

Song Yang
Nan He
Fan Li
Xiaoming Fu

Resource Allocation in Network Function Virtualization

Problems, Models and Algorithms



Springer


Resource Allocation in Network Function Virtualization


Song Yang • Nan He • Fan Li • Xiaoming Fu


Resource Allocation in Network Function Virtualization

Problems, Models and Algorithms

Song Yang 
Computer Science and Technology
Beijing Institute of Technology
Beijing, China

Nan He 
Computer Science and Technology
Beijing Institute of Technology
Beijing, China

Fan Li 
Computer Science and Technology
Beijing Institute of Technology
Beijing, China

Xiaoming Fu 
Institute of Computer Science
University of Göttingen
Göttingen, Germany

ISBN 978-981-19-4814-5 ISBN 978-981-19-4815-2 (eBook)
<https://doi.org/10.1007/978-981-19-4815-2>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2023

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Preface

Network Function Virtualization (NFV) has been emerging as an appealing solution that transforms complex network functions from dedicated hardware implementations to software instances running in a virtualized environment. Due to the numerous advantages such as flexibility, efficiency, scalability, short deployment cycles, as well as large reduction in CAPital EXpenditures (CAPEX) and OPerational EXpenditures (OPEX), NFV has been widely recognized as the next-generation network service provisioning paradigm. In NFV, the requested service is implemented by a sequence of Virtual Network Functions (VNFs) that can run on generic servers by leveraging the virtualization technology. These VNFs are pitched with a predefined order through which data flows traverse, and it is known as the Service Function Chaining (SFC). Nevertheless, facing these above benefits that NFV can bring, one of the main challenges is how to optimally place the VNFs on network nodes and find the routing path among VNF pairs, which is called the resource allocation problem in NFV. Due to the inherent nature of NFV, for example, there is a predefined order for the VNF pairs, the traffic rate gets changed after being processed by the VNF; the resource allocation problem in NFV is more complex and different from current similar resource allocation problem, for example, service placement problem and virtual network embedding problem. Because of this reason, this problem has received a broad attention from research community.

In this book, we first summarize and generalize the fundamental resource allocation problems and analyze their complexities. We also give the formal formulation and discuss about the current mainly adopted approaches for these problems. After that, we showcase our own research projects on resource allocation problem in NFV, including presenting approximation algorithm and Deep Reinforcement Learning (DRL) algorithm, and considering different QoS factors and in different network scenarios. Finally, we summarize the book by pointing out possible research directions in this area. We hope this book can lighten the beginners, especially senior undergraduate students and graduate students, to have a clear clue about resource allocation problems in NFV and inspire further research ideas in this area.

Acknowledgments

The work of Song Yang is partially supported by the National Natural Science Foundation of China (NSFC, No. 62172038, 61802018). The work of Fan Li is partially supported by the NSFC (No. 62072040). The work of Xiaoming Fu is partially supported by the EU H2020 RISE COSAFE project (No. 824019).

Beijing, China

Beijing, China

Beijing, China

Göttingen, Germany

May 2022

Song Yang

Nan He

Fan Li

Xiaoming Fu

Contents

1	Introduction	1
1.1	The Advantages and Disadvantages of NFV	2
1.2	NFV Standard and SDN	2
1.3	Book Organization	4
	References	5
2	Resource Allocation Problems Formulation and Analysis	7
2.1	Basic Problem Definition and Analysis	7
2.1.1	Examples	9
2.1.2	Problem Goals	9
2.1.3	Problem Formulation	11
2.1.4	Mainly Adopted Approaches	13
2.2	QoS Models in NFV	15
2.2.1	Delay Calculation of an SFC	15
2.2.2	NFV Resilience	17
2.3	Summary	20
	References	21
3	Delay-Aware Virtual Network Function Placement and Routing in Edge Clouds	23
3.1	Introduction	23
3.2	Related Work	25
3.2.1	Traffic/Cost-Aware VNF Placement and Routing in Generic Networks	26
3.2.2	Delay-Aware VNF Placement and Routing in Generic Networks	26
3.2.3	Delay-Aware VNF/Rule Placement and Routing in Edge Clouds and Software Defined Networks (SDN)	27
3.3	Network Delay Model	28
3.3.1	Service Function Chaining	28
3.3.2	Traversing Delay in a SFC in Edge Clouds	30

3.4	Problem Definition and Complexity Analysis	32
3.4.1	Problem Definition	32
3.4.2	An Exact Formulation	33
3.4.3	Complexity Analysis	35
3.5	Approximation Algorithm	36
3.5.1	Transformation from the INLP to the LP	36
3.5.2	Randomized Rounding Approximation Algorithm	38
3.5.3	Approximation Performance Analysis	39
3.6	Simulations	43
3.6.1	Simulation Setup	43
3.6.2	Simulation Results	45
3.7	Summary	50
	References	51
4	Delay-Sensitive and Availability-Aware Virtual Network Function Scheduling for NFV	55
4.1	Introduction	55
4.2	Related Work	57
4.2.1	Traffic and Cost-Aware VNF Placement and Routing	57
4.2.2	Delay-Sensitive VNF Scheduling	58
4.2.3	NFV Resiliency	59
4.3	Delay Calculation for a flow in a Service Function Chaining	59
4.3.1	Totally Ordered SFC	60
4.3.2	Partially Ordered SFC	61
4.4	VNF Placement Availability Calculation	62
4.5	Problem Definition and Complexity Analysis	65
4.5.1	Delay-Sensitive VNF Scheduling	65
4.5.2	Delay-Sensitive and Availability-Aware VNF Scheduling	65
4.5.3	Complexity Analysis	66
4.6	Solutions	67
4.6.1	Exact Solution	67
4.6.2	Heuristic	70
4.7	Simulations	72
4.7.1	Simulation Setup	72
4.7.2	Simulation Results	75
4.8	Summary	79
	References	80
5	Traffic Routing in Stochastic Network Function Virtualization Networks	83
5.1	Introduction	83
5.2	Related Work	85
5.2.1	Traffic Routing and VNF Placement in Deterministic NFV Networks	86
5.2.2	Traffic Routing and VNF Placement in Stochastic (NFV) Networks	88

5.3	Stochastic Link Weight	89
5.4	Problem Definition and Formulation	91
5.4.1	Problem Definition and Complexity Analysis	91
5.4.2	Problem Formulation	92
5.5	Multi-Constrained Traffic Routing Heuristic	93
5.6	Simulations	97
5.6.1	Simulation Setup	97
5.6.2	Simulation Results	99
5.7	Summary	103
	References	104
6	A-DDPG: Attention Mechanism-Based Deep Reinforcement Learning for NFV	107
6.1	Introduction	107
6.2	Related Work	109
6.2.1	Combinatorial Optimization Theory for NFV	109
6.2.2	DRL for NFV	110
6.3	Model and Problem formulate	111
6.3.1	Network Utility Model	111
6.3.2	Problem Definition and Formulation	113
6.4	Deep Reinforcement Learning	115
6.4.1	DRL Model Design	115
6.4.2	A-DDPG Framework	115
6.5	Performance Evaluation	121
6.5.1	Simulation Settings	121
6.5.2	Simulation Results	122
6.6	Summary	126
	References	126
7	Summarization and Future Work	129
7.1	Summarization of This Book	129
7.2	Emerging Topics and Future Works	131
7.2.1	State-of-the-Art	131
7.2.2	Security-Aware Resource Allocation	132
7.2.3	Wireless Virtual Network Functions and Other Network Application Domains	133
7.2.4	Mobility Management in NFV	133
7.2.5	SDN and NFV	133
	References	134

About the Authors

Song Yang received his BS degree in software engineering and MS degree in computer science from Dalian University of Technology, China, in 2008 and 2010, respectively, and his PhD degree from Delft University of Technology, the Netherlands, in 2015. From August 2015 to August 2017, he worked as postdoc researcher for the EU FP7 Marie Curie Actions CleanSky Project in Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG), Göttingen, Germany. He is currently an associate professor in the School of Computer Science and Technology at the Beijing Institute of Technology, China. His research interests focus on data communication networks, cloud/edge computing, and network function virtualization.

Nan He received her BE degree in computer and information from Hefei University of Technology, China, in 2016 and received her MS degree from Jilin University, China, in 2019. Currently, she is a PhD student in the School of Computer Science and Technology, Beijing Institute of Technology. Her research interests include network function virtualization and network resource optimization management based on reinforcement learning.

Fan Li received the PhD degree in computer science from the University of North Carolina at Charlotte in 2008, MEng degree in electrical engineering from the University of Delaware in 2004, and MEng and BEng degrees in communications and information system from Huazhong University of Science and Technology, China, in 2001 and 1998, respectively. She is currently a professor in the School of Computer Science and Technology at Beijing Institute of Technology, China. Her current research focuses on wireless networks, ad hoc and sensor networks, and mobile computing.

Xiaoming Fu received his PhD in computer science from Tsinghua University, Beijing, China, in 2000. He was then a research staff at the Technical University Berlin until joining the University of Göttingen, Germany, in 2002, where he has

been a professor in computer science and heading the Computer Networks Group since 2007. Prof. Fu's research interests include network architectures, protocols, and applications. He is currently an editorial board member of IEEE Network and IEEE Transactions on Network and Service Management, and has served on the organization or program committees of leading conferences such as INFOCOM, ICNP, ICDCS, MOBICOM, MOBIHOC, CoNEXT, ICN, and COSN. He is an IEEE Fellow, an ACM Distinguished Member, a fellow of IET, and member of the Academia Europaea.

Chapter 1

Introduction



With the continuous emergence¹ of new service patterns (e.g., Cloud Computing or Virtual Reality) and stringent demanding Quality of Service (QoS) to network services (e.g. High Dimensional Video), the IP traffic across the network increases exponentially. According to the Cisco Annual Internet Report (2020), the global average broadband speed continues to grow and will more than double from 2018 to 2023, from 45.9Mbps to 110.4Mbps. In the traditional network services provisioning paradigm, network functions (e.g., firewall or load balancer) which are also called middleboxes are usually implemented by the dedicated hardware appliances. Deploying hardware middleboxes is costly due to their high cost for design and production and also these middleboxes need to be configured and managed manually, which further increases the costs of service providers. Hence, the traditional network service paradigm fails to keep pace with satisfying the ever-increasing users' QoS requirements from the perspective of CAPital EXPenditures (CAPEX) and OPerational EXPenditures (OPEX), which poses a big challenge to network service providers.

Network Function Virtualization (NFV) which is first proposed by European Telecommunications Standards Institute (ETSI) (2013) in 2012 has emerged as an appealing solution, since it enables to replace dedicated hardware implementations with software instances running in a virtualized environment. In NFV, the requested service is implemented by a sequence of Virtual Network Functions (VNF) that can run on generic servers by leveraging the virtualization technology. These VNFs are pitched with a predefined order through which data flows traverse, and it is also known as the Service Function Chaining (SFC). Benefiting from virtualization technology, the SFC can be established by placing the requested VNFs on a network in a very efficient and agile manner. Moreover, one or more VNFs can be dynamically added or deleted with a very small cost and high efficiency to cope with the case when the requested SFCs have been changed. NFV allows to allocate

¹ Parts of this chapter is reprinted from Yang et al. (2021), with permission from IEEE.

network resources in a more scalable and elastic manner, offer a more efficient and agile management and operation mechanism for network functions and hence can lead to a significant reduction in CAPEX and OPEX for network service providers.

1.1 The Advantages and Disadvantages of NFV

From above, we can see that NFV can bring the following main benefits (Chayapathi et al. 2016):

- hardware flexibility: NFV enables network operators to upgrade system by adding more software-enabled functions, instead of buying hardware equipments.
- faster service life cycle: In NFV, the VNFs can be created and deleted in a dynamic manner. This is in contrast to the slow and costly deployment effort of network hardware.
- scalability and elasticity: NFV can satisfy the increasing (decreasing) requests in terms of CPU, bandwidth or storage by allocating (removing) more VNFs. This also makes NFV not to overprovision resources.
- simplifying the network operation and lower CAPEX and OPEX: NFV operates the network by managing VNFs, which is much easier and simpler than operating hardware equipments. In the meantime, NFV can bring significant reduction in CAPEX and OPEX.

In the meantime, NFV also has the following disadvantages:

- Since using the software-enabled function, the performance of NFV is largely affected by the virtualized architecture, data transmitting manners between software and hardware, etc. Hence, NFV still has a performance gap compared to the hardware-implemented network functions.
- It is still challenging to orchestrate VNFs in a resource efficient manner while still satisfying the user's QoS requirement.
- It is desired to design a uniform programming interface for NFV to implement various kinds of network functions.

1.2 NFV Standard and SDN

As shown in Fig. 1.1, the ETSI reference NFV framework mainly consists of three blocks (ETSI 2013):

- Network Function Virtualization Infrastructure (NFVI) block: This block sets the foundation of the whole architecture. The hardware resources of computing, storage and network, as well as the enabled virtualization resources by virtualizing technology are grouped in this block.

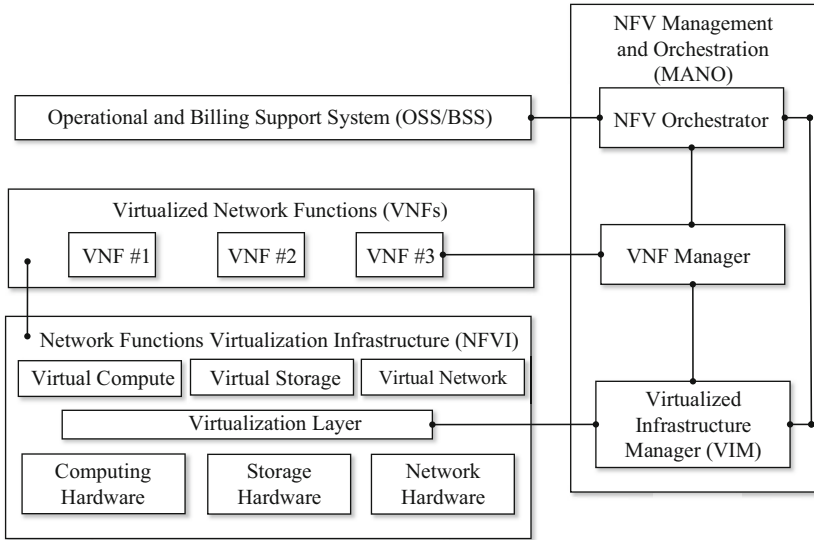
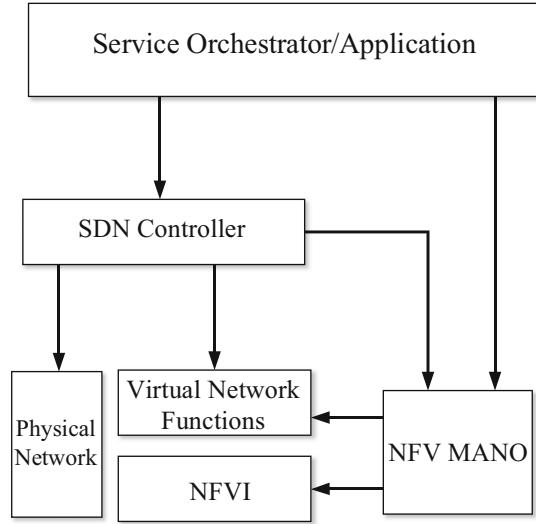


Fig. 1.1 The ETSI NFV framework (ETSI 2013; Chayapathi et al. 2016)

- **Virtualized Network Function (VNF) block:** This block builds on NFVI block and provide the software-implemented VNFs.
- **Management and Orchestration (MANO) block:** MANO is a separate block consisting of three subblocks in the architecture, and it interacts with both the NFVI and VNF blocks. More specifically, NFV orchestrator communicates with OSS/BSS system. VNF manager is in charge of VNFs block for lifecycle management of VNFs including instantiating, updating, scaling, and terminating. VIM is responsible for managing the NFVI resources such as computing, storage and network resources to VNFs.

Software-Defined Network (SDN) (Nunes et al. 2014) defines a network connection and management methodology that decouples control plane from data plane. In SDN, the network intelligence stays in a logical centralized software-controller (control plane), and network equipments (data plane) can be programmed via an open interface (e.g., OpenFlow). NFV and SDN are two independent innovation technologies, but they can work together for a joint flexible, efficient, agile network management and service development (Duan et al. 2016). For example, Fig. 1.2 presents a possible way (Chayapathi et al. 2016) how SDN can be integrated into a ETSI referenced NFV framework in order that they can work together. The addressed and surveyed resource allocation algorithms in this survey, can for instance be implemented in a SDN controller in a SDN-enabled NFV framework.

Fig. 1.2 A possible framework where SDN and NFV work together (Chayapathi et al. 2016)



1.3 Book Organization

In line with the benefits that NFV brings, one important question is how can we efficiently allocate network resources to establish requested SFC(s). This primarily deals with the (fundamental) VNF Placement and Traffic Routing (VPTR) problem and its variants, which is to place each user's requested VNFs on networks and find routes among each adjacent VNF pair without violating the node capacity and link bandwidth. Due to the inherent features and designing principles of NFV, the VPTR problem is different from the existing e.g., service placement and routing problem (Zhang et al. 2013) and virtual machine placement and routing problem (Jiang et al. 2012; Yang et al. 2017). For instance, when the requested VNFs are placed on networks, the route should traverse each located VNF one by one with a predefined order. In addition, we should also take the QoS parameters into account in the VPTR problem. For instance, in an SFC the end-to-end delay of the packets which traverse each VNF in this chaining is an important QoS to measure the performance of the NFV. To guarantee a satisfying network service, service providers need to promise a delay-sensitive performance and service to the customers. In this sense, the service providers may get revenue loss if the promised total delay is violated. Resilience is another important QoS parameter of NFV that defines the level the provided service can survive in the face of failures. Since, once a node or a link in an SFC fails, the whole SFC cannot operate and hence the provided service has to be stopped. Needless to say, the VPTR problem and its variants become more complicated when these QoS parameters are considered. Therefore, it is important to get an insight into the resource allocation problem in NFV from different dimensions, which is the main focus of this book.

This book consists of 7 chapters. In Chap. 2, we will summarize and generalize the fundamental resource allocation problems in NFV, analyze the problem complexities, present the formal problem formulation and discuss about the mainly adopted approaches to solve these problems. After that, Chap. 3 will study the Virtual Network Function (VNF) placement and routing problem by considering delay and availability QoS factors. In Chap. 4, we will continue to address the VNF placement and routing problem in edge clouds by proposing a randomized approximation algorithm. In Chap. 5, we will study the traffic routing problem in stochastic NFV networks. Chapter 6 will target how to apply Deep Reinforcement Learning (DRL) to solve the VNF placement and routing problem in NFV. Finally, we will conclude and present the future challenges for resource allocation problem in NFV in Chap. 7.

References

- Chayapathi R, Hassan SF, Shah P (2016). Network functions virtualization (NFV) with a touch of SDN. Addison-Wesley, Reading
- Cisco annual internet report (2018–2023) white paper (2020). <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html>
- Duan Q, Ansari N, Toy M et al. (2016) Software-defined network virtualization: an architectural framework for integrating SDN and NFV for service provisioning in future networks. *IEEE Netw* 30(5):10–16
- ETSI Publishes First Specifications for Network Functions Virtualisation. <http://www.etsi.org/news-events/news/700-2013-10-etsi-publishes-first-nfv-specifications>
- Jiang JW, Lan T, Ha S, Chen M, Chiang M (2012) Joint VM placement and routing for data center traffic engineering. In *Proc. of IEEE INFOCOM*, pp 2876–2880
- Nunes BA, Mendonca M, Nguyen XN, Obraczka K, Turletti T (2014) A survey of software-defined networking: past, present, and future of programmable networks. *IEEE Commun Surv Tutor* 16(3), 1617–1634 (2014)
- Yang S, Wieder P, Yahyapour R, Trajanovski S, Fu X (2017) Reliable virtual machine placement and routing in clouds. *IEEE Trans Parallel Distrib Syst* 28(10):2965–2978
- Yang S, Li F, Trajanovski S, Yahyapour R, Fu X (2021) Recent advances of resource allocation in network function virtualization. *IEEE Trans Parallel Distrib Syst* 32(2):295–314
- Zhang Q, Zhu Q, Zhani MF, Boutaba R, Hellerstein JL (2013) Dynamic service placement in geographically distributed clouds. *IEEE J Sel Areas Commun* 31(12):762–772

Chapter 2

Resource Allocation Problems

Formulation and Analysis



In this chapter, we focus on resource allocation problems formulation and analysis, including the basic problem definition and analysis, QoS models in NFV, and Resource Allocation in NFV.¹

2.1 Basic Problem Definition and Analysis

A network is represented by $\mathcal{G}(\mathcal{N}, \mathcal{L})$, where \mathcal{N} denotes a set of N nodes and \mathcal{L} stands for a set of L links. Each link $l \in \mathcal{L}$ is associated with capacity $c(l)$. In this book, the network is referred to a substrate network unless otherwise stated. We use R to represent the set of total requests and for a request $r(\alpha, F, \mathbf{w}) \in R$, α denotes the requested data rate (required bandwidth), F indicates the set of requested VNFs with orders, and $\mathbf{w} = [w_1, w_2, \dots, w_m]$ is a requirement vector with m requirements (e.g., cost, delay, availability, energy, etc.). The VNF $f \in F$ on node $n \in \mathcal{N}$ requires processing time of Ψ_n^f , which means the processing delay of packets arriving at the function f on node n .

Definition 2.1 In a given network $\mathcal{G}(\mathcal{N}, \mathcal{L})$ and for each request $r(\alpha, F, \mathbf{w}) \in R$, the VNF Placement and Traffic Routing (VPTR) problem is to place its requested VNFs on \mathcal{N} and find routes among each adjacent VNF pair without violating the node capacity and link bandwidth such that the requirement vector \mathbf{w} is satisfied.

¹ Parts of this chapter is reprinted from Yang et al. (2021b), with permission from IEEE.

Below are the VPTR problem variants.

The VNF Placement (VNFP) problem, without considering the traffic routing (sub)problem, is defined as follows:

Definition 2.2 For a network $\mathcal{G}(\mathcal{N}, \mathcal{L})$ and a set of requests R , suppose that the path between any node pair in the network is known/given, and each adjacent VNF pair chooses the given path to route the traffic. For each request $r(\alpha, F, \mathbf{w}) \in R$, the VNF Placement (VNFP) problem is to place its requested VNFs on \mathcal{N} without violating the node capacity such that \mathbf{w} is satisfied.

The VPTR problem turns into the TRAffic Routing (TRR) problem, when the requested VNFs are already placed on the network, and is defined as follows:

Definition 2.3 In a given network $\mathcal{G}(\mathcal{N}, \mathcal{L})$ and for a set of requests R , suppose that the requested VNFs for each r have already been placed on network nodes. For each request $r(\alpha, F, \mathbf{w}) \in R$, the TRAffic Routing (TRR) problem is to route the traffic between each VNF pair without violating the link capacity such that \mathbf{w} is satisfied.

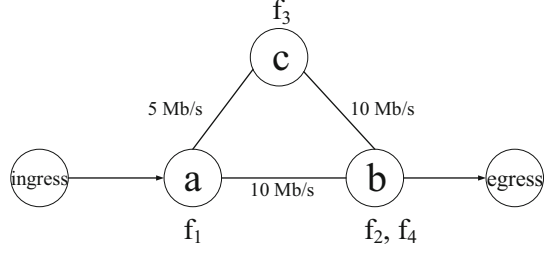
Another problem variant, called VNF Redeployment and Consolidation (VRC) problem, is defined as follows:

Definition 2.4 In a given network $\mathcal{G}(\mathcal{N}, \mathcal{L})$ where a set of existing SFCs are already deployed in the network, the VNF Redeployment and Consolidation (VRC) problem is to redeploy the already existing SFCs such that the requirement vector \mathbf{w} is satisfied.

It is worthwhile to mention that in the Virtual Network Embedding (VNE) problem (Fischer et al. 2013), a Virtual Network Request (VNR) consists of a set of required virtual nodes and virtual links. The VNE problem is to map each VNR to the substrate (physical) network such that each virtual node is placed on a different substrate node with sufficient capacity and each virtual link is allocated a path with sufficient bandwidth. Although the VNE problem shares some similarities with the VPTR problem, they have the following differences:

- In the VPTR problem, the required VNFs in an SFC have a predefined order and the traffic has to traverse each placed VNF under this order. On the contrary, there is no required order between the requested virtual nodes in the VNE problem.
- One or more required VNFs can be placed in one node in the VPTR problem, however, each requested virtual node in the VNE problem has to be mapped on different physical nodes.
- After the traffic goes through one VNF for processing, its rate may get changed (increased or decreased) in the VPTR problem. However, the traffic rate mostly keeps unchanged among different virtual node pairs in the VNE problem.

Fig. 2.1 An example of describing the VPTR problem



2.1.1 Examples

Let us use an example in Fig. 2.1 to describe the VPTR problem.² Consider a network with 5 nodes (including ingress node and egress node) in Fig. 2.1, and the capacity of nodes a , b and c are 8, 11 and 15, respectively. The link capacities are shown above the link. Suppose a request r asks for an SFC with an order $f_1 - f_2 - f_3 - f_4$ consisting of 4 VNFs, whose resource requirements are 8, 6, 14 and 5. We have to place these VNFs on the nodes as shown in the network without violating the node capacity. Let $\alpha = 5$ Mb/s for r , then the whole route in this SFC is $\text{ingress} \rightarrow a \rightarrow b \rightarrow c \rightarrow b \rightarrow \text{egress}$ without violating each traversed link capacity. We notice that this route is not a simple path since it contains a loop. From this example, we see that the route traversing an entire SFC is not necessarily a simple path (Even et al. 2016), which makes the VPTR problem more complex.

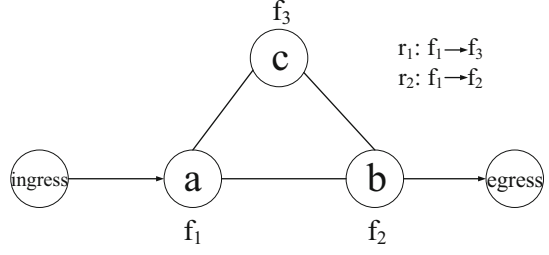
Next, we start with an example to illustrate the VRC problem. Suppose there are 2 requests, where r_1 asks for an SFC of $f_1 - f_3$ and r_2 requests an SFC of $f_1 - f_2$. In Fig. 2.2a f_1 , f_2 and f_3 are originally placed on a , b and c to serve these 2 requests. After consolidation and redeployment, f_3 is migrated from c to b and we switch off c for e.g., energy saving.

2.1.2 Problem Goals

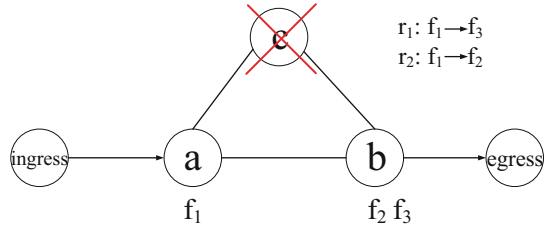
So far, we have formally defined the VPTR problem and its variants in Sect. 2.1.3 and described these problems by using examples in Sect. 2.1.1. However, we have not mentioned problem goals for these problems. For instance, in addition to finding a feasible VNF placement and routing solution for the VPTR problem, what is the

² The ingress node and egress node represent the entry and exit of a request, respectively. However, they are usually not considered in the problem and solution. For completeness, we put them in all the illustrating figures in this survey.

Fig. 2.2 An example of describing the VRC problem. (a) Before migration and consolidation. (b) After migration and consolidation



(a) Before migration and consolidation.



(b) After migration and consolidation.

problem goal to achieve? We could understand the problem goal as the objective function of the optimization problem. Below are frequently used problem goals:

- **Maximum link utilization (MLU):** For a link l , its link utilization (denoted by u_l) is defined as $\frac{v(l)}{c(l)}$, where $v(l)$ denotes the actual traversing flow on l , and $c(l)$ indicates the total capacity of l . The maximum link utilization of a network is the maximum of the link utilization over all its links. The MLU is to be minimized. Sometimes, the reciprocal of the MLU is also used, which is called *network throughput*. In these cases, maximizing the network throughput is equivalent to minimizing MLU.
- **QoS parameters:** The main QoS parameters include service delay, availability, energy consumption, and so on. These parameters can quantitatively reflect how well an NFV service provides to customers.
- **Costs and Revenue:** Mostly used in the context with network operators, the network cost represents the operational expenditure or financial aspect of deploying VNFs on the nodes and routing traffic among used links. Meanwhile, revenue, in particular, refers to the net value earned by serving traffic demands, optimized under capacity constraints.

2.1.3 Problem Formulation

In this subsection, we formulate the (basic) VPTR problem³ in the fully ordered SFC as an Integer Nonlinear Programming (INLP) (Yang et al. 2022). We start by some necessary notations and variables:

- R : A request set and for each request $r(\alpha, F, \mathbf{w}) \in R$, α denotes the requested data rate (required bandwidth), F indicates the set of requested VNFs with orders, and $\mathbf{w} = [w_1, w_2, \dots, w_m]$ is a requirement vector with m requirements (e.g., cost, delay, availability, energy, etc.).
- f_i and f_j : Two consecutive VNFs in the required SFC.
- $X_n^{f,r}$: A Boolean variable which returns 1 if the VNF f requested by r is placed on node $n \in \mathcal{N}$, and 0 otherwise.
- $Y_{f_i, f_j, r}^{(u,v)}$: A Boolean variable which returns 1 if the flows between VNFs f_i and f_j of r traverse link (u, v) , and 0 otherwise.

Routing Constraints

$$\sum_{(u,v) \in \mathcal{L}} Y_{f_i, f_j, r}^{(u,v)} = 1 \quad u \in \mathcal{N} : X_u^{f_i, r} = 1 \& X_u^{f_j, r} \neq 1, r \in R \quad (2.1)$$

$$\sum_{(u,v) \in \mathcal{L}} Y_{f_i, f_j, r}^{(u,v)} = 1 \quad v \in \mathcal{N} : X_v^{f_j, r} = 1 \& X_v^{f_i, r} \neq 1, r \in R \quad (2.2)$$

$$\begin{aligned} \sum_{(u,v) \in \mathcal{L}} Y_{f_i, f_j, r}^{(u,v)} &= \sum_{(v,w) \in \mathcal{L}} Y_{f_i, f_j, r}^{(v,w)} \\ v \in \mathcal{N} : X_v^{f_i, r} &\neq 1 \& X_v^{f_j, r} \neq 1, r \in R \end{aligned} \quad (2.3)$$

Placement Constraint

$$\sum_{n \in \mathcal{N}} X_n^{f,r} = 1 \quad \forall r \in R, f \in r \quad (2.4)$$

Link Capacity Constraint

$$\sum_{r(\alpha, F, \mathbf{w}) \in R, (f_i, f_j) \in r} Y_{f_i, f_j, r}^{(u,v)} \cdot \alpha \leq c(u, v) \quad \forall (u, v) \in \mathcal{L} \quad (2.5)$$

³ On basis of the basic VPTR problem, the availability/resilience-aware VPTR problem only additionally considers the reliability constraint on basis of the basic VPTR problem. It follows similarly for the delay-aware VPTR problem, energy-aware VPTR problem, and so on.

Node Processing Capacity Constraint

$$\sum_{r(\alpha, F, \mathbf{w}) \in R, f \in r} X_n^{f,r} \cdot \eta(f) \leq \pi(n) \quad \forall n \in \mathcal{N} \quad (2.6)$$

There is no objective (needed) in this formulation, but we can add the minimization of e.g., number of nodes or links, total costs, etc. which depends on specific network necessities. Equations (2.1)–(2.3) are the flow conservation constraints (Zhu and Mukherjee 2002; Wu et al. 2007) that account for the routing from a source to a destination and ensures to find a path from u to v . More specially, Eq. (2.1) ensures that if f_i is placed on node u and f_j is not placed on node u , then the sum of outgoing traffic from u is equal to 1. Equation (2.2) ensures that if f_j is placed on node v and f_i is not placed on node v , then the sum of incoming traffic to v is equal to 1. Equation (2.3) ensures that if f_i and f_j are not placed on node v , then the sum of incoming traffic to v is equal to its outgoing traffic. Equation (2.4) ensures that each required VNF f must be placed on one node in the network. Equation (2.5) ensures that the total allocated bandwidth on each link does not exceed its maximum capacity, which is denoted by $c(u, v)$. Equation (2.6) ensures that the total assigned processing capacity on each node does not exceed its maximum processing capacity, where $\eta(f)$ and $\pi(n)$ represent the required resource of f and total available resource of node n , respectively.

When $X_n^{f,r}$ or $Y_{f_i, f_j, r}^{(u,v)}$ is known as an input, the above INLP accordingly changes to solve the TRR problem or VNFP problem. To solve the VRC problem, it is set in R that (some of) requests have already been provisioned by placing the required VNFs on nodes and selecting appropriate paths. On the basis of that, we can run the INLP in Eqs. (2.1)–(2.6) to solve it. That is to say, Eqs. (2.1)–(2.6) are quite general. It is also worthwhile to mention that Eqs. (2.1)–(2.6) solve the basic VPTR problem in generic networks without considering QoS requirements such as cost, delay, availability, energy. However, we can extend this formulation to solve the VPTR problem by taking into account QoS requirements in different network architectures. For brevity, please refer to Yang et al. (2022) for the formulation of VPTR problem for delay-awareness and availability-awareness, Jang et al. (2017) for the formulation of VPTR problem for energy-awareness. Moreover, the formulation of the VPTR problem is not unique as we state above and also a slight change of the problem input will result in different formulation. For instance, Yang et al. (2021a) assumes that a set of paths between each node pair in the network is given, then the VPTR problem formulation becomes different from Eqs. (2.1)–(2.3) since the multi-community routing constraints do not exist in this scenario. Nevertheless, since the INLP has exponential running time, it cannot return a solution in a reasonable time especially when the problem size is large, which calls for polynomial running time approximation algorithms or efficient heuristics to solve this problem.

2.1.4 Mainly Adopted Approaches

As we summarize, the most frequent approaches to deal with NFV resource allocation include combinatorial optimization theory (e.g., randomized/LP rounding, primal-dual approximation), Deep Reinforcement Learning, Game theory, etc. Below, we will briefly provide the representative approaches and their descriptions.

1. Randomized/LP rounding: It first formulates the problem to be solved as an integer linear program (ILP), and then computes an optimal fractional solution to the linear programming relaxation (LP) of the ILP. After that, it rounds the fractional solution of the LP to an integer solution of the ILP (original problem) within a gap to the optimal solution.
2. Primal-Dual approximation: *The primal-dual method (or primal-dual schema) is another means of solving linear programs. The basic idea of this method is to start from a feasible solution y to the dual program, then attempt to find a feasible solution x to the primal program that satisfies the complementary slackness conditions. If such an x cannot be found, it turns out that we cannot find a better y in terms of its objective value. Then, another iteration is started. An approximate solution to the primal IP and a feasible solution to the dual LP can be constructed simultaneously and improved step by step. In the end, the approximate solution can be compared with the dual feasible solution to estimate the approximation ratio (Lecture).*
3. Markov Approximation: *It first derives log-sum-exp approximation of the optimal value of a combinatorial problem, and this leads to a solution that can be realized by time-reversible Markov chains. Certain carefully designed Markov chains among this class can yield distributed algorithms for solving the network optimization problem approximately (Chen et al. 2013).*
4. Local Search: *Local search moves from solution to solution in the space of candidate solutions (the search space) by applying local changes, until a solution deemed optimal is found or a time bound is elapsed (Wikipedia 2019).*
5. Alternating Direction Method of Multipliers (ADMM) (Boyd et al. 2011): It first separates the objective and variables into two parts, and then alternatively optimizes one set of variables that accounts for one part of the objective to iteratively reach the optimum.
6. Column Generation (Ford and Fulkerson 1958): It begins with a small part of a problem as a restricted master problem (RMP), and then solves that part by analyzing that partial solution to discover the next part of the problem. After that, one or more variables are added to the model, and then resolves the enlarged model. This process repeats until it achieves a satisfactory solution to the whole of the problem.
7. Generalized Benders Decomposition (GBD) (Geoffrion 1972): *the original problem is projected onto the space of first-stage variables and reformulated into a dual problem that contains an infinite number of constraints, which is then relaxed into a lower bounding problem with a finite subset of these constraints. After fixing the first-stage variables to the solution of the lower bounding*

problem, the original problem becomes an upper bounding problem, which can naturally be decomposed into smaller subproblems for each of the scenarios. The solutions of a sequence of upper bounding problems give a sequence of non-decreasing upper bounds on the optimal objective function value while the solutions of a sequence of lower bounding problems give a sequence of non-increasing lower bounds. An optimum of the original problem is obtained when the upper and lower bounds converge (Li et al. 2011).

8. Deep Reinforcement Learning (DRL): DRL is a method of solving decision-making problems, combining the idea of Reinforcement Learning (RL) and the structure of Deep Learning (DL). The essence of RL is learning through trial-and-error interactions with the environment. The RL agent first senses the state of the environment and then selects an action according to the current state. After reaching a new state, the agent receives a reward associated with the new state. The obtained reward tells the agent how good or bad the taken action was. The aim of the agent is to find the optimal policy. The policy is the strategy that the agent employs to determine the next action based on the current state so as to maximize the reward. The role of DL in DRL is to use the powerful representation capabilities of neural networks to fit the strategy of RL agent, such as to deal with the problems in a complex dynamic environment.
9. Game theory: Game theory is the study of theoretical models of strategic interaction among competing players. It is usually used to solve the problem of choosing the optimal decisions for each player in the presence of competition, cooperation, or conflict. A game model usually consists of two or more players, a set of strategies and utility functions. In general, the players which make decisions independently can be malicious, cooperative or selfish, and a player's success in making decisions depends on the choices of others. In game theory, players compete with other opponents taking turns sequentially to maximize their payoff until they achieve the Nash Equilibrium (NE). A NE is a steady state where no player has an incentive to deviate from its chosen strategy after considering the choices of other opponents.

In general, Randomized/LP rounding, Primal-Dual and Markov Approximation techniques are often used to devise the approximation algorithms. Local-search approach can aid to accelerate the speed to find local optimum solutions, but there is no guarantee to find the global optimal solution which makes it a heuristic solution. ADMM, Column generation and GBD can converge to find an optimal solution with fast speed in practice, but theoretically, they still have exponential running time. DRL provides a new perspective to solve combinatorial optimization problems (e.g., NFV resource allocation problem), but DRL has uncertain convergence time and/or long training time. Game theory provides a distributed manner to solve the resource allocation problem in NFV, but since due to the lack of global network information, it cannot essentially solve the resource allocation problem in NFV in order to get the optimal solution as well as taking into account more QoS parameters such as delay and availability.

2.2 QoS Models in NFV

While delay and availability are two important QoS parameters in an NFV service, we study how to quantitatively calculate them in NFV in general. Moreover, we present a VNF placement availability calculation model. The other QoS models will be introduced when the respective literature is mentioned due to its simplicity or similarity to the introduced models in this section.

2.2.1 Delay Calculation of an SFC

According to the feature of function chains, there are three kinds of dependencies between VNFs, namely, (1) non-ordered: there is no dependency between VNFs, (2) totally-ordered: there is a total dependency order on the VNF set, and (3) partially-ordered: there exists dependency among a subset of VNFs. The traversing delay calculation mainly depends on the totally-ordered and partially-order scenario, since the non-ordered scenario does not occur frequently in practice. For example, Fig. 2.3a illustrates a totally ordered SFC.

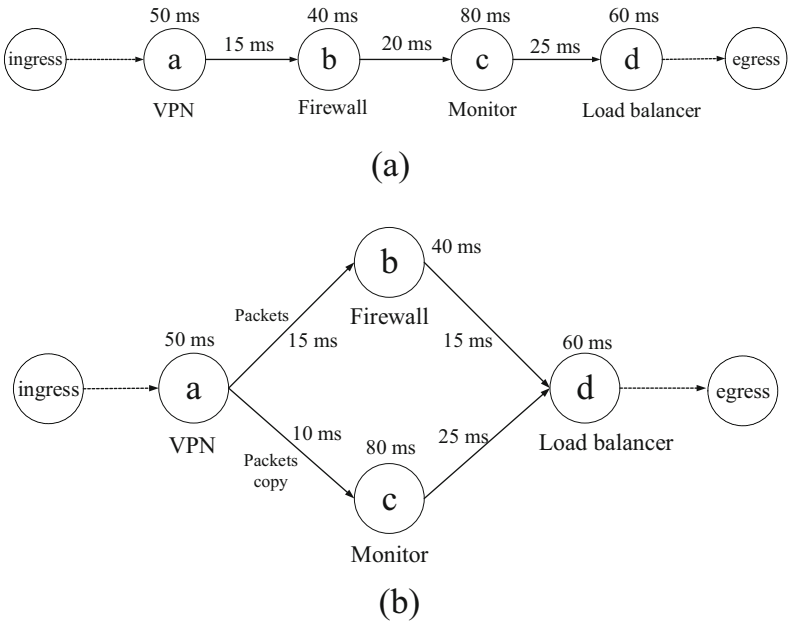


Fig. 2.3 Totally ordered SFC v.s. partially ordered SFC, derived from Sun et al. (2017). (a) A totally ordered SFC. (b) A partially ordered SFC

We assume that an SFC contains a set of $|F|$ ordered VNFs, $f_1, f_2, \dots, f_{|F|}$ which need to be placed in the network. Suppose there is a total dependency order on $|F|$ VNFs, then the traversing delay in a totally ordered SFC is calculated as follows:⁴

$$\sum_{f \in F, n \in N} \Psi_n^f + \sum_{l \in \mathcal{L}^s} d(l) \quad (2.7)$$

where Ψ_n^f represents the processing delay for function f on node n , \mathcal{L}^s denotes the link set that connects the placed VNF, and $d(l)$ indicates the flow delivering delay on link l . For example, in Fig. 2.3a where the flow delivering delay and node processing delay are associated, the traversing delay is $\underbrace{(50 + 40 + 80 + 60)}_{\text{total node delay}} + \underbrace{(15 + 20 + 25)}_{\text{total link delay}} = 290$ ms.

However, since a monitor function only maintains the packets and does not change the packets in practice, this indicates that firewall and monitor functions are not dependent on each other. In this sense, these two functions can work in two cases, namely, (1) they work sequentially with any order between each other in an SFC, and (2) they work in parallel in an SFC. In case (1), ingress-*a-c-b-d*-egress is also a feasible possible feasible SFC alternative to Fig. 2.3a. In case (2), firewall and monitor work in parallel as shown in Fig. 2.3b. After that, firewall and monitor functions send their individual processed packets to the load balancer. As a result, the load balancer only needs to select the packets from firewall and process them. Therefore, there may exist multiple data routes traversing from the ingress node to the egress node in a partially ordered chaining. As a result, the traversing delay of a parallel SFC is equal to the longest delay route, which means that the sum of node delay and link delay in this data delivering route is the largest. For example, the traversing delay of SFC in Fig. 2.3b is equal to the route ingress $\rightarrow -c-d \rightarrow$ egress, whose delay is $50 + 10 + 80 + 25 + 60 = 225$ ms, which is less than the one in Fig. 2.3a. It is obvious to see that a partially ordered SFC can shorten the service delay compared to a totally ordered delay, but this comes at the expense of consuming more link bandwidth. Therefore, whether to use totally ordered SFC or partially ordered SFC depends on the tradeoff of network budget and application requirement.

From the above example we see that given a required non-ordered chain, there exists more than one possible SFCs (either sequential or parallel). We call the process of generating such possible SFCs as chain composition problem (Herrera and Botero 2016). Moreover, Fig. 2.3 is also called as SFC forwarding graph (Herrera and Botero 2016). In this survey, we assume that the SFC forwarding graph is given an input to the resource allocation problem in NFV, as most of existing

⁴ We mention that it is possible more than one VNFs can be placed on one same node. In that case, we only calculate their processing time and regard that their communication delay is neglectable, leveraged by the NIC virtualization techniques (e.g., VIRTIO, DPDK, SRIOV).

literature do, we therefore do not address the chain composition problem and refer the readers to (Herrera and Botero 2016) for more details.

2.2.2 NFV Resilience

Resilience is another important QoS parameter of NFV that defines the level the provided service can survive in the face of failures. Since, once a node or a link in an SFC fails, the whole SFC cannot operate and hence the provided service has to be stopped. An efficient way to tackle this issue is to provide redundant SFCs, and these SFCs usually place requested VNFs on different nodes (called node-disjoint) to prevent node failure and/or select paths that do not traverse same link (called link-disjoint) to overcome link failure. By doing this, the service is normally provisioned by the primary SFC, and in case of any node/link failure in the primary SFC, the backup SFC will be activated to work. This method is also known as SFC protection.

Hmaity et al. (2016) study the SFC protection in three cases, namely, (1) end-to-end protection: a primary SFC and a backup SFC are completely (both node- and link-) disjoint; (2) link-protection, and (3) node-protection. Figure 2.4 reflects these three protection schemes for placing 3 VNFs in the network, where the red line indicates the primary path and the blue line represents the backup path.

However, the proposed SFC protection schemes in (Hmaity et al. 2016) do not consider the nodes' and links' failure probabilities. For instance, if both primary SFC and backup SFC contain nodes or links with higher failure probabilities, then this protection can also not guarantee a reliable service. Moreover, in (Hmaity et al. 2016) it is only considered $k = 2$ SFCs for protection (i.e., one VNF replica). However, in practice, in order to provide a more reliable service, we need to provide $k \geq 2$ SFCs sometimes. In order to deal with this issue, in (Yang et al. 2022), we present a quantitative model to calculate the VNF placement availability based on the proposed protected schemes in Fig. 2.4 (Hmaity et al. 2016). Similar to Kong et al. (2017), we consider both the node and link availabilities to quantitatively measure the reliability of SFC protection for any $k \geq 1$ VNF replicas/backups. More specifically, the availability of a system is the fraction of time that the system

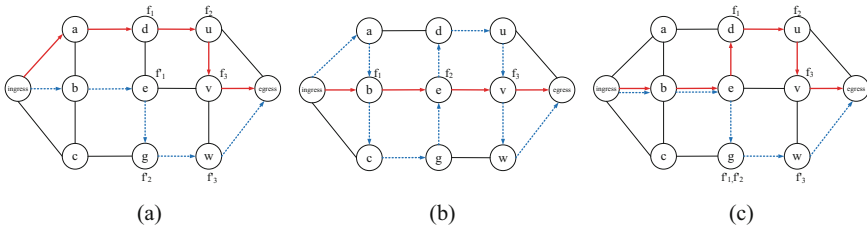


Fig. 2.4 Protection schemes (Hmaity et al. 2016). (a) End-to-end protection. (b) Link protection. (c) Node protection

is operational during the entire service time. The availability A_j of a network component j can be calculated as (McCool 2003):

$$A_j = \frac{MTTF}{MTTF + MTTR} \quad (2.8)$$

where $MTTF$ represents Mean Time To Failure and $MTTR$ denotes Mean Time To Repair. A node in the network represents a server, and a link denotes a physical link connecting two physical servers. Their availabilities are equal to the product of the availabilities of all their components (e.g., hard disk and memory for a node, amplifiers and fibers for a link). For a server n , let $A(n)$ and $p(n)$ denote its availability value and failure probability, respectively, then we have $A(n) = 1 - p(n)$. Node failures can vary from server age, the number of hard disks, etc. (Vishwanath and Nagappan 2010), and they can be caused by hardware component failure, software bugs, power loss events, etc. (Ford et al. 2010). In reality, we can obtain the server's availability value by accessing the detailed logs recording every hardware component repair/failure incident during the lifetime of the server. The details for characterizing server failures and statistically calculating node availability can be found in (Vishwanath and Nagappan 2010; Ford et al. 2010) and work therein. Reference (Gill et al. 2011) and the work therein provide failure statistics for other network devices (e.g., switches) in datacenters.

We distinguish and analyze the VNF placement availability under two different cases, namely (1) Unprotected SFC: only one SFC is allowed to be placed in the network, and (2) Protected SFC: at most $k \geq 2$ SFCs can be placed in the network.

Suppose an unprotected SFC places VNFs on w nodes n_1, n_2, \dots, n_w and traverses m links l_1, l_2, \dots, l_m , its availability is calculated as:

$$\prod_{i=1}^w A_{n_i} \cdot \prod_{j=1}^m A_{l_j} \quad (2.9)$$

where $\prod_{i=1}^w A_{n_i}$ denotes the availability of all the used nodes⁵ and $\prod_{j=1}^m A_{l_j}$ indicates the availability of all the traversed links.⁶

In the protected SFC, we stress that it is composed of (maximum) k unprotected SFCs. For the ease of clarification, we further term each of the k unprotected SFC in the protected placement as placement group ρ_i . We denote by ρ_1 the primary placement group. Since different placement groups may place one or more VNFs on the same node and/or traverse the same link, we distinguish the protected placement as two cases, namely (1) *fully protected SFC*: each one of k placement groups places VNF on different nodes and traverses different links and (2) *partially protected SFC*:

⁵ Even though one node can host more than one VNFs, its availability will be counted only once.

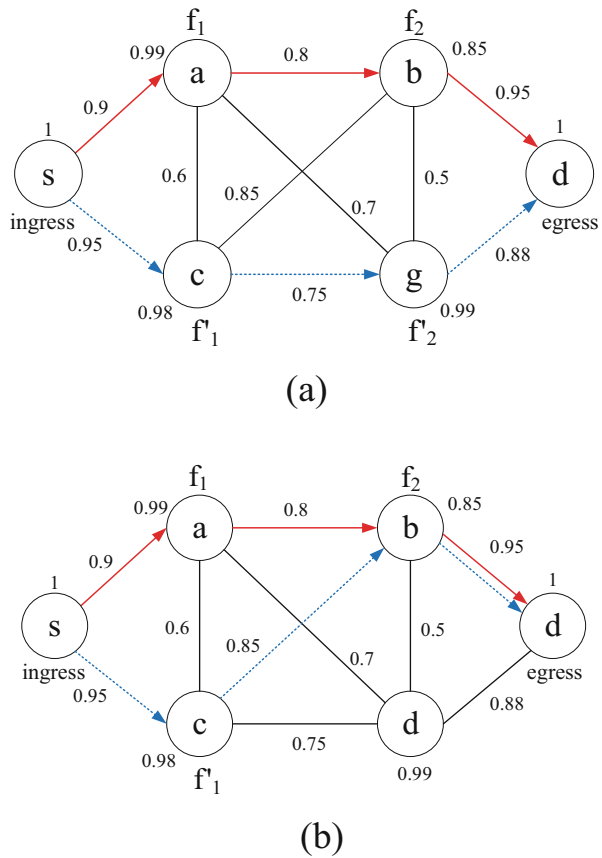
⁶ It is possible that one SFC traverses one link multiple times, but we consider the availability of each of the traversed links only once.

at least two placement groups place one or more VNFs on the same node and/or traverse the same link. In the fully protected placement case, the availability can be calculated as:

$$1 - \prod_{i=1}^k (1 - A_{\rho_i}) \quad (2.10)$$

Equation (2.10) indicates that the availability of k SFCs is equal to the probability that at least one SFC can work normally (does not fail). For example, in Fig. 2.5a where the node and link availabilities are associated, we use s and d to represent ingress and egress nodes. For the ease of expression, their availabilities are always 1. SFC s - a - b - d and s - c - g - d are fully protected, since they do not contain any same link or node. As a result, the total availability of these two SFCs are: $1 - (1 - 0.9 \cdot 0.99 \cdot 0.8 \cdot 0.85 \cdot 0.95) \cdot (1 - 0.95 \cdot 0.98 \cdot 0.75 \cdot 0.99 \cdot 0.88) \approx 0.8338$.

Fig. 2.5 VNF placement availability calculation. (a) Fully protected SFC. (b) Partially protected SFC



Next, we consider a general scenario where at least two of k placement groups place the same VNF on the same node or traverse the same links. In this case, Eq. (2.10) cannot be used to calculate the availability in this scenario since the availabilities of overlapped nodes or links will be counted more than once. To amend this, we use a new operator \circ . Suppose there are m different nodes n_1, n_2, \dots, n_m with availabilities $A_{n_1}, A_{n_2}, \dots, A_{n_m}$. For a node n_x with availability A_{n_x} , \circ can be defined as follows:

$$A_{n_1} \cdot A_{n_2} \cdot \dots \cdot A_{n_m} \circ A_{n_x} = \begin{cases} \prod_{i=1}^m A_{n_i} & \text{if } \exists n_i = n_x \\ \prod_{i=1}^m A_{n_i} \cdot A_{n_x} & \text{otherwise} \end{cases} \quad (2.11)$$

where the \circ computations for link availabilities can be defined analogously.

Let \coprod denote consecutive \circ operations of the different sets, and it is commutative, associative and distributive. Hence, the availability of k partially protected SFCs can now be represented as:

$$\begin{aligned} & 1 - \coprod_{i=1}^k (1 - A_{\rho_i}) \\ &= 1 - (1 - A_{\rho_1}) \circ (1 - A_{\rho_2}) \circ \dots \circ (1 - A_{\rho_k}) \\ &= \sum_{i=1}^k A_{\rho_i} - \sum_{0 < i < j \leq k} A_{\rho_i} \circ A_{\rho_j} + \\ & \quad \sum_{0 < i < j < u \leq k} A_{\rho_i} \circ A_{\rho_j} \circ A_{\rho_u} + \dots + (-1)^{k-1} \coprod_{i=1}^k A_{\rho_i} \end{aligned} \quad (2.12)$$

where A_{ρ_i} denotes the availability of placement group ρ_i and can be calculated from Eq. (2.9). Let us take Fig. 2.5b as an example. SFCs s-a-b-d and s-c-b-d are partially protected, since they jointly place VNF f_2 on node b and they traverse the same link (b, d) as well. Consequently, the total availability is calculated as $1 - (1 - A_{sa} \circ A_a \circ A_{ab} \circ A_b \circ A_{bd}) \circ (1 - A_{sc} \circ A_c \circ A_{cb} \circ A_b \circ A_{bd}) = A_a \circ A_b \circ A_d \circ A_{sa} \circ A_{ab} \circ A_{bd} + A_c \circ A_b \circ A_d \circ A_{sc} \circ A_{cb} \circ A_{bd} - A_a \circ A_b \circ A_d \circ A_{sa} \circ A_{ab} \circ A_{bd} \circ A_c \circ A_{cb} \circ A_{bd} = A_a \cdot A_b \cdot A_d \cdot A_{sa} \cdot A_{ab} \cdot A_{bd} + A_c \cdot A_b \cdot A_d \cdot A_{sc} \cdot A_{cb} \cdot A_{bd} - A_a \cdot A_b \cdot A_d \cdot A_{sa} \cdot A_{ab} \cdot A_{bd} \cdot A_c \cdot A_{cb} \cdot A_{bd} \approx 0.759$.

2.3 Summary

In this chapter, we introduce basic problem formulation and analysis and related work on QoS models in NFV. For basic problem formulation and analysis, we generalize and analyze four representative resource allocation problems (variants), namely, (1) the VNF Placement and Traffic Routing problem, (2) VNF Placement

problem, (3) Traffic Routing problem in NFV and (4) the VNF Redeployment and Consolidation problem in Sect. 2.1. After that, we briefly provide the representative approaches to deal with NFV resource allocation and their descriptions. Subsequently, we study how to quantitatively QoS parameters in a NFV service in Sect. 2.2. After introducing simple basic concepts in NFV, the corresponding theories and algorithms will be discussed in the next chapters of this book.

References

- Boyd S, Parikh N, Chu E, Peleato B, Eckstein J et al. (2011) Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found Trends® Mach Learn* 3(1):1–122
- Chen M, Liew SC, Shao Z, Kai C (2013) Markov approximation for combinatorial network optimization. *IEEE Trans Inform Theory* 59(10):6301–6327
- Even G, Rost M, Schmid S (2016) An approximation algorithm for path computation and function placement in SDNs. In: *International colloquium on structural information and communication complexity*. Springer, Berlin, pp 374–390
- Fischer A, Botero JF, Beck MT, de Meer H, Hesselbach X (2013) Virtual network embedding: a survey. *IEEE Commun Surv Tutor* 15(4):1888–1906
- Ford Jr. LR, Fulkerson DR (1958) A suggested computation for maximal multi-commodity network flows. *Manag Sci* 5(1):97–101
- Ford D, Labelle F, Popovici FI, Stokely M, Truong VA, Barroso L, Grimes C, Quinlan S (2010) Availability in globally distributed storage systems. In *OSDI*, vol. 10, pp 1–7
- Geoffrion AM (1972) Generalized benders decomposition. *J Optim Theory Appl* 10(4):237–260
- Gill P, Jain N, Nagappan N (2011) Understanding network failures in data centers: measurement, analysis, and implications. *ACM SIGCOMM Comput Commun Rev* 41(4):350–361
- Herrera, JG, Botero JF (2016) Resource allocation in NFV: a comprehensive survey. *IEEE Trans Netw Serv Manag* 13(3):518–532
- Hmaity A, Savi M, Musumeci F, Tornatore M, Pattavina A (2016) Virtual network function placement for resilient service chain provisioning. In: *8th IEEE/IFIP international workshop on resilient networks design and modeling (RNDM)*, pp 245–252
- Jang I, Suh D, Pack S, Dán G (2017) Joint optimization of service function placement and flow distribution for service function chaining. *IEEE J Sel Areas Commun* 35(11):2532–2541
- Kong J, Kim I, Wang X, Zhang Q, Cankaya HC, Xie W, Ikeuchi T, Jue JP (2017) Guaranteed-availability network function virtualization with network protection and VNF replication. In: *IEEE global communications conference*, pp 1–6
- Lecture Notes. <https://cse.buffalo.edu/~hungngo/classes/2006/594/notes/Primal-Dual.pdf>
- Li X, Tomagard A, Barton PI (2011) Nonconvex generalized benders decomposition for stochastic separable mixed-integer nonlinear programs. *J Optim Theory Appl* 151(3):425
- McCool JI (2003) Probability and statistics with reliability, queuing and computer science applications. Taylor & Francis, London
- Sun C, Bi J, Zheng Z, Yu H, Hu H (2017) NFP: enabling network function parallelism in NFV. In *ACM SIGCOMM*, pp 43–56
- Vishwanath, KV, Nagappan N (2010) Characterizing cloud computing hardware reliability. In *Proc. of the 1st ACM symposium on Cloud computing*, pp 193–204
- Wikipedia Contributors (2019) Local search (optimization)—Wikipedia, the free encyclopedia. [https://en.wikipedia.org/w/index.php?title=Local_search_\(optimization\)&oldid=928471895](https://en.wikipedia.org/w/index.php?title=Local_search_(optimization)&oldid=928471895). Accessed 3 Jan 2020
- Wu B, Yeung KL, Xu S (2007) ILP formulation for p-cycle construction based on flow conservation. In: *IEEE global telecommunications conference*, pp. 2310–2314

- Yang S, Li F, Trajanovski S, Chen X, Wang Y, Fu X (2021a) Delay-aware virtual network function placement and routing in edge clouds. *IEEE Trans Mobile Comput* 20(2):445–459
- Yang S, Li F, Trajanovski S, Yahyapour R, Fu X (2021b) Recent advances of resource allocation in network function virtualization. *IEEE Trans Parallel Distrib Syst* 32(2):295–314
- Yang S, Li F, Yahyapour R, Fu X (2022) Delay-sensitive and availability-aware virtual network function scheduling for NFV. *IEEE Trans Serv Comput* 15(1):188–201
- Zhu K, Mukherjee B (2002) Traffic grooming in an optical WDM mesh network. *IEEE J Sel Areas Commun* 20(1):122–133

Chapter 3

Delay-Aware Virtual Network Function Placement and Routing in Edge Clouds



As noted before, resource allocation problem in NFV can be formulated as an Integer Nonlinear Programming (INLP). And the most frequent approaches to deal with NFV resource allocation include combinatorial optimization theory (e.g., randomized/LP rounding, primal-dual approximation), Deep Reinforcement Learning, Game theory, etc. In this chapter, we study the problem of how to place VNFs on edge and public clouds and route the traffic among adjacent VNF pairs, such that the maximum link load ratio is minimized and each user's requested delay is satisfied.¹ We consider this problem for both totally ordered SFCs and partially ordered SFCs. We prove that this problem is NP-hard, even for the special case when only one VNF is requested. We subsequently propose an efficient randomized rounding approximation algorithm to solve this problem. Extensive simulation results show that the proposed approximation algorithm can achieve close-to-optimal performance in terms of acceptance ratio and maximum link load ratio.

3.1 Introduction

Cloud computing (Mell and Grance 2010) is a distributed computing and storage paradigm, which can provide virtually unlimited scalable service over the Internet for on-demand data-intensive applications. Data centers are usually the cloud computing-enabled infrastructures, and are located on some core router nodes in backbone networks. However, delay-sensitive applications for remote end users suffer from the long-distance network transmission delay, which remains a crucial drawback for cloud computing. The concept of Mobile Edge Computing (MEC) (Hu et al. 2015) has been proposed to bring the computing resources closer to end

¹ Parts of this chapter is reprinted from Yang et al. (2021), with permission from IEEE.

users by installing small resource-limited cloud infrastructures at the network edge (also called edge clouds), so as to provide delay-guaranteed services to end users.

Moreover, in the traditional network services provisioning paradigm, network functions (e.g., firewall or load balancer) which are also called middleboxes are usually implemented by the dedicated hardware appliances. Deploying hardware middleboxes is costly due to their high cost for design and production and also these middleboxes need to be configured and managed manually, which further increases the costs of service providers. Network Function Virtualization (NFV) which is first proposed by the European Telecommunications Standards Institute (ETSI) (ETSI; Chiosi et al. 2012) has been emerged as an appealing solution, since it enables to replace dedicated hardware implementations with software instances running in a virtualized environment. In NFV, the requested service is implemented by a sequence of Virtual Network Functions (VNF) that can run on generic servers by leveraging the virtualization technology. Service Function Chaining (SFC) is therefore defined as a chain-ordered set of placed VNFs that handles the traffic of the delivery and control of a specific application (Medhat et al. 2017). NFV allows to allocate network resources in a more scalable and elastic manner, offer a more efficient and agile management and operation mechanism for network functions and hence can largely reduce the overall costs.

Applying NFV to MEC will not only shorten servicing delay for end users but service providers will also benefit from lower expenditures and higher efficiency. The user requested service, in this context, consists of a sequence of VNFs that need to be placed on edge or public clouds. Considering that the edge cloud has limited capacity, it is also suggested (Xu et al. 2016; Ceselli et al. 2017; Lyu et al. 2018; Yang et al. 2018) to connect the edge clouds together in order to expose services to more nearby end users by efficiently utilizing and sharing the capacity and load of edge clouds. Even this, the connected edge clouds may sometimes fail to satisfy all the end users when users' requested resources increase. Hence, the edge clouds also need to collaborate with remote public cloud with sufficient capacity by placing requested VNFs on it for the users who ask for delay-tolerant services via e.g., long distance core networks, which can be shown in Fig. 3.1. This deals with where to place the VNFs and how to route the traffic by selecting appropriate paths during the whole SFC according to the user's delay requirement. In this chapter, we study how to place each user's requested VNFs on edge and public clouds, and route the packets among these VNFs in order to minimize the maximum link load ratio and meet each user's delay requirement. Our key contributions are as follows:

- We analyze the traversing delay calculation in edge clouds for both totally ordered SFC and partially ordered SFC.
- We define the Delay-aware VNF Placement and Routing (DVPR) problem in edge clouds and prove it is NP-hard.
- We propose a randomized rounding approximation algorithm to solve the DVPR problem.
- We conduct extensive simulations to validate the performances of the proposed algorithm with three heuristics.

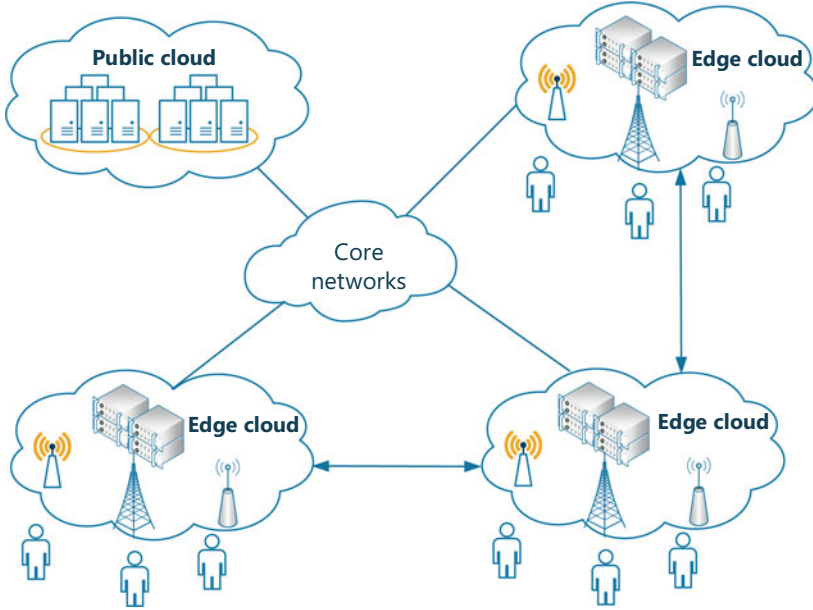


Fig. 3.1 A Mobile Edge Computing framework

The remainder of this chapter is organized as follows. Section 3.2 presents the related work. Section 3.3 models traversing delay for both totally ordered SFC and partially ordered SFC in edge clouds. Section 3.4 defines the delay-aware VNF placement and routing problem in edge clouds, formulates it as an exact INLP and proves it is NP-hard. In Sect. 3.5, we propose an approximation algorithm to solve this problem. Section 3.6 provides the simulation results and we conclude in Sect. 3.7.

3.2 Related Work

A survey about computation offloading and computation resources allocation in MEC can be found in (Mach and Becvar 2017). Mao et al. (2017) provide an overview about the communication models and resource management in MEC such as MEC server scheduling and cooperative computing. Moreover, a comprehensive survey about NFV can be found in (Mijumbi et al. 2016; Bhamare et al. 2016; Yi et al. 2018). Herrera and Botero (2016) provide a survey about resource allocation in NFV.

3.2.1 *Traffic/Cost-Aware VNF Placement and Routing in Generic Networks*

Mehraghdam et al. (2014) first explores the VNF placement problem by presenting an ILP to solve this problem. Eramo et al. (2017) target the VNF placement and routing problem with the goal of maximizing the amount of data that can be processed within the network at peak traffic time interval. They subsequently study how to consolidate VNFs and shut down unused servers when traffic demands decrease such that the total operation cost (energy consumption+revenue loss) is minimized. They propose an exact Integer Linear Programming (ILP) and an efficient heuristic to solve the problem. Cohen et al. (2015) propose a near-optimal approximation algorithm to place VNFs without considering network function dependency. Kuo et al. (2016) relax/approximate the VNF placement and routing problem based on the intuition that placing VNFs on a shorter path consumes less link bandwidth, but might also reduce VM reuse opportunities; reusing more VMs might lead to a longer path, and so it consumes more link bandwidth. Guo et al. (2018) jointly consider the VNF placement and routing problem in data centers. They propose a randomized approximation algorithm when the traffic matrix is known in advance and a competitive online algorithm when the future arriving traffic is not known. However, they assume that one configuration in data centers consists of one VNF placement and one routing path solution, and a (limited) set of configurations is given. The delay is not taken into account in these work. Gupta et al. (2018a) develop a column-generation-based approaches to solve the VNF placement and routing problem. Liu et al. (2017) present a column-generation-based approach to solve the SFC readjustment problem. Bhamare et al. (2017) deal with VNF placement in a multi-cloud scenario with constraints of deployment costs and servicing delay by proposing an ILP and an affinity-based approach heuristic. A Minimum-Residue heuristic is also presented in (Bhamare et al. 2015) for VNF placement in multi-cloud scenario. However, the path selection problem among VNF pairs are not considered in (Bhamare et al. 2015, 2017).

It is worthwhile to mention that the Virtual Machine (VM) placement problem is similar to the VNF placement. However, the communication/routing path between each VM pair do not necessarily form a SFC, which makes it different from the VNF placement problem. Please see these works in e.g. (Goudarzi and Pedram 2011; Su et al. 2015; Pires and Barán 2013; Mills et al. 2011).

3.2.2 *Delay-Aware VNF Placement and Routing in Generic Networks*

Qu et al. (2016) consider the VNF transmission and processing delays, and formulate the joint problem of VNF placement and routing as a Mixed Integer Linear Program (MILP). However, they assume that the virtual link between two

physical nodes can at most process one traffic flow at one time. Alameddine et al. (2017) jointly consider the network function mapping, traffic routing and the deadline-aware service scheduling problem, where it is assumed that the requested VNFs have already been placed on network nodes. They propose a tabu search-based algorithm and two straightforward heuristics to solve this problem. Zhang et al. (2017) devise an Alternating Direction Method of Multipliers (ADMM)-based algorithm to solve the delay-aware VNF placement and routing problem without considering the nodes' processing delay in their problem. Li et al. (2017) address the delay-aware middlebox placement and routing problem by leveraging a packet queuing model. Sun et al. (2017) implement a framework that enables (independent) network functions to work in parallel, which improves NFV performance in terms of delay, but this comes at the expense of increasing network resource (e.g., network bandwidth). By duplicating an original graph with m connected copy graphs, Huin et al. (2018) present a mathematical formulation with the aid of column generation (using a limited number of configurations) that can scale well with problem inputs (e.g., number of requests or nodes). Allybokus et al. (2018) propose an exact ILP and a greedy heuristic to solve the VNF placement and routing problem for both fully and partially ordered SFCs. However, their solutions only consider a simple path within a SFC. Dwaraki and Wolf (2016) devise a layered-graph based heuristic to solve the delay-aware traffic routing problem when the VNFs are assumed to be placed on network nodes. Similarly, Pei et al. (2018) devise a layered-graph based heuristic to jointly solve the delay-aware VNF placement and routing problem. In (Yang et al. 2022), we present an exact INLP and a recursive heuristic to solve the delay-aware and availability-aware VNF placement and routing problem in generic networks.

3.2.3 *Delay-Aware VNF/Rule Placement and Routing in Edge Clouds and Software Defined Networks (SDN)*

Hou et al. (2016) suppose that the edge servers can host limited κ number of services. When the requested service does not exist in the edge servers, downloading this service from cloud incurs a cost of ω , otherwise the provisioning cost is negligible. Without any prior knowledge of future traffic, they propose an online algorithm to configure κ services on edge servers. Ma et al. (2017b) model the edge cloud network as a queuing network, and they formulate the system servicing delay problem as a convex optimization problem. They subsequently present an algorithm with the linear complexity to solve the proposed convex optimization problem. Moreover, Saha et al. (2018) target the QoS-aware (such as delay and packet-loss) routing problem in software defined IoT networks by considering the SDN rule capacity constraint. They present an exact ILP as well as a Yen's k-shortest path-based heuristic. Bera et al. (2019) present an adaptive flow-rule placement scheme which consists of three phases, namely (1) forwarding path selection, (2) flow-rule

placement and (3) rule redistribution. To solve them, they respectively formulate (1) via a max-flow-min-cost problem, devise two greedy heuristic approaches for (2), and propose a rule redistribution scheme to detect rule congestion at a switch.

Cziva et al. (2018) tackle the VNF placement problem at the network edge in order to minimize the total expected latency from all users to their respective VNFs. They further employ the Optimal Stopping Theory to determine when to re-evaluate the optimal placement problem by taking into account the migration cost and random path delay. However, they assume that only one VNF is enough to provide a service to one user instead of a sequence of VNFs. Yang et al. (2018) study how to place NFV-enabled service on a minimum number of edge nodes and find routes from the access point to the requested service located node in order to meet the delay requirement. They propose a heuristic-based incremental allocation mechanism to solve this problem. Different from the work in (Cziva et al. 2018; Yang et al. 2018), we propose a general model to quantitatively calculate flow traversing delay of a SFC in edge clouds, and devise a randomized rounding algorithm to jointly solve the delay-aware VNF placement and traffic routing problem in edge clouds. Moreover, there are also some studies about VNF placement in virtual evolved packet core (4G/5G) networks (Dietrich et al. 2017; Gupta et al. 2018b), Cloud-Radio Access Networks (Bhamare et al. 2018), etc.

In the previous works, either an exact ILP algorithm or an efficient heuristic is proposed to solve the general VNF placement and routing problem. However, the exact ILP solution has exponential running time which means that it cannot scale well especially when the problem size increases. The heuristics are shown to perform well in the simulations under certain parameter settings, but it has no performance guarantee, meaning that it may not always output a feasible solution even if the optimal solution exists. Our contribution is devising a randomized rounding approximation algorithm to (jointly) solve the delay-aware VNF placement and routing problem in edge clouds. Moreover, our work can also be adjusted to solve the delay-aware VNF placement and routing problem in generic networks.

3.3 Network Delay Model

3.3.1 Service Function Chaining

We consider traversing delays under two cases for VNFs in a SFC, namely, (1) Totally Ordered SFC: there is a total dependency order on the VNF set, and (2) Partially Ordered SFC: there exists dependency among a subset of VNFs. That is, there exists at least two VNFs, such that they have the same predecessor function and

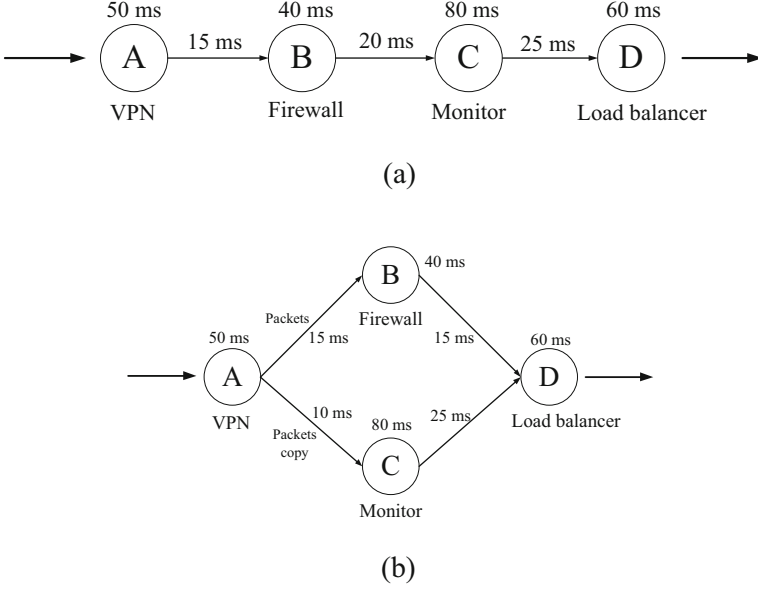


Fig. 3.2 Totally Ordered SFC v.s. Partially Ordered SFC, derived from Sun et al. (2017). (a) Totally ordered SFC. (b) Partially ordered SFC

successor function but they have no dependencies with each other. The traversing delay in the totally ordered SFC is calculated as follows:

$$\sum_{f \in F, n \in N} \Psi_n^f + \sum_{l \in p_k} T(l) \quad (3.1)$$

where Ψ_n^f represents the processing delay for function f on node n , p_k denotes the path that traverses each placed VNF, and $T(l)$ indicates the flow delivering delay on link l . For example, Fig. 3.2a is a totally ordered SFC and the traversing delay in this chain is $\underbrace{(50 + 40 + 80 + 60)}_{\text{total node delay}} + \underbrace{(15 + 20 + 25)}_{\text{total link delay}} = 290$ ms.

However, since function monitor only maintains the packets and does not change the packets, firewall and monitor can work in parallel as shown in Fig. 3.2b. After that, firewall and monitor functions send their individual processed packets to load balancer. As a result, load balancer only needs to select the packets from firewall and process them. The example in Fig. 3.2b belongs to the partially ordered SFC. We partition the partially ordered SFC into several segments and thus there is one or more independent VNFs in each segment. For example, there are 3 segments in Fig. 3.2b: Segment 1 contains the function VPN, segment 2 is composed of functions firewall and monitor, and segment 3 has the function load balancer. As a result, the traversing delay in a partially ordered SFC is equal to the longest path delay from the

node which hosts one of the VNFs in the first segment to the node which contains one of the VNFs in the last segment. For example, the traversing delay of SFC in Fig. 3.2b is equal to $\underbrace{(50 + 80 + 60)}_{\text{total node delay}} + \underbrace{(10 + 25)}_{\text{total link delay}} = 225 \text{ ms}$, which is less than

the one in Fig. 3.2a. This example also illustrates that the partially ordered SFC can reduce delay compared to the totally ordered SFC. We notice that the totally ordered SFC is a special case of the partially ordered SFC, since when each segment contains only one VNF, the partially ordered SFC becomes the totally ordered SFC. Without loss of generality, we assume a (totally ordered or partially ordered) SFC consists of a set of $|M|$ segments M , and each segment $m \in M$ has $|m|$ functions. We use m_i and m_j to represent two consecutive segments, where $j = i + 1$ and $1 \leq i \leq |M| - 1$. In order to calculate the traversing delay in a partially ordered SFC with segments M , we first make the definition of *totally ordered sub-SFC* as follows:

Definition 3.1 (Totally Ordered Sub-SFC) Suppose there is a partially ordered SFC C with segment set M . A totally ordered sub-SFC is a chaining consisting of exactly one VNF in each segment.

According to the above definition, there are in total $\prod_{i=1}^{|M|} |m_i|$ totally ordered sub-SFCs.

For example in Fig. 3.2b, there are $1 \cdot 2 \cdot 1 = 2$ totally ordered sub-SFCs, namely $A \rightarrow B \rightarrow D$ and $A \rightarrow C \rightarrow D$. As a result, the delay value of a partially ordered SFC is the maximum delay value among all the totally ordered sub-SFCs composing it.

3.3.2 Traversing Delay in a SFC in Edge Clouds

Assume that there is a set of edge clouds denoted by \mathcal{E} and one remote public cloud² represented by C . Each edge cloud $E \in \mathcal{E}$ builds a mount of edge servers along with the base station in an area and serve the end users within the coverage of the base station. The users within the coverage of E is denoted by R_E . In this sense, requests sent from R_E should be first received by the edge servers in E , and if E cannot process these requests because of capacity limit, it can relay these requests to other edge clouds or public cloud. In this chapter, we assume that any end user can only be within the coverage of one edge cloud. For simplicity, we assume that one end user sends only one request to the edge cloud. The notations used in this chapter are summarized in Table 3.1.

Suppose there is a user u_r who is within the coverage of edge cloud E , and for simplicity u_r requests a fully ordered SFC consisting of h virtual network functions f_1, f_2, \dots, f_h , where f_i is placed on node n_{f_i} for $1 \leq i \leq h$. If $T(n_i, n_j)$ denotes

² For simplicity, we only assume one public cloud in this chapter, but our work can also be extended to the case of multiple public clouds.

Table 3.1 Notations

Notation	Description
$\mathcal{E}, \mathcal{S}, \mathcal{C}$,	The set of edge clouds, switch nodes and public cloud
\mathcal{N}	The set of N nodes that can host VNFs, i.e., $\mathcal{N} = \mathcal{E} \cup \mathcal{C}$
$\mathcal{L}, b(l), t(l)$	The set of L links, traffic load and delay of link $l \in \mathcal{L}$
$\pi(n), c(l)$	Capacity of node $n \in \mathcal{N}$ and link $l \in \mathcal{L}$
$\mathcal{G}(\mathcal{N}, \mathcal{L})$	A network with set of nodes and links \mathcal{N} and \mathcal{L}
$P^{u,v}, K$	Path set between u and v , the number of paths in the set
$T(p_k)$	Delay of path p_k
R	The set of requests. For each $r(E, \delta, F, D) \in R$, E indicates the edge cloud the request belongs to, δ represents the data transmitting rate, F denotes a set of requested VNFs with predefined order, D means the requested delay
m_i, m_j	Two consecutive segments for r , where $j = i + 1$
f_i, f_j	The i -th and j -th requested VNF for a totally ordered (sub-)SFC, where $j = i + 1$
\mathbb{B}_r	The set of totally ordered sub-SFCs of request r
\mathbb{F}	A set of total requested VNFs
$\eta(f)$	The required processing capacity of VNF f
Ψ_n^f	Processing time for VNF f on node n
$H_{l,k}^{u,v}$	A given Boolean array indicating whether link l is traversed by path p_k between u and v
λ	A fractional variable meaning maximum link load ratio
$X_n^{r,f}$	A Boolean variable. It is 1 if r 's requested VNF f is placed on n ; and 0 otherwise
$Y_{u,v,k}^{r,i,j}$	A Boolean variable. It is 1 if r 's requested f_i and f_j are placed on node u and v , respectively, and $p_k \in P^{u,v}$ is selected; and 0 otherwise.

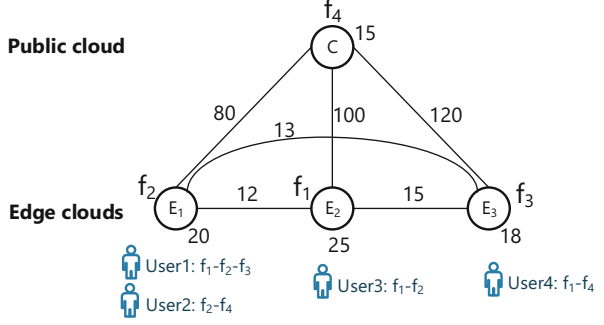
the traversing delay between edge or cloud nodes n_i and n_j , and Ψ_n^f indicates the VNF f 's processing time on node n , then the total delay for serving u_r is:³

$$T(E, n_{f_1}) + \sum_{i=1}^h \Psi_{n_{f_i}}^{f_i} + \sum_{j=1}^{h-1} T(n_{f_j}, n_{f_{j+1}}) \quad (3.2)$$

where $T(E, n_{f_1})$ indicates the traversing delay from E to n_{f_1} . $T(E, n_{f_1})$ means that if f_1 is not located in proximity to user in E , then we need to take into account the traversing delay from E to its located edge cloud or public cloud. $\sum_{i=1}^h \Psi_{n_{f_i}}^{f_i}$ calculates the total node processing time, and $\sum_{j=1}^{h-1} T(n_{f_j}, n_{f_{j+1}})$ calculates the total path delay. For example in Fig. 3.3, there are three edge clouds and one public cloud. Each edge cloud can only host one VNF because of limited capacity for

³ When an end user requests a partially ordered SFC, we can calculate the delay of each totally ordered sub-SFC according to Eq. (3.2) and take the maximum value as the final delay value.

Fig. 3.3 An example of calculating the delay in a SFC in edge clouds



simplicity, and the public cloud host two remaining VNFs. It is assumed that user 1 and user 2 belong to edge cloud E_1 . Moreover, edge cloud E_2 and edge cloud E_3 serve user 3 and user 4, respectively. The requested VNFs (SFC) are also shown in Fig. 3.3. When calculating the traversing delay of requested SFC from the request of user 1, the first requested VNF is not located in edge cloud E_1 , therefore we must calculate the delay from E_1 to E_2 (which is 12, since f_1 is placed on E_2), and then calculate the delay of SFC $f_1 - f_2 - f_3$, which is $\underbrace{(25 + 20 + 18)}_{\text{total node delay}} + \underbrace{(12 + 13)}_{\text{total link delay}} =$

88. As a result, the total delay for serving the request from user 1 is $12 + 88 = 100$. Analogously, the delay for serving user 4 is 155. On the other hand, the first requested VNF of user 2 and 3 is placed in the edge cloud they belong to, therefore serving user 2 consumes a delay of $20 + 15 + 80 = 115$, and serving user 3 incurs a delay of $25 + 20 + 12 = 57$.

3.4 Problem Definition and Complexity Analysis

3.4.1 Problem Definition

There is a set of edge clouds \mathcal{E} , a set of switch/router nodes \mathcal{S} and a remote cloud C . Each edge cloud $n \in \mathcal{E}$ consisting of a limited number of edge servers has a total capacity of $\pi(n)$, and the public cloud C is assumed to have infinite capacity. Switch nodes are used to forward the traffic. Assume there is a set \mathcal{L} of L links to connect different nodes in $\mathcal{E} \cup \mathcal{S} \cup C$. For ease of presentation, we let $\mathcal{N} = \mathcal{E} \cup C$ denote the node set that can host VNFs, and we use \mathcal{N} (which does not contain the switch node set) without loss of generality to stand for the node set in the network. Each VNF $f \in \mathbb{F}$ on node $n \in \mathcal{N}$ requires time of Ψ_n^f to process it, and we use $\eta(f)$ to denote its processing capacity. Each link $l \in \mathcal{L}$ has a capacity of $c(l)$, and traversing delay of $t(l)$. Let $b(l)$ denote the total traffic load on l , then the maximum link load ratio λ is defined as: $\max\{\frac{b(l)}{c(l)}, l \in \mathcal{L}\}$ and it is a single variable. For each node pair

(u, v) where $u, v \in \mathcal{N}$, we assume that the path set between them is known⁴ and denoted by $P^{u,v}$ with a number of K paths. R represents a set of $|R|$ requests, and for each request $r(E, \delta, F, D) \in R$, E denotes the edge cloud the request belongs to, δ stands for the data transmitted rate, $F \subset \mathbb{F}$ indicates a set of $|F|$ requested VNFs with predefined order (totally or partially ordered), and D stands for the total requested end-to-end delay. Similar to (Alameddine et al. 2017; Zhang et al. 2017; Huin et al. 2018; Pei et al. 2018; Yang et al. 2018), we assume that R is known or given. Formally, the Delay-aware VNF Placement and Routing problem in edge clouds can be defined as follows:

Definition 3.2 Given are a network $\mathcal{G}(\mathcal{N}, \mathcal{L})$, and a set of requests R . For each request $r(E, \delta, F, D) \in R$, the Delay-aware VNF Placement and Routing (DVPR) problem in edge clouds is to place its requested VNFs on \mathcal{N} and find routes among each adjacent VNF pair such that λ is minimized and the total flow traversing delay is no greater than D .

The purpose of minimizing λ is to avoid the network bottleneck, e.g., one or more links are highly loaded and while some other links are less loaded. In network bottleneck scenario, the highly loaded link(s) are unavailable to transport more data from a source to a destination because of the lack of free capacity. In the worst case, there may not exist bandwidth-free path from a specified source to a specified destination if these (critical) highly loaded links are not available, or it may take longer delay by traversing some more links from a source to a destination. Nevertheless, we mention that our problem definition/formulation and proposed approximation algorithm are general and flexible, since we could also remove the objective or change to select some other objectives. More specifically, if we remove the objective, then the problem becomes to find the VNF placement and traffic routing for each request in edge clouds without violating the specified delay requirement. And the respective approximation algorithm proposed in Sect. 3.5 can be slightly adjusted (by removing the objective and λ -related constraints) to solve it as well.

3.4.2 An Exact Formulation

In this subsection, we formulate the DVPR problem as an exact Integer Nonlinear Programming (INLP) formulation. We begin with some necessary notations and variables.

⁴ According to Leconte et al. (2018), at most 6 paths in GÉANT network are enough for serving 11460 traffic matrices during the entire 4 month duration without violating Quality of Service (QoS). We therefore assume that a (small) set of paths is sufficient for calculating the optimal solution. This set of paths is precalculated and given in the problem.

INLP Notations

- $H_{l,k}^{u,v}$: A given Boolean array. It is 1 (true) if link l is traversed by the path p_k between u and v , and 0 (false) otherwise.
- $T(p_k)$: Delay of path p_k .
- \mathbb{F} : The set of total requested $|\mathbb{F}|$ VNFs. For ease of calculation of traversing delay, for each request r , we add two dummy VNFs f_0 as the first VNF of the chain and $f_{|\mathbb{F}|+1}$ as the last VNF of the chain, with $\eta(f_0) = \eta(f_{|\mathbb{F}|+1}) = 0$ and $\Psi_n^{f_0} = \Psi_n^{f_{|\mathbb{F}|+1}} = 0$, where $n \in \mathcal{N}$.
- \mathbb{B}_r : The set of totally ordered sub-SFCs of request r .
- f_i, f_j : The i -th and j -th requested VNF for r , where $0 \leq i \leq |\mathbb{F}|$, $j = i + 1$ (two consecutive VNFs).

INLP Variables

- λ : A float variable ranging in $[0, 1]$ that represents the link load factor or ratio.
- $X_n^{r,f}$: A Boolean variable. It is 1 (true) if r 's requested VNF f is placed on n ; and 0 (false) otherwise.
- $Y_{u,v,k}^{r,i,j}$: A Boolean variable. It is 1 (true) if r 's requested f_i and f_j are placed on node u and v , respectively, and $p_k \in P^{u,v}$ is selected; and 0 (false) otherwise.

Objective

$$\min \lambda \quad (3.3)$$

Placement Constraints

$$\sum_{n \in \mathcal{N}} X_n^{r,f} = 1 \quad \forall r \in R : f \in r.F \quad (3.4)$$

$$\sum_{u,v} \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} = 1 \quad \forall r \in R, B \in \mathbb{B}_r, f_i, f_j \in B \quad (3.5)$$

Path Selection Constraint

$$X_u^{r,f_i} \cdot X_v^{r,f_j} = \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \quad \forall r \in R, B \in \mathbb{B}_r, \\ f_i, f_j \in B, u, v \in \mathcal{N} \quad (3.6)$$

Delay Constraint

$$\begin{aligned} & \sum_{f_i, f_j \in B} \sum_{u, v \in N} (X_u^{r, f_i} \cdot X_v^{r, f_j} \cdot \sum_{p_k \in P^{u, v}} Y_{u, v, k}^{r, i, j} \cdot (T(p_k) + \frac{\Phi_u^{f_i} + \Phi_v^{f_j}}{2})) \\ & \leq D \quad \forall r \in R, B \in \mathbb{B}_r \end{aligned} \quad (3.7)$$

Edge Cloud Capacity Constraint

$$\sum_{f \in \mathbb{F}} \max_{r \in R: f \in r.F} X_n^{r, f} \cdot \eta(f) \leq \pi(n) \quad \forall n \in \mathcal{N} \setminus C \quad (3.8)$$

Link Capacity Constraint

$$\begin{aligned} & \sum_{r \in R} \sum_{B \in \mathbb{B}_r} \sum_{f_i, f_j \in B} \sum_{u, v \in N} \sum_{p_k \in P^{u, v}} Y_{u, v, k}^{r, i, j} \cdot H_{l, k}^{u, v} \cdot r \cdot \delta \leq \lambda \cdot c(l) \\ & \forall l \in \mathcal{L} \end{aligned} \quad (3.9)$$

Equation (3.3) minimizes the maximum link load ratio. Equation (3.4) ensures that for each user's requested VNF f , it must be placed on one node in the network. Equation (3.5) ensures that for each requested VNF pair f_i, f_j in a totally ordered sub-SFC, they must be placed on one or two nodes. The above two placement constraints are set for variables $X_n^{r, f}$ and $Y_{u, v, k}^{r, i, j}$, respectively, and Eq. (3.6) establishes the equality relation between $X_n^{r, f}$ and $Y_{u, v, k}^{r, i, j}$. More specifically, Eq. (3.6) indicates that when request r places f_i and f_j on u and v , respectively, only one path between u and v can be selected to use. Equation (3.7) ensures that for each request the total traversing delay in each sub-ordered SFC is no greater than D . In particular, $\sum_{f_i, f_j} (\Phi_u^{f_i} + \Phi_v^{f_j})/2$ calculates the total node processing delay. This is because f_1, f_2, \dots have been counted twice in Eq. (3.7), we take the sum of $\Phi_u^{f_i} + \Phi_v^{f_j}$ and let it be divided by 2. Equation (3.8) ensures that each edge cloud's capacity is not violated. Equation (3.9) ensures that each link's load does not exceed to λ times its total capacity.

3.4.3 Complexity Analysis

Theorem 3.1 *The DVPR problem is NP-hard, even when $|\mathbb{F}| = 1$ and $|R| \geq 2$.*

Proof Since totally ordered SFC is a special case of partially ordered SFC, we only analyze the problem complexity for the totally ordered SFC without loss of generality. Ma et al. (2017a) prove that the general VNF placement and routing problem can be reduced to the NP-hard Hamiltonian cycle problem (Garey and

Johnson 1979), which means that the problem is also NP-hard. Next, we will prove that even when $|\mathbb{F}| = 1$ and $|R| \geq 2$, the DVPR problem is still NP-hard.

Assuming that there are two edge clouds, E_1 and E_2 . Two user requests r_1 and r_2 which are both in proximity of E_1 ask for VNF f_1 and f_2 , respectively. It is assumed that E_1 has no spare capacity and therefore f_1 and f_2 have to be placed on E_2 . Moreover, we assume $c(l) = r_1 \cdot \delta = r_2 \cdot \delta, \forall l \in \mathcal{L}$. In this sense, each link can only be traversed by at most one time and $\lambda = 1$ in the optimal solution. Hence, the DVPR problem is to find two link-disjoint paths from E_1 to E_2 such that each path delay is no greater than the specified delay value. Now, the DVPR problem is equivalent to the NP-hard min-max problem (Li et al. 1990), which is to find two link-disjoint paths from a source to a destination, such that the longer path delay (denoted by T_x) is minimized, if we assume $\max(r_1 \cdot D, r_2 \cdot D) = T_x$. The proof is therefore complete.

3.5 Approximation Algorithm

Considering that the exact INLP solution has exponential running time (Hemmecke et al. 2010) because of its Boolean variables and nonlinear constraints, it cannot scale well especially when the problem size increases. In this section we propose a polynomial-time approximation algorithm to solve the DVPR problem. We first transform the exact INLP in Eqs. (3.3)–(3.9) to a Linear Programming (LP). After solving the LP, we subsequently achieve a solution via the randomized rounding method with proved bounded approximation ratio.

3.5.1 Transformation from the INLP to the LP

First, using Eq. (3.6) into Eq. (3.7) we get:

$$\begin{aligned} & \sum_{f_i, f_j \in B} \sum_{u, v \in N} \left(\sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \cdot \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \cdot (T(p_k) + \frac{\Phi_u^{f_i} + \Phi_v^{f_j}}{2}) \right) \\ & \leq D \quad \forall r \in R, B \in \mathbb{B}_r \end{aligned} \quad (3.10)$$

It is important to notice that $Y_{u,v,k}^{r,i,j}$ takes binary values in $\{0, 1\}$ and if we decompose the two innermost sum into products we get pairwise products in $Y_{u,v,k}^{r,i,j}$. However, according to the placement constraint (Eq. (3.5)) there is a single $Y_{u,v,k}^{r,i,j}$ equals to 1 and all the rest are 0 for a fixed r . Hence, all the pairwise products of $Y_{u,v,k}^{r,i,j}$ in

Eq. (3.10) are 0, except for one that is $(Y_{u,v,k}^{r,i,j})^2 = Y_{u,v,k}^{r,i,j}$. Finally, Eq. (3.10) boils down to the following equivalent constraint of Eq. (3.7):

$$\sum_{f_i, f_j \in B} \sum_{u, v \in \mathcal{N}} \left(\sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \cdot (T(p_k) + \frac{\Phi_u^{f_i} + \Phi_v^{f_j}}{2}) \right) \leq D$$

$$\forall r \in R, B \in \mathbb{B}_r \quad (3.11)$$

Second, considering that Eq. (3.6) is nonlinear, we transform it to the following equivalent linear constraints without losing any accuracy:

$$\begin{cases} X_u^{r,f_i} \geq \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \\ X_v^{r,f_j} \geq \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \\ X_u^{r,f_i} + X_v^{r,f_j} - 1 \leq \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \\ \forall r \in R, B \in \mathbb{B}_r, f_i, f_j \in B, u, v \in \mathcal{N} \end{cases} \quad (3.12)$$

After the transformation, we have the following relaxed Linear Programming (LP) to solve the DVPR problem, where $Y_{u,v,k}^{r,i,j}$ and $X_n^{r,f}$ are set to be a fractional number ($\in [0, 1]$).

$$\begin{aligned} & \min \quad \lambda \\ & s.t. \quad \begin{cases} \sum_{n \in \mathcal{N}} X_n^{r,f} = 1 \quad \forall r \in R, f \in r.F \\ \sum_{u,v} \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} = 1 \quad \forall r \in R, B \in \mathbb{B}_r, f_i, f_j \in B \\ X_u^{r,f_i} \geq \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \\ X_v^{r,f_j} \geq \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \\ X_u^{r,f_i} + X_v^{r,f_j} - 1 \leq \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \\ \forall r \in R, B \in \mathbb{B}_r, f_i, f_j \in B, u, v \in \mathcal{N} \\ \sum_{f_i, f_j \in B} \sum_{u, v \in \mathcal{N}} \left(\sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \cdot (T(p_k) + \frac{\Phi_u^{f_i} + \Phi_v^{f_j}}{2}) \right) \\ \leq D \quad \forall r \in R, B \in \mathbb{B}_r \\ \sum_{f \in \mathbb{F}} \max_{r \in R: f \in r.f} X_n^{r,f} \cdot \eta(f) \leq \pi(n) \quad \forall n \in \mathcal{M} \setminus C \\ \sum_{r \in R} \sum_{B \in \mathbb{B}_r} \sum_{f_i, f_j \in B} \sum_{u, v \in \mathcal{N}} \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j} \cdot H_{l,k}^{u,v} \cdot r.\delta \\ \leq \lambda \cdot c(l) \quad \forall l \in \mathcal{L} \end{cases} \end{aligned} \quad (3.13)$$

3.5.2 Randomized Rounding Approximation Algorithm

In this subsection, we present the Randomized Rounding VNF placement and routing Algorithm (RRVA) in Algorithm 1 to solve the DVPR problem.

Algorithm 1: RRVA ($\mathcal{G}(\mathcal{N}, \mathcal{L}), R, \mathcal{F}$)

```

1  Solve the LP in Eq. (3.13) to obtain  $\widetilde{Y}_{u,v,k}^{r,i,j}$ 
2   $Ht[R][F] \leftarrow null, Rt[R][F][F] \leftarrow 0, At[R] \leftarrow true$ 
3  foreach request  $r \in R$  do
4      foreach  $B \in B_r$  do
5           $n_x \leftarrow r.E$ 
6          foreach  $f_i, f_j \in B$  do
7              if  $Ht[r][f_i] \neq null \ \&\& \ Ht[r][f_j] \neq null$  then
8                   $n_x \leftarrow Ht[r][f_j]$ 
9              else
10                  $P \leftarrow \emptyset, Q \leftarrow \emptyset$ 
11                 if  $\widetilde{Y}_{u,v,k}^{r,i,j} > 0 \ \&\& \ u == n_x$  then
12                     if  $u == v$  then
13                          $P.Add(\widetilde{Y}_{u,v,k}^{r,i,j})$ 
14                     else
15                          $Q.Add(\widetilde{Y}_{u,v,k}^{r,i,j})$ 
16                 if  $P \neq \emptyset$  then
17                     Pick  $v$  with maximum node capacity from  $P$  such that  $\widetilde{Y}_{u,v,k}^{r,i,j} > 0$  and
18                      $u == Ht[r][f_i]$  if  $Ht[r][f_i] \neq null$ 
19                      $\pi(v) \leftarrow \pi(v) - \eta(f_j), Ht[r][f_i] \leftarrow u, Ht[r][f_j] \leftarrow$ 
20                      $v, Rt[r][f_i][f_j] \leftarrow 0$ 
21                 else if  $Q \neq \emptyset$  then
22                     Randomly select  $\widetilde{Y}_{u,v,k}^{r,i,j} > 0$  from  $Q$  such that  $u == Ht[r][f_i]$  if
23                      $Ht[r][f_i] \neq null$ 
24                      $\pi(v) \leftarrow \pi(v) - \eta(f_j), n_x \leftarrow v, Ht[r][f_i] \leftarrow u, Ht[r][f_j] \leftarrow$ 
25                      $v, Rt[r][f_i][f_j] \leftarrow p_k$ 
26                     Prune the link bandwidth according to the selected path  $p_k$  based on
27                      $\widetilde{Y}_{u,v,k}^{r,i,j}$ 

```

The rational of RRVA in Algorithm 1 is that after solving LP in Eq. (3.13), we first get fractional solutions $\widetilde{Y}_{u,v,k}^{r,i,j}$ (and $\widetilde{X}_n^{r,f}$). Our aim is to derive an integer solution from $\widetilde{Y}_{u,v,k}^{r,i,j} > 0$ by randomized rounding. To that end, RRVA first uses $Ht[R][F]$ to store which node to host the requested VNF f of r , and uses $Rt[R][F][F]$ to store which path to route traffic from r 's requested f_i to f_j . When f_i and f_j are placed on the same node, $Rt[r][f_i][f_j] = 0$. Moreover, $At[R]$

represents whether r is successfully served, and initially is set to be true for all the requests. After that, for each requested totally ordered sub-SFC $B \in B_r$ of request r , and for each its requested VNF pair $(f_i, f_j) \in B$, RRVA finds the appropriate nodes for hosting f_i and f_j and the path to route the traffic between them if they are placed on different nodes. Let n_x be the node where f_j is placed on in each iteration. In particular, when f_i and f_j have already been placed on the nodes in the previous iterations by other totally ordered sub-SFC, Step 8 assigns n_x with $Ht[r][f_j]$. Otherwise, it indicates that at least f_j is not placed on the network. When $\widetilde{Y}_{u,v,k}^{r,i,j} > 0$, RRVA uses P to store the results where f_i and f_j are placed on the same node, and uses Q to store the results where f_i and f_j are placed on different nodes. If P is not empty, RRVA selects one $\widetilde{Y}_{u,v,k}^{r,i,j}$ from P where the node u has maximum residual capacity to host f_i and f_j . However, when $Ht[r][f_i] \neq null$, it indicates that f_i has already been placed on the network by other totally ordered sub-SFC determined by RRVA, and therefore we need to select $\widetilde{Y}_{u,v,k}^{r,i,j} > 0$ from P where $u = Ht[r][f_i]$. The reason is that one VNF can only be placed on one node. If P is empty and Q is not empty, RRVA randomly selects one $\widetilde{Y}_{u,v,k}^{r,i,j}$ from Q . Similarly, when $Ht[r][f_i] \neq null$ we need to select $\widetilde{Y}_{u,v,k}^{r,i,j} > 0$ from Q where $u = Ht[r][f_i]$. Because f_i except for f_0 and $f_{|\mathbb{F}|+1}$ occurs twice in the algorithm, we need to decrease the capacity of its hosted node only once, which can be seen in steps 18 and 21. Accordingly, RRVA prunes the link bandwidth that is traversed by the selected path p_k between u and v . The above procedure continues until all the requests have been iterated.

The time complexity of RRVA can be analyzed like this: The time complexity of RRVA is dominated by the LP in Eq. (3.13). There exists efficient polynomial time algorithm to solve the LP with the current best worst-case complexity of $O(\lfloor \frac{I^3}{\ln I} \rfloor \gamma)$ by an interior-point method according to Anstreicher (1999), where I is the number of variables and γ is the bit size of the problem (related to the number of bits in its binary representation). There are in total $O(|R||F|N + K|R||F|^2N^2) = O(K|R||F|^2N^2)$ variables in Eq. (3.13), leading to a total time complexity of $O(\frac{\gamma K^3|R|^3|F|^6N^6}{\ln(K|R||F|^2N^2)})$ for RRVA, which indicates it has polynomial running time.

3.5.3 Approximation Performance Analysis

The analysis leverages on the method of Upper Tail Chernoff bound (Tarjan 2009) and Union Bound (Boole's inequality) (Comtet 1974, Chapter 4.7)—a technique often used in other works (see e.g., Xu et al. 2017; Lin et al. 2018). For

completeness, we first give a formal definition of the Upper Tail Chernoff bound and Union Bound inequality:

Theorem 3.2 *Tarjan (2009)* Let x_1, x_2, \dots, x_n be n independent random variables, and $x_i \in [0, 1]$ for $1 \leq i \leq n$. Denote $\mu = E[\sum_{i=1}^n x_i]$, then for an arbitrary positive ϵ we have:

$$\Pr \left[\sum_{i=1}^n x_i \geq (1 + \epsilon)\mu \right] \leq e^{\frac{-\epsilon^2 \mu}{2 + \epsilon}} \quad (3.14)$$

Theorem 3.3 Let A_1, A_2, \dots, A_n be n events with happening probability $\Pr[A_1], \Pr[A_2], \dots, \Pr[A_n]$, then $\Pr[A_1 \cup A_2 \cup \dots \cup A_n] \leq \sum_{i=1}^n \Pr[A_i]$.

Moreover, α is used to ensure that the following defined expected values are fractional number and defined as follows:

$$\alpha = \min \left\{ \min \left\{ \frac{\tilde{\lambda} \min(c(l))}{r \cdot \delta} \right\}, \min \left\{ \frac{r \cdot D}{T(p_k)} \right\}, \min \left\{ \frac{\pi(n)}{\eta(f)} \right\} \right\} \\ \forall r \in R, l \in \mathcal{L}, n \in \mathcal{N}, f \in \mathbb{F}, p_k \in P^{u,v} : u, v \in \mathcal{N} \quad (3.15)$$

where $\tilde{\lambda}$ is the lower bound of maximum link load ratio value returned by the LP in Eq. (3.13). The proposed RRVA rounds the fractional $\widetilde{Y_{u,v,k}^{r,i,j}}$ solved from LP in Eq. (3.13) to integer value and then derive the routing and placement⁵ solution. In the following, we will analyze the violating factor in terms of link capacity, requested delay and node capacity of RRVA.

3.5.3.1 Link Capacity Violating Factor

Definition 3.3 For each request r and each link l , the traffic load z_l^r is defined as follows:

$$z_l^r = \begin{cases} r \cdot \delta & \text{with prob. } \sum_{r \in R} \sum_{B \in \mathbb{B}_r} \sum_{f_i, f_j \in B} \sum_{u, v \in \mathcal{N}} \sum_{p_k \in P^{u,v}} \widetilde{Y_{u,v,k}^{r,i,j}} \cdot H_{l,k}^{u,v} \\ 0 & \text{otherwise} \end{cases}$$

⁵ According to Eq. (3.13), $X_u^{r,f_i} \geq \sum_{p_k \in P^{u,v}} Y_{u,v,k}^{r,i,j}$, so the VNF placement solution achieved by RRVA can also be regarded as rounded from X_u^{r,f_i} .

Since $z_l^{r_1}, z_l^{r_2}, \dots$ are mutually independent according to their definition, the expected load on link l is:

$$\begin{aligned} E \left[\sum_{r \in R} z_l^r \right] &= \sum_{r \in R} E[z_l^r] = \sum_{r \in R} r \cdot \delta \cdot \sum_{f_i, f_j} \sum_{u, v \in N} \widetilde{Y_{u,v,k}^{r,i,j}} \cdot H_{l,k}^{u,v} \\ &\leq \widetilde{\lambda} \cdot c(l) \end{aligned} \quad (3.16)$$

According to the definition of α in Eq. (3.15), it holds that $0 \leq \frac{z_l^r \cdot \alpha}{\widetilde{\lambda} \cdot c(l)} \leq 1$. Therefore, by dividing Eq. (3.16) with $\frac{\widetilde{\lambda} \cdot c(l)}{\alpha}$ on both sides we have:

$$\mu_c = E \left[\sum_{r \in R} \frac{z_l^r \cdot \alpha}{\widetilde{\lambda} \cdot c(l)} \right] \leq \alpha \quad (3.17)$$

Since $\frac{\alpha}{\mu_c} \geq 1$, we have:

$$\Pr \left[\sum_{r \in R} \frac{z_l^r \cdot \alpha}{\widetilde{\lambda} \cdot c(l)} \geq (1 + \epsilon)\alpha \right] \leq \Pr \left[\sum_{r \in R} \frac{\alpha}{\mu_c} \frac{z_l^r \cdot \alpha}{\widetilde{\lambda} \cdot c(l)} \geq (1 + \epsilon)\alpha \right] \quad (3.18)$$

We cannot directly apply Theorem 3.2 for $\frac{z_l^r \cdot \alpha}{\widetilde{\lambda} \cdot c(l)}$ with α , however the following holds:

$$\alpha = \frac{\alpha}{\mu_c} \mu_c = \frac{\alpha}{\mu_c} E \left[\sum_{r \in R} \frac{z_l^r \cdot \alpha}{\widetilde{\lambda} \cdot c(l)} \right] = E \left[\sum_{r \in R} \frac{\alpha}{\mu_c} \frac{z_l^r \cdot \alpha}{\widetilde{\lambda} \cdot c(l)} \right] \quad (3.19)$$

By applying Theorem 3.2 for $\frac{\alpha}{\mu_c} \frac{z_l^r \cdot \alpha}{\widetilde{\lambda} \cdot c(l)}$, based on Eq. (3.19):

$$\Pr \left[\sum_{r \in R} \frac{\alpha}{\mu_c} \frac{z_l^r \cdot \alpha}{\widetilde{\lambda} \cdot c(l)} \geq (1 + \epsilon)\alpha \right] \leq e^{\frac{-\epsilon^2 \alpha}{2 + \epsilon}} \quad (3.20)$$

where ϵ is an arbitrary positive value. Further, using inequalities from (3.18) and (3.20) together and introducing Δ , we have:

$$\Pr \left[\sum_{r \in R} \frac{z_l^r}{\widetilde{\lambda} \cdot c(l)} \geq (1 + \epsilon) \right] \leq e^{\frac{-\epsilon^2 \alpha}{2 + \epsilon}} \leq \frac{\Delta}{N^2} \quad (3.21)$$

where Δ is a network related variables and $\Delta \rightarrow 0$ when the network size grows. By solving Eq. (3.21), we have that

$$\epsilon \geq \frac{-\log \frac{\Delta}{N} + \sqrt{\log^2 \frac{\Delta}{N^2} - 8\alpha \log \frac{\Delta}{N^2}}}{2\alpha} \quad (3.22)$$

Theorem 3.4 *RRVA can achieve a link capacity violating factor of $\frac{4 \log N}{\alpha} + 3$.*

Proof By setting $\Delta = \frac{1}{N^2}$, Eq. (3.21) becomes:

$$\Pr \left[\sum_{r \in R} \frac{z_l^r}{\tilde{\lambda} \cdot c(l)} \geq (1 + \epsilon) \right] \leq \frac{1}{N^4}, \quad \text{where } \epsilon \geq \frac{4 \log N}{\alpha} + 2. \quad (3.23)$$

By using Union Bound inequality in Theorem 3.3 for all the links:

$$\begin{aligned} \Pr \left[\bigcup_{l \in \mathcal{L}} \sum_{r \in R} \frac{z_l^r}{\tilde{\lambda} \cdot c(l)} \geq (1 + \epsilon) \right] &\leq \sum_{l \in \mathcal{L}} \Pr \left[\frac{z_l^r}{\tilde{\lambda} \cdot c(l)} \geq (1 + \epsilon) \right] \\ &\leq N^2 \cdot \frac{1}{N^4} = \frac{1}{N^2}, \quad \text{where } \epsilon \geq \frac{4 \log N}{\alpha} + 2 \end{aligned} \quad (3.24)$$

The last inequality holds since there are at most N^2 links in a network with N nodes. Finally, Eq. (3.24) indicates that the probability the expected load on any link violates $\tilde{\lambda} \cdot c(l)$ with a factor of $1 + \epsilon = \frac{4 \log N}{\alpha} + 3$ approaches 0 when $N \rightarrow +\infty$.

Theorem 3.5 *RRVA can achieve a delay violating factor of $\frac{\log |R| |B_x| + 2 \log N}{\alpha} + 3$.*

Proof For brevity, we have proved it in Appendix A of Yang et al. (2021).

Theorem 3.6 *The randomized approximation algorithm can achieve a node capacity violating factor of $\frac{3 \log N}{\alpha} + 3$.*

Proof For brevity, we have proved it in Appendix B of Yang et al. (2021).

Theorem 3.7 *The randomized approximation algorithm can achieve a link bandwidth violating factor of $\frac{4 \log N}{\alpha} + 3$, delay violating factor of $\frac{\log |R| |B_x| + 2 \log N}{\alpha} + 3$, and node capacity violating factor of $\frac{3 \log N}{\alpha} + 3$.*

Proof The proof follows from Theorems 3.4, 3.5 and 3.6.

3.6 Simulations

3.6.1 Simulation Setup

We conduct simulations⁶ on two networks: USANet, displayed in Fig. 3.4, which is a realistic carrier backbone network, and GÉANT, shown in Fig. 3.5, which is a pan-European communications infrastructure. In both networks, the solid red circled nodes are assumed to be edge clouds, whose capacities range from 15 to 20 units, and the dashed red circled nodes are assumed to be the public cloud nodes, whose capacities are infinite. For each link, its capacity is randomly picked in [3, 4] Gb/s and its delay takes value in [10, 50] ms. It is assumed that there are in total 15 VNFs whose capacities vary from 5 to 10 units, and the node processing time for each VNF varies from 50 to 150 ms (Ghaznavi et al. 2015). We randomly generate 50 sets of $|R| = 20, 40, 60, 80, 100$ requests for requested both totally ordered SFC (the request set is denoted by R^t) and partially ordered SFC (the request set denoted by R^p), separately. R^t and R^p are uniformly distributed in each edge cloud. For each request $r(E, \delta, F, D) \in R^t$, δ is between 10 and 50 Mb/s, $|F|$ is between 5 and 10, and D is between 500 and 800 ms. The traffic request setting is in line with (Savi et al. 2015), and reflects the conventional application of e.g., Web Service, VoIP, etc. The request $r(E, \delta, F, D) \in R^p$ is generated the same with the request in R^t except that we set 4 segments of VNFs in each requested partially ordered SFC (the requested VNFs in R^p are identical with the ones in R^t for each request), and therefore the number of VNFs in each segment is in [1, 3].

The solution of the LP from Eq. (3.13) provides a lower bound of the optimal solution (denoted by OPT_LB), so we compare OPT_LB and our proposed approximation algorithm RRVA with the relevant (slightly modified) SFC-eMbedding Approach (MAP) (Pei et al. 2018) and two straightforward heuristics as follows. We initialize $i = 1$.

1. SFC-MAP: For each request $r(E, \delta, F, D)$, suppose that f_0 is placed on $r.E$ (also for the Greedy and Random algorithm in below) as in segment m_0 . For each totally ordered sub-SFC m of r , SFC-MAP first constructs a multi-layer graph by creating $|F| + 1$ copies of the original graph. The inter-layer links are created to connect the same node in different layers, and the inter-layer nodes are created to represent the possible places to host each VNF in each layer. By assigning link delay weights, SFC-MAP runs shortest path from the ingress node in layer 1 to the egress node in layer $|F| + 1$. If the solution is not found or the delay requirement is violated, a penalty is imposed for each link weight. The above procedure continues at most τ times, where we set $\tau = 30$ in our simulation.

⁶ The simulations were run on a high-performance desktop PC with 3.40 GHz and 16 GB memory. We used IBM ILOG CPLEX (CPLEX Callable Library interface) 12.6 to implement OPT_LB, IBM ILOG CPLEX 12.6 together with C# to implement RRVA, and C# to implement the heuristic algorithms.

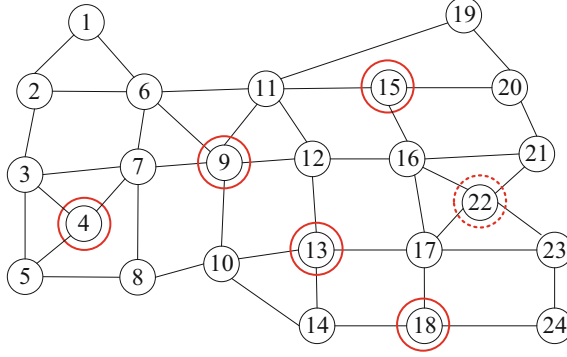


Fig. 3.4 USA carrier backbone network

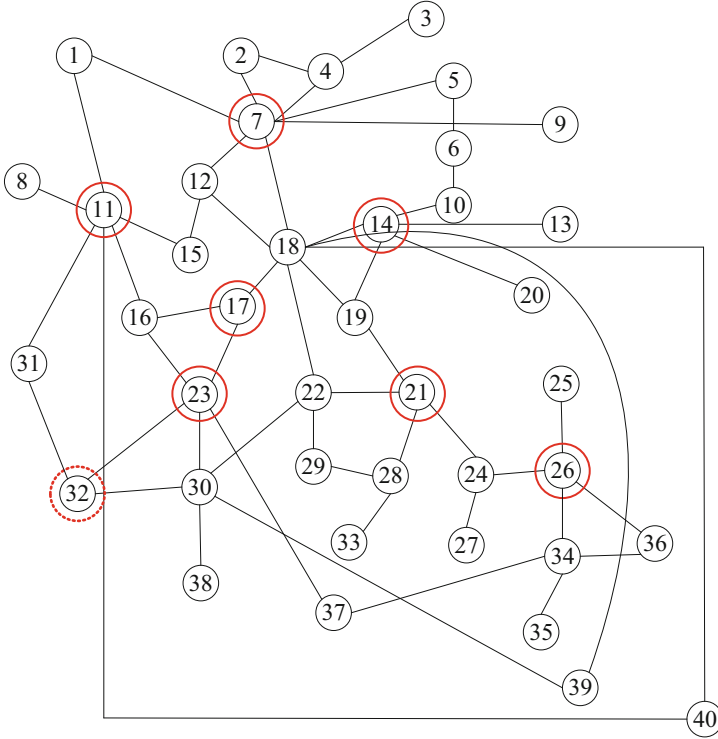


Fig. 3.5 GÉANT pan-European research network

2. Greedy Algorithm: For each request $r(E, \delta, F, D)$, suppose that f_0 is placed on $r.E$ as in segment m_0 . For each requested VNF f in segment m_i , Greedy algorithm first tries to place it on one of the nodes where the VNF(s) in segment m_{i-1} is hosted (denoted by the set N_{f_x}). If this step fails, Greedy algorithm places

f on a node n_{f_y} with sufficient capacity, such that the $\frac{\min_{l \in p_k} (c(l) - b(l))}{T(p_k)}$ value is maximum among all the nodes $E \setminus \{N_{f_x}\}$, where $\min_{l \in p_k} (c(l) - b(l))$ denotes the minimum link residual capacity in path p_k from n_{f_y} to each node in N_{f_x} . Moreover, the sum of each path's delay and node processing delay for f together with the delay of serving previous VNFs should be less than D .

3. Random Algorithm: For each requested VNF f in segment m_i , Random algorithm first tries to place it on one of the node(s) where the VNFs in segment m_{i-1} are hosted (denoted by N'_{f_x}). If this step fails, it randomly chooses one node n'_{f_y} with sufficient capacity such that there exists at least one path from each node in N'_{f_x} to n'_{f_y} whose accumulating delay is less than D and the path bandwidth is no less than δ . If succeeds, it randomly selects one satisfied path from each node in N'_{f_x} to n'_{f_y} .

For both Greedy and Random algorithms, the above procedures continue by increasing i by 1 until all the requested VNFs have been placed and associated paths have been found without violating delay, node capacity and link capacity constraints, otherwise the algorithms continue to serve the next request using the same routine until all the requests have been iterated.

3.6.2 Simulation Results

We first compare the algorithms in terms of Acceptance Ratio (AR), which is defined as the number of accepted requests divided by the total number of requests. Here, if a request is accepted, it means that its specified total flow traversing delay D is obeyed returned by the found solution. From the simulations we found that the variance of AR and maximum Link Load Ratio (LLR) for all the algorithms is small. To not clutter the figures, we only present the mean value of AR and Max. LLR in below. Moreover, since Opt_LB only returns the fractional solutions, we do not know the feasible integer solutions corresponding to different requests and thus cannot calculate its returned AR value. We therefore omit the AR values for Opt_LB. From Figs. 3.6 and 3.7 we can see that our approximation algorithm can accept almost all the requests (above around 90%) for both totally ordered SFC and partially ordered SFC. The AR values returned by Greedy algorithm and Random algorithm are much lower than the AR value achieved by RRVA. SFC-MAP has a better performance than Greedy and Random, but its achieved AR value is still lower than RRVA. For all the algorithms, their achieved AR value for the partially ordered SFC request in Fig. 3.7 is higher than the AR value for the partially ordered SFC request in Fig. 3.6. The reason is that as we stated in Sect. 3.3.1, the traversing delay in partially ordered SFC is lower than the delay in the totally ordered SFC, which in turns satisfy more requests regarding delay requirement. In particular, Fig. 3.8 illustrates the average delay value returned from all the algorithms among 100 requests. Accordingly, RRVA achieves the least average delay, followed SFC-MAP and Greedy, and the Random has the largest average delay.

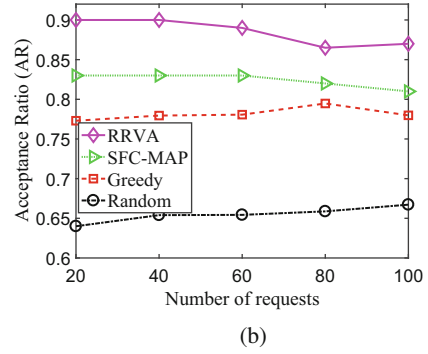
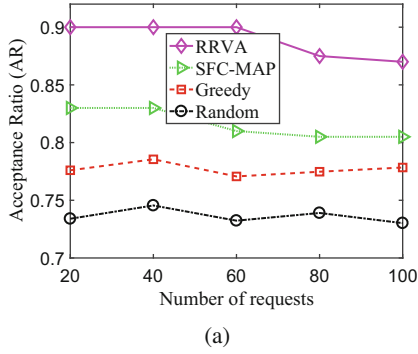


Fig. 3.6 Acceptance Ratio for the totally ordered SFC. (a) USANet. (b) GÉANT

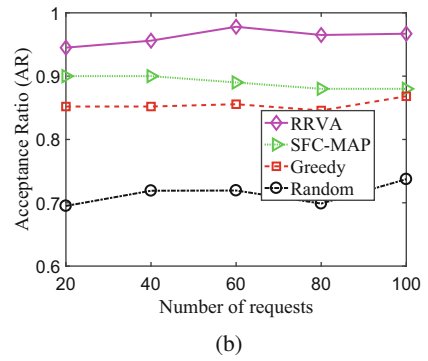
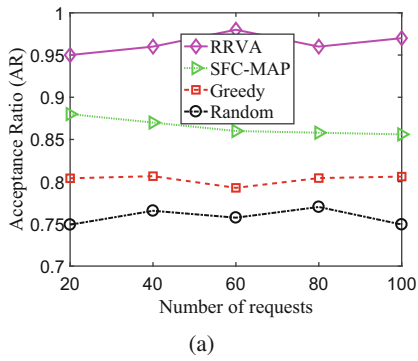


Fig. 3.7 Acceptance Ratio for the partially ordered SFC. (a) USANet. (b) GÉANT

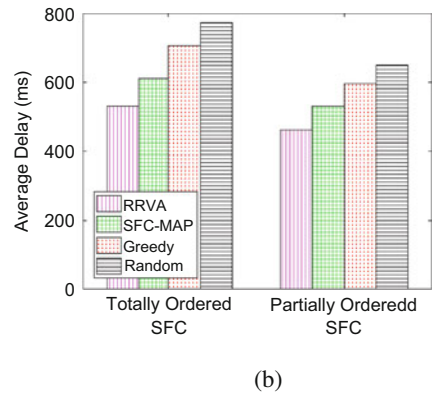
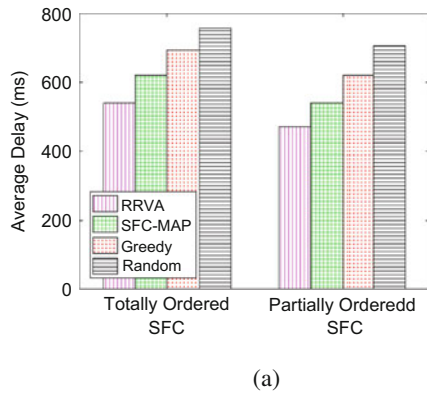


Fig. 3.8 Average Delay of $|R| = 100$ requests. (a) USANet. (b) GÉANT

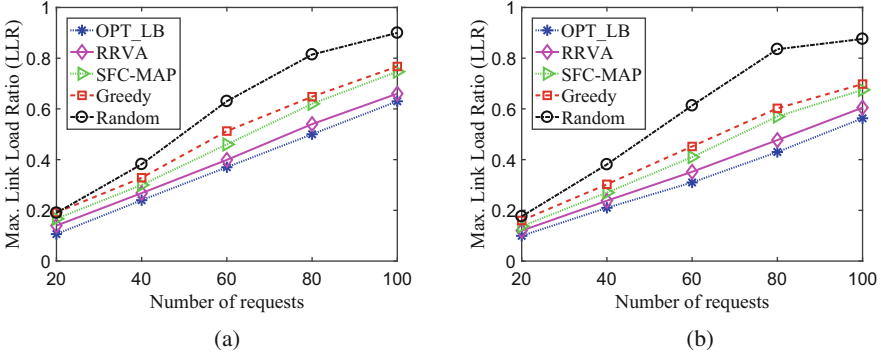


Fig. 3.9 Maximum Link Load Ratio (LLR) for the totally ordered SFC. (a) USANet. (b) GÉANT

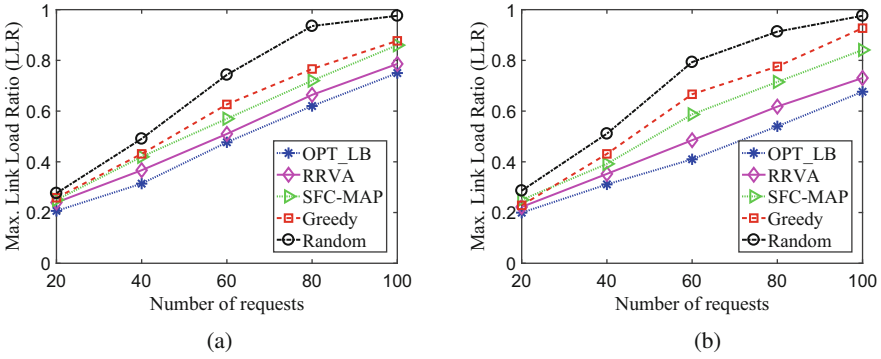


Fig. 3.10 Maximum Link Load Ratio (LLR) for the partially ordered SFC. (a) USANet. (b) GÉANT

Next, we evaluate the maximum Link Load Ratio (LLR) of all the algorithms. Figures 3.9 and 3.10 show the Max. LLR value for all the algorithms. We find that Opt_LB achieves the lowest Max. LLR value, and the approximation algorithm achieves a very close performance with OPT_LB. SFC-MAP performs ranks the third regarding the performance of Max. LLR value. Random algorithm performs poorly by having the highest Max. LLR value because of its randomness in choosing placed node and paths regardless of link residual bandwidth. For all the algorithms, we found that their achieved Max. LLR value for the partially ordered SFC request in Fig. 3.10 is larger than the Max. LLR value for the partially ordered SFC request in Fig. 3.9. This is because more paths need to be established in the partially ordered SFC, and this will consume more bandwidth which results in larger Max. LLR.

After that, we verify the performances of the algorithms when different K (number of paths between node pairs) is set. As stated above, we omit the AR values for OPT_LB. For simplicity we only present the results when $|R| = 40$. We see that when K increases from 1 to 10, the AR value increases for all the algorithms except for SFC-MAP in Figs. 3.11 and 3.12, and the Max. LLR value decreases for all the algorithms in Figs. 3.13 and 3.14. But this trend does not change from 10 to 20 for

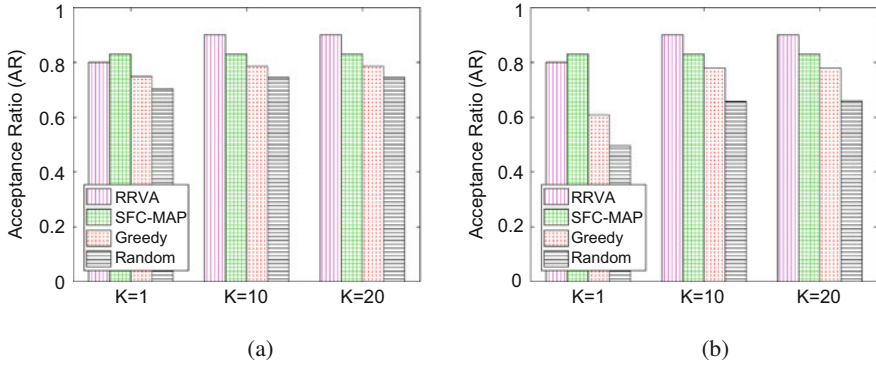


Fig. 3.11 Acceptance Ratio for K paths and totally ordered SFC ($|R| = 40$). (a) USANet. (b) GÉANT

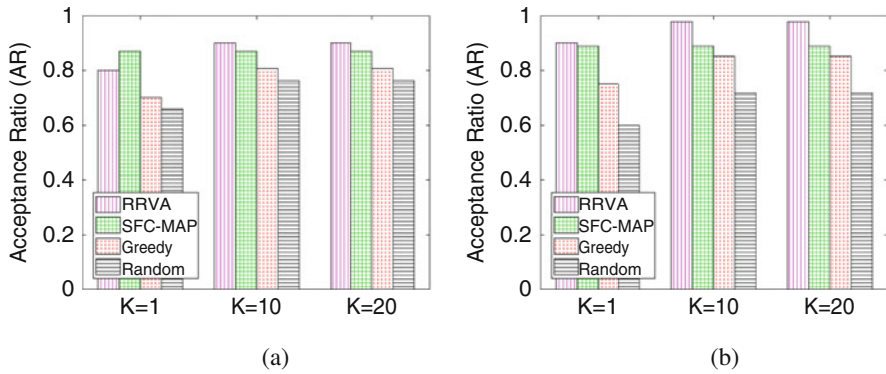


Fig. 3.12 Acceptance Ratio for K paths and partially ordered SFC ($|R| = 40$). (a) (b) GÉANT

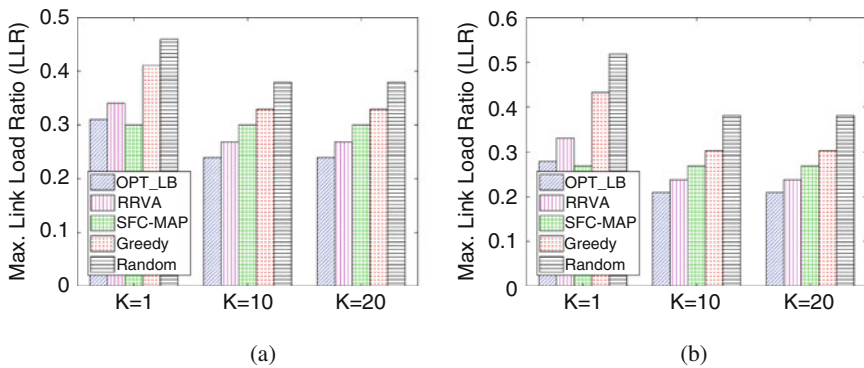


Fig. 3.13 Maximum Link Load Ratio (LLR) for K paths and totally ordered SFC ($|R| = 40$). (a) USANet. (b) GÉANT

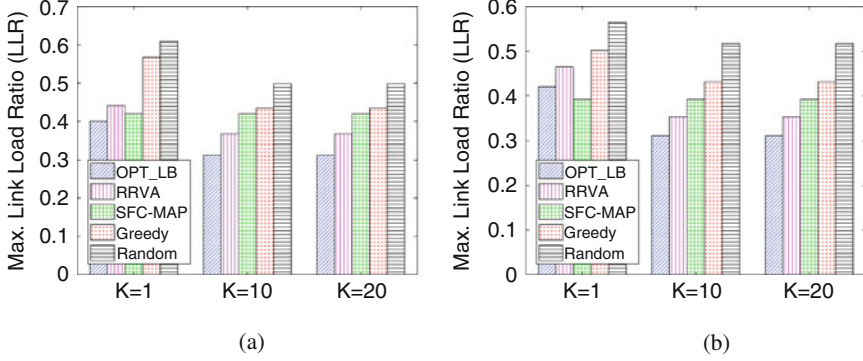


Fig. 3.14 Maximum Link Load Ratio (LLR) for K paths and partially ordered SFC ($|R| = 40$). (a) USANet. (b) GÉANT

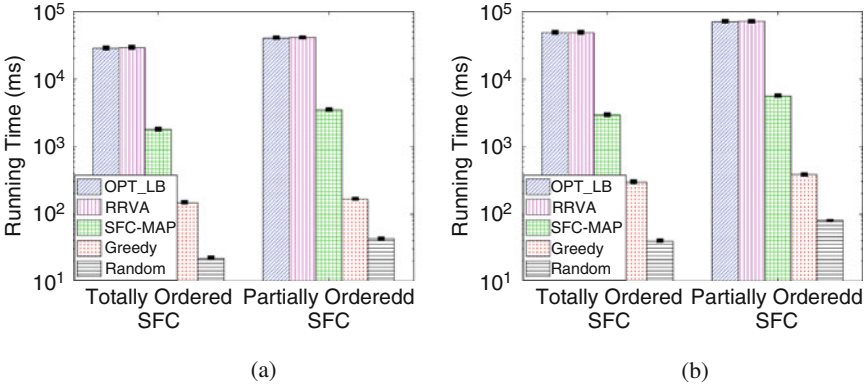


Fig. 3.15 Total Running Time for $|R| = 40$ requests (95% confidence interval). (a) USANet. (b) GÉANT

both AR and Max. LLR, which implies that $K = 10$ paths are enough to guarantee QoS (e.g., delay). This is because SFC-MAP only applies shortest path algorithm to find route between VNF pair and therefore the number of paths does not affect its performance.

Finally, Fig. 3.15 presents the total running time of all the algorithms (in log scale) for $|R| = 40$, where the confidence interval is set to 95%. We see that the confidence intervals of all the algorithms are not very visible,⁷ which indicates that the variances are also very small. We see that the OPT_LB has similar running time with RRVA since their time complexities are dominated by the LP, but both of them are higher than SFC-MAP and two other heuristics. Nevertheless, it is still acceptable since RRVA can achieve a close-to-optimal performance as shown in

⁷ Figure 3.15 is log-scale that additionally contributes to the confidence interval visibility.

Figs. 3.6, 3.7, 3.8, 3.9, and 3.10. SFC-MAP has much higher running time than Greedy and Random heuristic due to its increased problem size ($|F| + 1$ times nodes and links) and maximum iteration times. Another observation is that when $|R| = 1$, we found our proposed approximation algorithm can return the solution in less than several seconds (We omit the figure for brevity). In practice, in many cases the traffic requests can be known or predicted (Alawe et al. 2018) in advance, therefore the service provider can apply the proposed approximation algorithm to calculate the solutions in an offline fashion (in acceptably short time as implied in Fig. 3.15) for the (delay-sensitive) incoming traffic requests. In addition, our proposed approximation algorithm can also be applied only periodically or as complementary solution for existing methods. Moreover, for the same algorithm, we found that its running time for the totally ordered SFC is lower than the value for the partially ordered SFC. This can be explained that since the partially ordered SFC consists of one or more totally ordered sub-SFC, then there are more data input or constraints in the OPT_LB and RRVA, which results in higher running time. Similarly, Greedy and Random needs to establish more paths for VNF pairs in different segments, which also consumes more time.

3.7 Summary

In this chapter, we have studied the Delay-aware VNF Placement and Routing (DVPR) problem in edge clouds, which is to place each end user's requested VNFs on network nodes and find routes among each adjacent VNF pair such that the maximum link load ratio λ is minimized and the total traversing delay is no greater than the specified value. We have considered this problem for both the totally ordered SFC and partially ordered SFC. We have proved that the DVPR problem is NP-hard, even for the special case when only one VNF is requested. We subsequently propose a randomized rounding approximation algorithm to solve it. Simulation results reveal that the proposed randomized rounding approximation algorithm outperforms three other heuristics and achieves close-to-optimal performances in terms of acceptance ratio and maximum link load ratio. In practice, the proposed approach can provide the service provider with the VNF placement and routing solutions for end users' requests such that each user's NFV delay requirement is satisfied while the maximum link load ratio is minimized. By minimizing the maximum link load ratio, the actual network flow is evenly distributed across the network, and more future requests can be accommodated since we try our best to reduce the network bottleneck. These scenarios, requiring robust latency and availability requirements, need accurate mathematical modeling and implementation of algorithms to support QoS, as discussed in the following chapters of this book.

References

- Alameddine HA, Qu L, Assi C (2017) Scheduling service function chains for ultra-low latency network services. In: 2017 13th international conference on network and service management (CNSM), pp 1–9
- Alawe I, Ksentini A, Hadjadj-Aoul Y, Bertin P (2018) Improving traffic forecasting for 5G core network scalability: a machine learning approach. *IEEE Netw* 32(6):42–49
- Allybokus Z, Perrot N, Leguay J, Maggi L, Gourdin E (2018) Virtual function placement for service chaining with partial orders and anti-affinity rules. *Networks* 71(2):97–106
- Anstreicher KM (1999) Linear programming in $O((n^3/\ln n) \log n)$ operations. *SIAM J Optim* 9(4):803–812
- Bera S, Misra S, Jamalipour A (2019) Flowstat: adaptive flow-rule placement for per-flow statistics in SDN. *IEEE J Sel Areas Commun* 37(3):530–539
- Bhamare D, Jain R, Samaka M, Vaszkun G, Erbad A (2015) Multi-cloud distribution of virtual functions and dynamic service deployment: open ADN perspective. In: IEEE international conference on cloud engineering, pp 299–304
- Bhamare D, Jain R, Samaka M, Erbad A (2016) A survey on service function chaining. *J Netw Comput Appl* 75:138–155
- Bhamare D, Samaka M, Erbad A, Jain R, Gupta L, Chan HA (2017) Optimal virtual network function placement in multi-cloud service function chaining architecture. *Comput Commun* 102:1–16
- Bhamare D, Erbad A, Jain R, Zolanvari M, Samaka M (2018) Efficient virtual network function placement strategies for cloud radio access networks. *Comput Commun* 127:50–60
- Ceselli A, Premoli M, Secci S (2017) Mobile edge cloud network design optimization. *IEEE/ACM Trans Netw* 25(3):1818–1831
- Chiosi M, Clarke D, Willis P, Reid A, Feger J, Bugenhagen M, Khan W, Fargano M, Cui C, Deng H et al. (2012) Network functions virtualisation introductory white paper. In: SDN and OpenFlow world congress
- Cohen R, Lewin-Eytan L, Naor JS, Raz D (2015) Near optimal placement of virtual network functions. In: IEEE INFOCOM
- Comtet L (1974) Advanced combinatorics: the art of finite and infinite expansions. Springer Netherlands
- Cziva R, Anagnostopoulos C, Pezaros DP (2018) Dynamic, latency-optimal vNF placement at the network edge. In: IEEE INFOCOM
- Dietrich D, Papagianni C, Papadimitriou P, Baras JS (2017) Network function placement on virtualized cellular cores. In: IEEE international conference on communication systems and networks (COMSNETS), pp 259–266
- Dwaraki A, Wolf T (2016) Adaptive service-chain routing for virtual network functions in software-defined networks. In: Proceedings of the 2016 workshop on hot topics in middleboxes and network function virtualization, ser. ACM HotMiddlebox, pp 32–37
- Eramo V, Miucci E, Ammar M, Lavacca FG (2017) An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures. *IEEE/ACM Trans Netw* 25(4):2008–2025
- ETSI Publishes First Specifications for Network Functions Virtualisation. <http://www.etsi.org/news-events/news/700-2013-10-etsi-publishes-first-nfv-specifications>
- Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of np-completeness. W. H. Freeman, New York
- Ghaznavi M, Khan A, Shahriar N, Alsubhi K, Ahmed R, Boutaba R (2015) Elastic virtual network function placement. In: IEEE 4th international conference on cloud networking (CloudNet), pp 255–260
- Goudarzi H, Pedram M (2011) Multi-dimensional SLA-based resource allocation for multi-tier cloud computing systems. In: IEEE 4th international conference on cloud computing, pp 324–331

- Guo L, Pang J, Walid A (2018) Joint placement and routing of network function chains in data centers. In: IEEE INFOCOM
- Gupta A, Jaumard B, Tornatore M, Mukherjee B (2018a) A scalable approach for service chain mapping with multiple sc instances in a wide-area network. *IEEE J Sel Areas Commun* 36(3):529–541
- Gupta A, Tornatore M, Jaumard B, Mukherjee B (2018b) Virtual-mobile-core placement for metro network. In: IEEE conference on network softwarization and workshops (NetSoft), pp 308–312
- Hemmecke R, Köppe M, Lee J, Weismantel, R (2010) Nonlinear integer programming. In: 50 Years of integer programming 1958–2008. Springer, Berlin, pp 561–618
- Herrera JG, Botero JF (2016) Resource allocation in NFV: a comprehensive survey. *IEEE Trans Netw Serv Manag* 13(3):518–532
- Hou I-H, Zhao T, Wang S, Chan K (2016) Asymptotically optimal algorithm for online reconfiguration of edge-clouds. In: ACM MobiHoc, pp 291–300
- Hu YC, Patel M, Sabella D, Sprecher N, Young V (2015) Mobile edge computing: a key technology towards 5G. ETSI White paper
- Huin N, Jaumard B, Giroire F (2018) Optimal network service chain provisioning. *IEEE/ACM Trans Netw* 26(3):1320–1233
- Kuo T-W, Liou B-H, Lin KC-J, Tsai M-J (2016) Deploying chains of virtual network functions: On the relation between link and server usage. In: IEEE INFOCOM
- Leconte M, Destounis A, Paschos G (2018) Traffic engineering with precomputed pathbooks. In: IEEE INFOCOM
- Li C-L, McCormick S, Simchi-Levi D (1990) The complexity of finding two disjoint paths with min-max objective function. *Discrete Appl Math* 26(1):105–115
- Li Q, Jiang Y, Duan P, Xu M, Xiao X, Quokka (2017) Quokka: latency-aware middlebox scheduling with dynamic resource allocation. *J Netw Comput Appl* 78:253–266
- Lin S-C, Wang P, Akyildiz I, Luo M (2018) Towards optimal network planning for software-defined networks. *IEEE Trans Mobile Comput* 17(12):2953–2967
- Liu J, Lu W, Zhou F, Lu P, Zhu Z (2017) On dynamic service function chain deployment and readjustment. *IEEE Trans Netw Serv Manag* 14(3):543–553
- Lyu X, Tian H, Jiang L, Vinel A, Maharjan S, Gjessing S, Zhang Y (2018) Selective offloading in mobile edge computing for the green internet of things. *IEEE Netw* 32(1):54–60
- Ma W, Sandoval O, Beltran J, Pan D, Pissinou N (2017a) Traffic aware placement of interdependent NFV middleboxes. In: IEEE INFOCOM
- Ma X, Zhang S, Li W, Zhang P, Lin C, Shen X (2017b) Cost-efficient workload scheduling in cloud assisted mobile edge computing. In: IEEE/ACM IWQoS, pp 1–10
- Mach P, Becvar Z (2017) Mobile edge computing: a survey on architecture and computation offloading. *IEEE Commun Surv Tutor* 19(3):1628–1656
- Mao Y, You C, Zhang J, Huang K, Letaief KB (2017) A survey on mobile edge computing: the communication perspective. *IEEE Commun Surv Tutor* 19(4):2322–2358
- Medhat AM, Taleb T, Elmangoush A, Carella GA, Covaci S, Magedanz T (2017) Service function chaining in next generation networks: state of the art and research challenges. *IEEE Commun Mag* 55(2):216–223
- Mehraghdam S, Keller M, Karl H (2014) Specifying and placing chains of virtual network functions. In: IEEE 3rd international conference on cloud networking (CloudNet), pp 7–13
- Mell P, Grance T (2010) The NIST definition of cloud computing. *Commun ACM* 53(6):50,
- Mijumbi R, Serrat J, Gorricho J-L, Bouten N, De Turck F, Boutaba R (2016) Network function virtualization: state-of-the-art and research challenges. *IEEE Commun Surv Tutor* 18(1):236–262
- Mills K, Filliben J, Dabrowski C (2011) Comparing VM-placement algorithms for on-demand clouds. In: IEEE third international conference on cloud computing technology and science, pp 91–98
- Pei J, Hong P, Xue K, Li D (2018) Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system. *IEEE Trans Parallel Distrib Syst* 30(10):2179–2192

- Pires FL, Barán B (2013) Multi-objective virtual machine placement with service level agreement: a memetic algorithm approach. In: Proceedings of the 2013 IEEE/ACM 6th international conference on utility and cloud computing, pp 203–210
- Qu L, Assi C, Shaban K (2016) Delay-aware scheduling and resource optimization with network function virtualization. *IEEE Trans Commun* 64(9):3746–3758
- Saha N, Bera S, Misra S (2018) Sway: traffic-aware QoS routing in software-defined IoT. *IEEE Trans Emer Topics Comput* 9(1):390–401
- Savi M, Tornatore M, Verticale G (2015) Impact of processing costs on service chain placement in network functions virtualization. In: IEEE conference on network function virtualization and software defined network (NFV-SDN), pp 191–197
- Su M, Zhang L, Wu Y, Chen K, Li K (2015) Systematic data placement optimization in multi-cloud storage for complex requirements. *IEEE Trans Comput* 65(6):1964–1977
- Sun C, Bi J, Zheng Z, Yu H, Hu H (2017) NFP: enabling network function parallelism in NFV. In: ACM SIGCOMM, pp 43–56
- Tarjan R (2009) Course: advanced algorithm design. Lecture: Chernoff, probability and computing. Princeton University, Princeton
- Xu Z, Liang W, Xu W, Jia M, Guo S (2016) Efficient algorithms for capacitated cloudlet placements. *IEEE Trans Parallel Distrib Syst* 27(10):2866–2880
- Xu H, Yu Z, Li XY, Huang L, Qian C, Jung T (2017) Joint route selection and update scheduling for low-latency update in SDNs. *IEEE/ACM Trans Netw* 25(5):3073–3087
- Yang B, Chai WK, Xu Z, Katsaros KV, Pavlou G (2018) Cost-efficient NFV-enabled mobile edge-cloud for low latency mobile applications. *IEEE Trans Netw Serv Manag* 15(1):475–488
- Yang S, Li F, Trajanovski S, Chen X, Wang Y, Fu X (2021) Delay-aware virtual network function placement and routing in edge clouds. *IEEE Trans Mobile Comput* 20(2):445–459
- Yang S, Li F, Yahyapour R, Fu X (2022) Delay-sensitive and availability-aware virtual network function scheduling for NFV. *IEEE Trans Serv Comput* 15(1):188–201
- Yi B, Wang X, Li K, Huang M et al. (2018) A comprehensive survey of network function virtualization. *Comput Netw* 133:212–262
- Zhang Z, Li Z, Wu C, Huang C (2017) A scalable and distributed approach for NFV service chain cost minimization. In: IEEE ICDCS, pp 2151–2156

Chapter 4

Delay-Sensitive and Availability-Aware Virtual Network Function Scheduling for NFV



As noted before, the VNF placement and routing problem can be addressed with the goal of minimizing the maximum link load ratio and each user's requested delay is satisfied. However, in addition to requested delay, resiliency is also an important Service Level Agreements (SLA) in a NFV service. In this chapter, we first investigate how to quantitatively model the traversing delay of a flow in both totally ordered and partially ordered SFCs.¹ Subsequently, we study how to calculate the VNF placement availability mathematically for both unprotected and protected SFCs. After that, we study the delay-sensitive Virtual Network Function (VNF) placement and routing problem with and without resiliency concerns. We prove that this problem is NP-hard under two cases. We subsequently propose an exact Integer Nonlinear Programming (INLP) formulation and an efficient heuristic for this problem in each case. Finally, we evaluate the proposed algorithms in terms of acceptance ratio, average number of used nodes and total running time via extensive simulations.

4.1 Introduction

In the traditional network services provisioning paradigm, network functions (e.g., firewall or load balancer) which are also called middleboxes are usually implemented by the dedicated hardware appliances. Deploying hardware middleboxes is costly due to their high cost for design and production and also these middleboxes need to be configured and managed manually, which further increases the costs of service providers. Network Function Virtualization (NFV) which is first proposed by has been emerged as an appealing solution, since it enables to replace dedicated hardware implementations with software instances running in a virtualized

¹ Parts of this chapter is reprinted from Yang et al. (2022), with permission from IEEE.

environment. In NFV, the requested service is implemented by a sequence of Virtual Network Functions (VNF) that can run on generic servers by leveraging the virtualization technology. Service Function Chaining (SFC) is therefore defined as a chain-ordered set of placed VNFs that handles the traffic of the delivery and control of a specific application (Medhat et al. 2017). NFV allows to allocate network resources in a more scalable and elastic manner, offer a more efficient and agile management and operation mechanism for network functions and hence can largely reduce the overall costs.

In a SFC, the end-to-end delay (Verma 2004) of the packets which traverse each VNF in this chaining is an important Service Level Agreement (SLA) to measure the performance of NFV. To guarantee a reliable network service, service providers need to promise a delay-sensitive performance and service to the customers. In this sense, the service providers may get a revenue loss if the promised total delay is violated. This deals with how to place the required ordered VNFs in the network and how to route the packets among these VNFs.

Resiliency (Franke 2012) is another important SLA of NFV that defines the level the provided service can survive in the face of failures. Since, once a node or a link in a SFC fails, the whole SFC cannot operate and hence the provided service has to be stopped. To tackle this concern, we need to provide redundant SFCs to protect the primary SFC, although this comes at the expense of more network resource consumption. As a result, how to provide a customer's satisfied resilient NFV service by using minimum costs remains to be an important problem for network providers to solve.

In this chapter, we first study the Delay-Sensitive VNF Scheduling problem, which is to place all the requested VNFs according to their predefined order and route the traffic among these VNFs without exceeding the end-to-end delay such that the number of used nodes is minimized. On the basis of the DSVS problem, we study the Delay-sensitive and Availability-aware NFV Scheduling problem, which additionally takes the VNF placement availability constraint into account. Our key contributions are as follows:

- We propose a model to quantitatively calculate the traversing delay for a flow in a SFC.
- We present a model to quantitatively measure the SFC availability for NFV resiliency.
- We propose an exact Integer Nonlinear Programming (INLP) formulation and an efficient heuristic for placing and routing the delay-sensitive virtual network functions with and without resiliency concerns.
- We validate and investigate the performance of the proposed algorithms via extensive simulations.

The outline of this chapter is organized as follows: Sect. 4.2 presents the related work. Section 4.3 analyzes and defines the delay calculation model in both totally ordered and partially ordered SFCs, and Sect. 4.4 presents the VNF placement availability calculation model. Section 4.5 defines the Delay-Sensitive VNF Scheduling (DSVS) problem and the Delay-sensitive and Availability-aware

VNF Scheduling (DAVS) problem and analyzes their complexities. In Sect. 4.6, we propose an exact INLP formulation and an efficient heuristic for each problem. Section 4.7 provides the simulation results and we conclude in Sect. 4.8.

4.2 Related Work

A comprehensive survey about NFV can be found in (Bhamare et al. 2015; Mijumbi et al. 2016; Yi et al. 2018; Hantouti et al. 2018). Herrera and Botero (2016) provides a survey about resource allocation in NFV.

4.2.1 *Traffic and Cost-Aware VNF Placement and Routing*

Eramo et al. (2017) first target the VNF routing and placement problem with the goal of maximizing the amount of data that can be processed within the network at peak traffic. They subsequently study how to consolidate VNFs and shut down unused servers such that the total operation cost (energy consumption+revenue loss) is minimized. They propose an exact Integer Linear Programming (ILP) and an efficient heuristic to solve each problem. Li et al. (2016) consider a fat tree data center topology, and study (1) service chaining consolidation, (2) pod assignment, and (3) machine assignment per pod problems in both offline and online situation. The delay concern is also considered in these 3 problems. Consequently, efficient heuristics are proposed to solve these problems. Ma et al. (2017) study the VNF placement problem to minimize the maximum link load in three cases: (1) when there is no dependency between VNFs, (2) totally-ordered: there is a total dependency order on the VNF set, and (3) partially-ordered: there exists dependency among a subset of VNFs. Assuming that the path between any network node pair is precalculated, they propose a polynomial-time algorithm for case (1), and prove that cases (2) and (3) are NP-hard. To solve them, a dynamic programming and an efficient heuristic are proposed to solve (2) and (3), respectively. Cohen et al. (2015) propose a near-optimal approximation algorithm to place VNF without considering network function dependency. Kuo et al. (2016) relax/approximate the VNF placement and routing problem based on the intuition that placing VNFs on a shorter path consumes less link bandwidth, but might also reduce VM reuse opportunities; reusing more VMs might lead to a longer path, and so it consumes more link bandwidth. Feng et al. (2017) propose an approximation algorithm to find the minimum cost solution for placing VNFs and steering traffic between them for a given set of requests. Pham et al. (2017) propose a sampling-based Markov approximation algorithm to jointly minimize the operational and network traffic cost for placing VNFs. Zhang et al. (2018) transform the VNF placement problem into min-cost flow problem and the devise an efficient heuristic to solve this problem with the gola of minimizing total costs. Guo et al. (2018) jointly

consider the VNF placement and routing problem in data centers. They propose a randomized approximation algorithm when the traffic matrix is known in advance and a competitive online algorithm when the future arriving traffic is not known. However, they assume that one configuration in data centers consists of one VNF placement and one routing path solution, and a (limited) set of configurations is given. Nevertheless, the delay is not taken into account in these work.

4.2.2 Delay-Sensitive VNF Scheduling

Qu et al. (2016) consider the VNF transmission and processing delays, and formulate the joint problem of VNF scheduling and traffic steering as a Mixed Integer Linear Program (MILP). However, they assume that the virtual link between two physical nodes can at most process one traffic flow at one time, irrespective of the flow size. Vizarreta et al. (2017) solve the QoS VNF placement problem, where the end-to-end delay and service chaining availability are considered by proposing an ILP and a heuristic. However, their adopted delay model does not relate with packet size and also the VNF placement availability in the VNF protection case is not considered in their work. Zhang et al. (2017) devise an Alternating Direction Method of Multipliers (ADMM)-based algorithm to solve the delay-aware VNF placement and routing problem. However, they do not consider the nodes' delay in their problem. Li et al. (2017) address the latency-aware middlebox routing and placement problem by leveraging a packet queuing model. However, fixed link transmission delay is assumed in their model. Dwaraki and Wolf (2016) devise a layered-graph based heuristic to find delay-aware routes when the requested VNFs have been placed on network nodes. Allybokus et al. (2017) propose an exact ILP and a greedy heuristic to solve the VNF placement and routing problem for both full and partially ordered SFCs. However, their solutions only consider a simple path within a SFC. Chen et al. (2018) propose a hidden Markov Chain-based heuristic to place VNFs which jointly minimize the cost and delay for a set of given flows. Sun et al. (2017) propose a framework that enables (independent) network function to work in parallel, which largely improve NFV performance in terms of delay. Gouareb et al. (2018) present several heuristics to solve the VNF placement and routing problem from holistic view by leveraging queueing theory in edge clouds. Nevertheless, different from above work, our proposed delay calculation model is related with the packet size of the flow. Moreover, we consider both node and link delay and the required route traversing an entire SFC in the proposed problems is not necessarily a simple path, which is more general.

4.2.3 NFV Resiliency

Hmaity et al. (2016) study the VNF protection in three cases, namely, (1) end-to-end protection: a primary SFC and a backup SFC are completely (both node- and link-) disjoint; (2) link-protection, and (3) node-protection. They propose an ILP for each case in order to place VNFs to minimize the number of used nodes. Beck et al. (2016) propose a recursive heuristic for survivable VNF placement to guarantee an end-to-end VNF protection. Han et al. (2017) explore resilient respects of the individual VNF in terms of fault management (e.g., failure detection and automated recovery), state management, etc. They also discuss existing solutions for these aspects. Herker et al. (2015) formulate how to calculate the VNF placement availability in data center topology. They also present an efficient heuristic on how to place VNFs and their backups to satisfy the requested availability. Fan et al. (2015) target how to compute one backup SFC when the primary SFC is given such that the overall availability is satisfied. However, the considered availability model ignores the global information of the entire SFC, which is not precise enough. Ding et al. (2017) formulate how to calculate VNF placement availability when at most one backup chaining is allowed. On basis of Fan et al. (2015), given that the primary SFCs are already placed in the network, they Ding et al. (2017) propose a heuristic on how to calculate the respective backup SFCs with minimum cost such that the total availability for each request is satisfied. Qu et al. (2017) consider both delay and availability constraints to route and place SFCs. However, both node delay and link availability are not taken into account in their model. In addition, in their proposed VNF placement availability calculation model, the delay is not considered in each SFC when calculating the whole availability, which is not precise enough. Different from above work, our VNF placement availability model is general since it can precisely calculate the total availability of any $k \geq 1$ SFCs in both fully protected and partially protection scenario.

4.3 Delay Calculation for a flow in a Service Function Chaining

We consider traversing delays under two cases for VNFs in a SFC, namely, (1) Totally Ordered: there is a total dependency order on the VNF set, and (2) Partially Ordered (Sun et al. 2017): there exists dependency among a subset of VNFs. That is, \exists at least two VNFs m and n , such that they have the same predecessor function and successor function but they have no dependencies with each other. For example, Fig. 4.1a is a totally ordered SFC. However, since function monitor only maintains the packets and does not change the packets, firewall and monitor can work in parallel as shown in Fig. 4.1b. After that, firewall and monitor functions send their individual processed packets to load balancer. As a result, load balancer only needs to select the packets from firewall and process them. In this way, as we will show

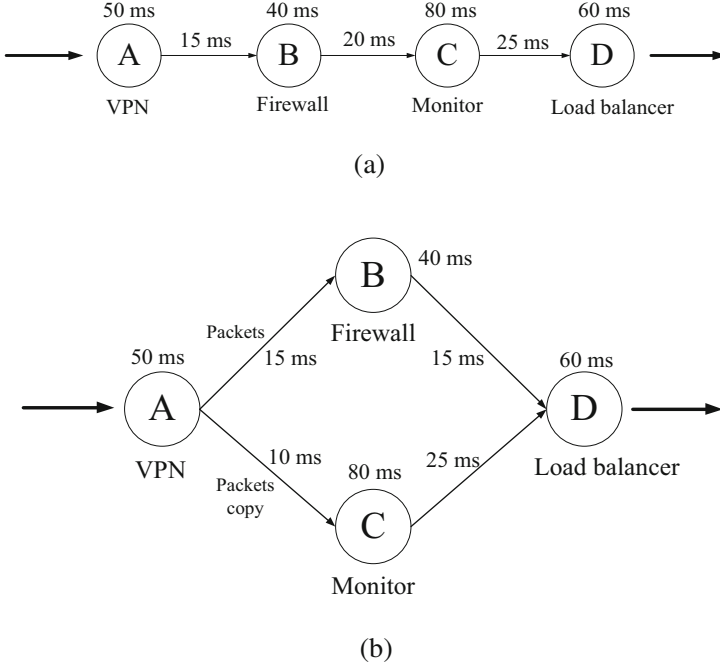


Fig. 4.1 Totally Ordered Chaining v.s. Partially Ordered Chaining, derived from Sun et al. (2017). (a) Totally ordered chaining. (b) Partially ordered chaining

later, the delay in Fig. 4.1b is less than the one in Fig. 4.1a. In the following, we will quantitatively show how to calculate the traversing delay for a flow in both totally ordered and partially ordered chainings.

4.3.1 Totally Ordered SFC

A network is represented by $\mathcal{G}(\mathcal{N}, \mathcal{L})$, where \mathcal{N} denotes a set of N nodes and \mathcal{L} stands for a set of L links. Each node $n \in \mathcal{N}$ has a maximum processing capability of $Cap(n)$ and each link $l \in \mathcal{L}$ is associated with bandwidth $b(l)$. A request is represented by $r(v, e, F, D)$, where v denotes the traffic volume (or total packet size) that needs to be delivered through the SFC, e means the requested network transmitting rate, F indicates the set of VNFs with orders, and D stands for the total requested end-to-end SFC delay. The VNF $f \in F^2$ (with required processing capacity $\eta(f)$) on node $n \in \mathcal{N}$ requires time of p_n^f to process it. The delay on a link

² In this chapter, F is the set of required ordered VNFs of request, and SFC is defined as a chain-ordered set of placed VNFs that handles the traffic of the delivery and control of a specific

l for a flow with total size of v can be calculated as:³

$$\frac{v}{e} \cdot \theta(l) \quad (4.1)$$

where $\theta(l)$ is a link attenuation parameter value and reflects the factors of the link that impact the transmitting delay such as link distance, signal impairmentness, etc. We assume that a SFC contains a set of $|F|$ ordered VNFs, $f_1, f_2, \dots, f_{|F|}$ which need to be placed in the network. Suppose there is a total dependency order on $|F|$ VNFs, then the traversing delay in this SFC is calculated as follows:

$$\sum_{f \in F, n \in N} p_n^f + \sum_{l \in \mathcal{L}^s} T(l) \quad (4.2)$$

where p_n^f represents the processing delay for function f on node n , \mathcal{L}^s denotes the link set that connects the placed VNF, and $T(l)$ indicates the flow delivering delay on link l . For example, in Fig. 4.1a where the flow delivering delay and node processing delay are associated, the traversing delay is $\underbrace{(50 + 40 + 80 + 60)}_{\text{total node delay}} + \underbrace{(15 + 20 + 25)}_{\text{total link delay}} = 290$ ms.

4.3.2 Partially Ordered SFC

Suppose at least two VNFs are not dependent with each other, and both of them need to receive the packets from the same prior VNF f_r and need to send packets to the same VNF f_s . We first partition the SFC into several segments and thus there is one or more independent VNFs in each segment. For example, there are 3 segments in Fig. 4.1b: Segment 1 contains the function VPN, segment 2 is composed of functions firewall and monitor, and segment 3 has the function load balancer. As a result, the traversing delay in a partially ordered SFC is equal to the longest path delay from the node which hosts one of the VNFs in the first segment to the node which contains one of the VNFs in the last segment. For example, the traversing delay of SFC in Fig. 4.1b is equal to $\underbrace{(50 + 80 + 60)}_{\text{total node delay}} + \underbrace{(10 + 25)}_{\text{total link delay}} = 225$ ms, which is less than

the one in Fig. 4.1a. This example also illustrates that the partially ordered SFC can reduce delay compared to the totally ordered SFC. We notice that the totally ordered SFC is a special case of the partially ordered SFC, since when each segment contains

application. In this sense, F means the VNFs are to be deployed, and SFC represents the VNFs that have already been placed for traffic delivery and packet processing in network.

³ In this chapter, we assume a proportional-to-flow-size link delay model, but our work can also be applied to the fixed link delay scenario.

only one VNF, the partially ordered SFC becomes the totally ordered SFC. Without loss of generality, we assume a (totally ordered or partially ordered) SFC consists of a set of $|M|$ segments M , and each segment $m \in M$ has $|m|$ functions. We use m_i and m_j to represent two consecutive segments, where $j = i + 1$ and $1 \leq i \leq |M| - 1$. In order to calculate the traversing delay in a partially ordered SFC with segments M , we first make the definition of *totally ordered sub-SFC* as follows:

Definition 4.1 (Totally Ordered Sub-SFC) Suppose there is a partially ordered SFC C with segment set M . A totally ordered sub-SFC is a chaining consisting of exactly one VNF in each segment.

According to the above definition, there are in total $\prod_{i=1}^{|M|} |m_i|$ totally ordered sub-SFCs. For example in Fig. 4.1b, there are $1 \cdot 2 \cdot 1 = 2$ totally ordered sub-SFCs, namely $A \rightarrow B \rightarrow D$ and $A \rightarrow C \rightarrow D$. As a result, the delay value of a partially ordered SFC is the maximum delay value among all the totally ordered sub-SFCs composing it.

4.4 VNF Placement Availability Calculation

The availability of a system is the fraction of time that the system is operational during the entire service time. The availability A_j of a network component j can be calculated as McCool (2003):

$$A_j = \frac{MTTF}{MTTF + MTTR} \quad (4.3)$$

where $MTTF$ represents Mean Time To Failure and $MTTR$ denotes Mean Time To Repair. In this chapter, a node in the network represents a server, and a link denotes a physical link connecting two physical servers. Their availabilities are equal to the product of the availabilities of all their components (e.g., hard disk and memory for a node, amplifiers and fibers for a link). For a server (or any other equipment) n , let A_n and B_n denote its availability value and failure probability, respectively, then we have $A_n = 1 - B_n$. In reality, we can obtain the equipment's availability value by accessing the detailed logs recording every hardware component repair/failure incident during its lifetime. The details for characterizing e.g., server node failures and statistically calculating node availability can be found in (Vishwanath and Nagappan 2010; Ford et al. 2010) and work therein.

We consider a general multiple node and link failure scenario. That is, multiple nodes and links may fail at one certain time point. We distinguish and analyze the VNF placement availability under two different cases, namely (1) Unprotected SFC: only one SFC is allowed to be placed in the network, and (2) Protected SFC: at most $k \geq 2$ SFCs (Hmaity et al. 2016) can be placed in the network.

Suppose an unprotected SFC places VNFs on w nodes n_1, n_2, \dots, n_w and traverses m links l_1, l_2, \dots, l_m , its availability is calculated as:

$$\prod_{i=1}^w A_{n_i} \cdot \prod_{j=1}^m A_{l_j} \quad (4.4)$$

where $\prod_{i=1}^w A_{n_i}$ denotes the availability of all the used nodes and $\prod_{j=1}^m A_{l_j}$ indicates the availability of all the traversed links.⁴

In the protected SFC, we regard that it is composed of (maximum) k unprotected SFCs. For clarification, we further term each of the k unprotected SFC in the protected placement as placement group Φ_i . We regard Φ_1 as the primary placement group. Since different placement groups may place one or more VNFs on the same node and/or traverse the same link, we distinguish the protected placement as two cases, namely (1) *fully protected SFC*, each one of k placement groups places VNF on different nodes and traverses different links and (2) *partially protected SFC*, at least two placement groups place one or more VNFs on the same node and/or traverse the same link. In the fully protected placement case, the availability can be calculated as:

$$A_{FCH}^k = 1 - \prod_{i=1}^k (1 - A_{\Phi_i}) \quad (4.5)$$

Equation (4.5) indicates that the availability of k SFCs is equal to the probability that at least one SFC can work normally (does not fail). For example, in Fig. 4.2a where the node and link availabilities are associated, SFC s-a-b-d and s-c-g-d are fully protected, since they do not contain any same link or node. Although s and d are overlapped by these two SFCs, these two nodes are the ingress and egress nodes in the SFC and we do not consider their availabilities in this chapter or we assume their availabilities are always 1. As a result, the total availability of these two SFCs are: $1 - (1 - 0.9 \cdot 0.99 \cdot 0.8 \cdot 0.85 \cdot 0.95) \cdot (1 - 0.95 \cdot 0.98 \cdot 0.75 \cdot 0.99 \cdot 0.88) \approx 0.8338$.

Next, we consider a general scenario where at least two among the k placement groups place the same VNF on the same node⁵ or traverse the same links. In this case, Eq. (4.5) cannot be used to calculate the availability in this scenario since the availabilities of overlapped nodes or links will be counted more than once. To amend this, we use a new operator \circ .⁶ Suppose there are m different nodes n_1, n_2, \dots, n_m

⁴ It is possible that one SFC traverses one link multiple times, but we consider the availability of each of the traversed links only once.

⁵ In this scenario, one VNF may serve for more than one placement groups. In this chapter we assume each VNF placed on some node can have sufficient capability for servicing at most k placement groups (Hmaity et al. 2016).

⁶ As in (Yang et al. 2015) for the partially link-disjoint paths connection availability and in (Yang et al. 2017) for the virtual machine partially protection.

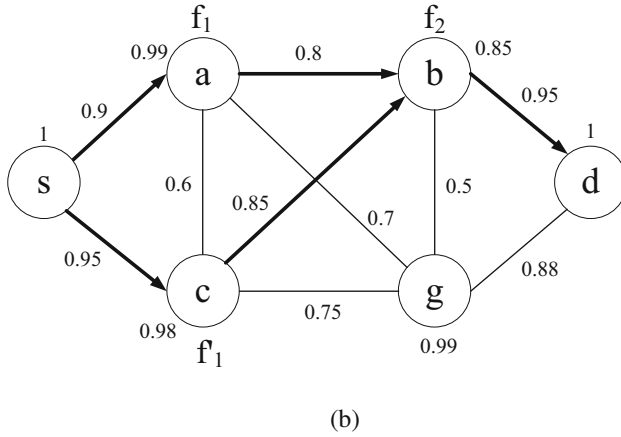
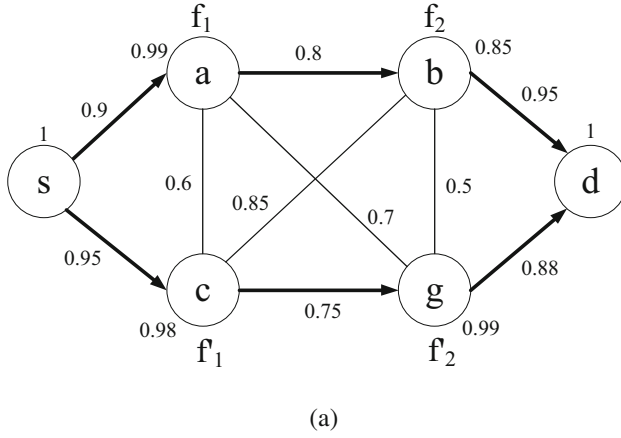


Fig. 4.2 SFC protection. (a) Fully protected SFC. (b) Partially protected SFC

with availabilities $A_{n_1}, A_{n_2}, \dots, A_{n_m}$. For a node n_x with availability A_{n_x} , \circ can be defined as follows:

$$A_{n_1} \cdot A_{n_2} \cdot \dots \cdot A_{n_m} \circ A_{n_x} = \begin{cases} \prod_{i=1}^m A_{n_i} & \text{if } \exists n_i = n_x \\ \prod_{i=1}^m A_{n_i} \cdot A_{n_x} & \text{otherwise} \end{cases} \quad (4.6)$$

where the \circ computations for link availabilities can be defined analogously.

Let \prod denote consecutive \circ operations of the different sets, then the availability (represented by A_{PCH}^k) of k SFCs can now be represented as:

$$\begin{aligned}
 A_{PCH}^k &= 1 - \prod_{i=1}^k (1 - A_{\Phi_i}) \\
 &= 1 - (1 - A_{\Phi_1}) \circ (1 - A_{\Phi_2}) \circ \cdots \circ (1 - A_{\Phi_k}) \quad (4.7) \\
 &= \sum_{i=1}^k A_{\Phi_i} - \sum_{0 < i < j \leq k} A_{\Phi_i} \circ A_{\Phi_j} + \\
 &\quad \sum_{0 < i < j < u \leq H} A_{\Phi_i} \circ A_{\Phi_j} \circ A_{\Phi_u} + \cdots + (-1)^{H-1} \prod_{i=1}^k A_{\Phi_i}
 \end{aligned}$$

where A_{Φ_i} denotes the availability of placement group Φ_i and can be calculated from Eq. (4.4). Let us take Fig. 4.2b as an example. SFCs s-a-b-d and s-c-b-d are partially protected, since they jointly place VNF f_2 on node b and they traverse the same link (b, d) as well. Consequently, the total availability is calculated as $1 - (1 - A_{sa} \circ A_a \circ A_{ab} \circ A_b \circ A_{bd}) \circ (1 - A_{sc} \circ A_c \circ A_{cb} \circ A_b \circ A_{bd}) = A_a \circ A_b \circ A_d \circ A_{sa} \circ A_{ab} \circ A_{bd} + A_c \circ A_b \circ A_d \circ A_{sc} \circ A_{cb} \circ A_{bd} - A_a \circ A_b \circ A_d \circ A_{sa} \circ A_{ab} \circ A_{bd} \circ A_c \circ A_{cb} \circ A_{bd} = A_a \cdot A_b \cdot A_d \cdot A_{sa} \cdot A_{ab} \cdot A_{bd} + A_c \cdot A_b \cdot A_d \cdot A_{sc} \cdot A_{cb} \cdot A_{bd} - A_a \cdot A_b \cdot A_d \cdot A_{sa} \cdot A_{ab} \cdot A_{bd} \cdot A_c \cdot A_{cb} \cdot A_{bd} \approx 0.759$.

4.5 Problem Definition and Complexity Analysis

4.5.1 Delay-Sensitive VNF Scheduling

Formally, the Delay-Sensitive VNF Scheduling (DSVS) problem is defined as follows:

Definition 4.2 For a request $r(v, e, F, D)$, the Delay-Sensitive VNF Scheduling (DSVS) problem is to place all the requested VNFs according to their predefined order and route the traffic among these VNFs without violating the link bandwidth constraint and exceeding the end-to-end delay such that the number of used nodes is minimized.

4.5.2 Delay-Sensitive and Availability-Aware VNF Scheduling

On basis of the DSVS problem, we assume that the node availabilities and link availabilities are associated in the given network $\mathcal{G}(\mathcal{N}, \mathcal{L})$. A request $r(v, e, F, D, \delta)$

indicates the same meaning with the defined request $r(v, e, F, D)$ in the DSVS problem, except that we additionally use δ to denote the requested availability value. Moreover, we assume each requested VNF can be placed at most k nodes in order to achieve a certain VNF placement availability, that is, each requested VNF has $k - 1$ copies (k is given as an input). Formally, the Delay-sensitive and Availability-aware VNF Scheduling (DAVS) problem is defined as follows:

Definition 4.3 For a request $r(v, e, F, D, \delta)$, the Delay-sensitive and Availability-aware VNF Scheduling (DAVS) problem is to place at most k groups of the requested VNFs (k SFCs) according to their predefined order and route the traffic among these VNFs within each SFC, such that:

- The link bandwidth constraint is not violated.
- The end-to-end delay within each SFC is no greater than D .
- The VNF placement availability is no less than δ .
- The number of used nodes is minimized.

By minimizing the number of used nodes, we want to save network cost consumption in terms of servers. Since the addressed problem and later proposed solutions are general, the other metrics such as number of used links can also be used as the objective of the problem. In that case, both the exact solution and proposed heuristic proposed in Sect. 4.6 only need to be slightly changed in order to solve the problem. Moreover, for simplicity, we do not consider the ingress and egress nodes in the SFC in both problems in this chapter, since our focus is on finding a delay-sensitive SFC with and without resiliency concerns. We also do not consider the case when the traffic volume are changed during the packets are processed on different nodes (Ma et al. 2017), but our approach can be easily extended to this scenario.

4.5.3 Complexity Analysis

Remark 4.1 The DSVS problem and DAVS problem are NP-hard.

Proof When only the totally sequential SFC is considered and the delay constraint is not imposed, Ghaznavi et al. (2017) have proved that the DSVS problem is NP-hard. Since the DSVS problem is a special case of the DAVS problem (when $k = 1$ and $\delta = 0$), the DAVS problem is also NP-hard. The proof is therefore complete.

Remark 4.2 When all the VNFs are placed on network nodes and there is sufficient link bandwidth, the DSVS problem is polynomial-time solvable.

Proof When the link bandwidth constraint is not considered, for each two adjacent segments (m_i, m_j) and any two VNFs $f_i \in m_i$ which is hosted on node n_i and $f_j \in m_j$ which is located on node n_j , if $n_i \neq n_j$, we apply shortest path algorithm from n_i to n_j . By doing this, we find routes for every adjacent VNF pairs according to the orders in the SFC in the network. Therefore, the proof is complete.

4.6 Solutions

4.6.1 Exact Solution

In this subsection, we propose an exact INLP formulation to solve the DSVS and DAVS problem. We first solve the DSVS problem where $k = 1$ in the respective variables and start by some necessary notations and variables:

$r(v, e, F, D)$, $r(v, e, F, D, \delta)$: Request r for the DSVS problem and DAVS problem, respectively, and the details can be seen in Table 4.1.

\mathbb{B}_r :	The set of totally ordered sub-SFCs of request r .
$p_n^{f,h}$:	Processing delay if VNF f is placed on node n by the placement group h .
(m_i, m_j) :	Segment pairs that partition the whole SFC. Segment m_i contains F_i VNFs and Segment m_j contains F_j VNFs. VNFs pairs f_i and f_j which are dependent with each other in a SFC. That is, the packets which are processed by VNF f_i needs to traverse function f_j subsequently.
$X_n^{f,h}$:	A Boolean variable which returns 1 if the VNF f is placed on node $n \in \mathcal{N}$ by the h -th placement group, and 0 otherwise.
$Y_{f_i, f_j}^{l,h}$:	A Boolean variable which returns 1 if the flows between VNFs f_i and f_j traverse link l by the h -th placement group, and 0 otherwise.

Table 4.1 Notations

Notation	Description
$\mathcal{G}(\mathcal{N}, \mathcal{L})$	A network \mathcal{G} with node set \mathcal{N} and link set \mathcal{L}
$b(l), \theta(l), Cap(n)$	Bandwidth of link l , link attenuation parameter, maximum processing capacity of node n
$r(v, e, F, D)$	Request r , where v denotes the traffic volume that needs to be delivered through the SFC, e means the requested network transmitting rate, and F indicates the set of VNFs with orders in the DSVS problem.
$r(v, e, F, D, \delta)$	On basis of $r(v, e, F, D)$, δ indicates the requested VNF placement availability requirement in the DAVS problem.
k	Maximum times of a VNF can be placed on a network for the DAVS problem
\mathbb{B}_r	The set of totally ordered sub-SFCs of request r
$p_n^f, \eta(f)$	Processing delay if VNF f is placed on node n , required processing capacity of f
A_l, A_n	Availability of link l , node n .
M	A set of segments in a (partially ordered) SFC.
m_i, m_j	Two consecutive segments in a SFC.
$X_n^{f,h}, Y_{f_i, f_j}^{l,h}$	INLP Boolean variables, which identify whether f is placed on node n and whether the flows between f_i and f_j traverse link l by the h -th placement group, respectively.

Objective

$$\min \sum_{n \in \mathcal{N}} \max_{f \in F, 1 \leq h \leq k} X_n^{f,h} \quad (4.8)$$

Constraints

Routing Constraints

$$\begin{aligned} \sum_{(u,v) \in \mathcal{L}} Y_{f_i, f_j}^{(u,v),h} &= 1 \quad \forall 1 \leq h \leq k, \\ (m_i, m_j) : f_i &\in m_i, f_j \in m_j, \\ u \in \mathcal{N} : X_u^{f_i,h} &= 1 \& X_u^{f_j,h} \neq 1 \end{aligned} \quad (4.9)$$

$$\begin{aligned} \sum_{(u,v) \in \mathcal{L}} Y_{f_i, f_j}^{(u,v),h} &= 1 \quad \forall 1 \leq h \leq k, \\ (m_i, m_j) : f_i &\in m_i, f_j \in m_j, \\ v \in \mathcal{N} : X_v^{f_j,h} &= 1 \& X_v^{f_i,h} \neq 1 \end{aligned} \quad (4.10)$$

$$\begin{aligned} \sum_{(u,v) \in \mathcal{L}} Y_{f_i, f_j}^{(u,v),h} &= \sum_{(v,w) \in \mathcal{L}} Y_{f_i, f_j}^{(v,w),h} \\ \forall 1 \leq h \leq k, (m_i, m_j) : f_i &\in m_i, f_j \in m_j, \\ v \in \mathcal{N} : X_v^{f_i,h} &\neq 1 \& X_v^{f_j,h} \neq 1 \end{aligned} \quad (4.11)$$

Placement Constraint

$$\sum_{n \in \mathcal{N}, 1 \leq h \leq k} X_n^{f,h} \geq 1 \quad \forall f \in r.F \quad (4.12)$$

Link Bandwidth Constraint

$$\sum_{(m_i, m_j), 1 \leq h \leq k} Y_{f_i, f_j}^{l,h} \cdot e \leq b(l) \quad \forall l \in \mathcal{L} \quad (4.13)$$

Node Processing Capacity Constraint

$$\sum_{f \in F, 1 \leq h \leq k} X_n^{f,h} \cdot \eta(f) \leq Cap(n) \quad \forall n \in \mathcal{N} \quad (4.14)$$

Traversing Delay Constraint

$$\sum_{f \in B} \sum_{n \in \mathcal{N}} X_n^{f,h} \cdot p_n^f + \sum_{f_i, f_j \in B} \sum_{l \in \mathcal{L}} \left(\frac{v}{e} \cdot \theta(l) \cdot Y_{f_i, f_j}^{l,h} \right) \leq D$$

$$\forall 1 \leq h \leq k, B \in \mathbb{B}_r \quad (4.15)$$

$$\sum_{h=1}^k \left(\prod_{n \in \mathcal{N}} \min_{f \in F} (1 - X_n^{f,h} + X_n^{f,h} A_n) \cdot \prod_{l \in \mathcal{L}} \min_{m_i, m_j} (1 - Y_{f_i, f_j}^{l,h} + Y_{f_i, f_j}^{l,h} A_l) \right) -$$

$$\sum_{1 \leq g < h \leq k} \left(\prod_{n \in \mathcal{N}} \min_{f \in F} \left(\min(1 - X_n^{f,g} + X_n^{f,g} A_n), \min(1 - X_n^{f,h} + X_n^{f,h} A_n) \right) \right) \cdot$$

$$(4.16)$$

$$\sum_{1 \leq g < h \leq k} \left(\prod_{l \in \mathcal{L}} \min \left(\min_{m_i, m_j} (1 - Y_{f_i, f_j}^{l,h} + Y_{f_i, f_j}^{l,h} A_l), \min_{m_i, m_j} (1 - Y_{f_i, f_j}^{l,g} + Y_{f_i, f_j}^{l,g} A_l) \right) \right)$$

$$+ \dots + (-1)^{k-1} \left(\prod_{n \in \mathcal{N}} \min_{1 \leq h \leq k} \left(\min_{f \in F} (1 - X_n^{f,g} + X_n^{f,g} A_n) \right) \right.$$

$$\cdot \left. \prod_{l \in \mathcal{L}} \min_{1 \leq h \leq k} \left(\min_{m_i, m_j} (1 - Y_{f_i, f_j}^{l,h} + Y_{f_i, f_j}^{l,h} A_l) \right) \right) \geq \delta \quad (4.17)$$

$$\prod_{n \in \mathcal{N}} \min_{f \in F} (1 - X_n^{f,1} + X_n^{f,1} A_n) \cdot \prod_{l \in \mathcal{L}} \min_{m_i, m_j} (1 - Y_{f_i, f_j}^{l,1} + Y_{f_i, f_j}^{l,1} A_l) + \quad (4.18)$$

$$\prod_{n \in \mathcal{N}} \min_{f \in F} (1 - X_n^{f,2} + X_n^{f,2} A_n) \cdot \prod_{l \in \mathcal{L}} \min_{m_i, m_j} (1 - Y_{f_i, f_j}^{l,2} + Y_{f_i, f_j}^{l,1} A_l) \quad (4.19)$$

$$- \prod_{n \in \mathcal{N}} \min_{f \in F} \left(\min(1 - X_n^{f,1} + X_n^{f,1} A_n), \min(1 - X_n^{f,2} + X_n^{f,2} A_n) \right) \cdot$$

$$\prod_{l \in \mathcal{L}} \min_{m_i, m_j} \left(\min_{m_i, m_j} (1 - Y_{f_i, f_j}^{l,1} + Y_{f_i, f_j}^{l,1} A_l), \min_{m_i, m_j} (1 - Y_{f_i, f_j}^{l,2} + Y_{f_i, f_j}^{l,2} A_l) \right) \quad (4.20)$$

Equation (4.8) minimizes the number of used nodes. For instance, we first calculate the maximum value of $X_n^{f,h}$ for node n , and as long as $X_n^{f,h} = 1$ for some $1 \leq h \leq k$ and $f \in F$, it means that node n is in use to host VNF f . After that, we take the sum of $\max_{f \in F, 1 \leq h \leq k} X_n^{f,h}$ for all the nodes in \mathcal{N} and try to minimize this value. Equations (4.9)–(4.11) are the multicommunity constraints that account for the routing from a source to a destination and ensures to find a path from u to v . More specially, Eq. (4.9) ensures that if f_i in the segment m_i is placed on node u

and f_j is not placed on node u , then the sum of outgoing traffic from u is equal to 1. Equation (4.10) ensures that if f_j in the segment m_j is placed on node v and f_i is not placed on node v , then the sum of incoming traffic to v is equal to 1. Equation (4.11) ensures that if f_i and f_j are not placed on node v , then the sum of incoming traffic to v is equal to its outgoing traffic. Equation (4.12) ensures that each required VNF f must be placed on at least one node in the network. Equation (4.13) ensures that the total allocated bandwidth on each link does not exceed its maximum bandwidth. Equation (4.14) ensures that the total assigned processing capacity on each node does not exceed its maximum processing capacity. Equation (4.15) means that the total traversing delay of the SFC is no greater than the requested for both the totally (when $k = 1$) and the partially ordered SFC (when $k > 1$). More specifically, for each placement group h and each totally ordered sub-SFC B , $\sum_{f \in B, n \in N} X_n^{f,h} \cdot p_n^f$ calculates the total processing delay, and $\sum_{f_i, f_j \in B} \sum_{l \in \mathcal{L}} (\frac{v}{e} \cdot \theta(l) \cdot Y_{f_i, f_j}^{l,h})$ computes the total path traversing delay. Consequently, by letting each totally ordered sub-SFC of each placement group be less than D , the total traversing delay requirement is satisfied.

When solving the DAVS problem, we keep Eqs. (4.8)–(4.15) the same and add one more constraint in Eq. (4.17). Equation (4.17) ensures that the availability of total k groups of requested VNFs requirement is obeyed, and it is derived based on inclusive-exclusive principle. For example, when $k = 2$, Eq. (4.17) turns into Eq. (4.18). More specifically, in the first term of Eq. (4.18), $\prod_{n \in N} \min(1 -$

$X_n^{f,1} + X_n^{f,1} A_n)$ calculates the product of all the traversed nodes by group 1, and $\prod_{l \in \mathcal{L}} \min_{m_i, m_j} (1 - Y_{f_i, f_j}^{l,1} + Y_{f_i, f_j}^{l,1} A_l)$ calculates the product of all the traversed links by group 1. The second term in Eq. (4.18) follows similarly. In the third term

of Eq. (4.17), $\prod_{n \in N} \min \left(\min_{f \in F} (1 - X_n^{f,1} + X_n^{f,1} A_n), \min_{f \in F} (1 - X_n^{f,2} + X_n^{f,2} A_n) \right)$

calculates the distinct product of all the traversed nodes by group 1 and 2, and

$\prod_{l \in \mathcal{L}} \min \left(\min_{m_i, m_j} (1 - Y_{f_i, f_j}^{l,1} + Y_{f_i, f_j}^{l,1} A_l), \min_{m_i, m_j} (1 - Y_{f_i, f_j}^{l,2} + Y_{f_i, f_j}^{l,2} A_l) \right)$

calculates the distinct product of all the traversed links by group 1 and group 2. It is worthy to mention that our INLP can solve the DSVS and DAVS problem exactly. It means that it can return a solution with both VNF placement and how to route the traffic among those VNFs without violating the delay constraint (and the SFC placement availability) if any.

4.6.2 Heuristic

We emphasize that the route that packets traverse through a whole set of functions which are placed in the network is not necessarily a simple path. Especially with link bandwidth constraint, finding routes among VNFs is more difficult. Therefore, classical shortest path-based algorithms cannot apply well in the delay-sensitive

Algorithm 2: RVSA ($\mathcal{G}(\mathcal{N}, \mathcal{L}), M, r, H, TL$)

```

1  $T_x \leftarrow D, b(l) \leftarrow Cap(l), \pi[f][n] \leftarrow true, p[f][n] \leftarrow 0, \forall l \in \mathcal{L}, \forall f \in F, n \in \mathcal{N};$ 
2 while  $T_x \geq 0$  & Time limit TL is not reached do
3   foreach segment  $m \in M$  do
4     foreach VNF  $f \in m$  do
5       Place  $f$  on node  $n_x$  such that  $\pi[f][n_x] = true,$ 
         $T_x \geq p_{n_x}^f + \max(SP[n_x][N_y])$  and there are sufficient link bandwidth, where
         $N_y$  denotes the set of nodes that host VNFs in the previous segment  $prev[m]$ .
6       if Step 5 succeeds then
7          $p[f][n_x] \leftarrow 1;$ 
8          $T_x \leftarrow T_x - p_{n_x}^f - \max(SP[n_x][N_y]);$ 
9         Establish shortest paths  $\Psi_u$  between  $n_x$  and all the nodes in  $N_y$ .
10         $b(l) \leftarrow b(l) - e, \forall$  traversed link  $l \in \Psi_u$ .
11        if All the VNFs are placed then
12           $\text{Return } p$  and  $\Psi_u;$ 
13        else
14           $p[f_v][n_v] \leftarrow 0, \pi[f_v][n_v] \leftarrow false, \forall f_v \in m$  which is originally placed
            by node  $n_v$  in the segment  $m$ 
15           $m \leftarrow prev[m]$  and goes back Step 2.

```

16 Output there is no solution.

VNF placement and routing problem. As we will show in Sect. 4.7, the shortest path based heuristic performs poorly due to nature of the DSVS and DAVS problem.

Our proposed heuristic, called Recursive VNF Scheduling Algorithm (RVSA), is presented in Algorithm 2. The general idea of VNF is that it iteratively places VNFs per-segment, when it cannot place VNFs of segment m because of node processing capacity or delay violating, it goes back the previous iteration and place the VNFs again. This procedure continues until when all the VNFs are placed or the total running time is exceeded. We first present the details of RVSA to solve the DSVS problem. More specifically, in Step 1 of Algorithm 2, T_x represents the “remaining” allowed delay value, $b(l)$ denotes the available bandwidth of link l , $\pi[f][n]$ is an indicator which records whether VNF f can be placed on node n , and $p[f][n]$ stores the solution found by the algorithm which means whether VNF f has been placed on node n . We initialize these variables as shown in Step 1. As long as $T_x \geq 0$ and the time limit is not reached, Steps 2–15 are going to find an VNF placement and routing solution. More specifically, for each segment $m \in M$ and for each VNF $f \in m$, we first try to place f on a node with sufficient available processing capacity and also the induced delay composed of processing delay and (maximum) traversing delay is less than T_x . If there exists such a node n_x , we let $p[f][n_x] = 1$ and $T_x \leftarrow T_x - p_{n_x}^f - \max(SP[n_x][N_y])$ in Step 7 and 8, where N_y denotes the set of nodes that host VNFs in the previous segment $prev[m]$ and $SP[n_x][N_y]$ represents the shortest paths between n_x and the nodes in N_y . In Step 9, we establish shortest paths between n_x and all the nodes in N_y . We subsequently let each of the traversed

link's bandwidth be decreased by e in Step 10. If all the required VNFs are placed on network, it indicates that we find a feasible solution and return the result in Step 12. When Step 5 fails, it indicates that the algorithm cannot find a placement solution for placing VNFs belonging to Segment m . Therefore, we change the placement results in segment m by letting $p[f_v][n_v] \leftarrow 1$ and $\pi[f_v][n_v] \leftarrow false$ in Step 14. After that, we set $m \leftarrow prev[m]$ and the algorithm goes back to Step 2. In Step 16, when there is no feasible solution found by the algorithm, it outputs that there is no solution.

To solve the DAVS problem, we sequentially run the (slightly) modified RVSA algorithm at most k times. The only modification is that in Step 9, instead of running shortest path, RVSA applies the TAMCRA (Van Mieghem and Kuipers 2004), which is a heuristic multi-constrained path selection algorithm we use to find restricted shortest path. In the restricted shortest path problem, each link is associated with two weights (say delay and cost), and the problem is to find a path from the source to the destination such that the total delay is no greater than T and the total cost is minimized. The multi-constrained path selection problem is similar to the Restricted Shortest path problem, and it is assumed that there can be multiple additive link weights on each link. The goal is to find a path from a source to a destination such that the link weight sum in each dimension during the path is no greater than the specified. Since each link has link and availability weights in the DVAS problem, we employ TAMCRA to find a path with minimum delay and the product of link availabilities is less than a certain value κ . We let $\kappa = \delta$ and the number of stored paths for each node in TAMCRA is equal to 1 for simplicity. Consequently, we first run RVSA on the network and if it can return the solutions with satisfied delay constraint, we record the solution as P_1 . In particular, in Step 5, we always select a node with the largest availability. After that, we decrease the link bandwidth of all the links that are traversed by P_1 , and reduce the node's available processing capacity accordingly of all the nodes that are used to place VNFs. If P_1 cannot satisfy the VNF placement availability, we run RVSA again on the modified network and record the feasible solution as P_2 if any. This procedure continues until the VNF placement availability requirement is satisfied by the so-far found solutions or we run RVSA at most k times but there is no feasible solution for the DAVS problem. In all, the time complexity of RVSA to solve the DSVS problem and DAVS problem are $O(TL)$ and $O(TL \cdot k)$, respectively.

4.7 Simulations

4.7.1 Simulation Setup

We conduct simulations on two realistic carrier backbone networks (Shen and Tucker 2009), namely the NSFNet of 14 nodes and 21 links shown in Fig. 4.3 and the USANet of 24 nodes and 43 links displayed in Fig. 4.4. Each node has a

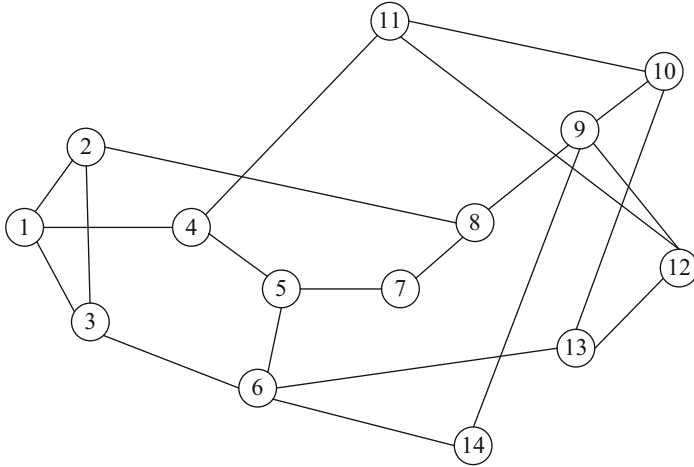


Fig. 4.3 NSFNet carrier backbone network

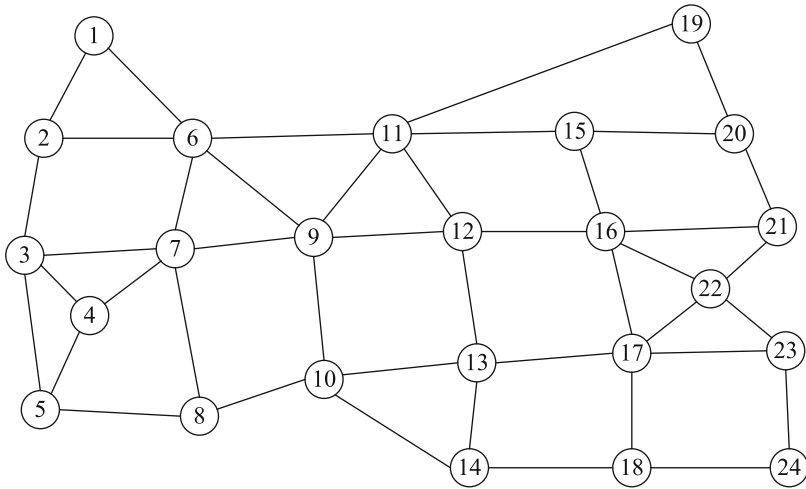


Fig. 4.4 USA carrier backbone network

processing capacity ranging from 10 to 15 units, and the node availability value is randomly distributed in the set of $\{0.999, 0.9995, 0.9999, 0.99999\}$. The link bandwidth is randomly generated from 30 to 50, and the link availability is randomly chosen from the set of $\{0.99, 0.999, 0.9999\}$. Moreover, the value of $\theta(l)$ falls in the range of $[20, 50]$. There are in total 20 different VNFs that can be requested in the service and the size of the VNF is randomly distributed between 5 and 10. The time for different node to process different VNF is randomly generated from 10 to 25 ms. For each request $r(v, e, F, D)$ in the DSVS problem and each request $r(v, e, F, D, \delta)$ in the DAVS problem which demand totally ordered SFCs,

$v \in [1, 10]$, $e \in [50, 100]$, $|F| \in [5, 10]$ and $D \in [100, 200]$. The traffic request setting is similar to Savi et al. (2015), and reflects the conventional application of e.g., Web Service or VoIP. Moreover, δ is randomly distributed in the set of $[0.95, 0.99, 0.995, 0.999, 0.9995, 0.9999]$, similar to Yang et al. (2017). The request which asks for partially ordered SFCs follow the same with the request generated for the totally ordered SFCs, but we let that each segment contains 2 VNFs and the number of segments ranges from 2 to 5. In the DSVS problem, we compare our proposed algorithms with two heuristics that are derived from literature:⁷

1. Traffic And Space Aware Routing (TASAR) (Ma et al. 2017): It firstly selects one node to place as many VNFs as possible, and then iteratively route to a nearby node with spaces until sufficient spaces have been accumulated for all the required VNFs.
2. Greedy Shortest Path (GSP) (Qu et al. 2017): For each node pair (s, t) in the network, it firstly finds z -shortest paths from s to t . After that, it iteratively places VNFs according to the requested SFC one by one on the nodes along each of these z paths. It selects one feasible placement solution which uses the minimum number of nodes from those z paths for each (s, t) pair. Finally, it returns the “best” feasible placement solution with the minimum node usage among $\frac{N \cdot (N-1)}{2}$ different (s, t) pairs of solutions⁸ if any. We set $z = 10$ for GSP in the simulation.

In the DAVS problem where we set $k = 2$, we first run TASAR or GSP in the network, if it returns a feasible solution with satisfied delay constraint (say P_1), we prune the nodes and links that are used by them by decreasing their node available processing capacity and link bandwidth, respectively. Subsequently, we apply TASAR or GSP again on the remainder of the network. The returned feasible solution in this time is denoted by P_2 . Finally, if P_1 and P_2 satisfy the availability requirement, then we return the solution, otherwise we regard that TASAR or GSP cannot find solutions for the DAVS problem. We randomly and respectively generate $R = 100, 200$, and 300 requests for both problems and for both totally ordered and partially ordered SFCs. In this sense, all the compared algorithms run R times⁹ for corresponding R requests in each scenario in order to collect the simulation results. For example, for each generated request the INLP runs one time on \mathcal{G} (the parameters in \mathcal{G} like link bandwidth keep the same for each time of algorithm’s running) and we evaluate the algorithm’s performance over all the requests.

The simulations are run on a high-performance desktop PC with 3.40GHz and 16 GB memory. We use IBM ILOG CPLEX 12.6 to implement the proposed INLP.

⁷ Most of current relevant literature deal with different problems in NFV or adopt different NFV models with us, so we have to modify two relevant proposed algorithms from Ma et al. (2017); Qu et al. (2017) in order to solve our proposed problems accordingly.

⁸ Since the returned solutions by GSP for node pair (s, t) and (t, s) are the same, we only need to consider possible solutions from $\frac{N \cdot (N-1)}{2}$ node pairs.

⁹ In this chapter, the proposed and compared algorithm take the input of single request to find corresponding VNF placement and traffic routing solution. However, we can also extend the proposed algorithms to simultaneously find solutions for multiple requests.

All the heuristics are implemented by C# and compiled on Visual Studio 2015 (using .NET Framework 4.5). In order that the INLP can return a feasible solution in a reasonable time, we set the time limit for the INLP to be 20 min for each request and the gap of the INLP is set to 0.02. Moreover, we let $TL = 1$ second for RVSA.

4.7.2 Simulation Results

4.7.2.1 Delay-Sensitive VNF Scheduling

We first evaluate the performance of the algorithms in terms of Acceptance Ratio (AR), which is defined as the number of accepted requests over all the requests (between 0 and 1). Here, the “accepted request” means that its SFC delay requirement is satisfied and each link’s consumed bandwidth is not exceeded. Figures 4.5a and 4.6a give the AR value returned by all the algorithms in NSFNet. More specifically, when totally ordered SFC is requested, in Fig. 4.5a, we find out that the INLP achieves the highest AR value, followed by our proposed heuristic RVSA with a slight difference. TASAR ranks the third, but its returned AR value is largely less than RVSA. This is because TASAR does not consider link delay

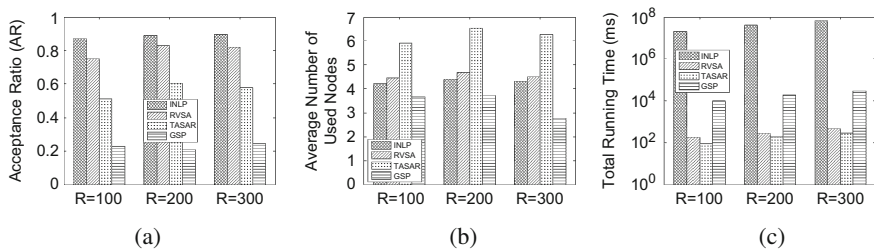


Fig. 4.5 Simulation results on NSFNet over different amount of requests with totally ordered SFC for the DSVS problem: (a) Acceptance ratio (AR), (b) Average number of used nodes (ANUN) and (c) Running time

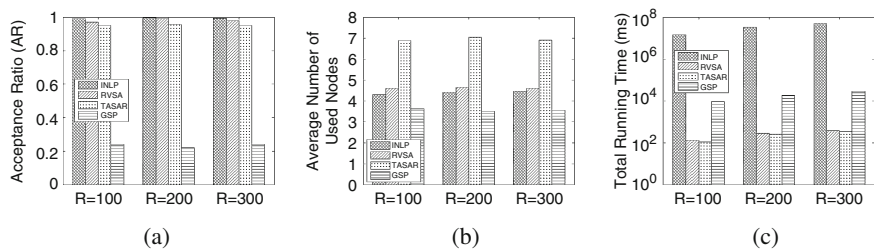


Fig. 4.6 Simulation results on NSFNet over different amount of requests with partially ordered SFC for the DSVS problem: (a) Acceptance ratio (AR), (b) Average number of used nodes (ANUN) and (c) Running time

factor when placing VNFs, which may violate the requested delay constraint and results in request blocking. GSP performs the worst. The reason is that although it first determines z -shortest paths which may have the least path delay, it ignores the VNF size and node processing capacity. This means that the VNF(s) cannot be placed on the node along the path whose maximum processing capacity is fewer than the VNF's. When partially ordered SFC is requested, the AR value achieved for all the algorithms except for GSP in this case (see Fig. 4.6a) is larger than the case for the totally ordered SFC (see Fig. 4.5a). This can be explained that there are at most 5 segments in the partially ordered SFC, so the total link delay value is the sum of at most 4 subpaths' delay (between 5 segment pairs) which is much less than the case in the totally ordered SFC with the sum of at most 9 subpaths' delay (between 10 segment pairs). Consequently, the total delay in the partially ordered SFC become less and hence becomes easier for the algorithms to find feasible solutions to satisfy the requested delay constraint. Nevertheless, INLP can accept almost all the requests, while RVSA and TASAR can achieve a close to optimal AR performance. GSP, on the other hand, still performs very poorly, due to the reasons as we analyzed above. We found that the results in USANet (Figs. 4.7 and 4.8) follow similarly with the ones in the NSFNet (Figs. 4.5 and 4.6). The reason is that both NSFNet and USANet are well connected and their parameters such as node's

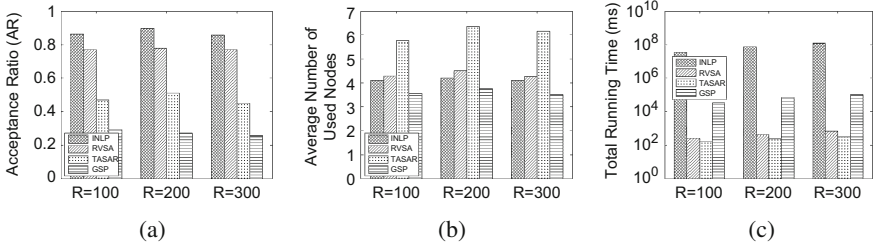


Fig. 4.7 Simulation results on USANet over different amount of requests with totally ordered SFC for the DSVS problem: (a) Acceptance ratio (AR), (b) Average number of used nodes (ANUN) and (c) Running time

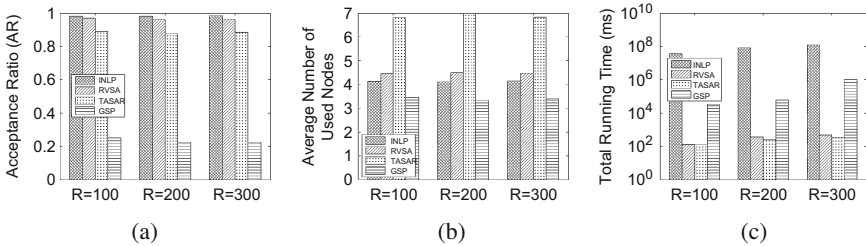


Fig. 4.8 Simulation results on USANet over different amount of requests with partially ordered SFC for the DSVS problem: (a) Acceptance ratio (AR), (b) Average number of used nodes (ANUN) and (c) Running time

maximum processing capacity are generated similarly. Hence, the routing aspect of variety in terms of topology does not affect the DSVS problem too much, since the solution is dominated by how to place the VNFs in the network in this context. For simplicity, in the following we only demonstrate the results for the algorithms in NSFNet.

Next, we compare the algorithms in terms of Average Number of Used Nodes (ANUN). The ANUN is defined as the total number of nodes consumed by the accepted requests divided by the number of accepted requests of the algorithm. In Figs. 4.5b and 4.6b, we see that the achieved ANUN value by GSP is the lowest. This is because its acceptance ratio in those scenarios is too low, and it only finds solutions for some “easier” requests. Except for those cases, the INLP achieves the minimum value of ANUN, and our proposed RVSA obtains the second lowest ANUN value. From above, we observe that even under the constrained simulation setup, the exact INLP can always accept most requests and consume the least amount of nodes as well, which validates its correctness.

Finally, Figs. 4.5c and 4.6c present the total running time for all the algorithms (in log scale). We found that the running time for the same algorithm increases as the number of traffic requests raises. The INLP is significantly more time-consuming than all the 3 heuristics. The GSP consumes more time than the other 2 heuristics, since it iterates $\frac{N \cdot (N-1)}{2}$ possible solutions (rounds) by finding $z = 10$ shortest paths in each solution. RVSA is only slightly higher than TASAR, but both of them are very quick. In addition, although we set $TL = 1$ second in RVSA, we found it returns a feasible solution within less than 1 s in most of the times. From above we see that, due to the nature that RVSA applies the recursive technique whose returned route is not necessarily a simple path, it can efficiently find feasible (close-to-optimal) solutions for the DSVS (and DAVS problem in below) in most of times at a reasonable short time.

4.7.2.2 Delay-Sensitive and Availability-Aware VNF Scheduling

In this subsection, we evaluate the performance of the algorithms to solve the DAVS problem. Due to the nonlinear constraint in Eq. (4.17) for the INLP, it becomes very time-consuming and keeps running for at least one day without returning a feasible solution. We therefore only evaluate the performance of heuristic algorithms. Due to the lack of the exact solution, we generate 100 sets of $R = 100, 200, 300$ traffic requests, respectively, and evaluate all the heuristic algorithms for those sets of traffic requests (100 runs). By doing this, we want to establish confidence on the performance of heuristics. Similar to Sect. 4.7.2.1, for all the algorithms, the results in the USANet (Figs. 4.11 and 4.12) follow similarly to the results in the NSFNet (Figs. 4.9 and 4.10), we therefore only describe and analyze the results in the NSFNet for all the heuristics for simplicity.

Figures 4.9 and 4.10 depict the AR, ANUN and running time (in log scale) of all these algorithms, where the confidence interval is set to 95%. The 95% confidence interval is calculated for all the figures, but in those where it is not

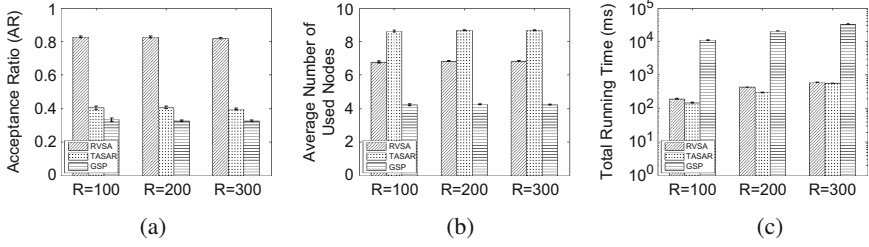


Fig. 4.9 Simulation results on NSFNet over 100 sets of different amount of requests with totally ordered SFC for the DAVS problem: (a) Acceptance ratio (AR), (b) Average number of used nodes (ANUN) and (c) Running time

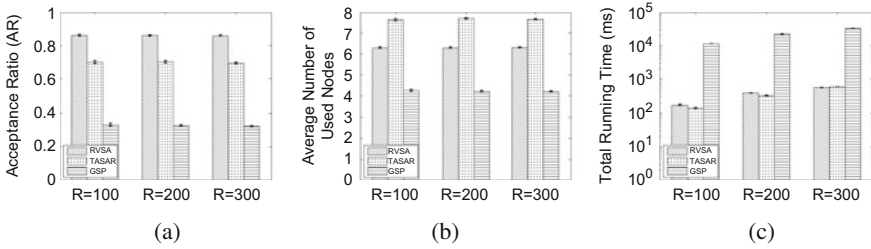


Fig. 4.10 Simulation results on NSFNet over 100 sets of different amount of requests with partially ordered SFC for the DAVS problem: (a) Acceptance ratio (AR), (b) Average number of used nodes (ANUN) and (c) Running time

visible, the interval is negligibly small.¹⁰ RVSA always achieves better performance than the other 2 heuristics in terms of AR (see Figs. 4.9a and 4.10a) and ANUN (see Figs. 4.9b and 4.10b). The reason can be explained that RVSA leverages TAMCRA (Van Mieghem and Kuipers 2004) which takes both availability and delay factors into account when finding a path. It indicates that adopting TAMCAR together with its recursive nature can make RVSA to efficiently solve DAVS problem as well. Different from Sect. 4.7.2.1, for the same algorithm, the AR value in Fig. 4.10a is not always (much) higher the AR value in Fig. 4.9a. This is due to the fact that apart from the delay requirement (constraint), the VNF placement availability requirement (constraint) is also imposed in the DAVS problem. In this sense, the acceptance of a request depends on satisfying both two requirements. Moreover, Figs. 4.9c and 4.10c show that GSP is more time consuming than the other heuristics, and our proposed heuristic RVSA is very comparable with TASAR with much quicker running time (Figs. 4.11 and 4.12).

In all, we conclude that the exact solution can always find optimal solution (if there exists), but this comes at the expense of much higher running time. Especially

¹⁰ We note here that some plots are log-scale that additionally contributes to the confidence interval visibility.

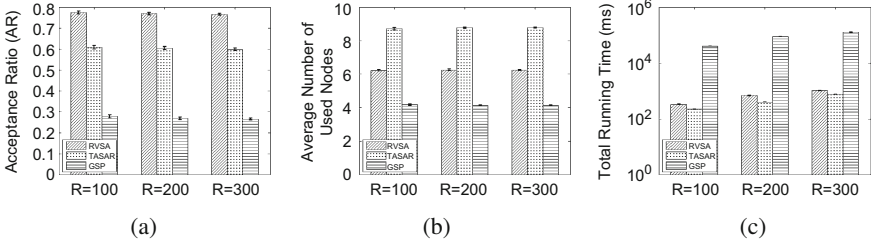


Fig. 4.11 Simulation results on USANet over 100 sets of different amount of requests with totally ordered SFC for the DAVS problem: (a) Acceptance ratio (AR), (b) Average number of used nodes (ANUN) and (c) Running time

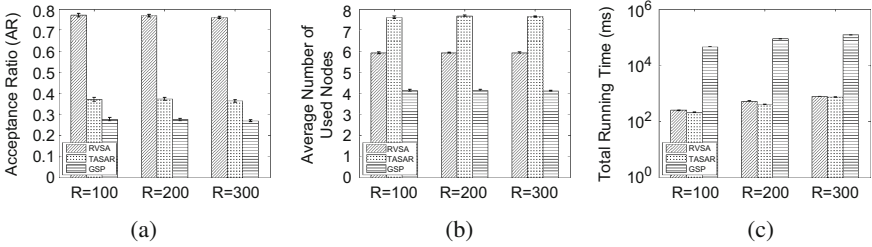


Fig. 4.12 Simulation results on USANet over 100 sets of different amount of requests with partially ordered SFC for the DAVS problem: (a) Acceptance ratio (AR), (b) Average number of used nodes (ANUN) and (c) Running time

when to solve the DAVS problem, the exact solution cannot return a solution even within quite a long time. Our proposed RVSA, can return a feasible solution in most of times at a much quicker time, which can be used when the traffic requests arrive in an online fashion.

4.8 Summary

In this chapter, we first investigate how to quantitatively model the traversing delay of a flow in a SFC and study how to calculate VNF placement availability mathematically for both the totally ordered SFC and partially ordered SFC. After that, we study the Delay-Sensitive VNF Scheduling (DSVS) problem and the Delay-Sensitive and Availability-aware VNF Scheduling (DAVS) problem. We have proved that both problems are NP-hard. To solve them, we propose an exact INLP and an efficient heuristic. Extensive simulations reveal that the exact INLP can always return feasible solutions if there exists, but its running time is significantly higher. On the other hand, our proposed heuristic RVSA, can achieve a close to optimal performance with a much quicker running time. In the next chapter, we plan to study the traffic routing when traffic demands vary in different time intervals.

References

- Allybokus Z, Perrot N, Leguay J, Maggi L, Gourdin E (2017) Virtual function placement for service chaining with partial orders and anti-affinity rules. *Networks* 71(2):97–106
- Beck MT, Botero JF, Samelin K (2016) Resilient allocation of service function chains. In: *IEEE conference on network function virtualization and software defined networks (NFV-SDN)*, pp 128–133
- Chen H, Wang X, Zhao Y, Song T, Wang Y, Xu S, Li L (2018) MOSC: a method to assign the outsourcing of service function chain across multiple clouds. *Comput Netw* 133:166–182
- Cohen R, Lewin-Eytan L, Naor JS, Raz D (2015) Near optimal placement of virtual network functions. In: *IEEE INFOCOM*
- Ding W, Yu H, Luo S (2017) Enhancing the reliability of services in nfv with the cost-efficient redundancy scheme. In: *IEEE international conference on communications (ICC)*, pp 1–6
- Dwaraki A, Wolf T (2016) Adaptive service-chain routing for virtual network functions in software-defined networks. In: *Proceedings of the 2016 workshop on hot topics in middleboxes and network function virtualization*, ser. *ACM HotMiddlebox*, pp 32–37
- Eramo V, Miucci E, Ammar M, Lavacca FG (2017) An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures. *IEEE/ACM Trans Netw* 25(4):2008–2025
- ETSI Publishes First Specifications for Network Functions Virtualisation. <http://www.etsi.org/news-events/news/700-2013-10-etsi-publishes-first-nfv-specifications>
- Fan J, Ye Z, Guan C, Gao X, Ren K, Qiao C (2015) Grep: guaranteeing reliability with enhanced protection in NFV. In: *ACM SIGCOMM workshop on hot topics in middleboxes and network function virtualization*, pp 13–18
- Feng H, Llorca J, Tulino AM, Raz D, Molisch AF (2017) Approximation algorithms for the NFV service distribution problem. In: *IEEE INFOCOM*
- Ford D, Labelle F, Popovici FI, Stokely M, Truong VA, Barroso L, Grimes C, Quinlan S (2010) Availability in globally distributed storage systems. In: *OSDI*, vol. 10, pp 1–7
- Franke U (2012) Optimal IT service availability: shorter outages, or fewer? *IEEE Trans Netw Serv Manag* 9(1):22–33
- Bhamare D, Jain R, Samaka M, Erbad A (2015) A survey on service function chaining. *J Netw Comput Appl* 75:138–155
- Ghaznavi M, Shahriar N, Kamali S, Ahmed R, Boutaba R (2017) Distributed service function chaining. *IEEE J Sel Areas Commun*. 35(11):2479–2489
- Gouareb R, Friderikos V, Aghvami A-H (2018) Virtual network functions routing and placement for edge cloud latency minimization. *IEEE J Sel Areas Commun* 36(10):2346–2357
- Guo L, Pang J, Walid A (2018) Joint placement and routing of network function chains in data centers. In: *IEEE INFOCOM*
- Han B, Gopalakrishnan V, Kathirvel G, Shaikh A (2017) On the resiliency of virtual network functions. *IEEE Commun Mag* 55(7):152–157
- Hantouti H, Benamar N, Taleb T, Laghrissi A (2018) Traffic steering for service function chaining. *IEEE Commun Surv Tutor* 21(1):487–507
- Herker S, An X, Kiess W, Beker S, Kirstaedter A (2015) Data-center architecture impacts on virtualized network functions service chain embedding with high availability requirements. In: *Globecom workshops (GC Wkshps)*. IEEE, pp 1–7
- Herrera JG, Botero JF (2016) Resource allocation in NFV: a comprehensive survey. *IEEE Trans Netw Serv Manag* 13(3):518–532
- Hmaity A, Savi M, Musumeci F, Tornatore M, Pattavina A (2016) Virtual network function placement for resilient service chain provisioning. In: *8th IEEE/IFIP international workshop on resilient networks design and modeling (RNDM)*, pp 245–252
- Kuo T-W, Liou B-H, Lin KC-J, Tsai M-J (2016) Deploying chains of virtual network functions: On the relation between link and server usage. In: *IEEE INFOCOM*

- Li Y, Phan LTX, Loo BT (2016) Network functions virtualization with soft real-time guarantees. In: IEEE INFOCOM, pp 1–9
- Li Q, Jiang Y, Duan P, Xu M, Xiao X (2017) Quokka: latency-aware middlebox scheduling with dynamic resource allocation. *J Netw Comput Appl* 78:253–266
- Ma W, Sandoval O, Beltran J, Pan D, Pissinou N (2017) Traffic aware placement of interdependent NFV middleboxes. In: IEEE INFOCOM
- McCool JI (2003) Probability and statistics with reliability, queuing and computer science applications. Taylor & Francis, London
- Medhat AM, Taleb T, Elmangoush A, Carella GA, Covaci S, Magedanz T (2017) Service function chaining in next generation networks: state of the art and research challenges. *IEEE Commun Mag* 55(2):216–223
- Mijumbi R, Serrat J, Gorricho J-L, Bouten N, De Turck F, Boutaba R (2016) Network function virtualization: state-of-the-art and research challenges. *IEEE Commun Surv Tutor* 18(1):236–262
- Pham C, Tran NH, Ren S, Saad W, Hong CS (2017) Traffic-aware and energy-efficient vNF placement for service chaining: joint sampling and matching approach. *IEEE Trans. Serv. Comput.* 13(1):172–185
- Qu L, Assi C, Shaban K (2016) Delay-aware scheduling and resource optimization with network function virtualization. *IEEE Trans Commun* 64(9):3746–3758
- Qu L, Assi C, Shaban K, Khabbaz MJ (2017) A reliability-aware network service chain provisioning with delay guarantees in nfv-enabled enterprise datacenter networks. *IEEE Trans Netw Serv Manag* 14(3):554–568
- Savi M, Tornatore M, Verticale G (2015) Impact of processing costs on service chain placement in network functions virtualization. In: IEEE conference on network function virtualization and software defined network (NFV-SDN), pp 191–197
- Shen G, Tucker RS (2009) Energy-minimized design for IP over WDM networks. *IEEE/OSA J Opt Commun Netw* 1(1):176–186
- Sun C, Bi J, Zheng Z, Yu H, Hu H (2017) NFP: enabling network function parallelism in NFV. In: ACM SIGCOMM, pp 43–56.
- Van Mieghem P, Kuipers FA (2004) Concepts of exact QoS routing algorithms. *IEEE/ACM Trans Netw* 12(5):851–864
- Verma DC (2004) Service level agreements on IP networks. *Proc IEEE* 92(9):1382–1388
- Vishwanath, KV, Nagappan N (2010) Characterizing cloud computing hardware reliability. In: Proc. of the 1st ACM symposium on cloud computing, pp 193–204
- Vizarreta P, Condoluci M, Machuca CM, Mahmoodi T, Kellerer W (2017) QoS-driven function placement reducing expenditures in NFV deployments. In: IEEE ICC
- Yang S, Trajanovski S, Kuipers FA (2015) Availability-based path selection and network vulnerability assessment. *Networks* 66(4):306–319
- Yang S, Wiedner P, Yahyapour R, Trajanovski S, Fu X (2017) Reliable virtual machine placement and routing in clouds. *IEEE Trans Parallel Distrib Syst* 28(10):2965–2978
- Yang S, Li F, Yahyapour R, Fu X (2022) Delay-sensitive and availability-aware virtual network function scheduling for NFV. *IEEE Trans Serv Comput* 15(1):188–201
- Yi B, Wang X, Li K, Huang M et al. (2018) A comprehensive survey of network function virtualization. *Comput Netw* 133:212–262
- Zhang Z, Li Z, Wu C, Huang C (2017) A scalable and distributed approach for NFV service chain cost minimization. In: IEEE ICDCS, pp 2151–2156
- Zhang J, Wu W, Lui J (2018) On the theory of function placement and chaining for network function virtualization. In: Proceedings of the eighteenth ACM international symposium on mobile ad hoc networking and computing, pp 91–100.

Chapter 5

Traffic Routing in Stochastic Network Function Virtualization Networks



While virtual network function scheduling in the previous chapter assume deterministic link delay and bandwidth, the real-life network usually behaves in a stochastic manner, due to e.g., inaccurate data, expired exchanged information, insufficient estimation to the network. Motivated by this, we consider the stochastic NFV networks in this chapter, where the link bandwidth and delay are assumed to be random variables and their cumulative distribution functions are known.¹ We first study how to calculate the delay and bandwidth value in an SFC such that their realizing probabilities are satisfied. Subsequently, we formally define the traffic routing problem in stochastic NFV networks and prove it is NP-hard. We present an exact solution and a tunable heuristic to solve this problem. The proposed heuristic is a sampling-based algorithm, and it leverages the Tunable Accuracy Multiple Constraints Routing Algorithm (TAMCRA) to find a multi-constraint path for each adjacent VNF pair. It dynamically adjusts the link weights as well as delay and bandwidth realizing probability constraint after finding the path for each VNF pair so that the cumulated probabilities will not violate the specified values. Finally, we evaluate the performance of the proposed algorithms via extensive simulations.

5.1 Introduction

Nowadays, data communication networks have been witnessing exponential growth in users' data traffic. According to the cisco annual Internet report (Cisco 2020), the global average broadband speed continues to grow and will more than double from 2018 to 2023, from 45.9Mbps to 110.4Mbps. In the traditional network services provisioning paradigm, network functions (e.g., firewall or web proxy) which are also called middleboxes are usually implemented by dedicated hardware

¹ Parts of this chapter is reprinted from Yang et al. (2020), with permission from Elsevier.

appliances. Needless to say, it is costly to deploy these hardware middleboxes due to their high design and production cost and also these middleboxes need to be configured and managed manually, which further increases the costs of service providers. Therefore, the traditional network service paradigm fails to keep pace with satisfying the ever-increasing users' QoS requirements from the perspective of CAPital EXpenditures (CAPEX) and OPERational EXpenditures (OPEX), which poses a big challenge to network service providers.

Network Function Virtualization (NFV) which is first proposed by European Telecommunications Standards Institute (ETSI) (ETSI) in 2012 has emerged as an appealing solution, since it offers replacement of dedicated hardware implementations with software instances running in a virtualized environment. In NFV, the requested service is implemented by a sequence of Virtual Network Functions (VNF) that can run on generic servers by leveraging the virtualization technology. Service Function Chaining (SFC) is therefore defined as a chain-ordered set of placed VNFs that handles the traffic of the delivery and control of a specific application (Medhat et al. 2017). NFV is therefore able to allocate network resources in a more scalable and elastic manner, offer a more efficient and agile management and operation mechanism for network functions, leading to a large reduction of the overall costs.

Traffic routing (and VNF placement)² in NFV networks is a fundamental issue to construct an SFC without violating the QoS requirement, and it has been extensively studied in the recent works (Herrera and Botero 2016; Yi et al. 2018; Hantouti et al. 2018) where the link weight is assumed to be deterministic. However, in many real-life networks (e.g., data communication networks, wireless sensor networks Krishnan et al. 2018), the link weight such as bandwidth and delay usually varies and is uncertain (Yang and Kuipers 2014). These uncertainties (Guerin and Orda 1999; Korkmaz and Krunz 2003) usually arise from inaccurate data, expired exchanged information or insufficient estimation to the network. For example, especially in large networks, it is difficult to obtain an accurate view on the link characteristics like bandwidth utilization or latency, because their dynamics are usually of the same order as the time it would take to distribute information on the link state throughout the network (Yang and Kuipers 2014). Similarly in Software Defined Networks (SDN) (Nunes et al. 2014), the controller collects network statistics periodically to achieve the link conditions in the network. The dynamics such as configuration changes (Reitblatt et al. 2012) may lead to inaccurate/stochastic network state information in terms of delay or bandwidth. Another example is that the delay and available bandwidth are affected by diurnal patterns, interference in wireless networks (Krishnan et al. 2019), or by failure and maintenance events.

² As we will show later, the traffic routing problem alone in (stochastic) NFV networks is already NP-hard, hence jointly considering traffic routing and VNF placement will be more difficult to solve. We therefore only consider the traffic routing problem in stochastic NFV networks in this chapter.

Based on the above concerns, we assume the link delay and bandwidth are random variables (stochastic) and assume their cumulative distribution functions (CDF) are known, which can be easily achieved based on historical data. Suppose that the VNFs are placed in the network, we investigate how to construct an SFC by finding the appropriate path for each requested VNF pair. To the best of our knowledge, this work is the first to address the traffic routing problem in stochastic NFV networks. Our key contributions are as follows:

- We mathematically model the stochastic link weight in terms of delay and bandwidth. More specifically, given that the link delay and bandwidth follow a known distribution, we show how to calculate the delay and bandwidth in an SFC such that their realizing probabilities satisfy the required values.
- We define the Traffic Routing problem in Stochastic NFV Networks and prove it to be NP-hard. We further formulate this problem as an exact optimization solution.
- We devise a tunable heuristic that first discretizes the network by k -samplings, and then dynamically applies Tunable Accuracy Multiple Constraints Routing Algorithm (TAMCRA) to find the multi-constraint path for each adjacent VNF pair.

The remainder of this chapter is organized as follows. Section 5.2 presents the related work. Section 5.3 introduces stochastic link weight model. Section 5.4 defines the Traffic Routing in Stochastic NFV Networks (TRSN) problem and presents an exact solution to formulate it. In Sect. 5.5, we propose a tunable sampling-based heuristic to solve the TRSN problem. Section 5.6 provides the simulation results and we conclude in Sect. 5.7.

5.2 Related Work

In this section, we will divide and present the related work about traffic routing and VNF placement in both deterministic and stochastic NFV networks. In deterministic NFV networks, we will further present the related work on three categories, namely, (1) traffic routing in deterministic NFV networks, where VNFs are assumed to be placed on networks, (2) VNF Placement in Deterministic NFV Networks, where the path between each node pair is known, (3) (jointly) VNF Placement and Routing in Deterministic NFV Networks.

5.2.1 *Traffic Routing and VNF Placement in Deterministic NFV Networks*

A comprehensive survey about NFV can be found in (Bhamare et al. 2016; Mijumbi et al. 2016; Yi et al. 2018). Herrera and Botero (2016), Mirjalily (2018) provides a survey about resource allocation in NFV networks and (Hantouti et al. 2018) presents a survey about traffic steering in NFV networks. Laghrissi and Taleb (2019) present a survey of Virtual Machine placement and VNF placement.

5.2.1.1 Traffic Routing in Deterministic NFV Networks

Suppose that the VNFs are placed in networks, the traffic routing problem in NFV networks refers to find a path among each requested VNF pair. This problem can be proved to be NP-hard by a reduction to the NP-hard Hamiltonian path problem (Garey and Johnson 1979). Therefore, only approximation algorithm or heuristic can exist to solve it in polynomial time. Dwaraki and Wolf (2016) devise a layered-graph based heuristic to solve the delay-aware traffic routing problem in NFV networks. Wang et al. (2017) propose a distributed Alternating Direction Method of Multipliers (ADMM) algorithm to solve the traffic routing problem in NFV networks for multiple requests. The proposed algorithm in (Wang et al. 2017) is proved to have a fixed coverage rate. Yu et al. (2017) propose a Fully-Polynomial Time Approximation Scheme (FPTAS)³ to find the multipath between each VNF pair such that during an arbitrary single service failure, at most a portion of bandwidth is lost for each request. Gao and Rouskas (2019) present an online competitive traffic routing algorithm based on the Shortest Path Tour problem to minimize the congestion.

5.2.1.2 VNF Placement in Deterministic NFV Networks

When the path between each node pair is known/fixed, Ma et al. (2017) study the VNF placement problem to minimize the maximum link load in three cases: (1) when there is no dependency between VNFs, (2) there is a total dependency order on the VNF set, and (3) there exists dependency among a subset of VNFs. They propose a polynomial-time algorithm for case (1), and prove that cases (2) and (3) are NP-hard. To solve them, a dynamic programming and an efficient heuristic are proposed to solve (2) and (3), respectively. Pham et al. (2017) propose a sampling-based Markov approximation algorithm to jointly minimize the operational and network traffic costs for the VNF placement problem. Kuo et al. (2016) relax/approximate

³ An FPTAS has a time complexity that is polynomial in both the problem size and $\frac{1}{\epsilon}$ and produces a solution that is within a factor $(1 + \epsilon)$ of the optimal solution (or $(1 - \epsilon)$ for maximization problems).

the VNF placement problem based on the intuition that placing VNFs on a shorter path consumes less link bandwidth, but might also reduce VM reuse opportunities; reusing more VMs might lead to a longer path, and so it consumes more link bandwidth. Tomassillik et al. (2018) study the VNF placement problem with the aim of minimizing total costs for servicing a set of requests. By transforming this problem into a hitting-cut problem, Tomassillik et al. (2018) propose two logarithmic factor approximation algorithms. The first algorithm is based on LP rounding and the second one is a greedy algorithm. Marotta et al. (2017) devise a three-phase heuristic for the robust power-aware VNF placement problem by considering the uncertainty of the demand. Song et al. (2019) study the VNF placement problem in 5G edge networks by considering the user's mobility. They Song et al. (2019) first propose a user grouping model based on users' context geographical information and then define (and compute the optimum number of the) clusters to minimize the end-to-end delay of network services. Subsequently, a graph partitioning algorithm assigning VNFs to clusters in the edge network is presented to minimize user movement between clusters and optimize the data rate that users lose due to VNF migration.

5.2.1.3 VNF Placement and Routing in Deterministic NFV Networks

As we see above, only consider traffic routing or VNF placement in NFV networks has already been NP-hard to solve, therefore jointly considering the VNF placement and routing in NFV networks will make this problem ever harder. Ma et al. (2017) prove that jointly considering the VNF placement and routing problem is NP-hard even under the non-ordered VNF dependence case by a reduction to the NP-hard Hamiltonian Cycle problem (Garey and Johnson 1979). Guo et al. (2018) propose a randomized approximation algorithm when the traffic matrix is known in advance and a competitive online algorithm when the future arriving traffic is not known. However, they assume that one configuration in data centers consists of one VNF placement and one routing path solution, and a (limited) set of configurations is given. Qu et al. (2016) consider the VNF transmission and processing delays, and formulate the VNF Placement and Routing problem as a Mixed Integer Linear Program (MILP). However, they assume that the virtual link between two physical nodes can at most process one traffic flow at a time. Li et al. (2017) address the latency-aware middlebox routing and placement problem by leveraging a packet queuing model. However, a fixed link transmission delay is assumed in their model. Hamann and Fischer (2019) formulate the VNF placement and routing problem as an ILP where a set of k paths between each node pair in the network are precalculated and known. Yang et al. (2021) propose an approximation algorithm to solve the VNF placement and routing problem in edge clouds. The proposed approximation in (Yang et al. 2021) leverages randomized rounding technology and assumes that the paths between each node pair are known/given.

Apart from that, there is also work about availability-aware VNF provisioning (Fan et al. 2018; Yang et al. 2022), VNF placement in operator data centers (Tang

et al. 2019), the resource allocation of NFV in 5G mobile networks (Blanco et al. 2017), IoT (Fu et al. 2019) and Radio Access Networks (RAN) (Garcia-Saavedra et al. 2018), etc. Due to the complex nature of the VNF placement and/or traffic routing problem in NFV networks, we observe that it is difficult to design a general approximation algorithm to solve it. Instead, the current literature mainly simplify some problem inputs or constraints (e.g., assuming the paths between node pair are known Yang et al. 2021 or multipath routing Yu et al. 2017) and devise proper approximation/heuristic algorithms. Nevertheless, all the above work assume deterministic link weight, therefore they cannot solve the proposed problem in this chapter.

5.2.2 *Traffic Routing and VNF Placement in Stochastic (NFV) Networks*

So far, the above literature assume that the link delay and bandwidth are deterministic, and this assumption is not always very true in practice. The reason is that in practice, the network usually behaves in a stochastic manner, which is mainly caused by inaccurate data, expired exchanged information or insufficient estimation (Guerin and Orda 1999; Korkmaz and Krunz 2003). In what follows, we will provide the related work about routing and VNF placement on stochastic (NFV) networks:

Traffic Routing in Stochastic Networks A survey about traffic planning models and routing algorithms in stochastic networks can be found in (Yang and Kuipers 2014). Lorenz and Orda (1998) assume that each link l has a function $p_l(d)$ that represents the probability that link (l) introduces a delay of no more than d time units. This so-called Delay-Based Routing (DBR) problem is to find a path that has the biggest probability of not exceeding D . Lorenz and Orda prove that the DBR problem is NP-hard, and by decomposing the end-to-end delay constraint D into local delay constraints, they manage to develop an FPTAS. However, the proposed algorithms in (Lorenz and Orda 1998) cannot solve the proposed problem in this chapter. The reason is that it is required to find a path for each VNF pair and also the bandwidth requirement should be taken into account. Assuming the link's bandwidth and delay follows a log-concave distribution, we Kuipers et al. (2014) propose a polynomial-time convex optimization formulation to find the maximum flow in the so-called stochastic networks. When the delay constraint is imposed on each path, the maximum flow problem is NP-hard. To solve it, we Kuipers et al. (2014) propose an approximation algorithm and a tunable heuristic algorithm. However, the proposed approximation algorithm in (Kuipers et al. 2014) is to find multipath and hence cannot solve the considered problem in this chapter accordingly.

VNF Placement in Stochastic NFV Networks The most relevant work with us is Cheng et al. (2018), but it assumes that the service rate demands and available amounts of wireless resources of nodes are random and tackles the VNF placement problem in the so-called dynamic networks. Cheng et al. (2018) further develop a distributed computing framework with two-level decomposition to solve this problem. Zeng et al. (2018) present an online VNF placement and data packets scheduling framework in edge clouds by applying the Lyapunov optimization theory. However, the routing issues are not considered in (Zeng et al. 2018). While our work is on traffic routing in stochastic networks, and therefore this chapter is orthogonal with (Cheng et al. 2018; Zeng et al. 2018).

Moreover, Miao et al. (2019) propose an analytical model based on stochastic network calculus to calculate the delay bound of the stochastic NFV networks. They further leverage the property of convolution associativity and leftover service technologies to calculate the available resources for VNF networks. However, Miao et al. (2019) tackles the resource allocation problem in stochastic NFV networks from the system side, and routing issues are not considered.

In all, we conclude that the traffic routing problem in stochastic NFV networks has never been tackled by the existing literature, which remains our contribution in this chapter. To that end, we first mathematically model the stochastic link delay and bandwidth in stochastic NFV networks and present an exact algorithm as well as a tunable heuristic MCTR for this problem. We also conduct simulations to evaluate the performance of the proposed algorithms.

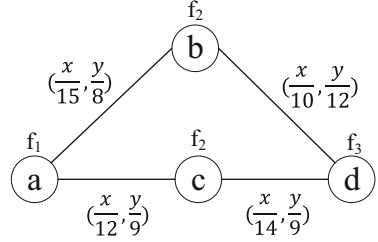
5.3 Stochastic Link Weight

For each link $l \in \mathcal{L}$, we assume that its delay and bandwidth are stochastic and follow a given distribution. More specifically, it is assumed that the allocated bandwidth has a known cumulative distribution function $c_l(b)$, which indicates the probability of being able to allocate b units of bandwidth; the allocated delay follows a known cumulative distribution function $p_l(d)$, which gives the probability of transporting data with no more than d_l units of delay. We assume that the bandwidth allocated on each link l ranges from 0 to b_l^{\max} , and the delay on each link l ranges from $d_l^{\min} > 0$ to d_l^{\max} . For ease of notation we will sometimes write c_l and p_l to denote $c_l(b_l)$ and $p_l(d_l)$. The traffic request set is denoted by R , and for each $r(F, B, P_B, D, P_D) \in R$, F symbolizes a set of required ordered VNFs, P_B refers to the probability to realize B available bandwidth, and P_D represents the probability of realizing a total traversing delay D . Consequently, by finding an appropriate path between each VNF pair which is located on different nodes (if the VNF pair are placed on the same node, then it is not necessary to find path), we get an SFC for request r . The notations used in this chapter are summarized in Table 5.1.

Let us consider the example from Fig. 5.1, where f_1 is placed on node a , f_2 is placed on node b and c , and f_3 is placed on node d . The allocated link bandwidth x and delay y are assumed to follow a uniform distribution,

Table 5.1 Notations

Notation	Description
$\mathcal{G}(\mathcal{N}, \mathcal{L})$	A network \mathcal{G} with node set \mathcal{N} and link set \mathcal{L}
$c_l(b)$	CDF of being able to allocate b units of bandwidth
$p_l(d)$	CDF of transporting data with d units of delay
R	Traffic request set and for each $r(F, B, P_B, D, P_D) \in R$, F symbolizes a set of required ordered VNFs, P_B refers to the probability to realize B available bandwidth, and P_D represents the probability of realizing a total traversing delay D
$\pi(f)$	The set of nodes on which VNF f is placed
$\Psi(n)$	The set of VNFs placed on n
k	The number of samplings used in the heuristic
q	The number of stored path per node in MCTR, where $q = ck$
$W_{u,v}^{r,f_i,f_j}$	A Boolean variable and it is 1 (true) if r 's requested VNF pair (f_i, f_j) traverses link (u, v) , and 0 (false) otherwise
$Z_{u,v}^{r,f_i,f_j}$	A float variable indicating the allocated delay on link (u, v) for r 's requested VNF pair (f_i, f_j)

Fig. 5.1 An Example of illustrating stochastic link bandwidth and delay

and their CDFs are labelled above each link. Now, suppose there is a request $r(\{f_1, f_2, f_3\}, 5, 0.35, 16, 0.7)$. According to Fig. 5.1, if we follow the path $a - b - d$, then the probability of realizing available bandwidth of **at least** 5 is $(1 - \frac{5}{15}) \cdot (1 - \frac{5}{10}) \approx 0.33 < P_B$, and the probability of realizing a delay **at most** 16 is to solve the following equation:

$$\begin{aligned} \max \quad & \frac{y_1 \cdot y_2}{96} \\ \text{subject to: } & y_1 + y_2 \leq 16 \end{aligned} \quad (5.1)$$

The objective is achieved when $y_1 = y_2 = 8$ (theoretically can be derived from the inequality of geometric arithmetic Hardy et al. 1934) which approximately leads to realizing probability of $\frac{8 \cdot 8}{96} \approx 0.67 < P_D$.

However, both the requested P_B and P_D are violated if using this path, which indicates that this is not a feasible path to satisfy r . Similarly, we can calculate that if we choose the path $a - c - d$, the probability of realizing available bandwidth of **at least** 5 is $(1 - \frac{5}{12}) \cdot (1 - \frac{5}{14}) \approx 0.375 > P_B$ and the probability of realizing a delay of **at most** 16 is approximately 0.81, which is greater than P_D . As a result, we see that $a - c - d$ constitutes a feasible SFC and can satisfy r .

5.4 Problem Definition and Formulation

5.4.1 Problem Definition and Complexity Analysis

We consider a network $G(\mathcal{N}, \mathcal{L})$, where \mathcal{N} represents a set of N nodes and \mathcal{L} denotes a set of L links. For each link $l \in \mathcal{L}$ the allocated bandwidth has a known CDF $c_l(b_l)$, which gives the probability of being able to allocate no more than b_l units of bandwidth. Moreover, if the possible bandwidth allocated on each link l ranges from 0 to b_l^{\max} ($0 < b_l^{\max}$), then the probability of allocating a bandwidth out of this range is 0, i.e. $c_l(b_l^{\max}) = 1$.

A certain number of VNFs are placed on network nodes, and for each $n \in \mathcal{N}$, we denote $\Psi(n)$ as the set of VNFs placed on it. Formally, the Traffic Routing in Stochastic NFV Networks (TRSN) problem can be defined as follows:

Definition 5.1 For each request $r \in R$, the TRSN problem is to find route in each required VNF pair such that:

- The path between each VNF pair has **at least** B bandwidth and the total probability to realize the bandwidth is **no less** than P_B .
- The total traversing delay in an SFC is **at most** D and the probability to realize the delay is **no less** than P_D .

Theorem 5.1 *The TRSN problem is NP-hard.*

Proof Let us first introduce the NP-hard Hamiltonian path problem (Garey and Johnson 1979). A Hamiltonian path is a path in a graph that visits each node exactly once. For simplicity, we assume that the allocated bandwidth x on each link l follows

$$x = \begin{cases} B & \text{with probability}=1 \\ 2B & \text{with probability} < 1 \end{cases} \quad (5.2)$$

and the allocated delay y on each link l follows

$$y = \begin{cases} D_l & \text{with probability}=1 \\ D_l/\alpha & \text{with probability} < 1 \end{cases} \quad (5.3)$$

where $\alpha_l > 1$ symbolizes a coefficient for each link. We assume $N = |F|$, and on each node there is one distinct VNF placed on it. Now, suppose for a request $r(\{f_1, f_2, \dots, f_N\}, B, 1, D, 1)$ where D is set to $\sum_{l \in \mathcal{L}} D_l$, then the TRSN problem is equivalent to the Hamiltonian path problem. The proof is therefore complete.

5.4.2 Problem Formulation

In this section, we present an exact solution to formulate the TRSN problem. We start with some necessary notations and variables.

- R : Traffic request set R .
 $\pi(f)$: The set of nodes on which the VNF f is placed.
 f_i, f_j : The required VNF pair.
 $W_{u,v}^{r,f_i,f_j}$: A Boolean variable and it is 1 (true) if r 's requested VNF pair (f_i, f_j) traverses link (u, v) , and 0 (false) otherwise.
 $Z_{u,v}^{r,f_i,f_j}$: A float variable indicating the allocated delay on link (u, v) for r 's requested VNF pair (f_i, f_j) .

Constraints

Routing Constraints

$$\sum_{u \in \pi(f_i): (u,v) \in \mathcal{L}} W_{u,v}^{r,f_i,f_j} = 1 \quad \forall r \in R, (f_i, f_j) \in r, r \in R \quad (5.4)$$

$$\sum_{v \in \pi(f_j): (u,v) \in \mathcal{L}} W_{u,v}^{r,f_i,f_j} = 1 \quad \forall r \in R, (f_i, f_j) \in r, r \in R \quad (5.5)$$

$$\sum_{(u,v) \in \mathcal{L}: v \notin \pi(f_i) \cup \pi(f_j)} W_{u,v}^{r,f_i,f_j} = \sum_{(v,w) \in \mathcal{L}: v \notin \pi(f_i) \cup \pi(f_j)} W_{v,w}^{r,f_i,f_j} \quad \forall r \in R, (f_i, f_j) \in r \quad (5.6)$$

Delay Constraints

$$W_{u,v}^{r,f_i,f_j} \cdot M \geq Z_{u,v}^{r,f_i,f_j} \quad \forall r \in R, f_i, f_j \in r, (u, v) \in \mathcal{L} \quad (5.7)$$

$$\sum_{f_i, f_j \in r} \sum_{(u,v) \in \mathcal{L}} Z_{u,v}^{r,f_i,f_j} \leq r.D \quad \forall r \in R \quad (5.8)$$

$$\prod_{(u,v) \in \mathcal{L}} \max_{f_i, f_j \in r} p_{u,v}(Z_{u,v}^{r,f_i,f_j}) \geq P_D \quad \forall r \in R \quad (5.9)$$

Bandwidth Constraint

$$\prod_{f_i, f_j \in r} \prod_{(u,v) \in \mathcal{L}} (1 - c_{u,v}(r.B)) \geq P_B \quad \forall r \in R \quad (5.10)$$

Link Capacity Constraint

$$\sum_{r \in R, f_i, f_j \in r} W_{u,v}^{r, f_i, f_j} \cdot r.B \leq b_{(u,v)}^{max} \quad \forall (u, v) \in \mathcal{L} \quad (5.11)$$

There is no objective (needed) in the proposed exact formulation, but one could include the objective of e.g., minimizing the number of links used. Equations (5.4)–(5.6) are the multicommunity constraints that account for the routing from a source to a destination and ensures to find a path from the node where f_i is located to the node where f_j is placed. More specially, for each VNF pair f_i and f_j of request r , Eq. (5.4) ensures that if f_i is placed on node u and r selects u as the node to place f_i and transfer data (since more than 1 nodes can host f_i in the network), then the sum of outgoing traffic from u is equal to 1. Equation (5.5) ensures that if f_j is placed on node v and r selects v as the node to place f_j and transfer data (since more than 1 nodes can host f_j in the network), then the sum of incoming traffic to v is equal to 1. Equation (5.6) ensures that for any intermediate node v which does not host f_i and f_j (by setting $v \notin \pi(f_i) \cup \pi(f_j)$), the sum of incoming traffic to v is equal to its outgoing traffic. Equation (5.7) applies big M -method to set the relation between W and Z . More specifically, by setting M as a large enough number, when Z is greater than 0, then W has to be 1, otherwise when Z is equal to 0, then W is forced to be 0. Equation (5.8) ensures that the total delay during the entire SFC is no greater than the specified. Equation (5.9) ensures that the probability of realizing the total allocated delay is at least P_D . Equation (5.10) ensures that the probability of realizing the link bandwidth is at least P_B . Equation (5.11) ensures that the total allocated bandwidth on each link cannot exceed its maximum possible link capacity.

It is worthwhile to mention that the complexity of the proposed exact formulation depends on the convexity of link delay and bandwidth distribution. For example, when link CDF and CCDF are log-concave functions (e.g., exponential distribution), Eqs. 5.4–5.11 become a convex optimization formulation.

5.5 Multi-Constrained Traffic Routing Heuristic

Considering that the running time of the proposed exact solution in Sect. 5.4.2 is exponential, especially when the problem size is large, the exact solution cannot return a solution in an extremely long time. Therefore, we propose a fast heuristic, called Multi-Constrained Traffic Routing (MCTR) for the problem in this section. The idea of MCTR is that it first discretizes the network in terms of link delay and link bandwidth, and then runs a heuristic multi-constrained routing algorithm. MCTR has three main features: (1) MCTR is a sampling-based heuristic, that is, the more samplings are used to discretize the network, the more accurate that the solution returned by the heuristic will have, but more running time it will consume since the problem size is enlarged. (2) MCTR leverages the Tunable

Accuracy Multiple Constraints Routing Algorithm (TAMCRA) to find the multi-constraint path for each adjacent VNF pair. TAMCRA (De Neve and Van Mieghem 2000) is scalable in the number of constraints and in the size of the graphs by applying Dijkstra-like algorithm, and that multiple constraints routing problems can be solved to a very high accuracy within a short time that increases linearly with. (3) MCTR dynamically adjusts the link weights and delay and bandwidth realizing probability constraint after finding the path for each VNF pair so that the cumulated probabilities will not exceed the specified. For completeness, we first give a formal definition of the multi-constrained path selection problem as follows (Kuipers and Van Mieghem 2005):

Definition 5.2 Consider a network $\mathcal{G}(\mathcal{N}, \mathcal{L})$, where each link $l \in \mathcal{L}$ is specified by a link weight vector with m additive QoS link weights $w_i(l) \geq 0$ for all $1 \leq i \leq m$. Given m constraints T_i , where $1 \leq i \leq m$, the problem is to find a path P from a source to a destination such that

$$\sum_{l \in P} w_i(l) \leq T_i \quad (5.12)$$

A path that satisfies all m constraints is referred to be a feasible path. The multi-constrained path selection problem is proved to be NP-hard (Wang and Crowcroft 1996), we therefore have chosen a heuristic for the multi-constrained routing problem.

Next, we will illustrate how to discretize the network. For a link, k samples are taken for each delay distribution and bandwidth distribution, in the format of link weight vector (delay, $-\log(\text{delay probability})$, $-\log(\text{bandwidth probability})$). It is worthwhile to mention that the bandwidth value does not exist in the weight vector, since we only need to allocate just enough requested bandwidth. As a result, each link will be transformed into k parallel links as illustrated in Fig. 5.2.

Subsequently, for each VNF pair f_i and f_j , we run the Tunable Accuracy Multiple Constraints Routing Algorithm (TAMCRA) (De Neve and Van Mieghem 2000), a heuristic for the multi-constrained path problem. The objective in TAMCRA is to minimize $\sum_{l \in p} d_l$, where d_l indicates the delay of traversed link l in path p . The delay probability constraint in TAMCRA is $-\frac{\log(P_D)}{\gamma}$ and the bandwidth probability constraint in TAMCRA is $-\frac{\log(P_B)}{\gamma}$. γ is used to “scale” the original requirement to each requested VNF pair and is first set to be the number of requested VNF pairs for simplicity. In case under $-\frac{\log(P_D)}{\gamma}$ the path is not found by applying TAMCRA, we can enlarge γ and run TAMCRA again. After that, the delay probability constraint is set $-\frac{\log(P_D/P_x)}{\gamma'}$, where P_x stands for the product of already consumed probability for each VNF pair, and γ' represents the number of VNF pairs that have not been iterated. TAMCRA never returns a path that violates QoS constraint, but it may fail to find a solution since it is a heuristic. TAMCRA is tunable in how many paths it stores per node, a parameter that we set equal to $q = ck$, a constant c times the number of samples k .

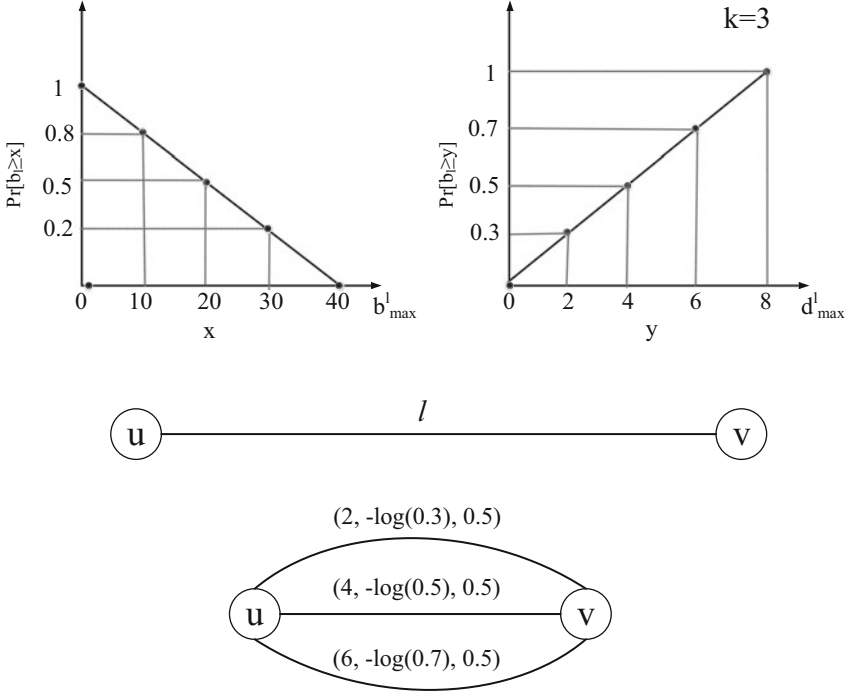


Fig. 5.2 Link transformation for the heuristic algorithm

An example is given in Fig. 5.3. It is assumed that the bandwidth CDF for link (b, c) is $\frac{x}{6}$, and the bandwidth CDF for all the other links is $\frac{x}{10}$. Moreover, except for link (a, c) , the delay distribution are the same for all the other links. Assuming there is a request $r(\{f_1, f_2, f_3\}, 4, 0.5, 15, 0.1)$. The link weight assignment are shown in Fig. 5.3a. Since r asks for allocating at least a bandwidth of $B = 4$ and link (b, c) has a bandwidth CDF $\frac{x}{6}$, then allocating at least a bandwidth of 4 leads to a probability of $1 - \frac{4}{6} \approx 0.33$. Similarly, the probability of allocating at least a bandwidth of 4 for the other links is equal to $1 - \frac{4}{10} = 0.6$. Since f_1 and f_2 are located on a and c , respectively, we take the path $a - b - c$ to route traffic from $a(f_1)$ to $c(f_2)$. The path $a - c$ cannot be selected, otherwise P_D will be violated. Afterwards, the link weights of (a, b) and (b, c) will be adjusted. More specifically, since 4 bits of bandwidth have already been allocated, the probability of allocating another 4 bits bandwidth on link (a, b) is $\frac{c_{(a,b)}(8)}{c_{(a,b)}(4)} = \frac{1-8/10}{1-4/10} \approx 0.33$. Moreover, since the maximum possible bandwidth of link (b, c) is 6 and it has already allocated 4 bits of bandwidth, link (b, c) cannot allocate another 4 bits bandwidth. We therefore used dashed line to represent it which means it cannot be used. Afterwards in Fig. 5.3b, we continue to find a path from f_2 to f_3 . It can be seen that path $c - d$ is the only feasible path. In all, the final path is $a - b - c - d$ has a total delay of 15 by summing up the link delay of (a, b) , (b, c) and (c, d) and its realizing probability is equal to $0.9 \cdot 0.9 \cdot 0.9 = 0.729$

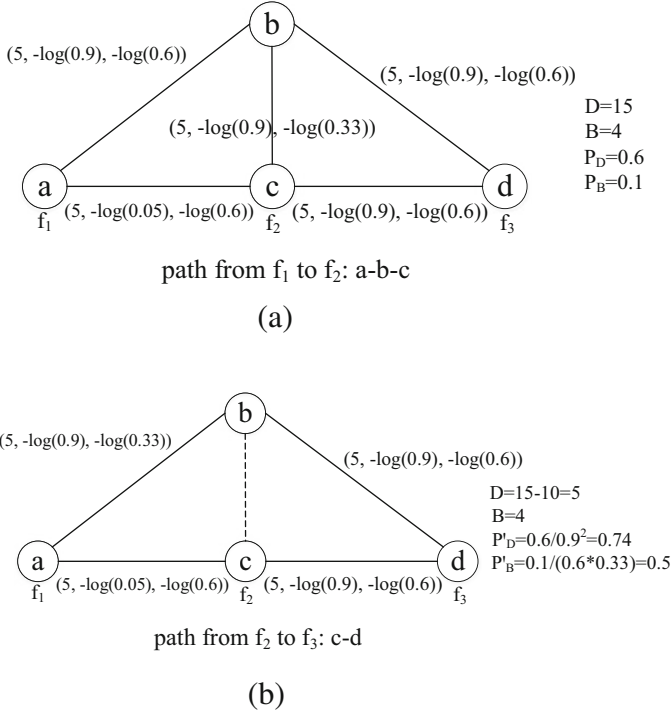


Fig. 5.3 An example of the heuristic link changes for $k = 1$. (a) Path finding from f_1 to f_2 . (b) Path finding from f_2 to f_3

by taking the product of probabilities to realize these delay. Similarly, each link in path is $a - b - c - d$ can allocate available bandwidth of 4 and the total realizing probability is equal to $0.6 \cdot 0.33 \cdot 0.6 = 0.1188$. The pseudo code of MCTR can be seen in Algorithm 3.

The complexity of MCTR is analyzed like this. There are in total $|R|$ requests, and for each request, there are constant number of possible corresponding SFCs and at most $|F| - 1$ required VNF pairs. For each VNF pair, running TAMCRA calls for a time complexity of $O(qN \log(qN) + q^2kL)$, where q is the maximum number of stored paths at each node. Consequently, the total time complexity of MCTR is $O(|R||F|qN \log(qN) + |R||F|q^2kL)$.

Algorithm 3: MCTR ($\mathcal{G}(\mathcal{N}, \mathcal{L})$, R , k)

```

1 foreach request  $r \in R$  do
2   foreach possible corresponding SFC  $\Omega$  do
3     foreach  $f_i, f_j \in \Omega$  which are located on  $n_i$  and  $n_j$  do
4       if  $n_i$  and  $n_j$  have sufficient capability then
5         Discrete the network link weights by  $k$  samplings.
6         Apply TAMCRA to find a path from  $n_i$  to  $n_j$ .
7         if step 6 succeeds then
8           Adjust link weights and requirement.
9           if  $f_j$  is the last required VNF then
10            Return this solution.
11         else
12           break;

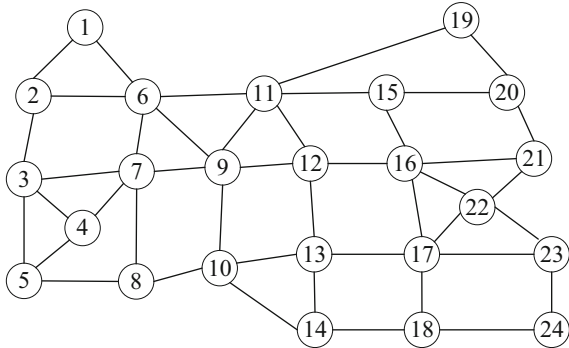
```

5.6 Simulations

5.6.1 Simulation Setup

In this section, we first conduct simulations on two backbone networks: USANet, displayed in Fig. 5.4, which is a realistic carrier backbone network, and GÉANT, shown in Fig. 5.5, which is a pan-European communications infrastructure. We also conduct simulations on a 100-node network: we generate 100 nodes and the link existence in each node pair occurs with a probability of $\frac{1}{2}$, which follows an Erdős-Rényi model (Erdős and Rényi 1959). We assume there are in total 15 VNFs, and each VNF is randomly placed on m nodes, where $m \in [2, 4]$. The link delay and bandwidth in these 3 networks are assumed to follow three distributions, namely the exponential, uniform and weibull distributions. In the exponential distribution $1 - e^{-\lambda x}$, we choose $\lambda \in [0.001, 0.01]$ for the bandwidth distributions of different links and $\lambda \in [0.5, 1.5]$ for the delay distributions of different links. In the

Fig. 5.4 USA carrier backbone network



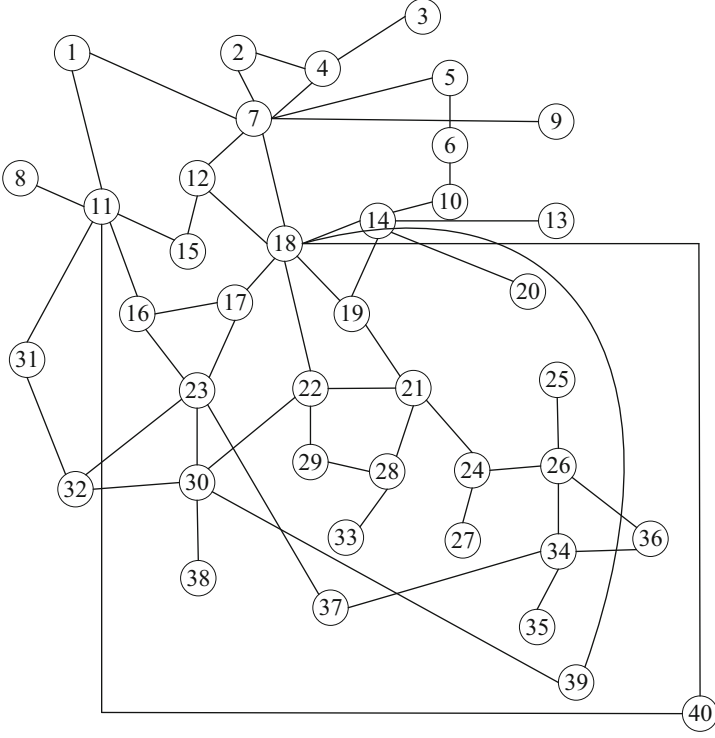


Fig. 5.5 GÉANT pan-European research network

uniform distribution $\frac{x-\alpha}{\beta-\alpha}$, we choose $\alpha = 0$ for both bandwidth and delay, and $\beta \in [200, 300]$ for bandwidth and $\beta \in [3, 7]$ for delay. In the weibull distribution $1 - e^{-(x/\lambda)^k}$, we choose $k = 1$ for both bandwidth and delay, and $\lambda \in [1, 5]$ for delay and $\lambda \in [0.0001, 0.0005]$ for bandwidth.

Considering that there is no existing work dealing with the traffic routing problem in stochastic networks, we compare the exact solution and heuristic MCTR with two benchmark heuristics, namely Least Expected Path (LEP) and Random algorithm. For each request and its possible corresponding SFC, for each VNF pair that are located on different nodes n_i and n_j , these two heuristics perform as follows:

- LEP: It first assigns each link with the expected delay value, and then runs the shortest path from n_i to n_j .
- Random algorithm: It finds w -shortest hop path from n_i to n_j , and randomly selects one path from these k paths.

After determining the path for each VNF pair, we let L_x represent the total number of links (hops) that the required SFC traverses. Subsequently, for both LEP and Random algorithm, each traversed link l will allocate $\frac{r \cdot D}{L_x}$ delay and $r \cdot B \cdot t_l$

Table 5.2 Simulation parameters

Distribution	$r(F, B, P_B, D, P_D)$	Link bandwidth	Link delay
$1 - e^{-\lambda x}$	$B \in [1, 10], P_B \in [0.1, 0.3]$ $D \in [5, 15], P_D \in [0.1, 0.4]$	$\lambda \in [0.001, 0.01]$	$\lambda \in [0.5, 1.5]$
$\frac{x-\alpha}{\beta-\alpha}$	$B \in [1, 10], P_B \in [0.05, 0.1]$ $D \in [15, 30], P_D \in [0.1, 0.2]$	$\alpha = 0 \beta \in [200, 300]$	$\alpha = 0 \beta \in [3, 7]$
$1 - e^{-(x/\lambda)^k}$	$B \in [1, 10], P_B \in [0.005, 0.01]$ $D \in [15, 30], P_D \in [0.01, 0.02]$	$k = 1 \lambda \in [0.0001, 0.0005]$	$k = 1 \lambda \in [1, 5]$

bandwidth, where t_l denotes the times that the link is traversed.⁴ For each traversed link l , the probability of allocating delay is calculated as $CDF_l(\frac{r \cdot D}{L_x})$, and the probability of realizing bandwidth calculated as $CCDF_l(r \cdot B \cdot t_l)$. By taking the product of all allocated probabilities, we can calculate whether the request can be satisfied or not. Random algorithm will try at most p times from these w paths if the cumulated constraint (e.g., delay, bandwidth, realizing probability) is not obeyed. In particular, we set $w = 10$ and $p = 3$ in Random algorithm. We first set in MCTR that $k = 4, 8, 16$ and $c = 1$, which means that the maximum number of stored paths is set $q = 4, 8, 16$, respectively, when compared to LEP and Random algorithm. Later for a fixed $k = 8$, we will vary q in order to further evaluate the heuristic algorithm.

In order that the feasible solution exists and increase the difficulty for the algorithms to find solutions, the request $r(F, B, P_B, D, P_D)$ is set like this (in line with Kuipers et al. 2014): For exponential distribution, we have set $D \in [5, 15]$, $P_D \in [0.1, 0.4]$, $B \in [1, 10]$ and $P_B \in [0.1, 0.3]$; for uniform distribution, we have set $D \in [15, 30]$, $P_D \in [0.1, 0.2]$, $B \in [1, 10]$ and $P_B \in [0.05, 0.1]$; for weibull distribution, we have set $D \in [15, 30]$, $P_D \in [0.01, 0.02]$, $B \in [1, 10]$ and $P_B \in [0.005, 0.01]$. Moreover, F is between 2 and 5 for all 3 distributions. Table 5.2 provides the simulation parameters.

The simulations are run on a high-performance desktop PC with 3.4 GHz and 16GB memory. We use C# to implement all the heuristics, and we use CVX in Matlab to implement the exact solution, a package for specifying and solving convex optimization problems (Grant and Boyd 2014).

5.6.2 Simulation Results

5.6.2.1 Backbone Networks

We first generate in total 100 requests for each distribution, and it is assumed that the request arrives in network one by one in an online fashion. Accordingly,

⁴ It is possible that one link is traversed more than once in an SFC.

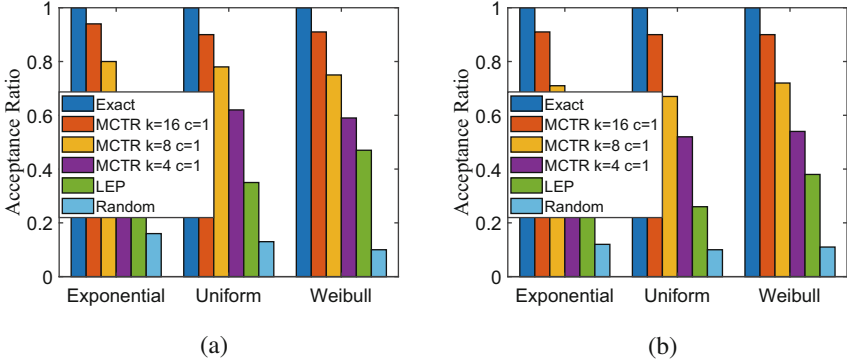


Fig. 5.6 Acceptance Ratio on two backbone networks: (a) USANet and (b) GÉANT

the algorithm runs 100 times sequentially for each request. We first compare the algorithms in terms of Acceptance Ratio (AR), which is defined as the number of accepted requests divided by the total number of requests. As expected, Fig. 5.6 shows that the exact solution can always find paths for each request (if it exists) with $AR=1$, which also verifies its exactness and correctness. With k (and q) increasing, we found that the achieved AR value of MCTR approaches to the close-to-optimal (above 90% when $k = 16$). The reason is that more samples are created in networks, and therefore there are more parallel links with various delay and bandwidth allocation weights between each node pair, which makes MCTR to have more choices to find feasible solutions. Since MCTR leverages TAMCRA, which is a heuristic to find the multi-constrained path, MCTR may not always find a feasible solution. However, with the maximum number of stored paths increasing, it increases the chance for MCTR to return the feasible solution. On the other hand, both the LEP and Random algorithm behave poorly by having a much lower AR value. For LEP, it indicates that the least expected path may not always satisfy the request with specified delay and bandwidth probability requirement. For the Random algorithm, although it adopts $w = 10$ paths for each VNF pair, the quality of paths is not defined very well according to the request, and its randomness further leads to the worst performance especially under a stricter problem condition.

Figure 5.7 plots the total running time of all the algorithms. We see that the exact solution consumes a significantly higher running time than 3 heuristics. Although it can achieve the best performance in terms of AR, it still cannot be adopted to compute the solution for when the request arrives in an online fashion. The proposed MCTR has much lower running time than the exact solution, but its running time is higher than the two benchmark heuristics. However, it is still acceptable since it can achieve close-to-optimal AR value. We observe that with k increasing, its running time also increases. This is caused by the larger network scale and larger maximum number of stored paths. Nevertheless, we can tune the parameters of MCTR in practice to strike a tradeoff between accuracy and running time.

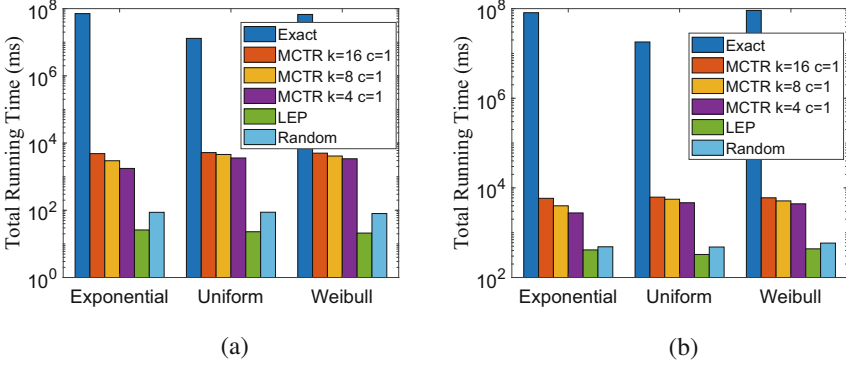


Fig. 5.7 Running Time on two backbone networks: (a) USANet and (b) GÉANT

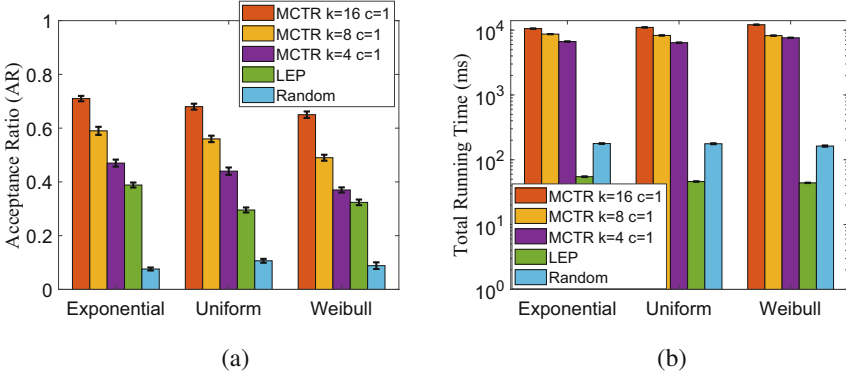


Fig. 5.8 Performance on a 100-node network: (a) Acceptance ratio and (b) running time

5.6.2.2 100-Node Network

In this scenario, we run all the algorithms on a larger 100-node network, but we found that the exact algorithm cannot return a solution because of its exponential time complexity. This also indicates that the exact solution cannot scale well with the problem size and cannot be adopted in practice. Due to the lack of the exact solution, we generate 100 sets of 100 traffic requests for distributions, and evaluate all the heuristic algorithms for those 100 sets of 100 traffic requests (100 runs). By doing this, we want to establish confidence on the performance of heuristics. Figures 5.8a and b respectively depict the AR and running time (in log scale) of all these algorithms, where the confidence interval is set to 95%. The 95% confidence interval is calculated for all the figures, but in those where it is not visible, the

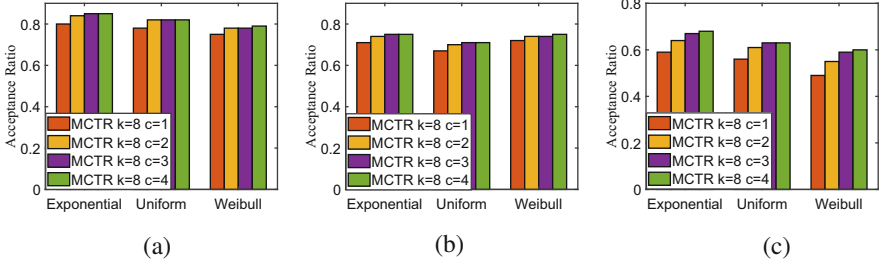


Fig. 5.9 Acceptance Ratio of MCTR on three networks for $k = 8$ and $q = ck$, where $c = 1, 2, 3, 4$: (a) USANet (b) GÉANT (c) 100-node network

interval is negligibly small.⁵ More specifically, in Fig. 5.8a, the confidence intervals of AR values for all the algorithms stay around 0.01. In Fig. 5.8b, the confidence intervals of running time for MCTR and LEP ranges from 0.3 to 0.6, and the confidence intervals of running time for Random ranges from 1.9 to 3.3, which is mainly caused by its randomness. We see that the AR values of all the algorithms in 100-node network are lower than the AR values in backbone networks, due to the reason that SFC may traverse more links which consumes more delay and hence violate the required delay value. Also, the enlarged network scale makes all the algorithms to consume more time than backbone networks. Nevertheless, the proposed MCTR achieves much higher AR value than LEP and Random, but this comes at the expense of a bit higher running time.

5.6.2.3 Varying the Maximum Number of Stored Paths for MCTR

Finally, in MCTR we keep $k = 8$ the same and vary the value of $c = 1, 2, 3, 4$. Our aim is to evaluate the performance of MCTR when the maximum number of stored paths $q = ck$ changes. In particular, we only show the average value of all the algorithms in the 100-node network since the confidence intervals in this case are very small. As expected, Fig. 5.9 shows that with q increasing, its achieved AR value increases, which indicates that a bigger q may lead to better performance. However, the running time of MCTR increases when q increasing, as we see in Fig. 5.10.

In all, we conclude that the exact solution can always find the optimal solution (if there exists), but this comes at the expense of significantly higher running time. When the problem size becomes larger (e.g., $|N| = 100$), the exact solution cannot return a feasible solution within quite a long time. Our proposed MCTR, can return a feasible solution at a much quicker time regardless of the problem size, which can be used when the traffic requests arrive in an online fashion.

⁵ We note here that some plots are log-scale that additionally contributes to the confidence interval visibility.

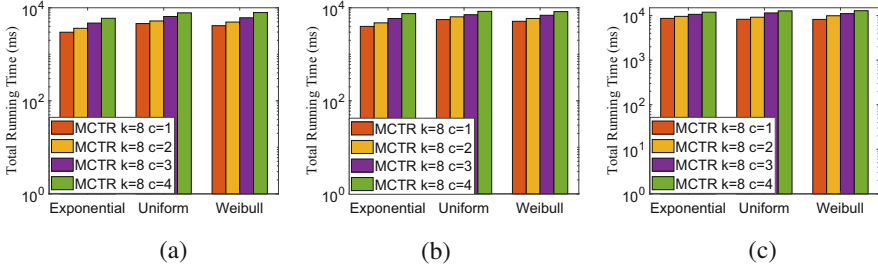


Fig. 5.10 Running Time of MCTR on three networks for $k = 8$ and $q = ck$, where $c = 1, 2, 3, 4$: (a) USANet (b) GÉANT (c) 100-node network

5.7 Summary

In this chapter, we have studied the Traffic Routing problem in Stochastic NFV Networks. The randomness/uncertainties in networks usually arise from inaccurate data, expired exchanged information, insufficient estimation to the network, etc. Under the assumption that the link delay and link bandwidth are random variables and their CDFs are known, we have shown that the problem is NP-hard. To solve it, we present an exact optimization formulation as well as a tunable sampling-based heuristic MCTR. MCTR leverages the Tunable Accuracy Multiple Constraints Routing Algorithm (TAMCRA) to find the multi-constraint path for each adjacent VNF pair. Moreover, it dynamically adjusts the link weights and delay and bandwidth realizing probability constraint after finding the path for each VNF pair in order that the cumulated probabilities will not exceed the specified. The simulation results demonstrate that the proposed heuristic can achieve close-to-optimal performance in terms of acceptance ratio, but its running time is much lower than the exact solution. Moreover, MCTR can scale well when the network size is enlarged to 100-node, but the exact solution cannot return a feasible in this scenario within a reasonably long time, which indicates that the exact solution cannot be adopted for when the provisioning time needs to be short. We found that with more samplings are allowed in MCTR, its achieved AR value gets higher which means it can accept more requests. Meanwhile, when the number of stored paths is increased in MCTR, its performance also gets better. This has verified that MCTR is a tunable and efficient heuristic that can achieve a tradeoff between accuracy and running time. For the VNF placement and routing problem in random NFV networks, using the heuristic algorithms provided in the previous chapters does not meet the real-time service requirements. As discussed in the next chapter of this book, VNF placement and routing optimization algorithms based on deep reinforcement learning provide superior answers to real-time service requirements.

References

- Bhamare D, Jain R, Samaka M, Erbad A (2016) A survey on service function chaining. *J Netw Comput Appl* 75:138–155
- Blanco B, Fajardo JO, Giannoulakis I, Kafetzakis E, Peng S, Pérez-Romero J, Trajkovska I, Khodashenas PS, Goratti L, Paolino M et al. (2017) Technology pillars in the architecture of future 5G mobile networks: NFV, MEC and SDN. *Comput Stand Interfaces* 54:216–228
- Cheng X, Wu Y, Min G, Zomaya AY (2018) Network function virtualization in dynamic networks: A stochastic perspective. *IEEE J Sel Areas Commun* 36(10):2218–2232
- Cisco annual internet report (2018–2023) white paper (2020). <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html>
- De Neve H, Van Mieghem P (2000) Tamcra: a tunable accuracy multiple constraints routing algorithm. *Comput Commun* 23(7):667–679
- Dwaraki A, Wolf T (2016) Adaptive service-chain routing for virtual network functions in software-defined networks. In: *Proceedings of the 2016 workshop on hot topics in middleboxes and network function virtualization*, ser. ACM HotMiddlebox, pp 32–37
- Erdős P, Rényi A (1959) On random graphs I. *Publicationes Mathematicae (Debrecen)*, pp 290–297
- ETSI Publishes First Specifications for Network Functions Virtualisation. <http://www.etsi.org/news-events/news/700-2013-10-etsi-publishes-first-nfv-specifications>
- Fan J, Jiang M, Rottenstreich O, Zhao Y, Guan T, Ramesh R, Das S, Qiao C (2018) A framework for provisioning availability of NFV in data center networks. *IEEE J Sel Areas Commun* 36(10):2246–2259
- Fu X, Yu FR, Wang J, Qi Q, Liao J (2019) Dynamic service function chain embedding for NFV-enabled IoT: a deep reinforcement learning approach. *IEEE Trans Wirel Commun* 19(1):507–519
- Gao, L, Rouskas GN (2019) On congestion minimization for service chain routing problems. In: *IEEE international conference on communications (ICC)*, pp 1–6
- Garcia-Saavedra A, Costa-Perez X, Leith DJ, Iosifidis G (2018) FluidRAN: optimized vRAN/MEC orchestration. In: *IEEE INFOCOM*
- Garey MR, Johnson DS (1979) *Computers and intractability: a guide to the theory of np-completeness*. W. H. Freeman, New York
- Grant M, Boyd S (2014) CVX: matlab software for disciplined convex programming, version 2.1. <http://cvxr.com/cvx>
- Guerin RA, Orda A (1999) Qos routing in networks with inaccurate information: theory and algorithms. *IEEE/ACM Trans Netw* 7(3):350–364
- Guo L, Pang J, Walid A (2018) Joint placement and routing of network function chains in data centers. In: *IEEE INFOCOM*
- Hamann M, Fischer M (2019) Path-based optimization of nf-v-resource allocation in SDN networks. In: *ICC 2019–2019 IEEE international conference on communications (ICC)*, pp 1–6
- Hantouti H, Benamar N, Taleb T, Laghrissi A (2018) Traffic steering for service function chaining. *IEEE Commun Surv Tutor* 21(1):487–507
- Hardy G, Littlewood J, Polya G (1934) *Inequalities*. Cambridge University Press, Cambridge
- Herrera JG, Botero JF (2016) Resource allocation in NFV: a comprehensive survey. *IEEE Trans Netw Serv Manag* 13(3):518–532
- Korkmaz T, Krunz M (2003) Bandwidth-delay constrained path selection under inaccurate state information. *IEEE/ACM Trans Netw* 11(3):384–398
- Krishnan M, Yun S, Jung YM (2018) Improved clustering with firefly-optimization-based mobile data collector for wireless sensor networks. *AEU - Int J Electron Commun* 97:242–251
- Krishnan M, Yun S, Jung YM (2019) Enhanced clustering and aco-based multiple mobile sinks for efficiency improvement of wireless sensor networks. *Comput Netw* 160:33–40

- Kuipers FA, Van Mieghem PF (2005) Conditions that impact the complexity of QoS routing. *IEEE/ACM Trans Netw* 13(4):717–730
- Kuipers FA, Yang S, Trajanovski S, Orda A (2014) Constrained maximum flow in stochastic networks. In: *Proc. of IEEE ICNP 2014*, North Carolina
- Kuo T-W, Liou B-H, Lin KC-J, Tsai M-J (2016) Deploying chains of virtual network functions: On the relation between link and server usage. In: *IEEE INFOCOM*
- Laghriissi, A, Taleb T (2019) A survey on the placement of virtual resources and virtual network functions. *IEEE Commun Surv Tutor* 21(2):1409–1434
- Li Q, Jiang Y, Duan P, Xu M, Xiao X (2017) Quokka: latency-aware middlebox scheduling with dynamic resource allocation. *J Netw Comput Appl* 78:253–266
- Lorenz DH, Orda A (1998) QoS routing in networks with uncertain parameters. *IEEE/ACM Trans Netw* 6(6):768–778
- Ma W, Sandoval O, Beltran J, Pan D, Pissinou N (2017) Traffic aware placement of interdependent NFV middleboxes. In: *IEEE INFOCOM*
- Marotta A, Zola E, D’Andreagiovanni F, Kassler A (2017) A fast robust optimization-based heuristic for the deployment of green virtual network functions. *J Netw Comput Appl* 95:42–53
- Medhat AM, Taleb T, Elmangoush A, Carella GA, Covaci S, Magedanz T (2017) Service function chaining in next generation networks: state of the art and research challenges. *IEEE Commun Mag* 55(2):216–223
- Miao W, Min G, Wu Y, Huang H, Zhao Z, Wang H, Luo C (2019) Stochastic performance analysis of network function virtualization in future internet. *IEEE J Sel Areas Commun* 37(3):613–626
- Mijumbi R, Serrat J, Gorricho J-L, Bouten N, De Turck F, Boutaba R (2016) Network function virtualization: state-of-the-art and research challenges. *IEEE Commun Surv Tutor* 18(1):236–262
- Mirjalili G (2018) Optimal network function virtualization and service function chaining: a survey. *Chin J Electron* 27:704–717
- Nunes BA, Mendonca M, Nguyen XN, Obraczka K, Turletti T (2014) A survey of software-defined networking: past, present, and future of programmable networks. *IEEE Commun Surv Tutor* 16(3), 1617–1634 (2014)
- Pham C, Tran NH, Ren S, Saad W, Hong CS (2017) Traffic-aware and energy-efficient vNF placement for service chaining: joint sampling and matching approach. *IEEE Trans. Serv. Comput.* 13(1):172–185
- Qu L, Assi C, Shaban K (2016) Delay-aware scheduling and resource optimization with network function virtualization. *IEEE Trans Commun* 64(9):3746–3758
- Reitblatt M, Foster N, Rexford J, Schlesinger C, Walker D (2012) Abstractions for network update. *SIGCOMM Comput Commun Rev* 42(4):323–334
- Song S, Lee C, Cho H, Lim G, Chung J (2019) Clustered virtualized network functions resource allocation based on context-aware grouping in 5G edge networks. *IEEE Trans Mobile Comput* 19(5):1072–1083
- Tang H, Zhou D, Chen D (2019) Dynamic network function instance scaling based on traffic forecasting and VNF placement in operator data centers. *IEEE Trans Parallel Distrib Syst* 30(3):530–543
- Tomassillik A, Giroire F, Huin N, Pérennes S (2018) Provably efficient algorithms for placement of service function chains with ordering constraints. In: *IEEE INFOCOM*
- Wang Z, Crowcroft J (1996) Quality-of-service routing for supporting multimedia applications. *IEEE J Sel Areas Commun* 14(7):1228–1234
- Wang T, Xu H, Liu F (2017) Multi-resource load balancing for virtual network functions. In: *IEEE 37th international conference on distributed computing systems (ICDCS)*, pp 1322–1332
- Yang S, Kuipers FA (2014) Traffic uncertainty models in network planning. *IEEE Commun Mag* 52(2):172–177
- Yang S, Li F, Trajanovski S, Fu X (2020) Traffic routing in stochastic network function virtualization networks. *J Netw Comput Appl* 169:102765
- Yang S, Li F, Trajanovski S, Chen X, Wang Y, Fu X (2021a) Delay-aware virtual network function placement and routing in edge clouds. *IEEE Trans Mobile Comput* 20(2):445–459

- Yang S, Li F, Yahyapour R, Fu X (2022) Delay-sensitive and availability-aware virtual network function scheduling for NFV. *IEEE Trans Serv Comput* 15(1):188–201
- Yi B, Wang X, Li K, Huang M et al. (2018) A comprehensive survey of network function virtualization. *Comput Netw*
- Yu R, Xue G, Zhang X (2017) QoS-aware and reliable traffic steering for service function chaining in mobile networks. *IEEE J Sel Areas Commun* 35(11):2522–2531
- Zeng D, Zhang J, Gu L, Guo S (2018) Stochastic scheduling towards cost efficient network function virtualization in edge cloud. In: *IEEE international conference on sensing, communication, and networking (SECON)*, pp 1–9

Chapter 6

A-DDPG: Attention Mechanism-Based Deep Reinforcement Learning for NFV



Chapters 3–5 transform the VNF placement and traffic routing problem into some well-known NP-hard problems, and then a heuristic or approximation method is proposed to solve it, at the expense of ignoring the network state dynamics. To bridge that gap, in this chapter, we formulate the VNF placement and traffic routing problem as a Markov Decision Process model to capture the dynamic network state transitions.¹ In order to jointly minimize the delay and cost of NFV providers and maximize the revenue, we devise a customized Deep Reinforcement Learning (DRL) algorithm, called A-DDPG, for VNF placement and traffic routing in a real-time network. A-DDPG uses the attention mechanism to ascertain smooth network behavior within the general framework of network utility maximization (NUM). The simulation results show that A-DDPG outperforms the state-of-the-art in terms of network utility, delay, and cost.

6.1 Introduction

Traditionally, Network Functions (NFs), such as firewalls and load balancers, are implemented on physical devices, called middleboxes, which are costly, lack flexibility, and are difficult to operate. Network Function Virtualization (NFV) has emerged as an innovative technique that can deal with these challenges by decoupling network functions from dedicated hardware and realizing them in the form of Virtual Network Functions (VNFs) (Bonfim et al. 2019; Ren et al. 2020). Because this technique shows great potential in promoting openness, innovation, flexibility, and scalability of networks, NFV attracts a good deal of interest from the networking community (Scazzariello et al. 2020; Ruiz et al. 2020; Yang et al. 2021). To build more complex services, the notion of Service Function Chaining

¹ Parts of this chapter is reprinted from He et al. (2021), with permission from IEEE.

(SFC) can be used, where a sequence of VNFs must be processed in a pre-defined order to collectively deliver a certain service. Therefore, an important problem is to determine the positions for placing VNFs and select the paths for routing traffic, such that the service requirement can be satisfied. The problem of VNF placement and traffic routing is referred to as VNF-PR in this chapter. In solving the VNF-PR problem, service providers typically strive for network utility maximization (NUM). Therefore, jointly considering cost and QoS (e.g., delay) schemes is required, which will lead to a better user experience and higher profit.

Existing works on the VNF-PR problem is either based on linear programming (Yala et al. 2018; Nguyen et al. 2019) or the VNF-PR problem is translated into some well-known NP-hard problems (Zhang et al. 2017) such as the knapsack problem, and then a heuristic or approximation method is proposed to solve it (Allybokus et al. 2018; Pei et al. 2018; Sallam and Ji 2019), at the expense of ignoring the network state dynamics. For better performance, existing works formulate a one-shot optimization problem in a dynamic environment (Golkarifard et al. 2021), and some works consider the VNF-PR problem over the entire system lifespan (Soualah et al. 2019). However, they only focus on the revenue for NFV operators and do not pay attention to the network utility consisting of revenue and cost.

Another line of work applies (Deep) Reinforcement Learning (DRL) (Henderson et al. 2018) to solve the VNF-PR problem (Guo et al. 2019; Khezri et al. 2019; Quang et al. 2020). More specifically, to solve the VNF-PR problem, an RL agent interacts with the real-time NFV-enabled environment through the implementation of placement and routing strategies. Subsequently, the RL agent continuously optimizes the strategies according to the reward value of the environment feedback (e.g., delay, capacity, and overhead). However, given the large state space involved, RL methods become impractical and inefficient for large networks. On the contrary, deep neural networks can be applied to high-dimensional state space. Different from these approaches, in this chapter, we leverage the feature of deep neural networks and introduce a Markov Decision Process (MDP) to capture the dynamic network state transitions and process them within a DRL architecture. We adopt a Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al. 2015) algorithm to deal with the high-dimensional and time-varying network state and complex network environment.

Typically, a DRL agent will not pay equal attention to all the available placement nodes. The agent usually chooses the current action based on information of higher levels of cognitive skills and ignores other perceivable information. In this chapter, we introduce the concept of attention mechanism, which is widely used in neural image caption generation (Courville et al. 2015) to simulate the agent's action for DDPG. We find that during the training process of DDPG, the attention mechanism will automatically focus on the feasible neighbor node that may affect the agent's selection behavior. It ultimately helps to reduce the attention to other unnecessary nodes and improve the training efficiency of the model. With this motivation, we design a customized attention mechanism-based DDPG to train our DRL model.

The main contributions of this chapter are as follows:

- We formulate the VNF-PR problem as an optimization model and establish a utility function aiming to trade off between revenue and cost.
- We propose a novel Attention mechanism-based Deep Deterministic Policy Gradients (A-DDPG) framework, using the Actor-Critic network structure, in which both the Actor and Critic networks adopt double networks (namely the main network and the target network).
- Through extensive simulation experiments, we show that our A-DDPG framework outperforms the state-of-the-art in terms of network utility, delay, and cost.

The remainder of this chapter is organized as follows. Section 6.3 defines the VNF-PR problem. In Sect. 6.4, we devise the A-DDPG algorithm to solve the VNF-PR problem. Section 6.5 provides the simulation results. Section 6.2 describes related work and we conclude in Sect. 6.6.

6.2 Related Work

6.2.1 Combinatorial Optimization Theory for NFV

The VNF-PR problem has been studied for different objectives (Sharma et al. 2020), such as cost minimization (Gao et al. 2020; Bunyakitanon et al. 2020), performance improvement (Pei et al. 2019; Dinh-Xuan et al. 2020), and utility maximization (Kuo et al. 2017; Gu et al. 2019b; Shah and Zhao 2020). In most studies, VNF placement, either alone or jointly with traffic routing, is investigated by using combinatorial optimization theory (e.g., primal-dual, rounding, Markov approximation). For instance, Ma et al. (2017) target the VNF placement problem to load balance the traffic at base stations. They subsequently solve the problem when the flow path is predetermined, and propose a traffic and space aware routing heuristic for a non-ordered or ordered middlebox set. Feng et al. (2017) present a VNF placement and traffic routing model to minimize the network cost. The authors formulate the problem as an ILP and then devise an approximate algorithm to effectively consolidate flows into a limited number of active resources. A Minimum-Residue heuristic is presented in (Bhamare et al. 2015) for VNF placement in a multi-cloud scenario with constraints of deployment cost. Sampaio et al. (2018) study how to achieve load balancing in networks to reduce the number of overloaded links in NFV/SDN-enabled networks. However, the aforementioned studies do not consider QoS, such as the delay of the data flow in SFC. Gao et al. (2020) propose a cost-efficient scheme to address the VNFs placement problem in public cloud networks with the goal of low cost and latency. Cziva et al. (2018) study how to use the optimal stopping theory to place VNFs under the edge cloud to minimize the total delay expectation. However, the authors in (Cziva et al. 2018) assume

that one VNF is sufficient to meet the users' requirements. Nevertheless, applying combinatorial optimization theory cannot work well with the real-time dynamic network variations (Xiao et al. 2019).

6.2.2 *DRL for NFV*

Some studies solve the VNF-PR problem by using DRL. For instance, Quang et al. (2019) solve the VNF-PR or VNF forwarding graph embedding problem in multiple non-cooperative domains by jointly considering the delay and underlying infrastructure constraints. They first introduce a DRL framework in which each domain determines the bidding price of using its resources selfishly. After that, the final decision is made by the owner of VNF-PR by executing a Cost-based First Fit (CFF)-based heuristic algorithm. Xiao et al. (2019) present an adaptive deep reinforcement learning approach to automatically deploy SFCs for the optimization of throughput and operation cost. Tong et al. (2020) propose a Gated Recurrent Units (GRU)-based traffic prediction model and place VNF instances in advance based on the prediction result. They apply a DRL algorithm called Asynchronous Advantage Actor-Critic (A3C) to train the agent and then obtain the optimal strategy. Nakanoya et al. (2019) propose an accelerated reinforcement learning method to shorten the delivery time of services. According to Nakanoya et al. (2019), the reinforcement learning agent learns the optimal placement strategy of VNFs according to the state value function and simulates the model in various environments. Sun et al. (2020) combine the DRL and GNN to solve the VNF placement problem with the minimum deployment cost. However, the methods in (Quang et al. 2019; Xiao et al. 2019; Tong et al. 2020; Nakanoya et al. 2019; Sun et al. 2020) ignore the end-to-end delay, especially the processing delay. Gu et al. (2019a) minimize the deployment cost based on geographic location and the SFC processing delay, and jointly solve the VNF placement and routing problem using a DRL algorithm. However, the authors in (Gu et al. 2019a) suppose that all VNF instances have already been placed on network nodes, so the VNF-PR problem is simplified to the deployment of paths and allocation of traffic load on the links. None of the aforementioned works considers the impact of surrounding nodes' resources on network states. In fact, the importance of neighbors to the learning agent is distinguishable according to their remaining resources in the DRL model. The attention mechanism enables to focus on neighbor nodes with sufficient resources and contributes to the generation of neighbor interaction behaviors. Li et al. (2020) make a preliminary attempt to solve the VNF placement problem with the combination of attention based sequence model and RL algorithm, where the RL agent incorporates an entropy maximization strategy and the goal is formalized as optimizing the power consumption of the service chain. However, Li et al. (2020) assumes that the problem model is only for a star topology with 10 nodes, which is not always the case in practice. Our proposed A-DDPG solves the VNF-PR problem by applying the attention mechanism to the DRL architecture, using the Actor-Critic network structure.

6.3 Model and Problem formulate

In this section, we begin with describing the network utility model in Sect. 6.3.1 and then we formulate the VNF-PR problem with the objective and constraints in Sect. 6.3.2. For the convenience of reading, we summarize the notations used in this chapter in Table 6.1.

6.3.1 Network Utility Model

Firstly, we consider a physical network which is presented as a graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$, where \mathcal{N} and \mathcal{E} stand for the node set and the link set, respectively. We mainly consider two kinds of resource constraints, including node and link resource

Table 6.1 Notations

Variable	Definition
\mathcal{G}	Physical network
\mathcal{N}	The set of nodes of the network
\mathcal{E}	The set of links of the network
\mathcal{R}	The set of request. For each $r(\xi, \mathcal{F}, D) \in \mathcal{R}$, ξ indicates the flow rate, \mathcal{F} represents a set of requested VNFs, D denotes the requested delay
\mathcal{F}_r	The set of requested VNFs of $r \in \mathcal{R}$
$u_{s,a}^r, u_{s,a}^c$	The revenue function and cost function
T^r	The total delay of network request $r \in \mathcal{R}$
C_n^f	Capacity demand for $f \in \mathcal{F}_r$ on node $n \in \mathcal{N}$
$C_{u,v}$	The link capacity demand between u and v
d_n^f	Delay demand for $f \in \mathcal{F}_r$ on node n
$d_{u,v}$	Delay demand between u and v
δ_n, η_e	Capacity of node $n \in \mathcal{N}$ and link $e \in \mathcal{E}$
Ψ	The expected service payment from consumers
$\Phi^{op}, \Phi^{de}, \Phi^{tr}$	The unit operation cost, unit deployment cost and unit transmission cost, respectively
S, A, R	The state space, action space, and reward, respectively
$x_n^{r,f}$	A Boolean variable. It is 1 if r 's requested VNF f is placed on n ; and 0 otherwise
$y_{u,v}^r$	A Boolean variable. It is 1 if path between u and v is used for delivering the requested task of r ; and 0 otherwise
z_r	A Boolean variable. It is 1 if r is accepted; and 0 otherwise
k_i, v_i, q_i	The key, value and query for node $i \in \mathcal{N}$
sc_i^j	The compatibility of query q_i with key k_j
θ^μ and θ^Q	The weights of actor and critic networks
\mathbb{N}_τ	The distribution with a time τ for the exploration noise

constraints. Each node $n \in \mathcal{N}$ has a capacity of δ_n (i.e., CPU cycles per second) and a delay of d_n . Each link $e \in \mathcal{E}$ has a capacity of η_e and a delay d_e . We use \mathcal{R} to represent a set of $|\mathcal{R}|$ requests, and each $r_l(\xi, \mathcal{F}, D) \in \mathcal{R}$ has multiple VNFs that are used in sequence, where ξ indicates the flow rate, \mathcal{F} represents a set of requested VNFs, and D denotes the requested delay. We define the VNFs set as $\mathcal{F}_r = \{f_1, f_2, \dots, f_K\}$. Each VNF $f \in \mathcal{F}_r$ on node $n \in \mathcal{N}$ requires time of D_n^f to process it. Before SFC, we define k as a constant in the range $(0, K)$. When all VNFs in SFC are successfully placed and routed, $k = K$. We use the high-order matrix $K * \mathcal{N}$ to represent the deployment status of VNF on a physical server.

The total network utility to serve a request r consists of revenue and cost. More specifically, we define the utility function U^r of request r as:

$$U^r = u_{s,a}^r - u_{s,a}^c \quad (6.1)$$

where $u_{s,a}^r$ is the revenue function, and $u_{s,a}^c$ is the total cost. We use the concept of Shannon's entropy (Shannon 2001) and define the revenue function as:

$$u_{s,a}^r = \sum_{r \in \mathcal{R}} z_r \cdot \xi \cdot \Psi - \sum_{r \in \mathcal{R}} \left(-\frac{1}{T^r} \log \frac{1}{T^r} \right) \quad (6.2)$$

where ξ represents the traffic of the request r , and Ψ is the expected service revenue from consumers according to the SLA (Chase et al. 2006). T^r represents the total delay of network request r , and it is the sum of the processing delays of all nodes. The purpose of using information entropy is to unify service revenue and delay that ensures the additivity between data. The transmission delay in the SFC is defined as:

$$T^r = \sum_{f_i \in \mathcal{F}_r} x_n^{r,f} \cdot d_n^f + \sum_{e^{u,v} \in \mathcal{E}} y_{u,v}^r \cdot d_{u,v} \quad \forall u, v, n \in \mathcal{N} \quad (6.3)$$

where d_n^f represents the delay demand for $f \in \mathcal{F}_r$ on node n , $d_{u,v}$ indicates the delay demand between nodes u and v .

The cost function $u_{s,a}^c$ includes three parts: operation cost, deployment cost and transmission cost.

6.3.1.1 Operation Cost

Each physical node needs to complete the preparatory work before deploying VNFs, such as the pre-configuration of different types of VNFs. We define the unit operating cost as Φ^{op} , and then the total operation cost is defined as:

$$u_{s,a}^{op} = \sum_{n \in \mathcal{N}} x_n^{r,f} \cdot \Phi^{op} \quad \forall r \in \mathcal{R}, f \in \mathcal{F}_r \quad (6.4)$$

6.3.1.2 Deployment Cost

The deployment cost of a server is directly proportional to the resources consumed. Therefore, we stipulate that VNF deployment cost is mainly generated by the server of the deployed function. If no VNF is placed on the server, the deployment cost is not considered. We define the unit deployment cost as Φ^{de} . The total deployment cost is defined as:

$$u_{s,a}^{de} = \sum_{n \in \mathcal{N}} x_n^{r,f} \cdot \Phi^{de} \quad \forall r \in \mathcal{R}, f \in \mathcal{F}_r \quad (6.5)$$

6.3.1.3 Transmission Cost

The transmission cost is the communication cost for transferring traffic between nodes. In practice, the deployment cost is negatively correlated with the transmission cost (Verbrugge et al. 2006). When the number of VNFs is reduced to save on deployment cost, the average transmission cost of the network will increase. We define the transmission unit cost as Φ^{tr} . The total transmission cost is defined as:

$$u_{s,a}^{tr} = \sum_{e \in \mathcal{E}} y_{u,v}^r \cdot \xi_r^e \cdot \Phi^{tr} \quad \forall r \in \mathcal{R}, u, v \in \mathcal{N} \quad (6.6)$$

Finally, $u_{s,a}^c$ is a combination of above mentioned three kinds of cost:

$$u_{s,a}^c = u_{s,a}^{op} + u_{s,a}^{de} + u_{s,a}^{tr} \quad (6.7)$$

6.3.2 Problem Definition and Formulation

After the VNF is placed on a physical node in \mathcal{G} , the path for link mapping follows the order of the SFC, to reduce the dimension of the action space and solve the problem appropriately. Formally, the VNF Placement and traffic Routing (VNF-PR) problem can be defined as follows:

Definition 6.1 Given are a network $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$ and a set of requests R , for each request $r(\xi, \mathcal{F}, D) \in \mathcal{R}$, the VNF-PR problem is to place the VNFs on N and to route the traffic in a specific order, such that the network utility $\sum_{r \in R} U^r$ is maximized.

The VNF placement and traffic routing problem is known to be NP-hard (Zhang et al. 2017). We first formally present the VNF-PR problem with objectives and constraints. We begin with some necessary variables.

Boolean Variables

- $x_n^{r,f}$: It is 1 if r 's requested VNF f is placed on n ; and 0 otherwise.
 $y_{u,v}^r$: It is 1 if a path between u and v is used for delivering the requested task of r ; and 0 otherwise.

Objective

$$\max \sum_{r \in R} U^r \quad (6.8)$$

Placement Constraints

$$\sum_{n \in N} x_n^{r,f} = 1 \quad \forall r \in R, f \in \mathcal{F}_r \quad (6.9)$$

Node Capacity Constraints

$$\sum_{r \in R} \sum_{f \in \mathcal{F}_r} x_n^{r,f} \cdot C_n^f \leq \delta_n \quad \forall n \in N \quad (6.10)$$

Link Capacity Constraints

$$\sum_{r \in R} \sum_{e \in \mathcal{E}} y_{u,v}^r \cdot C_{u,v} \leq \eta_e \quad \forall u, v \in N \quad (6.11)$$

Delay Constraints

$$\sum_{f \in R} \sum_{n \in N} x_n^{r,f_i} \cdot d_n^f + \sum_{u,v \in N} \sum_{e^{u,v} \in \mathcal{E}} y_{u,v}^r \cdot d_{u,v} \leq D \quad \forall r \in R \quad (6.12)$$

Equation (6.8) maximizes the network utility. Equation (6.9) ensures that for each requested VNF f , it must be placed on one node in the network. Equation (6.10) indicates that each node's capacity is not violated. Equation (6.11) ensures that each link's load is not greater than η_e . Equation (6.12) ensures that the total delay for each request does not exceed D .

Without considering the variability of network states, the optimization problem can be solved by using ILP or heuristic algorithms (Zhang et al. 2017). However, it is non-trivial to use those techniques to model real-time metrics. DRL can capture the dynamic states of networks. Therefore, in Sect. 6.4, we exploit DRL to solve the VNF placement and traffic routing problem.

6.4 Deep Reinforcement Learning

In this section, we begin with the DRL model design in Sect. 6.4.1. This is a Markov decision process including state, action, and reward. Then we propose our A-DDPG framework to solve the VNF-PR problem in Sect. 6.4.2.

6.4.1 DRL Model Design

In the DRL model, three elements, which are based on a Markov decision process, can be described by a tuple (S, A, R) , referring to the state space, action space, and reward, respectively. To deal with the real-time network state changes caused by VNF-PR, we consider a discrete-time period T . From state S , after taking action A , the agent transfers to the next state S' , and generates a return R (reward or penalty) that guides the DRL. Then the agent makes new decisions and the procedure repeats. We define the 3-tuple (S, A, R) for the VNF-PR problem as follows:

State The state space can be described by a vector $S = \{s_1, s_2, s_3, \dots, s_T\}$, where each term $s_t \in S$ represents the remaining resources of the virtual links and nodes at time t . T represents a time period.

Action The action of any agent is a vector A with each term $a \in A$ representing VNF placement and traffic routing. Therefore, we define an action as $a = \{x_n^{r,f}, \xi_r^f\} \forall r \in \mathcal{R}, f \in \mathcal{F}_r, n \in \mathcal{N}$.

Reward The value of the reward is a value indicating correct action. Whether the action can bring profit and whether the user's demand is met is taken as the criteria to affect the reward value. The reward received at time-slot t is set as the objective of our utility function, defined as $R = U^r = u_{s,a}^r - u_{s,a}^c$ according to Eq. (6.1). If the action brings benefits to the network and saves cost, the reward will be a positive value to encourage the operation. However, if the cost increases or the constraint is violated, a negative reward is returned.

6.4.2 A-DDPG Framework

Incorporating the above definitions, we start to design our A-DDPG framework. We first present the attention model in Sect. 6.4.2.1. Then, we describe the Actor-Critic network in Sect. 6.4.2.2. Finally, we complete the algorithm in Sect. 6.4.2.3.

6.4.2.1 Attention Model

We argue that the neighbor nodes of each server node are of great significance to the performance of VNFs placement and routing. Therefore, we introduce an attention mechanism into the neural network, which allows the network to better obtain neighbor node information. The attention mechanism assumes that weights of nodes measure matching degree between neighbors in the attention layer.

Formally, we define h_i as the state of the node, and the key k_i , value v_i , and q_i can be calculated as follows:

$$k_i = W^K \cdot h_i, v_i = W^V \cdot h_i, q_i = W^Q \cdot h_i \quad \forall i \in \mathcal{N} \quad (6.13)$$

where W^Q , W^K , and W^V are the parameter matrices that can be learned, and h_i is equal to s_i defined in Sect. 6.4. The compatibility sc_i^j of the query q_i of node i with the key k_j of node j is calculated as the active function (e.g., dot-product):

$$sc_i^j = active(q_i, k_j) \quad \forall i, j \in \mathcal{N} \quad (6.14)$$

According to Eq. (6.14), we compute the weights a_i^j using a softmax function:

$$a_i^j = softmax(sc_i^j) = \frac{e^{sc_i^j}}{\sum_{i=1}^N e^{sc_i^j}} \quad \forall i, j \in \mathcal{N} \quad (6.15)$$

Then the attention value is equal to:

$$at((k_i, v_i), q_i) = \sum_{i=1}^N a_i^j \cdot v_i = \sum_{i=1}^N \frac{e^{sc_i^j}}{\sum_{i=1}^N e^{sc_i^j}} \cdot v_i \quad \forall i, j \in \mathcal{N} \quad (6.16)$$

6.4.2.2 Actor-Critic Network Design

Our A-DDPG algorithm constructs a deep reinforcement network fitting state-action value function to solve the state space explosion problem. We use the Actor-Critic network structure, in which the Actor and Critic networks both adopt double-networks, namely the main network and the target network. Therefore, our framework has four networks for DRL agent training to solve the VNF-PR problem. The four network structures are the same, as shown in Fig. 6.1. In the Actor-Critic network, the observation state and action are taken as the input, and two-layer hidden networks with 32 and 16 neurons are used to process the input to understand the deployment status of the physical server in the current network. Then the processing results are imported into a full connection layer with *Relu* function. The *Relu* function is a non-linear activation function in neural network. The network

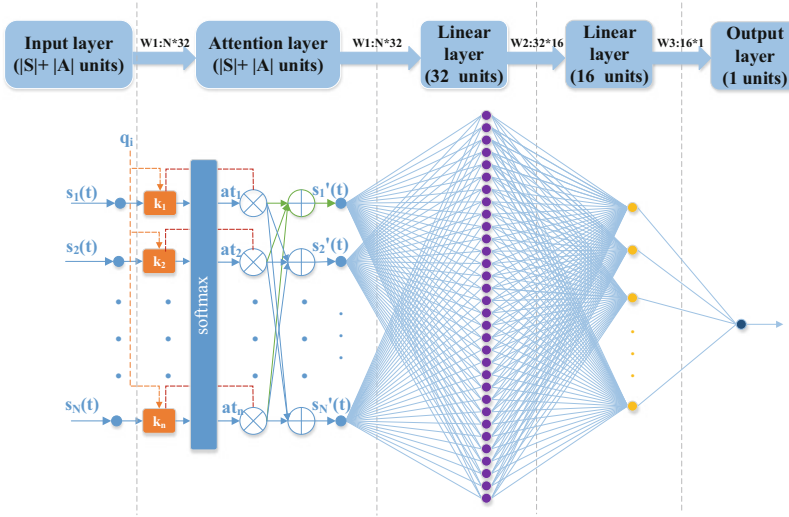


Fig. 6.1 Actor-critic network design of our A-DDPG framework

uses the cumulative reward as the target value and the expected cumulative reward as the predicted value. The purpose of training is to make the predicted value as close as possible to the target value. The equation to define the loss function is as follows:

$$L(\theta) = \frac{1}{B} \sum_t (y_t - Q(s_t, a_t | \theta^Q))^2 \quad (6.17)$$

where θ represents the parameter of actor for sampling and B indicates the size of the replay buffer. Then, the partial derivative of the loss function to the weight of the neural network can be calculated as:

$$\frac{\partial L(\theta)}{\partial \theta} = \frac{1}{B} \sum_t (y_t - Q(s_t, a_t | \theta^Q))^2 \frac{\partial Q(s_t, a_t | \theta^Q)}{\partial \theta} \quad (6.18)$$

where $Q(s_t, a_t | \theta^Q)$ refers to the long-term return of an action, taking a specific a_t under a specific policy from the current state s_t , and y_t denotes the predicted return. Through multiple iterations of the gradient descent method and the back-propagation mechanism, the $Q(s_t, a_t | \theta^Q)$ value can be obtained.

6.4.2.3 Algorithm Design

The A-DDPG framework is divided into three main processes: observation process, training process, and running process, as shown in Fig. 6.2.

Observation Process

The whole observation process mainly consists of two parts: Step 1 and Step 2 in Fig. 6.2.

Step 1, which is the initial phase of observation, begins with the agent interacting with the environment. The step is mainly used to obtain the initial state and store the historical samples. More specifically, the agent obtains the original state of the environment (including the routing status of the server, etc.) and collects environmental historical samples that need to be trained. These samples contain a

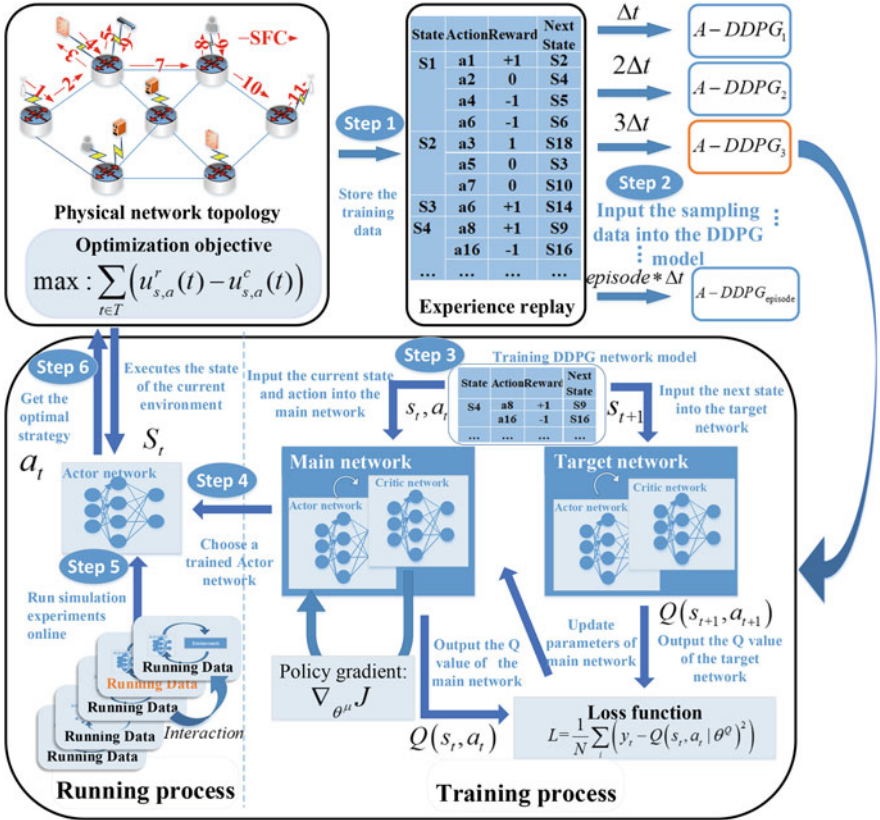


Fig. 6.2 The A-DDPG framework

Algorithm 4: A-DDPG training procedure

```

1: Randomly initialize critic network  $Q(s, a|\theta^Q)$  and actor  $\mu(s|\theta^\mu)$  with weights  $\theta^Q$  and  $\theta^\mu$ 
2: Initialize target network  $Q'$  and  $\mu'$  with  $\theta^{Q'}$ ,  $\theta^{\mu'}$ 
3: Initialize replay buffer R
4: for  $episode = 0, 1, \dots, M$  do
5:   Initialize a random process  $\mathbb{N}$ , choose a time  $\tau$  to get a distribution  $\mathbb{N}_\tau$  for action
   exploration
6:   Receive initial observation state  $s_1$ 
7:   for  $t = 0, 1, \dots, T$  do
8:     Select action  $a_t = \mu(s_t|\theta^\mu) + \rho$  according to the current policy  $\theta^\mu$  and exploration
     noise  $\rho$ , where  $\rho$  is randomly chosen from  $\mathbb{N}_\tau$ 
9:     Execute action  $a_t$  and observe reward  $r_t$  and observe new state  $s_{t+1}$ 
10:    Store a random minibatch of  $N$  transitions  $(s_t, a_t, r_t, s_{t+1})$  from replay buffer
11:    Set  $y_t = r_t + \gamma Q'(s_{t+1}, \mu'(s_{t+1}|\theta^{\mu'}))|\theta^{Q'}$ 
12:    Update critic by minimizing the loss:
      
$$L = \frac{1}{B} \sum_t (y_t - Q(s_t, a_t|\theta^Q))^2$$

13:    Update the actor policy using the sampled gradient:
      
$$\nabla_{\theta^\mu} J \approx \frac{1}{B} \sum \nabla_a Q(s_t, a_t|\theta^Q) \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_t}$$

14:    Update the target network:
      
$$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$$

      
$$\theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$$

15:   end for
16: end for

```

sequence composed of the initial state s_t , action a_t , reward r_t , and the next state s_{t+1} . Then the samples are put into the replay memory (Step 1). Subsequently, the action is obtained according to the ϵ greedy strategy (since the neural network parameters are also randomly initialized, the parameters will not be updated at the step, and they are collectively called random actions). Next, ϵ is reduced according to the number of iterations. Afterwards, the simulator performs the selecting action and returns a new state and reward.

Step 2 of the observation process begins in the replay buffer. The samples of the replay buffer must be independent and identically distributed. However, the adjacent training samples of RL are related to each other. Therefore, an experience replay and target network are introduced into the network to break up the correlation. More specifically, the previous state s_t , action a_t , new state s_{t+1} , and reward r_t are assembled into (s_t, a_t, r_t, s_{t+1}) to enter the replay memory for parameter updating. Finally, the action to be executed next is selected according to the ϵ greedy strategy, and the cycle is repeated until the number of iterations reaches the limitation (depending on the size of the replay buffer).

Training Process

After the observation process, the sufficient samples required for A-DDPG training are obtained. Algorithm 4 describes the training process of Step 3 in Fig. 6.2. More specifically, the agent first initializes the weights of the critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ as θ^Q and θ^μ , respectively (Line 1), and the target networks are cloned from the critic and actor networks (Line 2). Then a batch of data is sampled from the replay memory R as the input parameters of the A-DDPG model (Line 3). During the m -th episode (Line 4), in order to increase the randomness of the training process and the coverage of learning, the model adds random noise to the selected action, the noise is simulated by \mathbb{N}_τ (Line 5), and the agent receives the first state s_0 of the current environment (Line 6). During the t -th time-slot (Line 7), A-DDPG adds exploration noise to the current policy (Line 8). Next, the agent executes action a_t and transfers to next state s_{t+1} , rewards r , and decides whether to terminate the state (Line 9). Subsequently, the agent stores the quadruple (s_t, a_t, r_t, s_{t+1}) into the experience replay B (Line 10). The next step is to update the actor network and the critic network. First we need to prepare significant training data: (1) calculate the predicted baseline y_t (Line 11) and (2) calculate the policy $\mu(s_t|\theta^\mu) + \rho$ and Q baseline $Q(s_t, a_t|\theta^Q)$. Drawing on the DDPG method, the value loss function is the mean square error (MSE) of the predicted baseline and the actual baseline according to (6.17) (Line 12). Afterwards, gradient descent (6.18) is used to train the neural network (Line 13), and the weight parameters of the network are updated regularly (Line 14). If s_{t+1} is the terminal state, the current round of iteration is finished, otherwise, it goes to Line 8. The total number of episodes is denoted by M and each training episode contains T training rounds. The above procedures iterate until convergence or reaching the predefined episode bound.

Running Process

The whole running process of the A-DDPG algorithm mainly consists of three parts: Step 4, Step 5, and Step 6 in Fig. 6.2.

In Step 4, the well-trained A-DDPG model is selected, and the long-term cumulative reward of the action is preliminarily evaluated by inputting the current state. The purpose is to avoid operations with poor performance and statistically select operations that may achieve good performance to optimize the solution space size. In Step 5, the performance of each action in the optimized solution space, based on the predicted value in the simulated environment, is evaluated to obtain rewards, and the results are recorded in the database to further update the A-DDPG model. In Step 6, the action with greatest reward is performed in the physical network.

6.5 Performance Evaluation

6.5.1 Simulation Settings

The simulation experiments are all implemented on an Intel (R) Core (TM) i7 Windows 10 64-bit system. We conduct simulations on a 50-node network: We generate 50 nodes in the network and draw a link for each node pair. Moreover, the network parameters, computing capabilities, and traffic requests are randomly generated similar to existing works (Li et al. 2017; Gardner et al. 2015). The node capacity is randomly distributed in $[1, 100]$. For each link, its capacity is randomly assigned from the range $[2, 4]$ Gb/s and its delay takes value in $[30, 50]$ ms. We simulate $[10, 100]$ requests and each request requires an SFC consisting of 3 to 6 different VNFs (e.g., firewall, NAT, IDS, load balancer, WAN optimizer and flow monitor) according to Savi et al. (2015). For the unit cost of Eqs. (6.4), (6.5), (6.6), we set $\Phi^{op} = 0.2$, $\Phi^{de} = 0.4$, and $\Phi^{tr} = 0.1$.

We set our attention-based deep neural network structure with an input layer, an output layer, and 3 hidden layers. The 3 hidden layers comprise an attention layer and 2 fully connected layers. The number of hidden nodes of the 2 fully connected layers is 32 and 16, respectively. The hyperparameters for DRL are shown in Table 6.2, and the target network parameters are updated once every 200 steps.

The implementation of the A-DDPG algorithm is divided into three modules. The first is the construction of the underlying network environment, including the simulation of network topology nodes and link resources. Next, the request generation module. Each request contains an SFC and each SFC contains 3 to 6 VNFs. Finally, the DRL algorithm module runs the A-DDPG algorithm. Once the agent is well-trained after convergence, it can make the right decision for the VNF-PR problem.

We compare our A-DDPG method with three counterpart algorithms: DDPG, NFVdeep (Xiao et al. 2019), and Q-learning.

DDPG DDPG is a model-free DRL algorithm to solve the VNF-PR problem. The difference with A-DDPG is that it does not add an attention mechanism. For each request $r(\xi, \mathcal{F}, D) \in \mathcal{R}$, the DDPG algorithm tries to place $f \in \mathcal{F}_r$ on node $n \in \mathcal{N}$ only considering its current remaining resource capacity, regardless of the neighbors' states. Moreover, it considers actor and critic networks with two fully connected layers, in which the number of nodes is 32 and 16, respectively. Meanwhile, we set $\alpha = 0.01$, the batch size is 64 and $\gamma = 0.8$, which is consistent with the parameter settings of A-DDPG.

Table 6.2 Hyperparameters for DRL

Replay buffer size	10,000	Learning rate	0.1, 0.01, 0.001
Hidden nodes	32, 64	Number of episodes	3000
Discounted factor	0.8	Hidden layer	3

NFVdeep NFVdeep is a state-of-the-art method for VNF-PR problems. NFVdeep methods are a type of RL technique that relies upon optimizing parameterized policies concerning the expected return (long-term cumulative reward) by gradient descent. The policy gradient of the parameter θ is defined as:

$$\nabla_{\theta} J(\theta) = \left(\frac{\partial \mathcal{J}(\theta)}{\partial \theta_1}, \dots, \frac{\partial \mathcal{J}(\theta)}{\partial \theta_n} \right) \quad (6.19)$$

where the parameter θ is updated as $\theta_{i+1} = \theta_i + \alpha \nabla_{\theta_i} \mathcal{J}(\theta_i)$,

α is the learning rate and n is the number of neurons. During the training process, the agent processes one VNF of SFC in each MDP state transition. Then the reward for each state s is calculated, and the physical network gives the reward to the NFVdeep agent. Subsequently, the NFVdeep agent is trained for updating the policy circularly until the reward converges.

Q-Learning Q-learning is an off-policy RL method where the agent queries the Q-value table to make decisions. For each request $r(\xi, \mathcal{F}, D) \in \mathcal{R}$, the Q-learning algorithm tries to place $f \in \mathcal{F}_r$ on a well-resourced node $n \in \mathcal{N}$, such that the total delay of r shall be less than D . Afterwards, the agent calculates the cumulative reward by the utility function of the network. We set the learning rate to 10^{-2} and halved it every 200 episodes. Meanwhile, we set $\gamma = 0.8$ to ensure the best performance of the algorithm. The Q-value is updated as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (6.20)$$

where s represents the state at a certain moment, a_t indicates the action taken at that moment, $Q(s_t, a_t)$ denotes the Q-value corresponding to the (state, action) pair, r reflects the reward function, $Q(s_{t+1}, a_{t+1})$ represents the state transition function, and a_{t+1} denotes the action corresponding to the next state.

For the above three methods, we compare their network utility, delay, and running time. Among them, the network utility reflects the resource occupancy of the nodes and links of the network according to Eq. (6.1), and the total delay is calculated by the sum of each path and node processing delay to reflect the network utility of VNF placement and traffic routing.

6.5.2 Simulation Results

Figure 6.3a shows reward returned by A-DDPG under different learning rates (0.1, 0.01 and 0.001). The placement and routing of all VNFs are completed in each training episode. It can be seen from Fig. 6.3a that the learning rate affects the value of reward in the algorithm's training progress. The reason is that the learning rate represents the amount by which the weights are updated (a.k.a. the step size)

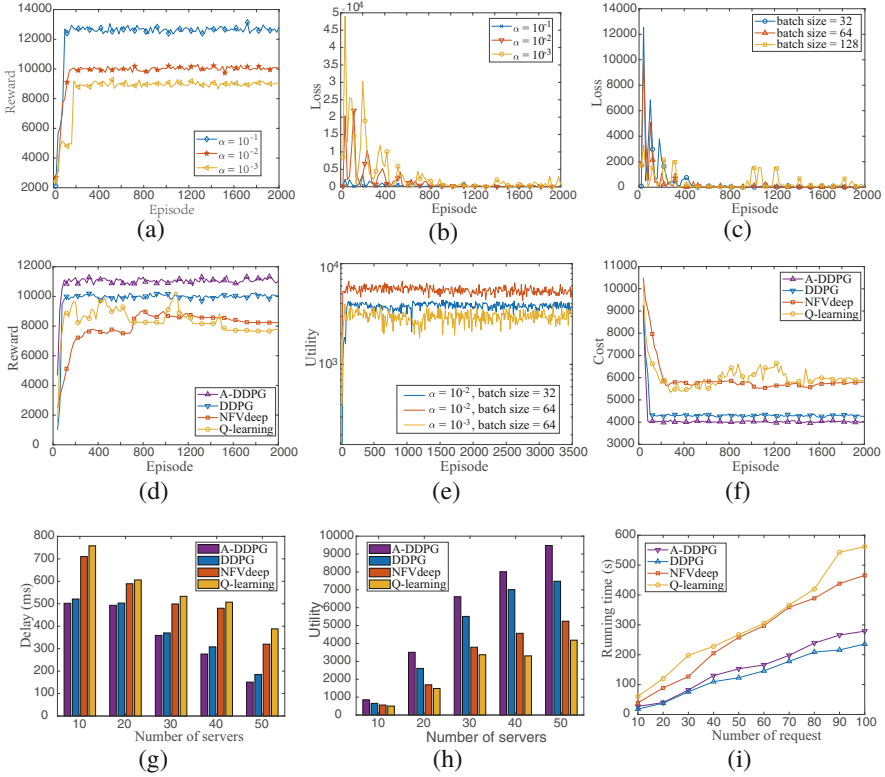


Fig. 6.3 Performance comparison of A-DDPG, DDPG, NFDveep, and Q-learning. (a) The reward returned by A-DDPG under different learning rates. (b) The loss value of the A-DDPG under different learning rates. (c) The loss value of A-DDPG with different batch sizes. (d) The reward returned by all the algorithms. (e) The influence of learning rate and batch size on the utility for A-DDPG. (f) The cost returned by all the algorithms. (g) The influence of the number of servers on the delay for all the algorithms. (h) The influence of the number of servers on the utility for all the algorithms. (i) The influence of the number of servers on the running time for all the algorithms

during training. Smaller learning rates may lead to a slower weight update, so more training episodes are needed to achieve convergence of reward, whereas larger learning rates cause rapid changes and require fewer training episodes. According to our simulation results, when the learning rate is 0.01, A-DDPG achieves the best performance in terms of reward. Therefore, we will take the best learning rate for comparison with other algorithms. Its learning speed is acceptable, and it leads to faster convergence of reward function.

Figure 6.3b shows the loss value of the A-DDPG method under different learning rates (0.1, 0.01 and 0.001). It can be seen from Fig. 6.3b that the learning rate affects the loss value in the algorithm's training step. The reason is that if the learning rate is too large, the loss function may directly exceed the global optimization of the learning process, whereas a small learning rate can cause the process to get stuck.

When the learning rate is small, the changing speed of the loss function is slow. In this context, it will greatly increase the convergence complexity of the network, and the loss function is easy to be trapped in a local minimum. Our simulation shows that the learning rate of 0.01 provides the best performance for A-DDPG.

Figure 6.3c shows the loss of A-DDPG with different batch sizes, where the batch sizes are 32, 64, and 128, respectively. As can be seen from Fig. 6.3c, as the episodes increase, the batch size will affect the value of the loss. We use batch gradient descent in the simulation to complete the iteration, which processes a portion of the samples at a time. A small portion of the samples will bring a large variance, which will cause the loss function to oscillate and slow down the convergence speed of the algorithm, especially when the network is complex. If the sample size is too large, the gradient estimation will be more accurate and stable, which may cause the neural network to converge to a poor local optimal solution point. Therefore, the batch size cannot be set too small or too large. According to our simulation results, the batch size can be set to 64. In addition, we find that after a series of shocks, the loss value in Fig. 6.3c can always be stable near 0 within a certain range. This indicates that our proposed algorithm can achieve convergence in VNF-PR.

Figure 6.3d shows the reward returned by all the algorithms. As the episodes increase, the value of the reward gradually converges. In particular, we find that the A-DDPG algorithm is stable after being trained for 200 episodes. Subsequently, the reward value of A-DDPG somewhat fluctuates. On the one hand, this is due to the random generation of network requests, and the reward value is related to the completion of the network request. On the other hand, when poor samples are selected, one may end up in a local optimum, which results in a low reward value. It can be observed from Fig. 6.3d that Q-learning and NFVdeep always return the lowest reward because they do not directly use the deep network to select actions. DDPG performs better due to its capability to select actions directly. However, it is not as good as A-DDPG, since it fails to capture the neighbors' states. The A-DDPG algorithm can always achieve the highest reward after 200 episodes of training among the simulated algorithms. The reason is that the A-DDPG agent takes action by additionally paying attention to the states of neighbors, and the correct behavior enables the agent to obtain positive rewards faster during training, which accelerates the learning process. The results from the training process imply that the A-DDPG agent is more intelligent than other agents.

Figure 6.3e depicts the influence of learning rate and batch size on the utility for A-DDPG. As the A-DDPG model iterates, the utility gradually converges towards a maximum where the model optimizes the weights. The utility value of $\alpha = 0.01$ is higher than that of $\alpha = 0.001$. We analyze that this is because the higher learning rate may miss the global optimization of the learning process, so it will cause the network to converge to a local optimum and obtain a low utility. Moreover, the speed of convergence when *batch size* = 64 is higher than the value when *batch size* = 32. This is because, as the batch size increases, the data processing speed becomes faster, which can reduce training time and enhance system stability.

Figure 6.3f shows the cost returned by all the algorithms. As can be seen from Fig. 6.3f, the training efficiency of Q-learning is lower than that of A-DDPG.

More specifically, the cost of NFVdeep fluctuates at 5800 after 200 episodes. The cost of Q-learning fluctuates at 6000 after 1800 episodes. The cost of A-DDPG stabilizes after 100 episodes with slight fluctuations at 4000. Given these points, A-DDPG converges faster than NFVdeep and Q-learning. This is because, during initial training, VNFs are randomly placed on the different servers, which incurs large operation and transmission costs. Through the training of Q-table and neural networks, Q-learning and A-DDPG agents can reduce unnecessary costs through training results. However, due to the use of neural networks, the A-DDPG agent is conducive to expressing complex network states. Compared with the discrete strategy in NFVdeep, A-DDPG can directly optimize the strategy (e.g., request rate) to meet the time-varying network states. Under the same conditions, it can process more network requests and reduce cost, thereby improving the processing capacity of the network. The result shows that an A-DDPG agent is more intelligent since it can achieve lower costs.

Figure 6.3g shows the influence of the number of servers on the delay for all the algorithms. As shown in Fig. 6.3g, the delay shows a downward trend as the number of servers increases. With the increase in the number of servers, it guarantees enough resources and applicable paths to accommodate requests. Our A-DDPG algorithm achieves a lower delay compared with the other three approaches. This is because, as the number of servers increases, the random topology becomes more complicated. The placement and routing of VNFs using NFVdeep and Q-learning can easily cause the server to fall into a local bottleneck due to performance degradation, whereas A-DDPG adds incentives for delay optimization in the reward function. The value of the reward increases more and more as delay decreases. Therefore, the node with the smaller delay will be selected to deploy the VNF in the strategic choice.

Figure 6.3h shows the influence of the number of servers on the utility for all the algorithms. As shown in Fig. 6.3h, A-DDPG achieves higher utility than the others. It reflects the superiority of A-DDPG in expressing decision-making in complex network environments. A-DDPG can obtain better utility under time-varying network states compared with the DDPG algorithm. NFVdeep performs well in an environment with 30 servers. Q-learning is not sensitive to the number of servers. In short, the A-DDPG method can attain greater utility by adopting an adaptive selection policy in a complex network environment.

Finally, we compare the running time performance of all the algorithms. As seen in Fig. 6.3i, the running time of Q-learning is significantly higher than those of A-DDPG, DDPG, and NFVdeep. In addition, when there are more than 80 requests, the running time of Q-learning increases rapidly. This is because when there are fewer requests, the servers have numbers of capacity available on-demand, and there exist more feasible solutions for the VNF-PR problem to achieve the best performance. However, with an increase in resource demands, the state space and action space in the Q-table greatly increase. In that case, finding the optimal strategy by looking up the table becomes difficult, and considering the complex calculations of nodes and links in large NFV networks, it takes more time to find the optimal strategy in large networks. Although A-DDPG and DDPG spend some more time to train the neural networks, after the training is completed and deployed, it only needs to use the well-

trained neural networks for reasoning. A-DDPG has more running time than DDPG. This is because A-DDPG's neural network architecture has one more attention layer than DDPG, which causes a slight increase in running time. Therefore, A-DDPG consumes reasonable running times due to the powerful representation capabilities of the neural networks.

6.6 Summary

As previously discussed in Chap. 2 for simple basic concepts in NFV, some heuristic algorithms are applied to VNF-PR problem in Chaps. 3–4. However, these methods are not applicable to the stochastic NFV network discussed in Chap. 5. To overcome the uncertainty caused by the stochastic network, in this chapter, we present a DRL framework with an attention mechanism, called A-DDPG, which consists of three processes: observation process, training process, and online running process. In the observation process, the agent first obtains historical samples through observation for training. Second, the agent updates the network parameters according to the historical samples and reduces the random exploration rate in the training process. Finally, the agent selects the trained network model and then executes the action of the maximum reward to get the optimized VNF-PR policy. By introducing an attention mechanism, we can focus on more critical information, such as the state of neighbors, to reduce the attention to other unnecessary nodes and improve the training efficiency of the model. Via extensive simulations, we find that our proposed algorithm A-DDPG can outperform the state-of-the-art in terms of network utility, delay, and cost.

References

- Allybokus Z, Perrot N, Leguay J, Maggi L, Gourdin E (2018) Virtual function placement for service chaining with partial orders and anti-affinity rules. *Networks* 71(2):97–106
- Bhamare D, Jain R, Samaka M, Vaszkun G, Erbad A (2015) Multi-cloud distribution of virtual functions and dynamic service deployment: open ADN perspective. In: *IEEE international conference on cloud engineering*, pp 299–304
- Bonfim MS, Dias KL, Fernandes SF (2019) Integrated NFV/SDN architectures: a systematic literature review. *ACM Comput Surv* 51(6):1–39
- Bunyakitanon M, Vasilakos X, Nejabati R, Simeonidou D (2020) End-to-end performance-based autonomous VNF placement with adopted reinforcement learning. *IEEE Trans Cogn Commun Netw* 6(2):534–547
- Chase JS, Doyle RP, Ims SD (2006) Multi-tier service level agreement method and system
- Courville RZ, Salakhudinov R, Bengio Y (2015) Show, attend and tell: neural image caption generation with visual attention. In: *International conference on machine learning*
- Cziva R, Anagnostopoulos C, Pezaros DP (2018) Dynamic, latency-optimal vNF placement at the network edge. In: *IEEE INFOCOM*
- Dinh-Xuan L, Popp C, Burger V, Wamser F, Hoffeld T (2020) Impact of VNF placements on qoe monitoring in the cloud. *Int J Netw Manag* 30(3):e2053

- Feng H, Llorca J, Tulino AM, Raz D, Molisch AF (2017) Approximation algorithms for the NFV service distribution problem. In: IEEE INFOCOM
- Gao T, Li X, Wu Y, Zou W, Huang S, Tornatore M, Mukherjee B (2020) Cost-efficient VNF placement and scheduling in public cloud networks. *IEEE Trans Commun* 68(8):4946–4959
- Gardner K, Borst S, Harchol-Balter M (2015) Optimal scheduling for jobs with progressive deadlines. In: IEEE INFOCOM, pp 1113–1121
- Golkarifard M, Chiasserini CF, Malandrino F, Movaghar A (2021) Dynamic VNF placement, resource allocation and traffic routing in 5G. *Comput Netw* 188:107830
- Gu L, Zeng D, Li W, Guo S, Zomaya A, Jin H (2019a) Deep reinforcement learning based VNF management in geo-distributed edge computing. In: IEEE ICDCS, pp 934–943
- Gu L, Zeng D, Li W, Guo S, Zomaya AY, Jin H (2019b) Intelligent VNF orchestration and flow scheduling via model-assisted deep reinforcement learning. *IEEE J-SAC* 38(2):279–291
- Guo S, Dai Y, Xu S, Qiu X, Qi F (2019) Trusted cloud-edge network resource management: DRL-driven service function chain orchestration for IoT. *IEEE IoT J* 7(7):6010–6022
- He N, Yang S, Li F, Trajanovski S, Kuipers FA, Fu X (2021) A-DDPG: attention mechanism-based deep reinforcement learning for NFV. In: IEEE IWQoS, pp. 1–10
- Henderson P, Islam R, Bachman P, Pineau J, Precup D, Meger D (2018) Deep reinforcement learning that matters. In: AAAI conference on artificial intelligence
- Khezri HR, Moghadam PA, Farshbafan MK, Shah-Mansouri V, Kebriaei H, Niyato D (2019) Deep reinforcement learning for dynamic reliability aware NFV-based service provisioning. In: IEEE GLOBECOM, pp 1–6
- Kuo J-J, Shen S-H, Kang H-Y, Yang D-N, Tsai M-J, Chen W-T (2017) Service chain embedding with maximum flow in software defined network and application to the next-generation cellular network architecture. In: IEEE INFOCOM, pp 1–9
- Li T, Magurawalage CS, Wang K, Xu K, Yang K, Wang H (2017) On efficient offloading control in cloud radio access network with mobile edge computing. In: IEEE ICDCS, pp 2258–2263
- Li S, Zhang S, Chen L, Chen H, Liu X, Lin S (2020) An attention based deep reinforcement learning method for virtual network function placement. In: IEEE ICC, pp 1005–1009
- Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D (2015) Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*
- Ma W, Sandoval O, Beltran J, Pan D, Pissinou N (2017) Traffic aware placement of interdependent NFV middleboxes. In: IEEE INFOCOM
- Nakanoya M, Sato Y, Shimonishi H (2019) Environment-adaptive sizing and placement of NFV service chains with accelerated reinforcement learning. In: IEEE IM, pp 36–44
- Nguyen DT, Pham C, Nguyen KK, Cheriet M (2019) Placement and chaining for run-time IoT service deployment in edge-cloud. *IEEE Trans Netw Serv Manag* 17(1):459–472
- Pei J, Hong P, Xue K, Li D (2018) Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system. *IEEE Trans Parallel Distrib Syst* 30(10):2179–2192
- Pei J, Hong P, Pan M, Liu J, Zhou J (2019) Optimal VNF placement via deep reinforcement learning in SDN/NFV-enabled networks. *IEEE J-SAC* 38(2):263–278
- Quang PTA, Bradai A, Singh KD, Hadjadj-Aoul Y (2019) Multi-domain non-cooperative VNF-FG embedding: a deep reinforcement learning approach. In: IEEE INFOCOM WKSHPS, pp 886–891
- Quang PTA, Hadjadj-Aoul Y, Outtagarts A (2020) On using deep reinforcement learning for VNF forwarding graphs placement. In: IEEE NoF, pp 126–128
- Ren H, Xu Z, Liang W, Xia Q, Zhou P, Rana OF, Galis A, Wu G (2020) Efficient algorithms for delay-aware NFV-enabled multicasting in mobile edge clouds with resource sharing. *IEEE Trans Parallel Distrib Syst* 31(9):2050–2066
- Ruiz L, Durán RJ, de Miguel I, Merayo N, Aguado JC, Fernández P, Lorenzo RM, Abril EJ (2020) Comparison of different protection schemes in the design of VNF-mapping with VNF resiliency. In: IEEE ICTON, pp 1–4
- Sallam G, Ji B (2019) Joint placement and allocation of virtual network functions with budget and capacity constraints. In: IEEE INFOCOM, pp 523–531

- Sampaio LS, Faustini PH, Silva AS, Granville LZ, Schaeffer-Filho A (2018) Using NFV and reinforcement learning for anomalies detection and mitigation in SDN. In: IEEE ISCC, pp 00432–00437
- Savi M, Tornatore M, Verticale G (2015) Impact of processing costs on service chain placement in network functions virtualization. In: IEEE conference on network function virtualization and software defined network (NFV-SDN), pp 191–197
- Scazzariello M, Ariemma L, Di Battista G, Patrignani M (2020) Megalos: a scalable architecture for the virtualization of network scenarios. In: IEEE NOMS, pp 1–7
- Shah HA, Zhao L (2020) Multi-agent deep reinforcement learning based virtual resource allocation through network function virtualization in Internet of Things. *IEEE IoT* 8(5):3410–3421
- Shannon CE (2001) A mathematical theory of communication. *ACM SIGMOBILE* 5(1):3–55
- Sharma GP, Tavernier W, Colle D, Pickavet M (2020) VNF-AAPC: accelerator-aware VNF placement and chaining. *Elsevier Comput Netw* 477:107329
- Soualah O, Mechtri M, Ghribi C, Zeghlache D (2019) Online and batch algorithms for VNFs placement and chaining. *Comput Netw* 158:98–113
- Sun P, Lan J, Li J, Guo Z, Hu Y (2020) Combining deep reinforcement learning with graph neural networks for optimal VNF placement. *IEEE Commun Lett* 25(1):176–180
- Tong R, Xu S, Hu B, Zhao J, Jin L, Guo S, Li W (2020) VNF dynamic scaling and deployment algorithm based on traffic prediction. In: IEEE IWCMC, pp 789–794
- Verbrugge S, Colle D, Pickavet M, Demeester P, Pasqualini S, Iselt A, Kirstädter A, Hülsermann R, Westphal F-J, Jaeger M (2006) Methodology and input availability parameters for calculating OpEx and CapEx costs for realistic network scenarios. *J Opt Netw* 5(6):509–520
- Xiao Y, Zhang Q, Liu F, Wang J, Zhao M, Zhang Z, Zhang J (2019) NFVdeep: adaptive online service function chain deployment with deep reinforcement learning. In: Proceedings of the international symposium on quality of service. ACM, New York, p 21
- Yala L, Frangoudis PA, Ksentini A (2018) Latency and availability driven VNF placement in a MEC-NFV environment. In: IEEE GLOBECOM, pp 1–7
- S. Yang, F. Li, S. Trajanovski, R. Yahyapour, and X. Fu (2021) Recent advances of resource allocation in network function virtualization. *IEEE Trans Parallel Distrib Syst* 32(2):295–314
- Zhang Z, Li Z, Wu C, Huang C (2017) A scalable and distributed approach for NFV service chain cost minimization. In: IEEE ICDCS, pp 2151–2156

Chapter 7

Summarization and Future Work



7.1 Summarization of This Book

NFV enables to replace dedicated hardware implementations with software instances and it provides more possibilities for large cost savings, high flexibility and scalability, short deployment cycles, etc.¹ These advantages of NFV makes it very promising to become network provisioning service paradigm in the next generation. This book focuses on a dedicated but important aspect of NFV, that is, resource allocation problem in NFV. More specifically, we first summarize and generalize four fundamental resource allocation problems in NFV, namely (1) the VNF Placement and Traffic Routing (VPTR) problem, (2) the VNF Placement (VNFP) problem, (3) the TRaffic Routing (TRR) problem, and (4) the VNF Redeployment and Consolidation (VRC) problem. If we do not induce any other QoS factors such as delay, and only network cost is taken into account, these problems ask for whether a corresponding solution exists when only the (basic) link and node capacity constraints are satisfied. Nevertheless, even under this case, the VPTR problem together with all its variants are still NP-hard in a generic case, which indicates the difficulty of the problem(s). So far, there is no approximation algorithm to solve the problem in the generic case. Even though some approximation algorithms (e.g., Fend et al. 2017; Guo et al. 2018) are proposed, they simplify the problem inputs or constraints to some extent. We present an uniform ILP formulation for these problems and also survey the current mainly adopted approaches to solve these four problems and analyze their pros and cons.

¹ Parts of this chapter is reprinted from Yang et al. (2021), with permission from IEEE.

After that, we study the Virtual Network Function (VNF) placement and routing problem by considering delay and availability QoS factors in Chap. 3. Our contributions are:

- We propose a model to quantitatively calculate the traversing delay for a flow in a SFC.
- We present a model to quantitatively measure the SFC availability for NFV resiliency.
- We propose an exact Integer Nonlinear Programming (INLP) formulation and an efficient heuristic for placing and routing the delay-sensitive virtual network functions with and without resiliency concerns.
- We validate and investigate the performance of the proposed algorithms via extensive simulations.

When SFC delay constraint is additionally taken into account, we found that the VPTR problem and its variants become more difficult to solve. The reliability (level) definition varies by different literature, which results in different problem input and constraints as well as solutions. Some literature only considers node/link backup, while some literature additionally takes into account the node and link availability. When $k \geq 2$ disjoint SFCs are provided, the service can recover from $k - 1$ simultaneously node and/or link failures. However, this approach ignores the node/link failure probability and its cost is high. On the contrary, the proposed SFC availability calculation is more exact and can quantitatively measure how reliable the NFV service is provided, even though the node and link availability value is not easy to achieve.

We then continue to address the VNF placement and routing problem in edge clouds by proposing a randomized approximation algorithm in Chap. 4. Our contributions are:

- We analyze the traversing delay calculation in edge clouds for both totally ordered SFC and partially ordered SFC.
- We define the Delay-aware VNF Placement and Routing (DVPR) problem in edge clouds and prove it is NP-hard.
- We propose a randomized rounding approximation algorithm to solve the DVPR problem.
- We conduct extensive simulations to validate the performances of the proposed algorithm with three heuristics.

In the context of edge clouds, the VPTR problem needs to additionally consider the user's location as well as both the wireless transmission between the user and edge server and wireline link transmission between edge server and cloud server. In this sense, jointly considering multiple QoS parameters will make this problem difficult to solve.

In Chap. 5, we assume the link delay and bandwidth are random variables (stochastic) and assume their cumulative distribution functions (CDF) are known, which can be easily achieved based on historical data. Suppose that the VNFs are

placed in the network, we investigate the TRR problem. Our key contributions are as follows:

- We mathematically model the stochastic link weight in terms of delay and bandwidth. More specifically, given that the link delay and bandwidth follow a known distribution, we show how to calculate the delay and bandwidth in an SFC such that their realizing probabilities satisfy the required values.
- We define the Traffic Routing problem in Stochastic NFV Networks and prove it to be NP-hard. We further formulate this problem as an exact optimization solution.
- We devise a tunable heuristic that first discretizes the network by k -samplings, and then dynamically applies Tunable Accuracy Multiple Constraints Routing Algorithm (TAMCRA) to find the multi-constraint path for each adjacent VNF pair.

In Chap. 6, we target how to apply Deep Reinforcement Learning (DRL) to solve the VNF placement and routing problem in NFV. We propose a novel Attention mechanism-based Deep Deterministic Policy Gradients (A-DDPG) framework, using the Actor-Critic network structure, in which both the Actor and Critic networks adopt double networks (namely the main network and the target network). The insights on this domain are:

- Since an ML-based approach cannot guarantee to find an optimal solution due to the use of training, can we improve the accuracy of ML-based approach?
- How to further reduce the convergence/training time of ML-based approach.
- What is the performance ratio of using ML-based approach for solving NFV resource allocation problem in theory (if any).
- Apart from DRL related approach, can some other ML-based approaches be used to solve NFV resource allocation problem? (e.g., graph neural networks)

7.2 Emerging Topics and Future Works

There are still open domains or interesting topics that need to be considered in depth about resource allocation in NFV which we will suggest below.

7.2.1 *State-of-the-Art*

Since the VPTR problem and its variants are proved to be NP-hard, approximation algorithms with proved approximation performance ratio have been devised, but with an assumption of a loose constraint (e.g., non-ordered SFC, a set of placement and routing configurations are known.) of the problem. We even found that no approximation algorithm has been devised to solve the VRC problem. Hence, we

still need to further investigate the “hardness” of the addressed problems, analyze the problem properties and devise approximation algorithms (in the general form) or efficient heuristics. Moreover, in this survey, we assume a single-path routing scheme (unless otherwise specified), which means that the data packets cannot be split through an SFC. Therefore it is interesting to further study the VPTR problem with multiple QoS parameters when multipath routing (Pham and Pham 2016; Wang et al. 2018; Fend et al. 2017) or multicast routing (Xu et al. 2017) is allowed.

7.2.2 *Security-Aware Resource Allocation*

As NFV yields numerous benefits such as lower CAPEX and OPEX, the software-enabled functions are vulnerable to a number of security threats (Lal et al. 2017), compared to the hardware-implemented middleboxes. Even though there is research (see Pattaranantakul et al. 2018 and work therein) that analyzes security threats and conduct studies on security mechanisms that are applied in traditional scenarios and in NFV environments, the security-aware resource allocation in NFV receives less attention. For example, how to securely place VNFs and (re)route traffic to defend against DDoS attack (Rashidi et al. 2017), VM escape attack or other threats. Among the security-related research in NFV, Guan et al. (2018) claim a subset of network nodes in the network are “key” nodes, and present a Path Routing Betweenness Centrality (PRBC) metric to represent this group of nodes, which implies the maximum probability that packets go through at least one node in this group. They subsequently present a successive heuristic to place Virtual Security Network Function (VSNF) such as firewall and intrusion detection on the calculated PRBC nodes. Park et al (2017) presents a light intrusion detection system by utilizing a chain of network functions in NFV based on ClickOS. In Doriguzzi-Corin et al. (2017, 2020), the security constraints in NFV refer to that VSNF must be placed on a certain subset of nodes (e.g., close to the user for a shorter delay) and prevented to be placed on a certain subset of nodes (e.g., critical regions from potentially malicious user traffic). An exact ILP and an efficient heuristic are proposed in (Doriguzzi-Corin et al. 2020) to solve the VNF placement and routing problem where both delay and security are taken into account. Similarly, in (Jmila and Blanc 2019), the security constraints include: (1) a set of VNFs that can be co-located, (2) a set of VNFs that should be co-located, (3) a set of VNFs that should not be co-located, and (4) a set of VNFs that should not share instances. Shameli-Sendi et al. (2019) define a set of network security defense patterns, which means the sequence/relation among required VNFs. More specifically, for any two VNFs, the patterns include, unordered, ordered, composition, location-aware, collaborative, etc. According to these defined security patterns, Shameli-Sendi et al. (2019) further propose a partitioning and segmentation-based heuristic to solve the VNFP problem.

7.2.3 Wireless Virtual Network Functions and Other Network Application Domains

While the majority of work as we summarized above have been directed to the scenario where data flow traverses in wireline networks in an SFC, the VPTR problem over wireless networks has received less attention. More specifically, when the nodes are connected via wireless channels, the traffic routing in terms of delay and packet loss among these nodes depends on the signal attenuation intensity, transmitting frequency or other metrics. The data transmission and communication models (Ramanathan [1999](#); Akyildiz et al. [2002](#)) differ from the ones in the wireline networks, and hence the VPTR problem should be reconsidered. As such, the NFV resource allocation problem in 5G mobile networks (Alhussein et al. [2018](#)), network slicing (Barakabitze et al. [2020](#)), IoT (Mouradian et al. [2016](#)) and other network application domains such as Content Delivery Networks (CDN) (Benkacem et al. [2018](#)) and Radio Access Networks (RAN) (Garcia-Saavedra et al. [2018](#)) need to be further investigated in future.

7.2.4 Mobility Management in NFV

In 5G-enabled mobile edge computing networks, the network service region is partitioned into different cells and each cell is located with edge servers. The end users stay in one cell whose NFV service request is accommodated by its local edge servers in this cell. Typically, the end user moves erratically, and when an end user roams to another cell, it will trigger handover delay. In this sense, the NFV services should be dynamically migrated (Chen and Liao [2019](#)) among multiple edge clouds to maintain the service performance (e.g., guarantee the predictability of migration times Tasiopoulos et al. [2017](#), the performance of network Kulkarni et al. [2017](#), and the QoS Kulkarni et al. [2018, 2020](#)). Therefore, it is necessary to consider the user's mobility by solving which VNF to migrate to which edge server, how to find appropriate paths within an SFC after migration, etc.

7.2.5 SDN and NFV

Software-Defined Network (SDN) (Nunes et al. [2014](#)) defines a network connection and management methodology that decouples the control plane from the data plane. In SDN, the network intelligence stays in a logically centralized software-controller (control plane), and network equipment (data plane) can be programmed via an open interface (like OpenFlow McKeown et al. [2008](#)). NFV and SDN are two independent innovation technologies, but they can work together for a joint flexible, efficient, agile network management and service development (Duan et al. [2016](#)).

The surveyed resource allocation algorithms in this survey can, for instance, be implemented in an SDN controller in an SDN-enabled NFV framework. Hence, it is necessary to consider network management and control related metrics such as network control overhead when designing resource allocation algorithms (Bu et al. 2017; Zhang et al. 2016).

References

- Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E (2002) Wireless sensor networks: a survey. *Comput Netw* 38(4):393–422
- Alhussein O, Do PT, Li J, Ye Q, Shi W, Zhuang W, Shen X, Li X, Rao J (2018) Joint VNF placement and multicast traffic routing in 5G core networks. In: IEEE global communications conference (GLOBECOM), pp 1–6
- Barakabitze AA, Ahmad A, Mijumbi R, Hines A (2020) 5G network slicing using SDN and NFV: a survey of taxonomy, architectures and future challenges. *Comput Netw* 167:106984
- Benkacem I, Taleb T, Bagaa M, Flinck H (2018) Optimal VNFs placement in CDN slicing over multi-cloud environment. *IEEE J Sel Areas Commun* 36(3):616–627
- Bu C, Wang X, Cheng H, Huang M, Li K, Das SK (2017) Enabling adaptive routing service customization via the integration of SDN and NFV. *J Netw Comput Appl* 93:123–136
- Chen Y-T, Liao W (2019) Mobility-aware service function chaining in 5G wireless networks with mobile edge computing. In: IEEE international conference on communications (ICC), pp 1–6
- Doriguzzi-Corin R, Scott-Hayward S, Siracusa D, Salvadori E (2017) Application-centric provisioning of virtual security network functions. In: IEEE conference on network function virtualization and software defined networks (NFV-SDN), pp 276–279
- Doriguzzi-Corin R, Scott-Hayward S, Siracusa D, Savi M, Salvadori E (2020) Dynamic and application-aware provisioning of chained virtual security network functions. *IEEE Trans Netw Serv Manag* 17(1):294–307
- Duan Q, Ansari N, Toy M et al. (2016) Software-defined network virtualization: an architectural framework for integrating SDN and NFV for service provisioning in future networks. *IEEE Netw* 30(5):10–16
- Feng H, Llorca J, Tulino AM, Raz D, Molisch AF (2017) Approximation algorithms for the NFV service distribution problem. In: IEEE INFOCOM
- Garcia-Saavedra A, Costa-Perez X, Leith DJ, Iosifidis G (2018) FluidRAN: optimized vRAN/MEC orchestration. In: IEEE INFOCOM
- Guan J, Wei Z, You I (2018) GRBC-based network security functions placement scheme in SDS for 5G security. *J Netw Comput Appl* 114:48–56
- Guo L, Pang J, Walid A (2018) Joint placement and routing of network function chains in data centers. In: IEEE INFOCOM
- Jmila H, Blanc G (2019) Designing security-aware service requests for nfv-enabled networks. In: IEEE 28th international conference on computer communication and networks (ICCCN), pp 1–9
- Kulkarni S, Arumathurai M, Ramakrishnan KK, Fu X (2017) Neo-NSH: towards scalable and efficient dynamic service function chaining of elastic network functions. In: ICIN. IEEE, Piscataway, pp 308–312
- Kulkarni SG, Liu G, Ramakrishnan KK, Arumathurai M, Wood T, Fu X (2018) Reinforce: achieving efficient failure resiliency for network function virtualization based services. In: Proceedings of the 14th international conference on emerging networking experiments and technologies, pp 41–53

- Kulkarni SG, Zhang W, Hwang J, Rajagopalan S, Ramakrishnan K, Wood T, Arumaithurai M, Fu X (2020) NFVnice: dynamic backpressure and scheduling for NFV service chains. *IEEE Trans Netw* 28(2):639–652
- Lal S, Taleb T, Dutta A (2017) NFV: security threats and best practices. *IEEE Commun Mag* 55(8):211–217
- McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J (2008) Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.* 38(2):69–74
- Mouradian C, Saha T, Sahoo J, Abu-Lebdeh M, Glitho R, Morrow M, Polakos P (2016) Network functions virtualization architecture for gateways for virtualized wireless sensor and actuator networks. *IEEE Netw* 30(3):72–80
- Nunes BA, Mendonca M, Nguyen XN, Obraczka K, Turletti T (2014) A survey of software-defined networking: past, present, and future of programmable networks. *IEEE Commun Surv Tutor* 16(3), 1617–1634 (2014)
- Park Y, Chandaliya P, Muralidharan A, Kumar N, Hu H (2017) Dynamic defense provision via network functions virtualization. In: *Proceedings of the ACM international workshop on security in software defined networks & network function virtualization*, pp 43–46
- Pattaranantakul M, He R, Song Q, Zhang Z, Meddahi A (2018) Nfv security survey: from use case driven threat analysis to state-of-the-art countermeasures. *IEEE Commun Surv Tutor* 20(4):3330–3368
- Pham T-M, Pham LM (2016) Load balancing using multipath routing in network functions virtualization. In: *IEEE RIVF international conference on computing & communication technologies, research, innovation, and vision for the future (RIVF)*, pp 85–90
- Ramanathan S (1999) A unified framework and algorithm for channel assignment in wireless networks. *Wirel Netw* 5(2):81–94
- Rashidi B, Fung C, Bertino E (2017) A collaborative DDoS defence framework using network function virtualization. *IEEE Trans Inform Forensics Secur* 12(10):2483–2497
- Shameli-Sendi A, Jarraya Y, Pourzandi M, Cheriet M (2019) Efficient provisioning of security service function chaining using network security defense patterns. *IEEE Trans Serv Comput* 12(4):534–549
- A. G. Tasiopoulos, S. G. Kulkarni, M. Arumaithurai, I. Psaras, K. Ramakrishnan, X. Fu, and G. Pavlou (2017) DRENCH: a semi-distributed resource management framework for NFV based service function chaining. In: *IFIP networking conference (IFIP networking) and workshops*. IEEE, Piscataway, pp 1–9
- Wang Q, Shou G, Liu Y, Hu Y, Guo Z, Chang W (2018) Implementation of multipath network virtualization with SDN and NFV. *IEEE Access* 6:32460–2470
- Xu H, Yu Z, Li XY, Huang L, Qian C, Jung T (2017) Joint route selection and update scheduling for low-latency update in SDNs. *IEEE/ACM Trans Netw* 25(5):3073–3087
- Yang S, Li F, Trajanovski S, Yahyapour R, Fu X (2021) Recent advances of resource allocation in network function virtualization. *IEEE Trans Parallel Distrib Syst* 32(2):295–314
- Zhang SQ, Tizghadam A, Park B, Bannazadeh H, Leon-Garcia A (2016) Joint NFV placement and routing for multicast service on SDN. In: *IEEE/IFIP network operations and management symposium*, pp 333–341