Provisioning Ultra-Low Latency Services in Softwarized Network for the Tactile Internet

NATTAKORN PROMWONGSA

A THESIS

IN

THE CONCORDIA INSTITUTE

FOR

INFORMATION SYSTEMS ENGINEERING

PRESENTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF DOCTOR OF

DOCTOR OF PHILOSOPHY (INFORMATION SYSTEM ENGINEERING) AT

CONCORDIA UNIVERSITY

MONTREAL, QUEBEC, CANADA

MAY 2022

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Nattakorn Promwongsa

Entitled: Provisioning Ultra-Low Latency Services in Softwarized Network for the Tactile Internet

and submitted in partial fulfillment of the requirements for the degree of

Doctor Of Philosophy INFORMATION SYSTEM ENGINEERING

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Pouya Valizadeh

_____ External Examiner
Dr. Soumaya Cherkaoui

_____ External to Program
Dr. Yann-Gaël Guéhéneuc

_____ Examiner
Dr. Chadi Assi

_____ Examiner
Dr. Jamal Bentahar

_____ Thesis Supervisor (s)
Dr. Roch Glitho

_____ Thesis Co-supervisor (s)
Dr. Noël Crespi

Approved by _____
Dr. Mohammad Mannan, Graduate Program Director

May 27, 2022
Date of Defense

_____
Dr. Mourad Debbabi, Dean, Gina Cody School of Engineering and Computer Science

ABSTRACT

Provisioning Ultra-Low Latency Services in Softwarized Networks for the Tactile Internet

Nattakorn Promwongsa, Ph.D.

Concordia University, 2022

The Internet has made several giant leaps over the years, from a fixed to a mobile Internet, then to the Internet of Things, and now to a Tactile Internet. The Tactile Internet is envisioned to deliver real-time control and physical tactile experiences remotely in addition to conventional audiovisual data to enable immersive human-to-machine interaction and allow skill-set delivery over networks. To realize the Tactile Internet, two key performance requirements, namely ultra-low latency and ultra-high reliability need to be achieved. However, currently deployed networks are far from meeting these stringent requirements and cannot efficiently cope with dynamic service arrivals/departures and the significant growth of traffic demands. To fulfill these requirements, a softwarized network enabled by network function virtualization (NFV) and software-defined network (SDN) technologies is introduced as a new promising concept of a future network due to its flexibility, agility, scalability and cost efficiency. Despite these benefits, provisioning Tactile Internet network services (NSs) in an NFV-based infrastructure remains a challenge, as network resources must be allocated for virtual network function (VNF) deployment and traffic routing in such a way that the stringent requirements are met, and network operator's objectives are optimized. This problem is also well-known, as NFV resource allocation (NFV-RA) and can be further divided into three stages: (i) VNF composition, (ii) VNF embedding/placement and (iii) VNF scheduling.

This thesis addresses challenges on NFV-RA for Tactile Internet NSs, especially ultra-low latency NSs. We first conduct a survey on architectural and algorithmic solutions proposed so far for the Tactile Internet. Second, we propose a joint VNF composition and embedding algorithm to efficiently determine the number of VNF instances to form a VNF forward graph (VNF-FG) and their embedding locations to serve ultra-low latency NSs, as in some cases, multiple instances of each VNF type with proper embedding may be needed to guarantee the stringent latency requirements. The proposed algorithm relies on a Tabu search method to solve the problem with a

reasonable time. Third, we introduce real-time VNF embedding algorithms to efficiently support ultra-low latency NSs that require fast service provisioning. By assuming that a VNF-FG is given, our proposed algorithms aim to minimize the cost while meeting the stringent latency requirement. Finally, we focus on a joint VNF embedding and scheduling problem, assuming that ultra-low latency NSs can arrive in the network any time and have specific service deadlines. Moreover, VNF instances once deployed can be shared by multiple NSs. With these assumptions, we aim to optimally determine whether to schedule NSs on already deployed VNFs or to deploy new VNFs and schedule them on newly deployed VNFs to maximize profits while guaranteeing the stringent service deadlines. Two efficient heuristics are introduced to solve this problem with a feasible time.

Acknowledgments

Firstly, I am grateful to my Ph.D. supervisor Dr. Roch Glitho and my Ph.D co-supervisor Dr. Noel Crespi for their guidance, support, patience, and encouragement. They both are a dedicated and caring advisor. Thank you.

Secondly, I gratefully acknowledge my Ph.D. committee members, Dr. Yann-Gaël Guéhéneuc, Dr. Chadi Assi, Dr. Jamal Bentahar and Dr. Soumay Cherkaoui for their time, effort, and constructive comments.

I am also thankful to Dr. Diala Naboulsi, Dr. Amin Ebrahimzadeh, Dr. Carla Mouradian, Dr. Wubin Li, Ákos Recse and Dr. Róbert Szabó for all the enlightening discussions, comments, and collaborations. It was a great pleasure working with you. Furthermore, I am thankful to all of my colleagues in the telecommunication service engineering lab at Concordia University for their companionship, support, ideas, and fruitful discussions.

Finally, I am forever indebted to my parents for their encouragement, continuous support, and love. Without them and their support, not only this thesis, but also none of my achievements would have been possible. There are no words that can express my gratitude and love for you.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
AP	Access Point
BD	Bender's Decomposition
BS	Base Station
CO	Central Office
D2D	Device-to-device
DoF	Degree of Freedom
EDCA	Enhanced Distributed Channel Access
EPON	Ethernet Passive Optical Network
HCCA	Hybrid Coordination Function-controlled Channel Access
HMM	Hidden Markov Model
ILP	Integer Linear Programming
IoT	Internet of Things
LTE-A	Long Term Evolution Advanced
MAC	Medium Access Control
MEC	Multi-access Edge Computing
MILP	Mixed Integer Linear Programming
MINLP	Mixed Integer Nonlinear Programming
MLP	Multi-layer Perceptron
MPR	Multi-plane Routing
NFV	Network Function Virtualization
NFV-RA	NFV Resource Allocation
NO	Network Operators
NOMA	Non-orthogonal Multiple Access
NS	Network Service
OFDMA	Orthogonal Frequency Division Multiple Access
OLT	Optical Line Terminal
ONU	Optical Network Units

OSPF	Open Shortest Path First
PHY	Physical Layer
PON	Passive Optical Network
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
SDN	Software Defined Networking
SFC	Service Function Chain
SLA	Service Level Agreement
TTI	Transmission Time Interval
UE	User Equipment
VM	Virtual Machines
VNF	Virtual Network Functions
VNF-FG	VNF Forward Graph
VR	Virtual Reality
VNE	Virtual Network Embedding
WBAN	Wireless Body Area Network
WBANN	Body Area Nanonetworks
WLAN	Wireless Local Area Network

Chapter 1

1. Introduction

1.1 Overview

Today's Internet is ushering in a new era of communications, where interactive cyber-physical systems can exchange not only the conventional triple-play data (i.e., audio, video, and text), but also haptic control messages in real-time through the so-called Tactile Internet [1]. The Tactile Internet is a natural evolution of the Internet, which went from a fixed and text-based Internet to a multimedia mobile Internet, and then to an Internet of Things (IoT). It is geared towards haptic communications over the network. Using the Tactile Internet, a physician may be able to perform a remote surgical operation on a distant patient by perceiving not only real-time visual and auditory, but also touching senses of the distant environment. These new possibilities will revolutionize the set of applications and services provided to date by the Internet and take the next-generation systems to an unprecedented level of human-like communication. The Tactile Internet will revolutionize and combine machine-to-machine and human-to-machine interactions [2].

Tactile Internet applications, such as tele-surgery mostly require ultra-low latency and a high level of reliability and security to function correctly and safely. Latency requirements of Tactile Internet applications may vary from <10 ms up to tens of milliseconds, depending on the type of application and dynamicity of the environment [3][4]. Similar to the latency requirement, reliability requirement of the Tactile Internet applications can also vary, depending on the given

application. It can start from a failure rate of 10^{-3} [3][4]. The most critical Tactile Internet applications (e.g., tele-surgery) require a reliability of up to 10^{-7} [3].

However, currently deployed networks (e.g., 5G) cannot still potentially realize Tactile Internet applications because of the stringent Tactile Internet requirements [1][5]. Moreover, the next-generation networks are expected to support a wide range of applications with different requirements [5]. Therefore, a softwarized network is envisioned as a new promising concept of a future network to support services in flexible, agile and cost-efficient manners to efficiently guarantee the unique requirements of these vertical services, and to cope with their dynamic arrivals/departures and the significant growth of traffic demands [5][6]. Network function virtualization (NFV) and software defined networking (SDN) are two key technologies for realizing such networks [6][7]. NFV allows network functions required by network services (NSs) to be deployed using virtualization technology, which is more flexible, scalable and cost-efficient than a traditional method (e.g., relying on propriety hardware) [8]. Meanwhile, SDN provides flexible and agile management process to establish efficient traffic routing for NSs through centralized network control [9].

Although an NFV/SDN-based softwarized network brings several benefits, provisioning Tactile Internet NSs in this network is not a simple task due to their stringent requirements. With NFV settings, Tactile Internet NSs can generally be provisioned by steering their traffic through an ordered set of virtual network functions (VNFs) between end-points [10]. In order to meet the stringent requirements of Tactile Internet NSs while considering network operators (NOs)' objectives (e.g., minimizing cost), network resources for VNF deployment and traffic routing need to be efficiently allocated. This challenge is well-known as NFV resource allocation (NFV-RA) [11]. In general, the NFV-RA problem can be divided into three main stages: (i) VNF composition, (ii) VNF embedding/placement, and (iii) VNF scheduling [11]. The first stage involves determining the number of VNF instances and their order in order to form a service function chain (SFC) or, more generally, a VNF forward graph (VNF-FG). The second stage aims at embedding the VNFs in the VNF-FG into NFV-enabled nodes and to embed the virtual links between the VNFs into substrate links. The third stage focuses on determining the execution schemes of the VNFs in the VNF-FG required to run a given service.

This thesis mainly focuses on NFV-RA for Tactile Internet NSs in an NFV-based infrastructure, paying particular attention on ultra-low latency NSs, as ultra-low latency is the most

critical performance requirement for most of Tactile Internet NSs. In the following subsections, we first discuss the challenges on NFV-RA for ultra-low latency NSs followed by our main thesis contributions. Next, we provide background information about important concepts related to our thesis. Finally, we present our thesis outline.

1.2 Challenges

The main challenges on NFV-RA for ultra-low latency NSs in an NFV-based infrastructure are summarized as follows:

- **Joint VNF composition and embedding for ultra-low latency NSs:** It is critical to take all delay components including processing delay and communication delay into consideration when solving the NFV-RA problem for ultra-low latency NSs in order to efficiently meet their stringent latency requirement. In the NFV-RA problem, processing delay is generally influenced by VNF instances processing traffic of NSs. On the other hand, communication delay, including transmission delay and propagation delay highly depend on embedding locations of VNF instances and traffic routing. In some cases, especially when the latency requirement is very stringent, multiple instances of each VNF type required by a NS with proper VNF embedding may be needed to share processing loads to accelerate the processing delay [12][13] in order to meet the very stringent latency requirement. This is very challenging, as VNF composition and embedding stages need to be jointly determined. Moreover, splitting traffic to be processed by multiple instances of each VNF type to serve a NS could lead to a decrease in reliability which is also critical for some ultra-low latency NSs. This is because the number of NFV-enabled nodes involved in hosting these VNF instances may increase. Here, reliability of a NS can be defined as probability that all VNF instances in its VNF-FG can operate for a given period of time [14]. Also, resource costs of NFV-enabled nodes and physical links can generally be heterogeneous. Therefore, an efficient algorithm is needed to tackle this challenge to satisfy stringent latency requirements of ultra-low latency NSs while ensuring low reliability degradation and cost.
- **Real-time VNF embedding for ultra-low latency NSs:** Some ultra-low latency NSs require to be provisioned immediately once they arrive in the network to be able to operate. One example is teleoperation for wildfire suppression, in which a service needs to operate immediately once wildfire occurs to suppress the wildfire in time. However, finding a high-

quality VNF embedding solution for an ultra-low latency NSs with timely deployment time is not a simple task, as this does not only involve determining where to embed VNFs into NFV-enabled nodes, but also routing traffic through a set of VNFs in a specific order while ensuring that overall latency does not violate the stringent latency requirement. We assume that a VNF-FG from the VNF composition stage is given. Moreover, resource costs need to be considered to ensure high profits of NOs. One simple approach to tackle this problem is to embed each VNF of a given VNF-FG of a NS in a sequential order, based on its best local decision. However, this approach could generally lead to poor embedding outcomes, as embedding decision of any given VNF could be influenced by embedding decisions of other VNFs in a VNF-FG. Therefore, there is still a need for efficient VNF embedding algorithms to take VNF dependencies into account to find high-quality embedding solutions while ensuring fast service deployment times.

- **Joint VNF embedding and scheduling for ultra-low latency NSs:** Some ultra-low latency NSs can arrive in and depart from the network at any point in time [10]. Some of them may have specific service deadlines that need to be met [15][16]. Specifically, these NSs require their traffic to be processed through an ordered set of VNFs within specific service deadlines before they depart from the network. A cost-efficient approach to serve NSs is to allow them to share VNF instances that were deployed in the network to optimize resource usages. However, this is not trivial for ultra-low latency NSs, as a scheduling scheme of each VNF instance to serve the NSs need to be properly determined to satisfy the service deadlines. Moreover, when the service deadline becomes more stringent, new VNF instances may be also needed to be deployed in the network to reduce waiting delays caused by VNFs and link sharing and shorten communication delays. These new deployments certainly come with additional costs. Therefore, efficient algorithms are needed to jointly address VNF embedding and scheduling stages for ultra-low latency NSs to meet the stringent service deadlines while optimizing the costs.

1.3 Thesis Contributions

Our survey indicates that the solutions proposed so far for the Tactile Internet do not address the challenges related to NFV-RA for Tactile Internet ultra-low latency NSs, as described in Section 1.2. Therefore, this thesis aims to tackle these challenges to efficiently provision ultra-low

latency NSs in an NFV/SDN-based softwarized network for the Tactile Internet. To this end, this thesis consists of four main contributions. Each of them is described in the following subsections.

1.3.1 Survey of the Tactile Internet

The Tactile Internet is still its early stage. Improvement of every segment of today's networks is still needed to be able to potentially realize the Tactile Internet. Therefore, in the first contribution, we conduct a survey, covering both architectural and algorithmic aspects of the Tactile Internet [1]. More precisely, in this survey, we propose a classification of works done so far for the Tactile Internet. We then critically review them and derive some useful insights and lessons learned from our literature review. Finally, we discuss some promising research directions.

Based on this survey, none of NFV-RA algorithms addresses the challenges discussed in this thesis. Hence, for the next three contributions, we propose efficient approaches to tackle each of these three challenges as described in Section 1.2.

1.3.2 Joint VNF Composition and Embedding Algorithm

In the second contribution, we tackle joint VNF composition and embedding for ultra-low latency NSs [17]. As we discussed earlier, in some cases, multiple VNF instances of each type with proper VNF embedding, traffic splitting and traffic routing are needed to meet the very stringent latency requirements. However, increasing the number of VNF instances of each type to process traffic of a NS could lead to a reliability degradation, which is critical for some ultra-low latency NSs. Therefore, in this contribution, we propose a joint VNF composition and embedding algorithm to address this challenge. Our proposed algorithm aims at efficiently determining where traffic of a NS should be split and processed by multiple instances of each VNF type embedded on different physical nodes to guarantee the stringent latency requirements while optimizing reliability degradation and cost. We formulate the problem as a mixed integer linear programming (MILP). Due to the complexity of the formulated problem, we introduce a Tabu search-based algorithm to solve the problem with a feasible time. The simulation results indicate that our proposed algorithm can achieve high-quality solutions and outperforms a late acceptance hill climbing-based algorithm and the existing algorithm.

1.3.3 Real-Time VNF Embedding Algorithm

In the third contribution, we focus on real-time VNF embedding for ultra-low latency NSs to ensure fast service provisioning. Generally, one viable approach to solve the VNF embedding problem to find a satisfactory solution with an acceptable execution time is a heuristic-based approach. However, this approach typically relies on greedy and sequential VNF embedding, which could easily suffer from the so-called causality issue. This issue may degrade the embedding solution quality. The causality issue means that embedding decisions cannot be optimally determined before all neighboring dependencies are known. To tackle this issue, we introduce our *h*-horizon sequential lookahead greedy embedding algorithms [18][19]. Our proposed algorithms rely on a greedy and sequential heuristic-based approach to ensure timely service provisioning. Furthermore, they provide efficient VNF embedding and re-embedding strategies to alleviate the impact of the causality issue to ensure high-quality solutions. Our objective is to minimize total costs while satisfying the stringent latency requirement. We also propose a mathematical formulation of the online VNF embedding problem and adopt a Bender's decomposition method to solve it to serve as a benchmark for our proposed algorithms. The simulation results show that our proposed algorithms achieve lower total cost then the existing algorithm while being much more scalable than optimization-based approaches.

1.3.4 Joint VNF Embedding and Scheduling Algorithm

In the fourth contribution, we address joint VNF embedding and scheduling for ultra-low latency NSs. As we discussed earlier, some ultra-low latency NSs can arrive and depart the network at any point in time and have stringent service deadlines [15][16]. To efficiently guarantee such stringent service deadlines while maximizing profit of NSs, VNF embedding and scheduling need to be jointly considered. In this contribution, we propose joint VNF embedding and scheduling algorithms to tackle this challenge [20]. They aim at efficiently determining whether to schedule NSs on VNF instances already deployed in the network, or to deploy new VNF instances and schedule NSs on newly deployed VNF instances to maximize profit of NOs while guaranteeing the stringent service deadlines. The profit is defined as total revenue subtracted by total cost. We formulate the problem as an integer linear programming (ILP). To tackle the scalability issue of the formulated problem, we propose two efficient algorithms, namely greedy-

based and Tabu search-based algorithms to solve the problem. The simulation results indicate that our proposed algorithms outperform the existing algorithms in terms of the profits.

1.4 Background Information

In this subsection, we provide brief background information related to this thesis. This background information covers the Tactile Internet, the softwarized network and the NFV-RA.

1.4.1 Tactile Internet

The term Tactile Internet was first coined in 2014 by Fettweis in his seminal paper [2]. Fettweis defined the Tactile Internet as the enabling technology for control and steering of real and/or virtual objects through the Internet by requiring a very low round-trip latency [2]. Not long afterwards, in March 2016, the IEEE P1918.1 standards and working group was formed, which aims to define the reference architecture of the Tactile Internet framework [4]. The IEEE P1918.1 standard working group defines the Tactile Internet as "A network, or a network of networks, for remotely accessing, perceiving, manipulating, or controlling real and virtual objects or processes in perceived real-time".

The Tactile Internet is expected to allow the provisioning of a broad range of new applications that will enrich the set already offered by today's Internet. Interactive real-time haptic applications will comprise the bulk of these, and they will play a pivotal role in enhancing everyday life. Remotely-controlled systems may become key to the functioning of several very important and sensitive domains, such as healthcare (e.g., tele-diagnosis, tele-surgery), industry (e.g., dangerous and difficult-to-reach environments), virtual and augmented reality (e.g., a firefighter training system), road traffic (e.g., automated and cooperative driving), education and serious gaming (e.g., games for personalized cardio-training), and many others [2][21].

Haptic applications bring a set of stringent requirements in terms of latency and reliability. An ultra-low latency of 1 ms is required to ensure the timely delivery of control messages (e.g., a surgeon's hand movement) and avoid cybersickness (e.g., in the case of virtual reality (VR)). Cybersickness occurs when multiple senses (audio, video and touch) participate in an interaction, but the feedback of the different senses are notably unsynchronized (e.g., the time-lag between tactile and visual movement exceeds 1 ms) [22]. When a Tactile Internet signal includes the synchronization between several senses, this implies a multi-modal signal. On another note, ultra-

high reliability is required for several of the applications envisioned for the Tactile Internet. For instance, it is inconceivable that a data communication error (e.g., malicious or network-based) might occur during remote open-heart surgery or when controlling an airplane in flight.

Given the targeted applications and their requirements, the realization of the Tactile Internet clearly brings many challenges. As captured in [23], these challenges can be categorized under four streams: haptics, intelligence, computing and communication. Each of these are briefly discussed below and summarized in Figure 1.1.

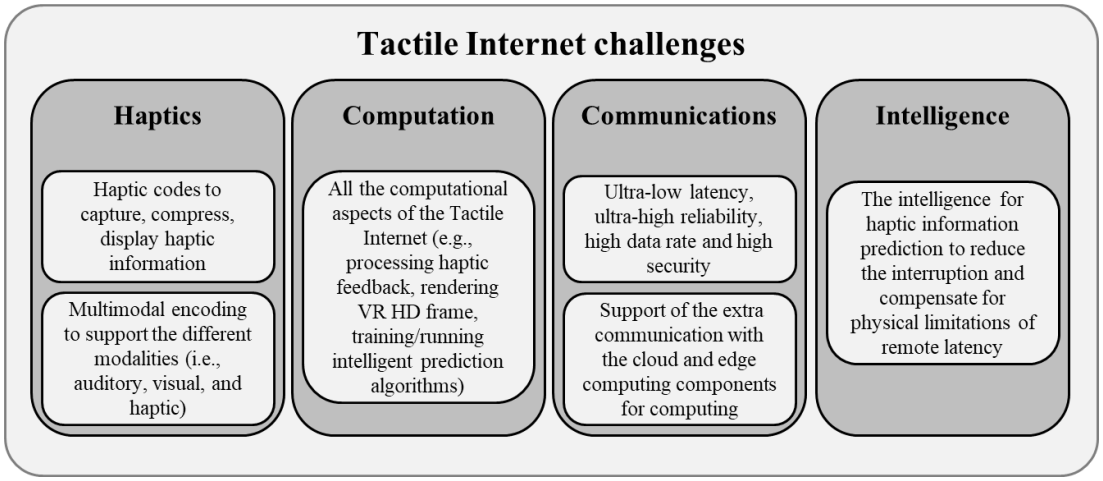


Figure 1.1: Tactile Internet Challenges [23].

Haptic challenges: These cover capture, compress, transmit and display haptic information with minimum latency. Haptic devices capable of sensing and delivering both tactile and kinesthetic feedback are needed at both ends [24]. Multimodal encoding schemes must be defined to support the different modalities (i.e., vocal, visual, and haptic) of Tactile Internet information, without increasing the end-to-end latency [25]. Steinbach *et al.* [25] report that visual and auditory information encoding have been studied extensively, but not haptic encoding. Meeting the challenge of haptics will also require customized interfaces to support and distinguish between human-to-machine, machine-to-human and machine-to-machine interactions.

Intelligence challenges: These are related to the intelligence required for haptic information prediction (e.g., to predict a surgeon's next move). This covers the actions, movements, and feedback predictions to enhance the latency and reduce communication delays and interruptions. Predictions may fill a critical gap; for instance, if the command signal for the next action is not received on time, due to either the distance (more than 150 km), or a connection problem.

Computation challenges: These deal with all the computational aspects of haptic communication, including processing haptic feedback for a real-time interaction 'feel' and running the prediction intelligence program. While computations may be resource- and time-consuming, they should not affect the end-to-end latency requirement.

Communication challenges: This category encompasses the requirements for ultra-low latency, ultra-high reliability, a high exchange rate, and the support of the extra communication with the cloud and edge computing components that can be used for computing. A communication infrastructure that meets all these requirements remains to be developed. Security mechanisms and multiplexing schemes combining multiple modalities and respecting the 1ms latency are also needed [22].

1.4.2 Softwarized Network

A softwarized network can refer to a network that supports services and applications through software-oriented operations [6][7][26]. This network is expected to offer flexibility in terms of configuration adaptation, network function location and scalability to efficiently deal with dynamic changes in user demands, network conditions and application requirements in order to ensure high quality of service (QoS) [7][26]. NFV and SDN are well known as two key technologies to realize the softwarized network. With NFV and SDN, setting up (virtual) network functions and networks as well as configuring traffic routing for services and applications can be achieved simply through software interfaces [7].

NFV is a network architecture that takes the process of network building and operating to a new level of flexibility by decoupling the network functions from the hardware infrastructure running those functions. NFV relies on standard virtualization technologies to virtualize network functions and elements, creating VNFs [8]. These VNFs are instantiated, chained, and run as needed to provide targeted services, thereby shortening the life cycle of provisioning new services. With the utilization of NFV, VNFs required by services can be flexibly instantiated and migrated anywhere and anytime in the network to guarantee service-level agreement (SLA) requirements of services.

SDN refers to a networking architecture where the network control plane is separated from the data plane [9]. The control plane is responsible for making decisions on how to handle the network traffic, while the data plane performs the actual traffic forwarding based on those decisions. With

SDN technology, the networking devices (e.g., switches, routers) are managed by a centralized controller, thereby simplifying the whole management process (e.g., network configuration and policy enforcement). Being aware of global views of the entire network and having the agile, flexible and simple management process can efficiently help handle traffic guarantee QoS requirements of services.

1.4.3 NFV Resource Allocation

NFV-RA is one of the main challenges for provisioning NSs in an NFV-based infrastructure, as it highly influences SLA requirements of the NSs and profits of NOs. With NFV settings, NSs can generally be provisioned by steering their traffic through an ordered set of VNFs between end-points [10]. According to [11], NFV-RA can be generally divided into three stages: (i) VNF composition, (ii) VNF embedding/placement and (iii) VNF scheduling. Each of these stages is briefly described below.

VNF composition: In general, a NS request consists of a set of VNFs with dependency constraints and SLA requirements (e.g., resource demands, traffic demands, latency requirements) [13]. Given this NS request, VNF composition aims to compose chains of VNFs to build a VNF-FG by determining an order of VNFs, the number of VNF instances of each type and connections between the instances with respect to NO's objectives and the SLA requirements. Figure 1.2 shows an example of the VNF composition, where Figure 1.2a shows a set of required VNFs in an original NS request while Figure 1.2b demonstrates one possible VNF-FG of the original NS request.

VNF embedding/placement: The second stage of NFV-RA is VNF embedding, also known as VNF placement. Given a VNF-FG obtained from the VNF composition stage, the VNF embedding aims to allocate network resources for VNF deployment and traffic routing in order to embed the VNF-FG into an NFV-based infrastructure. This VNF-FG embedding includes VNF and virtual link embedding. In the VNF embedding, each VNF in a VNF-FG is mapped into an NFV-enabled node, meaning that NFV-enabled node's resources (e.g., CPU, memory, storage) must be allocated to deploy the VNF. The VNF could be also embedded into a VNF instance that were already deployed in the network if that VNF instance is capable of supporting the VNF. Meanwhile, in the virtual link embedding, each virtual link between the VNFs in the VNF-FG is embedded into physical links, meaning that links' resources (i.e., link bandwidth) must be reserved to facilitate traffic routing of the virtual link. Note that each virtual link may be embedded into

more than one physical links (a path). These VNF and virtual link embedding need to be carried out with respect to NOs' objectives and SLA requirements. An example of VNF embedding/placement is demonstrated in Figure 1.3. In this figure, VNFs A, B and E in a VNF-FG are embedded into Node 2 by deploying new VNF A, B and E instances while VNF C is embedded into Node 3 by reusing the VNF C instance already deployed in that node. Also, virtual links between a source and VNF A, between VNF E and VNF C and between VNF C and a destination are embedded into Links 1, 2 and 4, respectively.

VNF scheduling: The VNF scheduling stage, the last one, focuses on determining execution schemes of VNFs required by NSs. Given sets of VNF instances and physical links assigned to NSs from the VNF embedding stage and with assumption that VNF instances can be shared by multiple NSs, the VNF scheduling aims to determine time intervals on which the VNF instances and links serve NSs with respect to SLA requirements (e.g., service deadlines) and NOs' objectives (e.g., power consumption minimization).

(a) (b)

Figure 1.2: An illustration of VNF composition: (a) an original set of required VNFs and (b) one possible VNF-FG.

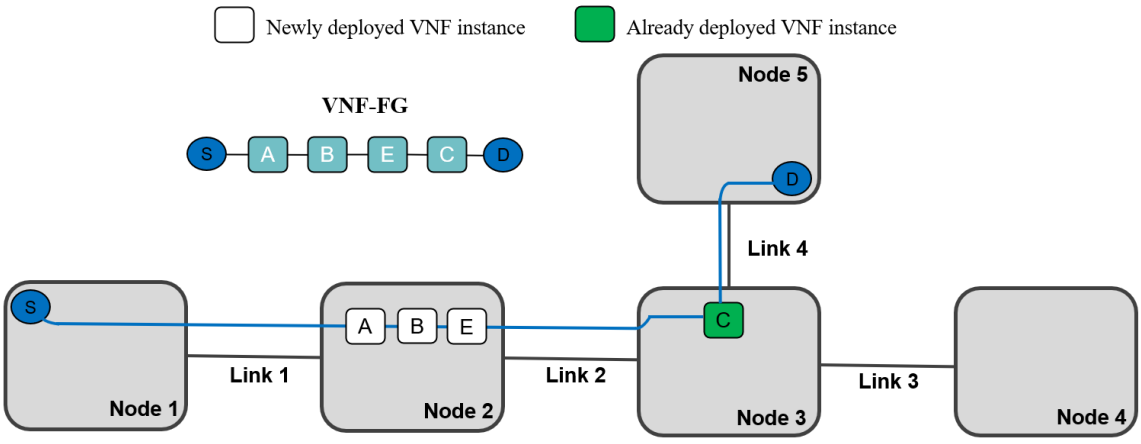


Figure 1.3: An example of VNF embedding.

1.5 Thesis Outline

The rest of the thesis is organized as follows. Chapter 2 discusses the requirements related to the challenges on NFV-RA for ultra-low latency NSs and presents a critical review of the state-of-the-art. Before discussing our proposed NFV-RA algorithms for ultra-low latency NSs, we also provide a survey of the architectures and algorithms proposed up to date and not only specific to NFV-RA for the Tactile Internet in Chapter 3. While Chapter 4 presents a joint VNF composition and embedding algorithm for ultra-low latency NSs, Chapter 5 presents real-time VNF embedding algorithms for ultra-low latency NSs. In Chapter 6, we propose joint VNF embedding and scheduling algorithms for ultra-low latency NSs. Finally, we conclude this thesis and provide future directions for this research work in Chapter 7.

Chapter 2

2. Requirements and Related Work

2.1 Requirements of NFV-RA Algorithms for Ultra-Low Latency NSs

This thesis focuses on ultra-low latency NSs for the Tactile Internet. Given the ultra-low latency requirements, we first derive the general requirements in designing NFV-RA algorithms for ultra-low latency NSs. After that, we discuss the requirements specific to the NFV-RA algorithms to address the challenges described in Section 1.2.

2.1.1 General Requirements

There are three main requirements NFV-RA algorithms should generally consider to efficiently support ultra-low latency NSs while optimizing NO's objectives. Each of them is described below.

Latency: As their name suggests, ultra-low latency NSs are latency-sensitive services that have specific latency requirements to be satisfied to ensure high quality of experiences (QoEs). These latency requirements may vary, depending on services. To efficiently guarantee the latency requirements, an NFV-RA algorithm must take all latency components into account, including processing latency and communication latency when solving the NFV-RA problem for ultra-low latency NSs. Furthermore, the algorithm should consider the latency as a constraint it needs to satisfy because considering the latency as the main objective it should optimize may not be able to guarantee the latency requirements.

Traffic routing: Traffic routing is another requirement an NFV-RA algorithm must take into consideration, as it has a huge impact on communication latency. Specifically, improper traffic routing could lead to long communication delay, which may not allow the latency requirements of ultra-low latency NSs to be met. The traffic routing must include finding suitable routing paths and ensuring bandwidth requirements on physical links for reliable transmission.

Cost: A cost is always the main concern of a NO, as it highly impacts NO's profit. Therefore, an NFV-RA algorithm must optimize the cost while serving ultra-low latency NSs. The cost can be represented in terms of resource usage or resource costs.

2.1.2 Algorithm-Specific Requirements

In this subsection, we identify requirements that are specific to the NFV-RA algorithms to address the three challenges we discussed in Section 1.2.

2.1.2.1 Joint VNF composition and embedding algorithm

In addition to the three main requirements as discussed above, there are two additional requirements a joint VNF composition and embedding algorithm must consider to efficiently support ultra-low latency NSs. These two requirements are described as follows:

Joint design approach: As the VNF composition and VNF embedding stages can be mutually dependent [27], solving these two stages independently might not be an efficient way to meet the latency requirements of ultra-low latency NSs, especially when the latency requirements become very stringent. Therefore, to efficiently ensure that the overall latency including both processing and communication latency does not violate these stringent latency requirements, it is necessary to jointly address these two stages.

Reliability: As discussed in Section 1.2, although allowing multiple VNF instances of each type to serve a NS can improve the overall latency, this could lead to a decrease in the reliability. Hence, an algorithm should ensure a minimum reliability degradation while meeting the stringent latency requirements when jointly determining VNF composition and embedding because the reliability is also critical for some ultra-low latency NSs.

2.1.2.2 Real-time VNF embedding algorithm

Four additional requirements are needed in designing a real-time VNF embedding algorithm to serve ultra-low latency NSs that require fast service provisioning.

Fast NS provisioning: An algorithm should enable fast NS provisioning, as some ultra-low latency NSs need to be provisioned immediately once they arrive in the network to be able to operate (e.g., wildfire suppression).

General VNF-FG topologies: VNF-FG topologies of some ultra-low latency NSs can go beyond a simple linear topology [10]. Thus, general VNF-FG topologies need to be considered when solving the real-time VNF embedding.

Causality issue: The causality issue could significantly impact on embedding quality solutions, as embedding decision of any given VNF in a VNF-FG highly influences those of other VNFs. Thus, it is essential to take the causality issue into consideration when solving the real-time VNF embedding for ultra-low latency NSs to ensure high-quality embedding solutions.

VNF reusability: One viable way to further reduce a cost is to reuse VNF instances that were already deployed in the network to save costs for new VNF deployment (e.g., license costs, instantiation costs etc.). Therefore, an algorithm should consider the VNF reusability to maximize NO's profit.

2.1.2.3 Joint VNF embedding and scheduling algorithm

Here, we derive three additional requirements that are specific to a joint VNF embedding and scheduling algorithm to efficiently serve ultra-low latency NSs.

Service deadline: Given that some ultra-low latency NSs can arrive and depart the network at any point in time, these NSs have specific service deadlines [15][16]. Thus, an algorithm should be able to guarantee the service deadlines. In addition, to meet these service deadlines, both processing latency and communication latency must be taken into account when serving ultra-low latency NSs.

Joint design approach: As discussed in Section 1.2**,** considering a scheduling-only scheme with an assumption that VNF embedding is done in advance may not allow stringent service deadlines to be met, as already deployed VNF instances may be shared by multiple NSs, thereby leading to long waiting times. Therefore, to efficiently guarantee these stringent service deadlines, the VNF embedding and scheduling need to be jointly considered when scheduling ultra-low latency NSs.

VNF reusability: Relying only on newly deployed VNF instances to schedule NSs is not cost-efficient, as these new instances impose additional costs (e.g., license costs, instantiation costs).

Therefore, an algorithm should reuse as many already deployed VNF instances as possible to schedule NSs before deploying new VNF instances to reduce costs.

2.2 Related Work

This subsection reviews the existing NFV-RA algorithms proposed in the literature. We classify them into two main categories, (i) single design-based and (ii) joint design-based approaches. The single design-based approach aims to solve three NFV-RA stages (i.e., VNF composition, embedding and scheduling) separately whereas the joint design-based approach tackles two of these three stages jointly.

2.2.1 Single Design-Based Approach

Most of studies rely on a single design-based approach to tackle the NFV-RA problem due to its complexity. We further categorize them into three main categories, (i) VNF composition, (ii) VNF embedding and (iii) VNF scheduling. Each of them is described below.

2.2.1.1 VNF Composition

A few studies have tackled the VNF composition problem. Mehraghdam *et al.* [28] introduce a context-free language to formalize service requests and propose a greedy heuristic algorithm to compose a VNF-FG. This algorithm aims to minimize the overall data rate by prioritizing a VNF that reduces the data rate of flows the most in each composition step. Gil-Herrera *et al.* [29] also focus on addressing VNF composition for service requests. The authors assume that the service requests may have more than one termination points and propose a scalable metaheuristic algorithm, based on Tabu search to tackle this problem with the objective of minimizing total bandwidths. In addition, a security service chaining composition problem is studied by Liu *et al.* [30]. The authors formulate the problem as an ILP problem with respect to security and resource requirements, and propose a heuristic algorithm based on a breath first search to solve it. We note that these studies tackle the VNF composition solely and do not consider the latency aspect.

2.2.1.2 VNF Embedding/Placement

The VNF embedding problem is one of the most challenges of NS deployment in the NFV ecosystem, as it has a huge impact on profit gains of network operators and service performance.

Multiple studies in the literature are thus dedicated to find efficient ways to tackle this problem (for a survey, the interested reader is referred to the recently published survey on the topic [31]). In this section, we review the existing VNF embedding algorithms, classifying them into four main categories of (i) optimization-, (ii) metaheuristic-, (iii) heuristic-, and (iv) machine learning-based approaches. Each of these approaches is described below.

Optimization-based approach: An optimization-based approach generally aims at formulating the VNF embedding problem as an optimization problem and finding the optimal solution through optimization solvers (e.g., CPLEX). For instance, Oljira *et al.* [32] tackle the problem of virtual mobile core network function embedding in NFV-based core networks. They formulate the problem as a MILP, aiming to minimize the total cost while satisfying the latency requirement. They also consider virtualization overhead in the delay model, which occurs when multiple VNFs are hosted in the same physical node. In [33], Alleg *et al.* propose a latency-aware VNF embedding method, designed to minimize the resource consumption while considering the impact of allocated resources on the processing delay. The problem is formulated as a mixed integer quadratically constrained programming. Cai *et al.* [34] focus on VNF embedding for disaster-resilient services. They formulate the problem as an ILP with the main objective of minimizing resource consumption while considering the disaster protection using multi-path routing. It is important to note that although the optimization-based approach provides the optimal solution, they usually require a very high execution time, especially for large-scale scenarios, as the VNF-FG embedding problem is well known to be NP-hard [31].

Metaheuristic-based approach: Some studies have relied on a metaheuristic-based approach to solve the VNF-FG embedding problem. For instance, Yala *et al.* [35] focus on the VNF placement over edge-central cloud network and aim to find a good trade-off between service access delay and service availability while embedding VNFs into edge nodes. They propose a genetic-based algorithm to solve the problem. In [36], Khoshkholghi *et al.* study the problem of VNF-FG embedding with the objective of jointly optimizing the embedding cost and service latency. They introduce genetic- and bee colony-based algorithms to tackle the problem. Another genetic-based VNF-FG embedding algorithm is proposed by Magoula *et al.* [37]. The authors aim to minimize the overall latency while satisfying the latency requirement of latency-sensitive services. Generally, the metaheuristic-based approach relies on randomized and advanced search mechanisms [31] to improve solution quality. This, however, may lead to slow convergence,

especially for large-scale networks, thus making these solutions sometimes inappropriate for NSs that require fast service provisioning.

Heuristic-based approach: To ensure fast NS provisioning, a large number of studies (e.g., [38][39][40][41][42][43][44][45]) are based on heuristics to find satisfactory embedding solutions with acceptable execution times. Gouareb *et al.* [38] study the problem of VNF-FG embedding over edge cloud networks, aiming to minimize the overall latency. The authors formulate the problem as a MILP and then propose their heuristics based on a best-fit decreasing method to solve the problem. Jin *et al.* [40] focus on latency-aware VNF-FG embedding over multi-access edge computing (MEC)-based networks, where each VNF instance can be shared by multiple NSs. They formulate the problem as a MILP, aiming to minimize the total resource consumption while ensuring the given latency requirement. To ensure a fast NS deployment, they propose a heuristic algorithm to solve the developed MILP. The heuristic proposed in [40] divides the problem into two subproblems, namely, (i) path selection and (ii) VNF assignment. In the path selection subproblem, the algorithm finds a set of feasible paths for each NS that satisfy the latency requirement. After that, in the VNF assignment subproblem, the algorithm greedily and sequentially assigns VNFs of each NS on the nodes along the feasible path with the objective of reusing as many deployed VNFs as possible. Similarly, the authors of [41] introduce a heuristic algorithm for VNF-FG embedding. This algorithm aims to greedily embed each VNF of a given VNF-FG into the substrate network in a sequential order in such a way that the total embedding cost is minimized. Recently, in [43], Chen *et al.* propose a latency-aware VNF-FG embedding algorithm for reliable NSs. Specifically, the authors formulate the problem as an ILP and then solve it using their proposed heuristic, especially given that the problem is NP-hard. The proposed algorithm in [43] aims to find backup VNFs as well as routing paths to guarantee service availability while minimizing the transmission delay. In [45], Thanh *et al.* aim to solve the energy-aware VNF-FG embedding problem in edge-cloud environments for IoT services. Aiming to optimally determine whether to embed VNFs into either edge or cloud nodes to minimize the power consumption, they propose a sequential heuristic algorithm to solve the problem. We note that although these works can provide acceptable deployment times, they still suffer from the causality issue, which may lead to poor embedding solutions.

Machine learning-based approach: Recently, a machine learning-based approach has gained some attentions from researchers to tackle the VNF-FG embedding problem. This approach can

be applied to the VNF embedding problem in two ways. First, it is directly used to provide decision making on VNF and virtual link mappings for VNF-FG embedding (i.e., [46][47][48]). For instance, Pei *et al.* [46] introduce a deep learning-based algorithm, which is divided into two steps, (i) VNF placement and assignment and (ii) VNF chaining, to solve the VNF-FG embedding problem. Deep belief network models are deployed to provide decision making in both steps. Second, the machine learning-based approach can also be applied to reduce the size of the original VNF-FG problem, thereby reducing its complexity. For example, Araujo *et al.* in [49] propose a hybrid optimization-machine learning algorithm for VNF-FG embedding. This algorithm first relies on clustering techniques to identify a set of promising candidate nodes to embed a VNF-FG of a NS, thus significantly reducing the number of variables and constraints of a given ILP model. Then, it solves the ILP model to obtain the embedding solution. We note that the applicability of the machine learning-based approach is limited, as it usually requires training data (e.g., historical data of embedding of the previous NS requests), which may not always be available. Alternatively, reinforcement learning can train models using the feedback received from environments, though its key shortcoming is the lack of agility. If there are significant changes in the network (e.g., new SLA requirements, network topologies), the models should be retrained [50]. This, as a result, may lead to extra service deployment times.

2.2.1.3 VNF scheduling

Some studies have focused on the scheduling problem for NFV-RA [15][16][51][52][53][54]. These works assume that the deployment of VNF instances on physical nodes while satisfying their resource requirements is known in advance, and the objective is to map and schedule the VNFs of a given NS onto the already deployed VNF instances that support them. For instance, Mijumbi *et al.* [53] focus on the online scheduling problem. The authors assume that virtual machines (VMs) for running VNFs have already been deployed on physical nodes, and each VM can support multiple VNFs, but can only run one VNF at a time. The authors aim to map and schedule the VNFs required by the NS onto the already deployed VMs that can support them. They propose three simple greedy algorithms with different greedy criteria as well as a Tabu search-based algorithm to solve the problem. In [54], Assi *et al.* introduce energy-aware service mapping and scheduling with deadline guarantees by considering server consolidation. They formulate the problem as an ILP and then propose a Genetic-based heuristic algorithm to solve the problem. We

note that the works in [52]-[54] do not consider the traffic routing or transmission delay when scheduling NSs.

On the other hand, the authors of [51] formulated the service mapping and scheduling problem as a MILP designed to minimize the maximum completion time of service requests while considering traffic routing. In [51], they assume that each VM is dedicated to one VNF, and the deployment of VNF instances is known in advance. To cope with the complexity of the developed MILP, they also propose a column generation approach-based algorithm. Alameddine *et al.* in Refs. [15] and [16] focus on service scheduling for low latency NSs. They formulate the problem as a MILP problem, which accounts for both traffic routing and deadline guarantees. The authors of [16] assume that a node buffer is infinite and propose a Tabu search-based algorithm to solve the problem, while the work in [15] considers a finite node buffer model and introduces a game theory-based algorithm to tackle the problem. It should be noted that the above-mentioned works (i.e., [15],[16],[51]-[54]) assume that the deployment of VNF instances is done in advance.

2.2.2 Joint Design-Based Approach

Some studies aim to address three NFV-RA stages in a joint manner. They either consider a joint VNF composition and embedding or a joint VNF embedding and scheduling.

2.2.2.1 Joint VNF Composition and Embedding

Some studies have focused on a joint VNF composition and embedding. For instance, Wang *et al.* [55] model a joint VNF composition and embedding problem as a weighted graph matching problem, aiming to determine order of VNFs, the number of their instances and their embedding locations while optimizing resource consumption. The authors also propose a Hungarian-based algorithm to solve the problem. In [56], Li *et al.* focus on joint VNF composition and embedding in 5G networks. The authors propose a two-stage heuristic algorithm to jointly compose and embed a VNF-FG, aiming to maximize service acceptance ratios under resource capacity constraints. In [57], Wang *et al.* study a joint VNF composition and placement problem that takes VNF reusability into account. The authors formulate the problem as an ILP problem with objective of minimizing VNF instance utilization and bandwidth consumption and then propose a heuristic algorithm to solve it. Carpio *et al.* [58] aim to find the optimal number of instances of each VNF type and their optimal embedding locations such that the link cost is minimized. The authors assume that the

costs of all links increase linearly as link utilization grows, and an order of VNFs is fixed. They propose both MILP and heuristics to find an optimum network load balancing. We note that the works [55]-[58] do not consider the latency requirements of NSs. In [27], the authors consider latency as a penalty to service revenue that need to be optimized when jointly solving VNF composition and embedding. They formulated the problem as a MILP problem and propose a heuristic algorithm to solve it. However, this work [27] does not consider the latency requirements of NSs.

2.2.2.2 Joint VNF Embedding and Scheduling

A few works have tackled the joint VNF embedding and scheduling problem. Gholipoor *et al.* [59] investigate a joint radio and NFV resource allocation for the Tactile Internet. The authors in [59] consider a joint VNF embedding and scheduling problem for NFV-RA. However, they do not consider VNF reusability or data rate guarantees for transmission when scheduling services. Another work that proposes a joint VNF embedding and scheduling algorithm is Ref. [60], where the authors propose a two-stage online algorithm to solve the problem. However, applicability of this work is rather limited because it assumes that each physical node can run at most one VNF at a time. With this assumption, the NFV technology cannot be fully leveraged, especially given that NFV-enabled nodes can generally host and run multiple VNFs at the same time to efficiently utilize their resources. Moreover, Ref. [60] does not consider communication delay when scheduling NSs, which may not allow the service deadlines to be satisfied, since the communication delay can have a huge impact on overall service completion time.

2.3 Evaluation of the Related Work

In this subsection, we evaluate the related work with respect to the requirement of the NFV-RA algorithms for ultra-low latency NSs derived in Section 2.1.

2.3.1 Joint VNF Composition and Embedding algorithm

The evaluation of the existing studies on VNF composition and/or embedding with respect to the requirements of the joint VNF composition and embedding algorithm, is summarized in Table 2.1. As shown in this table, most of the studies mainly tackle the VNF embedding problem. Some of them also focus on the joint VNF composition and embedding problem. However, none of them

considers the joint design, cost, traffic routing, latency requirement and reliability degradation at the same time to efficiently support ultra-low latency NSs, especially when the latency requirements become very stringent.

Table 2.1: Evaluation of the related works with respect to the requirements of the joint VNF composition and embedding algorithm.

References	Joint design		Cost	Traffic routing	Latency	Reliability
	VNF composition	VNF embedding				
[28]	✓	-	-	-	-	✓
[29][30]	✓	-	✓	-	-	-
[32][33][36],[39]-[42], [47]-[49]	-	✓	✓	✓	✓	-
[37][44][46]	-	✓	-	✓	✓	-
[43]	-	✓	-	✓	✓	✓
[34]	-	✓	✓	✓	-	✓
[38]	-	✓	-	✓	-	-
[45]	-	✓	✓	✓	-	-
[35]	-	✓	-	-	✓	✓
[27],[55]-[58]	✓	✓	✓	✓	-	-

2.3.2 Real-Time VNF Embedding Algorithm

As we discussed in Section 2.2, the heuristic-based VNF embedding approach is the most promising way to ensure fast NS provisioning. In Table 2.2, we evaluate the existing VNF embedding works that rely on the heuristic-based approach with respect to the requirements of the real-time VNF embedding algorithm derived in Section 2.1.2.2. As demonstrated in the table, most of the works consider the cost, VNF reusability, traffic routing and latency to guarantee the latency requirements of ultra-low latency NSs with a minimum cost. Nevertheless, all of them still suffer from the causality issue, which may lead to low-quality embedding solutions.

Table 2.2: Evaluation of the related works with respect to the requirements of the real-time VNF embedding algorithm.

References	Approach	Fast provisioning	Causality solution	Cost	VNF reusability	Traffic routing	Latency	General VNF-FG Topologies
[38]	Heuristic	Fast	-	-	✓	✓	✓	-
[39][40] [42]	Heuristic	Fast	-	✓	✓	✓	✓	-
[41]	Heuristic	Fast	-	✓	✓	✓	✓	✓
[43][44]	Heuristic	Fast	-	-	-	✓	✓	-
[45]	Heuristic	Fast	-	✓	-	✓	-	-

2.3.3 Joint VNF Embedding and Scheduling Algorithm

Table 2.3 demonstrates the evaluation of the related works on the VNF scheduling and the joint VNF embedding and scheduling with respect to the requirements of the joint VNF embedding and scheduling algorithm discussed in Section 2.1.2.3. The works that consider only the VNF embedding not are evaluated here, because they cannot solve the VNF scheduling problem. As shown in Table 2.3, only two works tackle the joint VNF embedding and scheduling. However, none of them considers the cost, VNF reusability, traffic routing and service deadline requirement at the same time to efficiently schedule ultra-low latency NSs. It should also be noted that although [60] can meet almost all of the requirements, it does not consider communication delay when scheduling the NSs, which may result in violating the given service deadlines. In addition, it assumes that NFV-enabled nodes can run at most one VNF at a time. This assumption does not allow the NFV technology to be fully leveraged.

Table 2.3: Evaluation of the related works with respect to the requirements of the joint VNF embedding and scheduling algorithm.

References	Joint design		Cost	VNF reusability	Traffic routing	Service deadline
	VNF embedding	VNF scheduling				
[15][16]	-	✓	✓	✓	✓	✓
[51]	-	✓	-	✓	✓	-
[52]	-	✓	-	✓	-	-
[53]	-	✓	✓	✓	-	-
[54]	-	✓	✓	✓	-	-
[59]	✓	✓	✓	-	-	✓
[60]	✓	✓	✓	✓	✓	-

2.4 Conclusion

In this chapter, we first derived the requirements for designing the NFV-RA algorithms to support ultra-low latency NSs. Then, we critically reviewed the existing NFV-RA algorithms proposed in the literature and evaluated them, based on our derived requirements. Our literature review indicates that efficient NFV-RA algorithms are still needed to address the three challenges, discussed in Section 1.2.

Chapter 3

3. [1]Survey of the Tactile Internet

3.1 Introduction

The Tactile Internet is known as the next generation of the Internet that is expected to offer a plethora of new applications including teleoperation and remote robotic surgery. Unlike the previous generations of the Internet (e.g., IoTs), the Tactile Internet aims to enable haptic communications by allowing real-time human-to-machine interactions over the networks. More specifically, using the Tactile Internet, humans will be able to interact with remotely located machines in real-time via not only visual and auditory sensations, but also the sensation of touch. It will revolutionize how humans interacts with machines.

Due to its potential applications, the Tactile Internet has spurred a great deal of interest in both academia and industry and is gaining more and more momentum. Back in 2014 when the term Tactile Internet was coined, only one paper was published on the topic, and by early 2020, the number has increased to 135. As mentioned earlier, dedicated standards are being developed by the IEEE 1918.1 working group. Reference [4] gives an overview of this standardization effort. The first goal is to produce a baseline standard that includes the terminology, application scenarios,

[1]This chapter is based on a published paper [1]. N. Promwongsa *et al.*, "A Comprehensive Survey of the Tactile Internet: State-of-the-Art and Research Directions," *IEEE Commun. Surv. Tutor.*, vol. 23, no. 1, pp. 472–523, Firstquarter 2021.

requirements, and even the basic architecture(s). More specific standards, such as haptic codecs, are also under being developed [25].

There are several surveys and tutorials among the numerous papers published so far on the Tactile Internet. However, to the best of our knowledge, none of them covers both architectures and algorithms. To address these lacunae, we provide a survey of the architectures and algorithms proposed to date for the Tactile Internet in this chapter [1]. In addition, we critically review them and discuss some insights and lessons learned as well as the most promising research directions.

The rest of this chapter is organized as follows: we first propose a classification of the solutions proposed to date for the Tactile Internet. Then, we critically review the architectural solutions and derive insights and lessons learned from the literature review followed by the algorithmic solutions for the Tactile Internet. After that, we discuss the potential research directions. Finally, we conclude this chapter.

3.2 Literature Classification

We have identified a grand total of seventy-five papers to be reviewed in this survey. These papers are classified into four main categories: (i) architectures, protocols, and intelligent prediction, (ii) radio resource allocation algorithms, (iii) non-radio resource allocation algorithms on lower layers (i.e., PHY and MAC layers), and (iv) non-radio resource allocation algorithms beyond lower layers for the Tactile Internet. It should be noted that the intelligent prediction covers the works that aim at tackling the intelligence challenges as discussed in Section 1.4.1. We include them in the same category as architectures and protocols given that they use machine learning techniques to intelligently predict tactile/haptic information (e.g., control commands and tactile/haptic feedbacks), which is expected from a remote functional entity and not received on time.

The proposed taxonomy of the reviewed papers is illustrated in Figure 3.1. The first category comprises of the following four subcategories: (i) architectures for non-specific applications, (ii) architectures for specific applications, (iii) protocols, and (iv) intelligent predication schemes. In the second category (i.e., radio resource allocation algorithms), four subcategories are identified: (i) uplink transmission, (ii) downlink transmission, (iii) joint uplink and downlink transmission, and (iv) network slicing. The third category (i.e., non-radio resource allocation on lower layers), is further classified into three subcategories: (i) resource allocation in wireless local area network

(WLAN) and wireless body area network (WBAN), (ii) wavelength and bandwidth allocation in passive optical network (PON), and (iii) PHY layer. Finally, the fourth and last category (i.e., non-radio resource allocation beyond lower layers) consists of five subcategories: (i) edge resource allocation, (ii) NFV-RA, (iii) co-design, (iv) routing, and (v) switching.

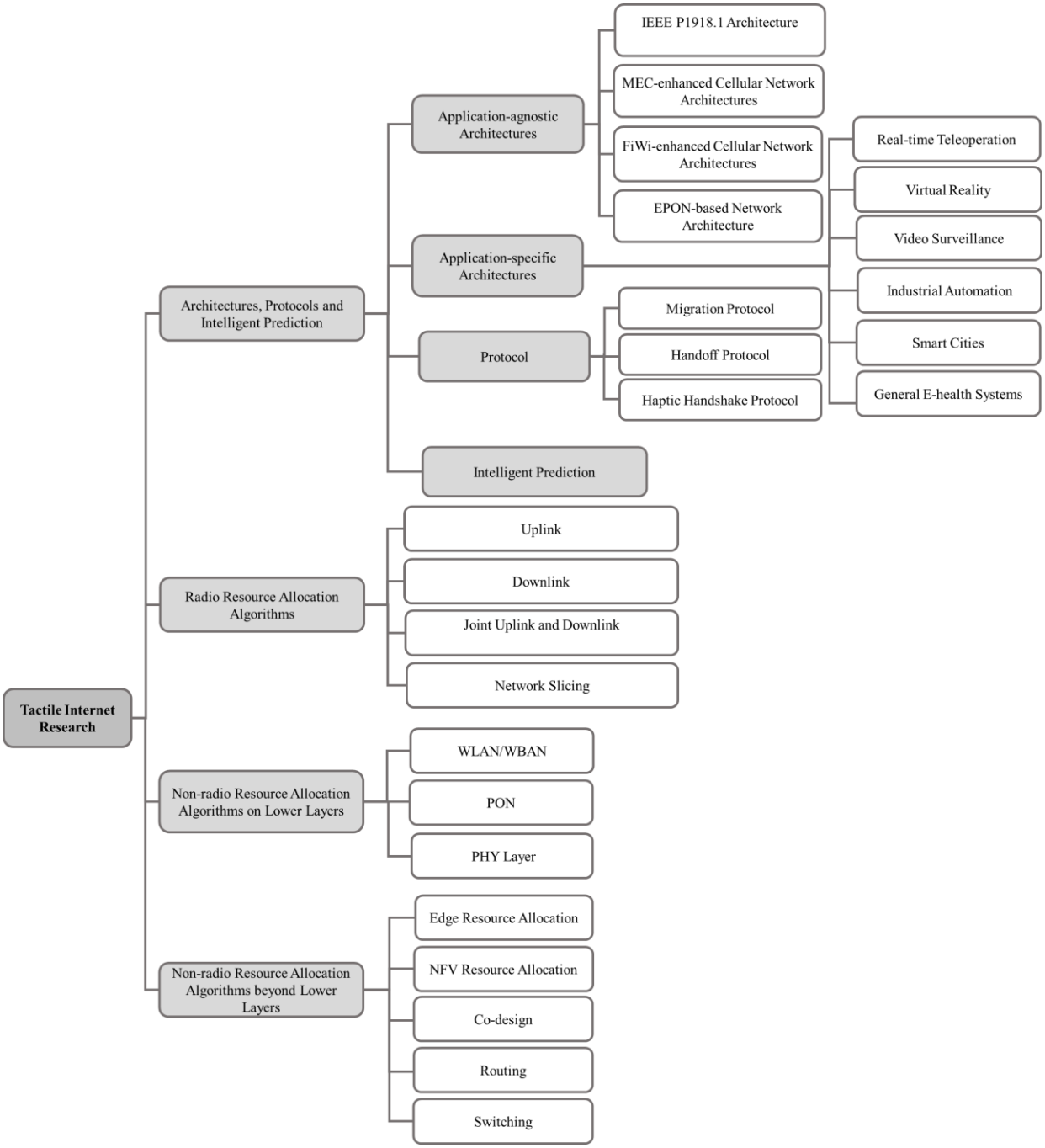


Figure 3.1: Overview of the surveyed research works according to the proposed classification system.

3.3 Architectures, Protocols and Intelligent Predication

Several works have introduced architectures for the Tactile Internet. All of these architectures are either application-agnostic or application-specific. Some works have also focused exclusively on the protocol aspects of architectures without proposing full-blown architectures. In addition, some of them have proposed intelligent prediction schemes to tackle the critical issue of haptic/tactile information (e.g., control command, haptic/tactile feedbacks) that is expected by remote functional entities and does not arrive on time. In the following subsections, we critically review these works. After that, we identify some promising insights and lessons learned.

3.3.1 Summary

While multiple works have introduced application-agnostic and application-specific architectures for the Tactile Internet, a few have proposed protocols as architectural building blocks as well as intelligent prediction schemes.

A standard architecture has been proposed by IEEE P1918.1[4]. It aims at interoperability and defines functional entities and interfaces. However, the implementation of the functional entities is outside the scope of standardization. Four of the applications-agnostic architectures harness MEC in a cellular environment [61][62][63][64], one of them harnesses Fi-Wi in a cellular environment [65], and the sixth relies on Ethernet passive optical network (EPON) [66]. The first application agnostic architecture [61] that harnesses MEC in a cellular environment relies on a multi-level cloud system for offloading. The second [62] builds on the first by leveraging SDN in addition to cloud computing. The third [63] relies on a cloud radio access network for offloading and the fourth [64] proposes a new low latency approach for service function chaining. The only architecture proposed to date for harnessing Fi-Wi in cellular environment aims at enhancing LTE-Advanced (LTE-A) heterogeneous network (HetNet) with Fi-Wi access for high capacity and offloading purposes [65]. On the other hand, the architecture based on next-generation EPONs (NG-EPON) allows optical network units to transmit over multiple wavelength channels at the same time to improve the network capacity [66].

Two of the proposed application-specific architectures focus on real-time tele-operations [67][68], two on VR [69][70], one on real-time video surveillance [71], one on industrial automation [72] and one on smart city applications [73], and one on general health applications [74]. One of the architectures that deals with tele-operations [67] combines the EPON-based

architecture presented in reference [65] with AI-enhanced MEC. Another relies on sub-GHz technology for the transmission of tactile feedback and robotic control [68]. As far as VR is concerned, one of the proposals builds on MEC and mmWave communications for a multiplayer VR game [69]. The second [70] builds on fog computing for a remote VR phobia treatment application. For video surveillance, a proposed architecture [71] aims at a system with very stringent requirements. The architecture for industrial automation [72] aims at providing a testbed for evaluating the use of NFV and SDN networks for industrial applications. The architecture for smart city applications [73] not only aims at achieving low latency and high reliability, but also at enabling high QoE. Finally, the sixth one relies on EPONs and aims at health applications in a campus hospital [74].

A migration protocol over MEC [75] and a handoff protocol [76] are the two protocols proposed so far for Tactile Internet, besides the protocols proposed for the haptic aspects [77]. The migration protocol aims at transferring only the application state over MEC, with a goal of zero downtime [75]. The second protocol produces a handoff between vehicles and infrastructure, with the assumption that the vehicles can be connected to both LTE and 802.11p networks [76]. A recent haptic communication protocol worth mentioning is the haptic handshake scheme proposed in reference [78]. It aims at orchestration between heterogeneous tactile devices.

As for the intelligent prediction schemes, multi-layer perceptron (MLP)-artificial neural networks (ANNs) is exploited in [67] and [79] to predict the delayed and/or lost samples in the feedback path of a teleoperation system. It is shown that the proposed edge sample forecaster is capable of making accurate predictions within the 1 ms deadline. Moreover, after training a hidden Markov model (HMM) model with a set of expert force/torque profiles, the authors of [80] made predictions using a Gaussian mixture regression during a remote needle insertion in robotic telesurgery applications. Further, the authors of [81][82][83] have designed a two-stage AI model, where a set of ANN-based binary classifiers are used along with reinforcement learning to forecast the haptic feedback.

3.3.2 Insights and Lessons Learned

Five insights and lessons learned are identified from reviewing the architectural, protocol and intelligent prediction solutions. The first insight is that the number of papers published so far on the architectural aspects of the Tactile Internet is relatively low: a grand total of 15 papers from

2014 to early 2020, 7 on application-agnostic architectures and 8 on application-specific architectures. In comparison, the number of papers published on fog computing through 2017 (i.e., 2013-2017) was 32, 19 on application-agnostic architectures and 16 on application-specific architectures [84]. A lesson we can learn from this is that the architectural aspects of the Tactile Internet are still being investigated. Much more research is needed, especially on the application-agnostic aspects.

Second, the most popular techniques used to date for meeting the requirements are offloading and running components at the edge. Offloading consists of offloading either computation [61][62][69][71] or traffic [63][65]. This is currently the case for both application-agnostic architectures [61][62][63][65] and application-specific ones [69][71]. In [67], [64], [70] [72], the technique consists of running components at the edge. However, the outcome so far is not compelling, as none of these proposals meets both requirements for all Tactile Internet applications. A lesson we can learn here is that research in this area needs to go beyond offloading and running components at the edge of the network. Some of the potential avenues will be sketched in the research direction subsection.

A third insight is that most of the available works do not provide experimental validation, even though experimental validation is a sine qua condition for any sound architecture design. Instead, they rely on simulations and do not even include proof-of-concept prototypes. The three notable exceptions are [64], [70] and [72], which include prototypes and measurements made on the prototypes. The lesson we can infer from this observation is that researchers should aim more and more at comprehensive architectures that include experimental validation.

The fourth insight is from the protocol perspective. The protocol works remain scarce, aside from the protocols for haptic communications which are well discussed in [77]. To date, this area only contains two proposals, one targeting the Tactile Internet at large [75] and one focusing on a specific Tactile Internet application, vehicular networks [76]. Clearly, more research is required in this area.

An important insight learned from the intelligent prediction schemes is the feasibility of using ANN-based classifiers and predictors in the context of haptic-enabled VR, teleoperation, and telesurgery applications. Note, however, that the research in this avenue is still in its infancy and much further efforts are needed to design more sophisticated prediction modules with the capability of being generalized to a larger group of applications and user behaviors.

3.4 Radio Resource Allocation Algorithms

Significant research efforts have focused on radio resource allocation algorithms to achieve ultra-low latency and ultra-high reliability in cellular radio access networks (RANs) for the Tactile Internet. These algorithms can be further classified as: (i) uplink transmission, (ii) downlink transmission, and (iii) joint uplink and downlink transmission algorithms. Furthermore, some works have leveraged on the network slicing concept to design efficient radio resource allocation schemes by creating multiple logical radio networks to guarantee the unique requirements of Tactile Internet applications. We first review these algorithms, and then identify potential insights and important lessons learned.

3.4.1 Summary

In cellular wireless networks, a key challenge to support Tactile Internet applications is to address the problem of multiple access, especially in the uplink direction, where the user equipment (UE) needs to enter a scheduling and grant procedure. Typically, this procedure starts when a UE sends a grant request to a base station (BS) and then waits for the scheduling grants (i.e., radio resources) before transmitting its data. In the downlink direction, however, the BS transmits the downlink data immediately after the data reaches the buffer. Clearly, the contribution of the uplink latency in total end-to-end latency in RANs is much larger than that of the downlink latency, especially given that the contention in the uplink direction may lead to failures and retransmissions. This has spurred a great deal of interest in designing low-latency radio resource allocation algorithms for the uplink direction. Both grant-based [85][86][87] and grant-free algorithms [88][89][90] [91][92] for the uplink direction are proposed for the Tactile Internet. In grant-based schemes, the BS is responsible for arbitrating the channel access in the uplink direction, and the allocation is carried out only after it receives the users' grant requests. In contrast, grant-free schemes aim to eliminate the scheduling and grant procedures, thus accelerating the channel access process in the uplink direction. The proposed algorithms in this group are based on dedicated reservation, soft reservation, predictive reservation, semi-persistent scheduling, or deep learning.

Although downlink delay is relatively smaller than uplink delay in RANs, the downlink delay may not be negligible, especially since the downlink queuing delay could grow longer as the incoming aggregated traffic to the BS increases. In the context of the Tactile Internet, two works

[93][94] address the radio resource allocation problem in the downlink direction. The main idea in these works is to adopt a queue-scheduling mechanism at a BS to ensure the timely delivery of delay-sensitive Tactile Internet packets.

Several papers [95][96][97][98][99][100][101][102] have considered joint uplink and downlink transmission, which stands as a very important research direction, especially for haptic communications with its inherent bidirectional exchange of haptic information. The works in this category mainly consider communication for general Tactile Internet applications, haptic communication, frequency division duplex-based device-to-device (D2D) communication, non-orthogonal multiple access (NOMA)-based D2D communication and vehicular communication.

There are three network slicing-based radio resource allocation algorithms proposed for the Tactile Internet [63][103][104]. These works are mainly based on reinforced learning, game theory, and two-level MAC scheduling to provide radio network slices to serve Tactile Internet applications.

3.4.2 Insights and Lessons Learned

One overall insight from our literature review of radio resource allocation algorithms for the Tactile Internet is that most of the studies in this context focus on resource allocation in orthogonal frequency division multiple access (OFDMA)-based cellular systems. These studies mainly aim at allocating orthogonal resources to the users to avoid interference while considering delay and reliability either as the main objective functions to be optimized or as the constraints to be satisfied. We note, however, that the main drawback of these systems is their spectral efficiency. For this reason, OFDMA-based cellular systems may not be appropriate to support the Tactile Internet, especially for large numbers of users. To cope with this drawback of OFDMA-based systems, NOMA is an emerging concept that enables non-orthogonal resource allocation among multiple users and utilizes sophisticated inter-user interference cancellation at the receiver side to separate the signals from different users. Not only is NOMA capable of supporting a larger number of users than OFDMA, it can also achieve smaller transmission delay, as it does not necessarily require a scheduling and grant procedure for uplink transmission [105]. Despite its potential, only three studies [91][96][99] consider NOMA-based cellular systems for the Tactile Internet. A lesson learned here is that more effort is needed on designing radio resource allocation for NOMA-based cellular system to unleash the potentials of NOMA in realizing the Tactile Internet.

Yet another insight is that the studies that could potentially meet both Tactile Internet requirements (i.e., [85][100][101]) rely on short packet transmission. These studies are essential for haptic communications, because packets generated by haptic devices are generally very small in size (e.g., 8, 24, and 48 Bytes for 1-, 3- and 6-degree of freedom (DoF) devices, respectively [4]). Nevertheless, we note that some Tactile Internet applications may require larger packet sizes (e.g., 1.5 kB for video traffic in VR applications [4]). Therefore, the above-mentioned studies may not be suitable for large-size packet transmission. This is more critical for the ultra-high reliability requirement, as packet error rate can easily grow as packet size increases. The lesson we can infer from this insight is that further in-depth studies are needed to design proper radio resource allocation schemes for large-size packet transmission in the context of the Tactile Internet.

A third insight is that only a few works (e.g., [63][94]) have tapped into the significant potential of low-cost data-centric Ethernet technologies (e.g., Wi-Fi) in conjunction with the coverage-centric cellular mobile networks to increase the overall network capacity. This leads to the conclusion that there is a need to complement cellular networks with already widely deployed Wi-Fi access networks as well as other technology networks. This will open the possibility to tap into the unlicensed frequency band, thus helping meet the Tactile Internet requirements at a lower cost.

3.5 Non-Radio Resource Allocation Algorithms on Lower Layers

In this subsection, we compile the research on non-radio resource allocation algorithms that mainly focus on lower layers (i.e., PHY and MAC layers) for the Tactile Internet. These algorithms include bandwidth allocation and scheduling in WLAN, WBAN, and PON for the Tactile Internet. They also cover algorithms whose main focus is on the physical layer (PHY layer) and which go beyond shortening the air interface to realize ultra-low latency communications. This subsection is concluded by presenting the insights and important lessons learned from this review.

3.5.1 Summary

A number of researchers have focused on introducing WLAN [106][107][108][109] and WBAN [110][111] technologies as potentially alternative wireless access networks to serve Tactile Internet applications due to their low cost, low power consumption, and wide deployment. Some mechanisms have been developed to enhance the WLAN and WBAN technologies to meet the stringent Tactile Internet requirements, as conventional mechanisms cannot fulfill these

requirements. More specifically, four of them focus on WLANs, aiming to either propose an efficient mechanism to enhance the existing IEEE 802.11 MAC protocols (i.e., hybrid coordination function-controlled channel access (HCCA), enhanced distributed channel access (EDCA)) [106][107] or to design novel MAC schemes [108][109] to meet the strict Tactile Internet requirements. As for the other two works targeting WBANs, the enhanced SmartBAN MAC protocol is proposed in [110], and an efficient resource allocation framework is proposed in [111] for haptic communications over body area nanonetworks (WBANN).

Due to their high bandwidth and flexibility, PONs are also a promising candidate to realize low-latency Tactile Internet applications, especially for fixed users. Several algorithmic works [66][74], [112][113][114][115] have studied the problem of scheduling and bandwidth allocation for PONs. These works consider a similar architecture, which mainly consist of multiple optical network units (ONUs) integrated with wireless access point (APs) to interact with wireless access networks. All ONU/APs are also connected to a single optical line terminal (OLT) via fiber links. In addition, the OLT is co-located with a tactile control server at a central office (CO). Considering this specific architecture, the main objective of these works is to introduce an efficient bandwidth allocation mechanism to ensure ultra-low latency and ultra-high reliability communication between the ONU/APs and the tactile control server. The existing dynamic wavelength and bandwidth allocation algorithms in PONs for the Tactile Internet are either non-predictive [66][112] or predictive [74][113][114][115]. The non-predictive algorithms mainly aim to optimally allocate the bandwidths based on the requested bandwidths reported from each ONU in each polling cycle while prioritizing Tactile Internet services to ensure their requirements compared to other services. On the other hand, the predictive algorithms allow the bandwidths to be proactively allocated based on traffic prediction to further reduce latency. The predictive algorithms rely on either Bayesian and maximum-likelihood sequence estimation methods [74] or machine learning-based methods [113][114][115] for traffic prediction.

As discussed in [4], an important step towards achieving 1 ms latency in LTE cellular networks is to reduce the air interface by shortening transmission time interval (TTI) in the PHY layer to allow for an accelerated user scheduling. Shortening the TTI can also improve the reliability, as the number of retransmission opportunities within the latency bound can increase. However, this improvement comes at the expense of increased subcarrier spacing. Therefore, tackling other aspects of the PHY layer rather than just shortening the TTI remains a key challenge to achieve

ultra-low latency and ultra-high reliability. Several algorithmic works have tried to improve the latency and reliability from a PHY layer point of view. These works include proposing multi-input multi-output systems for the Tactile Internet [116][117], mmWave communications for haptic-enabled VR applications [118], a novel modulation technique [119], reliable and low-latency fronthaul network [120] and network coding for the Tactile Internet [121][122].

3.5.2 Insights and Lessons Learned

Despite initial efforts on realizing the Tactile Internet over WLAN and WBAN technologies, none of the studies has considered joint resource allocation in both uplink and downlink directions. This aspect is particularly crucial for haptic communications due to their bidirectional exchange of control commands in the forward path (i.e., from the master domain to the slave domain) as well as haptic feedback in the reverse path (i.e., from the slave domain to the master domain). Therefore, allocating the resources in each direction independently from each other may not be an efficient approach to ensure the timely real-time exchange of control commends and haptic feedback for a frictionless, immersive haptic communication. The lesson we can learn from this insight is that the joint resource allocation in both uplink and downlink directions in WLAN and WBAN should be further investigated to efficiently meet the stringent Tactile Internet requirements. This could push forward WLAN and WBAN technologies as potential alternative wireless access technologies for the Tactile Internet.

Another insight on WLAN and WBAN resource allocation is that none of the studies has addressed the reliability aspect which is also critical for Tactile Internet applications. We note that although most of the proposed algorithms rely on contention-free mechanisms [106][108][109] to avoid packet collision, packet loss can still occur due to erroneous transmission caused by excessive path loss, given that packet dropping occurs mainly due to buffer overflow [107][111]. The lesson we can infer from this is that the reliability aspect of the WLAN and WBAN technologies must be further investigated in order to fulfill the Tactile Internet reliability requirement.

In the PHY layer, [120] shows that the application of erasure coding on the MAC frames transported by the fronthaul network can reduce the transport overhead significantly. Further, multi-path transmission can be used along with erasure coding to allow the central unit to split the original MAC frames into smaller blocks and send them over multiple paths, thereby helping to

achieve improvements over the conventional single-path transmission fronthaul solution. Despite all these efforts, further investigations are needed to achieve the ultra-high reliability requirement.

Finally, to cope with the emerging haptic-enabled VR systems' need for ultra-high data rates, one viable solution is to leverage on the mmWave frequency bands via mmWave communications. Investigating the feasibility of using mmWave communications to realize haptic-enabled VR applications has begun to attract real interest (e.g., [118]). Despite all the benefits that mmWave communications may offer in terms of ultra-high data rates and security, such systems still suffer from a major disadvantage. Specifically, the narrow beamwidth makes the mmWave system susceptible to link misalignment, which may deteriorate the reliability-latency performance. This reinforces the lesson that it is necessary to investigate the feasibility of smart beamwidth adaptation techniques designed to stabilize the link performance of mobile end-users.

3.6 Non-Radio Resource Allocation Algorithms beyond Lower Layers

This subsection focuses on non-radio resource allocation algorithms that go beyond the lower layers. They cover the allocation of edge resources including computation, storage, and communication resources for computation task offloading, NFV-RA, co-design algorithms, and algorithms on the traffic routing and switching aspects for the Tactile Internet. We first summarize these works, and then identify some key insights and discuss the important lessons learned. It should be noted that the works on NFV-RA are not included in this subsection, as they are discussed in Chapter 2.

3.6.1 Summary

Typically, mobile users have limited computation resources, and thus local processing of computation-intensive, low-latency tasks at the mobile devices may not meet the stringent requirements of the Tactile Internet. A recent viable approach is the so-called computation task offloading, where mobile users can offload their computation-intensive tasks to computing and storage resources—variously referred to as cloudlets, micro datacenters, or fog nodes—which are placed at the Internet's edge in proximity to wireless end devices in order to achieve low end-to-end latency, low jitter, and scalability. Several papers have studied the feasibility of leveraging on computation task offloading in the Tactile Internet context. Their main objective is to efficiently utilize resources (i.e., computation, storage, and communication) at the network edge to help end-

users with limited resources meet the strict Tactile Internet requirements. These studies include the computation resource allocation over fog systems [123][124], resource sharing among mobile edge devices [125], joint computation and radio resource allocation [126], and content edge caching [127] for the Tactile Internet.

Given the multi-disciplinary nature of the Tactile Internet, co-design approaches are an interesting group of algorithms with which to tackle the latency and reliability challenges by taking into account the crucial aspects of communication, computation, and control. Despite its paramount importance, this area of research has not received much attention. While four works [128][129][130][131] have leveraged the interdependence between the communication and control domains of telerobotic systems to increase the QoE, one work [132] has considered a communication-computation approach.

Due to its stringent QoS requirements, the Tactile Internet also requires novel routing schemes that consider the coexistence between haptic traffic and conventional triple-play traffic. To date, three studies have aimed to design novel routing algorithms for the Tactile Internet. The works that consider the routing aspect mostly aim to address the reliability. The proposed routing algorithms are based on multi-plane routing (MPR) [133][134] and minimum spanning tree [135] approaches. The MPR protocol outperforms intra-domain routing protocols (e.g., open shortest path first (OSPF)). The dynamic edge-elimination-based algorithm [135] can efficiently construct a rooted delay-constrained minimum spanning tree framework for message broadcasts while ensuring the delay constraints.

Finally, two papers have proposed scheduling policies in switches and/or routers to efficiently handle tactile traffic. Reference [136] focuses on cut-through switching-based networks, while Reference [137] investigates delay-constrained input-queued switching using virtual output queueing for the Tactile Internet.

3.6.2 Insights and Lessons learned

An insight on edge resource allocation is that most of the studies do not consider user mobility. These studies may not be suitable for Tactile Internet applications that require (almost) constant mobility (e.g., autonomous driving) because computation resources need to be allocated according to the user's movement in order to meet the stringent latency and reliability requirements. For instance, in autonomous driving, the application migration and edge resource allocation to run an

application server have to be made according to the movement of vehicular users. If not, the stringent latency and reliability requirements between the application server and the vehicular users are not likely to be met. The lesson we can learn from this insight is that more research efforts are needed on mobility management while allocating resources at the edge for the Tactile Internet.

From a routing perspective, the first insight is that the MPR scheme can ensure that the QoS requirements of haptic packets are met while maximizing the total network flow as well as minimizing the routing cost (e.g., [133][134]). It is also shown that shortest-path algorithms cannot handle haptic traffic. An optimized MPR algorithm has achieved significant improvements over equal-cost multi-path algorithm in terms of throughput, delay, and packet delivery ratio [134]. We note, however, that optimization approaches to find the best routing plane in an MPR scheme may add extra computational burdens, especially for large-sized graphs. This mandates the need for designing efficient heuristics to solve achieve (sub)optimal optimized MPR solutions to realize ultra-reliable, low-latency routing for haptic traffic.

Another important insight from a routing perspective is that none of the existing studies has tackled the problem of traffic routing for wireless multi-hop networks. This is especially important for the cases where establishing a direct communication link between the BS/AP and the mobile users becomes infeasible. This might be also necessary in case of excessive network congestion in the core network, which may result in an unacceptable end-to-end latency. In these cases, a traffic routing mechanism is needed to establish an efficient routing path using D2D-enabled wireless multi-hop networks in order to meet the stringent Tactile Internet requirements. The lesson we can learn from this insight is that more research efforts on this aspect are still needed to realize the Tactile Internet over D2D-enabled multi-hop wireless access network.

3.7 Research Directions

Based on the lessons learned we discuss above, we present some potential research directions for the Tactile Internet. We summarize them in Table 3.1 and Table 3.2. Further discussions on each of these research directions can be found in [1]. It should be noted that exhaustiveness is impossible due to the richness and novelty of the topic.

Table 3.1: Research directions on architectures, protocols and intelligence predictions.

Scope	Potential Research Directions
Architectures	• Architectures for enabling application components to be run on other devices than MEC servers • Architectures for enabling parallel processing of traffic that goes through VNFs in 5G: • Architectures for enabling streams, generated by multi-stream transport protocols, to be transmitted through multiple paths in networks
Protocols	• Designing new transport protocols (to be implemented in the user space) with having ultra-low latency as the key design goal. • Designing application-specific transport protocols (to be implemented in the user space), with the specific requirements of an application, including ultra-low latency, incorporated as part of the objective.
Intelligence Predictions	• Investigating machine learning techniques in actual remote scenarios with realistic network conditions. • Investigating machine learning techniques in high DoF teleoperation use cases • Conducting subjective and objective studies to allow for both qualitative and quantitative measuring of how closely a human operator is coupled with the involved experience enhanced by machine learning-based prediction.

Table 3.2: Research directions on algorithms.

Scope	Potential Research Directions
Radio Resource Allocation	• Radio resource allocation with mobility management • Radio resource allocation in NOMA-based cellular systems • Radio resource allocation for large-size packet transmission • Radio resource allocation in cellular networks in coordination with other low-cost networks
Resource Allocation in WBAN/WLAN	• Joint resource allocation in both uplink and downlink directions • Ultra-reliable resource allocation
Edge Resource Allocation	• Edge resource allocation with mobility management
NFV-RA	• Joint design-based NFV-RA • Real-time NFV-RA
Routing	• Traffic routing in wireless multi-hop networks

3.8 Conclusion

The Tactile Internet will completely revolutionize the way humans communicate once it becomes a reality. The main challenges are ultra-low latency and ultra-highly reliable communications. However, the Tactile Internet is still in its early stages. Research on all of its aspects is still needed to overcome its unique challenges. This chapter provides a survey of the solutions proposed to date in the literature on this topic. The solutions covering architectures, protocols, intelligent predication and algorithms were then critically reviewed. In addition, insights and lessoned from our literature review were derived. Finally, we discussed the remaining challenges and promising research directions in this topic.

Chapter 4

4. [2]Joint VNF Composition and Embedding for Ultra-Low Latency Services

4.1 Introduction

Provisioning ultra-low latency NSs for the Tactile Internet in an NFV-based infrastructure remains a challenge, as these NSs require the stringent latency requirements. To meet such stringent latency requirement, in some cases, traffic of a NS should be able to split into sub-flows, and multiple instances of each required VNF type should be deployed in order to process a fraction of the split traffic in parallel to share the processing load, thereby accelerating the processing delay [12][13]. However, this requires VNF composition and embedding stages to be jointly determined. Moreover, relying on multiple instances of each required VNF type to serve a NS could decrease reliability, as the number of NFV-enabled nodes involved in hosting these VNF instances may increase.

To address this challenge, we study a joint VNF composition and embedding problem for ultra-low latency NSs in this chapter. We aim at optimally determining the number of instances of each

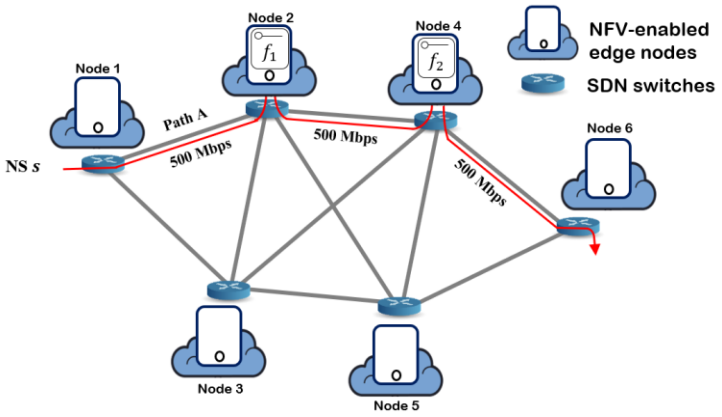
[2]This chapter is based on a published paper [17]: N. Promwongsa *et al.*, "Ensuring Reliability and Low Cost When Using a Parallel VNF Processing Approach to Embed Delay-Constrained Slices," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 4, pp. 2226–2241, Dec. 2020.

VNF type to serve a NS and where its traffic of a NS should be split and processed by these instances embedded into physical nodes to guarantee a stringent latency requirement while optimizing reliability degradation and cost. We formulate the problem as a mixed integer nonlinear programming (MINLP) problem, which takes VNF composition, VNF embedding and traffic routing into account. Based on our observation, MINLP formulation can be also simplified into a MILP formulation when all physical nodes have the same reliability. Due to the complexity of the formulated problem, we propose a Tabu search-based algorithm to find a high-quality solution within feasible time and evaluate it through extensive simulations.

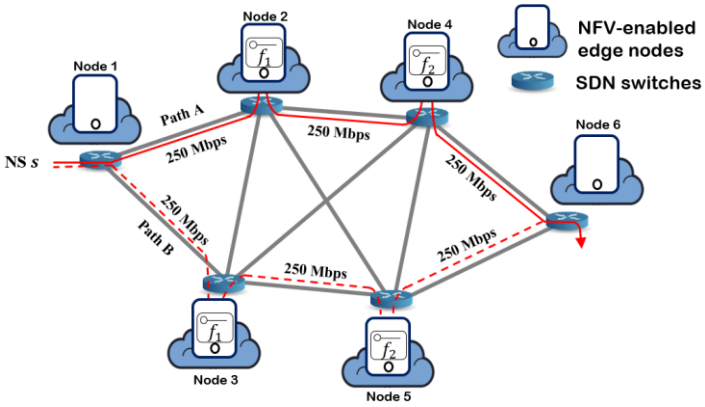
The rest of this chapter is organized as follows. First, we present the motivating scenario followed by the description of the system model and the problem formulations. Then, we describe our proposed Tabu search-based joint VNF composition and embedding algorithm. After that, we discuss the performance evaluation of the proposed algorithm. Finally, we conclude this chapter.

4.2 Motivation Scenario

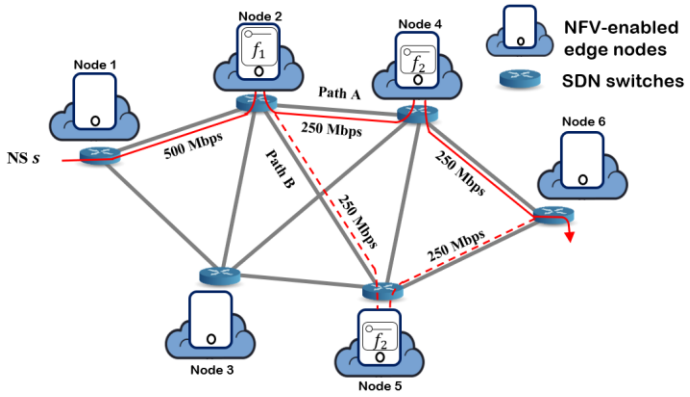
In this subsection, we demonstrate our joint VNF composition and embedding problem. First, let us assume that NFV-based infrastructure consists of six NFV-enabled nodes and ten links, as shown in Figure 4.1. We also assume that these NFV-enabled nodes reside in close proximity to the users, as embedding VNFs in remote NFV-enabled nodes might not be suitable for latency-sensitive applications due to high communication delay. Each node is associated with a node capacity and a reliability. For simplicity and without loss of generality, the node capacity and reliability of all the nodes are assumed to be 1 Gbps and 0.90, respectively. Moreover, we assume that the processing delay of each VNF depends on the resource utilization of its physical host node [12][138]. As suggested in [138], the processing delay model can be represented as piecewise linearization. To simplify the explanation here, we assume that this delay is computed by a single linear function of $1.5u_k$ + 0.2 ms, where u_k is the node utilization of node k. All links have a bandwidth of 2 Gbps, and the link delay is a constant value, 0.1 ms. The reliability of a NS is computed by a product of the reliability of all physical nodes hosting its VNFs [139][140]. Next, we assume that NS s arrives in the network. This NS requires its traffic of 500 Mbps to be transmitted from node k_1 and then processed by VNFs f_1 and then f_2 before it reaches node k_6. Also, the latency requirement of NS s is 2 ms.



(a)



(b)



(c)

Figure 4.1: Illustration of motivation of joint VNF composition and embedding for ultra-low latency NSs.

Provisioning a NS by deploying only one instance of each required VNF type with a single path may not allow a stringent latency requirement to be met. As illustrated in Figure 4.1a, the traffic of NS s is routed through Path A: $k_1 \rightarrow k_2 \rightarrow k_4 \rightarrow k_6$ and is processed by f_1 and f_2 at node

k_2 and node k_4, respectively. In this case, the obtained end-to-end latency is 2.2 ms, which does not satisfy the latency requirement, and the reliability is 0.81.

In order to meet the latency requirement of 2 ms, the traffic of NS s should be split and processed by multiple instances of each VNF type over different nodes, as illustrated in Figure 4.1b. The traffic is split into two paths – path A: $k_1 \rightarrow k_2 \rightarrow k_4 \rightarrow k_6$ and path B: $k_1 \rightarrow k_3 \rightarrow k_5 \rightarrow k_6$. The traffic in path A is processed by f_1 and f_2 at k_2 and node k_4, respectively while the one in path B is processed by f_1 and f_2 at k_3 and k_5, respectively. In this case, the obtained end-to-end latency is reduced to 1.45 ms, which satisfies the latency requirement. However, the reliability is decreased to approximately 0.66 because of the increase in the number of nodes hosting the VNFs.

In order to minimize the reliability degradation while satisfying the latency requirement, the number of physical nodes hosting VNFs should be as small as possible. Figure 4.1c shows an example of joint VNF composition and VNF embedding that meets the latency requirement while improving the reliability compared to Figure.4.1b. The traffic is routed first through $k_1 \rightarrow k_2$ and is processed by f_1 at k_2. Next, the traffic is split into two paths: path A: $k_2 \rightarrow k_4 \rightarrow k_6$ and path B: $k_2 \rightarrow k_5 \rightarrow k_6$. The traffic in path A is processed by f_2 at k_4 while that in path B is processed by f_2 at k_5. As a result, the obtained end-to-end latency is 1.825 ms which satisfies the latency requirement, and the reliability is close to maximal, at roughly 0.73.

Furthermore, processing and bandwidth costs may vary, depending on nodes and links, respectively. Therefore, to efficiently support ultra-low latency NSs, a joint VNF composition and embedding algorithm should optimize reliability degradation and cost while meeting the stringent latency requirements.

4.3 System Model

In our joint VNF composition and embedding problem, we represent a substrate network as a graph $\{\mathcal{K}, \mathcal{L}\}$ where $\mathcal{K}$ is a set of NFV-enabled nodes, and $\mathcal{L}$ is a set of physical links. Every node $k \in \mathcal{K}$ is associated with a node capacity p_k, processing cost per traffic unit c_k and reliability ρ_k. Similarly, every physical link $l = (k_1, k_2) \in \mathcal{L}$, has a link bandwidth b_l, bandwidth cost per traffic unit c_l and link delay d_l, where k_1 and $k_2 \in \mathcal{K}$. We define $\mathcal{S}$ as a set of NS requests. Every NS $s \in \mathcal{S}$ demands a traffic rate t^s between source node k^s_{src} and destination node k^s_{dst} to be processed by an ordered set of VNFs $\mathcal{N}^s$ within a latency bound δ^s, where k^s_{src} and $k^s_{dst} \in \mathcal{K}$. We also

define two binary parameters, μ_k^s and π_k^s to indicate whether physical node k is a source node or a destination node of NS s, respectively. Precisely, μ_k^s is 1 if $k = k_{src}^s$, while π_k^s is 1 if $k = k_{dst}^s$. Otherwise, both are zero. In addition, considering a given NS s, $\mathcal{E}^s$ is defined as a set of virtual links, including the virtual links between the virtual source node and the first VNF, and the last VNF and virtual destination node. For example, if $\mathcal{N}^s = \{f_1, f_2\}$, then $\mathcal{E}^s = \{(f_0, f_1), (f_1, f_2), (f_2, f_{|\mathcal{N}^s|+1})\}$, where f_0 and $f_{|\mathcal{N}^s|+1}$ represent virtual source and destination nodes, respectively. For each $e \in \mathcal{E}^s$, α_e^s and β_e^s are also defined to represent the origin and destination of virtual link e. For instance, if $e = (f_1, f_2)$, $\alpha_e^s = f_1$ and $\beta_e^s = f_2$. We assume that processing one unit of traffic rate requires one unit of node capacity of NFV-enabled nodes to host each VNF [141]. The required bandwidth of each virtual link $e \in \mathcal{E}^s$ is proportional to the traffic rate. Moreover, traffic flow of a NS is assumed to be splittable.

Table 4.1: Summary of key notations for the joint VNF composition and embedding problem.

Input Parameters	
$\mathcal{K}$	Set of NFV-enabled nodes
$\mathcal{L}$	Set of physical links
p_k	Node capacity of node $k \in \mathcal{K}$
c_k	Processing cost per of node $k \in \mathcal{K}$
ρ_k	Reliability of node $k \in \mathcal{K}$
b_l	Link bandwidth of link $l \in \mathcal{L}$
c_l	Link cost of link $l \in \mathcal{L}$
d_l	Link delay of link $l \in \mathcal{L}$
$\mathcal{S}$	Set of NSs
t^s	Traffic rate of NS $s \in \mathcal{S}$
$\mathcal{N}^s$	Ordered set of VNFs for NS $s \in \mathcal{S}$
k_{src}^s	Source node of NS $s \in \mathcal{S}$
k_{dst}^s	Destination node of NS $s \in \mathcal{S}$
δ^s	Latency requirement of NS $s \in \mathcal{S}$
$\mathcal{E}^s$	Set of virtual links of NS $s \in \mathcal{S}$
α_e^s	Origin VNF of virtual link $e \in \mathcal{E}^s$ of NS $s \in \mathcal{S}$
β_e^s	Destination VNF of virtual link $e \in \mathcal{E}^s$ of NS $s \in \mathcal{S}$
μ_k^s	A binary parameter, indicating whether node k is a source node of NS $s \in \mathcal{S}$
π_k^s	A binary parameter, indicating whether node k is a destination node of NS $s \in \mathcal{S}$

4.4 Problem Formulation

In this subsection, we first propose a problem formulation for a general case. After that, we present a simplified problem formulation for a case where all physical nodes have the same reliability.

4.4.1 Problem Formulation for a General Case

In our formulated problem, we define $\mathcal{I}^s = \{1, 2, .., |\mathcal{I}^s|\}$ as a set of sub-flows, where traffic of NS s can be split. $|\mathcal{I}^s|$ is the maximum number of sub-flows, which is an input parameter. It could also be considered as the maximum number of instances of each required VNF type that can be hosted in different physical nodes for NS s. It is important to note that the value of $|\mathcal{I}^s|$ does not imply that a NS is always split into $|\mathcal{I}^s|$ separate sub-flows that are mapped into different physical nodes and links. Instead, as many sub-flows of the same NS as possible are forced to be processed by the same VNF instance to ensure optimal reliability degradation. The sub-flows that are routed through the same physical nodes and links can be considered as a single sub-flow at the substrate network. For example, consider Figure 4.1c with $|\mathcal{I}^s|$ set to 4. The traffic of NS s is thus split into four sub-flows. However, three of them are routed through $k_1 \rightarrow k_2 \rightarrow k_4 \rightarrow k_6$. The fractions of the traffic split among these three sub-flows are 0.2, 0.1 and 0.1, respectively. Meanwhile, the fourth sub-flow is routed through $k_1 \rightarrow k_2 \rightarrow k_5 \rightarrow k_6$, and its fraction of the traffic split is 0.5. Therefore, eventually there will only be two actual sub-flows for this NS in the network.

All decision variables in this problem are defined as follows:

- $G_{f,k}^{s,i} \in \{0,1\}$ is a binary variable, indicating whether VNF type $f \in \mathcal{N}^s$ is hosted in node k for sub-flow i of NS s.
- $x_{f,k}^{s,i} \in [0,1]$ is a continuous variable, indicating a fraction of split traffic of sub-flow $i \in \mathcal{I}^s$ of NS s processed by VNF type $f \in \mathcal{N}^s$ on node k.
- $U_{e,l}^{s,i} \in \{0,1\}$ is a binary variable, indicating whether virtual link $e \in \mathcal{E}^s$ is mapped into link l for sub-flow $i \in \mathcal{I}^s$ of NS s.
- $y_{e,l}^{s,i} \in [0,1]$ is a continuous variable, indicating a fraction of split traffic of sub-flow $i \in \mathcal{I}^s$ of NS s on virtual link $e \in \mathcal{E}^s$ routed through link l.
- $H_{f,k}^{s} \in \{0,1\}$ is a binary variable, indicating whether node k is used to host VNF type f for NS s.
- $q_{f,k}^{s} \in [0,1]$ is a continuous variable, indicating the reliability of VNF type f hosted in node k for NS s.
- $\varphi^{s,i} \in [0,1]$ is a continuous variable, indicating a fraction of split traffic of sub-flow $i \in \mathcal{I}^s$ of NS s.

Next, we describe all the constraints considered in the joint VNF composition and embedding problem. Constraint (4.1) ensures that VNF f will be embedded into node k for sub-flow i of NS s only if the traffic of that sub-flow is processed by the VNF f in that node.

$$G_{f,k}^{s,i} - x_{f,k}^{s,i} \leq 1 - \epsilon, \forall s \in \mathcal{S}, \forall i \in \mathcal{I}^s, \forall f \in \mathcal{N}^s, \forall k \in \mathcal{K} \quad (4.1a)$$

$$G_{f,k}^{s,i} - x_{f,k}^{s,i} \geq 0, \forall s \in \mathcal{S}, \forall i \in \mathcal{I}^s, \forall f \in \mathcal{N}^s, \forall k \in \mathcal{K} \quad (4.1b)$$

where ϵ is a small constant value which should be less than the minimum traffic fraction allowed to represent traffic of a sub-flow. For instance, if the minimum traffic fraction is required to be at least 0.01 ($\varphi^{s,i} \geq 0.01$), then ϵ should be less than 0.01.

Constraint (4.2) ensures that entire traffic of NS s must be processed by the ordered set of VNFs $\mathcal{N}^s$.

$$\sum_{i \in \mathcal{I}^s} \sum_{k \in \mathcal{K}} x_{f,k}^{s,i} = 1, \forall s \in \mathcal{S}, \forall f \in \mathcal{N}^s \quad (4.2)$$

Constraint (4.3) ensures that only one instance of each required VNF type is deployed in the network for sub-flow i of NS s.

$$\sum_{k \in \mathcal{K}} G_{f,k}^{s,i} = 1, \forall s \in \mathcal{S}, \forall i \in \mathcal{I}^s, \forall f \in \mathcal{N}^s \quad (4.3)$$

Constraint (4.4) ensures that instances of different VNF types required by NS s must not be deployed on the same node in order to avoid a long recovery time if there is a node failure [142].

$$\sum_{f \in \mathcal{N}^s} G_{f,k}^{s,i} \leq 1, \forall s \in \mathcal{S}, \forall i \in \mathcal{I}^s, \forall k \in \mathcal{K} \quad (4.4)$$

Constraint (4.5) ensures that traffic of all sub-flows of NS s must be processed by all VNF types $f \in \mathcal{N}^s$.

$$\sum_{k \in \mathcal{K}} x_{f,k}^{s,i} = \varphi^{s,i}, \forall s \in \mathcal{S}, \forall i \in \mathcal{I}^s, \forall f \in \mathcal{N}^s \quad (4.5)$$

Constraint (4.6) ensures that any node hosting any VNF must not exceed its capacity.

$$\sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{I}^s} \sum_{f \in \mathcal{N}^s} t^s x_{f,k}^{s,i} \leq p_k, \forall k \in \mathcal{K} \quad (4.6)$$

Constraint (4.7) ensures that there is a continuous path for each sub-flow of NS s from its source to destination and guarantee that the path must pass through nodes that are hosting its VNFs and the ordered set of the required VNFs.

$$\sum_{l=(k_1,k_2)\in\mathcal{L}|k_1=k} U_{e,l}^{s,i} - \sum_{l=(k_1,k_2)\in\mathcal{L}|k_2=k} U_{e,l}^{s,i} = \begin{cases} \mu_k^s - G_{\beta_e^s,k}^{s,i} & if\ \alpha_e^s = f_0 \\ G_{\alpha_e^s,k}^{s,i} - \pi_k^s & if\ \beta_e^s = f_{|\mathcal{N}^s|+1} \\ G_{\alpha_e^s,k}^{s,i} - G_{\beta_e^s,k}^{s,i} & if\ \alpha_e^s \neq f_0 \\ & and\ \beta_e^s \neq f_{|\mathcal{N}^s|+1} \end{cases}, \quad (4.7)$$

$$\forall s \in \mathcal{S}, \forall i \in \mathcal{I}^s, \forall e \in \mathcal{E}^s, \forall k \in \mathcal{K}$$

Constraint (4.8) ensures that if any physical link is assigned to any virtual link of any sub-flow, the traffic of that sub-flow must be routed through that physical link. We note that although this constraint is non-linear; however, it can be simply linearized.

$$\varphi^{s,i} U_{e,l}^{s,i} = y_{e,l}^{s,i}, \forall s \in \mathcal{S}, \forall i \in \mathcal{I}^s, \forall e \in \mathcal{E}^s, \forall l \in \mathcal{L} \quad (4.8)$$

Constraint (4.9) ensures that any physical link assigned to any NS must not exceed its capacity.

$$\sum_{s\in\mathcal{S}} \sum_{i\in\mathcal{I}^s} \sum_{e\in\mathcal{E}^s} t^s y_{e,l}^{s,i} \leq b_l, \forall l \in \mathcal{L} \quad (4.9)$$

Constraint (4.10) ensures that the total latency of all sub-flows of NS s, including processing and communication delays, does not violate the latency requirement. Here, we follow the processing delay model used in [32][138], where the processing delay of executing any VNF depends on the processing loads of its host physical node and is presented as a convex function of node utilization based on the M/M/1 queuing model. The node utilization is a ratio of the total processing demands and the node capacity of the node. Piecewise linearization is then adopted to estimate the convex delay curve. The communication delay considered here is a fixed value. To ensure this constraint, we first define a helping variable $u_k \in [0,1]$ to represent the node utilization of physical node k, which can be computed by

$$u_k = \frac{1}{p_k} \sum_{s\in\mathcal{S}} \sum_{i\in\mathcal{I}^s} \sum_{f\in\mathcal{N}^s} t^s x_{f,k}^{s,i} \quad (4.10a)$$

Next, we define another helping variable $\omega_{f,k}^{s,i}$ to denote the processing delay of executing the VNF type f on physical node k for sub-flow i of NS s. The processing delay $\omega_{f,k}^{s,i}$ at node k must be zero if the required VNF of sub-flow i of NS s is not hosted in that node. Otherwise, $\omega_{f,k}^{s,i}$ depends on linear functions of the node utilization. This condition is enforced by constraints (4.10b) and (4.10c).

$$\omega_{f,k}^{s,i} \leq G_{f,k}^{s,i} \omega_k^{max}, \forall s \in \mathcal{S}, \forall i \in \mathcal{I}^s, \forall f \in \mathcal{N}^s, \forall k \in \mathcal{K} \quad (4.10b)$$

$$\theta_j u_k + \partial_j \leq \omega_{f,k}^{s,i} + \left(1 - G_{f,k}^{s,i}\right) \omega_k^{max}, \forall s \in \mathcal{S}, \forall i \in \mathcal{I}^s, \forall f \in \mathcal{N}^s, \forall k \in \mathcal{K} \quad (4.10c)$$

where θ_j and ∂_j are linear coefficients of a j^{th} linear function, and ω_k^{max} is an input parameter that denotes the maximum processing delay of node k. This input parameter can be obtained from the delay model by setting the node utilization equal to one. Finally, Constraint (4.10d) ensures that the sum of the processing and link delays does not violate the latency requirement of a NS.

$$\sum_{f\in\mathcal{N}^s}\sum_{k\in\mathcal{K}}\omega_{f,k}^{s,i}+\sum_{e\in\mathcal{E}}\sum_{l\in\mathcal{L}}d_l U_{e,l}^{s,i}\leq\ \delta^s,\forall s\in\mathcal{S},\forall i\in\mathcal{I}^s \tag{4.10d}$$

Constraints (4.11a), (4.11b) and (4.11c) indicate a value of variable $q_{f,k}^s$ in order to help computing the reliability of a NS. Precisely, $q_{f,k}^s = \rho_k$ if $H_{f,k}^s = 1$. Otherwise, $q_{f,k}^s = 1$.

$$\sum_{i\in\mathcal{I}^s}G_{f,k}^{s,i}\ \geq\ H_{f,k}^s\ ,\forall s\in\mathcal{S},\forall f\in\mathcal{N}^s,\forall k\in\mathcal{K} \tag{4.11a}$$

$$G_{f,k}^{s,i}\ \leq\ H_{f,k}^s\ ,\forall s\in\mathcal{S},\forall f\in\mathcal{N}^s,\forall k\in\mathcal{K} \tag{4.11b}$$

$$q_{f,k}^s=1-[1\ -\ \rho_k]H_{f,k}^s\ ,\forall s\in\mathcal{S},\forall f\in\mathcal{N}^s,\forall k\in\mathcal{K} \tag{4.11c}$$

The objective of our problem is to optimize the operational cost and the reliability degradation. The operational cost (C_{op}) consists of two components, processing cost (C_{proc}) and communication cost (C_{com}), which can be expressed in Eqs. (4.12) and (4.13), respectively.

$$C_{proc}=\sum_{s\in\mathcal{S}}\sum_{i\in\mathcal{I}^s}\sum_{f\in\mathcal{N}^s}\sum_{k\in\mathcal{K}}t^s c_k x_{f,k}^{s,i} \tag{4.12}$$

$$C_{com}=\sum_{s\in\mathcal{S}}\sum_{i\in\mathcal{I}^s}\sum_{e\in\mathcal{E}}\sum_{l\in\mathcal{L}}t^s c_l y_{e,l}^{s,i} \tag{4.13}$$

The reliability of a NS can be computed as the product of the reliability of all physical nodes that are hosting its required VNFs. Thus, the overall reliability of all NSs can be represented as

$$R=\sum_{s\in\mathcal{S}}\prod_{f\in\mathcal{N}^s}\prod_{k\in\mathcal{K}}q_{f,k}^s \tag{4.14}$$

Finally, our problem is formulated as an optimization problem with the objective to

$$min\ \ Obj=w_{cost}C_{op}-\ w_{rel}R \tag{4.15}$$

Subject to Constraints (4.1) to (4.11)

where $C_{op} = C_{proc} + C_{com}$. w_{cost} and w_{rel} are the cost and reliability weights, respectively, that can be tuned accordingly. To achieve our intended goal, w_{rel} should be much greater than w_{cost} to ensure that maximizing the reliability function (R) will be prioritized in the optimization, and the operation cost will be regarded as a secondary objective within the optimization. We denote this problem as the general problem (GP). GP is a MINLP problem as R is a non-linear function.

4.4.2 Problem Formulation for the Case Where Physical Nodes Have the Same Reliability

As discussed in our motivating scenario in Section 4.2, if all physical nodes have the same reliability, the reliability degradation could depend on the number of physical nodes involved in hosting VNFs for NSs. Therefore, minimizing the number of physical nodes used for each NS while being able to ensure its stringent latency requirement could optimize the reliability degradation in this case. Based on this observation, we simplify the objective function of the GP by minimizing the number of physical nodes used by each NS instead of maximizing the complex non-linear function R. The sum of the number of physical nodes used by all NSs can be presented as in the following equation.

$$M = \sum_{s \in \mathcal{S}} \sum_{f \in \mathcal{N}^s} \sum_{k \in \mathcal{K}} H_{f,k}^s \tag{4.16}$$

Therefore, the GP can be simplified to a MILP problem as follows:

$$min\ [w_{cost} C_{op} + w_{rel} M]$$
$$\text{Subject to Constraints (4.1) to (4.11)} \tag{4.17}$$

This MILP problem is denoted by the same reliability problem (SRP).

4.5 Tabu Search-Based joint VNF Composition and Embedding

The GP and the SRP consist of the VNF composition and VNF embedding problem. Generally, the VNF embedding problem falls into the same problem domain as virtual network embedding (VNE) [11], which is well-known to be NP-hard [143]. Therefore, the GP and the SRP are NP-hard. This implies that optimally solving a SRP becomes unaffordable for medium to large-scale scenarios. Even for small-scale scenarios, an exhaustive search algorithm takes quite a long time to find the optimal solutions, as will be seen in the next subsection. Moreover, to the best of our knowledge, a GP cannot be solved by CPLEX. Therefore, in this section, we propose a meta-heuristic algorithm to tackle these problems within a feasible execution time. Our proposed algorithm relies on Tabu search [144], a local search procedure that uses memory structure to track search history and avoid cycling and being trapped in local optima. Tabu search has also proven to provide outstanding sub-optimal solutions for the VNF embedding problem (e.g., [145][146]).

Our proposed Tabu Search-based joint composition and embedding (TJCE) algorithm is outlined in Algorithm 4.1. It starts from an initial solution (Φ_0). It then explores the search space by iteratively performing moves to transit from the current solution (Φ_{cur}) to an improving solution in its neighborhood if there is one available. If not, a non-improving solution can also be accepted to help the local search process escape from a local optimum. Initially, the current solution (Φ_{cur}) is set to be the initial solution (Φ_0). After that, it is set to be a solution that is accepted from the previous iteration. In order to avoid getting stuck in a local optimum and cycling back to the previously accepted solutions again, the moves of the previously accepted solutions are stored in its memory structure, called the Tabu list. This will prevent any selection of moves that would lead to the previously accepted solutions for a specific number of iterations. However, if those moves lead to solutions that are better than the best-known solution thus far, those moves could still be selected, and the solutions corresponding to those moves could still be accepted. This condition is called aspiration criteria. The search process is repeated until the best-known solution (Φ_{best}) does not improve for a certain number of consecutive iterations (j_{stop}). Next, we explain the components of our proposed TJCE algorithm in detail.

Algorithm 4.1 Tabu Search-based Joint Composition and Embedding Algorithm

Input: $\mathcal{S}$, $\mathcal{K}$, $\mathcal{L}$
Output: Φ_{best}
1: $\Phi_0 \leftarrow$ **generate** an initial solution by executing the IniSolGen function
2: $\Phi_{cur} \leftarrow \Phi_0$, $\Phi_{best} \leftarrow \Phi_0$, $j \leftarrow 0$
3: **while** $j < j_{stop}$ **do**
4: $\mathcal{H} \leftarrow$ **create** candidate neighbor solutions of Φ_{cur}
5: $h_{best} \leftarrow$ **select** from $\mathcal{H}$ the best solution in terms of the fitness value
6: **if** $\left[\xi(h_{best}) < \xi(\Phi_{cur})$ and h_{best}'s move is not Tabu $\right]$ or $\xi(h_{best}) < \xi(\Phi_{best})$ **then**
7: **add** h_{best}'s move to a Tabu list
8: **else**
9: $h_{best} \leftarrow$ **select** randomly the non-Tabu neighbor solution with probabilities according to the fitness value
10: **add** h_{best}'s move to a Tabu list
11: **end if**
12: $\Phi_{cur} \leftarrow h_{best}$
13: **if** $\xi(\Phi_{cur}) < \xi(\Phi_{best})$ **then**
14: $\Phi_{best} \leftarrow \Phi_{cur}$
15: $j \leftarrow 0$
16: **end if**
17: $j \leftarrow j + 1$
18: **end while**

4.5.1 Initial Solution

We design a simple VNF embedding algorithm to generate an initial solution (Φ_0). It aims at deploying only one instance of each VNF type required by a given NS on a node that has minimum resource utilization and then routing the traffic through the path with minimum delay. The pseudo-code of our proposed algorithm for initial solution generation is outlined in Algorithm 4.2. It works as follows: for each NS $s \in \mathcal{S}$, it starts finding the shortest path between a source and a destination based on the link delay such that the number of physical nodes in that path is equal to or greater than the number of required VNFs $|\mathcal{N}^s|$. We recall that different VNF types required by a given NS are not allowed to be hosted on the same physical nodes to avoid a long recovery time in case of a node failure [142]. Next, it finds a set of $|\mathcal{N}^s|$ physical nodes in the path that have the lowest resource utilization, and then it orders a set of nodes according to their position (i.e., location) in the path, from the source to the destination. This step ensures that the VNFs will be embedded in the correct order along the path. Each VNF $f \in \mathcal{N}^s$ is then embedded into a different physical node from the ordered set. Finally, all the virtual links are assigned to the path.

Algorithm 4.2 IniSolGen Algorithm

Input: $\mathcal{S}$, $\mathcal{K}$, $\mathcal{L}$
Output: Φ_0

for each $s \in S$ **do**
  $p \leftarrow$ find the shortest path based on the link delay from source to destination for NS s
  $\mathcal{K}_p \leftarrow$ **get** nodes in path p
  while $|\mathcal{K}_p| < |\mathcal{N}^s|$ **do**
    $p \leftarrow$ find the next shortest path
    $\mathcal{K}_p \leftarrow$ **get** nodes in path p
  end while
  $\mathcal{K}_{cd} \leftarrow \{\emptyset\}$
  while $|\mathcal{K}_{cd}| \neq |\mathcal{N}^s|$ **do**
    $k_{cd} \leftarrow$ **find** the node with the lowest node resource utilization in $\mathcal{K}_p$
    add k_{cd} to $\mathcal{K}_{cd}$
    remove k_{cd} from $\mathcal{K}_p$
  end while
  sort the order of nodes in $\mathcal{K}_{cd}$ according to their orders appearing in path p from source to destination
  for each $f \in \mathcal{N}^s$ **do**
    $k_{cd} \leftarrow$ **select** the first node in $\mathcal{K}_{cd}$
    embed VNF f into k_{cd}
    remove k_{cd} from $\mathcal{K}_{cd}$
  end for
  embed all virtual links $e \in \mathcal{E}^s$ into path p
end for

4.5.2 Neighborhood Structure

We define two moves to generate neighbor solutions for exploration in our proposed TJCE algorithm. The first one is the VNF relocation move which aims at re-embedding the VNF of each sub-flow of a NS to a new physical node in order to satisfy the strict latency requirement. The second one is the traffic splitting readjustment move designed to readjust the traffic fraction of sub-flows of a NS. The procedure for generating the neighbor solutions from these moves is explained as follows:

- VNF relocation: VNF $f \in \mathcal{N}^s$ of sub-flow $i \in \mathcal{I}^s$ of NS $s \in \mathcal{S}$ is randomly selected and then re-embedded into another candidate physical node. To find the best candidate physical node, we first find the shortest path between the physical nodes hosting the predecessor VNF and the successor VNF of VNF f. Then, a physical node in the path that has enough capacity and minimum resource utilization is selected as a candidate node to host VNF f. Finally, the virtual link between the predecessor VNF and the successor VNF of VNF f is re-embedded into the path that contains the candidate node.
- Traffic splitting readjustment: NS $s \in \mathcal{S}$ whose sub-flows are not embedded into the same nodes and links is randomly selected. Next, two sub-flows in $\mathcal{I}^s$ are randomly selected. The ratio of the traffic split between these two sub-flows is then adjusted such that the sum of the traffic of these two sub-flows remains the same. For example, let us assume that NS s was selected, and the maximum number of sub-flows allowed is 3. To deal with the continuous values of splitting ratio variables, we also assume that traffic rate t^s is divided into m discrete units. Now, let the traffic units handled by sub-flows i_1, i_2 and i_3 be m_1, m_2 and m_3, respectively, where $m_1 + m_2 + m_3 = m$. We first randomly select two of these three sub-flows. These could be i_1 and i_3. Next, we select a random number between 1 and $m_1 + m_3$. Let us assume that the random number is n. Finally, we adjust the traffic rates handled by i_1 and i_3 to be $t^s[n/(m_1 + m_3)]$ and $t^s[(m_1 + m_3 - n)/(m_1 + m_3)]$, respectively.

4.5.3 Tabu List

To avoid repeatedly visiting the same solution and to help escape from the local optimum, the Tabu search algorithm declares the move of the accepted candidate solution as "Tabu" and stores it in Tabu list for a certain number of iterations (j_{tabu}). In our proposed TJCE algorithm, we set

j_{tabu} to be 7 because that number provides the best solution quality based on our experiments. Moreover, the algorithm forbids selecting a move if it exists in the Tabu list unless that move leads to a solution that is better than the best-known solution (Φ_{best}) thus far (aspiration criteria).

4.5.4 Neighborhood Evaluation and Selection

In each iteration, our proposed TJCE evaluates a set of candidate solutions and then selects one of them as the new current solution (i.e., to transit to a neighbor solution). Our proposed TJCE algorithm uses the fitness function (ξ) expressed in Eq. (4.18) in evaluating the candidate solutions.

$$\xi = Obj(h) + w_{rel}M(h) + P(h) - R(h) \tag{4.18}$$

In Eq. (4.18), $Obj(h)$ is the objective function as defined in Eq. (4.15), and h is a neighbor solution. $M(h)$ is the sum of the number of physical nodes used by all NSs, as defined in Eq. (4.16). Here, $M(h)$ is considered as an auxiliary function to help guide the search process because minimizing the number of physical nodes can generally lead to better reliability of NSs. $P(h)$ is a constraint penalty function imposed because of constraint violations. In our proposed TJCE algorithm, the violation of Constraints (4.6), (4.9) and (4.10d) is panelized as the search process is allowed to explore infeasible solutions. In addition, the penalty of each neighbor solution is associated with the level of constraint violations. Consequently, $P(h)$ can be computed by

$$P(h) = \sum_{c \in \mathcal{C}} w_{cst}^{c} \max(0, u_c - v_c) \tag{4.19}$$

where w_{cst}^{c} is a constraint penalty weight for constraint c. u_c and v_c are the left and right sides of Constraints (4.6), (4.9) and (4.10d), respectively.

Finally, $R(h)$ is a NS satisfaction function defined as how many NS requests meet their requirements. It can be represented as

$$R(h) = w_{sat}R_T \tag{4.20}$$

where w_{sat} is a reward weight for NS satisfaction. R_T is defined as the number of satisfied NSs for a given neighbor solution (h). The rationale for that is to reward the move that satisfies all the requirements of NSs, especially the latency requirement. In addition, regarding weight selection, w_{sat} must be greater than w_{cst}^{c}, and w_{cst}^{c} must be greater than w_{rel}. This ensures that our proposed algorithm will prioritize a feasible solution.

In each iteration, after evaluating the generated neighbor solutions, the solution that has the best fitness value is always selected, if it is better than the current solution, and its move is not

Tabu. If the move is Tabu, but it leads to a solution that is better than the best-known solution, then the solution is still accepted, since the aspiration criteria are met. However, if none of the above criteria are satisfied, our proposed algorithm probabilistically selects a non-Tabu neighbor solution such that the neighbor solution with a better fitness value has a higher probability to be chosen.

4.5.5 Termination Criterion

The algorithm stops when the best-known solution does not improve for a certain number of consecutive iterations (j_{stop}). We define this stop criterion as a function of $|\mathcal{K}|$, $|\mathcal{S}|$, $\max(|\mathcal{I}^1|,..,|\mathcal{I}^{|\mathcal{S}|}|)$ and $\max(|\mathcal{N}^1|,..,|\mathcal{N}^{|\mathcal{S}|}|)$, as shown in Eq. (4.21). This formulation allows the number to grow with respect to the number of problem inputs. The multiplier 125 was obtained from experiments. It has been adjusted to ensure a trade-off between the execution time and solution quality.

$$j_{stop} = 125\left[|\mathcal{K}| + |\mathcal{S}| + \max\left(|\mathcal{I}^1|,..,|\mathcal{I}^{|\mathcal{S}|}|\right) + \max(|\mathcal{N}^1|,..,|\mathcal{N}^{|\mathcal{S}|}|)\right] \quad (4.21)$$

4.6 Performance Evaluation

In this subsection, we evaluate the performance of the proposed TJCE algorithm. We compare it with the solutions derived by solving the MILP formulation of the SRP using CPLEX. To the best of our knowledge, none of the existing algorithms in the literature consider the same joint VNF composition and embedding problem as ours. Therefore, we also developed a Late Acceptance Hill-Climbing (LAHC) algorithm [147] and extended the existing Heuristic-based Single- Feature Single-Request (HSFHR) algorithm proposed in [38] to solve our problem and compare them with our proposed TJCE algorithm. We denote the extended version of the HSFHR as E-HSFHR. Furthermore, we implemented the TJCE, LAHC and E-HSFHR algorithms in JAVA, while the SRP model was implemented and solved in CPLEX. All the simulations were executed on a server with 2×12-core 2.20 GHz Intel Xeon E5 2650 v4 CPUs and 128GB of memory. As the TJCE and the LAHC involve some randomness inside the algorithms, we ran the simulations five times for each scenario and took the average results.

4.6.1 Simulation Setup

In our simulations, we consider five network topologies in evaluating the performance of the proposed TJCE algorithm. One of them is generated with the following number of nodes and links (nodes, links): 10 and 15, while the rest are taken from SNDlib [148] to emulate NFV-based infrastructures. These topologies include POLSKA (PK) (12 nodes, 18 links), NOBEL GERMANY (NG) (17, 26), JANOS-US (26, 42) and TA2 (65, 108). We assume that all NFV-enabled nodes have 2 Gbps of node capacity and randomly assign the reliability between 0.9 and 0.99. The processing cost per traffic unit of each node is randomly selected between 1 and 9 per Mbps. The processing delay of each node is calculated based on the M/M/1 queuing model [138]. Similarly, all links have 2 Gbps of bandwidth. The cost per traffic unit of each link is selected randomly between 0.1 and 0.2 per Mbps, and the communication delay of all the links is selected from 0.1 and 0.2 ms. Multiple NSs are generated, whose the traffic rates are randomly chosen among 300, 400, 500 and 600 Mbps [3]. Each NS requires 1 to 3 VNFs, assigned randomly. The latency requirement is selected between 2 and 5 ms [3].

4.6.2 Algorithm Performance Evaluation

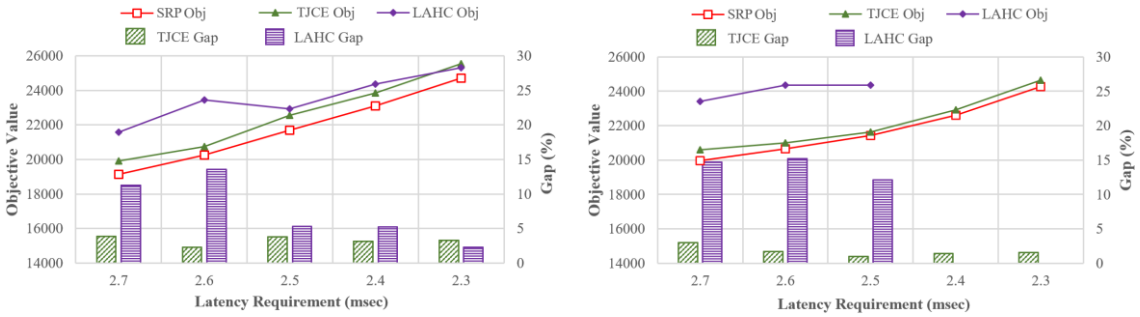
In this subsection, we evaluate the performance of our proposed TJCE. We first consider a case where all physical nodes have the same reliability. In this case, the SRP formulation is solved to provide a benchmark for our proposed TJCE. Then, we evaluate the performance of our proposed TJCE in a case where all physical nodes have different reliability.

4.6.2.1 Equal node reliability

Here, we assume that all physical nodes have the same reliability, which is equal to 0.9. To thoroughly investigate the performance of our proposed TJCE algorithm, we start by considering the case where both cost (w_{cost}) and reliability weights (w_{rel}) are equal to 1. In this case, the cost is prioritized over the reliability, as the cost is the dominant part in the objective function. Figure 4.2 illustrates the objective function value of the solutions obtained by the TJCE and LAHC algorithms, compared to the solutions obtained by solving the SRP for two network topologies, and when $|\mathcal{J}| = 2$. The results are derived with respect to varying latency requirements. The results demonstrate that the TJCE provides very high-quality solutions and outperforms the LAHC in most of the latency requirement cases. Precisely, the average gap of the TJCE remains smaller than

5% for all cases, while that of the LAHC could be as high as 16% in comparison to the SRP solutions. We also observe that the LAHC could not find feasible solutions when the latency requirement was very strict (for example, δ^s = 2.3 ms in Figure 4.2b). These results show the superiority of the TJCE over the LAHC.

Next, we focus on our intended case, where w_{rel} must be much larger than w_{cost} to ensure that the reliability is prioritized over the cost. This means that traffic will be split and multiple instances of each VNF type will be deployed only if it is necessary to satisfy the latency requirement in order to minimize the reliability degradation. In this case, w_{cost} and w_{rel} are set to be 1 and 10^5, respectively. The objective values of all the algorithms under our consideration with different values of $|\mathcal{I}|$ are plotted in Figure 4.3 for two network topologies with varying latency requirements. It is worth noting that the results of the E-HSFHR algorithm are plotted only for $|\mathcal{I}|$ = 2, as the E-HSFHR algorithm does not utilize $|\mathcal{I}|$ in its logic. The results confirm that the TJCE leads to high-quality solutions and attains a very small average gap, compared to the SRP solutions.

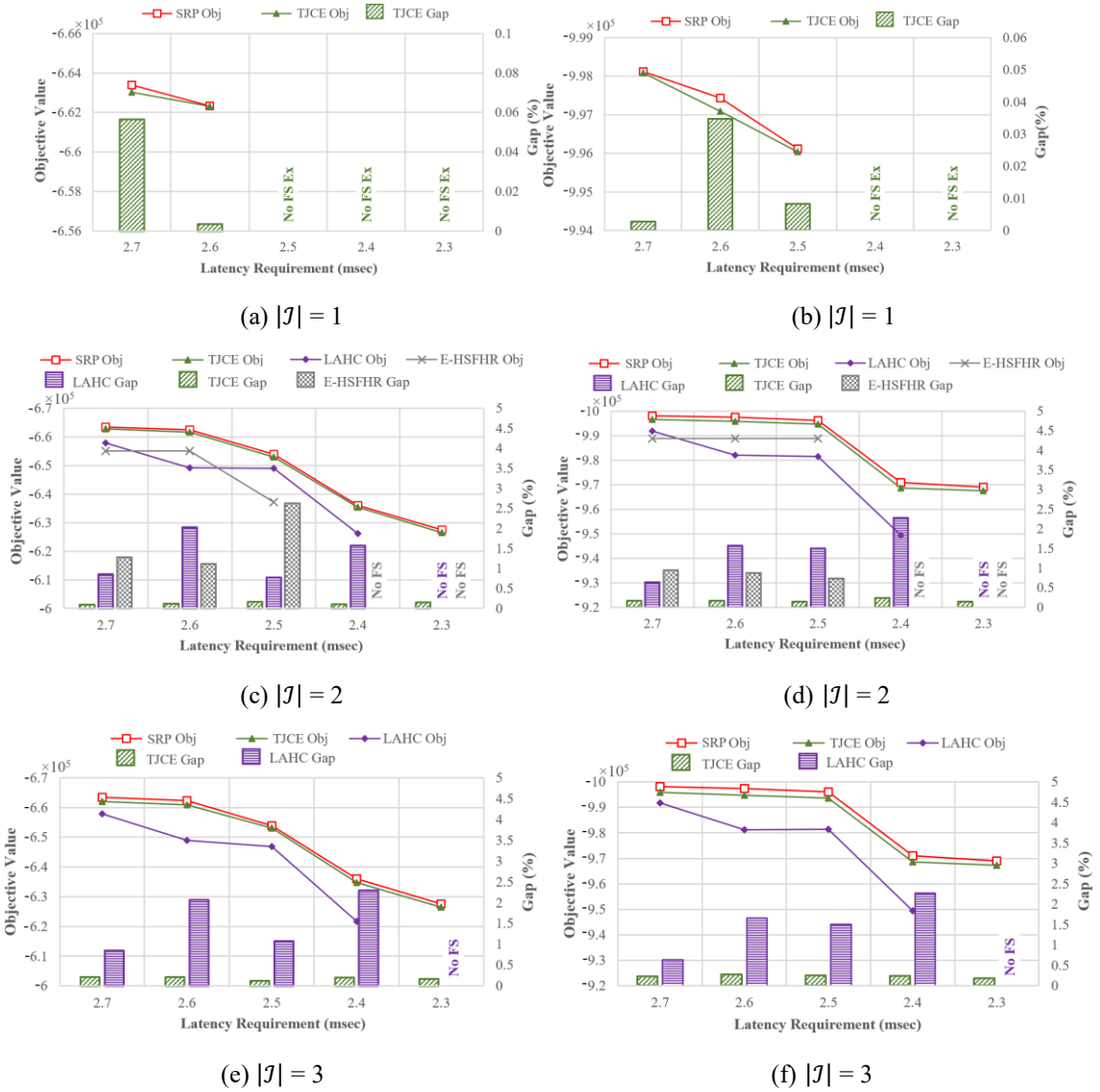


(a) (b)

Figure 4.2: The objective value with w_{cost} = 1 and w_{rel} = 1 for the TJCE and LAHC algorithms together with the gap, compared with the solutions obtained by solving the SRP model in the case, where physical nodes have the same. "No FS" means no feasible solution found. (a) $|\mathcal{I}|$ = 2, $|\mathcal{K}|$ = 10, and $|\mathcal{S}|$ = 6, and (b) $|\mathcal{I}|$ = 2, $|\mathcal{K}|$ = 12, and $|\mathcal{S}|$ = 8.

The results in Figure 4.3 also indicate that there are no feasible solutions in the search space for $|\mathcal{I}|$ = 1 when the latency requirement is very stringent. For instance, in Figure 4.3b, the feasible solutions exist for $|\mathcal{I}|$ = 1 only when the latency requirements are 2.7, 2.6 and 2.5 ms. The rationale is that traffic splitting is not considered, and only one instance of each VNF type is deployed when $|\mathcal{I}|$ = 1, which means that traffic cannot be split to accelerate the processing delay to satisfy the stringent latency requirements for all of the NSs. In other words, with $|\mathcal{I}|$ = 1, both the mathematical model and our TJCE algorithm lead to results similar to those obtained from VNF

embedding algorithms that consider non-traffic splitting and only one instance of each VNF type to provision a given NS. In contrast, when $|\mathcal{J}| = 2$ and $|\mathcal{J}| = 3$, traffic is allowed to be flexibly split, and multiple instances of each VNF type can be deployed to ensure the stringent latency requirements. This indicates the necessity of using our joint VNF composition and embedding approach in provisioning ultra-low latency NSs. We also observe that the E-HSFHR has the worst performance in finding feasible solutions when there are the strict latency requirements. This is because it cannot split traffic flexibly enough to meet the stringent latency requirement for all NSs, as shown in Figures 4.3c and 4.3d.



(a) $|\mathcal{J}| = 1$ (b) $|\mathcal{J}| = 1$

(c) $|\mathcal{J}| = 2$ (d) $|\mathcal{J}| = 2$

(e) $|\mathcal{J}| = 3$ (f) $|\mathcal{J}| = 3$

Figure 4.3: The objective values with $w_{cost} = 1$ and $w_{rel} = 10^5$ for all the algorithms together with the gap, compared with the solutions obtained by solving the SRP model for two network topologies in the case, where physical nodes have the same reliability. "No FS" means no feasible solution found. "No FS Ex" means no feasible solution exists in the search space. (a), (c) and (d) $|\mathcal{K}| = 12$, and $|\mathcal{S}| = 8$, and (b), (d) and (f) $|\mathcal{K}| = 17$, and $|\mathcal{S}| = 12$.

Table 4.2 shows the average execution time of the TJCE and that of solving the SRP using CPLEX. We observe that the execution times of the CPLEX not only depend on the size of the network instances and NSs, but also on the values of w_{rel} and δ^s. The execution time is much longer when w_{rel} and δ^s are smaller, given that w_{cost} is fixed to 1. In contrast, the execution times of the TJCE are approximately the same for different values of w_{rel} and δ^s. As can be seen in Table 4.2, the TJCE significantly outperforms CPLEX in all scenarios and, does so by many orders of magnitude.

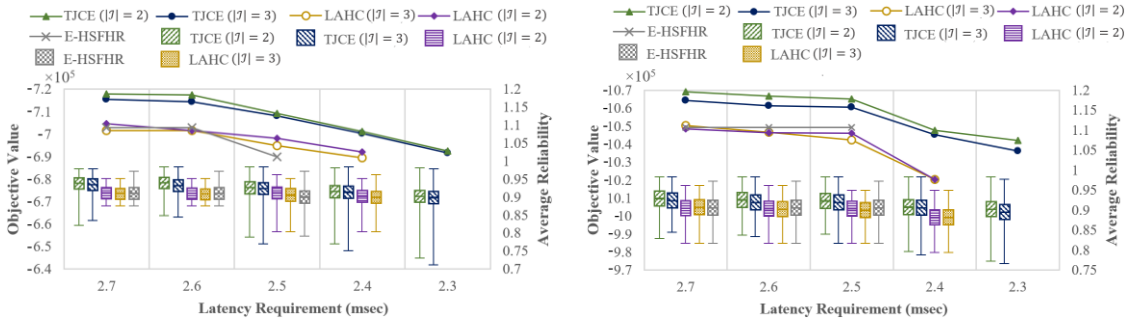
Table 4.2: Average execution time.

Experiment Parameters			**Execution Time (second)**		
Nodes	**NSs**	w_{rel}	**SRP (CPLEX)**		**TJCE**
			Non-stringent δ^s	**Stringent δ^s**	
7	4	1	97	240	2
7	4	10000	7	74	
10	6	1	> a day	> a day	3.6
10	6	10000	180	17340	
12	8	1	> a day	> a day	6.8
12	8	10000	147	35100	
17	12	10000	1740	> a day	15.4
26	24	10000	> a day	> a day	88.88
65	45	10000	> a day	> a day	731.8

4.6.2.1 Different node reliability

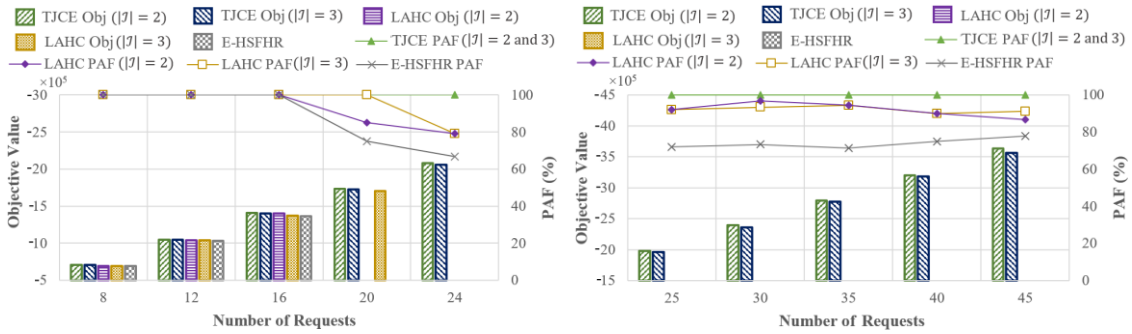
Here, we focus on a case, where physical nodes have different reliability. In this case, we only compare our proposed TJCE algorithm with the LAHC and E-HSFHR algorithms, as the GP model cannot be solved by CPLEX. Considering the same network topologies and NS requests as in Figure 4.3, Figure 4.4 illustrates the objective values and the average reliability of all the algorithms when the latency requirement is varied. The results indicate that the TJCE can provide the lowest objective values and achieve the highest average reliability for all of the given latency requirements. This implies that the TJCE can allow traffic of the NSs to be split and be processed by multiple instances of each VNF type efficiently to meet their latency requirements while maintaining the lowest reliability degradation. Similar to the case, where all physical nodes have the same reliability, the LAHC and E-HSFHR could not find feasible solutions when the latency requirement became more stringent, with the E-HSFHR showing the worst performance.

Figure 4.5 shows the objective values and the percentages of admitted NS requests (PAF) for two network topologies with respect to a varying number of NS requests. As can be seen in Figure 4.5, the TJCE provides slightly better performance compared to the LAHC and the E-HSFHR in scenarios with a small amount of NSs requests. However, as the number of NS requests increases, the PAF of both LAHC and E-HSFHR starts to drop, while the TJCE is still able to admit all NS requests and satisfy their latency requirements. For instance, in Figure 4.5b, when the number of NS requests is greater than 25, the LAHC and the E-HSFHR could not find a feasible solution that satisfies the latency requirements of all of the NS requests. In contrast, the TJCE could still meet the requirements of all the NS requests in our evaluation scenarios. In other words, the TJCE can admit significantly more NS requests into networks than the LAHC and the E-HSFHR.



(a) (b)

Figure 4.4: The objective values and the average reliability with $w_{cost} = 1$ and $w_{rel} = 10^5$ for all the algorithms with varying the latency requirements in the case, where physical nodes have different reliability. (a) $|\mathcal{K}| = 12$, and $|\mathcal{S}| = 8$ and (b) $|\mathcal{K}| = 17$, and $|\mathcal{S}| = 12$.



(a) (b)

Figure 4.5: The objective values and the percentages of admitted NS requests (PAF) for all the algorithms with physical nodes of different reliability, and $w_{cost} = 1$ and $w_{rel} = 10^5$ when the number of NS requests is varied. (a) $|\mathcal{K}| = 26$ and (b) $|\mathcal{K}| = 65$.

4.7 Conclusion

In this chapter, we have proposed an efficient joint VNF composition and embedding algorithm for ultra-low latency NSs. It aims at determining the number of instances of each VNF type required by a NS and their embedding locations as well as how the traffic is split to be processed by these VNF instances in order to meet the stringent latency requirement while optimizing the reliability degradation and cost. We have conducted simulations and compared our proposed algorithm with the existing approaches and the optimization-based approach. The results indicate that our proposed algorithm achieves a high-quality solution in the case, where all physical nodes have the same reliability. It also outperforms the existing approaches in terms of both solution quality and the percentage of admitted NS requests for both cases, i.e., where the physical nodes have the same and different reliability.

Chapter 5

5. [3]Real-Time VNF Embedding for Ultra-Low Latency Services

5.1 Introduction

Service deployment time is also another critical metrics for some ultra-low latency NSs (e.g., wildfire suppression), as these NSs require to be provisioned immediately in order to perform their tasks efficiently once they arrive in the network. In other words, a VNF-FG of a NS must be embedded into the substrate network in a timely manner while satisfying the SLA requirements and optimizing NO's objective, given that the VNF-FG from the VNF composition stage is given. To ensure fast NS provisioning, several works have aimed to design heuristics to solve the VNF embedding problem to find a satisfactory solution within an acceptable execution time, e.g., [31]. Typically, the main idea of this approach is to embed each VNF of a given VNF-FG into the substrate network in a sequential order in such a way that it leads to satisfactory results while considering embedding decisions of only the predecessor VNFs, e.g., [40][41]. However, making local decisions in a sequential manner may lead to poor embedding outcomes, as the embedding decision of a given VNF could be influenced by embedding decisions of other VNFs (i.e., both

[3]This chapter is based on a submitted paper [19]: N. Promwongsa *et al.*, "Look-ahead VNF-FG Embedding Framework for Latency-sensitive Network Services," *Submitted To IEEE Trans. Netw. Serv. Manag.* (under revision).

predecessor and successor VNFs). In other words, the optimal embedding decision of a given VNF cannot be known until the embeddings of all other VNFs in a VNF-FG are determined. This problem is referred to as the causality issue, which may hinder the heuristic approaches from achieving high-quality solutions. Therefore, there is a need for an efficient mechanism to further improve the quality of embedding solutions by enabling progressive embedding and re-embedding of VNFs of a VNF-FG while considering the contextual information of the neighboring VNFs.

In this chapter, we propose an efficient VNF embedding algorithm called h-horizon sequential lookahead greedy algorithm (h-HSLGA) to tackle the aforementioned causality issue. The proposed h-HSLGA relies on a window-based approach along with go-back and move-forward mechanisms to offer efficient embedding and re-embedding strategies to alleviate the impact of the causality issue. Its main objective is to minimize the total cost while satisfying the latency requirements. It also offers a general VNF embedding framework, which is compatible with any sequential embedding algorithm. Moreover, our proposed h-HSLGA is a heuristic-based algorithm, so that it can ensure timely service provisioning. To evaluate our proposed algorithm, we conduct extensive simulations. We also compare the performance of our proposed algorithm with those of the optimization-based approaches and the existing VNF embedding algorithm.

The rest of this chapter is organized as follows. We first provide illustrative examples of the causality issue and present the system model. Next, we develop an online VNF embedding formulation and propose a Bender's decomposition-based approach to solve the developed formulation to serve as benchmarks. Then, we describe our proposed look-ahead embedding framework. After that, we present our evaluation results and discussions. Finally, we conclude this chapter.

5.2 Illustrative Examples of the Causality Issue

To better understand the causality issue, in this subsection, we demonstrate how it can impact the quality of VNF embedding solutions. First, let us consider a substrate network, comprising 7 physical nodes and 7 physical links, as shown in Figure 5.1. Each node is associated with node resource capacity, cost per resource unit, and a set of supported VNF types. Similarly, each link has a bandwidth capacity, cost per bandwidth unit, and link latency (see Figure 5.1). Furthermore, each VNF type is associated with a predefined resource requirement and a fixed instantiation cost. Once a VNF instance is instantiated, it is attributed with a processing capacity and a fixed reusable

cost when it is reused by another NS. For simplicity and without loss of generality, we assume that the resource requirement and instantiation cost of all VNF types are 1 resource unit and $1, respectively. In addition, we assume that VNF instances of all types have 2 units of processing capacity and $0 of reusable cost. Next, let us assume that NS 1 needs to be deployed in the network. NS 1 requires its traffic to be transmitted from Node 1 and then processed by a VNF-FG (which is given in the form of an ordered set of VNFs) before reaching Node 7 with some specific requirements, which are all shown in Figure 5.1. In addition, we assume that there are some VNF instances that have already been deployed in the network to serve other NSs, and all of them have enough capacities to serve NS 1.

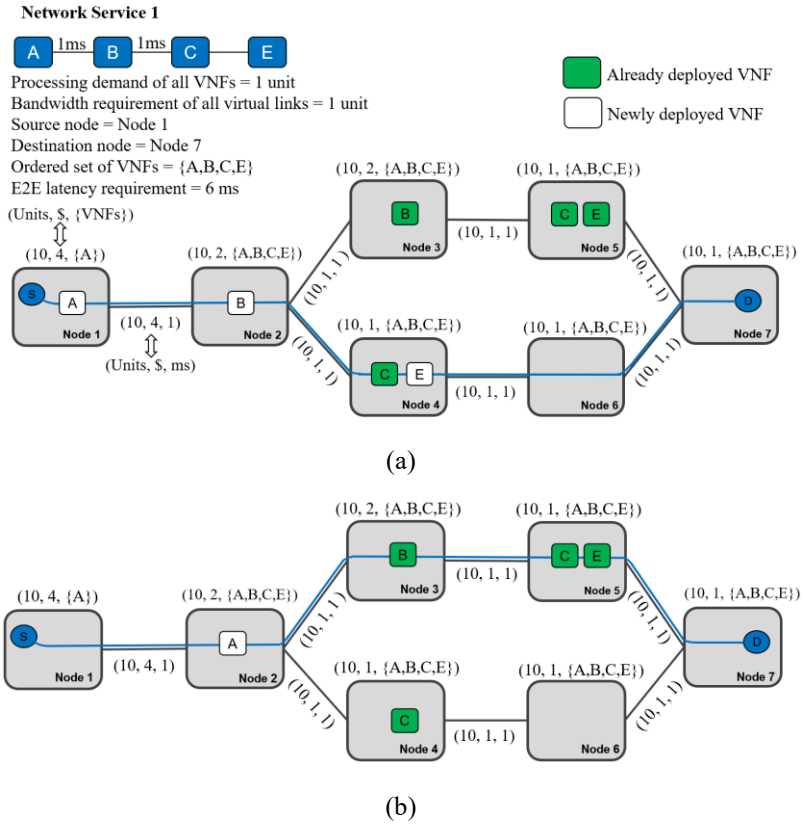


Figure 5.1: Illustrations of the causality issue: (a) embedding solution obtained by SGA [41] and (b) optimal embedding solution for NS 1.

Now, let us focus on the sequential greedy algorithm (SGA) introduced in [41] to demonstrate the limitations of the existing sequential algorithms. The key idea of SGA is to make local decisions to embed each VNF of a given VNF-FG into a physical node in a sequential order while satisfying the given constraints. More specifically, SGA starts by setting the source node (i.e., Node 1) as an anchor point. It then searches for the best hosting node and path (between the anchor

point and the hosting node) in terms of total cost to embed the first VNF of the given VNF-FG (i.e., VNF A). In this scenario, SGA embeds VNF A into Node 1 because this leads to the smallest total embedding cost, which includes resource cost, instantiation cost, and transmission cost (see Figure 5.1a). It should be noted that although the cost of Node 2 is smaller than that of Node 1, embedding VNF A into Node 1 leads to a smaller total cost due to the transmission cost between Nodes 1 and 2. Next, SGA sets the physical node hosting VNF A as a new anchor point (i.e., Node 1) and moves to embed the next VNF (i.e., VNF B). Repeating the same procedure, the best hosting node and routing path are selected for embedding VNF B and the remaining VNFs until all the VNFs are embedded into the network. Accordingly, VNF B is embedded into Node 2 due to the latency constraint between VNFs A and B. After that, VNF C is embedded into Node 4 by reusing the already deployed VNF C. Finally, VNF E is embedded into Node 4. Consequently, the total embedding cost for the above explained solution is $17.

As demonstrated in this particular example, embedding VNF A without knowing the embedding location of VNF B causes a poor embedding decision for VNF A. More importantly, such poor embedding decision for VNF A also leads to a propagation of poor embedding decisions for VNFs B, C and E. To see this, we note that if the embedding location of VNF B was known, VNF A could have been embedded into Node 2 instead of Node 1, resulting in a cost reduction of $2. Accordingly, if VNF A is embedded into Node 2, VNF B could be embedded into Node 3 by reusing the already deployed VNF B therein. Furthermore, this new embedding decision for VNF B will eventually allow VNFs C and E to be embedded into Node 5 by reusing the already deployed VNFs C and E therein (see Figure 5.1b). As a result, the total cost would be $10, which translates into a 41% cost reduction in comparison to the solution obtained by SGA.

The discussion above demonstrates that embedding a given VNF without knowing the embedding locations of other VNFs in a VNF-FG can potentially lead to a poor embedding solution. This problem is referred to as the causality issue. To address this issue, efficient mechanisms are needed to enable the already embedded VNFs to be re-visited and re-embedded once new contextual information about the embedding locations of other VNFs arise. Referring again to the example explained above, embedding VNF B into Node 2 creates a new context, which can then be used to revisit VNF A and re-embed it into Node 2. In this chapter, we aim to overcome the causality issue by allowing the VNFs to be revisited and re-embedded, when needed.

5.3 System Model

In our VNF embedding problem, we represent a substrate network as a graph ($\mathcal{K}$,$\mathcal{L}$) where $\mathcal{K}$ is the set of NFV-enabled nodes, and $\mathcal{L}$ is the set of physical links. Each NFV-enabled node is associated with an available node capacity r_k (e.g., CPUs, memories, storages), cost per resource unit c_k, and a set of supported VNF types $\mathcal{F}_k \subseteq \mathcal{F}$, where $\mathcal{F}$ is a set of all VNF types that are supported in the network. Each physical link $l = (k_1,k_2) \in \mathcal{L}$ has an available bandwidth b_l, cost per bandwidth unit c_l, and link delay d_l, where k_1 and $k_2 \in \mathcal{K}$. Each VNF type $f \in \mathcal{F}$ is associated with a predefined resource requirement u_f (measured by the number of CPUs, memory, and/or storage) for instantiation and a processing capacity p_f (measured by traffic per time unit). Each VNF type $f \in \mathcal{F}$ also has a fixed instantiation cost (e.g., license cost) which can generally depend on its hosting physical node. Thus, we define $\sigma_{f,k}$ to indicate the instantiation cost of VNF f on physical node k. Furthermore, we define $\mathcal{I}_f = \{1,2,\ldots,|\mathcal{I}_f|\}$ to specify a set of VNF instances corresponding to VNF type $f \in \mathcal{F}$, where $|\mathcal{I}_f|$ is the maximum number of instances of VNF type f, allowed to be instantiated across the network. Each instance i of VNF type f has a fixed reusable cost $\theta_{i,f}$ when it is reused by a NS. Note that generally the reusable cost $\theta_{i,f}$ is relatively small in comparison to the instantiation cost for instantiating a new instance. In addition, we define $\emptyset_{i,f}$ to specify the amount of processing load that has already been assigned to instance i of VNF type f.

Now, let us focus on an online scenario, where only one NS arrives in the network at any given point in time. The NS requires its traffic to be traversed through a set of VNFs connected in the form of a VNF-FG. In this problem, we consider three basic VNF-FG topologies [149], including linear/chain topology, split topology with multiple destinations, and split and-merged topology with a single destination, as illustrated in Figure. 5.2. We note that more complex VNF-FG topologies can be realized by combining these three basic VNF-FG topologies with one another [150]. We denote $\mathcal{N} = \{1,\ldots,|\mathcal{N}|\}$ as the set of VNF indices of the NS in its VNF-FG, where $|\mathcal{N}|$ is the number of required VNFs. We define γ^n to indicate a VNF type associated to VNF index $n \in \mathcal{N}$. For instance, in Figure 5.2b, $|\mathcal{N}| = 3$, so $\mathcal{N} = \{1, 2, 3\}$, $\gamma^1 = \text{A}$, $\gamma^2 = \text{B}$ and $\gamma^3 = \text{C}$. Note that the numbers inside parentheses in Figure 5.2 indicate VNF indices in the VNF-FG. Each VNF index n is also associated with processing demand t^n. In addition, the NS can have one source along with one or multiple destinations.

Next, we define $\mathcal{E}$ to specify the set of all virtual links in the NS's VNF-FG. For each $e = (n_1, n_2) \in \mathcal{E}$, n_1 and n_2 represents VNF indices of the virtual link e, where n_1 and $n_2 \in \mathcal{N} \cup \{0, |\mathcal{N}| + 1, \ldots, |\mathcal{N}| + m\}$. We use 0 to indicate a source index while using $|\mathcal{N}| + 1$ to $|\mathcal{N}| + m$ to specify destination indices, where m is the number of destinations in the VNF-FG. For example, for the VNF-FG shown in Figure 5.2b, $\mathcal{E} = \{(0,1), (1,2), (1,3), (2,4), (3.5)\}$. For each virtual link $e \in \mathcal{E}$, we define α^e and β^e to represent the origin VNF index and destination VNF index of the virtual link e, respectively. For instance, for $e = (1,2)$, $\alpha^e = 1$ and $\beta^e = 2$. Each virtual link e also has a specific bandwidth requirement o^e. Furthermore, the NS can have one or multiple service chains in its VNF-FG, each having a specific latency requirement. We denote $\mathcal{G}$ to indicate the set of indices of those service chains. We also define $\mathcal{E}^g$ to specify the set of virtual links, where $\mathcal{E}^g \subseteq \mathcal{E}$, and define δ^g to indicate the latency requirement of service chain g in the VNF-FG. For example, let us again consider the VNF-FG shown in Figure 5.2b. Now, let us also assume that there are two latency requirements between S and D1, and between S and D2 that need to be satisfied. Therefore, $\mathcal{G} = \{1,2\}$, $\mathcal{E}^1 = \{(0,1), (1,2), (2,4)\}$ and $\mathcal{E}^2 = \{(0,1), (1,3), (3,5)\}$.

Finally, we define five binary parameters that are used in our problem formulation. The first binary parameter is $\tilde{z}_k^{i,f} \in \{0, 1\}$ that indicates whether instance $i \in \mathcal{I}_f$ of VNF type $f \in \mathcal{F}$ has already been deployed in physical node $k \in \mathcal{K}$ (1) or not (0). The second one is w_k^f that indicates whether physical node k can support VNF type $f \in \mathcal{F}$ (1) or not (0). The third one is τ_f^n which specifies whether the f type of VNF index n of a NS is VNF type $f \in \mathcal{F}$ (1) or not (0). The fourth one is $\mu_k \in \{0,1\}$ that indicates whether physical node k is a source of a NS (1) or not (0). The last one is $\pi_k^n \in \{0,1\}$ that specifies whether physical node k is a destination of a NS associated with index n (1) or not (0), where $n \in \{|\mathcal{N}| + 1, \ldots, |\mathcal{N}| + m\}$.

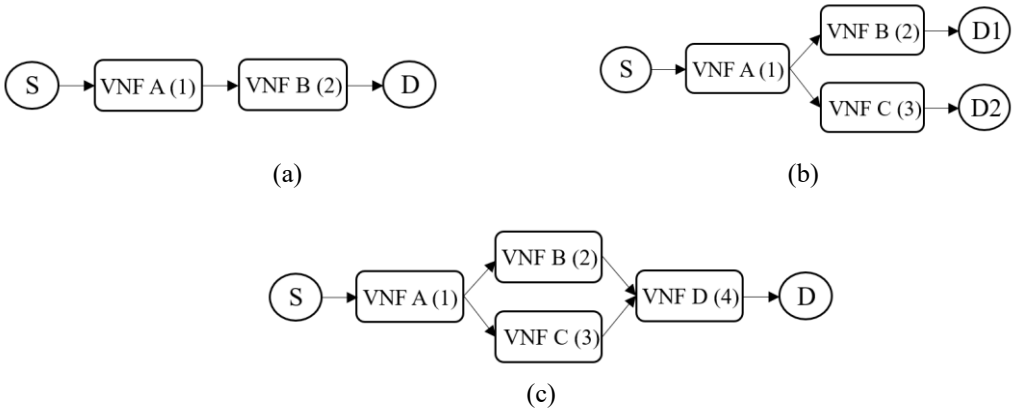


Figure 5.2: Three basic VNF-FG topologies [150].

Table 5.1 Summary of key notations for the online VNF embedding problem.

Input Parameters	
$\mathcal{K}$	Set of NFV-enabled nodes
$\mathcal{L}$	Set of physical links
r_k	Available node capacity of physical node k
c_k	Node cost per resource unit of physical node k
$\mathcal{F}_k$	Set of supported VNF types of physical node k
b_l	Available link bandwidth of physical link l
c_l	Link cost per bandwidth unit of physical link l
d_l	Link delay of physical link l
u_f	Resource requirement for instantiation of VNF type f
$\sigma_{f,k}$	Instantiation cost of VNF type f on physical node k
p_f	Processing capacity of VNF type f
$\mathcal{I}_f$	Set of instances associated with VNF type f
$\theta_{i,f}$	Reusable cost of instance i of VNF type f
$\emptyset_{i,f}$	Processing loads already assigned to instance i of VNF type f
$\mathcal{N}$	Set of VNF indices in a VNF-FG of a NS
γ^n	VNF type corresponding to VNF index n in a VNF-FG of a NS
t^n	Processing demand of VNF index n in a VNF-FG of a NS
m	Number of destinations in a VNF-FG of a NS
$\mathcal{E}$	Set of virtual links in a VNF-FG of a NS
o^e	Bandwidth requirement of virtual link e of a NS
α^e	Origin VNF index of virtual link e of a NS
β^e	destination VNF index of virtual link e of a NS
$\mathcal{G}$	Set of service chain indices in a VNF-FG of a NS that have specific latency requirements
$\mathcal{E}^g$	Set of virtual links of service chain g in a VNF-FG of a NS
δ^g	Latency requirement of service chain g in a VNF-FG of a NS
$\bar{z}_k^{i,f}$	Binary parameter, indicating whether instance $i \in \mathcal{I}_f$ of VNF type f has been already deployed in physical node $k \in \mathcal{K}$
w_k^f	Binary parameter, indicating whether physical node k can support VNF type f
τ_f^n	Binary parameter, indicating whether a type of VNF index n is VNF type f
μ_k	Binary parameter, indicating whether physical node k is a source in a VNF-FG of a NS
π_k^n	Binary parameter, indicating whether physical node k is a destination in a VNF-FG of a NS associated with index n, where $n \in \{\lvert\mathcal{N}\rvert + 1, \ldots, \lvert\mathcal{N}\rvert + m\}$

5.4 Problem Formulation

Five decision variables are defined to formulate our online VNF embedding problem. They are described as follows:

- $z_k^{i,f} \in \{0,1\}$ is a binary variable, specifying whether instance $i \in \mathcal{I}_f$ of VNF type $f \in \mathcal{F}$ is deployed in physical node $k \in \mathcal{K}$ (1) or not (0).
- $\partial_k^{i,f} \in \{0,1\}$ is a binary variable, specifying whether instance $i \in \mathcal{I}_f$ of VNF type $f \in \mathcal{F}$ on physical node $k \in \mathcal{K}$ is a newly deployed instance (1) or an already deployed instance/not deployed (0) in the network.

- $x_{i,f,k}^{n} \in \{0,1\}$ is a binary variable, specifying whether VNF index $n \in \mathcal{N}$ of a NS is assigned to instance $i \in \mathcal{I}_f$ of VNF type $f \in \mathcal{F}$ hosted on physical node $k \in \mathcal{K}$ (1) or not (0).
- $q_k^n \in \{0,1\}$ is a binary variable, indicating whether an instance assigned to VNF index $n \in \mathcal{N}$ of a NS is hosted on physical node $k \in \mathcal{K}$ (1) or not (0).
- $y_l^e \in \{0,1\}$ is a binary variable, indicating whether virtual link $e \in \mathcal{E}$ of a NS is mapped into physical link $l \in \mathcal{L}$ (1) or not (0).

Next, we describe the constraints of the problem. Constraint (5.1) ensures that any VNF index n of a NS must be assigned one of already deployed or newly deployed instances of its corresponding VNF type f.

$$\sum_{f \in \mathcal{F}} \sum_{i \in \mathcal{I}_f} \sum_{k \in \mathcal{K}} \tau_f^n x_{i,f,k}^n = 1, \forall n \in \mathcal{N} \tag{5.1}$$

Constraint (5.2) ensures that if VNF index n of a NS is assigned instance i of VNF type f hosted in physical node k, that instance must be deployed in that physical node.

$$x_{i,f,k}^n \leq z_k^{i,f}, \forall n \in \mathcal{N}, \forall f \in \mathcal{F}, \forall i \in \mathcal{I}_f, \forall k \in \mathcal{K} \tag{5.2}$$

Constraint (5.3) ensures that instance i of VNF type f can be deployed in physical node k only if the physical node can support VNF type f.

$$z_k^{i,f} \leq w_k^f, \forall f \in \mathcal{F}, \forall i \in \mathcal{I}_f, \forall k \in \mathcal{K} \tag{5.3}$$

Constraint (5.4) ensures that any instance of any VNF type must be deployed only once in the network.

$$\sum_{k \in \mathcal{K}} z_k^{i,f} \leq 1, \forall f \in \mathcal{F}, \forall i \in \mathcal{I}_f \tag{5.4}$$

Constraint (5.5) ensures that if instance i of VNF type f was already deployed in physical node k, the instance cannot be deployed in any other physical nodes.

$$\tilde{z}_k^{i,f} \leq z_k^{i,f}, \forall f \in \mathcal{F}, \forall i \in \mathcal{I}_f, \forall k \in \mathcal{K} \tag{5.5}$$

Constraint (5.6) helps indicate that if instance i of VNF type f is newly deployed in physical node k, $\partial_k^{i,f}$ is equal to 1; otherwise, it is equal to 0.

$$\partial_k^{i,f} = z_k^{i,f} - \tilde{z}_k^{i,f}, \forall f \in \mathcal{F}, \forall i \in \mathcal{I}_f, \forall k \in \mathcal{K} \tag{5.6}$$

Constraint (5.7) ensures that the processing load assigned to instance i of VNF type f must not exceed its available processing capacity.

$$\sum_{n \in \mathcal{N}} \sum_{k \in \mathcal{K}} t^n x_{i,f,k}^n \leq p_f - \emptyset_{i,f}, \forall f \in \mathcal{F}, \forall i \in \mathcal{I}_f \tag{5.7}$$

Constraint (5.8) ensures that any node $k \in \mathcal{K}$ must have enough capacity if VNF instances are to be newly deployed in that node.

$$\sum_{f \in \mathcal{F}} \sum_{i \in \mathcal{I}_f} u_f Z_k^{i,f} \leq r_k \,, \forall k \in \mathcal{K} \tag{5.8}$$

Constraint (5.9) helps indicate a physical node on which an instance assigned to VNF index n of a NS is hosted.

$$q_k^n = \sum_{f \in \mathcal{F}} \sum_{i \in \mathcal{I}_f} x_{i,f,k}^n \,, \forall n \in \mathcal{N}, \forall k \in \mathcal{K} \tag{5.9}$$

Constraint (5.10) ensures that each virtual link e should be assigned physical links that connect between physical nodes hosting instances assigned to the origin VNF index and the destination VNF index of virtual link e.

$$\sum_{l=(k_1,k_2) \in \mathcal{L} | k_1 = k} y_l^e - \sum_{l=(k_1,k_2) \in \mathcal{L} | k_2 = k} y_l^e = \begin{cases} \mu_k - q_k^{\beta^e} & if\ \alpha^e = 0 \\ q_k^{\alpha^e} - \pi_k^{\beta^e} & if\ \beta^e > |\mathcal{N}| \\ q_k^{\alpha^e} - q_k^{\beta^e} & if\ \alpha^e \neq 0 \\ & and\ \beta^e \leq |\mathcal{N}| \end{cases}, \quad \forall e \in \mathcal{E}, \forall k \in \mathcal{K} \tag{5.10}$$

Constraint (5.11) ensures that the bandwidth demand assigned to any given physical must not exceed its bandwidth capacity.

$$\sum_{e \in \mathcal{E}} o^e y_l^e \leq b_l, \forall l \in \mathcal{L} \tag{5.11}$$

Constraint (5.12) ensures that the latency requirements of all the given service chains are satisfied.

$$\sum_{e \in \mathcal{E}^g} \sum_{l \in \mathcal{L}} d_l y_l^e \leq \delta^g \,, \forall g \in \mathcal{G} \tag{5.12}$$

With all these considerations, our objective is to minimize the total embedding cost. To this end, we formulate the VNF embedding problem as an ILP problem, as follows:

$$min \sum_{n \in \mathcal{N}} \sum_{f \in \mathcal{F}} \sum_{i \in \mathcal{I}_f} \sum_{k \in \mathcal{K}} \theta_{i,f} \tilde{z}_k^{i,f} x_{i,f,k}^n + \sum_{f \in \mathcal{F}} \sum_{i \in \mathcal{I}_f} \sum_{k \in \mathcal{K}} (u_f c_k + \sigma_{f,k}) \partial_k^{i,f} + \sum_{e \in \mathcal{E}} \sum_{l \in \mathcal{L}} o^e c_l y_l^e \tag{5.13}$$

Subject to Constraints (5.1) – (5.12)

In Eq. (5.13), the first term of the objective function specifies the cost for reusing the already deployed instances. The second term is the cost for instantiating new instances. Finally, the last term indicates the transmission cost.

5.5 Bender's Decomposition-Based Approach

In this subsection, we utilize a Bender's decomposition (BD) method to solve the VNF embedding problem described in Section 5.4. Generally, the BD method [151] divides the original problem (OP) into two simpler problems, (i) master problem (MP) and (ii) subproblem (SP), which are easier to solve. The MP represents a simpler version of the OP while the SP is the subproblem of the OP, where some variables in the OP are fixed. In the BD method, both MP and SP are solved iteratively. At every iteration, a Bender's cut (i.e., a new constraint) is added to the MP to tighten the MP's search space. This cut is generally derived from the SP. This process is iterated until objective values of both MP and SP converge.

5.5.1 Subproblem

To formulate the SP, we fix variable $\boldsymbol{y}$ in the OP to be $\tilde{\boldsymbol{y}}$. This makes the SP (also known as ILP SP) become only the VNF placement and assignment problem, which is easier to solve than the OP that also involves the routing problem. The ILP SP can be expressed as follows:

$$min \sum_{n\in\mathcal{N}} \sum_{f\in\mathcal{F}} \sum_{i\in\mathcal{I}_f} \sum_{k\in\mathcal{K}} \theta_{i,f} \tilde{z}_k^{i,f} x_{i,f,k}^n + \sum_{f\in\mathcal{F}} \sum_{i\in\mathcal{I}_f} \sum_{k\in\mathcal{K}} (u_f c_k + \sigma_{f,k}) \partial_k^{i,f} \tag{5.14}$$

Subject to Constraints (5.1) – (5.9)

$$\sum_{l=(k_1,k_2)\in\mathcal{L}|k_1=k} \tilde{y}_l^e - \sum_{l=(k_1,k_2)\in\mathcal{L}|k_2=k} \tilde{y}_l^e = \begin{cases} \mu_k - q_k^{\beta^e} & if\ \alpha^e = 0 \\ q_k^{\alpha^e} - \pi_k^{\beta^e} & if\ \beta^e > |\mathcal{N}| \\ q_k^{\alpha^e} - q_k^{\beta^e} & if\ \alpha^e \neq 0 \\ & and\ \beta^e \leq |\mathcal{N}| \end{cases}, \quad \forall e \in \mathcal{E}, \forall k \in \mathcal{K} \tag{5.15}$$

Generally, classical Bender's cuts can be derived from dual variables of the SP. However, the standard duality theory does not exist in an ILP, and our SP is an ILP. Therefore, to generate the Bender's cuts, we first relax integrality constraints of the ILP SP. Then, we apply a three-step approach to obtain integer solutions, which will be discussed shortly. By relaxing variables $\boldsymbol{x}$, $\boldsymbol{z}$, $\boldsymbol{Z}$, and $\boldsymbol{q}$ to be non-negative continuous variables, the dual LP SP can be formulated as follows:

$$\max \xi = \sum_{f\in\mathcal{F}}\sum_{i\in\mathcal{I}_f}\sum_{k\in\mathcal{K}}\left[w_k^f C_k^{i,f} + \tilde{z}_k^{i,f}\left(E_k^{i,f} + F_k^{i,f}\right)\right] + \sum_{n\in\mathcal{N}} A^n + \sum_{f\in\mathcal{F}}\sum_{i\in\mathcal{I}_f}[D_k^{i,f} + (p_f - \emptyset_{i,f})G_{i,f}] + \sum_{k\in\mathcal{K}} r_k H_k + \sum_{e\in\mathcal{E}}\sum_{k\in\mathcal{K}}\left[\sum_{\substack{l=(k_1,k_2)\in\mathcal{L}:\\ k_1=k}} \tilde{y}_l^e - \sum_{\substack{l=(k_1,k_2)\in\mathcal{L}:\\ k_2=k}} \tilde{y}_l^e\right] J_k^e - \sum_{e\in\mathcal{E}}\sum_{k\in\mathcal{K}:\alpha^e=0} \mu_k J_k^e + \sum_{e\in\mathcal{E}}\sum_{k\in\mathcal{K}:\beta^e\geq|\mathcal{N}|} \pi_k^{\beta^e} J_k^e \tag{5.16}$$

Subject to

$$\tau_f^n A^n + B_{i,f,k}^n + t^n G_{i,f} + I_k^n \leq \theta_{i,f}\tilde{z}_k^{i,f},\ \forall n\in\mathcal{N}, \forall f\in\mathcal{F}, \forall i\in\mathcal{I}_f, \forall k\in\mathcal{K} \tag{5.17}$$

$$C_k^{i,f} + D_{i,f} + E_k^{i,f} + F_k^{i,f} - \sum_{n\in\mathcal{N}} B_{i,f,k}^n \leq 0, \forall f\in\mathcal{F}, \forall i\in\mathcal{I}_f, \forall k\in\mathcal{K} \tag{5.18}$$

$$u_f H_k - F_k^{i,f} \leq u_f c_k + \sigma_{f,k}, \forall f\in\mathcal{F}, \forall i\in\mathcal{I}_f, \forall k\in\mathcal{K} \tag{5.19}$$

$$\sum_{e\in\mathcal{E}:\alpha^e=n} J_k^e - \sum_{e\in\mathcal{E}:\beta^e=n} J_k^e - I_k^n \leq 0, \forall n\in\mathcal{N}, \forall k\in\mathcal{K} \tag{5.20}$$

$$B_{i,f,k}^n, C_k^{i,f}, D_{i,f}, G_{i,f}, H_k \leq 0, E_k^{i,f} \geq 0$$

where $\boldsymbol{A}$, $\boldsymbol{B}$,..., $\boldsymbol{I}$ are the dual variables associated with Constraints (5.1) to (5.9), respectively, and $\boldsymbol{J}$ is the dual variable corresponding to Constraint (5.15).

5.5.2 Master Problem

With a new variable v, the ILP MP can be formulated as follows:

$$min\, v \tag{5.21}$$

Subject to Constraints (5.11) and (5.12)

$$\xi(\boldsymbol{y}) \leq 0 \tag{5.22}$$

$$v \geq \sum_{e\in\mathcal{E}}\sum_{l\in\mathcal{L}} o^e c_l y_l^e + \xi(\boldsymbol{y}) \tag{5.23}$$

Constraints (5.11) and (5.12) ensure that the bandwidth and latency requirements are satisfied in the ILP MP. Constraint (5.22) is known as a feasibility cut which can be derived by solving the

dual LP SP. The feasible cut aims to ensure that values of variable $\boldsymbol{y}$ lead to the bounded dual LP SP. For each iteration, the feasible cut will be added to the ILP MP if the dual LP SP is unbounded or infeasible. Also, the values of the dual variables $\boldsymbol{A}$, $\boldsymbol{B}$,..., $\boldsymbol{J}$ in $\xi(\boldsymbol{y})$ are extreme rays in a polyhedron defined by Constraints (5.17)-(5.20) in the dual LP SP. Constraint (5.23) is called an optimality cut, which aims at tightening the ILP MP's search space in order to reach the optimality of the OP. Similar to the feasibility cut, the optimality cut can also be derived by solving the dual LP SP. For each iteration, the optimality cut will be added to the ILP MP if the dual LP SP is bounded, and the values of the dual variables in $\xi(\boldsymbol{y})$ are extreme points in the polyhedron formed by the dual LP SP's constraints.

5.5.3 Three-Step Approach

Since we relax the integrality constraints of the ILP SP in order to generate the feasibility and optimality cuts, integer solutions to the OP may not be achieved. To address this issue, we design our three-step BD approach, which is similar to the one proposed in [152]. The flowchart of our proposed three-step BD approach is illustrated in Figure 5.3. In Step 1, all integrality constraints of both ILP SP and ILP MP are relaxed, and then the LP SP and LP MP are solved to the optimality. More specially, the algorithm starts by generating an initial solution, which can be achieved using the existing SGA proposed in [41]. Given $\tilde{\boldsymbol{y}}^{(j)}$ in iteration j, it solves the dual LP SP to generate a Bender cut. If the dual LP SP is feasible, it then adds an optimality cut to the LP MP; otherwise, it adds the feasibility cut to the LP MP. Solving the dual LP SP also provides upper bound (UB) of the LP OP. After that, it solves the LP MP to find lower bound (LB) of the LP OP and a new solution of $\boldsymbol{y}$. The algorithm will then check whether UB and LB are equal. If yes, it goes to the next step; otherwise, it moves to the next iteration, updates $\tilde{\boldsymbol{y}}^{(j)}$, and repeats the same process. In Step 2, the algorithm solves the ILP MP with all the cuts generated in Step 1 to obtain an integer solution of $\boldsymbol{y}$. Then, it checks the feasibility of the integer solution of $\boldsymbol{y}$ by solving the dual LP SP. If the dual LP SP is not feasible, a new feasibility cut is added to the ILP MP, and the same process is repeated; otherwise, the algorithm goes to Step 3. In Step 3, given the feasible and integer solution of $\boldsymbol{y}$, it solves the ILP SP to obtain integer solutions of variables $\boldsymbol{x}$, $\boldsymbol{z}$, $\boldsymbol{Z}$, and $\boldsymbol{q}$.

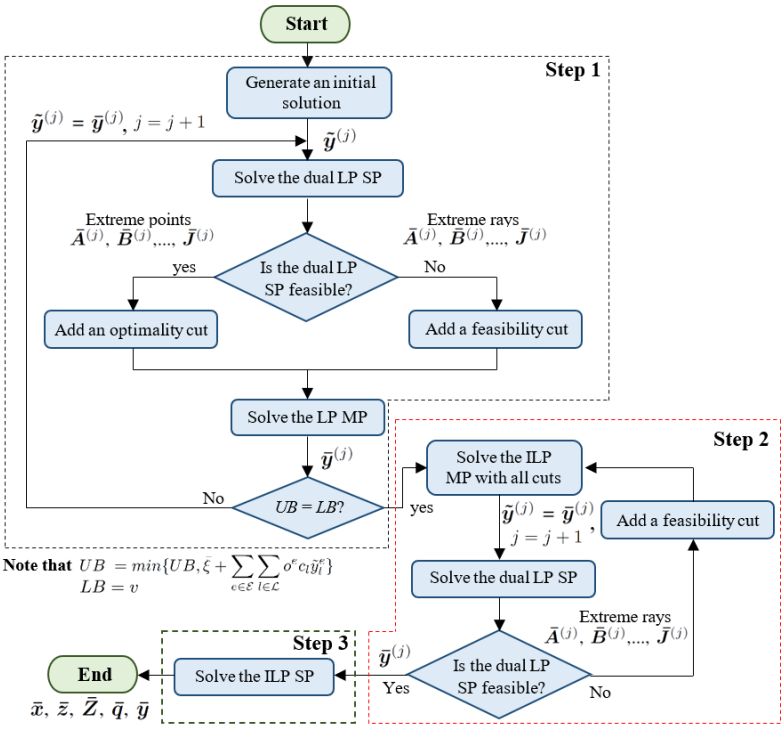


Figure 5.3: Flowchart of our proposed three-step BD approach.

It should be noted that the BD generally guarantees to converge to an optimal solution only if SPs can be formulated as a LP [151]. However, our LP SP does not always return integer solutions. This means that our original ILP SP cannot be transformed equivalently to a LP.

Theorem 1. $P = \{\boldsymbol{x} : \boldsymbol{Ax} \leq \boldsymbol{b}\}$ has integer solutions if coefficient matrix $\boldsymbol{A}$ is totally unimodular and coefficient matrix $\boldsymbol{b}$ is an integer matrix.

Proof. See page 75 of Ref. [153].

Lemma 2. *The coefficient matrix of all the constraints in our LP SP is not totally unimodular. Therefore, our LP SP does not necessarily return integer solutions*.

Proof. To prove this, we need to show that the coefficient matrix of all the constraints in our LP SP is not totally unimodular. By definition, a matrix is totally unimodular if every square submatrix has a determinant of 0, +1, or -1. However, u_f in constraint (8) can be any non-negative number, which does not allow the determinants of all the square submatrix of the coefficient matrix to be 0, +1, or -1. Thus, the coefficient matrix of all the constraints in our LP SP is not totally unimodular.

Thus, our proposed BD does not guarantee to achieve the optimal solution. The performance of our proposed BD will be further discussed in Section 5.7.

5.6 Look-Ahead VNF Embedding Framework

In this subsection, we describe our proposed h-horizon sequential look-ahead greedy algorithm (h-HSLGA) to tackle the causality issue as discussed in Section 5.2. Our proposed h-HSLGA relies on a window-based approach along with go-back and move-forward mechanisms to alleviate the impact of the causality issue while ensuring acceptable service deployment times. It consists of two variants, namely, (i) fixed-size sliding window and (ii) variable-size sliding widow. Next, we first explain the main concept of our proposed h-HSLGA. After that, we discuss our proposed h-HSLGA with fixed- and variable-size sliding windows.

5.6.1 h-HSLGA

The main idea of our proposed h-HSLGA is to allow VNFs in a given VNF-FG to be embedded into a substrate network by considering not only the embedding context related to its predecessor VNFs, but also those of its successor VNFs to tackle the causality issue. Parameter h in our proposed h-HSLGA is defined as the look-ahead horizon, which governs the size of the state space to be explored while taking an irreversible decision. Our proposed h-HSLGA also relies on a window-based approach which defines a sliding window to identify a subset of VNFs in a VNF-FG of a NS in each exploration step, where a sliding window size w is set to be $h + 1$. Furthermore, we utilize move-forward and go-back mechanisms to allow VNFs within a given sliding window to be embedded and then re-embedded based on the embedding decisions of other VNFs that reside within the sliding window at a given exploration step. The re-embeddings of the VNFs within the window are carried out until it does not lead to any improvement of embedding cost. We note that our proposed h-HSLGA is compatible with any sequential VNF-FG embedding algorithm. Without loss of generality, we adopt SGA [41] as our embedding solution. Once no further re-embedding is needed, we move to the next exploration step, where the sliding window is shifted one VNF towards the next VNF. In each exploration step, some VNFs may be left out of the sliding window. The embedding decisions of such VNFs are considered irreversible, meaning that they cannot be re-embedded again. Our proposed h-HSLGA repeats the aforementioned process until all the VNFs of the given VNF-FG are successfully embedded into the substrate network.

5.6.2 Fixed-Size Sliding Window

In this subsection, we present our h-HSLGA with a fixed-size sliding window (h-HSLGA-FSW). In this algorithm, the size of the sliding window is fixed at every exploration step. The pseudo-code of our proposed h-HSLGA-FSW is presented in Algorithm 5.1. More specifically, it acquires NS and substrate network information, and horizon parameter h as the input and returns the embedding of the VNF-FG of the NS into the substrate network as the output. The horizon parameter h indicates a fixed window size w in all exploration steps, where $w = h + 1$. The proposed h-HSLGA-FSW starts by setting the embedding decisions of all VNFs in the VNF-FG to be reversible. Next, it identifies a set of all service chains between the source and destinations in the VNF-FG and the set of VNFs for each service chain. It then sorts the service chains according to the ratio of the number of VNFs and the latency requirement of the service chain in a descending order (see lines 1-4, Algorithm 5.1). This prioritizes the service chain with larger number of VNFs and more stringent latency requirements. For instance, in the VNF-FG illustrated in Figure. 5.4, there are two service chains in total. The first one consists of VNFs A, B, C, D, E, and F, while the second one contains VNFs, A, B, G, and H. Let us also assume that the latency requirements of the first and second service chains are 5 ms and 4 ms, respectively. After running our sorting procedure explained above, the first service chain is selected to be embedded first.

Next, our proposed h-HSLGA-FSW aims to embed the VNFs of the service chain into the substrate network in a sequential manner. It first starts by defining a sliding window of size w for the given service chain g. Generally, $w = h + 1$, and $w \leq |\mathcal{N}_g| - n_f + 1$. n_f is the first VNF index in service chain g, where the embedding decision of its corresponding VNF is still reversible. If $w > |\mathcal{N}_g| - n_f + 1$, w is set to be $|\mathcal{N}_g| - n_f + 1$ (see lines 6-11, Algorithm 5.1). After that, it starts the sliding window with a size of w at the VNF corresponding to index n_f. For example, Figure 5.4a shows the sliding window at exploration step 1 for the first service chain for $h = 2$. Then, it embeds all the VNFs within the current window that have not been embedded yet using a sequential embedding algorithm (see lines 13-15, Algorithm 5.1). Next, it attempts to re-embed the VNFs within the current window, based on the new contexts generated by embeddings/re-embeddings of the other VNFs (see lines 16-23, Algorithm 5.1). The re-embedding process can be done in any arbitrary order of the VNFs, though we argue that the process should start from the

youngest VNF[4] in the current window, as it is the VNF that can benefit the most from the newly generated context by its neighbor VNFs. For instance, at exploration step 1, the re-embedding order starts from VNF B, as shown in Figure. 5.4a. The re-embedding process is repeated until it does not lead to any cost improvement.

Algorithm 5.1 h-HSLGA-FSW

Input: VNF-FG of the incoming NS (including its VNFs, virtual links, and requirements), substrate network information, and horizon parameter h

Output: Embedding of the VNF-FG onto the substrate network

1: **Set** embedding decisions of all VNFs in the VNF-FG to be reversible
2: $\mathcal{G} = \{1, .., |\mathcal{G}|\} \leftarrow$ **find** a set of chain indices between the source and destinations in the VNF-FG
3: $\mathcal{N}_g = \{1, .., |\mathcal{N}_g|\} \leftarrow$ **find** a set of VNF indices associated with chain $g \in \mathcal{G}$
4: $\mathcal{G}' \leftarrow$ **sort** all $g \in \mathcal{G}$ according to a ratio between the number of VNFs $|\mathcal{N}_g|$ and the latency requirement δ_g in descending order
5: **for** each $g \in \mathcal{G}'$ **do**
6: $n_f \leftarrow$ **find** the first index $n \in \mathcal{N}_g$, where embedding decision of its corresponding VNF is still reversible
7: $w \leftarrow h + 1$
8: **if** $w > |\mathcal{N}_g| - n_f + 1$ **then**
9: $w \leftarrow |\mathcal{N}_g| - n_f + 1$
10: **end if**
11: $n_l \leftarrow n_f + w - 1$
12: **while** $n_l \leq |\mathcal{N}_h|$ **do**
13: **for** each $n \in \{n_f, n_f + 1, ..., n_l\}$ **do**
14: **Embed** a VNF corresponding to index n using a sequential embedding algorithm if it is not embedded yet
15: **end for**
16: **Calculate** total embedding cost C_t
17: $C_t^{old} \leftarrow C_t, \Delta \leftarrow 1$
18: **while** $\Delta > 0$ **do**
19: **Re-embed** all VNFs corresponding to indices $n \in \{n_f, n_f + 1, ..., n_l\}$, starting from the youngest VNF if the re-embedding is beneficial using a sequential embedding algorithm
20: **Re-calculate** C_t
21: $\Delta \leftarrow C_t^{old} - C_t$,
22: $C_t^{old} \leftarrow C_t$
23: **end while**
24: **Set** embedding decisions of a VNF corresponding to n_f to be irreversible if they used to be in the same window as all its neighbor VNFs
25: $n_f \leftarrow n_f + 1$, $n_l \leftarrow n_l + 1$
26: **end while**
27: **end for**

Upon termination of the re-embedding process, we move to the next exploration step, where the sliding window is shifted one VNF to the next VNF. We repeat the same (re)embedding processes until the sliding window reaches the last VNF of the given service chain (see lines 12-26 of Algorithm 5.1). Figures 5.4a-5.4d demonstrate all exploration steps of our proposed h-HSLGA-FSW for the first service chain for h = 2. Essentially, we view the embedding decisions

[4]Youngest VNF refers to the most recently (re-)visited VNF, which is the last VNF of the previous sliding window.

of VNFs to be irreversible if they are left out of the window and used to be within the same window as any of their neighbor VNFs in the previous exploration steps (see line 24, Algorithm 5.1). For example, at exploration step 2 shown in Figure 5.4b, the embedding decision of VNF A is irreversible because it has been left out of the sliding window, and it was within the same window as VNF B in the previous exploration step (i.e., step 1, see Figure 5.4a). However, at exploration step 3 (see Figure 5.4c), the embedding decision of VNF B is still reversible because it was not within the same window as all its neighbor VNFs (i.e., VNF G) in the previous exploration steps. Once all the VNFs of the given service chain have been embedded, we move on to the next service chain. We repeat the same procedure until all the service chains are successfully embedded into the substrate network (see lines 5-27 of Algorithm 5.1). Figure 5.4e demonstrates the sliding window at the first exploration step of our proposed h-HSLGA-FSW ($h = 2$) for the second service chain. It is noteworthy to mention that in our proposed h-HSLGA-FSW, h is an input parameter, used to control the trade-off between complexity and solution quality.

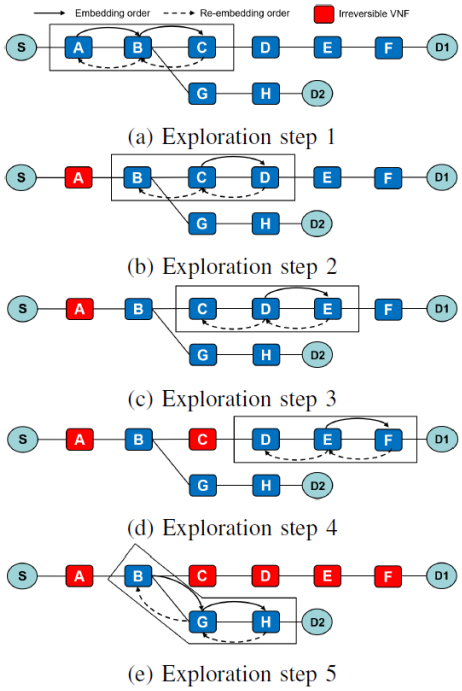


Figure 5.4: Illustration of sliding window and exploration steps for the proposed h-HSLGA-FSW with $h = 2$.

5.6.3 Variable-Size Sliding Window

The discussion above indicates that our proposed h-HSLGA-FSW is highly influenced by the selection of horizon parameter h. We note that the optimal value of horizon parameter h may vary

from one scenario to another. More precisely, a larger value of h does not always lead to a better embedding solution. Moreover, the optimal value of h is not trivial to be derived. To address this limitation, we also propose h-HSLGA with a variable-size sliding window (h-HSLGA-VSW). Unlike h-HSLGA-FSW, h-HSLGA-VSW aims at adjusting the size of the sliding window at each exploration step in order to improve the total embedding cost. The key idea of the proposed h-HSLGA-VSW can be summarized as follows. First, it increases the window size in the next exploration step whenever there is no re-embedding in the current exploration step. Second, whenever there are some re-embeddings in the current exploration step, it aims to leave out of the window the VNFs that are not likely to be re-embedded in the next exploration step. This could result in a decrease in the window size in the next exploration step, which reduces the exploration space, thus reducing the computational burden. This also gives the remaining VNFs a chance to be explored with a smaller window size in the next exploration steps once the window size has been increased in the previous exploration steps.

The pseudo-code of our proposed h-HSLGA-VSW is presented in Algorithm 5.2. Similar to h-HSLGA-FSW, the proposed h-HSLGA-VSW attempts to embed the service chain of the given VNF-FG one by one, starting from the one that has the largest number of VNFs and the most stringent latency requirement (see lines 1-4, Algorithm 5.2). Unlike h-HSLGA-FSW, it first sets the size of the sliding window to be the minimum window size w_{min} (see lines 6-7, Algorithm 5.2). Generally, w_{min} is set to be 2 (h = 1), as it is the smallest value of the window size in our proposed h-HSLGA. For instance, Figure 5.5a shows the sliding window at exploration step 1 of the proposed h-HSLGA-VSW for the first service chain. Next, the h-HSLGA-VSW applies the same embedding and re-embedding processes as h-HSLGA-FSW within the current window (see lines 9-20, Algorithm 5.2). After that, it moves to the next exploration step.

We design three main conditions for adjusting the size of the next sliding window at the next exploration step (see lines 21-33, Algorithm 5.2). First, if there are no re-embeddings of VNFs at all in the current exploration step as well as in the previous exploration step, the window is first shifted one VNF to the next VNF, and then its size is increased by one from the window tail. Note that r_{prev} is a binary parameter that indicates whether there are re-embeddings of VNFs in the previous exploration step (1) or not (0). For example, in Figure 5.5a, none of the VNFs is re-embedded at exploration step 1 and at the previous exploration step (i.e., step 0). Thus, the window is first shifted one VNF to the next VNF (i.e., VNF C). Its size is then increased by one from its

tail, so that the window size becomes 3, covering VNFs A, B, and C in the next exploration step (i.e., step 2), as shown in Figure 5.5b. This process ensures that the VNFs within the current window that are not re-embedded still stay within the next window. The reason for this is that they may have a chance to be re-embedded if they are exposed to the new context generated by other VNFs outside the window.

Second, if some of the VNFs in the current exploration step are re-embedded, the window is first shifted one VNF to the next VNF. Then, the size of the window is adjusted from its tail in such a way that the VNFs that are re-embedded in the current exploration step still stay within the window in the next exploration step. The reason for this is that these VNFs are likely to be re-embedded again if they are exposed to new contexts. This process also ensures that the VNFs that are not re-embedded at the current exploration step will be left out of the window at the next exploration step. This is due to the fact that these VNFs have already been exposed to the new context generated by re-embeddings of their neighbor VNFs in the current exploration step, though they can still not be re-embedded. For this reason, they are not likely to be re-embedded again in the next exploration step even though they have the new context of the next VNFs. For example, at exploration step 2 shown in Figure 5.5b, all the VNFs in the window (i.e., VNFs A, B, and C) are re-embedded. Therefore, the window is first shifted one VNF to the next VNF, and then its size is increased by one to ensure that VNFs A, B, and C still stay within the window in the next exploration step (i.e., step 3), see Figure. 5.5c. Moreover, at exploration step 3, only VNFs C and D are re-embedded. Consequently, after the window is first shifted one VNF to the next VNF, the window size is decreased by one to ensure that VNFs A and B are left out of the window at the next exploration step (i.e., step 4), see Figure 5.5d.

Lastly, if there are no re-embeddings of VNFs at all in the current exploration step, but some of the VNFs are re-embedded in the previous exploration step, the window is first shifted one VNF to the next VNF. Then, its size is adjusted from its tail in such a way that the VNFs that are not re-embedded in the current exploration step are left out of the next window in the next exploration step except the last VNF in the current window. The reason for keeping the last VNF in the next window is that it is not aware of the embedding location of its successor VNF yet. Therefore, it still has a high chance to be re-embedded at the next exploration step. For example, at exploration step 4 shown in Figure 5.5d, none of the VNFs is re-embedded, while at exploration step 3 shown in Figure 5.5c, some of the VNFs are re-embedded. Thus, after the window is first shifted to the

next VNF (i.e., VNF F), VNFs C and D are left out of the window in the next exploration step (i.e., step 5), as shown in Figure 5.5e.

Algorithm 5.2 h-HSLA-VSW

Input: VNF-FG of the incoming NS (including its VNFs, virtual links, and requirements), substrate network information, and minimum window size w_{min}
Output: Embedding of the VNF-FG onto the substrate network

1: **Set** embedding decisions of all VNFs in the VNF-FG to be reversible
2: $\mathcal{G} = \{1, .., |\mathcal{G}|\} \leftarrow$ **find** a set of chain indices between the source and destinations in the VNF-FG
3: $\mathcal{N}_g = \{1, .., |\mathcal{N}_g|\} \leftarrow$ **find** a set of VNF indices associated with chain $g \in \mathcal{G}$
4: $\mathcal{G}' \leftarrow$ **sort** all $g \in \mathcal{G}$ according to a ratio between the number of VNFs $|\mathcal{N}_g|$ and the latency requirement δ_g in descending order
5: **for** each $g \in \mathcal{G}'$ **do**
6: $n_f \leftarrow$ **find** the first index $n \in \mathcal{N}_g$, where embedding decision of its corresponding VNF is still reversible
7: $n_l \leftarrow n_f + w_{min} - 1$
8: **while** $n_l \leq |\mathcal{N}_g|$ **do**
9: **for** each $n \in \{n_f, n_f + 1, ..., n_l\}$ **do**
10: **Embed** a VNF corresponding to index n using a sequential embedding algorithm if it is not embedded yet
11: **end for**
12: **Calculate** total embedding cost C_t
13: $C_t^{old} \leftarrow C_t, \Delta \leftarrow 1$
14: $r_{prev} \leftarrow 0$
15: **while** $\Delta > 0$ **do**
16: **Re-embed** all VNFs corresponding to indices $n \in \{n_f, n_f + 1, ..., n_l\}$, starting from the youngest VNF if the re-embedding is beneficial using a sequential embedding algorithm
17: **Re-calculate** C_t
18: $\Delta \leftarrow C_t^{old} - C_t$
19: $C_t^{old} \leftarrow C_t$
20: **end while**
21: $n_f^{emb} \leftarrow$ **find** the first index $n \in \{n_f, n_f + 1, .., n_l\}$, where its corresponding VNF is re-embedded
22: **if** $n_f^{emb} \neq null$ **then**
23: **Set** embedding decisions of VNFs corresponding to index $n \in \{n_f, n_f + 1, .., n_l\}/\{n_f^{emb}, n_f^{emb} + 1, .., n_l\}$ to be irreversible if they used to be in the same window as all its neighbor VNFs
24: $n_f \leftarrow n_f^{emb}$
25: $r_{prev} \leftarrow 1$
26: **else**
27: **if** $r_{prev} = 1$ **then**
28: **Set** embedding decisions of VNFs corresponding to index $n \in \{n_f, n_f + 1, .., n_l - 1\}$to be irreversible if they used to be in the same window as all its neighbor VNFs
29: $n_f \leftarrow n_l$
30: **end if**
31: $r_{prev} \leftarrow 0$
32: **end if**
33: $n_l \leftarrow n_l + 1$
34: **end while**
35: **end for**

Similar to the h-HSLGA-FSW, the proposed h-HSLGAVSW sets embedding decisions of VNFs to be irreversible if they are left out of the window and used to be within the same window as their all neighbor VNFs (see line 23 and 28, Algorithm 5.2). Also, it terminates when all the

service chains are successfully embedded into the substrate network (see lines 5-35, Algorithm 5.2). Figure 5.5f shows the sliding window at the first exploration step for the second service chain for our proposed h-HSLGA-VSW.

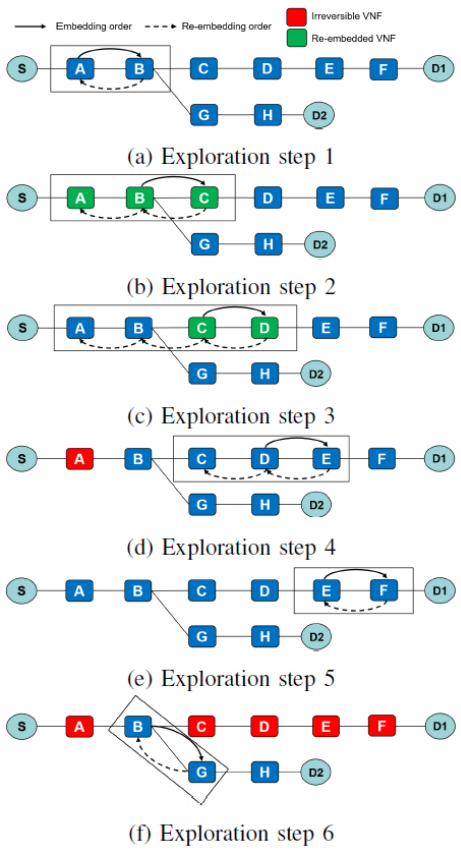


Figure 5.5: Illustration of sliding window and exploration steps for h-HSLGA-VSW.

5.7 Performance Evaluation

In this section, we conduct extensive simulations to evaluate the performance of our proposed h-HSLGA-FSW and h-HSLGA-VSW. We compare them with three benchmarks, namely SGA [41], online optimal algorithm, and online BD algorithm. In the online optimal algorithm, the ILP model proposed in Section 5.4 is solved by an exact algorithm using CPLEX to find the embedding solution of the NS, depending on its arrival while in the online BD algorithm, the ILP model is solved using the BD algorithm proposed in Section 5.5 All the simulations are conducted on a laptop with 1.60 GHz Intel(R) Core(TM) i5-8250U CPU and 8 GB of RAM. Furthermore, all simulation results are presented with 95% confidence interval.

5.7.1 Simulation Setup

In our simulations, we consider four network topologies with different sizes, including NOBEL-GERMANY (17 nodes, 26 links) [35], COST266 (37 nodes, 57 links) [148], TA2 (65 nodes, 108 links) [148] and ITC Deltacom (113 nodes and 184 links) [154]. We also assume that the number of available vCPUs of each node is 30, and the node cost is selected in the range of $(1 to 5) per vCPU. In addition, each link has 50 Gbps of available link bandwidth. The link cost and the link delay are chosen in the range of $(1 to 5) per Gbps, and between 1 and 3 ms, respectively. We also assume that the network can support 12 types of VNFs. Each VNF type requires 1 to 3 vCPUs for instantiation. The processing capacity of all VNF types is set to be 4 Gbps. While we set the reusable cost to be $1, we determine the instantiation cost of each VNF type based on parameter r_0, which is defined as the ratio of the instantiation cost to the reusable cost. It should be noted that the reusable cost is typically much smaller than the instantiation cost.

Multiple NSs are generated with three VNF-FG topologies (i.e., linear, split, and split-and-merged topologies). Each NS requires 6 to 12 VNFs in its VNF-FG. Each required VNF has the processing demand selected between 1 and 3 Gbps, and each virtual link connecting the required VNFs has the bandwidth requirement chosen between 1 and 3 Gbps. Moreover, the latency requirements of service chains in a VNF-FG are set to be from 6 to 15 ms.

5.7.2 Algorithm Performance Evaluation

First, we investigate the impact of the horizon parameter h on the performance of h-HSLGA-FSW. Figure 5.6 depicts the total embedding cost vs. horizon parameter h for different values of r_0. We note that in this figure, increasing the value of h greater than 7 does not help improve the total embedding cost, as the window size w in h-HSLGA that allows VNFs within the window to be (re)visited for (re)embedding does not exceed the number of required VNFs. Since in this scenario, the number of the required VNFs is between 6 and 8, the maximum window size w_{max} is 8, i.e., h_{max} = 7. In addition, in Figure 5.6, h = 0 represents the SGA [41], serving as our benchmark. First, we observe that h-HSLGA-FSW outperforms the SGA for a given h and/or r_0. This happens because the SGA embeds a given VNF based on only the embedding contexts related to its predecessor VNFs, thus being subject to the causality issue. In contrast, h-HSLGA-FSW allows the VNF to be revisited for re-embedding once the embedding contexts related to its successor VNFs are known as well as the new embedding contexts of its predecessor VNFs to

further improve the total embedding cost. Second, we notice that for different values of r_0, as h increases, the total embedding cost tends to be smaller. This is because for increased h, h-HSLGA-FSW allows a VNF to have more embedding contexts of its predecessor VNFs and its successor VNFs in order to find a better embedding solution via re-embedding. However, we also observe that higher values of h do not always lead to a better solution. For instance, as shown in Figure 5.6, the total embedding cost of $h = 5$ is smaller than that of $h = 7$ for $r_0 = 6$. Therefore, the horizon parameter h in h-HSLGA-FSW needs to be tuned properly to find the best embedding solution for each scenario, to be examined next. Figure 5.7 shows an upper bound and lower bound convergence of the online BD algorithm in the first step of the proposed three-step approach discussed in Section 5.5.3 to solve the VNF embedding problem for a single NS. As shown in this figure, after solving the dual LP SP and the LP MP for a number of iterations, the upper bound and the lower bound of the online BD algorithm converge. This indicates the stability of the online BD algorithm.

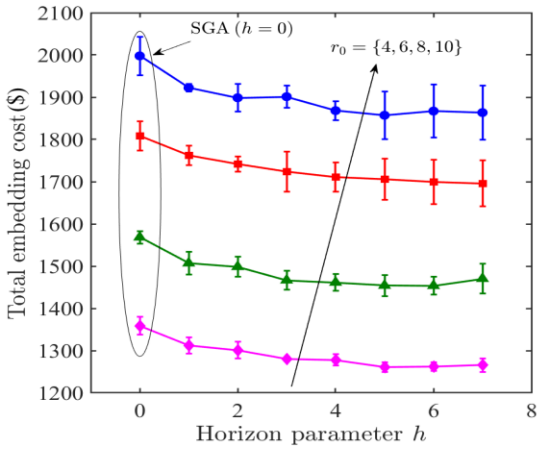


Figure 5.6: Total embedding cost vs. horizon parameter h for our proposed h-HSLGA-FSW with 17 nodes, 30 NSs and 6 to 8 VNFs per NS.

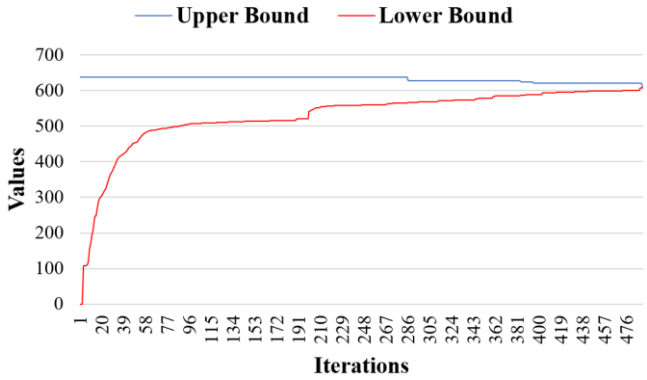


Figure 5.7. Upper bound and lower bound convergence of the online BD algorithm to solve the VNF embedding problem for a single NS with the network size of 17 nodes and r_0 = 10.

Next, we compare the performance of h-HSLGA-FSW and h-HSLGA-VSW with that of the online optimal algorithm and the online BD algorithm. For fair comparison, in h-HSLGA-FSW, we set the horizon parameter h to h_{max}, which is equal to the number of required VNFs. The reason for this is that based on our observation in Figure 5.6, setting h to be h_{max} generally tends to lead to a high-quality solution. Figure 5.8 illustrates the total embedding cost vs. number of NSs for different values of r_0. As shown in the figure, our proposed h-HSLGA-FSW and h-HSLGA-VSW both outperform SGA in all cases, because SGA suffers from the causality issue. Overall, h-HSLGA-VSW seems to perform better than h-HSLGA-FSW, even though their performance is close to each other in most cases. This is because h-HSLGA-VSW adjusts the sliding window at every exploration step to look for more possibilities to re-embed VNFs in order to further improve the total embedding cost. However, the online optimal algorithm and the online BD algorithm are better than our proposed h-HSLGA-FSW and h-HSLGA-VSW in all cases for the online optimal algorithm and most of the cases for the online BD algorithm. This is due to the fact that the online optimal algorithm relies on solving the ILP model, which provides the optimal embedding solution of a given NS at that particular network condition. The online BD algorithm also relies on solving the ILP model to find an embedding solution of a given NS. However, as we discussed earlier, the online BD algorithm does not always guarantee to return the optimal solution. Therefore, it is inferior to the online optimal algorithm in terms of the total embedding cost. It should be noted that both the online optimal and BD algorithms aim to embed a NS one by one without considering future NSs. Thus, the optimal VNF placement and assignment for embedding the current NS might not be optimal for embedding the future NSs.

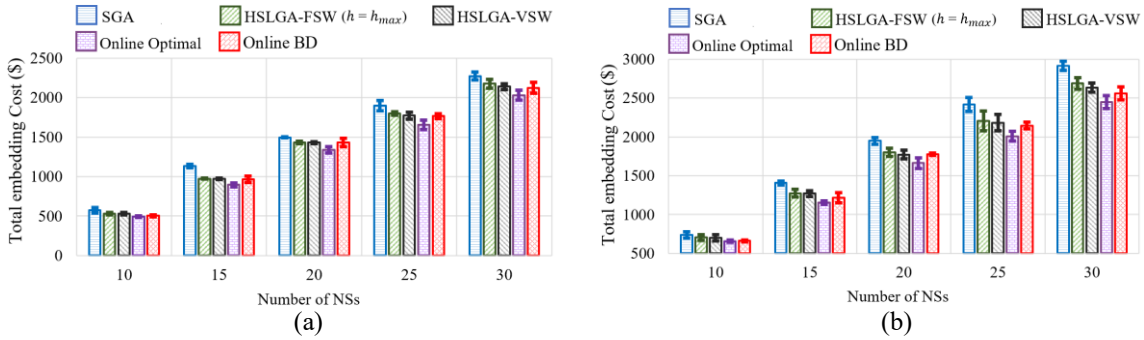


Figure 5.8: Total embedding cost vs. number of NSs with 17 nodes, 6 to 8 VNFs per NS. (a) r_0 = 4 and (b) r_0 = 10.

Figure 5.9 depicts the total embedding cost vs. number of nodes for 30 NSs, 10 to 12 VNFs per NS and r_0 = 10. We set r_0 to be 10 because the instantiation cost is typically much larger than the reusable cost. We also note that the results of the online BD algorithm are not shown here because it takes too long to obtain the embedding solutions of all the NSs. Similar to Figure 5.8, the simulation results shown in Figure 5.9 indicate that our proposed h-HSLGA-FSW and h-HSLGAVSW outperform SGA in all cases, and h-HSLGA-VSW can generally achieve a lower total embedding cost, compared to h-HSLGA-FSW. Furthermore, the total embedding costs of our proposed h-HSLGA-FSW and h-HSLGA-VSW are, on average, only 5.16% and 3.66% higher than that of the online optimal algorithm, respectively.

According to the results shown above, our proposed h-HSLGA-FSW and h-HSLGA-VSW are inferior to the online optimal algorithm and the online BD algorithm in terms of the total embedding cost. Nevertheless, they are expected to scale a lot better than both the online optimal algorithm and the online BD algorithm. Table 5.2 illustrates the execution times (in seconds) of all the algorithms under consideration. We observe from Table 5.2 that the execution times of our proposed h-HSLGA-FSW and h-HSLGA-VSW are not only much smaller, but they also grow at a lower rate than those of the online optimal algorithm and the online BD algorithm. For instance, when the numbers of nodes and NSs are 113 and 30, respectively, and r_0 = 10, the execution times of our proposed h-HSLGA-FSW and h-HSLGA-VSW are only 0.3 and 0.6 times higher than that of the SGA in order to obtain 9.52% and 9.7% improvement of the total embedding cost, respectively (see Figure 5.8 and Table 5.2). On the other hand, the online optimal algorithm took 63 times more execution time than that of the SGA in order to gain only 13.25% improvement of the total embedding cost. We note that the online BD algorithm could not even find the embedding solution within 2 hours. It is worth mentioning that the execution times of the online optimal algorithm and the online BD algorithm could become much higher if other requirements are also taken into consideration, such as VNF affinity, VNF anti-affinity, different node capabilities (i.e., memory, storage), as the ILP model becomes much more complex. Unexpectedly, the online BD algorithm is much worse than the online optimal algorithm in terms of the execution times. We believe that the main reason is that the online BD algorithm relies on only classic Bender's cuts to tighten the master problem's search space for our VNF embedding problem. This is not efficient enough to achieve a fast solution convergence, especially when the subproblem (i.e., the LP SP) for our VNF embedding problem is infeasible [151]. This suggests that more efficient approaches

are needed to generate valid cuts, especially feasibility cuts in order to see an advantage of BD in solving our VNF embedding problem.

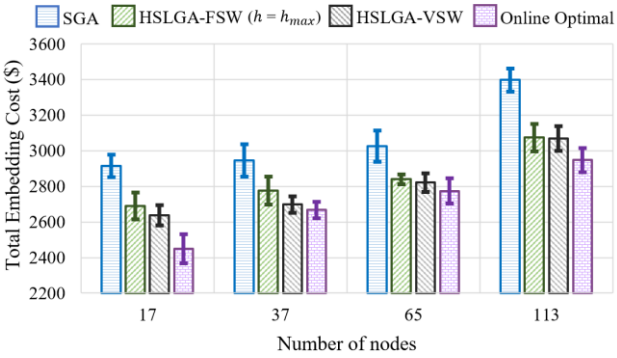


Figure 5.9: Total embedding cost vs. number of nodes for 30 NSs, 10 to 12 VNFs per NS and r_0 = 10.

Table 5.2: Average execution time.

Parameters		Algorithms (second)				
Nodes	**NSs**	**SGA**	**HSLGA-FSW**	**HSLGA-VSW**	**Online Optimal**	**Online BD**
17	10	<1	<1	<1	1	143
17	15	<1	<1	<1	2	443.8
17	20	<1	<1	<1	3	540
17	25	<1	<1	<1	3.6	869
17	30	<1	<1	<1	4.2	1194
37	30	1	1.4	1.6	30.6	> 2 hours
65	30	1.4	1.8	2	69.4	> 2 hours
113	30	2	2.6	3.2	129	> 2 hours

5.8 Conclusion

In this chapter, we addressed the so-called causality issue in the VNF embedding problem, which is generally suffered by a sequential heuristic approach to find a low-cost embedding solution for ultra-low latency NSs with timely service deployment times. This causality issue generally occurs when any given VNF is embedded without considering embedding contexts of all other VNFs in a VNF-FG. To this end, we proposed our novel VNF embedding algorithm, called h-horizon sequential look-ahead greedy algorithm, which relies on a window-based approach to provide efficient embedding and re-embedding mechanisms to alleviate the impact of the causality issue. The objective is to minimize the total embedding costs while satisfying the latency requirements. Our proposed algorithm is designed in two variants, namely, (i) fixed-size and (ii) variable-size sliding windows. Extensive simulations were conducted to evaluate the

performance of our proposed algorithm. The simulation results show that our proposed algorithm can achieve up to 9.7% of a total cost improvement with slight increase in execution times in comparison to the existing sequential heuristic algorithm. Moreover, although the optimization-based approach can provide higher total cost improvement than our proposed algorithm by up to 3.55%, it imposes up to 62.9 times higher execution times than that of our proposed algorithm with respect to the existing sequential heuristic algorithm.

Chapter 6

6. [5]Joint VNF Embedding and Scheduling for Ultra-Low Latency Services

6.1 Introduction

Given that some ultra-low latency NSs can arrive in and depart from the network at any point in time [10], these NSs may have specific service deadlines that need to be met [15][16]. Specifically, these NSs require their traffic to be processed through an ordered set of VNFs within specific service deadlines before they depart from the network. One way to provision these NSs is to schedule these NSs on VNF instances that were already deployed in the network in order to minimize a cost while ensuring that their service deadlines must be satisfied. However, relying only on the already deployed VNF instances to serve these NSs may lead to long waiting times due to resource sharing and long communication times, which may not allow the stringent service deadlines to be met. On the other hand, although relying on newly deployed VNF instances to serve the NSs could provide an efficient way to guarantee the stringent service deadlines, it

[5]This chapter is based on a published paper [20]: N. Promwongsa *et al*, "Joint VNF Placement and Scheduling for Latency-sensitive Services," *IEEE Trans. Netw. Sci. Eng.*, vol. Early access, 2022.

imposes an additional cost (e.g., instantiation costs). Therefore, an efficient mechanism is needed to guarantee the stringent service deadlines with a minimum cost.

To address this challenge, in this chapter, we study a joint VNF embedding and scheduling problem for ultra-low latency NSs. Our aim is to optimally determine whether to schedule NSs on already deployed VNFs, or to deploy new VNFs and schedule them on newly deployed VNFs to maximize profits while guaranteeing stringent service deadlines. We formulate the joint VNF embedding and scheduling problem as an ILP problem. This problem not only considers VNF embedding and scheduling jointly, but it also accounts for traffic routing. Due to lack of scalability of the developed ILP, we propose our greedy-based heuristic algorithm to solve the problem within a feasible time. Since the greedy-based heuristic algorithm may get stuck in local optimum, we also introduce our Tabu search-based algorithm to further improve the solution quality. We evaluate our proposed heuristic algorithms through extensive simulations in both offline and online scenarios. We also compare them with the existing algorithms.

The rest of this chapter is organized as follows. We first demonstrate motivation scenarios of our joint VNF embedding and scheduling. Then, we present the system model and the problem formulation. After that, we describe our proposed greedy-based heuristic algorithm followed by our proposed Tabu search-based heuristic algorithm. Next, we present our detailed performance evaluation. Finally, we conclude this chapter.

6.2 Motivation Scenario

In this subsection, we describe the joint VNF embedding and scheduling problem for ultra-low latency NSs and illustrate its advantages compared to the scheduling-only case. First, let us assume that an NFV-based infrastructure consists of four NFV-enabled nodes and five links, as shown in Figure 6.1. We also assume that NFV-enabled nodes reside in close proximity to users, given that deploying VNFs in remote NFV-enabled nodes may not be suitable for latency-sensitive NSs due to high communication delays. Each node is associated with an available node capacity. Each link is attributed to a link capacity and a propagation delay represented in number of time slots, assuming that time is divided into slots (i.e., t_1, t_2, t_3, ...). In our example, we assume that the available capacities of Node 1, Node 2, Node 3, and Node 4 are 0, 0, 2 and 2 units, respectively. Also, the link capacities and propagation delays of all the links are set to 34 Mbps and 1 time slot, respectively.

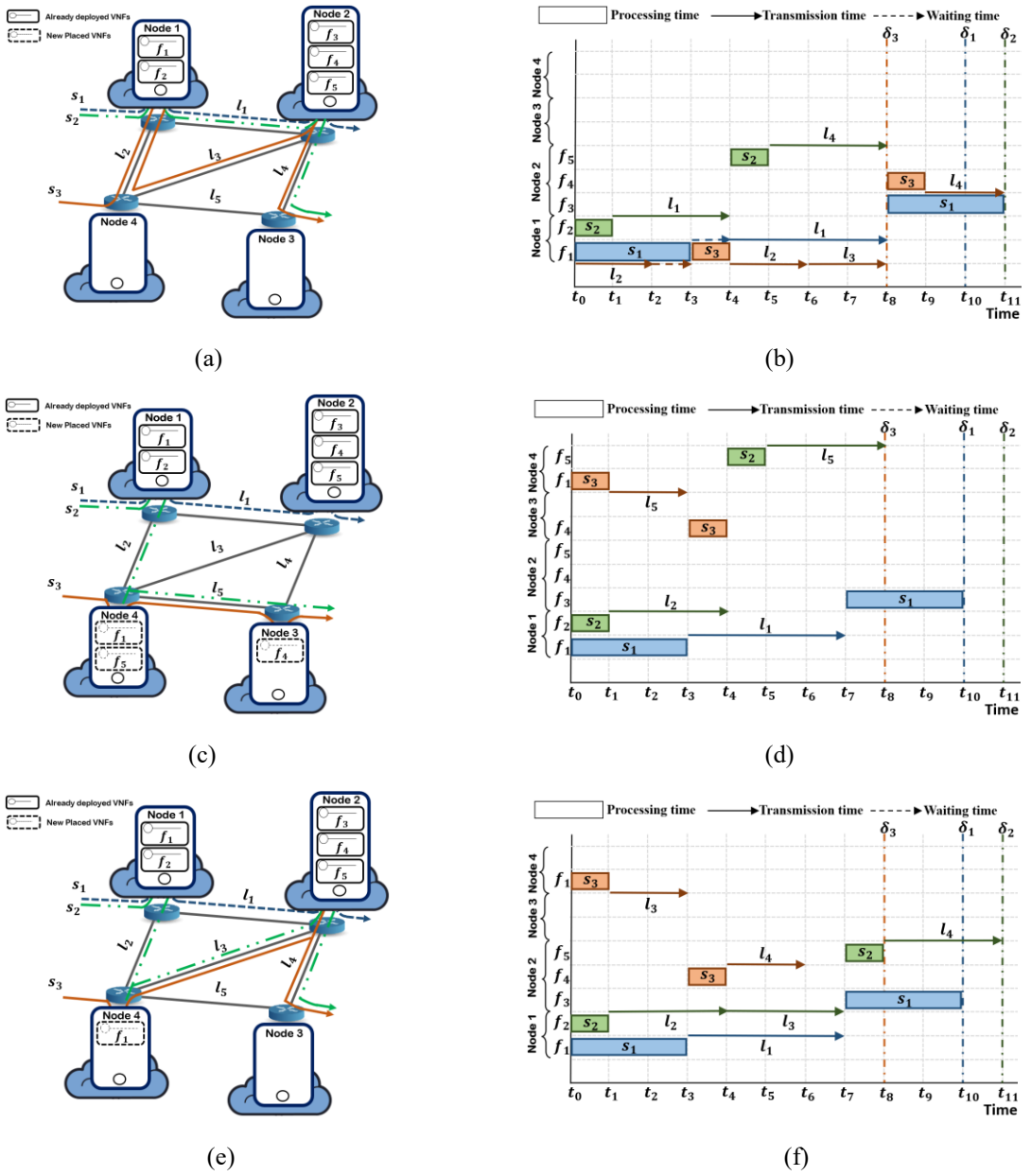


Figure 6.1: Illustrations of the joint VNF embedding and scheduling for latency-sensitive services. (a) and (b) aim to minimize the cost, (c) and (d) aim to minimize the completion time, and (e) and (f) aim to minimize the costs while satisfying the service deadlines.

Let us assume that some instances of VNF types f_1 to f_5 have already been deployed in the NFV-enabled nodes. These VNFs can be shared by multiple latency-sensitive NSs, but each of them has enough capacity to process the traffic of only one NS at a time [15][16]. Each VNF requires a certain computing resource (i.e., CPU, RAM, storage) for instantiation and is associated with a processing capacity. It runs on a dedicated virtual machine hosted in NFV-enabled nodes to avoid any switching overhead [64]. Moreover, we assume that new VNF instances can be

deployed in the network, if the already deployed VNFs are not sufficient to support NSs at any specific time. We note that the newly deployed VNFs come at the expense of an additional cost (e.g., license costs, instantiation costs). Without loss of generality, we assume that all VNFs have the same processing capacities of 24 Mbps. In addition, instantiating a new VNF instance imposes 10 additional cost units, and each VNF requires one unit of node capacity. With all these considerations in mind, let us consider three NSs that arrive in the network at t_0. These NSs can be represented as a tuple ($\mathcal{F}^s$, k^s_{src}, k^s_{dst}, a^s, w^s, b^s, δ^s), where $\mathcal{F}^s$ is an ordered set of required VNFs, k^s_{src} is the source node, k^s_{dst} is the destination node, a^s is the service arrival time, w^s is the traffic size, b^s is the required bandwidth, and δ^s is the service deadline of NS s. The incoming NSs s_1, s_2, and s_3 are given as follows: s_1 = ($\{f_1, f_3\}$, k_1, k_2, t_0, 72 Mb, 24 Mbps, t_{10}), s_2 = ($\{f_2, f_5\}$, k_1, k_3, t_0, 24 Mb, 12 Mbps, t_{11}), and s_3 = ($\{f_1, f_4\}$, k_4, k_3, t_0, 24 Mb, 24 Mbps, t_8).

A cost-efficient way to schedule these three NSs is to steer their traffic through the already deployed VNFs, as illustrated in Figure 6.1a and 6.1b. s_1, s_2, and s_3 are scheduled on the already deployed VNFs in Node 1 and Node 2. While the traffic of s_1 and s_2 are routing through link l_1 and links $l_1 \rightarrow l_4$, respectively, that of s_3 is routed through links $l_2 \rightarrow l_2 \rightarrow l_3 \rightarrow l_4$, as shown in Figure 6.1a. Although scheduling NSs only on the already deployed VNFs can minimize the cost, this may cause long waiting times, which could lead to the violation of the given service deadlines, as shown in Figure 6.1b. For instance, at t_3, s_1 needs to wait for 1 time slot before its traffic is routed through l_1, as at that time, l_1 does not have enough capacity to guarantee the required bandwidths of both s_1 and s_2 at the same time (link capacity = 34 Mbps, but the total required bandwidths of s_1 and s_2 = 24 Mbps + 12 Mbps = 36 Mbps). This assumption aligns with that in [15][16]. We note that the communication delay of a NS depends on link delay, which consists of the transmission and propagation delays. Moreover, at t_2, s_3 needs to wait for 1 time slot before its traffic is processed by f_1 in Node 1, as at that time, f_1 is processing the s_1 traffic. Consequently, s_1 and s_3 do not meet their service deadlines.

One way to reduce the completion time is to deploy new VNFs f_1 and f_5 on Node 4 and f_4 on Node 3, and schedule s_2 and s_3 on these newly deployed VNFs along with a proper traffic routing to minimize the waiting time, as illustrated in Figure 6.1c and 6.1d. In this case, none of the NSs experiences any waiting time for traffic processing and transmission. Therefore, all the NSs meet their service deadlines. This example indicates that instantiating new VNFs and using proper traffic routing in NS scheduling make it easier to meet the service deadlines. However,

instantiation of new VNF comes at the expense of an additional cost. In this case, the additional cost is 30 cost units.

In order to minimize the cost while guaranteeing the service deadlines, the NSs should be scheduled on as many already deployed VNFs as possible before instantiating new VNFs, as shown in Figure 6.1e and 6.1f. In this case, only one new instance is deployed in the network (i.e., f_1 in Node 4) while the service deadlines of all the NSs are still satisfied. Also, the additional cost is only 10 cost units. Therefore, our objective is to optimally determine whether to schedule NSs on already deployed VNFs, or to deploy new VNFs and schedule the NSs on the newly deployed VNFs while guaranteeing the service deadlines in order to optimize the profit, which is defined as the difference between the total revenue and total cost.

6.3 System Model

In our problem, we represent a substrate network as a graph $(\mathcal{K}, \mathcal{L})$ where $\mathcal{K}$ is the set of NFV-enabled nodes, and $\mathcal{L}$ is the set of physical links. Each node $k \in \mathcal{K}$ is associated with an available node capacity r_k, and every link $l = (k_1, k_2) \in \mathcal{L}$ has a link capacity c_l and a fixed propagation delay d_l where k_1 and $k_2 \in \mathcal{K}$. Recall from Section 6.2 that some VNF instances were already deployed in NFV-enabled nodes and can be shared by a number of NSs with similar service requirements. Thus, we denote a binary parameter $\tilde{z}_{i,k}^{f} \in \{0,1\}$ to indicate whether instance $i \in \mathcal{I}_f$ of VNF type $f \in \mathcal{F}$ was deployed in node $k \in \mathcal{K}$ (1) or not (0). While $\mathcal{F}$ is the set of VNF types, $\mathcal{I}_f = \{1,2,...,|\mathcal{I}_f|\}$ indicates a set of VNF instances associated with VNF type $f \in \mathcal{F}$, where $|\mathcal{I}_f|$ is also an input parameter that determines the maximum number of instances of VNF type f allowed to be deployed in the network. In addition, each VNF type $f \in \mathcal{F}$ is associated with a processing capacity p_f (measured by processing speed) and has a predefined resource requirement u_f (measured by the number of CPUs). Furthermore, if VNF type f is newly deployed in the network, it imposes an additional cost Θ_f (e.g., license cost, instantiation cost).

Let $\mathcal{S}$ be the set of latency-sensitive NSs. Each NS $s \in \mathcal{S}$ arrives at time instant a^s and requires traffic size w^s to be transmitted from a source node k_{src}^s with guaranteed bandwidth b^s and processed by an ordered set of VNFs before reaching a destination node k_{dst}^s within a service deadline δ^s, where k_{src}^s and $k_{dst}^s \in \mathcal{K}$. We also define two binary parameters, μ_k^s and $\pi_k^s \in \{0,1\}$ to indicate whether node k is a source node or a destination node of NS s, respectively.

More precisely, $\mu_k^s = 1$ if $k = k_{src}^s$ while $\pi_k^s = 1$ if $k = k_{dst}^s$. Otherwise, both are zero. In addition, we denote $\mathcal{N}^s \in \{0,1,\ldots, \eta^s, \eta^s{+}1\}$ as a set of VNF indices of NS $s \in \mathcal{S}$, where η^s is the number of required VNFs, 0 and $\eta^s + 1$ indicate virtual source and destination node indices, respectively, and 1 to η^s indicates indices of VNFs required by NS s according to their order. We also define v_n^s to specify a type of VNF index $n \in \mathcal{N}^s$ required by NS s. It should be noted that we use VNF type 0 and $|\mathcal{F}| + 1$ to represent a virtual source node and a virtual destination node of NSs, respectively, and they must be mapped into nodes that are associated with a source node and a destination node of NSs. For instance, for any NS $s \in \mathcal{S}$, $v_0^s = 0$ and $v_{\eta^s+1}^s = |\mathcal{F}| + 1$, and v_0^s and $v_{\eta^s}^s$ must be mapped into k_{src}^s and k_{dst}^s, respectively. Moreover, considering a given NS s, we define $\mathcal{E}^s = \{(v_0^s, v_1^s), (v_1^s, v_2^s), \ldots, (v_{\eta^s}^s, v_{\eta^s+1}^s)\}$ to represent a set of its virtual links that also includes the virtual links between its virtual source node and first VNF, and its last VNF and virtual destination node. For each $e \in \mathcal{E}^s$, α_e^s and β_e^s are also defined to represent origin and destination of virtual link e, respectively. For example, if $e = (v_n^s, v_{n+1}^s)$, $\alpha_e^s = v_n^s$ and $\beta_e^s = v_{n+1}^s$. Finally, we assume that the time domain is divided into time slots and $\mathcal{T}$ denotes the set of time slots.

Table 6.1: Summary of key notations for the joint VNF embedding and scheduling problem.

Input Parameters	
$\mathcal{K}$	Set of NFV-enabled nodes
$\mathcal{L}$	Set of physical links
r_k	Available node capacity of physical node $k \in \mathcal{K}$
c_l	Link capacity of link $l \in \mathcal{L}$
d_l	Propagation delay of link $l \in \mathcal{L}$
$\mathcal{F}$	Set of supported VNF types
$\mathcal{I}_f$	Set of instances associated with VNF type f
p_f	Processing capacity of VNF type $f \in \mathcal{F}$
u_f	Resource requirement of VNF type $f \in \mathcal{F}$ for instantiation
Θ_f	Additional cost for instantiating VNF type $f \in \mathcal{F}$
$\mathcal{S}$	Set of NSs
a^s	Service arrival time of NS $s \in \mathcal{S}$
w^s	Traffic size of NS $s \in \mathcal{S}$
b^s	Bandwidth requirement of NS $s \in \mathcal{S}$
k_{src}^s	Source node of NS $s \in \mathcal{S}$, where $k_{src}^s \in \mathcal{K}$
k_{dst}^s	Destination node of NS $s \in \mathcal{S}$, where $k_{dst}^s \in \mathcal{K}$
$\mathcal{N}^s$	Set of VNF indices of NS $s \in \mathcal{S}$
η^s	Number of required VNFs of NS $s \in \mathcal{S}$
v_n^s	Required VNF type of VNF index n of NS $s \in \mathcal{S}$
$\mathcal{E}^s$	Set of virtual links of NS $s \in \mathcal{S}$
α_e^s	Origin of virtual link $e \in \mathcal{E}^s$ of NS $s \in \mathcal{S}$
β_e^s	Destination of virtual link $e \in \mathcal{E}^s$ of NS $s \in \mathcal{S}$
$\mathcal{T}$	Set of time slots
$\tilde{z}_{i,k}^f$	Binary parameter, indicating that instance $i \in \mathcal{I}_f$ of VNF type f was already deployed on node $k \in \mathcal{K}$
μ_k^s	Binary parameter, indicating whether node $k \in \mathcal{K}$ is a source node of NS $s \in \mathcal{S}$
π_k^s	Binary parameter, indicating whether node $k \in \mathcal{K}$ is a destination node of NS $s \in \mathcal{S}$

6.4 Problem Formulation

To formulate our joint VNF embedding and scheduling problem, we define seven decision variables as follows:

- $z_{i,k}^{f} \in \{0,1\}$ is a binary variable, indicating whether instance $i \in \mathcal{I}_f$ of VNF type $f \in \mathcal{F}$ is deployed in node $k \in \mathcal{K}$ (1) or not (0).
- $x_{i,k,t}^{s,f} \in \{0,1\}$ is a binary variable, specifying whether traffic of NS $s \in \mathcal{S}$ starts being processed by instance $i \in \mathcal{I}_f$ of VNF type $f = v_n^s$ required by NS s on node $k \in \mathcal{K}$ at time slot $t \in \mathcal{T}$ (1) or not (0), where $n \in \mathcal{N}^s \backslash \{0, \eta^s+1\}$.
- $\psi_k^{s,f} \in \{0,1\}$ is a binary variable, indicating whether an instance of VNF type $f = v_n^s$ assigned to NS $s \in S$ is hosted on node $k \in \mathcal{K}$ (1) or not (0), where $n \in \mathcal{N}^s$.
- $h_{l,t}^{s,e} \in \{0,1\}$ is a binary variable, indicating whether traffic of NS $s \in \mathcal{S}$ on virtual link $e \in \mathcal{E}^s$ starts being transmitted through link $l \in \mathcal{L}$ at time slot $t \in \mathcal{T}$ (1) or (0).
- $g_{l,t}^{s,e} \in \{0,1\}$ is a binary variable, indicating whether traffic of NS $s \in \mathcal{S}$ on virtual link $e \in \mathcal{E}^s$ is transmitted through link $l \in \mathcal{L}$ at time slot $t \in \mathcal{T}$ (1) or (0).
- $y_l^{s,e} \in \{0,1\}$ is a binary variable, specifying whether traffic of NS $s \in \mathcal{S}$ on virtual link $e \in \mathcal{E}^s$ is routed through link $l \in \mathcal{L}$ (1) or not (0).
- $\tau^s \in \{0,1\}$ is a binary variable, indicating whether NS $s \in \mathcal{S}$ is admitted into the network (1) or not (0).

Next, we describe the constraints that need to be considered in our problem. Constraint (6.1) ensures that if NS s is admitted into the network, only one of any instance i of its required VNF type v_n^s hosted on any node $k \in \mathcal{K}$ must be assigned to process its traffic.

$$\sum_{i \in \mathcal{I}_{v_n^s}} \sum_{k \in \mathcal{K}} \sum_{t \in \mathcal{T}} x_{i,k,t}^{s,v_n^s} = \tau^s, \forall s \in \mathcal{S}, \forall n \in \mathcal{N}^s \backslash \{0, \eta^s + 1\} \tag{6.1}$$

Constraint (6.2) ensures that if instance i of VNF type v_n^s required by NS s is assigned to process its traffic on node $k \in \mathcal{K}$ at any point in time, this instance must be deployed in that node.

$$\sum_{t \in \mathcal{T}} x_{i,k,t}^{s,v_n^s} \leq z_{i,k}^{v_n^s}, \forall s \in \mathcal{S}, \forall n \in \mathcal{N}^s \backslash \{0, \eta^s + 1\}, \forall i \in \mathcal{I}_{v_n^s}, \forall k \in \mathcal{K} \tag{6.2}$$

Constraint (6.3) specifies that instance i of VNF type $f \in \mathcal{F}$ must not be deployed more than once across the network.

$$\sum_{k \in \mathcal{K}} z_{i,k}^{f} \leq 1, \forall f \in \mathcal{F}, \forall i \in \mathcal{I}_f \tag{6.3}$$

Constraint (6.4) indicates that if instance i of VNF type $f \in \mathcal{F}$ was already deployed in node $k \in \mathcal{K}$, it cannot be deployed in any other node.

$$\tilde{z}_{i,k}^{f} \leq z_{i,k}^{f}, \forall f \in \mathcal{F}, \forall i \in \mathcal{I}_f, \forall k \in \mathcal{K} \tag{6.4}$$

Constraint (6.5) ensures that any node $k \in \mathcal{K}$ must have enough capacity if VNF instances are to be newly deployed in that node.

$$\sum_{f \in \mathcal{F}} \sum_{i \in \mathcal{I}_f} u_f \left(z_{i,k}^{f} - \tilde{z}_{i,k}^{f} \right) \leq r_k, \forall k \in \mathcal{K} \tag{6.5}$$

Constraints (6.6a)-(6.6d) specify node $k \in \mathcal{K}$ on which an instance of VNF type v_n^s assigned to NS s is hosted.

$$\psi_k^{s,v_n^s} \leq \sum_{i \in \mathcal{I}_{v_n^s}} \sum_{t \in \mathcal{T}} x_{i,k,t}^{s,v_n^s}, \forall s \in \mathcal{S}, \forall n \in \mathcal{N}^s \backslash \{0, \eta^s + 1\}, \forall k \in \mathcal{K} \tag{6.6a}$$

$$\psi_k^{s,v_n^s} \geq \sum_{t \in \mathcal{T}} x_{i,k,t}^{s,v_n^s}, \forall s \in \mathcal{S}, \forall n \in \mathcal{N}^s \backslash \{0, \eta^s + 1\}, \forall i \in \mathcal{I}_{v_n^s}, \forall k \in \mathcal{K} \tag{6.6b}$$

$$\psi_k^{s,v_0^s} = \tau^s \mu_k^s, \forall s \in \mathcal{S}, \forall k \in \mathcal{K} \tag{6.6c}$$

$$\psi_k^{s,v_{\eta^s+1}^s} = \tau^s \pi_k^s, \forall s \in \mathcal{S}, \forall k \in \mathcal{K} \tag{6.6d}$$

Constraint (6.7) ensures that the transmission of NS s traffic on virtual link $e \in \mathcal{E}^s$ transmitted through link $l \in \mathcal{L}$ will not happen if NS s is not admitted into the network.

$$\sum_{t \in \mathcal{T}} h_{l,t}^{s,e} \leq \tau^s, \forall s \in \mathcal{S}, \forall e \in \mathcal{E}^s, \forall l \in \mathcal{L} \tag{6.7}$$

Constraint (6.8) ensures that the NS s traffic on virtual link $e \in \mathcal{E}^s$ can start being transmitted through $l \in \mathcal{L}$ if link l is assigned to virtual link e.

$$\sum_{t \in \mathcal{T}} h_{l,t}^{s,e} = y_l^{s,e}, \forall s \in \mathcal{S}, \forall e \in \mathcal{E}^s, \forall l \in \mathcal{L} \tag{6.8}$$

Constraints (6.9a) and (6.9b) ensure that NS s should not be served before its arrival time. More precisely, transmission on the first virtual link (i.e., (v_0^s, v_1^s)) and the processing of the first VNF of NS s (i.e., v_1^s) should not happen before its arrival time.

$$\sum_{l \in \mathcal{L}} h_{l,t}^{s,(v_0^s, v_1^s)} = 0, \forall s \in \mathcal{S}, \forall t \in \mathcal{T} : t < a^s \tag{6.9a}$$

$$\sum_{i \in \mathcal{I}_{v_1^s}} \sum_{k \in \mathcal{K}} x_{i,k,t}^{s,v_1^s} = 0, \forall s \in \mathcal{S}, \forall t \in \mathcal{T} : t < a^s \tag{6.9b}$$

Constraint (6.10) ensures that the transmission of NS s traffic on virtual link $e \in \mathcal{E}^s$ through link $l \in \mathcal{L}$, excluding the first virtual link (i.e., (v_0^s, v_1^s)), will not happen unless its traffic processing on VNF type α_e^s of virtual link e's origin has finished.

$$h_{l,t'}^{s,e} \leq 1 - \sum_{i \in \mathcal{I}_{\alpha_e^s}} \sum_{k \in \mathcal{K}} x_{i,k,t}^{s,\alpha_e^s}, \forall s \in \mathcal{S}, \forall e \in \mathcal{E}^s \backslash \{(v_0^s, v_1^s)\}, \forall l \in \mathcal{L}, \forall t, t' \in \mathcal{T}: t' < t + \frac{w^s}{p_{\alpha_e^s}} \quad (6.10)$$

Constraint (6.11) ensures that the traffic of NS s will not be processed by VNF type β_e^s of virtual link e's destination unless the transmission on that virtual link has completed.

$$\sum_{i \in \mathcal{I}_{\beta_e^s}} \sum_{k \in \mathcal{K}} x_{i,k,t'}^{s,\beta_e^s} \leq 1 - h_{l,t}^{s,e}, \forall s \in \mathcal{S}, \forall e \in \mathcal{E}^s, \forall l \in \mathcal{L}, \forall t, t' \in \mathcal{T}: t' < t + \frac{w^s}{b^s} + d_l \quad (6.11)$$

Constraint (6.12) ensures that if virtual link $e \in \mathcal{E}^s$ of NS s is mapped into multiple links, the transmission of NS s traffic through any given link must finish before it starts being transmitted through the next link.

$$h_{l',t'}^{s,e} \leq 1 - h_{l,t}^{s,e}, \forall s \in \mathcal{S}, \forall e \in \mathcal{E}^s, \forall l = (k_1, k_2), l' = (k_1', k_2') \in \mathcal{L}: k_2 = k_1', \quad (6.12)$$
$$\forall t, t' \in \mathcal{T}: t' < t + \frac{w^s}{b^s} + d_l$$

Constraint (6.13) ensures that the processing of NS s traffic on VNF type v_n^s of VNF index n must finish before VNF type $v_{n'}^s$ of its successor VNF index $n' = n + 1 \in \{1,\ldots,\eta^s\}$ starts processing the traffic.

$$\sum_{i \in \mathcal{I}_{v_{n'}^s}} \sum_{k \in \mathcal{K}} x_{i,k,t'}^{s,v_{n'}^s} \leq 1 - \sum_{i \in \mathcal{I}_{v_n^s}} \sum_{k \in \mathcal{K}} x_{i,k,t}^{s,v_n^s}, \forall s \in \mathcal{S}, \forall n, n' \in \mathcal{N}^s \backslash \{0, \eta^s + 1\}: n' = n + 1, \quad (6.13)$$
$$\forall t, t' \in \mathcal{T}: t' < t + \frac{w^s}{p_{v_n^s}}$$

Constraint (6.14) ensures that if instance i of VNF type v_n^s required by NS s is processing the traffic of NS s, it cannot process the traffic of any other NSs until the processing is finished.

$$\sum_{\substack{s' \in \mathcal{S}: \\ s \neq s'}} \sum_{\substack{n' \in \mathcal{N}^{s'} \backslash \{0, \eta^{s'}+1\}: \\ v_n^s = v_{n'}^{s'}}} x_{i,k,t'}^{s',v_{n'}^{s'}} \leq 1 - x_{i,k,t}^{s,v_n^s}, \forall s \in \mathcal{S}, \forall n \in \mathcal{N}^s \backslash \{0, \eta^s + 1\}, \forall i \in \mathcal{I}_{v_n^s} \forall k \in \mathcal{K}, \quad (6.14)$$
$$\forall t, t' \in \mathcal{T}: t \leq t' < t + \frac{w^s}{p_{v_n^s}}$$

Constraint (6.15) indicates that if the transmission of NS s traffic on virtual link $e \in \mathcal{E}^s$ through link $l \in \mathcal{L}$ occurs, the total amount of time slots for the transmission is equal to its total amount of required transmission time w^s/b^s plus propagation delay d_l of link l.

$$\sum_{t\in\mathcal{T}} g_{l,t}^{s,e} = \left(\frac{w^s}{b^s} + d_l\right) \sum_{t\in\mathcal{T}} h_{l,t}^{s,e}, \forall s \in \mathcal{S}, \forall e \in \mathcal{E}^s, \forall l \in \mathcal{L} \quad (6.15)$$

Constraint (6.16) specifies the time slots in which transmission of NS s traffic on virtual link $e \in \mathcal{E}^s$ through link $l \in \mathcal{L}$ occurs.

$$\sum_{t'\in\mathcal{T}: t\le t' < t+\frac{w^s}{b^s}+d_l} g_{l,t'}^{s,e} \ge \left(\frac{w^s}{b^s} + d_l\right) h_{l,t}^{s,e}, \forall s \in \mathcal{S}, \forall e \in \mathcal{E}^s, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \quad (6.16)$$

Constraint (6.17) ensures that at each time slot t, the bandwidth demands assigned to any link l must not exceed its bandwidth capacity.

$$\sum_{s\in\mathcal{S}} \sum_{e\in\mathcal{E}^s} g_{l,t}^{s,e}\, b^s \le c_l, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \quad (6.17)$$

Constraint (6.18) is a flow conservation constraint that specifies the links through which the NS s traffic on virtual link $e \in \mathcal{E}^s$ is routed. It also ensures that the traffic is steered through the ordered set of required VNFs.

$$\sum_{\substack{l=(k_1,k_2)\in\mathcal{L}:\\ k_1=k}} y_l^{s,e} - \sum_{\substack{l=(k_1,k_2)\in\mathcal{L}:\\ k_2=k}} y_l^{s,e} = \psi_k^{s,\alpha_e^s} - \psi_k^{s,\beta_e^s}, \forall s \in \mathcal{S}, \forall e \in \mathcal{E}^s, \forall k \in \mathcal{K} \quad (6.18)$$

Finally, Constraints (6.19a) and (6.19b) ensure that NS s must be served before its deadline. More specifically, if the last VNF of NS s (i.e., $v_{\eta^s}^s$) is mapped into the same node as its destination, the processing of the last VNF must finish before the deadline. Otherwise, transmission on the last virtual link i.e., $(v_{\eta^s}^s, v_{\eta^s+1}^s)$ must finish before the deadline.

$$\sum_{i\in\mathcal{I}_{v_{\eta^s}^s}} \sum_{k\in\mathcal{K}} \sum_{t\in\mathcal{T}} x_{i,k,t}^{s,v_{\eta^s}^s} \left(t + \frac{w^s}{p_{v_{\eta^s}^s}}\right) \le \delta^s, \forall s \in \mathcal{S} \quad (6.19a)$$

$$\sum_{t\in\mathcal{T}} h_{l,t}^{s,(v_{\eta^s}^s, v_{\eta^s+1}^s)} \left(t + \frac{w^s}{b^s} + d_l\right) \le \delta_s, \forall s \in \mathcal{S}, \forall l \in \mathcal{L} \quad (6.19b)$$

The objective of our joint VNF embedding and scheduling problem is to optimize the profit (γ), defined as total revenue R_{total} subtracted by total cost C_{total}. We note that the total revenue R_{total} is given by:

$$R_{total} = \omega \sum_{s\in\mathcal{S}} \tau^s \quad (6.20)$$

where ω is a reward weight, a non-negative constant value. In addition, the total cost C_{total} can be estimated by:

$$C_{total} = \sum_{f\in\mathcal{F}} \sum_{i\in\mathcal{I}_f} \sum_{k\in\mathcal{K}} \Theta_f \left(z_{i,k}^f - \tilde{z}_{i,k}^f\right) \quad (6.21)$$

Consequently, our problem can be formulated as an ILP problem, as follows:

$$max\ \gamma = R_{total} - C_{total} \tag{6.22}$$

Subject to Constraints (6.1) – (6.19)

6.5 Greedy-Based Joint VNF Embedding and Scheduling

Recall from Section 2.2 that due to its low complexity, the greedy-based approach has been widely used to solve the scheduling-only sub-problem to provide fast NS provisioning (e.g., [53] [54][59]). Inspired by this approach, we propose a greedy-based joint embedding and scheduling algorithm (GJESA) to solve the problem. Specifically, the main idea of our proposed GJESA is to schedule each NS on as many already deployed VNF instances as possible while minimizing the completion time before any new VNF instances are embedded into the network if the deadline of the NS is not satisfied. Algorithm 6.1 describes the pseudo-code of our proposed GJESA, which receives $\mathcal{S}$, $\mathcal{K}$, $\mathcal{L}$, and $\mathcal{I}_1$,..., $\mathcal{I}_{|\mathcal{F}|}$ as inputs and returns the joint embedding and scheduling solutions of each NS $s \in \mathcal{S}$, denoted by Ω_s, as outputs. To ensure that the most stringent service deadlines of NSs will be satisfied, these NSs should be scheduled as soon as possible. Therefore, we start by sorting all NSs according to their service deadlines in ascending order to make sure that the NSs with the most stringent service deadlines will be first prioritized (see line 1 in Algorithm 6.1). After that, we schedule each NS one by one.

To schedule a given NS, our proposed GJESA is divided into four phases: (i) admission control phase, (ii) scheduling phase, (iii) rescheduling phase, and (iv) new embedding phase. The admission control phase aims to pre-evaluate whether the service deadline of the NS could be satisfied based on the best-case VNF embedding and scheduling solution in terms of the completion time (see lines 2-7 in Algorithm 6.1). More specifically, in this phase, we first find the shortest path from the source to the destination of the NS taking into account the communication and waiting times. Next, we greedily and sequentially schedule all required VNFs on the already deployed or newly deployed instances on nodes along the shortest path with the objective of minimizing the completion time. If the obtained completion time is greater than the service deadline, the NS is rejected because the feasible solution of this NS cannot be found. This could reduce the computational burden. Otherwise, we move to the next phase (i.e., Phase 2).

Algorithm 6.1 Greedy-based Joint Embedding and Scheduling Algorithm

Input: $\mathcal{S}$, $\mathcal{K}$, $\mathcal{L}$, $\mathcal{I}_1$,..., $\mathcal{I}_{|\mathcal{F}|}$
Output: Ω_1,...,$\Omega_{|\mathcal{S}|}$

1: **sort** all $s \in \mathcal{S}$ according to their service deadlines ascendingly
2: **for** each $s \in \mathcal{S}$ **do**
3: $\rho \leftarrow$ **find** the shortest path from k^s_{src} to k^s_{dst}, based on the sum of the communication and waiting times, starting at arrive time a^s
4: $\delta^{best} \leftarrow$ **compute** the completion time if all VNF indices $n \in \mathcal{N}^s \setminus \{0, \eta^s + 1\}$ are greedily and sequentially scheduled on already deployed or newly placed instances corresponding to VNF type v^s_n on nodes along path ρ
5: **if** $\delta^{best} > \delta^s$ **then**
6: **reject** s
7: **else**
8: $t_{ref} \leftarrow a^s$, $k_{ref} \leftarrow k^s_{src}$
9: **for** each $n \in \mathcal{N}^s \setminus \{0\}$ **do**
10: $\mathcal{J}^{v^s_n} \leftarrow$ **find** all already deployed instances of VNF type v^s_n
11: $[i^{best}, \rho^{best}, \Delta^{best}] \leftarrow$ GetBestCandidate$(s, t_{ref}, k_{ref},$
12: $\mathcal{J}^{v^s_n}, \mathcal{K}, \mathcal{L})$
13: **schedule** VNF index n on i^{best} through path ρ^{best}
14: **update** scheduling of virtual link (v^s_{n-1}, v^s_n) and VNF index n in Ω^s
15: $k_{ref} \leftarrow$ **set** node k hosting i^{best}
16: $t_{ref} \leftarrow t_{ref} + \Delta^{best}$
17: **end for**
18: $\delta^{best} \leftarrow t_{ref}$
19: **if** $\delta^{best} \leq \delta^s$ **then**
20: **accept** s
21: **else**
22: Rescheduling$(s, \Omega^s, \mathcal{K}, \mathcal{L}, \mathcal{I}_1, ..., \mathcal{I}_{|\mathcal{F}|})$
23: **end if**
24: **end if**
25: **end for**

In Phase 2, we aim to sequentially schedule each VNF index $n \in \mathcal{N}^s \setminus \{0\}$ and virtual link (v^s_{n-1}, v^s_n) on already deployed instances of VNF type v^s_n and links, respectively with objective of minimizing the communication, waiting and processing times (see lines 8-17 in Algorithm 6.1). More precisely, for each VNF index n, we find the best candidate instance i^{best} of VNF type v^s_n from the set of already deployed instances $\mathcal{J}^{v^s_n}$, and the best candidate path ρ^{best} to that instance. To obtain i^{best} and ρ^{best}, a GetBestCandidate function is called (see line 11 of Algorithm 6.1). Algorithm 6.2 presents the pseudo-code of the GetBestCandidate function, which aims to select instance i from $\mathcal{J}^{v^s_n}$ that leads to the smallest completion time for scheduling virtual link (v^s_{n-1}, v^s_n) and VNF index n. In this function, we first find the shortest path ρ^i from node k_{ref} to node

k hosting each instance $i \in \mathcal{J}^{v_n^s}$, based on the sum of the communication and the waiting times starting at t_{ref}. Here, k_{ref} refers to a node hosting an instance of VNF type v_{n-1}^s on which VNF index $n-1$ is currently scheduled while t_{ref} is a time slot at which that instance of VNF type v_{n-1}^s finishes processing the NS s traffic. We note that when $n = 1$, k_{ref} and t_{ref} are equal to k_{src}^s and a^s, respectively. Next, we compute the completion time, which includes the communication, waiting, and processing times for each instance $i \in \mathcal{J}^{v_n^s}$, denoted as Δ^i. Finally, the function returns instance i and path ρ^i associated with the smallest completion time Δ^i as the best candidate instance i^{best} and path ρ^{best}, respectively. After all VNF indices of NS s are successfully scheduled, we check whether the service deadline is met or not. If yes, we accept NS s in the network. Otherwise, we move to the next phase (i.e., Phase 3) by calling the Rescheduling function (see line 22 in Algorithm 6.1).

Algorithm 6.2 GetBestCandidate

Input: s, t_{ref}, k_{ref}, $\mathcal{J}^{v_n^s}$, $\mathcal{K}$, $\mathcal{L}$
Output: i^{best}, ρ^{best}, Δ^{best}
1: $\mathcal{P}^{best} \leftarrow \{\emptyset\}$, $\Lambda^{best} \leftarrow \{\emptyset\}$
2: **if** $n \neq \eta^s + 1$ **then**
3: **for** each $i \in \mathcal{J}^{v_n^s}$ **do**
4: $\rho^i \leftarrow$ **find** the shortest path from k_{ref} to node k hosting instance i, based on the sum of the communication and waiting times, starting at t_{ref}
5: $\Delta^i \leftarrow$ **compute** the completion time if VNF index n and virtual link (v_{n-1}^s, v_n^s) are scheduled on instance i and path ρ^i, respectively
6: **add** ρ^i to $\mathcal{P}^{best}$ and **add** Δ^i to Λ^{best}
7: **end for**
8: $\Delta^{best} \leftarrow$ **select** the smallest completion time $\Delta^i \in \Lambda^{best}$
9: $i^{best} \leftarrow$ **select** instance $i \in \mathcal{J}^{v_n^s}$ corresponding to Δ^{best}
10: $\rho^{best} \leftarrow$ **select** path $\rho^i \in \mathcal{P}^{best}$ corresponding to Δ^{best}
11: **else**
12: $\rho^{best} \leftarrow$ **find** the shortest path from k_{ref} to k_{dst}^s, based on the sum of the communication and waiting times, starting at t_{ref}
13: $\Delta^{best} \leftarrow$ **compute** the completion time if virtual link (v_{n-1}^s, v_n^s) is scheduled on path ρ^{best}
14: **end if**

In Phase 3, we aim to alleviate the impact of the so-called causality issue to improve the scheduling solution obtained in Phase 2 to be able to meet the service deadline. We observe that this causality issue also exists in a sequential scheduling approach, which is similar to the sequential VNF embedding approach discussed in Chapter 5. In particular, our sequential greedy scheduling approach in Phase 2 could encounter the causality issue, as it considers only the

scheduling information of VNF index $n-1$ when scheduling VNF index n. Therefore, in this phase, we introduce our rescheduling strategy inspired by our proposed h-HSLGA discussed in Chapter 5 to tackle this issue. Algorithm 6.3 outlines the pseudocode of our proposed strategy, where we start by sorting all $n' \in \mathcal{N}^s \backslash \{0, \eta^s+1\}$ according to the completion time of traffic scheduling of virtual links $(v^s_{n'-1}, v^s_{n'})$ and $(v^s_{n'}, v^s_{n'+1})$, and VNF index n' in Ω^s in descending order. Then, we aim to reschedule each VNF index n' by considering the scheduling information of both VNF indices n' - 1 and n' + 1. More precisely, we first find the shortest path from the nodes hosting VNF instances on which VNF indices n' - 1 and n' + 1 are currently scheduled. After that, we find the best instance i^{best} from the list of all already deployed instances of VNF type $v^s_{n'}$ along that shortest path (see lines 3-13 in Algorithm 6.3). If i^{best} leads to a better completion time of traffic scheduling of virtual links $(v^s_{n'-1}, v^s_{n'})$ and $(v^s_{n'}, v^s_{n'+1})$ and VNF index n' than that of the old instance, we reschedule VNF index n' on i^{best}. Since the rescheduling of VNF index n' could impact the scheduling of all other VNF indices that are higher than n', we also reschedule them (see lines 14-17 in Algorithm 6.3). If the service deadline is satisfied, we accept NS s and stop Algorithm 6.3; otherwise, we attempt to reschedule the next VNF index $n' \in \mathcal{M}^s$. If all indices $n' \in \mathcal{M}^s$ are attempted to be rescheduled and the service deadline is still not satisfied, we move to the last phase (i.e., Phase 6.4) and call the NewEmbedding function (see line 27 in Algorithm 6.3).

In Phase 4, we aim to incrementally embed new instances of VNFs required by NS s into the network and reschedule it to further improve the completion time until the service deadline is satisfied. Algorithm 6.4 outlines the pseudo-code of the NewEmbedding function. In this function, we start by sorting all $n' \in \mathcal{N}^s \backslash \{0, \eta^s+1\}$ according to the total waiting time of traffic scheduling of virtual links $(v^s_{n'-1}, v^s_{n'})$ and $(v^s_{n'}, v^s_{n'+1})$, and VNF index n' in Ω^s in descending order. For each $n' \in \mathcal{M}$, a new instance of VNF type $v^s_{n'}$ is then deployed in the network. To find a hosting node for this new instance, we first find the shortest path from nodes hosting VNF instances on which VNF indices $n'-1$ and $n'+1$ are currently scheduled in Ω^s, based on the sum of the communication and waiting times. We next select the node with the highest available resources in that path as the hosting node. If the node has enough resources to host a VNF instance of VNF type $v^s_{n'}$, a new instance is deployed in that node. Then, it reschedules VNF index n' on i^{new} as well as rescheduling all VNF indices n that are higher than n' (see lines 3-9 in Algorithm 6.4).

After that, we check whether the service deadline of NS s is satisfied or not. If yes, we accept NS s and stop the function (see lines 10-14 in Algorithm 6.4). Otherwise, the next VNF index $n' \in \mathcal{M}^s$ is selected, and a new instance corresponding to this VNF index is deployed in the network. This process is repeated until the service deadline is satisfied. However, if new instances of all the required VNFs are attempted to be embedded, and the service deadline is still not met, NS s is rejected. After that, we move to schedule the next NS $s \in \mathcal{S}$ and schedule it until all NSs are scheduled.

Algorithm 6.3 Rescheduling

Input: s, Ω^s, δ^{best}, $\mathcal{K}$, $\mathcal{L}$, $\mathcal{I}_1$,..., $\mathcal{I}_{|\mathcal{F}|}$
Output: Ω^s

1: $\mathcal{M}^s \leftarrow$ **sort** all $n' \in \mathcal{N}^s \setminus \{0, \eta^s + 1\}$ according to the completion time of traffic scheduling of virtual links $(v^s_{n'-1}, v^s_{n'})$ and $(v^s_{n'}, v^s_{n'+1})$, and VNF index n' descendingly in Ω^s.
2: **for** each $n' \in \mathcal{M}^s$ **do**
3: $\Delta^{old} \leftarrow$ **compute** the completion time of traffic scheduling of virtual links $(v^s_{n'-1}, v^s_{n'})$ and $(v^s_{n'}, v^s_{n'+1})$, and VNF index n' in Ω^s
4: $\rho^{n'} \leftarrow$ **find** the shortest path from nodes hosting VNF instances on which VNF indices $n'-1$ and $n'+1$ are currently scheduled in Ω^s, based on the sum of communication and waiting times
5: $\mathcal{J}^{v^s_{n'}} \leftarrow$ **find** all already deployed instances of VNF type $v^s_{n'}$ on nodes along path $\rho^{n'}$
6: **if** $|\mathcal{J}^{v^s_{n'}}| \neq \emptyset$ **then**
7: $\Delta^{new} \leftarrow \infty$
8: **for** each $i \in \mathcal{J}^{v^s_{n'}}$ **do**
9: $\Delta \leftarrow$ **compute** the completion time if virtual links $(v^s_{n'-1}, v^s_{n'})$ and $(v^s_{n'}, v^s_{n'+1})$, and VNF index n' are rescheduled on path $\rho^{n'}$ and instance i, respectively.
10: **if** $\Delta < \Delta^{new}$ **then**
11: $\Delta^{new} \leftarrow \Delta$ and $i^{best} \leftarrow i$
12: **end if**
13: **end for**
14: **if** $\Delta^{new} < \Delta^{old}$ **then**
15: **reschedule** VNF index n' on i^{new} through the shortest path in terms of the completion time.
16: **reschedule** all VNF indices n, where $n > n'$ by executing lines 9-18 in **Algorithm 6.1** if n is not selected by the rescheduling function yet. Otherwise, n is scheduled, based on i^{best} obtained from the previous iteration, when it was selected.
17: **update** scheduling of all the VNF indices and the virtual links in Ω^s
18: $\delta^{best} \leftarrow$ **compute** the completion time of NS s
19: **if** $\delta^{best} \leq \delta^s$ **then**
20: **accept** s
21: **stop Algorithm 6.3**
22: **end if**
23: **end if**
24: **end if**
25: **end for**
26: **if** $\delta^{best} > \delta^s$ **then**
27: NewEmbedding$(s, \Omega^s, \mathcal{K}, \mathcal{L}, \mathcal{I}_1, ..., \mathcal{I}_{|\mathcal{F}|})$
28: **end if**

Algorithm 6.4 NewEmbedding

Input: s, Ω^s, δ^{best}, $\mathcal{K}$, $\mathcal{L}$, $\mathcal{I}_1$,..., $\mathcal{I}_{|\mathcal{F}|}$
Output: Ω^s

1: $\mathcal{M}^s \leftarrow$ **sort** all $n' \in \mathcal{N}^s \setminus \{0, \eta^s + 1\}$ according to the total waiting time of traffic scheduling of virtual links $(v^s_{n'-1}, v^s_{n'})$ and $(v^s_{n'}, v^s_{n'+1})$, and VNF index n' descendingly in Ω^s.
2: **for** each $n' \in \mathcal{M}^s$ **do**
3: $\rho^{n'} \leftarrow$ **find** the shortest path from nodes k hosting VNF instances on which VNF indices $n'-1$ and $n'+1$ are currently scheduled in Ω^s, based on the sum of the communication and waiting times
4: $k' \leftarrow$ **find** node k in path ρ^n with the highest available resources
5: **if** $u_{v^s_{n'}} \leq r_{k'}$ **then**
6: **place** new VNF instance i^{new} of type $v^s_{n'}$ on node k'
7: **reschedule** VNF index n' on i^{new} through the shortest path in terms of the completion time.
8: **reschedule** all VNF indices n, where $n > n'$ by executing lines 9-23 in **Algorithm 6.1** except lines 26-28 in **Algorithm 6.3** if n is not selected by the NewPlacement function yet. Otherwise, n is scheduled, based on i^{new} obtained from the previous iteration, when it was selected.
9: **update** scheduling of all the VNF indices and the virtual links in Ω^s.
10: $\delta^{best} \leftarrow$ **compute** the completion time of NS s
11: **if** $\delta^{best} \leq \delta^s$ **then**
12: **accept** s
13: **stop Algorithm 6.4**
14: **end if**
15: **end if**
16: **end for**
17: **reject** s

6.6 Tabu Search-Based Joint Embedding and Scheduling Algorithm

Although the greedy-based algorithm can provide a feasible solution very quickly due to its low complexity, its solution can sometimes be of low quality, as the greedy-based algorithm does not have any mechanism to escape from local optimum. Therefore, to further improve the solution quality, we also introduce a Tabu search-based joint VNF embedding and scheduling algorithm (TJESA), as explained next.

Tabu search [144] is a metaheuristic search method widely used to solve combinatorial optimization problems. It has also proven to be an efficient method to tackle the scheduling problem [16][53]. Unlike traditional local search methods (e.g., hill-climbing), the key idea of Tabu search is to use memory structure as a means to monitor search history in order to avoid local

search processes from cycling back to the same solution over and over again, and from being stuck in local optimum.

Algorithm 6.5 outlines the pseudo-code of our proposed TJESA. It starts by generating an initial solution Φ^0 and then sets Φ^0 as the current solution Φ^{cur} and the best-known solution Φ^{best}. Note that Φ denotes the embedding and scheduling solution of all the NSs in $\mathcal{S}$. Next, it performs a search process by iteratively generating a list of neighbor solutions of Φ^{cur}, denoted by $\mathcal{H}$ and then making a move to transit from Φ^{cur} to an improved solution among those neighbor solutions. If there is no non-improving solution, the best neighbor solution in the list can still be accepted to help the search process move from a local optimum. In addition, in every iteration, a move corresponding to the accepted solution is marked as Tabu and stored in a memory structure, called Tabu list. This will prevent a solution, whose move is marked as Tabu, from being accepted for a specific number of iterations to keep the search process away from cycling back to the previously accepted solutions again and getting stuck in the local optimum. However, if a solution, whose move is Tabu, is better than Φ^{best}, it can still be accepted. This condition is called aspiration criteria. Subsequently, in each iteration, the accepted solution becomes Φ^{cur} for the next iteration, and Φ^{best} is updated. We denote a solution that is accepted in each iteration by h^{best}. Finally, the search process is repeated until Φ^{best} does not improve for a given number j_{stop} of consecutive iterations. In the following subsections, the main components of our proposed TJESA are described in detail.

Algorithm 6.5 Tabu Search-based Joint Embedding and Scheduling Algorithm

Input: $\mathcal{S}$, $\mathcal{K}$, $\mathcal{L}$, $\mathcal{I}_1$,..., $\mathcal{I}_{|\mathcal{F}|}$
Output: Φ

1: $\Phi^0 \leftarrow$ **generate** an initial solution by executing **Algorithm 6.1** except the NewEmbedding function
2: $\Phi^{cur} \leftarrow \Phi^0$, $\Phi^{best} \leftarrow \Phi^0$, $j \leftarrow 0$
3: **while** $j < j_{stop}$ **do**
4: $\quad\mathcal{H} \leftarrow$ **create** candidate neighbor solutions of Φ^{cur}
5: $\quad h^{best} \leftarrow$ **select** from $\mathcal{H}$ a solution with the highest profits
6: $\quad$**if** h^{best}'s move is Tabu **then**
7: $\quad\quad$**if** $\gamma(h^{best}) \leq \gamma(\Phi^{best})$ **then**
8: $\quad\quad\quad h^{best} \leftarrow$ **select** a non-Tabu solution with the highest profits from $\mathcal{H}$
9: $\quad\quad$**end if**
10: $\quad$**end if**
11: $\quad$**add** h^{best}'s move to a Tabu list
12: $\quad\Phi^{cur} \leftarrow h^{best}$
13: $\quad$**if** $\gamma(\Phi^{cur}) > \gamma(\Phi^{best})$ **then**
14: $\quad\quad\Phi^{best} \leftarrow \Phi^{cur}$
15: $\quad\quad j \leftarrow 0$
16: $\quad$**end if**
17: $\quad j \leftarrow j + 1$
18: **end while**

6.6.1 Initial Solution

An initial solution Φ^0 is generated based on the greedy-based heuristic algorithm that aims to minimize the completion time without considering new VNF embedding. This can be carried out using Algorithm 6.1 without considering the NewEmbedding function (see line 1, Algorithm 6.5).

6.6.2 Neighborhood Structure

In this algorithm, two moves are defined to generate neighbor solutions for each iteration (line 4, Algorithm 6.5). The first one is the NS order-swapping move that aims to swap the scheduling order of NSs in $\mathcal{S}$, as their scheduling order can highly influence their completion times. The second one is the new VNF embedding move designed to embed new instances of VNFs required by a particular NS, whose service deadline cannot be satisfied by relying only on the already deployed VNF instances. The procedure for generating the neighbor solutions based on these two moves is explained below.

- **NS Order-Swapping:** Given the set of NSs $\mathcal{S}$ of the current solution Φ^{cur}, two NSs in $\mathcal{S}$ are first randomly selected. The order of these two selected NSs are then swapped in $\mathcal{S}$. Finally, two neighbor solutions are generated for this move. The first and second solutions are generated by rescheduling all NSs by using Algorithm 6.1 without considering the NewEmbedding function, and without considering both Rescheduling and NewEmbedding functions, respectively. The rationale behind these two rescheduling strategies is to make sure that the search process does not miss good neighbor solutions.
- **New VNF Embedding:** Given the current solution Φ^{cur}, NS $s \in \mathcal{S}$ whose deadline is not satisfied is first randomly selected. Next, VNF index $n \in \{1,..,\eta^s\}$ of the selected NS s, whose total waiting time of traffic scheduling of virtual links (v^s_{n-1}, v^s_n) and (v^s_n, v^s_{n+1}), and VNF index n is the longest, is selected. A node with the highest available resources in the shortest path from nodes hosting VNF instances on which VNF indices n - 1 and n + 1 are currently scheduled, is then selected to host a new instance of VNF type v^s_n. This shortest path is based on the sum of the communication and waiting times. Finally, similar to the first move, two neighbor solutions are also generated for this move. For the first solution, all NSs are rescheduled using Algorithm 6.1 without considering the NewEmbedding function, while for the second one, all NSs are rescheduled using Algorithm 6.1 without both considering the Rescheduling and NewEmbedding functions.

6.6.3 Tabu List

In each iteration, a move corresponding to the accepted neighbor solution is marked and stored in a Tabu list for a certain number j_{tabu} of iterations (also known as Tabu tenure) to avoid the search process from repeatedly visiting the same solution and to help it escape from the local optimum (line 11, Algorithm 6.5).

6.6.4 Neighborhood Evaluation and Selection

In each iteration, neighbor solutions are evaluated based on the objective function, as defined in Eq. (6.22). The neighbor solution h^{best} that has the best profits and whose move is not Tabu is always selected to become the new current solution Φ^{cur} for the next iteration. However, if the neighbor solution h^{best} that has the best profits and whose move is Tabu is better than the best-known solution Φ^{best}, it can still be selected, as the aspiration criteria are met (lines 5-10, Algorithm 6.5).

6.6.5 Termination Criteria

Our proposed TJESA will terminate if the best-known solution Φ^{best} does not improve for a given number j_{stop} of consecutive iterations (lines 13-17, Algorithm 6.5). We note that j_{stop} can be adjusted to ensure a suitable trade-off between the execution time and solution quality.

6.7 Performance Evaluation

In this subsection, we evaluate the performance of our two proposed algorithms, GJESA and TJESA, in solving the joint VNF embedding and scheduling problem. We compare their solutions with those obtained by solving the ILP model developed in Section 6.4 using CPLEX. To the best of our knowledge, none of the algorithms available in the literature considers the same joint VNF embedding and scheduling problem as in this chapter. Therefore, to demonstrate the effectiveness of our proposed algorithms in relation to the state-of-the-art, we developed three benchmarks, which solve the VNF embedding and scheduling problems separately. We describe them as follows:

- The first benchmark EE+SS adopts the existing embedding algorithm proposed in [40] to solve the VNF embedding problem. This embedding algorithm aims at reusing already deployed instances as many as possible before deploying new instances if service deadlines

cannot be met without considering waiting times. The EE+SS algorithm then schedules NSs on the assigned instances and links obtained from executing the embedding algorithm as soon as they are available.

- The second benchmark RE+ES relies on the Tabu search-based scheduling algorithm introduced in [16] without considering any new VNF embedding to schedule NSs. For fair comparison, it generates new instances with different VNF types uniformly and randomly deploy them in the network until all nodes are fully utilized before scheduling the NSs.
- The third benchmark EE+ES utilizes the embedding algorithm proposed in [40] to solve the VNF embedding problem. It then adopts the Tabu search-based scheduling algorithm introduced in [16] to solve the scheduling problem.

We implemented the GJESA, TJESA, EE+SS, RE+ES and EE+ES in JAVA and used CPLEX to implement and solve the ILP model. Furthermore, all the simulations were conducted on a machine with 2.40 GHz Intel(R) Xeon(R) CPU and 16 GB of RAM. As the TJESA, RE+ES and EE+ES involve some sort of randomness, the simulations for these algorithms in each scenario were also conducted 5 times, and the average results were taken. The results of the TJESA, RE+ES and EE+ES are also presented with 95% confidence interval.

6.7.1 Simulation Setup

In the simulations, we consider three network topologies with different sizes including a small-scale network (7 nodes and 12 links), NOBEL-GERMANY (17 nodes and 26 links) [148] and ITC Deltacom (113 nodes and 184 links) taken from the Internet Zoo Topology [154]. We also assume that a number of VNF instances were already deployed and randomly embedded into the network, and the number of VNF types is chosen between 5 and 8 [15][16], depending on the scenario. Each node has an available capacity selected randomly between 0 and 2 units, which can be used to host new VNF instances. The link capacities of all links are assumed to be 400 traffic per time unit, and each link has a propagation delay selected between 0 and 2 time slots. In addition, the resource requirement and additional cost for instantiating new instances of all the VNF types on any node in the network are 1 resource unit and 10 cost units per resource unit, respectively. The processing capacity of each VNF type is randomly selected between 200 and 300 traffic per time unit. Furthermore, we generate multiple NSs, whose traffic size is randomly chosen among 400 and 600 traffic units, and the bandwidth requirement is randomly selected between 200 and 300 traffic per

time unit. Each NS requires 2 to 4 VNF types, assigned randomly. Similar to the work in [15] and [16], the deadline requirement of each NS in all scenarios is set to be 4/3 of the sum of its processing and communication delays, without accounting for any waiting delays unless stated otherwise. Note that the communication delay of each NS is computed based on the shortest path between its source and its destination.

In the ILP model, we set the maximum number of instances of VNF types allowed to be deployed in the network to 5. For the TJESA, RE+ES and EE+ES, we adopted the same Tabu tenure and number of termination iterations, set to be 7 and 100, respectively. The Tabu tenure was selected, as it provides the best solution quality based on our experiments, while the number of termination iterations was chosen because it allows a suitable trade-off between solution quality and execution time.

Similar to the work in [15] and [16], we consider both offline and online scenarios to evaluate the performance of our proposed algorithms. In the offline scenario, all NSs are known in advance. We also assume that in this scenario, all NSs arrive at the same time at t_0 [15][16]. On the other hand, in the online scenario, NSs can arrive randomly at any time. To evaluate the performance of our proposed algorithms in this scenario, we rely on an online batch scheduling approach. The performance of our proposed algorithms in both online and offline scenarios are discussed in more detail next.

6.7.2 Offline Scenario

In the offline scenario, we consider the small-scale network (7 nodes and 12 links), and assume that 10 instances with 5 different VNF types were already deployed in the network. We also assume that the reward weight ω is set to be 100, which is large enough to allow the algorithms to prioritize maximizing the total revenue over minimizing the total cost in order to maximize the total profit. Figure 6.2 shows the results of different algorithms vs. number of NSs along with the optimal results obtained by solving the ILP model. As shown in Figure 6.2a, our proposed TJESA and GJESA outperform the EE+SS, RE+ES, and EE+ES algorithms in terms of total profit in all cases. More precisely, while the average gaps of our proposed TJESA and GJESA remain smaller than 14% and 28%, respectively, those of the EE+SS, RE+ES, and EE+ES algorithms can be as high as 52%, 80%, and 51%, respectively, in comparison to the optimal solution. This highlights the advantage of our proposed joint embedding and scheduling approach over the approaches that

solve the VNF embedding and scheduling problems separately. We also observe that the TJESA performs better than the GJESA because it offers more flexibility compared to the GJESA in exploring different scheduling orders as well as VNF embedding in order to find the best joint embedding and scheduling solutions through an efficient search mechanism. In addition, we observe that the RE+ES algorithm has the worst performance in this scenario. This is due to the fact that the RE+ES relies on random VNF embedding without considering service deadlines of NSs. This, as a result, may lead to large communication times, which may not allow the service deadline to be met. By contrast, our proposed TJESA and GJESA as well as the EE+SS and EE+ES algorithms take into consideration the service deadline when embedding new VNF instances into the network. Furthermore, we observe that the EE+ES algorithm is superior to the EE+SS algorithm in most of the case. This is because contrary to the EE+SS algorithm, the EP+ES algorithm relies on the search process-based scheduling algorithm, which allows different NS scheduling orders to be explored to find the best scheduling solution. Our results show that the EE+ES algorithm is inferior to our proposed TJESA and GJESA, as it does not consider waiting times when deploying new VNF instances. For this reason, service deadlines may not be met, especially when NSs have to be scheduled on shared VNF instances.

Next, we evaluate the performance of various algorithms under consideration in terms of admission rate. Figure 6.2b depicts the admission rate vs. the number of NSs for the offline scenario. As shown in this figure, the admission rate of our proposed GJESA and TJESA is higher than those of the EE+SS, RE+ES, and EE+ES algorithms and close to the optimal solution in all cases. This shows that our proposed algorithms allow VNF instances to be shared more efficiently in order to meet the service deadlines than the EE+SS, RE+ES and EE+ES for large number of NSs. In addition, the node utilization results are plotted in Figure 6.2c to show how node resources are being utilized while the number of NSs increases. Here, the node utilization is computed as the ratio of the total node resources used for hosting new instances to the total available node resources before the new instances are deployed. As shown in Figure 6.2c, the node utilization of all the algorithms, except for the RE+ES, tends to increase, as the number of NSs increases. This is because as the number of NSs increases, the already deployed VNF instances are shared by a larger number of NSs. This in turn leads to longer waiting times, which does not allow the deadlines of some NSs to be satisfied. Therefore, new instances need to be embedded to ensure that the service deadlines are satisfied, thereby increasing the node utilization. It should be note that although the

node utilization of our proposed GJESA and TJEPA is higher than those of the EE+SS and EE+ES algorithms, both the GJESA and TJEPA achieve higher admission rate. As mentioned earlier, since the RE+ES algorithm randomly embeds new VNF instances into the network until all nodes are fully utilized before scheduling NSs, its node utilization is always equal to one.

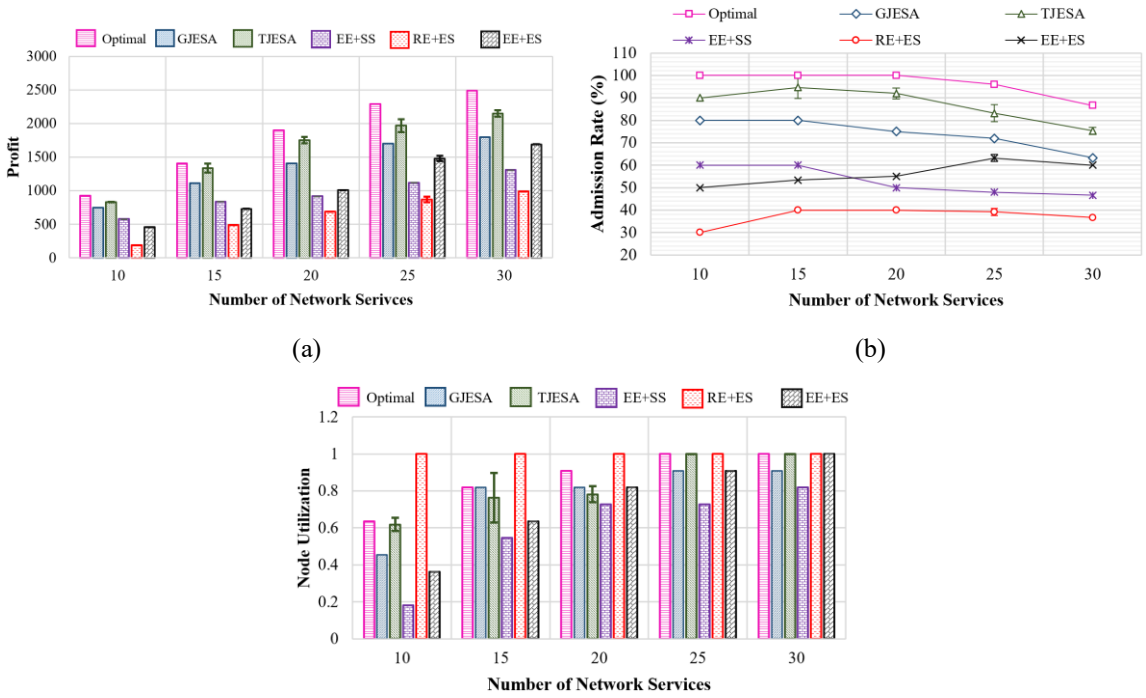


Figure 6.2: (a) Profit, (b) admission rate and (c) node utilization vs. NSs (offline scenario).

The profit, admission rate, and node utilization for 30 NSs for varying deadlines are illustrated in Figure 6.3. In the x-axis of this figure, the values 4/3, 5/3, 6/3 and 7/3 indicate that the service deadlines of the NSs are set to be 4/3, 5/3, 6/3 and 7/3 of the sum of their processing and communication times, respectively, without considering any waiting time. As shown in Figure 6.3a and 6.3b, the profit and admission rate of all the algorithms increase when the service deadlines become less stringent. This occurs because the NSs can tolerate longer waiting times, thus allowing for a larger number of NSs to be scheduled on the already deployed instances. Our proposed GJESA and TJESA outperform the EE+SS, RE+ES and EE+ES for any given service deadline. In addition, we notice that the profit and admission rate of the RE-ES grow significantly as the service deadlines increase. This is due to the fact that the NSs can tolerate longer communication times, which allows a number of NSs to be scheduled on the VNF instances which

were randomly deployed in advances. However, the RE+ES always fully utilizes node resources, as shown in Figure 6.3c.

Table 6.2 illustrates the average execution times of all the algorithms considered, along with the time required by solving the ILP model using CPLEX. It is clear from Table 6.2 that the average execution times of our proposed GJESA and TJESA are much smaller than those of the ILP model. For instance, when the number of NSs is 30, the execution time for solving the ILP model can be up to 5 hours, while the execution times for GJESA and TJESA are close to only 1 and 14.6 seconds, respectively. This implies that the GJESA and TJESA algorithms are much more scalable than the ILP model. Furthermore, even though the average optimality gap of the GJESA is higher than that of the TJESA by 14% as illustrated in Figure 6.2, the GJESA is much more scalable than the TJESA.

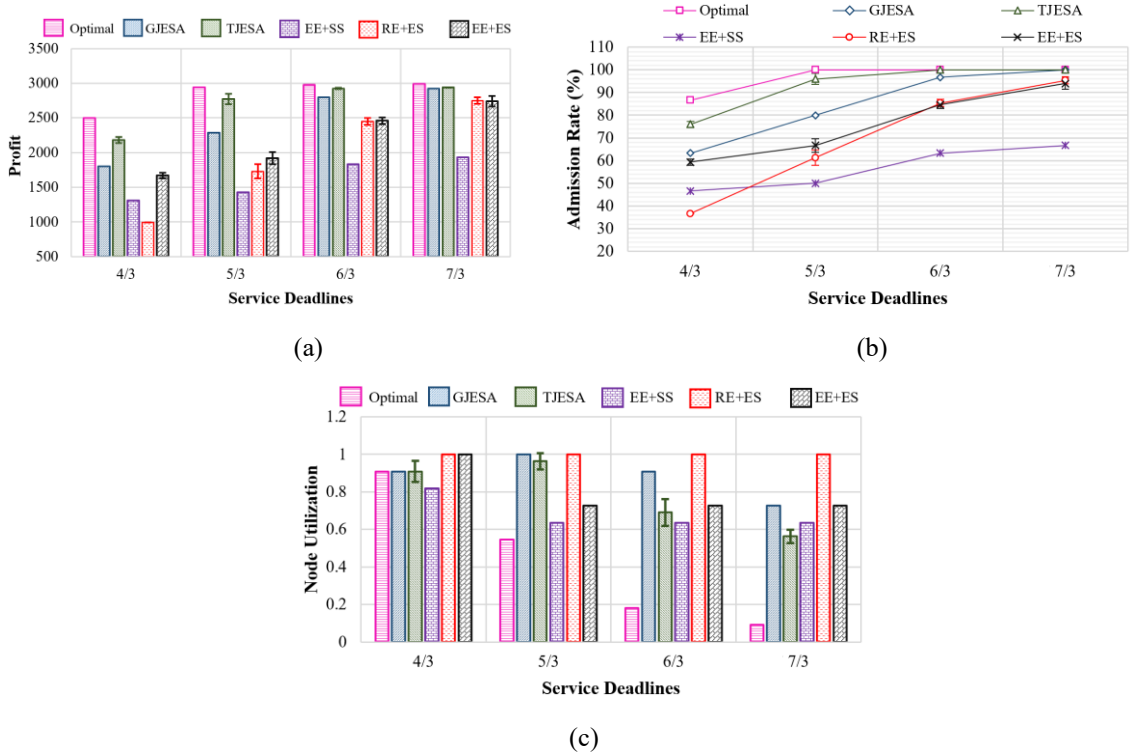


Figure 6.3: (a) Profit, (b) admission rate and (c) node utilization vs. deadlines for 30 NSs (offline scenario).

Table 6.2: Average execution time (second).

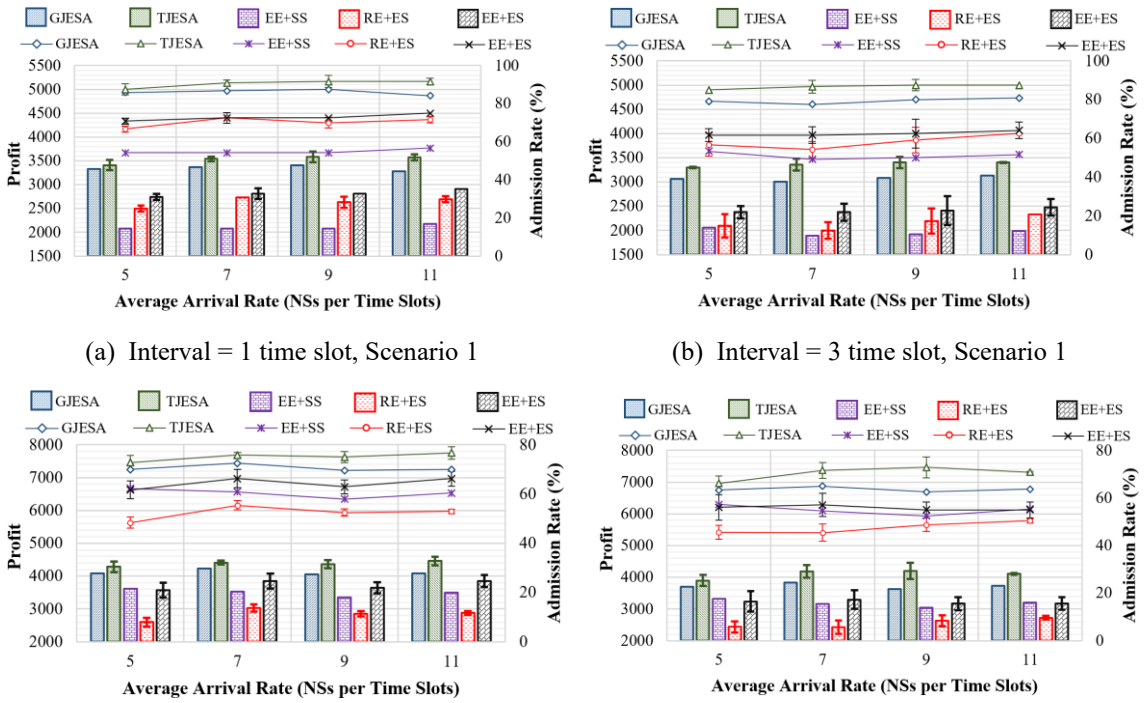
NSs	ILP Model	GJESA	TJESA	EE+SS	RE+ES	EE+ES
5	72	<1	<1	<1	1	1
10	138	<1	3.4	<1	3.2	2.8
15	219	<1	4.8	<1	9.4	9
20	317	<1	6.8	<1	15.4	12.2
25	1123	<1	10.4	<1	17.4	17.6
30	17100	<1	14.6	<1	19.6	25.4

6.7.3 Online Scenario

To conduct a more realistic evaluation, we evaluate the performance of our proposed algorithms in an online batch scheduling mode, where NSs randomly arrive in the network. In this mode, the algorithms are executed periodically with a batch of NSs that arrive within a fixed interval. In addition, we assume that NSs randomly arrive at each time slot, following a Poisson process [15][16] with an average arrival rate λ (measured in the number of NSs per time slot).

First, we investigate the performance of our proposed algorithms with respect to varying average arrival rate λ. We consider two network scenarios. Scenario 1 consists of 17 nodes, 34 already deployed instances with 8 different VNF types, and 40 NSs. Scenario 2 includes 113 Nodes, 60 already deployed instances with 8 different VNF types, and 60 NSs. Figure 6.4 shows the profit and admission rates of all the algorithms for these two network scenarios with fixed intervals of 1 and 3 time slots vs. the average arrival rate λ. Note that when the interval is set to 1 time slot, the algorithms are executed in every time slot. As shown in Figure 6.4, our proposed GJESA and TJESA perform better than the EE+SS, RE+ES and EE+ES algorithms for all the scenarios described above. This shows the effectiveness of our proposed algorithms over the existing algorithms in online scenarios and indicates a disadvantage of the VNF embedding without considering waiting times, which is always the case for approaches that aim to solve the VNF embedding and scheduling problems separately. Similar to the offline scenario, the RE+ES algorithm has the worst performance in most of the cases, as it relies on random VNF embedding. Furthermore, we notice that for a given average arrival rate λ, increasing the time interval leads to lower profit and admission rate, because as the time interval increases, the waiting time increases as well. For instance, if the time interval is equal to 3 time slots, NSs that arrive at the first time slot need to wait for 2 time slots before they can be scheduled. For this reason, their deadlines may not be met.

Finally, we investigate the performance of our proposed algorithms for varying total available node resources. In this case, scenario 1 is considered. We assume that the NSs randomly arrive at each time slot with λ = 3 NSs per time slot, and the interval is set to 1 time slot. Figures 6.5a and 6.5b illustrate the admission rate and node utilization vs. the total available node resources, respectively. The results shown in Figure 6.5a indicate that the admission rate of all the algorithms increases when the total available node resources increase. This was expected because increasing the total available node resources offers more opportunities to deploy new VNF instances to reduce the waiting times of NSs for VNF processing and transmission as well as to shorten the communication times. The results also show that our proposed GJESA and TJESA can achieve a higher admission rate compared to the EE+SS, RE+ES, and EE+ES algorithms for all the cases. This means that our proposed algorithms can use node resources more efficiently to satisfy the service deadlines of the NSs. It is important to note that although TJESA is superior to GJESA in terms of both admission rates and node utilization in most cases, this comes at the expense of a higher execution time as shown in Table 6.2.



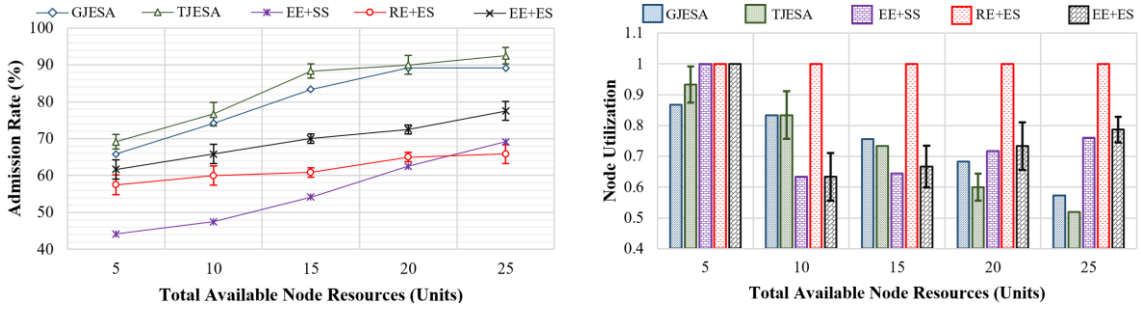
(a) Interval = 1 time slot, Scenario 1

(b) Interval = 3 time slot, Scenario 1

(c) Interval = 1 time slot, Scenario 2

(d) Interval = 3 time slot, Scenario 2

Figure 6.4: Profit and admission rate (AR) vs. average arrival rate λ for all the algorithms under consideration with two network scenarios and intervals = 1 and 3 time slots (online scenario).



(a) Interval = 1 time slot, Scenario 1

(b) Interval = 3 time slot, Scenario 1

Figure 6.5: (a) Admission rate and (b) node utilization vs. total available node resources for scenario 1 with λ = 3 NSs per time slot and interval = 1 time slot (online scenario).

6.8 Conclusion

This chapter studied the joint VNF embedding and scheduling problem, which are two subproblems of the NFV-RA problem for latency-sensitive NSs, as they have a significant impact on service deadlines. We aim at efficiently determining whether to deploy new VNF instances or reuse already deployed VNF instances for NS scheduling to maximize profits while guaranteeing the stringent service deadlines. We first formulated our problem as an ILP. To address the scalability issue of our developed ILP problem, we then proposed two heuristics, namely, greedy-based and Tabu search-based algorithms to solve the problem. We also conducted extensive simulations in both offline and online scenarios to show the effectiveness of our proposed algorithms versus the existing algorithms. The simulation results indicate that our proposed algorithms can achieve small gaps with respect to the optimal solution in an offline scenario. We also showed that our proposed algorithms achieve a higher profit than the existing algorithms in both offline and online scenarios. Furthermore, while the Tabu search-based algorithm can provide the best solution quality, the greedy-based algorithm can achieve the shortest execution time, making the latter more scalable with large-size network scenarios.

Chapter 7

7. Conclusion, Discussion and Future Work

7.1 Conclusion

Although the Tactile Internet will bring a new set of applications and services that could potentially revolutionize how humans live and communicate, it is still in its early stage because of its stringent performance requirements (i.e., ultra-low latency and ultra-high reliability). Due to some limitations of the currently deployed networks, a softwarized network enabled NFV and SDN technologies is introduced as a key enabler for realizing the Tactile Internet. Even though the softwarized network brings several benefits in terms of flexibility and scalability, provisioning Tactile Internet NSs in the softwarized network remains several challenges, especially for ultra-low latency NSs. This is because these NSs have the stringent latency requirements. Moreover, some of them require fast service provisioning and have dynamic service arrival/departure. Therefore, this thesis aims at addressing these challenges. More specifically, we first provided a survey on the solutions proposed so far for the Tactile Internet. After that, we introduced efficient algorithms to address the three challenges on provisioning ultra-low latency NSs in the softwarized network for the Tactile Internet.

In Chapter 3, we surveyed both architectural and algorithmic solutions in the literature for the Tactile Internet. We classified them into four main categories, (i) architectures, protocols, and intelligent prediction, (ii) radio resource allocation algorithms, (iii) non-radio resource allocation

algorithms on lower layers and (iv) non-radio resource allocation beyond lower layers. In each category, we provided a critical review and derived some potential insights and lessons learned. We also discussed some promising research directions for the Tactile Internet.

In order to meet very stringent latency requirements, in some cases, multiple instances of each VNF type with proper VNF embedding and traffic routing are needed to serve ultra-low latency NSs. However, relying on multiple instances of each VNF type to serve these NSs could lead to reliability degradation. To address this challenge, we introduced a joint VNF composition and embedding algorithm in Chapter 4. More precisely, we formulated the problem as a MILP, and proposed a Tabu search-based algorithm to solve the problem. Our proposed algorithm aims at optimally determining the number of VNF instances, their embedding locations and traffic routing to guarantee the stringent latency requirement while optimizing reliability degradation and cost. The simulation results indicate that it can significantly improve cost and reliability while satisfying stringent latency requirements of ultra-low latency NSs in comparison to the existing algorithms.

Chapter 5 tackled real-time VNF embedding to provide fast service provisioning for ultra-low latency NSs. Unlike in Chapter 4, we here assume that a VNF-FG is given. We proposed a h-horizon sequential look-ahead greedy algorithm, which aims at finding a low-cost embedding solution while guaranteeing stringent latency requirements with timely service deployment times. Our proposed algorithm relies on a sequential heuristic approach to ensure acceptable execution times. It also offers efficient embedding and re-embedding strategies to address the causality issue, which are generally suffered by existing sequential heuristic algorithms to improve the solution quality. The causality issue generally occurs when any given VNF is embedded without considering embedding contexts of all other VNFs in a VNF-FG, which could lead to poor embedding solutions. The simulation results show that our proposed algorithm achieves lower cost than the existing sequential heuristic algorithms while ensuring latency requirements and timely service deployment times.

Finally, we studied a joint VNF embedding and scheduling problem for ultra-low latency NSs in Chapter 6. In this problem, we considered that these services require to be served within specific service deadlines before they depart from the network, and these service deadlines can be stringent. We aim at optimally determining whether to schedule NSs on already deployed or newly deployed VNF instances to guarantee the stringent service deadlines while maximizing profits. We formulated the problem as an ILP and proposed two efficient algorithms, namely greed-based

algorithm and Tabu search-based algorithm to solve the problem. The simulation results show that our proposed algorithms outperform the existing algorithms that solve the VNF embedding and scheduling problems separately in terms of both profits and admission rates.

7.2 Limitations of the Proposed Algorithms

As demonstrated in the performance evaluation sections, the proposed algorithms in this thesis bring several potential benefits in provisioning ultra-low latency NSs in NFV-based softwarized networks. However, there are still some limitations. In this subsection, we describe those limitations of the proposed algorithms.

The joint VNF composition and embedding algorithm introduced in Chapter 4 provides an efficient method to guarantee the stringent latency requirements when provisioning ultra-low latency NSs. Specifically, it allows traffic of the NSs to be processed by multiple instances of each VNF type with efficient traffic splitting and routing to accelerate the processing delay to ensure that the stringent latency requirements are satisfied. However, one limitation of the proposed algorithm is its execution times. The proposed algorithm is based on a Tabu search method that relies on randomized and advanced search mechanisms to find a high-quality solution. This sometimes leads to slow solution convergence, especially for large-scale networks due to the complexity of the problem. Therefore, it might not be suitable for ultra-low latency NSs that require fast service provisioning (e.g., teleoperation for wildfire suppression).

When it comes to the real-time VNF embedding algorithm proposed in Chapter 5, one limitation is that the proposed algorithm might not be able to meet the latency requirement when the latency requirement becomes very stringent. This is due to the fact that the proposed algorithm relies on a single instance of each VNF type in a VNF-FG to process traffic of a NS. Therefore, it does not offer an efficient mechanism to accelerate the processing delay in order to meet the stringent latency requirement. Similarly, the joint VNF embedding and scheduling algorithm proposed in Chapter 6 also relies on a single instance of each VNF type in a VNF-FG to serve a NS. As a result, the proposed algorithm might not be able to meet the very stringent service deadlines. Furthermore, the joint embedding and scheduling algorithm only works with a NS that has a simple linear topology of a VNF-FG.

It is important to note that all the proposed algorithms in this thesis also neglect some service requirements to make the NFV-RA problem easier to solve. Examples of these requirements are

VNF affinity, VNF anti-affinity and different resource requirements. These requirements make the problem more practical and more challenging to solve. Finally, the proposed algorithms mainly focus on the static NFV-RA problem. More precisely, they do not provide an efficient mechanism to cope with dynamic changes in user demands, network conditions and service requirements, which could happen in real-world scenarios, once NSs are deployed in the network.

7.3 Applicability of the Proposed Algorithms to Network Slicing

Network slicing has emerged as a key concept to support a wide range of applications and services in 5G networks [155]. It also continues to play a crucial role in the next generation 6G mobile networks [5]. With this concept, dedicated virtualized networks, called slices, can be created on a shared infrastructure to serve application domains and related services with their unique requirements. These slices are compartmentalized and independently managed virtual infrastructures, and the required resources are instantiated and allocated on demand.

NFV and SDN technologies are two key enablers for realizing network slicing [155][156]. The NFV enables network service chaining, network capacity and QoS-oriented VNF embedding, and VNF management and orchestration to create and manage network slices in a shared infrastructure. With NFV technology, a slice can be defined as a collection of end-to-end traffic flows that are processed through an ordered set of VNFs, also known as a collection of VNF chains [157]. In addition, the SDN offers simple network management to facilitate connectivity necessary to create and manage network slices. The network slice orchestration architecture considered by 5G Infrastructure Public Private Partnership (5GPPP) is illustrated in Figure 7.1 [156]. This architecture relies on a NFV architecture integrated with SDN to create and orchestrate network slices. The architecture consists of three main components. The first one is end-to-end service management and orchestration, which is responsible for admission control, policy provisioning and slice customization. It also creates network service graphs for network slice requests. The second one is virtual resource orchestration, which includes all components (e.g., VNF manager, NFV orchestrator, virtualized infrastructure manager (VIM)) in the management and orchestration plan except the end-to-end service management and orchestration. The virtual resource orchestration aims at embedding and instantiating VNFs of the virtual network service graph, and managing and orchestrating life cycles of VNF instances for network slices. The proposed algorithms in this thesis could reside within the NFV orchestration. They could provide efficient

mechanisms to embed and chain VNFs required by network slices, especially slices for ultra-low latency services to efficiently ensure the service requirements. The last component is network resource programmable controller, which offers flexible VNF service chaining, QoE control and resource programmability for network slices.

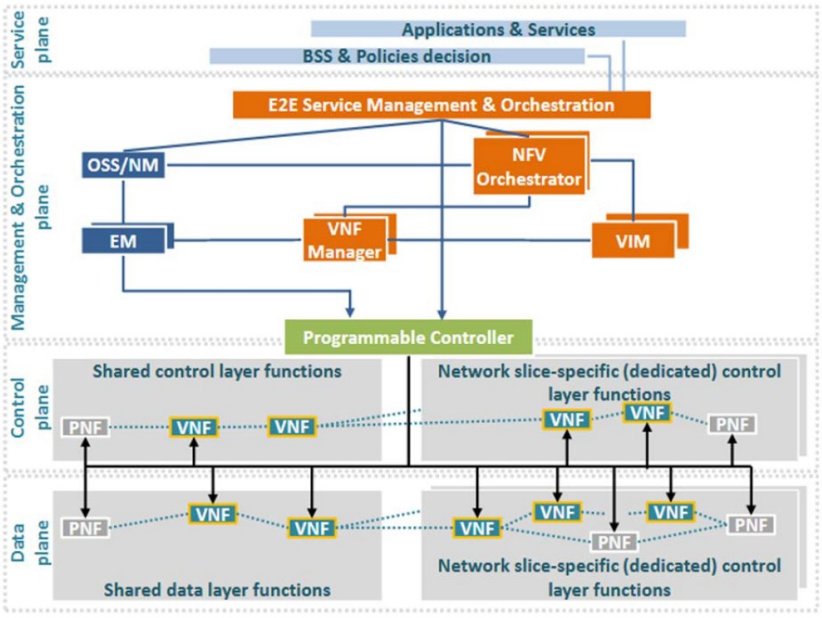


Figure 7.1: 5GPPP network orchestration and architecture [156].

7.4 Future Work

This thesis presented several contributions on ultra-low latency service provisioning in a softwarized network for the Tactile Internet. However, there still exist some future research directions in this area.

7.4.1 Additional Service Requirements

Although the main requirement of ultra-low latency NSs is latency, these services may have additional service requirements as well. One example is VNF affinity and anti-affinity in which VNFs required by a given NS must and must not be deployed in the same location, respectively to ensure that they do not degrade the service performance. Another example is different resource requirements, where multiple types of resources (e.g., CPUs, memory, storage) may be needed to run a given VNF required by a NS. These additional requirements make NFV-RA much more complex. Therefore, efficient mechanisms are needed to take these requirements into consideration

while guaranteeing stringent latency requirements of ultra-low latency NSs and maximizing profits of NOs.

7.4.2 Dynamic NFV-RA

In practice, service requirements and network conditions may change over time. Therefore, solving the NFV-RA problem in a static manner might sometimes not be an efficient way to guarantee the service requirements of ultra-low latency NSs as well as ensuring high profits of NOs. Moreover, once the NSs are deployed, resource reconfiguration may also be needed to ensure that the service requirement is still satisfied, and NO's objective is still optimized because of dynamic changes in the service requirements and network conditions. However, we still lack efficient mechanisms to address the dynamicity of the NFV-RA problem for ultra-low latency NSs.

7.4.3 Joint VNF Decomposition and Embedding

As discussed in [158], a VNF can be decomposed into multiple sub-functions and have multiple options of decompositions/implementations, where each of them is associated with different performance and costs. This brings flexibility to NOs in selecting a suitable decomposition option for each VNF to serve NSs to optimize their objectives (e.g., cost minimization) while satisfying the SLA requirements. Despite this benefit, the VNF decomposition also raises a challenge, especially when it comes to provisioning ultra-low latency NSs. Specifically, a decomposition option of each VNF needs to be selected while taking stringent latency requirements into account. This is not trivial, as overall latency performance of a NS cannot be obtained until decomposition options of all VNFs required by the NS are actually deployed into the network. Therefore, to take the benefit of the VNF decomposition when serving ultra-low latency NSs, there is a need for efficient mechanisms to jointly determine VNF decomposition selection and VNF embedding.

Bibliography

[1] N. Promwongsa *et al.*, "A Comprehensive Survey of the Tactile Internet: State-of-the-Art and Research Directions," *IEEE Commun. Surv. Tutor.*, vol. 23, no. 1, pp. 472–523, Firstquarter 2021.

[2] G. P. Fettweis, "The Tactile Internet: Applications and Challenges," *IEEE Veh. Technol. Mag.*, vol. 9, no. 1, pp. 64–70, Mar. 2014.

[3] K. S. Kim *et al.*, "Ultrareliable and Low-Latency Communication Techniques for Tactile Internet Services," *Proc IEEE*, vol. 107, no. 2, pp. 376–393, Feb. 2019.

[4] O. Holland *et al.*, "The IEEE 1918.1 'Tactile Internet' Standards Working Group and its Standards," *Proc IEEE*, vol. 107, no. 2, pp. 256–279, Feb. 2019.

[5] Z. Zhang *et al.*, "6G Wireless Networks: Vision, Requirements, Architecture, and Key Technologies," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 28–41, Sep. 2019.

[6] M. He, A. M. Alba, A. Basta, A. Blenk, and W. Kellerer, "Flexibility in Softwarized Networks: Classifications and Research Challenges," *IEEE Commun. Surv. Tutor.*, vol. 21, no. 3, pp. 2600–2636, thirdquarter 2019.

[7] S. D'Oro, S. Palazzo, and G. Schembra, "Orchestrating Softwarized Networks with a Marketplace Approach," *Procedia Comput. Sci.*, vol. 110, pp. 352–360, Jan. 2017.

[8] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges," *IEEE Commun. Surv. Tutor.*, vol. 18, no. 1, pp. 236–262, Firstquarter 2016.

[9] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proc IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[10] M. F. Zhani and H. ElBakoury, "FlexNGIA: A Flexible Internet Architecture for the Next-Generation Tactile Internet," *J. Netw. Syst. Manag.*, Mar. 2020.

[11] J. G. Herrera and J. F. Botero, "Resource Allocation in NFV: A Comprehensive Survey," *IEEE Trans. Netw. Serv. Manag.*, vol. 13, no. 3, pp. 518–532, Sep. 2016.

[12] F. C. Chua, J. Ward, Y. Zhang, P. Sharma, and B. A. Huberman, "Stringer: Balancing Latency and Resource Usage in Service Function Chain Provisioning," *IEEE Internet Comput.*, vol. 20, no. 6, pp. 22–31, Nov. 2016.

[13] K. Somayeh and R. Glith, "Cost-Efficient Server Provisioning for Deadline-Constrained VNFs Chains: A Parallel VNF Processing Approach," in *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Jan. 2019, pp. 1–6.

[14] "'Network Functions Virtualisation (NFV); Reliability; Report on Models and Features for End-to-End Reliability,' ETSI ISG NFV, Tech. Rep., 2016."

[15] H. Alameddine, M. H. K. Tushar, and C. Assi, "Scheduling of Low Latency Services in Softwarized Networks," *IEEE Trans. Cloud Comput.*, pp. 1–1, 2019.

[16] 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)*, Nov. 2017, pp. 1–9.

[17] N. Promwongsa *et al.*, "Ensuring Reliability and Low Cost When Using a Parallel VNF Processing Approach to Embed Delay-Constrained Slices," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 4, pp. 2226–2241, Dec. 2020.

[18] A. Ebrahimzadeh *et al.*, "h-Horizon Sequential Look-ahead Greedy Algorithm for VNF-FG Embedding," in *2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2021, pp. 41–46.
[19] N. Promwongsa *et al.*, "Look-ahead VNF-FG Embedding Framework for Latency-sensitive Network Services," *Submited ToIEEE Trans. Netw. Serv. Manag.*.
[20] N. Promwongsa, A. Ebrahimzadeh, R. H. Glitho, and N. Crespi, "Joint VNF Placement and Scheduling for Latency-sensitive Services," *IEEE Trans. Netw. Sci. Eng.*, vol. Early access, 2022.
[21] M. Simsek, A. Aijaz, M. Dohler, J. Sachs, and G. Fettweis, "5G-Enabled Tactile Internet," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 460–473, Mar. 2016.
[22] D. Van Den Berg *et al.*, "Challenges in Haptic Communications Over the Tactile Internet," *IEEE Access*, vol. 5, pp. 23502–23518, 2017.
[23] S. M. A. Oteafy and H. S. Hassanein, "Leveraging Tactile Internet Cognizance and Operation via IoT and Edge Technologies," *Proc IEEE*, vol. 107, no. 2, pp. 364–375, Feb. 2019.
[24] M. Dohler *et al.*, "Internet of Skills, Where Robotics Meets AI, 5G and the Tactile Internet," in *Proc. European Conference on Networks and Communications (EuCNC)*, Jun. 2017, pp. 1–5.
[25] E. Steinbach *et al.*, "Haptic Codecs for the Tactile Internet," *Proc. IEEE*, vol. 107, no. 2, pp. 447–470, 2019.
[26] W. Kellerer *et al.*, "How to Measure Network Flexibility? A Proposal for Evaluating Softwarized Networks," *IEEE Commun. Mag.*, vol. 56, no. 10, pp. 186–192, Oct. 2018.
[27] 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.
[28] 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)*, Oct. 2014, pp. 7–13.
[29] J. Gil-Herrera and J. F. Botero, "A scalable metaheuristic for service function chain composition," in *2017 IEEE 9th Latin-American Conference on Communications (LATINCOM)*, Nov. 2017, pp. 1–6.
[30] Y. Liu, Y. Lu, W. Qiao, and X. Chen, "A Dynamic Composition Mechanism of Security Service Chaining Oriented to SDN/NFV-Enabled Networks," *IEEE Access*, vol. 6, pp. 53918–53929, 2018.
[31] B. Zhang *et al.*, "A survey of VNF forwarding graph embedding in B5G/6G networks," *Wirel. Netw.*, Aug. 2021.
[32] D. B. Oljira, K. Grinnemo, J. Taheri, and A. Brunstrom, "A model for QoS-aware VNF placement and provisioning," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov. 2017, pp. 1–7.
[33] 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 *2017 13th International Conference on Network and Service Management (CNSM)*, Nov. 2017, pp. 1–7.
[34] S. Cai, F. Zhou, Z. Zhang, and A. Meddahi, "Disaster-Resilient Service Function Chain Embedding Based on Multi-Path Routing," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2021, pp. 1–7.

[35] 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)*, 2018, pp. 1–7.
[36] M. A. Khoshkholghi *et al.*, “Service Function Chain Placement for Joint Cost and Latency Optimization,” *Mob. Netw. Appl.*, vol. 25, no. 6, pp. 2191–2205, 2020.
[37] L. Magoula, S. Barmpounakis, I. Stavrakakis, and N. Alonistioti, “A genetic algorithm approach for service function chain placement in 5G and beyond, virtualized edge networks,” *Comput. Netw.*, vol. 195, p. 108157, Aug. 2021.
[38] R. Gouareb, V. Friderikos, and A. Aghvami, “Virtual Network Functions Routing and Placement for Edge Cloud Latency Minimization,” *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2346–2357, Oct. 2018.
[39] Y. T. Woldeyohannes, A. Mohammadkhan, K. K. Ramakrishnan, and Y. Jiang, “ClusPR: Balancing Multiple Objectives at Scale for NFV Resource Allocation,” *IEEE Trans. Netw. Serv. Manag.*, vol. 15, no. 4, pp. 1307–1321, Dec. 2018.
[40] P. Jin, X. Fei, Q. Zhang, F. Liu, and B. Li, “Latency-aware VNF Chain Deployment with Efficient Resource Reuse at Network Edge,” in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, Jul. 2020, pp. 267–276.
[41] A. Recse, R. Szabo, and B. Nemeth, “Elastic resource management and network slicing for IoT over edge clouds,” in *Proceedings of the 10th International Conference on the Internet of Things*, New York, NY, USA, Oct. 2020, pp. 1–8.
[42] Y. Yue, B. Cheng, X. Liu, M. Wang, B. Li, and J. Chen, “Resource Optimization and Delay Guarantee Virtual Network Function Placement for Mapping SFC Requests in Cloud Networks,” *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 2, pp. 1508–1523, Jun. 2021.
[43] Y. Chen and J. Wu, “Latency-Efficient VNF Deployment and Path Routing for Reliable Service Chain,” *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 1, pp. 651–661, Jan. 2021.
[44] J. Sun, F. Liu, H. Wang, M. Ahmed, Y. Li, and M. Liu, “Efficient VNF Placement for Poisson Arrived Traffic,” *IEEE Trans. Netw. Serv. Manag.*, pp. 1–1, 2021.
[45] N. H. Thanh, N. Trung Kien, N. V. Hoa, T. T. Huong, F. Wamser, and T. Hossfeld, “Energy-Aware Service Function Chain Embedding in Edge–Cloud Environments for IoT Applications,” *IEEE Internet Things J.*, vol. 8, no. 17, pp. 13465–13486, Sep. 2021.
[46] J. Pei, P. Hong, K. Xue, D. Li, D. S. L. Wei, and F. Wu, “Two-Phase Virtual Network Function Selection and Chaining Algorithm Based on Deep Learning in SDN/NFV-Enabled Networks,” *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1102–1117, Jun. 2020.
[47] J. Pei, P. Hong, M. Pan, J. Liu, and J. Zhou, “Optimal VNF Placement via Deep Reinforcement Learning in SDN/NFV-Enabled Networks,” *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 263–278, 2020.
[48] N. Toumi, M. Bagaa, and A. Ksentini, “Hierarchical Multi-Agent Deep Reinforcement Learning for SFC Placement on Multiple Domains,” in *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, 2021, pp. 299–304.
[49] S. M. A. Araújo, F. S. H. de Souza, and G. R. Mateus, “A hybrid optimization-Machine Learning approach for the VNF placement and chaining problem,” *Comput. Netw.*, vol. 199, p. 108474, Nov. 2021.
[50] J. Mei, X. Wang, and K. Zheng, “An intelligent self-sustained RAN slicing framework for diverse service provisioning in 5G-beyond and 6G networks,” *Intell. Converg. Netw.*, vol. 1, no. 3, pp. 281–294, 2020.

[51] H. A. Alameddine, S. Sebbah, and C. Assi, “On the Interplay Between Network Function Mapping and Scheduling in VNF-Based Networks: A Column Generation Approach,” *IEEE Trans. Netw. Serv. Manag.*, vol. 14, no. 4, pp. 860–874, Dec. 2017.
[52] J. F. Riera, E. Escalona, J. Batallé, E. Grasa, and J. A. García-Espín, “Virtual network function scheduling: Concept and challenges,” in *2014 International Conference on Smart Communications in Network Technologies (SaCoNeT)*, Jun. 2014, pp. 1–5.
[53] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and S. Davy, “Design and evaluation of algorithms for mapping and scheduling of virtual network functions,” in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, Apr. 2015, pp. 1–9.
[54] C. Assi, S. Ayoubi, N. El Khoury, and L. Qu, “Energy-Aware Mapping and Scheduling of Network Flows With Deadlines on VNFs,” *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 1, pp. 192–204, Mar. 2019.
[55] M. Wang, B. Cheng, B. Li, and J. Chen, “Service Function Chain Composition and Mapping in NFV-Enabled Networks,” in *2019 IEEE World Congress on Services (SERVICES)*, Jul. 2019, vol. 2642–939X, pp. 331–334.
[56] J. Li, W. Shi, Q. Ye, W. Zhuang, X. Shen, and X. Li, “Online Joint VNF Chain Composition and Embedding for 5G Networks,” in *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2018, pp. 1–6.
[57] Z. Wang, J. Zhang, T. Huang, and Y. Liu, “Service Function Chain Composition, Placement, and Assignment in Data Centers,” *IEEE Trans. Netw. Serv. Manag.*, vol. 16, no. 4, pp. 1638–1650, Dec. 2019.
[58] F. Carpio, S. Dhahri, and A. Jukan, “VNF placement with replication for Loac balancing in NFV networks,” in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.
[59] N. Gholipoor, H. Saeedi, N. Mokari, and E. A. Jorswieck, “E2E QoS Guarantee for the Tactile Internet via Joint NFV and Radio Resource Allocation,” *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 3, pp. 1788–1804, Sep. 2020.
[60] J. Li, W. Shi, P. Yang, and X. Shen, “On Dynamic Mapping and Scheduling of Service Function Chains in SDN/NFV-Enabled Networks,” in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.
[61] A. A. Ateya, A. Vybornova, R. Kirichek, and A. Koucheryavy, “Multilevel Cloud Based Tactile Internet System,” in *Proc. 19th International Conference on Advanced Communication Technology (ICACT)*, Feb. 2017, pp. 105–110.
[62] A. A. Ateya, A. Muthanna, I. Gudkova, A. Vybornova, and A. Koucheryavy, “Intelligent Core Network for Tactile Internet System,” in *Proc. International Conference on Future Networks and Distributed Systems*, New York, NY, USA, 2017, p. 22:1-22:6.
[63] A. S. Shafigh, S. Glisic, and B. Lorenzo, “Dynamic Network Slicing for Flexible Radio Access in Tactile Internet,” in *Proc. IEEE Global Communications Conference*, Dec. 2017, pp. 1–7.
[64] Z. Xiang, F. Gabriel, E. Urbano, G. T. Nguyen, M. Reisslein, and F. H. P. Fitzek, “Reducing Latency in Virtual Machines: Enabling Tactile Internet for Human-Machine Co-Working,” *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1098–1116, May 2019.
[65] H. Beyranvand, M. Lévesque, M. Maier, J. A. Salehi, C. Verikoukis, and D. Tipper, “Toward 5G: FiWi Enhanced LTE-A HetNets With Reliable Low-Latency Fiber Backhaul Sharing and WiFi Offloading,” *IEEEACM Trans. Netw.*, vol. 25, no. 2, pp. 690–707, Apr. 2017.

[66] J. Neaime and A. R. Dhaini, "Resource Management in Cloud and Tactile-capable Next-generation Optical Access Networks," *IEEEOSA J. Opt. Commun. Netw.*, vol. 10, no. 11, pp. 902–914, Nov. 2018.
[67] M. Maier, A. Ebrahimzadeh, and M. Chowdhury, "The Tactile Internet: Automation or Augmentation of the Human?," *IEEE Access*, vol. 6, pp. 41607–41618, 2018.
[68] O. Holland, S. Wong, V. Friderikos, Z. Qin, and Y. Gao, "Virtualized sub-GHz Transmission Paired with Mobile Access for the Tactile Internet," in *Proc. 23rd International Conference on Telecommunications (ICT)*, May 2016, pp. 1–5.
[69] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Toward Low-Latency and Ultra-Reliable Virtual Reality," *IEEE Netw.*, vol. 32, no. 2, pp. 78–84, Mar. 2018.
[70] Y. Jebbar, F. Belqasmi, R. Glitho, and O. Alfandi, "A Fog-Based Architecture for Remote Phobia Treatment," in *Proc. IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Dec. 2019, pp. 271–278.
[71] C. Grasso and G. Schembra, "Design of a UAV-Based Videosurveillance System with Tactile Internet Constraints in a 5G Ecosystem," in *Proc. 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, Jun. 2018, pp. 449–455.
[72] P.-V. Mekikis *et al.*, "NFV-Enabled Experimental Platform for 5G Tactile Internet Support in Industrial Environments," *IEEE Trans. Ind. Inform.*, vol. 16, no. 3, pp. 1895–1903, Mar. 2020.
[73] X. Wei, Q. Duan, and L. Zhou, "A QoE-Driven Tactile Internet Architecture for Smart City," *IEEE Netw.*, vol. 34, no. 1, pp. 130–136, Jan. 2020.
[74] E. Wong, M. P. I. Dias, and L. Ruan, "Predictive Resource Allocation for Tactile Internet Capable Passive Optical LANs," *J. Light. Technol.*, vol. 35, no. 13, pp. 2629–2641, Jul. 2017.
[75] P. J. Braun, S. Pandi, R. Schmoll, and F. H. P. Fitzek, "On the Study and Deployment of Mobile Edge Cloud for Tactile Internet Using a 5G Gaming Application," in *Proc. 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan. 2017, pp. 154–159.
[76] W. Du, Q. Liu, Z. Gao, and G. Tan, "Seamless Vertical Handoff Protocol for LTE-802.11p Hybrid Vehicular Communications Over the Tactile Internet," in *Proc. IEEE International Symposium on Haptic, Audio and Visual Environments and Games (HAVE)*, Sep. 2018, pp. 1–5.
[77] K. Antonakoglou, X. Xu, E. Steinbach, T. Mahmoodi, and M. Dohler, "Toward Haptic Communications Over the 5G Tactile Internet," *IEEE Commun. Surv. Tutor.*, vol. 20, no. 4, pp. 3034–3059, Fourthquarter 2018.
[78] K. Iiyoshi, M. Tauseef, R. Gebremedhin, V. Gokhale, and M. Eid, "Towards Standardization of Haptic Handshake for Tactile Internet: A WebRTC-Based Implementation," in *Proc. IEEE International Symposium on Haptic, Audio and Visual Environments and Games (HAVE)*, Oct. 2019, pp. 1–6.
[79] M. Maier and A. Ebrahimzadeh, "Towards Immersive Tactile Internet Experiences: Low-latency FiWi Enhanced Mobile Networks with Edge Intelligence [Invited]," *IEEEOSA J. Opt. Commun. Netw.*, vol. 11, no. 4, pp. B10–B25, Apr. 2019.
[80] F. Boabang, R. Glitho, H. Elbiaze, F. Belqasmi, and O. Alfandi, "A Framework for Predicting Haptic Feedback in Needle Insertion in Remote Robotic Surgery," presented at the Proc. 17th Annual IEEE Consumer Communications & Networking Conference (CCNC 2020), Las Vegas, USA, Jan. 2020.

[81] S. Mondal, L. Ruan, M. Maier, D. Larrabeiti, G. Das, and E. Wong, "Enabling Remote Human-to-Machine Applications with AI-Enhanced Servers over Access Networks," *IEEE Open J. Commun. Soc.*, pp. 1–1, 2020.
[82] S. Mondal, L. Ruan, and E. Wong, "Remote Human-to-Machine Distance Emulation through AI-Enhanced Servers for Tactile Internet Applications," in *Proc. Optical Fiber Communications Conference and Exhibition (OFC)*, Mar. 2020, pp. 1–3.
[83] E. Wong, S. Mondal, and L. Ruan, "Alleviating the Master-Slave Distance Limitation in H2M Communications through Remote Environment Emulation," in *Proc. International Conference on Optical Network Design and Modeling (ONDM)*, May 2020, pp. 1–3.
[84] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges," *IEEE Commun. Surv. Tutor.*, vol. 20, no. 1, pp. 416–464, Firstquarter 2018.
[85] C. She, C. Yang, and T. Q. S. Quek, "Uplink Transmission Design with Massive Machine Type Devices in Tactile Internet," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2016, pp. 1–6.
[86] Z. Zhou, Y. Guo, Y. He, X. Zhao, and W. M. Bazzi, "Access Control and Resource Allocation for M2M Communications in Industrial Automation," *IEEE Trans. Ind. Inform.*, vol. 15, no. 5, pp. 3093–3103, May 2019.
[87] M. Elsayed and M. Erol-Kantarci, "Deep Q-Learning for Low-Latency Tactile Applications: Microgrid Communications," in *Proc. IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, Oct. 2018, pp. 1–6.
[88] L. Ma and B. K. Yi, "Diversity and Subcarrier Index Modulation Based Multiple Access for the Tactile Internet," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2018, pp. 1–6.
[89] M. Condoluci, T. Mahmoodi, E. Steinbach, and M. Dohler, "Soft Resource Reservation for Low-Delayed Teleoperation Over Mobile Networks," *IEEE Access*, vol. 5, pp. 10445–10455, 2017.
[90] Z. Hou, C. She, Y. Li, T. Q. Quek, and B. Vucetic, "Burstiness-Aware Bandwidth Reservation for Ultra-Reliable and Low-Latency Communications in Tactile Internet," *EEE J. Sel. Areas Commun.*, vol. 36, no. 11, pp. 2401–2410, Nov. 2018.
[91] N. Ye, X. Li, H. Yu, A. Wang, W. Liu, and X. Hou, "Deep Learning Aided Grant-Free NOMA Toward Reliable Low-Latency Access in Tactile Internet of Things," *IEEE Trans. Ind. Inform.*, vol. 15, no. 5, pp. 2995–3005, May 2019.
[92] Y. Feng, A. Nirmalathas, and E. Wong, "A Predictive Semi-Persistent Scheduling Scheme for Low-Latency Applications in LTE and NR Networks," in *Proc. 2019 IEEE International Conference on Communications (ICC)*, May 2019, pp. 1–6.
[93] E. Khorov, A. Krasilov, and A. Malyshev, "Radio Resource Scheduling for Low-Latency Communications in LTE and Beyond," in *Proc. IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, Jun. 2017, pp. 1–6.
[94] Y. Su, X. Lu, L. Huang, X. Du, and M. Guizani, "Tac-U: A Traffic Balancing Scheme over Licensed and Unlicensed Bands for Tactile Internet," *Future Gener. Comput. Syst.*, vol. 97, pp. 41–49, Aug. 2019.
[95] C. She, C. Yang, and T. Q. S. Quek, "Cross-Layer Transmission Design for Tactile Internet," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, Dec. 2016, pp. 1–6.

[96] N. Gholipoor, H. Saeedi, and N. Mokari, “Cross-layer Resource Allocation for Mixed Tactile Internet and Traditional Data in SCMA Based Wireless Networks,” in *Proc. IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, Apr. 2018, pp. 356–361.
[97] A. Aijaz, “Toward Human-in-the-Loop Mobile Networks: A Radio Resource Allocation Perspective on Haptic Communications,” *IEEE Trans. Wirel. Commun.*, vol. 17, no. 7, pp. 4493–4508, Jul. 2018.
[98] B. Singh, Z. Li, and M. A. Uusitalo, “Flexible Resource Allocation for Device-to-device Communication in FDD System for Ultra-reliable and Low Latency Communications,” in *Proc. Advances in Wireless and Optical Communications (RTUWO)*, Nov. 2017, pp. 186–191.
[99] I. Budhiraja, S. Tyagi, S. Tanwar, N. Kumar, and J. J. Rodrigues, “DIYA: Tactile Internet Driven Delay Assessment NOMA-based Scheme for D2D Communication,” *IEEE Trans. Ind. Inform.*, pp. 1–1, 2019.
[100] C. She and C. Yang, “Ensuring the Quality-of-Service of Tactile Internet,” in *Proc. IEEE 83rd Vehicular Technology Conference (VTC Spring)*, May 2016, pp. 1–5.
[101] C. She and C. Yang, “Energy Efficient Design for Tactile Internet,” in *Proc. IEEE/CIC International Conference on Communications in China (ICCC)*, Jul. 2016, pp. 1–6.
[102] A. Avranas, M. Kountouris, and P. Ciblat, “Energy-Latency Tradeoff in Ultra-Reliable Low-Latency Communication with Retransmissions,” *IEEE J. Sel. Areas Commun.*, vol. 36, no. 11, pp. 2475–2485, Nov. 2018.
[103] A. Aijaz, “$\mathsfHap-SliceR$: A Radio Resource Slicing Framework for 5G Networks With Haptic Communications,” *IEEE Syst. J.*, vol. 12, no. 3, pp. 2285–2296, Sep. 2018.
[104] A. Ksentini, P. A. Frangoudis, A. PC, and N. Nikaein, “Providing Low Latency Guarantees for Slicing-Ready 5G Systems via Two-Level MAC Scheduling,” *IEEE Netw.*, vol. 32, no. 6, pp. 116–123, Nov. 2018.
[105] L. Dai, B. Wang, Z. Ding, Z. Wang, S. Chen, and L. Hanzo, “A Survey of Non-Orthogonal Multiple Access for 5G,” *IEEE Commun. Surv. Tutor.*, vol. 20, no. 3, pp. 2294–2323, thirdquarter 2018.
[106] Y. Feng, C. Jayasundara, A. Nirmalathas, and E. Wong, “A Feasibility Study of IEEE 802.11 HCCA for Low-Latency Applications,” *IEEE Trans. Commun.*, vol. 67, no. 7, pp. 4928–4938, Jul. 2019.
[107] F. Engelhardt, C. Rong, and M. Güneş, “Towards Tactile Wireless Multi-Hop Networks : The Tactile Coordination Function as EDCA Supplement,” in *Proc. Wireless Telecommunications Symposium (WTS)*, Apr. 2019, pp. 1–7.
[108] Y. Lv *et al.*, “Request-Based Polling Access: Investigation of Novel Wireless LAN MAC Scheme for Low-Latency E-Health Applications,” *IEEE Commun. Lett.*, vol. 23, no. 5, pp. 896–899, May 2019.
[109] Y. Lv *et al.*, “Dynamic Polling Sequence Arrangement for Low-Latency Wireless LAN,” in *Proc. Asia Communications and Photonics Conference (ACP)*, Oct. 2018, pp. 1–3.
[110] L. Ruan, M. P. I. Dias, and E. Wong, “Towards Tactile Internet Capable E-Health: A Delay Performance Study of Downlink-Dominated SmartBANs,” in *Proc. IEEE Global Communications Conference*, Dec. 2017, pp. 1–6.
[111] L. Feng, A. Ali, M. Iqbal, A. K. Bashir, S. A. Hussain, and S. Pack, “Optimal Haptic Communications Over Nanonetworks for E-Health Systems,” *IEEE Trans. Ind. Inform.*, vol. 15, no. 5, pp. 3016–3027, May 2019.

[112] A. Valkanis, P. Nicopolitidis, G. Papadimitriou, D. Kallergis, C. Douligeris, and P. D. Bamidis, "Efficient Resource Allocation in Tactile-Capable Ethernet Passive Optical Healthcare LANs," *IEEE Access*, pp. 1–1, 2020.
[113] L. Ruan, M. P. I. Dias, and E. Wong, "Machine Learning-Based Bandwidth Prediction for Low-Latency H2M Applications," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3743–3752, Apr. 2019.
[114] L. Ruan, I. Dias, and E. Wong, "Machine Intelligence in Supervising Bandwidth Allocation for Low-latency Communications," in *Proc. IEEE 20th International Conference on High Performance Switching and Routing (HPSR)*, May 2019, pp. 1–6.
[115] L. Ruan, M. P. I. Dias, M. Maier, and E. Wong, "Understanding the Traffic Causality for Low-Latency Human-to-Machine Applications," *IEEE Netw. Lett.*, vol. 1, no. 3, pp. 128–131, Sep. 2019.
[116] W. Tarneberg, M. Karaca, A. Robertsson, F. Tufvesson, and M. Kihl, "Utilizing Massive MIMO for the Tactile Internet: Advantages and Trade-Offs," in *Proc. IEEE International Conference on Sensing, Communication and Networking (SECON Workshops)*, Jun. 2017, pp. 1–6.
[117] C. Li, S. Yan, and N. Yang, "On Channel Reciprocity to Activate Uplink Channel Training for Downlink Wireless Transmission in Tactile Internet Applications," in *Proc. IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2018, pp. 1–6.
[118] K. C. Joshi, S. Niknam, R. V. Prasad, and B. Natarajan, "Analyzing the Tradeoffs in Using Millimeter Wave Directional Links for High Data-Rate Tactile Internet Applications," *IEEE Trans. Ind. Inform.*, vol. 16, no. 3, pp. 1924–1932, Mar. 2020.
[119] H. Ma, G. Cai, Y. Fang, J. Wen, P. Chen, and S. Akhtar, "A New Enhanced Energy-Detector-Based FM-DCSK UWB System for Tactile Internet," *IEEE Trans. Ind. Inform.*, vol. 15, no. 5, pp. 3028–3039, May 2019.
[120] G. Mountaser, T. Mahmoodi, and O. Simeone, "Reliable and Low-Latency Fronthaul for Tactile Internet Applications," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 11, pp. 2455–2463, Nov. 2018.
[121] D. Szabo, A. Gulyas, F. H. P. Fitzek, and D. E. Lucani, "Towards the Tactile Internet: Decreasing Communication Latency with Network Coding and Software Defined Networking," in *Proc. European Wireless 2015; 21th European Wireless Conference*, May 2015, pp. 1–6.
[122] F. Gabriel, J. Acevedo, and F. H. P. Fitzek, "Network Coding on Wireless Multipath for Tactile Internet with Latency and Resilience Requirements," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, Dec. 2018, pp. 1–6.
[123] Y. Xiao and M. Krunz, "Distributed Optimization for Energy-Efficient Fog Computing in the Tactile Internet," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 11, pp. 2390–2400, Nov. 2018.
[124] M. Aazam, K. A. Harras, and S. Zeadally, "Fog Computing for 5G Tactile Industrial Internet of Things: QoE-Aware Resource Allocation Model," *IEEE Trans. Ind. Inform.*, vol. 15, no. 5, pp. 3085–3092, May 2019.
[125] M. Tang, L. Gao, and J. Huang, "Enabling Edge Cooperation in Tactile Internet via 3C Resource Sharing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 11, pp. 2444–2454, Nov. 2018.

[126] Z. Ning, X. Wang, J. J. P. C. Rodrigues, and F. Xia, "Joint Computation Offloading, Power Allocation, and Channel Assignment for 5G-Enabled Traffic Management Systems," *IEEE Trans. Ind. Inform.*, vol. 15, no. 5, pp. 3058–3067, May 2019.
[127] J. Xu, K. Ota, and M. Dong, "Energy Efficient Hybrid Edge Caching Scheme for Tactile Internet in 5G," *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 2, pp. 483–493, Jun. 2019.
[128] X. Xu, Q. Liu, and E. Steinbach, "Toward QoE-driven Dynamic Control Scheme Switching for Time-delayed Teleoperation Systems: A Dedicated Case Study," in *Proc. IEEE International Symposium on Haptic, Audio and Visual Environments and Games (HAVE)*, Oct. 2017, pp. 1–6.
[129] S. Liu, M. Li, X. Xu, E. Steinbach, and Q. Liu, "QoE-Driven Uplink Scheduling for Haptic Communications Over 5G Enabled Tactile Internet," in *Proc. IEEE International Symposium on Haptic, Audio and Visual Environments and Games (HAVE)*, Sep. 2018, pp. 1–5.
[130] B. Chang, G. Zhao, M. A. Imran, L. Li, and Z. Chen, "Dynamic QoS Allocation for Real-Time Wireless Control in Tactile Internet," in *2018 IEEE 5G World Forum (5GWF)*, Jul. 2018, pp. 273–277.
[131] N. Ashraf, A. Hasan, H. K. Qureshi, and M. Lestas, "Combined Data Rate and Energy Management in Harvesting Enabled Tactile IoT Sensing Devices," *IEEE Trans. Ind. Inform.*, vol. 15, no. 5, pp. 3006–3015, May 2019.
[132] A. Garcia-Saavedra, G. Iosifidis, X. Costa-Perez, and D. J. Leith, "Joint Optimization of Edge Computing Architectures and Radio Access Networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 11, pp. 2433–2443, Nov. 2018.
[133] D. Lumbantoruan, M. Farhoudi, A. Mihailovic, and A. H. Aghvami, "Towards an Efficient Path Selection for Tactile Internet Traffic via Multi-Plane Routing," in *Proc. 26th International Conference on Telecommunications (ICT)*, Apr. 2019, pp. 60–65.
[134] M. Farhoudi, P. Palantas, B. Abrishamchi, A. Mihailovic, and A. H. Aghvami, "A Novel Reliable Routing Scheme for Tactile-oriented Internet Traffic," in *Proc. 24th International Conference on Telecommunications (ICT)*, May 2017, pp. 1–7.
[135] J. Ren, C. Lin, Q. Liu, M. S. Obaidat, G. Wu, and G. Tan, "Broadcast Tree Construction Framework in Tactile Internet via Dynamic Algorithm," *J. Syst. Softw.*, vol. 136, pp. 59–73, Feb. 2018.
[136] K. Shin, S. Choi, and H. Kim, "Flit Scheduling for Cut-Through Switching: Towards Near-Zero End-to-End Latency," *IEEE Access*, vol. 7, pp. 66369–66383, 2019.
[137] L. Deng, W. S. Wong, P.-N. Chen, Y. S. Han, and H. Hou, "Delay-Constrained Input-Queued Switch," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 11, pp. 2464–2474, Nov. 2018.
[138] 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*, Sep. 2015, pp. 97–102.
[139] L. Qu, C. Assi, K. Shaban, and M. J. Khabbaz, "A Reliability-Aware Network Service Chain Provisioning With Delay Guarantees in NFV-Enabled Enterprise Datacenter Networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, no. 3, pp. 554–568, Sep. 2017.
[140] L. Qu, C. Assi, M. Khabbaz, and Y. Ye, "Reliability-Aware Service Function Chaining With Function Decomposition and Multipath Routing," *IEEE Trans. Netw. Serv. Manag.*, pp. 1–1, 2019.
[141] N. Zhang, Y. Liu, H. Farmanbar, T. Chang, M. Hong, and Z. Luo, "Network Slicing for Service-Oriented Networks Under Resource Constraints," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2512–2521, Nov. 2017.

[142] Y. Xu and V. P. Kafle, “Reliable service function chain provisioning in software-defined networking,” in *2017 13th International Conference on Network and Service Management (CNSM)*, Nov. 2017, pp. 1–4.
[143] E. Amaldi, S. Coniglio, A. M. C. A. Koster, and M. Tieves, “On the computational complexity of the virtual network embedding problem,” *Electron. Notes Discrete Math.*, vol. 52, pp. 213–220, Jun. 2016.
[144] F. Glover, “Tabu Search—Part I,” *ORSA J. Comput.*, vol. 1, no. 3, pp. 190–206, Aug. 1989.
[145] 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)*, Oct. 2017, pp. 1–6.
[146] C. Mouradian, S. Kianpisheh, M. Abu-Lebdeh, F. Ebrahimnezhad, N. T. Jahromi, and R. H. Glitho, “Application Component Placement in NFV-Based Hybrid Cloud/Fog Systems With Mobile Fog Nodes,” *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1130–1143, May 2019.
[147] E. K. Burke and Y. Bykov, “A late acceptance strategy in hill-climbing for exam timetabling problems,” presented at the Proceedings of the PATAT 2008 Conference.
[148] “SNDlib 1.0—Survivable Network Design Library - Orlowski - 2010 - Networks - Wiley Online Library.” https://onlinelibrary.wiley.com/doi/abs/10.1002/net.20371 (accessed Nov. 02, 2019).
[149] Q. Zhang, F. Liu, and C. Zeng, “Adaptive Interference-Aware VNF Placement for Service-Customized 5G Network Slices,” in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, Apr. 2019, pp. 2449–2457.
[150] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, “Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions,” in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 98–106.
[151] R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei, “The Benders decomposition algorithm: A literature review,” *Eur. J. Oper. Res.*, vol. 259, no. 3, pp. 801–817, Jun. 2017.
[152] J.-F. Cordeau, G. Stojković, F. Soumis, and J. Desrosiers, “Benders Decomposition for Simultaneous Aircraft Routing and Crew Scheduling,” *Transp. Sci.*, vol. 35, no. 4, pp. 375–388, Nov. 2001.
[153] G. Cornuejols, *Combinatorial Optimization: Packing and Covering*, 74 vols. Society for Industrial and Applied Mathematics (SIAM), 2014.
[154] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, “The Internet Topology Zoo,” *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.
[155] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, “Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges - IEEE Journals & Magazine.” https://ieeexplore.ieee.org/document/7926921 (accessed Jan. 24, 2019).
[156] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, “Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions,” *IEEE Commun. Surv. Tutor.*, vol. 20, no. 3, pp. 2429–2453, 2018.
[157] A. Baumgartner, T. Bauschert, A. M. C. A. Koster, and V. S. Reddy, “Optimisation Models for Robust and Survivable Network Slice Design: A Comparative Analysis,” in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec. 2017, pp. 1–7.

[158] R. Szabo, M. Kind, F. Westphal, H. Woesner, D. Jocha, and A. Csaszar, “Elastic network functions: opportunities and challenges,” *IEEE Netw.*, vol. 29, no. 3, pp. 15–21, May 2015.