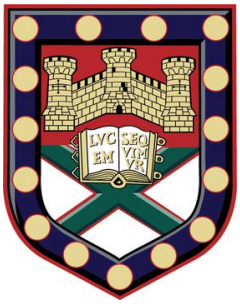
Learning Augmented Optimization for Network Softwarization in 5G



Xiangle Cheng

College of Engineering, Mathematics and Physical Sciences
University of Exeter

Submitted by Xiangle Cheng to the University of Exeter for the degree of
Doctor of Philosophy in Computer Science

Signature:.....

Computer Science Department August 2019

I would like to dedicate this thesis to my loving grandparents Fayan Cheng and Ronglian Hong for their greatest love and support.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 50 figures.

Xiangle Cheng
August 2019

Acknowledgements

I would like to thank my supervisors, Dr. Yulei Wu, Prof. Geyong Min for their kind helps and supports in supervising my works. In addition, I would like to thank Dr. Jia Hu, Dr. Chunbo Luo, Dr. Ke Li, Prof. Edward Keedwell, Prof. Xuming Fang, Prof. Albert Y. Zomaya and Prof. Ligang He for their helpful discussions and comments for my works.

In particular, I would like to thank all my family members, who always love me and support me. Also, I would like to have special thanks to Zhiwei Zhao, Lejun Chen, Haozhe Wang, Wang Miao, Chengqiang Huang, Yuan Zuo, Yang Mi, Zhengxin Yu, Yujia Zhu, He Zhang, Ning Wang and Joseph Billingsley, Mike Rosser, Carolyn Harris, Jerome Jeans and Polly Jeans for their supports and helps during my whole Ph.D. period.

Abstract

The rapid uptake of mobile devices and applications are posing unprecedented traffic burdens on the existing networking infrastructures. In order to maximize both user experience and investment return, the networking and communications systems are evolving to the next generation – 5G, which is expected to support more flexibility, agility, and intelligence towards provisioned services and infrastructure management. Fulfilling these tasks is challenging, as nowadays networks are increasingly heterogeneous, dynamic and expanded with large sizes. Network softwarization is one of the critical enabling technologies to implement these requirements in 5G. In addition to these problems investigated in preliminary researches about this technology, many new emerging application requirements and advanced optimization & learning technologies are introducing more challenges & opportunities for its fully application in practical production environment. This motivates this thesis to develop a new learning augmented optimization technology, which merges both the advanced optimization and learning techniques to meet the distinct characteristics of the new application environment. To be more specific, the abstracts of the key contents in this thesis are listed as follows:

- We first develop a stochastic solution to augment the optimization of the Network Function Virtualization (NFV) services in dynamical networks. In contrast to the dominant NFV solutions applied for the deterministic networking environments, the inherent network dynamics and uncertainties from 5G infrastructure are impeding the rollout of NFV in many emerging networking applications. Therefore, Chapter 3 investigates the issues of network utility degradation when implementing NFV in dynamical networks, and proposes a robust NFV solution with full respect to the underlying stochastic features. By exploiting the hierarchical decision structures in this problem, a distributed computing framework with two-level decomposition is designed to facilitate a distributed implementation of the proposed model in large-scale networks.

- Next, Chapter 4 aims to intertwin the traditional optimization and learning technologies. In order to reap the merits of both optimization and learning technologies but avoid

their limitations, promissing integrative approaches are investigated to combine the traditional optimization theories with advanced learning methods. Subsequently, an online optimization process is designed to learn the system dynamics for the network slicing problem, another critical challenge for network softwarization. Specifically, we first present a two-stage slicing optimization model with time-averaged constraints and objective to safeguard the network slicing operations in time-varying networks. Directly solving an off-line solution to this problem is intractable since the future system realizations are unknown before decisions. To address this, we combine the historical learning and Lyapunov stability theories, and develop a learning augmented online optimization approach. This facilitates the system to learn a safe slicing solution from both historical records and real-time observations. We prove that the proposed solution is always feasible and nearly optimal, up to a constant additive factor. Finally, simulation experiments are also provided to demonstrate the considerable improvement of the proposals.

- The success of traditional solutions to optimizing the stochastic systems often requires solving a base optimization program repeatedly until convergence. For each iteration, the base program exhibits the same model structure, but only differing in their input data. Such properties of the stochastic optimization systems encourage the work of Chapter 5, in which we apply the latest deep learning technologies to abstract the core structures of an optimization model and then use the learned deep learning model to directly generate the solutions to the equivalent optimization model. In this respect, an encoder-decoder based learning model is developed in Chapter 5 to improve the optimization of network slices. In order to facilitate the solving of the constrained combinatorial optimization program in a deep learning manner, we design a problem-specific decoding process by integrating program constraints and problem context information into the training process. The deep learning model, once trained, can be used to directly generate the solution to any specific problem instance. This avoids the extensive computation in traditional approaches, which re-solve the whole combinatorial optimization problem for every instance from the scratch. With the help of the REINFORCE gradient estimator, the obtained deep learning model in the experiments achieves significantly reduced computation time and optimality loss.

Table of contents

List of figures

List of tables

List of Publications

1. X. Cheng, Y. Wu, G. Min, A.Y. Zomaya, "Network function virtualization in dynamic networks: a stochastic perspective," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2218–2232, 2018.

2. L. Yan, X. Fang, Y. Fang, X. Cheng, "Dual-scheduler design for C/U-plane decoupled railway wireless networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 7, pp. 1842–1855, 2017.

Nomenclature

Acronyms / Abbreviations

3GPP Third Generation Partnership Project

AI Artificial Intelligence

AR Augmented Reality

BBU BaseBand Unit

C-RAN Cloud-Radio Access Network

DBN Deep Belief Network

DPI Deep Packet Inspection

E2E End-to-End

eMBB enhanced Mobile BroadBand

EU European Union

GAP Generalized Assignment Problem

GNN Graph Neural Network

HetNet Heterogeneous Network

ILP Integer Linear Program

IoT Internet of Things

IP Integer Programming

ISP Internet Service Provider

ITU International Telecommunication Union

KPI Key Performance Indicator

LP Linear Programming

LTE Long Term Evoluation

MAB Multi-Armed bandits

MAC Medium Access Control

MANO Management & Orchestration

MDP Markov Decision Process

MF-TDMA Multiple Frequencies Time Division Multiple Access

MIP Mixed Integer Programming

MIQP Mixed Integer Quadratic Programming

ML Machine Learning

mMTC massive Machine-Type Communications

NF Network Function

NFFG Network Function Forwarding Graph

NFV Network Function Virtualization

NFVI Network Function Virtualization Infrastructure

NLP Natural Language Processing

NN Neural Network

NP Network Provider

PDF Probability Distribution Function

QoE Quality of Experience

QoS Quality of Service

QP Quadratic Programming

RAT Radio Access Technology

RRH Remote Radio Heads

SDN Software Defined Networking

SDP SemiDefinite Programming

SFC Service Function Chain

SLA Service Level Agreement

SRA Stochastic Resource Allocator

UHD Ultra-High Definition

URLLC Ultra-Reliable and Low Latency Communications

VM Virtual Machine

VNE Virtual Network Embedding

VR Virtual Reality

4G The 4^{th} generation mobile communication technology

5G The 5^{th} generation mobile communication technology

Chapter 1

Introduction

1.1 Definition of Network Softwarization

The rising popularity of connected services and devices are actuating the explosive growth of networking traffic volume. According to [1], by 2021, global mobile data traffic will grow 7-fold, and the number of mobile users will be up to 5.5 Billion. The ever increasing proliferation of smart devices, introduction of new emerging applications, together with an exponential rise in service demands are posing a significant burden on existing networking infrastructures. This urges the network domains to evolve a pressing solution in 5G networks to handle the surging network traffic with satisfactory performance on both service experience and investment return. 5G systems, with significantly improved performances on data rates, capacity, latency, and Quality of Service (QoS)/Quality of Experience (QoE) , are expected to be the panacea of most of the current networks' problems [2], [3], [4].

Unlike the "one-size-fit-all" type of the current 4G technology, 5G era is touted as the generation of mobile networks that will offer multi-tenancy support and service-tailored connectivity [5]. Evolving from the success of 4G, 5G is anticipated to satisfy diverse business demands with even conflicting requirements. By allowing different parties to instantiate and run a software-based architecture, 5G becomes inherently a multi-tenant ecosystem, provisioning a truly differentiated service on top of a shared network infrastructure.

To meet these extreme demands, 5G leverages the benefits of network virtualization [6] to accommodate flexibility in providing carrier-grade differentiated networking services. The notion of network virtualization concentrates on the concept of a software-based representation of both the hardware and software resources for data and/or control-plane Network Functions (NFs). This lays the main foundation of network softwarization for 5G. **Network softwarization is the concept of designing, architecting, deploying and managing network components, primarily based on software programmability properties** [7]. It

enables flexibility, adaptability, and even total reconfiguration of a network on the fly based on timely requirements and behaviors. This, on the one hand, makes it possible to accommodate the diverse new emerging applications with 5G. On the other hand, more advanced technologies for intelligent operation and management are also required by considering cost and process optimization in the overall maintenance of the network lifecycle. This motivates this thesis to develop a new learning augmented optimization technology to facilitate the implementation of network softwarization in the context of 5G.

Apart from the extensively studied network virtualization technologies, two new enabling technologies to implement the vision of network softwarization in 5G are the concepts of network slicing and Network Function Virtualization (NFV).

Table 1.1 Typical 5G use cases and their QoS requirements

Use Cases	Examples	Requirements	Mobility
Enhanced Mobile Broad Band	4K/8K ultra high definition (UHD) video, hologram, Augmented Reality (AR), Virtual Reality (VR)	High capacity, video cache	Yes
Massive Machine Type Communications	Sensor Networks (smart metering, logistics, city, home, etc.)	Massive connection covering a very large area of mostly immobile devices	No
Ultra Reliable and Low Latency Communications	Autonomous driving, smart grid, remote surgery	Low latency and high reliability	Yes

1.1.1 Network Slicing

5G networks aim to support a number of vertical services that are characterized by diverse performance requirements. Many organizations e.g., International Telecommunication Union (ITU), Third Generation Partnership Project (3GPP), have categorized these services into serveral major use cases [8], [9]: enhanced mobile broadband (eMBB), massive machine-type communications (mMTC) and ultra-reliable and low latency communications (URLLC). As implied in Table 1.1, these emerging applications are subject to a wide spectrum of service requirements for latency, throughput, reliability, as well as the underlying resources, which are significantly beyond the capabilities of current networks. The diverse characteristics of these categories require different solutions. In contrast to current architectural model of "one size fits all", the network slicing technology is enabling the networking systems to support such functional and operational diversity.

According to the definition of 3GPP, **network slicing is a technology that enables the operator to create customized networks for different market scenarios with diverse**

requirements, e.g., in terms of functionality, performance and isolation [10]. A network slice is an end-to-end (E2E) logical network, including access and core networks, for a given application scenario to flexibly provide one or more network services according to the instructions of the controller [11]. A network slice instance consists of NFs and their corresponding computing, storage, and networking resources. The controller translates the use cases and business models into network slices, chains the relevant modular NFs, assigns the relevant performance configurations and maps all of these functions on the infrastructure resources. Hence, a network slice can span all domains of the network: software modules running on cloud nodes, specific configurations of the transport network supporting flexible location of functions, a dedicated radio configuration or even a specific radio access technology (RAT), as well as configurations of the 5G devices. Essentially, with network slicing an operator can deploy multiple logical networks over the same physical infrastructure. This, on the other hand, also makes it more challenging to deploy and operate these sliced network services under such complex environment.

The deployment of network slices relies on NFV and software-defined networking (SDN) paradigms. The former allows, through the virtualization of NFs, the achievement of a modular logical architecture and flexible placement of Virtual NFs (VNFs) throughout the network. The latter allows simplification of forwarding functions but, more importantly, a more advanced separation of control and user plane functionality [12]. As another key concept in this thesis, the details of NFV is introduced next.

1.1.2 Network Function Virtualization

By harvesting the benefits of virtualization and cloud computing, NFV allows the deployment of originally hardware-based proprietary NFs on virtual environments in the form of VNFs [13]. The hosting environment of VNFs can be either Virtual Machines (VMs) or containers [14], depending on their specific resource demands. A set of VNFs can be chained together in a co-located or distributed cloud environment, offering network or value added services. As shown in [15], an NFV architecture defines in gereral:

- VNFs that are software implementations of NFs deployed in virtual environments.
- NFV Infrastructure (NFVI), which comprises the logical environment's building blocks, i.e., storage, compute, network and their respective assisting hardware components.
- Management & Orchestration (MANO) that is responsible for the network wide management and orchestration of VNFs and the NFVI.

Through running VNFs as softwarized instances on a common NFVI, the paradigm of NFV is becoming a key enabler to implement swift, flexible and scalable service deployment in 5G. This makes it possible to support a more fine-grained control and optimization towards the underlying networking services and resources.

In the context of network slicing, the NFV framework enables service chaining, capacity and latency-oriented VNF embedding and management [16]. In an NFV based network service, a dynamic Network Function Forwarding Graph (NFFG) [17] is established to chain the required VNFs in a flexible way. The use of a NFFG enables an on-the-fly deployment of network services with solely required network functions, depending on the service needs. With a given NFFG, the MANO will reserve the required amount of infrastructure resources to instantiate the service accordingly. Such process is non-trial since the underlying networking environment is becoming increasingly complicated and unpredictable in 5G. This requires more advanced optimization technologies to facilitate the operation and management towards the newly constructed 5G architecture. Next, we will introduce a learning augmented optimization technology to address this.

1.2 Definition of Learning Augmented Optimization

System design and operation often lead to diverse allocation problems, where limited resources must be assigned to competing objects so as to achieve the best overall system performance. Depending on the context, the allocation decisions may pertain to costs, tasks, goods, or other resources that can be assigned to one or several agents. Most of such problems can be interpreted as resource optimization problems. This thesis is targeted at the **nework optimization problems, which refer to the management of network resources and functions in a given environment, with the goal of improving the network performance.**

Traditionally, such problems are extensively addressed through classic optimization models and solvers, such as convex optimization, game theory, metaheuristics etc. However, with the rapidly improved model scales and compuation difficulties, traditional network optimization approaches are being challenged by diverse new emerging network applications. Motivated by the latest success of Artificial Intelligence (AI), there is increasing momentum recently in deploying learning based intelligent mechanisms for the forthcoming 5G to control the massive traffic volume in diverse networking landscapes. Different from the prosperity of AI techniques in their dominant applications e.g., image/video/speech recognitions, the application of many advanced AI techniques is still in its infancy in communication and networking systems. This motivates this thesis to start a comprehensive

research into **the concept of learning augmented optimization, which aims at promoting a greater integration of multiple AI tools (e.g., classic optimization, control theories and machine learning) into networking architecture**. This is bulit upon the theories of both the traditional optimization modelling and data-driven learning.

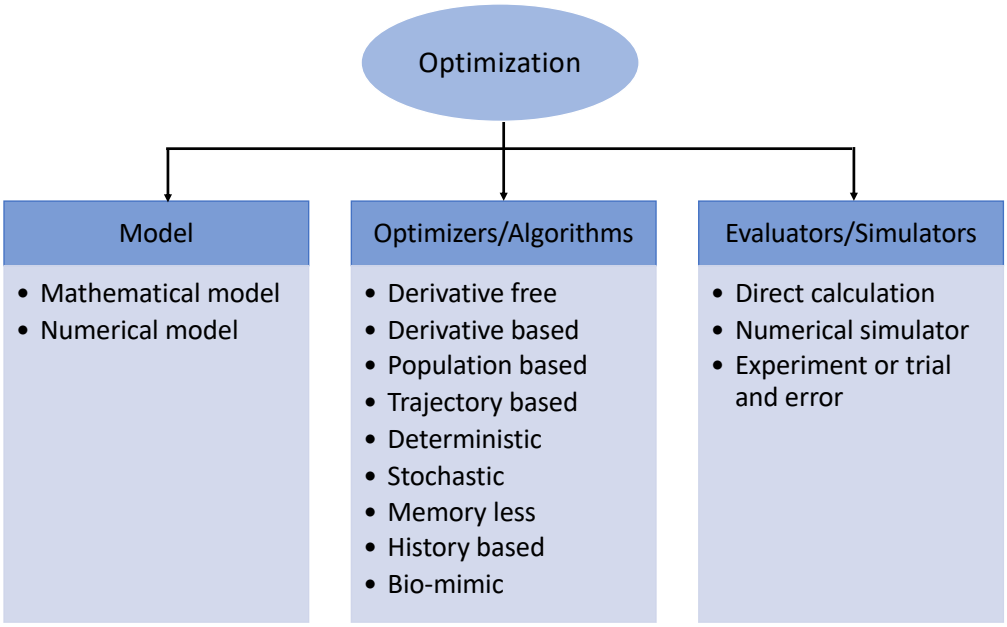


Fig. 1.1 A summary of optimization process.

1.2.1 Traditional Network Optimization

A typical optimization process is composed of three components [18], [19]: the model, the optimizer/algorithm and the evaluator/simulator, as shown in Fig. 1.1. Typically, a general network optimization problem can be formulated as:

$$\begin{cases} \min_x \ f(x) \\ s.t. \\ \quad h_i(x) \leq b_i, i = 1,2,\ldots,m \\ \quad g_j(x) = c_j, j = 1,2,\ldots,n \end{cases} \tag{1.1}$$

where x is the set of optimization variables. The function $f(x)$ is the objective function. The constraint conditions $h_i(x) \leq b_i$ and $g_j(x) = c_j$ are the inequality and equality constraints, respectively. The optimization problem formulated in (1.1) describes the problem of finding an optimal x^* that minimizes $f(x)$ among all x satisfying the constraints $h_i(x) \leq b_i$ and $g_j(x) = c_j$.

When all functions involved in (1.1) are convex, problem (1.1) is a convex optimization problem, which can be solved optimally by many efficient algorithms, such as interior-point methods [20]. However, many of network optimization problems are nonconvex due to the combinatorial property of allocating distributed network node & link resources. The complexity of global optimization methods for nonconvex problems may grow exponentially with the problem sizes, which are usually intractable. Alternatively, approximation and/or heuristic algorithms are often used to seek a suboptimal solution to overcome the difficulties of solving nonconvex problems.

In communication and networking areas, we can roughly list several optimization problems usually appeared in this field as follows.

- Integer Programming (IP) in which some or all optimization variables are constrained to be integer values. This kind of problems is usually raised in the designs of allocating multiple elementary network resources.
- Mixed Integer Programming (MIP) that concerns the problems having both discrete and continuous variables. Such model is often used to deal with the problems of jointly allocating discrete and continuous resources.
- Quadratic Programming (QP) where the objective function has quadratic terms.
- Semi-Definite Programming (SDP) which optimizes a linear function of the variables subject to linear equality constraints and a nonnegativity constraint on the variables. In many cases, nonconvex problems are usually transformed into SDP to get an efficient algorithm that is easy to implement.

To cope with the nonconvexity in these optimization problems, many optimization techniques have been proposed [21], such as dual decomposition, alternating search, penalty function method, sequential parametric convex approximation, semidefinite relaxation, and etc. These algorithms usually attempt to find an suboptimal but acceptable solution within a given computation budget. As the advent of many latest machine learning techniques, researchers are putting increasing attention on applying learning techniques to solve these optimization models.

1.2.2 Learning for Optimization

Most of the existing studies to solve optimization problems focused on sub-optimal or heuristic algorithms, whose optimality gaps are difficult to quantify and control. Instead of a hand-engineered optimization algorithm, the concept of “learn to optimize” aims to cast an

optimization model as a learning problem, allowing the algorithm to learn to exploit structure in the problems of interest in an automatic way [22], [23]. With the capabilities of inferring decisions from history experiences and observations, learning based solutions present a great complement to the existing optimization based solutions. This success can be attributed to the data-driven philosophy that underpins machine learning, which favours automatic discovery of patterns from data over manual design of systems using expert knowledge.

Machine learning-based methods have gained great momentum as a disruptive way to learn a near-optimal solution for difficult optimization problems. Recent attempts in [22]–[24] have demonstrated the promissing performance in a number of optimization problems. In spite of the preliminary success, applying fully learned optimization algorithms into real systems still faces many additional difficulties. As exposed in [25], a prominent shortcoming of these methods is that they require large amounts of training problem instances, e.g., millions of samples are needed for a small-size system. This incurs a significant cost for sample acquisition, and may not be feasible in particularly many NP-hard combinatorial optimization problems. Secondly, resource management problems are constrained by nature, but the ability of existing machine learning-based methods in dealing with constraints is limited. Finally, networks are inherently dynamic. Thus, a pre-trained machine learning model may be useless or suffer from severe performance deterioration as the network setting changes.

Consequently, instead of an exclusive choice, an alternative path foward is to commit to an integration of multiple AI tools to harvest their complementary strengths for existing problems. All of these motivate the research work on the learning augmented optimization technology in this thesis.

1.3 Research Problems, Challenges and Objectives

1.3.1 Problems and Challenges

Network softwarization, as a new conceptual technology, is still at its infancy. Currently, it remains a formidable challenge to implement, particularly in the dynamical 5G networking environment. In this respect, optimization and learning based approaches are two solid tools to move the way forward. Despite the great success of traditional optimization and intelligent learning technologies in their dominant applications, both of them struggle to confront these emerging challenges in current and near-future networking systems.

Concretely, the learning models (normally manifested with a neural network structure in deep learning) and classic optimization models (composed by a set of constraint and objective

functions) work in quite different ways. Optimization technology excells at generating stable and theorectically supported solutions and has successful applications in a wide range of domains. However, the drawbacks are that these solutions are usually dedicated only to the specific problems and require proficient domain knowledge to design and deploy them. In contrast, intelligent learning technologies, e.g., deep learning, are normally data-driven and exhibit great generality towards a set of problems that share similar structural patterns. Nevertheless, the black-box operations in learning technologies limit their application in many industrial fields, since controllable and interpretive solutions are more desired for many industrial applications.

In this setting, among the diverse optimization and learning methods, which of them are more preferable in the considered networking scenarios? How to intertwin the two distinct technnologies to reap their individual strengths but avoid their weaknesses? As a result, a holistic solution that integrates the merits of both traditional optimization and advanced learning technologies is timely needed. With this spirit in mind, we can summarize three critical problems along with their challenges to be targeted at in this thesis as follows:

- **Optimization with stochastic learning for netowork softwarization in dynamic networks**: The optimization of network softwarization involves solving probably very high-dimensional and complicated combinatorial optimization programs. This presents great computational difficulty since these programs are often NP-hard. Additionally, the dynamicity of the underlying networks are exacerbating the solving process, which normally requires solving a complicated stochastic optimization program in an iterative way. When some priori knowledge about the network changing patterns (e.g., historical statistics or stochastic probability models) are presented, how can we integrate these priori knowledge to augment the optimization decisions timely and robustly so that they can adapt the system to the fast changing networks? This is still a challenging problem for the implementation of network softwarization.

- **Optimization with online learning for network softwarization**: However, explicit knowledge about the environment is not always available, e.g., zero-day traffic. In this case, environmental knowledge and traffic statistics can only be learned progressively through run-time observations, which are then used to aid the system optimizaiton. Therein, among the diverse optimization techniques, many classic optimization technologies are applicable for our targeted problems, such as convex optimization, IP, online optimization etc. In the field of learning technologies, at hand are a wide range of machine/deep learning and reinforcement learning approaches. How to integrate these solid theories to compose a learning augmented online optimization for the

targeted problems? Particularly, when explicit statistics/knowledge are absent or only partially observed, how can we learn progressively and adapt the optimization online to safeguard the system performance in the long run? This is another challenging work in this thesis.

- **Deep learning to solving the network optimization problems**: Many latest deep learning technologies have demonstrated their promissing performance in areas e.g., speech recognition, image processing etc. However, the applications in networking domains present more constraints and requirements. The extension of these data-driven solutions to networking domains is non-trival. It requires to integrate a learning progress to reduce the optimality loss meanwhile respecting all the optimization constraints. However, deep learning techniques and optimization programs are attributed to completely different model structures. How to deploy a deep learning approach effiently within an optimization framework is still an open problem.

1.3.2 Objectives

This thesis aims at providing a systematic analysis to conceptualize the technology of learning aumented optimization. Accordingly, practical solutions will be developed to address these optimization problems for network softwarization. More specifically, methods will be introduced to 1) implement the efficient deployment solution of NFV in dynamic networks, 2) develop a learning augmented online optimization for network slicing in 5G, and 3) cope with the challenge of applying deep learning to solve combinatorial optimization problems. Further details will be outlined in Section 1.4.

1.4 Thesis Outline and Contribution

In this chapter, we have provided an introduction insight to the technologies of network softwarization and learning augmented optimization. In the remainder of this thesis, we will examine various state-of-the-art work and elaborate the designs of learning augmented optimization in the diverse problems of network softwarization. The key contents of this thesis are outlined in Fig. 1.2. The contributions of this thesis are summarized as follows.

- Chapter 2 presents a thorough review of the existing work on network softwarization and optimization techniques. In light of the problems in network softwarization, the overall 5G backgrounds and two critical enabling technologies, NFV and network slicing, are summarized. Then, we analyze three types of optimization techniques

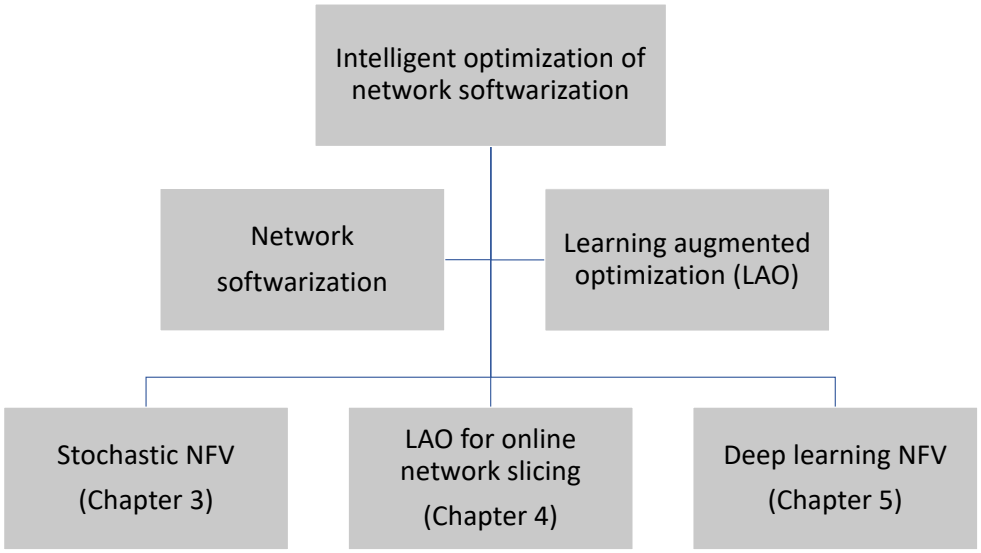


Fig. 1.2 Key contents of this thesis.

for the targeted problems, including deterministic, stochastic and online optimization technologies. This review builds a solid background for the methods proposed in this thesis.

- With the observations into the network utility degradation when implementing NFV in dynamic networks, Chapter 3 designs a robust NFV solution that exploits the prior stochastic knowledge to improve the system optimization. Unlike existing deterministic NFV solutions, which assume given network capacities and/or static service quality demands, this Chapter explicitly integrates the prior knowledge about the network dynamics into a two-stage stochastic resource utilization model. By exploiting the hierarchical decision structures in this problem, a distributed computing framework with two-level decomposition is designed to facilitate a distributed implementation of the proposed model in large-scale networks. The experimental results demonstrate a multi-fold performance improvement from the proposed solution.

- The robust NFV solution in Chapter 3 still requires the stochastic pattern information so that we can evaluate the candidate solutions by stochastic sampling technologies. In Chapter 4, we further release this requirement and consider the underlying system is fully unknown but can be observed through the real-time running. In this setting, Chapter 4 first classifies both the model-based and data-driven learning techniques that can be combined to imporve an optimization process. Subsequently, a learning augmented online optimization approach is developed to learn a safe network slicing solution from both prior knowledge and real-time observations. This is based on a

two-stage slicing optimization model with time-averaged constraints and objective. An offline solution to this problem is intractable since the future realizations of the system states are unknown before decisions. In contrast, an online solution can improve the optimization progressively by integrating knowledge learned from new observations. This eschews the biased decisions that only customize immediate performance over the instant system state in the existing network slicing solutions in the literature. We prove that the proposed solution is always feasible and nearly optimal, up to a constant additive factor. Finally, we demonstrate a huge improvement over the compared state-of-the-art algorithms.

- Distinct to the above optimization approaches for the targeted problems, Chapter 5 opens a new research direction, in which latest deep learning tools are introduced to these emerging problems. Specifically, we propose an encoder-decoder graph learning structure to represent and solve the traditional combinatorial optimization problems in a deep-learning manner. With the powerful data analysis of deep learning approaches, the proposed solution is able to solve the traditional optimization problem at tens of times faster speed with a less optimality loss. In addition, the optimality loss can be possibly further reduced by providing a stronger rollout policy as the baseline model. This presents a promissing research direction for the traditional optimization process to work in concert with advanced deep learning approches to build the future intelligent networks.

To summarize, this thesis researches the technology of learning augmented optimization and its application for the problems of network softwarization. The proposed methods are expected to contribute positively to both the optimization technologies and network softwarization. In the next chapter, a comprehensive review of existing solutions is provided. In addition to this, each primary chapter also makes some extra efforts to explain its contributions and most related works to make clear the contents in that chapter.

Chapter 2

Related Work

In Chapter 1, an overview of 5G network softwarization and network optimization techniques is presented coupled with the research problems, challenges and aims of this thesis. In this chapter, we provide a comprehensive review of existing research efforts around this new networking concept and optimization techniques. This chapter starts with the introduction of 5G network softwarization in Section 2.1, which elaborates both the achievements towards the overall 5G expectations and enabling techniques involved in network softwarization. Optimization techniques for network softwarization are then presented in Section 2.2, including both traditional and learning based network optimization techniques.

2.1 Network Softwarization in 5G

With the diverse new emerging networking services and more challenging operational requirements, the architectures of mobile networks (both core and radio access networks), fixed networks, and service delivery platforms are subject to an unprecedented technology-economic transformation. To improve the situation, many organizations are resorting to network softwarization for the next-wave network evolution through the technologies of NFV and SDN. In such a transition, more commodity servers and shared hardware devices will be introduced to replace these special-purpose devices in 4G. This will yield significant benefits in terms of reducing expenditure and operational costs of next generation (5G and beyond) networks. Also, it enables fully programmable, flexible, service/vertical-tailored, and automated end-to-end networks (i.e., network slices). The concept of network softwarization is expected to serve diverse services and verticals, including, but not limited to, Tactile Internet of Things, Pervasive Robotics, Self-driving, Immersive communications, Industry 4.0, and Augmented Reality.

Network softwarization works in concert with many other enabling techniques to achieve the overall expectations of 5G. In light of the overall challenges and efforts in 5G, the existing achievements in 5G are first introduced in this section.

2.1.1 Overall Challenges and Efforts in 5G

The combined effects of emerging millimeter wave (mm-wave) spectrum access, hyper-connected vision and new application-specific requirements are triggering the next major evolution in communications and networking– 5G [2], [26]. 5G envisions magnitudes of increase in data rates, bandwidth, coverage and connectivity, with a massive reduction in round trip latency and energy consumption. Blending the different research initiatives by industries and academia, eight major requirements [26] of next generation 5G systems are identified as:

1) 1 $\sim$ 10 Gbps data rates in real networks: This is almost 10 times increase from traditional Long Term Evoluation (LTE) network's theoretical peak data rate of 150 Mbps.

2) 1 ms round trip latency: Almost 10 times reduction from 4G's 10 ms round trip time.

3) High bandwidth in unit area: It is needed to enable large number of connected devices with higher bandwidths for longer durations in a specific area.

4) Enormous number of connected devices: In order to realize the vision of Internet of Things (IoT), emerging 5G networks need to provide connectivity to thousands of devices.

5) Perceived availability of 99.999%: 5G envisions that network should practically be always available.

6) Almost 100% coverage for 'anytime anywhere' connectivity: 5G wireless networks need to ensure complete coverage irrespective of users' locations.

7) Reduction in energy usage by almost 90%: Development of green technology is already being considered by standard bodies. This is going to be even more crucial with high data rates and massive connectivity of 5G wireless.

8) High battery life: Reduction in power consumption by devices is fundamentally important in emerging 5G networks.

With these eight above-mentioned requirements, industries, academia and research organizations have started collaborating in different aspects of 5G systems. Particularly, this section provides a thorough reviews on the overall 5G network architectures and standarization activities, which are closely connected to the solution designs of network softwarization.

First, many new network architectures are proposed in order to provide a holistic support towards the End-to-End (E2E) networking services in 5G. The promising new paradigms includes Cloud-Radio Access Network (C-RAN), Heterogeneous Network (HetNet), split control and user planes and etc. C-RAN resolves some of the major problems associated with increasing demands for high data rates [27]. C-RAN is based on fundamentals of centralization and virtualization. The baseband resources are pooled at BaseBand Unit (BBU), situated at remote central office (not at the cell sites). Remote Radio Heads (RRH), comprising of transreciever components, amplifiers and duplexers enable digital processing, analog-digital conversions, power amplification and filtering. RRHs are connected to BBU pool [28]. This simplified BS architecture offers to improve system architecture, mobility, coverage performance and energy efficiency while at the same time reducing the cost of network deployment and operation [27], [28]. Another way to handle the wireless traffic explosion, expected in 5G communication, is deployment of large number of small cells giving rise to HetNets [29]. By deploying low power small BSs, network capacity is improved and the coverage is extended to coverage holes. Moreover, the overlap of all small, pico, femto cells with the existing macro cells, leads to improved and efficient frequency reuse [29], [30]. The changes in architecture and air interface emphasizes on small cells and increased number of antennas. Configuration and maintenance of many servers and routers, in such a dense 5G deployment, is a complex challenge. SDN offers a simplified solution for this complex challenge. SDN considers a split between control and data planes, thereby introducing swiftness and flexibility in 5G networks [31].

The broad overview of 5G standardization activities is summarized in Fig. 2.1. It points out that the first standard is expected to mature by 2020. To deploy 5G in alignment with the market demands, 5GPPP is working for early agreements with major stakeholders for multitenancy and single digital market [33], [34]. METIS (Mobile and wireless communications Enablers for the Twenty-twenty (2020) Information Society) and HORIZON 2020 are the major 5G research project initiated and funded by the European Union (EU) [35], [36]. Moreover, different globally vendors and operators are also working towards their vision of 5G. Ericsson [37] expects 5G development should start in a backward compatible way with existing 4G LTE networks. This will help in continuing services using the same carrier frequency to traditional devices. Qualcomm [38] is developing and driving 4G and 5G in parallel to achieve the maximum potential. The unified platform should help in improving

cost and energy efficiency, while enabling a vast range of new services. Huawei is collaborating with international trade associations, universities, governments and ecosystem partners to establish crucial 5G innovations [39]. Docomo network has identified two important trends: (i) pervasive wireless connectivity (ii) extensive rich content delivery in real time [40]. It believes integration of both the higher and lower frequency bands holds the key to 5G deployment. The lower frequencies will be responsible for basic coverage and the higher frequencies will provide high data rates. Optimizing spectrum usage, revolutionary advances in 5G, dense small cells and improved performance are key concepts of Nokia's realization for 5G wireless [41].

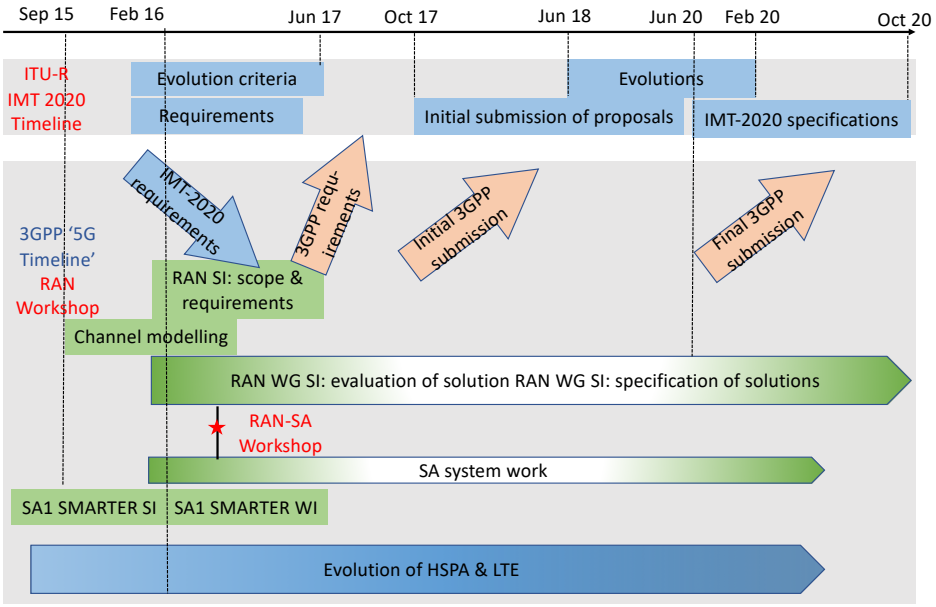


Fig. 2.1 Timeline of 5G standardization activities.

The above-mentioned challenges and visions of 5G networks lays the foundations of this thesis and motivate us to perform a comprehensive research into the technologies of network softwarization under the overall blueprint. Next, we provide a view on the existing achivements already done around the key enablers of network softwarization in 5G.

2.1.2 Network Slicing

Network slicing logically isolates NFs and resources that are specifically tailored to a vertical market's need on a single common network infrastructure. This enables a flexible stakeholder ecosystem that allows technical and business innovation integrating physical and/or logical network and cloud resources into a programmable, open software-oriented multi-tenant network environment.

Network slicing relies on the concept of virtualization. The idea of virtualization, i.e., creating a virtual form of a physical entity through software methods and processes, formed the vision of virtual systems spanning across computing platforms, network resources, and storage devices [65]. This requires that a network slicing solution must be able to coordinate efficiently diverse heterogenous resources.

In the context of 5G, many researchers have discussed the architecture of network slicing. A representative example of the network slice orchestration architecture given by 5GPPP [66] is illustrated in Fig. 2.2. Such an architecture considers a flexible separation of the control and data planes across a shared and dedicated network segments. A realization of network slice orchestration architecture, focusing on the RAN and on distributed core network, is considered in [69], [45]. The architecture relies on the principles of multi-service and multi-tenancy support. A network slicing architecture for integrated 5G communications is analyzed in [70], which demonstrates its realization for LTE considering different orchestration and control technologies.

A network slice is an E2E logical network that might span across multiple subnetworks, including RAN and core network. In order to meet the slicing demands of diverse network types, there are a broad researches working on the composition of the network infrastructure and its virtualization. For example, a mix of central and edge cloud computing infrastructures are proposed in [71], [72] , where resources can be allocated to either of them, depending on the slice requirements. This provides a flexible approach to meet the sub-millisecond latency requirement for some tactile services, such as remote surgery and autonomous driving.

Considering the virtualization of the core network infrastructure, researches done in the context of cloud computing can be leveraged. Specifically, technologies like kernel-based virtual machines and Linux containers can provide isolation guarantees in terms of processing, storage, and network resources at the operating system or process level [73]. On the other hand, virtualization approaches for the RAN are at an early stage. Applying VM and container-based solutions in this domain does not fully address the problem as they do not deal with the additional dimension of virtualizing and isolating radio resources (spectrum and radio hardware). Existing RAN virtualization approaches that account for this dimension fall into one of two categories [73]:

- Providing a dedicated chunk of spectrum for each virtual base station (slice) to deploy a full virtual network stack on top of it [72];
- Dynamically sharing the spectrum between different virtual base station instances (slices) by employing common underlying physical and lower medium access control (MAC) layers [74].

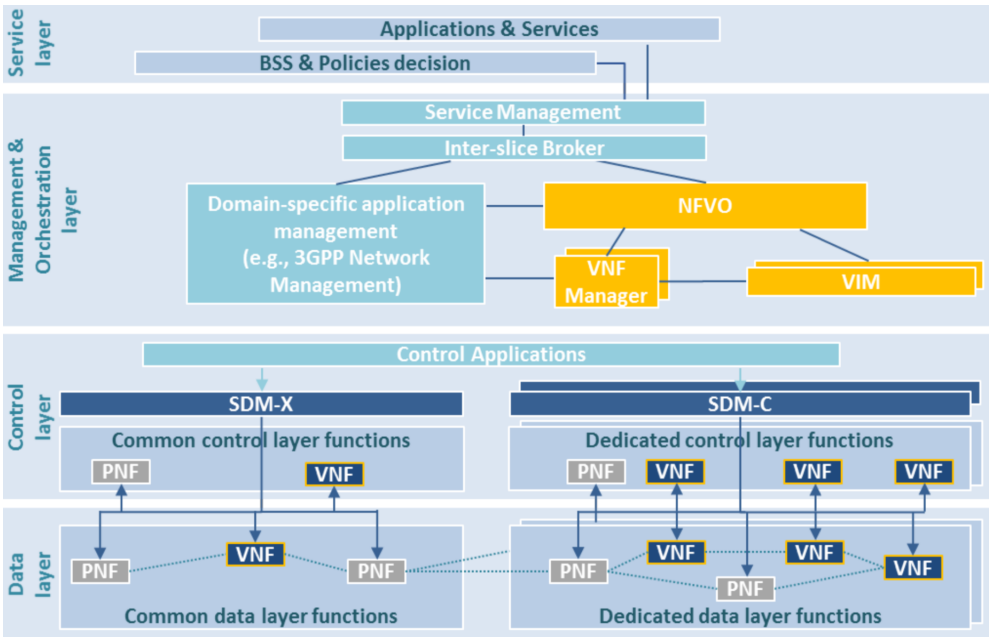


Fig. 2.2 A representative network slice orchestration architecture.

Common to all proposals, the deployment of network slices relies on NFV & SDN paradigms. As a result, 5G network control and data planes (C-/D-plane) will be organized by VNFs, instantiated in general purpose cloud infrastructures and communicating via programmable interfaces. Next, we provide a review on the existing work on NFV.

2.1.3 Network Function Virtualization

In spite of the performance superiority, 5G network functions face critical functional and architectural challenges, including the increased network complexity and equipment cost. A growing group of companies and standardization bodies are pushing researches of implementing the 5G network functions as software components by using the NFV paradigm to improve cost efficiency, flexibility, and performance guarantees. NFV is one of the key enabling technologies to implement the network softwarization. Generally, NFV can overcome some challenges of 5G by [15], [51]:

- Optimizing resource provisioning of the VNFs for cost and energy efficiency;
- Mobilizing and scaling VNFs from one hardware resource to the other;
- Ensuring performance guarantees of VNFs operations, including maximum failure rate, maximum latency, and tolerable unplanned packet loss;
- Ensuring coexistence of VNFs with non-virtualized network functions.

Fig. 2.3 depicts a referenced NFV architecture, which supports a wide range of services described as forwarding graphs by orchestrating the VNF deployment and operation across diverse computing, storage, and networking resources [15]. In Fig. 2.3, NFV management and orchestration comprises resource provisioning modules that achieve the promised benefits of NFV.

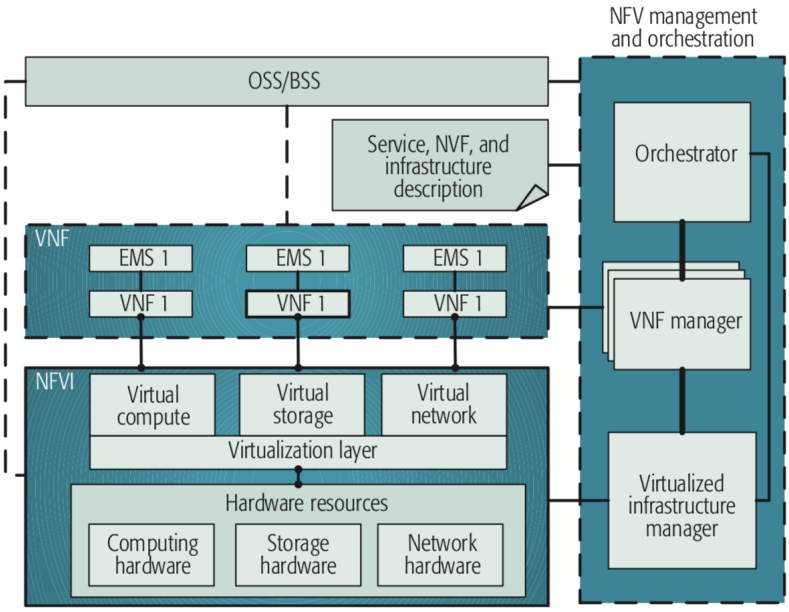


Fig. 2.3 A reference architecture of NFV.

Driven by the demand of industrial application and technical evolution, NFV based service orchestration is recently receiving increasing attentions by all participants. The service orchestration takes many similarities with the previous problem of virtual network embedding (VNE) [52] in placement aspect, but additional topological and service-specific requirements are posed:

- Different from the isolation requirement in VNE, virtual nodes (i.e., VNFs) in a service chain are allowed to share their service capacities to multiple service requests for same processing. This will associate the shared VNFs with multiple services, coupling the placement decisions with the admission results of multiple services when admission control is performed.

- Virtual nodes within a single service chain can be embedded into same substrate nodes. Whereas, different hosts are selected for different nodes in a virtual network so as to construct a forwarding network.

- The topology of a Service Function Chain (SFC) is flow-shaped, comprised by flow paths from pairs of source and destination. Nodes in each flow path usually constitute the dedicated processing for some specific flow(s). While VNE exhibits no such topological context information.

- The processing of a virtual node in a service chain would probably cause different rate variation to the served flow, resulting in heterogeneous resource requirements on nodes and links. In contrast, nodes in virtual network only perform data forwarding and resource requirements and node types are homogeneous within the topology.

To realize the envisions of NFV, Network Providers (NPs) need to orchestrate the network service at an abstract level by constructing a SFC, in which the involved VNFs and their logical connections that a service flow needs to traverse are defined (i.e., VNF chaining phase). The SFC is then optimally placed in instantiations bound to particular resources in the substrate network (i.e., SFC placement phase).

The SFC construction is similar to the problem of web service composition [53], which addresses the problem of how to compose a complex web service by chaining a set of elementary services. However, the chaining criteria for a service chain are quite different from those for web services due to the additional requirements on substrate resource allocation for placing VNFs. In the field of VNF chaining, intuitive semantic description is currently widely used [55], [56], [57]. This mechanism mainly relies on obvious mutual dependency relation between VNFs to determine their logical connectivity in a chain. [54] presents a flexible framework for the chain graph construction. Whereas, the proposed framework is designed from the perspective of high-level policy description, rather than from the perspective of topology and resource optimization.

The initial VNF/SFC placement researches mainly focus on optimizing the placement solutions with acceptable computing complexity under a given capacitated wired substrate network. The authors in [58] address the NFV node location problem, but leaving the link chaining problem into future research. Service decomposition is studied in [59] by supporting selecting a more refined topology. While the global topology optimization and rate variation problems are not addressed. [60] initiates the rate variation problem in NFV, in which only a simple rate variation model is presented based on a fixed node chaining order. However, when the global chaining order of NFs is subject to optimization, different ordering solutions would create different rate variation effect in a SFC graph.

Recently, there also exists a few work studing the elastic and online algorithms when the underlying system is dynamic. Among the very few of them, in [61], the authors model the elastic placement solution towards the dynamic workload, which is different

from the dynamicity problem of substrate network. The temporal scheduling problem is studied in [62], [63]. [62] addresses the VNF scheduling problem by jointly considering the scheduling and traffic steering. [63] models the temporal flexibilities for the VNE problem. Whereas, homogeneous wired network scenarios are targeted in all above work. As an extension, [64] studies the similar problem in distributed and wireless networks. Compared to the simple considerations of wireless network in [64], it is neccessary to conside the detailed network fluctuation and jointly combine the temporal and spatial features of substrate resources for the placement problem, which is more important from a long-term perspective.

2.2 Optimization Techniques for Network Softwarization

Following the general form of an optimization model in (1.1), we can, without loss of generality, classify the models to be used in the context of network optimization into three categories: *deterministic network optimization*, *dynamic/stochastic network optimization* and *online network optimization*.

2.2.1 Deterministic Network Optimization

When all system states (i.e., the coefficiencies of all objective and constraint functions) are known as a priori, we can formulate the involved problem as a *deterministic optimization model*, which can be an instance in any form of IP, MIP or QP etc. The deterministic optimization models capture the shared structures of network optimization problems, which are the basic foundations of the other optimization techniques. In the context of NFV, we can enumerate a list of Resource Allocation problems (i.e., NFV-RA problems) that are solved through some deterministic optimization approaches.

For example, Riera et al. [65] propose an analytic model for the VNF Forwarding Graph aiming to optimize the execution time of the network services deployed. This work presents a formulation for an optimal resource allocation for the NFV-RA problem. In addition, common economic metrics, performance metrics, etc. are introduced to evaluate and compare different approaches. Therefore, this analytic model can be used by different variants. Ghaznavi et al. [61] model the NFV-RA problem as MIP, different from the basic model, the authors consider distributed VNFs and workload balancing, so the formulation is much more complex than basic formulation. Therefore, a local search heuristic is proposed. Similarly, Luizelli et al. [66] decompose the problem into tree phase: (i) placement of VNFs, (ii) assigning VNFs to service requests, (iii) chaining the VNFs, and this insight is similar with the idea presented in [55]. The authors use Integer Linear Program (ILP) to model this problem

and present a heuristic. Cohen et al. [58] claim that the NFV-RA problem can be reduced to two NP-hard problems, the facility location problem and the Generalized Assignment Problem (GAP), which implies that the SFC placement problem is also NP-hard. Therefore, the authors propose an approximation algorithm based on solving GAP and then rounding the fractional solution computed into an integral solution.

In general, depending on the types of variables (continuous, discrete or mixed), linearity of all objective and constraint functions (linear, quadratic, convex or non-convex etc.), three general classic optimization solvers can be summarized: *optimal solver*, *approximate solver* and *heuristic solver*.

Optimal solvers find global optimal solutions. When a problem is convex or in small size, optimal solution can be solved by algorithms e.g., interior-point methods, branch and bound, dynamic programming, cutting plane methods, etc [20], [67]. A good example is presented in [68], which uses ILP to formulate VNF placement in mobile network, and use CPLEX, MATLAB, and CVX to solve the problem. Other similar studies can also be found in e.g., [69], [70]. However, the complexity of global optimization methods for nonconvex problems may grow exponentially with the problem sizes. Therefore, they are commonly used to solve small instances and present an optimal bound reference for heuristic solutions.

Approximate solvers give a trade-off between optimal solution and algorithm complexity. Therefore, a polynomial time algorithm can be found according to the compromising of optimality. A good application is the work in [71], which presents a polynomial time service chain approximation algorithms both for the case with admission and without admission control. This is achieved based on an extension of the classic Linear Programming (LP) and randomized rounding techniques. Moreover, the ILP formulation in [58] is also solved by reducing the problem to the GAP and then solving it with an approximation algorithm that shows proven performance bound.

Heuristic solvers try to exploit the problem-specific knowledge and for which we have no guarantee that they find the optimal solution [72]. As another related technique, a metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring the search space. Learing strategies are used to abstract structure information in order to find efficiently near-optimal solutions [73]. Service chain deployment is supposed to be low delay or real-time, therefore, fast heuristics are preferred in NFV-RA. Threre are a large quantity of heuristic-based solutions for NFV-RA problem. For example, Bari et al. [55] propose a dynamic programming based heuristic to solve large instances, which provides solutions within 1.3 times of the optimal solution obtained by solving ILP using CPLEX. The authors in [74] give a MILP formulation and a heuristic algorithm that solves the problem incrementally, which can solve

the problem for incoming flows without impacting existing flows. Metaheuristic algorithms are another way to overcome the hardness of NFV-RA problem. A lot of metaheuristics can be used to find better solutions, e.g., simulated annealing, tabu search, genetic algorithms, etc. For example, Bouet et al. [75] consider the problem of dynamically deploying Deep Packet Inspection (DPI) in NFV environment. The authors propose a method based on genetic algorithms, which provides a tradeoff between the number of engines and the network load to minimize the global cost of the deployment. Mijumbi et al. [76] consider the placement and scheduling of network functions in NFV. Then a greedy algorithm and a tabu search-based algorithm are proposed to solve the problem effectively.

2.2.2 Dynamic and Online Network Optimization

Normally, the above literatures assume that the network is static. In contrast, how to allocate resource at run time in a dynamic environment is a much more complex problem. Although this problem is similar to the basic NFV-RA problem, real-time NFV-RA has significant new challenges due to its dynamic features. First, resource that a VNF occupies may scale due to dynamic traffic. For example, a DPI needs less computing resource when the traffic decreases. Second, the QoS demands of VNFs may change due to changes of service requests. For example, when an established service request asks for low latency, reallocation of VNFs is required. Third, we should monitor the VNFs for reliability problem. For example, when VNF failures happen, we need reassign VNFs for corresponding service requests [77]. Therefore, in order to overcome those challenges, we should rethink the NFV-RA problem and propose new solutions.

Callegati et al. [78] use OpenFlow to properly steer traffic flows. According to the case study and proof of concept, the authors claim that both layer 2 and layer 3 approaches are functionally viable to implement the dynamic SFC. Shi et al. [77] have the insight that VNFs resources are not allocated simultaneously. Therefore, a preemptive resource allocation strategy is proposed. To realize the strategy, the authors model the SFC-RA problem as Markov Decision Process (MDP). In addition, Bayesian learning is used to predict future resource reliability. Leveraging the concept of asynchronous partition [79], the authors propose an algorithm based on MDP.

Another research direction to address the dynamic problem is to use online network optimization techniques. In this respect, we consider the service requests arrive one by one and are embedded when its arrival. Those solution typically belongs to Online Algorithms [80]. In such scenario, migration of VNFs may be necessary due to new requests arrival.

The authors in [74] give a MILP formulation to determine the placement of SFC while minimizing the resource utilization of nodes and links, in order to decrease delay. The

highlight of this paper is that the authors develop a heuristic to solve the problem incrementally, which support large size instances and can solve problem for incoming flows without impacting existing flows. Lukovszki and Schmid [81] propose a deterministic online algorithm which achieves a competitive ratio of $O(logl)$, where the node capacities are at least logarithmic. In addition, the authors prove that the proposed algorithm is asymptotically optimal in the class of both deterministic and randomized situations. At last, an ILP formulation is presented to show that the problem is NP-complete. Mijumbi et al. [76] consider the problem of online mapping and scheduling of VNFs. In this situation, each service is created and embedded as its need arises, and VMs can be shared by multiple VNFs. In addition, the authors propose three greedy algorithms and a tabu search-based heuristic.

2.2.3 Learning for Network Optimization Problems

5G systems are expected to support exploding mobile traffic volumes, real-time extraction of fine-grained analytics, and agile management of network resources, so as to maximize user experience. Fulfilling these tasks is challenging, as the networking environments are increasingly complex, heterogeneous, and dynamic. One potential solution is to resort to advanced machine learning (ML) techniques, in order to help manage the rise in data volumes and algorithm-driven applications. ML enables systematic mining of valuable information from traffic data and automatically uncover correlations that would otherwise have been too complex to extract by human experts [82]. This facilitates network analysis and management with high accuracy and in a timely manner, overcoming the run-time limitations of traditional mathematical techniques (e.g. convex optimization, game theory, meta heuristics).

The up-to-date studies that are at the intersection of deep learning and network optimization have underpined the new and powerful tools in this space. For example, Liu et al. exploit a Deep Belief Network (DBN) to discover the correlations between multi-commodity flow demand information and link usage in wireless networks [83]. Based on the predictions made, they remove the links that are unlikely to be scheduled, so as to reduce the size of data for the demand constrained energy minimization. Their method reduces runtime by up to 50%, without compromising optimality. Deep learning can also improve the efficiency of routing rules. Lee exploits a 3-layer deep neural network to classify node degree, given detailed information of the routing nodes [84]. The classification results along with temporary routes are exploited for subsequent virtual route generation using the Viterbi algorithm. Mao et al. employ a DBN to decide the next routing node and construct a software defined router [85]. By considering Open Shortest Path First as the optimal routing strategy, their method achieves up to 95% accuracy, while reducing significantly the overhead and delay, and achieving higher throughput with a signalling interval of 240 milliseconds. The application of deep

learning for scheduling problems can also be found in [86], [87]. In their studies, deep reforcement learning techniques are exploited to solve the corresponding network scheduling problems. More studies can be found in a number of survey papers e.g. [92]– [95].

With a similar resource allocation problem as this thesis, Ferreira et al. employ deep State-Action-Reward-State-Action to address resource allocation management in cognitive communications [88]. By forecasting the effects of radio parameters, this framework avoids wasted trials of poor parameters, which reduces the computational resources required. In [89], Mennes et al. employ multilayer perceptrons to precisely forecast free slots prediction in a Multiple Frequencies Time Division Multiple Access (MF-TDMA) network, thereby achieving efficient scheduling. Zhou et al. adopt LSTMs to predict traffic load at base stations in ultra dense networks [90]. Based on the predictions, their method changes the resource allocation policy to avoid congestion. The work in [91] sheds light on the radio control and signal detection problems. With accurate traffic forecasting, their proposal improves the performance of spectrum sharing in dynamic wireless environments, as it attains near-optimal spectrum assignments.

In summary, from these existing work in the literature, deep learning is more frequently used with its strong capability of prediction, which is different from our objective that aims to learn directly a solution for network optimization problems.

2.3 Conclusion

In this chapter, an organised overview of existing studies for both network softwarization and network optimization is present. Generally, it is very challenging to implement the visions of network softwarization in the context 5G since the basic problem is often NP-hard and the underlying environment is frequently changing. From the next chapter, three key research topics will be detailed concerning the involved resource allocation problems and optimization techniques: 1) stochastic NFV optimization solution, 2) learning augmented online optimization for network slicing and 3) a deep learning solution to solving the involved combinatorial optimization problems.

Chapter 3

Network Function Virtualization: A Stochastic Perspective

In this chapter, we present a stochastic NFV solution, in which we introduce prior stochastic knowledge about the environment to improve the NFV optimization in dynamic networks. Part of the contents in this chapter are summarized based on our prior publication in [99].

3.1 Introduction

The increasing mobility of humans and connected devices are actuating the explosive growth of mobile Internet traffic. According to [1], by 2021, global mobile data traffic will grow 7-fold, and the number of mobile users will be up to 5.5 Billion. To meet the extreme traffic demands, the next-generation networks (5G) are expected to be equipped with 5x as many as base stations and utilize 200x more spectrum than 4G [109]. This makes the orchestration of so many 5G elements to achieve the desired objectives get even more challenging than before.

To improve the situation, many organizations are resorting to the technologies of SDN & NFV [5], [45] for the next-wave network evolution. As illustrated in Fig. 3.1, in such a transition, more commodity servers and shared hardware devices will be introduced to replace these special-purpose devices in 4G. As a result, services will be constructed as individually optimized Service Function Chains (SFCs) [13]. These SFCs are then implemented with the isolated network resources sliced from the underlying network infrastructure. This enables prompt delivery of new services with better flexibility, agility and lower capital and operating expenditures [109].

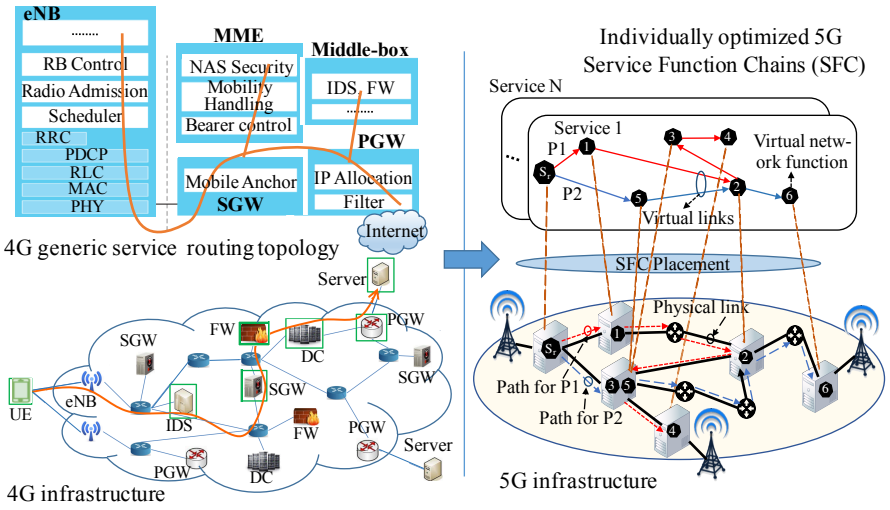


Fig. 3.1 NFV for next-generation network evolution.

As shown in Fig. 3.1, in an NFV based network service, VNFs will be dynamically chained as a specific SFC topology according to its service offerings. In order to implement such a service, one critical task is to perform the SFC placement in the underlying physical network bound to diverse resource and service requirements. This is achieved by placing VNFs in hosting servers and then connecting the placed VNFs with physical links through the proper allocation of infrastructure resources (*e.g.*, CPUs, cable bandwidth, radio spectrum).

The SFC placement problem (or service chain composition problem as termed in related work [67], [110]) is similar to the Virtual Network Embedding (VNE) problem [111] for network virtualization. Essentially, both problems aim to make efficient implementations of virtual requests in a physical network infrastructure. However, as discussed in [67], existing solutions to VNE problems are not sufficient for the SFC placement problems due to the specific features and applications of NFV.

Inheriting the methodologies built upon Integer Linear Programming (ILP) or Mixed Integer Linear Programming (MILP) for VNE, many preliminary studies have attempted to model and solve this new problem under a deterministic resource or traffic condition. However, with the penetration of NFV into more emerging networking applications [109], more network dynamics and uncertainties are expected than the current networks, which will make many existing deterministic NFV solutions not directly applicable. These network dynamics can be summarized as the following three classes:

a) ***Architecture level:*** In order to achieve an efficient utilization of higher and wider frequency spectrum beyond 6GHz, 5G networks will be heterogeneously constructed with more Radio Access Technologies (RAT), such as GSM, W-CDMA, LTE, W-LAN and new 5G RAT(s) [109]. The coexistence and cooperation of ever-increasing radio

systems will highly oscillate the network topology and bring more dynamics and uncertainties to the network operation and management.

b) ***Traffic level:*** In a virtualized operation environment in 5G, the traffic of each request may be collected from an individual mobile user, multi-tenant users, or a group of sensors. In many emerging 5G scenarios, the traffic profiles and demanded Service Level Agreements (SLAs) are to be highly varied and unpredictable. For example, in Internet-of-Things (IoT) applications, with the asynchronous activation and silence, node failures and mobility of large dynamic numbers of interconnected sensors and actuators, the collective traffic load injected into the traffic aggregation point (*e.g.*, gateway) of a network slice is always time-varying.

c) ***Resource level:*** With the increasing network cloudification in NFV, more globally controlled resources will be pooled together for more efficient utilization. The frequent resource scaling by the expected network self-management and -optimization procedures [112] for 5G will result in high instability and uncertainty on the resource distributions in each substrate node and link. With the expected coexistence of legacy and millimeter Wave (mmWave) spectrum bands [113], the 5G radio environment is also becoming increasingly unpredictable because of fast-fading, shadowing effects and interference.

When these dynamics are presented, the resource and/or traffic conditions are always time-varying. In this case, the deterministic SFC placement decisions can only customize the networking performance over the instantly observed information. However, this would leave the system vulnerable to potential network changes after decisions are made.

A straightforward solution to handle this situation is to migrate and re-route the VNF instances reactively by re-invoking a deterministic model (*e.g.*, MILP) to compute a new or recourse solution against each network change. However, no matter whether they are executed dynamically or online, such a solution can lead to frequent network reconfiguration and instability. In addition, it is also unaffordable in terms of the additional service latency incurred by the expensive re-computation of these usually NP-hard models. For example, in many delay-sensitive 5G applications [109], a millisecond-order system response is required, which is even beyond the time required for most existing model solvers to find a converged solution. To maintain a seamless service provisioning in dynamic networks, it is desirable to have a service deployment strategy that can handle the network changes proactively.

Therefore, in this chapter, we highlight the network utility degradation problem for the implementation of NFV in dynamic networks and aim to design a robust NFV deployment solution in both centralized and distributed fashions. Different from the posterior scaling

or migration based dynamic algorithms in the literature, this chapter alternatively resorts to reinforce the temporal robustness of the obtained SFC placement decisions from every expensive attempt of model solving by integrating future stochastic information at the initial placement decision phase. The contributions of this chapter can be summarized as follows:

- We implement the SFC placement in dynamic networks with a carefully designed Stochastic Resource Allocator (SRA) that: 1) jointly exploits the already-observed and future stochastic information to infer the placement decisions, and 2) balances of the immediate reward with the impact of each decision on future rewards.
- We provide a centralized optimal solution by solving the SRA model with a two-stage stochastic program and identify the hardness involved in solving SRA in a large instance, including the need of enumerating an exponentially expanding constraint set, and computing the expected random functions.
- A distributed computing framework with two-level decomposition is developed to facilitate a distributed implementation of the SRA in large-scale networks. Supported by the classic decomposition theory, the complicated combinatorial program only needs to be solved during service initialization, while the subsequent service running only involves the solving of a simple linear program. This significantly reduces the computation complexity involved in the whole duration of service running controls.
- Extensive simulation experiments are conducted with the settings in accordance with 5G expectations. Through the comparisons with the incumbent placement solutions, the results confirm that the proposed solution not only achieves significant performance improvement, but also effectively reduces risks of service quality violation in dynamic networks.

The rest of this chapter is organized as follows. Section 3.2 summarizes the related work. In Section 3.3, we formally derive the resource utility model for SFC placement in dynamic networks and analyze its optimal implementation. The distributed implementation based on two-level decomposition is presented in Section 3.4. Section 3.5 illustrates the performance evaluation results. Finally, in Section 3.6, conclusions are made.

3.2 Related Work

As a key enabling 5G technology, NFV has been gaining momentum among an ever-growing community of researchers from both academia and industry. It has been also the focus of

different standardization bodies (*e.g.*, 3GPP, ETSI and 5GMF) [5]. A global architecture can be found in [15], which defines the modules and interfaces that ensure the life-cycle management of NFV services. In the envisioned architecture, the technical implementation of NFV plays a critical role on the provisioning efficiency and service performance. Next, we provide a concrete review to highlight the differences between our work and the existing NFV solutions.

3.2.1 Deterministic NFV Modelling and Solutions

The preliminary efforts for the SFC placement problem mainly focus on optimizing the SFC placement with acceptable computing complexity in static settings. In this era, many centralized solutions based on deterministic optimization methods were proposed. For example, based on the multiple-objective Mixed Integer Quadratic Programming (MIQP), an initial study on placing VNFs was provided in [56]. However, its solution is developed through a Pareto set analysis while the solution scalability issue is left unattended. In [55], the authours formulated the VNF orchestration problem as an ILP, and a dynamic programming-based heuristic was provided to solve the problem in large instances. Based on the tools of MILP and game theory, extensive studies on deterministic VNF placement algorithms can also be found in [114]– [118]. Nevertheless, as aforementioned, these centralized solutions usually have scalability and/or accuracy issues and are often insufficient for large-scale dynamic networks.

In addition to these centralized solutions, the authors in [110] investigated the drawbacks of centralized placement solutions and proposed a distributed NFV solution by exploiting congestion games. A similar attempt is the work in [119] which provides a Markov approximated algorithm to solve the centralized placement model in a distributed way. In these existing studies, their solutions are solved only over the already observed network and service conditions. Consequently, these deterministic placement solutions, no matter in a centralized or distributed fashion, are not directly applicable to dynamic networks. In contrast, the SFC placement problem in dynamic networks is more complex. Beyond the considerations for a deterministic problem, more network dynamics are needed to handle in order to guarantee the effectiveness of the obtained solutions.

3.2.2 Dynamic Resource Utility for NFV Networks

There also exists a few studies in the literature striving to address similar resource utility problems for dynamic NFV networks.

Among the very few studies, Jia *et al.* in [120] proposed an online NFV scaling solution to handle the time-varying traffic volumes in geo-distributed Datacenters. However, the cost and drawbacks of dynamic scaling of VNF instances are not addressed in their solution. With a similar motivation, the resource provisioning solution proposed by Li *et al.* [121] is also proactive, although its objective is to assign requests with bounded response time. This is achieved by using SFC consolidation with timing abstraction, but the placement of SFCs is still based on deterministic models and VNF instance migration. Ghaznavi *et al.* in [61] optimized the VNF placement under changing workload. This is achieved by dynamically migrating the VNF instances on the basis of the migration costs in the current instant. This work was then extended in [122] by taking into account the benefits and penalties of these migrations in successive instants. However, different from the work in these existing studies, this chapter highlights the challenges of network dynamics that potentially limit the application of reactive scaling or migration strategies. Alternatively, we focus on generating SFC placement policies that can work robustly even when network state changes.

3.2.3 Network Applications of Decomposition Methods

As a complement, the applications of decomposition theories are also surveyed to show both the wide theoretical effectiveness and the differences between our application and other related networking problems. Decomposition methods are widely used in large-scale networks where a centralized solution is infeasible, non-scalable or too costly. Deniel *et al.* in [123] developed a generic application framework of decomposition methods for network utility maximization problem. A related survey of decomposition methods on many practical network applications can be found in [124]. Among these applications, the work in [111] comes closest to ours, in which they applied the Column Generation technique to decompose the deterministic centralized VNE model into two smaller subproblems. As a similar resource utility problem in supply chain network design, the authors in [125] proposed an accelerated benders' decomposition approach to expedite the solution time of the centralized MILP model.

Decomposition approaches require a good decomposable model structure. By exploiting the problem-specific structures in the proposed SRA model, we build a two-level decomposition framework to facilitate the distributed implementation of the proposed SRA solution.

3.3 A Robust Placement Model

As implied in Fig. 3.1, the SFC placement is essentially a graph-embedding problem. That is, mapping VNF nodes into substrate nodes and connecting VNFs with substrate links to implement the SFC graphs in the physical network topology. As discussed in Section 3.1, this is non-trivial as many problem-specific features are presented in the target problem. In what follows, we will design a stochastic resource utility model to implement the SFC placement with fully respect to the features of NFV paradigms and network dynamics. In this chapter, we treat the networking system as a discrete time stochastic system in which the network dynamics are assumed to follow stationary random processes. Moreover, resource scaling or migration of VNF instances are not considered due to the aforementioned challenges. The symbol notations used in this chapter are listed in Table 3.1.

3.3.1 Model Formulation

In this chapter, we consider a resource-limited network system, in which partial admission control is applied. That is, requests can be accepted by compromising service quality rather than be directly rejected when the available physical resources are not enough to fully meet the required demands. In addition, once accepted, each service will occupy isolated resources to instantiate its sliced network until new scheduling decisions are made or service is terminated. Such a pay-as-you-go admission policy is more practical than the existing all-or-nothing policy when the networks and/or resource demands of a service are time-variant.

To present the network dynamics, we consider a discrete time stochastic networking system, in which the service rate demands $\beta^s(t)$ and the available amounts of wireless resources at access nodes (*e.g.*, wireless transmission capacity), $c_v(t)$, are subject to random variations. The Probability Distribution Functions (PDF) of random variables are assumed known as a priori (via *e.g.*, estimations from historical statistics). At the beginning of every time slot $t \in \{0,1,2,\cdots\}$, the network controller observes a state update $\omega(t) = (\beta^s(t)_{s\in S}, c_v(t)_{v\in V})$, which specifies the current realizations of rate demands and resource state. Depending on $\omega(t_0)$ and the statistics about network dynamics, the controller, at the beginning of every scheduling interval T, decides a robust placement policy $\pi = (\pi_s, \pi_{f\to v}, \pi_{e\to l})^T_{s\in S}$ for the running duration $[t_0, t_0+T]$, and then adapts users' service rates to real-time observations at each time t.

From an algorithmic point of view, the design of placement policy in such a system requires to decide policies for admission control, VNF placement, and VNF chaining in a sequential order. This is fundamentally a combinatorial optimization process, which decides

Table 3.1 Notations for Chapter 3

System parameters			
$(V,\mathscr{L})$	Directed graph for physical network topology with node $v \in V$ and link $l_{uv} \in \mathscr{L}$ connecting u to v	c_{vr}	Residual resource capacity on physical node $v \in V$ for resource $r \in R$
c_l	Residual bandwidth capacity on physical link l	c_v	Residual wireless capacity on access node v
k_l, k_r	Usage price for per-unit link & node resources	T	Scheduling interval
Request parameters			
S	Received service requests	$f \in \mathscr{F}^s$	VNFs in service s, and let $\mathscr{F} = \cup_{s\in S}\mathscr{F}^s$
d_{fr}	Resource demand of $r \in R$ required to instantiate an instance of VNF f on a hosting node	$e_{ij} \in E^s$	Virtual link connecting VNF i to j for service s, and let $E = \cup_{s\in S}E^s$
b^s	Service price or benefit when unit rate of s is routed	$\beta^s \le \beta_0^s$	Rate demand requested by s, which is assumed to be up bounded by β_0^s
Decision variables			
π_s	Binary, 1 *iff* service request $s \in S$ is accepted	$\pi_{e\to l}$	Binary, 1 *iff* the routing of virtual link $e \in E^s$ uses physical link $l \in \mathscr{L}$
$\pi_{f\to v}$	Binary, 1 *iff* $f \in \mathscr{F}$ is placed on node $v \in V$	γ^s	Allocated rate for request s
Key auxiliary mathematical operators and symbols			
$(\cdot)^T$	Vector transpose	$\lvert\cdot\rvert$	Return the cardinality of a vector
$\mathbf{0}$,$\mathbf{1}$	All-one and all-zero vectors, respectively	$\mathbb{E}_{\boldsymbol{\gamma}}[\cdot]$	Take expectation over $\boldsymbol{\gamma}$
$u_s(\cdot)$	Revenue function for $s \in S$	$U_s(\cdot)$	Weighted revenue function for $s \in S$
$Q_s(\cdot)$	Utility function of s for policy approximation	w	Weight for future revenue
C^s_{nd}	Node resource cost for $s \in S$	$\Phi_{e\to l}$	Auxiliary variable for linearization
$\tilde{\pi}$	Fractional placement solution	$\bar{\pi}$	Approximate binary placement solution

a long-term optimal placement policy π under a stochastic environment. Following the pay-as-you-go billing model for network services [126], this chapter defines the following revenue oriented utility function for each $s \in S$:

$$u_s(\gamma^s, \pi^s) = \gamma^s(b^s - \sum_{\substack{e\in E^s \\ l\in\mathscr{L}}} k_l \pi_{e\to l}) - \sum_{\substack{f\in\mathscr{F}^s \\ v\in V, r\in R}} \pi_{f\to v} d_{fr} k_r \tag{3.1}$$

where $\pi^s = (\pi_s, \pi_{f\to v}, \pi_{e\to l})^T$ is the placement policy for s, $C^s_{lk} = \gamma^s \sum_{\substack{e\in E^s \\ l\in\mathscr{L}}} k_l \pi_{e\to l}$ and $C^s_{nd} = \sum_{\substack{f\in\mathscr{F}^s \\ v\in V, r\in R}} \pi_{f\to v} d_{fr} k_r$ are the cost for using the link and node resources, respectively.

Once a service is instantiated, a fixed node installation cost C^s_{nd} is charged, but the practical link cost $C^s_{lk}(t)$ and the benefit from this service are dynamically decided by the allocated service rate $\gamma^s(t)$ at runtime. Clearly, only services with benefit larger than all costs will be accepted by providers. Based on this insight, the concept of beneficial placement is defined

as follows:

Definition 1 ***(Beneficial placement)***. *Given a placement policy π^s, the placement of service request s is beneficial if the placement action for this request incurs a positive collective revenue (i.e., $\sum_{t\in[t_0,t_0+T]} u_s(\gamma^s(t)|\pi^s) > 0$).*

Based on the observed information $\omega(t_0)$, an *immediate revenue* can be counted as follows:

$$U(\gamma(t_0),\pi) = \sum\nolimits_{s\in S} u_s(\gamma^s(t_0),\pi^s) \tag{3.2}$$

For any future time $t_f > t_0$, the realizations of $\omega(t_f)$ are to be observed after the placement decisions. Consider the stochastic nature of the network, an *expected future revenue* under a given placement policy π made at t_0 can be calculated as follows:

$$\bar{U}(\gamma(t_f)|\pi) = \mathbb{E}_\gamma\big[\sum_{s\in S} \gamma^s(t_f)(b^s - \sum_{\substack{e\in E^s\\ l\in\mathscr{L}}} k_l\pi_{e\to l}) - C^s_{nd}\big] \tag{3.3}$$

where $\gamma = \gamma^s(t_f)_{s\in S}$ is a random vector dependent on the random outcome of $\omega(t_f)$.

The current placement decision has an impact not only on the immediate revenue, but also on the future revenues. From the network provider's point of view, the objective is always to maximize the long-term revenue under as minimum resource cost as possible. Consequently, efficient policies have to balance the benefits of an immediate reward with the expected impact of each decision on future rewards. This leads to the following global objective function designed to maximize the long-term revenue:

$$\begin{aligned} U(\gamma,\pi) &= \overbrace{U(\gamma(t_0),\pi)}^{\text{immediate exploitation}} + \mathrm{w}\ \overbrace{\bar{U}(\gamma(t_f)|\pi)}^{\text{future exploration}} \\ &= \sum_{s\in S} \underbrace{\big\{\big(b^s - \sum_{e\in E^s, l\in\mathscr{L}} k_l\pi_{e\to l}\big)\big(\gamma^s(t_0) + \mathrm{w}\mathbb{E}_\gamma\big[\gamma^s(t_f)\big]\big) - (1+\mathrm{w})C^s_{nd}\big\}}_{U_s(\gamma^s,\pi^s)} \end{aligned} \tag{3.4}$$

where $U_s(\cdot)$ is the weighted utility function for an individual service, and $\mathrm{w} \geq 0$ is a weighting factor to control the decisions' balance between exploiting immediate revenue and exploring the potentially better future revenue after network state changes.

Then, the intended SFC placement process can be readily formulated as the Stochastic Resource Allocation (SRA) program in Algorithm 1.

In Algorithm 1, (3.5b) and (3.5e) are the capacity upper bounds for link and node resources, respectively. (3.5c) guarantees that the total allocated rates for the set of services

Algorithm 1 The stochastic resource allocation for SFC placement in dynamic networks

Input: resource and traffic states at t_0, PDFs of $\omega(t_f)$, network and SFC topologies.
Output: placement policy π^* and running rate $\gamma^*(t_0)$.

$$U(\gamma^*,\pi^*) = \max_{\substack{\pi\in\{\mathbf{0},\mathbf{1}\}\\ \gamma\geq\mathbf{0}}} \sum_{s\in S} U_s(\gamma^s,\pi^s) \tag{3.5a}$$

$$s.t. \quad \sum_{e\in E^s, s\in S} \pi_{e\to l}\gamma^s(t) \leq c_l, \forall l\in\mathscr{L}, t\in\{t_0,t_f\} \tag{3.5b}$$

$$\sum_{s\in S_v} \gamma^s(t) \leq c_v(t), \forall v\in V, t\in\{t_0,t_f\} \tag{3.5c}$$

$$\gamma^s(t) \leq \pi_s\beta^s(t), \forall s\in S, t\in\{t_0,t_f\} \tag{3.5d}$$

$$\sum_{f\in\mathscr{F}} \pi_{f\to v} d_{fr} \leq c_{vr}, \forall v\in V, r\in R \tag{3.5e}$$

$$\sum_{v\in V} \pi_{f\to v} = \pi_s, \forall f\in\mathscr{F}^s, s\in S \tag{3.5f}$$

$$\sum_{l_{uv}\in O(u)} \pi_{e_{ij}\to l_{uv}} - \sum_{l_{vu}\in I(u)} \pi_{e_{ij}\in l_{vu}} = \pi_{i\to u} - \pi_{j\to u}, \forall e_{ij}\in E^s, s\in S, u\in V \tag{3.5g}$$

S_v attached to access node v will not overload its real-time wireless resource capacity. (3.5d) sets the rate upper bound that should be allocated for each service. (3.5f) imposes the variable dependencies and guarantees that each VNF will be placed at most once. In this chapter, unsplittable flow [127] is considered for constructing each virtual link. Let $O(u)$ and $I(u)$ denote the outgoing and incidental edges of node u, respectively. Then, the correlated connection between VNF placement decisions and VNF chaining decisions is finally expressed as (3.5g). Dependent on the practical applications, this model is versatile enough to integrate more problem-specific constraints. In production environment, service-related parameters are from the received request information. The network-related parameters can be collected from network configurations and real-time network telemetry [128].

For any deterministic realization (*i.e.*, a problem instance with all parameters determined), the model in Algorithm 1 corresponds to an MIQP. However, by exploiting the binary structure, this model can be readily linearized to a pure MILP. Let us define auxiliary variable $\Phi_{e\to l} = \pi_{e\to l}\gamma^s$ to substitute the quadratic expressions in (3.5a) and (3.5b) with the following two extra constraints:

$$\Phi_{e\to l} \leq \gamma^s, \forall e\in E^s, s\in S, l\in\mathscr{L} \tag{3.6}$$

$$\frac{\gamma^s}{\beta_0^s} - 1 + \pi_{e\to l} \leq \frac{\Phi_{e\to l}}{\beta_0^s} \leq \frac{\gamma^s}{\beta_0^s} + 1 - \pi_{e\to l} \tag{3.7}$$

Note that (3.6) is redundant when the link cost term is counted in the objective function.

In contrast to the dominant deterministic NFV resource utilization models in the literature, the proposed SRA jointly refers to both currently observed network information and future variation information at the placement decision phase. The added extra information can help exclude non-beneficial service placement more accurately, but also drive the model to the following more challenging dilemma when solving it.

Exploitation-exploration dilemma: *One needs to balance the exploitation of the placement action currently optimal with the exploration of other actions that currently appear suboptimal but may turn out to be superior in the long run.*

Algorithm 1 can be directly solved with all possible realizations of $\omega(t_f)$. However, this may require solving the resultant MILP model under an unmanageably large set of realizations of these random parameters, which is usually intractable. Next, we attempt to address this problem through a two-stage equivalent process.

3.3.2 The Global Optimality Solved through A Two-Stage Equivalence

Recall the structure of the model in Algorithm 1, the whole program can be re-arranged to a hierarchical two-stage process by separating the binary variables from continuous variables. Let us treat the case at t_0 as a special realization of future randomness. Then, the maximization problem in SRA can be reformulated as the following equivalent two-stage minimization problem (in linearized format):

$$U(\boldsymbol{\gamma}^*, \boldsymbol{\pi}^*) = \min_{\boldsymbol{\pi} \in \{\mathbf{0},\mathbf{1}\}} \Big\{ (1+\mathrm{w}) \sum_{s \in S} C^s_{nd} + \min_{\substack{\boldsymbol{\gamma} \geq \mathbf{0} \\ \boldsymbol{\Phi} \geq \mathbf{0}}} \mathbb{E}_{\boldsymbol{\gamma}} \Big[\sum_{\substack{s \in S \\ t \in \{0,1\}}} \mathrm{w}_t \Big(\sum_{\substack{e \in E^s \\ l \in \mathscr{L}}} k_l \Phi_{e \to l}(t) - b^s \boldsymbol{\gamma}^s(t) \Big) \Big] \Big\} \tag{3.8}$$

where w_t is the weighting factor for current and future time. Thus, we have $\mathrm{w}_0 = 1, \mathrm{w}_1 = \mathrm{w}$.

At the first stage, the program in (3.8) manages to decide a placement policy $\boldsymbol{\pi}$ with the constraints solely related to binary variables. Under the given policy $\boldsymbol{\pi}$, a policy evaluation program with only continuous variables (*i.e.,* $\boldsymbol{\gamma}$) is then applied at the second stage to evaluate the achievable average revenue.

For any determined placement policy $\bar{\pi}$ at each t, the inner minimization problem in (3.8) can be reduced to the following linear subproblem (policy evaluation program):

$$(\text{SP}) \quad \min_{\gamma \geq \mathbf{0}, \Phi \geq \mathbf{0}} \ \mathbb{E}_{\gamma}\Big[\sum_{s\in S} \mathrm{w}_t \Big(\sum_{e\in E^s, l\in \mathscr{L}} k_l \Phi_{e\to l}(t) - b^s \gamma^s(t) \Big)\Big] \tag{3.9a}$$

$$s.t. \qquad \sum_{e\in E^s, s\in S} \Phi_{e\to l}(t) \leq c_l, \forall l \in \mathscr{L} \tag{3.9b}$$

$$\sum_{s\in S_v} \gamma^s(t) \leq c_v(t), \forall v \in V \tag{3.9c}$$

$$\gamma^s(t) \leq \bar{\pi}_s \beta^s(t), \forall s \in S \tag{3.9d}$$

$$\Phi_{e\to l}(t) - \gamma^s(t) \leq \beta_0^s(1-\bar{\pi}_{e\to l}), \forall e \in E^s, l \in \mathscr{L}, s \in S \tag{3.9e}$$

$$\gamma^s(t) - \Phi_{e\to l}(t) \leq \beta_0^s(1-\bar{\pi}_{e\to l}), \forall e \in E^s, l \in \mathscr{L}, s \in S \tag{3.9f}$$

Define column vector $\mu := (\mu_l^0, \mu_v^1, \mu_s^2, \mu_{sel}^3, \mu_{sel}^4)^T$ as dual variables associated with each constraint in SP. Then, the dual of SP can be formulated as:

$$(\text{DSP}) \qquad \max_{\mu \leq \mathbf{0}} \ \mathbb{E}_{c_v, \beta^s}\big[-D(\mu, \bar{\pi}, t)\big] \tag{3.10a}$$

$$s.t. \quad \mu_{sel}^4 - \mu_{sel}^3 - \mu_l^0 \leq \mathrm{w}_t k_l, \forall e \in E^s, l \in \mathscr{L}, s \in S \tag{3.10b}$$

$$\sum_{e\in E^s, l\in\mathscr{L}} (\mu_{sel}^s - \mu_{sel}^4) - \mu_s^2 - \mu_{v_s}^1 \leq -\mathrm{w}_t b^s, \forall s \in S \tag{3.10c}$$

where v_s is the attached access node[1] for s, and $D(\mu, \bar{\pi}, t)$ is defined as follows:

$$D(\mu, \bar{\pi}, t) = \sum_{l\in\mathscr{L}} c_l \mu_l^0 + \sum_{v\in V} c_v(t)\mu_v^1 + \sum_{s\in S} \bar{\pi}_s \beta^s(t) \mu_s^2 + \sum_{\substack{l\in\mathscr{L} \\ s\in S, e\in E^s}} (\mu_{sel}^3 + \mu_{sel}^4)\beta_0^s(1-\bar{\pi}_{e\to l}) \tag{3.11}$$

The SP in (3.9) is a parametric linear program and always has a feasible solution under any given policy $\bar{\pi}$. In this case, according to duality theory [20], the DSP in (3.10) has always a bounded optimal solution corresponding to an extreme point of the polyhedron in dual space. After the transition from the primal SP to its dual, we can see that the uncertain parameters only exist in the objective function of dual problem, but the constraints of dual problem constitute a fixed polyhedron whose space is independent of the network variations and the chosen placement policy. Therefore, through the complete enumeration of extreme points, the original problem in (3.8) can be equivalently solved by the following master

[1] For simplifying exposition, single access node is considered for each s.

problem:

$$(\text{MP}) \qquad \min_{\substack{U,\pi\in P_\pi \\ \pi\in\{\mathbf{0},\mathbf{1}\}}} U \tag{3.12a}$$

$$s.t. \quad U \geq -\sum_{t\in\{0,1\}} \mathbb{E}_{c_v,\beta^s}[D(\bar{\mu}_i,\pi,t)] + (1+\mathrm{w})\sum_{s\in S} C^s_{nd}, \forall \bar{\mu}_i \in P_\Delta \tag{3.12b}$$

where P_π is the policy space defined by (3.5e)-(3.5g), and P_Δ is the set of extreme points in the DSP's polyhedron.

Recall the structure of the function $D(\cdot)$ in (3.11), c_v is independent of π when evaluating any given extreme point $\bar{\mu}_i$. Accordingly, we can reduce (3.12b) with

$$\mathbb{E}_{c_v,\beta^s}[D(\bar{\mu}_i,\pi,t)] = \mathbb{E}_{\beta^s}[D(\bar{\mu}_i,\pi,t)|c_v = \bar{c}_v] \tag{3.13}$$

where $\bar{c}_v$ is the mean value of c_v.

As a consequence, under the above separated two-stage structure, we only need to know the mean values of resource variations, although the detailed realization distributions of rate demands are still required. This property significantly reduces the number of random samples that are required to calculate the expectation of future average revenue.

When the problem is presented in a small instance, the MP can be solved efficiently with global optimality by enumerating all extreme points and possible realizations of β^s in (3.12b). Compared with the state-of-the-art methods, *e.g.,* directly adapting Column Generation [111] or Sample Average Approximation [125] to solve the SRA model, the above two-stage strategy requires less samples and thus results in a smaller problem to solve.

However, it would get very hard to do this for large-scale networks. Tri-fold challenges can be identified when solving the SRA model in a large instance with global optimality.

First, by removing the constraints (3.12b), the original problem is relaxed to the combination of classic facility location and multi-commodity flow problems [129], which are both NP-hard. Consequently, any single attempt of solving the problem with global optimality in a large instance is time consuming if not impossible. Second, to solve the problem in a large instance, there is typically an exponentially increased number of extreme points in the dual polyhedron. However, a more challenging problem is the computation of the expected value for the random function $D(\bar{\mu}_i,\pi,t)$. When the number of service requests gets large, this may involve an unmanageably large set of realization combinations for $(\beta^s)_{s\in S}$. All of these factors make it essentially impractical to completely enumerate all the constraints in (3.12b).

Consequently, in the following sections, we consider to design distributed models and approximate algorithms to alleviate the computational challenges of implementing this model in large-scale networks.

3.4 A Distributed Implementation Based on Two-Level Decomposition

It is well known that the computation load and the required memory for solving an optimization program increase exponentially with the number of variables and constraints. Therefore, by harvesting the above separated two-stage structure of the SRA model, we first design the following higher-level decomposition to reduce the large scale of the SRA model brought by the enumeration of extreme points and possible realizations of $(\beta^s)_{s\in S}$ in (3.12b). This is achieved based on the theory of stochastic decomposition [130].

3.4.1 Higher-Level Decomposition

By exploiting the method of stochastic decomposition, it is possible to decompose the complicated monolithic model in a large instance into a series of solvable submodules in a distributed way. The solution of original model can then be reached by solving these submodules in an iterative manner. This is implemented through the concepts of variable partition and constraint delay.

Following the two-stage structure of the SRA model in Section 3.3.2, we first construct the Higher-level Sub-Problem (HSP) exactly same as the SP model in (3.9). The HSPs are a series of linear programs with only continuous variables. Then, a dimension-reduced Higher-level Master Problem (HMP) can be initially constructed as a relaxed version of the MP model in (3.12) without the constraint (3.12b).

Instead of a complete enumeration of constraints in (3.12b), the method of stochastic decomposition alternatively solves the HMP model to generate a trial decision for the placement policy. The trial placement policy is then fed into the HSPs with randomly sampled parameters. Accordingly, the associated DSPs are solved to obtain the resultant extreme point and an approximation towards the original objective function under current samples. Next, a constraint of (3.12b) related to this extreme point will be inserted into the HMP, which is then solved again until a predefined termination criterion achieved. The overall progress is outlined in Algorithm 2.

Note that the extra terms in (3.14) are introduced to exclude unnecessary policy trials. Normally, we have $\sum_{s\in S}\{\pi_s - \sum_{e\in E^s, l\in\mathscr{L}} \frac{k_l \pi_{e\to l}}{\beta^s}\} \ll U$, thus the impact of introduced terms on the optimality of U is negligible.

Algorithm 2 Approximate stochastic decomposition algorithm for SRA

Input: resource and traffic states at t_0, PDFs of $(c_v)_{v\in V}$ and $(\beta^s)_{s\in S}$, network and SFC topologies.

Output: placement policy $\bar{\pi}$ and running rate $\gamma^*(t_0)$.

1: *Initialization:* set $m = 0$, collect current observation $\omega(t_0)$.

2: do $m = m+1$ and solve the following HMP to obtain trail policy $\bar{\pi}_m$:

$$U_m^l(\bar{\pi}_m) = \min_{\substack{U,\pi\in P_\pi \\ \pi\in\{\mathbf{0},\mathbf{1}\}}} U - \sum_{s\in S}\{\pi_s - \sum_{e\in E^s, l\in\mathscr{L}} \frac{k_l \pi_{e\to l}}{b^s}\} \tag{3.14}$$

3: draw random samples for the future realizations of $(c_v)_{v\in V}$ and $(\beta^s)_{s\in S}$ according to their PDFs.

4: solve the DSPs in (3.10) for each t with the generated samples and policy $\bar{\pi}_m$ to obtain the extreme point $\bar{\mu}_m$ and an empirical estimation to the original expected objective value:

$$U_m^u = -\sum_{t\in\{0,1\}} D(\bar{\mu}_m, \bar{\pi}_m, t) + (1+\mathrm{w})\sum_{s\in S} C_{nd}^s \tag{3.15}$$

5: **if** ***termination criteria*** meet **then**

6: solve HSPs with $\bar{\pi}_m$ for t_0 to get the allocated rate γ^* for current time.

7: **else**

8: add the following optimality constraint to the program in (3.14):

$$U \geq -\frac{1}{m}\sum_{t\in\{0,1\}} D(\bar{\mu}_m, \pi, t) + (1+\mathrm{w})\sum_{s\in S} C_{nd}^s \tag{3.16}$$

9: update the coefficients in previous optimality cuts as follows and go to step 2:

$$D(\bar{\mu}_i, \pi, t) = \frac{m-1}{m} D(\bar{\mu}_i, \pi, t), \forall t \in \{0,1\}, i = 1, \cdots, m-1 \tag{3.17}$$

10: **end if**

Termination criteria: For discrete distributions of random parameters, the sample space is deterministic. Therefore, by using the whole sample space at each iteration, the lower and upper bounds of original objective function can be derived precisely from the results of U_m^l and U_m^u, respectively. In this case, a deterministic criterion can be used by monitoring the optimality gap between upper and lower bounds. However, for continuous distributions, the bound gap is subject to statistical variation due to the random sampling outcomes. Then,

an alternative criterion is to monitor the progress of incumbent solutions. For example, the iteration stops when the incumbent solution has remained unchanged for a certain number of iterations within a tolerant variation on the expected objective value. Such criteria can provide better safeguards to prevent the sensitivity of the selected solution to additional sampling.

The proof of the asymptotic optimality of Algorithm 2 is similar to many existing stochastic approximation applications [132], thus is not explicitly presented here.

For each iteration in Algorithm 2, all sampled subproblems are linear programs, which can be solved easily via many standard LP solvers [20]. However, the HMP corresponds to an MILP problem, which is still NP-hard, although the dimension has already been reduced compared with the original problem. Therefore, the handicap regarding solving the NP-hard HMP still persists for large-scale networks. In the following, we will build a lower-level decomposition to turn the HMP into a decoupled resource utility problem for each service request, which can then be solved with better scalability.

3.4.2 Lower-Level Decomposition

Recall the structure of HMP model, we can see that the constraints in (3.12b) and (3.5e) are coupled among all service requests. This results in an exponentially increased computation complexity as more service requests are required to schedule. However, if these coupling constraints are relaxed, the original HMP model naturally turns into an individual resource utility problem for each service request. Each of these problems can then be, independently and parallelly, solved. This is implemented as follows through linear relaxation and dual decomposition.

Since the HMP is only introduced to generate the bound and trial placement policy for HSPs, the accurate but expensive solving for the optimal solution at every iteration is not essential. Alternatively, simple and faster approximation algorithms can be adopted to generate a new trial policy. The trial policy can then be gradually improved through fast iterations. Therefore, instead of directly getting an optimal solution for the HMP in (3.14) at each iteration, a linear relaxed version of HMP can be first solved as follows:

$$\text{(RHMP)} \quad \tilde{\pi} = \underset{\substack{U,\pi\in P_\pi \\ \pi\in[\mathbf{0},\mathbf{1}]}}{\operatorname{argmin}} U - \sum_{s\in S}\Big\{\pi_s - \sum_{e\in E^s, l\in\mathscr{L}} \frac{k_l \pi_{e\to l}}{b^s}\Big\} \tag{3.18}$$

The RHMP relaxes the binary constraint in HMP to the continuous value ranging from $[0,1]$. The fractional placement solution $\tilde{\pi}$ obtained from the RHMP conveys the globally coordinated resource allocation information when all requests compete for the shared re-

sources. We can anticipate that larger fractional values of decision variables would suggest a better revenue if the corresponding service is accepted and placed accordingly. Therefore, by proportionally weighting the placement selections with the corresponding fractional solutions, an approximate solution to the HMP can be solved from a weighted HMP program as follows:

$$\text{(WHMP)} \qquad \bar{\pi} = \underset{\pi \in P_\pi, \pi \in \{\mathbf{0},\mathbf{1}\}}{\operatorname{argmax}} \sum_{s \in S} Q_s(\pi^s) \tag{3.19}$$

where $Q_s(\pi^s)$ is the utility function for individual service, which is defined as

$$Q_s(\pi^s) = \alpha_1 \tilde{\pi}_s \pi_s - \sum_{\substack{f \in \mathscr{F}^s \\ v \in V}} \frac{\alpha_2 \pi_{f \to v}}{\tilde{\pi}_{f \to v} + 1} - \sum_{\substack{e \in E^s \\ l \in \mathscr{L}}} \frac{\alpha_3 \pi_{e \to l}}{\tilde{\pi}_{e \to l} + 1} \tag{3.20}$$

where $\alpha_1 \gg \alpha_2 \gg \alpha_3$ are weights to preserve the hierarchical decision order along $\pi_s \to \pi_{f \to v} \to \pi_{e \to l}$.

In WHMP, the objective function has a separable structure, but the resource constraint (3.5e) in P_π is coupled across all $s \in S$. Since strong duality holds when solving the WHMP through its Lagrange dual problem, this constraint can be decoupled for each $s \in S$ by means of dual decomposition [123].

We define the Lagrangian of the WHMP by relaxing the coupling constraint (3.5e) in P_π as

$$\begin{aligned} L(\pi, \lambda) &= \sum_{s \in S} Q_s(\pi^s) + \sum_{v \in V, r \in R} \lambda_{vr} \big(c_{vr} - \sum_{s \in S} \sum_{f \in \mathscr{F}^s} \pi_{f \to v} d_{fr} \big) \\ &= \sum_{s \in S} \big\{ Q_s(\pi^s) - \sum_{\substack{f \in \mathscr{F}^s \\ v \in V, r \in R}} \lambda_{vr} \pi_{f \to v} d_{fr} \big\} + \sum_{\substack{v \in V \\ r \in R}} \lambda_{vr} c_{vr} \end{aligned} \tag{3.21}$$

where λ is the non-negative Lagrange multiplier associated with the constraint (3.5e).

Clearly, for a given λ, the resulted Lagrangian dual problem can be decomposed into solving, independently for each $s \in S$, the following Lower-level Sub-Problem (LSP):

$$\text{(LSP)} \quad L_s(\bar{\pi}^s, \lambda) = \max_{\pi^s \in \{\mathbf{0},\mathbf{1}\}} Q_s(\pi^s) - \sum_{\substack{f \in \mathscr{F}^s \\ v \in V, r \in R}} \lambda_{vr} \pi_{f \to v} d_{fr} \tag{3.22a}$$

$$s.t. \qquad \sum_{v \in V} \pi_{f \to v} = \pi_s, \forall f \in \mathscr{F}^s \tag{3.22b}$$

$$\sum_{l_{uv} \in O(u)} \pi_{e_{ij} \to l_{uv}} - \sum_{l_{vu} \in I(u)} \pi_{e_{ij} \to l_{vu}} = \pi_{i \to u} - \pi_{j \to u}, \forall e_{ij} \in E^s, u \in V \tag{3.22c}$$

LSPs can be directly solved through existing integer program solvers since only several VNFs and virtual links are involved in the placement process. Additionally, linear relaxation and rounding based approximate solvers (*e.g.,* [129]) are also applicable in order to further reduce the computation complexity.

Physically, the Lagrange multiplier λ corresponds to the resource congestion price, on which each service has to depend to decide the amount of resources to be used at their own benefits. Then, in order to achieve the original global optimality, the minimum congestion price can be solved by using the following Lower-level Master Problem (LMP) to coordinate all LSPs:

$$\text{(LMP)} \quad \hat{\lambda} = \underset{\lambda \geq \mathbf{0}}{\operatorname{argmin}} \sum_{s \in S} L_s(\bar{\pi}^s, \lambda) + \sum_{v \in V, r \in R} \lambda_{vr} c_{vr} \tag{3.23}$$

When the analytical expression of L_s is absent, the (LMP) can be recursively solved through the following gradient method:

$$\lambda_{vr}(n+1) = \left[\lambda_{vr}(n) - \delta(c_{vr} - \sum_{f \in \mathscr{F}} \bar{\pi}_{f \to v} d_{fr})\right]^+, \forall v \in V, r \in R \tag{3.24}$$

where n is the iteration index, $\delta > 0$ is a positive step size, and $\lceil \cdot \rceil^+$ denotes the projection onto the non-negative orthant.

In Algorithm 3, we provide the detailed implementation of the approximate dual decomposition for solving the HMP.

Algorithm 3 Approximate dual decomposition algorithm for solving the HMP

Input: coefficient matrix for HMP, mean values of network states $(\mathbb{E}_{\beta^s}[\beta^s]_{s \in S}, (c_v)_{v \in V})$.
Output: trial placement policy $\bar{\pi}$.

1: set $n = 0$ and $\lambda_{vr}(0)$ equal to some non-negative value for all (v, r).
2: solve RHMP to obtain the fractional solution $\tilde{\pi}$.

▷ *Solving WHMP under* $\tilde{\pi}$

3: do $n = n + 1$, and solve all LSPs to obtain a local placement policy $\bar{\pi} = \{\bar{\pi}^s\}_{s \in S}$.
4: update congestion prices according to (3.24) and broadcast the new prices to all LSPs.
5: go to step 3 until maximum iterates or tolerant variation on the collective utility $L(\pi, \lambda)$ reached.

▷ *Revenue evaluation*

6: solve HSP with $\bar{\pi}$ and statistical mean values, *i.e.*, $\omega(t_f) = (\mathbb{E}_{\beta^s}[\beta^s]_{s \in S}, (c_v)_{v \in V})$.
7: calculate the individual revenue as follows based on the solution obtained in step 6:

$$U_s = u_s(\gamma^s(t_0), \bar{\pi}^s) + T u_s(\gamma^s(t_f), \bar{\pi}^s) \tag{3.25}$$

8: reject requests with non-beneficial placement (*i.e.*, $U_s \leq 0$) and return all accepted $\bar{\pi}^s$.

Note that in Algorithm 3, the WHMP will only generate a placement solution that fully respects the fractional placement results obtained in RHMP. But this does not guarantee that every admitted request has a beneficial placement for the duration $[t_0, t_0 + T]$ since the optimality constraints are not evaluated in WHMP. Therefore, under the placement solution from the WHMP, a final revenue evaluation process is invoked to exclude the requests with non-beneficial placement when the network is in the mean state.

Finally, by putting all together, the overall distributed computing framework for the SRA model is illustrated in Fig. 3.2 and summarized as follows:

Step 0: Preprocessing the request information and collect the coefficient matrix of SRA program.

Step 1: Solve HMP with lower-level decomposition algorithm:

- ***1.1:*** Solve RHMP to obtain fractional solution;
- ***1.2:*** For each $s \in S$, solve LSPs with given congestion price λ;
- ***1.3:*** Update congestion price and return to **1.2** until termination;
- ***1.4:*** Individual revenue evaluation and return beneficial placement policy.

Step 2: Solve all DSPs to evaluate the optimality gap of current trial policy.

Step 3: Add new optimality constraint to HMP and return to **Step 1** until termination.

Step 4: Network slice running management until next-round scheduling.

Considering the random iteration progress involved in the computing framework, it is quite difficult to provide analytical results to the overall optimality and time complexity of the proposed algorithm. In light of these, a set of numerical simulation analysis are provided in next section.

3.5 Simulation Results

In this section, we conduct extensive simulation experiments with the settings in accordance with 5G expectations to evaluate the proposed solution.

3.5.1 Simulation Setup

Following the similar setups used in the existing NFV/VNE experimental studies, *e.g.* [129], [133], we generate synthetic network slices and random resource demands to support the following simulation experiments. Current BT's IP network topology within Europe[2] is considered as the physical network, which includes 21 nodes and 34 bidirectional links. 5

[2]http://www.topology-zoo.org/maps/BtEurope.jpg

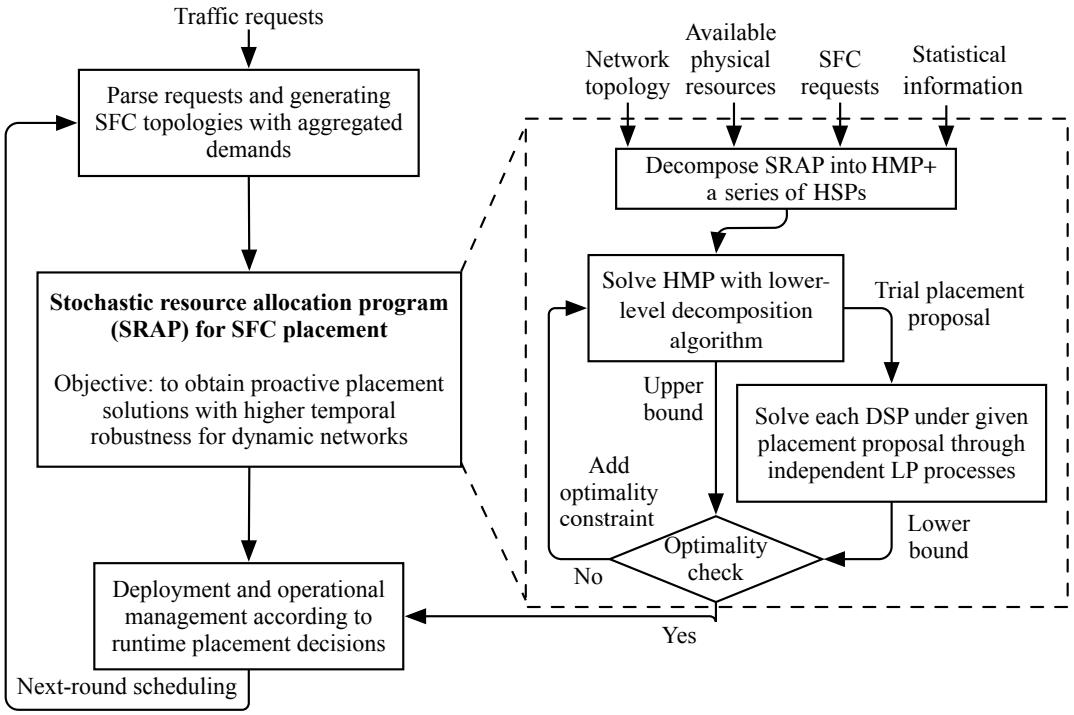


Fig. 3.2 Distributed SFC placement diagram based on two-level decomposition.

nodes from these 21 nodes are randomly selected to act as the wireless access nodes. For each node in the network, a fixed amount of computing resources is configured. For each fibre link, the transmission capacity is set proportionally scaled from the practical BT core network bandwidth[3].

In the following simulations, we emulate the envisioned 5G small cells with mmWave spectrum to model the capacities of wireless access links connected to each access node. Considering the fast transitions among Line-of-Sight (LOS), non-LOS (NLOS) and outage network stages in mmWave channels [113], we use the Rician fading for LOS stage and Rayleigh fading [134] for NLOS and outage stages. The transition probabilities between any two channel states are set as equal. The channel parameters are configured so that the resulting wireless capacity of each access node is on average within the envisioned capacity range for a 5G cell [109]. Table 3.2 lists the main configurations for the simulation experiments below.

[3]https://www.globalservices.bt.com/static/assets/pdf/products/optical_connect/BT_Optical_Connect_datasheet.pdf

Table 3.2 Simulation Setup for Chapter 3

Parameters	Value
Node resource capacity c_{vr}	5–10, uniformly distributed
Fixed link capacity c_l	$10Gbps$
Resource prices $[k_l, k_r]$	[$20/Gbps$, 20]
# node resource type $\|R\|$	1
# VNFs $\|\mathscr{F}^s\|$	2
Node resource demand d_{fr}	1–3, uniformly distributed
Aggregated rate demand β^s	1–$3Gbps$, uniformly distributed
Service price b^s	100–300, uniformly distributed
Rician factors K	$1dB$
Radio bandwidth B	$1GHz$
Normalized power allocation ρ	LOS: $31.3dB$; NLOS: $9.3dB$; Outage: -$4.3dB$

3.5.2 The Compared Algorithms and Performance Metrics

Two reference algorithms, CG_SP [108] and CMG_SP are compared. In CG_SP, the placement decisions are made only to optimize the immediate revenue at t_0 based on already observed network information. In CMG_SP, mean state information, $\mathbb{E}_{\beta^s}[\beta^s]_{s\in S}$ and $(\bar{c}_v)_{v\in V}$ are used to represent their future states. The decisions of CMG_SP are then made to optimize the same objective as the proposed SRA under the current observations and the mean states of futures. CMG_SP is a widely used policy to handle with system dynamics [135]. This comparison can provide an insight to the difference between the exploration of complete PDF knowledge and simple statistic knowledge.

All compared algorithms require run-time collection of the instant system states. Since the mean state information and PDF information are derived from the already collected statistics, no extra overhead is required to run the CMG_SP and our proposal in practice. For both reference algorithms, the corresponding placement models are solved through the greedy node mapping with shortest path based link mapping [108].

In SRA, the iterative progress is set to stop when the incumbent solution has remained unchanged for 5 iterates. In the lower-level decomposition, the tolerant variation on the monitored utility is 10%. For both levels, the maximum number of iterations is limited to 50.

The following four metrics are used to evaluate the performance of our algorithms against the compared ones.

1) ***Average revenue gain:*** This is defined as the ratio of average revenue achieved by SRA (or CMG_SP) and that by CG_SP within the running period $[0, T]$.

2) ***Provisioning cost gain:*** Provisioning cost defines the average cost for occupying physical node and link resources under each placement policy. Accordingly, the

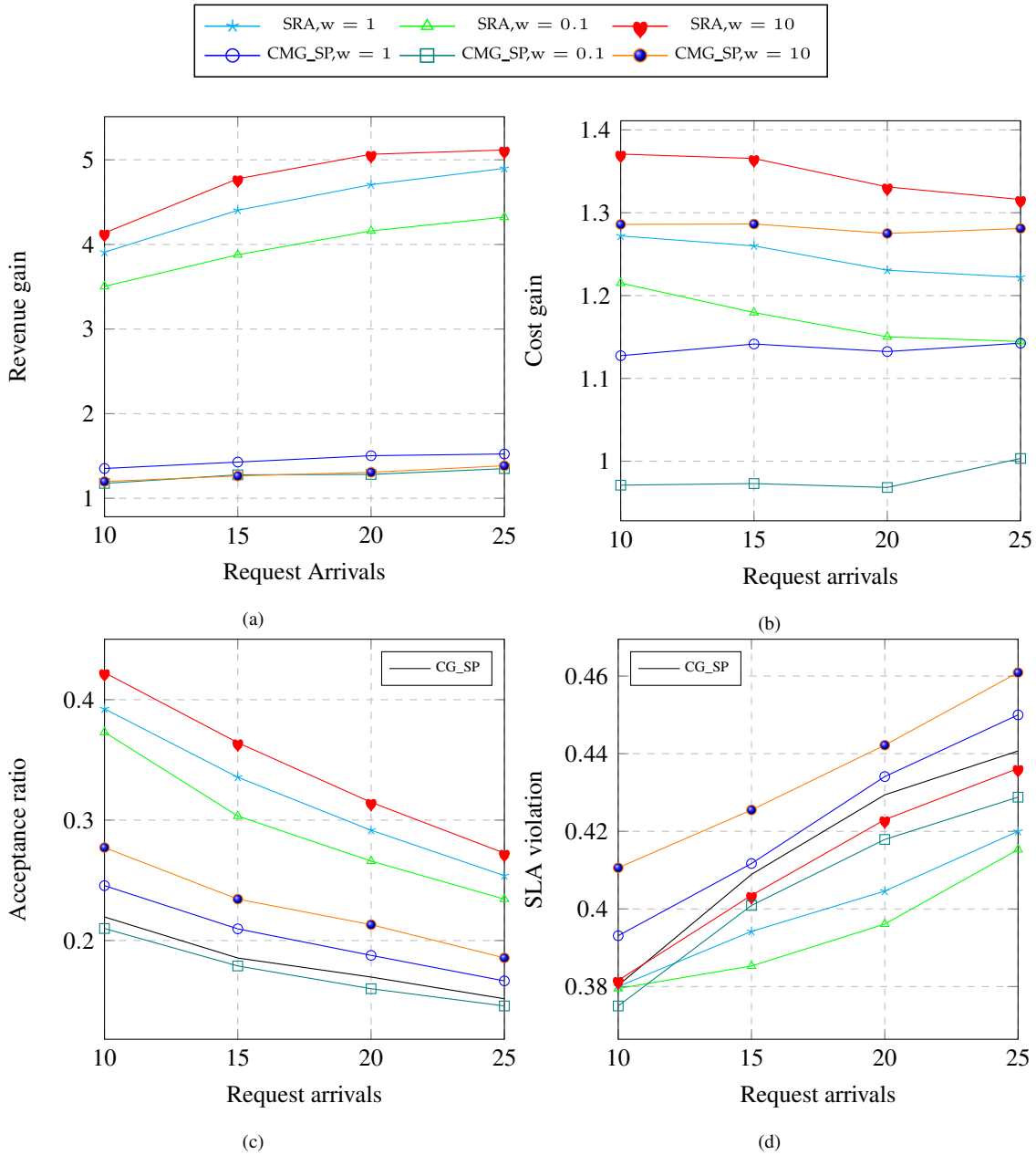


Fig. 3.3 Performance comparisons with $T = 10$: a) Average revenue gain, b) Provisioning cost gain, c) Acceptance ratio, and d) SLA violation.

provisioning cost gain is the ratio between the provisioning cost in SRA (or CMG_SP) and that in CG_SP.

3) ***Acceptance ratio:*** The acceptance ratio of an algorithm measures the percentage of total requests accepted by different algorithms. Combined with the revenue metric, the acceptance ratio gives a sense of how well an algorithm performs on excluding non-beneficial placements.

4) ***SLA violation:*** This is calculated as $\sum_{s\in S}\sum_{t\in[0,T]}(\pi_s - \gamma^s(t)/\beta^s(t))/\sum_{s\in S}(T+1)\pi_s$. SLA violation measures the average offset degree of the allocated running service rates within $[0,T]$ from the requested rate demands over all **accepted** requests, which is an important metric reflecting users' quality of experience towards the provisioned services.

3.5.3 Performance Analysis

Fig. 3.3 depicts the compared performance under different settings. All performance metrics are calculated by generating 1000 random samples to evaluate each placement policy. Based on the simulation results, our key observations are summarized in the following.

1) From long-term consideration, extra future statistic information can enhance the placement policy with $3 \sim 5$x *better revenue.* Fig. 3.3a shows the average revenues collected from different algorithms. Under the given settings, the simulation results confirm that the significant revenue improvement of the proposed SRA approach over the referenced two algorithms. Compared with the only 1.5x revenue gain made by the deterministic algorithm in [129] over the same CG_SP benchmark, the proposed SRA presents a more positive results with up to $3 \sim 5$x revenue gain when network dynamics are considered.

Specifically, when the available physical resources are abundant, the possible network variations have little impact on the placement decisions. In this case, the performance gain in SRA are mainly contributed by the more efficient policy computing than the greedy policies. With the increased requests, however, the resource competition among requests gets intensified. As a result, any over-optimistic or -pessimistic placement decisions in CG_SP would be detrimental to the long-term revenue performance. This is avoided in SRA with the joint reference of future statistic information, thus creating a higher gain as resources become scarce.

Benefiting from the calibration to the decisions by the statistical mean values, CMG_SP, on the other hand, also achieves around 1.5x revenue gain over CG_SP. However, due to the non-convexity of the achievable revenue under each combinatorial policy option, the

accuracy from statistical mean values is highly compressed than that when complete PDFs are used to capture the future network variations.

2) The nearly 5x *better revenue of SRA only raises about 30% more resource cost.* Fig. 3.3b shows the compared results on the provisioning cost gain over CG_SP. Combined with the revenue results in Fig. 3.3a, we can see that SRA achieves up to 5x better revenue but using only 30% more resources than CG_SP. This indicates that the available physical resources are coordinated more efficiently to serve more requests when physical resources become more scarce. However, with the greedy placement policies, about 20% more resource investment only contributes 1.5x revenue gain for CMG_SP.

3) SRA makes more good-quality acceptances. As depicted in Fig. 3.3c, SRA accepts 2x more services than CM_SP, and the ratio for CMG_SP is 1.5x. Then, we can get that the revenue gains contributed by unit acceptance are $5/2$ and $1.5/1.5$ for SRA and CMG_SP, respectively. This shows that the acceptances made by SRA are more beneficial, which collectively contribute the higher revenue gain. The degradation of the compared algorithms results from both the acceptance to the requests that are currently beneficial but long-term non-beneficial and the exclusion of requests that are temporarily non-beneficial but long-term beneficial.

4) Services deployed according to SRA policy present lower SLA violation risk. Fig. 3.3d shows that benefiting from the accurate capture of future network variations, the SLA violation is significantly lower in SRA than the compared ones. This reflects a better long-term robustness and users' quality of experience towards the provisioned services in SRA when network dynamics are presented. In contrast, the statistical mean values only decrease a little the SLA violation risk in CMG_SP.

3.5.4 Effect of Different Weighting Balance

In SRA, the weight settings of parameter w show different emphasis on the future expected revenue, which results in a performance tradeoff. The optimal value of w is subject to parameter tuning, depending on the runtime estimation towards the quality of current and future network states. For example, when the observed current network state is believed overwhelmingly better than the average cases, setting a small value of w is more reasonable so that the current good state can be fully exploited. Conversely, if the network state is currently observed to be very bad, a large value of w should be set to leave more spaces to explore potentially better performance in the future. However, estimating the quality of an observed network state is non-trivial when the explicit expression of the system performance over observations is absent.

Table 3.3 Performance of SRA under Different w, $T = 10$.

Weight w	Average revenue gain	Provisioning cost gain	Acceptance ratio	SLA violation
0.1	4.32	1.14	23.4%	41.5%
1.0	4.90	1.22	25.4%	42.0%
10	5.12	1.32	27.3%	43.6%
15	4.23	1.10	23.7%	38.3%
20	3.90	1.08	23.3%	38.1%

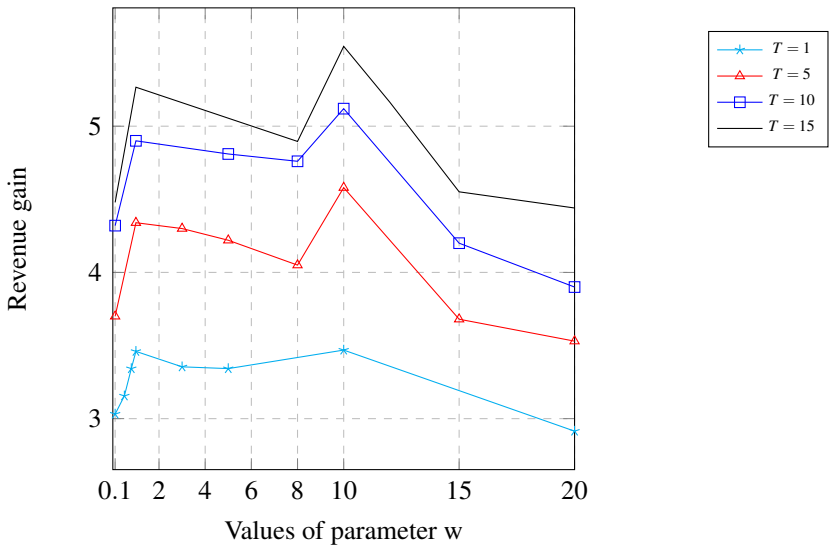


Fig. 3.4 Weighting effects under different scheduling intervals.

In this section, we evaluate the effect of different weight settings on the performance of SRA. Table 3.3 and Fig. 3.4 enumerate the compared performance when the system is loaded with request arrivals = 25. Based on the simulation results, the following two behaviors of the proposed SRA solution are observed.

1) Selecting an appropriate weighting balance for each decision is a tradeoff. We observe from Table 3.3 that the considered performance metrics exhibit different changes over the setting of weight w. Under the given settings, the difference gap among these average revenue gains is up to 1. Moreover, the average revenue gain and provisioning cost gain show more sensitivity towards the setting of w. In contrast, the variation of w only makes little changes on the SLA violation. This stands to the reason that the SLA violation is averaged over all the accepted requests, thus is normally less sensible to the changes of w than the other metrics.

2) When the network variations follow stationary processes, best weight is not around T, but a value approximately ranging between $[1, 10]$. Stationarity is the property of a stochastic process whose probability distribution is the same at all times [136]. In this case, the averaged long-term observations will finally converge to the mean values of network variations. As a consequence, if weighting according to the average revenue contribution of the immediate and future ones in the objective function (5a), $\mathrm{w} = T$ should be a reasonable weight option to balance the immediate and future revenues in the objective function (5a). However, according to the simulation results in Fig. 3.4, $\mathrm{w} = T$ is not always better when T takes different values. This shows the non-convexity property for the average revenue of SRA in terms of w. The calculation of the optimal weight requires the accurate modelling of the system performance over observations, which is complicated in the considered combinatorial optimization scenario. Another alternative option is to set a dynamic weight through some heuristic rules according to every observation. For the practical industrial application of the proposed SRA solution, taking a value between $[1, 10]$ could be a mild weighting option since this setting retains nearly 90% revenue gain in Fig. 3.4.

3.5.5 Effect of Statistical Error for Future

Stationary random processes are assumed for the network variations in the SRA model. Therefore, a proactively improved decision can be made to create a better long-term revenue based on the given PDFs of network variations. In this section, we release the stationarity assumption and evaluate the performance of SRA when the estimated PDFs are subject to statistical errors or temporal evolution. The error is presented by setting an offset between the mean values of the practical and estimated PDFs. We summarize the observed behaviors of the proposed SRA as follows.

1) The superiority of the proposed SRA is preserved even when the estimated network variations are subject to 50% statistical errors. The results for the considered performance metrics with $w = 1, T = 10$ are depicted in Fig. 3.5. We use a positive ε to denote the optimistic case when the statistical mean of future network variation is estimated 20% more than its practical value. Likewise, pessimistic estimations are evaluated with a negative ε to show the effect when the statistical mean of future network variation is estimated 20% less than its practical value. In Fig. 3.5, multi-fold performance gains are still presented for the SRA under statistical errors. However, it also causes a degradation up to 1 in terms of the revenue gain for the case of $\varepsilon = -50\%$ when compared with the results under accurate PDF information (*i.e.*,$\varepsilon = 0$).

2) Pessimistic estimation decreases the overall service capacity, while optimistic estimation leads to more bad-quality acceptances. Also depicted in Fig. 3.5 are the performances of SRA when ε takes different offset values. In the pessimistic estimation case, we can observe that with the increased estimation errors, the accepted maximum loads are gradually plummeted from the amount that the system can really serve. The decreased acceptance directly results in the under utilization of network resources and considerable revenue loss. On the other hand, SRA can make more acceptances in the optimistic case than the amount that the system can really serve. However, the increased acceptances only take negative effects, resulting in more revenue loss, provisioning cost and also SLA violation. This turns out that the extra acceptances are actually non-beneficial that should not have been accepted.

3) The revenue loss due to the sub-optimality of SRA can be compensated by pre-setting $\varepsilon = +10\%$ statistical error. In the case of $\varepsilon = +10\%$, we can observe from Fig. 3.6 that the SRA solver can load more requests than the case with error-free resource estimation. The slightly increased acceptances turn out to be beneficial and finally contribute nearly 10% revenue improvement. This confirms that due to the sub-optimality of the solution in SRA, there are nearly 10% potential revenue loss. However, such loss can be compensated by pre-setting the resource estimation with $\varepsilon = +10\%$ statistical error.

3.5.6 Convergence Analysis

The main computation cost of the proposed SRA solution comes from recursively calculating the approximate HMP and sample averaged approximation in SPs. In the designed termination criteria, the maximum number of iterations is limited to 50 so that the solution can be generated with a controlled time budget. Fig. 3.7 shows the average number of iterations under different SRA weighting balance. The following two behaviors can be observed.

1) More iterations are required to make the solution converge when the future revenue is considered with a higher weight than the immediate revenue. The future revenue in the

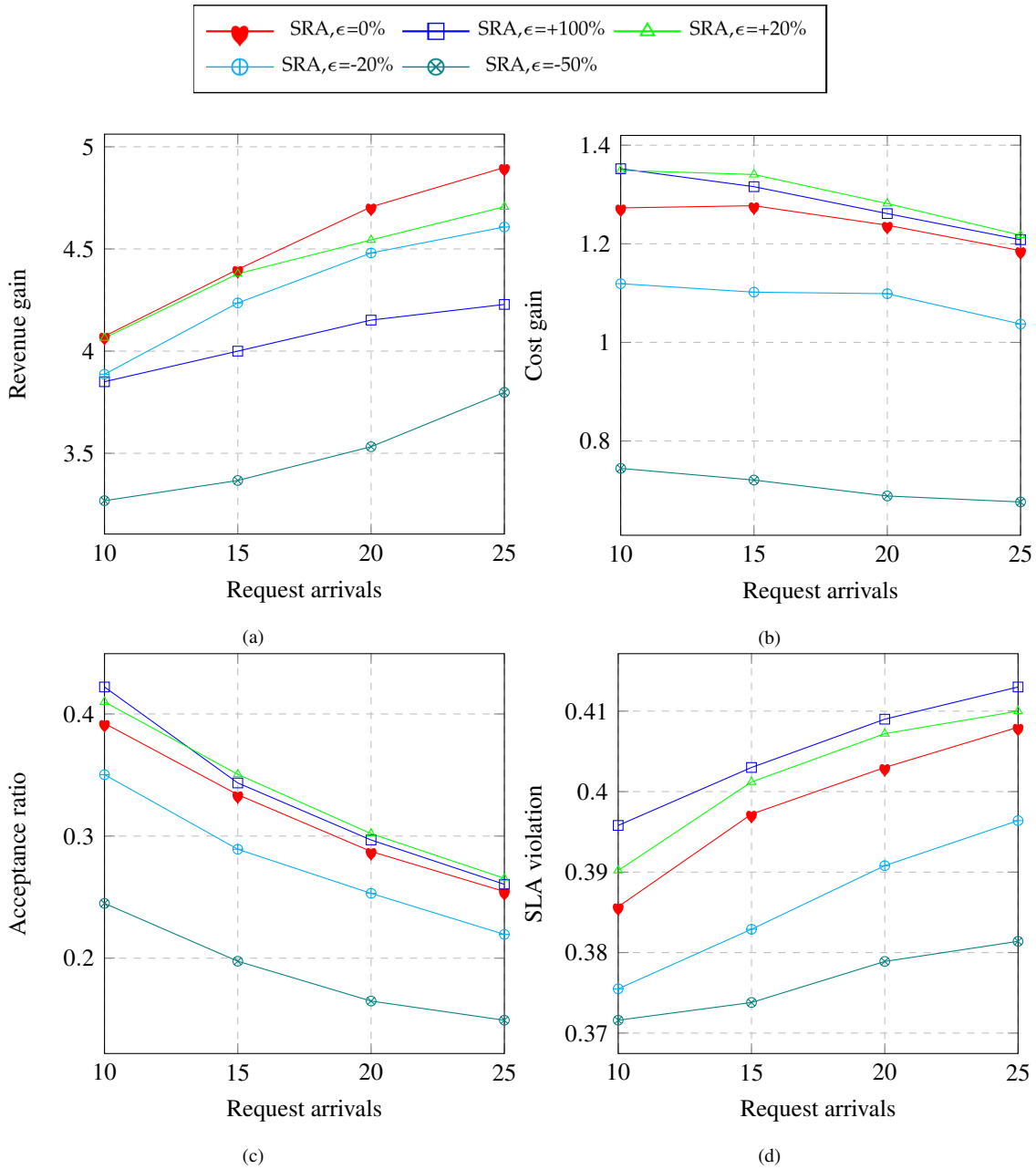


Fig. 3.5 Performance comparisons under statistical error with w $= 1, T = 10$: a) Average revenue gain. b) Provisioning cost gain. c) Acceptance ratio. d) SLA violation.

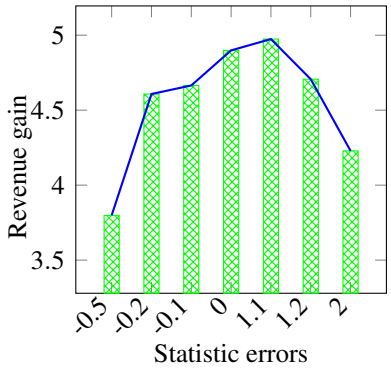


Fig. 3.6 Revenue performance under different statistic errors with w $= 1, T = 10$, request arrivals = 25.

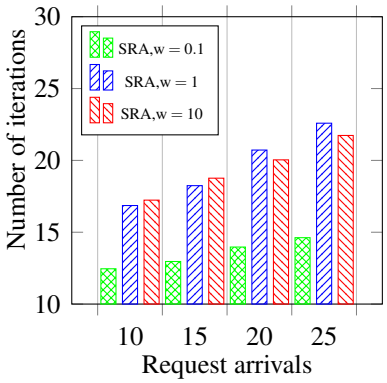


Fig. 3.7 Number of iterations under different weights.

proposed solution is evaluated through random samples at each iteration. When the future revenue takes a higher weight than the immediate revenue, the achieved objective value will get more sensitive to the outcomes of random sampling at each iteration. Consequently, more samplings are required to converge the objective of the model to the tolerant value variation.

2) The required iterations to converge increase when more requests are presented. As shown in Fig. 3.7, with the increase of request arrivals, more iterations are required to find the accepted placement policies. This is reasonable, since more arrivals lead to more similar policy options to compare with. However, with the fractional placement information used in solving HMP, many unnecessary placement policy trials can be avoided. We can see from Fig. 3.7 that all experiments finish the computation within an average number of 20 iterations. Combined with the results provided in Fig. 3.5, the performance achieved under such iteration criteria retains nearly 90% optimality.

Although multiple iterations are required, the complicated combinatorial program in the proposed solution only needs to be solved during service initialization. The robust deployment decisions, once solved, can be used with robustness across the whole scheduling interval. However, the subsequent service running controls only need to solve a simple linear program. This significantly reduces the computation complexity involved in the course of service running controls. Moreover, this proposed solution can be further accelerated by harvesting distributed and parallel computing technologies in the proposed computing framework.

Similar to the analysis to the VNE problem in [129], for the general version of the considered problem, theoretical bounds do not exist. It is quite challenging to model the analytic optimality bounds and convergence rate, due to possibly random termination of the iteration progress. A reasonable direction is to explore stochastic and approximate ratio analysis [137] in future work.

3.6 Conclusions

In this chapter, we have highlighted the network utility degradation problem for NFV in dynamic networks, and accordingly proposed a robust NFV deployment solution SRA that is robust against network state changes within a certain running period. By exploiting the problem-specific structures, a distributed computing framework with two-level decomposition has been designed to facilitate a distributed implementation of the proposed SRA model in large-scale networks. The simulation experiments have confirmed the performance degradation of existing NFV solutions in dynamic networks, and demonstrated that the proposed SRA solution can achieve up to 5x performance improvement against the compared algorithms. The obtained solution has presented low sensitivity towards parameter errors and even worked robustly with up to 50% statistical errors.

For the future work, more considerations are to be explored in terms of the more general implementation of the SRA algorithm and the challenges in theoretical analysis.

First, in this chapter, a fixed pricing strategy is used to control the whole admission and placement decisions. Considering the dynamics in networking market and traffic patterns, more dynamic pricing model is expected. For example, if a request has a long lease time and the market price of resources allocated to that request keeps fluctuating over the lease period, a full-fledged economic model will be required in order to model the revenue function.

Additionally, the model or statistical information of networking environment, such as traffic patterns and the PDFs of network variations, are required in this chapter. The accurate and timely acquisition of these knowledge in a dynamic networking environment is non-trivial. Therefore, a knowledge-free model extension is expected for future work to release the assumption of complete and stationary PDF information with technologies *e.g.*, multi-armed bandit learning theory [135], reinforcement learning [138], *etc*.

Chapter 4

Learning Augmented Online Optimization for Safe Network Slicing in 5G

The stochastic NFV solutions in previous chapter still require the input of the statistics information about the underlying environment. In this chapter, we further release this assuption and consider a networking environment where prior knowledge about the networking environment is not available but can be collectively observed at runtime. Under this setting, this chapter first presents promising approaches to intertwine learning and optimization technologies and then designs in detail a learning augmeted online optimization approach for the targeted problem. Part of the contents in this chapter are summarized from our work in [139].

4.1 Introduction

With the evolution of SDN and NFV, 5G networks have advocated a revolutionary paradigm called network slicing [5] [45] to construct network services. Unlike the large deployment of dedicatedly built network devices in conventional networks, network slicing utilizes VNF to implement individually optimized services on top of the same physical infrastructure. This enables a more flexible, scalable and agile management towards the end-to-end physical resources, including communication and computation resources, radio spectrum, energy, *etc*.

Network slicing has been accepted as an integrated part of the latest 3GPP release [140], but its technical implementation is solution agnostic. With proper assumptions or knowledge on traffic patterns and networking environment, a myriad of slicing solutions based on classic

optimization theories (*e.g.*, combinatorial/convex optimization) have been proposed (related surveys can be found in [5], [141]). As with the dominant implementations of network slicing in the literature, if the full knowledge of traffic and network states is assumed known, *e.g.*, observed/predicted states or Probability Distribution Functions (PDFs), we can always compute in offline an optimal or approximate slicing policy to best respond to these given network states.

However, with today's networks becoming increasingly dynamic, heterogeneous, and complex, the real-time tracking and explicitly modelling of the networking environment are getting increasingly costly or even intractable. The concrete examples can be easily found from these typical 5G scenarios where massive human-/machine-type connections are presented or millimeter wave enabled small cells are densely deployed [35]. As observed in the practical measurement campaigns in [113], the system states of small cells with millimeter wave spectrum evolve rapidly on nearly a millisecond order. Meanwhile, the number of system parameters per 5G node is expected to more than 2000 [96]. In these popular 5G cases, it is strongly vulnerable to the environmental changes for traditional slicing schemes that customize the immediate performance over a given deterministic environment information in the literature. This requires that a slicing system should be able to make decisions in the absence of partial system state information while the resulted solution efficiencies are safeguarded across the whole running trajectory. Compared with the existing challenges addressed in the literature, *e.g.*, solving the NP-hard combinatorial slicing model, this leads to three additional challenges on implementing of a safe network slicing in the considered complex 5G context:

- The slice operation environment frequently evolves with great uncertainties, and explicit environment knowledge/models are unavailable ahead of schedule;
- Timely control response, possibly on a millisecond order, is required in order to secure the system running as desired. This precludes the traditional scaling or migration based dynamic slicing solutions, which re-solve slicing models after each network change. This is because each attempt of solving a usually NP-hard slicing model in large-scale networks is costly and time-consuming;
- In addition to the real-time performance and constraints, the chosen slicing policy should also be able to work properly across the whole system trajectory and to safeguard the long-term system performance and constraints.

Based on the above insights, it is essential for a slicing system to observe environment variations, learn uncertainties, and accordingly plan response actions properly. Therefore, in

this chapter, we aim to safeguard the network slicing in dynamic 5G networks so that the returned slicing policy is always feasible and the resultant system performance is long-term safeguarded.

To address these challenges and objectives, we first investigate the promising approaches to intertwine the learning and optimization technologies and then build a two-stage slicing optimization model with time-averaged constraints and objective. This provides a two-phase control to secure the slicing process: initial slice deployment and long-term slice operation. Directly solving an off-line solution to this problem is intractable since the future realizations of the objective and constraint functions are unknown before decision-making in the considered 5G context. Therefore, we propose a learning augmented online optimization approach by intertwining historical learning and online learning to deploy and operate network slices with both historical records and real-time observations. We prove that the proposed slicing solution is always feasible and nearly optimal, up to a constant additive factor. In the proposed solution, the involved combinatorial model only needs to be solved during the initial deployment phase, while the subsequent operation controls only involve solving a simple continuous program. Therefore, this solution is robust and agile to respond to any dynamics. In the simulation, we demonstrate up to $2.6\times$ improvement when compared with the state-of-the-art baselines.

The major contributions of this chapter can be summarized as follows:

- We present a two-stage model structure with the learning augmented optimization framework to safeguard the network slicing in 5G. This facilitates great robustness and prompt adaptation for the network slicing in a dynamic environment.

- By intertwining the classic learning techniques and traditional optimization tools, we present an approximate solution to solve the hard combinatorial program for the robust slice deployment policy under incomplete system knowledge. An analytical probabilistic bound is provided, which can be improved by increasing the sample size.

- We propose an online slice operation policy with misloading calibration, which enables the deployed network slices to learn from and adapt to real-time observations. This provides a proven better performance bound than the referenced online learning algorithm while maintaining a same solution feasibility.

- Finally, extensive simulation experiments are conducted with the settings in accordance with 5G expectations. Through the comparison with the incumbent network slicing solutions, we demonstrate the efficacy of the proposed learning augmented online optimization approach for the targeted network slicing problem.

The rest of this chapter is structured as follows. We first investigate the related works in Section 4.2. Section 4.3 presents the promising approaches to intertwine the learning and optimization technologies. The implementation of the safe network slicing optimization model is presented in Section 4.4. To obtain a robust slice deployment policy, we propose a historical learning approach in Section 4.5 to approximate the proposed slicing model from historical records. Then, the safe slice operation control solution is presented in Section 4.6. The simulation results are summarized in Section 4.7. Finally, Section 4.8 concludes this chapter.

4.2 Related Work

Network slicing has been identified as the backbone of the rapidly evolving 5G technology [5]. By allowing different parties to instantiate and run software-based network services, this paradigm facilitates the development of service-tailed and truly differentiated services on top of a shared underlying network infrastructure. Gaining momentum from immense 5G applications [35], network slicing has been the focus of an ever-growing community of researchers from both academia and industry.

4.2.1 Network Slice Modelling and Optimization

There are many preliminary deterministic network slicing modelling and solutions reported in the literature, which extensively study the basic implementation of network slicing in static networks. These studies mainly focus on the optimization of the resource solutions with acceptable computing complexity under a given network state. Thus, these solutions are not directly applicable in the considered problem. Related works can be found in *e.g.*, [141], [118]- [68].

There also exist a few studies in the literature striving to address similar resource utility problems for dynamic NFV networks. Among these very few work, Ying *et al.* in [150] addressed the joint optimization problem of dynamic radio resources and computing resource. A robust resource allocation framework is presented in [142] with an iterative algorithm to auto-scale slices in response to the changing environment. With a similar motivation, the resource provisioning solution proposed by Li *et al.* [152] is also proactive although their objective is to assign requests with bounded response time. This is achieved by using slice consolidation with timing abstraction, but the placement of slices is still based on deterministic models, and the instance migration of VNFs is involved when new requests arrive. Split/Merge [153] provides system support for achieving efficient, load-balanced

elasticity when scaling in and out of virtual middleboxes. However, as aforementioned, scaling or migration based dynamic solutions are precluded in our problem. In our previous work in [99], we provided a stochastic solution to the similar problem and showed that explicit PDF models of environmental knowledge can contribute considerable improvements. In this chapter, we release the dependence of explicit environmental models of our prior solution by exploiting the learning and online optimization tools. This makes the resulted slicing solution applicable to more generic settings.

4.2.2 Network Optimization with Online Learning Approaches

Recently, there is also an increasing traction on developing model-based online solutions to handle dynamic networking problems. In this regard, the dynamic systems are explicitly modelled and then controlled by exploiting the structural information or statistical knowledge.

The pioneering works can be traced back to the studies that are based on online computation and competitive analysis [156]. Among the most related studies, Jia *et al.* in [151] investigated the online scaling problem of NFV service chains across geo-distributed datacenters. Their solution aims to handle practical time-varying traffic volumes and relies on the dynamic scaling of VNF instances. Evan *et al.* in [52] studied the online embedding of virtual networks. Their goal is to select high-benefit requests in a way that the likelihood that future requests can be embedded as well is maximized.

Other major attempts are the applications of *Lyapunov* optimization [146] and multi-armed bandits (MAB) theories [135] to model the resource allocation problems operating in dynamic systems. In this case, these classic online learning theories are exploited to make a sequence of control decisions with progressively learned knowledge about system dynamics and to optimize the long-term cumulative rewards. For example, Mao *et al.* developed in [154] a *Lyapunov* optimization based dynamic computation offloading solution for mobile-edge computing applications.

Huang *et al.* in [144] studied the learning-aided stochastic network optimization with dual learning and online queue-based control. Neely in [147] investigated the application of *Lyapunov* theories for the distributed stochastic sensor network problem. The applications of MAB can also be found in the problems for antenna beam selection [102], mobile edge computing [103], *etc*.

However, all these works are mainly focused on addressing the sequential online control problems under the imperfect system state information. This is achieved by repeatedly solving their base models with progressively collected new observations across time. In contrast, this chapter highlights the stochastic combinatorial hardness of the considered problem. Herein, it is not supported to repeatedly solve the combinatorial models with

new knowledge. Additionally, we attempt to intertwine the classic learning and online optimization theories so that we can integrate both empirical data experience and domain expertise to enhance the slicing solution.

4.3 Learning Augmented Optimization: Promising Integrative Approaches

Before the analysis towards the safe network slicing problem, we first present the promising integrative approaches to implement the learning augmented optimization. Nowadays, deploying learning based intelligent mechanisms is gaining momentum for 5G networks to control the massive traffic volume in diverse networking landscapes, such as high mobility and machine-to-machine connectivity [97]. With the capabilities of inferring decisions from historical experiences and observations, learning based solutions present a great complement to the existing optimization based solutions in particular in the following three situations:

Problems are un-modelable: 5G expands its coverage to more emerging applications. To support the slicing of these services with diverse differentiated policies, multiple KPIs are required to be balanced and optimized for the new system designs, including delay, reliability, connection density, *etc*. These KPIs might be either dependent or conflict, making the modelling of the problem with a unified optimization model already intractable, not to mention the complexity in tracking the ever-changing service requirements and networking environment. Nevertheless, learning based solutions are able to avoid the needs of explicitly modelling the target problems and environment. The learned solutions have also higher-level generality and are versatile to a set of general problems with the same patterns.

Problems need unified solutions: In traditional communication system designs, critical modules are usually optimized with individual models and problem-specific algorithms. Such a processing has huge complexity and efficiency problem in the implementation of slicing systems, since a single network slice instance could include the whole end-to-end network components and resources from the radio boundary to datacenter. With the data records of history system operations, it is possible to learn a unified solution from past controls and network states for the cross-layer optimization of all cascaded modules without going into the details of system structures.

Models are expensive to solve: For network slicing operations, the involved service and resource optimization problems are often formulated as constrained combinatorial programs, which are NP-hard and possess exponentially increased computation complexity. Existing heuristic or approximation approaches to solving this kind of models do not work well in

terms of either performance or computation cost. Data-driven approaches, on the other hand, open a new window for such problems since these learned decisions from data do not require directly solving these complicated models any more.

However, different from the prosperity of AI techniques in the data analysis applications *e.g.*, image/video/speech recognitions, the application of many advanced AI techniques like machine/deep learning is still in its infancy in communication and networking systems. The challenges include the difficulties in characterizing the appropriate input and output patterns for a learning system to correctly reflect the highly dynamic nature of large-scale heterogeneous networks. Moreover, the huge training cost in terms of both computation and convergence time are still open problems for their applications in real-time networking systems. Finally, the requirement of carrier-grade service reliability is also beyond the accuracy of existing learning based optimization solutions. For example, at least 10% of the results are invalid tours when applying the latest pointer networks to solve the classic Travelling Salesman Problem (TSP) [98].

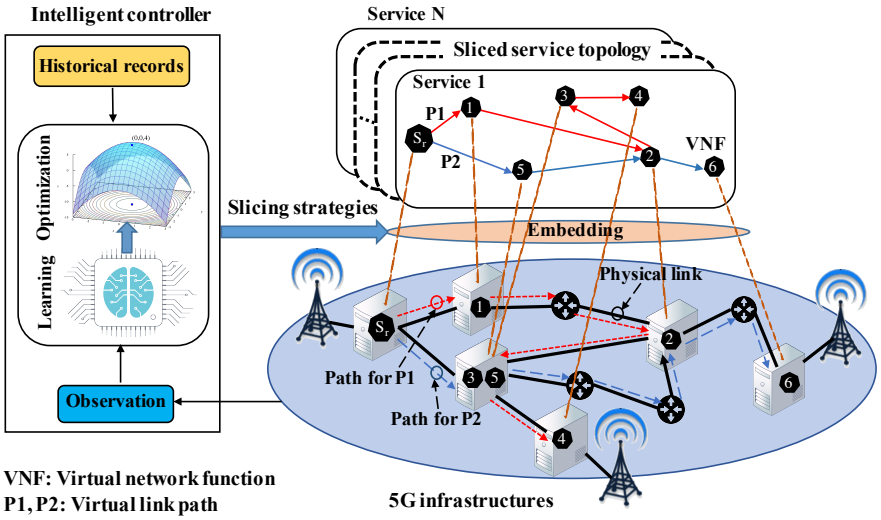


Fig. 4.1 A general framework of learning augmented optimization system for network slicing in 5G: The intelligent controller decides the VNF embedding and routing strategies based on the learning augmented optimization, and then the requested network slices are embedded accordingly in the available 5G infrastructures.

Looking ahead, learning augmented optimization, through greater integration of multiple AI tools (*e.g.*, classic optimization, control theories and machine learning) into networking architecture, is gaining increasing traction. As depicted in Fig. 4.1, such an approach can integrate both the knowledge learned from empirical data and domain expertise with explicit optimization models. This provides a promising way to develop a cognitive network that will show network-wide intelligence to meet the management requirements of sliced resources and services in 5G.

Fig. 4.1 illustrates the general framework for learning augmented optimization and its application in the network slicing problem. Essentially, this retains the basic optimization structure, but more learned information from historical records and run-time observations are introduced to improve the optimization outcomes. The efficacy of learning components here includes reduced computation time and/or optimality loss, or solution robustness. Depending on the the availability of prior knowledge about the environment and the needs of different optimization problems, there exists serveral feasible approaches to augment the optimization process with learning. Next, we investigate the learning techniques that are feasible to augment the applications of classic optimization theories for network softwarization in 5G. Fig. 4.2 shows a family-tree of these techniques. Depending on whether structural models are available, these approaches can be summarized as the following two categories.

4.3.1 Optimization with Model-driven Learning Approaches

If structural information or statistical knowledge are available so that explicitly mathematical models can be developed for the stochastic dynamic systems, the optimal policies can be found analytically. In this regard, classic learning theories from stochastic learning, multi-armed bandit and Lyapunov optimization are powerful for constructing such optimization frameworks.

Optimization with stochastic learning: This approach models the system uncertainties and dynamics as certain stochastic processes. Historical records are first used to construct an internal model of the transitions and outcomes in the environment with given probability distributions. Appropriate actions can then be chosen by searching or planning under the model constructed from the data. This is a statistically efficient way to use experience. Provided that constant re-planning is possible, this allows action selection to be readily adaptive to the changes during the transition contingencies. This approach is particularly useful for problems when system mechanisms are well understood, *e.g.*, optimization problems for radio resource management and utilization [100], [101], since many well designed channel models have been readily available to capture the critical system features. The work of Chapter 3 can be linked to this category.

Multi-armed bandits: Multi-armed bandits (MAB) have been used to model the problems of proportioning resources among competing objectives under a fixed budget. The properties of these objectives are only partially known at the time of decision making, but which may become better understood as time passes. In this case, MAB aims to make a sequence of control decisions with progressively learned knowledge about these objectives to optimize long-term cumulative rewards. The critical idea of MAB is to balance the trade-off between the exploitation of specific control action that has the highest immediate payoff and

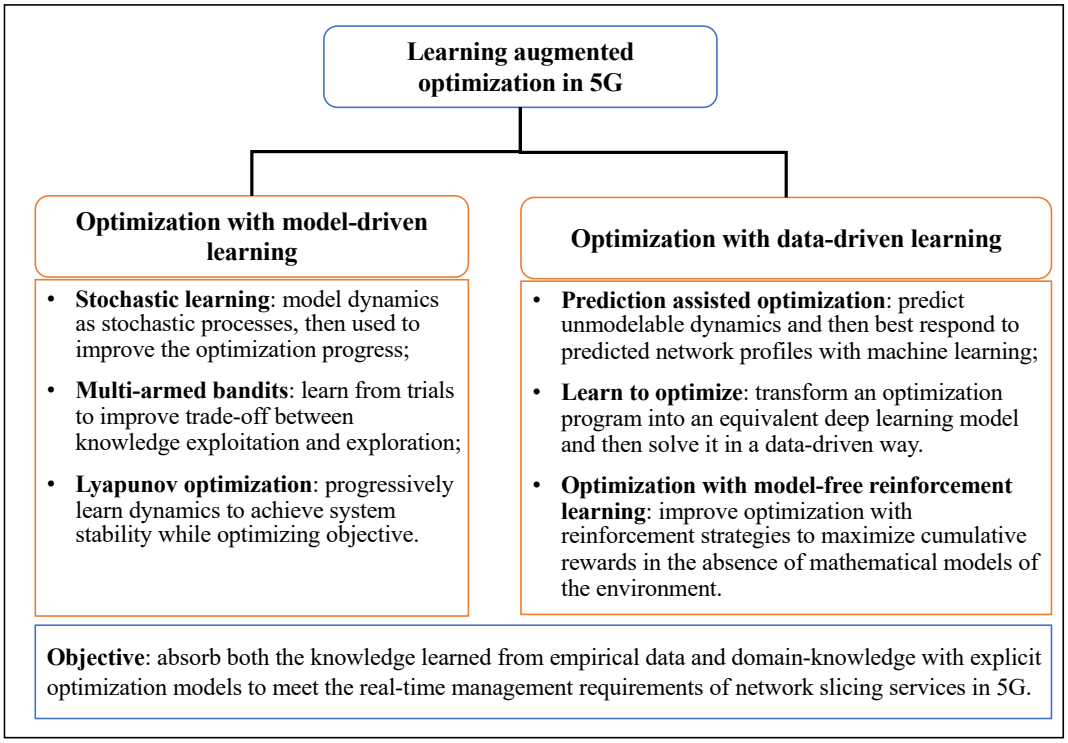


Fig. 4.2 Learning augmented optimization techniques.

the exploration of the other actions that might have a better payoff after more knowledge are learned. The applications of MAB can be found in problems for beam selection [102], mobile edge computing [103] *etc*.

Lyapunov optimization: Lyapunov functions have been extensively used in control theory to achieve the stability of dynamic systems. The goal of Lyapunov optimization for dynamic systems is to achieve system stability (*i.e.*, satisfy cumulative system constraints) while optimizing some performance objectives. This is achieved by progressively learning system uncertainties from new observations and then augmenting the control actions accordingly. With the learned knowledge, the resulting control solution can be steered to achieve the exact same efficiency as the optimal control strategy that is derived when all system dynamics are known as a priori. This approach is particularly powerful for optimizing a stochastic system with time-averaged objectives and constraints, *e.g.*, minimizing averaged cumulative energy consumption [104] or maximizing long-term averaged throughput [105].

4.3.2 Optimization with Data-driven Machine Learning

In many cases, however, mathematical models of environment and even action rewards are not known, and we need to observe and analyze the sample paths of the system to

determine the performance and make improvement decisions. In this case, the diverse data-driven machine learning tools, including supervised/unsupervised/model-free reinforcement learning, are applicable to improve the optimization processes. Machine learning is a subset of AI that uses statistical techniques to learn from and make predictions on data, without being explicitly programmed [97]. Considering the aforementioned challenges of directly applying machine learning in our problem, the traditional optimization procedures can be improved with machine learning algorithms in the following two ways.

Prediction-aided optimization through machine/deep learning: One critical issue hindering the further development of network slicing is the complex system uncertainties and dynamics, which are difficult to model analytically in the considered 5G context. To address this challenge, a natural approach is to first predict the realizations of these uncertainties in the near future and then to proceed by best responding to the predicted profiles. When compared with the existing optimization approaches that customize immediate rewards/costs over observed states, the additional predicted knowledge can enhance these traditional solutions with a better robustness. Depending on whether prior labels are available in the database, supervised/unsupervised learning has been widely used for the tasks of data-driven prediction. For example, in [106], the authors exploited supervised learning to predict wireless data and location interface configurations that can be used to optimize energy consumption in mobile devices. Their experimental results showed that it is possible to achieve up to 90% successful prediction and 50% improvement of energy saving with the aid of neural networks and k-nearest neighbor algorithms.

Optimization with model-free reinforcement learning: Adequate history observations are required for learning a model of how the environment works. When the data acquisitions are difficult or time costly, model-free reinforcement learning will then be a way out. The problems of interest in the optimization with model-free reinforcement learning are particularly concerned with an exact computation of optimal solutions but without estimation or use of an explicit environment model. The objective is to take proper real-time actions to maximize certain cumulative rewards. This is coincident with the design objective for the 5G network slicing systems, since long-term averaged performance is more meaningful than an instant outcome for a stochastic system. One of the most classic examples is Q-learning [107], which directly estimates the optimal Q-values of each action in each state, from which a strategy is derived by choosing the action with the highest Q-value in the current state.

Deep learning to directly solve an optimization program: Instead of serving as an auxiliary component, deep learning in this kind of approaches is directly used to generate a solution that respects the whole constraints of an optimization program. Once trained with labelled data set or reinforcement policies, the obtained learning model can directly

generate an optimal/suboptimal solution to the targeted problem without searching repeatedly the whole solution space from the scratch for every problem instance. This points out a promising direction to solve particularly many NP-hard combinatorial optimizaiton problems. Preliminary testaments to the applications of these approaches can be found in e.g., [172]-[173], [98]. A prominent example is from [175], where the graph attention network has been proven very effective for solving the classic TSP problem.

As a showcase, an learning augmented online optimization with Lyapunov theory will be developed in the following parts of this chapter. In next chapter, an approach that applies deep learning tools to directly solve the optimization process will also be provided. We note that the integrative technology of learning augmented optimization supports a wide range of combinations between the diverse learning and optimizaiton technologies. The work of this thesis serves as a start point to stimulate the innovative use of learning augmented optimization approaches in more applications.

4.4 System Model

With the technology of NFV, a sliced network service is constructed with solely required VNFs that are chained in a specific order according to its service policies. The installation/teardown and operations of network slices are mutually independent, although the underlying infrastructures are shared. In this case, one critical task for network slicing is to plan the utilization of the shared resources with adherence to diverse service and resource constraints. This is non-trivial and more challenging than planning 4G network services due to the rapidly expanded scales of network elements and connections as well as the frequently changing network environment in 5G.

Let us treat the 5G slicing system as a discrete-time stochastic system with time-varying resources c_t. Upon the given limited system resources, slicing requests with random resource demands d_t are required to schedule and control. At the beginning of each time slot $t \in \{0,1,2,\dots\}$, the system controller is able to observe a state update $\omega_t = (d_t, c_t)$, which specifies the independent realizations of current traffic and network states. The network slicing optimization problem aims to decide a robust slicing policy at the beginning of every scheduling interval T so that the slicing system can be configured and operated with full respect to the desired objectives and constraints.

4.4.1 Preliminaries

Considering the dynamic system feature in this chapter, we apply partial admission control policy. Consequently, requests can be accepted (*i.e.*, deployed) but run with compressed service rates rather than be directly rejected when the available resources at the decision-making time are not enough to fully meet the required demands. This ameliorates the over-/under-loading problem caused by partial observations at decision time. In this setting, the slicing policy can be split into two parts: slice deployment policy and slice control policy.

We first clearly define the two policies to be optimized in this chapter as follows.

Definition 1 (**Slice deployment policy** $\pi_+ \in P_+$)*:* is a policy vector specifying which slice requests are accepted, where to instantiate and how to chain the required VNFs along with routing paths in the underlying physical networks.

Definition 2 (**Slice control policy** $\pi_- \in P_-$)*:* is a contingency plan for choosing a single running control action to adapt the flow rates of deployed slices to a given network event observation under the chosen slice deployment policy π_+.

where P_+, P_- are the solution spaces for the two policies, respectively.

In this chapter, we denote a slicing policy $\pi := \{\pi_+, \pi_-\}$ to be ***safe*** if it is feasible across the runtime (*i.e.*, without violating any real-time or time-averaged constraints) while optimizing the long-term system performance.

4.4.2 The Deterministic Network Slicing Optimization Problem

As aforementioned, real-time scaling and migration of deployed slice instances are not considered. In this case, the slicing system can be sequentially controlled as a two-stage process: (i) decide a long-term feasible and optimized slice deployment policy with an initial knowledge, and (ii) then under the given deployment, adaptively control the running flow rates allocated to the deployed network slices according to the real-time environment changing. As with the extensively studied slicing models in the literature, we can abstract, without loss of generality[1], the network slicing optimization problem under any given system state ω as the following **T**wo-**S**tage **S**licing (**T2S**) Program:

$$(\textbf{T2S}\text{ model}) \quad \pi = \underset{\pi_+, \pi_-}{\operatorname{argmin}} \; f(\pi_+, \pi_-, \omega) \tag{4.1}$$

$$s.t. \qquad \pi_+ \in P_+ \text{ and } \pi_+ \text{ is binary} \tag{4.2}$$

$$\pi_- \in \underset{\pi_- \geq \mathbf{0}}{\operatorname{argmin}} \Big\{ f(\pi_-, \omega | \pi_+) \Big| u_l(\pi_-, \omega) \leq 0, l = 1, 2, \ldots, L \Big\} \tag{4.3}$$

[1]Equality constraints can be equivalently expressed through two inequality constraints.

where

- $\{f, u_l\}$: the objective function and a set of utility functions specifying the real-time constraints required for running the deployed network slices in the physical network. Their values are random, dependent on the realization of system state ω;
- P_+: the solution space to be defined by a set of constraints to guarantee that the chosen π_+ is a valid slice deployment policy (*e.g.*, satisfying flow conservation). A concrete example is provided in (4.45)–(4.47);
- π_+: a vector of decision variables indicating which physical nodes and link paths are used to construct the accepted network slices;
- π_-: real-time running flow rates allocated to the accepted network slices under the given slice deployment policy π_+ and system state ω;

As claimed in extensive existing studies *e.g.*, [142], under the given system state ω, solving this problem is NP-hard. By this token, many reported exact or approximated computing strategies in the literature can be used to solve this problem, such as dynamic programming [55] or integer relaxation [143]. However, the resulted solution can only customize the system performance that is best respond to the given system state. When the system evolves frequently and explicit models of the system environment are unavailable a-priori, the obtained slicing solutions through these traditional approaches are not safe across time.

Next, we safeguard the long-term performance of this **T2S** model by incorporating time-averaged performance metrics.

4.4.3 Safeguarding Long-Term Performance with Time-Averaged Metrics

Let us denote $\{\pi_-^t\}_{t=0}^{\infty}$ as the slice control policy across t and $f(\pi_+, \pi_-^t, \omega_t)$ as the time-variant objective function. Then, we define the time-averaged objective function as follows:

$$\bar{f}\big(\pi_+, \{\pi_-^t\}_{t=0}^{\infty}\big) = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}_{\omega}\big[f(\pi_+, \pi_-^{\tau}, \omega_{\tau})\big] \tag{4.4}$$

Likewise, we define a set of time-averaged utility functions $\bar{g}_k, k \in \{1,2,\ldots,K\}$ through its time-variant function $g_k(\pi_+, \pi_-^t, \omega_t)$ as follows:

$$\bar{g}_k\big(\pi_+, \{\pi_-^t\}_{t=0}^{\infty}\big) = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}_{\omega}\big[g_k(\pi_+, \pi_-^{\tau}, \omega_{\tau})\big] \tag{4.5}$$

In order to safeguard the slicing optimization solutions in 5G, we first extend the **T2S** model in (4.1) with its time-averaged objective and constraints. This results in our new proposed **T**wo-**S**tage **S**afe **S**licing (**T3S**) model as follows:

$$(\textbf{T3S model}) \quad \pi = \underset{\pi_+, \{\pi_-^t\}_{t=0}^{\infty}}{\operatorname{argmin}} \ \bar{f}\big(\pi_+, \{\pi_-^t\}_{t=0}^{\infty}\big) \tag{4.6}$$

$$s.t. \quad \bar{g}_k\big(\pi_+, \{\pi_-^t\}_{t=0}^{\infty}\big) \leq \bar{c}_k, k \in \{1,2,\ldots,K\} \tag{4.7}$$

$$\text{Constraints } (4.2) - (4.3), \forall t \tag{4.8}$$

where $\bar{c}_k$ is the upper bound (or cost budget) of the time-averaged utility $\bar{g}_k$.

In this new slicing model, three differences can be highlighted when compared with the existing system formulations. First, by optimizing the time-averaged objective, it captures the concerns on the long-term performance safety in the considered stochastic system. Second, with the time-averaged constraints in (4.7), this model provides a higher flexibility to consolidate both the long-term and real-time running requirements for deploying the sliced services in dynamic networks. Finally, the combinatorial solution for π_+ is optimized to respond to long-term system behaviours. This circumvents the complicated amendments to π_+ (*e.g.*, dynamic scaling or migrating deployed slice instances). Alternatively, the adaptation to real-time changes is handled through simplified continuous programs in the slice operation process.

A concrete application example of the **T3S** model is that network providers want to optimize their long-term revenues from the provisioning of network slices. The available resource capacities are time-varying due to *e.g.*, wireless channel fluctuation, traffic variations [99], [155]. In this case, the providers need to keep the average operation cost for *e.g.*, energy or bandwidth within given budgets. Meanwhile, both the real-time (*e.g.*, delay, jitter) and long-term (*e.g.*, packet loss/service interruption rate) service qualities should meet the contracted service level agreements. This is fundamentally more complicated than the deterministic counterparts.

In **T3S** model, we need to infer at the beginning of every scheduling interval a robust slicing policy that not only has an impact on the immediate system performance but also on the future ones. However, the objective function $f(\pi_+, \pi_-^t, \omega_t)$, utility functions $g_k(\pi_+, \pi_-^t, \omega_t)$

and $g_l(\pi_-^t, \omega_t)$ are unknown before decisions. Thus it is intractable to directly solve an optimal off-line solution to this problem.

Based on the above analyses, we resort to enhance the traditional stochastic optimization algorithms with learning components to solve this problem in two sequential steps. Specifically, a historical learning process is first introduced to extract a robust slice deployment policy. This eschews frequently re-solving the complicated combinatorial program in **T3S**. Considering the imperfect matching of the long-term optimal policy with the real-time operation environment, an online adaptation scheme is presented in the slice operation process. This provides an extra tunnel to learn from the real-time observations to further secure the slicing system.

4.5 Robust Slice Deployment with Historical Learning

A system's behavior is usually represented by a model, or by the trajectories that record the operation history of a system. As the inevitable absence of future system knowledge in the targeted problem, we resort to improve the slice deployment policy by learning from historical system records, which are commonly available in real-life communications systems.

Let $\{\hat{\omega}_t\}_{t=1}^{W_t}$ be a learned trajectory of the system state ω from historical records for the next W_t time window. Then, through approximating the cost and objective functions with the learned trajectory, we can get a **S**ampled reduction of the stochastic **T3S** model (**T4S**) as the following deterministic program:

$$(\textbf{T4S model})\ \pi_+ = \underset{\pi_+}{\operatorname{argmin}}\ \frac{1}{W_t+1}\sum_{t=0}^{W_t} f(\pi_+, \pi_-^t, \hat{\omega}_t) \tag{4.9}$$

$$\begin{cases} s.t. \\ \quad \frac{1}{W_t+1}\sum_{t=0}^{W_t} g_k(\pi_+, \pi_-^t, \hat{\omega}_t) \le \bar{c}_k, k = 1,2,\ldots,K \\ \quad \pi_+ \in P_+ \text{ and } \pi_+ \text{ is binary} \\ \quad \pi_-^t \in \operatorname{argmin}_{\pi_-^t \ge \mathbf{0}} f(\pi_-^t, \hat{\omega}_t | \pi_+), \forall t = 0,1,\ldots,W_t \\ \quad \begin{cases} s.t. \\ \quad\quad u_l(\pi_-^t, \hat{\omega}_t) \le 0, l = 1,2,\ldots,L \end{cases} \end{cases} \tag{4.10}$$

In **T4S**, the reduction of a stochastic problem to its deterministic approximation resembles the sampling based techniques (*e.g.*, [144], [125], [158]). Instead of a direct implementation of **T4S** with i.i.d historical samples, the manipulation with learned outcomes reaps two-fold benefits: i) this provides an interface for integrating advanced prediction tools to extract the temporal dependencies of the underlying system evolution, and 2) saves tremendous

computation on unnecessary samples when solving the combinatorial model. As a deterministic problem, many existing algorithms (fractional rounding, heuristic or decomposition, as surveyed in [67]) can be invoked to solve an exact or suboptimal solution for the **T4S** model, which is out of the focus in this work.

Misloading discussion: When an approximate solver is applied, the resultant suboptimal policy π_+ tends to accept less requests (*i.e.*, under-loading case) so that the given cost budget constraint will not be violated. Once under-loaded at the beginning, the system will not be able to override π_+ across the scheduling interval even a better networking environment is detected. In contrast, if over-loaded due to imperfect prediction accuracy, more cost will be devoted to maintaining the active of deployed slices. Thus, both cases will lead to system degradation. In Section 4.6.1, a misloading (i.e., under-/over-loading) calibration scheme will be introduced to ameliorate this issue.

The optimality of the learned deployment policy depends on both the prediction accuracy and the distribution consistency between historical samples and future outcomes. When the underlying stochastic processes are ergodic[2], the simplest learning strategy would be a directly random sampling from the historical records. However, the **T4S** model is versatile to work with any suitable learning technique. We validated in Section 4.7 that the proposed solution gains remarkable improvement even with a random sampling based learning strategy. When the objective function f is linear with respect to ω_t (as the case in the Benchmark problem in Section 4.7.1), we can further get an analytical *probabilistic bounds* on the objective value that improves (in expectation) with increasing sample/prediction window size. This is presented in the following *Proposition*.

Proposition 1*:* Assume f is linear with respect to the ergodic process ω_t. Let π_+^* be the optimal solution of **T3S** achieved through some '*genius*' algorithm that knows the true system trajectory $\{\omega_t\}_{t=1}^{\infty}$ a-priori. Then, we have

$$\bar{f}\big(\{\pi_-^t\}_{t=0}^{W} \mid \check{\pi}_+\big) \geq \bar{f}\big(\pi_+^*, \{\pi_-^t\}_{t=0}^{\infty}\big) \geq \mathbb{E}\big[\bar{f}\big(\hat{\pi}_+^*, \{\pi_-^t\}_{t=0}^{W_t}\big)\big] \tag{4.11}$$

$$\mathbb{E}\big[\bar{f}\big(\hat{\pi}_+^*, \{\pi_-^t\}_{t=0}^{W_t+1}\big)\big] \geq \mathbb{E}\big[\bar{f}\big(\hat{\pi}_+^*, \{\pi_-^t\}_{t=0}^{W_t}\big)\big] \tag{4.12}$$

where $\check{\pi}_+$ is a fixed policy obtained by some approximation procedure, *e.g.*, solving **T4S** with W samples; $\hat{\pi}_+^*$ is the optimal solution of **T4S** under the samples $\{\hat{\omega}_t\}_{t=1}^{W_t}$.

Proof. First, under any feasible deployment policy, say $\check{\pi}_+$, the achievable value $\bar{f}\big(\{\pi_-^t\}_{t=0}^{\infty} \mid \check{\pi}_+\big)$ is clearly a *rigorous upper bound* on $\bar{f}\big(\pi_+^*, \{\pi_-^t\}_{t=0}^{\infty}\big)$. As presented in [125], $\bar{f}\big(\{\pi_-^t\}_{t=0}^{\infty} \mid \check{\pi}_+\big)$ can be estimated by solving the resultant continuous program under a large set of historical

[2]An ergodic process is a stochastic process whose behavior does not depend on the initial conditions and whose statistical properties do not vary with time [157].

samples. Therefore, the left-side inequality in (4.11) can provides an upper bound estimation towards the true objective value. The right-side inequality of (4.11) shows a *probabilistic lower bound*, which can be calculated by solving a set of sampled instances of **T4S**. This is proved as follows.

According to the definition of (4.4), we can have

$$\bar{f}\big(\pi_+^*, \{\pi_-^t\}_{t=0}^{\infty}\big) = \min \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\big[f(\pi_+, \pi_-^\tau, \omega_\tau)\big] \tag{4.13}$$

$$= \min_{\pi} \mathbb{E}\big[\lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} f(\pi_+, \pi_-^\tau, \omega_\tau)\big] \tag{4.14}$$

$$\overset{(a)}{=} \min_{\pi} \frac{1}{W_t+1} \mathbb{E}\big[\sum_{t=0}^{W_t} f(\pi_+, \pi_-^t, \omega_t)\big] \tag{4.15}$$

$$\geq \mathbb{E}\big[\min_{\pi} \frac{1}{W_T+1} \sum_{t=0}^{W_t} f(\pi_+, \pi_-^t, \omega_t)\big] \tag{4.16}$$

$$\overset{(b)}{\geq} \mathbb{E}\big[\frac{1}{W_t+1} \sum_{t=0}^{W_t} f(\hat{\pi}_+^*, \pi_-^t, \omega_t)\big] \tag{4.17}$$

$$= \mathbb{E}[\bar{f}(\hat{\pi}_+^*, \{\pi_-^t\}_{t=0}^{W_t})] \tag{4.18}$$

where (a) follows with the ergodic theorem [157]; (b) follows since $\hat{\pi}_+^*$ is the optimal solution of the **T4S** model.

Next, we prove the monotonic feature of $\mathbb{E}\big[\bar{f}\big(\hat{\pi}_+^*, \{\pi_-^t\}_{t=0}^{W_t}\big)\big]$ with respect to sample size as follows:

$$\mathbb{E}\big[\bar{f}\big(\hat{\pi}_+^*, \{\pi_-^t\}_{t=0}^{W_t+1}\big)\big] = \mathbb{E}\big[\min_{\pi} \frac{1}{W_t+2} \sum_{t=0}^{W_t+1} f(\pi_+, \pi_-^t, \omega_t)\big] \tag{4.19}$$

$$= \mathbb{E}\big[\min_{\pi} \frac{1}{W_t+2} \sum_{t=0}^{W_t+1} \frac{1}{W_t+1} \sum_{\tau=0,\tau\neq t}^{W_t+1} f(\pi_+, \pi_-^\tau, \omega_\tau)\big] \tag{4.20}$$

$$\geq \frac{1}{W_t+2} \sum_{t=0}^{W_t+1} \mathbb{E}\big[\min_{\pi} \frac{1}{W_t+1} \sum_{\tau=0,\tau\neq t}^{W_t+1} f(\pi_+, \pi_-^\tau, \omega_\tau)\big] \tag{4.21}$$

$$= \frac{1}{W_t+2} \sum_{t=0}^{W_t+1} \mathbb{E}\big[\min_{\pi} \frac{1}{W_t+1} \sum_{\tau=0}^{W_t} f(\pi_+, \pi_-^\tau, \omega_\tau)\big] \tag{4.22}$$

$$\geq \mathbb{E}\big[\bar{f}\big(\hat{\pi}_+^*, \{\pi_-^t\}_{t=0}^{W_t}\big)\big] \tag{4.23}$$

Proof ends. ■

4.6 Adaptive Online Slice Operation with Misloading Calibration

Once instantiated under the learned slice deployment policy $\hat{\pi}_+$ from the history records, the original **T3S** model reduces to a simplified stochastic continuous program, which aims to exert a slice running control process to further secure both the real-time and long-term system performances. Since the future realizations and PDF models of ω_t are unavailable a-prior, we now propose an online learning approach to solve the **T3S** for the slice **R**unning control policy (called **T3S-R** model hereafter). In the following, $\hat{\pi}_+$ is identified as a known parameter and will not be shown explicitly in the **T3S-R** problem. Additionally, we assume the **T3S-R** is convex on π_-^t, which is aligned with most of existing resource allocation problems for network slicing.

4.6.1 Online Running Control with Misloading Calibration

In an online process, a solution has the following structure: At the beginning of every slot t, the system controller observes a realization of ω_t, and then a slice running control π_-^t is derived accordingly. In order to achieve the long-term objective, we construct the online process with the theoretical support of queuing networks [146].

Let us treat each time-averaged constraint in (4.7) as a virtual queuing process. For each constraint $k \in \{1,2,\ldots,K\}$, define a virtual queue Q_k^t with initial condition $Q_k^0 = 0$. The queue backlog updates over time via:

$$Q_k^{t+1} = \max\{Q_k^t + g_k(\pi_-^t, \omega_t) - \bar{c}_k, 0\} \tag{4.24}$$

The connection of such a queuing network with the **T3S-R** is that if we control to stabilize the queue Q_k^t, the average of the "arrival process" $g_k(\pi_-^t, \omega_t)$ must be less than or equal to that of the "service process" $\bar{c}_k$. Consequently, the resultant control sequence will be a feasible solution meeting the time-averaged constraint $\bar{g}_k\big(\hat{\pi}_+, \{\pi_-^t\}_{t=0}^{\infty}\big) \leq \bar{c}_k$.

Let $Q_t = [Q_1^t, Q_2^t, \ldots, Q_K^t]$ be the vector of queue backlogs, and define the *Lyapunov function* L_t as follows:

$$L_t = \frac{1}{2}\|Q_t\|^2 = \frac{1}{2}\sum_{k=1}^{K}[Q_k^t]^2 \tag{4.25}$$

The value L_t is a scalar measure of the size of the queue backlogs till t. If we take actions to consistently push this value down, then the queues will be stabilized (*i.e.*, satisfy the time-averaged constraints in (4.7)). Fig. 4.3 illustrates the online slice running control process with the virtual queuing network.

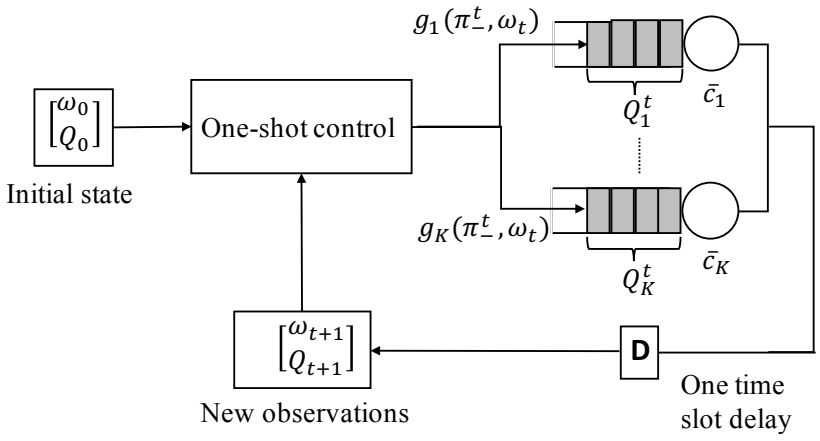


Fig. 4.3 Online slice running with the virtual queuing network.

In this online framework, the system observes the current queuing state Q_t and new realizations of ω_t at the beginning of every slot t. Then, an one-shot control is exerted accordingly to adapt the slice running to the new observations. Motivated by the theory in [146], we implement the one-shot control by minimizing the following drift-plus-penalty expression:

$$\min_{\pi_-^t \geq \mathbf{0}} \quad V f(\pi_-^t, \omega_t) + \sum_{k=1}^{K} Q_k^t g_k(\pi_-^t, \omega_t) \tag{4.26}$$

$$s.t. \qquad u_l(\pi_-^t, \omega_t) \leq 0, l = 1, 2, \ldots, L \tag{4.27}$$

where V is a non-negative weight that will be shown to affect the performance tradeoff.

Behind the one-shot control strategy is the intuition that more budget than the average $\bar{c}_k$ can be provided so that we can fully exploit the '*good*' state ω_t to collect a better objective value $f(\pi_-^t, \omega_t)$. Balanced by the surplus budget from the under utilization of $\bar{c}_k$ when ω_t is in '*bad*' quality, the time-averaged utility budget constraints in (4.7) can still be preserved.

With the convergence analysis in [159], the control strategy from (4.26) is known to provide an $O(1/V)$ approximation to the optimality of **T3S-R** with a convergence time of $O(V^2)$. However, considering the impacts of misloading resulted from imperfect prediction accuracy and the approximation for solving the **T4S** model, we introduce a misloading calibration scheme by extending the queue update function as following:

$$Q_k^{t+1} = \max\{Q_k^t + g_k(\pi_-^t, \omega_t) - (\bar{c}_k + \delta_k), 0\} \tag{4.28}$$

where $\delta_k = \max\{\bar{c}_k - \frac{1}{W_t+1}\sum_{\tau=0}^{W_t} g_k(\pi_-^\tau, \hat{\omega}_\tau), 0\}$.

δ_k measures the positive cost gap between the cost budget and the estimated utility due to the suboptimality of $\hat{\pi}_+$. With the calibration from δ_k, (4.28) imposes a positive utility offset

to the available cost budget. This avoids the unnecessary penalty on the objective $f(\pi_-^t, \omega_t)$ in subsequent controls. As analyzed in Section 4.6.2, such a calibration leads to a better convergence performance.

Finally, by putting all together, we are now ready to present the overall learning augmented optimization approach for the safe network slicing in Algorithm 4.

Algorithm 4 The proposed learning augmented optimization approach for safe network slicing.

Input: Historical system trajectory, network and requested slice topologies, V, W_t, T.
Output: Slice deployment policy $\hat{\pi}_+$ and online running flow rate π_-^t.

▷ *Slice deployment at the beginning of every scheduling interval*

1: *(Historical Learning)* Generate the realizations of ω_t for the next W_t time slot.
2: *(Deployment Policy Approximation)* Infer the slice deployment policy $\hat{\pi}_+$ by solving **T4S** model under samples $\{\hat{\omega}_t\}_{t=1}^{W_t}$.
3: Instantiate the accepted network slices according to policy $\hat{\pi}_+$.

▷ *Online slice running control*

4: *(New Observations)* Collect new observations for ω_t, Q_t at the beginning of time slot t.
5: *(One-shot Control)* Decide the current slice running policy π_-^t by solving the deterministic problem defined in (4.26).
6: *(Queue Update)* Observe the resulting utility $g_k(\pi_-^t, \omega_t)$, and update virtual queues by (4.28).
7: Go to Step 4 if $t = t+1$ is not the beginning of new scheduling interval
8: Otherwise go to Step 1.

4.6.2 Theoretical Analysis

We first show the running constraint violation across iteration in Algorithm 4.

Lemma 1*:* Let $Q_k^0 = 0$ and Q_k^t updates according to (4.28). Then, Algorithm 4 satisfies: for all $t > 0$, i) Q_k^t is mean-rate stable (*i.e.*, $\mathbb{E}[Q_k^t]$ is upper bounded by a finite value), and ii)

$$\bar{g}_k\big(\hat{\pi}_+, \{\pi_-^\tau\}_{\tau=0}^{t-1}\big) - \bar{c}_k \leq \mathbb{E}[Q_k^t]/t + \delta_k, \forall k = 1, 2, \ldots, K \tag{4.29}$$

Proof. The boundedness of $\mathbb{E}[Q_k^t]$ can be proved by contradiction. Assume Q_k^t is infinite large at some t. Then, in order to minimize the expression in (4.26), it must return a flow running rate with $\pi_-^t = \mathbf{0}$ and lead to the decrease of queue length for next slot, Q_k^{t+1}, by $\bar{c}_k + \delta_k$. This continues until all Q_k^t stabilize with certain finite queue backlogs. Consequently, under the control of Algorithm 4, it is impossible for Q_k^t to grow to infinity and thus all Q_k^t are mean-rate stable. As shown in (4.29), this property is useful to guarantee a declined constraint violation across iterations. Next, we prove the constraint violation in (4.29).

From (4.28), we have:

$$Q_k^{t+1} \geq Q_k^t + g_k(\pi_-^t, \omega_t) - (\bar{c}_k + \delta_k) \tag{4.30}$$

Summing over $\tau \in \{0, 1, \ldots, t-1\}$ gives:

$$Q_k^t - Q_k^0 \geq \sum_{\tau=0}^{t-1} g_k(\pi_-^\tau, \omega_\tau) - t(\bar{c}_k + \delta_k) \tag{4.31}$$

Dividing by t and using the fact that $Q_k^0 = 0$ gives

$$\frac{Q_k^t}{t} \geq \frac{1}{t} \sum_{\tau=0}^{t-1} g_k(\pi_-^\tau, \omega_\tau) - \bar{c}_k - \delta_k \tag{4.32}$$

Taking expectations and re-arranging terms yield (4.29). ■

The right side of (4.29) presents the running violation on the time-averaged constraints in (4.7). With the boundness of Q_k^t, it is clear that the term $\mathbb{E}[Q_k^t]/t$ vanishes as $t \to \infty$. Additionally, the added calibration offset only takes positive values in the event of system underloading. In this case, $\hat{\pi}_+$ tends to accept less loads, and the real-time running utility $g_k(\pi_-^t, \omega_t)$ is less likely to exceed the average budget $\bar{c}_k$. Consequently, the added calibration offset is inoffensive to the constraint in (4.7). This shows that Algorithm 4 maintains the same solution feasibility as the existing drift-plus-penalty algorithm [159]. Meanwhile, as we will shown below, the action of the misloading calibration can enhance Algorithm 4 with a better objective convergence than the vanilla drift-plus-penalty algorithm.

Theorem 1*:* Let $\bar{f}\big(\{\hat{\pi}_-^t\}_{t=0}^{t-1}\big)$ be the achieved objective value under the online control sequence $\{\hat{\pi}_-^t\}_{t=0}^{t-1}$ by recursively solving program (4.26), and $\bar{f}^*$ be the optimum of **T3S-R** obtained by some '*genius*' decision maker who holds a complete knowledge about the true system trajectory $\{\omega_t\}_{t=0}^{t-1}$. Then, we have:

$$\bar{f}\big(\{\hat{\pi}_-^t\}_{t=0}^{t-1}\big) \leq \bar{f}^* + \frac{1}{V}\big(B - \frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{k=1}^{K}\mathbb{E}[Q_k^\tau]\delta_k\big) \tag{4.33}$$

where B is a positive constant that upper bounds the second moments of the "arrival" and "service" processes of Q_k^t as follows:

$$\frac{1}{2}\sum_{k=1}^{K}\mathbb{E}\big[\big(g_k(\pi_-^t, \omega_t) - (\bar{c}_k + \delta_k^t)\big)^2\big] \leq B \tag{4.34}$$

Proof. This can be proved by extending the result in [159] with the misloading calibration scheme. Based on the objective function analysis in [159], we can bound the weighted-sum expression in (4.26) under the new queue update function in (4.28) as follows:

$$\mathbb{E}[\Delta_\tau] + V\mathbb{E}[f(\pi_-^\tau, \omega_\tau)] \leq B + V\bar{f}^* - \sum_{k=1}^{K} \mathbb{E}[Q_k^\tau]\delta_k \tag{4.35}$$

where $\Delta_\tau = L_{\tau+1} - L_\tau$, called the *Lyapunov drift*.

Summing (4.35) over the first t slots gives:

$$\mathbb{E}[L_t] - \mathbb{E}[L_0] + V\sum_{\tau=0}^{t-1} \mathbb{E}[f(\pi_-^\tau, \omega_\tau)] \leq (B + V\bar{f}^*)t - \sum_{\tau=0}^{t-1}\sum_{k=1}^{K} \mathbb{E}[Q_k^\tau]\delta_k \tag{4.36}$$

Dividing the above by Vt and using the fact that $\mathbb{E}[L_0] = 0, \mathbb{E}[L_t] \geq 0$, we have:

$$\bar{f}\big(\{\hat{\pi}_-^t\}_{t=0}^{t-1}\big) = \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}[f(\hat{\pi}_-^\tau, \omega_\tau)] \tag{4.37}$$

$$\overset{\textbf{(a)}}{\leq} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}[f(\pi_-^\tau, \omega_\tau)] \tag{4.38}$$

$$\leq \bar{f}^* + \frac{1}{V}\big(B - \frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{k=1}^{K} \mathbb{E}[Q_k^\tau]\delta_k\big) \tag{4.39}$$

where **(a)** follows because $\hat{\pi}_-^\tau$ is the optimal solution of program (4.26). *Proof* ends. ■

With the fact that $\mathbb{E}[Q_k^\tau] \geq 0$ and $\delta_k \geq 0$, (4.33) shows that *Theorem* 1 always achieves a tighter running performance bound than the vanilla drift-plus-penalty algorithm. This particularly improves the performance when Q_k^τ spurts due to a sudden improvement of the system state when the system is under-loaded. This stands to the reason that it is less likely to break the time-averaged constraints in this case. Then, the action of misloading calibration provides a pullback on Q_k^t to avoid the unnecessary penalty on $f(\pi_-^t, \omega_t)$ in (4.26), which contributes the extra performance here.

4.7 Performance Evaluation and Analysis

In this section, we use synthetic scenarios to evaluate the proposed solutions. Current BT's IP network topology within Europe[3] is considered as the physical network, from which arbitrary node pair is chosen as the source and destination of each network slice request. 5 nodes

[3]http://www.topology-zoo.org/dataset.html

in the whole 21 nodes are randomly selected to act as the wireless access nodes. For each node in the network, a fixed number of computing resources is considered. Nevertheless, the available wireless transmission capacity within each access node is time-variant. Considering the long-term co-existence of 5G and legacy networks (3G/4G, *etc*.) and the tremendous changes among Line-of-Sight (LOS), non-LOS (NLOS) and outage stages in 5G wireless channels [113], we use the Rician fading and Rayleigh fading[4] [148] to emulate the adopted wireless networking environment. For each fading status, a fixed duration T_Δ is set, and the transition probabilities between statuses are equal. The channel parameters are configured so that the resulted capacity of each access node is on average within the envisioned capacity range for a 5G cell [109].

4.7.1 A Benchmarking Problem

As a benchmark, we extend the deterministic network slicing model in [142] to maximize the time-averaged system revenue while safeguarding the time-averaged link resource cost not exceeding a given budget. For clarity, the involved objective[5] and constraint functions are clearly defined as follows:

$$f(\cdot) = -\sum_{s\in S}\Big(\pi_-^{st}(b_s - \overbrace{\sum_{l\in\mathscr{L}^s,e\in\mathscr{E}}\pi_+^{le}k_e}^{\text{dynamic link cost}}) + \overbrace{\sum_{f\in\mathscr{F}^s,n\in\mathscr{N}}\pi_+^{fn}d_f k_n}^{\text{fixed node cost}}\Big) \tag{4.40}$$

$$g(\cdot) = \sum_{s\in S,l\in\mathscr{L}^s,e\in\mathscr{E}}\pi_-^{st}\pi_+^{le}k_e \tag{4.41}$$

$$u_l(\cdot): \begin{cases} \pi_-^{st} \le \pi_+^s d_s, \forall s\in S & (4.42)\\ \sum_{l\in\mathscr{L}^s,s\in S}\pi_+^{le}\pi_-^{st} \le c_e, \forall e\in\mathscr{E} & (4.43)\\ \sum_{s\in\{S|Sr(s)=n_a\}}\pi_-^{st} \le c_{n_at}, \forall n_a\in N_{access} & (4.44)\end{cases}$$

and the solution space for a valid slice deployment policy $\pi_+\in P_+$ is defined as follows:

$$P_+: \begin{cases} \sum_{f\in\mathscr{F}^s,s\in\mathscr{S}}\pi_+^{fn}d_f \le c_n, \forall n\in\mathscr{N} & (4.45)\\ \sum_{n\in\mathscr{N}}\pi_+^{fn} = \pi_+^s, \forall f\in\mathscr{F}^s, s\in S & (4.46)\\ \sum_{e_{uv}\in O(u)}\pi_+^{l_{ij}e_{uv}} - \sum_{e_{vu}\in I(u)}\pi_+^{l_{ij}e_{vu}} = \pi_+^{iu} - \pi_+^{ju}, \forall l_{ij}\in\mathscr{L}^s, s\in S, u\in\mathscr{N} & (4.47)\end{cases}$$

[4]But the proposed solution is not limited to any specific type of dynamics.

[5]Take minus to change to a minimization problem as defined in our models.

where

- $\{\mathscr{N},\mathscr{E}\}$ are the node and directed link sets in the physical network, respectively;
- $\{\mathscr{L}^s,\mathscr{F}^s\}$ are the virtual link and VNF sets for slice request $s \in S$, respectively;
- $\pi_+ = \{\pi_+^s, \pi_+^{le}, \pi_+^{fn}\}, \forall s,l,e,f,n$ are binary variables that decide whether slice request $s \in S$ should be accepted, whether physical link e should be used to construct the virtual link l, and whether VNF f should be installed in physical node n, respectively;
- $\pi_- = \{\pi_-^{st}\}, \forall s,t$ are variables that decide the running flow rates allocated to s at t;
- $\{Sr(s), N_{access}\}$ are the source node of s and access node set, respectively;
- $\{c_n, c_e, c_{n_{at}}\}$ are the capacities of node, link and access resources, respectively;
- $\{b_s, k_e, k_n\}$ are the service price per unit rate and prices for using per unit physical link and node resources, respectively;
- $\{d_s, d_f\}$ are the demanded service rate of s, and the required computing resources to instantiate f, respectively;
- $\{O(u), I(u)\}$ are the outgoing and incidental links of physical node u, respectively;
- $\{e_{uv}, l_{ij}\}$ are the physical link connecting node v from node u and virtual link connecting VNF j from VNF i, respectively.

In this example, we aim to maximize the time-averaged system revenue $\bar{f}$ under a given time-averaged link cost budget $\bar{c}$ (*i.e.*, only one constraint, $\bar{g}\big(\pi_+, \{\pi_-^t\}_{t=0}^{\infty}\big) \leq \bar{c}$, for (4.7)). Specifically, Constraints (4.42) and (4.46) enforce the admission control on correlated variables; (4.43) – (4.45) bound the capacities of corresponding link and node resources; (4.47) expresses the single-path flow conservation [127].

The proposed solution is compared with three existing reference algorithms, Current-Greedy (CG) [108] and Prediction Average Approximation (PAA) and Learning Augmented Optimization (LAO)+CG. In CG, slicing decisions are made only to optimize instant system revenue while guaranteeing that the resource cost constraint is not violated under the current network state. In PAA, however, predictions of network states in the next W_t time window are used to approximate the deployment policy in **T4S**. The slice running control policies in PAA are then constructed to best respond to the predicted mean states. In contrast, in our proposed approach (LAO), the slice deployment policy is first derived through historical learning. Thereon, the deployed slices are adaptively operated with the *Lyapunov* stability and the

misloading calibration scheme. In light of the separate efficacy of the slice deployment and running control policies in LAO, we further exam the combinations of the proposed slice deployment policy with CG based slice running control policy (for clarity, this is called LAO+CG).

In the following experiments, upfront observation trajectories were first generated according to the corresponding distributions. For simplyfication, we directly compared the performance of PAA under the approximation with W_t error-free predictions, which is an upper bound that PAA can achieve if a real prediction process is imposed. The historical learning in LAO was implemented with a random sampling strategy, *i.e.*, directly sampling at random from these upfront trajectories. All compared algorithms were solved under a same greedy solver, which greedily decides the slice placement policy π_+^s for each s through Benders' decomposition algorithm [125].

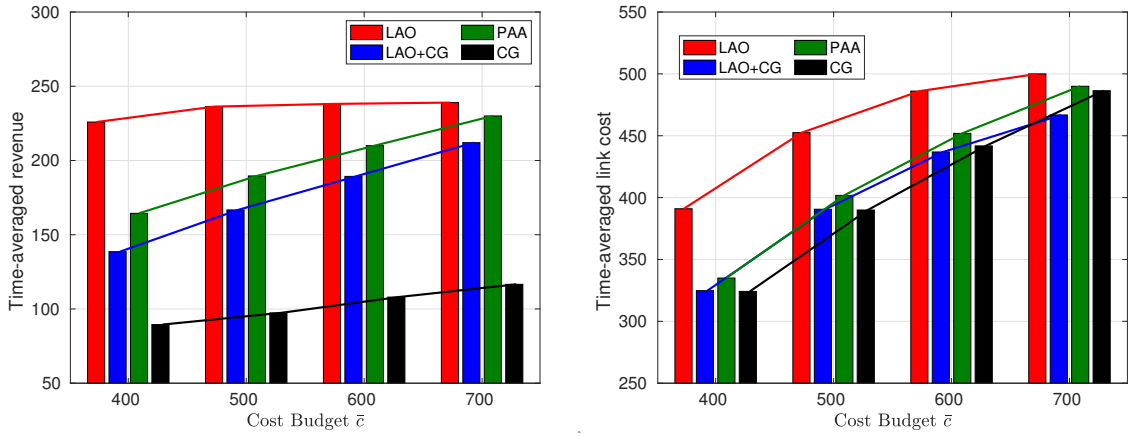
All algorithms are measured with the following five performance metrics: (1) time-averaged system revenue, (2) time-averaged system throughput, (3) time-averaged link cost, (4) number of accepted slice requests, and (5) time-averaged outage ratio of deployed services. A service is considered outage if the real-time service rate is below a certain threshold (set as $0.2d_s$ in experiments). The simulation parameters are shown in Table 4.1.

Table 4.1 Simulation Setup for Chapter 4

Parameters	**Value**
$c_n, n \in \mathcal{N}$	Fixed with an initial value uniformly distributed within [5,10]
$c_e, e \in \mathcal{E}$	10*Gbps*
$[k_n, k_e, b_s]$	[10, 20/*Gbps*, 100/*Gbps*]
# of VNFs $\lvert\mathcal{F}^s\rvert$	3
Rate demand d_s	1–3 *Gbps*, uniformly distributed
Node resource demand d_f	1–3, uniformly distributed
Radio bandwidth	1 *GHz*
Rician factors	1*dB*
Power ratio of signal to noise plus interference	Rician: 31.3 *dB*; Rayleigh: 12.8 *dB*
Channel duration T_Δ	10
Scheduling interval T	20

4.7.2 Time-Averaged System Performance

We first test the time-averaged performance of compared algorithms when the system is converged. In the following experiments, we apply Poison arrival process with the arrive rate



(a) (b)

Fig. 4.4 Performance comparisons with $V = 50, T_s = 40, W_t = 40$: a) Time-averaged system revenue, b) Time-averaged link cost.

= 20 per scheduling interval and average service lifespan $T_s = 40$ for the received network slice requests.

In Fig. 4.4, we plot the compared time-averaged objective value (*i.e.*, system revenue) as a function of cost budget $\bar{c}$. We observe that the proposed slicing solution achieves an obvious system performance improvement over the compared algorithms in all cases. Moreover, under all cases, the converged time-averaged cost constraint is well preserved. Such improvement manifests particularly when the system is greatly limited by the cost budget constraint. For the best case, say $\bar{c} = 400$ in Fig. 4.4a, LAO gains even up to $2.6\times$, $1.63\times$ and $1.37\times$ better performance than CG, LAO+CG and PAA, respectively. This can be interpreted as follows:

For CG, the slicing policies only best respond to the instant network states at the decision time. As a consequence, any over-optimistic decision will lead to high resource cost (node cost) for maintaining the over-loaded slices. Likewise, any over-pessimistic decision will not be able to make full use of the available cost budget to improve the system performance. Benefited from the approximation with the partial future predictions, these situations are slightly alleviated in PAA. However, the predicted mean state information still fails to capture the long-term system evolution and then adapt itself efficiently. By contrast, with full respect to large history samples and learned dynamics, the proposed slicing solution well matches the system dynamics, which finally contributes the significant system improvements.

Fig. 4.4 also shows that the performance of the compared three algorithms keeps quicker growing than LAO as $\bar{c}$ increases. This indicates that the cost budget plays more critical role than the underlying available physical resources for the compared ones. When $\bar{c}$ is large

enough, other factors (*e.g.*, wireless access resources) will become the key scarce resources for the system to further improve.

4.7.3 Solution Feasibility and Convergence Analysis

Fig. 4.5 presents the real-time average performance. The achieved numerical results in Fig. 4.5c show that all the compared algorithms meet the required time-averaged resource cost constraint throughout the running. However, this is achieved in CG and PAA by restricting the real-time resource cost at each running time slot according to the fixed cost budget and predicted mean state information, respectively. These drawbacks lead to very low utilization of the available cost budget. However, combined with the virtual queue based slice running policy in the proposed solution, the practical running link cost is below but very close to the given cost budget line. This confirms the solution feasibility of the proposed slicing policies, as analyzed in *Lemma* 1. On the other hand, we notice from both Fig. 4.4b and Fig. 4.5c that the available cost budget is always under-exploited for all compared algorithms. This is resulted from the suboptimality of the solution to the **T4S** program. When an approximate solver is applied during solving the **T4S** model, only solutions preserving the time-averaged cost constraint will be returned. Depending on the suboptimality gap of the applied approximate solver, the estimated cost is always inevitably lower than the given cost budget by a certain value.

Under a feasible slicing solution, Fig. 4.5a, Fig. 4.5b and Fig. 4.5d confirm that the benefits of network providers and the provisioned service qualities are all superior in LAO when compared with the other three. In Fig. 4.5e, the serrated changes on the number of running services across t are the results of the scheduling towards new slice requests and the service expiration of deployed slices. Fig. 4.5e shows that all compared algorithms nearly maintains the same level of service loads when the overloading and underloading cases are averaged. However, when combined with other performance metrics, we can see that a similar loading level in LAO contributes up to $2.6\times$ better revenue than CG. This indicates that LAO coordinates the available resources for the competing requests in a more efficient manner. The degradation of the compared algorithms results from both the misloading during the slice deployment phase and the inefficient utilization of cost budget during the slice running control.

4.7.4 Parameter Effectiveness

We now test the solution effectiveness of our proposal under different parameter settings. Fig. 4.6 plots the running time-averaged performances under different V. It shows that

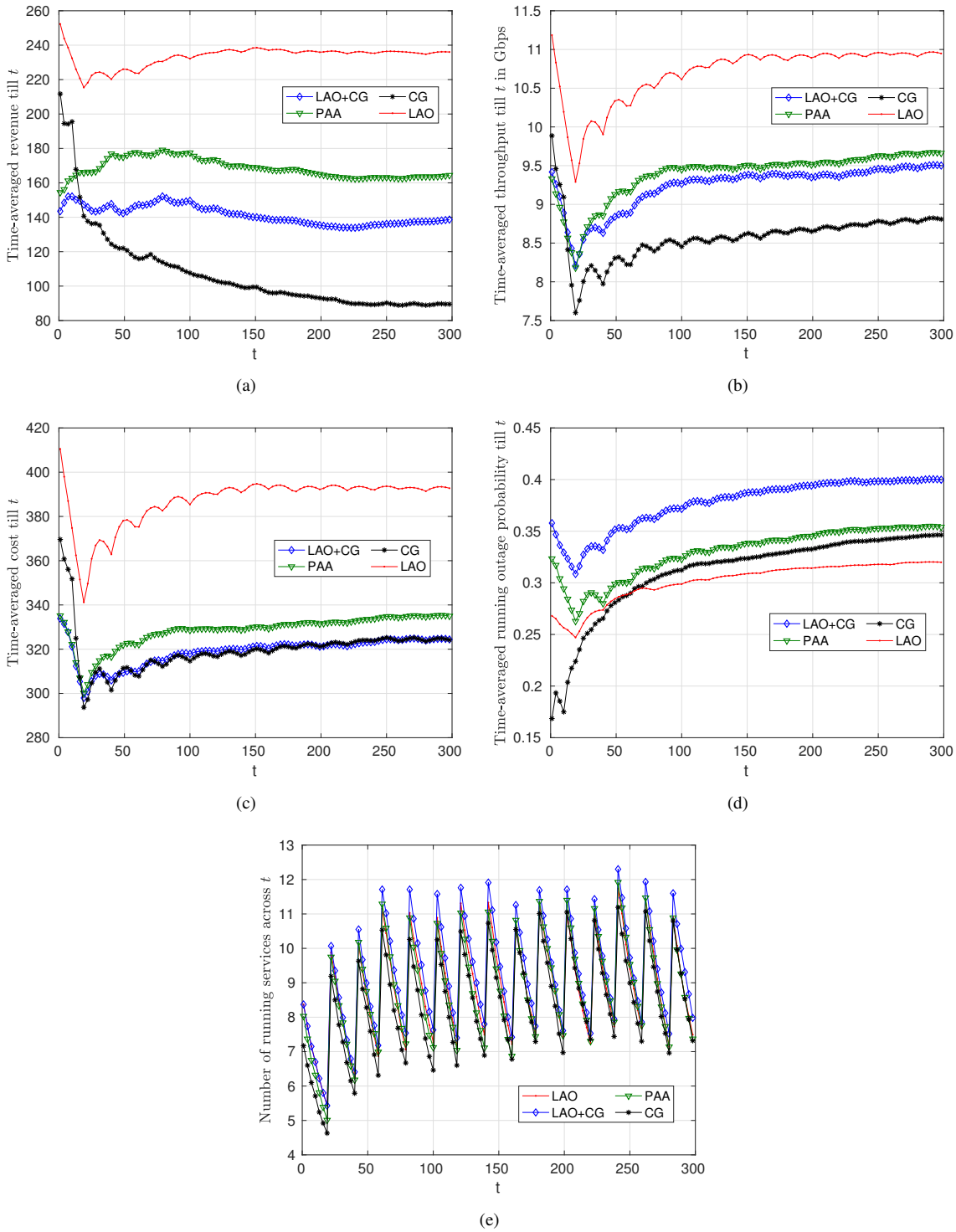


Fig. 4.5 Performance comparisons with $V = 50, T_s = 40, W_t = 40, \bar{c} = 400$: a) Time-averaged system revenue till t, b) Time-averaged system throughput till t, c) Time-averaged link cost till t, d) Time-averaged running outage probability till t, e) Number of running services across t.

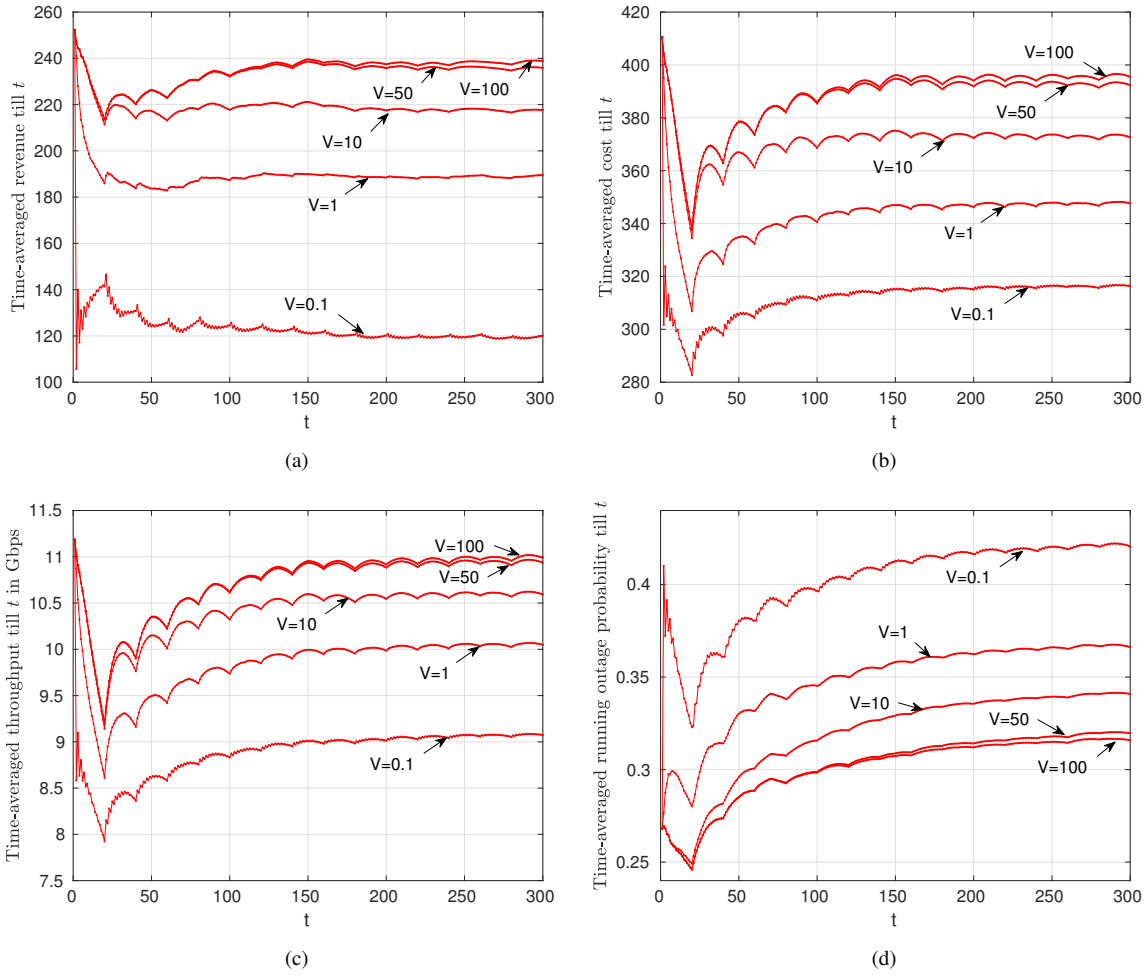


Fig. 4.6 Performance comparisons of LAO under different V with $\bar{c} = 400, W_t = 40, T_s = 40$: a) Time-averaged revenue till t, b) Time-averaged link cost till t, c) Timed-average throughput till t, d) Time-averaged running outage probability till t.

the solution superiority of our proposal is preserved under all values of V. Moreover, all the resulted time-average system metrics keep sustainable improvement as V goes larger, although the improvement gradually vanishes when V reaches a certain value.

Also revealed in Fig. 4.6, is the property that the running time-averaged system performance converges more quickly when controlled by a smaller V. These results are consistent with the theoretical analysis in Section 4.6.2 and [159]. The choice of a '*good*' V depends on the balancing requirements between the convergence speed and desired system performance, which is subject to practical applications.

In Fig. 4.6, we can also observe that there is always a descending slop during the start-up time along t axis. This is caused by the service expiration and tear-down of the deployed services with lifespan less than scheduling interval (*i.e.*, $T_s \leq T$). After a slow warm-up, more active services with lifespan $T_s > T$ will be accumulated and keep active in the following scheduling intervals. Since these services, once deployed, will have impacts throughout their lifespans, the service lifespan impose a direct influence on the decision accuracy of the obtained slicing policies. In next section, we provide a more detailed analysis on the efficacy of different service lifespans.

4.7.5 Efficacy of Different Service Lifespan

Intuitively, if the deployed services only possess a short lifespan, say $T_s < T$, the system will get under-loaded during the remaining running time of a scheduling interval. In contrast, if $T_s \gg T$, say $T_s = \infty$, the system will keep loading new requests until the system is saturated during the peak networking status. This will lead to overwhelming system overloading. To illustrate this, we examined three different lifespan settings $T_s = \{20, 50, 100\}$. The performance comparisons are shown in Fig. 4.7.

When services with short lifespans are largely loaded as the case of $T_s = 20$, Fig. 4.7 discloses that the system always maintains very low number of running services, resulting in the underutilization of available resources. Inversely, if it comes to the case of $T_s = 100$, more loaded services only lead to noticeable build-up of resource cost and service competition. This deteriorates both the benefit of network providers and provisioned service quality. By contrast, $T_s = 50$ achieves the best control towards the misloading problem. In practical networking environment, the scheduling interval T is fine-tuned according to the features of the supported services. Also, different network slices with policies and system settings dedicated to the service features can be created. This is exactly one of the critical envision when network slicing is advocated.

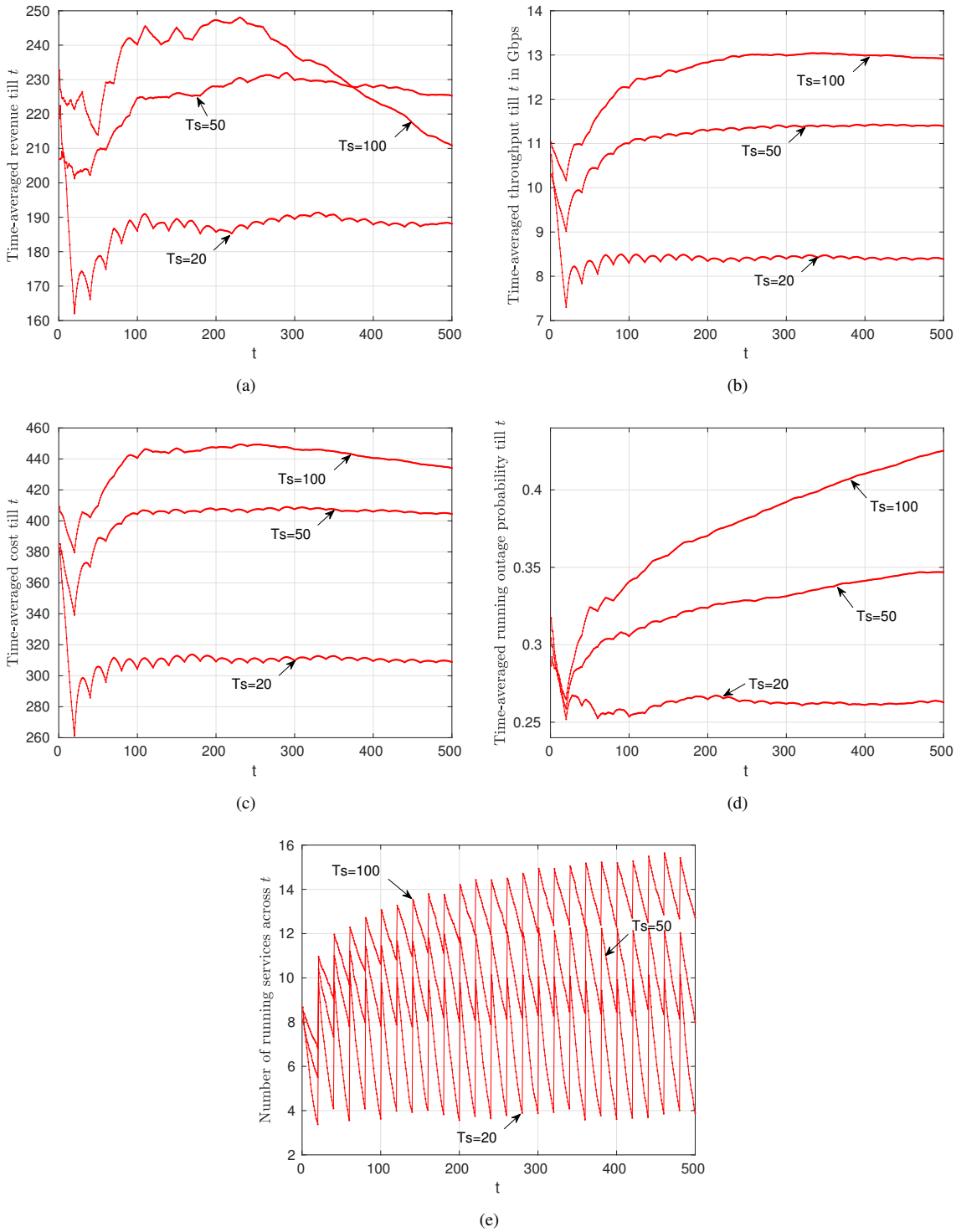


Fig. 4.7 Performance comparisons of LAO under different T_s with $V = 50, T = 40, W_t = 40, \bar{c} = 400$: a) Time-averaged system revenue till t, b) Time-averaged system throughput till t, c) Time-averaged link cost till t, d) Time-averaged running outage probability till t, e) Number of running services across t.

4.8 Conclusion

This chapter has highlighted the performance safety problem of slicing operations when the environment is time-varying and difficult to track with explicit models due to its complexity and heterogeneity. Based on the stochastic historical learning and online optimization, we have accordingly developed a learning augmented online optimization approach to learn a safe slicing solution from historical records and real-time observations. We have proved that the feasibility of proposed solution with a sub-optimality, up to a constant additive factor. Finally, we have demonstrated up to $2.6\times$ improvement in the simulation when compared with the referenced algorithms. This work is a good start to stimulate the further researches on the innovative use of learning augmented optimization approaches for more dynamic and stochastic networking problems.

Chapter 5

Deep Learning Solves the Optimization of Service Function Chaining in 5G

The solutions presented in the previous chapters are basically model based solutions supported with solid optimization/control theories. They are more dedicated to specific problems and often require some assumptions and strong domain expertise. Additionally, the success of these solutions to optimize the stochastic systems requires solving a base optimization program repeatedly until convergence. The optimality loss and computation cost of the final solutions are strongly dependent on the solving efficiency to the base optimization program. However, for each iteration, the base program exhibits the same model structure, but only differing in the input data. This makes it possible to apply the latest graph neural network to abstract the core structures of an optimization model and then use the learned deep learning model to directly generate the solutions to the equivalent optimization model. In this chapter, we will present a new deep learning based solution to represent and solve the targeted optimization problems in this thesis. The work of this chapter is able to bridge the two distinct technologies (i.e., deep learning and optimization) and facilitate their joint application in the field of communication and networks.

5.1 Introduction

The rapid uptake of mobile devices and the rising popularity of mobile applications and services pose unprecedented demands on mobile and wireless networking infrastructure. Upcoming 5G systems are evolving to support exploding mobile traffic volumes, real-time extraction of fine-grained analytics, and agile management of network resources, so as to maximize user experience [95]. Fulfilling these tasks is challenging, as nowadays networking

environments are increasingly complex, heterogeneous, and evolving. One potential solution is to resort to advanced Machine Learning (ML) techniques, in order to help manage the rise in data volumes and data-driven applications. ML enables systematic mining of valuable information from traffic data and automatically uncover correlations that would otherwise have been too complex to extract by human experts [160]. The recent success of deep learning underpins new and powerful tools that tackle problems in this space. As the flagship of ML, deep learning has achieved remarkable performance in areas such as computer vision and natural language processing (NLP) [161]. Networking researchers are also beginning to recognize the power and importance of deep learning, and are exploring its potential to solve problems specific to the networking domain [96], [162].

Traditionally, many networking control problems have been solved by constrained optimization, dynamic programming and game theory approaches. Unfortunately, these methods either make strong assumptions about the objective functions (e.g. function convexity) or data distribution (e.g. Gaussian or Poisson distributed), or suffer from high time and space complexity. As the networks become increasingly complex, such assumptions sometimes turn unrealistic.

Additionally, as a typical application of the graph optimization, networking optimization problems (especially when the problem is combinatorially NP-hard) have been traditionally tackled by three main approaches [163]: exact algorithms, approximation algorithms and heuristics. Exact algorithms are based on enumeration or branch-and-bound with an integer programming formulation, but may be prohibitive for large instances. On the other hand, polynomial-time approximation algorithms are desirable, but may suffer from weak optimality guarantees or empirical performance, or may not even exist for inapproximable problems. Heuristics are often fast, effective algorithms that lack theoretical guarantees, and may also require substantial problem-specific research and trial-and-error designing.

As claimed in [163], all three paradigms seldom exploit a common trait of real-world optimization problems: instances of the same type of problems are solved again and again on a regular basis, maintaining the same combinatorial structure, but differing mainly in their data. That is, in many applications, values of the coefficients in the objective function or constraints can be thought of as being sampled from the same underlying distribution. For instance, a package delivery company routes trucks on a daily basis in a given city. Thousands of similar optimizations need to be solved, since the underlying demand locations can differ.

In contrast, ML methods have the potential to be applicable across many optimization tasks by automatically discovering their own heuristics based on the training data, thus requiring less hand-engineering than solvers that are optimized for one task or one specific

problem instance only. Recently, there has been some seminal work on using deep learning architectures to learn optimization solutions for combinatorial problems, including the Travelling Salesman Problem (TSP) [98], routing problems [164] etc. However, the architectures used in these works are generic and tested with only some toy problems.

In this chapter, we aim to address the challenging placement tasks of Service Function Chain (SFC) during network softwarization by applying latest deep learning techniques to learn a combinatorial solution directly from the historical data. The deep learning model is constructed based on the latest graph attention mechanism and encoder-decoder architecture. In order to facilitate the solving of the constrained optimization program for the targeted network slicing problem in a deep learning manner, we design a problem-specific decoding process by integrating program constraints and problem context information. The deep learning model, once trained, can be used to directly generate the solution to any specific problem instance through only some simple linear operations. This avoids the extensive computation in traditional approaches of re-solving the whole combinatorial optimization problem from the scratch and is promising to facilitate a wider technology penetration of network softwarization in different applications. The contributions of this chapter can be summarized as follows:

- Following the latest Transformer architecture, we construct a problem-specific attention based graph encoder to learn the network topology and state information for the network slice placement problem targeted in this chapter.
- A context-aware decoder is designed to generate the desire combinatorial solution directly for the constrained optimization problem in network slicing. The constraints are fully preserved by a specific masking mechanism, which are then integrated into a context node to aid the decision of the combinatorial solution at each step.
- We train the new encoder-decoder model with the REINFORCE gradient estimator to solve a realistic benchmarking problem for network slicing and verify the superiority of the proposed approach through simulation.

5.2 Related Work

Inspired by the great success of deep learning in other domains, there exists a growing number of recent papers that attempt to bring deep learning into the computer networking domain. Alsheikh et al. identified the benefits and challenges of using big data for mobile analytics and proposed a Spark based deep learning framework for this purpose [165]. Wang et al. presented a traffic-driven deep learning solution to infer the network topology online

for the problem of network topology configuration optimization. Graph CNN and LSTM were leveraged in [166] to predict dynamic macroscopic traffic congestion. More recently, Fadlullah et al. delivered a survey on the progress of deep learning in a board range of areas, highlighting its potential application to network traffic control systems [92]. Zhang et al. bridged the gap between deep learning and mobile and wireless networking research, by presenting a comprehensive survey of the crossovers between the two areas [95]. In general, most of these preliminary researches use deep learning tools to extract additional information (e.g. pattern recognition or network state prediction), which are then used to aid the decisions of network optimization and controls. This presents great difference with our work, in which we aim to use deep learning techniques to directly solve constrained combinatorial optimization problems for networking applications.

The application of Neural Networks (NNs) for optimizing decisions in combinatorial optimization problems can be dated back to Hopfield and Tank [167], who applied a Hopfield-network for solving small TSP instances. NNs have been applied to many related problems [168], although in most cases solutions are learned separately for every instance. More recently, there is also an increasing attention among academia paid to directly learn a solution for combinatorial optimization problems. In this respect, the majority of reported researches only present the proof-of-concept results on some classic combinatorial optimization problems, including TSP, vehicle routing problem, knapsack problem, and etc. These pioneering work, although still quite tutorial and only validated on some *toy* problems, open a promising new direction for solving diverse practical combinatorial optimization problems.

Specifically, the first attempt was proposed by Vinyals et al. in [98], who introduced the Pointer Network (PN) as a model that used attention to output a permutation of the input, and trained this model offline in a supervised manner to solve the TSP. Bello et al. in [169] introduced an Actor-Critic algorithm to train the PN without supervised solutions. They considered each instance as a training sample and used the cost (tour length) of a sampled solution for an unbiased Monte-Carlo estimate of the policy gradient. They introduced extra model depth in the decoder by an additional glimpse [170] at the embeddings, masking nodes already visited. For both papers, only small problem instances are tested and the reported results only show margin gains when compared with traditional optimization solvers. Nazari et al. in [164] replaced the LSTM encoder of the PN by element-wise projections, such that the updated embeddings after state-changes can be effectively computed. They applied this model on the Vehicle Routing Problem (VRP) with split deliveries and a stochastic variant.

Apart from the applications of separate encoder and decoder in above researches, the authors in [163] presented a single model based on graph embeddings. They trained the model to output the order in which nodes were inserted into a partial tour, using a helper function

to insert at the best possible location. Their 1-step DQN [171] training method trained the algorithm per step and incremental rewards provided to the agent at every step effectively encouraged a greedy behavior. Nowak et al. in [172] trained a Graph Neural Network (GNN) in a supervised manner to directly output a tour as an adjacency matrix, which was converted into a feasible solution by a beam search. The model is non-autoregressive, so cannot condition its output on the partial tour. Kaempfer and Wolf in [173] trained a model based on the Transformer architecture [174] that outputs a fractional solution to the multiple TSP (mTSP). The result can be seen as a solution to the linear relaxation of the problem and they use a beam search to obtain a feasible integer solution. More recently, closest to our work, the authors in [175] presented a general attention based model to learn solutions for the diverse routing problems. They integrated decoding state information into a context node and then updated the context node to generate sequential outputs. However, the encoder-decoder structure is tied to the given routing problems, which limits its applications in general combinatorial optimization problems. Therefore, a new learning model with problem-specific structure is still absent for the placement problem of SFC.

The direct application of deep learning for VNF related networking problems is still in its infancy. Among the very few work, Li et al. in [176] investigated the application of Deep Reinforcement Learning (DRL) in the resource management of radio resource slicing and priority-based core network slicing, and demonstrated the advantage of DRL over existing competing schemes. Similarly, in [177], a DRL solution based on deep Q-learning was presented for multi-tenant cross-slice resource orchestration, where a discrete number of communication and computation resources have to be allocated to different slice tenants. In [178], a deep deterministic policy gradient (DDPG) method with advantage function was employed to allocate bandwidth resources to different network slices. Similar DRL solution was also presented in [179], in which the authors leveraged an actor-critic architecture to learn to provision resources to the VNFs in an online manner. The testaments in the abobe work have demonstrated the promissing efficacy of DRL for the tasks of resource management/allocation, which, however, are inherently different from the SFC placement problem in this chapter. Moreover, the authors in [180] proposed a Graph Neural Network (GNN)-based algorithm, which exploits the VNF forwarding graph topology information to predict future resources requirements for each VNF component. Similarly, Artificial neural network was applied in [181] to predict affinity of VNFs. Both of the above researches are the applications of the deep learning techniques for the prediction tasks, which do not directly output network control actions. Instead, the extracted additional prediction information in their work are used to aid the decisions of subsequent network controls. To

the best knowledge of the author, this work is the first application of the concept of neural combinatorial optimization for the SFC placement problem.

5.3 Attention Model

Without loss of generality, we abstract the optimization problem for the network slicing as the following generic combinatorial program:

$$\begin{cases} \min \ \ f(\pi) \\ s.t. \\ \quad g_k(\pi) \leq c_k, k = 1,2,\ldots,K \\ \quad \pi \text{ are integers} \end{cases} \tag{5.1}$$

where f denotes the objective function, g is a set of the specific resource and service utilities for constructing network slices under a capacitated physical network, and π is a vector of decision variables indicating which physical nodes and link paths are used to construct each network slice in the final slicing strategy.

In the previous chapters, we have solved this problem/variants through several problem-specific approaches. As we discussed before, each attempt of solving this problem in a large dimension is time-consuming and computationally expensive. Moreover, any change to the coefficients of objective or constraint functions will lead to the re-solving of the program from the scratch. Next, we resort to the latest deep learning technologies to learn the solution directly from the input data and without revoking any traditional optimization solvers. This is achieved by an encoder-decoder architecture.

5.3.1 Encoder

The topology of network infrastructures presents a typical graph structure, in which the traffic flows through a source-destination routing path with a set of nodes and connection links in the graph. Different with the popular data structures (time-series data, image/video) in the dominant deep learning applications (e.g., NLP, autonomous driving), the graph-structured data in networking domains presents an irregular distribution pattern and strong dependence among nodes and links, which make it difficult to represent and learn through a deep learning model. In order to efficiently exploit the structure patterns and represent the states of a network graph, a graph attention neural network [182] will be applied in this work. Specifically, both the individual node features (e.g., coordination, available resources) and topological features (e.g., neighbouring and connection relationship) will be embedded

to form the initial states. Then, a Multi-Head Attention (MHA) mechanism [174] will be applied to pass the initial state information along the connection topology so that the final state of each node can represent the aggregated information from all other connected nodes. Finally, the encoded graph state information will be fed to a problem-specific decoder to generate the final optimization solution.

The encoder that we use is similar to the one in the attention routing problem [175]. But we provide the problem-specific input features and replace the fully connected network topology with a realistic directional-connected network topology. Thus, the attention weights are updated according to the connection relationship in the graph structure. Specifically, the encoder processes the input in following manner:

1. Initial embedding for input features x_i:

$$h_i^{(0)} = W^x x_i + b^x \tag{5.2}$$

   In contrast to the simple input feature (i.e., node coordinates) for TSP problem in [175], the input features for the target problem exhibit a more complicated structure. The placement of VNFs requires the information about the resource capacities of each node in the graph, since this is directly connected with the placement decisions. Moreover, information about node locations is required to chain the placed VNFs. Consequently, in the SFC placement problem, the input features in x_i consist of both the node coordinates and resource capacities. The input dimension is $d_x = 2 + N_r$, where N_r is the number of resource types in each node. The output dimension of $h_i^{(0)}$ is d_h.

2. N-layer attention for $h_i^{(0)}$:
   The initial embeddings are updated using N sequential attention layers, each consisting of two sublayers, which follows the Transformer architecture. Specificially, the first attention sublayer computes a MHA that executes message passing between the nodes with direct connections in the graph. Then, the second sublayer imposes a node-wise fully connected feed-forward operation on the results of the multi-head attention. Let us denote $h_i^{(l)}$ as the embedding of node $i \in \{1, \cdots, \mathcal{N}\}$ produced by layer $l \in \{1, \cdots, N\}$. Then, the attention procedure can be formally described as the following formulas:

$$\text{First sublayer :} \qquad \hat{h}_i = BN^l\left(h_i^{l-1} + MHA_i^l\left(h_1^{l-1}, \cdots, h_n^{l-1}\right)\right) \tag{5.3}$$

$$\text{Second sublayer :} \qquad h_i^{(l)} = BN^l\left(\hat{h}_i + FF^l(\hat{h}_i)\right) \tag{5.4}$$

   where a skip-connection [183] and batch normalization (BN) [184] are added in each sublayer. These components help accelerate the training and also allows better model

generality. The details of attention mechanism can be reviewed by referring to work e.g. [174], [175].

3. Encoder masking:
   To respect the connection structure in the graph, two different masking schemes are introduced during computing the MHA process. Specifically, for the VNF placement problem, although this is irrelevant to the practical topology structure, the selection of each node to place a VNF is decided by the status of all other nodes. Therefore, a fully connected network topology will be applied to calculate the message passing between nodes. However, for the chaining problem, non-adjacent nodes in the practical network graph will be masked during computing the attended node states.

4. Compute the graph embedding by aggregating the final node embedding results $h_i^{(N)}$:

$$\bar{h}^{(N)} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} h_i^{(N)} \tag{5.5}$$

Both the node embeddings $h_i^{(N)}, i = \{1, \cdots, \mathcal{N}\}$ and the graph embedding $\bar{h}^{(N)}$ are used as input to the decoder. Fig. 5.1 depicts the structure of the applied encoder.

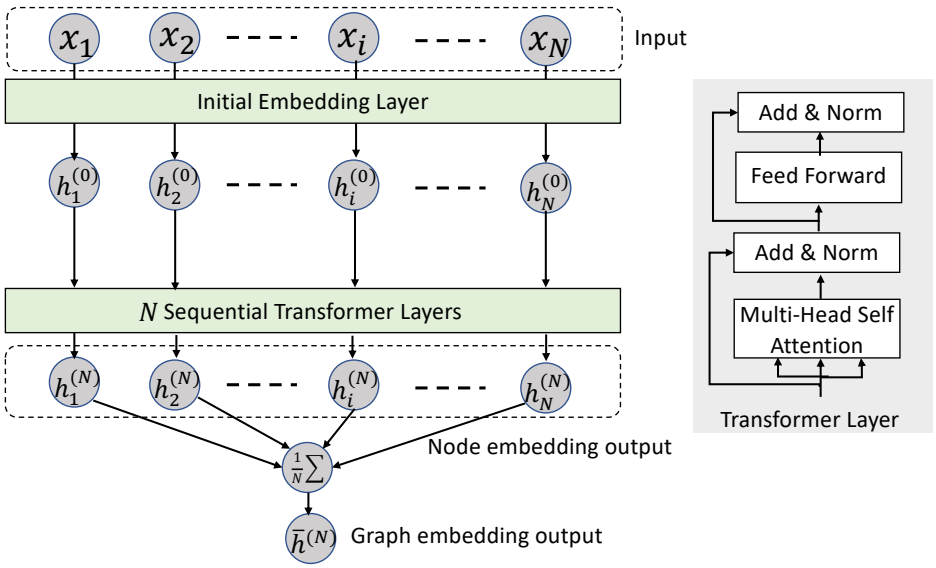


Fig. 5.1 Attention based graph encoder.

5.3.2 Decoder for VNF Placement

The learning models (normally manifested with a neural network structure in deep learning) and classic optimization models (composed by a set of constraint and objective functions) work in quite different ways. As two distinct technologies to solve a constrained optimization problem, we need to guarantee an equivalent representation of the original constrained optimization model with the deep learning model. However, the huge searching space of the original problem makes it very challenging to decide the training output at each iteration. Considering the possibly dependent decision structures in the problem, we will transform the solving process into a problem-specific sequential decoding process by integrating program constraints and other context information to the training process.

In general, we aim to use the learned graph state information from the Encoder to calculate the output probability of selecting each node/link at each decoding step. To facilitate this process, we will construct the context node with the encoded graph state information and output information in previous steps so that the remaining selections can be updated accordingly based on both the graph states and previous sequential outputs. Moreover, masking scheme will be designed to exclude the selections (thus reduce the feasible selection space) that do not meet all the constraints under existing output sequence. This guarantees that the learned outcomes during both the training and test stages respect all the constraints and objectives in the optimization model.

Specifically, the VNF placement problem here aims to find a combination of the physical nodes to host the required VNFs, which is subject to a set of constraints e.g. resource capacity constraints etc. Clearly, the decision of placing any VNF will has a direct influence on the placement results of the remaining VNFs. Such a decision dependency relationship can be learned with a parametric model, which selects a solution π given a problem instance s. This can be factorized and parameterized by θ as the following probability chain rule:

$$p_\theta(\pi|s) = \prod_{t=1}^{N_f} p_\theta(\pi_t|s, \pi_{1:t-1}) \tag{5.6}$$

where $\pi_{1:t-1}$ specifies the placement decisions at all previous steps before t.

The parameters θ of the model are learnt by maximizing the conditional probability for the training set as follows:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_s \sum_{t=1}^{N_f} \log p(\pi_t|s, \theta) \tag{5.7}$$

where N_f is the number of VNFs, and the sum is over all training instances.

With the stochastic policy $p(\pi|s)$ for a given problem instance s, the solution of this VNF placement problem can be decoded sequentially. At each step t, the decoder outputs the placement decision for current VNF and then updates the network status accordingly. Once selected to host the current VNF f, the selected node's residual resources will change, which will then influence the decisions for remaining VNFs. Following the structure in [175], we integrate these decoding information into a special *context node* (c) and augment the graph by connecting the context node to all graph nodes. The decoder computes an attention sublayer on top of the encoder, but with messages only to the context node. Finally, the final probabilities are computed using a single-head attention mechanism. The decoding process is illustrated in Fig. 5.2 and is described in details as follows.

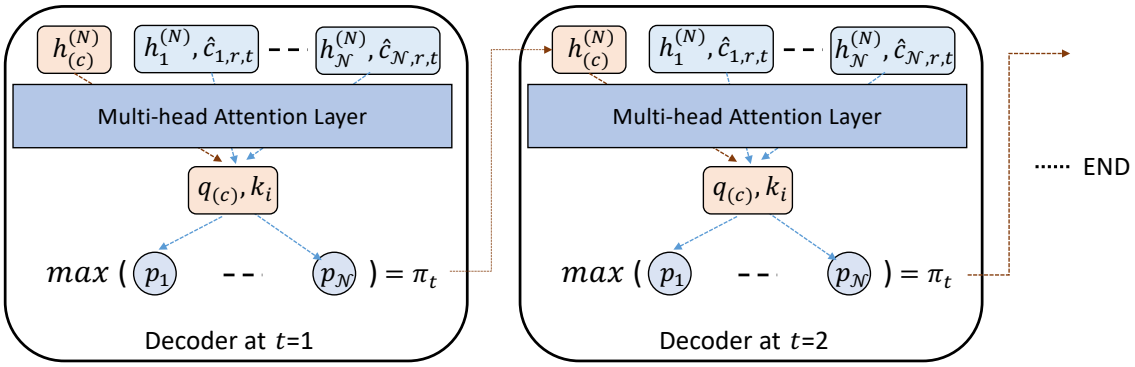


Fig. 5.2 Context-aware decoder with attention for the VNF placement problem.

1. Construct node embedding for context node (c):
   To facilitate the capacity constraints, we keep track of the residual resource capacities $\hat{c}_{i,r,t}$ for each type of resources $r \in \{1,\cdots,\mathscr{R}\}$ in nodes $i \in \{1,\cdots,\mathscr{N}\}$ at step t. After a node, say i, is selected at step t, the resource status of node i is then updated as follows:

$$\hat{c}_{i,r,t} = \begin{cases} \max(0, \hat{c}_{i,r,t} - d_{fr}) & \text{if } t > 0 \\ C_{i,r} & \text{if } t = 0 \end{cases} \tag{5.8}$$

   where $C_{i,r}$ is the initial resource capacity and d_{fr} is the required amount of resources to instantiate VNF f.

   Based on the features of the VNF placement problem, it is reasonable to construct the context of the decoder at step t with the graph embedding, the node $\pi_{t_{f-1}}$ selected to host the previous VNF $(f-1)$ at step t_{f-1}, the destination node $D_{st}(s)$ of current service s, resource demands d_{fr} of current VNF f, and residual resource capacities $\hat{c}_{i,r,t}$ of all graph nodes $i \in \{1,\cdots,\mathscr{N}\}$. However, the residual resource capacity

corresponds to individual nodes. For efficiency, we include it in the computation of the attention layer in (5.10). Therefore, the context node can be constructed as following:

$$h_{(c)}^{(N)} = \begin{cases} \left[\bar{h}^{(N)}, h_{\pi_{t_{f-1}}}^{(N)}, h_{D_{st}(s)}^{(N)}, \{d_{fr}\}_{r\in\{1,\cdots,\mathscr{R}\}}\right] & \text{if } t > 0 \\ \left[\bar{h}^{(N)}, h_{S_r(s)}^{(N)}, h_{D_{st}(s)}^{(N)}, \{d_{fr}\}_{r\in\{1,\cdots,\mathscr{R}\}}\right] & \text{if } t = 0 \end{cases} \tag{5.9}$$

where $S_r(s)$ is the source/access node of current service s that includes current VNF f.

In (5.9), $\bar{h}^{(N)}$ provides the holistic graph state information. The following two terms guide the decoder to select nodes between $\pi_{t_{f-1}}$ and $h_{D_{st}(s)}^{(N)}$, which will be useful for shortening the source-to-destination traversing path during the subsequent VNF chaining process.

Next, we can compute the compatibility of this context node with all graph nodes by using the above MHA mechanism. In the attention layer, we can first obtain the query $q_{(c)}$ from the context node, and the keys k_i and values v_i from the node embeddings $h_i^{(N)}$ as following:

$$q_{(c)} = W^Q h_{(c)}^{(N)} \qquad k_i = W^K h_i^{(N)} + W_d^K \hat{c}_{i,r,t} \qquad v_i = W^V h_i^{(N)} + W_d^V \hat{c}_{i,r,t} \tag{5.10}$$

where (W^Q, W^K, W^V) are the weighting coefficiencies to be learnt. (W_d^K, W_d^V) are $(d_k \times 1)$ parameter matrices, which project the context information from the residual resource capacity of individual node.

Finally, the compatibility of the query of context node with all graph nodes can be computed as:

$$u_{(c)j} = \begin{cases} \frac{q_{(c)}^T k_j}{\sqrt{d_k}} & if\ j \text{ is not masked} \\ -\infty & \text{otherwise} \end{cases} \tag{5.11}$$

where $d_k = d_h/M$ and M is the number of heads in MHA.

2. Decoding masking:
To preserve other constraints, the following masking mechanism is applied during every decoding step: (1) masking nodes with any residual resource capacity less than the required resource demand to place current VNF; (2) if any VNF is failed to place in a service, all nodes are masked for the remaining VNFs in this service (i.e., directly reject this service).

3. Calculation of log-probabilities:
   In order to decode the selection at current step, we follow the operations in [169], using a *single* attention head and clipping the result within $[-C, C](C = 10)$ with tanh in the final decoder layer. Then, we can get the final compatibility as

$$u_{(c)j} = \begin{cases} C \cdot \tanh\left(\frac{q_{(c)}^T k_j}{\sqrt{d_k}}\right) & if\ j \text{ is not masked} \\ -\infty & \text{otherwise} \end{cases} \tag{5.12}$$

   These compatibilities can be interpreted as unnormalized log-probabilities of selecting each node. The final output probability vector p can be computed by using a softmax:

$$p_i = p_\theta(\pi_t = i | s, \pi_{1:t-1}) = \frac{e^{u_{(c)i}}}{\sum_{j=1}^{\mathcal{N}} e^{u_{(c)j}}} \tag{5.13}$$

   where p_i manifests the probability of selecting node i to host current VNF. With some decoding strategy, e.g. greedy decoding, the selected node to host current VNF can be directly decided by finding the node with maximum p_i value.

5.3.3 Decoder for VNF Chaining

The VNF chaining problem aims to find the traversing path to reach the placed VNFs according to the required order. The strong dependency among the sequential decisions makes the problem quite complicated in traditional optimization approaches. Working beyond the decoding structure for the VNF placement problem, we now re-construct the context node (c) by integrating the link information to facilitate the decoding for the VNF chaining progress. This is illustrated in details as follows:

1. Construct and embed context node (c):
   Similar to the process for the context node c in Section 5.3.2, we keep track of the residual link bandwidth $\hat{c}_{e,t}$ for each graph link/edge $e \in \{1, \cdots, L\}$ at step t. After a link, say e, is selected at step t to construct the chaining path, the resource status of link e is then updated as follows:

$$\hat{c}_{e,t} = \begin{cases} \max(0, \hat{c}_{e,t} - r_s) & \text{if } t > 0 \\ C_e & \text{if } t = 0 \end{cases} \tag{5.14}$$

   where C_e is the initial resource capacity of link e, and r_s is the requested flow rate for current service s.

Note that selecting a link is equivalent to select an adjacent node of the end node of previously selected link. Therefore, the link context of the decoder at time t can be constructed with the graph embedding, states of previously selected node and destination node, and the requested flow rate of s as following:

$$h_{(c)}^{(N)} = \begin{cases} \left[\bar{h}^{(N)}, h_{\pi_{t-1}}^{(N)}, h_{D_{st}(s)}^{(N)}, r_s\right] & \text{if } t > 0 \\ \left[\bar{h}^{(N)}, h_{S_r(s)}^{(N)}, h_{D_{st}(s)}^{(N)}, r_s\right] & \text{if } t = 0 \end{cases} \tag{5.15}$$

Likewise, we can now compute the compatibility $u_{(c)j}$ of $h_{(c)}^{(N)}$ with all nodes through (5.10)–(5.12) accordingly. However, the resource context information in (5.10) now becomes residual link resource capacity $\hat{c}_{e,t}$. Finally, the output probability vector of selecting node j (and thus the corresponding outgoing link e connecting node j from π_{t-1} or $S_r(s)$) to construct the current chaining path can be computed as the following:

$$p_i = p_\theta(\pi_t = i | s, \pi_{1:t-1}) = \frac{e^{u_{(c)i}}}{\sum_j e^{u_{(c)j}}} \tag{5.16}$$

where the above sum along j only includes the nodes that are not masked.

2. Decoding masking:
   During the chaining process for each virtual link, all non-adjacent nodes and the node selected at the last step are masked. Moreover, to preserve the link resource constraints, the nodes are also masked if their remaining link capacities are less than the required flow rate of current service. Finally, for any decoding step for service s, if the returned output is *NULL* (i.e., all nodes are masked at some step), then the service request will be rejected and all previously allocated resources for already placed VNFs and links in service s will be released.

5.3.4 Decoding for Joint VNF Placement and Chaining

To this end, we can present the decoding procedures for the joint VNF placement and chaining problem as follows:

1. Select a service request s according to some strategy, e.g., greedy approach and start the decoding process from the source node $S_r(s)$ of this service (i.e., previously selected node π_{t-1} is the source node, $\pi_{t-1} = S_r(s)$);

2. Obtain the node π_{t_f} to host the first VNF in the remaining ordered VNF set though the decoder for the VNF placement. Here, t_f denotes the step placing the current VNF;

3. For each incremental step $t = t + 1$, select a node π_t according to the decoding result for the VNF chaining to construct the chaining path from previously selected node. This happens sequentially until reaching the node π_{t_f} or $\pi_t = NULL$ (if this occurs, service s will be directly rejected);

4. Update the residual node resources, remaining VNF set and remaining link resources;

5. Go to Step 2 if the remaining VNF set is not empty and s is not rejected. Otherwise go to Step 1 until all service requests have been iterated.

5.4 Numerical Evaluation

5.4.1 Experiment Setup

In order to train the whole model, we construct a benchmarking problem by using the constraints (4.42)–(4.47). The objective function is to optimize the following cost function:

$$C_{st}(\pi) = \sum_{s} \Big(\sum_{f} k_n d_f + k_l L_{th}^{s} r_s - k_p \Big) \tag{5.17}$$

where k_n, k_l, k_p are the coefficiencies specifying the resource/service prices, L_{th}^{s} is the length of the chaining path for service s, and d_f, r_s indicate the corresponding resource demands required to deploy such a SFC.

Finally, similar to [175], we use the REINFORCE gradient estimator [185] with baseline $b(s)$ to optimize the following loss function:

$$\bigtriangledown \mathscr{L}(\theta|s) = \mathbb{E}_{p_\theta(\pi|s)} \big[(C_{st}(\pi) - b(s)) \bigtriangledown \log p_\theta(\pi|s) \big] \tag{5.18}$$

where the baseline $b(s)$ is defined as the cost achieved by the best model trained so far.

With the baseline as the supervisor, the training will always try to challenge the best model achieved so far so that the model can improve itself progressively. Specifically, the function $C_{st}(\pi) - b(s)$ is negative if cost achieved by current solution π is better than the baseline, causing actions to be reinforced.

To set up the simulation, we use current BT's IP network topology within Europe[1] as the physical network. Arbitrary node pair is chosen as the source and destination of each network slice request. 5 nodes are randomly selected to act as the access nodes. For each node in the network, a fixed number (i.e., a random number in $[5, 10]$ in the experiments) of

[1] http://www.topology-zoo.org/dataset.html

computing resources is considered. The capacity of each network link is configured as 10. For each service, in addition to the access & destination nodes, 1 VNF demanding a random computing resource within $[1,3]$ is added. This leads to 2 virtual links to chain. The flow rate of each service is configured as a random number within $[1,3]$. Coefficiencies of k_n, k_l, k_p in (5.17) are set to 200, 10, 2000, respectively. In the experiments, normalized objective value $-C_{st}(\pi)/k_p$ is calculated as a measure of the revenue-related performance of the compared algorithms.

Hyperparameters: We initialize model parameters with Uniform$(-1/\sqrt{d_x}, 1/\sqrt{d_x})$. For every epoch, we generate at random 51200 instances as the training data. The batch sizes for training and evalution are set to 128. A constant learning rate $\eta = 10^{-4}$ is applied. At the end of each epoch, we use another 5000 random instances to evaluate the model trained so far. Model parameters are updated with Adam [188] as optimizer during training. All experiments are run on a single GPU (1080TI) with Pytorch[2]. The placement algorithm proposed in [108] is chosen as a comparison, which decides the policies of VNF placement and the subsequent VNF chaining through two separate optimization programs. Herein, the OR-Tools[3], an open-source optimization library developed by Google, is invoked to solve the optimization programs in [108].

5.4.2 Performance Analysis

As a comparison, we first test the performance solved with the OR-Tools over 10000 instances, which reports that the normalized objective value is 0.95 with a runtime of 32ms per iterate in average. OR-Tools is written in C++, which is tested nearly 100x faster than Python[4]. Since two different programming language are used, we do not directly compare the time complexity between the traditional optimizer and the proposed neural one. The run-time cost provided here only provides an insight to the time complexity of the corresponding implementations. Next, we demonstrate the superiority of the proposed deep learning model over the traditional optimization sovers. As manifested in (5.13), the decoding strategies to map the probabilities to corresponding selections play an important role on the efficacy of the trained model. Therefore, we test the results of the trained model with two different decoding strategies to show the robustness of the trained model.

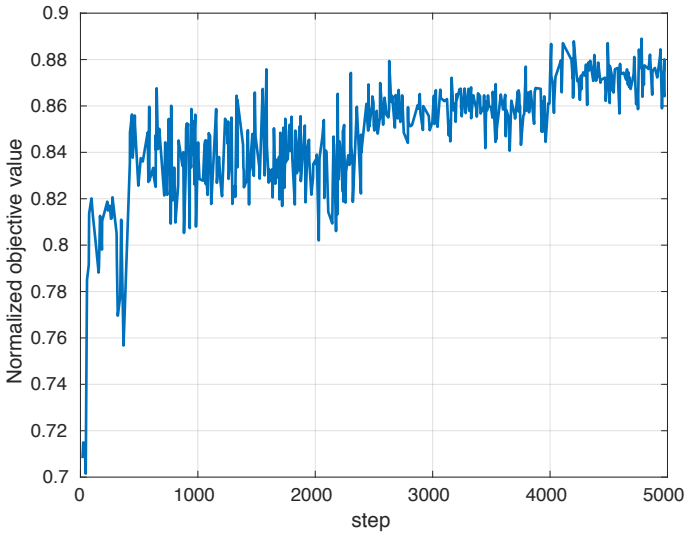


Fig. 5.3 Normalized revenue achieved by the baseline model during training with greedy decoding.

Results achieved with greedy decoding

We first test the results achieved with the greedy decoding strategy. In this case, the node/link with maximum probability in (5.13) is selected at each step. Fig. 5.3 shows the results achieved by the baseline model. We can see that the performance is steadily improved across iterations, which results in a better baseline model to challenge the run-time model training. For the baseline model, we finally obtain an evaluation result of around 0.87 on normalized revenue, which reaches 91.58% of the performance of the compared traditional optimization algorithm. As the training is actually supervized by the run-time baseline model, the trained model presents a 2.1% optimality loss when compared with the baseline model, but still reaches 89.47% of the performance of the compared algorithm.

Additionally, the test results in this experiment report that there is nearly 2.5% of probability that the learned model fails to find a valid solution for the targeted problem. This presents one of the shared limitations faced by existing learning-based solutions. That is, due to the black-box operation, deep learning methods can only generate a solution with some high probablity (can be possibly 99.9%) but without 100% guaranty. For a simple TSP problem, for example, the latest Pointer Network in [98] can only find a valid solution with a chance of around 90%. In practical application, the invalid solutions can be further filtered by a beam search procedure [98] .

[2]https://pytorch.org/

[3]https://developers.google.com/optimization/

[4]https://www.ibm.com/developerworks/community/blogs/jfp/entry/A_Comparison_Of_C_Julia_Python_Numba_Cython_Scipy_and_BLAS_on_LU_Factorization?lang=en

Finally, we obtain a test runtime of about 70ms per iterate according to the above model evaluation experiments. In practical production environment, the optimality gap between the learning based solution and the traditional ones depends on the quality of labels (or baseline models) used to supervise the training process. In our case, 10% optimality loss is presented by the simple greedy decoding and self-learning strategies, in which we only use the best trained model so far as the supervisor. More advanced rollout strategies proposed in latest deep reinforcement learning methods can be exploited to provide a better training supervision, such as deep Q learning [171], actor-critic networks [186] etc. However, in this thesis, we only focus on how to apply the latest deep learning approaches to represent and solve a traditional optimization model in a learning manner. We leave these further attempts to future work.

Results achieved with sampling decoding

Next, we train and test the proposed deep learning model with the sampling decoding strategy to show that a smarter decoding strategy at each step will reach a less optimality loss. In this case, we use the probabilistic sampling results to select the decision at each step in (5.13). Under this decoding strategy, we give a higher chance to select the node/link with higher probability value, but also leave chances to explore the node/link with lower probability. This reduces the constraint violation occurred in the greedy decoding case, in which only node/link with maximum probablity can be selected.

Fig. 5.4 presents the training convergence of the baseline model under sampling decoding strategy. The convergenced result reaches 0.96, even better than the result gained by Google OR-Tools. This further demonstrates the superiority of the REINFORCE training policy, which is able to generate a even better result than a supervised learning label. From the results in Fig. 5.5, we can see that the test results also obtain significantly improvement when the training model reaches convergence. Fig. 5.6 further depicts the length of the returned chaining path. With an adequate learning into the paths explored previously, the return chaining path is progressively optimized and finally converged to nearly optimal one. However, due to the probabilistic sampling strategy, the performance of the convergenced learning model experiences sharp jitters occasionally, as manifested in Fig. 5.5 – Fig. 5.7. This can be avoided by taking multiple sampling trials and reporting the best as the final solution. As shown in Fig. 5.7, the occasional performance jitters can be well eliminated with only several more sampling trials.

5.5 Conclusions

Traditional combinatorial optimization approaches are widely used to model and solve the diverse control and management problems in the field of communication and networking. Since these models are often NP-hard to solve, the traditional optimization solutions exhibit a critical tradeoff between optimality loss and computation time. In order to address these challenges and bring the latest deep learning technologies into the networking applications, we have proposed in this chapter a new encoder-decoder based deep-learning solution, which can represent and solve a traditional combinatorial optimization problem in a learning manner. Through a simulation test over the emerging service function chaining problem in 5G, we have demonstrated that the proposed solution can achieve a less optimality loss than the compared traditional solution. This presents a promising direction for introducing the advanced learning technologies to empower the current and near-future 5G networks with more intelligence.

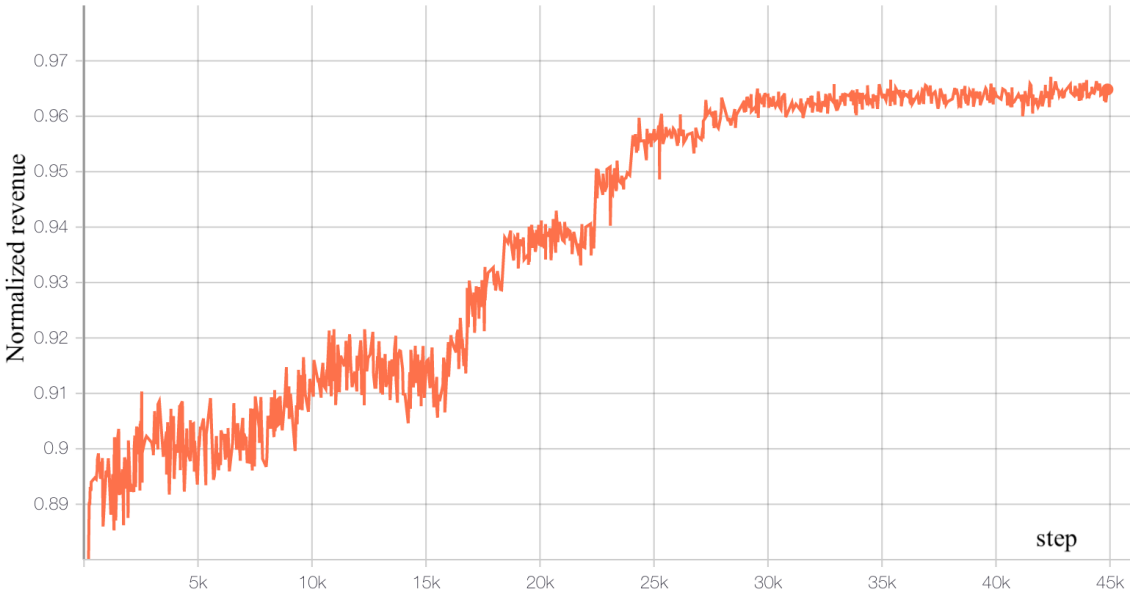


Fig. 5.4 Normalized revenue achieved by the baseline model with sampling decoding during training.

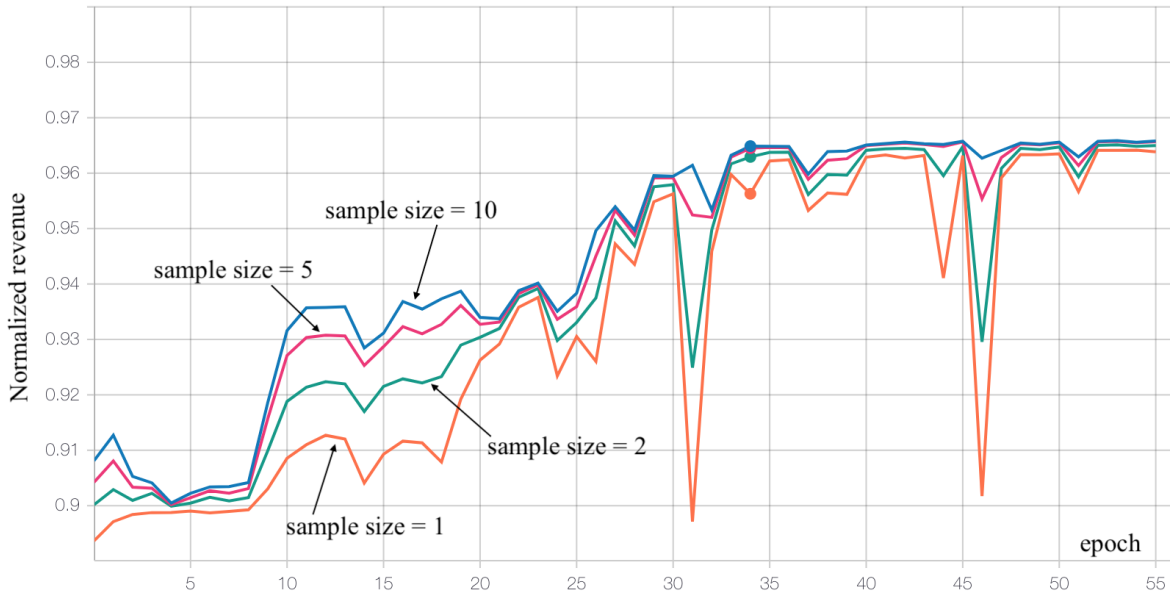


Fig. 5.5 Normalized revenue achieved by evaluating the training model with test data set and sampling decoding strategy.

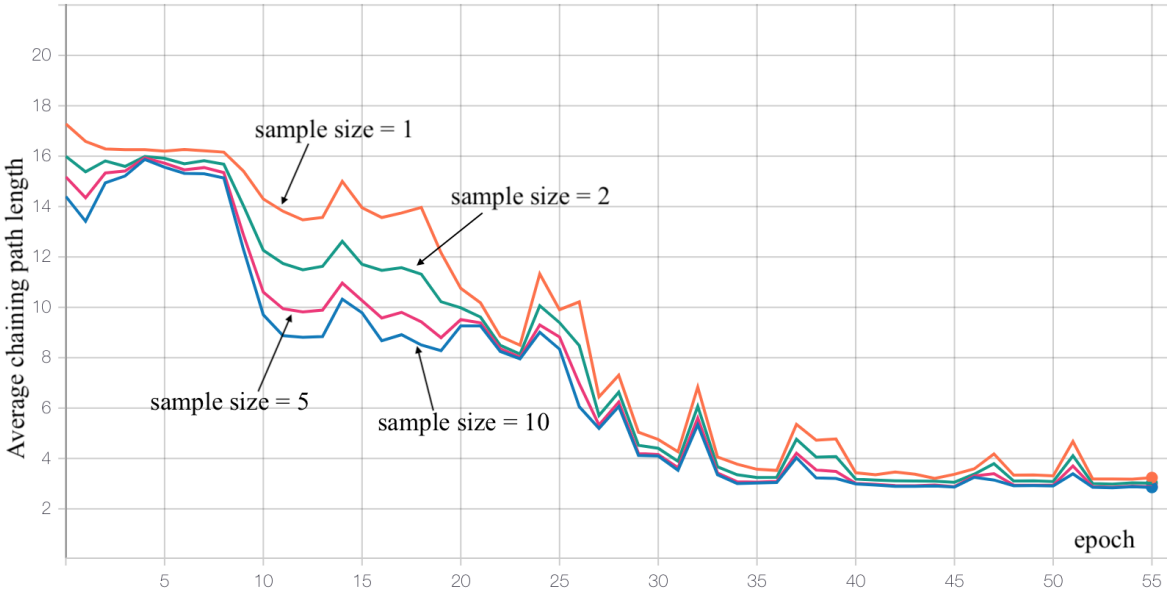


Fig. 5.6 Average chaining path length achieved by evaluating the training model with test data set and sampling decoding strategy.

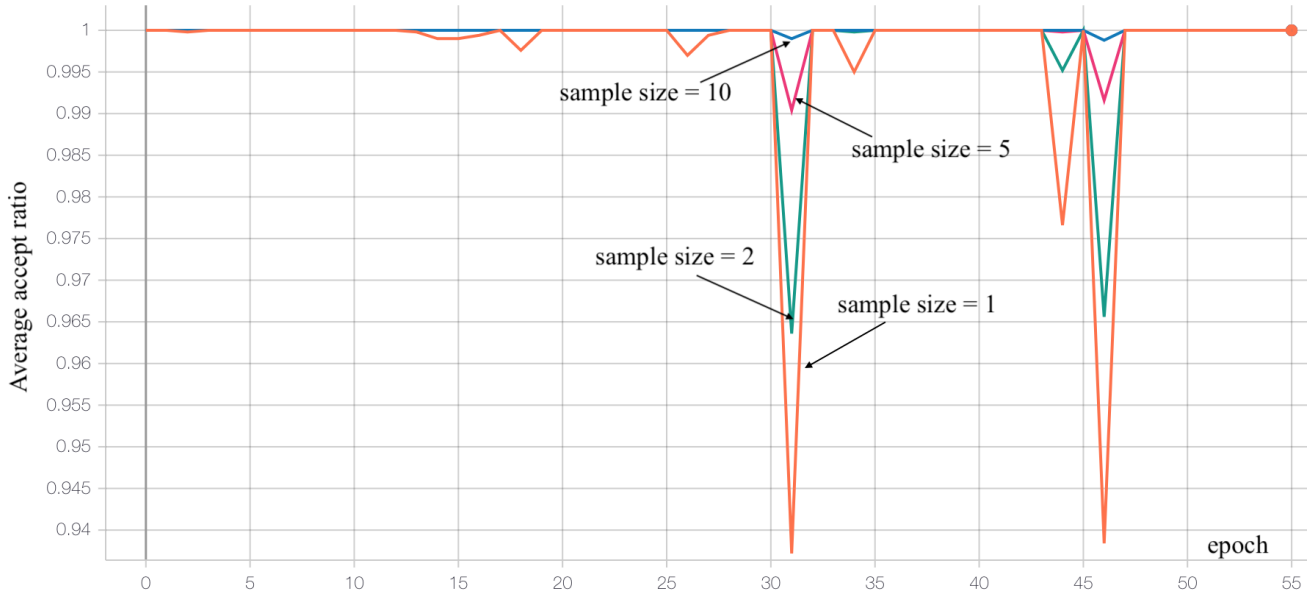


Fig. 5.7 Accep ratio achieved by evaluating the training model with test data set and sampling decoding strategy.

Chapter 6

Conclusions and Future Work

Recent years have witnessed intense research efforts in developing novel optimization and learning methods that strive to meet the various requirements for the implementation of network softwarization in 5G. With the ever-growing demands of the diverse 5G emerging applications and the networking intelligence, the research on network softwarization is expected to experience much speedy advancement in the upcoming years. In this thesis, several aspects of network softwarization optimization problems have been examined, especially with the learning augmented optimization. Three primary contributions have been made with the faith that they will give rise to more research ideas and benefit practical applications in this domain:

- Considering the inherently dynamic nature of the underlying physical networking environment in 5G, a stochastic NFV solution along with its distributed implementation have been designed. By explicitly integrating the knowledge of influential network variations into a two-stage stochastic resource utilization model, the proposed solution has presented a robust and swift NFV implementation for its application in dynamic networks.

- With the proposed concept of learning augmented optimization, this thesis has investigated the promissing approaches to integrate the traditional optimization and learning technologies together for the optimization of network softwarization. Accodingly, considering the possible absence of the statistics information about the underlying stochastic environment, this thesis has proposed an online learning approach by combining the stochastic combinatorial optimization and Lyapunov based learning theory. This solution makes it possible to adjust the ongoing networking services to the changing network status according to the newly observed information at runtime.

- Finally, distinct to the above work, this thesis also starts the new attempt of fully utilizing the latest deep learning technologies to solve the traditional cosntraint combinatorial optimization problems. This further facilitates the merging of the two distinct technologies to empower the underlying networking systems with more intelligence.

Generally speaking, the technologies developed in this thesis aim to supply advanced optimization technologies in an attempt to meet the distinct requirements of intelligent network softwarization in 5G. They are initially driven from the research point of view and expected to provide a solid theorectical guidance for the rollout of network softwarization in practical engineering environment. To make further progress towards this direction, the future work will primarily consider the following four aspects that may bring immediate consequences:

- It is normally very difficult to obtain a large number of good-quality labels as the training data set in many NP-hard combinatorial optimization problems. When training the model by following a not so good supervisor/baseline, the performance of the resulted model will suffer from great optimality loss. Therefore, a stronger basedline or rollout policy is required to further improve the optimality gap between the deep learning based optimization solvers and traditional optimization solvers. In this respect, many latest deep reinforcement learning approaches bring a promissing direction to move this work forward.

- In addition, the processing capabilities of a single machine have been greatly improved by many new computing technologies, e.g., super-computer, GPU. In spite of this fact, the memory and computation of a single machine are still significantly lagging behind the requirements of processing the exploding big data in todays networking environment. Therefore, a natural solution is to develop distributed and parallelized solutions for the further improvement of intelligent optimization technologies. Apart from the extensively applied data parallelization, there is still a huge space to explore the parallelizations for both the mathematical optimization programs and machine learning models. We believe more parallelization solutions in this field are anticipated in the near future.

- Graphs arise naturally in many real-world applications including the diverse optimization problems in the networking area. Traditionally, machine learning models for graphs have been mostly designed for static graphs. However, many applications involve evolving graphs. This introduces important challenges for the learning and inference since nodes, attributes, and edges change over time. How to deal with graphs

with dynamic structures is still an open problem. Dynamic graph neural network is being actively researched on. We believe it would be a big milestone to achieve the stability and adaptability of general graph neural networks.

- The performance of existing machine/deep learning techniques is still barely satisfactory in real-time networking systems. Therefore, more efforts are still required so as to develop a light-weight machine learning process for real-time system controls. For example, to accelerate the data training process with analytical models (*e.g.*, [187]).

- Finally, timely data monitoring and collection are normally required in these proposed solutions so that the controller can get a real insight to the run-time networking environment and make the response accordingly. This requires more efforts to develop advanced network telemetry technology to support the intelligent optimization in this thesis. This is non-trival since both modelling the environmental dynamics and data-driven intelligence require a vast volume of data to work. A promising direction for this problem would be to install edge/device-side intelligence. This avoids the delay and overhead of data collections and brings intelligence directly to the locations where the data is generated.

Although it still requires further efforts in improving the related methods and implementing them in practical products, it is believed that the developments of these methods will contribute positively to practical applications and inspire more competent solutions to implement the network softwarization with more advanced optimization and intelligent learning technologies.

References

[1] Cisco Public, "Cisco visual networking index: Forecast and methodology, 2016–2021," Cisco White Paper, 2017. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networkingindex-vni/complete-white-paper-c11-481360.pdf

[2] J. G. Andrews et al., "What will 5G be?" *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, Jun. 2014.

[3] S. Chen and J. Zhao, "The requirements, challenges, and technologies for 5G of terrestrial mobile telecommunication," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 36–43, May 2014.

[4] GSMA Intelligence,"Understanding 5G: Perspectives on future technological advancements in mobile," White paper, 2014.

[5] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing & softwarization: A survey on principles, enabling technologies & solutions,", *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2429–2453, third quarter, 2018.

[6] J. van de Belt, H. Ahmadi, and L. E. Doyle, "Defining and surveying wireless link virtualization and wireless network virtualization," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1603–1627, 3rd Quart., 2017.

[7] A. Manzalini et al., "Towards 5G software-defined ecosystems: Technical challenges, business sustainability and policy issues," IEEE Future Directions, White Paper, Jul. 2016.

[8] ITU-R Rec. M.2083, "IMT Vision – Framework and Overall Objectives of the Future Development of IMT for 2020 and Beyond," 2015.

[9] 3GPP, TR 22.891, "Technical Specification Group Services and System Aspects; Feasibility Study on New Services and Markets Technology Enablers, Stage 1," Release 14, v. 14.2.0, Sept. 2016.

[10] 3GPP, TS 23.501, "System Architecture for the 5G System," Release-15, v. 2.0.1, Dec. 2017.

[11] R. Wen et al., "Robust network slicing in software-defined 5G networks," in *Proc. IEEE GLOBECOM*, Singapore, Dec. 2017, pp. 1–6.

[12] L. Yan, X. Fang, Y. Fang, and X. Cheng, "Dual-scheduler design for C/U-plane decoupled railway wireless networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 7, pp. 1842–1855, Jul. 2017.

[13] R. Mijumbi et al., "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 1st Quart., 2016.

[14] M. G. Xavier et al., "Performance evaluation of container-based virtualization for high performance computing environments," in *Proc. IEEE PDP*, Belfast, U.K., Feb. 2013, pp. 233–240.

[15] ETSI NFV, Network Functions Virtualisation (NFV); Architectural framework , GS NFV 002, Oct. 2013.

[16] A. M. Medhat et al., "Service function chaining in next generation networks: State of the art and research challenges," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 216–223, Feb. 2017.

[17] Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV, document GS NFV 002, ETSI NFV, Sophia Antipolis, France, Oct. 2013.

[18] Z. Fei et al., "A survey of multi-objective optimization in wireless sensor networks: Metrics, algorithms, and open problems," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 550–586, 1st Quart., 2017.

[19] M. A. Adnan, M. A. Razzaque, I. Ahmed, and I. F. Isnin, "Bio-mimic optimization strategies in wireless sensor networks: A survey," *Sensors*, vol. 14, no. 1, pp. 299–345, 2014.

[20] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge Univ. Press, 2004.

[21] D. Wang, B. Bai, W. Zhao, and Z. Han, "A survey of optimization approaches for wireless physical layer security," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1878–1911, 2rd Quart., 2019.

[22] M. Andrychowicz et al.,"Learning to learn by gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3981–3989.

[23] K. Li and J. Malik, "Learning to optimize," in *Proc. Int. Conf. Learn. Represent.*, 2017.

[24] H. Sun et al., "Learning to optimize: Training deep neural networks for interference management," *IEEE Trans. Signal Process.*, vol. 66, no. 20, pp. 5438–5453, Oct. 2018.

[25] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief, "LORA: learning to optimize for resource allocation in wireless networks with few training samples," arXiv preprint arXiv:1812.07998 v1, Dec. 2018.

[26] GSMAIntelligence, "Understanding 5G: Perspectives on future technological advancements in mobile," White paper, 2014.

[27] A. Checko et al., "Cloud RAN for mobile networks-a technology overview," *IEEE Commun. Surv. Tuts.*, vol. 17, no. 1, pp. 405–426, First Quart. 2015.

[28] K. Chen and R. Duan, "C-RAN: The road towards green RAN," China Mobile Research Institute, Beijing, White paper, 2011.

[29] N. Zhang et al., "Cloud assisted HetNets toward 5G wireless networks," *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 59–65, Jun. 2015.

[30] S. M. Abd El-atty and Z. M. Gharsseldien, "On performance of HetNet with coexisting small cell technology," in *Proc. IEEE Conf. Wireless Mobile Netw.*, 2013, pp. 1–8.

[31] P. Agyapong, M. Iwamura, D. Staehle, W. Kiess, and A. Benjebbour, "Design considerations for a 5G network architecture," *IEEE Commun. Mag.*, vol. 52, no. 11, pp. 65–75, Nov. 2014.

[32] 3GPP, The Mobile Broadband Standard (2015), [Online]. Available: http://www.3gpp.org/news-events/3gpp-news/1674-timeline_5g.

[33] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5G wireless networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 1617–1655, 3rd Quart., 2016.

[34] 5G-Infrastructure Public-Private Partnership, 2013. [Online]. Available: http://5g-ppp.eu/.

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

[36] European Commission. (2011). HORIZON 2020, The EU Framework Programme for Research and Innovation [Online]. Available: http://ec.europa.eu/programmes/horizon2020/.

[37] Ericsson, "5Gradioaccess," White paper,2015.

[38] Qualcomm Technologies, Inc., "Qualcomm's 5G vision," White paper, 2014.

[39] Huawei, "5Gatechnologyvision," White paper,2013.

[40] NTTDocomo, "5G radio access: Requirements, concepts technologies," White paper, 2015.

[41] Nokia Networks, "Looking ahead to 5G: Building a virtual zero latency gigabit experience," White paper, 2014.

[42] R. P. Goldberg, "Survey of virtual machine research," *IEEE Comput.*, vol. C-7, no. 6, pp. 34–45, Jun. 1974.

[43] 5GPPP, "View on 5G architecture," 5G PPP Architecture Working Group, EuCNC, Athens, Greece, Jul. 2016.

[44] A. Banchs et al., "A novel radio multiservice adaptive network architecture for 5G networks," in *Proc. IEEE VTC-Spring*, Glasgow, U.K., May 2015, pp. 1–5.

[45] P. Rost *et al.*, "Network slicing to enable scalability and flexibility in 5G mobile networks," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 72—79, May 2017.

[46] K. Katsalis, N. Nikaein, E. Schiller, A. Ksentini, and T. Braun, "Network slices toward 5G communications: Slicing the LTE network," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 146–154, Aug. 2017.

[47] P. Rost et al., "Mobile Network Architecture Evolution Toward 5G," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 84–91, May 2016.

[48] N. Nikaein et al., "Network Store: Exploring Slicing in Future 5G Networks," *Proc. 10th ACM Int'l. Wksp. Mobility in the Evolving Internet Architecture*, Sept. 2015, pp. 1–6.

[49] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 94–100, May 2017.

[50] X. Costa-Perez et al., "Radio Access Network Virtualization for Future Mobile Carrier Networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 27–35, July 2013.

[51] S. Abdelwahab et al., "Network Functions Virtualization in 5G," *IEEE Commun. Mag.*, vol. 54, no. 4, Apr. 2016, pp. 84–91.

[52] G. Even, M. Medina, G. Schaffrath, and S. Schmid, "Competitive and deterministic embeddings of virtual networks," in *Proc. 13th International Conference on Distributed Computing and Networking*, 2012, pp. 106–121.

[53] H. Tong, J. Cao, S. Zhang, and M. Li, "A distributed algorithm for web service composition based on service agent model," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 12, pp. 2008–2021, Dec. 2011.

[54] R. Soulé et al, "Merlin: A language for provisioning network resources," In *CoNEXT*, Dec. 2014, pp. 213–226.

[55] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," in IEEE Trans. Netw. Service Manag., vol. 13, no. 4, pp. 725—739, Dec. 2016.

[56] S.Mehraghdam, M.Keller, and H.Karl, "Specifying and placing chains of virtual network functions," in *Proc. 3rd IEEE Conf. Cloud Netw. (CloudNet)*, Oct. 2014, pp. 7–13.

[57] M. T. Beck and J. F. Botero, "Coordinated allocation of service function chains," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2015, pp. 1–6

[58] R. Cohen, L. Lewin-Eytan, J. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 1346–1354.

[59] S. Sahhaf, W. Tavernier, D. Colle, and M. Pickavet, "Network service chaining with efficient network function mapping based on service decompositions," in *Proc. 1st IEEE Conf. Netw. Softwarization (NetSoft)*, London, U.K., 2015, pp. 1–5.

[60] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *Proc. IEEE Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2015, pp. 171–177.

[61] M. Ghaznavi *et al.*, "Elastic virtual network function placement," in *Proc. 4th IEEE Conf. Cloud Netw. (CloudNet)*, Oct. 2015, pp. 1–6.

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

[63] M. Rost, S. Schmid, and A. Feldmann, "It's about time: On optimal virtual network embeddings under temporal flexibilities," *IEEE 28th International Parallel and Distributed Processing Symposium (IPDPS)*, 2014, pp. 17–26.

[64] R. Riggio, A. Bradai, D. Harutyunyan, T. Rasheed, and T. Ahmed, "Scheduling wireless virtual networks functions," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 2, pp. 240–252, Jun. 2016.

[65] J. F. Riera, X. Hesselbach, M. Zotkiewicz, M. Szostak, and J. F. Botero, "Modelling the nfv forwarding graph for an optimal network service deployment," in *17th International Conference on Transparent Optical Networks (ICTON)*, 5-9 July, 2015, pp. 1–4.

[66] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May, 2015, pp. 98–106.

[67] Z. Liu, S. Wang, and Y. Wang, "Service function chaining resource allocation: A survey," *Cornell Univ. Library*, arXiv: 1608.00095, 2016. [Online]. Available: https://arxiv.org/pdf/1608.00095.pdf.

[68] T. Taleb, M. Bagaa, and A. Ksentini, "User mobility-aware virtual network function placement for virtual 5G network infrastructure," in *Proc. IEEE ICC 2015*, Jun. 2015, pp. 3879–3884.

[69] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *Proc. 10th Int. Conf. Netw. Service Manag. (CNSM)*, Nov. 2014, pp. 418–423.

[70] A. Gupta, M. F. Habib, P. Chowdhury, M. Tornatore, and B. Mukherjee, "On service chaining using virtual network functions in network-enabled cloud systems," in *Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst. (ANTS)*, Dec. 2015, pp. 1–3.

[71] M. Rost and S. Schmid, "Service Chain and Virtual Network Embeddings: Approximations Using Randomized Rounding," *arXiv:1604.02180*, 2016.

[72] F. Rothlauf, *Design of modern heuristics: principles and application*. Springer Science & Business Media, 2011.

[73] I. H. Osman and G. Laporte, "Metaheuristics: A bibliography," *Annals of Operations Research*, vol. 63, no. 5, pp. 511–623.

[74] A. Mohammadkhan et al., "Virtual function placement and traffic steering in flexible and dynamic software defined networks," in *IEEE International Workshop on Local and Metropolitan Area Networks (LANMAN)*, April 2015, pp. 1–6.

[75] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of virtualized deep packet inspection functions in sdn," in *IEEE Military Communications Conference*, Nov. 2013 2013, pp. 992–997.

[76] R. Mijumbi et al., "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in *IEEE Conference on Network Softwarization (NetSoft)*, 13-17 April 2015 2015, pp. 1–9.

[77] R. Shi et al., "Mdp and machine learning-based cost-optimization of dynamic resource allocation for network function virtualization," in *IEEE International Conference on Services Computing (SCC)*, July, 2015, pp. 65–73.

[78] F. Callegati, W. Cerroni, C. Contoli, and G. Santandrea, "Dynamic chaining of virtual network functions in cloud-based edge networks," in *IEEE Conference on Network Softwarization (NetSoft)*, 2015, pp. 1–5.

[79] Y. Jia, R. Buyya, and T. Chen Khong, "Cost-based scheduling of scientific workflow applications on utility grids," in *First International Conference on e-Science and Grid Computing*, July 2005.

[80] A. Fiat, *Online Algorithms: The State of the Art (Lecture Notes in Computer Science)*. Springer, Sep. 1998.

[81] T. Lukovszki and S. Schmid, "Online admission control and embedding of service chains," in *Structural Information and Communication Complexity*, Springer, 2014, pp. 104–118.

[82] Paolini, Monica and Fili, Senza, "Mastering Analytics: How to benefit from big data and network complexity: An Analyst Report," *RCR Wireless News*, 2017.

[83] L. Liu et al., "Deep learning based optimization in wireless network," In *Proc. IEEE International Conference on Communications (ICC)*, 2017, pp. 1–6.

[84] Y. Lee, "Classification of node degree based on deep learning and routing method applied for virtual route assignment," *Ad Hoc Networks*, vol. 58, pp. 70–85, 2017.

[85] B. Mao et al., "Routing or computing? the paradigm shift towards intelligent computer network packet trans- mission based on deep learning," *IEEE Transactions on Computers*, vol. 6, no. 11, pp. 1946–1960, 2017.

[86] R. Atallah, C. Assi, and M.Khabbaz, "Deep reinforcement learning-based scheduling for roadside communication networks," In *Proc. 15th IEEE International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2017, pp. 1–8.

[87] S. Chinchali et al., "Cellular network traffic scheduling with deep reinforcement learning," In *Proc. National Conference on Artificial Intelligence (AAAI)*, 2018.

[88] P. Ferreira et al., "Multi-objective reinforcement learning-based deep neural networks for cognitive space communications," In *Proc. IEEE Cognitive Communications for Aerospace Applications Workshop (CCAA)*, 2017, pages 1–8.

[89] R. Mennes, M.Camelo, M. Claeys, and S. Latre, "A neural-network-based MF-TDMA MAC scheduler for collaborative wireless networks," In *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, 2018, pages 1–6.

[90] Y. Zhou et al., "A deep learning based radio resource assignment technique for 5G ultra dense networks," *IEEE Network*, vol. 32, no. 6, pp. 28–34, 2018.

[91] T. J O Shea and T C. Clancy, "Deep reinforcement learning radio control and signal detection with KeRLym, a gym RL agent," *arXiv preprint arXiv:1605.09221*, 2016.

[92] Z. Fadlullah et al., "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2432–2455, 2017.

[93] N. Ahad, J. Qadir, and N. Ahsan, "Neural networks in wireless networks: Techniques, applications and guidelines," *Journal of Network and Computer Applications*, vol. 68, pp. 1–27, 2016.

[94] Q. Mao, F. Hu, and Q. Hao, "Deep learning for intelligent wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2595–2621, 2018.

[95] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys & Tutorials*, early access, DOI 10.1109/COMST.2019.2904897, 2018.

[96] R. Li *et al.*, "Intelligent 5G: When Cellular Networks Meet Artificial Intelligence," *IEEE Commun. Mag.*, vol. 24, no. 5, 2017, pp. 175–183.

[97] C. Jiang *et al.*, "Machine Learning Paradigms for Next-Generation Wireless Networks," *IEEE Wireless Commun.*, vol. 24, no. 2, pp. 98—105, April 2017.

[98] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer Networks," in *Proc. NIPS*, Dec. 2015.

[99] X. Cheng *et al.*, "Network Function Virtualization in Dynamic Networks: A Stochastic Perspective," *IEEE JSAC*, vol. 36, no. 10, Oct. 2018.

[100] S. Deb and P. Monogioudis, "Learning-based Uplink Interference Management in 4G LTE Cellular Systems," *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 398–411, April 2015.

[101] Q. Zheng *et al.*, "Delay-Optimal Virtualized Radio Resource Scheduling in Software-Defined Vehicular Networks via Stochastic Learning," *IEEE Trans. Veh. Technol.*, vol. 65, no. 10, pp. 7857–7867, Oct. 2016.

[102] M. Wang *et al.*, "Smart Exploration in HetNets: Minimizing Total Regret with mmWave," in *Proc. IEEE Int. Conf. Sens., Commun. Netw.*, June 2016.

[103] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-Aware Mobility Management for Mobile Edge Computing in Ultra Dense Networks," *IEEE JSAC*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017.

[104] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices," *IEEE JSAC*, vol. 34, no. 12, pp. 3590–3605, Dec. 2017.

[105] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*, San Rafael, CA, USA: Morgan and Calypool, 2010.

[106] B. K. Donohoo *et al.*, "Context-Aware Energy Enhancements for Smart Mobile Devices," *IEEE Trans. Mobile Comput.*, vol. 13, no. 8, Aug. 2014, pp. 1720–1732.

[107] R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction*, Cambridge, MA, USA: MIT Press, vol. 1, 2017.

[108] Y. Zhu and M. Ammar, "Algorithms for Assigning Substrate Network Resources to Virtual Network Components," in *Proc. INFOCOM*, April 2006.

[109] 5GMF White Paper, 5G Mobile Communications Systems for 2020 and beyond, 2017, [Online]. Available: https://5gmf.jp/en/whitepaper/5gmf-white-paper-1-1/.

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

[111] A. Jarray and A. Karmouch, "Decomposition approaches for virtual network embedding with one-shot node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 1012—1025, Jun. 2015.

[112] Z. Zhao *et al.*, "Autonomic communications in software-driven networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2431–2445, Nov. 2017.

[113] M. R. Akdeniz *et al.*, "Millimeter wave channel modeling and cellular capacity evaluation," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1164–1179, Jun. 2014.

[114] M. Bagaa, T. Taleb, A. Laghrissi, A. Ksentini, and H. Flinck, "Coalitional game for efficient virtual evolved packet core in 5G networks," in *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 469–484, Mar. 2018.

[115] A. Laghrissi, T. Taleb, and M. Bagaa, "Conformal mapping for optimal network slice planning based on canonical domains," in *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 519–528, Mar. 2018.

[116] M. Bagaa, T. Taleb, and A. Ksentini, "Service-aware network function placement for efficient traffic handling in carrier cloud," in *Proc. IEEE WCNC'14*, Apr. 2014, pp. 2402–2407.

[117] A. Laghrissi, T. Taleb, M. Bagaa, and H. Flinck, "Towards edge slicing: VNF placement algorithms for a dynamic & realistic edge cloud environment," in *Proc. IEEE Globecom 2017*, Dec. 2017, pp. 1–6.

[118] I. Benkacem, T. Taleb, M. Bagaa, and H. Flinck, "Optimal VNFs placement in CDN slicing over multi-cloud environment," in *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 616–627, Mar. 2018.

[119] P. Wang, J. Lan, X. Zhang, Y. Hu, and S. Chen, “Dynamic function composition for network service chain: Model and optimization," *Computer Networks*, vol. 92, Part 2, pp. 408–418, Dec. 2015.

[120] Y. Jia, C. Wu, Z. Li, F. Le, and A. Liu, “Online scaling of NFV service chains across geo-distributed datacenters," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 2008–2025, April 2018.

[121] Y. Li, L. T. X. Phan, and B. T. Loo, “Network functions virtualization with soft real-time guarantees," in *Proc. IEEE INFOCOM 2016*, Apr. 2016, pp. 1–9.

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

[123] D. P. Palomar and M. Chiang, “A tutorial on decomposition methods for network utility maximization," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1439–1451, Aug. 2006.

[124] Q.-V. Pham and W.-J Hwang, “Network utility maximization-based congestion control over wireless networks: A survey and potential directives," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 1173–1200, 2nd Quart., 2017.

[125] H. M. Bidhandi, J. Patrick, “Accelerated sample average approximation method for two-stage stochastic programming with binary first-stage variables," *Applied Mathematic Modelling,* vol. 41, pp. 582–595, 2017.

[126] S. Su *et al.*, “Energy-aware virtual network embedding," in *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1607–1620, Oct. 2014.

[127] Y. Dinitz, N. Garg, M.X. Goemans, “On the single source unsplittable flow problem," *Proceedings of the 39th Symposium on the Foundations of Computer Science*, Palo Alto, CA, 1998, pp. 290–299.

[128] C. Kim *et al.*, “In-band Network Telemetry via Programmable Dataplanes,” Industrial demo, *ACM SIGCOMM*, 2015.

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

[130] A. J. Conejo, E. Castillo, R. Mnguez, and R. Garca-Bertrand, D*ecomposition Techniques in Mathematical Programming: Engineering and Science Applications*, Springer, 2006.

[131] V. Borkar, *Stochastic Approximation: A Dynamical Systems Viewpoint.* Cambridge, U.K.: Cambridge Univ. Press, 2008.

[132] J. Higle and S. Sen, “Stochastic decomposition: An algorithm for two stage linear programs with recourse," *Math. Oper. Res.*, vol. 16, no. 3, pp. 650–669, 1991.

[133] T. W. Kuo, B. H. Liou, K. C. Lin, and M. J. Tsai, “Deploying Chains of Virtual Network Functions: On the Relation Between Link and Server Usage," in *Proc. IEEE INFOCOM 2016*, Apr. 2016, pp.1–9.

[134] G. L. Stuber, *Principles of Mobile Communication*, Second Edition, Kluwer Academic Publishers, 2001.

[135] Y. Gai, B. Krishnamachari, and R. Jain, “Combinatorial network optimization with unknown variables: multi-armed bandits with linear rewards and individual observations”, *IEEE/ACM TRANS. Netw.*, vol. 20, no. 5, pp. 1466–1478, Oct. 2012.

[136] P. Z. Peebles, *Probability, Random Variables and Random Signal Principles.* New York, NY, USA: McGraw-Hill, 1993.

[137] G. G. Yin and H. J. Kushner, *Stochastic Approximation and Recursive Algorithms and Applications*. New York, NY, USA: Springer-Verlag, 2003.

[138] X. Chen, J. Wu, Y. Cai, H. Zhang, and T. Chen, “Energy-efficiency oriented traffic offloading in wireless networks: A brief survey and a learning approach for heterogeneous cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 4, pp. 627–640, April 2015.

[139] X. Cheng, Y. Wu, G. Min, A.Y. Zomaya, and X. Fang, “Safeguard network slicing optimization in 5G: a learning augmented optimization approach,” *IEEE Journal on Selected Areas in Communications*, under review.

[140] 3GPP Release 15 specifications, http://www.3gpp.org/release-15, June, 2018.

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

[142] M. Leconte *et al.*, “A resource allocation framework for networkslicing,” in *Proc. INFOCOM*, April 2018.

[143] G. Even, M. Rost, and S. Schmid, “An approximation algorithm for path computation and function placement in SDNs,” In *Proc. SIROCCO*, July 2016, pp. 374–390.

[144] L. Huang, M. Chen, and Y. Liu, “Learning-aided stochastic network optimization with state prediction,” *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1810–1820, Aug. 2018.

[145] A. J. Kleywegt, A. Shapiro, T. Homem-De-Mello, The sample average approximation method for stochastic discrete optimization, SIAM J. Optim 12 (2) (2001) 479—502.

[146] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*, San Rafael, CA, USA: Morgan and Calypool, 2010.

[147] M. J. Neely, “Distributed stochastic optimization via correlated scheduling,” *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 759—772, April 2016.

[148] G. L. Stuber, *Principles of Mobile Communication*, Second Edition, Kluwer Academic Publishers, 2001.

[149] S. Su *et al.*, “Energy-aware virtual network embedding,” in *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1607—1620, Oct. 2014.

[150] Y. L. Lee, J. Loo, T. C. Chuah, and L. Wang, “Dynamic network slicing for multitenant heterogeneous cloud radio access networks,” *IEEE Trans. Wireless Commu.*, vol. 17, no. 4, pp. 2146–2161, April 2018.

[151] Y. Jia *et al.*, “Online scaling of NFV service chains across geo-distributed datacenters," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 2008–2025, April 2018.

[152] Y. Li, L. T. X. Phan, and B. T. Loo, “Network functions virtualization with soft real-time guarantees," in *Proc. IEEE INFOCOM 2016*, Apr. 2016, pp. 1–9.

[153] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Warfield, “Split/merge: System support for elastic execution in virtual middleboxes,” in *Proc. 10th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2015, pp. 227–240.

[154] Y. Mao, J. Zhang, and K. B. Letaief, “Dynamic computation offloading for mobile-edge computing with energy harvesting devices,” *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3604, Dec. 2016.

[155] C. Liang and F. R. Yu, “Wireless network virtualization: a survey, some research issues and challenges,” *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 358–380, first quarter 2015

[156] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1998.

[157] P. Z. Peebles, *Probability, Random Variables and Random Signal Principles.* New York, NY, USA: McGraw-Hill, 1993.

[158] W.-K. Mak, D.P. Morton, and R.K. Wood, “Monte carlo bounding techniques for determining solution quality in stochastic programs,” *Operations Research Letter*, vol. 24, no. 1, pp. 47–56, Feb. 1999.

[159] M. J. Neely, “A simple convergence time analysis of drift-plus penalty for stochastic optimization and convex programs,” *arXiv preprint arXiv:1412.0791*, 2014. [Online]. Available: https://arxiv.org/pdf/1412.0791.pdf

[160] R.Boutaba et al.,“A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities,” *J. Internet Services Appl.*, vol. 9, p. 16, pp.1–99, Jun. 2018.

[161] S. Pouyanfar et al., “A survey on deep learning: algorithms, techniques, and applications,” *ACM Computing Surveys (CSUR)*, vol. 51, no.5, pp. 1–36, 2018.

[162] N. Kato et al., “The deep learning vision for heterogeneous network traffic control: proposal, challenges, and future perspective,” *IEEE Wireless Commun.*, vol. 24, no. 3, Jun. 2017, pp. 146–53. DOI: 10.1109/ MWC.2016.1600317WC.

[163] H. Dai, E. B Khalil, Y. Zhang, B. Dilkina, and L. Song, “Learning combinatorial optimization algorithms over graphs,” In *Advances in Neural Information Processing Systems*, pp. 6348-–6358, 2017.

[164] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takac, "Reinforcement learning for solving the vehicle routing problem," In *Advances in Neural Information Processing Systems*, pp. 9860—-9870, 2018.

[165] M. A. Alsheikh, D. Niyato, S. Lin, H. Tan, and Z. Han, "Mobile big data analytics using deep learning and Apache Spark," *IEEE network*, vol. 30, no. 3, pp. 22—29, 2016.

[166] S. Mohanty and A. Pozdnukhov, "Graph cnn+ lstm framework for dynamic macroscopic traffic congestion prediction," In *International Workshop on Mining and Learning with Graphs*, 2018.

[167] J. J Hopfield and D. W Tank, "Neural computation of decisions in optimization problems," *Biological cybernetics*, vol. 52, no. 3, pp. 141—152, 1985.

[168] K. A Smith, "Neural networks for combinatorial optimization: a review of more than a decade of research," *INFORMS Journal on Computing*, vol. 11, no. 1, pp. 15–34, 1999.

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

[170] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets," In *International Conference on Learning Representations*, 2016.

[171] V. Mnih, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp.529–533, 2015.

[172] A. Nowak, S. Villar, A. S Bandeira, and J. Bruna, "A note on learning algorithms for quadratic assignment with graph neural networks," arXiv preprint arXiv:1706.07450, 2017.

[173] Y. Kaempfer and L. Wolf, "Learning the multiple traveling salesmen problem with permutation invariant pooling networks," arXiv preprint arXiv:1803.09621, 2018.

[174] A. Vaswani et al., "Attention is all you need," In *Advances in Neural Information Processing Systems*, pp. 5998—6008, 2017.

[175] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–25.

[176] R. Li, et al., "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, 74429 – 74441, Nov. 2018.

[177] X. Chen, et al, "Multi-tenant cross-slice resource orchestration: A deep reinforcement learning approach," IEEE Journal on Selected Areas in Communications, vol. 37, no. 10, pp. 2377–2392, 2019.

[178] C. Qi, et al., "Deep reinforcement learning with discrete normalized advantage functions for resource management in network slicing," *IEEE Communications Letters*, vol. 23, no. 8, pp. 1337–1341, Aug. 2019.

[179] J. S. P. Roig, D. M. Gutierrez-Estevez, and D. Gunduz, "Management and orchestration of virtual network functions via deep reinforcement learning," *JSAC*, in press, DOI: 10.1109/JSAC.2019.2959263, 2019.

[180] R. Mijumbi, et al., "A connectionist approach to dynamic resource management for virtualised network functions," in *Proc. 12th IEEE/IFIP/ACM Int. Conf. Netw. Service Manag.*, 2016, pp. 1—9.

[181] A. S. Jacobs, "Artificial neural network model to predict affinity for virtual network functions," *IEEE/IFIP Network Operations and Management Symposium*, 2018.

[182] P. Velickovic, et al., "Graph attention networks," in *Proc. of ICLR*, 2018.

[183] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," In *Proc. of CVPR*, pp. 770-–778, 2016.

[184] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," In *International Conference on Machine Learning*, pp. 448—456, 2015.

[185] R. J Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8 no. 3–4, pp. 229–256, 1992.

[186] V. R. Konda and J. Tsitsiklis, "Actor-critic algorithms," In *Proc. Advances in Neural Information Processing Systems*, 2000, pp. 1008–1014.

[187] N. Lei, et al., "A Geometric View of Optimal Transportation and Generative Model," *arXiv preprint*, arXiv:1710.05488v2, 2017.

[188] Di. P Kingma and J. Ba, "Adam: a method for stochastic optimization," In *Proc. ICLR*, 2015.