Resource Allocation in SDN/NFV-Enabled Core Networks

by

Junling Li

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2020

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Amiya Nayak Professor, University of Ottawa
Supervisor(s):	Xuemin (Sherman) Shen University Professor, University of Waterloo
Internal Member:	Zhou Wang Professor, University of Waterloo
Internal Member:	Xiaodong Lin Adjunct Associate Professor, University of Waterloo
Internal-External Member:	Wei-Chau Xie Professor, University of Waterloo

Author's Declaration

Abstract

For next generation core networks, it is anticipated to integrate communication, storage and computing resources into one unified, programmable and flexible infrastructure. Software-defined networking (SDN) and network function virtualization (NFV) become two enablers. SDN decouples the network control and forwarding functions, which facilitates network management and enables network programmability. NFV allows the network functions to be virtualized and placed on high capacity servers located anywhere in the network, not only on dedicated devices in current networks. Driven by SDN and NFV platforms, the future network architecture is expected to feature centralized network management, virtualized function chaining, reduced capital and operational costs, and enhanced service quality.

The combination of SDN and NFV provides a potential technical route to promote the future communication networks. It is imperative to efficiently manage, allocate and optimize the heterogeneous resources, including computing, storage, and communication resources, to the customized services to achieve better quality-of-service (QoS) provisioning. This thesis makes some in-depth researches on efficient resource allocation for SDN/NFV-enabled core networks in multiple aspects and dimensionality. Typically, the resource allocation task is implemented in three aspects. Given the traffic metrics, QoS requirements, and resource constraints of the substrate network, we first need to compose a virtual network function (VNF) chain to form a virtual network (VN) topology. Then, virtual resources allocated to each VNF or virtual link need to be optimized in order to minimize the provisioning cost while satisfying the QoS requirements. Next, we need to embed the virtual network (i.e., VNF chain) onto the substrate network, in which we need to assign the physical resources in an economical way to meet the resource demands of VNFs and links. This involves determining the locations of NFV nodes to host the VNFs and the routing from source to destination. Finally, we need to schedule the VNFs for multiple services to minimize the service completion time and maximize the network performance.

In this thesis, we study resource allocation in SDN/NFV-enabled core networks from the aforementioned three aspects. First, we jointly study how to design the topology of a VN and embed the resultant VN onto a substrate network with the objective of minimizing the embedding cost while satisfying the QoS requirements. In VN topology design, optimizing the resource requirement for each virtual node and link is necessary. Without topology optimization, the resources assigned to the virtual network may be insufficient or redundant, leading to degraded service quality or increased embedding cost. The joint problem is formulated as a Mixed Integer Nonlinear Programming (MINLP), where queueing theory is utilized as the methodology to analyze the network delay and help to define the optimal set of physical resource requirements at network elements. Two algorithms are proposed to obtain the optimal/near-optimal solutions of the MINLP model.

Second, we address the multi-SFC embedding problem by a game theoretical approach, considering the heterogeneity of NFV nodes, the effect of processing-resource sharing among various VNFs, and the capacity constraints of NFV nodes. In the proposed resource constrained multi-SFC embedding game (RC-MSEG), each SFC is treated as a player whose objective is to minimize the overall latency experienced by the supported service flow, while satisfying the capacity constraints of all its NFV nodes. Due to processing-resource sharing, additional delay is incurred and integrated into the overall latency for each SFC. The capacity constraints of NFV nodes are considered by adding a penalty term into the cost function of each player, and are guaranteed by a prioritized admission control mechanism. We first prove that the proposed game RC-MSEG is an exact potential game admitting at least one pure Nash Equilibrium (NE) and has the finite improvement property (FIP). Then, we design two iterative algorithms, namely, the best response (BR) algorithm with fast convergence and the spatial adaptive play (SAP) algorithm with great potential to obtain the best NE of the proposed game.

Third, the VNF scheduling problem is investigated to minimize the makespan (i.e., overall completion time) of all services, while satisfying their different end-to-end (E2E) delay requirements. The problem is formulated as a mixed integer linear program (MILP) which is NP-hard with exponentially increasing computational complexity as the network size expands. To solve the MILP with high efficiency and accuracy, the original problem is reformulated as a Markov decision process (MDP) problem with variable action set. Then, a reinforcement learning (RL) algorithm is developed to learn the best scheduling policy by continuously interacting with the network environment. The proposed learning algorithm determines the variable action set at each decision-making state and accommodates different execution time of the actions. The reward function in the proposed algorithm is carefully designed to realize delay-aware VNF scheduling.

To sum up, it is of great importance to integrate SDN and NFV in the same network to accelerate the evolution toward software-enabled network services. We have studied VN topology design, multi-VNF chain embedding, and delay-aware VNF scheduling to achieve efficient resource allocation in different dimensions. The proposed approaches pave the way for exploiting network slicing to improve resource utilization and facilitate QoS-guaranteed service provisioning in SDN/NFV-enabled networks.

KEY WORDS: Core network, network function virtualization, software-defined networking, VN topology design, VNF chain embedding, VNF scheduling.

Acknowledgements

I would like to express my sincere appreciation to my supervisor, Professor Xuemin (Sherman) Shen for the constant support and expert guidance during my Ph.D. study in University of Waterloo. His rigorous academic attitude, warmth encouragement, and innovative insights have broadened my academic vision and improved my research abilities. His great support has not only gave me the direction to march forward on the academic road but also helped me made a clear career plan. It is my great honor to work under his supervision and guidance.

I would also like to express my appreciation to Professor Weihua Zhuang. It is my great honor to participate in the Huawei SONAC project under the knowledgeable instructions of Professor Zhuang and Professor Shen. The three years' guidance and inspiration in the Huawei Project has greatly motivated me to do in-depth thinking on my research.

Specially, I would like to gratefully acknowledge the members of my Ph.D. thesis examination committee: Prof. Amiya Nayak, Prof. Wei-Chau Xie, Prof. Zhou Wang, and Prof. Xiaodong Lin for their time, invaluable suggestions and corrections. Their suggestions and valuable questions have helped me significantly to improve the quality of my thesis.

I would like to thank all the BBCR members. The weekly group meetings gave me a great opportunity to enrich my knowledge and improve my presentation skills. The harmonious and friendly learning atmosphere has also helped me to discover new research topics and improved my problem-solving skills. Except for the group meeting, the individual meetings also allows us to discuss the most recent research works between each other and learn from each other.

I would like to thank all the BBCR-SDN subgroup members for their enthusiasm and valuable discussions. I would like to especially thank Dr. Qiang Ye, Dr. Ning Zhang, Dr. Shan Zhang, Dr. Wei Quan, Dr. Phu Thinh Do, Dr. Peng Yang, Weisen Shi, Omar Alhussein, Kaige Qu, Jiayin Chen, Si Yan for their inspiring discussions and valuable insights on my research and life.

Finally, I wish to thank my parents and my grandparents for their love, belief, support, and encouragement throughout my study. A special thank to my husband Jianan Zhang for his constant accompany, encouragement, and motivation. The nine years' company through my undergraduate, master, and Ph.D. studies have encouraged me to keep pursuing my dream. Together we have enjoyed countless happy times and overcome many challenges through my whole academic career.

Junling Li
May 13, 2020
Waterloo, Ontario, Canada

Dedication

To my dear parents, my grandparents and my husband.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

5G	Fifth generation
B5G	Beyond 5G
BR	Best Response
CAPEX	Capital Expenditure
CN	Core Network
eMBB	enhanced Mobile Broadband
E2E	End-to-End
FIP	Finite Improvement Property
InP	Infrastructure Provider
IoT	Internet-of-Things
ISP	Internet service provider
JSP	Job-Shop Problem
JSSD	Job-Shop Scheduling Problem with Deadlines
MDP	Markov Decision Process
MILP	Mixed Integer Linear Programming
MINLP	Mixed Integer Nonlinear Programming
mMTC	massive Machine-type Communication
NBI	Northbound Interface
NE	Nash Equilibrium
NF	Network Function
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
NFV-RA	NFV Resource Allocation
NV	Network Virtualization
QoS	Quality-of-Service

OPEX	Operational Expenditure
PSO	Particle Swarm Optimization
RAN	Radio Access Network
RL	Reinforcement Learning
SDN	Software-Defined Networking
SFC	Service Function Chain
SAP	Spatial Adaptive Play
SBI	Southbound Interface
TAP	Traffic Aggregation Points
URLLC	Ultra-reliable and Low-latency Communication
VN	Virtual Network
VNE	Virtual Network Embedding
VNF	Virtual Network Function
VNO	Virtual Network Operator

Chapter 1

Introduction

With the prevalence of Internet-of-Things (IoT), the communication networks are envisioned to accommodate a proliferation of connected devices and diversified services, with largely increased traffic volume and differentiated quality-of-service (QoS) demands [1], [2]. These rapid-growing and on-demand services require efficient and effective network organization and configuration to achieve fast service provisioning. To realize service oriented networking, significant changes are to be made in the currently deployed networking architecture and a number of technical challenges arise for resource management [3]. Core networks as the central element of current communication networks can provide customized services to end users connected by the access networks. Software-defined networking (SDN) and network function virtualization (NFV) are two promising and complementary technologies to reduce the function provisioning cost and improve the heterogeneous resource utilization for customized end-to-end (E2E) service deliveries [4]. In this chapter, we provide an overview of SDN and NFV, followed by the new challenges of resource allocation in SDN/NFV-enabled core networks. Finally, we present three key components/stages for efficient resource allocation in SDN/NFV-enabled core networks.

1.1 Overview of NFV and SDN for Next Generation Networks

The fifth generation networks (5G) and beyond, which are commercially available in 2020, will unlock the possibilities to enhance people's life while bringing in many new opportunities and user experiences. Massive number of intelligent devices (e.g., sensors, vehicles, wearable devices) are expected to be connected to accommodate various newly emerging services [5] [6], including ultra-reliable and low-latency communication (URLLC) services, massive machine-type communication (mMTC) services, and enhanced mobile broadband (eMBB) services [5], [7]. These new services demonstrate highly diversified traffic characteristics and differentiated quality-of-service (QoS) requirements [8] in terms of latency, security, reliability, and complexity [9]. In particular, URLLC is crucial to many prospective applications, such as industry automation, autonomous driving, and remote surgery, which require a low end-to-end (E2E) communication delay in the order of milliseconds. Meanwhile, the reliability requirements of URLLC services can be higher than 99.999% [8]. mMTC services usually have to support massive number of devices with low mobility, thus having less stringent requirements on connection delay and reliability. In contrast, eMBB services require high data rates (up to the order of Gbps) supported on moving devices over a wide coverage area. Typical applications include 4K/8K ultra-high definition video streaming, virtual reality, and augmented reality. In addition, in the 5G era, new requirements arise for the evolving network paradigm, including 1) cost-effective network deployment for seamless device access, 2) enhanced resource utilization to accommodate high traffic volume, and 3) flexible function placement for service customization [10][11]. The current one-type-fits-all network is not able to provide customized services with various quality-of-service (QoS) requirements well.

Therefore, how to satisfy diverse and stringent E2E QoS requirements for multiple services remains a significant challenge. Moreover, to accommodate the aforementioned differentiated services, heterogeneous resources should be intelligently integrated into 5G networks with efficient allocation [9]. The physical resource pool manifests high heterogeneity, ranging from radio access networks (RANs) to core networks (CNs). For the RAN segment, spectrum and time slots are the major resources that can be allocated by operators, while for the CN segment, the resources include computing, storage and networking resources at each CN element (nodes and links). In addition, to support massive connected intelligent terminals and provide seamless connectivity, macro-cells, small cells, and femtocells are expected to be densely deployed, which composite heterogeneous multi-tier access networks in 5G networks and beyond [12]. Various wireless access technologies (e.g. LTE, Wi-Fi, WCDMA, etc.) will coexist and cooperate with each other to better utilize network resources. Through efficient integration, the service quality is expected to be enhanced. The heterogeneity in services, resources, and access technologies increases the complexity of 5G networks, leading to high costs of deployment and maintenance. Second, to provide customized services, the strict service isolation is a necessity when considering the dynamics of one service, such as the topology changes and traffic fluctuations. Last, to better support the heterogeneity of services without increasing the network deployment cost, it is expected to have a unified framework to integrate computing, networking, and storage resources for more efficient global resource allocation. Therefore, the development of 5G

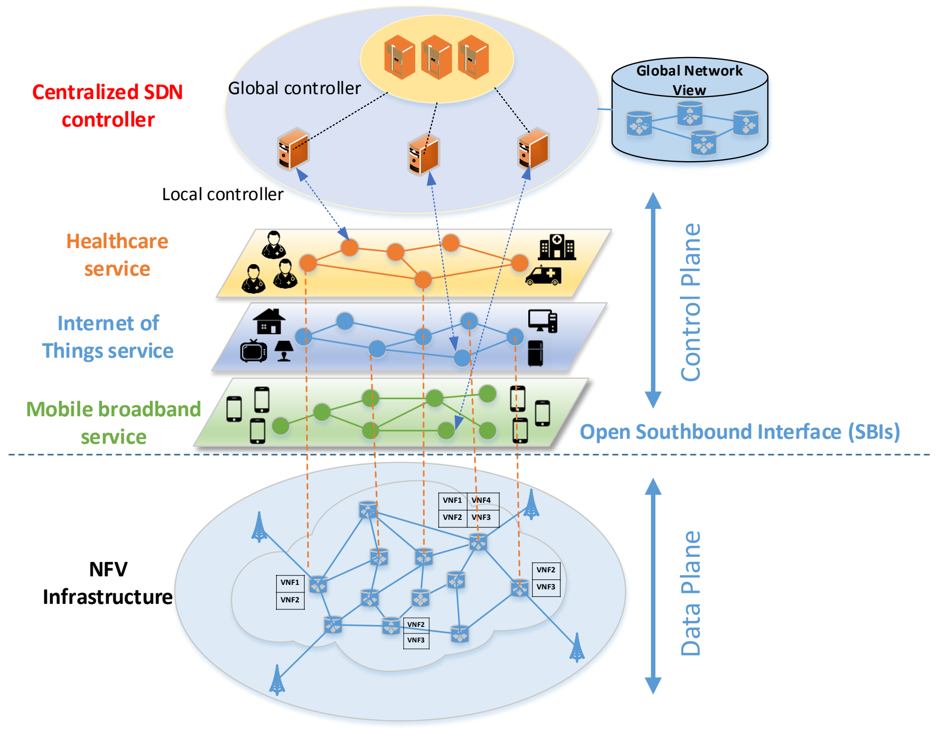


Figure 1.1: A high-level overview of SDN/NFV-enabled core networks.

networks is not simply an inheritance from the current 3GPP mobile network, but an evolution of the network architecture. The B5G networks are expected to feature enhanced mobile broadband, massive machine-type communication, and ultra reliable low latency. A scalable, delay-optimal, highly adaptable and flexible network architecture is desired, which has attracted great attentions from both the industry and the academia.

Software-Defined Networking (SDN) and Network Function Virtualization (NFV) are recognized as the two most promising technologies to address the aforementioned challenges [4], [13]. SDN decouples the network control and forwarding functions, which facilitates network management and enables network programmability. NFV allows the network functions to be virtualized and placed on high capacity servers located anywhere in the network, not only on dedicated devices in traditional networks. Fig. 1.1 illustrates a typical SDN/NFV-enabled core network scenario. Each customized service is supported by chaining several virtual network functions (VNFs) and embedding the VNF chain onto the substrate network. To support the diversified services, a SDN-based hierarchical controller is employed to manage virtual networks on top of a NFV infrastructure. The logically centralized controller has a global view of the whole network to monitor the network states and manage the traffic flows. Additionally, the central controller also helps to achieve better load balancing in both control plane and data plane. In this way, different services are strictly isolated and can be managed and operated independently. Bringing SDN and NFV into 5G mobile communication creates unique benefits, such as flexible service deployment, reduced costs, and enhanced service quality.

In the following sections, I illustrate the fundamental concepts of network virtualization (NV), network function virtualization and software-defined networking. The typical network architectures as well as the the main network features enabled by the three technologies are provided.

1.1.1 Network Virtualization/Slicing

Network virtualization (NV), or network slicing, holds a great potential to support various services by logically partitioning the network resources into multiple virtual slices for customized services [14], [15] [16] [17]. For 5G core networks, with network virtualization, each Internet service providers (ISPs) is decomposed into two network entities: the infrastructure provider (InP) and the virtual network operator (VNO). InPs take charge of deploying and maintaining network infrastructures. Each VNO partitions the computing/bandwidth resources on network elements (i.e., network servers, routers, and transmission links) to create different virtual network slices, a. k. a. *virtual networks* (VNs), for supporting different customized end-to-end (E2E) services over a physical substrate network. As shown in Fig. 1.2, NV creates multiple VNs on top of a common physical substrate network. Each of the VNs represents a customized end-to-end service and is composed of a set of virtual nodes interconnected by virtual links. Each virtual node (virtual link) is physically operated on a network server/switch (transmission link) and occupies a portion of physical computing (bandwidth) resources. Through NV, multiple VNs coexist on a common substrate network and their resources are logically isolated and managed independently by different VNOs, which not only allows efficient resource sharing among heterogeneous services but also improves resource utilization and provide more flexibilities for the network.

1.1.1.1 Virtual Network Embedding

A VN typically consists of several virtual nodes interconnected via (either wired or wireless) virtual links. Those virtual nodes and links form a virtual topology. By virtualizing both node and link resources of a SN, multiple VNs can be co-hosted and existing on the same physical hardware. The problem of embedding multiple virtual networks onto a

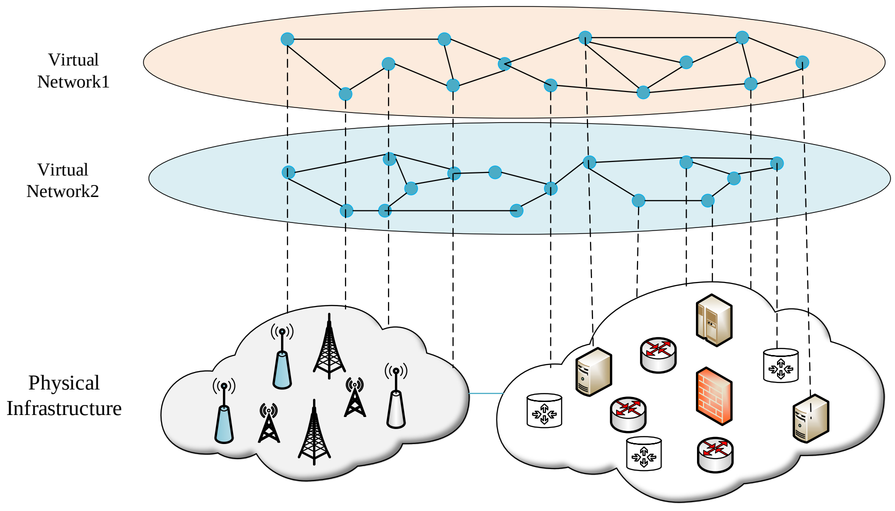


Figure 1.2: Network virtualization environment.

common substrate network is the main resource allocation task in network virtualization and is usually termed as Virtual Network Embedding (VNE). Efficient VNE algorithms can dramatically maximize physical resource utilization via resource sharing. There are various metrics to measure the optimality of VNE algorithms, such as the acceptance ratio, revenue, and revenue-to-cost ratio, etc.

VNE algorithms deal with the allocation of virtual resources in both nodes and links, so it can be divided into two sub-problems, i.e., Virtual Node Mapping (VNoM) and Virtual Link Mapping (VLiM) [18]. VNoM deals with how to allocate virtual nodes in physical nodes while VLiM focuses on how to map the virtual links connecting these virtual nodes to paths connecting the corresponding physical nodes. The node mappings are usually achieved by employing greedy methods, while the link mappings are typically obtained using shortest path, k-shortest paths [19], and multi-commodity flow algorithms [18]. Both of these two sub-problems are NP-hard problems [18], and can be jointly studied to achieve better mapping strategy but with increased complexity.

In last few years, many research efforts have been devoted to the development of VNE approaches for different scenarios. In what follows, we give a brief overview of the seminal works. A comprehensive survey of existing VNE approaches can be found in [18]. The authors of [20] first introduced path splitting and path migration to VNE. The original VNE problem is first divided into two sub-problems, i.e., the node mapping problem and the link mapping problem, then the two sub-problems are solved independently. The coordination in node mapping and link mapping was first illustrated by the approach presented in [21], [22]. The node mapping is solved by a relaxed MIP formulation. The substrate nodes are chosen in a way so that the mapping cost for the virtual links are likely to be low. It is widely recognized that the end-to-end delay requirement is one of the most important QoS requirements for customized services in B5G networks. The satisfaction of the delay requirement of each VN remains an important aspect of VNE. There exist some works in the VNE literature that try to consider the delay requirement as a constraint for the VNE problem, such as [23] and [24]. In [23], the VNE problem with node and link resource constraints and path delay constraint has been considered for evolving networks. The problem has been formulated as integer programming then solved by a heuristic algorithm. In [24], an optimization framework that minimizes the end-to-end delay for VNE in mobile networks has been proposed.

1.1.2 Network Function Virtualization

Network services provided by operators can be considered as chains of network functions (NFs). Various NFs, such as firewall, deep packet inspection, data monitoring, encryption and decryption, have specific hardware requirements and configurations. Currently, these NFs are implemented on physical middleboxes (hardware appliances). Due to the explosive development of mobile Internet and Internet of things, the number of diversified new services will prominently increase in B5G networks [25]. High capital and operational expenditures are demanded for network operators to deploy, maintain, and upgrade physical middleboxes when new network services are offered.

Network function virtualization, which was initiated in October 2012 [26], is a promising technology to address the aforementioned challenges. As the home of the Industry Specification Group for NFV, the European Telecommunications Standards Institute (ETSI) has proposed a number of use cases driven by NFV since its initiative [27]. The new use cases include a variety of novel applications not only for individual customers but also for the evolved core network.

The basic concept of NFV is to decouple the functions from the hardware equipment on which they run [28]. In NFV, the NFs, which can only run on dedicated devices in traditional networks, are virtualized to be placed on high capacity servers located anywhere in the network. By implementing NFV, the physical resources in the network can be virtualized to allow efficient resource sharing among different services. The customized services can also be accommodated in a cost-effective, agile and flexible manner [10].

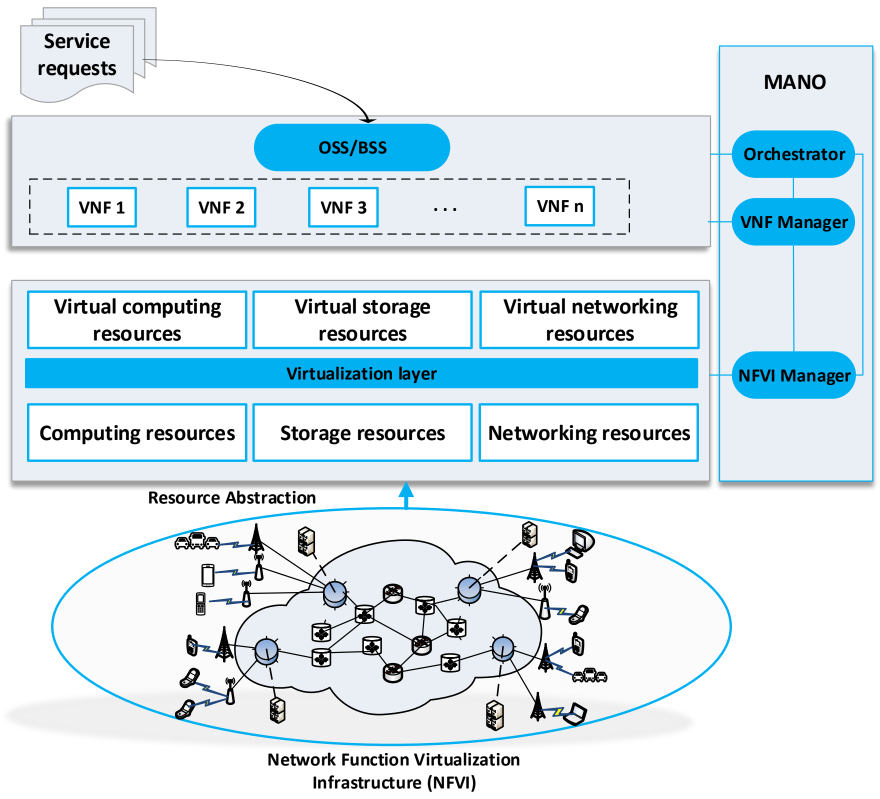


Figure 1.3: Network function virtualization architecture.

1.1.2.1 NFV Architecture

As presented in Fig. 1.3, the NFV architecture has three key components: NFVI, VNFs, and NFV Management and Orchestration (MANO) [29]. NFVI contains all the hardware and software resources which are utilized to run the VNFs. The physical resources may include computing, storage and networking resources. Through a virtualization layer, the physical resources are abstracted as virtual resources.

Network functions (NFs) are functional building blocks implemented on specific network elements (middleboxs), which have well-defined interfaces to end users and specific functional behaviors. In the NFV environment, a network service is a set of chained VNFs, defined by its types of VNFs, their respective order in the chain, and the resource allocation of the chain in the NFVI. Network services can be supported by chaining the VNFs to form VNF chains then embedding the resultant chains onto an NFV-enabled infrastructure

[30] [31]. VNF chain is in fact an ordered list of network function instances that traffic traverses through. It can be considered as a novel service chain deployment model in NFV and SDN environment.

The role of the NFV MANO is to fulfill all the tasks for virtualizing the network functions and the infrastructure [29]. Specifically, those tasks include orchestrating and managing the resources (including hardware and software) for virtualizing the infrastructure and executing the VNFs, and also include defining the interfaces to allow communications between different network elements in the NFV infrastructure.

1.1.2.2 Benefits of NFV

As a new network architecture, network function virtualization enables network functions to be implemented through software running on general-purpose high volume servers. These network functions were traditionally implemented on dedicated hardware such as middleboxes or network appliances. The key benefits brought by NFV can be summarized as follows [26],[32]

1. Increased Deployment Flexibility

   In the core network domain of the current network architecture, a set of dedicated middleboxes is deployed for each specific network service. When new services are required, new middleboxes have to be deployed, while increasing the number of network elements for service customization. With NFV, one can use commodity servers to host VNFs and deploy a new service by installing VNFs on commodity servers instead of adding new hardware devices. In this way, flexible function placement can be achieved for service customization.

2. Reduced Capital Expenditure (CAPEX) and Operational Expenditure (OPEX)

   With increased deployment flexibility and agility to integrate new network services, significant reductions in CAPEX and OPEX are expected to be obtained. With the help of software, NFV allows a common physical network to support multiple customized services simultaneously. Improved hardware upgrading and maintaining efficiency also leads to reduced CAPEX and OPEX.

3. Faster Service Life Cycle

   Compared with the time-consuming and costly function update process in an existing network, in an NFV-enabled network, adding new functions is much easier. The function update can be completed by only updating the software. Because of this, the life cycle of VNFs can be much shorter and services deployment time can be greatly reduced.

4. Better Network Function Provisioning and Isolation

   In the NFV framework, traditional middleboxes are managed and orchestrated in the form of VNFs, which are in fact software modules that can be installed on general-purpose commodity servers. This allows programmability and isolation of each function, so they can be managed independently.

5. Improved Resource Utilization

   Through network virtualization, the whole network is sliced into multiple virtual networks. Each VN is used to support a customized service in a specific scenario. Multiple VNs are mutually isolated and can coexist in the common infrastructure sharing the same physical resource pool. The heterogeneous resources are virtualized to allow resource sharing among different VNs and to improve resource utilization in both RANs and CNs.

1.1.3 Software-Defined Networking

1.1.3.1 Conventional SDN

In the traditional networking approaches, most network functionalities are implemented using dedicated hardware, i.e., router, switch, and firewall. In the contrast, SDN networking architectures aim to separate the network control plane from data plane, so that hardware products can be standardized and the whole network can be made directly programmable. As a result, enterprises can get rid of hardware restrictions of the network architecture. Network adjustment, expansion and upgrade can be completed by installing and upgrading the software. Both the capital and operational costs are reduced while simpler management and more flexible control are achieved. Fig. 1.4 shows the illustration of a conventional physical SDN network, where network applications (App1, ..., Appn) interact through the Northbound Interfaces (NBIs) with the SDN controller, which in turn interact with the physical SDN network through Southbound Interfaces (SBIs) [33].

1.1.3.2 Combining NV and SDN

In the last a few years, network virtualization and software-defined networking have been the two hottest topics in networking. They both have their own distinct advantages

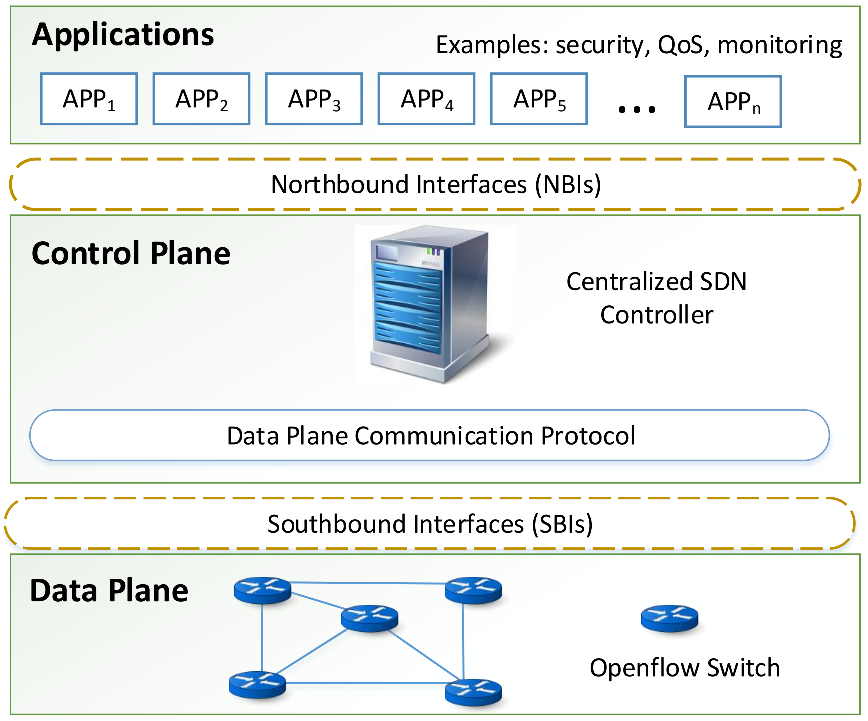

Applications
Examples: security, QoS, monitoring
APP$_1$
APP$_2$
APP$_3$
APP$_4$
APP$_5$
…
APP$_n$
Northbound Interfaces (NBIs)
Control Plane
Centralized SDN Controller
Data Plane Communication Protocol
Southbound Interfaces (SBIs)
Data Plane
Openflow Switch

Figure 1.4: Conventional (non-virtualized) SDN network.

and can be combined to enable more powerful technologies for future networks. Different from NV, SDN requires the construction of new network where the control and data layers are decoupled. In other words, SDN modifies the physical network, while NV can reside on the servers of the existing network. SDN creates network abstractions to enable faster innovation, open innovation creates competitive supply of innovative applications by third parties, while NV reduces CAPEX, OPEX, space and power consumption.

One of the enablers for future networking technologies is virtual SDN networks, in which a hypervisor is incorporated between the physical SDN network and the SDN control plane [33]. The hypervisor has a global view of the whole physical SDN network and interacts with it via the Southbound API. The physical SDN network is virtualized to create isolated virtual SDN networks, which are controlled by different virtual SDN controllers. Each virtual SDN controller is able to perceive the status and interact with the virtual SDN network under its management. An illustration of the concept of SDN network virtualization is shown in Fig. 1.5.

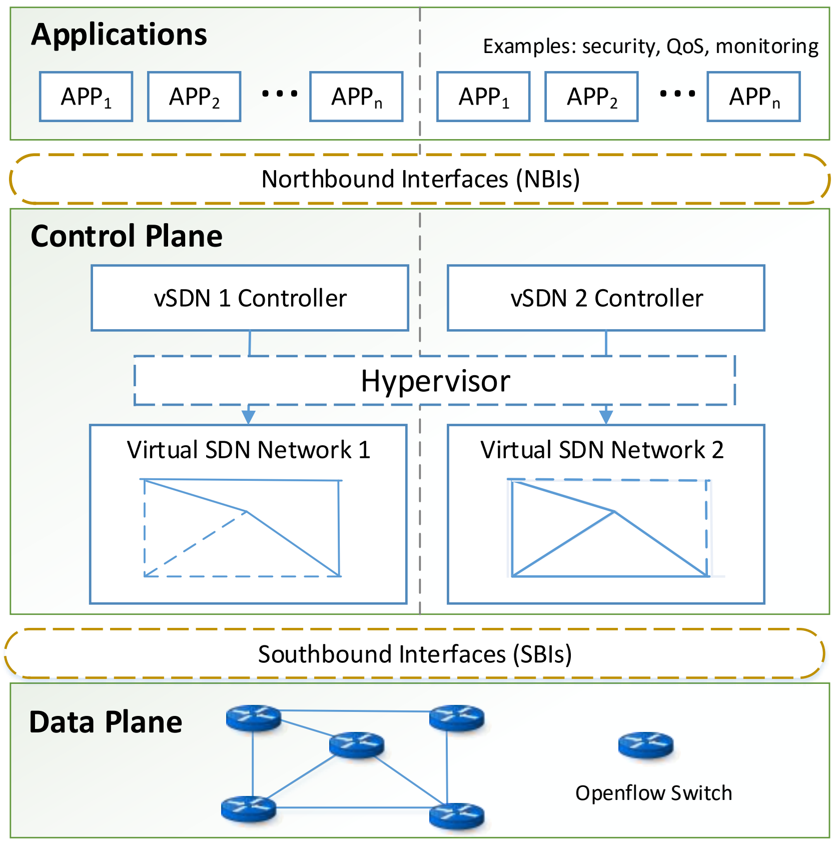


Figure 1.5: Virtualization of the SDN network.

1.1.3.3 Benefits of SDN

SDN enables the programmability of networks such that operators are able to support services with finite network resources [34]. The biggest promise of SDN is that it will centralize and simplify the control of network management by separating the network control plane from data plane. Below we summarize some highlights of the specific advantages of SDN [34]-[35]:

1. Centralized Control Provisioning

   SDN provide a centralized view of the whole network, which enables easier centralized network management and provisioning. Through separating the control and data planes, SDN is able to facilitate service delivery and offer more agility in provisioning both virtual and physical network resources at a central location.

2. Comprehensive Infrastructure Management

   Many new applications and virtual machines have to be set up in order to accommodate new processing requests such as big data. SDN allows IT teams to be able to run experiments with network configurations without causing any effects to the network. Moreover, in SDN, there exists a single centralized controller that manages physical and virtual switches and network equipment. This cannot be achieved with simple network management protocol (SNMP).

3. Reduced Capital and Operational Expenditures (CAPEX and OPEX)

   The implementation of SDN makes the optimization of existing network devices easier. Existing hardware can be re-purposed with the instruction given by the SDN controller. In addition, since all the intelligence of new devices are centered at the SDN controller, new devices actually becomes “white box” switches and less expensive hardware can be deployed. On the other hand, the operating expenditures can also be reduced due to administrative efficiency, centralized management, and better control of virtualization.

4. Guaranteed Data Delivery

   One of the primary advantages of SDN is the ability to shape and control data traffic. The QoS requirements of multimedia services can be better guaranteed as SDN is able to direct and automate data traffic.

5. Enhanced Security

   SDN centralizes security control into one entity, such as the SDN controller. By doing this, SDN can be used effectively to manage security throughout the whole network provided that it is securely and properly implemented.

1.1.4 Relationships Between NFV and SDN

NFV and SDN share a lot of similarities as they both pursue the openness of network and the use of standard network hardware. Both of them attempt to exploit virtualization and automation to achieve their own potential goals. Actually, they are highly complementary to each other and may be combined to enable more beneficial networking technologies. NFV is able to support SDN and vice versa. The performance of NFV can be enhanced via SDN-based approaches and vice versa, though most of the advantages coming from combining the two technologies have not been observed in practice.

On the other hand, there are also significant differences between SDN and NFV since they are different concepts. One of the differences is that, SDN aims to separate the control and forwarding planes and orchestrate the network-wide resources via a centralized controller. On contrary, NFV focuses on optimizing network services themselves by replacing the expensive dedicated middleboxes with generic servers that can provide a number of different VNFs through software. Another difference is that, NFV can be implemented on the servers of an existing network and thus does not require the construction of new network. On contrary, the deployment of SDN relies on decoupled control and data layers, therefore, it requires the construction of a new network architecture.

It is possible and important to integrate SDN and NFV in the same network to accelerate the evolution toward software-enabled network services. NFV promotes the diversity of network functions offered by service providers [36]. SDN has been recognized as a key technology for the realization of a more efficient and flexible data center infrastructure, which enables network services to be dynamically deployed among shared NFV nodes. It is also a key driver for the development of edge computing and cloud computing. With the help of SDN, it becomes easier for the service provider to support diversified services with the most cost-effective resources.

1.2 Resource Allocation in Core Networks

As mentioned above, SDN and NFV are able to create more efficient and centralized networking management, reduce operational costs, and enable other new technologies. Thus, integrating SDN and NFV into the design of next generation networks remains a hot research topic in recent years. However, one important yet challenging problem of implementing SDN and NFV is how to manage and allocate the physical resources in an efficient and cost-effective manner, which is usually referred to as the NFV resource allocation (NFV-RA) problem.

Traditionally, network infrastructure are managed and maintained by different network operators independently. In this business model, to deploy a network service for each network operator, the data traffic needs to flow through a certain fixed group of middleboxes, such as firewall, DNS (Domain Name System), and IDS (Intrusion Detection System), in a predetermined order [37]. The problem of choosing the set of required middleboxes and deciding the corresponding routing paths is called "middleboxes orchestration." In an existing network, this task is manually executed through a "trial-and-error" approach, which

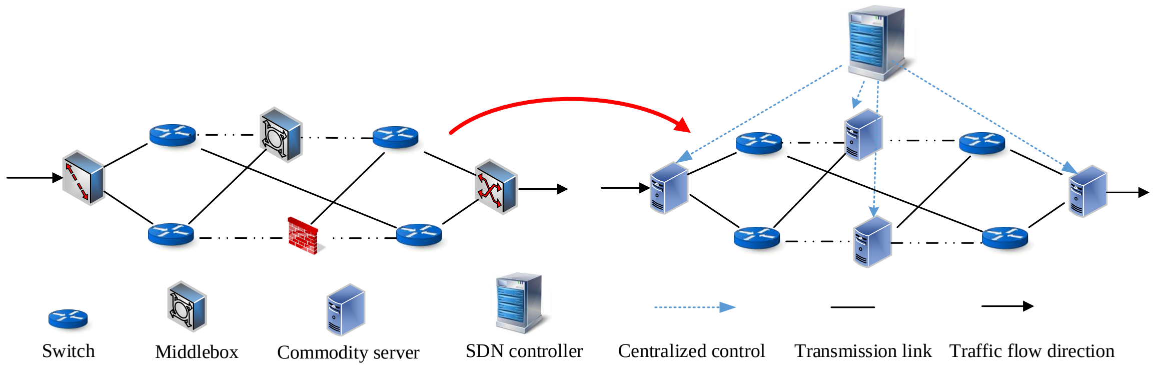


Figure 1.6: Global resource management and allocation via centralized SDN controller in core networks.

is very costly and time-consuming. Furthermore, once placed, it is costly and impractical to change the location of a middlebox according to the variations of network conditions.

In the context of SDN/NFV-enabled core networks, a network service can be formed with a set of chained virtual network functions (VNFs). Therefore, a network service is typically defined by the types of the VNFs chained, the processing order of the VNFs in the chain, and the embedding of the chain onto the network function virtualization infrastructure (NFVI). Fig. 1.6 illustrates the transition from traditional network to SDN/NFV-enabled core network. Note that the function-specific middleboxes are replaced by commodity servers to host VNFs. A logically centralized SDN controller, which has a global view of the whole network, is used to manage and allocate the resources in the network. By doing this, new services can be deployed by installing VNFs on commodity servers instead of adding new hardware devices to the network, leading to reduced CAPEX and OPEX. Such a network service has its specific time-varying and stochastic properties in terms of traffic volume and user distribution. Additionally, to accommodate the differentiated network services with diverse QoS requirements, heterogeneous resources (i.e., computing, storage and networking resources) need to be allocated to the services simultaneously and efficiently. There exist significant challenges in achieving a fast, efficient, and dynamic resource allocation scheme to accommodate multiple network services since efficient algorithms are desired to (1) compose the VNFs chain for a network service; (2) embed the chained network functions onto the substrate network (SN); and (3) schedule the VNFs of the network service to minimize the execution time and obtain improved performance. Through virtualizing network functions, new services can be accommodated by updating the software on the network servers instead of deploying new devices. In this way, the service provisioning cost can be reduced while the resource utilization can be improved. On the other hand, the use of SDN facilitates the realization of network openness and programmability. Despite the benefits brought by NFV and SDN, one of the main challenges in SDN/NFV-enabled core networks is, how to efficiently manage, allocate, and optimize the heterogeneous resources, including computing, storage, and networking resources, to the customized services for better QoS provisioning [38].

1.3 Research Motivations and Contributions

In the 5G era, new requirements arise for the evolving network paradigm, including 1) cost-effective network deployment for seamless device access, 2) enhanced resource utilization to accommodate high traffic volume, and 3) flexible function placement for service customization [3], [11]. However, the heterogeneity in services, resources, and access technologies increases the complexity of 5G networks. Therefore, heterogeneous resources should be intelligently integrated into 5G networks with efficient allocation. On the other hand, the centralized networking management offered by SDN and the increased service deployment flexibility provided by NFV create new opportunities for the development of resource allocation schemes for 5G networks. Therefore, resource allocation becomes one of the most crucial problems that calls for in-depth investigations for deploying SDN/NFV-enabled 5G networks. With efficient and dynamic resource allocation algorithms, the advantages obtained from existing hardware service can be maximized. In addition, optimally dynamic resource allocation helps the network to provide QoS-guaranteed services and accommodate services in a flexible, agile, and cost-effective manner.

The objective of this research is to develop efficient resource allocation schemes for SDN/NFV-enabled core networks to improve resource utilization, enable better QoS provisioning, and increase the network capability of accommodating customized services. Specifically, we will focus on the following three problems:

- In a single-service scenario, we investigate how to design the VN topology (i.e., reasonably allocate physical resources to the virtual nodes and virtual links) and embed the resultant VN onto substrate network to guarantee that the embedding cost is minimized and the QoS requirement can be satisfied. The objective is to find the optimal VN topology design and embedding solutions that can assign optimal physical resources for VNFs and virtual links so as to reduce the deployment cost and to increase the number accommodated services in the long run.

- In a multi-service scenario, we devise a game theoretical approach to address the multi-VNF chain embedding problem taking into account the additional latency increase introduced by processing-resource sharing and the node capacity constraints. The objective is to minimize the overall latency experienced by the traffic flow associated with each VNF chain so as to facilitate QoS-guaranteed service provisioning. An efficient embedding algorithm can not only find the optimal VNF placement and traffic routing strategies but also achieve fairness among NFV nodes to realize load balancing.

- In a multi-service scenario, we develop a reinforcement learning based VNF scheduling scheme to minimize the completion time of network services while satisfying their different E2E delay requirements. The original formulation of VNF scheduling falls into the class of NP-complete problems. To make it tractable, we reformulate it as an MDP and leverage a modified RL approach to solve it with reduced complexity and high accuracy. An efficient scheduling algorithm can minimize service completion time to improve the resource utilization.

1.4 Thesis Outline

The remainder of the thesis is organized as follows: Chapter 2 presents a comprehensive background and overview of related works of resource allocation in SDN/NFV-enabled core networks. We introduce the RA task from three aspects, i.e, VNF chain topology design, VNF chain embedding, and VNF scheduling. In Chapter 3, we jointly study how to design the topology of a VN and embed the resultant VN onto a substrate network with the objective of minimizing the embedding cost while satisfying the QoS requirements. In Chapter 4, a game theoretical approach is proposed to address the multi-VNF chain embedding problem. Then, two iterative algorithms are designed to find the NE of the proposed potential game. routes. In Chapter 5, the VNF scheduling problem is investigated to minimize the makespan (i.e., overall completion time) of all services, while satisfying their different E2E delay requirements. Then, a reinforcement learning based approach is developed to learn the best scheduling policy by continuously interacting with the network environment. Finally, Chapter 6 concludes the thesis and presents future works.

Chapter 2

Background and Literature Survey

In SDN/NFV-enabled core networks, a network service has its specific time-varying and stochastic properties in terms of traffic volume and user distribution. To accommodate the differentiated network services with diverse QoS requirements, heterogeneous resources (i.e., computing, storage and networking resources) need to be allocated to the services simultaneously and efficiently. Moreover, both the number and the types of network services will grow exponentially in the evolution of 5G networks. There exist significant challenges in achieving a fast, efficient, and dynamic resource allocation scheme to accommodate multiple network services. The resource allocation task (also referred to as network function virtualization resource allocation, NFV-RA) is implemented in three steps [32], i.e., given the traffic metrics, QoS requirements, and resource constraints of the substrate network, we first need to compose a VNF chain to form a virtual network (VN) topology. Then virtual resources allocated to each element (VNF or virtual link) need to be optimized in order to minimize the provisioning cost and to satisfy the QoS requirements. This process is usually referred to as *VNF chain topology design*. Next, we need to embed the virtual network (i.e., the VNF chain) onto the substrate network. During this stage, we need to assign the physical resources in an economical way to meet the resource demands of virtual nodes and links. This involves determining the locations of NFV nodes to host the VNFs and the routing from source to the destination. This process is referred to as *VNF chain embedding*. Finally, we need to schedule the VNFs for multiple services to minimize the service completion time and maximize the network performance. In other words, when multiple VNFs are embedded on the same NFV node, each of them should be assigned an optimal time slot. This process is referred to as *VNF scheduling* [39]. The main outcome for each step can be summarized as follows [32]:

1) VNF chain topology design: Given the traffic metrics, QoS requirements, and re-

quired VNFs of the service, we need to determine the required physical resources of the VNFs and virtual links, and chain them together to form a VNF chain; A optimized topology design can assign optimal physical resources for VNFs and virtual links to reduce deployment cost.

2) VNF chain embedding: Given the VNF chain and the resource constraints of the substrate network, we need to first determine the locations of NFV nodes onto which the VNFs should be placed, then determine the corresponding routing from source to destination in the NFVI. An efficient embedding algorithm can not only find optimal VNF placement and traffic routing strategy but also achieve fairness among VNFs embedded onto the same NFV node to realize load balancing.

3) VNF scheduling: When multiple VNFs are placed on the same NFV node, the execution time of each VNF needs to be scheduled to minimize the total execution time and achieve efficient resource sharing. An efficient scheduling algorithm can minimize service completion time to improve the resource utilization.

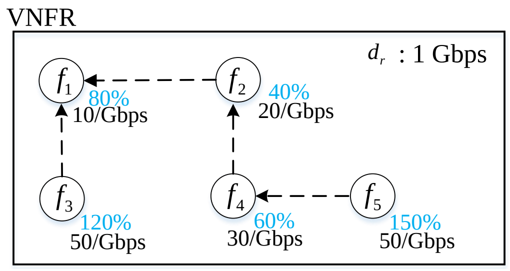
In what follows, we provide a detailed description of the three steps, along with some research challenges and existing approaches proposed in the literature.

2.1 VNF Chain Topology Design

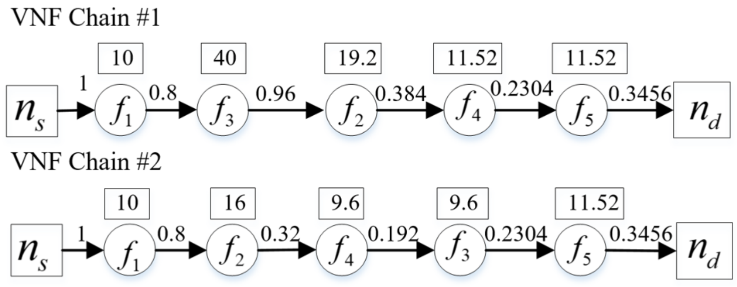
In the NFV environment, a network service request, or a VNF request (VNFR), is an entity composed by a number of VNFs in a predetermined order [29]. This means, the data traffic should pass through a given number of VNFs in a given order to complete the execution of the network service. Different from an existing network where network functions are run on dedicated hardware appliances which have fixed locations, VNFs are software with much higher deployment flexibility, leading to increased complexity in composing the VNF chains. Therefore, it is crucial to investigate the chain composition process so that VNF chains can be efficiently composed to deploy customized and dynamic NFV-enabled network services.

For a given network service request, the following information is typically known:

- Set of VNFs in the request and the dependencies between them;
- Fixed substrate nodes of initial and terminal points for the traffic flow, denoted as n_s and n_d, respectively;
- Initial data rate of the traffic flow, denoted as d_r;



(a)



(b)

Figure 2.1: Example of (a) a VNFR and (b) two possible VNF chain topology design results.

- The ratio of outgoing to incoming data rate of each VNF;
- The processing capacity demand per bandwidth unit of each VNF.

Fig. 2.1(a) shows an example of a VNFR, where the number of required VNFs is 5 and the initial date rate entering the chained functions is 1 Gbps. The percentage (indicated as blue) beside each VNF represents the ratio of outgoing to incoming data rate for each VNF. For example, when $d_r = 1$, the capacity demand for processing $f_1 = 10$ units. The outcome data rate of f_1 is 0.8 Gbps. The dependencies between multiple VNFs are denoted by dashed arrows. For example, the dashed arrow from f_4 pointing to f_2 indicates that f_4 must be processed after f_2. Two possible VNF chain topology design based on the VNFR information are shown in Fig. 2.1(b). The numbers on the links represent the bandwidth requirements (in GBps) while the numbers in the rectangles indicate the CPU requirements.

2.1.1 Problem Description

The task of allocating physical resources to a service request consists of two major parts, i.e., composing a VNF chain for the VNFR in an optimal way and embedding the resultant VNF chain onto the substrate network efficiently. Specifically, the following five aspects have to be taken into consideration:

1) Design the objective function for optimization

Let $C(G')$ represent the embedding cost of the VNF chain. $C(G')$ can be defined as follows [22]

$$C(G') = \alpha \sum_{l' \in L'} \sum_{l \in L} b_l^{l'} + \beta \sum_{n' \in N'} c(n') \tag{2.1}$$

where $b_l^{l'}$ represents the total amount of bandwidth resource allocated on the substrate link l for virtual link l', α and β are parameters to weigh the contributions of node and link resources to $C(G')$. To allow efficient resource sharing among customized services, $C(G')$ should be minimized.

2) Determine the order of VNFs in the chain

The inputs of the problem in the preliminary study are static VN topologies in which the nodes order are fixed and pre-determined. As the inputs are changed to network service requests composed of several VNFs. The order of the VNFs needs to be determined where the dependencies between any two VNFs should be respected.

3) Determine the optimal resources allocated to each VNF and virtual link in the VNF chain

Given the end-to-end delay requirement D, initial data rate entering the chained functions d_{init}, and the ratio of outgoing to incoming data rate $r(f_i)$ of each VNF f_i, we need to find the optimal physical resources allocated to each VNF and virtual link to minimize the overall embedding cost $C(G')$ while satisfying the delay requirement. This work can be solved together with VNF ordering.

4) Find the locations of NFV nodes to host the VNFs in the chain

The VNF chain embedding process need to consider both the host ability and resource constraints of each NFV node. During the node embedding process, we try to find a embedding $\mathcal{M}^N : N' \rightarrow N$ from the virtual nodes to substrate nodes, such that for all $n' \in N'$

$$\mathcal{M}^N(n') \in N$$

subject to

$$f^{n'} \in \mathcal{H}(\mathcal{M}^N(n')) \tag{2.2}$$

$$c(\mathcal{M}^N(n')) \geq c(n'). \tag{2.3}$$

Constraint (2.2) ensures that the VNF corresponding to n' can be hosted by the substrate node to which n' is embedded, while constraint (2.3) ensures that the CPU requirement of n' can be satisfied.

5) Determine the routing from source to destination, passing through all the deployed VNFs

In the link embedding step, one aims to find a mapping $\mathcal{M}^L$ from the virtual links to substrate paths, such that for all $l' \in L'$

$$\mathcal{M}^L(l') \in \mathcal{P}^s$$

subject to

$$b(\mathcal{M}^L(l')) \geq b(l') \tag{2.4}$$

Constraint (2.4) ensures that the bandwidth requirement of l' can be satisfied.

Some other constraints also need to be included in formalizing the optimization. For example, routing constraints have to be satisfied, which ensures that there is a path from the source to the first VNF, from each VNF to its consecutive VNF, and from the last VNF to the destination. Also, the delay constraint has to be satisfied, which guarantees that the total end-to-end delay does not violate the delay requirement. The delay analysis need to consider both the propagation and transmission delay over the links and the processing delay on the NFV nodes. In terms of the objective function, it should include the binary variables indicating whether a node is used to host a specific function, and the binary variables indicating whether a link is used to forwarding the data traffic between source and destination.

There are many factors that increase the complexity and hardness of the VNF chain topology design problem: First of all, the ordering of VNFs are flexible and has to respect the dependencies of the VNFs. Different ordering of VNFs may result in different physical resources requirements (which is related to the embedding cost) for the resultant VNF chain. This flexibility increases the searching space of the optimal VNF chaining. Secondly, traffic splitting may be required in situations where the residual physical resources of a node (or a link) are not enough to support the network service. In such scenarios, some VNF has to be instantiated on multiple NFV-enabled nodes and the data traffic has to traverse through different VNF instances running on multiple nodes. This will further increase the hardness of the VNF chain topology design problem. Thirdly, once the order of VNFs is determined, we need to determine the set of physical resources allocated to the VNFs and virtual links, to minimize the embedding cost while achieving QoS-guaranteed service provisioning. Lastly, as the physical resources of the substrate network is limited, it is also important to take the substrate network condition, more specifically, the substrate node locations, residual capacities and host abilities into the VNF chaining process to improve the overall network performance. This will make it even harder to obtain the optimal solution to the VNF chain topology design problem.

2.1.2 Related Works

As mentioned earlier, the performance of a network service relies on how its VNF chain is composed and in which order the VNFs are processed. However, most of existing studies neglect this stage of NFV-RA while focusing on the second and the third stages. One of the pioneering works that focuses on the VNF chain topology design problem is presented in [40]. The authors design a context-free language to formalize the requests for chaining several VNFs together. Then, a heuristic was used for choosing one of the possibilities to compose the VNF chain. The heuristic attempts to minimize the total data rate of the resultant VNF chain by assigning higher priority to VNFs with a smaller "inflation factor" (i.e., smaller ratio of outgoing to incoming data rate) when chaining the VNFs together. In this way, the total cost of deploying the service request is expected to be minimized.

Some previous studies have been conducted in literature that tackle with the VNF chain topology design and embedding together, such as those in [37], [40], [41], [42], [43], [44], [45], [46], [47]. For example, in [40], the VNF chaining and placement problem has been considered as an extension of VNE. The problem of finding the optimal placement of VNFs and chaining them together has been formulated as a Mixed Integer Quadratically Constrained Program (MIQCP) and solved accordingly. In [41], a coordinated approach which deals with the first two stages of NFV-RA has been proposed, where VNF chains are composed and embedded simultaneously. The resultant VNF chain from VNF chain topology design is prepared to optimize VNF chain embedding, leading to a higher probability of successful embedding. In [46], it has been shown how traditional VNE optimization models can be modified to cope with the virtual core network function placement and topology optimization. A novel ILP formulation has been presented with the objective of minimizing the cost of occupied link and node resources. In our work [37], a two-stage approach is proposed to jointly optimize the chaining and embedding of virtual network functions (VNFs), to obtain feasible composition and embedding results with low complexity, while the average embedding cost is minimized and the total revenue is increased. In the first stage, the VNF chaining order is optimized based on the location and functionality of substrate nodes, and the ratio of outgoing data rate over incoming data rate for each required VNF. In the second stage, we allocate the physical resources based on the preliminary

VNF ordering under the resource capacity constraints. A node splitting mechanism is also employed to improve the resource allocation fairness and increase the service acceptance ratio for the substrate network.

However, most previous studies on VNF chain embedding assume that the virtual resources allocated to the VNF instances and virtual links are pre-defined and further assume that QoS-guaranteed services can be provided once the given virtual resource requirements are satisfied. Without optimization, the physical resources required by each VNF chain may be redundant or insufficient, leading to increased embedding cost for the network operator or degraded QoS for end users. Therefore, to provide end users with guaranteed service quality in a cost-effective manner, VNF chain topology design and embedding should be solved in a combined way, while considering the optimization of the physical resources allocated to the VNF chain.

2.2 VNF Chain Embedding

Once the VNF chain is composed via the chain composition process, the input to the second stage of NFV-RA is obtained. The VNF chain embedding process is to place the VNFs on NFV nodes in the NFVI in the most suitable way, considering multiple network service requests. The VNFs should be consolidated in a way such that the physical resources are optimally allocated to the VNF chains with respect to a specific network operator's objective (e.g., maximizing the total revenue). The problem of embedding VNF chains onto the network function virtualization infrastructure is another challenging resource allocation task in the implementation of NFV. Through dynamic mapping VNFs onto physical hardware, the advantages obtained from existing hardware service can be maximized. This calls for algorithms to determine onto which substrate nodes in the NFVI the VNFs should be placed.

2.2.1 Problem Description

The left side of Fig. 2.2 shows three basic types of VNF chains. Note that they may be combined with one another to form more complex requests. The right side of Fig. 2.2 shows the physical infrastructure as well as an example of VNF deployment on it to fulfill a number of service requests [42].

VNF chain embedding shares many similarities with virtual network embedding (VNE), which can also be divided into two sub-problems: virtual node mapping and virtual link

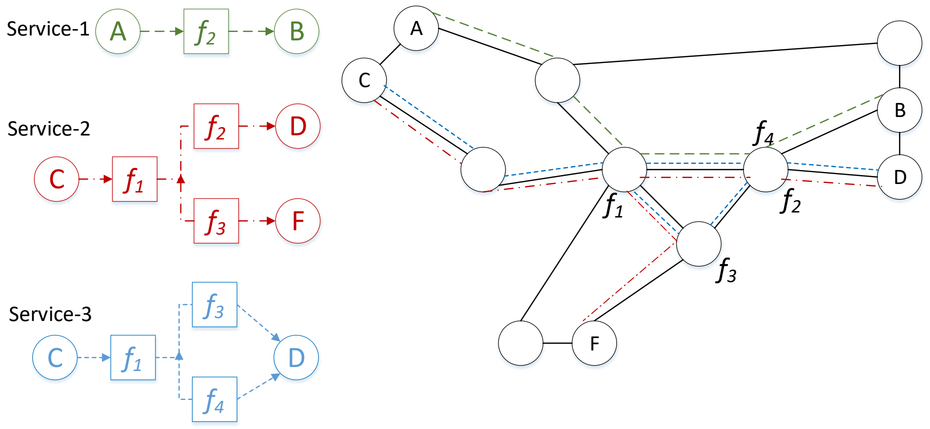


Figure 2.2: Illustration of VNF chain embedding, the left side shows three basic types of VNF chains, while the right side shows the physical NFV-enabled infrastructure and the VNF chain embedding results.

mapping [48]. Nevertheless, different from VNE where only the ability of processing and switching data packets are considered for all the substrate nodes, in VNF chain embedding, each substrate node can host a set of VNFs of different types. Since each VNF uniquely requires specific physical (computing, storage or networking) resources, a VNF can only be assigned onto a substrate node that has the ability to host it. This makes it harder to embed a VNF chain onto an NFVI than to map a virtual network onto a substrate network in the NV environment. Furthermore, the VNF chain embedding stage needs to deal with dynamic/changing networking condition and varying service description. That is to say, (1) With new services arriving and old services leaving the network, the existing VNF placement scheme may need to be modified to accommodate the newly coming service requests and improve resource utilization; (2) When the data rate entering an existing function chain changes, the orchestrator should notice that and trigger an adjustment algorithm to re-embed the corresponding VNF chain and re-allocate the physical resources to achieve better overall network performance. Lastly, as the middle stage of the NFV-RA problem, VNF chain embedding is closely related to the other two stages. It is particularly important to study VNF chain topology design and VNF chain embedding jointly, VNF chain embedding and VNF scheduling jointly, and even the three stages jointly to obtain the optimal resource allocation solution for the NFV environment, though the problem complexity would also greatly increases.

2.2.2 Centralized Approaches to VNF Chain Embedding

Due to the importance of VNF chain embedding in NFV, many research efforts have been devoted to the development of efficient VNF chain embedding algorithms. A centralized approach usually addresses VNF chain embedding by formulating the embedding process as an optimization problem [39]. The objective can be: 1) to maximize the number of accommodated VNF chains; 2) to minimize the average long-run embedding cost; and 3) to minimize the E2E latency of the flow corresponding to each VNF chain, to name a few.

For example, in [42], the VNF chain embedding problem has been formulated as an ILP with the objective of minimizing the number of VNF instances deployed onto the substrate network. In order to improve the scalability of their approach, a binary search based heuristic algorithm has been presented to quickly obtain feasible and high-quality solutions to the ILP model. In [45], VNF placement has been modeled as a combination of the facility location problem and the generalized assignment problem. The objective of their optimization model is to minimize the total system cost which includes the cost for function setup and the cost for traffic delivery. The linear relaxation technique is used to address the NP-hardness of the established ILP model. In [49], the VNF chain embedding problem has been formulated as an integer programming model with the objective of minimizing the latency.

While most of existing studies focused on the VNF chain embedding problem only, some other works address VNF chain embedding with other phases of NFV-RA in an coordinated manner. In [50], the problems of VNF chain topology design and embedding for multiple service requests are jointly studied with the objective of minimizing the total bandwidth consumption. A heuristic algorithm was proposed to coordinate the two processes using the feedback from mapping the key sub-modules of a VNF chain. In [43], a coordinated approach has been studied to jointly optimize VNF chain composition, embedding, and scheduling, where the former two phases are jointly formulated as an MILP with the objective of minimizing the total cost, while the last phase is addressed by a one-hop scheduling algorithm with low-complexity.

The existing centralized approaches face the challenges of how to deal with the computational complexity which increases exponentially as the network size expands. Moreover, the availability of a centralized entity collecting the information of all SFCs existing in the entire network needs to be stated. In reality, different SFCs could be controlled by different operators from different network domains, which motivates researchers to rethink the SFC embedding problem and investigate the possibility of performing SFC embedding in a distributed manner, as will be discussed in the next section.

2.2.3 Distributed Approaches to VNF Chain Embedding

Among all the distributed approaches for SFC embedding, game theoretical approach is the most widely adopted one due to its great potential to model the strategic interaction among a number of decision-makers. Recently, some research activities have been carried out to develop game theoretical approaches for SFC embedding [52]-[54].

In [52], the process of NFV servers providing network functions to users has been modeled as a two-stage Stackelberg game, where the servers and the users are considered as leaders and followers of the game, respectively. Similar to [52], the authors of [57] have leveraged a mixed-strategy gaming approach to facilitate SFC provisioning over inter-datacenter elastic optical networks, where resource brokers and users play the leader game and the follower game, respectively. In [55], the work in [57] has been extended from a single-broker scenario to a multi-broker scenario. In [56], the SFC embedding problem has been envisioned as a cooperative graph partitioning game and a heuristic algorithm has been designed to achieve the NE corresponding to the optimal solution.

On the other hand, the SFC embedding problem has also been formulated as a congestion game in [51]-[54]. D'Oro *et al.* first modeled the SFC embedding problem as a weighted congestion game in [51]. The game has been proved to be a weighted potential game which admits at least one pure strategy NE. Following the study in [51], Bian *et al.* have proved in [53] that, with the consideration of user and resource failures in the NFV system, the weighted congestion game is still a weighted potential game whose NE can be obtained with their proposed distributed algorithm. In [54], Le *et al.* have developed a congestion game with player-specific (CGPS) utility functions which has been proved to be a weighted potential game. They then applied CGPS to model the SFC embedding problem where different traffic flows have different priorities.

Some machine learning based approaches that adapt to dynamic environment and uncertain network conditions have also been proposed in the literature [106], [58]. For example, in [58], a decentralized optimization framework that addresses multi-domain SFC embedding for wireless networks has been presented. The framework exploits a recently reported Alternating Directions Dual Decomposition (AD^3) algorithm to achieve consensus among different operators quickly with guaranteed convergence and stability.

2.2.4 Processing-Resource Sharing

The E2E delay requirement is one of the most important metrics for NFV-enabled service provisioning in next generation communication networks. Under this consideration, some approaches have been developed to achieve latency-aware VNF chain embedding from different aspects. In [44], the latency for a given traffic flow is calculated from the aggregate propagation delay of all the links the traffic passing through. The network OPEX then consists of a penalty for service level objective (SLO) violation, i.e., if a traffic experienced more than the maximum allowed propagation delay. A similar way to deal with the latency constraint in the VNF chain embedding context can be found in [40] and [59]. In [42], the propagation delay of links and the processing delay of network functions are considered as constants. Both of the delays are then involved in the formulation of VNF chaining and placement to ensure that the end-to-end latency constraints of a specific service is satisfied. As an extension to [46], the work in [60] include processing, queueing as well as propagation delays into the optimization of virtual mobile core network function placement and topology design.

Almost all the existing VNF chain embedding approaches neglect one important impact on the latency of a service caused by placing multiple VNFs on the same substrate node. Due to processing-resource sharing, additional latency penalties are introduced to the execution of network services which have multiple VNFs running on the same NFV-enabled node. This impact is not desired as services have their specific latency requirements to be satisfied. Thus, the impact of processing-resource sharing on VNF placement needs to be evaluated and taken into consideration when dealing with the VNF chain embedding problem.

Compared with its single-service counterpart, the VNF chain embedding problem in a multi-service scenario has additional challenges. One practical yet challenging aspect of multi-service VNF chain embedding is to take into consideration the impact of processing-resource sharing on VNF placement. Suppose that every NFV-enabled node is equipped with a multi-core CPU and is able to run multiple VNFs (either in the same or in different service chains) simultaneously. Then multiple services can share the processing resources of a same NFV node by placing a VNF to the same NFV node. In such a scenario, signaling overhead among different VNFs and synchronization overhead among different cores may result in additional latency for each service.

Efficient management of physical resources and control of flow latency are two of the most crucial yet challenging tasks for SDN/NFV-enabled 5G networks. Placing multiple VNFs on the same NFV node equipped with multi-core CPU may result in performance degradation due to processing-resource sharing. The degradation effects reflect in an increase in the latency caused by the NFV node and a reduction of the actual processing resources that can be used to host VNFs of the NFV node. Therefore, to manage the physical resources efficiently as well as to achieve better latency control of flows, the degradation effects mentioned above should be taken into account when solving the VNF chain embedding problem.

2.2.5 Challenges of Multi-VNF Chain Embedding

The embedding of multiple services faces three challenges: (1) The impact of processing-resource sharing on the VNF chain embedding process is difficult to be evaluated. Two types of penalties related to the processing-resource sharing among multiple VNFs can be identified, i.e., the *context switching costs* and the *upscaling costs* [61]. The former is due to multiple VNFs running on the same NFV node while the latter is caused by performing load balancing for one VNF running on a NFV node with multi-core CPU. In [61], the context-switch costs are modeled as linearly related to the number of VNFs sharing the NFV node, and the upscaling costs are modeled as linearly related to the number of CPU cores among which the traffic is balanced. However, the linear relationship may not be a practical assumption. To estimate the two types of costs accurately, a more advanced modeling method needs to be investigated. The model will be utilized to help formalize the VNF chain embedding problem; (2) The objective function of the VNF chain embedding problem should be carefully designed. In [61], the paper minimizes the number of active NFV nodes in deploying a network service. However, by minimizing the number of active NFV nodes, the embedding cost can be reduced but the additional latency penalty is also increased. Thus, there is a trade-off between reducing the number of active NFV nodes and satisfying the latency requirements. Moreover, simply minimizing the number of active NFV nodes may lead to unbalanced load among the NFV nodes in the network. Therefore, the objective function should be devised in a way that by minimizing the objective function one can achieve load balancing among NFV nodes and reduce the embedding cost while satisfying the latency requirement; (3) The optimization problem can be formulated as an ILP/MILP model, which is NP-hard. To obtain a suboptimal solution in a shorter time, heuristic algorithms need to be designed. The performance of the heuristic algorithms should be comparable with the ILP/MILP model when evaluated in various network scenarios.

2.3 VNF Scheduling

The last stage of NFV-RA is the VNF scheduling process. Given the embedding results of VNF chains, this process is to find a proper scheduling of VNFs' execution on each substrate node to minimize the "makespan" (i.e., the final completion time of the last VNF of the last executed network service). The results of a VNF scheduling process is closely related to the VNF chain embedding stage. The embedding process should aim to maximize the performance of the VNF scheduling stage. Therefore, it is desired to solve the VNF chain embedding and VNF scheduling problems in a coordinated way. Alternatively

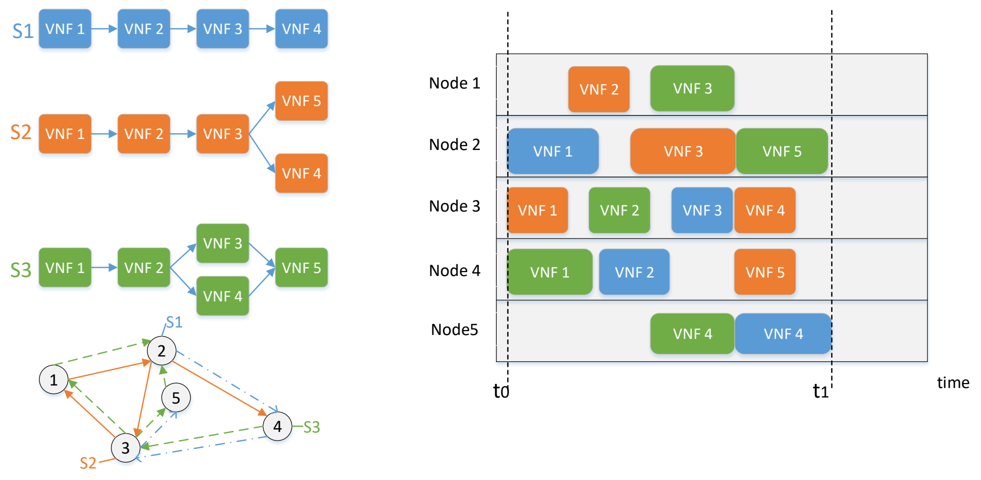


Figure 2.3: Illustration of static VNF scheduling.

for simplicity, the two problems can also be solved independently where the results of VNF chain embedding are taken as the input of VNF scheduling [62].

2.3.1 Static VNF Scheduling

Fig. 2.3 illustrates an example of VNF scheduling. There are three network services composed of several VNFs. The processing of S_1, S_2, S_3 begin immediately on their arrivals at time t_0 at node 2, node 3 and node 4, respectively. Thereafter, the processing of each of the subsequent functions in a network service cannot start until the processing of its preceding function is finished. The processing of S_1 ends when its last function has been processed at time t_1 at node 5. The total flow time of S_1 is given by $t_1 - t_0$, and is equal to the sum of processing times of the VNFs at multiple substrate nodes. The VNF scheduling process is to determine the time slots for the VNFs of all network services to be executed over a given set of servers.

The VNF scheduling process is related to the embedding process. The embedding should maximize the performance of the VNF scheduling stage. To illustrate, if we minimize the flow time of network services as one objective in the embedding stage, the service delays due to queuing for both processing and transmission can be minimized [43], [63]. To minimize the flow time of network services and achieve maximum resource utilization, the embedding and VNF scheduling problems should be jointly considered.

2.3.1.1 Related Works

VNF chain embedding problems are studied extensively to either maximize the number of accommodated services or minimize the average long-run embedding cost under the assumption that each NFV node can support multiple network functions [32], [42], [43]. When multiple VNFs are embedded onto an NFV node, a new and challenging research issue is how to properly schedule the embedded VNFs to minimize the overall completion time of packets processing for all service requests. Compared with the research on VNF chain embedding, the investigations focusing on VNF scheduling remain limited.

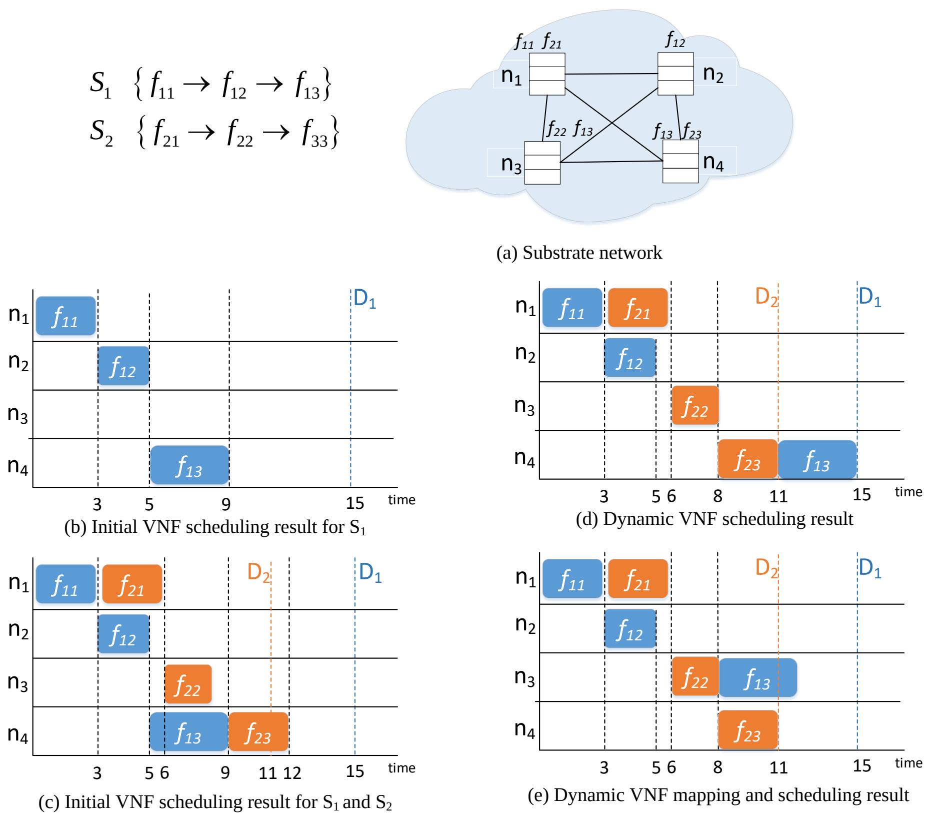
As the first study on VNF scheduling, Riera *et al.* formulate a joint VNF assignment and scheduling problem as a job-shop problem (JSP) which can be solved in two separate stages [64]. Another seminal work on VNF scheduling can be found in [63], where VNF chain embedding and VNF scheduling are heuristically solved in a coordinated way. The authors introduce the time to execute each VNF for a specific service and include a constraint that forbids a substrate node to execute more than one VNF at a certain time as part of the embedding phase. Three greedy heuristic algorithms and a tabu search meta-heuristic algorithm are presented to solve the problem with the objective of minimizing the total flow time (i.e., the final execution time of the last VNF in the last executed network service). On the other hand, an uncoordinated approach solves VNF chain embedding and VNF scheduling separately. In the VNF chain embedding stage, the goal is to find where to allocate the VNFs in each network service in the NFVI in a suitable way, without considering the optimality for the scheduling process. The embedding results are then considered as an input to the VNF scheduling stage. Then, in the VNF scheduling stage, the goal is to find the time slots for the VNFs composing different network services to be executed on the servers to which they have been assigned. A good example of such approach can be found in [62], where the VNF assignment sub-problem is first solved to allocate VNFs of each network service to substrate nodes. Then, a genetic algorithm (GA) based heuristic algorithm is presented to obtain local optimal solutions, without QoS consideration. Wang *et al.* present a one-hop greedy scheduling algorithm, which is part of the NFV-RA optimization framework [43]. In [65], Alameddine *et al.* study the deadline-aware VNF scheduling problem and present an MILP formulation for the joint problem of VNF chain embedding, traffic routing, and VNF scheduling based on a discrete-time model. To address the MILP, a tabu search-based heuristic algorithm is provided to obtain the near-optimal solutions. In [66], Pham *et al.* present a matching game based approach to address deadline-aware VNF scheduling. Although the existing heuristic algorithms can help to find local optimal solutions, the optimality gaps of the algorithms are difficult to validate, and the delay requirements are yet to be taken into account.

Recently, the reinforcement learning (RL) approach has been applied to solving combinatorial optimization problems [67], [68], including standard JSP [69]-[71]. Specifically, in [69]-[71], various multi-agent RL algorithms have been presented to solve the JSP problem in different scenarios, where a centralized control of network states is hard to be instantiated. The job scheduling problem has been formulated as a decentralized MDP with changing action sets, and then been solved in a distributed manner with inter-agent coordinations. However, the optimal scheduling results may not be achieved with only local network information, as the restrictions imposed on the learning agents degrade the accuracy of the solutions [72]. Also, when E2E delay constraints are considered in dealing with VNF scheduling, both the makespan evaluation and the delay constraints verification require the status information of the VNFs in the whole system. For decentralized learning, each agent only has a local view of the system and thus may not optimize the makespan with satisfied delay constraints well. Introducing inter-agent coordination to exchange necessary information among all the agents can introduce some overheads into the learning process. To address this issue, in our work [73], we leverage the SDN/NFV-enabled network architecture, where global network state information is collected to enhance the performance of VNF scheduling. By introducing a single scheduling agent placed in the SDN control module, the agent is able to make network-wide scheduling decisions and learn the optimal VNF scheduling policy. Moreover, a specific reward function is designed to incorporate the E2E delay requirements of diverse services into the VNF scheduling process.

2.3.2 Dynamic VNF Scheduling

A simple illustration of the dynamic VNF mapping and scheduling process is shown in Fig. 2.4 [74]. Suppose that there are three VNF chains of network services S_1, S_2, S_3 (labeled by different colours) to be scheduled. On each NFV node, the VNF scheduling sequence has to be determined to minimize the makespan. The VNF chain embedding results are shown in Fig. 2.4(b). In Fig. 2.4(c), packet batches start traversing the VNFs of S_1, S_2, and S_3 at nodes n_1, n_2, and n_3, respectively, from time instant 0. Packet batch processing time at each scheduled VNF is also displayed by different colours. Time instant t_1 indicates the makespan of one VNF scheduling cycle.

In the form of a sequence of VNFs, service requests dynamically arrive at the system with a set of resource and E2E delay requirements. The substrate network is limited in resources, including processing capacity, buffer capacity, etc. Through dynamic VNF instantiation, we aim to find the optimal VNF mapping and scheduling strategies for the services to maximize the acceptance ratio of service requests and service provider's profit in the long run.



(a) Substrate network

(b) Initial VNF scheduling result for S_1

(c) Initial VNF scheduling result for S_1 and S_2

(d) Dynamic VNF scheduling result

(e) Dynamic VNF mapping and scheduling result

Figure 2.4: Illustration of dynamic VNF scheduling.

Fig. 2.4 shows an example of dynamic VNF mapping and scheduling process containing two service requests, i.e., $S_1 = \{f_{11} \to f_{12} \to f_{13}, D_1\}$ and $S_2 = \{f_{21} \to f_{22} \to f_{23}, D_2\}$. Fig. 2.4(a) shows the capabilities of different nodes to support VNFs, in which n_1 can support f_{11} and f_{21}, n_2 can support f_{12}, n_3 has the capability to support f_{13} and f_{22}, and n_4 can support f_{23} and f_{13}. S_1 arrives at the system at time $t = 0$ with E2E delay requirement $D_1 = 15$ (time units), while S_2 arrives at the system at time $t = 3$ with E2E delay requirement $D_2 = 8$ (time units). The initial VNF mapping for S_1 at $t = 0$ is shown in Fig. 2.4(b). S_2 arrives at $t = 3$ with initial VNF mapping results shown in Fig. 2.4(c). According to the VNF mapping results in Fig. 2.4(c), the service completion time of S_2 is $t = 12$, which violates the delay requirement. Therefore, S_2 will be rejected in the static VNF mapping and scheduling algorithms such as those in [65] and [63]. For dynamic VNF mapping and scheduling considered in this paper, when the initial scheduling for S_2 is failed, the proposed Tabu search-based algorithm is triggered to remap and/or reschedule the existing VNFs in the system. An example of VNF rescheduling strategy is illustrated in Fig. 2.3(d) (For illustration purpose, additional delay incurred by VNF re-instantiation is not shown explicitly.). After switching the processing sequence of f_{13} and f_{23}, which are both mapped onto server n_4, the delay requirements of S_1 and S_2 can be satisfied simultaneously. Fig. 2.3(e) illustrates an example of VNF remapping, where f_{13} of S_1 is re-instantiated on n_4. As a result, f_{13} and f_{23} can both start processing at $t = 8$. Additionally, VNF remapping reduces the completion time of S_1, compared with that of VNF rescheduling only solution in Fig. 2.3(d).

Clearly, by performing dynamic VNF mapping and scheduling, we are allowed to 1) increase the acceptance ratio of the substrate network to accommodate more services, thus obtain a higher profit; 2) potentially reduce the completion time of the customized services, which enhances the QoS; and 3) balance the traffic load among the NFV nodes in the system.

2.3.2.1 Related Works

Recently, dynamic VNF mapping approaches that allow VNF remapping (also referred to as VNF migration) mechanisms have emerged as a hot topic in the literature [75]-[81]. For instance, in [76], the authors highlight that the E2E delay of services will be affected once a VNF is migrated (or re-instantiated) from one hosting server to another. In addition, they establish a mathematical model to help determine if the VNFs in a given VNF chain should be migrated or re-instantiated for a better mapping strategy with shorter E2E delay. In [80], the problem of dynamic VNF chain deployment and readjustment is investigated to maximize the service provider's profit. In [81], a hybrid service provisioning algorithm that combines an online heuristic algorithm and an offline mixed integer program based optimization has been presented. The online heuristic algorithm maps VNF chains greedily to achieve fast deployment of network services while the offline optimization is running in the background to periodically re-optimize the mappings of several VNF chains at once. Nevertheless, none of these works considers the problem of VNF rescheduling after remapping existing VNFs onto new NFV nodes, which makes their studies different from ours. How to readjust the current mapping and scheduling strategies together to achieve enhanced performance of service provisioning in an online scenario has not been fully investigated yet. In our work [74], dynamic VNF mapping and scheduling are jointly investigated. Specifically, to achieve load balancing with QoS guarantee, we first formulate the VNF mapping and scheduling problem as a mixed integer linear programming (MILP). We then propose a two-stage online algorithm to address the NP-hardness of the MILP. In particular, when new service arrives, we map and schedule the VNFs on a VNF chain by greedily minimizing the waiting time of VNFs. If the delay requirement cannot be satisfied after the first stage, a delay-aware rescheduling scheme is triggered, in which selected existing VNFs are remapped and rescheduled. The proposed dynamic approach achieves more flexible function placement and higher service acceptance ratio in comparison with its static counterpart.

Chapter 3

Joint VN Topology Design and Embedding for Service-Oriented Core Networks

Network virtualization is a promising technology to efficiently allocate and manage heterogeneous resources for service customization in fifth generation (5G) core networks. In this chapter, we study how to jointly determine the virtual network (VN) topology, composed of virtual nodes and virtual links, and embed the resultant VN onto a physical substrate network. An optimization problem is formulated to minimize the VN embedding cost while satisfying the end-to-end (E2E) packet delay constraints for different service requests. The E2E packet delay for each service is evaluated as a function of resources allocated to the embedded virtual nodes and virtual links, by employing the network queueing theory. The formulated problem is a mixed integer nonlinear program (MINLP) with binary and continuous variables, nonlinear objective, and nonlinear constraints to be dealt with. An enhanced brute-force search algorithm is proposed to obtain the optimal solutions. To reduce the computational complexity and improve the scalability of the algorithm, we further propose a low-complexity heuristic algorithm to determine a set of near-optimal solutions for large-scale networks. Simulation results demonstrate the effectiveness of both algorithms and indicate that the low-complexity heuristic algorithm can reduce the running time significantly.

3.1 Introduction

The 5G networks are envisioned to accommodate a proliferation of connected devices and diversified services, with largely increased traffic volume and differentiated quality-of-service (QoS) demands [4]. Network virtualization (NV) and software-defined networking (SDN) hold great potentials to realize service-oriented networking in both wireless and wired domains [25], [82], [83]. One of the fundamental research issues in NV is how to determine the amount of physical resources allocated to each VN (*VN topology design*) and embed the resultant VN onto a substrate network (*VN embedding, VNE*) [17], [18]. Specifically, given traffic statistics, QoS requirements of different services, and physical network resource constraints, it is desired to compose virtual nodes and virtual links to form a VN topology. The physical resources allocated to each virtual node and virtual link need to be optimized to reduce the potential embedding cost with differentiated service requirements being satisfied. Then, we need to embed the VN onto the substrate network in an economical way while satisfying the resource requirements of virtual nodes and virtual links. Many existing works study the VN embedding (VNE) problem considering two objectives: 1) to minimize the VN embedding cost or to maximize the number of accommodated VNs in the long run, and 2) to provide QoS-guaranteed services by satisfying the resource requirements of virtual nodes and virtual links specified by VNOs. Most of the studies assume a VN topology with predefined resource demands on virtual nodes and virtual links [20], [84]-[89]. However, resource allocation (RA) for each VN is not optimized for customized QoS requirements. To improve network resource utilization and provide end users with guaranteed service quality in a cost-effective manner, the VN topology design and VN embedding should be jointly studied, where the amount of resources allocated to each virtual node and virtual link are determined for each VN to satisfy differentiated E2E QoS requirements.

To support time-critical services in 5G core networks, E2E delay modeling for traffic passing through each embedded VN needs to be considered. Most existing approaches only take into account link propagation delay for E2E delay modeling, while neglecting the queueing delay on each virtual node [90]-[92]. Therefore, a more comprehensive analytical E2E delay model is required to optimize the resource allocation for the VNE problem. In this chapter, we study a QoS-guaranteed VNE problem in which the packet-level E2E delay is considered as the QoS metric. Both traffic routing configuration and resource allocation for each VN are jointly optimized in the problem. The objective is to minimize the embedding cost while satisfying the E2E delay requirements of different services and the resource constraints of the physical network. Towards this objective, we present an NV-based network architecture to support E2E service deliveries in the 5G core networks, where the data traffic from different access networks are aggregated and grouped by traffic aggregation points (TAPs). Based on the NV architecture, we formulate a QoS-guaranteed VNE problem as a mixed integer nonlinear program (MINLP), where the E2E packet delay of each service is analyzed as a function of the amount of resources allocated to the VN. The main contributions of this chapter are four-folded:

1. Existing VNE problem formulations only consider the VN to physical network mapping without optimizing the physical resources set allocated for each VN. Different from those formulations, we formulate the joint VN topology design and embedding problem as an MINLP to solve the VN resource allocation and VN embedding problems together. In this way, the resource utilization is improved and differentiated QoS requirements are guaranteed.

2. We design a cost function for utilizing physical resources to balance the load over the network in the long run. By using the designed cost function, substrate nodes and links with scarce resources will be less likely chosen in the VN resource allocation and embedding process.

3. We propose an enhanced brute-force search algorithm to solve the MINLP for optimal VN embedding solutions. To improve the scalability in large-scale substrate network scenarios, a low-complexity heuristic algorithm is further designed to obtain near-optimal solutions with low computational overhead.

4. Compared with propagation-only delay models, we analyze the E2E packet delay using the queueing network theory, in which both packet processing delay on each virtual node and packet transmission delay over each virtual link are considered. The proposed analytical E2E packet delay model is used as a constraint in the VNE problem to achieve QoS-guaranteed embedding results.

3.2 Related Works

Network virtualization is one of the key technologies for the development of future service-oriented 5G core networks. As an essential part of NV, VNE tackles the problem of virtual resource allocation in both nodes and links. Therefore, it naturally consists of two sub-problems: virtual node mapping and virtual link mapping. The former deals with how to embed virtual nodes on physical infrastructures while the latter focuses on how to embed the virtual links between virtual nodes on physical paths. The node mappings are usually achieved by employing greedy methods, while the link mappings are typically obtained using k-shortest paths and multi-commodity flow algorithms [18].

Many research works have been carried out on the VNE problem in different scenarios. A comprehensive survey can be referred to as in [18]. Existing studies on VNE can be classified into two main categories, i.e., the uncoordinated ones [20], [84]-[86] which solve the two sub-problems separately, and the coordinated ones [21], [87]-[89] which solve the two sub-problems in one stage. In [20], Yu *et al.* have investigated the VNE problem with the consideration of path splitting and path migration to increase the VN acceptance ratio. For tractability, the formulated VNE problem has been divided into node embedding and link embedding sub-problems which are solved independently. Later, several node ranking approaches based on topological attributes have been developed to improve the quality of node mapping solutions [84]-[86]. The lack of correlation between node and link embedding may increase the embedding cost and lead to a low VN acceptance ratio. To tackle with this issue, the dependency between the two sub-problems has been considered [21],[84]-[89]. For example, in [87], the authors have conducted in-depth investigations on the complexity of the location-constrained (LC-VNE) problem. By using graph bisection, it has been proved that LC-VNE is NP-complete. With the concept of compatibility graph (CG), two heuristic algorithms have been designed to facilitate coordinated node and link mapping. In [89], a particle swarm optimization (PSO)-based coordinated VNE algorithm has been presented. The algorithm exploits a step-by-step updating rule of particle positions to allow a combination of node and link mapping, outperforming the existing two-stage approaches.

In all the aforementioned VNE problem formulations, the set of physical resources required by a VN is predefined, and it is assumed that the QoS of each service can be guaranteed with the allocated physical resource set. This makes these existing works different from our study, in which the set of physical resources allocated to each VN is optimized. Moreover, to achieve QoS-aware VN embedding, we establish an analytical model between the QoS metric (i.e., E2E packet delay) and the resources allocated to each VN, which is then incorporated in our VNE problem formulation.

On the other hand, there exist some VNE formulations that consider the E2E packet delay of each service as constraint functions [90]-[92]. For example, in [90], the VNE problem has been investigated in the context of a multicast service-oriented network with the consideration of delay and jitter constraints, and a sliding window-based heuristic algorithm has been devised to solve the problem. In [23], the authors have studied the VNE problem for evolving networks where reconfiguration of the virtual network infrastructure is allowed. The problem is formulated as an integer program with delay constraints, and is then solved by a heuristic algorithm that greedily migrate virtual nodes if the delay constraint is violated. In [24], a VNE optimization framework has been established to minimize the E2E packet delay in mobile networks taking into account user mobility. However, most existing approaches only consider the link propagation delay when analyzing

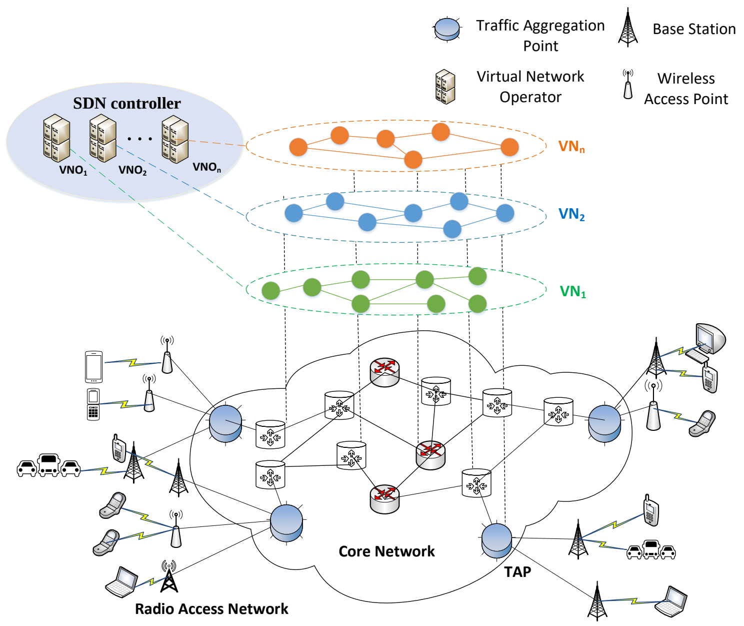


Figure 3.1: The NV-based network architecture to support E2E service deliveries in 5G core networks.

the E2E packet delay. In this study, to achieve a comprehensive delay modeling, we also take into account the packet processing delays on each virtual node and transmission delay on virtual links, and propagation delay. Moreover, we use the E2E packet delay model as a constraint in our problem formulation to achieve delay-aware VNE. To sum up, existing VNE studies either assume predefined physical resources required by a VN or only consider the link propagation delay when analyzing the E2E packet delay for a VN. How to jointly solve the VN topology design and VN embedding to achieve E2E delay guaranteed service provisioning needs further investigation.

3.3 System Model

3.3.1 Network Architecture

We consider an NV-based network architecture to support E2E service provisioning in the 5G core networks, as shown in Fig. 3.1. Network resources are sliced for different VNs to achieve service customization and improves the physical resource utilization through inter-VN resource sharing. The E2E network architecture consists of 1) radio access networks (RAN), where end devices connect to different wireless access points (WAPs) with their traffic aggregated at traffic aggregation points (TAPs) [46], and 2) the core network, where VNOs slice the physical network resources to create different VNs for customized services and the SDN controller orchestrates the network-wide resources. For RAN, we consider the inter-working of heterogeneous wireless access technologies, e.g., long-term evolution (LTE), WiFi, direct short range communications (DSRC), where different WAPs are deployed to support massive device access. In the core network, each VN is managed by a VNO and any dynamics of the VN do not influence other VNs. In this way, the resource management for each VN becomes more flexible as VNs are conceptually isolated for customized QoS guarantee, and are managed by the associated VNOs.

3.3.2 Traffic Aggregation and Traffic Grouping

For E2E service deliveries, traffic from heterogeneous RAN are aggregated at TAPs deployed between WAPs and the edge of the core network (CN), as shown in Fig. 3.1. Each TAP is connected to one edge node in the CN. All traffic aggregated at a TAP is forwarded to the CN through the edge nodes connected to it. After traffic aggregation, traffic grouping is performed by each TAP based on the destination edge nodes in the CN. The packets belonging to the same service type and the same source-destination edge node pair is grouped as one *traffic flow* (or *one service flow*).

3.3.3 Service Request

After traffic aggregation and grouping, multiple service requests are formed in different VNs of different VNOs. A service request consists of a source-destination (S-D) node pair, traffic arrival rate at the source node, and E2E packet delay requirement. We consider that a certain amount of CPU processing/bandwidth resources are allocated to each virtual node/link to process/transmit the data packets. The optimal set of physical resources allocated to the virtual nodes and virtual links can be determined through the minimization of the embedding cost for leasing physical resources from InPs, while satisfying the E2E packet delay requirement of each service. We give two service request examples at the top of Fig. 3.2, where service request #1 is with a delay requirement of 10 ms and an arrival data rate of 150 packets/s, while service request #2 is with a delay requirement of 20 ms and an arrival data rate of 250 packets/s.

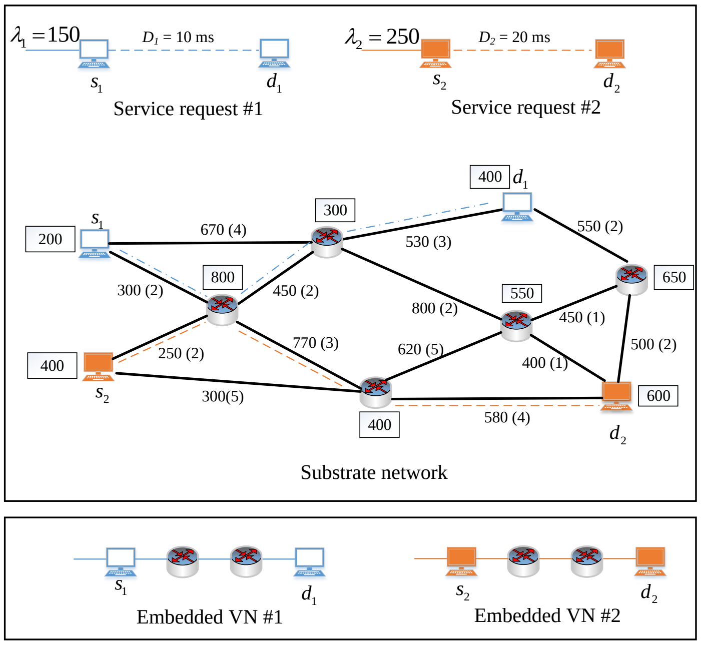


Figure 3.2: Illustration of the substrate network and two E2E VN requests. The joint VN topology design and embedding problem includes determining the set of physical resources for the VN and embedding the resultant VN onto the substrate network.

3.3.4 Substrate Network

We describe the substrate network (i.e., the CN) as an abstracted undirected graph, denoted by $G = (N, L)$, where N is the set of physical nodes (i.e., network servers/switches), and L is the set of undirected physical transmission links. All the physical nodes have the ability of processing and forwarding data packets. Let C_n denote the available CPU processing rate for each node $n \in N$. Let B_l represent the available transmission rate for each link $l \in L$. Each link has a constant propagation delay δ_l. The set of neighboring nodes of node $v \in N$ is denote by $\mathcal{N}_v$, where node u is in $\mathcal{N}_v$ if the direct link $l = (u, v)$ between u and v exists in L. In Fig. 3.2, we show an example of a substrate network, where the numbers over each link represent the available link bandwidth and the constant propagation delay (the number in brackets), respectively. The numbers beside the physical nodes represent the available CPU processing rates.

3.3.5 Embedded VN Topology

An embedded VN topology is obtained after embedding a service request onto a substrate network. Denote an embedded VN by $G' = (N', L')$, where $N' \subset N$ is a set of physical nodes used to operate (embed) the virtual nodes in the VN, and $L' \subset L$ contains all the links connecting the embedded virtual nodes. We consider single-path traffic routing between embedded virtual nodes of each VN. The S-D node pair for a VN indicates the physical nodes where a traffic flow starts and ends in the core network. We denote the source and destination nodes of a VN as $s \in N'$ and $d \in N' \setminus \{s\}$, respectively. Let λ and D be the arrival data rate and the E2E delay requirement of the VN, respectively. At the bottom of Fig. 3.2, we show an example of two service requests being embedded onto the substrate network. The circles in each embedded VN represent the intermediate network switches used to forward the data traffic from source to destination nodes.

3.4 Problem Formulation

Given traffic statistics and service demands supported by each VN, how to determine the amount of resources allocated to virtual nodes and virtual links of the VN, and embed the resultant VN onto the substrate is referred to as the QoS-guaranteed VNE problem. The formulation should take into account the physical resource constraints, the routing constraints, and the E2E packet delay constraints. In addition, the coupling of VN topology design and VN embedding should be considered, as the embedding results affect the amount of available physical resources that can be allocated to each VN. In this chapter, we consider that the service requests are accommodated in a First-Come-First-Serve (FCFS) manner. Therefore, the problem formulation for different service requests remains in the same form. In this section, we provide the formulation of the joint problem for a single service request. Table 3.1 provides a summary of important symbols that are used in the following formulation. Table 3.2 summarizes a list of decision variables for the problem.

To formulate the problem, we first identify the set of constraints that need to be considered, including physical resource constraints, traffic routing constraints, and E2E delay constraints. The amount of bandwidth resources, $B_l^{'}$, allocated to link $l \in L'$ of a VN is upper bounded by B_l. That is,

$$B_l' \leq B_l, \quad \forall l \in L'. \tag{3.1}$$

As traffic splitting is not considered and all the nodes are without traffic inflation and deflation, the link bandwidth demands for any $l \in L'$ are assumed the same. We denote

Table 3.1: Summary of Important Symbols

Notation	Description
C_n	Available CPU processing rate of node n
B_l	Available bandwidth resource of link l
δ_l	Propagation delay of link l
s	Source node
d	Destination node
$\mathcal{N}_v$	Neighboring nodes of physical node $v \in N$
λ	Arrival data rate of the traffic flow
D	E2E delay requirement
p_n	Unit cost of CPU resource at physical node n
p_{uv}	Unit cost of bandwidth resource at physical link (u, v)
T_q	Average queueing delay from source to destination
T	Total delay from source to destination

Table 3.2: Decision Variables

Decision variable	Description
C'_n	CPU resource demand at node n
B'_{sd}	Bandwidth resource demand at each link $l' \in L'$
x_n	Binary variable indicating whether or not node n is used to forward the traffic from source to destination
y_{uv}	Binary variable indicating whether or not link (u, v) is used to forward the traffic from source to destination

this bandwidth requirement as B'_{sd}. Thus, Constraint (3.1) is written as:

$$B'_{sd} \leq B_l, \quad \forall l \in L'. \tag{3.2}$$

Let C'_n be the CPU processing rate allocated to a virtual node embedded on physical node n. C'_n is upper bounded by C_n, given as

$$C'_n \leq C_n, \quad \forall n \in N'. \tag{3.3}$$

For E2E delay modeling, we establish a network of queues to analyze the packet delay of a service flow passing through an embedded VN. An example is shown in Fig. 3.3, where

n_1 and n_2 represent the network servers operating the embedded source and destination nodes, respectively. The network also contains a number of intermediate network servers and transmission links for traffic forwarding. A customized service flow enters the network at the source node n_1 and leaves the network at the destination node n_2. We assume traffic arrivals of one service flow at a source node follows a Poisson process. For VN embedding, when multiple service flows are placed over a common (or a partially common) network path, both CPU processing rate at each network server and transmission rate over each physical link along the path, need to be properly shared among the flows. This sharing is typically made according to a certain multi-resource sharing policy (e.g., dominant-resource generalized processor sharing [93]). Based on a proper resource sharing, a comprehensive E2E delay model can be established for each embedded service flow [94], which, on the other hand, complicates the VNE problem formulation. To simplify the resource sharing for problem tractability, we assume that CPU processing rates and link transmission rates allocated to each service flow are independent and exponentially distributed random variables [95]. In this way, both packet processing at an embedded virtual node and packet transmission over an embedded virtual link are modeled as an M/M/1 queue system [95]-[97]. Consequently, the queueing process of a service flow traversing an entire embedded VN is modeled as an open Jackson queueing network. We use λ to denote the arrival rate of the traffic flow at the source node of a VN. For each processing node in the embedded VN, the traffic intensity is given by

$$\rho_n = \lambda / C'_n < 1, \quad \forall n \in N'. \tag{3.4}$$

For each transmission link, the traffic intensity is given by

$$\rho_l = \lambda / B'_{sd} < 1, \quad \forall l \in L'. \tag{3.5}$$

Let T_n denote the packet waiting time at the processing queue of node $n \in N'$. Let T_l denote the packet waiting time at the transmission queue of link $l \in L'$. For the M/M/1 queues at equilibrium, the expectation of T_n and T_l are given by [97]

$$E[T_n] = \frac{1}{C'_n - \lambda}, \tag{3.6}$$

and

$$E[T_l] = \frac{1}{B'_{sd} - \lambda}. \tag{3.7}$$

Thus, the average packet queueing delay for the entire queueing network is given by [97]

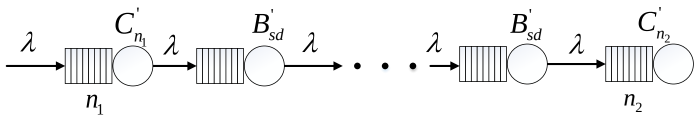


Figure 3.3: The queueing network to determine the optimal set of physical resource requirements at virtual nodes and links for each E2E VN.

$$T_q = \sum_{n \in N'} E[T_n] + \sum_{l \in L'} E[T_l]$$
$$= \sum_{n \in N'} \frac{1}{C'_n - \lambda} + \sum_{l \in L'} \frac{1}{B'_{sd} - \lambda}. \tag{3.8}$$

By considering the propagation delay δ_l at each link $l \in L'$, we calculate the average packet delay of traffic passing through the embedded VN as [97]:

$$T = T_q + \sum_{l \in L'} \delta_l. \tag{3.9}$$

To satisfy the E2E delay requirement of the accommodated service, we have

$$T \leq D. \tag{3.10}$$

To describe the VNE problem formulation, we define binary variables x_n ($n \in N$), where $x_n = 1$ if physical node n operates a virtual node of an embedded VN, and $x_n = 0$ otherwise. We further define binary variables y_{uv} ($u, v \in N$) to indicate whether the link $l = (u, v)$ is chosen to forward the traffic from s to d ($y_{uv} = 1$) or not ($y_{uv} = 0$). For each physical node and physical link, we have

$$y_{uv} B'_{sd} \leq B_{uv}, \quad \forall (u, v) \in L, \tag{3.11}$$

$$x_n C'_n \leq C_n, \quad \forall n \in N. \tag{3.12}$$

By introducing x_n and y_{uv}, Equation (3.8) and (3.9) can be transformed into the following forms:

$$T_q = \sum_{n \in N} x_n \frac{1}{C'_n - \lambda} + \sum_{l \in L} y_{uv} \frac{1}{B'_{sd} - \lambda}, \tag{3.13}$$

$$T = T_q + \sum_{(u,v) \in L} y_{uv} \delta_{uv}. \tag{3.14}$$

Substituting Equation (3.13) and (3.14) into Constraint (3.10) yields

$$\sum_{n\in N} x_n \frac{1}{C'_n - \lambda} + \sum_{(u,v)\in L} y_{uv}\left(\frac{1}{B'_{sd} - \lambda} + \delta_{uv}\right) \leq D. \tag{3.15}$$

The traffic routing constraints at the source node s, destination node d, and each embedded node in between are given, respectively, by [98]

$$\sum_{v\in\mathcal{N}_s} y_{sv} - \sum_{v\in\mathcal{N}_s} y_{vs} = 1 \tag{3.16}$$

$$\sum_{u\in\mathcal{N}_d} y_{ud} - \sum_{u\in\mathcal{N}_d} y_{du} = 1 \tag{3.17}$$

$$\sum_{v\in\mathcal{N}_u} y_{vu} = \sum_{v\in\mathcal{N}_u} y_{uv}, \quad \forall u \in N \setminus \{s,d\}. \tag{3.18}$$

where constraint (3.16) ensures that at least one node v from $\mathcal{N}_s$ needs to be selected with $y_{sv} = 1$; Constraint (3.17) indicates that at least one node u from $\mathcal{N}_d$ needs to be selected with $y_{ud} = 1$; For every other node $u \in N$, constraint (3.18) indicates that whether physical node u is chosen for embedding on the path from s to d.

The objective of joint VN topology design and embedding problem is to minimize the embedding cost $\mathcal{C}$, which consists of the costs of utilizing substrate links and substrate nodes:

$$\mathcal{C} = w_1 \sum_{n\in N} x_n p_n C'_n + w_2 \sum_{(u,v)\in L} y_{uv} p_{uv} B'_{sd}. \tag{3.19}$$

C'_n and B'_{sd} represent the allocated CPU resources at node n and the allocated bandwidth resources over each link for traffic forwarding from s to d, respectively. p_n represents the unit cost of allocating CPU resources at physical node n, while p_{uv} denotes the unit cost of bandwidth resources over physical transmission link (u,v). w_1 and w_2 are two weighting factors used to reflect the importance of node embedding and link embedding in contributing to the total embedding cost. It is seen from (3.19) that the embedding cost is calculated as a weighted summation of the physical resources allocated to the virtual nodes and virtual links in the VN. The weighting factors are the unit prices (depending on the residual resources) of utilizing the physical resource on each node/link. By minimizing the cost of embedding each VN, the resource utilization among the substrate nodes/links can be better balanced. This helps to reduce the congestion level of the bottleneck nodes in the system. Consequently, the substrate network is able to accommodate more services and the ISPs' total profit can be increased in the long run [20], [22], [89].

Design of the unit cost of physical resources: In the design of p_n and p_{uv}, we take the following three aspects into consideration:

- Intuitively, the unit cost of utilizing the CPU (bandwidth) resources at a node (link), p_n (p_{uv}), should be a function of the available physical resources at the node (link);
- p_n (p_{uv}) should be a monotonically decreasing function so that physical nodes and links with more remaining resources have smaller unit costs and are more likely embedded to achieve load-balancing in the long run [21];
- It is expected that the unit cost remains steady for physical nodes (links) with sufficient resources, while p_n (p_{uv}) changes more significantly as C_n (B_{uv}) varies for the nodes (links) with less remaining resources.

Based on these considerations, we define p_n and p_{uv} as

$$p_n = e^{-\frac{C_n - C_{min}}{K}}, \tag{3.20}$$

$$p_{uv} = e^{-\frac{B_l - B_{min}}{K}}, \tag{3.21}$$

where C_{min} and B_{min} are the minimum residual CPU resources at a node and the minimum residual bandwidth resources over a link for the entire network, respectively. K is a positive constant which represents the changing rate of the unit cost at a node/link when its physical resource is rare. Fig. 3.4 gives an illustration of p_n, where $C_{min} = 600$. Clearly, with different values of K, the impacts of the proposed unit cost function on load-balancing can be different.

Now, we formulate the joint VN topology design and embedding problem as an MINLP, given by

$$\begin{aligned}
\text{(P1)} \quad & \min_{x_n, y_{uv}, C'_n, B'_{sd}} w_1 \sum_{n \in N} x_n p_n C'_n + w_2 \sum_{(u,v) \in L} y_{uv} p_{uv} B'_{sd} \\
& \text{s.t.} \quad (3.4), (3.5), (3.11) - (3.18) \\
& \qquad\;\; x_n \in \{0, 1\}, \ \forall n \in N \\
& \qquad\;\; y_{uv} \in \{0, 1\}, \ \forall (u, v) \in L
\end{aligned} \tag{3.22}$$

By minimizing the total embedding cost in (P1), an embedded VN topology from the source node to the destination node can be found, with the set of optimal resources allocated to each embedded virtual node and virtual link to satisfy the E2E delay requirement.

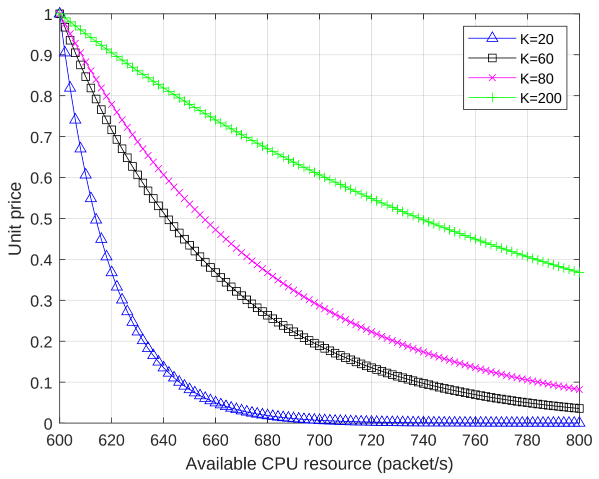


Figure 3.4: An illustration of the exponential unit cost function for utilizing substrate node resources.

Remark. Note that problem (P1) involves continuous variables C'_n and B'_{sd}, as well as binary variables x_n and y_{uv}. It contains a nonlinear objective function and a constraint in fractional form with x_n and y_{uv} as numerators, and C'_n and B'_{sd} as denominators. Therefore, (P1) is an MINLP which is NP-hard [99]. It is challenging in handling nonlinearities in (3.22) with a large number of combinatorial variables, and thus it is computationally complex to determine the optimal solution by solving the problem directly, especially for large-scale substrate networks.

3.5 Problem Transformation and Optimal Solutions

Although (P1) involves nonlinear constraints for the decision variables, it is possible to make it tractable by transforming it into a nonlinear programming (NLP). Recall that x_n and y_{uv} are variables indicating the intermediate nodes and links chosen to form the routing path. Therefore, (P1) can be transformed into an NLP if the routing path can be found in a certain way so that binary variables x_n and y_{uv} can be set as constants. Based on these intuitions, in this section, we propose an enhanced brute-force search algorithm, which can compute the optimal solution to (P1) but with relatively high computational overhead. In the algorithm, the substrate network is pruned in a way that the search space of the algorithm is reduced. In subsequent section, we further develop a low-complexity heuristic algorithm to improve the scalability of the algorithm.

3.5.1 Enhanced Brute-Force Search Algorithm

We first find alternative routing paths for embedding a VN to fix x_n and y_{uv} in (P1). In this way, the MINLP in (P1) is simplified as a NLP. Then, we determine the optimal physical resources allocated to embedded virtual nodes and links on the alternative paths by solving the NLP using an existing solver. By using this idea, we present an enhanced brute-force search algorithm, which is able to calculate the optimal solution to the MINLP, but with relatively high computational complexity. The algorithm enumerates all the possible substrate paths from the source node to the destination node. As shown in Algorithm 1, to reduce the computational complexity of searching for the optimal path, we first prune the substrate network based on constraints (3.4) and (3.5). Let $\tilde{G} = (\tilde{N}, \tilde{L})$ be the substrate network after pruning, where $\tilde{N} = \{n | n \in N, C_n > \lambda\}$, $\tilde{L} = \{l | l \in L, B_l > \lambda\}$. Next, a brute-force search process is performed in the pruned substrate network. Specifically, we find in $\tilde{G}$ all the substrate paths from s to d, along which the aggregate propagation link delay is smaller than D. Let $\mathcal{P}$ denote the set of these substrate E2E paths. Then,

$$\mathcal{P} = \{P_k | \sum_{l \in \bar{L}(P_k)} \delta_l < D\}, k = 1, \cdots, K, \tag{3.23}$$

where K is the total number of path candidates, and $\bar{L}(P_k)$ represents the set of substrate links corresponding to the kth path P_k. $P_1, P_2, \cdots, P_K$ are considered as the routing path candidates among which the one with the minimum embedding cost is chosen to forward the data traffic of the embedded VN from s to d.

Let $N'(P_k)$ and $L'(P_k)$ represent the set of substrate nodes and the set of substrate links corresponding to the kth path candidate, respectively. Then, we solve the following NLP to obtain the set of optimal physical resources and the embedding cost for path candidate P_k:

$$\begin{aligned}
\text{(P2)} \quad & \min_{C'_n, B'_{sd}} \mathcal{C}(P_k) = w_1 \sum_{n \in N'(P_k)} p_n C'_n + w_2 \sum_{l \in L'(P_k)} p_l B'_{sd} \\
& \text{s.t.} \quad C'_n \le C_n, \quad \forall n \in N'(P_k) \\
& \qquad\; B'_{sd} \le B_l, \quad \forall l \in L'(P_k) \\
& \sum_{n \in N'(P_k)} \frac{1}{C'_n - \lambda} + \sum_{l \in L'(P_k)} \frac{1}{B'_{sd} - \lambda} + \sum_{l \in L'(P_k)} \delta_l \le D.
\end{aligned} \tag{3.24}$$

In (P2), the objective of this constrained optimization problem is to minimize a linear function with a nonlinear inequality constraint and linear constraints. The interior-point or active-set algorithm can be applied to solve the problem to obtain the set of optimal allocated resources that achieve the minimal embedding cost for P_k.

```
Algorithm 1: Enhanced Brute-Force Search Algorithm
  Input  : $G = (N, L)$, $s$, $d$, $\lambda$, $D$
1 Find the pruned substrate network $\tilde{G}$ based on Constraints (3.4) and (3.5);
2 Find the set of all the path candidates $\mathcal{P} = \{P_1, P_2, \cdots, P_K\}$ in $\tilde{G}$ based on (3.23);
3 for $P_k \in \mathcal{P}$ do
4     Fix the values of $x_n$ and $y_{uv}$ corresponding to $P_k$;
5     Solve the NLP (3.24) to obtain the optimal set of physical resource
        requirements and the embedding cost $\mathcal{C}(P_k)$;
6 Compare the embedding costs of all the path candidates $\mathcal{C}(P_k)$, $k \in \{1, \ldots, K\}$;
7 Choose the path with minimum embedding cost as the optimal routing path, i.e.,
    $P_k^* = \arg\min_{P_k} \mathcal{C}(P_k)$;
8 Impose the corresponding set of physical resource requirements on substrate nodes
    and links on the selected routing path $P_k^*$;
  Output: Physical resource requirements $C'_n$ and $B'_{sd}$
          The routing path from $s$ to $d$
```

Finally, we compare the embedding costs of all the path candidates, $\mathcal{C}(P_k)$, $k = 1, \cdots, K$. The path with the minimum embedding cost is chosen as the optimal path. Let P_k^* represent the optimal path. Then, this step can be denoted as

$$P_k^* = \arg\min_{P_k} \mathcal{C}(P_k), \quad k \in \{1, 2, \cdots, K\}. \tag{3.25}$$

Then, the corresponding set of optimal physical resources are allocated to each embedded virtual node and virtual link.

3.6 Low-Complexity Heuristic Algorithm

The enhanced brute-force search algorithm can be used to find the optimal solution to (P1) with high computational complexity, which is not scalable in a large-scale network. For example, in a substrate network with 50 nodes and 123 links, the number of path candidates to embed a VN (with the E2E delay requirement of 40 ms) is 445 and it takes around 14.8 min (with 2.20 GHz CPUs) to find the optimal path with the minimum embedding cost. In reality, service requests arrive randomly independent of other requests, and the VN embedding process has to be executed faster than the service request arriving rate. In the following part, a heuristic algorithm is proposed to obtain near-optimal solutions to (3.22) with low computational complexity.

Algorithm 2: Low-Complexity Heuristic Algorithm

```
Input  : $G = (N, L)$, $s$, $d$, $\lambda$, $D$
1 Find the pruned substrate network $\tilde{G}$ based on constraints (3.4) and (3.5);
2 Define and compute a new weight $w_{uv}$ for each link $(u, v)$ in $\tilde{G} = (\tilde{N}, \tilde{L})$ using
    (3.26);
3 Find the routing path from $s$ to $d$ using shortest path algorithms;
4 Fix the binary variables $x_n$ and $y_{uv}$ in (3.22) corresponding to the shortest path;
5 Solve (3.22) with the fixed binary variables to obtain the set of physical resource
    requirements and the embedding cost;
Output: Physical resource requirements $C'_n$ and $B'_{sd}$
        The routing path from $s$ to $d$
```

3.6.1 Single-Service Scenario

On the arrival of a service request, the enhanced brute-force search algorithm enumerates all the possible routing paths from source to destination. The main advantage of the heuristic algorithm is that it finds one routing path without losing too much optimality of the solution. The algorithm contains two steps: First, a substrate path with low embedding cost from the source node to the destination node is found; Then, the set of physical resources allocated to each virtual node and virtual link are determined for the path found in the previous step.

The details of the low-complexity heuristic algorithm is given in Algorithm 2. In the first step, the substrate path is chosen in a way that the path is likely to support a VN with low embedding cost. To achieve this, a new weight is assigned to all physical links. Then, the shortest path from s to d is found using the k-shortest path algorithm [19] and chosen as the routing path from source to destination. In the second step, the set of physical resources allocated to each virtual node and virtual link is determined on the chosen routing path. Similar to the enhanced brute-force search algorithm, the original substrate network is first pruned based on constraints (3.4) and (3.5). After that, a weight w_{uv} is computed for each link (u, v) in the pruned substrate network $\tilde{G} = (\tilde{N}, \tilde{L})$ as follows

$$w_{uv} = \eta_d \delta_{uv} + \eta_c (e^{-\frac{B_{uv} - B_{min}}{K}} + e^{-\frac{C_u - C_{min}}{K}} + e^{-\frac{C_v - C_{min}}{K}}), \tag{3.26}$$

where δ_{uv} and B_{uv} are the propagation delay and available bandwidth resource over link $(u, v) \in \tilde{L}$, respectively. C_u and C_v are the available CPU resources of node $u \in \tilde{N}$ and node $v \in \tilde{N}$, respectively. η_d and η_c are weighting coefficients, with $\eta_d + \eta_c = 1$. Equation (3.26) indicates that physical links with smaller propagation delay and larger amount of bandwidth resources, and physical nodes with larger amount of CPU resources, are preferred in selecting the routing path from s to d. In this way, a low embedding cost is expected to be obtained after solving (P1). For example, $\eta_d = 1$ means that the path with minimum aggregate link propagation delay from s to d is chosen, while $\eta_c = 1$ indicates that the path with the most physical resources is selected. In practice, η_d and η_c should be set as different values for different scenarios, depending on network conditions and service types. After the routing path is found by the shortest path algorithms, the set of optimal physical resource resources allocated to the embedded VN are obtained by solving (P1).

Complexity Analysis: Here we provide the complexity analysis of Algorithm 2. The algorithm starts with pruning the original substrate network G to be $\tilde{G}$, whose complexity is $O(|N| + |L|)$. The next step is to compute the weight for each link in $\tilde{G}$ using $O(|\tilde{L}|)$ time. The complexity of finding the routing path from source to destination using the Dijkstra's algorithm is typically $O(|\tilde{N}|^2)$. The last step in Algorithm 2 is to solve an NLP once using the interior-point method.

3.6.2 Handling Online Service Requests

The low-complexity heuristic algorithm is able to perform VN topology design and embedding with low computational overhead. Therefore, it can be extended to embed multiple service requests in an online scenario [1], where the VN requests arrive and leave the substrate network consistently over time. Fig. 3.5 illustrates the overall online VN embedding procedure. The low-complexity heuristic algorithm is invoked each time a new VN request arrives for VN topology design and embedding. If the topology of the service request is successfully designed and embedded, it will be accommodated on the substrate network until its lifetime expires. At the same time, the status of substrate nodes and links is updated. Otherwise, the service request will be rejected. When the lifetime of an existing service request expires, the resources occupied by the embedded VN are released and the status of the substrate nodes and links is updated.

3.7 Performance Evaluation

In this section, we evaluate the performance of the proposed algorithms through extensive simulations. There are two main components in the simulation setting: 1) Generation of

[1] It is not efficient to apply the enhanced brute-force search algorithm in online scenarios due to its high complexity, as will be shown in Section 3.7.

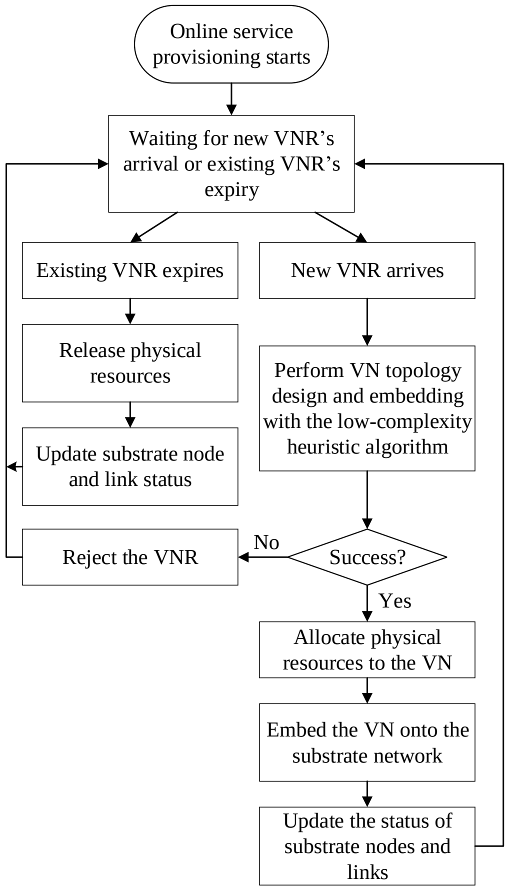

Online service provisioning starts
Waiting for new VNR's arrival or existing VNR's expiry
Existing VNR expires
New VNR arrives
Release physical resources
Update substrate node and link status
Perform VN topology design and embedding with the low-complexity heuristic algorithm
No
Success?
Reject the VNR
Yes
Allocate physical resources to the VN
Embed the VN onto the substrate network
Update the status of substrate nodes and links

Figure 3.5: Online VN topology design and embedding procedure.

the substrate network and service requests; 2) Evaluation of the proposed algorithms in static/online scenarios. The square-grid shaped substrate network and service requests are generated using the GT-ITM tool [100]. We have implemented a discrete-event network simulator developed by C++ to evaluate our proposed algorithms. The simulator allows efficient customization of the substrate network and service requests. The capacities of all substrate nodes and links are randomly generated following a certain uniform distribution. The link propagation delay is proportional to the distance between two end nodes. The value of K in the unit cost function is set as 60.0. The weighting factors in (3.19) are set as $w_1 = 0.1$ and $w_2 = 0.1$. We set $\eta_d = 0.4$ and $\eta_c = 0.6$ for the proposed low-complexity heuristic algorithm.

For comparison purposes, we choose the following two benchmark algorithms: 1) Capacity-based greedy link embedding with equal-delay resource allocation, in which the substrate path with the maximum residual resources is chosen as the routing path to forward the traffic from source to destination. Then, the CPU/bandwidth resource demands imposed on the nodes/links along the routing path are determined by setting the packet queueing delay over each node/link equal; 2) Shortest path link embedding with equal-delay resource allocation, in which the routing path is chosen by the shortest path algorithm [20], and the set of CPU/bandwidth resource demands is determined by performing equal-delay resource allocation. In the following, we first evaluate the performance of the proposed algorithms in VN topology design and embedding for a single service. Then, we demonstrate the advantages of the proposed low-complexity heuristic algorithm over the benchmarks in online scenarios.

3.7.1 Single-Service VN Topology Design and Embedding

3.7.1.1 Embedding cost

To compare the performance between the proposed algorithms and the two benchmarks in terms of the embedding cost, we perform VN embedding for a service request over a substrate with 50 nodes and 130 links. The capacities of all substrate nodes and links are uniformly distributed in range of [600 packets/s, 800 packets/s]. The traffic arrival rate of the service request is set as 150 packets/s, while the delay requirement is varied from 20 ms to 50 ms. The source and destination nodes are randomly chosen from the substrate nodes. Fig. 3.6 compares the embedding costs between the four algorithms. It can be seen that for all the four algorithms in comparison, the cost increases as the delay requirement becomes stringent. This is due to the fact that more CPU processing and link bandwidth resources are consumed to satisfy a more stringent delay requirement, leading to a higher embedding cost. Also, the embedding cost of the low-complexity heuristic algorithm and that of the brute-force search algorithm stay the same for all the cases, and are both lower than those of the two benchmarks. Moreover, the gap between them becomes more significant as the delay requirement becomes more stringent. The reason is that in the proposed algorithms, the optimal set of resource demands imposed on the nodes/links is optimized. In contrast, in the two benchmarks, the set of resource demands imposed on the nodes/link is determined based on equal-delay RA.

Next, we perform VN embedding for the same service request with fixed delay requirement and varying traffic arrival rates. Fig. 3.7 shows the cost comparison among the

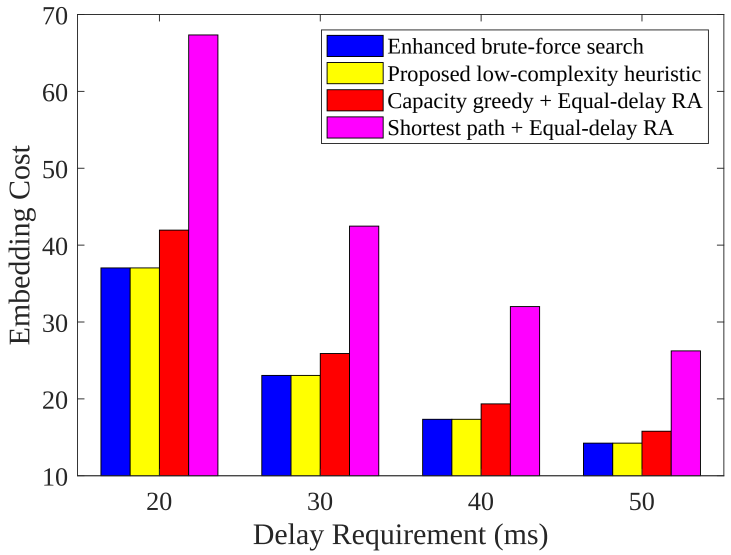


Figure 3.6: Comparison of VN embedding costs between the four algorithms under different delay requirements over the 50-node substrate network (traffic arrival rate is set as 150 packets/s).

four algorithms. It can be seen that, except for the case where the arrival data rate (λ) is 200 packets/s, the cost of the proposed low-complexity heuristic algorithm remains the same with that of the brute-force search algorithm, and keeps lower than those of the two benchmark algorithms. The embedding costs for the two benchmarks are not obtainable when $\lambda = 200$ because the delay requirement cannot be satisfied and the request is rejected. We can also observe that, for all the algorithms in comparison, the embedding cost increases with the traffic arrival rate.

Lastly, we present the allocation of CPU processing rate at various substrate nodes from source to destination for the given VN in Fig. 3.8. It is observed that two substrate nodes are chosen as the intermediate network servers to route the traffic from source to destination, and the optimal CPU processing rates allocated at each node along the routing path have different relationships with the arrival data rate. The difference between them is caused by their different unit costs of CPU resource. A similar tendency can be observed in Fig. 3.9, where the optimal transmission rates allocated over two consecutive substrate links along the routing path manifests a nearly linear relationship to the arrival data rate.

3.7.1.2 Time complexity

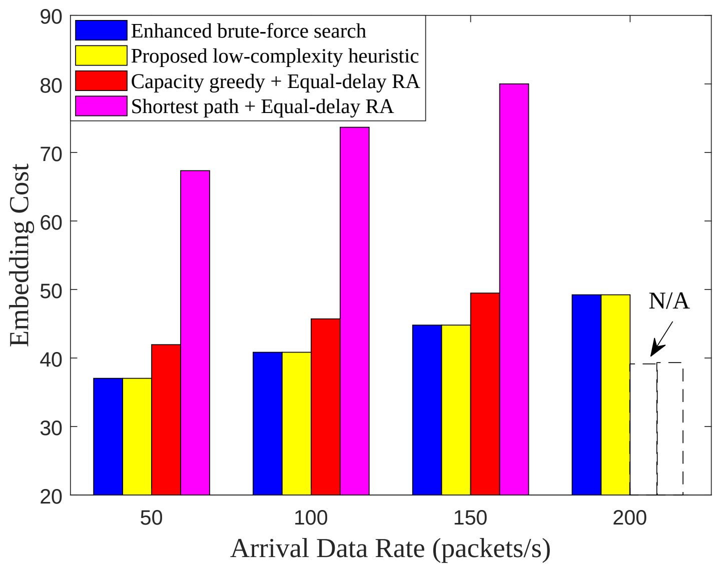


Figure 3.7: Comparison of VN embedding costs between the proposed algorithms and the two benchmarks over the 50-node substrate network (packet delay requirement is set as 35 ms).

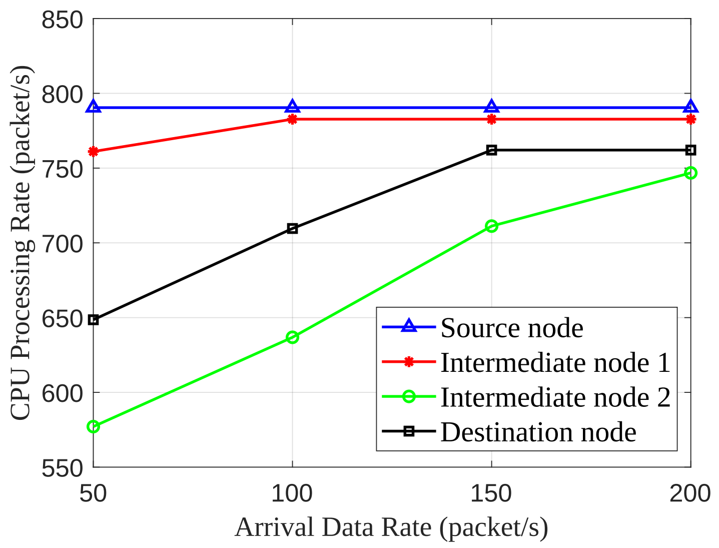


Figure 3.8: Optimal allocation of CPU processing rate at various substrate nodes under different arrival data rates (packet delay requirement is 35 ms).

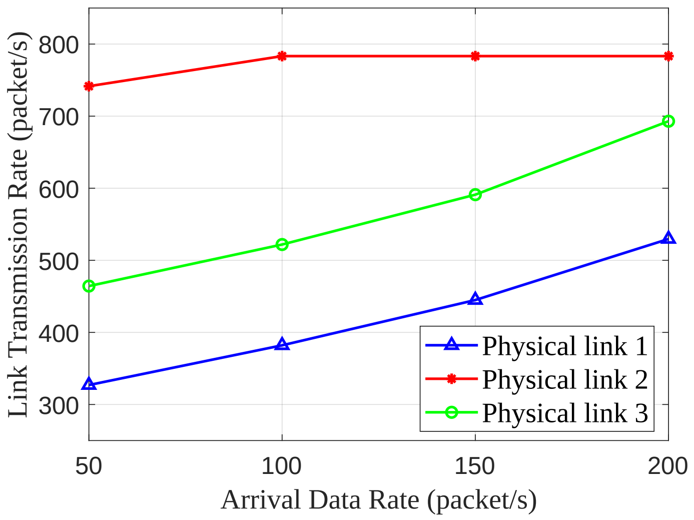


Figure 3.9: Optimal allocation of transmission rate over various substrate links under different arrival data rates (packet delay requirement is 35 ms).

Table 3.3: Parameters of The Three Substrate Networks

Parameter	Substrate I	Substrate II	Substrate III
Number of nodes	50	75	100
Number of links	130	335	552

We analyze the running time of the two proposed algorithms over three different substrate network scales, as shown in Table 3.3. Different service requests are considered to be embedded over different substrate networks. For each service request, the traffic arrival rate and delay requirement are random numbers within the range of [150, 250] and [20 ms, 50 ms], respectively. The source and destination nodes are randomly chosen from the substrate nodes.

Table 3.4 shows the running time between the two algorithms. For each substrate network, the running time of the low-complexity heuristic algorithm stays steady around 1.5 seconds. This is because the running time depends on the number of intermediate nodes along the weighted shortest path. For Substrate I, the running time of the enhanced brute-force search algorithm ranges from 2.95s to 10.31s under different delay requirements, which is 2-5 times than that of the low-complexity heuristic algorithm. For Substrate II (Substrate III), the running time of the brute-force search algorithm is around 10-300 times

Table 3.4: Comparison of the Running Time Between the Two Algorithms Over Three Substrate Networks

Running time / **Delay requirement**	Substrate I		Substrate II		Substrate III	
	Enhanced brute-force search	Low-complexity heuristic	Enhanced brute-force search	Low-complexity heuristic	Enhanced brute-force search	Low-complexity heuristic
30 ms	2.95 s	1.41 s	13 s	1.07 s	85 s	1.17 s
35 ms	5.89 s	1.45 s	103 s	1.14 s	408 s	1.21 s
40 ms	7.37 s	1.45 s	346 s	2.04 s	1358 s	1.25 s
45 ms	10.31 s	2.08 s	720 s	2.13 s	2306 s	1.58 s

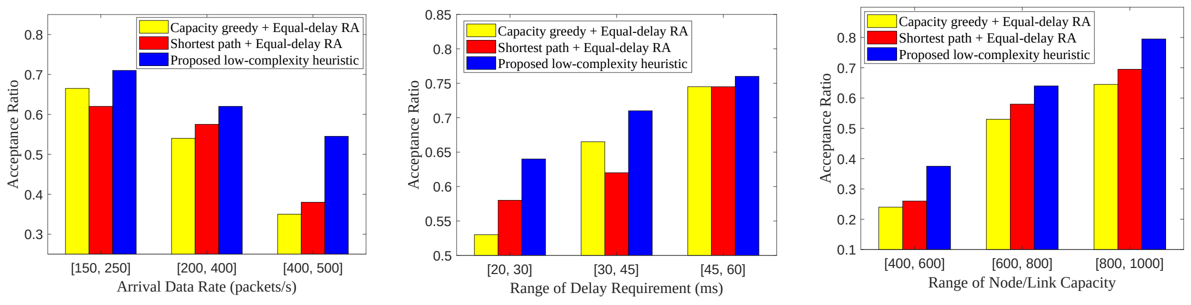
(60-1400 times) than that of the low-complexity running time, which indicates that the low-complexity heuristic algorithm has significantly better scalability than the enhanced brute-force search algorithm. The reason is that the low-complexity heuristic algorithm finds the weighted shortest path, while the running time of the enhanced brute-force search highly relies on the number of candidate routing paths.

3.7.2 Online VN Topology Design and Embedding

The effectiveness of the proposed low-complexity heuristic algorithm is further validated in online scenarios. The VN requests are generated by the VNOs and arrive to the system following a Poisson process. The average arrival rate is 4 VNs per second. The lifetime of each VN request has an exponential distribution with an average of 10 seconds. The total simulation time is 45 seconds. We adopt the VN acceptance ratio as the performance metric, which is defined as the number of accepted VN requests over the number of all the requests arrived to the system. In the following, we investigate the impacts of different factors on the performance of the proposed low-complexity heuristic algorithm and its advantages over the two benchmarks.

3.7.2.1 Impact of Arriving Data Rate

We first investigate the impact of arriving data rates (λ) of VN requests. For this purpose, we consider three cases of online VN topology design and embedding over a 50-node substrate network. For all the three cases, the delay requirement of the VN requests is uniformly distributed between 30 ms and 45 ms. The ranges of arrival data rate for



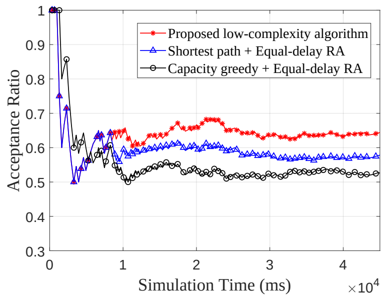
(a) (b) (c)

Figure 3.10: Performance comparison between the proposed low-complexity heuristic algorithm and the two benchmarks from different aspects: (a) Varying average arrival data rate of VN requests, (b) Varying average delay requirement of VN requests, (c) Average node/link capacities of the substrate network.

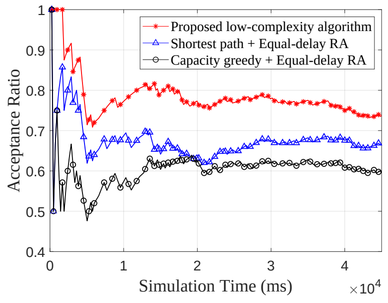
the service requests for the three cases are [150, 250], [200, 400], and [400, 500] (all in packets/s), respectively. Fig. 3.10 (a) compares the performance of the three algorithms in terms of service acceptance ratio. It can be seen from the figure that the proposed algorithm achieves the highest acceptance ratio for all the cases. For all the three algorithms, the acceptance ratio decreases as the average arrival data rate increases. This is because when λ becomes larger, more CPU/bandwidth resources on nodes/links are consumed to guarantee a shorter packet queueing delay, which leads to a higher chance of rejecting a new service. It can be also observed that the performance gap between the proposed algorithm and the two shortest path based algorithms increases with λ, demonstrating the superiority of the proposed algorithm when the physical resources in the substrate network are scarce.

3.7.2.2 Impact of Delay Requirement

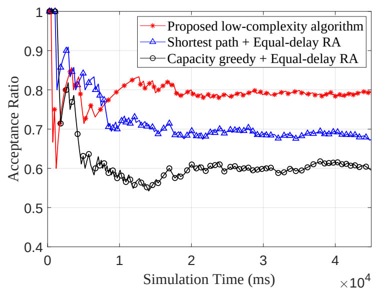
Next, we investigate the impact of delay requirements (D) of VN requests. Three cases are considered for online VN topology design and embedding over the same 50-node substrate, where the ranges of the delay requirement for the service requests are [20, 30], [30, 45], and [45, 60] (all in ms), respectively. The arrival data rates for the service requests are uniformly distributed between 150 and 250 (packets/s). The performance comparison of the three algorithms is shown in Fig. 3.10 (b). We can see that the proposed algorithm performs the best. On average, the proposed algorithm can achieve around 10% more service requests than the two benchmarks. In addition, the acceptance ratio increases as the average delay requirement become loose for all the three algorithms. This is reasonable



(a)



(b)



(c)

Figure 3.11: Performance comparison between the proposed heuristic algorithm and the two benchmarks over different substrate networks: (a) Substrate I, (b) Substrate II, (c) Substrate III.

since a loose delay requirements can tolerate a longer packet queueing delay, which means less amount of node/link processing and propagation resource consumption. Hence, the substrate network can accommodate more services.

3.7.2.3 Impact of Delay Requirement

The impact of node/link capacities is investigated by setting the average node/link capacities in the 50-node substrate network to different values. Three cases are considered, where the ranges of the node/link capacities are [400, 600], [600, 800], and [800, 1000], respectively. The delay requirement of the VN requests are uniformly distributed in the range of [20, 30] ms, while the arrival data rates for the service requests are uniformly distributed in the range of [150, 250] packets/s. Fig. 3.10 (c) compares the performance between the three different algorithms. As can be seen, in all the three cases, the proposed algorithm achieves a higher acceptance ratio than the other two algorithms. The main reason is that the set of resources allocated to the virtual nodes and links is optimized, which holds the potential to balance the load among substrate nodes/links. Additionally, the acceptance ratio increases as the node/link capacities become large, which is reasonable since more available physical resources in the network allows more service requests to be accommodated.

3.7.2.3 Performance Comparison over Different Network Scales

We next compare the performance between the proposed low-complexity heuristic algorithm and the two benchmarks over different substrate networks, i.e., Substrate I, II, and III as given in Table 3.3. For all the three substrate networks, the available CPU resources and bandwidth resources are uniformly distributed in the range of [600, 800]. The ranges of delay requirement and arrival data rate for the VN requests are [20, 30] (ms) and [150, 250] (packets/s), respectively. Fig. 3.11 compares the service acceptance ratio over time for the three network scales. The highest acceptance ratio is achieved by the proposed algorithms for all three network scales. The main reason is that the resources allocated to each VN in the proposed solution is optimized to guarantee that the QoS requirement can be satisfied and to avoid resource redundancy for supporting each VN. Consequently, the resource utilization of the substrate network is greatly improved.

3.8 Summary

In this chapter, the VN topology design and VN embedding problems have been jointly studied to improve service customization and resource utilization in 5G core networks. A joint optimization problem has been formulated to minimize the VN embedding cost while satisfying the E2E packet delay requirements for differentiated services. The E2E delay has been analyzed as a function of the allocated physical resources for embedded virtual nodes and virtual links. Two algorithms have been proposed to solve the problem for small and large scale substrate networks, respectively. Simulation results have shown the effectiveness and efficiency of the proposed algorithms for both single-service and online scenarios. For future work, an algorithm will be developed to adjust the weighting factors (η_d and η_c) adaptively for the proposed low-complexity heuristic algorithm.

Chapter 4

Multi-VNF Chain Embedding

Virtual network functions (VNFs) are mapped onto substrate networks as service function chains (SFC), which provides customized services and guaranteed quality-of-service (QoS) for network function virtualization (NFV) enabled networks. In this chapter, we address the multi-SFC embedding problem by a game theoretical approach, considering the heterogeneity of NFV nodes, the impact of processing-resource sharing among various VNFs, and the capacity constraints of NFV nodes. In the proposed resource constrained multi-SFC embedding game (RC-MSEG), each SFC is treated as a player whose objective is to minimize the overall latency experienced by the supported service flow, while satisfying the capacity constraints of all its NFV nodes. Due to processing-resource sharing, additional delay is incurred and integrated into the overall latency for each SFC. The capacity constraints of NFV nodes are considered by adding a penalty term into the cost function of each player, and are guaranteed by a prioritized admission control mechanism. The proposed game is proved to be an exact potential game that admits at least one pure Nash Equilibrium (NE) and has the finite improvement property (FIP). We then design two iterative algorithms, namely, the best response (BR) algorithm with fast convergence and the spatial adaptive play (SAP) algorithm with great potential to obtain the best NE of the proposed game. Simulations are conducted to validate the proposed approach.

4.1 Introduction

To achieve high network performance with load balancing in the next generation mobile communication, network slicing is one of the most promising solutions. Through network slicing, the whole network is sliced into multiple virtual networks (VNs). Each VN is used to support a customized service in a specific scenario. Multiple VNs are mutually isolated and can coexist in the common infrastructure sharing the same physical resource pool. The heterogeneous resources are virtualized to allow resource sharing among different VNs and to improve resource utilization in both access and core networks. Network function virtualization (NFV) is one of the key components for network slicing. NFV allows the network functions to be virtualized and placed on high capacity servers located anywhere in the network, not only on dedicated devices in current networks. As a result, One virtual network is modeled as a service function chain (SFC) composed of virtual nodes and virtual links. Driven by NFV platform, the future network architecture is expected to feature virtualized function chaining, reduced capital and operational costs, and enhanced service quality [38].

Efficient management of physical resources is one of the most crucial issues for NFV-enabled networks. Specifically, given the traffic statistics, QoS requirements of heterogeneous services, and resource constraints of the substrate network, SFCs are embedded onto the substrate network (referred to as *SFC embedding*), during which the physical resources need to be allocated efficiently to meet the resource and QoS requirements of each SFC [39]. SFC embedding determines the routing path from source to destination and finds the locations of NFV nodes to host the virtual network functions (VNFs) along the path, which increases the resource utilization with guaranteed QoS requirements [37].

Existing research efforts devoted to the SFC embedding problem can be generally categorized into two categories. In centralized approaches, a centralized entity holds a global view of the whole substrate network and orchestrates the VNFs to support all the SFCs [42]-[50]. These works either seek for optimal solutions through the optimization with high computation complexity, or attempt to obtain near-optimal solutions by proposing heuristic algorithms. Centralized approaches are typically developed from a service provider to achieve the network-wide optimization without considering competitive and non-cooperative behaviors. Moreover, centralized approaches usually suffer from high complexity, which makes them not scalable to large-scale networks. Distributed approaches, however, do not require a centralized controller, and allow each SFC to find its own embedding strategy independently [51]-[58]. Among these approaches, game theory has been recognized as a powerful tool to solve the SFC embedding problem for rational entities with conflicting objectives. Once the system reaches the Nash equilibrium (NE), no player has the intention to change the strategy unilaterally. In that sense, the SFC embedding problem is usually formulated from the user's perspective in a distributed manner with the objective of minimizing/maximizing their individual costs/payoffs. Compared with centralized approaches, performing SFC embedding in a distributed manner requires less information exchange between the centralized controller and the users, thereby greatly reducing the signaling overhead. However, the global optimality of the solution is not easy to be guaranteed. Therefore, distributed approaches can be used to balance the trade-off between computation complexity and optimality, especially for complex and large-scale network scenarios.

In this chapter, we propose a game theoretical approach for multi-SFC embedding, considering the effect of processing-resource sharing among different VNFs and the capacity constraints of different NFV nodes. In the proposed game, each SFC is treated as a player aiming to minimize the overall latency experienced by the traffic flow traversing the SFC. The proposed game is a resource-specific congestion game in that the NFV nodes are considered to be heterogeneous in terms of the ability of hosting VNFs, the time for processing data traffic of each VNF, and the latency parameter used to evaluate the effect of processing-resource sharing. Our main contributions are four-fold:

1. We formulate the multi-SFC embedding problem as a resource-specific congestion game, which is proved to be an exact potential game. In the proposed multi-SFC embedding game (MSEG), the effect of processing-resource sharing among SFCs is considered. Considering the additional delay due to processing-resource sharing, we have established an accurate end-to-end (E2E) delay model for each SFC with delay satisfaction;

2. To guarantee the capacity constraints on different NFV nodes, we formulate a new resource constrained multi-SFC embedding game (RC-MSEG) where a penalty term is added into the original cost function of each player. Both MSEG and RC-MSEG are proved to be exact potential games admitting at least one pure NE;

3. To obtain the NE of RC-MSEG, we design two iterative algorithms, namely, the best response (BR) algorithm with fast convergence and the spatial adaptive play (SAP) algorithm with great potential to obtain the best NE of the proposed game. The two algorithms correspond to local and global optimal solutions to the original problem, respectively;

4. To further guarantee the capacity constraints of NFV nodes as well as to achieve better load-balancing, we propose a prioritized admission control mechanism where the maximum number of embedded SFCs is predicted. Then, the size of the original SFC set is adjusted to accommodate the maximum set of SFCs with relatively high bit rates, while all capacity constraints are satisfied.

Table 4.1: Summary of Important Notations

Notation	Description
$\mathcal{N}$	set of players (SFCs)
N	number of players (SFCs) in the system
F_i	set of VNFs in SFC i
f_{ij}	jth VNF in SFC i
$\mathcal{M}$	set of VNF types in the whole system
$\mathcal{M}_i$	set of VNF types in SFC i
m_{ij}	type of jth VNF in SFC i
$\mathcal{V}$	set of NFV nodes in the system
V	number of NFV nodes in the system
$\mathcal{V}(m)$	set of NFV nodes that can support VNF type m
λ_i	bit rate of player $i \in N$
$\delta(v_1, v_2)$	link propagation delay over the link between $v_1 \in \mathcal{V}$ and $v_2 \in \mathcal{V}$
μ_v	maximum service rate of NFV node $v \in \mathcal{V}$
$\rho_{v,m}$	unit bit rate processing time for VNF type $m \in \mathcal{M}$ on NFV node $v \in \mathcal{V}$
$\rho(v, f_{ij})$	processing time for VNF f_{ij} on NFV node $v \in \mathcal{V}$
K_v	context-switching latency parameter for $v \in \mathcal{V}$
$\lvert\Upsilon_v\rvert$	number of SFCs that chooses server v to support a certain VNF
$d_i^{(prop)}(\bar{s}_i)$	total link propagation delay experienced by traffic flow i in strategy $\bar{s}_i$
$d_i^{(proc)}(\bar{s}_i)$	total VNF processing delay experienced by traffic flow i in strategy $\bar{s}_i$
$d_i^{(cs)}(\bar{s}_i, \boldsymbol{s}_{-i})$	additional context switching delay experienced by traffic flow i in strategy profile $(\bar{s}_i, \boldsymbol{s}_{-i})$

4.2 System Models and Game Formulation

In this section, we present the system models and the game formulation for the multi-SFC embedding problem. Table 4.1 summarizes the important notations used in this section.

4.2.1 Network Model

We consider an NFV system with a set of NFV nodes/servers denoted by $\mathcal{V}$, where $\mathcal{V} = \{1, 2, \ldots, V\}$, and a set of SFCs denoted by $\mathcal{N}$, where $\mathcal{N} = \{1, 2, \ldots, N\}$. Let $\mathcal{M}$ be the set of VNF types available in the whole system, where $\mathcal{M} = \{1, 2, \ldots, M\}$. Each SFC $i \in \mathcal{N}$ is composed by a set of VNFs $F_i = \{f_{i1}, f_{i2}, \ldots, f_{i|F_i|}\}$ corresponding to a set of VNF types $\mathcal{M}_i = \{m_{i1}, m_{i2}, \ldots, m_{i|F_i|}\}$, where $m_{ij} \in \mathcal{M}$ and $j = 1, \ldots, |F_i|$. Each SFC is also associated with a bit rate λ_i ($i \in \mathcal{N}$). Each NFV node is capable of hosting multiple VNFs of different types. Let $\mathcal{M}(v) \subset \mathcal{M}$ be the set of VNF types that can be supported by NFV node $v \in \mathcal{V}$. Each VNF type $m \in \mathcal{M}$ can be hosted by a set of NFV nodes, denoted by $\mathcal{V}(m) \subset \mathcal{V}$. Let $\rho_{v,m}$ be the unit bit rate processing time for a certain VNF type $m \in \mathcal{M}$ on NFV node $v \in \mathcal{V}(m)$. Accordingly, the processing time of VNF f_{ij} on node $v \in \mathcal{V}(m_{ij})$ is given by

$$\rho(v, f_{ij}) = \rho_{v,m_{ij}} \lambda_i. \tag{4.1}$$

Note that different NFV nodes have different abilities of processing VNFs of the same type. Denote the maximum service rate that NFV node v can offer by μ_v. Let $\delta(v_1, v_2)$ represent the link propagation delay over the link between NFV nodes v_1 and v_2, where $v_1 \in \mathcal{V}$ and $v_2 \in \mathcal{V}$.

4.2.2 Multi-SFC Processing-Resource Sharing Model

To reduce the VNF provisioning cost, we consider that multiple VNFs belonging to different SFCs can be embedded onto a common NFV server. However, embedding multiple VNFs on the same NFV node may result in performance degradation due to context switching among different VNFs [61]. The performance degradation is reflected in an increased VNF processing latency, referred to as the *context switching delay.* Therefore, to manage the physical resources efficiently and to achieve a better latency control over the traffic flows, the effect of processing-resource sharing should be taken into account when dealing with the multi-SFC embedding problem.

Let $d^{(cs)}(v)$ denote the additional context-switching delay imposed on the processing of all the VNFs embedded onto v. According to [61], this additional delay can be modeled as

$$d^{(cs)}(v) = |\Upsilon_v| K_v, \tag{4.2}$$

where $|\Upsilon_v|$ represents the number of SFCs that chooses server v to support a certain VNF; K_v is the context-switching latency parameter of node $v \in \mathcal{V}$.

Fig. 4.1 gives an example of embedding multiple SFCs onto the same substrate network, where multiple VNFs from different SFCs can share the processing resources of a common

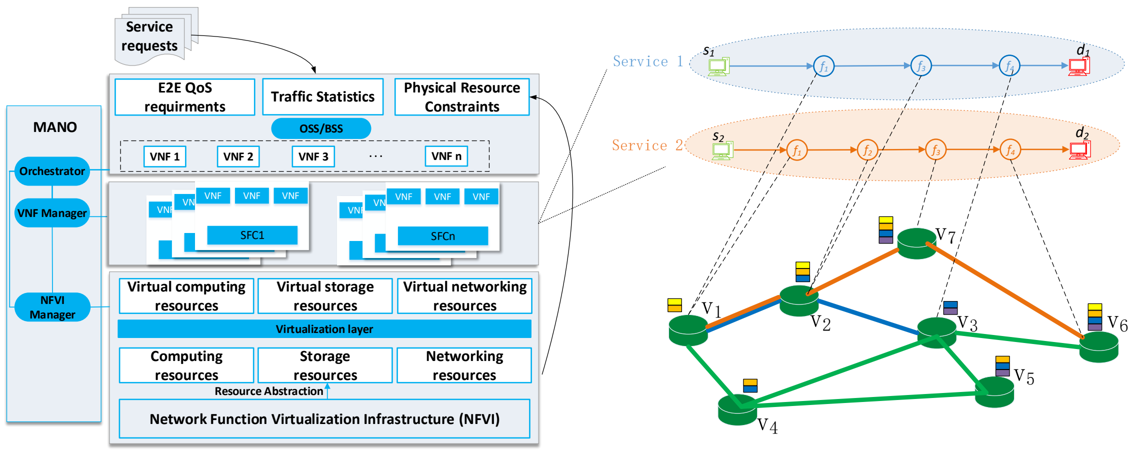


Figure 4.1: Illustration of multi-service SFC embedding, where different VNFs can share the physical resources of a common NFV node and different SFCs can share the same VNF.

NFV node. In the figure, we assume that all the substrate nodes are NFV nodes equipped with multiple CPU cores. Suppose that we have two SFCs to be embedded onto the substrate network. For a given service, the source and the destination nodes in the substrate network are known. Let $S_1 = \{s_1, f_1, f_3, f_4, d_1\}$ and $S_2 = \{s_2, f_1, f_2, f_3, f_4, d_2\}$ denote the first and the second SFCs, respectively (other available information associated with the SFC are omitted for brevity). f_1 in both SFC 1 and SFC 2 are embedded on the NFV node v_1. They will share the processing and other physical (such as storage and disk) resources of v_1. The impact of this resource-sharing among different VNFs is an increase on the end-to-end latency of both services. On the other hand, f_3 in SFC 1 and f_2 in SFC 2 are both embedded onto NFV node v_2, which also leads to latency increase for the two services.

4.2.3 Delay Models and Capacity Constraints

We refer to the ith SFC as player i with its embedding strategy being denoted by $\bar{s}_i$. Specifically, $\bar{s}_i = \left(s_{i1}, s_{i2}, \ldots, s_{i|F_i|}\right)$, where $s_{ij} \in \mathcal{V}$ represents the NFV node chosen by player i to process f_{ij}, $j = 1, \ldots, |F_i|$. Let S_i represent the strategy set of player i. In this work, we assume that different VNFs from the same SFC cannot be embedded onto the same NFV node. Therefore, the total number of elements in S_i is given by $|S_i| = |\mathcal{V}(m_{i1})| \times |\mathcal{V}(m_{i2})| \times \cdots \times |\mathcal{V}(m_{i|F_i|})|$.

For any player $i \in \mathcal{N}$ and any strategy $\bar{s}_i \in S_i$, the total link propagation delay experienced by the traffic flow corresponding to player i is given by

$$d_i^{(prop)}(\bar{s}_i) = \sum_{j=1}^{|F_i|-1} \delta(s_{ij}, s_{i,j+1}), \tag{4.3}$$

where $\delta(s_{ij}, s_{i,j+1})$ represents the link propagation delay between the two NFV nodes s_{ij} and $s_{i,j+1}$.

Based on (4.1), the overall delay for processing all the VNFs in SFC i in strategy $\bar{s}_i$ is given by

$$d_i^{(proc)}(\bar{s}_i) = \sum_{j=1}^{|F_i|} \rho_{s_{ij},m_{ij}} \lambda_i. \tag{4.4}$$

Let $\boldsymbol{s}_{-i}$ be the set containing all the strategies chosen by the players in the set $\mathcal{N} \setminus \{i\}$, i.e., $\boldsymbol{s}_{-i} = \{\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_{i-1}, \bar{s}_{i+1} \ldots, \bar{s}_N\}$. Considering that multiple VNFs managed by different players can be embedded onto the same NFV node, we define $\Upsilon_v(\bar{s}_i, \boldsymbol{s}_{-i})$ as the set containing the players that chooses server v to execute a certain VNF. Specifically,

$$\begin{aligned}\Upsilon_v(\bar{s}_i, \boldsymbol{s}_{-i}) = \{i' \in \mathcal{N} : \exists j \in \{1, \ldots, |F_{i'}|\}, s_{i'j} = v, \\ s_{i'j} \in \bar{s}_{i'}, \bar{s}_{i'} \in (\bar{s}_i, \boldsymbol{s}_{-i})\}.\end{aligned} \tag{4.5}$$

According to (4.2), the additional delay experienced by the traffic flow of player i due to context-switching is given by

$$d_i^{(cs)}(\bar{s}_i, \boldsymbol{s}_{-i}) = \sum_{v \in \bar{s}_i} |\Upsilon_v(\bar{s}_i, \boldsymbol{s}_{-i})| K_v, \tag{4.6}$$

where $|\cdot|$ represents the cardinality of a set.

Finally, to satisfy the capacity constraint of each NFV server, the total bit rates of different SFCs embedded onto the same server should not exceed the maximum service rate that can be provided by that server, i.e.,

$$\sum_{i' \in \Upsilon_v(\bar{s}_i, \boldsymbol{s}_{-i})} \lambda_{i'} \leq \mu_v, \ \forall v \in \mathcal{V}. \tag{4.7}$$

4.2.4 Game Formulation

The objective of player i is to minimize the overall latency experienced by its flow, while satisfying the capacity constraints of all the NFV nodes, i.e.,

$$\begin{aligned} &\min_{\bar{s}_i \in S_i} D_i(\bar{s}_i, \boldsymbol{s}_{-i}) \\ \text{s.t.} \quad &\sum_{j \in \Upsilon_v(\bar{s}_i, \boldsymbol{s}_{-i})} \lambda_j \leq \mu_v, \ \forall v \in V \end{aligned} \tag{4.8}$$

where $D_i(\bar{s}_i, \boldsymbol{s}_{-i})$ represents the overall latency experienced by the traffic flow managed by player i under strategy profile $(\bar{s}_i, \boldsymbol{s}_{-i})$. The overall latency for an embedded SFC consists of the total link propagation delay, the total processing delay, and the total additional context-switching delay imposed on all the VNFs in the SFC, i.e.,

$$D_i(\bar{s}_i, \boldsymbol{s}_{-i}) = d_i^{(prop)}(\bar{s}_i) + d_i^{(proc)}(\bar{s}_i) + d_i^{(cs)}(\bar{s}_i, \boldsymbol{s}_{-i}). \tag{4.9}$$

Accordingly, the multi-SFC embedding problem without considering the capacity constraints can be formulated as a congestion game: $\mathcal{G} = \{\mathcal{N}, \mathcal{V}, \mathcal{S}, (D_i(\bar{s}_i, \boldsymbol{s}_{-i}))_{i \in \mathcal{N}}\}$, where $\mathcal{N}$ represents the set of players, $\mathcal{V}$ the set of NFV servers, and $D_i(\bar{s}_i, \boldsymbol{s}_{-i})$ the cost function of player i as given in (4.9). $\mathcal{S}$ denotes the set of all the possible strategy profiles, i.e., $\mathcal{S} = S_1 \bigotimes S_2 \bigotimes \cdots \bigotimes S_N$. We refer to $\mathcal{G}$ as the multi-SFC embedding game (MSEG) in this chapter. Notably, MSEG is a resource-specific congestion game in that different NFV nodes have different ability of hosting VNFs ($\mathcal{M}(v)$), different processing times for the same VNF type ($\rho_{v,m}$), different maximum service rates (μ_v), and different context-switching latency parameters (K_v).

To relax the capacity constraints of NFV nodes, we modify the original cost function of each player by adding a penalty term to it as follows [112]-[114]

$$\bar{D}_i(\bar{s}_i, \boldsymbol{s}_{-i}) = D_i(\bar{s}_i, \boldsymbol{s}_{-i}) + wP(\bar{s}_i, \boldsymbol{s}_{-i}), \tag{4.10}$$

where w is the weighting parameter for the penalty term $P(\bar{s}_i, \boldsymbol{s}_{-i})$. The penalty term reflects the violation degree of the capacity constraints for all the NFV nodes, given by

$$P(\bar{s}_i, \boldsymbol{s}_{-i}) = \sum_{v \in V} p_v(\bar{s}_i, \boldsymbol{s}_{-i}), \tag{4.11}$$

where $p_v(\bar{s}_i, \boldsymbol{s}_{-i})$ is defined as:

$$p_v(\bar{s}_i, \boldsymbol{s}_{-i}) = \begin{cases} M & \text{if } \sum_{i' \in \Upsilon_v(\bar{s}_i, \boldsymbol{s}_{-i})} \lambda_{i'} > \mu_v, \\ 0 & \text{otherwise.} \end{cases} \tag{4.12}$$

Here M is a positive number used to penalize the strategy $\bar{s}_i$ if the capacity constraint on server v is violated. Based on (4.10)-(4.12), we reformulate the multi-SFC embedding problem subject to the capacity constraints as a new congestion game: $\bar{\mathcal{G}} = \{\mathcal{N}, \mathcal{V}, \mathcal{S}, (\bar{D}_i(\bar{s}_i, \boldsymbol{s}_{-i}))_{i\in\mathcal{N}}\}$, where $\bar{D}_i(\bar{s}_i, \boldsymbol{s}_{-i}))$ is the modified cost function given by (4.10).

Definition 1. A strategy profile $(\bar{s}_1^*, \bar{s}_2^*, \dots, \bar{s}_N^*) \in \mathcal{S}$ is a Nash Equilibrium (NE) if and only if,

$$D_i(\bar{s}_i^*, \boldsymbol{s}_{-i}^*) \leq D_i(\bar{s}_i, \boldsymbol{s}_{-i}^*),\ \forall i \in \mathcal{N},\ \forall \bar{s}_i \in S_i, \tag{4.13}$$

which means that $(\bar{s}_1^*, \bar{s}_2^*, \dots, \bar{s}_N^*)$ is a strategy profile where no player has the incentive to change its strategy since its overall latency cannot be decreased unilaterally.

Definition 2. A game is an exact potential game if there exists a function, Φ, that satisfies the following condition

$$D_i(\bar{a}, \boldsymbol{s}_{-i}) - D_i(\bar{b}, \boldsymbol{s}_{-i}) = \Phi(\bar{a}, \boldsymbol{s}_{-i}) - \Phi(\bar{b}, \boldsymbol{s}_{-i}), \tag{4.14}$$

where D_i is the cost function of player i, while $\bar{a}$ and $\bar{b}$ are two strategies chosen by player i. The function Φ is called the exact potential function of the game.

Theorem 1. The original game MSEG ($\mathcal{G}$) is an exact potential game which admits at least one pure strategy NE.

Proof. Define the potential function Φ for game $\mathcal{G}$ as follows:

$$\Phi(\bar{s}_i, \boldsymbol{s}_{-i}) = \sum_{i\in\mathcal{N}} \tilde{d}_i(\bar{s}_i) + \sum_{v\in\mathcal{V}} \sum_{n=1}^{|\Upsilon_v(\bar{s}_i, \boldsymbol{s}_{-i})|} nK_v \tag{4.15}$$

where $|\Upsilon_v(\bar{s}_i, \boldsymbol{s}_{-i})|$ represents the number of players that choose v to process a certain VNF, and $\tilde{d}_i(\bar{s}_i)$ is defined as

$$\tilde{d}_i(\bar{s}_i) = d_i^{(prop)}(\bar{s}_i) + d_i^{(proc)}(\bar{s}_i). \tag{4.16}$$

Suppose that player i changes its strategy from $\bar{a}$ to $\bar{b}$, while other players remain their current strategies. From (4.9), we obtain the change of the cost function for player i as

$$\begin{aligned} D_i(\bar{a},&\boldsymbol{s}_{-i}) - D_i(\bar{b}, \boldsymbol{s}_{-i}) \\ &= \tilde{d}_i(\bar{a}) - \tilde{d}_i(\bar{b}) + d_i^{(cs)}(\bar{a}, \boldsymbol{s}_{-i}) - d_i^{(cs)}(\bar{b}, \boldsymbol{s}_{-i}) \\ &= \tilde{d}_i(\bar{a}) - \tilde{d}_i(\bar{b}) \\ &+ \sum_{v\in\bar{a}} |\Upsilon_v(\bar{a}, \boldsymbol{s}_{-i})|K_v - \sum_{v\in\bar{b}} |\Upsilon_v(\bar{b}, \boldsymbol{s}_{-i})|K_v. \end{aligned} \tag{4.17}$$

From (4.15), we obtain the change of the potential function Φ between two strategy profiles $(\bar{a}, \boldsymbol{s}_{-i})$ and $(\bar{b}, \boldsymbol{s}_{-i})$ as

$$\begin{aligned}\Phi(\bar{a}, \boldsymbol{s}_{-i}) &- \Phi(\bar{b}, \boldsymbol{s}_{-i}) \\ &= \sum_{i\in\mathcal{N}} \tilde{d}_i(\bar{a}) - \sum_{i\in\mathcal{N}} \tilde{d}_i(\bar{b}) \\ &+ \sum_{v\in\mathcal{V}} (\sum_{n=1}^{|\Upsilon_v(\bar{a},\boldsymbol{s}_{-i})|} nK_v - \sum_{n=1}^{|\Upsilon_v(\bar{b},\boldsymbol{s}_{-i})|} nK_v).\end{aligned} \tag{4.18}$$

Define $\mathcal{F}_{\bar{a}-\bar{b}}$ as the set containing the VNFs of player i that are processed by different NFV nodes between $\bar{a}$ and $\bar{b}$, i.e., $\mathcal{F}_{\bar{a}-\bar{b}} = \{f_{ij} \in \mathcal{F}_i : \bar{a}(f_{ij}) \neq \bar{b}(f_{ij})\}$, where $\bar{a}(f_{ij})$ and $\bar{b}(f_{ij})$ are the NFV nodes chosen to support f_{ij} in strategy $\bar{a}$ and $\bar{b}$, respectively. Define $\mathcal{V}_{\bar{a}\backslash\bar{b}}$ as the set of NFV nodes that are chosen to process the VNFs in $\mathcal{F}_{\bar{a}-\bar{b}}$ in strategy $\bar{a}$, i.e., $\mathcal{V}_{\bar{a}\backslash\bar{b}} = \{\bar{a}(f_{ij}) \in \mathcal{V} : f_{ij} \in \mathcal{F}_{\bar{a}-\bar{b}}\}$. Similarly, define $\mathcal{V}_{\bar{b}\backslash\bar{a}}$ as the set of NFV nodes that are chosen to process the VNFs in $\mathcal{F}_{\bar{a}-\bar{b}}$ in strategy $\bar{b}$, i.e., $\mathcal{V}_{\bar{b}\backslash\bar{a}} = \{\bar{b}(f_{ij}) \in \mathcal{V} : f_{ij} \in \mathcal{F}_{\bar{a}-\bar{b}}\}$. Fig. 4.2 illustrates the definitions of $\mathcal{F}_{\bar{a}-\bar{b}}$, $\mathcal{V}_{\bar{a}\backslash\bar{b}}$, and $\mathcal{V}_{\bar{b}\backslash\bar{a}}$ using a simple example with $|F_i| = 3$ and $V = 4$.

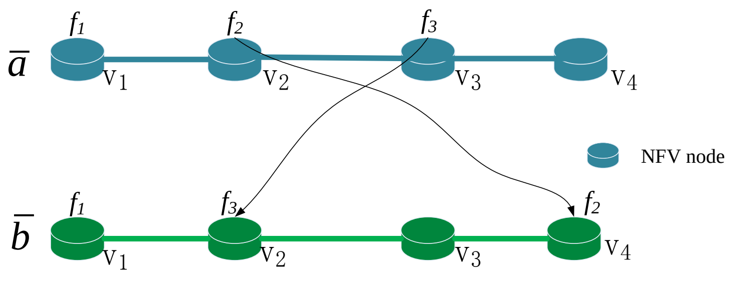
We can rewrite the change of the cost function of player i in (4.17) into

$$\begin{aligned}D_i(\bar{a}, \boldsymbol{s}_{-i}) &- D_i(\bar{b}, \boldsymbol{s}_{-i}) \\ &= \tilde{d}_i(\bar{a}) - \tilde{d}_i(\bar{b}) \\ &+ \sum_{v\in\mathcal{V}_{\bar{a}\backslash\bar{b}}} |\Upsilon_v(\bar{a}, \boldsymbol{s}_{-i})|K_v - \sum_{v\in\mathcal{V}_{\bar{b}\backslash\bar{a}}} |\Upsilon_v(\bar{b}, \boldsymbol{s}_{-i})|K_v.\end{aligned} \tag{4.19}$$

And the change of the potential function Φ given by (4.18) can be now expressed as

$$\begin{aligned}\Phi(\bar{a},\boldsymbol{s}_{-i}) &- \Phi(\bar{b}, \boldsymbol{s}_{-i}) \\ &= \tilde{d}_i(\bar{a}) - \tilde{d}_i(\bar{b}) \\ &+ \sum_{v\in\mathcal{V}_{\bar{a}\backslash\bar{b}}\cup\mathcal{V}_{\bar{b}\backslash\bar{a}}} \left(\sum_{n=1}^{|\Upsilon_v(\bar{a},\boldsymbol{s}_{-i})|} nK_v - \sum_{n=1}^{|\Upsilon_v(\bar{b},\boldsymbol{s}_{-i})|} nK_v \right).\end{aligned} \tag{4.20}$$

Now, let $\mathcal{V}_0 = \mathcal{V}_{\bar{a}\backslash\bar{b}} \cup \mathcal{V}_{\bar{b}\backslash\bar{a}}$. Then, $\mathcal{V}_0$ can be considered as the union of three disjoint sets, i.e., $\mathcal{V}_0 = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \mathcal{V}_3$, where $\mathcal{V}_1 = \mathcal{V}_{\bar{a}\backslash\bar{b}} \cap \mathcal{V}_{\bar{b}\backslash\bar{a}}$, $\mathcal{V}_2 = \mathcal{V}_{\bar{a}\backslash\bar{b}} \setminus \mathcal{V}_1$, and $\mathcal{V}_3 = \mathcal{V}_{\bar{b}\backslash\bar{a}} \setminus \mathcal{V}_1$. Fig. 4.3 illustrates the relationships between the sets $\mathcal{V}_{\bar{a}\backslash\bar{b}}$, $\mathcal{V}_{\bar{b}\backslash\bar{a}}$, $\mathcal{V}_1$, $\mathcal{V}_2$, and $\mathcal{V}_3$. Given an NFV node $v \in \mathcal{V}_0$, depending on which subset (among $\mathcal{V}_1$, $\mathcal{V}_2$, and $\mathcal{V}_3$) v belongs to, we have the following three cases to describe the relationship between $|\Upsilon_v(\bar{a}, \boldsymbol{s}_{-i})|$ and $|\Upsilon_v(\bar{b}, \boldsymbol{s}_{-i})|$:



$$\mathcal{F}_{\bar{a}-\bar{b}} = \{f_2, f_3\} \quad \mathcal{V}_{\bar{a}\backslash\bar{b}} = \{v_2, v_3\} \quad \mathcal{V}_{\bar{b}\backslash\bar{a}} = \{v_2, v_4\}$$

$$\mathcal{V}_1 = \{v_2\} \qquad \mathcal{V}_2 = \{v_3\} \qquad \mathcal{V}_3 = \{v_4\}$$

Figure 4.2: Illustration of the definitions of $\mathcal{F}_{\bar{a}-\bar{b}}$, $\mathcal{V}_{\bar{a}\backslash\bar{b}}$, and $\mathcal{V}_{\bar{b}\backslash\bar{a}}$.

Case 1: $v \in \mathcal{V}_1$, i.e., the NFV server v is chosen by player i in both $\bar{a}$ and $\bar{b}$, but to process two different VNFs. For the example shown in Fig. 4.2, $\mathcal{V}_1 = \{v_2\}$ and this case refers to the NFV node v_2. In such case, the number of VNFs received by v in the old strategy $\bar{a}$ and that in the new strategy $\bar{b}$ are the same. Accordingly, we have $|\Upsilon_v(\bar{a}, \boldsymbol{s}_{-i})| = |\Upsilon_v(\bar{b}, \boldsymbol{s}_{-i})|$.

Case 2: $v \in \mathcal{V}_2$, i.e., the NFV server v is chosen by player i to process a certain VNF in the old strategy $\bar{a}$, but that VNF is moved to a different server in the new strategy $\bar{b}$. As for the example shown in Fig. 4.2, this case refers to the NFV node v_3. In such case, the number of VNFs received by v in strategy $\bar{a}$ is one more than that in strategy $\bar{b}$. Therefore, we have $|\Upsilon_v(\bar{a}, \boldsymbol{s}_{-i})| = |\Upsilon_v(\bar{b}, \boldsymbol{s}_{-i})| + 1$.

Case 3: $v \in \mathcal{V}_3$, i.e., in the new strategy $\bar{b}$, player i places a certain VNF on server v, which was not chosen by the old strategy $\bar{a}$. This case refers to the NFV node v_4 for the example shown in Fig. 4.2. In this case, the number of VNFs received by v in $\bar{a}$ is one less than that in $\bar{b}$. i.e., $|\Upsilon_v(\bar{a}, \boldsymbol{s}_{-i})| = |\Upsilon_v(\bar{b}, \boldsymbol{s}_{-i})| - 1$.

With the three cases above in mind, we obtain the change of the potential function Φ

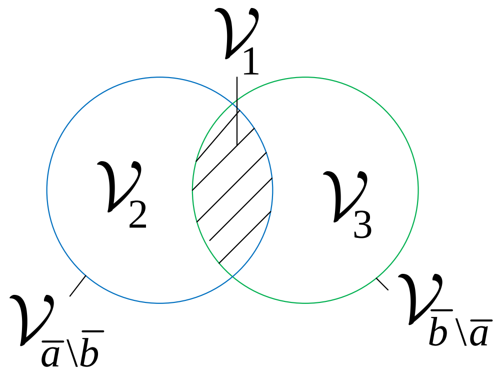

$\mathcal{V}_1$
$\mathcal{V}_2$
$\mathcal{V}_3$
$\mathcal{V}_{\bar{a}\backslash\bar{b}}$
$\mathcal{V}_{\bar{b}\backslash\bar{a}}$

Figure 4.3: Illustration of the relationships between the sets $\mathcal{V}_{\bar{a}\backslash\bar{b}}$, $\mathcal{V}_{\bar{b}\backslash\bar{a}}$, $\mathcal{V}_1$, $\mathcal{V}_2$, and $\mathcal{V}_3$.

in (4.20) as follows

$$\begin{aligned}
&\Phi(\bar{a},\boldsymbol{s}_{-i}) - \Phi(\bar{b},\boldsymbol{s}_{-i}) \\
&\quad= \tilde{d}_i(\bar{a}) - \tilde{d}_i(\bar{b}) \\
&\quad+ \sum_{v\in\mathcal{V}_2} |\Upsilon_v(\bar{a},\boldsymbol{s}_{-i})|K_v - \sum_{v\in\mathcal{V}_3} |\Upsilon_v(\bar{b},\boldsymbol{s}_{-i})|K_v \\
&\quad= \tilde{d}_i(\bar{a}) - \tilde{d}_i(\bar{b}) \\
&\quad+ \left(\sum_{v\in\mathcal{V}_2} |\Upsilon_v(\bar{a},\boldsymbol{s}_{-i})|K_v + \sum_{v\in\mathcal{V}_1} |\Upsilon_v(\bar{a},\boldsymbol{s}_{-i})|K_v\right) \\
&\quad- \left(\sum_{v\in\mathcal{V}_1} |\Upsilon_v(\bar{b},\boldsymbol{s}_{-i})|K_v + \sum_{v\in\mathcal{V}_3} |\Upsilon_v(\bar{b},\boldsymbol{s}_{-i})|K_v\right) \\
&\quad= \tilde{d}_i(\bar{a}) - \tilde{d}_i(\bar{b}) \\
&\quad+ \sum_{v\in\mathcal{V}_{\bar{a}\backslash\bar{b}}} |\Upsilon_v(\bar{a},\boldsymbol{s}_{-i})|K_v - \sum_{v\in\mathcal{V}_{\bar{b}\backslash\bar{a}}} |\Upsilon_v(\bar{b},\boldsymbol{s}_{-i})|K_v \\
&\quad= D_i(\bar{a},\boldsymbol{s}_{-i}) - D_i(\bar{b},\boldsymbol{s}_{-i}).
\end{aligned} \tag{4.21}$$

Therefore, the potential function Φ is an exact potential function for the original game $\mathcal{G}$. This completes the proof. □

Now, based on the previous proof, we will show that the RC-MSEG $\bar{\mathcal{G}}$ is also an exact potential function with the following potential function:

$$\bar{\Phi}(\bar{s}_i,\boldsymbol{s}_{-i}) = \Phi(\bar{s}_i,\boldsymbol{s}_{-i}) + wP(\bar{s}_i,\boldsymbol{s}_{-i}), \tag{4.22}$$

where the definitions of $P(\bar{s}_i, \boldsymbol{s}_{-i})$ and $p_v(\bar{s}_i, \boldsymbol{s}_{-i})$ are given by (4.11) and (4.12), respectively.

Proof. Suppose that the strategy of player i is changed from $\bar{a} \in S_i$ to $\bar{b} \in S_i$, then, the change of the modified cost function $\bar{D}_i(\bar{s}_i, \boldsymbol{s}_{-i})$ is given by

$$\begin{aligned} &\bar{D}_i(\bar{a}, \boldsymbol{s}_{-i}) - \bar{D}_i(\bar{b}, \boldsymbol{s}_{-i}) \\ &= D_i(\bar{a}, \boldsymbol{s}_{-i}) - D_i(\bar{b}, \boldsymbol{s}_{-i}) + w(P(\bar{a}, \boldsymbol{s}_{-i}) - P(\bar{b}, \boldsymbol{s}_{-i})) \end{aligned} \tag{4.23}$$

Based on (4.21) and (4.22), we obtain the change of the potential function $\bar{\Phi}$ as follows

$$\begin{aligned} &\bar{\Phi}(\bar{a}, \boldsymbol{s}_{-i}) - \bar{\Phi}(\bar{b}, \boldsymbol{s}_{-i}) \\ &= \Phi(\bar{a}, \boldsymbol{s}_{-i}) - \Phi(\bar{b}, \boldsymbol{s}_{-i}) + w(P(\bar{a}, \boldsymbol{s}_{-i}) - P(\bar{b}, \boldsymbol{s}_{-i})) \\ &= \bar{D}_i(\bar{a}, \boldsymbol{s}_{-i}) - \bar{D}_i(\bar{b}, \boldsymbol{s}_{-i}). \end{aligned} \tag{4.24}$$

That is, the potential function $\bar{\Phi}$ is an exact potential function for the new potential game $\bar{\mathcal{G}}$. This completes the proof. □

4.3 Algorithm Design

There are two main properties for potential games, i.e., every finite ordinal potential game has a pure strategy equilibrium, and every finite ordinal potential game has the Finite Improvement Property (FIP). Therefore, in this section, we first employ the best response (BR) algorithm to seek for an NE. The BR algorithm can in general achieve an NE quickly, but may result in a local optimal solution at which the potential function is not maximized. To improve the solution quality at NE, we further propose a learning algorithm, i.e., the spatial adaptive play (SAP) algorithm, to find the best NE.

4.3.1 Best Response Iterative Algorithm

For finite ordinal potential games, it is well-known that an equilibrium point exists and that every maximal improvement path must terminate at an equilibrium point. With the FIP property, the basic best response algorithm can be employed to find the NE of the proposed potential game $\bar{\mathcal{G}}$. The BR algorithm is executed in a round-robin manner. In each round, one player is chosen to update its strategy to the best strategy such that the cost function is minimized, while other players keep their strategies unchanged. The

Algorithm 3: Best response algorithm to find the NE

```
1  Initialize the round counter $r \leftarrow 0$;
2  for $i$ in $\mathcal{N}$ do
3      Randomly select a strategy from the whole strategy set to initialize $\bar{s}_i(0)$;
4      Execute the initial strategy;
5  Set $r \leftarrow r + 1$;
6  while $stopConditionNotMet()$ do
7      for $i$ in $\mathcal{N}$ do
8          Keep $\boldsymbol{s}_{-i}$ unchanged, i.e., $\boldsymbol{s}_{-i}(r) = \boldsymbol{s}_{-i}(r-1)$
9          Loop all the possible strategies of player $i$ to find the best response strategy
             $\bar{s}_i^*$, i.e.
10         $\bar{s}_i^* = \arg\min_{\bar{s}_i} \bar{D}_i(\bar{s}_i, \boldsymbol{s}_{-i}(r)), \forall \bar{s}_i \in \mathcal{S}_i$;
11         Set $\bar{s}_i(r) = \bar{s}_i^*$;
12         Execute the best response strategy $\bar{s}_i^*$;
13         Update the round counter $r \leftarrow r + 1$;
```

algorithm is terminated once the stopping criterion is met (e.g., the value of potential function does not change for a number of successive rounds, or the maximum number of rounds is reached). The details of BR algorithm is shown in Algorithm 3.

Due to the FIP feature of ordinal potential game, after a finite number of rounds, the BR algorithm can convergence to a stable solution which corresponding to an NE. The solution obtained from BR is at least locally optimal.

4.3.2 Spatial Adaptive Play Algorithm

We now present a learning algorithm, i.e., the SAP algorithm [116]-[118], to find the best NE of our game $\bar{\mathcal{G}}$. In the SAP algorithm, an exploration parameter is used to determine the probability of escaping from a local minimum/maximum of the potential function.

The strategies of the players are updated in a round-robin manner. Let $(\bar{s}_i(r), \boldsymbol{s}_{-i}(r))$ be the strategy profile at round r. Initially ($r = 0$), each player selects a strategy from $\mathcal{S}_i$ following a uniform distribution $p_{i,j}(r = 0) = 1/|\mathcal{S}_i|$, where $j = 1, \ldots, |\mathcal{S}_i|$ and $p_{i,j}$ is the probability of player i selecting the jth strategy from $\mathcal{S}_i$. At round $r + 1$, one player (say player i) is to update its strategy from $\bar{s}_i(r)$ to $\bar{s}_i(r+1)$. The new strategy is selected from

Algorithm 4: SAP algorithm to find the best NE

```
1  Initialize the round counter $r \leftarrow 0$;
2  Initialize the exploration parameter $\beta$;
3  Initialize the probability vector $p_{i,j}(r=0) = 1/|S_i|, \forall i \in \mathcal{N}, \forall j = 1, \dots, |S_i|$;
4  for $i$ in $\mathcal{N}$ do
5      Select an initial strategy from the whole strategy set according to $p_{i,j}(r=0)$;
6      Execute the initial strategy;
7  Set $r \leftarrow r+1$;
8  while $stopConditionNotMet()$ do
9      for $i$ in $\mathcal{N}$ do
10         Keep $\boldsymbol{s}_{-i}$ unchanged, i.e., $\boldsymbol{s}_{-i}(r) = \boldsymbol{s}_{-i}(r-1)$;
11         Loop all the possible strategies of player $i$ to calculate $\bar{D}_i(\bar{s}_{i,j}, \boldsymbol{s}_{-i}(r))$,
           $\forall j = 1, \dots, |S_i|$;
12         Update the probability vector:
13         $p_{i,j}(r) = \frac{\exp(-\beta \bar{D}_i(\bar{s}_{i,j}, \boldsymbol{s}_{-i}(r)))}{\sum_{\bar{s}_{i,j} \in S_i} \exp(-\beta \bar{D}_i(\bar{s}_{i,j}, \boldsymbol{s}_{-i}(r)))}$;
14         Select the new strategy $\bar{s}_i(r)$ according to the updated probability vector
           $p_{i,j}(r)$;
15         Execute the updated strategy $\bar{s}_i(r)$;
16         Update the round counter $r \leftarrow r+1$;
```

whole strategy set $\mathcal{S}_i$ according to the following probability distribution $p_{i,j}(r+1)$:

$$p_{i,j}(r+1) = \frac{\exp(-\beta \bar{D}_i(\bar{s}_{i,j}, \boldsymbol{s}_{-i}(r)))}{\sum_{\bar{s}_{i,j} \in S_i} \exp(-\beta \bar{D}_i(\bar{s}_{i,j}, \boldsymbol{s}_{-i}(r)))}, \tag{4.25}$$

where $\bar{s}_{i,j}$ represents the jth strategy in S_i; $\bar{D}_i(\bar{s}_{i,j}, \boldsymbol{s}_{-i}(r))$ is the cost when the strategy profile is $(\bar{s}_{i,j}, \boldsymbol{s}_{-i}(r))$; $\beta \geq 0$ is the exploration parameter. A large β will force the players to select the best response strategy with high probability, while a small β in general leads to slower convergence. The SAP algorithm terminates if the stopping criterion is satisfied (e.g., the maximum number of rounds is reached). The details of the SAP algorithm is shown in Algorithm 4.

Theorem 2. The proposed SAP can converge to the following stationary distribution $\pi(\boldsymbol{s})$

$$\pi(\boldsymbol{s}) = \frac{\exp\left(-\beta(\boldsymbol{s})\bar{\Phi}(\boldsymbol{s})\right)}{\sum_{\boldsymbol{s}' \in \mathcal{S}} \exp\left(-\beta \bar{\Phi}(\boldsymbol{s}')\right)}. \tag{4.26}$$

Proof. Following similar proofs in [109], [115]-[118], let us denote the network state at the rth round as $\boldsymbol{s}(r) = (\bar{s}_1(r), \bar{s}_2(r), \ldots, \bar{s}_N(r))$, where $\bar{s}_i(r)$ represents the strategy of player i. Obviously, $s(r)$ is a discrete time, irreducible, and aperiodic Markov process, which has a unique stationary distribution. Let $\boldsymbol{s}_1 \in \mathcal{S}$ and $\boldsymbol{s}_2 \in \mathcal{S}$ represent any two arbitrary network states. Specifically, let $\boldsymbol{s}_1 = (\bar{s}_1, \ldots, \bar{s}_{m-1}, \bar{s}_m, \bar{s}_{m+1} \ldots, \bar{s}_N)$ and $\boldsymbol{s}_2 = (\bar{s}_1, \ldots, \bar{s}_{m-1}, \bar{s}'_m, \bar{s}_{m+1} \ldots, \bar{s}_N)$, where $m \in \mathcal{N}$ is the player chosen to change her strategy between states $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$. Denote the transition probability from $\boldsymbol{s}_1$ to $\boldsymbol{s}_2$ by $P(\boldsymbol{s}_2|\boldsymbol{s}_1)$. To prove that the unique distribution of $\boldsymbol{s}(r)$ must be (4.26), we only need to prove that the following balanced equation holds

$$\pi(\boldsymbol{s}_1)P(\boldsymbol{s}_2|\boldsymbol{s}_1) = \pi(\boldsymbol{s}_2)P(\boldsymbol{s}_1|\boldsymbol{s}_2). \tag{4.27}$$

Given $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ defined earlier, the L.H.S. of above equation is given by

$$\begin{aligned} &\pi(\boldsymbol{s}_1)P(\boldsymbol{s}_2|\boldsymbol{s}_1) \\ &= \frac{\exp(-\beta\bar{\Phi}(\boldsymbol{s}_1))}{\sum_{\boldsymbol{s}_1\in\mathcal{S}} \exp(-\beta\bar{\Phi}(\boldsymbol{s}_1))} \cdot \frac{\exp(-\beta\bar{D}_m(\bar{s}'_m, \boldsymbol{s}_{-m}))}{\sum_{\bar{s}'_m\in S_m} \exp(-\beta\bar{D}_m(\bar{s}'_m, \boldsymbol{s}_{-m}))} \\ &= \eta \exp\{-\beta(\bar{\Phi}(\boldsymbol{s}_1) + \bar{D}_m(\bar{s}'_m, \boldsymbol{s}_{-m}))\} \end{aligned} \tag{4.28}$$

where the parameter η is defined as

$$\eta = 1/\{\sum_{\boldsymbol{s}_1\in\mathcal{S}} \exp(-\beta\bar{\Phi}(\boldsymbol{s}_1)) \cdot \sum_{\bar{s}'_m\in S_m} \exp(-\beta\bar{D}_m(\bar{s}'_m, \boldsymbol{s}_{-m}))\}.$$

In a similar manner, we can also obtain the R.H.S. of Equation (4.27) as follows

$$\pi(\boldsymbol{s}_2)P(\boldsymbol{s}_1|\boldsymbol{s}_2) = \eta \exp\{-\beta(\bar{\Phi}(\boldsymbol{s}_2) + \bar{D}_m(\bar{s}_m, \boldsymbol{s}_{-m}))\} \tag{4.29}$$

Considering that from $\boldsymbol{s}_1$ to $\boldsymbol{s}_2$ there is only one element (i.e., player m's strategy) changed, and that the proposed game is an exact potential game, we have

$$\bar{\Phi}(\boldsymbol{s}_1) - \bar{\Phi}(\boldsymbol{s}_2) = \bar{D}_m(\bar{s}_m, \boldsymbol{s}_{-m}) - \bar{D}_m(\bar{s}'_m, \boldsymbol{s}_{-m}). \tag{4.30}$$

According to (4.28)-(4.30), we can conclude that the stationary distribution shown in (4.26) satisfies the balanced equation (4.27) and therefore is the unique stationary distribution of the proposed SAP algorithm. □

Complexity Analysis: In each round of the SAP algorithm, one player i is chosen to calculate the potential cost over the whole strategy space S_i. The computational complexity is $O(|S_i|) = O(|\mathcal{V}(m_{i1})| \times |\mathcal{V}(m_{i2})| \times \cdots \times |\mathcal{V}(m_{i|F_i|})|)$, which is upper bounded by $O(V^{|F_i|})$, where $|F_i|$ is the number of VNFs managed by player i. Let the maximum number of rounds be r_{max}. Then the total complexity of SAP is given by $O(r_{max} \cdot V^{|F_i|})$.

Optimality Analysis: As for optimality, given sufficiently large β, the SAP algorithm can achieve the global optimal solution with an arbitrarily large probability [116]-[118].

Algorithm 5: Proposed prioritized admission control mechanism

```
1  Calculate $N_{max}$ using (4.31);
2  if $N > N_{max}$ then
3      Sort $\mathcal{N}$ in descending order according to $\lambda_i$;
4      Choose the first $N_{max}$ players in $\mathcal{N}$ to form $\tilde{\mathcal{N}}$;
5  else
6      Set $\tilde{\mathcal{N}} \leftarrow \mathcal{N}$;
7  All players in $\tilde{\mathcal{N}}$ play game $\bar{\mathcal{G}}$ to obtain an NE $\boldsymbol{s}^*$;
8  if $P(\boldsymbol{s}^*) = 0$ is true then
9      if $N > N_{max}$ then
10         while $P(\boldsymbol{s}^*) = 0$ is true do
11             Set $\boldsymbol{s}^{AC} \leftarrow \boldsymbol{s}^*$;
12             Add one more player to $\tilde{\mathcal{N}}$, i.e.,
13             $i^* = \arg\min_{i \in \mathcal{N} \setminus \tilde{\mathcal{N}}} \lambda_i$; $\tilde{\mathcal{N}} = \tilde{\mathcal{N}} \cup \{i^*\}$;
14             All players in $\tilde{\mathcal{N}}$ re-play the game $\bar{\mathcal{G}}$ to obtain a new NE $\boldsymbol{s}^*$;
15         Remove the lastly added player from $\tilde{\mathcal{N}}$;
16         return $\boldsymbol{s}^{AC}$, $\tilde{\mathcal{N}}$;
17     else
18         Set $\boldsymbol{s}^{AC} \leftarrow \boldsymbol{s}^*$;
19         return $\boldsymbol{s}^{AC}$, $\tilde{\mathcal{N}}$;
20 else
21     while $P(\boldsymbol{s}^*) = 0$ is false do
22         Remove one player from $\tilde{\mathcal{N}}$, i.e.,
23         $i^* = \arg\min_{i \in \tilde{\mathcal{N}}} \lambda_i$; $\tilde{\mathcal{N}} = \tilde{\mathcal{N}} \setminus \{i^*\}$;
24         All players in $\tilde{\mathcal{N}}$ re-play the game $\bar{\mathcal{G}}$ to obtain a new NE $\boldsymbol{s}^*$;
25     Set $\boldsymbol{s}^{AC} \leftarrow \boldsymbol{s}^*$;
26     return $\boldsymbol{s}^{AC}$, $\tilde{\mathcal{N}}$;
```

4.3.3 Proposed Prioritized Admission Control Mechanism

When the physical resources are scarce or the traffic load is high, simply adding a penalty term to the cost function may not be able to ensure the capacity constraints of NFV nodes. Therefore, we further design an admission control (AC) mechanism to address over-loaded scenarios where the capacity constraints of NFV nodes are difficult to be guaranteed. The details of the proposed AC mechanism is shown in Algorithm 5.

First, given the statistical characteristics of the NFV system and the SFCs, the maximum allowed number of SFCs that can be embedded, N_{max}, is estimated as

$$N_{max} \approx \frac{E(\mu_v) \cdot V}{E(\lambda_i) \cdot E(|F_i|)}, \tag{4.31}$$

where $E(\mu_v)$ denotes the expectation of the service rate of an NFV node, $E(\lambda_i)$ denotes the expectation of the bit rate of a VNF chain, and $E(|F_i|)$ is the average number of VNFs in a VNF chain. In (4.31), the numerator approximates the total amount of physical resources available in the whole system, while the denominator approximates the amount of physical resources demanded by an VNF chain. Next, the mechanism checks if the actual number of VNF chains to embedded (N) is greater than N_{max}. If we have $N \leq N_{max}$, all the players in $\mathcal{N}$ are sorted in a descending order according to their bit rates (λ_i). Then, the first N_{max} players form the set of VNF chains (denoted by $\tilde{\mathcal{N}}$) that are chosen to be embedded (Line 2-4). Otherwise, we keep the original VNF chain set $\mathcal{N}$ unchanged (Line 5-6). After that, all the players in $\tilde{\mathcal{N}}$ play the game $\bar{\mathcal{G}}$ (with either BR or SAP algorithm) to obtain an NE $\boldsymbol{s}^*$ (Line 7). If all the capacity constraints are satisfied and we have $N \leq N_{max}$, the AC mechanism terminates and outputs the final accommodated SFC set $\tilde{\mathcal{N}}$ and their corresponding embedding strategies $\boldsymbol{s}^{AC}$ (Line 17-19). Otherwise, the mechanism attempts to find the maximum number of SFCs that can be accommodated. Each time the player with the smallest bit rate is chosen and added into $\tilde{\mathcal{N}}$, and the game is replayed to find a new NE (Line 9-16). Similarly, if at least one capacity constraint is violated, one player is removed from $\tilde{\mathcal{N}}$ each time and the game is re-played until all the capacity constraints are satisfied (Line 20-26). Notably, by executing the proposed admission control mechanism, a maximum set of SFCs with relatively high bit rates are accommodated by the system where all the capacity constraints of NFV nodes are satisfied.

4.4 Performance Evaluation

In this section, we evaluate the performance of the proposed algorithms in various scenarios. The parameters in the penalty term in the cost function is set as $w = 1.0$ and $M = 10.0$. Detailed simulation parameters are provided in Table 4.2.

Table 4.2: Parameters Used in Simulations

Parameter	Minimum	Maximum
Total number of VNF types $\lvert\mathcal{M}\rvert$	10	10
Number of VNF types in each SFC $\lvert\mathcal{M}_i\rvert$	3	5
Service rate μ_v	100	200
Latency parameter K_v	0.1	0.5
VNF processing time $\rho(v, m_{ij})$	0.5	1.0
Link propagation delay $\delta(v_1, v_2)$	10.0	20.0
Bit rate λ_i	10.0	20.0

4.4.1 Optimality at Convergence of the Proposed Algorithms

We first validate the optimality at convergence of the proposed algorithms. The exhaustive search (ES) algorithm is used as the benchmark to obtain the global optimal solution. To make the comparison feasible, we consider a small network instance with 5 NFV nodes and 4 SFCs. Each SFC is associated with 3 VNFs. We consider that each NFV node can support all the VNF types in the system, i.e., $\mathcal{M}(v) = \mathcal{M}, \forall v \in \mathcal{V}$. Then, the size of search space for the ES algorithms is $|\mathcal{S}| = (5 \times 4 \times 3)^4 = 12,960,000$. For the SAP algorithm, the exploration parameter β is selected from $[0.5, 1.0, 6.0]$. The maximum number of rounds is set as 100. The parameters of the penalty term in the cost function are set as $w = 1.0$ and $M = 10.0$. The convergence behavior of the proposed BR and SAP algorithms is shown in Fig. 4.4. As can be seen from the figure, after a small number of rounds, both BR algorithm and SAP algorithm (with $\beta = 6.0$) converge to the global optimal solution found by the ES algorithm. Notice that for this small network scenario, the SAP algorithm converges to the global optimal solution as quickly as the BR algorithm does. On the other hand, after 100 rounds, the SAP algorithm with $\beta = 0.5$ or $\beta = 1.0$ has not converged. To achieve convergence, a larger number of rounds are needed. Notably, a larger value of exploration parameter (β) leads to a faster convergence of the SAP algorithm.

4.4.2 Performance Comparison Between Proposed Algorithms

Next, the performance of the two proposed algorithms are compared in terms of the ability of finding the best NE. Consider an NFV system with 8 nodes and 20 SFCs (i.e., $V = 8$ and $N = 20$). The number of VNFs in each SFC is a random number chosen from $[3, 4, 5]$. The parameters in the penalty term are set as $w = 1.0$ and $M = 10.0$. For both algorithms, the total number of rounds is set to be 250 to ensure convergence. For the SAP algorithm,

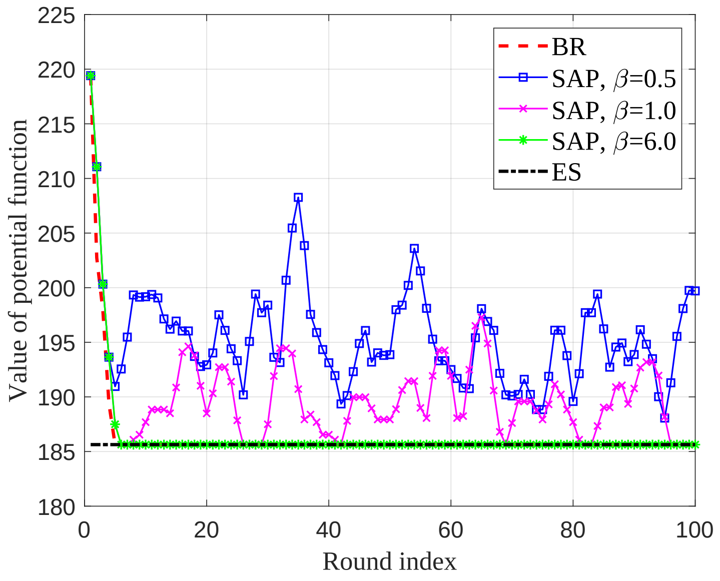


Figure 4.4: Performance of the BR and SAP algorithms at convergence, where $N = 4$, $V = 5$, and $|F_i| = 3$, $\forall i \in \mathcal{N}$.

the initial value of the exploration parameter is chosen from $[0.5, 1.0, 2.0, 4.0]$. Fig. 4.5 and Fig. 4.6 compare the performance of minimizing the potential function and the sum of SFCs' latency between the two algorithms, respectively. To achieve an NE of the proposed game, the BR algorithm takes about 50 rounds, while the SAP algorithm (with $\beta = 2.0$ or $\beta = 4.0$) requires about 110 rounds. However, the values of the potential function and the sum of overall latency achieved by SAP are both smaller than that achieved by BR, which demonstrates that the SAP algorithm has greater potential to achieve the best NE of the proposed game. Moreover, it is seen that a larger value of β leads to a faster convergence of the SAP algorithm.

Fig. 4.7 and Fig. 4.8 show the dynamic evolution of the costs of five chosen players (i.e. Player 1, 3, 5, 7, and 9) for BR and SAP (with $\beta = 4.0$) algorithms, respectively. From Fig. 4.7, we can see that, with BR algorithm the costs of the five players keep decreasing as the round index increases, reaching to an equilibrium after about 50 rounds. The final cost of each player, which corresponds to the total delay experienced by each SFC, at the NE depends on the final embedding strategy chosen by the SFC. A similar tendency can be observed from Fig. 4.8, where the cost of each player oscillates at the initial phase and converges to a stable value after 100 rounds.

Fig. 4.9 and Fig. 4.10 illustrate the evolution of the VNF embedding strategies of two

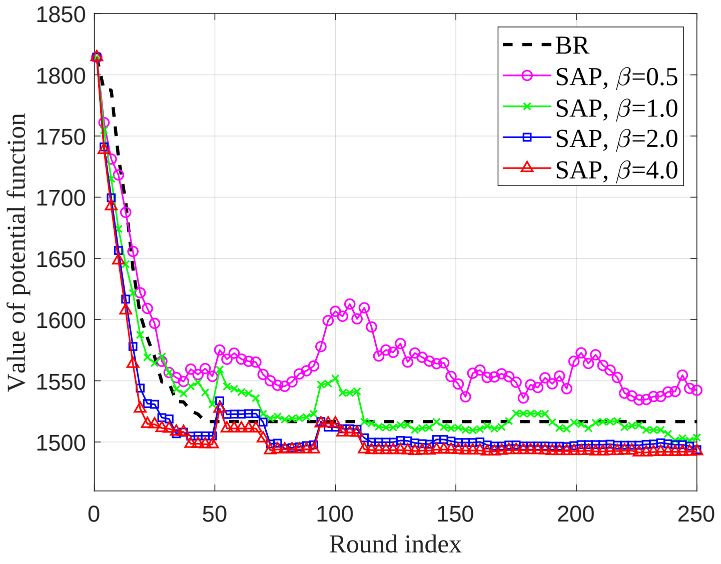


Figure 4.5: Comparison between BR and SAP algorithms in terms of the value of potential function.

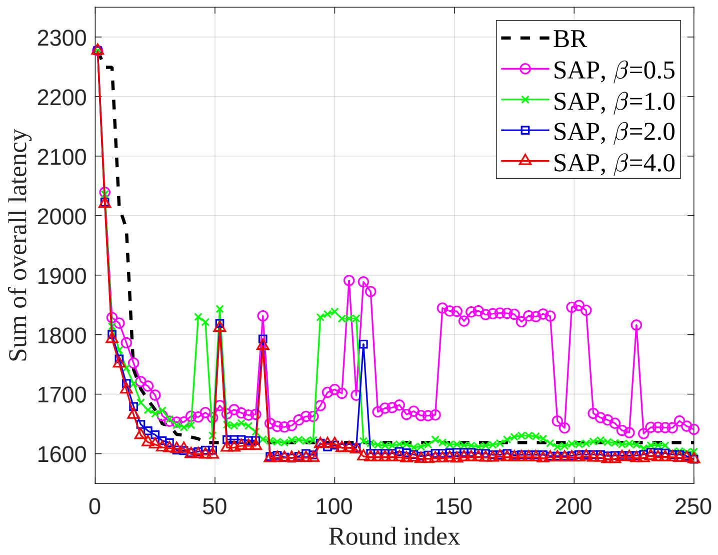


Figure 4.6: Comparison between BR and SAP algorithms in terms of the sum of overall latency for all the SFCs.

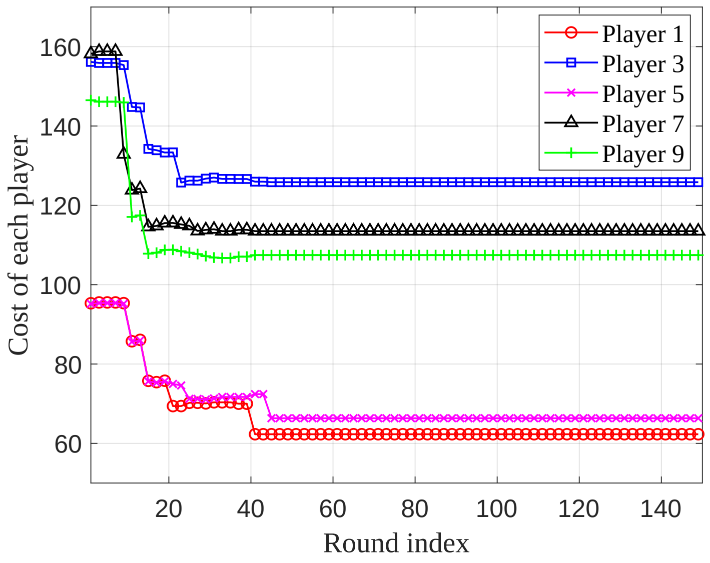


Figure 4.7: Dynamic evolution of the costs of the five chosen players when using the BR algorithm for the scenario $N = 20$, $V = 8$, and $|F_i| \in \{3, 4, 5\}$.

selected players (i.e., player 1 and player 6) using BR and SAP algorithms, respectively. Both players are associated with three VNFs. With the BR algorithm, it can be seen that the VNF embedding strategies of both players keep unchanged after 40 rounds. With the SAP algorithm, it is shown in Fig. 4.10 that the strategies of both players remain unchanged after about 150 rounds. These results further demonstrate that both algorithms are able to obtain an NE of the proposed multi-SFC embedding game.

To show that the proposed game can effectively handle capacity constraints, we present the dynamic evolution of the penalty term (i.e., $P(\bar{s}_i, \boldsymbol{s}_{-i})$) for BR and SAP algorithms in Fig. 4.11 and Fig. 4.12, respectively. A larger value of penalty term indicates a higher violation degree of the capacity constraints of the NFV nodes. It can be observed from Fig. 4.11 that the capacity constraints of two NFV nodes (we set $w = 1.0$ and $M = 10.0$) are violated with the initial strategies of the players. After 15 rounds, the BR algorithm finds an embedding strategy where the capacity constraints of all the NFV nodes can be satisfied (i.e., $P(\bar{s}_i, \boldsymbol{s}_{-i}) = 0$). For the SAP algorithm, we can see from Fig. 4.12 that the number of NFV nodes with violated capacity constraints fluctuates between 0 and 2, and stabilizes at 0 after about 120 rounds for all the three exploration parameter settings. These results demonstrate that our proposed approach is able to ensure the capacity constraints of NFV nodes effectively.

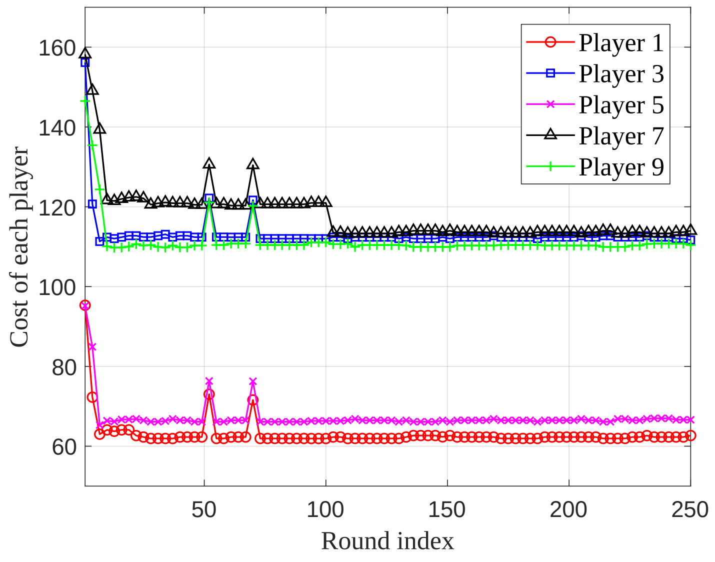


Figure 4.8: Dynamic evolution of the costs of the five chosen players when using the SAP algorithm with $\beta = 4.0$ for the scenario $N = 20$, $V = 8$, and $|F_i| \in \{3, 4, 5\}$.

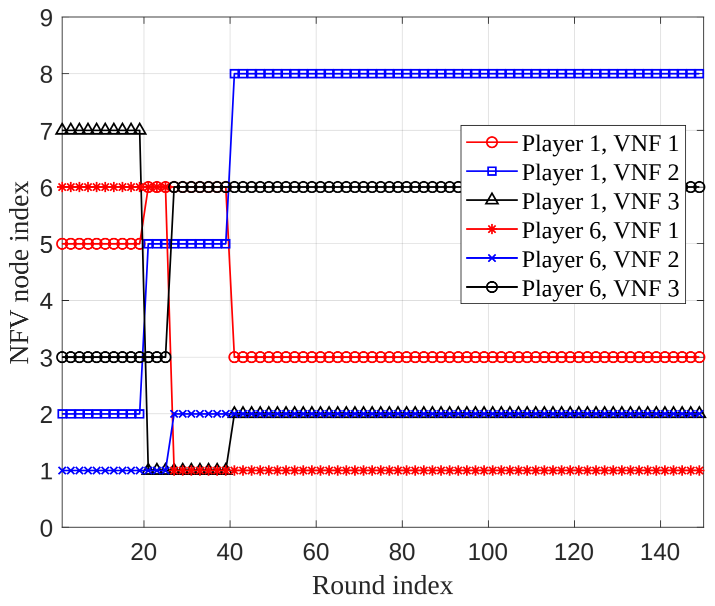


Figure 4.9: VNF embedding strategy evolution using the BR algorithm.

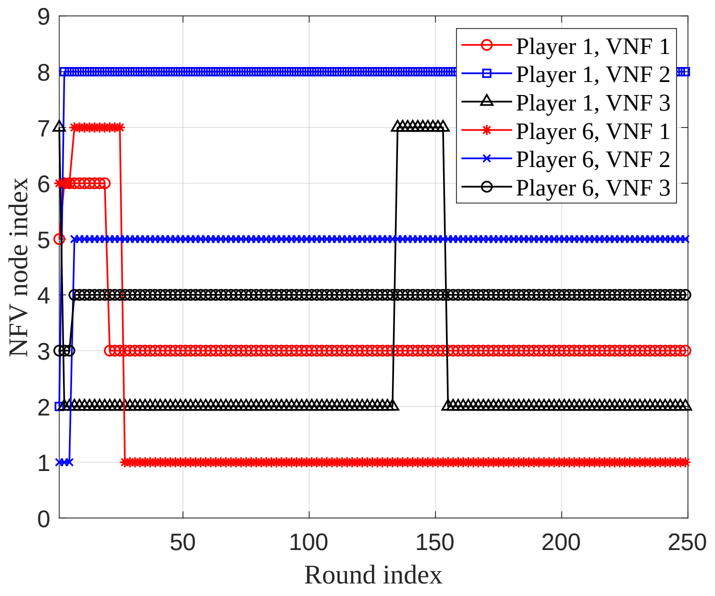


Figure 4.10: VNF embedding strategy evolution using SAP algorithm ($\beta = 4.0$).

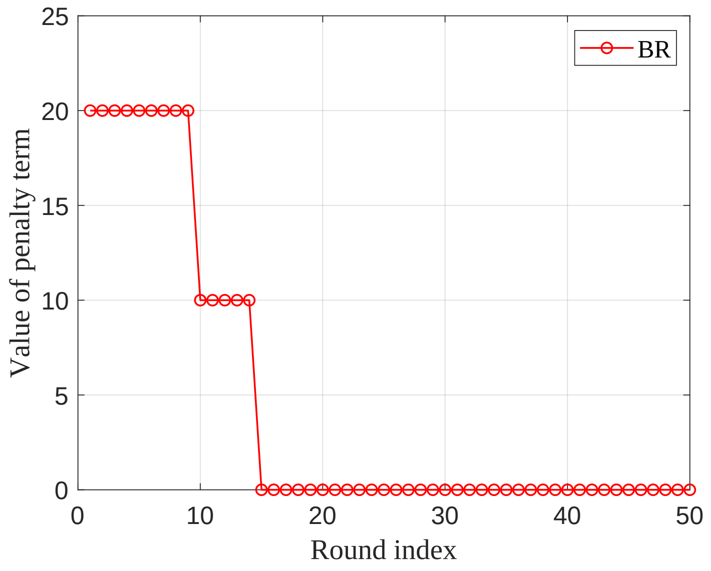


Figure 4.11: Dynamic evolution of the penalty term of the BR algorithm.

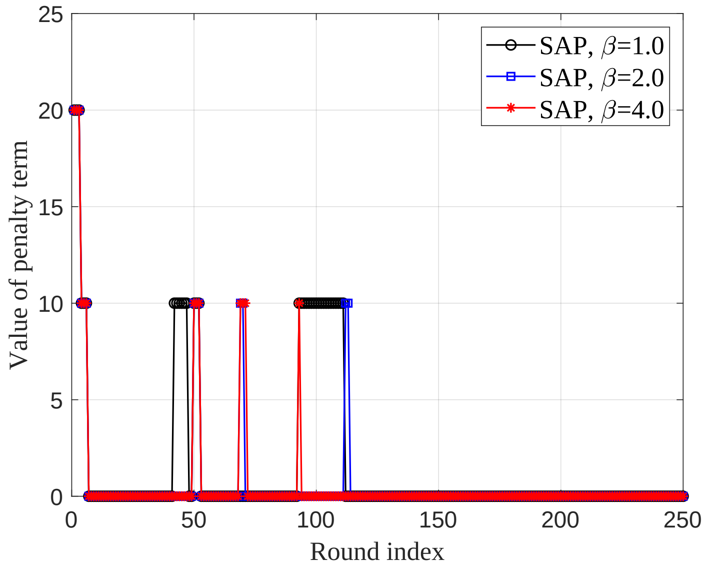


Figure 4.12: Dynamic evolution of the penalty term of the SAP algorithm.

4.4.3 Performance Verification of the Proposed Admission Control Mechanism

To validate the effectiveness of the proposed prioritized AC mechanism, we consider another network scenario where the number of SFCs to be embedded is 22. We adopt the resource utilization ratio at each NFV node (defined as the ratio of total traffic load on the NFV node over the service rate of that node) as the performance metric. Fig. 4.13 compares the performance between the BR algorithm with the proposed AC mechanism and that without AC mechanism. By using the AC mechanism, 20 SFCs are embedded at the NE point. It can be seen from the figure that, without the AC mechanism, there are three NFV nodes whose the capacity constraints are violated at the NE. With the AC mechanism applied, all the NFV nodes have their capacity constraints satisfied at the NE, demonstrating the effectiveness of our proposed AC mechanism. Moreover, with the proposed AC mechanism, the resource utilization ratios of all NFV nodes are quite similar to each other and close to 100%, which demonstrate better traffic load balancing among NFV nodes for the proposed approach.

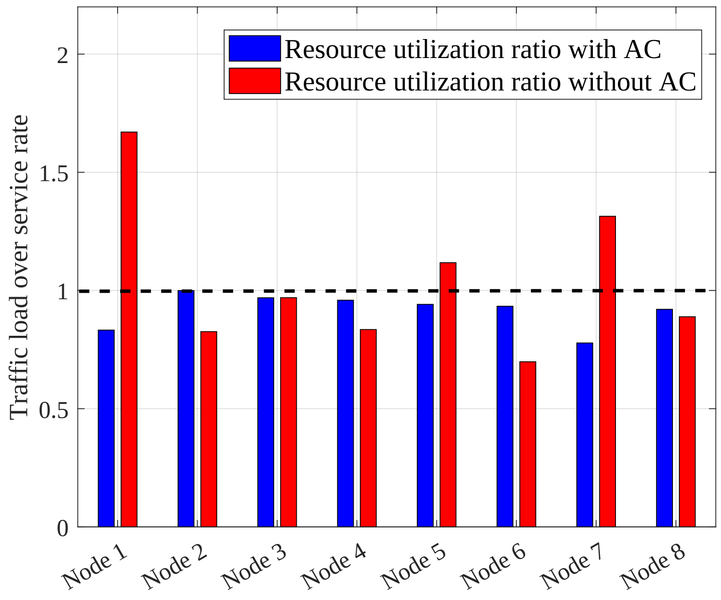


Figure 4.13: Comparison of resource utilization ratio between the BR algorithm with admission control and that without admission control.

4.5 Summary

In this chapter, we have proposed a game theoretical approach to deal with the multi-SFC embedding problem whose objective is minimizing the E2E latency for each service while satisfying capacity constraints of NFV nodes. The problem is formulated as resource-specific congestion game, where a penalty term is added to the cost function of each player to involve the impact of capacity constraints. The proposed game RC-MSEG is proved to be an exact potential game which admits at least one pure strategy NE. Two iterative algorithms are devised to find the NE of RC-MSEG, which corresponds to the local/global optimal solution for the multi-SFC embedding problem. A novel prioritized admission control mechanism is proposed to handle over-loaded scenarios where the violation of the capacity constraints is avoided. Simulations are carried out to validate the effectiveness of the proposed approach. The proposed approach achieves low latency for SFC provisioning with reduced computational complexity in an NFV-enabled future network.

Chapter 5

Delay-Aware VNF Scheduling

Software defined networking (SDN) and network function virtualization (NFV) are the key enabling technologies for service customization in next generation networks to support various applications. In such a circumstance, virtual network function (VNF) scheduling plays an essential role in enhancing resource utilization and achieving better quality-of-service (QoS). In this chapter, the VNF scheduling problem is investigated to minimize the makespan (i.e., overall completion time) of all services, while satisfying their different end-to-end (E2E) delay requirements. The problem is formulated as a mixed integer linear program (MILP) which is NP-hard with exponentially increasing computational complexity as the network size expands. To solve the MILP with high efficiency and accuracy, the original problem is reformulated as a Markov decision process (MDP) problem with variable action set. Then, a reinforcement learning (RL) approach is developed to learn the best scheduling policy by continuously interacting with the network environment. The proposed learning algorithm determines the variable action set at each decision-making state and accommodates different execution time of the actions. The reward function in the proposed algorithm is carefully designed to realize delay-aware VNF scheduling. Simulation results are presented to demonstrate the convergence and high accuracy of the proposed approach against other benchmark algorithms.

5.1 Introduction

With the fast evolvement of communication technologies, the 5G wireless networks are anticipated to accommodate a massive number of Internet-of-Things (IoT) devices with highly diversified quality-of-service (QoS) requirements [1], [9]. Conventional network functions are placed at function-specific network servers or middleboxes, which are cost-ineffective for differentiated service customization since a large number of servers need to be augmented to support different types of services. In addition, the traffic routing paths among network elements are distributedly calculated with considerable overhead, which is not efficient in achieving high network performance with load balancing [25], [38]. Integrating SDN and NFV for future networking is expected to feature centralized network management, virtualized service function chaining, reduced costs, and enhanced service quality [10].

One of the fundamental research issues in NFV is how to efficiently and fairly allocate physical resources in the substrate network to support the embedding of multiple VNF chains, referred to as the NFV resource allocation (NFV-RA) problem [32]. The NFV-RA problem typically consists of VNF composition, VNF chain embedding, and VNF scheduling on embedded NFV nodes. During the embedding process, multiple VNFs can be placed onto a common NFV-enabled network commodity server (i.e., NFV node) to reduce function provisioning cost and improve physical resource utilization. In VNF scheduling, the execution timings of embedded VNFs on NFV nodes are scheduled to minimize the makespan (i.e., the time period from the execution of the first VNF to the completion of last VNF among all the scheduled VNFs for all the services). Existing studies have shown that the classical VNF scheduling problem can be formulated as a job-shop problem (JSP) [64], [121], which is an NP-hard combinatorial optimization problem [122]. To obtain its near-optimal solutions, a number of heuristic/metaheuristic algorithms have been developed [62]-[66], [123]. These heuristic algorithms (e.g., greedy algorithms) are in general fast and easy to implement, but their performance highly depends on the characteristics of the problem and may deteriorate as the network size expands. On the other hand, the metaheuristic algorithms such as particle swarm optimization (PSO) and genetic algorithm (GA) may suffer a low convergence rate in the iterative process, leading to increased computational cost and operational time. Their performance also relies on the initialization of parameters and may converge prematurely, falling into a local optimum especially for complex problems. Moreover, strict E2E delay requirements are important to 5G service provisioning. The incorporation of E2E delay constraints for different services in the standard JSP problem poses additional challenge to conventional VNF scheduling. With different E2E delay requirements, the existing heuristic algorithms for standard JSP need to be carefully adjusted to achieve delay-constrained VNF scheduling. This requires manual designs for appropriate rules of reducing the search space for the heuristic algorithms.

Recently, reinforcement learning (RL) has emerged as a promising approach to solve combinatorial optimization problems with reduced complexity and high accuracy [67]-[72],[124]. In RL approaches, a learning agent learns the best policy iteratively by directly interacting with the environment, and multiple objectives can be supported simultaneously by designing an appropriate reward function for the learning system. In this way, the useful information for solving the problem at hand can be automatically extracted. The delay-aware VNF scheduling problem has a long-term objective which is to minimize the overall makespan. Also, the task of scheduling the VNFs in the system can be considered as a sequential decision process where the future status of the system depend on the current status and decision only. Therefore, the VNF scheduling problem can be naturally modeled as a Markov decision process (MDP). This motivates us to employ RL to obtain near-optimal/optimal solutions to delay-aware VNF scheduling with reduced computational complexity.

In this chapter, the delay-guaranteed VNF scheduling problem is first formulated as a mixed-integer linear program (MILP) which is NP-hard. To solve the MILP with high efficiency and accuracy, the original problem is reformulated as a Markov decision process problem. Then, we develop an RL framework to address the VNF scheduling problem with E2E delay guarantee. With the SDN/NFV-enabled network architecture, the logically centralized VNF scheduler has global network information and makes scheduling decisions. Based on this, we propose a centralized learning algorithm to solve the VNF scheduling problem. The action of the agent (i.e., the VNF scheduler) is improved through continuously interacting with the network environment. Different from conventional decentralized approaches for standard job-shop problems [69]-[71], the centralized RL improves the solution accuracy of the VNF scheduling, thanks to the global view of the whole system. Also, without the coordination message overhead between multiple agents, the solution complexity is reduced [72]. In the proposed RL approach, the action set for each state is finite, state-dependent, and fixed for a given state. At each decision-making state, a feasible/admissible action set is updated with the consideration of VNF dependency. Each action can take varying amount of time to complete, i.e., the length of the time period between two decision-making states is not fixed, depending on both states and actions taken. The effectiveness of our proposed scheduling algorithm is evaluated under different network scales and is compared with other heuristic algorithms. The main contributions of this work are four-fold:

1. We formulate the VNF scheduling problem as an MILP, considering the E2E delay requirements of differentiated services. To solve it, we transform the formulation into a discrete-time problem, where the VNF scheduling decisions are made at the beginning of each time slot;

2. A novel Q-learning based VNF scheduling algorithm is developed, in which system states, actions, variable action set, reward functions are designed for the learning algorithm. In conventional RL approaches, the feasible action set for the agent is

state-independent and the execution time for all the actions are identical. However, in VNF scheduling, the action set for the agent are state-dependent since a VNF can be traversed only after its previous VNF has been passed through by the packet batch, and each action can take a varying amount of time to complete. Thus, our proposed Q-learning algorithm determines the variable action set at each decision-making state and accommodates diverse execution time of the actions;

3. The RL approach efficiently guarantees the QoS requirements for different services via a customized reward function that is composed of two parts. The first part reflects the makespan, where a short makespan leads to a large reward; The second part reflects whether the E2E delay requirements of the services are satisfied. If the delay requirement is satisfied for a given service, a positive reward is fed back to the agent. By iteratively accumulating the reward, the agent learns the optimal scheduling policy that results in the minimal makespan with E2E delay guarantee.

4. Extensive simulations are conducted to demonstrate the convergence of the proposed RL algorithm and to compare the performance of the proposed approach with those of several heuristic/metaheuristic algorithms and the decentralized approach. Simulation results show that our proposed approach can address the delay-aware VNF scheduling problem effectively with reduced complexity and high accuracy, and that the RL approach outperforms the benchmark algorithms in comparison.

5.2 System Model

Consider an SDN/NFV-enabled network architecture as shown in Fig. 5.1, where the control plane is decoupled from the data plane and migrated to a centralized SDN control module. The SDN controller is logically centralized and can be physically deployed in a decentralized way, in which case some information change between different local SDN controllers is required. The control module orchestrates the placement of VNFs from different VNF chains and configures traffic routing between consecutive VNFs for load balancing in the data plane. Through open southbound Interface (SBI) between the control module and the substrate network, the state information from the underlying substrate network can be collected. Multiple types of services, e.g., machine-type services with stringent E2E delay requirements and data services with non-stringent delay requirements, are considered and supported by different VNF chains. These services manifest different levels of delay requirements [3]. In the SDN/NFV-enabled network architecture, service customization and isolation should be achieved by embedding different VNF chains onto the

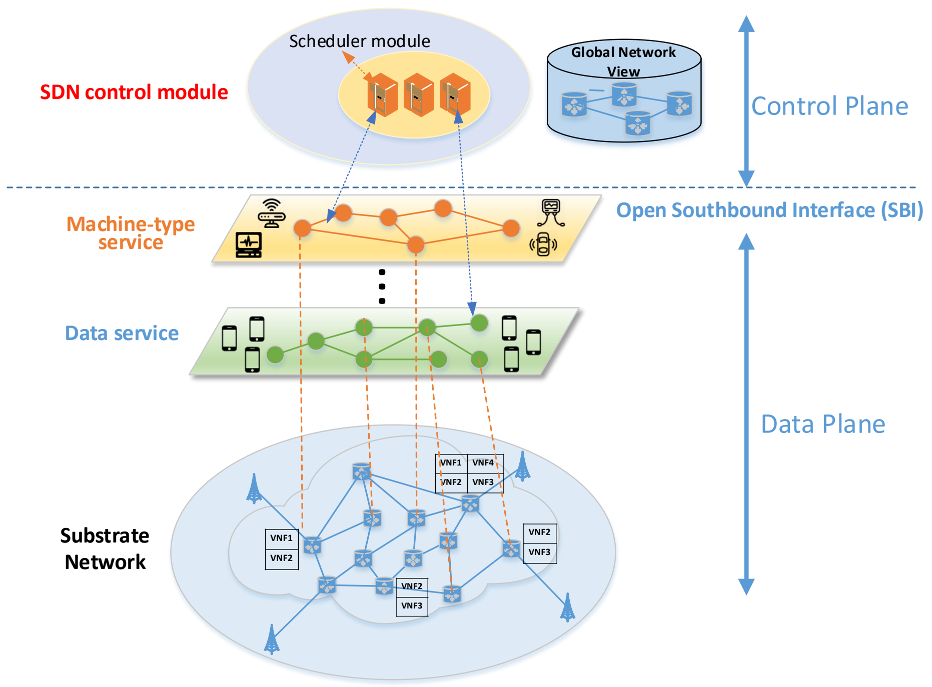


Figure 5.1: An SDN/NFV-enabled network architecture.

same physical substrate network. VNF scheduling is periodically performed by the VNF scheduler (as a sub-module integrated inside the SDN control module) as service requests arrive. All the arrived service requests are scheduled simultaneously at the beginning of every scheduling period. Important parameters and variables of the system model are listed in Table 5.1.

Substrate network – The substrate network under consideration is represented by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, where $\mathcal{V}$ denotes the set of physical nodes in the network and $\mathcal{L}$ represents the set of physical links connecting any pair of nodes. Define $\mathcal{N}(\subseteq \mathcal{V})$ as the set of NFV nodes, and NFV node $n_k \in \mathcal{N}$ ($k \in \mathbb{N}$, where $\mathbb{N} = \{1, 2, \ldots, |\mathcal{N}|^1\}$) is capable of hosting multiple VNFs of different types. Denote the CPU processing capacity of n_k for a VNF f as $c_k(f)$.

VNF chain – A network service is composed of an ordered sequence of VNFs, referred to as a VNF chain. Suppose that there are $|\mathcal{R}|$ services, denoted by $S_1, S_2, \ldots, S_{|\mathcal{R}|}$, and

$^1|\cdot|$ represents the cardinality of a finite set.

Table 5.1: Summary of Important Notations and Decision Variables

Notation	Description
$\mathcal{R}$	set of network services
S_i	ith network service
D_i	processing time deadline of S_i for one packet batch
$\mathcal{F}$	set of VNFs in all the services
$\mathcal{F}_i$	set of VNFs in S_i
f_{ij}	jth VNF in S_i
$\mathcal{N}$	set of NFV nodes
n_k	kth NFV node
$\mathcal{I}_{ij}$	set of NFV nodes that can support VNF f_{ij}
ρ_{ijk}	packet batch processing time at $f_{ij} \in \mathcal{F}_i$ on node n_k
y_{ijk}	binary constant indicating if f_{ij} is assigned to node n_k
T	total number of time slots
τ	duration time of a time slot
$\mathcal{M}$	overall makespan
M	a big positive number
Decision variable	**Description**
t^s_{ij}	time when the packet batch of S_i starts processing at f_{ij}
x^t_{ik}	binary variable indicating whether or not S_i starts being supported on n_k at the tth time slot
$z^k_{ij,pq}$	binary variable indicating whether f_{ij} starts execution before f_{pq}, if f_{ij} and f_{pq} are both embedded onto n_k

define $\mathbb{R} = \{1, 2, \ldots, |\mathcal{R}|\}$. Denote the set of VNFs in S_i by $\mathcal{F}_i$. Let f_{ij} represent the jth VNF in $\mathcal{F}_i$, where $j \in \mathbb{F}_i$ and $\mathbb{F}_i = \{1, 2, \ldots, |\mathcal{F}_i|\}$. Let $\mathcal{I}_{ij}$ be the set of NFV nodes that can support VNF f_{ij}.

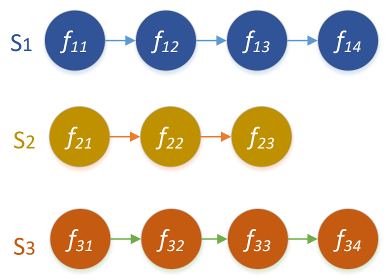
Traffic model – Each NFV node is assumed to have a number of processing queues, where each packet from a specific service is buffered for processing at certain VNF embedded onto the NFV node. Packets of service S_i arrive at one of the processing queues at the first NFV node of an embedded VNF chain as a Poisson process with arrival rate λ_i (packet/s). When multiple VNFs are embedded at a common NFV node, it is required to determine the scheduling sequence of VNFs for packet processing since each VNF is scheduled for packet processing one at a time. In a real system, the computing resources on each NFV node are indivisible. Each NFV node can only support at most one VNF at a time and all the computing resources are allocated to the VNF for packet batch processing [32], [62], [66], [125]. To reduce the switching overhead for VNF scheduling, we assume that a VNF is scheduled for packet processing only if its associated buffer occupancy is above a threshold B. The number of packets to be processed for one-time VNF scheduling is B, which is also called one *packet batch*. Thus, the processing time of one packet batch at f_{ij} on node n_k ($n_k \in \mathcal{I}_{ij}$) for service S_i is given by

$$\rho_{ijk} = B/c_k(f_{ij}). \tag{5.1}$$

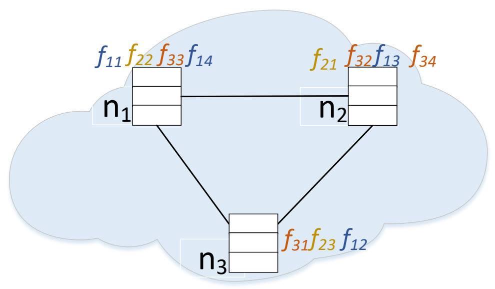
VNF scheduling – VNF scheduling is performed cyclically. We define the *makespan* of one VNF scheduling cycle as the time duration that all embedded VNFs are scheduled once for processing a single packet batch. For simplicity, we assume that the processing queue occupancy of a subsequent VNF on an NFV node is above the threshold B once the previous VNF scheduling completes[2]. For the first scheduling cycle, we assume that VNFs at different NFV nodes start to be scheduled at the same time instant (i.e., the time instant 0 in Fig. 5.2(c)). The scheduling starting and completion time for a packet batch of S_i traversing f_{ij} in a scheduling cycle are represented by t^s_{ij} and t^c_{ij}, respectively. Between t^c_{ij} and $t^s_{i,j+1}$, the packet batch waiting time can exist due to packet queueing on the NFV node where $f_{i,j+1}$ is embedded. The packet batch of S_i has to be processed at a chain of VNFs in a predefined order, with the duration of ($t^s_{i|\mathcal{F}_i|} + \rho_{i|\mathcal{F}_i|k}$) for traversing all services' VNFs in a scheduling cycle. The scheduling results include two parts: 1) For service S_i, we determine the time when a packet batch starts being processed at f_{ij}; 2) For NFV node n_k, we determine the scheduling sequence of all embedded VNFs. Let D_i denote the maximum acceptable processing time for a packet batch of S_i passing through the VNF chain in one scheduling cycle. The VNF scheduling process is to jointly determine the scheduling sequence and the starting time for packet processing for the VNFs of all services to minimize the makespan. By conforming to an optimal scheduling sequence, each cycle of VNF scheduling proceeds repeatedly to reduce the overall processing delay of packets traversing each VNF chain.

A simple illustration of the VNF scheduling process is shown in Fig. 5.2. Suppose that there are three VNF chains of network services S_1, S_2, S_3 (labeled by different colours) to be scheduled at each scheduling cycle. On each NFV node, the VNF scheduling sequence has to be determined to minimize the makespan. The VNF chain embedding results are shown in Fig. 5.2(b). In Fig. 5.2(c), packet batches start traversing the VNFs of S_1, S_2, and S_3 at nodes n_1, n_2, and n_3, respectively, from time instant 0. Packet batch processing

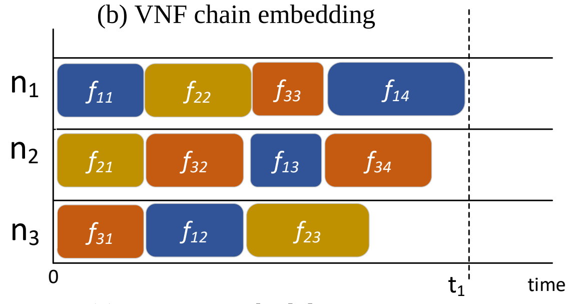
[2]Dynamic VNF scheduling by considering the buffer occupancy status will be studied in our future work.


S1
f_{11} f_{12} f_{13} f_{14}
S2
f_{21} f_{22} f_{23}
S3
f_{31} f_{32} f_{33} f_{34}

(a) Three VNF chains for embedding and scheduling


f_{11} f_{22} f_{33} f_{14}
f_{21} f_{32} f_{13} f_{34}
n_1
n_2
n_3
f_{31} f_{23} f_{12}

(b) VNF chain embedding



(c) One VNF scheduling pattern

Figure 5.2: A simple example to illustrate the VNF scheduling process.

time at each scheduled VNF is also displayed by different colours. Time instant t_1 indicates the makespan of one VNF scheduling cycle.

5.3 Problem Formulation

In this section, the delay-aware VNF scheduling problem is presented by a continuous-time formulation, in which the timings of the scheduling process are clearly described. For the continuous-time representation, additional binary variables are required to indicate the processing sequence of the scheduled batches, which complicates the mathematical formulation and the way of solving the problem. Hence, we transform the continuous-time formulation into a discrete-time interpretation in a time-slotted system, where the VNF scheduling decisions are made at the beginning of each time slot [126]. For both formulations, we incorporate delay constraints to achieve E2E delay guaranteed service provisioning.

5.3.1 Continuous-Time Formulation

Objective – The objective of the VNF scheduling problem (P1) is to minimize the makespan $\mathcal{M}$, which is expressed as

$$\min_{t^s_{ij}, z^k_{ij,pq}, z^k_{pq,ij}} \mathcal{M} \tag{5.2}$$

where the makespan $\mathcal{M}$ is the time duration of one VNF scheduling cycle for all services, given by

$$\mathcal{M} = \max_{i \in \mathbb{R}, j \in \mathbb{F}_i} \{t^s_{ij} + \sum_{n_k \in \mathcal{I}_{ij}} y_{ijk}\rho_{ijk}\}. \tag{5.3}$$

In (5.3), binary parameter y_{ijk} is introduced, where $y_{ijk} = 1$ indicates f_{ij} is embedded at server n_k; otherwise, $y_{ijk} = 0$.

Constraints – The following constraints are imposed to guarantee the feasibility of the VNF scheduling:

1) If any two VNFs (e.g., f_{ij} and f_{pq}) are embedded onto the same NFV node, it is required that the packet batch processing at one of the two VNFs cannot start before the processing at the other one finishes [64]. To impose these constraints, we define an auxiliary binary variable $z^k_{ij,pq}$ as

$$z^k_{ij,pq} = \begin{cases} 1 & \text{if } f_{ij} \text{ starts packet batch processing before} \\ & f_{pq} \text{ on NFV node } n_k, \\ 0 & \text{otherwise.} \end{cases}$$

Then, the constraints ensuring that preemption is not allowed at any time on any NFV node are expressed as

$$t^s_{ij} + z^k_{ij,pq} \sum_{n_k \in \mathcal{N}} y_{ijk}\rho_{ijk} \leq t^s_{pq} + z^k_{pq,ij} M, \tag{5.4}$$

$$t^s_{pq} + z^k_{pq,ij} \sum_{n_k \in \mathcal{N}} y_{pqk}\rho_{pqk} \leq t^s_{ij} + z^k_{ij,pq} M, \tag{5.5}$$

$$z^k_{ij,pq} + z^k_{pq,ij} = 1 \tag{5.6}$$

where $p \in \mathbb{R}$, $q \in \mathbb{F}_p$, $p \neq i$ or $q \neq j$ and M is a big positive number [98]. Note that since M is large, constraint (5.4) is non-restrictive when $z^k_{ij,pq} = 0$ and $z^k_{pq,ij} = 1$ (i.e., when f_{pq} starts packet batch processing before f_{ij}), and constraint (5.5) is non-restrictive when $z^k_{ij,pq} = 1$ and $z^k_{pq,ij} = 0$ (i.e., when f_{pq} starts packet batch processing after f_{ij}).

2) The specified processing sequence of the VNFs in each service, $f_{i1} \to f_{i2}, \ldots, \to f_{i|\mathcal{F}_i|}$, $\forall i \in \mathbb{R}$, is enforced by [64]

$$t^s_{i,j+1} - t^s_{ij} \geq \sum_{n_k \in \mathcal{I}_{ij}} y_{ijk}\rho_{ijk}, \qquad \forall i \in \mathbb{R}, j \in \mathbb{F}_i. \tag{5.7}$$

Constraint (5.7) guarantees that the processing of a packet batch at a subsequent VNF cannot start until the processing at its previous VNF is completed.

3) The duration time for a packet batch passing through the VNF chain of S_i in one scheduling cycle should satisfy

$$t^s_{i|\mathcal{F}_i|} + \sum_{n_k \in \mathcal{N}} y_{i|\mathcal{F}_i|k}\rho_{i|\mathcal{F}_i|k} \leq D_i, \qquad \forall i \in \mathbb{R}. \tag{5.8}$$

Therefore, the continuous time formulation for the delay-guaranteed VNF scheduling problem (P1) is presented as an MILP program, given by

$$\begin{aligned}
\min_{t^s_{ij}, z^k_{ij,pq}, z^k_{pq,ij}} \quad & \mathcal{M} \\
\text{s.t.} \quad & (5.4) - (5.8), \\
& t^s_{ij} \geq 0, \\
& z^k_{ij,pq} \in \{0,1\}, \\
& z^k_{pq,ij} \in \{0,1\}.
\end{aligned}$$

5.3.2 Discrete-Time Transformation

The continuous-time formulation presented above requires additional binary variables (i.e., $z^k_{ij,pq}$ and $z^k_{pq,ij}$) to indicate the packet batch processing sequence for scheduling VNFs. This makes both the mathematical formulation and the way of solving the problem more complicated. In addition, the difficulty of the algorithm design increases when continuous variables are involved. To overcome this problem, a discrete-time transformation of the continuous-time formulation is presented in this section.

Time is divided into T time slots of equal and fixed duration time τ [65]. It is assumed that a packet batch can only start processing at one VNF at the beginning of a certain time slot, and the processing time is an integer multiple of one time slot. We define binary variable x_{ik}^t to indicate the packet batch processing state of service i on NFV node n_k, where $x_{ik}^t = 1$ indicates the packet batch of service S_i starts being processed at the VNF embedded on NFV node n_k at (the beginning of) time slot t, and $x_{ik}^t = 0$ otherwise. The discrete time formulation for the VNF scheduling problem (P2) is presented as follows:

Objective – The objective is to minimize the makespan of packet batch processing for all services, given by

$$\min_{x_{ik}^t} \ \mathcal{M} \tag{5.9}$$

where

$$\mathcal{M} = \max_{i \in \mathbb{R}} \{ \sum_{k=1}^{|\mathcal{N}|} \sum_{t=1}^{T} x_{ik}^t y_{i|\mathcal{F}_i|k}((t-1)\tau + \rho_{i|\mathcal{F}_i|k}) \} \tag{5.10}$$

Constraints:

$$x_{ik}^t y_{ijk} + \sum_{i' \in \mathbb{R}, i' \neq i} x_{i'k}^{t'} \leq 1, \quad \forall i \in \mathbb{R}, \forall j \in \mathbb{F}_i, \forall k \in \mathbb{N}, \tag{5.11}$$

$$\forall t, t' \in [1, T], t \leq t' < t + \sum_{n_k \in \mathcal{I}_{ij}} y_{ijk} \rho_{ijk}$$

$$\sum_{k=1}^{|\mathcal{N}|} \sum_{t=1}^{T} x_{ik}^t (y_{i(j+1)k} - y_{ijk})(t-1)\tau \geq \sum_{k=1}^{|\mathcal{N}|} \sum_{t=1}^{T} x_{ik}^t y_{ijk} \rho_{ijk},$$
$$\forall i \in \mathbb{R}, \forall j \in \mathbb{F}_i \setminus \{|\mathcal{F}_i|\} \tag{5.12}$$

$$\sum_{i=1}^{|\mathcal{R}|} x_{ik}^t \leq 1, \quad \forall t \in [1, T], \forall k \in \mathbb{N} \tag{5.13}$$

$$\sum_{k=1}^{|\mathcal{N}|} \sum_{t=1}^{T} x_{ik}^t y_{ijk} = 1, \quad \forall i \in \mathbb{R}, \forall j \in \mathbb{F}_i \tag{5.14}$$

$$\sum_{k=1}^{|\mathcal{N}|} \sum_{t=1}^{T} x_{ik}^t y_{i|\mathcal{F}_i|k}((t-1)\tau + \rho_{i|\mathcal{F}_i|k}) \leq D_i, \quad \forall i \in \mathbb{R} \tag{5.15}$$

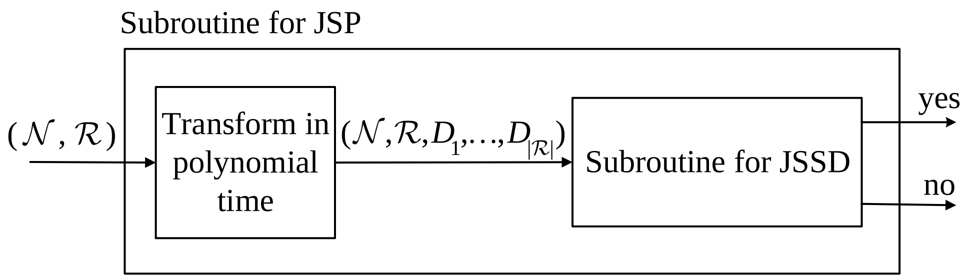

Subroutine for JSP
$(\mathcal{N}, \mathcal{R})$
Transform in polynomial time
$(\mathcal{N},\mathcal{R},D_1,\ldots,D_{|\mathcal{R}|})$
Subroutine for JSSD
yes
no

Figure 5.3: Illustration of reducing JSP to JSSD.

where T is the total number of time slots, and is a large number to ensure the completion of one packet batch processing for all services. $[1, T]$ represents the set of integers between 1 and T. Constraint (5.11) indicates that packet batch processing is conducted for only one VNF at a time on an NFV node [65]; Constraint (5.12) indicates that a packet batch processing at a VNF cannot start until its previous VNF completes the processing; Constraint (5.13) guarantees that at any time slot t, the packet batch of at most one service is processed by a NFV node; Constraint (5.14) ensures that the packet batch processing at a VNF will not be repetitively conducted; Constraint (5.15) is the processing time deadline constraint of one packet batch for each service.

From (5.2)-(5.8), it is observed that Problem (P1) involves continuous variables t^s_{ij} as well as binary variables $z^k_{ij,pq}$ and $z^k_{pq,ij}$. Therefore, (P1) is an MILP. The objective is to minimize the overall makespan while respecting the precedence relations between VNFs and satisfying the delay requirements of the services. In fact, the formulation of (P1) falls into the category of job-shop scheduling problem with deadlines (JSSD).

Remark. JSSD is NP-hard.

Proof. The classical JSP is NP-complete [138], and it cannot be solved in polynomial time. In our work, we will show that JSP with deadlines (JSSD) is NP-hard, if JSP is NP-complete, i.e.,

$$(JSP \notin P) \Rightarrow (JSSD \notin P). \tag{5.16}$$

where P represents the class of problems that can be solved in polynomial time. We will prove the contrapositive:

$$(JSSD \in P) \Rightarrow (JSP \in P). \tag{5.17}$$

Assume that we have access to a polynomial time subroutine JSSD ($\mathcal{N}$,$\mathcal{R}$,D_1,...,$D_{|\mathcal{R}|}$). The inputs to the subroutine are a set of NFV nodes $\mathcal{N}$, a set of services $\mathcal{R}$, and their corresponding delay constraints D_i, $i = 1, \ldots, |\mathcal{R}|$. The output of this subroutine is *true* if a feasible schedule exists such that the makespan is smaller than or equal to M (where M is a positive integer) and all the delay constraints are satisfied, and is *false* otherwise [139]. Obviously, a problem instance ($\mathcal{N}$, $\mathcal{R}$) for JSP can be transformed in polynomial time into an instance ($\mathcal{N}$,$\mathcal{R}$,D_1,...,$D_{|\mathcal{R}|}$) for JSSD (see Fig. 5.3). We can also observe that both problems need to answer whether a schedule exists such that the makespan is smaller than or equal to M. Let D_1, D_2, ...,$D_{|\mathcal{R}|}$ all be equal to M. Then, the outputs of JSP and JSSD are consistent. Suppose (towards a contradiction) that a polynomial time algorithm for JSSD exists, we could use this algorithm to solve JSP in polynomial time. Therefore, (5.17) is true, and as the contrapositive of (5.17), (5.16) is also true. This shows that the JSSD can be reduced from JSP in polynomial time and thus is NP-hard. □

5.4 A Single Agent Q-Learning Algorithm

In this section, we reformulate the VNF scheduling problem (P2) in Section 5.3 as an MDP problem with variable action set. An MDP is typically composed of five parts, denoted by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$, where $\mathcal{S}$ represents a set of system states, $\mathcal{A}$ a set of actions, $\mathcal{P}$ the state transition probabilities, R a reward function, and γ the discount factor. In what follows, we provide their specific representations in the context of VNF scheduling, based on which a single-agent RL approach is proposed to allow the agent to learn the optimal decision policy for the MDP. Fig. 5.4 shows the overall reinforcement learning framework for VNF scheduling. At each time slot t, the agent (i.e., VNF scheduler) first finds the feasible action set based on the current system state $s(t)$, then chooses an action $A(t)$ from the feasible action set according to the current scheduling policy. The reward obtained from the network environment, $R(s(t), A(t))$, is fed back to the agent for updating the scheduling policy through the RL algorithm.

5.4.1 System State

The system state is captured at the beginning of each time slot t $(\in [1, T])$, and is defined as $s(t) = [\boldsymbol{M}(t), \boldsymbol{F}(t)]$, where $\boldsymbol{M}(t) = [m_1(t), m_2(t), \ldots, m_{|\mathcal{N}|}(t)]$ denotes the states of NFV nodes and $\boldsymbol{F}(t) = [\xi_1(t), \xi_2(t), \ldots, \xi_{|\mathcal{F}|}(t)]$ represents VNF states. $\mathcal{F}$ is the set of VNFs to

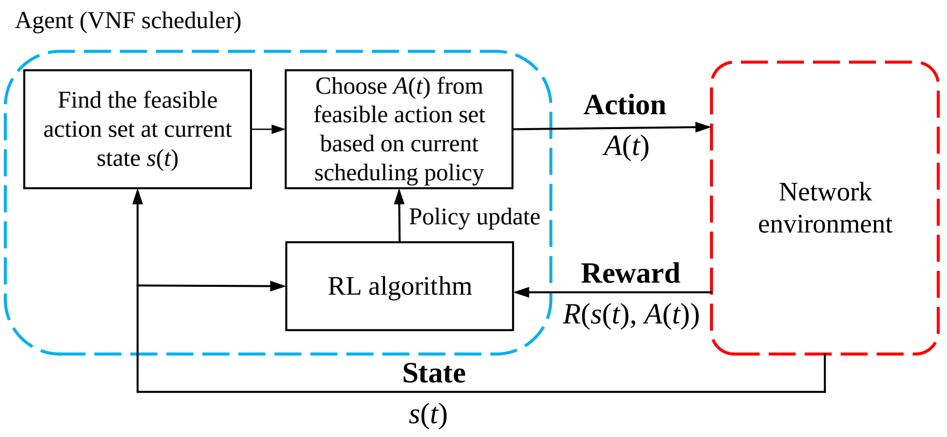

Agent (VNF scheduler)
Find the feasible action set at current state s(t)
Choose A(t) from feasible action set based on current scheduling policy
Action
A(t)
Network environment
Policy update
RL algorithm
Reward
R(s(t), A(t))
State
s(t)

Figure 5.4: Reinforcement learning framework for VNF scheduling.

be scheduled for all network services. At time slot t, the definitions of $m_k(t)$ and $\xi_l(t)$[3] with $k \in \mathbb{N}$ and $l \in \mathbb{F}$ are given by

$$m_k(t) = \begin{cases} 0, & \text{if } n_k \text{ is not processing packets,} \\ 1, & \text{if } n_k \text{ is processing packets,} \end{cases} \tag{5.18}$$

$$\xi_l(t) = \begin{cases} 0, & \text{if } f_l \text{ is waiting to be traversed,} \\ 1, & \text{if } f_l \text{ is being traversed,} \\ 2, & \text{if } f_l \text{ has been traversed.} \end{cases} \tag{5.19}$$

At the initial time slot, all the NFV nodes are idle and the VNFs in all the services are waiting to be traversed. Let the initial and completion states of one VNF scheduling process be s_{ini} and s_{ter}, respectively. According to (5.18) and (5.19), we have $s_{ini} = (0, \ldots, 0, 0, \ldots, 0)$ and $s_{ter} = (0, \ldots, 0, 2, \ldots, 2)$.

[3]We sort all VNFs to be scheduled according to the processing time of a packet batch at each VNF. There exists a one-to-one mapping between f_l and f_{ij}, where f_l represents the lth VNF in the sorted VNFs.

5.4.2 Action and Variable Action Set

In our problem, due to the dependency of VNFs, not all the actions are feasible for all the states. Therefore, it is useful to introduce an additional mapping which assigns the set of feasible/admissible actions to each state [127]. An action of the VNF scheduler (i.e., the agent in RL) at state $s(t)$, $A(t)$, indicates the VNFs chosen to be traversed at state $s(t)$ on all the NFV nodes, and is represented by $A(t) = [a_1(t), a_2(t), \ldots, a_{|\mathcal{N}|}(t)]$, where $a_k(s_t)$ is the action to take at s_t on NFV node n_k. For example, $A(t) = (1, 5, 2, 7)$ indicates that at state $s(t)$, the VNFs chosen to be traversed on n_1, n_2, n_3, and n_4 are f_1, f_5, f_2, and f_7, respectively. The feasible action set of the agent at state $s(t)$ is denoted by $\mathcal{A}(s_t) = \mathcal{A}_1(s_t) \otimes \mathcal{A}_2(s_t) \otimes \cdots \otimes \mathcal{A}_{|\mathcal{N}|}(s_t)$, where $\mathcal{A}_k(s_t)$ is the feasible action set of NFV node n_k at state $s(t)$, and $\otimes$ denotes the Cartesian product. $\mathcal{A}_k(s_t)$ contains the indices of all the VNFs that can be traversed on NFV node n_k at s_t. Note that since the packet processing time at different VNFs can be different, the length of the time period between two states that have scheduling decisions to be made is not fixed. If n_k does not have any VNF waiting to be traversed at state s_t, we set $\mathcal{A}_k(s_t) = \{0\}$ and $a_k(s_t) = 0$. We refer to them as a null feasible action set and a null action, respectively. A system state with a non-null feasible action set is called a decision-making state. For notation convenience, let $\mathcal{A}_{null}(s_t) = \{(0, 0, \ldots, 0)\}$. Since a null action is available for any state s_t, $\mathcal{A}_{null}(s_t) \subset \mathcal{A}(s_t)$ is always true.

The feasible action set $\mathcal{A}(s_t)$ depends on each state, and can be obtained without observing the environment. Therefore, in this paper, the feasible action set is finite, state-dependent, and fixed for a given state. This state-dependent feasible action set imposes additional challenge for designing the learning algorithm. Since the feasible actions are continually changing as the state changes, a specific algorithm has to be devised to find all the feasible actions for the VNF scheduler at a given system state. Then, the VNF scheduler can choose an action from the feasible action set before taking an action. Note that the feasible action set remains constant for each state, so it only needs to be found once before learning. The details for finding the feasible action set for the agent at state s_t is shown in Algorithm 6. At the beginning of the algorithm, all the idle servers are found and added to the set $\mathcal{N}_{idle}(s_t)$. For busy servers, let their feasible action set contain a null action only (Line 1 - Line 5). Then, the algorithm finds all the VNFs currently waiting for being traversed on each idle server (Line 7) and adds feasible actions (i.e., VNF indices) to the action set $\mathcal{A}_k(s_t)$ with the consideration of VNF dependency (Line 8 - Line 14). Finally, the feasible action set for the VNF scheduler is determined by the Cartesian product of the feasible action set of all the servers (Line 15).

Complexity Analysis for Algorithm 6: For a given system state s_t, the task of finding its feasible action set contains two steps. The first step is to find all the idle NFV nodes and

Algorithm 6: Algorithm for finding the feasible/admissible action set at s_t

```
   Input: Current system state $s_t$
   Output: Feasible action set $\mathcal{A}(s_t)$
1  for all $n_k \in \mathcal{N}$ do
2      if $n_k$ is idle at time slot $t$ then
3          Add $n_k$ to $\mathcal{N}_{idle}(s_t)$;
4      else
5          Set $\mathcal{A}_k(s_t) \leftarrow \{0\}$;
6  for all $n_k \in \mathcal{N}_{idle}(s_t)$ do
7      $\mathcal{F}_k(s_t) \leftarrow findWaitingFunctions(n_k, s_t)$;
8      if $\mathcal{F}_k(s_t)$ is empty then
9          Set $\mathcal{A}_k(s_t) \leftarrow \{0\}$;
10     else
11         Add $\{0\}$ to $\mathcal{A}_k(s_t)$;
12         for all $f_{ij} \in \mathcal{F}_k(s_t)$ do
13             if $j = 1$ or $\xi_{i,j-1}(t) = 2$ then
14                 Add $f_{ij}$ to $\mathcal{A}_k(s_t)$;
15 Set $\mathcal{A}(s_t) \leftarrow \mathcal{A}_1(s_t) \otimes \mathcal{A}_2(s_t) \otimes \cdots \otimes \mathcal{A}_{|\mathcal{N}|}(s_t)$.
```

add them into $\mathcal{N}_{idle}(s_t)$, which has a computational complexity of $O(|\mathcal{N}|)$. Then, for each idle NFV node $n_k \in \mathcal{N}_{idle}(s_t)$, we find all the waiting functions on it (denoted by $\mathcal{F}_k(s_t)$) and add all the feasible ones into $\mathcal{A}_k(s_t)$, whose complexity is $O(|\mathcal{N}_{idle}(s_t)| \cdot |\mathcal{F}_k(s_t)|)$. Therefore, the total computational complexity of finding feasible action set is $O(\max(|\mathcal{N}|, |\mathcal{N}_{idle}(s_t)| \cdot |\mathcal{F}_k(t)|))$, upper-bounded by $O(|\mathcal{N}| \cdot |\mathcal{F}|)$. Note that Algorithm 6 is performed only once for a given state during the whole learning process. Once the admissible/feasible action set is found for a given state, the mapping between feasible action set and state does not need to be found again for that state.

5.4.3 State Transition

At time slot $t = 1$, the system is in its initial state, i.e., $s(1) = s_{ini}$. Then, the system state is updated at the beginning of each time slot until it reaches the completion state s_{ter}. The state transitions include the transitions of server state $\boldsymbol{M}(t)$ and function state

$\boldsymbol{F}(t)$. The state transitions for $\boldsymbol{M}(t)$ are given by

$$m_k(t+1) = \begin{cases} 0, & \text{if } a_k(t) = 0, m_k(t) = 1, \theta_k(t) = 1 \\ 1, & \text{if } a_k(t) \neq 0, m_k(t) = 0, \theta_k(t) > 1 \\ m_k(t), & \text{otherwise} \end{cases} \tag{5.20}$$

where $\theta_k(t)$ denotes the remaining packet batch processing time at the VNF being traversed on server n_k at time slot t. In (5.20), the first condition indicates that the state of n_k transits from the packet batch processing state into the idle state if n_k takes a null action, and the remaining packet batch processing time at the VNF is exactly one time slot; The second condition indicates that the state of n_k transits from the idle state to the packet batch processing state if the node takes a non-null action at time slot t and the packet batch processing time at the VNF chosen to be traversed is more than one time slot. On the other hand, the state transitions for $\boldsymbol{F}(t)$ are given by

$$\xi_l(t+1) = \begin{cases} 1, & \text{if } \xi_l(t) = 0, l \in A(t), \tau_l(t) > 1 \\ 2, & \text{if } \xi_l(t) = 0, l \in A(t), \tau_l(t) = 1 \\ 2, & \text{if } \xi_l(t) = 1, \tau_l(t) = 1 \\ \xi_l(t), & \text{otherwise} \end{cases} \tag{5.21}$$

where $\tau_l(t)$ denotes the remaining packet batch processing time at f_l at time slot t. In (5.21), the first condition indicates that the state of f_l transits from waiting for being scheduled to being traversed if the agent starts the packet batch processing at time slot t and the corresponding packet batch processing time is greater than one time slot. The second condition indicates that the state of f_l transits from waiting for being scheduled to scheduling completion if the agent starts packet batch processing at time slot t and the function traversing time is exactly one time slot. The third condition indicates that ξ_l transits from being traversed to scheduling completion if the remaining packet batch processing time at f_l is one time slot at time slot t.

5.4.4 Reward Function

The objective of VNF scheduling is to minimize the overall makespan $\mathcal{M}$, while satisfying the delay requirements for different services. When an RL approach is applied to achieve delay-aware VNF scheduling, it is required that the accumulated reward (which is the feedback to the agent during the learning process) is consistent with the objective of VNF scheduling.

However, the makespan and the completion time instants of all services are not accessible until the system reaches the completion state. Therefore, instead of providing the agent with an instant reward after an action is taken at each time slot, we calculate the accumulated reward for continuous state-action pairs before an episode (i.e., a sequence of agent-environment interactions between initial and completion states) ends. In this study, we follow the reward feedback mechanism used/discussed in [128]-[130], where the reward for a state-action pair is received by the end of an episode. Although this mechanism may slow down the overall learning process, other advanced RL techniques such as reward shaping and parallel computation can be employed to speed up the overall learning process. The accumulated reward is then fed back to the agent to help improve its VNF scheduling policy. Specifically, we design an accumulated reward for a series of state-action pairs in an episode, given by

$$R(s_t, A_t) = c_0/\mathcal{M} + \sum_{i=1}^{|\mathcal{R}|} c_i \delta_i, \ \forall (s_t, A_t) \in \Omega \tag{5.22}$$

where $\mathcal{M}$ is the makespan for current episode, Ω is a set containing all the state-action pairs in the episode, and δ_i is a binary variable indicating if the delay requirement of S_i is satisfied ($\delta_i = 1$) or not ($\delta_i = 0$). c_0 and c_i are weighting coefficients reflecting the rewards obtained from minimizing the makespan and satisfying delay requirements, respectively.

If the packet batch processing of service S_i is finished before its deadline, the scheduling agent obtains an additional reward c_i. In this way, the scheduling agent tries to maximize the accumulated reward by scheduling the services to be completed before their deadlines. Setting $c_0 = 1$ and $c_i = 0$ implies that one only considers to minimize the overall makespan and ignores the delay requirements of the services, while setting $c_0 = 0$ and $c_i = 1$ means that one aims to have all the delay requirements satisfied no matter how long the makespan is. By setting proper values for c_0 and c_i, we minimize the makespan while having all the delay requirements satisfied. Note that the value of c_i also reflects the priority of services, and high priority services should have large c_i. For delay non-sensitive services, c_i should be set to a small value (or even zero), since the additional reward from satisfying the delay requirement can be obtained. For delay-sensitive services, c_i should be set as a large value to satisfy the delay requirement.

5.4.5 Q-Learning Algorithm for Optimal VNF Scheduling

Given the system states, actions, and reward functions, we develop an RL approach for the scheduling agent to learn the optimal scheduling policy π^* that maximizes the accumulated reward over time, given by

$$\pi^* = \max_{\pi} \sum_{t \in [1,T]; \mathcal{A}(s_t) \neq \mathcal{A}_{null}} R(s_t, A_t). \tag{5.23}$$

Q-learning [131] is adopted to allow the VNF scheduler to learn the optimal scheduling policy through continuously interacting with the system. During the learning process, a scheduling policy table (also called Q-table) is maintained and the entries in the Q-table (called Q-values) are updated iteratively by [131]

$$Q(s_t, A_t) = (1-\alpha)Q(s_t, A_t) + \alpha[R(s_t, A_t) + \gamma \max_{A_{t+1}} Q(s_{t+1}, A_{t+1})] \tag{5.24}$$

where α is the learning rate, γ is the discount factor, and both α and γ are real numbers set between 0 and 1. The Q-value at time slot t, $Q(s_t, A_t)$, represents the expected reward for the state-action pair (s_t, A_t), and thus can be interpreted as the probability of the agent taking action A_t at sate s_t. It is demonstrated in [133] that Q-learning can be used to achieve an optimal policy for discounted reward problems. Suppose that the Q-table converges to its optimum Q^* after sufficiently large number of episodes [134], then the optimal policy π^* can be obtained based on a greedy exploration. That is,

$$\pi^* = \arg\max_{A(t)} Q^*(s_t, A_t). \tag{5.25}$$

Since the VNF scheduling problem is transformed as an MDP with variable action set, the agent can take null actions at some time instants. Therefore, the general Q-value iterative equation in (5.24) cannot be directly applied to our problem. To make it adapt to the varying time period between two decision-making states, we modify (5.24) as

$$Q(s_t, A_t) = (1-\alpha)Q(s_t, A_t) + \alpha[R(s_t, A_t) + \gamma \max_{A_{t+\Delta t}} Q(s_{t+\Delta t}, A_{t+\Delta t})] \tag{5.26}$$

where $t+\Delta t$ denotes the next time slot when the decision-making state occurs, Δt depends on both the system state and the packet batch processing time at the VNFs being traversed at time slot t. Between any two consecutive decision-making states, Δt is same for all the NFV nodes. Note that in the implementation of large networks, the logically centralized VNF scheduler can be physically decentralized. In such a circumstance, if we apply Q-learning centrally (i.e., maintain the Q-table in a single server), then some information exchange among different servers will be required.

Algorithm 7: Q-learning algorithm to find the optimal scheduling policy

```
Initialization:
Initialize episode counter $n$ as 0;
Initialize $n_{max}, \alpha, \epsilon, \gamma$;
Set $t \leftarrow 0$;
while $n < n_{max}$ do
    Set $s_t \leftarrow s_{ini}$;
    Set $\Omega_n \leftarrow \emptyset$;
    while $s_t \neq s_{ter}$ do
        $A_t \leftarrow chooseAction(s_t, Q, t)$;
        Add $(s_t, A_t)$ to $\Omega_n$;
        $s_{t+1} \leftarrow TakeAction(s_t, A_t, t)$;
        while True do
            $t \leftarrow t + 1$;
            if $s_t = s_{ter}$ then
                break;
            $\mathcal{A}(s_t) \leftarrow findFeasibleActionSet(s_t, t)$;
            if $\mathcal{A}(s_t) \neq \mathcal{A}_{null}$ then
                break;
            else
                $s_{t+1} \leftarrow NullAction(s_t, t)$;
    $n \leftarrow n + 1$;
    for all $(s_t, A_t) \in \Omega_n$ do
        Calculate $R(s_t, A_t)$ according to (5.22);
        $Q(s_t, A_t) \leftarrow (1 - \alpha)Q(s_t, A_t) + \alpha(R(s_t, A_t) + \gamma \max_{A_{t+\Delta t}} Q(s_{t+\Delta t}, A_{t+\Delta t}))$;
```

The detailed Q-learning algorithm to determine the optimal scheduling policy is given in Algorithm 7, where n denotes the episode counter; n_{max} denotes the maximum number of episodes for Q-learning; and Ω_n represents the set containing all the state-action pairs in the nth episode. At the beginning of each episode, the system state is initialized as s_{ini} and Ω_n is empty (Line 6 - Line 7). In each episode, the agent takes an action by using the ϵ-greedy algorithm at the beginning of each time slot (Line 9 - Line 11). The detailed algorithm for the agent to take an action at each state is given in Algorithm 8. If an action is a null action, the system state keeps transiting over time until the next decision-making state occurs (Line 12 - Line 20). Whenever the agent takes a non-null

Algorithm 8: Algorithm for choosing action at each state

```
   Input: System state $s_t$, $Q$-table
   Output: The action to take $A_t$
1  $\mathcal{A}(s_t) \leftarrow$ findFeasibleActionSet($s_t$);
2  $\mathcal{N}_{idle}(s_t) \leftarrow$ findIdleServers($s_t$);
3  for all $n_k \notin \mathcal{N}_{idle}(s_t)$ do
4      Set $a_k(t) \leftarrow 0$;
5  for all $n_k \in \mathcal{N}_{idle}(s_t)$ do
6      if $\mathcal{A}_k(t) = \{0\}$ then
7          Set $a_k(t) \leftarrow 0$;
8      else
9          With probability $\epsilon$ to choose a random action $a_k(t)$ from $\mathcal{A}_k(s_t)$;
10         Otherwise, choose $a_k(t) = arg\max_{a_k(t)} Q(s_t, A_t)$;
11 Set $A_t \leftarrow [a_1(t), a_2(t), \ldots, a_{|\mathcal{N}|}(t)]$;
```

action, the state-action pair (s_t, A_t) is added to Ω_n (Line 10). When an episode ends, we calculate the accumulated reward for all the state-action pairs in this episode according to (5.22) (Line 23) and update Q-values according to (5.26) (Line 24).

Complexity Analysis: In each episode, the agent first chooses an action at each state using Algorithm 8 and then execute it. The worst-case complexity of this operation is $O(|\mathcal{N}||\mathcal{F}|)$. The state evolves until the completion state s_{ter} is reached. Let m represent the maximum number of steps in an episode. Then, the complexity of running an episode is $O(m \cdot |\mathcal{N}| \cdot |\mathcal{F}|)$. Finally, the worst-case running time of updating the Q-table for the state-action pairs in each episode is $O(m)$. Therefore, the total complexity in each episode of RL is upper-bounded by $O(|\mathcal{S}| \cdot |\mathcal{N}| \cdot |\mathcal{F}|)$. The total number of episodes required by RL is n_{max}. So the overall complexity of Algorithm 7 is upper-bounded by $O(n_{max}|\mathcal{S}| \cdot |\mathcal{N}| \cdot |\mathcal{F}|)$.

Convergence Analysis: As shown in Algorithm 6, we introduce an additional mapping which assigns the set of feasible/admissible actions to each state. Each state $s(t)$, where $s(t) = [\boldsymbol{M}(t), \boldsymbol{F}(t)]$, consists of the states of NFV nodes and the states of VNFs. The feasible action set $\mathcal{A}(s_t)$ depends on each state and can be obtained without observing the environment. Therefore, $\mathcal{A}(s_t)$ is finite, state-dependent, and fixed for a given state. The MDP under consideration has stable state-action pairs. When the state and action spaces are finite and stable, various proofs exist that the tabular Q-learning does converge to the optimal Q-function $Q^*(s_t, A_t)$, under very mild conditions [131]-[134]. In particular, a

Table 5.2: Parameter Ranges in Simulation for Different Network Scales

Parameter	Small-scale	Medium-scale	Large-scale
$\lvert\mathcal{N}\rvert$	[1,5]	[6,10]	[10,20]
$\lvert\mathcal{R}\rvert$	[1,5]	[6,15]	[16,30]
$\lvert\mathcal{F}_i\rvert$	[2,5]	[2,5]	[2,5]
ρ_{ijk}	[1,5]	[1,5]	[1,5]

Q-learning algorithm with the update rule

$$\begin{aligned}Q(s_t, A_t) = (1 - \alpha_t(s_t, A_t))Q(s_t, A_t)\\ + \alpha_t(s_t, A_t)[R(s_t, A_t) + \gamma \max_{A_{t+\Delta t}} Q(s_{t+\Delta t}, A_{t+\Delta t})],\end{aligned} \tag{5.27}$$

converges with probability one to the optimal Q-function as long as [134]

$$\sum_t \alpha_t(s_t, A_t) = \infty \text{ and } \sum_t \alpha_t^2(s_t, A_t) < \infty, \tag{5.28}$$

$\forall(s_t, A_t) \in \mathcal{S} \times \mathcal{A}$, where $\alpha_t(s_t, A_t)$ represents the learning rate at state-action pair (s_t, A_t). Since $0 \leq \alpha_t < 1$, (5.28) requires that all the state-action pairs can be visited infinitely often. For example, when $\alpha_t(s_t, A_t) = \frac{1}{t+1}$, the conditions in (5.28) can be guaranteed.

5.5 Performance Evaluation

In this section, we first verify the convergence of the proposed RL approach then compare its performance with other benchmark algorithms. In the following simulations, three different network scales (small, medium and large) are considered, which reflects different numbers of supported NFV nodes, network services, and VNFs. The parameter ranges in the simulation of the three network scales are listed in Table 5.2. For all the three network scales, the VNFs are randomly embedded onto the NFV nodes. The number of VNFs for each network service is a random integer between 2 and 5, while the packet batch processing time (in terms of number of time slots) at a VNF is a randomly selected integer from 1 to 5. Note that the size of a problem instance is sampled from parameters in Table 5.2 and remains constant across episodes during the whole learning process.

5.5.1 Convergence of the Proposed RL Approach

We first use a medium-scale network with 8 NFV nodes and 10 services to verify the convergence of the proposed RL approach. For all the simulations presented in this section, we consider all the services as delay non-sensitive ones except for S_3, S_6, and S_{10}. Accordingly, we set the coefficients in the reward function as $c_0 = 600$, $c_3 = 1.0$, $c_6 = 1.0$, $c_{10} = 1.0$, and $c_i = 0$ $(i = 1, 2, 4, 5, 7, 8, 9)$. The corresponding delay constraints are specified as $D_3 = 18$, $D_6 = 20$, and $D_{10} = 21$ (all in time units).

5.5.1.1 Impact of learning rate (α) and exploration rate (ϵ)

Fig. 5.5 illustrates the convergence of the learning process with different parameters. The discount factor γ is set to be 0.8. We consider a constant learning rate ($\alpha = 0.05, 0.2$) and a decayed one (decaying from 0.8 to 0.1 with a decay rate of 0.999) and compare their performance. From the figure, we can see that a decayed learning rate helps the RL approach converges faster than a constant one does. This is because a decayed α allows the learning algorithm to take bigger steps during the initial phase to have a fast learning, but to take smaller steps as learning approaches convergence.

On the other hand, the choice of the value for ϵ reflects the trade-off between exploration and exploitation [131]. Two cases of ϵ values are considered, i.e., $\epsilon = 0.1$ and ϵ decays from 1.0 to 0.1 with a decay rate 0.99 (referred to as "ϵ decay"). It is observed that the RL approach with a decayed ϵ has a higher convergence rate. This is because a decayed ϵ has a higher chance do exploration at the beginning but decreases the possibility dedicated for exploration as time goes by. Fig. 5.5 also shows that the proposed RL approach converges after about 1000 episodes with the optimal set of learning parameters among the four cases presented.

5.5.1.2 Impact of network scale

Next, we demonstrate the convergence of the proposed RL algorithm over all the three network scales, as shown in Fig. 5.6. The selected small-scale network is with 4 NFV nodes and 5 network services, while the selected large-scale network is with 15 NFV nodes and 20 network services. For the small-scale network, we set the coefficients in the reward function as $c_0 = 350$, $c_1 = 1.0$, $c_2 = 0$, $c_3 = 1.0$, $c_4 = 0$, and $c_5 = 1.0$. The delay constraints are specified as $D_1 = 18$, $D_3 = 17$, and $D_5 = 8$ (all in time units). For the large-scale network, we set $c_0 = 2000$, $c_9 = 1.0$, $c_{15} = 1.0$, $c_{17} = 1.0$, and $c_i = 0$ $(i = 1 \dots 20, i \neq 9, 15, 17)$. The delay constraints are specified as $D_9 = 17$, $D_{15} = 41$, and $D_{17} = 20$ (all in time units).

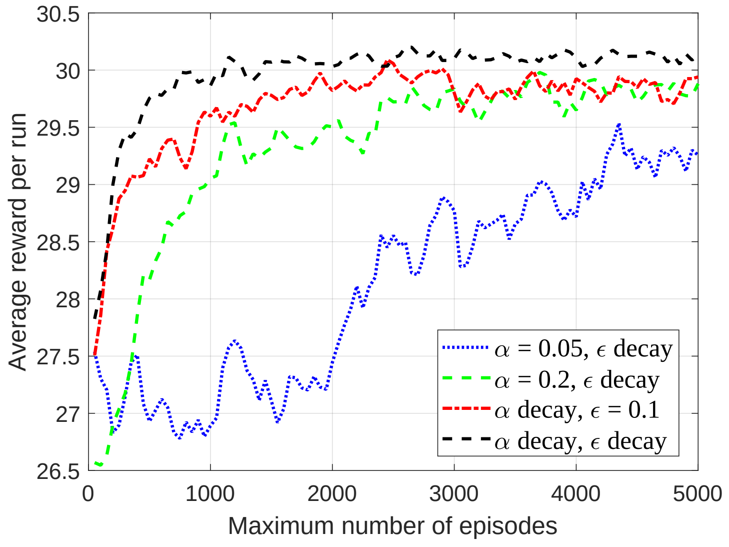
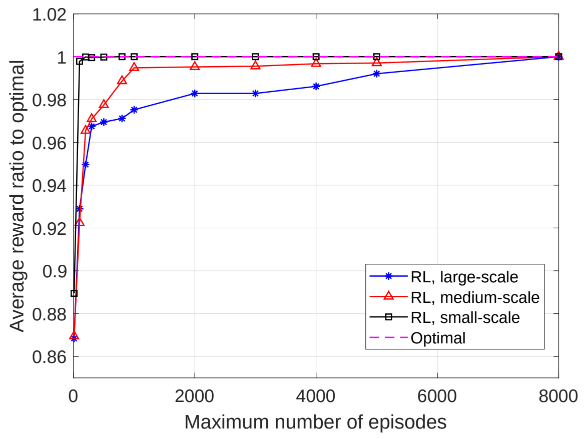
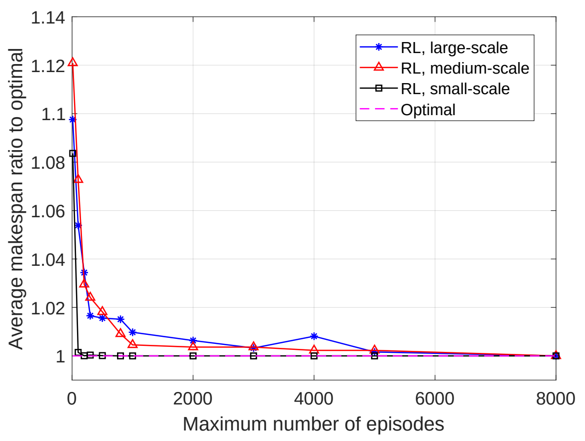


Figure 5.5: Convergence of the proposed RL approach over the medium-scale network.

For each network scale, we first tune the RL parameters to approach the optimum, and then apply to different maximum number of episodes n_{max} to observe a convergence. For each value of n_{max}, we run the RL approach 50 times and calculate the average reward and makespan per run for each network scale. These rewards and makespans are then used to find the ratios to optimality of different learning configurations. The optimal solutions are found by solving the ILP model given by $(P2)$ using Gurobi solver. Fig. 5.6(a) and Fig. 5.6(b) show the convergence of the RL algorithm in terms of average reward per run and average makespan per run, respectively. As can be seen from figures, the proposed RL approach is able to produce the optimal solution after a certain number of learning episodes. Also, the convergence rate of the RL approach decreases as the size of problem instance expands. Specifically, RL converges to the optimal solution after around 400, 1000, and 2000 episodes for the small, medium, and large problem instances, respectively. The reason for this is that a large problem instance has a bigger size of solution space, and thus the RL agent needs more episodes to explore the optimal policy.



(a)



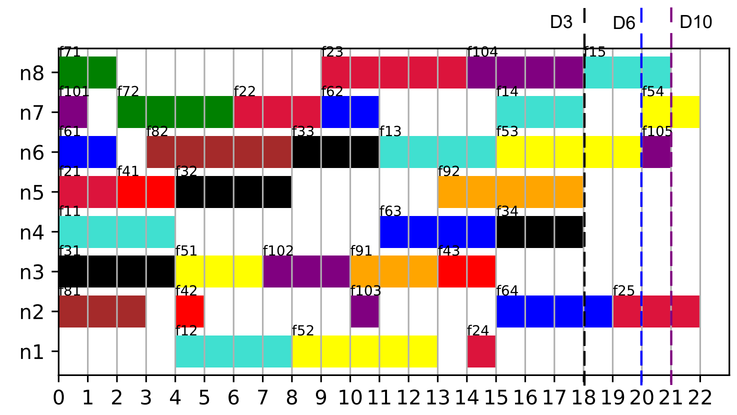
(b)

Figure 5.6: Convergence of the RL algorithm over the small-scale, medium-scale and large-scale networks: (a) Average reward ratio to optimal, and (b) Average makespan ratio to optimal.

5.5.2 Verification of Delay-Guaranteed VNF Scheduling Using the Proposed RL Approach

We focus on the medium-scale network to verify the proposed RL approach in guaranteeing the delay constraints. To this end, we consider two types of services, i.e., delay sensitive ones and delay non-sensitive ones, and two cases of service type assignments. In Case 1, S_3, S_6, and S_{10} are considered to be delay sensitive services, and all the remaining services are considered to be delay non-sensitive services. Accordingly, the coefficients in the reward function are set as $c_3 = 1.0$, $c_6 = 1.0$, $c_{10} = 1.0$, and $c_i = 0$ ($i = 1, 2, 4, 5, 7, 8, 9$). The corresponding delay constraints are specified as $D_3 = 18$, $D_6 = 20$, and $D_{10} = 21$ (all in time units). In Case 2, however, S_1, S_4, and S_7 are considered to be delay sensitive services, so set $c_1 = 1.0$, $c_4 = 1.0$, $c_7 = 1.0$, and $c_i = 0$ ($i = 2, 3, 5, 6, 8, 9, 10$). The corresponding delay constraints are specified as $D_1 = 19$, $D_4 = 18$, and $D_7 = 12$ (all in time units). For both cases, we set $c_0 = 600$, and run the RL approach for 1000 episodes ($n_{max} = 1000$) to allow a guaranteed convergence.

The detailed VNF scheduling results for the two cases are presented in Fig. 5.7, where the interval between two vertical lines represents a time slot, the bricks in the same color represent the VNFs in a network service, and the vertical dashed lines represent the delay constraints of the delay sensitive services. As can be seen from the figure, the scheduling


(a)

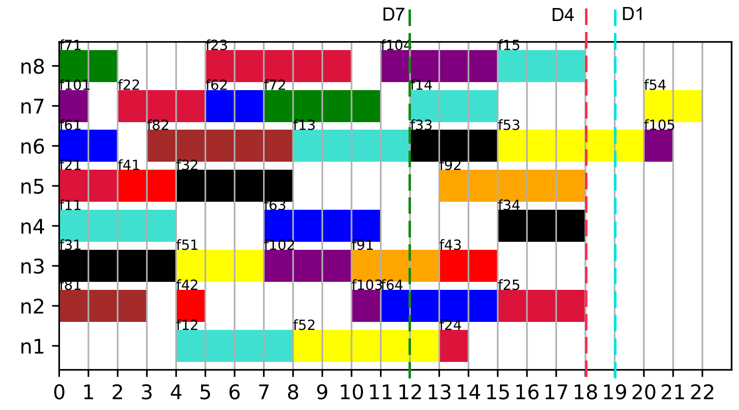

(b)

Figure 5.7: Verifying delay-guaranteed VNF scheduling using the proposed RL approach over the medium-scale network: (a) Case 1 with $c_3 = 1.0$, $c_6 = 1.0$, $c_{10} = 1.0$, and (b) Case 2 with $c_1 = 1.0$, $c_4 = 1.0$, $c_7 = 1.0$.

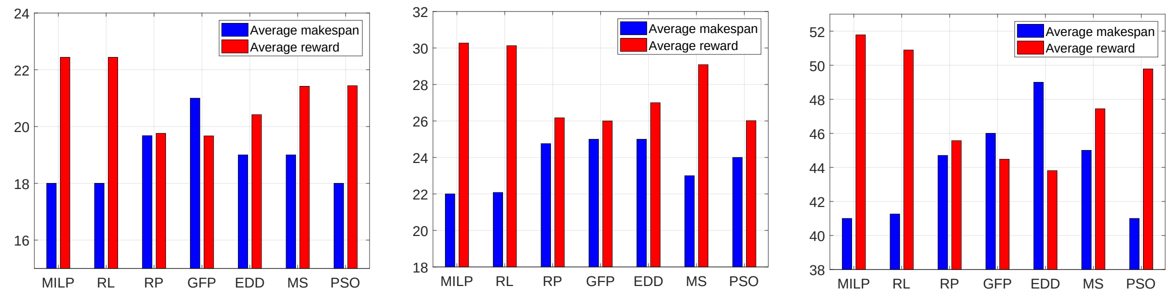
outcomes of the two cases have the same makespan but the detailed scheduling sequence of the VNFs are different. In both cases, the proposed RL approach is able guarantee the delay constraints of delay sensitive services, while achieving the optimal makespan. This demonstrates that the proposed approach achieves delay-aware VNF scheduling effectively and can support services with different priorities.

Table 5.3: Running Time Comparison Between RL Approach and MILP

Problem Instance	Proposed RL Approach		MILP Model
	Number of episodes	CPU time (Average reward)	CPU time (Reward)
Medium ($\|\mathcal{N}\|$=8, $\|\mathcal{R}\|$=10)	1	0.0039s	5.4103s (30.273)
	10	0.0486s (26.321)	
	100	0.3462s (27.924)	
	500	1.9865s (29.590)	
	800	2.6526s (29.928)	
	1000	3.0028s (30.115)	
	2000	5.9922s (30.128)	
Large ($\|\mathcal{N}\|$=15, $\|\mathcal{R}\|$=20)	1	0.0109s	41.3272s (51.780)
	10	0.1052s (44.967)	
	100	1.3739s (48.102)	
	1000	10.7934s (50.497)	
	2000	20.7799s (50.892)	
	4000	40.2382s (51.063)	
	5000	52.9593s (51.369)	
	10000	106.6593s (51.780)	

5.5.3 Running Time Comparison

We first compare the running time between the proposed RL approach and the MILP model to illustrate the time efficiency of the proposed approach. For the MILP formulation, we employ the Gurobi optimization solver to determine the optimal solution. For the RL approach, we tune its parameters to approach the optimal values. Table 5.3 compares the running times, number of episodes, and the corresponding average reward of the proposed RL approach with those of solving the MILP model. Both methods are run on a Intel i5-7200U CPU @2.5GHz. We can find that, for the medium-scale network, solving the MILP model using Gurobi solver takes about 5.4 s with a reward of 30.273 (calculated using the same reward function as RL). In comparison, the solutions from the proposed RL approach after 500 (or 1000) episodes are only 2.26% (or 0.52%) worse than the optimality, while the running times are both lower than that of the MILP model. As for the large-scale network, the solution from the RL approach with 2000 episodes already has a very small gap (1.71%) to the optimality, while the running time is almost half of that of the MILP model. It is also noticed that the advantage of the RL approach becomes greater as the problem



(a) (b) (c)

Figure 5.8: Performance comparison of the four methods in terms of the average reward and average makespan over the three scale networks: (a) small-scale network, (b) medium-scale network, and (c) large-scale network.

instance size expands. In summary, we can conclude that the proposed RL approach is time-efficient with negligible loss.

5.5.4 Performance Comparison with Heuristic Algorithms

In this section, we compare the performance of the proposed RL approach with the optimal solutions obtained from the Gurobi solver and those from classical heuristic/metaheuristic algorithms. In particular, we consider four greedy algorithms [135]: 1) Greedy fast processing (GFP) in which the VNF with the shortest processing time is always selected first; 2) Earliest due date first (EDD) in which the VNF with the earliest due time is always selected first; 3) Minimum slack first (MS) which always selects the VNF with the minimum slack (defined as $D_i - t_{cur} - \rho_{ij}$, where t_{cur} is the current system time); and 4) Random policy (RP) in which the agent randomly schedules a feasible VNF for processing at each time slot. In addition, we also consider a particle swarm optimization (PSO) based algorithm [136][137] for the purpose of comparison. According to [137], we modify the way to evaluate the fitness function for each particle by adding a penalty term. For a given particle, if one delay constraint is not satisfied, a penalty is added to the fitness function.

We compare the performance of the seven methods in terms of the (average) reward and the (average) makespan over the three scale networks. For the proposed approach, we set n_{max} to be 400 (0.343 s), 1000 (3.002 s), and 2000 (20.78 s) for the small, medium, and large problem instances, respectively, to allow the RL approach to converge. For the PSO-based algorithm, the number of particles, the number of iterations, and the corresponding running time for the three network scales are (100, 200, 0.65 s), (300, 1000, 10.11 s), (500, 1000, 32.377 s), respectively. The performance comparison for the seven methods are shown in Fig. 5.8. It is seen that for all the three problem instances, the average reward and makepsan from the RL approach stay very close to the optimal ones obtained from MILP, which indicates that in most cases the minimum makespan is found and the delay constraints are all satisfied. In contrast, using the PSO algorithm, in the small-scale network, the minimum makespan can be found with a delay constraint being violated. Note that the maximum reward can only be achieved when the PSO parameters are increased to (500, 200, 3.04 s), which indicates that the minimum makespan is found and the delay constraints are all satisfied. A similar conclusion can be drawn for the large-scale network. With 500 particles and 1000 iterations, PSO cannot obtain the maximum reward (with two delay constraints being violated). Even if we use 1000 particles and 2000 iterations, which takes 116.14 s, the minimum makespan still cannot be found with all the delay constraints being satisfied. This demonstrates that the proposed RL approach can achieve the minimum makespan while satisfying all the delay constraints using a shorter time than the PSO-based algorithm.

As for the other four heuristic algorithms, we observe that their solution qualities depend on the objective of the problem, and for different problem instances their performance cannot be guaranteed. Among the four heuristic algorithms, the MS algorithm performs the best in all the three network scales. The reason is that the MS algorithm aims to minimize the maximum lateness and takes the delay constraints into consideration when selecting the VNF for processing. However, performance gap exists between the solution from the MS algorithm and the optimal solution, and this gap increases as network scale becomes larger. Considering that the optimality gap between the RL solutions and the optimal ones are consistently small, we can conclude that the proposed RL approach outperforms classical heuristic algorithms, and also well competes with the MILP model over all of the three networks. In addition, the performance gaps between the RL approach and the heuristic algorithms in terms of the reward are bigger than those for the makespan for all the three network scales. This is because the RL approach achieves delay-aware VNF scheduling where not only the makespan is minimized but also the delay constraints are satisfied. Therefore, the additional rewards for satisfying the delay constraints are be obtained. Note that the makespan differences in Fig. 5.8 indicate the packet batch processing time gaps between RL and the heuristic algorithms in one scheduling cycle. Once the scheduling sequence is found, each cycle of VNF scheduling proceeds repeatedly for future packet batch processing, which accumulates the gap and leads to a considerable performance difference between the RL approach and the other five algorithms in comparison.

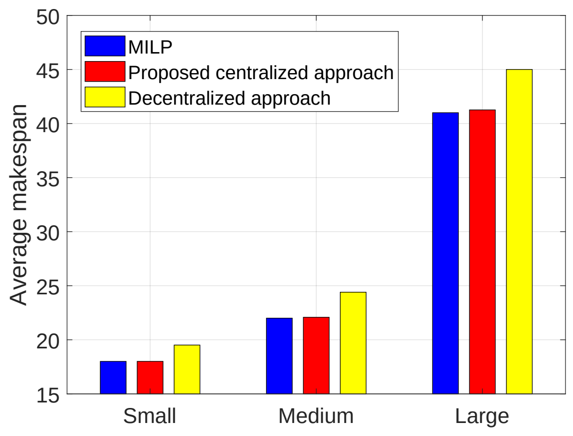


Figure 5.9: Comparison between the proposed centralized approach and the decentralized approach in terms of makespan.

5.5.5 Performance Comparison with Decentralized Solution

In this section, we compare the proposed centralized approach and the decentralized one [69]. For the decentralized approach, we attach to each NFV server an agent which improves its VNF scheduling policy independently with the help of RL. Each agent only has a local view of of the system, which contains the status of the NFV node it is associated with and the status of all the VNFs on that node. Similar to [69], the immediate cost $C(s_t, a_t)$ for taking action a_t at state s_t is given by

$$C(s_t, a_t) := \sum_{k=1}^{|\mathcal{N}|} |\{f_{ij} | f_{ij} \text{ is queued at NFV node } n_k\}|. \tag{5.29}$$

Similar to the reward function $R(s_t, A_t)$ in this chapter, this cost function $C(s_t, a_t)$ is utilized in updating the Q-table maintained within each learning agent. The intuition behind this cost function is that a high utilization of the resources (i.e., NFV nodes) implies a minimal makespan.

Since delay constraints are not considered in the decentralized approach, we set $c_i = 0$ $(i = 1, \ldots, |\mathcal{R}|)$ in the reward function in our approach and compare the performance in terms of makespan only. Fig. 5.9 shows the comparison between the two approaches over three scale networks. It can be seen that the proposed centralized approach outperforms the decentralized one. Our centralized approach has improved performance by taking advantage of a global view of the status of NFV nodes and services in the whole system.

5.6 Summary

In this chapter, we have investigated a VNF scheduling problem under an SDN/NFV-enabled network architecture, with the objective of minimizing the overall makespan of services while satisfying differentiated E2E delay requirements. The problem was formulated as an MILP and then reformulated as an MDP problem. A reinforcement learning approach has been proposed to obtain near-optimal solutions to the VNF scheduling problem with high efficiency and accuracy. Simulation results have been presented to demonstrate that the proposed approach outperforms other heuristic algorithms and can achieve near-optimal solutions. The proposed approach facilitates QoS-guaranteed service provisioning in SDN/NFV-enabled networks. Since the Q-table was learnt and applied on one fixed problem instance, as our future work, we will investigate a more general RL approach which can learn on a set of training problem instances but apply to unknown instances with similar patterns. We will also consider stochastic arrivals of service requests in the formulation of the VNF scheduling problem.

Chapter 6

Conclusions and Future Research

In this chapter, we summarize the main contributions and impacts of the presented works and highlight future research directions.

6.1 Main Research Contributions

In this thesis, we make some in-depth researches on efficient resource allocation for SDN/NFV-enabled core networks in multiple aspects and dimensionality.

- First, we jointly study how to design the topology of a VN and embed the resultant VN onto a substrate network with the objective of minimizing the embedding cost while satisfying the QoS requirements. In VN topology design, optimizing the resource requirement for each virtual node and link is necessary. Without topology optimization, the resources assigned to the virtual network may be insufficient or redundant, leading to degraded service quality or increased embedding cost. The joint problem is formulated as a Mixed Integer Nonlinear Programming (MINLP), where queueing theory is utilized as the methodology to analyze the network delay and help to define the optimal set of physical resource requirements at network elements.
- Second, we propose a game theoretical approach to address the multi-VNF chain embedding problem with the impact of processing-resource sharing and node capacity constraints being considered. In the proposed game, each VNF chain is treated as one player whose objective is to minimize the overall latency experienced by its flow, while satisfying the capacity constraints of all the NFV nodes. The additional

delay introduced by processing-resource sharing is modeled and used in analyzing the overall latency for each VNF chain, while the node capacity constraints are handled by adding a penalty term to the cost function of each player. The proposed game is proved to be an exact potential game which admits at least one pure Nash Equilibrium (NE) and has the finite improvement property (FIP) property. We then design two iterative algorithms, i.e., the best response (BR) algorithm and the spatial adaptive play (SAP) algorithm, to find the NE of the game, which corresponds to a locally/globally optimal solution to the original problem.

- Third, the VNF scheduling problem is investigated to minimize the makespan (i.e., overall completion time) of all services, while satisfying their different end-to-end (E2E) delay requirements. The problem is formulated as a mixed integer linear program (MILP) which is NP-hard with exponentially increasing computational complexity as the network size expands. To solve the MILP with high efficiency and accuracy, the original problem is reformulated as a Markov decision process (MDP) problem with variable action set. Then, a reinforcement learning (RL) approach is developed to learn the best scheduling policy by continuously interacting with the network environment. The proposed learning algorithm determines the variable action set at each decision-making state and accommodates different execution time of the actions. The reward function in the proposed algorithm is carefully designed to realize delay-aware VNF scheduling.

6.2 Future Research

Some possible future research directions based on the extension of the approaches proposed in this thesis are identified as follows:

- **Dynamic VNF scheduling:** One of the future research directions is to consider the dynamic VNF scheduling. Most of previous studies consider static scenarios, where the possibility of VNF remapping and rescheduling is not contemplated. However, to adapt to varying network conditions and service characteristics, it is also critical to consider dynamic scenarios where VNF live migration, VNF re-instantiation, and VNF rescheduling are allowed. The attempt of remapping and rescheduling can increase the possibility of admitting new services and thus increases the service provider's profit through a quick readjustment of the mapping and scheduling strategies of existing VNFs. However, the operational cost and additional delay incurred by VNF live migration and re-instantiation should be carefully modeled.

- **Cost-efficient resource allocation/slicing:** To fully unleash the potential of the SDN/NFV-enabled core network framework, it is imperative to design a multiple time-granularity resource slicing scheme to balance the network performance and the slicing cost. In particular, both the communication frequency between the SDN controller and NFV nodes, and the information exchange frequency among NFV nodes have significant impact on the performance of the slicing framework. Moreover, the amount of information required to be exchanged among different network entities for updating the slicing results also affect the slicing accuracy. Higher interaction frequency between different network elements leads to better adaptiveness, but inevitably increases the slicing cost. Therefore, it is required to develop a multi-time-granularity slicing solution to balance the trade-off between slicing cost and slicing accuracy.

- **Prediction-based proactive VNF chain embedding:** Nowadays, with the development of 5G networks, the services show high diversity not only in QoS requirements but also in traffic patterns. The pre-allocated resource for different services are highly related to their traffic patterns and service types. In our proposed schemes, we always assume that the service arriving follows Poisson process as most existing works, given the assumption that the traffic patterns are well characterized by the models. However, the uncertainty of traffic features and service types becomes more dominant in a increasingly complex 5G and beyond 5G environment, which decreases the accuracy of model based methods. For future works, a learning based traffic prediction can be designed to determine the resource allocation. Leveraging the emerging deep learning technologies, such as long short term memory neural network (LSTM) and deep Q-network (DQN), dynamic traffic and service types, heterogeneous QoS requirements are trained and predicted accurately with low complexity, upon which proactive resource allocation actions can be learned in an online manner.

References

[1] A. Osseiran, F. Boccardi, V. Braun, *et al.*, "Scenarios for 5G mobile and wireless communications: the vision of the METIS project," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 26-35, 2014.

[2] Q. Ye, and W. Zhuang, "Distributed and adaptive medium access control for Internet-of-Things enabled mobile networks," *IEEE Internet Things J.* vol. 4, no. 2, pp. 446-460, 2017.

[3] 5G Infrastructure Association, "5G vision: the 5G infrastructure public private partnership: the next generation of communication networks and services," *White Paper*, Feb. 2015.

[4] W. Zhuang, Q. Ye, F. Lyu, N. Cheng, and J. Ren, "SDN/NFV-empowered future IoV with enhanced communication, computing, and caching," in *Proc. IEEE*, vol. 108, no. 2, pp. 274-291, Feb. 2020.

[5] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5G: RAN, core network and caching solutions," *IEEE Commun. Surv. Tutor.*, vol. 20, no. 4, pp. 3098-3130, May 2018.

[6] 3GPP TR 38.913 V14.2.0, "5G; Study on scenario and requirements for next generation access technologies," 2018.

[7] 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, Third Quarter 2018.

[8] M. Bennis, M. Debbah, and H. V. Poor, "Ultra-reliable and low-latency wireless communication: Tail, risk, and scale," *Proceedings of the IEEE*, vol. 106, no. 10, pp. 1834-1853, Oct. 2018.

[9] Q. Ye, J. Li, K. Qu, W. Zhuang, X. Shen, and X. Li, "End-to-end quality of service in 5G networks - Examining the effectiveness of a network slicing framework," *IEEE Veh. Technol. Mag.*, vol. 13, no. 2, pp. 65-74, 2018.

[10] N. Zhang, P. Yang, S. Zhang, D. Chen, W. Zhuang, B. Liang, and X. Shen, "Software defined networking enabled wireless network virtualization: Challenges and solutions," *IEEE Netw.*, vol. 31, no. 5, pp. 42-49, 2017.

[11] S. Zhang, W. Quan, J. Li, W. Shi, P. Yang, and X. Shen, "Air-ground integrated vehicular network slicing with content pushing and caching," *IEEE J. Sel. Areas of Commun.*, vol. 36, no. 9, pp. 2114-2127, Oct. 2018.

[12] I. Da Silva, S. E. El Ayoubi, O. M. Boldi, *et al.*, "5G RAN architecture and functional design," METIS II White Paper.

[13] J. Chen *et al.*, "SDATP: An SDN-based traffic-adaptive and service-oriented transmission protocol," *IEEE Trans. Cogn. Commun. Netw.*, pp. 1-15, Dec. 2019.

[14] Q. Ye, W. Zhuang, S. Zhang, A. Jin, X. Shen, and X. Li, "Dynamic radio resource slicing for a two-tier heterogeneous wireless network", *IEEE Trans. Veh. Technol.*, vol. 67, no. 10, pp. 9896-9910, 2018.

[15] K. Tutschku, T. Zinner, A. Nakao, and P. Tran-Gia, "Network virtualization: implementation steps towards the future internet," *Electronic Communications of the EASST*, 2009.

[16] N. M. M. K. Chowdhury, and R. Boutaba, "Network virtualization: State of the art and research challenges," *IEEE Commun. Mag.*, vol. 47, no. 7, pp. 20-26, Jul. 2009.

[17] C. Liang and F. Richard Yu. "Wireless network virtualization: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 358-380, 2015.

[18] A. Fischer, J. Botero, M. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888-1906, 2013.

[19] D. Eppstein, "Finding the k shortest paths," *SIAM J. Comput.*, vol. 28, pp. 652-673, Feb. 1999.

[20] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," in *Proc. ACM SIGCOMM CCR*, vol. 38, no. 2, pp. 17-29, Apr. 2008.

[21] M. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. IEEE INFOCOM'09*, Rio de Janeiro, Brazil, Apr. 2009, pp. 783-791.

[22] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw*, vol. 20, no. 1, pp. 206-219, Feb. 2012.

[23] Z. Cai, F. Liu, N. Xiao, Q. Liu, and Z. Wang, "Virtual network embedding for evolving networks," in *Proc. IEEE GlOBECOM'10*, Miami, FL, USA, Dec. 2010, pp. 1-5.

[24] G. Chochlidakis and V. Friderikos, "Low latency virtual network embedding for mobile networks," in *Proc. IEEE ICC'16*, Kuala Lumpur, Malaysia, 2016, pp. 1-6.

[25] A. J. Gonzalez, G. Nencioni, A. Kamisinski, B. E. Helvik, and P. E. Heegaard, "Dependability of the NFV orchestrator: State of the art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3307-3329, 2018.

[26] M. Chiosi, C. Don, W. Peter, R. Andy *et al.*, "White paper: Network functions virtualisation: An introduction, benefits, enablers, challenges & call for action," *Tech. Rep.*, 2012.

[27] ETSI, "Network Functions Virtualisation (NFV); Use cases (v 0.0.5 2016-09)," *Tech. Rep.*, 2016.

[28] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surv. Tutor.*, vol. 18, no. 1, pp. 236-262, 2016.

[29] G. ETSI, "Network Functions Virtualisation (NFV): Architectural framework," *ETSI GSNFV*, vol. 2, no. 2, p. V1, 2013.

[30] P. Quinn and T. Nadeau, "Service function chaining problem statement," *draft-ietf-sfc-problem-statement-07 (work in progress)*, 2014.

[31] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE ACCESS*, vol. 3, pp. 2542-2553, 2015.

[32] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 518-532, 2016.

[33] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on network virtualization hypervisors for software defined networking," *IEEE Commun. Surv. Tutor.*, vol. 18, no. 1, pp. 655-685, 2016.

[34] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," *Computer Networks*, vol. 72, pp. 74-98, 2014.

[35] https://www.cisco.com/c/en/us/solutions/software-defined-networking/benefit.html.

[36] T.Wood, K. K. Ramakrishnan, J. Hwang, G. Liu, and W. Zhang, "Toward a software-based network: Integrating software defined networking and network function virtualization," *IEEE Netw.*, vol. 29, no. 3, pp. 36-41, May 2015.

[37] J. Li, W. Shi, Q. Ye, W. Zhuang, X. Shen, and X. Li, "Online joint VNF chain composition and embedding for 5G networks," in *Proc. IEEE GLOBECOM'18*, Abu Dhabi, UAE, Dec. 2018, pp. 1-6.

[38] J. Li, N. Zhang, Q. Ye, W. Shi, W. Zhuang, and X. Shen, "Joint resource allocation and online virtual network embedding for 5G networks," in *Proc. IEEE GLOBECOM'17*, Singapore, Dec. 2017, pp. 1-6.

[39] Y. Xie, Z. Liu, S. Wang, and Y. Wang, "Service function chaining resource allocation: A survey," *Arxiv preprint (arXiv:1608.00095)*, Jul 2016.

[40] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. IEEE Cloud Networking (CloudNet)*, 2014, pp. 7-13.

[41] M. T. Beck and J. F. Botero, "Coordinated allocation of service function chains," in *Proc. IEEE GLOBECOM'15*, Dec 2015, pp. 1-6.

[42] 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 *Proc. IFIP/IEEE IM*, May 2015, pp. 98-106.

[43] 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, Nov. 2016.

[44] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *Proc. Int. Conf. Network and Service Management (CNSM)*, Nov 2015, pp. 50-56.

[45] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. IEEE INFOCOM'15*, 2015, pp. 1346-1354.

[46] A. Baumgartner, V. S. Reddy, and T. Bauschert, "Mobile core network virtualization: A model for combined virtual core network function placement and topology optimization," in *Proc. IEEE Network Softwarization (NetSoft)*, London, UK, 2015, pp. 1-9.

[47] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of virtualized deep packet inspection functions in SDN," in *Proc. IEEE Military Communications (MILCOM)*, 2013, pp. 992-997.

[48] E. Amaldi, S. Coniglio, A. M. Koster, and M. Tieves, "On the computational complexity of the virtual network embedding problem," *Electronic Notes in Discrete Mathematics*, vol. 52, pp. 213-220, 2016.

[49] B. Martini, F. Paganelli, P. Cappanera, S. Turchi, and P. Castoldi, "Latency-aware composition of virtual functions in 5G," in *Proc. 1st IEEE Conf. Netw. Softwarization (NetSoft)*, pp. 1-6, Apr. 2015.

[50] Z. Ye, X. Cao, J. Wang, H. Yu, and C. Qiao, "Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization," *IEEE Netw.*, vol. 30, no. 3, pp. 81-87, May 2016.

[51] S. D'Oro, L. Galluccio, S. Palazzo, and G. Schembra, "Exploiting congestion games to achieve distributed service chaining in NFV networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 2, pp. 407-420, Feb. 2017.

[52] S. D'Oro, L. Galluccio, S. Palazzo, and G. Schembra, "A game theoretic approach for distributed resource allocation and orchestration of softwarized networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 3, pp. 721-735, Mar. 2017.

[53] S. Bian, X. Huang, Z. Shao, X. Gao, Y. Yang, "Service chain composition with failures in NFV systems: A game-theoretic perspective," in *Proc. IEEE ICC*, May 2019, pp. 1-6.

[54] S. Le, Y. Wu, and X. Sun"Congestion games with player-specific utility functions and its application to NFV networks," *IEEE Trans. Autom. Sci. Eng.*, Mar. 2019.

[55] X. Chen, Z. Zhu, R. Proietti, and S. J. B. Yoo, "On incentive-driven VNF service chaining in inter-datacenter elastic optical networks: A hierarchical game-theoretic mechanism," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 1, pp. 1-12, Mar. 2019.

[56] A. Leivadeas, G. Kesidis, M. Falkner, and I. Lambadaris, “A graph partitioning game theoretical approach for the VNF service chaining problem,” *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 4, pp. 890-903, Dec. 2017.

[57] X. Chen, Z. Zhu, J. Guo, S. Kang, R. Proietti, A. Castro, and S. J. B. Yoo, “Leveraging mixed-strategy gaming to realize incentive-driven VNF service chain provisioning in broker-based elastic optical inter-datacenter networks,” *IEEE Trans. Netw. Service Manag.*, vol. 10, no. 2, pp. 232-240, Feb. 2018.

[58] P. T. A. Quang, A. Bradai, K. D. Singh, G. Picard, and R. Riggio,“Single and multi-domain adaptive allocation algorithms for VNF forwarding graph embedding,” *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 1, pp. 98–112, Mar. 2019.

[59] L. Guo, J. Pang, and A. Walid, “Dynamic service function chaining in SDN-enabled networks with middleboxes,” in *Proc. IEEE ICNP*, Nov 2016, pp. 1-10.

[60] A. Baumgartner, V. S. Reddy, and T. Bauschert, “Combined virtual mobile core network function placement and topology optimization with latency bounds,” in *European Workshop on Software Defined Networks*, Sep 2015, pp. 97-102.

[61] M. Savi, M. Tornatore, and G. Verticale, “Impact of processing-resource sharing on the placement of chained virtual network functions,” *IEEE Trans. Cloud Comput.*, 2019.

[62] L. Qu, C. Assi, and K. Shaban, “Delay-aware scheduling and resource optimization with network function virtualization,” *IEEE Trans. Commun.*, vol. 64, no. 9, pp. 3746-3758, 2016.

[63] 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 *Proc. IEEE Netw. Softwarization (NetSoft)*, Apr. 2015, pp. 1-9.

[64] J. F. Riera, E. Escalona, J. Batallé, E. Grasa, and J. A. García-Espin, “Virtual network function scheduling: Concept and challenges,” in *Proc. IEEE Smart Commun. Netw. Technol. (SaCoNeT)*, Spain, Jun. 2014, pp. 1-5.

[65] H. A. Alameddine, L. Qu, and C. Assi, “Scheduling service function chains for ultra-low latency network services,” in *Proc. IEEE Netw. Service Manage. (CNSM)*, 2017, pp. 1-9.

[66] C. Pham, N. H. Tran, and C. S. Hong, “Virtual network function scheduling: A matching game approach,” *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 69-72, Jan, 2018.

[67] I. Bello, H. Pham, QV. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.

[68] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Proc. Advances Neural Information Processing Systems*, pp. 2692–2700, 2015.

[69] T. Gabel and M. Riedmiller, "Adaptive reactive job-shop scheduling with reinforcement learning agents," *Int. J. Information Technology and Intelligent Computing*, vol. 24, no. 4, 2008.

[70] T. Gabel and M. Riedmiller, "Distributed policy search reinforcement learning for job-shop scheduling tasks," *Int. J. Production Research*, vol. 50, no. 1, pp. 41-61, 2012.

[71] M. Lauer and M. Riedmiller, "An algorithm for distributed reinforcement learning in cooperative multi-agent systems," in *Proc. ICML'00*, 2000, pp. 535-542.

[72] M. M. Ling, K. A. Yau, J. Qadir, and Q. Ni, "A reinforcement learning-based trust model for cluster size adjustment scheme in distributed cognitive radio networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 1, pp. 28-43, 2019.

[73] J. Li, W. Shi, N. Zhang, and X. Shen, "Reinforcement learning based VNF scheduling with end-to-end delay guarantee," in *Proc. IEEE/CIC ICCC'19*, Aug. 2019, pp. 572-577.

[74] J. Li, W. Shi, P. Yang, and X. Shen, "On Dynamic mapping and scheduling of service function chains in SDN/NFV-enabled networks," *IEEE GLOBECOM'19*, Dec. 2019, pp. 1-6.

[75] L. Guo, J. Pang, and A. Walid, "Dynamic service function chaining in SDN-enabled networks with middleboxes," in *Proc. IEEE ICNP*, 2016, pp. 1-10.

[76] H. Hawilo, M. Jammal, and A. Shami, "Orchestrating network function virtualization platform: Migration or re-instantiation?" in *Proc. Cloud Networking (CloudNet)*, 2017, pp. 1-6.

[77] D. Cho, J. Taheri, AY. Zomaya, and P. Bouvry, "Real-time virtual network function (vnf) migration toward low network latency in cloud environments," in *Proc. IEEE Cloud Computing (CLOUD)*, 2017, PP. 798-801.

[78] J. Zhang, L. Li, and D. Wang, "Optimizing VNF live migration via para-virtualization driver and QuickAssist technology," in *Proc. IEEE ICC*, 2017, pp. 1-6.

[79] V. Eramo, E. Miucci, M. Ammar, and FG. Lavacca, "An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures," *IEEE/ACM Trans. Netw.*, vol. 25, no. 4, pp. 2008-2025, 2017.

[80] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu, "On dynamic service function chain deployment and readjustment," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 543-553, 2017.

[81] B. Németh, M. Szalay, J. Dóka, M. Rost, S. Schmid, L. Toka, and B. Sonkoly, "Fast and efficient network service embedding method with adaptive offloading to the edge," in *Proc. IEEE INFOCOM WKSHPS'18*, 2018, pp. 178-183.

[82] M. Nejad, S. Parsaeefard, M. Maddah-Ali, T. Mahmoodi, and B. Khalaj, "vSPACE: VNF simultaneous placement, admission control and embedding," *IEEE J. Sel. Areas Commun.* vol. 36, no. 3, pp. 542-557, 2018.

[83] R. Wen *et al.*, "On robustness of network slicing for next-generation mobile networks," *IEEE Trans. Commun.*, vol. 67, no. 1, pp. 430-444, 2018.

[84] X. Cheng, *et al.*, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 38-47, Apr. 2011.

[85] S. Zhang, Z. Qian, J. Wu, S. Lu, and L. Epstein, "Virtual network embedding with opportunistic resource sharing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 816-827, Mar. 2014.

[86] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. IEEE INFOCOM*, May 2014, pp. 1-9.

[87] L. Gong, H. Jiang, Y. Wang, and Z. Zhu, "Novel location-constrained virtual network embedding LC-VNE algorithms towards integrated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 24, no. 6, Dec. 2016.

[88] S. R. Chowdhury *et al.*, "Multi-layer virtual network embedding," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 3, pp. 1132-1145, Sept. 2018.

[89] A. Song, W. Chen, T. Gu, H. Zhang, and J. Zhang, "A constructive particle swarm optimizer for virtual network embedding," *IEEE Trans. Netw. Sci. Eng.*, Aug. 2019.

[90] M. Zhang, C. Wu, M. Jiang, and Y. Qiang, "Mapping multicast service-oriented virtual networks with delay and delay variation constraints," in *Proc. IEEE GLOBECOM'10*, Miami, FL, USA, Dec. 2010, pp. 1-6.

[91] M. M. A. Khan, N. Shahriar, R. Ahmed, and R. Boutaba, "Multi-path link embedding for survivability in virtual networks," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 2, pp. 253-266, Jun. 2016.

[92] S. Ayoubi, C. Assi, K. Shaban, and L. Narayanan, "Minted: Multicast virtual network embedding in cloud data centers with delay constraints," *IEEE Trans. Commun.*, vol. 63, no. 4, pp. 1291-1305, Apr. 2015.

[93] W. Wang, B. Liang, and B. Li, "Multi-resource generalized processor sharing for packet processing," in *Proc. ACM IWQoS*, Montreal, QC, Canada, Jun. 2013, pp. 1-10.

[94] Q. Ye, W. Zhuang, X. Li, and J. Rao, "End-to-end delay modeling for embedded VNF chains in 5G core networks," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 692-704, Feb. 2019.

[95] 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./Dec. 2016.

[96] K. Qu, W. Zhuang, Q. Ye, X. Shen, X. Li, and J. Rao, "Dynamic flow migration for embedded services in SDN/NFV-enabled 5G core networks," *IEEE Trans. Commun.*, pp. 1-15, 2020.

[97] D. Bertsekas and R. Gallager, *Data Networks.* Englewood Cliffs, NJ: Prentice-Hall, 1987.

[98] O. Alhussein *et al*, "A virtual network customization framework for multicast services in NFV-enabled core Networks," *IEEE J. Sel. Areas of Commun.*, Feb. 2020.

[99] S. Burer, and A. N. Letchford, "Non-convex mixed-integer nonlinear programming: A survey," *Surv. Oper. Res. Management Sci.*, vol. 17, no. 2, pp. 97-106, 2012.

[100] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE INFOCOM'96*, San Francisco, CA, USA, 1996, pp. 594-602.

[101] A. Blenk, P. Kalmbach, P. van der Smagt, and W. Kellerer, "Boost online virtual network embedding: Using neural networks for admission control," in *Proc. IEEE CNSM'16*, Oct. 2016, pp. 1-9.

[102] S. Liao, C. Wu, M. Zhang, and M, Jiang, "An efficient virtual network embedding algorithm with delay constraints," in *Proc. IEEE WPMC'13*, Atlantic City, NJ, USA, Jun. 2013, pp. 1-6.

[103] N. Shahriar *et al.*, "Virtual network survivability through joint spare capacity allocation and embedding," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 502-518, Mar. 2018.

[104] C. W. Huang, C. A. Shen, C. Y. Huang, T. L. Chin, and S. H. Shen, "An efficient joint node and link mapping approach based on genetic algorithm for network virtualization," in *Proc. IEEE Veh. Technol. Conf. (VTC Fall)*, Honolulu, HI, USA, Sept. 2019, pp. 1-5.

[105] O. Alhussein, T. Do, J. Li, Q. Ye, W. Shi, W. Zhuang, X. Shen, X. Li, and J. Rao, "Joint VNF placement and multicast traffic routing in 5G core networks," in *Proc. IEEE GLOBECOM'18*, Abu Dhabi, UAE, Dec. 2018, pp. 1-6.

[106] X. Fu, F. R. Yu, J. Wang, Q. Qi, and J. Liao, "Dynamic service function chain embedding for NFV-enabled IoT: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, Oct. 2019.

[107] S. Agarwal, F. Malandrino, C. F. Chiasserini, and S. De, "Joint VNF placement and CPU allocation in 5G," in *Proc. IEEE INFOCOM*, 2018, pp. 1-9.

[108] D. Monderer and L. S. Shapley, "Potential games," *Games Econ. Behavior*, vol. 14, no. 1, pp. 124–143, 1996.

[109] N. Zhang, S. Zhang, J. Zheng, X. Fang, Jon W. Mark, and X. Shen, "QoE Driven Decentralized Spectrum Sharing in 5G Networks: Potential Game Approach", *IEEE Trans. Veh. Technol.*, Vol. 66, No. 9, pp. 7797-7808, 2017.

[110] N. Cheng, N. Zhang, N. Lu, X.Shen, Jon W. Mark, and F. Liu, "Opportunistic Spectrum Access for CR-VANETs: A Game Theoretic Approach," *IEEE Trans. Veh. Technol.*, Vol. 63, No. 1, pp. 237-251, 2013.

[111] L. M. Law, J. Huang, and M. Liu, "Price of anarchy of wireless congestion games," *IEEE Trans. Wireless Commun.*, vol. 11, no. 10, pp. 3778-3787, Oct. 2012.

[112] Y. Pan and L. Pavel, "Global convergence of an iterative gradient algorithm for the nash equilibrium in an extended OSNR game," in *Proc. IEEE INFOCOM*, 2007, pp. 1-7.

[113] G. Arslan, M. F. Demirkol, and S. Yuksel, "On games with coupled constraints," *IEEE Trans. Autom. Control*, vol. 60, no. 2, pp. 358-372, Aug. 2014.

[114] W. Yuan, P. Wang, W. Liu, and W. Cheng, "Variable-width channel allocation for access points: A game-theoretic perspective," *IEEE Trans. Mobile Comput.*, vol. 12, no. 7, pp. 1428-1442, Jul. 2013.

[115] J. Zheng, Y. Cai, Y. Liu, Y. Xu, B. Duan, and X. Shen, "Optimal power allocation and user scheduling in multicell networks: Base station cooperation using a game-theoretic approach," *IEEE Transactions on Wireless Communications*, vol. 13, no. 12, pp. 6928-6942, 2014.

[116] H. P. Young, *Individual Strategy and Social Structure: An Evolutionary theory of Institutions*. Princeton University Press, 1998.

[117] Y. Xu, J. Wang, Q. Wu, A. Anpalagan, and Y. Yao, "Opportunistic spectrum access in cognitive radio networks: Global optimization using local interaction games," *IEEE J. Sel. Topics Signal Process.*, vol. 6, no. 2, pp. 180-194, Apr. 2012.

[118] J. Marden, G. Arslan, and J. Shamma, "Cooperative control and potential games," *IEEE Trans. Syst., Man, Cybern.*, vol. 39, no. 6, pp. 1393-1407, Dec. 2009.

[119] G. Baggio, R. Bassoli, and F. Granelli, "Cognitive software-defined networking using fuzzy cognitive maps," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 3, pp. 517-539, 2019.

[120] Z. Meng, J. Bi, H. Wang, C. Sun, and H. Hu, "MicroNF: An efficient framework for enabling modularized service chains in NFV," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1851-1865, 2019.

[121] J. F. Riera, X. Hesselbach, E. Escalona, J. A. García-Espin, and E. Grasa, "On the complex scheduling formulation of virtual network functions over optical networks," in *Proc. IEEE Transp. Opt. Netw. (ICTON)*, Graz, Austria, Jul. 2014, pp. 1-5.

[122] M. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Math. Oper. Res.*, vol. 1, pp. 117-129, 1976.

[123] F. Pezzella, G. Morganti, and G. Ciaschetti, "A genetic algorithm for the flexible job-shop scheduling problem," *Computers & Operations Research*, vol. 35, no. 10, pp. 3202-3212, 2008.

[124] S. Riedmiller and M. Riedmiller, "A neural reinforcement learning approach to learn local dispatching policies in production scheduling," in *Proc. IJCAI*, vol. 2, pp. 764-771, 1999.

[125] J. C. R. Bennett and H. Zhang, "WF^{2}Q: worst-case fair weighted fair queueing," in *Proc. IEEE INFOCOM'96*, 1996.

[126] C. A. Floudas and X. Lin, "Continuous-time versus discrete-time approaches for scheduling of chemical processes: A review," *Computers and Chemical Engineering*, vol. 28, no. 11, pp. 2109-2129, 2004.

[127] Csaba Szepesvári, "Algorithms for reinforcement learning," *Synthesis lectures on artificial intelligence and machine learning,* vol. 4, no. 1, pp. 1-103, 2010.

[128] R. Mijumbi, J. Gorricho, J. Serrat, M. Claeys, F. D. Turck, and S. Latré, "Design and evaluation of learning algorithms for dynamic resource management in virtual networks," In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, May 2014, pp. 1-9.

[129] V. Mnih, *et al.*,"Human-level control through deep reinforcement learning," *Nature*, 518(7540):529-533, 2015.

[130] R. S. Sutton and A. G. Barto, *Reinforcement learning: An Introduction.* Cambridge, U.K: Cambridge Univ. Press, 2011.

[131] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279-292, 1992.

[132] T. Jaakkola, M. I. Jordan, and S. P. Singh, "On the convergence of stachastic iterative dynamic programming algorithms," *Neural Comput.*, vol. 6, no. 6, pp. 1185-1201, 1994.

[133] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access in wireless networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 2, pp. 257-265, 2018.

[134] F. S. Melo, "Convergence of Q-learning: A simple proof," *Institute Of Systems and Robotics, Tech. Rep*, 2001, pp. 1-4.

[135] M. Pinedo, "Planning and scheduling in manufacturing and services." New York: Springer-Verlag, 2005.

[136] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Proc. IEEE Int. Conf. on Neural Networks*, 1995, pp. 1942-1948.

[137] K. Masuda, K. Kurihara, and E. Aiyoshi, "A penalty approach to handle inequality constraints in particle swarm optimization," *Proceedings of IEEE International Conference on Systems Man and Cybernetics*, Oct. 2010, pp. 2520-2525.

[138] J. D. Ullman, "NP-complete scheduling problems." *Journal of Computer and System sciences*, vol. 10, no. 3, pp. 384-393, 1975.

[139] G. Mosheiov, "Complexity analysis of job-shop scheduling with deteriorating jobs," *Discrete Applied Mathematics*, vol. 117, no. 1-3, pp. 195-209, Mar. 2002.

List of Publications

Journal Papers

1. **J. Li**, W. Shi, N. Zhang, and X. Shen, "Delay-aware VNF scheduling: A reinforcement learning approach with variable action set," *IEEE Transactions on Cognitive Communications and Networking (TCCN)*, 2020. Doi: 10.1109/TCCN.2020.2988908.

2. **J. Li**, W. Shi, P. Yang, Q. Ye, X. Shen, X. Li, and J. Rao, "A hierarchical soft RAN slicing framework for differentiated service provisioning," *IEEE Wireless Communications (WCM)*, 2020. Doi: 10.1109/MWC.001.2000010.

3. **J. Li**, W. Shi, N. Zhang, Q. Ye, S. Zhang, W. Zhuang, and X. Shen, "Joint virtual network topology design and embedding for service-oriented 5G core networks," *IEEE Transactions on Communications, (TCOM)*, under review, 2020.

4. **J. Li**, W. Shi, Q. Ye, N. Zhang, and X. Shen, "Multi-SFC embedding with end-to-end delay-guarantee: A game theoretical approach," to be submitted.

5. **J. Li**, W. Shi, P. Yang, and X. Shen, "Cost-aware dynamic service function chain provisioning in SDN-NFV enabled networks," to be submitted.

6. W. Shi, H. Zhou, **J. Li**, W. Xu, N. Zhang, and X. Shen, "Drone assisted vehicular networks: Architecture, challenges and opportunities," *IEEE Network*, vol. 32, no. 3, pp. 130-137, 2018.

7. Q. Ye, **J. Li**, K. Qu, W. Zhuang, X. Shen, and X. Li, "End-to-End quality of service in 5G networks: examining the effectiveness of a network slicing framework," *IEEE Vehicular Technology Magazine (MVT)*, vol. 13, no. 2, pp. 65-74, 2018.

8. W. Shi, **J. Li**, W. Xu, H. Zhou, N. Zhang, S. Zhang, and X. Shen, "Multiple drone-cell deployment analyses and optimization in drone assisted radio access networks," *IEEE Access*, vol. 6, pp. 12518-12529, 2018.

9. S. Zhang, W. Quan, **J. Li**, W. Shi, P. Yang, and X. Shen, "Air-ground integrated vehicular network slicing with content pushing and caching," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 36, no. 9, pp. 2114-2127, Oct. 2018.

10. W. Shi, **J. Li**, N. Cheng, F. Lyu, S. Zhang, H. Zhou, and X. Shen, "Multi-drone 3D trajectory planning and scheduling in drone-assisted radio access networks," *IEEE Transactions on Vehicular Technology (TVT)*, vol. 68, no. 8, pp. 8145-8158, 2019.

11. S. Zhang, B. Luo, **J. Li**, W. Shi, and X. Shen, "Hierarchical soft slicing to meet Multi-dimensional QoS demand in cache-enabled vehicular networks,"*IEEE Transactions on Wireless Communications (TWC)*, vol. 19, no. 3, pp. 2150-2162, Jan. 2020.

12. O. Alhussein, P. T. Do, Q. Ye, **J. Li**, W. Shi, W. Zhuang, X. Shen, X. Li, and J. Rao, "A virtual network customization framework for multicast services in 5G core networks," *IEEE Journal on Selected Areas in Communications (JSAC)*, accepted, Feb., 2020.

13. W. Shi, **J. Li**, H. Wu, C. Zhou, N. Cheng, and X. Shen, "Drone-cell trajectory planning and resource allocation for high-mobility users: A hierarchical DRL approach," *IEEE Internet of Things Journal*, submitted.

14. W. Shi, **J. Li**, P. Yang, Q. Ye, X. Shen, X. Li, and J. Rao, "Two-level soft RAN slicing for customized service provisioning," to be submitted.

Conference Papers

15. **J. Li**, N. Zhang, Q. Ye, W. Shi, W. Zhuang, and X. Shen, "Joint resource allocation and online virtual network embedding for 5G networks," in *Proc. IEEE GLOBECOM'17*, Singapore, Dec. 2017, pp. 1-6.

16. **J. Li**, W. Shi, Q. Ye, W. Zhuang, X. Shen, and X. Li, "Online joint VNF chain composition and embedding for 5G networks," in *Proc. IEEE GLOBECOM'18*, Abu Dhabi, UAE, Dec. 2018, pp. 1-6.

17. **J. Li**, W. Shi, P. Yang, and X. Shen, "On dynamic mapping and scheduling of service function chains in SDN/NFV-enabled networks," in *Proc. IEEE GLOBECOM'19*, Dec. 2019, pp. 1-6.

18. **J. Li**, W. Shi, N. Zhang, and X. Shen, "Reinforcement learning based VNF scheduling with end-to-end delay guarantee," in *Proc. IEEE/CIC ICCC'19*, Aug. 2019, pp. 572-577.

19. W. Shi, **J. Li**, W. Xu, H. Zhou, N. Zhang, and X. Shen, "3D drone-cell deployment optimization for drone assisted radio access networks", *Proc. IEEE/CIC ICCC'17*, Oct. 2017, pp. 1-6.

20. O. Alhussein, T.P. Do, **J. Li**, Q. Ye, W. Shi, W. Zhuang, X. Shen, X. Li, and J. Rao, "Joint VNF placement and multicast traffic routing in 5G core networks," *Proc. IEEE GLOBECOM'18*, Abu Dhabi, UAE, Dec. 2018, pp. 1-6.

21. W. Shi, **J. Li**, N. Cheng, F. Lyu, Y. Dai, H. Zhou, and X. Shen, "3D multi-drone-cell trajectory design for efficient IoT data collection", *Proc. IEEE ICC'19*, May. 2019, pp. 1-6.

Book Chapters

22. **J. Li**, W. Shi, and N. Zhang,"Resource allocation in SDN/NFV-enabled 5G networks," *Encyclopedia of Wireless Networks, Springer*, published, 2019.

23. W. Shi, **J. Li**, and N. Zhang,"Resource allocation in UAV-aided wireless networks," *Encyclopedia of Wireless Networks, Springer*, published, 2019.