A Study on Efficient Service Function Chain Placement in Network Function Virtualization Environment

by

Yansen Xu

A dissertation submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy in Engineering

Department of Informatics
Graduate School of Informatics and Engineering
THE UNIVERSITY OF ELECTRO-COMMUNICATIONS

June 2020

A Study on Efficient Service Function Chain Placement in Network Function Virtualization Environment

by

Yansen Xu

APPROVED by SUPERVISORY COMMITTEE:

Chairperson:	Prof. Toshihiko Kato
Member:	Prof. Hiroshi Yoshiura
Member:	Assoc. Prof. Mitsugu Iwamoto
Member:	Assoc. Prof. Celimuge Wu
Member:	Assoc. Prof. Satoshi Ohzahata
Member:	Assoc. Prof. Ved Prasad Kafle

仮想ネットワークにおけるサービスチェインの効率的な配置に関する研究

Yansen Xu

概要

ネットワーク機能仮想化(NFV)はネットワーク機能を専用のハードウェア装置から分離して仮想サーバーに移動し、複数の機能を汎用の物理サーバーに集約できる。このアプローチにより、ソフトウェアによる動的なネットワークの構成変更を可能とし，通信事業者の機器や運用のコストを低減することが期待されている。NFV より、ネットワークサービスはネットワーク機能を鎖状に接続したもの（SFC）を 1 つのサービスとして提供することが可能となる。SFC 配置は，NFV におけるネットワークサービス構築のコーアな技術である。しかし、SFC 配置問題は困難である。限られたネットワークリソース（CPU や帯域幅）とネットワークサービスパフォーマンスの要求を満たすため、適切に設計された SFC 配置手法が必要である。また，QoS を向上させるためには，分散構築やネットワーク遅延などの要求が考慮されるべきである。本論文では、コスト、エンドツーエンド遅延および分散配置の要求を満たした SFC 配置問題は課題とし、数理モデルと配置アルゴリズムを提案する。

コストはネットワークオペレーターにとって重要な側面であると考え、セットアップコストと運用コストを最小化する目的で数学モデルを作成した。セットアップコストと運用コストはより現実的な定義した。提案した数理モデルは先行研究と比較し、平均で最大 30％のコストを削減することが示された。さらに、提案したモデルでは、リアルタイムアプリケーションなどの SFC にとって重要な要求であるエンドツーエンド遅延デッドラインを保証することも考慮している。実験結果は提案したモデルが SFC リクエストのエンドツーエンド遅延デッドラインの要求を満たすことができることをしめしている。

エンドツーエンドの遅延が SFC にとって重要な QoS メトリックであり、SFC のエンドツーエンドの遅延が短いとユーザーエクスペリエンスが大幅に向上することを考慮している。本論文では、エンドツーエンドの遅延要求に対して、短い遅延対応できる SFC 配置アルゴリズムを提案した。先行研究と異なり、提案した配置アルゴリズムは、動的計画法を用いて、ローカル情報ではなくグローバル情報を探索して記録し、より短いエンドツーエンド遅延パスを算出することが可能となる。提案した配置アルゴリズムと先行研究における配置アルゴリズムを実装し評価を行った。その結果、提案した配置アルゴリズムが様々な物理ネットワーク上により短いエンドツーエンド遅延を達成することが示された。そして、ランダムな配置手法と比較したら、物理ネットワークモデルは JPN48 の場合、提案し

た配置アルゴリズムがエンドツーエンド遅延を最大 81.95％削減し、他の物理ネットワークモデルの場合、エンドツーエンド遅延を 50％以上削減したことを示した。その上、提案した配置アルゴリズムは、SFC リクエストに対して高いアクセプト率を達成した。本論文提案した配置アルゴリズムは短いエンドツーエンド遅延と高いアクセプト率が達成し、より優れた QoS 保証と効率的なネットワークリソース使用ができることを示している。

同一 SFC における複数の VNFs を一つの物理ネットワークノードに集約することにより、リソース使用量の過負荷、トラフィックループ、長い復旧時間などが発生する可能性があると考慮し、同一 SFC における VNFs を分散配置する必要性を検討した。本論文では、分散配置要求を備えた配置アルゴリズムを設計した。Layer graph というアプローチを用いて、分散配置要求を満たした上、エンドツーエンド遅延が短くすることが達成した。提案した配置アルゴリズムと先行研究における配置アルゴリズムを実装し評価を行った。その結果、提案した配置アルゴリズムが様々な物理ネットワークモデル上、分散配置要求を 100％満たすことができることを示している。さらに、提案した配置アルゴリズムは先行研究における二部グラフの最大マッチングに基づいた配置手法とランダム配置手法とそれぞれ比較した。物理ネットワークは JPN48 ネットワークとランダムグラフネットワークとし、提案した配置アルゴリズムは、二部グラフの最大マッチング配置手法より、それぞれのエンドツーエンド遅延を最大 46％と 60％削減でき、ランダム配置手法より、それぞれのエンドツーエンド遅延を最大 38％と 60％削減できる。本論文提案した配置アルゴリズムは分散配置要求を満たすこととエンドツーエンドの遅延の大幅な削減することが可能とし、適切な QoS 保証ができることを示している。

A Study on Efficient Service Function Chain Placement in Network Function Virtualization Environment

Yansen Xu

Abstract

Traditional network is equipped with many proprietary hardware-based network functions. These hardware-based network functions lead to network ossification which results in lack of flexibility and high cost to network operators. Network function virtualization (NFV) is a promising technology to solve this ossification problem in traditional network. NFV separates software implementation of network function from hardware. The software implementation of network function is called virtual network function, which can be installed and run on generic servers. A service function chain (SFC) in NFV is a set of ordered virtual network functions (VNFs) for end-to-end network service. The core technology to realize network service in NFV is SFC placement. SFC placement deals with placing VNFs of an SFC onto which hardware server, and chaining these VNFs along which edges in the hardware platform while satisfying particular requirements of SFC. Due to limited network resources and requirements of network service performance, network operators must carefully design scheme of SFC placement to efficiently utilize network resources and guarantee network service performances. In this thesis, we focus on SFC placement problem with satisfying minimized cost, optimal end-to-end delay, and distributed placement requirements. These requirements are critical to network operators constructing network service, and end users using network services. We contribute a mathematic model and algorithms to solve these problems.

From the aspect of cost, considering cost is one of critical aspects to network operator, we formulate a mathematic model with objective of minimizing setup costs and operational costs. With a realistic definition of setup cost and operation cost, our model can reduce cost up to 30% in average. Moreover, our model also considers guaranteeing the end-to-end delay deadline, which is a critical requirement to SFCs, such as real-time applications. Our model can satisfy 100% end-to-end delay deadline requirements of SFC requests.

From the aspect of end-to-end delay, it is one of critical QoS metrics to SFC. A shorter end-to-end delay to an SFC can significantly improve the user experience. Being aware this, we propose a delay aware SFC placement algorithm with achieving shorter end-to-end delay requirement. Unlike previous works, our algorithm explores and records global rather than local information to obtain a shorter end-to-end delay path. Evaluation results show that our algorithm achieved a maximum 81.95% reduction of end-to-end delay comparing with randomly deployment of SFC on JPN48 network model, and more than 50% reduction of end-to-end delay on the other network model. Besides, our algorithm achieves a good performance in terms of acceptance rate. The short end-to-end delay and high acceptance rate indicate that our algorithm can achieve a better QoS guarantee and efficient network resource utilization.

From the aspect of distributed placement requirements, we perceive that consolidating VNFs of an SFC onto a substrate network node will potentially lead to overload resource usage, traffic loops, long recovery time, etc. Motivated by this, we design an algorithm with distributed SFC deployment requirement, which is merely touched by previous works. Evaluation results show our proposed algorithm can 100% satisfy distributed deployment requirement on different network models. Besides, our algorithm can achieve a maximum 46% and 60% reduction of end-to-end delay by comparing with bipartite matching algorithm on JPN48 and random graph network models, respectively. Our algorithm achieves a maximum 38% and 60% reduction of end-to-end delay by comparing with random algorithm on JPN48 and random graph network models, respectively. The 100% satisfaction of distributed deployment requirements and high reduction of end-to-end delay indicates that our algorithm can perform a good QoS guarantee.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

1. Introduction

1.1. Traditional networking

In 1960s, ARPANET[1] proposed the idea of earliest computer network. With decades of development, computer network has been one of the most important infrastructures to modern society. Almost every activity in the world is depending on computer network, from ordering a pizza or watching movie, to control a factory.

At the very beginning, computer network was equipped with essential forwarding devices, such as switches and routers, for exchanging information traffic between end users. The traffic between end users were forwarded with merely touched.

Nevertheless, as numerous applications and services, such as online banking, are emerging in the network, for security but also many needs, many **middleboxes** [1][26] are introduced into the network. These middleboxes are also known as **network functions** (NFs). For instance, a firewall blocks unallowed packets to improve the security of the network; an intrusion detection system (IDS) is actively monitoring network traffic for alert of malicious activities and attacks; and a network address translator (NAT) hides internal devices as if it were the only device connecting to the network.

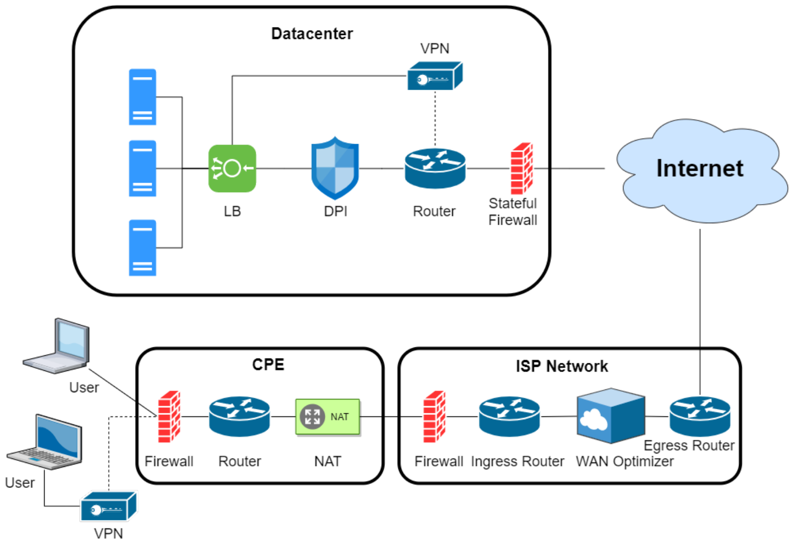

Figure 1-1 Network composed of forwarding devices and middleboxes.[2]

[1] ARPANET: The Advanced Research Projects Agency NETwork

Figure 1-1 illustrates a realistic network composed of both forwarding devices and middleboxes. End user devices connect to a Customer Premise Equipment (CPE) provided by Internet Service Provider (ISP) in the homes. The traffic travels through these middleboxes, such as firewall and NAT, and arrives at ISP network. The ISP network may also consist many middle boxes, such as firewall, WAN (Wide Area Network) Optimizer, which increases data transfer efficiency.

Another scenario of middleboxes deployment is in datacenter as shown in Figure 1-1. A datacenter is composed of a collection of servers, most of which provide network services. The firewall in datacenter can limit the number of connections from Internet to prevent Denial of Service (DoS). Deep Packet Inspector (DPI) analyzes the packet content to check if it is an attacking packet or banned content that should be rejected. The Load Balancers (LB) distributes the traffic to multiple servers to spread the load.

Above is only a limited non-exhaustive list of examples and scenarios for middleboxes. Middleboxes have been playing an important role to network. Previous studies show that the number of middleboxes in datacenter and enterprise networks have been similar to the number of forwarding devices [3][4][5]. For instance, a large network may contain 2850 L3 routers and 1946 total middle boxes [5].

Normally, middleboxes are implemented in a way of integration of a dedicated hardware and a dedicated software. The hardware of a middle box receives packets, forwards and processes packets according to the commands of software. As the hardware is a dedicated device for a specific purpose, the software is thus programmed only for this dedicated hardware and cannot be installed on other types of hardware devices. In addition, multiple vendors would provide same network function with different hardware and software, but the software and hardware are not interchangeable among vendors due to the integration of dedicated software and hardware.

However, this integration of dedicated software and hardware for a middlebox introduces multiple issues.

- **High Capital Expenditure (CAPEX):** The network infrastructure is expensive. The deployment of network devices is costly and requires network operator high investment on hardware. According to a survey on 57 enterprise networks in [5], the number of middleboxes is almost equal to the number of forwarding devices in a network. A medium network which consists 1000 to 10,000 servers may invest 50,000 to 500,000 dollars on hardware. A large network which is composed of more than 100k servers may require millions of dollars to invest hardware infrastructure.

- **High Operational Expenditure (OPEX) and high manage complexity**: The hardware-based network function devices are provided by multiple vendors and require large number of high-skilled labors to manage the complex heterogeneous devices. Even a small network contains 10 number of middleboxes may require 6-25 number of personnel according to the survey in [5]. Besides, as network functions are hardware-based and are deployed on the underlying infrastructure, to modify the topology of current network service, such as adding/removing network functions, it requires to reconfigure the underlying physical topology, which results in a high complex and error-prone reconfiguration [38]. Moreover, as stated in [39], management of middleboxes is high complex due to carefully design network topologies, manually setup rules for routing traffic across a specific order of network devices and implement safeguards to assure the correctness of operation of overload or failures.
- **Lack of scalability and flexibility:** Since the network functions are strong coupling with the physical topology, it is very difficult to reconstruct the network to adapt to a new topology or services [27]. To change the logical network topology of the service requires changes on configuration [40]. For example, inserting an SSL box for security before a web server needs to identify the point that all web traffic travel through, and requires personnel to setup the box at a location which needs extra space, power, and configuration. Moreover, as the network functions are vendor-specific, it will take a long time and high cost to upgrade the network functions or add new features to adapt to new services or business.
- **Inefficiency resource utilization:** The middleboxes in the network are deployed in an uncoordinated manner. They are developed for a specific need. The network resources are hard to be amortized across applications [4]. When constructing a network service, extra network resources must be provisioned in case the demand of resources inflates in future. The computation and bandwidth resources may be under-utilized due to changing of traffic pattern. As network functions are topology-dependent, it is impractical and leads a high operational cost to re-planning the chaining of network functions in a production environment [38].

1.2. New approach toward modern networking

As large amount of various middleboxes are populating in the network to satisfy needs of applications and services, network has been getting complex and involving many issues as mentioned above. To overcome these issues, ETSI [2] proposed **Network Function Virtualization (NFV)** [6][7].

NFV implements network services via a more programmable and flexible approach. NFV replaces the dedicated hardware of a middlebox with a generic hardware such as x86 server. The software of a middlebox thus can be installed on anyone of such generic hardware. Besides, by leveraging virtualization technology, multiple software can be installed onto a same server. In this way, a network function is no longer a dedicated hardware box integrated with software, but a software running on top of generic physical platform. This software-based network function is known as **virtual network function (VNF)**. As VNFs are software, they can be easily installed, terminated, and migrated among servers instantly according to the demands of network services.

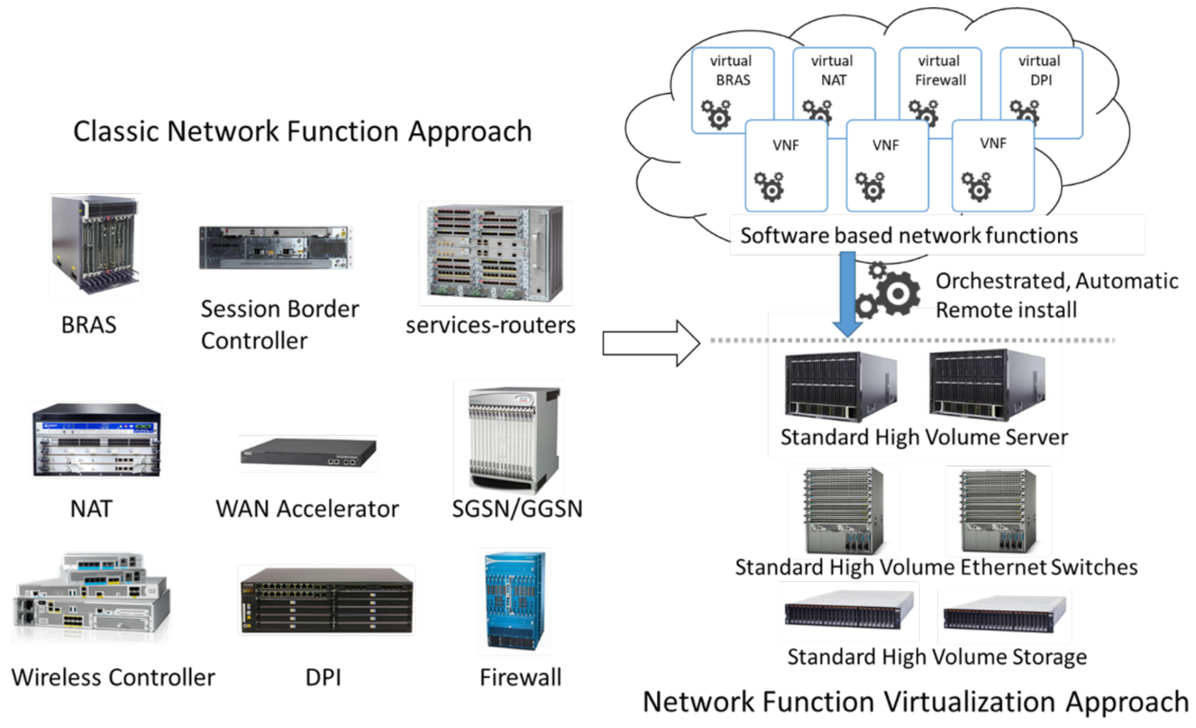
Classic Network Function Approach
BRAS
Session Border Controller
services-routers
NAT
WAN Accelerator
SGSN/GGSN
Wireless Controller
DPI
Firewall
virtual BRAS
virtual NAT
virtual Firewall
virtual DPI
VNF
VNF
VNF
Software based network functions
Orchestrated, Automatic Remote install
Standard High Volume Server
Standard High Volume Ethernet Switches
Standard High Volume Storage
Network Function Virtualization Approach

Figure 1-2 the transition from classic network function approach to the NFV based approach.

[2] ETSI: The European Telecommunications Standards Institute

Figure 1-2 shows the transition from classic network function approach to the NFV based approach. Left side of the Figure 1-2 are classic hardware based middle boxes. The software and hardware are coupled inside a dedicate hardware device. Each hardware device performs a specific network function, such as firewall, or DPI. Right side of Figure 1-2 are NFV based network function. VNFs such as virtual Firewall and virtual DPI are software installed on standard high-volume servers. These VNFs can be easily installed, terminated, removed, and migrated in an autonomous way.

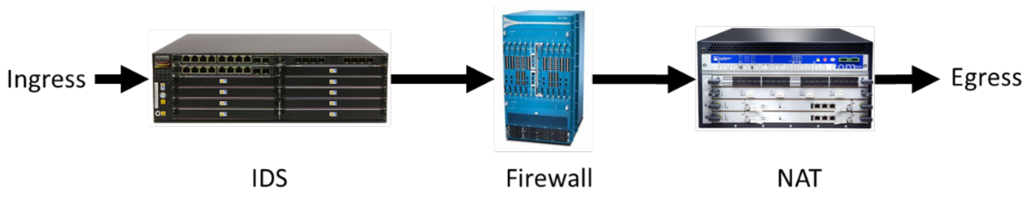

Figure 1-3 Hardware based network service.

Figure 1-3 illustrates an example of network service (network function chain) constructed based on classic network function approach. The network function chain is composed of three hardware-based network functions, IDS, Firewall, and NAT. Network traffic enters the function chain at ingress, travels through these three network functions, gets processed and finally leaves the function chain at egress. The logical links of network function chain are same to physical links, which are very hard to change the topology, for example, insert a new function to the chain.

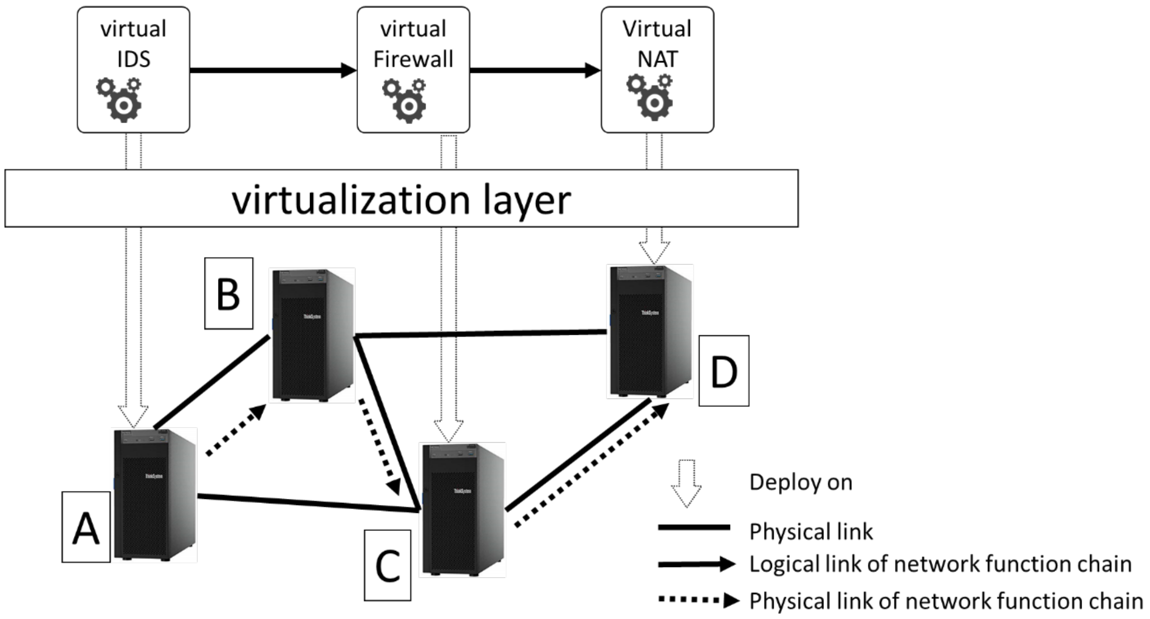

Figure 1-4 Network service in NFV.

Figure 1-4 demonstrates an example of network function chain based on NFV approach. This network function chain provides same functions as in Figure 1-4. Different from Figure 1-3, each network function in NFV is virtualized as an VNF and deployed physical network composed of four standard servers through virtualization layer. Specifically, IDS is deployed on server A, Firewall is deployed on server C, and NAT is deployed on server D, respectively. The logical links of the network function chain are different from the actual physical link. As shown in Figure 1-4, the actual physical link of network function chain is A->B->C->D. It is much easier to change the topology of network function chain, such as inserting a network function, in a way of installing a VNF on physical servers, and directing traffic flow through the server by leveraging technique such as Software Defined networking (SDN) that can control traffic flow in the network flexibly to any location by programming.

1.3. Research problem in NFV

In the previous sections, we introduced the transition from classic hardware-based network function approach to a flexible software-based network function approach. This new approach is known as Network Function Virtualization (NFV). In NFV, Network services consists a serials of Virtual Network Functions (VNFs) instead of hardware-based middleboxes. This serials of VNFs is called **Service Function Chain (SFC)**. Figure 1-4 illustrates an example of an SFC on top of virtualization layer in NFV. This SFC is composed of three VNFs: an IDS, a Firewall, and an NAT. Traffic is supposed to travel through these VNFs to be processed before arriving at destination.

In this context, initiating a network service (or SFC, same meaning in this thesis) is achieved by placing and chaining VNFs over a virtualization layer built on generic physical platform which consists generic servers, storage and networks. This introduces a new research problem known as **SFC deployment problem**. This problem deals with placing VNFs of an SFC onto which hardware server, and chaining these VNFs along which edges in the hardware platform while satisfying particular requirements of SFC.

To solve this problem, it requires an algorithm of VNF placement and chaining. This algorithm must be carefully designed with best strategy rather than a random process. Following describes the reason of necessity to such algorithm.

Reason 1: The resource of physical platform is limited. As multiple network services coexist on a same physical platform and share the resources, an arbitrary placement of

VNFs would result in computation resource (CPU) fragment in physical servers so that no VNF can use it, or overload of a physical link that cannot route any traffic between two VNFs in a network service. Deployment of SFCs should be carefully designed to efficiently use the physical resources.

Reason 2: The physical equipment must guarantee the performances of network services, such as end-to-end delay of network services. This indicates that VNFs of a network service must be placed and chained on the physical platform along a physical path that have a relative shorter end-to-end delay.

Due to these reasons, randomly deploying VNFs is not applicable. Designing an algorithm to deploy VNFs with best strategy is recognized as an SFC deployment problem as mentioned previous and is regarded as an NP-hard optimization problem[20][21].

In this thesis, we study this SFC deployment problem. SFC deployment is a core technology to realize network service in NFV. This problem has been studied from many aspects, but we still see that there are rooms to improve the previous works such as reduce delay of SFC, and some aspects such as distributed deployment requirements are merely touched by previous works.

Motivated by this observation, we study SFC deployment problem with satisfying particularly requirements of SFC. Specifically, the requirements in this thesis are cost reduction requirement, short end-to-end delay requirement, and distributed deployment requirement. We study how to deploy VNFs of SFC while simultaneously satisfying these requirements. Following summarized the scope of this research problem.

- **Cost reduction requirement in SFC deployment:** Cost reduction is one of benefit of NFV. Cost occurs when deploying VNFs onto physical platform and utilizing resources of physical platform. It is difficult to reduce cost as deployment of SFC is complex and involves many constraints. These costs should be carefully identified and defined.
- **End-to-end delay requirement:** End-to-end delay (forwarding delay) is one of critical QoS [22][23] metrics for network services. A shorter end-to-end delay will significantly impact on user experience in using applications such as conversational video or online game. Nevertheless, deploying SFC with a desired end-to-end delay in NFV environment is more challenge than that in traditional networking due to the distribution characteristics of physical resources in NFV. For example, deploying two VNFs of a

network services onto two generic servers which are far away from each other in the physical platform will results in a very long end-to-end delay to network services.

- **Distributed deployment requirement:** Distributed deployment requirement describes that VNFs of an SFC should be deployed on different generic servers in the physical platform to reduce risk of performance degradation. Collocation of VNFs of an SFC on a same generic server will potentially lead to overloaded resource usage, long recovery time, traffic loops, etc. Thus, it is necessary to consider to distributed deployment of SFC, i.e., distributed deployment requirement. Besides, the end-to-end delay should also be considered as distributed deployment would result a relative long end-to-end delay for network services.

1.4. Contributions of this study

The main purpose of this thesis is to provide a well-designed model and algorithms of SFC deployment problem with satisfying particular requirements. The main contributions of this study are summarized:

1) **Mathematic model for SFC deployment with reducing cost**

   The first contribution is a mathematic model that provides an exact solution which meets both end-to-end delay and order of VNFs constraints of an SFC while minimizing costs in SFC deployment process. The model reduces average 32% of overall costs occurring in the process of setup and operation of VNFs, achieves 100% end-to-end delay performance guarantee, in comparison of 50% guarantee performance of previous works.

2) **An end-to-end delay aware algorithm for SFC deployment**

   The second contribution is an algorithm to place and chain VNFs along a minimized end-to-end delay path for an SFC. The algorithm utilizes dynamic programming technique to find a relatively shorter end-to-end path for deployment of SFC. The algorithm also guarantees the order of VNFs of an SFC and satisfies the resource demands of SFCs. The algorithm achieved a maximum 81.95% reduction of end-to-end delay on JPN48 network model, respectively. On the other network model, the algorithm achieved more than 50% reduction of end-to-end delay. Meanwhile, the algorithm has a good performance with more or no less acceptance rate than other algorithms on different network models.

3) **A distributed algorithm for SFC deployment**

The third contribution is an algorithm of distributed placement and chain VNFs on the physical platform with a short end-to-end delay for SFCs. The algorithm leverages layered graph approach to find a deployment solution with a 100% satisfaction of distribution deployment requirements for SFC on different network modes. Meanwhile, the algorithm achieves a maximum 46% and 60% reduction of end-to-end delay by comparing with bipartite matching algorithm on JPN48 and random graph network models, respectively. Our proposed algorithm achieves a maximum 38% and 60% reduction of end-to-end delay by comparing with random algorithm on JPN48 and random graph network models, respectively.

1.5. Thesis outline

The remainder of the thesis is organized as follows:

In Chapter 2, we give an introduction on the related technologies of NFV, SFC, and state-of-the-art of SFC placement.

In Chapter 3, we present a mathematic model for SFC deployment problem. The model utilizes integer linear optimization technique to minimize the setup cost and operation cost which is more practical than previous works in literature. This model takes the computation and bandwidth resources as constraints, and it guarantees the end-to-end delay as well as the order of VNFs of an SFC.

In Chapter 4, we analyze the importance of a shorter end-to-end delay to service function chain. We design a heuristic SFC deployment algorithm based on the dynamic programming technique. This algorithm deploys service function chain with satisfying substrate network resource constraints, while simultaneously achieving a minimum end-to-end delay for service function chain. Experiments are conducted to show the superior of our algorithm in comparisons of other algorithms.

In Chapter 5, we perceive the necessity of distributed deployment of SFCs which can reduce the risk of degradation of network services. We present a distributed deployment algorithm by modifying layered graph approach. The proposed algorithm can avoid to deploying multiple VNFs of an SFC onto same substrate networks while simultaneously achieving a relative shorter end-to-end delay. Results of experimental studies on different algorithms show that our proposed algorithm outperform these algorithms in literature.

Finally, Chapter 6 draws the conclusions and describes the future work items.

Figure 1-5 illustrates the contribution and outline of this thesis.

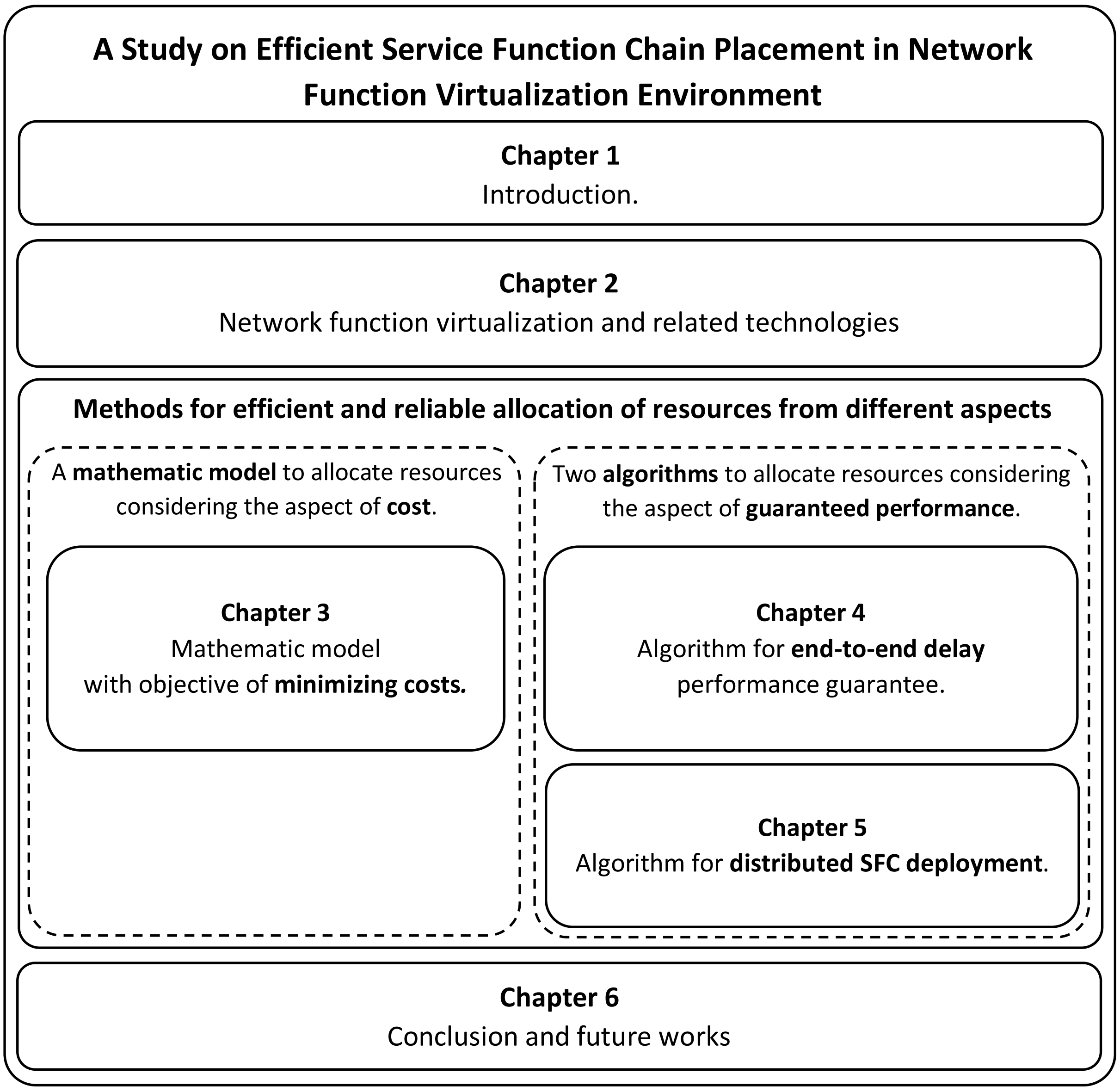
A Study on Efficient Service Function Chain Placement in Network Function Virtualization Environment
Chapter 1
Introduction.
Chapter 2
Network function virtualization and related technologies
Methods for efficient and reliable allocation of resources from different aspects
A mathematic model to allocate resources considering the aspect of cost.
Two algorithms to allocate resources considering the aspect of guaranteed performance.
Chapter 3
Mathematic model with objective of minimizing costs.
Chapter 4
Algorithm for end-to-end delay performance guarantee.
Chapter 5
Algorithm for distributed SFC deployment.
Chapter 6
Conclusion and future works

Figure 1-5 Contribution and outline of this thesis

Chapter 2

2. Network Function Virtualization and Related Technologies

2.1. Network function virtualization

Network function virtualization [6][7] aims to address the challenges in the traditional networks such as high cost and lack of scalability and flexibility due to dependence on proprietary hardware network functions. By leveraging virtualization technologies, NFV consolidates the software implementations of network functions onto standardized high-volume servers. NFV provides network operator a flexible and efficient management and operation to network and the capability to faster release new types of network services in lower cost. NFV changes the way that networks are designed, deployed and maintained by utilizing virtualized network functions (VNFs). As the software implementation of network function is no longer integrated into a dedicated hardware, both software and hardware can be evaluated independently which promotes the innovation of both hardware and software implement of network function. Furthermore, the separation of software and hardware allows network operators to deploy network functions flexibly by reassigning and sharing the same hardware platform, which can efficiently utilize the hardware resources and make new network service faster. Moreover, the network operator can dynamically scale the actual VNF performance according to the actual traffic or resource utilization [8]. Consequently, NFV potentially reduces CAPEX and OPEX and increases reliability and resilience and resource utilization [6][8][9][10].

NFV was first introduced by world's leading TSPs such as AT&T, BT, China Mobile, KDDI, NTT, and Verizon. The first version white paper [33] of NFV was released by European Telecommunications Standard Institute (ETSI) in 2012 to call for actions to work on the deployment of NFV. Since then, ETSI has released a series of white paper on the opportunities and challenges [33], use cases [34], architectural framework [8], industry progresses [35], terminology for main concepts [36] and NFV priorities for 5G [37]. As the significant benefit of NFV, both academy and industry are focusing on the research and development of NFV. Many standardize organizations such as NFV ISG[11], 3GPP[12], IETF SFC WG[13], ATIS[14] and BB Forum[15] are working on NFV. These standardize organizations have released many specifications on the development and standard of NFV. Besides, 93% of operators in a survey intend to integrate NFV [18]. Many ongoing projects such as OPNFV[16] and ZOOM[17] are working toward the deployment of platforms to promote NFV.

2.1.1. NFV reference architecture

Figure 2-1 shows ETSI NFV architectural framework[8]. The architectural framework consists of three main components: Network Function Virtualization Infrastructure (NFVI) component, Virtual Network Functions (VNFs) component, and NFV Management and Orchestration (MANO) component.

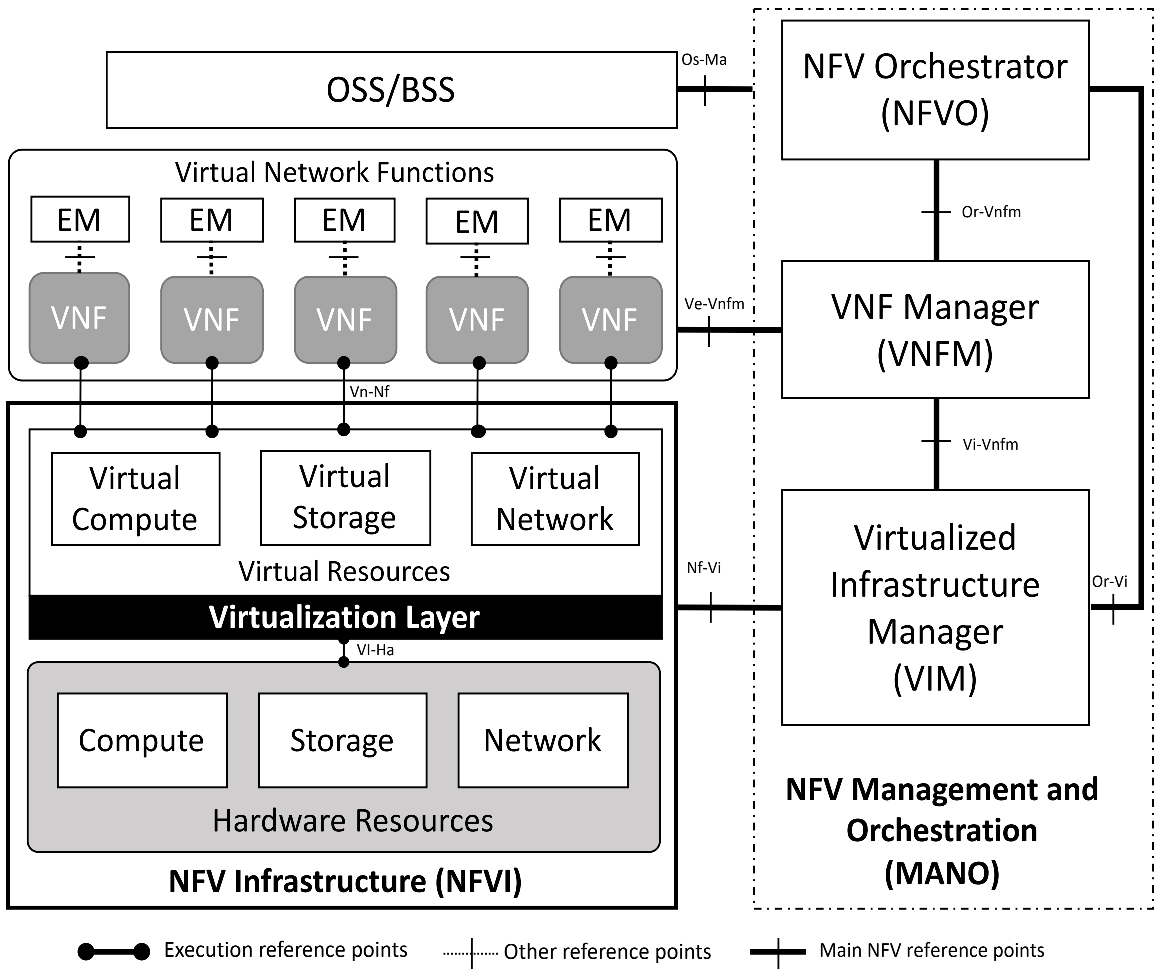

Figure 2-1 ETSI NFV architectural framework. [8]

2.1.2. Function blocks in NFV architecture

NFV Infrastructure (NFVI): NFVI is composed of hardware and software resources in NFV. Some example of such resources are server, network, storage, virtual machines, and hypervisors. NFVI virtualizes these resources to construct a virtualization layer where VNFs can be deployed and executed.

Virtual network functions (VNF): a VNF is a softwarized network function. By leveraging virtualization technology, a VNF can be packaged into a virtual machine and deployed on a specific server in NFVI. VNF can be started and terminated within few minutes and migrated across the NFVI to adapt to demand of network service. VNFs are managed by VNFM.

Element Management (EM): EM manages VNFs in collaboration with VNFM.

Management and Orchestration (MANO): MANO is composed of NFV Orchestrator (NFVO), a VNF Manager (VNFM), and a Virtual Infrastructure Management (VIM) component. MANO is delegated NFVO, VNFM, and VIM to manage the overall of NFV system.

NFV Orchestrator (NFVO): NFVO covers the orchestration and management of hardware and software resources in NFVI. NFVO realize the network services on NFVI, and is responsible for the network service lifecycle management, e.g., instantiate, scale, update and terminate the network service. NFVO controls the entire NFV infrastructure resources with the support of VIM and VNFM.

VNF Manager (VNFM): VNFM manages VNFs' lifecycle according to a deployment template. A deployment template describes requirements to realize a VNF. VNFM can perform functions such as instantiating, scaling, updating, and terminating VNFs. When instantiates a VNF, VNFM requests resources from NFVI based on the requirement in the deployment template, and NFVI assigns proper resources to the VNF. During the lifecycle of a VNF, VNFM monitors the VNF and scale resources for the VNFs accordingly.

Virtual Infrastructure Management (VIM): VIM is responsible for controlling and managing compute, storage, and network resources in the NFVI. For example, VIM can allocate or de-allocate compute and storage resources to a VNF. VIM collects infrastructure information for capacity planning, monitoring and optimization. VIM also monitors and reports infrastructure fault information. A VIM can manage multiple types of resources or be dedicated to handle a certain type of resource, such as compute-only VIM.

OSS/BSS: OSS/BSS are integrated with network operators' other systems. An OSS is an operational support system composed of a set of software applications that perform functions including service activation, service provisioning, network configure, network maintenance, and fault management. A BSS is a business support system composed of a set of software application that support customer-facing activities, such as processing billing, taking orders, collecting payments, and customer relationship management.

NFVI PoP: An NFVI PoP [36] is a single geographic location where a number of physical devices are sited. These physical devices provide computation, storage and networking resources and are responsible to provide NFVI functions to execute VNFs [30][31].

2.1.3. Reference points in NFV architecture

Reference points [32] are connections for functional blocks within and outside NFV-MANO to interact, communicate and work with each other for exchanging information.

Virtualization Layer - Hardware Resources (VI-Ha): Used in the middle of virtualization layer and hardware resource. The interface collects hardware state information and manage VNFs.

VNF - NFV Infrastructure (Vn-Nf): exchange information between VNF and NFVI. It ensures that the lifecycle, performance and portability requirement of the VNFs are independent from the hardware.

NFV Orchestrator - VNF Manager (Or-Vnfm): exchange information between NFVO and VNFM. This reference point supports resource related requests. It allocates and releases resources for VNFs. NFV Orchestrator can send configuration information to the VNF Manger to configure VNF according to the VNF Forwarding Graph. It supports the lifecycle management of VNF such as instantiation, query, update, scaling out/in, up/down, termination of VNF instance.

Virtualized Infrastructure Manager - VNF Manager (Vi-Vnfm): exchange information between VIM and VNFM. This reference point allocates/releases resources requested by VNFM. It exchanges the information of virtualized hardware resource configuration and state information.

NFV Orchestrator - Virtualized Infrastructure Manager (Or-Vi): exchange information between NFVO and VIM for reserving, releasing, updating, and allocating resources requested by NFVO.

NFVI - Virtualized Infrastructure Manage (Nf-Vi): exchange information between NFVI and VIM. It assigns computation resources and storage resources in response to resource allocation requests. It supports update resource allocation, migration, termination of VM. It also supports to create, configure and remove connections between VMs.

VNF/EM - VNF Manager (Ve-Vnfm): exchange information in middle of VNF/EM and VNFM. It manages lifecycle of VNF. It exchanges the information of configuration and events between VNF/EM and VNF Manager.

OSS/BSS - NFV Management and Orchestration (Os-Ma): exchange information in middle of OSS/BSS and NFVO. It manages NSD, VNF package, and lifecycle of NS, including instantiation, update, query, scaling, and termination of NS instance.

2.1.4. Information elements in NFV architecture

Information in NFV have three categories, information that resides in descriptors, information that resides in recorders, and information exchanged between functional blocks. Descriptors are deployment templates of static information for processing on-boarding virtual network functions and network services. Records are dynamic runtime data which represents virtual network functions and instances of network service [32].

An information element in NFV contains a specific information and can be built into a tree structure. An information element is classified into three types: leaf, which contains a specified value, reference elements, which carries a reference to another information element, and sub elements, which specifies another level in the tree.

To describe a Network Service, four information elements are defined: 1) VNF information element, 2) PNF information element, 3) VL (virtual link) information element, and 4) VNF Forwarding graph (VNFFG) information element.

Network Service Descriptor (NSD): An NSD is a network service deployment template, which refers to other descriptors that are part of the network service.

VNF Descriptor (VNFD): A VNFD is a VNF deployment template that describes requirements of deployment and operational behavior to a VNF. VNFM uses this template to instantiate VNF and manages VNF lifecycle. NFVO uses this template to manage NS and virtual resources on VNFI. NFV-MANO use the information of connectivity, interface and KPIs requirements for setting up Virtual Links within NFVI.

VNF Forwarding Graph Descriptor (VNFFGD): A VNFFGD describes network service's topology.

Virtual Link Descriptor (VLD): A VLD describes link resource requirements between VNFs in a Network Service.

Physical Network Function Descriptor (PNFD): A PNFD is a deployment template that describes Virtual Links requirements such as connectivity to an PNF.

By leveraging these descriptors with instantiation input parameters, the NFV Orchestrator or VNFM performs instantiation operation, and create record information such as NSR and VNFR as a result of the instantiation operation.

2.2. Service function chain (SFC)

Service function chain (SFC) is an end-to-end network service composed an ordered set of VNF for processing traffic flows [28]. The end-to-end network service is a combination of network functions [8], e.g., firewalls, HTTP header manipulation, and load balancers. The network service is represented as a Network Function Forwarding Graph [29] of a serials of connected network functions and end points. These network functions are deployed on a single or multiple interconnected operator network.

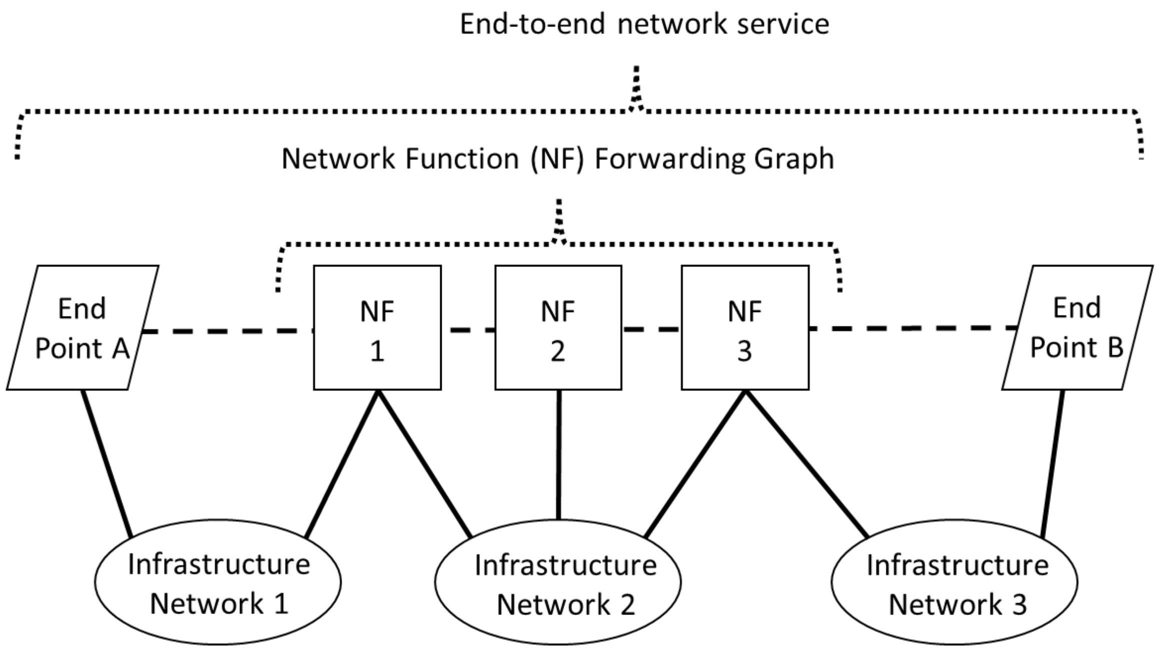

Figure 2-2 An End-to-end network service on the traditional hardware-based network.

Figure 2-2 shows end-to-end network service on the traditional hardware-based network. The end-to-end network service is composed of a network function forwarding graph, and two end points A, B. Network functions are deployed on the physical infrastructure networks. The dash line connected End Points and NFs are logical links of end-to-end network services. These logical links are connected by physical infrastructure network links. The end points are not included in the domain of network operator.

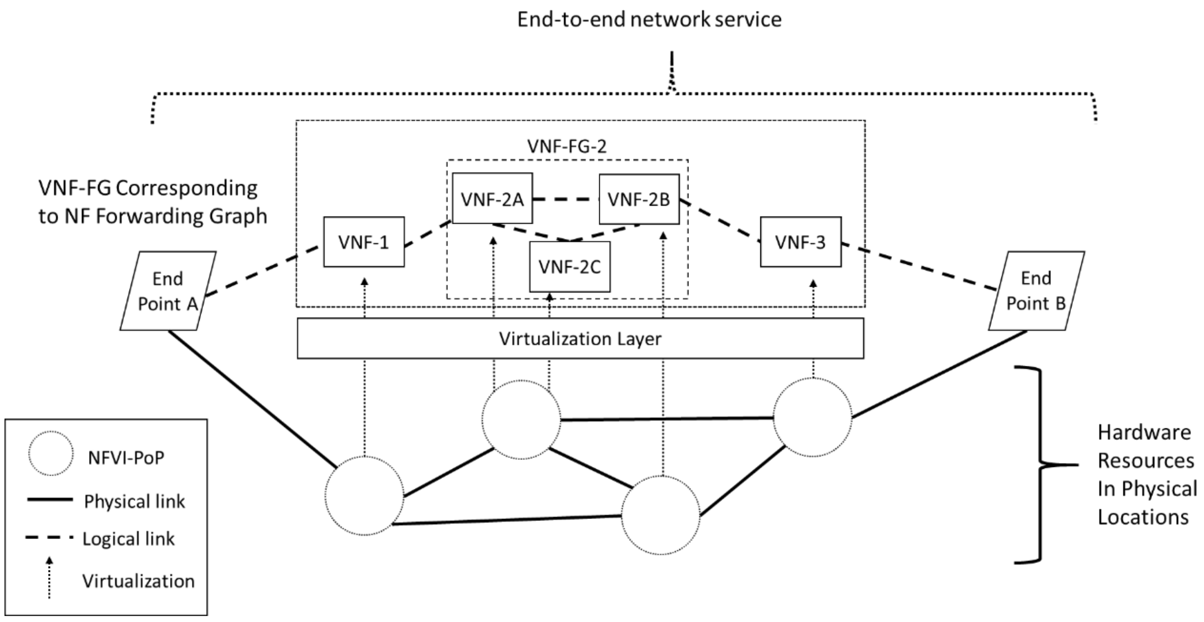

Figure 2-3 An example of an end-to-end network service realized by virtualization.

Figure 2-3 shows end-to-end network service realized by virtualization approach corresponding to Figure 2-2. The network functions are virtualized as VNFs and separated from hardware. VNFs are deployed on virtual machines provided by hardware resources of NFVI-PoP. From the view of end-to-end services, it does not know where the VNFs are deployed on the physical infrastructure.

2.3. SFC deployment scenarios

This section describes several scenarios of SFC deployment as below.

2.3.1. SFC in Broadband network

Service nodes can be deployed in broadband networks to add value-added services by network operator. Figure 2-4 shows a possible deployment of SFCs. The SFCs are deployed between BNG (Broadband Network Gateway) and CR (Core Router).

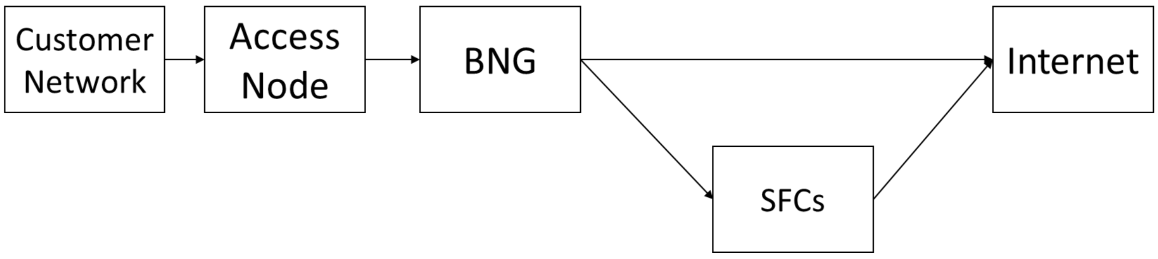

Figure 2-4 Service function chain deployment in broadband network scenario.

2.3.2. SFC in Mobile Networks Gi/SGi Interface

Figure 2-5 shows an example of SFC scenario in the Gi/SGi Interface. There are three SFCs in the figure. SFC1 is composed of an WAP GW. The DPI in SFC classifies the traffic and forwards WAP traffic to WAP GW. SFC2 is composed of an optimizer, a cache, a firewall and an NAT. DPI classifies HTTP traffic and directs the traffic through these functions in order. SFC3 is composed of a firewall and an NAT. Other traffic is directed by DPI to travel through these functions in order.

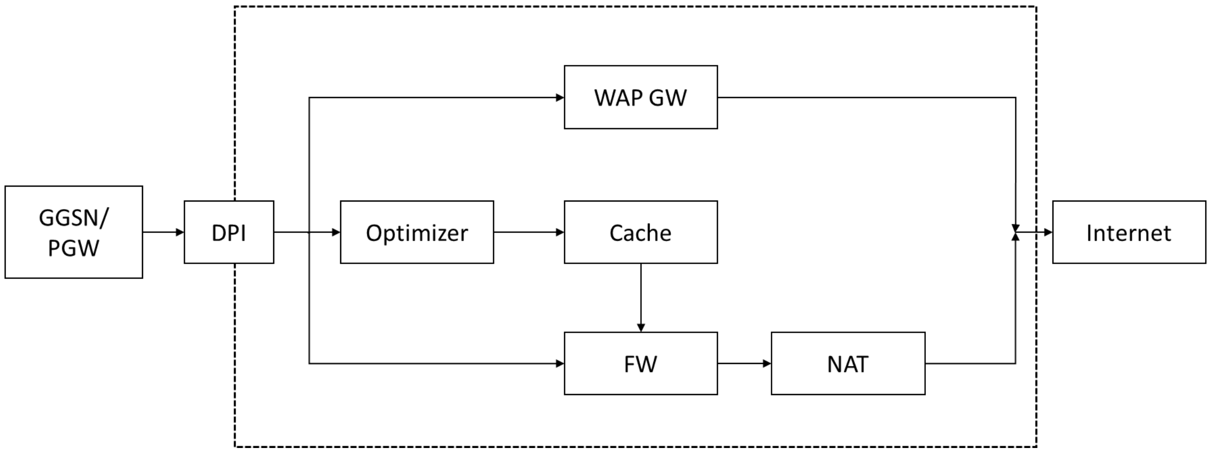

Figure 2-5 Service function chain deployment in mobile networks scenario.

2.3.3. SFC in Datacenter

Added-value services can be realized by deployment of SFC in Datacenter. Figure 2-6 depicts an example of SFC in Datacenter. The SFC is composed of an IDS/IPS, a firewall, an NAT. This SFC is deployed between the Servers and Datacenter route (DC) and processes all incoming traffic orderly.

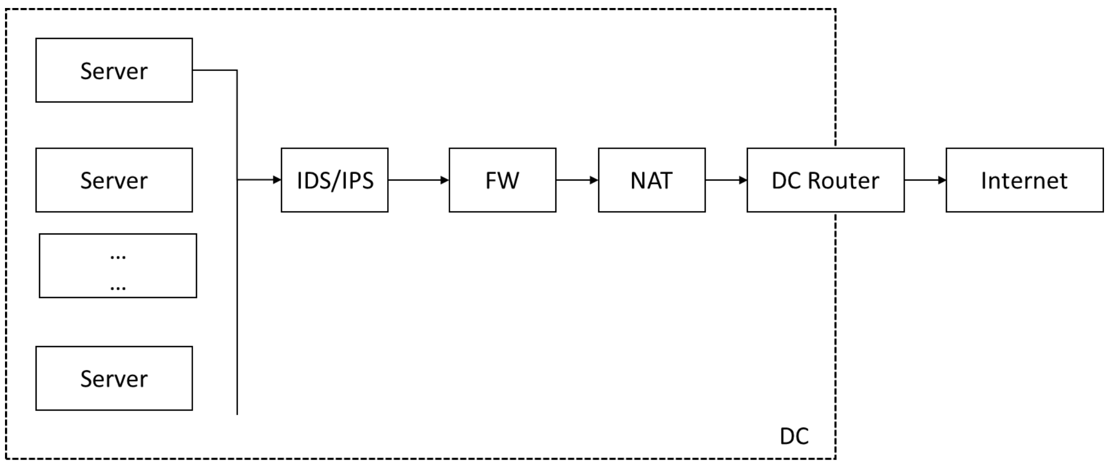

Figure 2-6 Service function chain deployment in datacenter scenario.

2.4. SFC placement and chaining problem

2.4.1. Overview of SFC placement and chaining problem

An SFC is composed of a set of ordered VNFs for processing traffic flow and delivery end-to-end network service. For example, securing flow can travel through an SFC composed of VNFs: firewall, DPI, and VPN. Though NFV addresses many issues in the traditional network, it arises additional challenges [10], and one of these challenges is VNF placement and chaining problem.

VNF placement and chaining is an important technology in the network function virtualization environment. It is a complex task since many constraints and requirements must be satisfied. It reflects the requirement of an SFC and how the network provider handles to satisfy these requirements, thus VNF placement and chaining can be regarded as resource allocation in NFVI. An algorithm of VNF placement and chaining is required to optimally deploy each VNF in an SFC request on a best location in the NFVI and interconnect these VNF instances orderly with satisfying resources and QoS constraints. Specifically, the algorithm must find location where have sufficient computation resources for processing VNF instances of an SFC and find a link who have sufficient bandwidth resources for routing traffic orderly to each VNF of SFC. QoS like delay and availability must be considered for SFC service delivery at same time.

Figure 2-7 depicts an example of SFC mapping over on NFVI. A NFVI contains six interconnected physical servers. Two service function chains have been mapped on the NFVI through the virtualization layer. SFC1 contains three VNFs: FW, IDS, and Proxy, between the src and dst nodes. This SFC can be deployed in the datacenter for enterprise. SFC2 contains a P-CSCF and a S-CSCF between src and dst node in IMS system [74]. The P-CSCF function is for call registration in IMS system which is located close to callers as the first contact point. S-CSCF is for session control in IMS [24]. From Figure 2-7, VNFs in SFC are deployed on physical server in NFVI, and traffic is routed through the servers that host the VNFs orderly. Note that VNFs from different SFCs can share the same physical server, e.g. IDS and S-CSCF. However, from the view of an SFC, each SFC has a fully controlled isolated resource. It no knowledge of physical server' location and the existence of other SFCs, as the physical resources are provided through the virtualization layer.

An SFC request is on-boarding to MANO. The NFVO receives the SFC requests in a form of NSD, that contains the information of the network service, such as number of VNFs, dependency of VNFs, computation and bandwidth resource requirement, and other information as describe in the information element section above. The NFVO perform instantiation operation in coordination with VNFM and VIM to install the VNFs on the NFVI and allocate proper computation and bandwidth resources for the SFC requests.

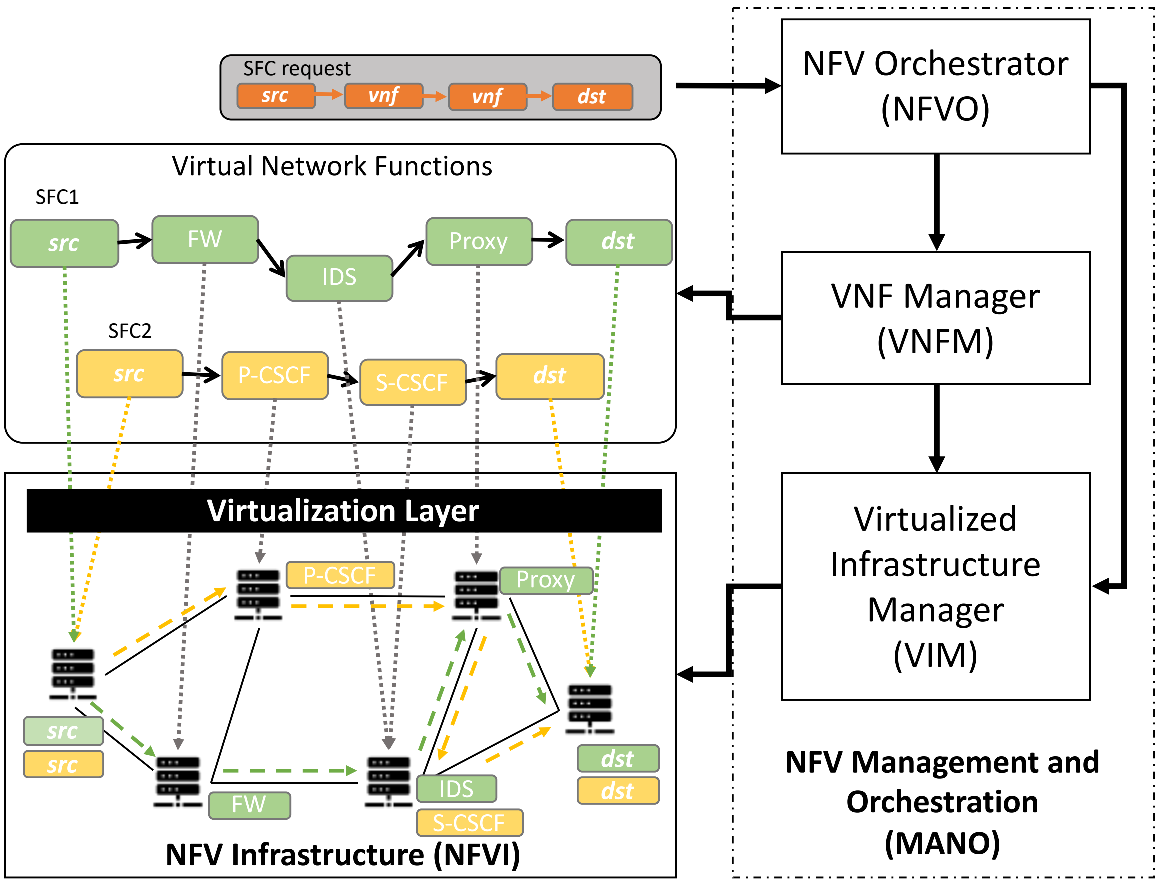

Figure 2-7 An example of SFC mapping over on NFVI.

2.4.2. Resource allocation taxonomy

When initializing an SFC on the physical infrastructure, the physical infrastructure provider must provide resources such as computation, bandwidth, and guarantee the performance for SFC to fulfill the end-to-end service deliver. We introduce the terminologies in the resource allocation for network slice.

VNF: A softwarized network function that can be installed onto a standardized generic server (e.g. x86 server) to realize the network function. For example, a softwarized firewall (vFirewall) is installed in an x86 server to act the firewall network function, rather than the dedicated firewall devices.

Substrate network: A substrate network is a physical infrastructure composed of a set of nodes interconnected through links. These nodes and links provide resources such as CPU and bandwidth. CPU and bandwidth capacity in this thesis are static. An example of substrate network is a collection of connected standardized generic physical device (e.g. x86 servers) in datacenter. A substrate network is regarded as NFV Infrastructure (NFVI) in NFV architecture.

Substrate network node / substrate node: A substrate node is a physical computation device. Substrate nodes are connected via substrate links (physical links). VNFs can be installed in the substrate node in a form of virtual machine, thus multiple VNFs can be installed on one substrate node. A substrate node provide computation resource (CPU) for VNFs installed in it to realize the functions of VNFs. A substrate node can receive and forward traffic flow acting like a switch (via technologies such as vSwitch).

Substrate network path / substrate path: Substrate network path is a physical link connecting multiple substrate nodes in a substrate network. Traffic travels on the path and through particular substrate nodes which host VNFs to realize end-to-end services. For a certain SFC, the substrate network path is also called end-to-end path.

Service function chain (SFC): an SFC consists a set of VNFs aiming to provide specific end-to-end service. An example of an SFC can be a chain of Firewall, IDS, and Proxy.

SFC request: an SFC with resources demands and performance requirements. An SFC request is created by a tenant or custom to request network resources from substrate network for deliver its particular end-to-end service. Resource demands describes the amount of resources the substrate network must provide. For instances, CPU demands for processing each VNFs of the SFC, and bandwidth demands for connecting two VNFs. In this study, we consider the resource demands are static. Performance requirements describe the performance objectives that the substrate network must guarantee. Some SFC requests require the VNFs processed in a specific order by substrate network, in order to realize the whole service correctly. The substrate network must allocate proper resources to satisfy resource demands and guarantee the performance of SFCs.

2.4.3. Resource allocation for SFC

To support multiple network slice coexisting on the top of a shared underlying substrate network, the substrate network must allocate resources to each network slice properly. From the view of substrate network, allocating resources for SFC can be regarded as a processing of deployment of SFC request. We consider two types of resource allocation, CPU resource allocation, and bandwidth resource allocation.

CPU resource allocation: CPU resource allocation deals with finding a location (substrate network node) whose CPU resource is sufficient to host the VNF of an SFC request. That is, to host a VNF, the amount of available CPU resources of a substrate network node should be larger than the CPU demand of the VNF.

Bandwidth resource allocation: bandwidth resource allocation deals with finding a substrate network path whose bandwidth resource is sufficient to direct traffic through the VNFs of an SFC.

2.5. State-of-the-art

This section presents state-of-the-art in the field of SFC deployment.

2.5.1. Hybrid of deployment of VNFs.

Hybrid of deployment of VNFs with physical network function is the first step for shifting traditional network to NFV based network. Hybrid of deployment can be classified into several aspects such as hybrid of different types of SFCs, hybrid of virtual network functions with physical network functions, and hybrid of deployment of VNFs in cloud and fog systems.

Zheng et al. in [41] investigated the problem of deployment of a hybrid SFC. The authors identified three types of SFCs, u-SFC, in which traffic is forwarded through ordered VNFs in one direction, b-SFC, in which symmetric traffic flows travel forward and backward through VNFs, and h-SFC, which is a hybrid of u-SFC and b-SFC, where some VNFs in the SFC need to process bidirectional traffic, and other VNFs process unidirectional traffic. Authors modeled the hybrid SFC deployment problem and proposed an approximate algorithm. The approximate algorithm first constructs a unidirectional service function path (SFP) and finds the shortest path to connect forwarding SFP and backward SFP.

Moens et al. in [42] presented a formal model for VNF placement which focus on a hybrid of physical network function and virtualized network function scenario. The scenario assumes that traditional hardware-based network function process normal traffic load, and VNFs process the variation of traffic load by dynamically installing VNFs. Huang in [43] observe that the collaboration of VNFs and physical network functions would be lasting a long term. The authors apply Markov Approximation (MA) based algorithm to efficiently deploy SFCs with satisfying traffic demands and individual policy chains simultaneously and maximize the total admitted traffic in the hybrid of VNFs and physical network functions environment.

Mouradian et al. in [44] and Afrasiabi et al. in [45] investigated the VNFs placement and migration in a hybrid of cloud and fog environment. Authors minimize the aggregated weighted function of makespan and cost with assumption of non-deterministic application graph. The authors formulate and evaluate an ILP model in a small-scale scenario by CPLEX optimization tool.

2.5.2. VNF interference and distributed deployment of VNFs.

Different types of VNFs will compete resources such as CPU or network I/O, if they are placed on the same hardware device. This competition will degrade VNFs' performance. Some researchers studied how the performance of VNF would be interfered among VNF instances deployed on the same server.

Zhou et al. in [19] studied vertical scalability for VNF placement. Authors consider a traffic flow data rate can alternated between normal rate and peak rate. The authors formulate a model for the problem and analyzed the reason that two VNFs in an SFC should not be placed on the same substrate node due to high packet loss ratio. The authors distributedly deploy VNFs by utilizing a bipartite matching algorithm. Zhang et al. in [46] investigated the interference problem and found that service performance will be degraded in 5G network slicing scenario, if VNFs are deployed on same place. The authors quantify the extent of VNF interference based a demand-supply model. The authors proposed an interference-aware algorithm for placement of VNFs in 5G slicing automatically. Veddy et al. in [47] observed that consolidation of VNFs would increases Snort's [48][49] response time. For example, the response time of Snort is around 0.18ms if it runs alone in the server, whereas the response time increases to 0.3ms if Snort runs with PktStat [50], and 0.4ms if multiple Snorts run in the same server. The authors proposed an algorithm to efficiently selects the VNFs and route the traffic flow with consideration of interference effect by utilizing dynamic programming technique. Zeng et al. in [51] measured performance interference by utilizing six types of VNFs. The authors investigated how the resource competition and packet characteristics would affect the performance interference. Based on the results of measurement, the authors provide suggestions on how to efficiently place VNFs from the perspectives of resource allocation, location selection and dynamic scaling. Kannan et al. in [52] identified the challenge of performance interference caused by visualization in datacenter environments. To address this challenge for production datacenters, authors designed Proctor, which can monitor the performance of physical infrastructure in a lightweight and scalable way.

2.5.3. Availability

Availability is one of key metrics of QoS of network services. Availability is more challenge in NFV environment as the network services could suffer completed service outage from both hardware failure and software failure.

To tackle availability problem, Moualla et al. [53] designed a placement algorithm based on an iterative linear program (LP). The algorithm leverages properties of fat-tree topology to realize online SFC placement. The algorithm has no any prior knowledge about the demand distribution. The algorithm places VNFs of SFCs considering availability which is requested by users. Herker et al. [54] analyzed different data-center topologies and provided different backup strategies for service function chains. By comparing costs in different network architectures and service availability performance, the authors observed a service chain can obtain a higher availability in a two-tier tree topology than a three-tier tree topology as a two-tier tree topology have more paths for backup. Kong et al. [55] proposed a mapping mechanism to guarantee SFC availability by deciding the number of VNF replicas that are necessary to each VNF. In such way, the mechanism can maximize the availability metric and guarantee the dependencies of VNFs. Yala et al. [56] designed an SFC placement algorithm which maximizes availability of SFC while simultaneously minimizing access latency for SFCs in MEC. The authors formulated a model considering availability, cost and latency, and proposed a genetic algorithm (GA). The authors evaluated the quality of solution and computation complexity by using IBM ILOG CPLEX Optimizer. Fan et al in [57] analyzed the challenges of availability issues in datacenter that different devices such as server, virtual machine, core switch could suffer failure and lead outage of SFCs. Authors proposed a VNF deployment scheme to minimize amount of resources by estimating SFC requests availability.

2.5.4. Energy

Existing studies show that in comparisons of traditional network, energy consumption can be saved up to 50% in NFV environment.[58]. To benefit advantages of NFV from efficient energy consumption, some researches focus on the energy-efficient deployment of SFC in NFV.

Xu et al. in [59] formulated an ILP model for energy-saving model and designed an algorithm by jointing optimization of VNF placement and traffic routing by utilizing Markov approximation technique to solve the problem in a polynomial time complexity. The results show the algorithm can save 14.84% energy consumption in telecom networks. Zheng [60] considered that a substrate network node can only provide a unique network function, which is practical in mobile edge computing system that the computation resources are limited. Authors formulated the problem into an ILP problem and proposed an efficient heuristic algorithm which can achieve a near-optimal performance in a small-scale network.

Mebarkia and Zsóka in [61] considered both the topology of network functions, network, and current load information of the underlying IP layer to avoid link overload in the underlying IP network layer. Authors classified SFC solutions into three architecture models based on the level of information exchanging between service function and underlying networking.

Garg in [62] considered the utilization of CPU can affect the VNF delay in the substrate network and proposed a selection algorithm with guaranteed end-to-end delay of SFC requests by setting the threshold of CPU utilization.

2.5.5. SFC deployment in 5G

NFV is a main enabler of 5G [63]. Many prior studies consider SFC deployment in 5G scenario.

Bi et al. in [64] designed a MILP SFC placement model for ultra-low latency services. The model minimizes the total latency with consideration of the sources of latency from processing, queueing, transmission, propagation, and optical-electronic-optical conversion. Authors proposed a heuristic algorithm based on data rate for service latency minimization. The evaluation shows the algorithm can achieve at least 1.7 times acceptance ratio as the baseline approach. Harutyunyan et al. in [65] formulated an ILP model of SFC placement which associates user equipment (UE) in 5G mobile networks. Authors designed a heuristic algorithm to solve the problems in scale. Alameddine et al. in [66] consider the placement and scheduling of VNFs of SFC in 5G ultra-low latency network. Authors solve the SFC schedule orchestration problem by jointly addressing the mapping and scheduling of services to VNFs. Authors formulate the problem into MILP and design a game-theoretic algorithm called ENCHAIN to exploit a scalable solution.

2.5.6. AI-based approach deployment

AI-based VNFs deployment is a promising way for network control and management as NFV paradigm improved the performability of network which could utilize AI technologies. Some related works have formulated VNF scaling and traffic predicting as an online learning (convex) optimization and solved by online-learning techniques [67][68][69]. Other related works leverage machine learning or deep learning for VNFs placement for efficient resource utilization.

Jia et al. in [70] addressed geo-distributed deployment of service function chain and scaling of VNF instances. Authors first formulate an online cost minimization problem for dynamic deployment and removal of VNFs across different datacenters, and dynamic traffic flow in each service function chain. By leveraging online learning algorithm of regularization technique, the algorithm relaxes the offline optimization problem. Finally, the author designed an online dependent rounding algorithm to obtain the final randomized mixed integer solution. Zhang et al. in [71] estimated traffic rate and adjusted VNF deployment in advance with a proactive approach. The authors solve the problem of VNF scaling by combination of online learning algorithm. The algorithm minimizes the regret of error prediction and the cost of sub-optimal online decisions. Kim et al in [72] proposed a machine learning-based algorithm to predict VNF resource demands. The method is a type of recurrent neural network that uses attention and embedding techniques in conjunction with a Long Short-Term Memory model. This model can simultaneously predict resource demands for all VNFs in SFC with a higher accuracy. Quang et al in [73] transform the VNF placement problem as a Markov Decision Process, and utilize deep reinforcement learning algorithm to solve the problem.

2.5.7. Virtual network embedding

Virtual network embedding (VNE) problem is considered to perform embedding of virtualized network functions into a shared substrate network with satisfying the constraints of substrate node computation and substrate link bandwidth capacity. Though the VNE problem is similar to the SFC placement problem, the SFC placement problem is aiming to map different virtual network functions on substrate by compromising the given constraints such as computation and bandwidth resources. In our research, the SFC placement also considers keeping the order of VNFs in the SFC for service delivery while satisfying a guaranteed end-to-end delay or availability requirements.

Yu et al. [75] proposed a greedy algorithm to deploy virtual network nodes on substrate network by ranking nodes of network. The algorithm calculates the amount of available resources of a node both in substrate network and each virtual network request by multiplying the available CPU resources with available bandwidth of all adjacent links of this node. Then algorithm sorts virtual network request according to the revenues and mapping virtual network request with maximum revenues by mapping the node with maximum available amount in virtual network request on to the node with maximum available amount in substrate network such that satisfying the CPU constraint. Then the algorithm maps virtual network links by finding k-shortest path. The authors in [75] also considered multi-path approach to map virtual network link efficiently. Cheng et al. [76] map virtual network nodes by ranking nodes as well, but in another way of rank nodes. They proposed a NodeRank algorithm similar to PageRank with transition matrix of the Markov chain to rank node and discover the network topology. Once the nodes in substrate network and virtual network are ranked, the algorithm maps node in a way same as [75] that mapped nodes large resources of virtual network on to nodes with large resources of substrate network. Once all nodes have been mapped, algorithm maps virtual network links by finding k-shortest path. Authors in [76] also introduce a one-stage mapping of nodes and links to eliminate unnecessary consumption of bandwidth resources of the substrate network such as two very closed nodes in virtual network may be mapped on two nodes with a long distance in substrate network. The algorithm introduces breadth-first search tree to solve this problem. Chowdhury [77] proposed a coordinate node and link mapping algorithm by extend the substrate network to an argument graph with meta node and meta edge. The algorithm formulates the embedding problem as a mix integer programming (MIP). The algorithm then relaxes the integer constraints to obtain a linear program and devise the algorithms using deterministic and rounding techniques. This algorithm also considered the location of nodes in virtual network. Gong et al. [78] proposed a global resource capacity algorithm to rank nodes in network. The global resource capacity algorithm overcomes locality problem and “big island” problem. The locality problem is that a node only considers the resources of the node itself and its adjacent links, but not the global resources capacity when evaluating the node rank. The “big island” problem is that the load distribution in a network is imbalanced so that some nodes cannot be utilized properly though they may still have large amount of available resources.

2.6. Conclusion

In this chapter, we introduced a brief overview of the reference NFV architecture. We described the definition of different function blocks, references point between function blocks, and the information elements flowing in the NFV architecture. We introduced the end-to-end network service in NFV as well as network slicing concept. Following SFC deployment scenarios, we stated the SFC placement and chaining problem and challenges in the problem. Finally, we presented the state-of-the-art of SFC placement.

Although many researches have focused on the SFC placement problem from different aspects, such as energy, we still see the rooms that can be improved such as reducing end-to-end delay. Some aspects of this problem such as distributed deployment requirement have not been researched well yet. In following chapters, we fill these gaps by studying SFC deployment with requirements of cost, end-to-end delay and distributed deployment.

Chapter 3

3. Mathematic Model for SFC Deployment Problem

3.1. Introduction

Network function virtualization (NFV) shifts the way of design, management, and deployment of network services into a more flexible, automated and vendor-independent manner. NFV has been considered as a promising technique for next generation networking. Many network operators, vendors and standard organizations are cooperating together to work on the development of NFV [79][80][81][82].

Service function chain (SFC) is an important technique to realize network service in NFV environment. As described in the previous chapter, an SFC is a set of VNFs deployed on the substrate network. A traffic flow travels through VNFs of SFC, such as virtual load balancer, virtual proxy, virtual DPI, and virtual firewalls, before it arrives the destination.

One of important benefits from the NFV is flexible and operationally efficient network management and deployment, which reduce both CAPEX and OPEX for service operators and end-users [83][84]. For example, by dynamically adjusting the number of running VMs for hosting VNFs, the network operator can reduce cost in terms of energy consumption. However, to gain full benefits from NFV, the scheme of deployment of SFCs must be carefully designed. One of the important considerations from the network operator's perspective is to reduce the cost criteria. Optimizing deployment cost means finding proper substrate nodes for hosting VNFs at a minimum cost while simultaneously satisfying the demands and constraints of network service. The cost in this context can be various. One of such cost we consider is setup cost, which is the cost to initiate a VNF instance on a substrate node, and another is operational cost, which is the cost to process a unit of VNF computation requirement in substrate network. We discuss the details of both costs in section 3.3.

However, the SFC placement problem is complex, involving many constraints and requirements. Nevertheless, due to the flexible of deployment of VNFs across the distributed substrate network nodes, it is a challenge for network operators to reduce the total CAPEX and OPEX with an optimal VNF deployment scheme while simultaneously guaranteeing the QoS of SFC such as end-to-end forwarding delay.

In this chapter, we present a mathematic modeling for the SFC deployment problem. A mathematic modeling is a special class of decision problem where concerns with the efficient use of limited resources to meet desired objectives [85]. It provides an effective approach of solving many different types of optimization problems.

Specifically, we formulate an ILP optimization problem with an objective of minimizing setup cost and operational cost. We show that the setup cost and operational cost in this model is practical and realistic. A practical definition is critical for network operator to estimate the costs when planning and operating networks. We conduct experimental study on our model. The results show that our model can reduce costs and guarantee an end-to-end delay deadline requirement.

In section 3.2, we introduce and compare related works which solve the SFC deployment by leveraging mathematic modeling method.

In section 3.3, we define the setup cost and operational cost and show differences and superior of our definitions comparing with previous works in literature.

In section 3.4, we formally formulate a mathematic model with optimization objective of minimizing costs, while simultaneously satisfying network resource capacity, order of VNFs, and end-to-end delay of an SFC.

In section 3.5 and 3.6, we conduct experimental study on our model.

In section 3.8 we make a conclusion on this chapter.

3.2. Related works

NFV is a promising technology for reduction of CAPEX and OPEX for network operators by leveraging generic hardware as a platform for deploying softwarized network functions. Many previous studies have utilized the mathematical programming to model the SFC deployment problem with various optimization objective and different constraints.

Normally, previous studies on deployment of SFC with cost reduction objective consider three main aspects:

a) General cost[86][87], which refers to reducing the cost to deploy and operate physical resource on node and edge.
b) Consolidation of nodes[42][90][91][93], which refers to deploy VNFs on less number of substrate network nodes to reduce the potential cost consumption such as energy.
c) Network load[88][92][94], which refers to reducing the network bandwidth utilization.

Regarding the constraints of the optimization model, as the network resources are limited, most previous studies consider CPU resource and bandwidth resource as the main constraints of the optimization models. These constraints guarantee that the total allocated resources not exceed the capacity of substrate network resources.

However, some of the previous works such as Lin [86] and Zeng [87] only consider the resource constraints. Although these studies provided effective model for minimizing the cost of deployment of SFC, they neglected the QoS constraints imposed on the SFC. As a result, this lack of QoS constraints will significantly impair the experiences of network services. One of such QoS metric is the delay (also known as forwarding delay or end-to-end delay) of an SFC. The delay is calculated as the summation of edges delays that a traffic of an SFC travel through. As described in the next chapter, a shorter end-to-end delay can significantly improve the experiences of users using the system. Some of previous works such as Moens [42] and Addis [88] noticed this neglection and added delay of an SFC as an additional constraint. These works formulated the optimization model with objective of cost reduction while simultaneously guaranteeing a desired end-to-end delay and satisfying the resource constraints in terms of CPU and bandwidth resources.

Another critical constraint is the order of VNFs in an SFC. Order constraint should be considered carefully as order is one of the distinct features for SFC compared with problem such as virtual network embedding problem (VNE) described in section 2.5.7. The order constraint guarantees a traffic flow travelling through VNFs of an SFC in a specific order. This is critical since some VNFs must be processed before or after another VNFs for completion of whole network function. For example, intrusion detection system (IDS) needs to come before compression or encryption functions to inspect payloads, and encryption function should be located in front of a decryption function to perform correctly[25][89]. Some researchers such as Savi [90] and Tajiki [91] have noticed the order constraint and added this constraint in their models.

Table 3-1 draws a summary of the previous study on cost reduction by utilizing mathematical modeling. Lin et al. in [86] and Zeng et al. in [87] consider reduce the general cost, which refers to reducing the cost to deploy and operate physical resource on node and edge. Lin et al. in [86] proposed a MIP model for solving the deployment network functions and allocations of physical resources. The model is aiming to minimize the cost to deploy an instance of network function and reduce the cost to use physical resource on both nodes and edges. The model guaranteed the constraints of CPU and bandwidth resource capacity. However, they did not consider the order of VNFs and the end-to-end delay of an SFC. Zeng et al. in [87] studied the orchestration of tree-type VNF forwarding graph in an inter-datecenter elastic optical network scenario. The authors formulated a MILP model to minimize the overall cost to orchestrate VNFs. As the VNF forwarding graph is a tree-like one, order of VNFs and end-to-end delay of an SFC are not considered.

Researchers in [42][90][91][93] consider to consolidate the VNFs onto few number of substrate network nodes to potentially reduce the cost. Moens et al. in [42] designed a formal model for placing VNFs. The model focuses on a hybrid of physical network function and virtualized network function scenario. The scenario assumes that traditional hardware-based network function process normal traffic load, and VNFs process the variation of traffic load by dynamically installing VNFs. Savi et al. in [90] deployed VNFs on a smaller number of substrate network nodes, to minimize the number of active nodes. The model consists the CPU resource, bandwidth resource, order, and max tolerant end-to-end delay constraints. However, the model did not take the cost to deployment and utilization of resources into account. Luizelli et al. in [93] aims to reducing the number of VNFs. The authors argued that number of VNFs would impact the network providers' cost most significantly. The model the authors proposed are composed of CPU constraint, bandwidth constraint, order and end-to-end delay constraints. However, we consider that the cost to deploy and utilize the VNFs should also be counted as an important

Table 3-1 Mathematic modeling reference comparison table.

Works	Category	Objective	Constraints			
			CPU	BW*	Order	Delay**
Our model	General cost	**Min cost**	**Y**	**Y**	**Y**	**Y**
Lin[86]		Min cost	Y	Y	N	N
Zeng[87]		Min cost	Y	Y	N	N
Moens[42]	Consolidation of nodes	Min #PM	Y	Y	N	Y
Savi[90]		Min #active NFV nodes	Y	Y	Y	Y
Luizelli[93]		Min #VNF instances	Y	Y	Y	Y
Tajiki[91]		Min #forwarding elements	Y	N	Y	Y
Addis[88]	Network load	Min Max network link utilization/Min #cores	Y	Y	N	Y
Jang[92]		Min network resource usage	N	Y	Y	N
Gupta[94]		Min bandwidth consumption	N	Y	Y	N

* BW: Bandwidth
** Delay: End-to-end delay of SFC or forwarding delay of SFC.
Y indicates the model includes the corresponding constraint.
N indicates the model excludes the corresponding constraint.

aspect for reducing network providers' cost, which are not considered by the authors. Tajiki et al. in [91] proposed a scheme to reduce energy. The authors modeled an ILP optimization model to minimize the number of hops along the SFC path. The model considered the constraints of order of VNFs, delay of SFC, link resource utilization, and server resource utilization.

Other researchers in [88][92][94] consider to minimizing the network resource load. Addis et al. in [88] formulate a model as a MILP optimization problem with minimizing the maximum network link utilization and number of cores (CPU) subjected to the constraints of node resource capacity, bandwidth resource capacity, and latency bound. However, they did not explicitly formulate the order constraints in the formulation. Jang et al. in [92] formulated an optimization problem based on IETF SFC architecture and topology. The objective of this formulation is to minimize the network bandwidth resource usage by determining the ratios of flows. The authors considered the constraints of flow conservation, bandwidth capacity, and SF processing order. As the author assumed the service functions have been already deployed on the substrate network, the model did not contain the CPU constraint and forwarding latency constraint. Gupat et al. in [94] consider a deployment of SFC in a NeC scenarios, where NeC is short for Network-enabled Cloud. The NeC scenarios can be classified into three types: 'DC only', in which SFC can be deployed in the Datacenter (DC) alone, 'DC NFV x', in which SFC can be deployed on Datacenter or x NFV-capable nodes, and 'ALL NFV', in which all nodes are NFV-capable nodes. The authors formulated the SFC deployment problem with object of minimizing the network resource consumption. The model consists a set of constraints including link capacity, and order requirement. However, the model considers one VNF can be hosted at only one NFV-capable network node, and k-shortest paths have been pre-calculated.

Our model, however, differs from these previous models. Our model considers to minimizing the general costs in terms of setup cost and operational cost. The setup cost refers to the cost to initiate a new VNF instance in a substrate node, thus, by consolidating same type of VNFs onto one substrate network node, the setup cost can be reduced. Previous works such as [42][90][91][93] considered the setup cost is identical to all types VNFs. In our model, the setup costs are different to different types of VNFs which is realistic.

The operational cost refers to the cost to process a unit resource of VNF computation requirement in substrate network node, thus can be reduced by deploying VNFs on the most optimal locations. Different from the previous works such as[86][87], which considered the operational cost is same to all types of VNFs, in our model, we consider the operational cost is different according to the type of request VNFs, which is realistic.

Moreover, our model also considers guaranteeing the CPU and bandwidth resource constraints as well as the requirements of order and end-to-end delay. By comparing with previous works, Lin [86] and Zeng [87] have the similar optimization objective that minimizes the deployment and operational costs. However, they did not consider the order and delay requirements as shown in Table 3-1. Furthermore, as described in the section 3.3, the setup cost and operational cost defined in our model are realistic than these previous works. A practical definition can guide network operators to estimate the cost more accuracy when planning and operating networks.

3.3. Cost in SFC deployment

In this section, we define two types of costs when deploying VNFs of SFC onto the substrate network. We show how our definition of setup cost and operational cost are practical and realistic.

3.3.1. Setup cost

The VNF setup cost is defined as the cost to initiate a VNF on substrate node. This cost is consumed once during the process of placing a VNF on a substrate node, thus, sharing VNF that have been already setup in the substrate network will not consume setup cost. To initiate different types of VNFs will result in different cost, i.e., different types of VNFs have different setup costs. Thus, by consolidating same type of VNFs onto one substrate node, i.e., initiate a type of VNF on one substrate network node and share this VNF with multiple SFCs, the setup cost can be reduced. On example of such cost could be the license fee for initiating a VNF as VNF is a kind of software and subject to a license fee, and different types of VNFs have different license fee.

Differing from previous works such as [42][90][91][93], which consider the setup cost is identical for all VNFs, we consider different type of VNFs have different setup cost, which is more practical. Previous works simply consolidate VNFs onto same substrate network node to reduce the number of active nodes in the substrate network, thus reduce the setup cost, whereas our model consolidates the VNFs based on the cost value rather than the number of substrate nodes. Simple consolidating VNFs onto same substrate network nodes in previous works implies that different combinations of VNFs consolidation results in a same setup cost and does not consider consolidating same type of VNFs to share a same substrate node. For example, consider two substrate network nodes node-A and node-B, each node can host two VNFs, and four VNFs requests, in which are two F-type VNFs (e.g. firewall), VNF-F1, VNF-F2, and two N-type VNFs (e.g. NAT), VNF-N1, VNF-N2. Then, there could be two possible collocation results, that one collocation is A: F1F2, B: N1N2, and the second collocation is A: F1N1, B: F2N2, where A: F1F2 means VNF-F1 and VNF-F2 are collocated in node-A, and so as F2N2. In simple consolidating VNFs case, both of two collocation results in a same cost, as their objective is to reduce the number of substrate network nodes, and both solution results in two number of substrate nodes. However, in our model, we consider the collocation A: F1F2, B: N1N2 is more cost-efficient, as VNF-F1 and VNF-F2 are same F-type VNFs, thus, it initiates F-type VNF only once, and this VNF instance is shared by both VNF-F1 and VNF-F2, so as N1N2. Furthermore, taking the license fee as an example, as software are usually charged by the number of nodes or hosts, the collocation A: F1F2, B: N1N2 only consumes two licenses each for F-type and N-type VNF, respectively. However, the collocation A: F1N1, B: F2N2 will consume four licenses as F-type VNFs needs two licenses for installed on node-A and node-B, so as N-type VNFs. Another extreme example is that consider only one substrate network node, with four VNFs requests, in which are two F-type VNFs (e.g. firewall), VNF-F1, VNF-F2, and two N-type VNFs (e.g. NAT), VNF-N1, VNF-N2. In simple consolidating, any combination of these four VNFs are possible solution as the objective is minimize the number of substrate network node. In our model, as we consider different type of VNFs have different setup cost, the cost of consolidation F1F2 and N1N2 results different setup costs, and our model will choose the consolidation with minimum setup cost.

In other words, as shown in the section 3.4.3, our model defines the setup cost as φ_γ and γ is the type of VNFs, which indicates the setup cost is relied on the type of VNFs, whereas previous works defined the setup cost as φ, which did not consider the types of VNFs for setup cost.

3.3.2. Operational cost

The operational cost is the cost to process a unit of computation resource on a substrate node. Both the type of substrate network and the type of VNFs affect the value of operational cost, i.e., different substrate network nodes process one unit of computation resource requested by different types of VNFs at different operational cost. Therefore, the operational cost can be minimized by finding an optimal substrate network node that processes the type of VNFs at the least cost.

Differing from previous works such as in [86][87], which assume the operational cost are uniform for all types of VNFs, our model steps forward to consider that the operational cost is different for one substrate network node processing different types of VNFs. This is reasonable since when processing one-unit of computation resource, the substrate network may require other resources such as storage, memory or I/O resources for different types of VNFs.

In other words, as shown in the section 3.4.3, our model defines the operational cost as $\omega_{\gamma,v}$, where v is the substrate network node and γ is the type of VNFs. This indicates that the operational cost is relied on both the γ type of VNFs and the substrate network v that processes the γ type of VNF. However, previous works define the operational cost as ω_v, which indicates that the operational cost is only relied on the substrate network node v, without considering the type of VNFs.

3.4. Mathematic modeling for SFC deployment

In this section, we formally formulate the deployment of SFC problem in a language of mathematical model. A mathematical model is helpful to explain a system and to study the effects of different components. The model casts the observed phenomenon into a mathematical description to simplify the complex of the problem so that the problem can be tractable.

3.4.1. Substrate network modeling

$G = (V, E)$: a directed graph represents substrate network. Here we use directed graph for guaranteeing orders of VNFs in SFC.

V: set of substrate network nodes.

E: set of substrate network edges.

C_v: computation resource capacity of $v \in V$.

$B_{(u,v)}$: bandwidth resource capacity of edge $(u, v) \in E$.

$L_{(u,v)}$: latency of edge $(u, v) \in E$.

3.4.2. SFC request modeling

Γ: a set of VNF types.

γ: $\gamma \in \Gamma$ is a type of VNF.

$\langle s_i, d_i, l_i, b_i, \overrightarrow{g_i} \rangle$: a 5-tuple to represent an SFC request. Here i is the i-th SFC request among SFC request set.

s_i: source node (or ingress node) of SFC i in the substrate network, where $s_i \in V$.

d_i: destination node (or egress node) of SFC i in the substrate network, where $d_i \in V$ and $d_i \neq s_i$

l_i: maximum value of end-to-end delay that SFC i can tolerate. The end-to-end delay of a path for an SFC request cannot exceed this value.

b_i: bandwidth requirement of SFC i. Here, we assume that the bandwidth requirements between any two neighbor VNFs in SFC are identical, that VNFs in SFC do not change the volume of traffic.

$\overrightarrow{g_i}$: a vector of ordered VNFs of SFC i, and $g_{i,j}$ is the j-th VNF in $\overrightarrow{g_i}$.

$\overrightarrow{t_i}$: a vector of ordered VNF types for SFC i. It has the same size as $\overrightarrow{g_i}$. Specifically, $t_{i,j} \in \Gamma$ represents the type of $g_{i,j}$, i.e., the type of the j-th VNF in $\overrightarrow{g_i}$.

$\overrightarrow{c_i}$: a vector of CPU resource requirement for SFC i. It has the same size as $\overrightarrow{g_i}$. Specifically, $c_{i,j}$ represents the CPU resource requirement of $g_{i,j}$, i.e., the CPU resource requirement of the j-th VNF in $\overrightarrow{g_i}$.

3.4.3. Cost model

φ_γ: setup cost of γ type VNF on an arbitrary substrate node such as VNF license fee, which is charged according to the number of nodes that install the VNF.

$\omega_{\gamma,v}$: per CPU operational cost for γ type VNF on substrate node $v \in V$.

The operational cost for an SFC, which requests a amount of CPU to operate VNF f on the substrate network node v, is calculated as

$$operation\ cost = a \cdot \omega_{f,v}$$

3.4.4. Decision variables

We use following notation to present decision variables:

$x_{\gamma,v}$: a Boolean variable that equals 1 if the type of VNF γ is deployed on substrate network node v, and 0 otherwise.

$y_{i,j,v}$: a Boolean variable that equals 1 if the j-th VNF in the i-th SFC is deployed on the substrate network node v, and 0 otherwise.

$z_{i,(u,v)}$: a Boolean variable that equals 1 if the edge (u,v) is on the path of the i-th SFC where $(u,v) \in E$.

$k_{i,v}$: an integer variable that indicates the rank of substrate node v along the path of the i-th SFC in the substrate network. This variable ranks the node along the path in the substrate network for one SFC request to avoid loop and keep VNFs' order in SFC.

3.4.5. Constraints

We describe constraints of SFC placement problem in this part.

1) Substrate network node CPU capacity constraints:

$$\sum_{i}\sum_{j} y_{i,j,v} \cdot c_{i,j} \leq C_v \qquad \forall v \in V \tag{1}$$

Eq. (1) ensures that total CPU resource requirements of VNFs allocated on substrate network node $v \in V$ cannot exceed the CPU capacity of node v for all SFC requests,

2) SFC placement constraints:

$$\sum_{v} y_{i,j,v} = 1 \qquad \forall j \; in \; i, \quad \forall \tag{2}$$

Eq. (2) ensures that for any VNF in any SFC request, the VNF should be placed on one and only one substrate network node.

$$\sum_{j} y_{i,j,v} \leq 1 \qquad \forall i, \forall v \in V \tag{3}$$

Eq. (3) ensures that no more than two VNFs in one SFC request are placed on the same substrate network node, i.e., Eq. (3) prevents the collocation of two or more VNFs in one SFC request.

$$x_{t_{i,j},v} = 1 \; if \; y_{i,j,v} = 1 \;\; \forall i, \forall v \in V \tag{4}$$

Eq. (4) ensures that if the j-th VNF in the i-th SFC is a $t_{i,j}$ type of VNF which is placed on the substrate network node v, the substrate network node v must have capability to host the $t_{i,j}$ type VNF.

$$y_{i,j,v} = 0, if \;\; v = s_i \;\; \forall i, j \tag{5}$$

$$y_{i,j,v} = 0, if\ v = d_i\ \ \forall i,j \tag{6}$$

Eqs. (5) and (6) ensure that VNFs in SFC requests cannot be placed on source and destination nodes of an SFC in the substrate network.

3) Bandwidth constraints:

$$\sum_i z_{i,(u,v)} \cdot b_i \leq B_{(u,v)}\ \ \forall (u,v) \in E \tag{7}$$

Eq. (7) ensures that total requirements of bandwidth resource of all SFC requests cannot exceed the total bandwidth capacity for any substrate edge $(u,v) \in E$.

4) End-to-end delay constraints:

$$\sum_{(u,v)\in E} z_{i,(u,v)} \cdot L_{(u,v)} \leq l_i\ \ \forall i \tag{8}$$

Eq. (8) ensures that for any SFC request, the path latency in the substrate network for this SFC request cannot exceed the maximum end-to-end delay requirement of the SFC.

5) Flow conservation constraints:

$$\sum_{u\in R} z_{i,(u,v)} - \sum_{u\in R} z_{i,(v,u)} = 0$$

$$\forall v, R\ is\ a\ set\ of\ v's\ neighbor\ nodes, v \neq s_i, v \neq d_i, \forall i \tag{9}$$

$$\sum_{u\in R} z_{i,(u,v)} - \sum_{u\in R} z_{i,(v,u)} = \begin{cases} -1 & v = s_i \\ 1 & v = d_i \end{cases}$$

$$R\ is\ a\ set\ of\ v's\ neighbor\ nodes, \forall i \tag{10}$$

Eq. (9) ensures that for any substrate network node v, if v is neither source node nor destination node of an SFC, the number of edges starting from node v is always same as the number of edges ending at node v (i.e., the amount of outgoing flow leaving node v, is always same as the amount of incoming flow entering node v).

Eq. (10) ensures that if substrate network node v is source of SFC, there should be always one edge starting from node v (i.e., a certain amount of outgoing flow leaving node v); if substrate network node v is destination of SFC, there should be always one edge ending at node v (i.e., a certain amount of incoming flow starting from node v).

$$z_{i,(u,v)} + z_{i,(v,u)} \leq 1 \quad \forall (u,v) \in E \quad \forall i \tag{11}$$

Eq. (11) ensures that the SFC path can travel through one substrate edge at most once.

$$\sum_{(u,v)\in E} z_{i,(u,v)} = 1 \; if \; y_{i,j,v} = 1 \quad \forall (u,v) \in E \; \forall i \tag{12}$$

$$\sum_{(v,u)\in E} z_{i,(v,u)} = 1 \; if \; y_{i,j,v} = 1 \quad \forall (u,v) \in E \quad \forall i \tag{13}$$

Eqs. (12) and (13) ensure if the j-th VNF in the i-th SFC is placed on substrate network node v, the SFC path must travel through this node

$$k_{i,v} - k_{i,u} \geq 1 - |V|\big(1 - z_{i,(u,v)}\big) \quad \forall (u,v) \in E \quad \forall i \tag{14}$$

$$k_{i,u} = 0 \; if \; u = s_i \quad \forall i \tag{15}$$

Eqs. (14) and (15) ensure that there is no isolated loop in the substrate network. An isolated loop is a path that starts from and ends at the same substrate node. This path can meet the conditions of constraints (9), but it is not on the path of an SFC, which starts from the source node of SFC, ends at destination node of SFC, and travel through substrate network nodes which host VNFs. To avoid to generating isolated loop, we assign rank numbers to substrate nodes along the path of SFC as eqs (14) and (15). If $z_{i,(u,v)}$ is equal to 1, which indicates $(u,v) \in E$ is on the path of SFC and node v comes after u as substrate network topology is modeled as a directed graph, the rank of v, $k_{i,v}$, is always higher than the rank of u, $k_{i,u}$. If $z_{i,(u,v)}$ is equal to 0, it has no impact on the rank of nodes u, v, since $|V|$ is the number of nodes in substrate network. It is obvious that the destination node will always has a higher rank than the source node, which indicates that the source node and destination node cannot be the same node. In such a way, isolated loops are eliminated.

6) VNFs order constraints:

$$k_{i,u} - k_{i,v} \geq 0 \; if \; y_{i,j+1,u} = 1 \; and \; y_{i,j,v} = 1 \; \forall i,j, \forall u,v \in V \tag{16}$$

Eq. (16) ensures that if substrate network node u, v can host the $(j+1)$-th and j-th VNF in an SFC request, respectively, then substrate network node u must come after substrate node v in the flow path to keep the order of VNFs.

3.4.6. Optimization objective

The objective of model is to minimize setup cost and operational cost. Total setup cost of all SFC requests S is defined as

$$S = \sum_{\gamma \in \Gamma} \sum_{v \in V} x_{\gamma,v} \cdot \varphi_\gamma \tag{17}$$

Here, $x_{\gamma,v}$ is a Boolean variable. $x_{\gamma,v}$ is equal to 1 if the type of VNF γ is placed on substrate network node v. φ_γ is the setup cost of γ type VNF on one substrate node. $\sum_{v \in V} x_{\gamma,v}$ indicates the number of substrate nodes which host γ type of VNF, and $\sum_{v \in V} x_{\gamma,v} \cdot \varphi_\gamma$ is the summation of setup cost for γ type VNF. To minimize setup cost, one type of VNF can be placed on as a smaller number of substrate network nodes as possible.

Operational cost O is defined as

$$O = \sum_{i} \sum_{j} \sum_{v} y_{i,j,v} \cdot c_{i,j} \cdot \omega_{t_{i,j},v} \tag{18}$$

Here, $\omega_{t_{i,j},v}$ is operational cost for substrate node v processing a unit of VNF $g_{i,j}$. $c_{i,j}$ is the amount of CPU requirement for VNF $g_{i,j}$. Therefore, $c_{i,j} \cdot \omega_{t_{i,j},v}$ is the total operational cost for substrate node v processing VNF $g_{i,j}$. This objective adds all the operational cost of VNFs in all SFC requests. To minimize operational cost, the VNFs can be placed on the proper substrate network nodes with minimum operational cost.

The overall cost $\mathbb{C}$, which is the summation of setup and operational costs, is given as below:

$$\mathbb{C} = S + O \tag{19}$$

3.4.7. Network resource utilization ratio

The network resource utilization ratio is defined as:

$$r = \frac{\xi_u}{\xi_t} \tag{20}$$

ξ_u is the total network resource requirements of all SFC requests that have been placed on substrate network. ξ_t is total resource capacity of substrate network that could be bandwidth or CPU. In case of bandwidth resource, ξ_u and ξ_t are defined as:

$$\xi_u = \sum_{i} \sum_{(u,v)\in E} z_{i,(u,v)} \cdot b_i \qquad \xi_t = \sum_{(u,v)\in E} B_{(u,v)} \tag{21}$$

and in case of CPU resource, ξ_u and ξ_t are defined as:

$$\xi_u = \sum_{i} \sum_{j} \sum_{v\in V} y_{i,j,v} \cdot c_{i,j} \qquad \xi_t = \sum_{v\in V} C_v \tag{22}$$

3.5. Experimental study and analysis

In this section, we conduct experimental study on our model. We evaluate our model from two aspects: cost reduction and end-to-end delay guarantee.

3.5.1. Substrate network model

The substrate network model used in experiments is NSFNET as shown in Figure 3-1 described as following:

NSFNET network model: National Science Foundation Network (NSFNET) was a program sponsored by National Science Foundation (NSF) to promote advanced research and education networking in United States [129]. Many prior works have utilized this network topology to test and evaluate their proposal [47][59][62][130][131] Figure 3-1 shows the NSFNET topology. The NSFNET contains 14 nodes and 21 edges. The number along the edges are delay of that edge in milliseconds similar to the setting in prior work[62].

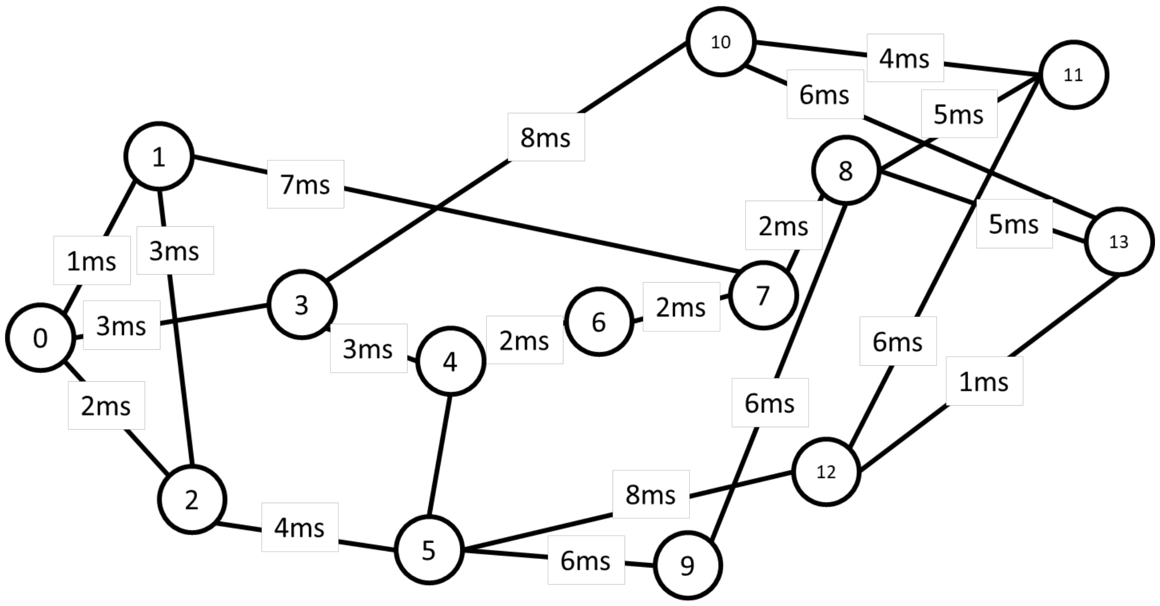

Figure 3-1 NSFNET network model.

Figure 3-2 shows the experiment system model. A batch of SFC requests arrives in NFV. The MANO receives the SFC requests and deploys them by utilizing the exact solution derived from the model. The model is solved by IBM ILOG CPLEX Optimization Studio v12.9 [95]. The parameters are introduced in the next section.

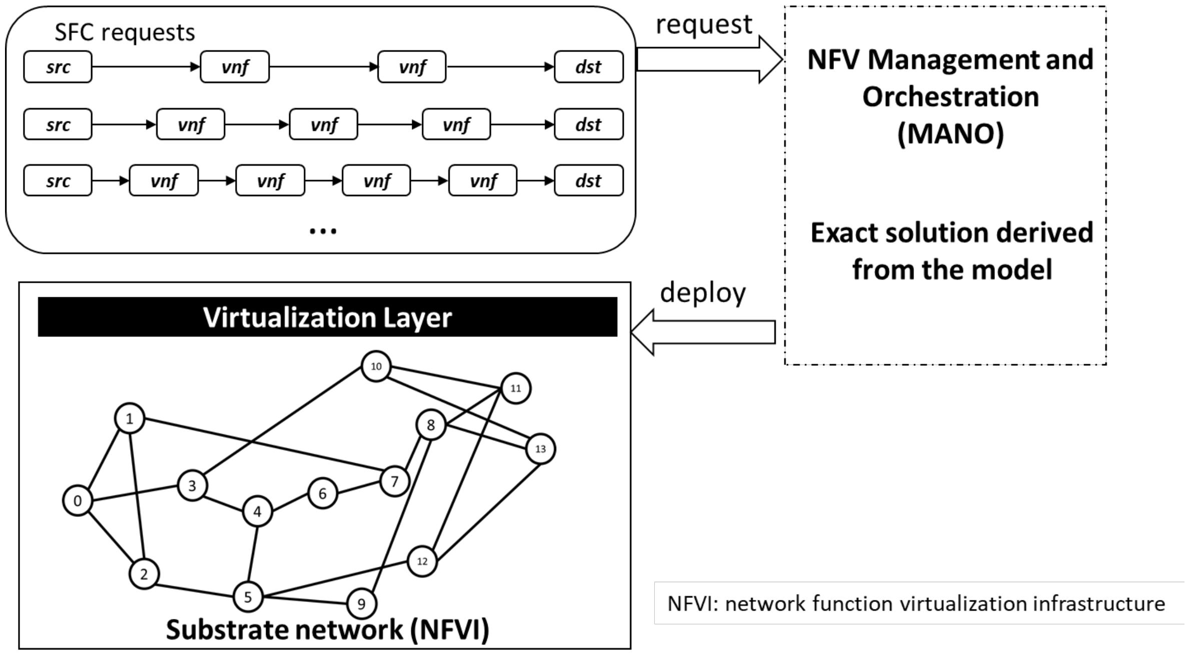
SFC requests
src
vnf
vnf
dst
src
vnf
vnf
vnf
dst
src
vnf
vnf
vnf
vnf
dst
...
request
NFV Management and Orchestration (MANO)
Exact solution derived from the model
deploy
Virtualization Layer
Substrate network (NFVI)
NFVI: network function virtualization infrastructure

Figure 3-2 Experiment system model.

3.5.2. Parameter setting

CPU capacity of a substrate node and bandwidth capacity of a substrate edge are set to 100 units, respectively. Delay of substrate network edges are set as shown in Figure 3-1. The substrate network supports 4 types of VNF. The number of VNFs in one SFC is randomly chosen between 2 and 4. CPU requirement of one VNF is randomly chosen between 4 and 6 units. Bandwidth requirement of an SFC is randomly chosen between 4 and 6 units. The end-to-end delay requirement of an SFC is randomly chosen between 30 and 50. The source node and destination node of an SFC are randomly chosen from the substrate network nodes. The source and destination nodes cannot be a same substrate node.

Table 3-2, Table 3-3, and Table 3-4 summarize the parameter settings of substrate network, service function chain requests, and setup and operational cost, respectively.

Table 3-2 Substrate network parameter settings.

parameters	**values**
CPU capacity	100
bandwidth capacity	100
latency of edge	as Figure 3-1
number of VNFs a node support	4

Table 3-3 Service function chain request parameter settings.

parameters	**values**
number of VNFs in SFC	2 - 4
CPU requirement	4 - 6
bandwidth requirement	4 – 6
end-to-end delay requirement	30 - 50
number of SFC request	6 - 10

Table 3-4 Setup cost and operational cost settings.

parameters	**values**
setup cost	5 - 10
operational cost	5 - 10

3.5.3. Evaluation of cost reduction

The objective of our model is to reduce the setup cost and operational cost with an exact solution. In this section, we evaluate the performance of reduction in comparison against previous works.

We conduct experiments on three models and derived the results shown in Figure 3-3. Specifically, the proposed model is our model that difference types of VNFs have different values of setup cost and different values of operational cost. The previous-1 and previous-2 models are the previous model that all types of VNFs have a same value of setup cost and a same value of operational cost.

Proposed model uses a setting of costs randomly chosen from (5, 10) as mentioned in the previous section, say *setting-1*. However, it must be careful when select different cost values to different types of VNFs in such random way for our proposed model, as it could select identical values which is not applicable to our model. We dropped the cost settings if they are happened to all same values. With *setting-1*, the model would derive a solution, which is a scheme describing how to deploy VNFs and chain the VNFs on the substrate network.

The solution of previous-1 and previous-2 models use another setting of costs which is also randomly chosen from (5, 10), say *setting-2*. With *setting-2*, Both previous-1 and previous-2 would derive a scheme describing how to deploy VNFs and chain the VNFs on the substrate network. These two schemes are exactly same as they are optimal solution.

The difference between previous-1 model and previous-2 model is how they calculate the total costs. Preious-1 model calculates the costs by using the deployment scheme with setting *setting-2,* whereas previous-2 model calculates the costs by using the deployment scheme with setting *setting-1*. In this way, we can thus understand how extent our realistic definition of costs could reduce the total setup cost and operational cost.

Appendix A and Appendix B show the details of the setup cost and operational cost setting data for both proposed model and previous model. As our model selects values of costs in a random way, we show the mean and deviation of cost values in our model in Appendices. Specifically, Appendix A describes the setup cost value settings, and Appendix B describes the operational cost value settings.

Figure 3-3 shows the total costs of deployment SFC with respects to the number of SFC requests. Each bar is composed of two parts, setup cost and operation cost of corresponding model at certain number of SFC requests.

The results show that our proposed model could achieve a lower total cost in comparison against previous works. Specifically, our model reduced the total cost up to 11% in average in comparison with previous-1 model, and up to 32% in average in comparison with previous-2 model, respectively. The results in comparison with previous-2 model indicates the gap of realistic cost and ideal cost, that without using our model, the costs will be over calculated. The over calculated cost could result in a problem that the estimated cost is higher than the actual cost that leads network operator making a wrong decision based on the estimation of cost.

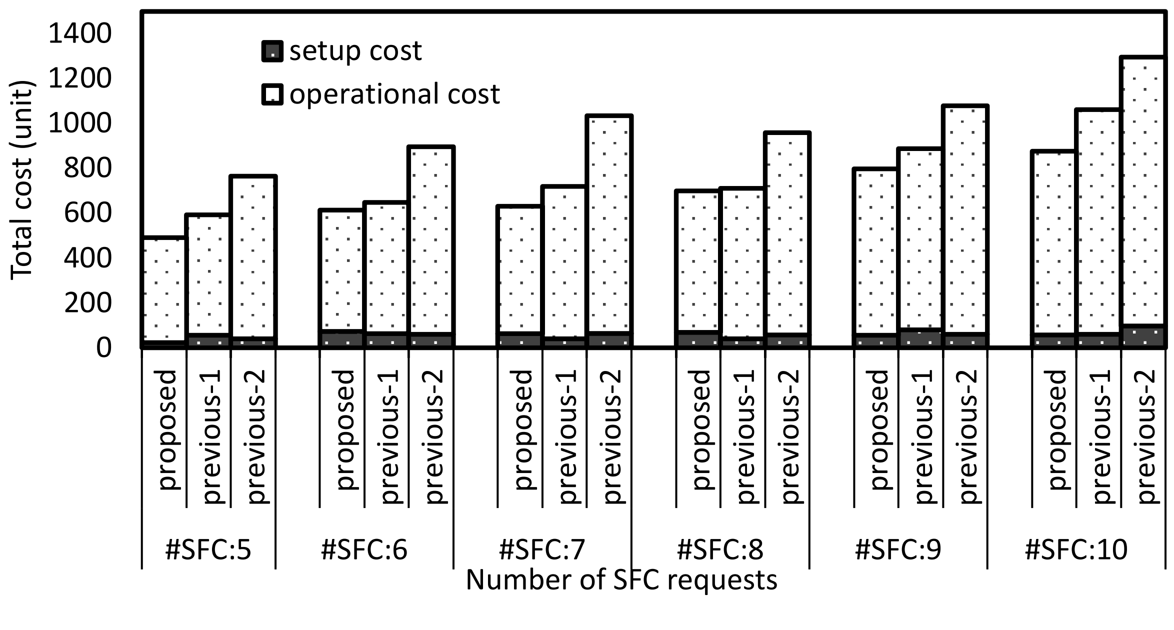

Figure 3-3 Total costs of different models.

The reason that our model could achieve a lower total cost is that our model consider costs are different to all types of VNFs rather than identical costs to all types of VNFs. Identical costs to all types of VNFs provide no information to the model that the VNFs should be deployed on which substrate nodes, and the model will consider all types of VNFs same weight. On the contrary, our model provides more detailed information to the model to deploy VNFs to locations at less costs. Consequently, our model could achieve a relatively lower costs than previous works.

3.5.4. Evaluation of end-to-end delay performance guarantee

In this section, we evaluate our model in terms of end-to-end delay performance guarantee. The end-to-end delay performance is one of requirements of some network services which are sensitive to end-to-end delay, such as autonomous vehicles. The end-to-end delay deadline should be considered carefully in this context.

We compare our model against modes in previous works that did not take end-to-end guarantee performance into account as shown in Table 3-1. Figure 3-4 shows the end-to-end delay deadline meet rate of our model and previous model. A higher value rate indicates that more SFC requests are satisfied within a certain end-to-end delay deadline. The results show that our model can guarantee all the end-to-end delay deadline requirement in comparison against previous works, which could guarantee end-to-end delay requirement for only around 50% SFC requests. This indicates that our model can accept all SFC requests but previous model can only accept around 50% SFC requests and reject other requests due to dissatisfaction of end-to-end delay requirement.

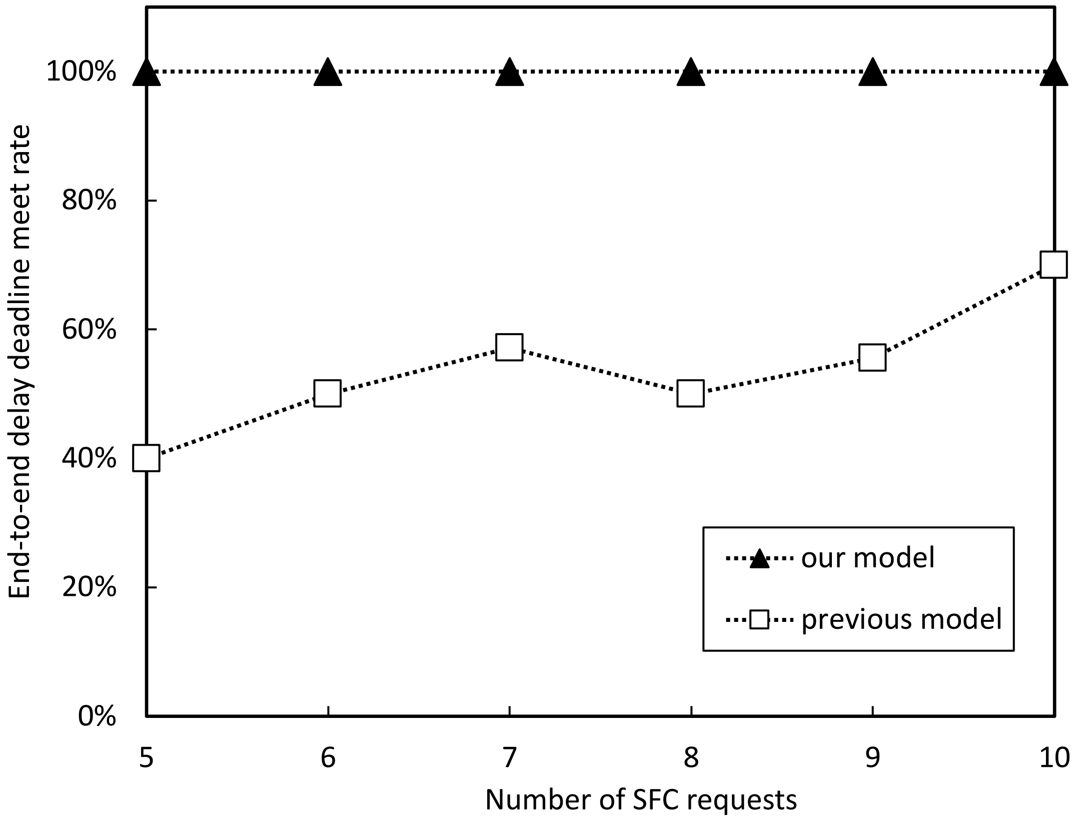

Figure 3-4 End-to-end delay deadline meet rate.

3.6. Extensive experimental study and analysis

In this section, we apply the mathematic model to analyze how the cost and network performance such as bandwidth and CPU utilization ratio are affected by different objectives. We also studied the cost and performance of network with and without keeping the order of VNFs in SFC requests.

3.6.1. Parameter setting

We evaluate the mathematic model in a small size network model due to the computation complexity of ILP problem. The substrate network topology is composed of six nodes and nine edges as shown in Figure 3-5. Similar to [78], CPU capacity of a substrate node and bandwidth capacity of a substrate edge are randomly chosen between 80 and 100 units. Delay of a substrate edge is randomly chosen from 2 or 3 units. The substrate network supports 4 types of VNF. The number of VNFs in an SFC is chosen between 2 and 4 randomly. CPU requirement of a VNF is randomly chosen between 3 and 5 units. Bandwidth requirement is randomly chosen between 4 and 6 units. End-to-end delay requirement of an SFC request is a value obtained by multiplying the number of VNFs with a number randomly selected between 4 and 6. The source node and destination node of an SFC are randomly selected from substrate network nodes. The source and destination nodes cannot be a same substrate node.

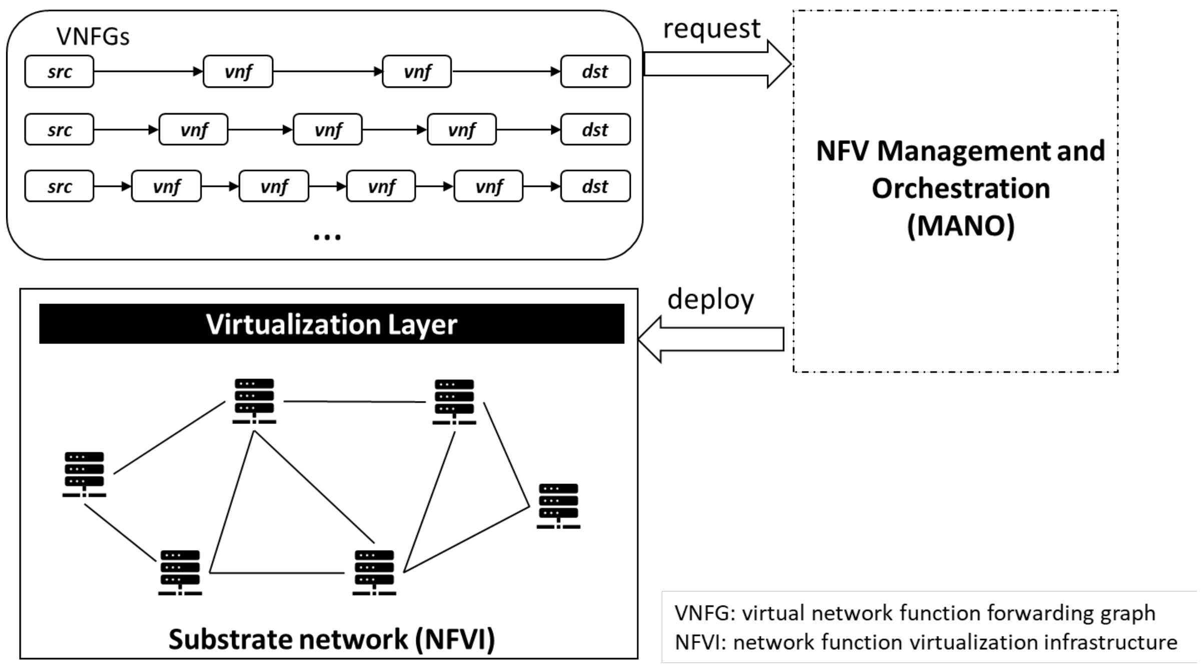
VNFGs
src
vnf
vnf
dst
src
vnf
vnf
vnf
dst
src
vnf
vnf
vnf
vnf
dst
...
request
NFV Management and Orchestration (MANO)
deploy
Virtualization Layer
Substrate network (NFVI)
VNFG: virtual network function forwarding graph
NFVI: network function virtualization infrastructure

Figure 3-5 Substrate network with six substrate network nodes.

The value of setup cost and operational cost is selected in a uniformly distribution within (5, 10). This cost setting could be suitable for early stage of operating network, when setup cost and operational cost are comparable. The numbers of SFC requests for once experiment varied from 5 to 40. All SFC requests arrive at substrate network at once, thus the model knows information of all SFCs. We generated 50 independent SFC requests with different settings and averaged the results to increase the statistical accuracy.

3.6.2. Impact of different optimization objective

In this section, we evaluate how different optimization objectives impact the setup cost and operational cost. By varying the optimization objectives, this model is helpful for the substrate network operator to compare the different cost.

In the evaluations, all SFCs arrive at once, that the SFCs do not dynamically arrive and depart at/from the substrate network. The model keeps the order of VNFs in SFCs, satisfies end-to-end delay performance objective, and all SFCs are successfully deployed on the substrate network.

We compare the results of different costs with the following three different optimization objectives:

1) Both costs objective, where both setup cost and operational cost are taken jointly as the objective of the model.
2) Operational cost objective, where only operational cost is taken as the objective of the model.
3) Setup cost objective, where only setup cost is taken as the objective of the model.

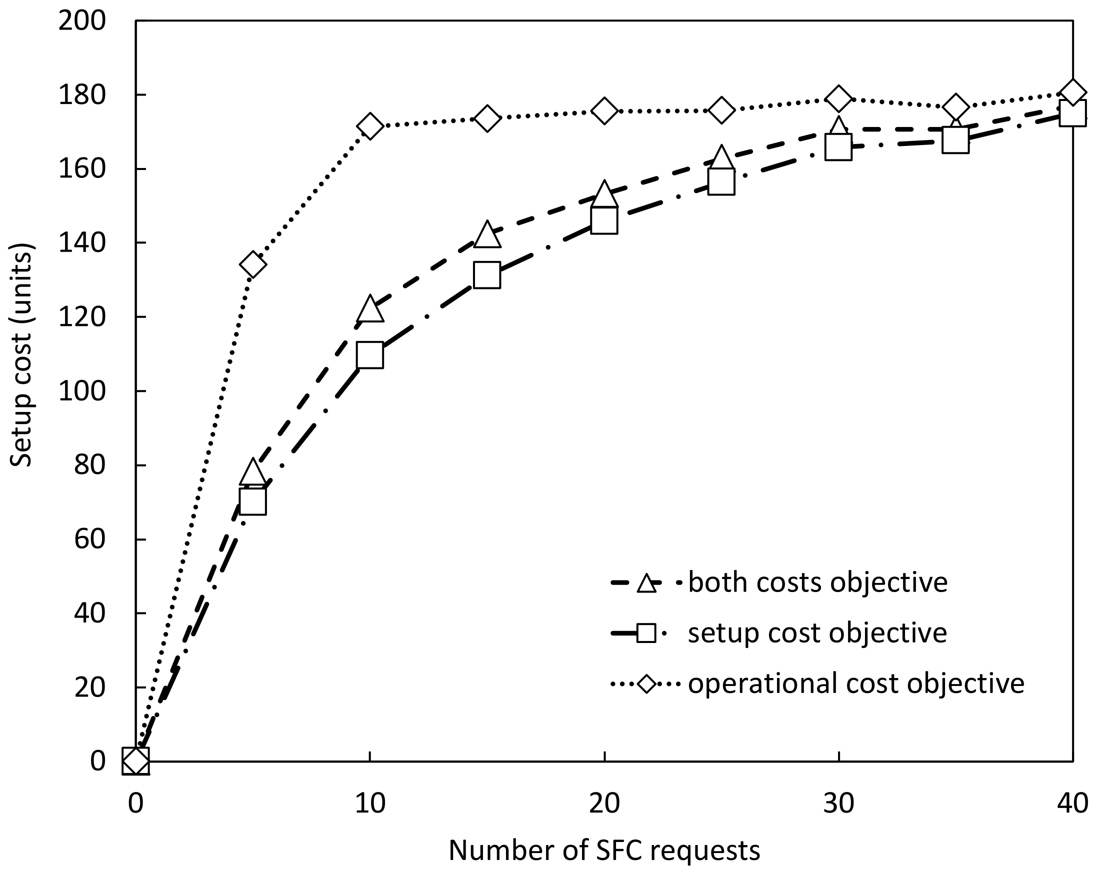

Figure 3-6 Setup cost at different optimization objectives. The setup cost is higher when network operator only wants to minimize the operational cost.

Figure 3-6 shows the results of setup cost at different optimization objectives. The x-axis is the number of SFC requests arrived at a time, varying from 5 to 40. For example, at number of SFC request 5, the value of long dash dot line with square marker is 70.02 units, which is the summation of total setup cost of these 5 request SFCs. The y-axis is the total setup cost in unit solved by the optimizer at corresponding number of SFC requests.

We compared the setup cost with different optimization objectives. The dash line with triangle markers shows the setup cost when the optimization objective is the summation of setup cost and operational cost jointly. The dot line with rhombus markers shows the setup cost when the optimization objective is only operational cost, and the long dash dot line with square markers shows the setup cost when the optimization objective is only setup cost.

When the number of SFC requests is a few, e.g., 5 or 10, the value of setup cost is small. With the number of SFC requests increasing, the setup cost increases. This is intuitive, since a greater number of SFC requests need more resource utilization and results in a higher cost. However, it is not a linear growth, but have an upper-limit value. This is reasonable. Recall that the setup cost gets charged according to the types of VNFs and the number of substrate nodes that hosts the VNFs. Every time more SFC requests arrive, it does not simply utilize more extra substrate nodes to host the new arrival VNFs but try to find and share existing substrate network nodes to host the new arrived VNFs. The setup finally cost reaches the upper-limit, indicates that for any one type of VNF, it has been placed on almost all substrate network nodes it could.

An interesting result is that the setup cost increases much faster when the optimization objective is only operational cost, which take the operational cost as the optimization objective. This is because when the optimization objective is only minimizing operational cost, the model finds solutions that have least operational cost without taking the collocation of the same type of VNF into account. Thus, as more SFC requests arrive, the model utilizes more substrate network nodes but not shares the existing substrate network nodes to achieve the minimum operational cost. This is the reason that the dot line with rhombus markers reaches the up limited faster than other lines. On the other hand, the long dash dot line with square markers gets the minimum values comparing the other two lines. This is because the model only considers minimizing the setup cost, thus, it shares as many number of substrate network nodes to host same type of VNFs, i.e., collocate the same type of VNFs on the substrate network nodes.

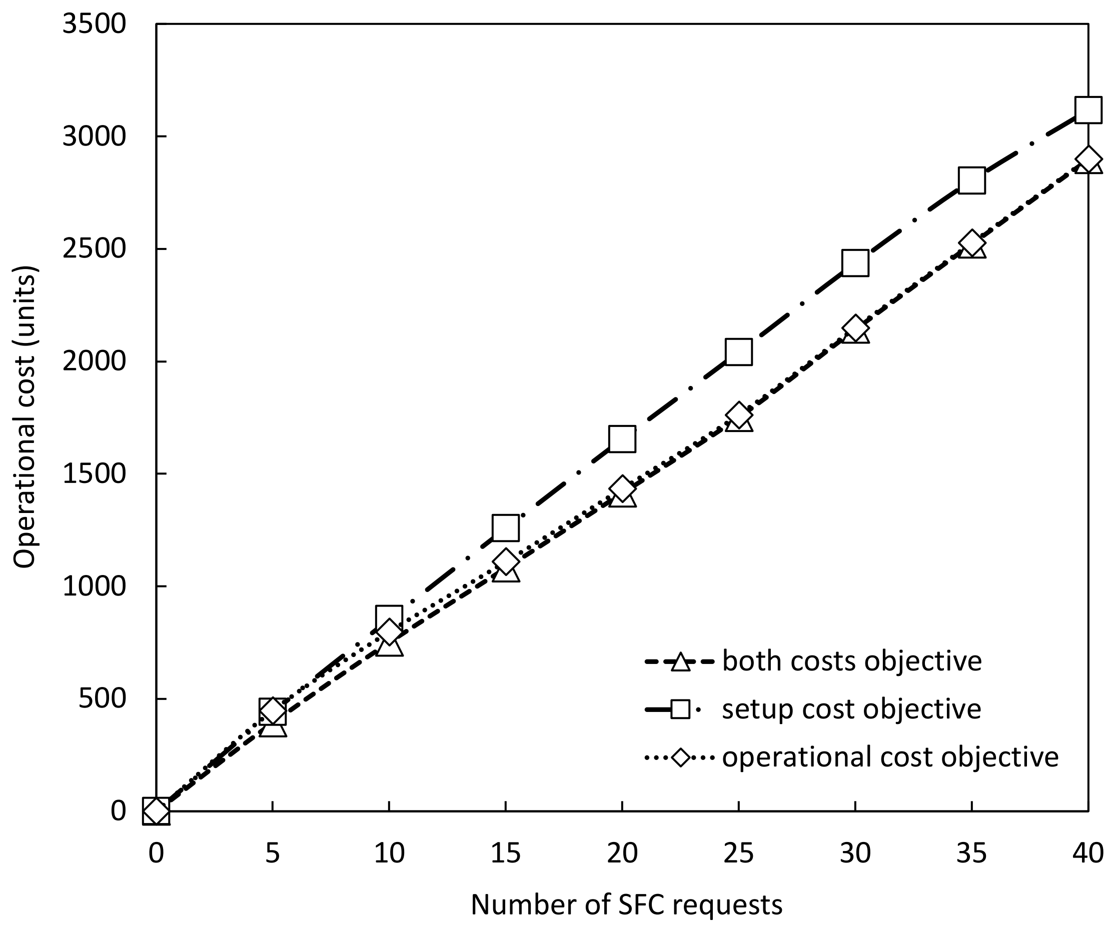

Figure 3-7 Operational cost at different optimization objectives. The operational cost is higher when network operator only wants to minimize the setup cost.

Figure 3-7 shows the results of operational cost at different optimization objectives. The x-axis is the number of SFC requests arrived at a time, varying from 5 to 40. The y-axis is the total operational cost in units given by the optimizer at corresponding number of SFC requests. For example, at the number of SFC request 5, the total operational cost of long dash dot line is 371 units, which is the summation of operational cost of these 5 SFC requests.

We compared the operational cost with different optimization objectives. The dash line with triangle markers shows the operational cost when the optimization objective is the summation of setup cost and operational cost jointly. The dot line with rhombus markers shows the operational cost when the optimization objective is only operational cost, and the long dash dot line with square markers shows the operational cost when the optimization cost is only setup cost.

The growth of operational cost shown by each line is linear for the reason that the operational cost is the cost that the substrate nodes process one unit of VNF requests and it is charged according to the amount of resources the SFC requested. Thus, the operational costs could not be reduced by the collocation of VNFs.

When only considering setup cost (setup cost objective) as the objective, its operational cost (long dash dot line with square markers) is larger than that at other objectives. This is because that the model attempts to collocate same type VNFs on as few numbers of substrate nodes as possible, and it does not consider whether a substrate node is an optimal one for processing VNF at lower operational cost.

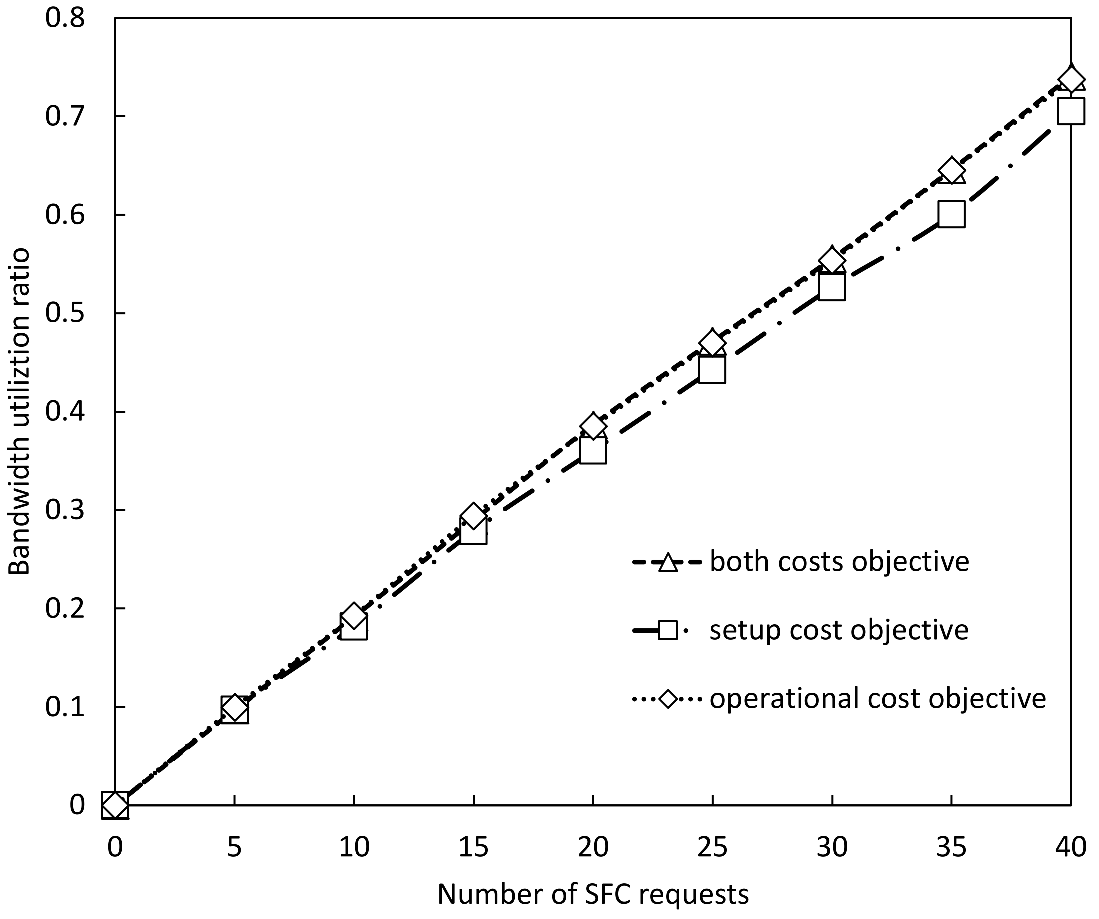

Figure 3-8 Bandwidth utilization ratio of substrate network. The bandwidth utilization ratio is lower when network operator only wants to minimize the setup cost.

Figure 3-8 shows the results of bandwidth utilization ratio of the substrate network with different optimization objectives. The x-axis represents the number of SFC requests arrived at a time, varying from 5 to 40. The y-axis represents the bandwidth utilization at the corresponding number of SFC requests. The bandwidth ratio is defined as in Eq. (20) and Eq. (21), that is the ratio of total requested bandwidth of SFCs with the total bandwidth of edges in the substrate network.

We compared the bandwidth utilization ratio of substrate network with different optimization objectives. The dash line with triangle markers shows the bandwidth utilization ratio when the optimization objective is the summation of setup cost and operation cost. The dot line with rhombus markers shows the bandwidth utilization ratio when the optimization objective is only the overall cost, and the long dash dot line with square markers shows the bandwidth utilization ratio when the optimization cost is only setup cost.

We found that the bandwidth utilization ratio achieves a lower value when only take only setup cost (only S) as the optimization objective. Recall that the setup cost is machine type independent cost defined in the beginning of the chapter. When the model minimizes the setup cost only, the model will be independent of the location of substrate nodes, and it can increase the probability of an optimized traffic path in the substrate network for SFC. However, when the model minimizes the operation cost, the model must consider the best substrate network node to host a VNFs, since the operation cost is machine dependent. Some SFCs may suffer a less optimized traffic path, such as detour to comprise the minimizing of the operation cost. For example, an optimal traffic path for SFC should be the shortest path between the ingress node and egress node of the SFC; however, to meet the minimum operation cost, the traffic path has to travel a longer path, such as detour, to reach the substrate network node, which has a less operation cost for processing the VNF.

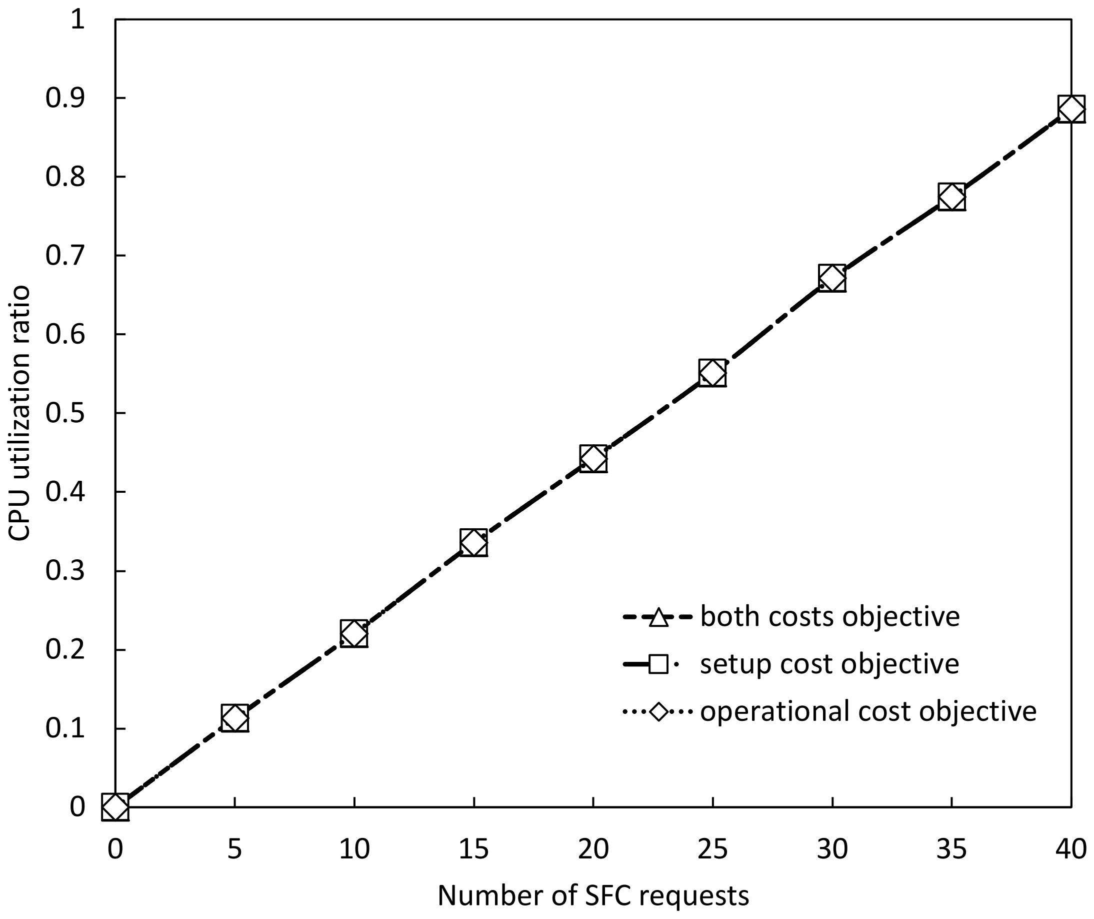

Figure 3-9 CPU utilization ratio of substrate network. The CPU utilization ratio are same for all optimization objectives.

Figure 3-9 shows results of the CPU utilization ratio of the substrate network. The x-axis is the number of SFC requests arrived at a time, varying from 5 to 40. The y-axis is the bandwidth utilization at the corresponding number of SFC requests. The definition of CPU utilization ratio is described in Eqs. (20) and (22), that is the ratio of total requested CPU units and the total CPU units of nodes in the substrate network. In contrast to the bandwidth utilization ratio, which different optimization objectives have different value of bandwidth utilization ratio, the values of CPU utilization ratio are same for all three objectives at each number of SFC requests. This is because the summation of CPU requirements of at each request is fixed as a constant value. Thus, the CPU utilization ratio remains the same no matter what the optimization objective is.

3.6.3. Impact of keeping order of VNFs in an SFC

It is an important requirement for some SFCs to keep the specified order of VNFs to realize the whole service correctly. For instances, a decryption function should come after an encryption function. In other cases, VNFs of an SFC could be independent and unrelated, thus it is unnecessary to place them in an order. For example, VNFs such as Firewall, Intrusion Prevention System (IPS) and Intrusion Detection System (IDS), could be placed in an arbitrary order, as illustrated in [28].

In addition, new types of VNFs would be emerging with rapid softwarization process of network functions. These VNFs may or may not need to be placed in a specific order. Therefore, as mentioned in [89], it is necessary to analysis how the cost is affected by the order of VNFs. We conduct experiments and analysis the impact of order of VNFs to setup and operational cost.

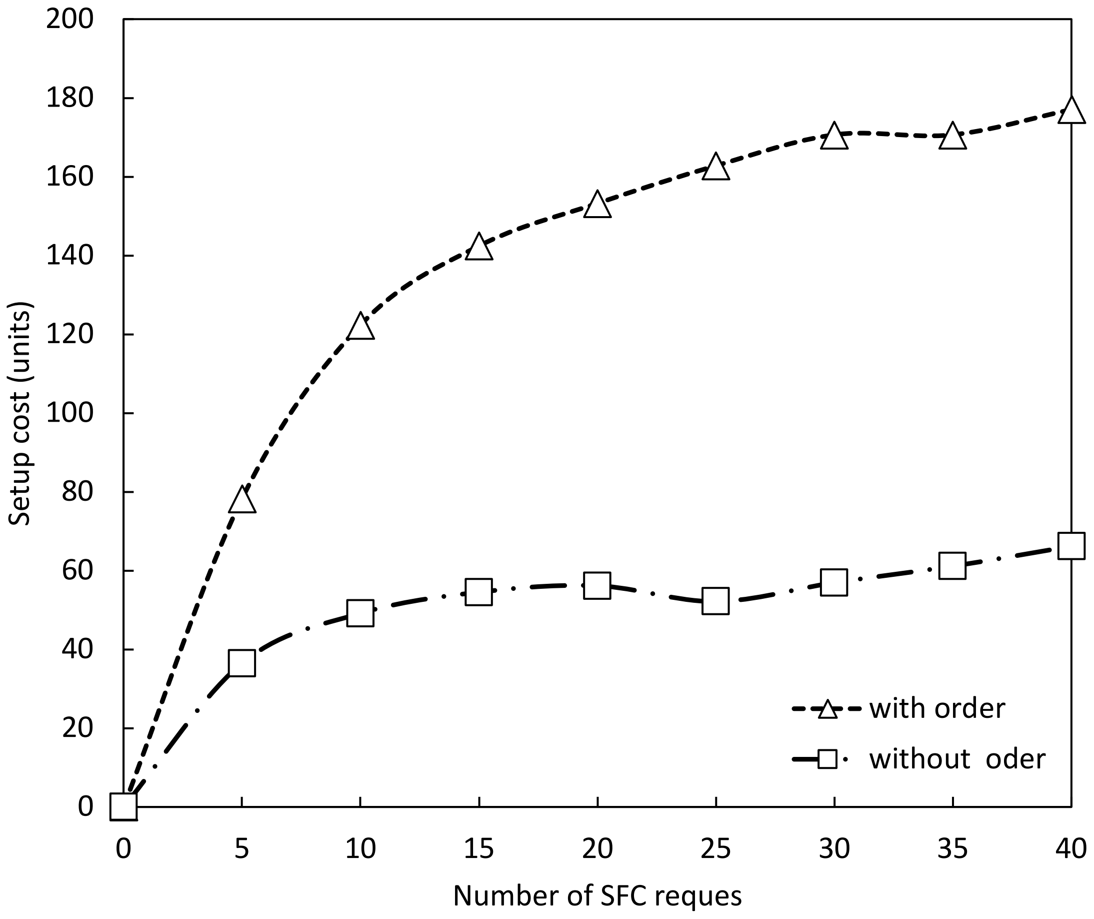

Figure 3-10 Setup cost with and without keeping the order of VNFs. The setup cost is higher when keeping the order of VNFs in an SFC.

Figure 3-10 shows setup cost with and without keeping the order of VNFs in the SFC request. The dash line with triangle markers shows the setup cost with keeping the order of VNFs in the SFCs, whereas the long dash dot line with square markers shows the setup cost without keeping the order of VNFs in the SFCs.

We found that the setup cost without keeping the order of VNFs is much smaller compared to that with keeping the order. This is because without keep the order of VNFs in the SFCs, the model has more degrees of freedom to find an optimal solution of placement of VNFs. By relaxing the order of VNFs constraint, the model can collocate the same types of VNFs on as less as the number of substrate network nodes by sharing these nodes, without considering whether the traffic path could travel through these substrate network nodes. On the contrary, in the case of consideration of order constraint, due to the limited CPU and bandwidth resources, the path may not be able to travel to some specific substrate nodes, and the mode has to utilize more substrate nodes to host a type of VNF. This results in a relative high setup cost.

Another observation is that the setup cost increases gradually when not keeping the order of VNFs. This is because the model would not look for a new node to host a type of VNFs if one node hosting this type of VNFs has sufficient CPU and bandwidth resources.

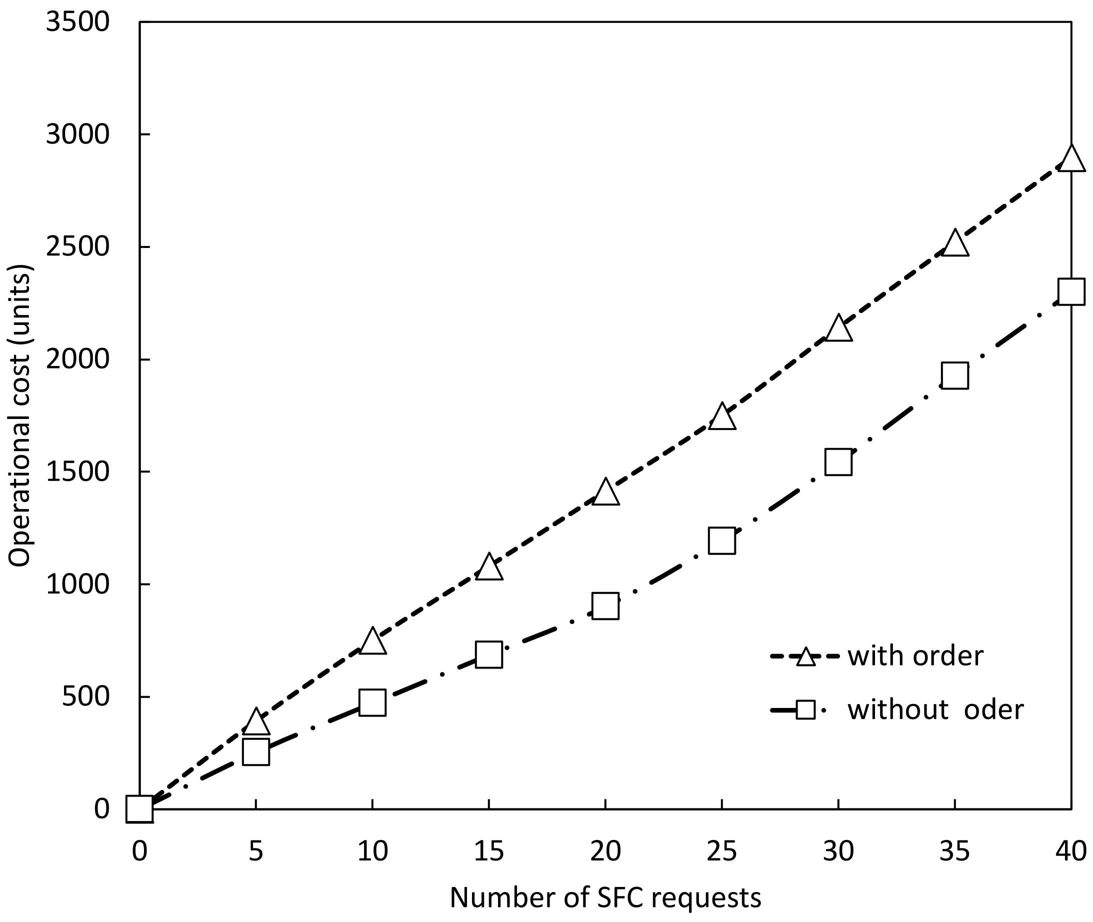

Figure 3-11 Operation cost with and without keeping the order of VNFs in SFC requests. The operational cost is higher when keeping the order of VNFs in an SFC.

Figure 3-11 shows the results of the operational cost with and without keeping the specific order of VNFs in SFC requests. The operational cost with keeping the order of VNFs is larger than that without keeping the order of VNFs. This is reasonable since without keeping the order of VNFs of SFC, the model can use substrate nodes whose operational cost for a type of VNF is least.

3.7. Discussion

3.7.1. Discussion on model objective

In section 3.4.6, we defined equations of setup cost and operational cost. We defined the overall cost as the summation of setup cost and operation costs as shown in Eq. (19). We attempted to use this equation as the objective of the model to find a solution with a reduced overall cost.

In the above setting, we consider the weight of setup cost and operational cost are same. In reality, the weight could be different to setup cost and operational cost. For example, the network operator may focus on setup cost more than operational cost for reasons like budget or policy. A better formulation of the optimization objective equation could be like following:

$$\mathbb{C} = (1 - \alpha)S + \alpha O$$

where we introduce a new variable α to balance the weight of setup cost and operation cost. Network operator could set a proper value to α to adapt their own network environment or policy, or vary the value of α to find a better cost estimation or balance. Note that extensive experiments in section 3.6 have demonstrated two special cases, only operational cost is taken as the objective of the model ($\alpha = 1$), and only setup cost is taken as the objective of the model ($\alpha = 0$).

3.8. Conclusions

SFC deployment problem with reducing cost requirement is complex due to many constraints and requirements in NFV. To better understand this problem, in this chapter, we formally formulated a mathematic model for the SFC deployment problem. The model takes CPU and bandwidth resources as constraints, as well as satisfying the end-to-end delay and order of VNFs in SFC. We defined the setup cost and operation cost. We showed that the definitions of cost are more practical than previous works in literature. We set the model with the objective to reduce these costs.

In the model, we defined a setup cost and operation cost. With these definitions, our model can reduce cost up to 30% in average. Moreover, our model guarantees the end-to-end delay deadline, which is a critical requirement to SFCs, such as real-time applications. The results show that our model can satisfy 100% end-to-end delay deadline requirements of SFC requests.

We also applied this model to analyze how the cost and network performance such as bandwidth and CPU utilization ratio are impacted by different objectives. We also studied the cost and performance of network with and without keeping the order of VNFs in SFC requests. We believe that our model can provide a way for network operators to analyze and estimate the cost incurred in deployment of SFCs with their own network environment and particular parameters.

Chapter 4

4. Delay-aware Service Function Chain Deployment Algorithm

4.1. Introduction

NFV transforms the design, construction, operation, and management of network in a way of separation of software implementation of network functions and hardware. This new paradigm increases the flexibility and scalability of network service deployment and reduces CAPEX and OPEX. Due to benefits of NFV, many network operators are considering integrating NFV into their networks.

However, different network services have their own requirements of Quality of Service (QoS). One critical QoS parameter is the end-to-end delay performance. Different end-to-end delay affects significantly the quality of users' experience. Researchers in [96][97] have identified three important limits regarding response times which impact user's experience. The first limitation is 10 seconds. A user would loss patient on the system if the response time is longer than 10 seconds. The second limitation is 1.0 seconds. A user could stay on the system but would be aware response delay. The third limitation is 0.1 second. A user would feel that the system responds instantaneously. S. Cheshire in [98] further argued that if the response is lower than 100ms, the user would feel it as an instantaneous system. Besides, Amazon has reported that every 100ms increase in response delay will cost 1% loss in e-commerce sales [99][100]. The ITU Recommendation in [101] has listed the performance considerations for different applications in terms of delay. For example, streaming audio/video may tolerate up to 10 seconds, web-browsing and e-mail access require around 1 second response time, whereas conversational voice and video, interactive games require less than 200ms end-to-end delay. Moreover, as mentioned in [102], delay is a main metric in next generation network and 5G communication. In future 5G network communication [103], applications require the network provides high throughput, high availability and latency-sensitive services, such as self-driving car, remote medical surgery, or wireless control of industrial manufacturing. End-to-end delay is a key factor on the network service response time and user's experience.

In this chapter, we investigate the challenge of deployment SFCs on substrate network (NFVI) with an optimized end-to-end delay. As described above, many network services and applications require strict QoS requirement of a low end-to-end delay. However, it is a challenge to place VNFs of SFC onto NFVI with an optimized end-to-end delay. An algorithm must find a proper location for VNFs and route the traffic through these locations in NFVI. As NFVI is a collection of substrate nodes (hosts or servers), and can be regarded as a distributed hardware resources, it is obvious that different deployment scheme will result in different resource consumption and affect the actual end-to-end delay of SFCs.

The algorithm to this problem must satisfy following requirements. The first requirements is that the path found by algorithm should have a shorter delay with enough number of substrate nodes to host all VNFs in the SFC request. The second requirement is the substrate nodes hosting VNFs should have sufficient computation resources. And the third requirement is the path should travel through VNFs in a specific order.

To tackle this problem, we design a dynamic programming-based algorithm for deploying SFC in this chapter. This algorithm deploys VNFs of an SFC with awareness of minimizing end-to-end delay while taking computation, bandwidth and the order of VNFs into account. Our algorithm assumes the substrate network nodes can host any type of VNFs and the VNFs of SFCs can be deployed on any substrate network nodes. By recording intermediate information such as resource usage, minimum end-to-end delay, our algorithm deploys the VNFs of SFC along a path with a minimized end-to-end delay in substrate network.

We perform experimental study on the proposed algorithm with comparison with other algorithms in literature on various network model. The results of the experiments show that our proposed algorithm outperform other algorithms in terms of short end-to-end delay of SFCs, high acceptance rate on various network models.

Section 4.2 presents the related works on the VNF placement with end-to-end delay requirement.

Section 4.3 introduces the problem of the SFC deployment with desire QoS requirement in terms of end-to-end delay of SFCs. This section also states the challenges to solve this problem.

Section 4.4 presents a dynamic programming based heuristic algorithm for solve SFC placement problem with end-to-end delay requirement.

In Section 4.5, we conduct numeric evaluations on our proposed algorithm with comparison with other algorithms on different network models. The results of evaluations show that our algorithm outperforms other algorithms in terms of end-to-end delay and acceptance rate.

Section 4.7 makes a conclusion on this chapter.

4.2. Related works

Previous works such as Miller [96], Nielsen [97] and Cheshire [98] have investigated how the end-to-end delay would impact users experiences on using the network system. The ITU Recommendation in [101] has listed the performance considerations for different applications in terms of delay.

Many previous works in literature also place SFCs with considering the end-to-end delay of SFCs. Luizelli et al. [93] classified SFC requests into three topological components: line path, bifurcated path and bifurcated path. Any SFC requests can be obtained from these three components. Then the authors formulate an ILP placement model to minimize end-to-end delay and resource overprovision ratio. However, it takes a relatively long time for large scale network to computing the exact solution. Our algorithm leverages the dynamical programming technique to record intermediate information which could reduce the computation time. Xia et al in [105] formulated VNF placement problem as a binary integer programming model. The authors proposed a heuristic algorithm to reduce conversions between optical/electrical/optical (O/E/O) by centralizing network functions in an SFC into fewer pods and reducing unnecessary flow traversals. However, although the authors consider the CPU resource limitation, other network resources such as pod capacity and bandwidth resource capacity are not considered. In contrast to this algorithm, our algorithm considers both CPU and bandwidth resources while finding a shorter end-to-end delay path.

Liu et al. in [104] studied the VNF placement problem aiming to improve the performance of service function chain from the aspect of end-to-end delay as well as bandwidth resource consumption. The authors first formulate a 0-1 programming problem and then proposed a greedy algorithm to solve the deployment problem. The greedy algorithm deploys VNFs according to the importance factor of the VNFs, and then finds a path to deploy the virtual links between deployed VNFs. However, this work did not consider the computation resource constraints for processing VNFs. Moreover, as greedy algorithm finds best solution in each iteration, it cannot have a global information on the delay of substrate network, thus, the whole paths of solution may not be optimal.

Qu et al. [115] proposed a delay guarantee route optimization frame for VNF placement and traffic routing based on k-shortest path algorithms. The algorithm first finds a shortest path, and then checks the resource capacity for VNF placement. If the resource capacity is not sufficient, it finds resources around the shortest path by branching the shortest path. As the resources are limited around the shortest path, the algorithm has no global information on the resource capacity. Our algorithm explores global information of substrate network to obtain a path with relative shorter end-to-end delay.

Hawilo et al. [116] formulate the problem as an MILP optimization model and proposed a betweenness centrality based VNF placement algorithms for minimizing intra-VNF end-to-end delay. Guan et al. [117] also utilized the concept of betweenness centrality for deployment of VNFs of an SFC in 5G wireless networking. Although betweenness centrality is a good metric for finding shortest path, a node with high betweenness centrality may not have a high resource capacity. Our algorithm checks the resource capacity while finding the path simultaneously. In such way, our algorithm can know global information of resource capacity rather than that limited around shortest path.

With regard to these algorithms, greedy-based algorithm coordinates deploying and chaining VNFs, but it utilizes only local information resulting in a local optimization rather than global optimization. *k*-shortest paths-based algorithm and betweenness centrality-based algorithm separately place and chain VNFs of an SFC, and they use global information to find the SFC path but local information to check network resources. We can see that there are still rooms to improve the performance of these works.

To fill these gaps, our algorithm explores substrate network to obtain global information rather than local information on delay and resources. By leveraging such global information, our algorithm deploys and chains VNFs coordinately to find a shorter end-to-end delay path with sufficient CPU and bandwidth resources. By storing the intermediate information, computation time could be reduced to an applicable time. Moreover, as shown in section 4.5, our proposed algorithm outperforms these works mentioned above in terms of end-to-end delay and acceptance rate.

Table 4-1 shows a comparison of our algorithm with related works.

Table 4-1 Related works comparison table.

Works	*Technique*	*Features*
Our algorithm	*DP*-based	Utilization of global information on delay and resource capacity. Coordination of placement and chaining of VNFs. Explore and record global network information. Low computation complexity with high performance.
Liu *[104]*	*GD*-based	Lack of global information on delay. Coordination of placement and chaining of VNFs. Find best solution in each iteration (local optimization).
Qu *[115]*	*k-SP*-based	Lack of global information on resource capacity. Separation of placement and chaining of VNFs (First find a shortest path, then check resource capacity).
Hawilo *[116]* ***Guan*** *[117]*	*BC*-based	Lack of global information on resource capacity. Utilization of only topology property. Overload substrate nodes and low performance.
Yang *[114]* ***Carpio*** *[123]*	*Random*-based	With no any strategy. Compared to show the necessity of a well-designed SFC deployment algorithm.
Luizelli *[93]* ***Xia*** *[105]*	*ILP*-based	Exact solution. High computation complexity.

DP*: Dynamic Programming*
GD*: Greedy*
k-SP*: k-Shortest Paths*
BC*: Betweenness Centrality*
ILP*: Integer Linear Programing*

4.3. Problem statement and challenges

4.3.1. Problem statement

In this chapter, we consider deploying VNFs of an SFC with a guaranteed QoS in terms of end-to-end delay of an SFC. The end-to-end delay of an SFC is the time that a packet entering the SFC from the src node, travel through all VNFs of the SFC and leaves from the dst node of SFC. Note that here we only consider the propagation delay and not consider the processing delay by VNFs. In other word, the end-to-end delay of an SFC is the summation of the delay of edges along the path of an SFC in the substrate network. When an SFC request arrived at NFV system, the MANO is responsible to find suitable locations in the substrate network for deploying VNFs while meeting all constraints, and chain VNFs along a path that has minimum delay.

4.3.2. Challenges

It is obvious that an arbitrary deployment of VNFs on the substrate network cannot achieve a minimized end-to-end delay for SFC. Nevertheless, the random deployment cannot guarantee that the substrate node or traffic route have sufficient resources for processing VNFs in terms of computation and bandwidth. Therefore, a carefully designed algorithm is needed to solve this problem. However, it is not easy to design such algorithm.

As mentioned in introduction, the algorithm designed should fulfill following requirements.

a) Finding a path with a minimum delay, with considering where there are enough number of nodes to host all VNFs of an SFC.
b) The path of SFC must have sufficient bandwidth resources for routing traffic, and the server or host on the path must have sufficient computation resources for processing VNFs.
c) The path of SFC must travel through the substrate nodes that host VNFs in a specific order requested by the SFC for end-to-end network service delivery.

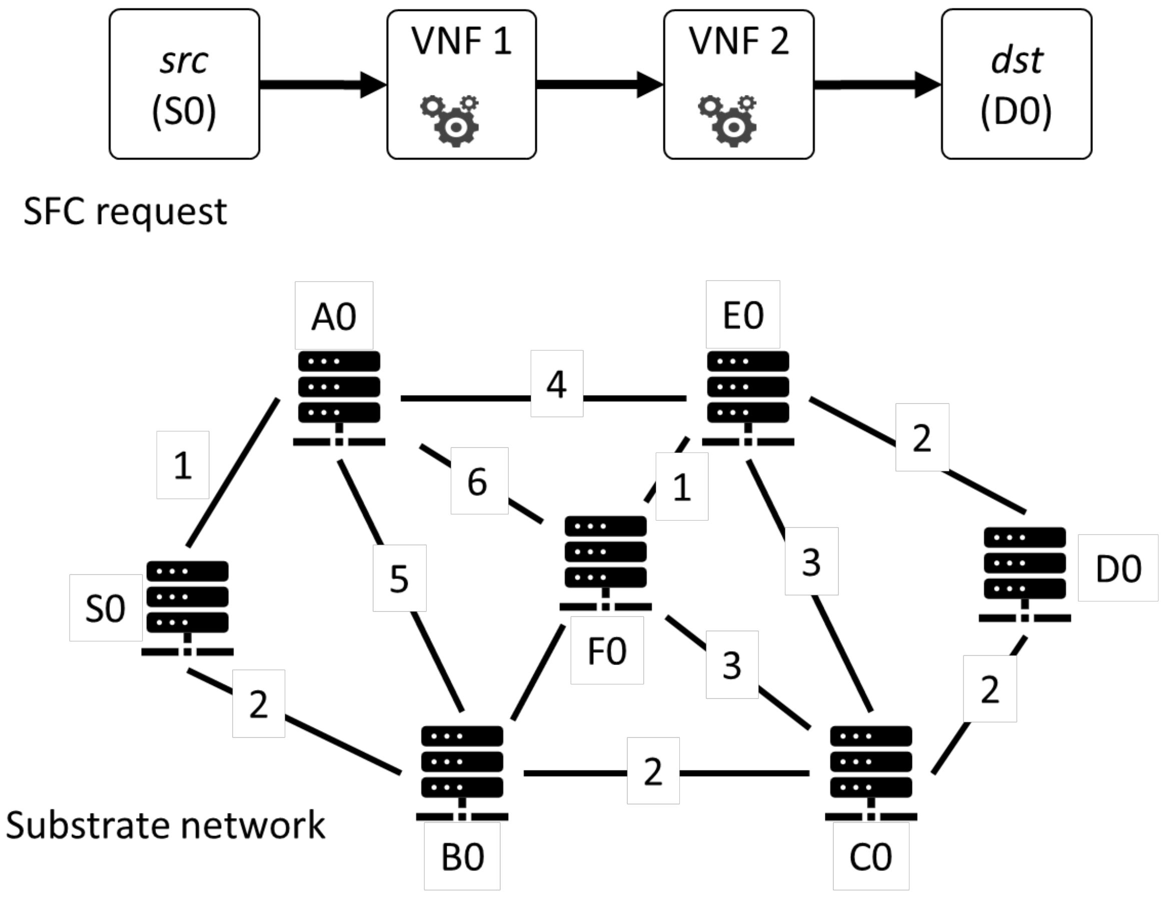

Figure 4-1 An illustration of difficulty to find a minimum delay path in substrate network.

Figure 4-1 shows an example of the difficulty to find a minimum delay path in the substrate network even without thinking the resource constraints. The linear graph on the top is an SFC request, and the seven-node graph at the bottom is a substrate network graph. A number along edges of the substrate network represents the latency or delay of each edge, such as 1-unit delay of edge (S0, A0), and 5 units delay of edge (A0, B0). VNF1 can be deployed on A0 as the delay is minimum from the ingress S0, and VNF2 can be deployed on E0, as edge (A0, E0) has the minimum delay of 4 units from A0, compared to the edge (A0, F0), whose delay is 5 units, and the edge (A0, B0) whose delay is 6 units. Then, the resultant path is S0→A0→E0 and its delay is 5 units. However, there is a better path, S0→B0→F0, whose delay is only 3 units, if we deploy VNF 1 on B0 and VNF 2 on F0, respectively. Thus, this greedy algorithm cannot find an optimal path with minimum delay. Moreover, even placing VNF 1 on B0 and VNF 2 on F0, the whole path for ingress S0 to egress D0 is S0→B0→F0→C0→D0, the path delay is 8 units. We could find a path S0→B0→C0→D0 with a shorter delay, which places the VNF 1 on B0, and VNF 2 on C0, respectively. This path delay is only 6 units. However, we must assure that

B0 and C0 have sufficient CPU resources for hosting VNF1 and VNF2, and the minimum bandwidth of path S0→B0→C0→D satisfies the bandwidth requirement of SFC.

In following sections, we introduce the system model and our proposed algorithm for solving this problem.

4.3.3. System model

In this chapter, NFV architecture consists an NFVI, a MANO and a set of VNFs. The NFVI is represented as a substrate network in which are a set of servers with computation resource capacity and edges interconnecting servers with bandwidth resource capacity. The MANO receives the requests of SFC requests and deploy the SFC on the substrate network. Both the substrate network and an SFC is modeled as a linear directed graph.

Substrate network model

A substrate network is modeled as an undirected graph $G^s = (V^s, E^s)$ where V and E represent the set of nodes and edges, respectively. c_n^c, c_n^f and c_n^u denote the capacity, available and used CPU resource of node $n \in V^s$, respectively, where $c_n^c = c_n^f + c_n^u$. b_e^c, b_e^f and b_e^u denote the capacity, available and used bandwidth resource of edge $e \in E^s$, respectively, where $b_e^c = b_e^f + b_e^u$. l_e denotes the latency of edge $e \in E^s$.

SFC request model

A linear directed graph $G^v = (V^v, E^v)$ represents an SFC request, where V^v represents the VNF in SFC and E^v represents the link between two neighbor VNFs. r_v^c represents the CPU request of VNF $v \in V^v$. $intf_v^i$ and $intf_v^o$ represent the input and output bandwidths, respectively, and f_v denotes a traffic processing function, where $intf_v^o = f_v(intf_v^i)$. $r_{(v_i,v_{i+1})}^b$ represents the bandwidth request of link $(v_i, v_{i+1}) \in E^v$ and $r_{(v_i,v_{i+1})}^b = intf_{v_i}^o = intf_{v_{i+1}}^i$. This means that the bandwidth request between two VNFs is equal to the output bandwidth request of prior VNF as well as the input bandwidth request of latter VNF. Two special VNFs src and dst represent the start and end points of SFC, i.e., ingress, and egress, respectively. Both src and dst do not request CPU resources, but only request bandwidth resources.

4.4. End-to-end delay aware SFC deployment algorithm

In this section, we introduce our proposed end-to-end delay aware SFC deployment algorithm. We first give a brief introduction on the dynamic programming technique, which we use it in our algorithm. Then, we describe our algorithm with details.

4.4.1. Dynamic programming

Dynamic programming (DP) is an advanced technique for designing an algorithm to find a solution with constraints. Usually, a complex problem contains multiple subproblems that are overlapped with each other. To solve the problem, DP first divides this complex problem into subproblems. Each of subproblems shares a sub-subproblems. DP then solves the sub-subproblems only once and records the results for this sub-subproblems. Next time when DP encounters the same sub-subproblem, it can just reuse the records previously calculated, thus saves much time. Finally, DP obtains the solution by computing each subproblems only once [106].

4.4.2. Algorithm description

From the above example in section 4.3, we infer that it is not easy to find a path with a minimum delay and allocate proper resources even for this simple case for deploying SFC. However, when calculating a path for one VNF, it can reuse the results that have been already calculated previously to save calculation time. This motives us to utilize the DP technique.

To address this problem, we propose a heuristic algorithm by leveraging DP technique to obtain a path with minimum delay for SFC. Our algorithm finds a proper substrate node for one VNF only once without extra recalculating precedents and decedents of this VNF. The algorithm stores information of feasible substrate network nodes, end-to-end delay of source to nodes for VNFs as the order of VNF in SFC. By backtracking the path from the destination node to source node of the SFC, the algorithm can find a minimum end-to-end delay path and guarantee the specific order of VNFs along the path.

To check whether a substrate node have capability to host a VNF, the algorithm examines following three conditions: CPU resource sufficiency, connectivity, and bandwidth resource sufficiency. Only if all three conditions are satisfied, the SFC can be deployed on the substrate network.

The CPU resource sufficiency condition is satisfied if the amount of available CPU resources of a substrate network node is larger than the requirement of a VNF of an SFC.

The connectivity condition is satisfied if there is at least a path for the VNF to connect itself to its precedent VNF.

The bandwidth resource sufficiency condition is satisfied if the available bandwidth of the substrate path is larger than the requested bandwidth requirement of the SFC.

The algorithm creates four matrices A, L, P, and T to keep tracking feasible location information.

- $A(v, f)$ is a Boolean value that it is true, if the substrate node v have capability to host VNF f. This matrix is applied to check the connectivity between two neighboring VNFs.

- $L(v, f)$ stores the minimum value of delay from src of the SFC to the substrate node v on which VNF f is placed. This matrix is applied to calcuate the minimum end-to-end delay of SFC.

- $P(v, f)$ records a set of paths. Each path is composed a set of ordered substrate nodes along this path. This matrix is used for backtracking the final end-to-end path from src to dst.

- $T(v, f)$ represents a temporary state of substrate network with residual CPU and bandwidth resources, when VNF $f$is supposed to be placed on substrate node v. It is necessary to create this matrix to track the available resources once a VNF is placed on a substrate node for succeeding placement of VNFs. As succeeding placement needs to refer to a network state in which all previous VNFs have been placed to assure that CPU and bandwidth resources are sufficient.

Algorithm 1 and Algorithm 2 describe details of our proposed SFC placement algorithm.

In Algorithm 1, our proposed SFC placement algorithm explores substrate nodes' capability to host VNF f in an SFC. If no one substrate node has the capability to host VNF f, the algorithm terminates with a $fail$ return value. If there is at least one substrate network node can host VNF f of SFC, the algorithm continues to place next VNF.

Algorithm 2 describes details of checking node's capability of hosting a VNF f. The procedure first checks CPU resource sufficiency of substrate node v for hosting VNF f. If CPU resources of substrate node v is not enough, the algorithm immediately returns $false$ (note $A(v, f)$ is initialized as $false$). If v has enough CPU resources for hosting VNF f, the algorithm then examines the connectivity and bandwidth sufficiency between n and all other substrate nodes which is capable to host VNF f's precedent.

Starting from line 6 in Algorithm 2, the algorithm finds the minimum delay path $p_{vv'}$ between v and v', for any substrate node v' $(v' \neq v)$, if $A(v', f) = true$. $(v', f) = true$ indicates that v' is capable to host VNF f_p, which is the precedent of VNF f. If the minimum delay path $p_{vv'}$ exists, and $p_{vv'}$ has sufficient bandwidth resource to meet the bandwidth request between VNFs f_p and f, then the algorithm summates $L(v', f_p)$ (i.e. the delay from src to v') and the delay of $p_{vv'}$. The algorithm returns $false$ if any one of following three conditions is not satisfied: $A(v', f_p)$ is $true$, shortest path $p_{vv'}$ exists, and the available bandwidth of $p_{vv'}$ is sufficient.

From line 11 in Algorithm 2, the algorithm calculates minimum delay paths for all pairs of v and v'. After all iterations of $v' \in substrate\ nodes, v' \neq v$, the algorithm set value of $L(v, f)$ to the minimum delay among all delays of possible paths from src to v, via v'. Meanwhile, the algorithm updates $A(v, f)$ by true, since substrate node v is capable to host VNF f, updates $P(v, f)$ by concatenating $P(v', f_p)$ and p, which is the path from src to v, updates $T(v, j)$ by $T(v', f_p)$, which is the temporary network state when f_p were placed on v'. Finally, the algorithm allocates CPU and bandwidth resources on $T(v, f)$ according to the request of f.

As Algorithm 1 iterates VNFs in SFC one by one from the first VNF of an SFC, our algorithm can guarantee the specific order of VNFs of SFC. Besides, for any VNF f, if there exists a substrate node v, so that Algorithm 2 could return $true$, it indicates that there must be a substrate network path connecting VNFs from src to f orerly. And this path has a minimum end-to-end delay with enough CPU and bandwidth resources.

Algorithm 1 Dynamic programming based SFC mapping scheme

```
1:  procedure MAPPING ($SFC$ , $substrate\ network$)
2:    for all $f \in SFC$ do
3:        $has\_feasible\_node \leftarrow fasle$
4:        for all $v \in substrate\ nodes$ do
5:            if FEASIBILITY ($v, f, substrate\ network$) then
6:                $has\_feasible\_node \leftarrow true$
7:            end if
8:        end for
9:        if $has\_feasible\_node = false$ then
10:           return $fail$
11:       end if
12:   end for
13:   return $success$
14: end procedure
```

Algorithm 2 Check node feasibility for VNF

```
1:  procedure FEASIBILITY ($v, f, substrate\ network$)
2:    $f_p \leftarrow f.precedent$
3:    $A(v,f) \leftarrow false$
4:    $L(v,f) \leftarrow \infty$
5:    if $cpu_v > cpu_f$ then
6:      for all $v' \in substrate\ nodes\ \ v' \neq v$ do
7:        if $A(v', f_p) = true$ then
8:          $p_{vv'} \leftarrow FIND\_SHORTEST\_PATH(v, v')$
9:          if $\exists p_{vv'}$ and $bw_{p_{vv'}} > bw_{f_p f}$ do
10:           $latency \leftarrow L(v', f_p) + LATENCY(p_{vv'})$
11:           if $latency \leq L(v,f)$ then
12:             $L(v,f) \leftarrow latency$
13:             $A(v,f) \leftarrow true$
14:             $P(v,f) \leftarrow P(v', f_p) \cup p_{vv'}$
15:             $T(v,f) \leftarrow T(v', f_p)$
16:              Allocate CPU and bandwidth of $v$ on $T(v,f)$
17:           end if
18:         end if
19:       end if
20:     end for
21:   end if
22:   return $A(v,f)$
23: end procedure
```

4.4.3. Route information extraction

In the above algorithms, if the connectivity, CPU and bandwidth requirements are satisfied, the algorithm returns success and the recorded information $P(v_{dst}, dst)$, where v_{dst} is the substrate network node (egress) which hosts dst in the SFC.

$P(v_{dst}, dst)$ consists a set of paths, each of which connects two neighboring VNFs of an SFC. For example, $P(v_{dst}, dst)$ = [[v_{src}], [v_{src}, …, v_{f_1}], [v_{f_1}, …, v_{f_2}], …, [$v_{f_{i-1}}$, …, v_{f_i}], [v_{f_i}, …, v_{dst}]] and each element of P, e.g. [v_{f_p}, …, v_f], is a set of nodes, in which the first node (say v_f) hosts VNF f_p , and the last node (say f) hosts VNF f. f_p and f are neighbors and f_p is preceding f. The nodes between v_{f_p} and v_f in [v_{f_p}, …, v_f] are in the path from v_{f_p} to v_f. Therefore, once the algorithm obtains P successfully, the route information can be easily extracted.

4.4.4. Matrices initialization

Our proposed algorithm utilizes the information in substrate network nodes. The information records feasibility of a node to host a VNF, the connectivity and the minimum end-to-end delay path between src and dst of SFC. Therefore, the algorithm needs to initialize matrices for recording the information as well as backtracking from dst to src to obtain the path. The algorithm initializes matrices as shown in below:

- Set $A(v_{src}, src) = true$ and other elements in A as $false$. v_{src} is the substrate network node hosting src of SFC.

- Set $L(v_{src}, src) = 0$ and other elements in L as infinity. Set $P(v_{src}, src) = [[src]]$ and other elements in P as null.

- Set $T(v_{src}, src)$ as a copy of current substrate network state which contains CPU and bandwidth resources. Other elements in T are set to null.

4.4.5. Computation complexity analysis

The algorithm calculates substrate network nodes for VNFs of SFC orderly. This needs $|g|$ times iterations, where $|g|$ is the number of VNFs. The algorithm examines all substrate network nodes for each VNF, which requires $|V|$ times. In each examination, the algorithm checks the connectivity $|V|-1$ times. Thus, the overall computation complexity of the algorithm is $O(|g||V|^2)$.

4.5. Evaluation and analysis

In this section, we evaluate our proposed SFC placement algorithm with comparison of other SFC placement algorithms in literatures on different network topologies. We wrote a simulator by using NetworkX [107] in Python [108]. We evaluate our algorithm in terms of end-to-end delay, acceptance rate, and resource utilization on two practical network models, JPN48 [124][125] and NSFNET [129], and random graph network models. The random graph topologies are generated on the basis of Erdős-Rényi graph [109][110]. The results show that our algorithm can outperform in terms of end-to-end delay, acceptance rate, and resource utilization.

4.5.1. Algorithms for comparison

We compare our proposed algorithm with other SFC deployment algorithms in literatures as described in the related works sections. Specifically, they are greedy algorithm based, k-shortest paths algorithm based, betweenness centrality-based and random algorithm based SFC deployment algorithms as described following:

a) **Greedy algorithm** [104][114]: Greedy algorithm is a popular approach in solving NP problem as it can obtain a sub-optimal solution within an acceptable short computation time. A greedy strategy solves a problem by making a sequence of decision. At each point, it makes the decision that seems the best choice at that moment [106]. In the SFC deployment context, this algorithm 1) starts from the substrate network node, which hosts the src of an SFC, 2) checks its neighbor nodes, 3) finds the neighbor node with the shortest latency edge, and 4) uses the node to host the next VNF. The algorithm greedily finds all nodes for hosting VNFs. Finally, the algorithm finds a shortest path from the substrate node who hosts the last VNF in the SFC to the substrate node who hosts the dst of the SFC.
b) ***k*-shortest paths algorithm** [115][116]: k-shortest paths is an very important and often used algorithm in network research. In the SFC deployment context, this algorithm first finds k shortest paths in terms of delay between src and dst of SFC. Then the algorithm uses the longest path as the candidate path. Here the longest path is the path with the greatest number of nodes along the path. If the number of nodes along the path

cannot host all the VNFs in the SFC, the algorithm finds one edge with least available bandwidth resource in the path, say edge (m, n), where m and n are substrate nodes. The algorithm removes the edge (m, n) from the path, checks and compares the CPU capacity of node m's neighbor nodes and node n's neighbor. The algorithm selects the node who has the most available CPU capacity among m and n's neighbor nodes to host VNF, say node c. Finally, the algorithm finds the shortest paths in terms of latency from c to m, and c to n, and connects the path. Notice if k is equal to one, it is a modified single shortest path algorithm.

c) **Betweenness centrality-based algorithm** [116][117]: BC [118][119] is a measure of centrality of a graph based on shortest paths. BC is given by the expression:

$$B(v) = \sum \frac{\sigma_{st}(v)}{\sigma_{st}} \qquad \forall v \neq s, v \neq t$$

where $\sigma_{st}(v)$ is the number of shortest paths from s to t that pass-through node v, and σ_{st} is the total number of shortest paths from s to t. It can be known that the node with a high value of betweenness centrality is the anchors and has the greatest number of shortest paths through it. In the SFC deployment context, the algorithm first deploys the middle VNF (say vnf between src and dst of SFC) on the substrate node with highest value of betweenness centrality. Then the algorithm recursively deploys the middle VNF on the left side of vnf that is between src and vnf and the middle on the right side of vnf that is between vnf and dst on the substrate nodes with higher value of betweenness centrality. Finally, the algorithm finds shortest paths to connect these substrate nodes. In the experiments, we utilize the algorithms[93][119][120][122] in NetworkX to calculate betweenness centrality value.

d) **Random algorithm** [114][123]: In the SFC deployment context, this algorithm first randomly selects k number of substrate nodes from substrate network, where k is equal to the VNFs of an SFC. Note that these k nodes cannot be the substrate nodes who host src and dst of SFC. Then the algorithm starts from src, finds shortest path one by one and connect the shortest paths between selected nodes. The order of substrate nodes selected for connected the shortest path is also randomly chosen. By comparing with this random algorithm, it can be known that whether our proposed algorithm has a considerable impact on reduce the end-to-end delay for service function chain, or alternatively simply deploy VNFs on the preferred nodes in substrate network is enough for end-to-end delay of an SFC.

4.5.2. Substrate network models for evaluation

In this section, we introduce network models used in the experiments. We conducted our evaluations on three network models, JPN 48, NSFNET, and Random Graph network model.

a) **JPN 48 network model:** JPN 48 is a network model developed based on publicly available information of Japan nationwide transportation network [124][125]. The network model contains 48 nodes and 82 edges. The nodes are placed on major cities in Japan. The model has been utilized for network research in Japan [126][127][128]. Figure 4-2 shows the topology of JPN 48. The number in the figure represents a city of Japan, e.g., node number 10 is Sapporo, node number 20 is Aomori, and node number 131 is Tokyo. We set the delay of each edge according to the real distance between two nodes. For instance, the delay between 10 and 20 is $1.5884322213336\ ms$, which is calculated as $476.2\ km/299792458(km/s) * 1000$, where 476.2km is the distance between city Sapporo (10) and city Aomori (20), and $299792458 km/s$ is the speed of light. 1000 is used to adjust the unit from second to millisecond.
b) **NSFNET network model:** National Science Foundation Network (NSFNET) was a program sponsored by National Science Foundation (NSF) to promote advanced research and education networking in United States [129]. Many prior works have utilized this network topology to test and evaluate their proposal [47][59][62][130][131] Figure 4-3 shows the NSFNET topology. The NSFNET contains 14 nodes and 21 edges. The number along the edges are delay of that edge in milliseconds similar to the setting in prior work[62].
c) **Random Graph network model:** To evaluate and compare our proposed algorithm on a larger, more complex but general substrate network without dependent on any specific topology, we introduce random graph model as the underlaying substrate network topology. Random graph model [109][110] is an intersection of graph theory and probability theory. There are two ways to generates a random graph, and we leverage the $G(n, p)$ model to generate a random graph topology, where n is the number of nodes and p is edge connectivity probability. Random Graph model has been utilized for evaluation algorithms such as in [78][111][112][127][131]. The delay of each edge is randomly chosen between 1 and 5 in milliseconds unit. Figure 4-4 shows an illustration of random graph network model.

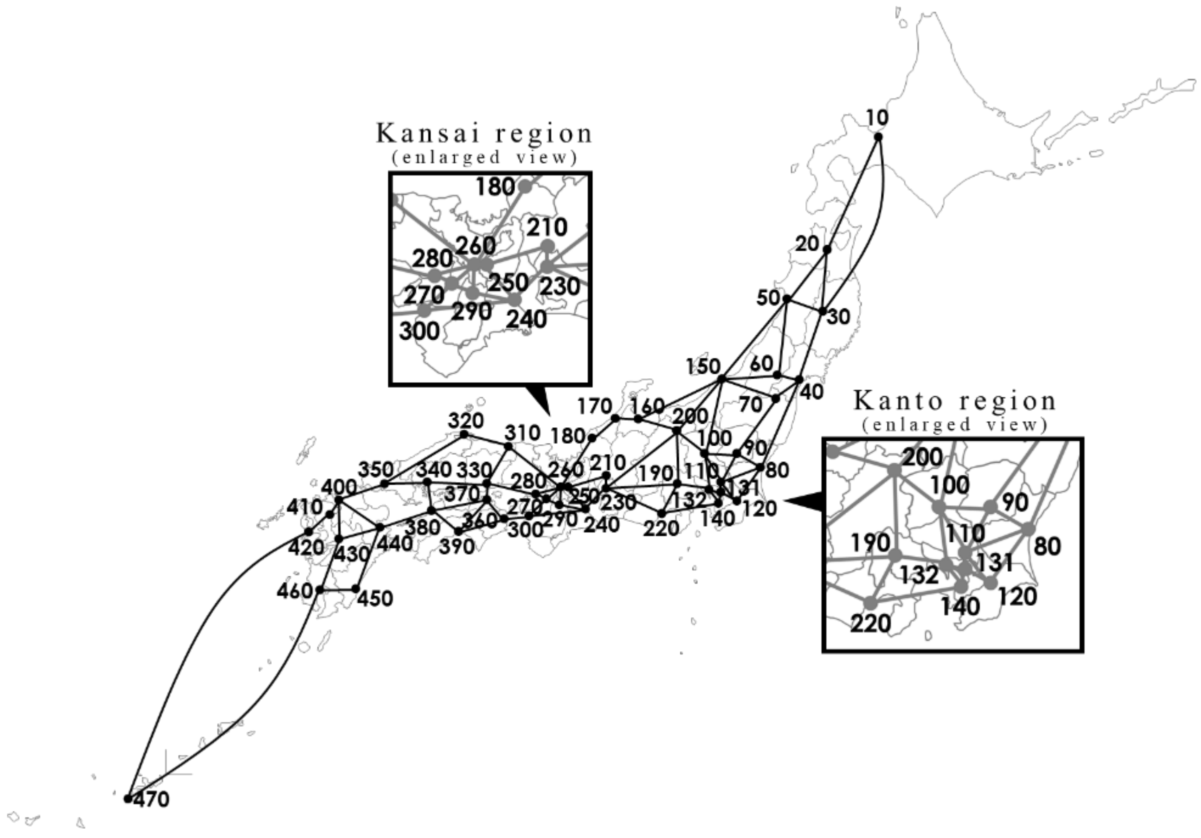

Figure 4-2 JPN 48 network model, number aside nodes represent a city in Japan.

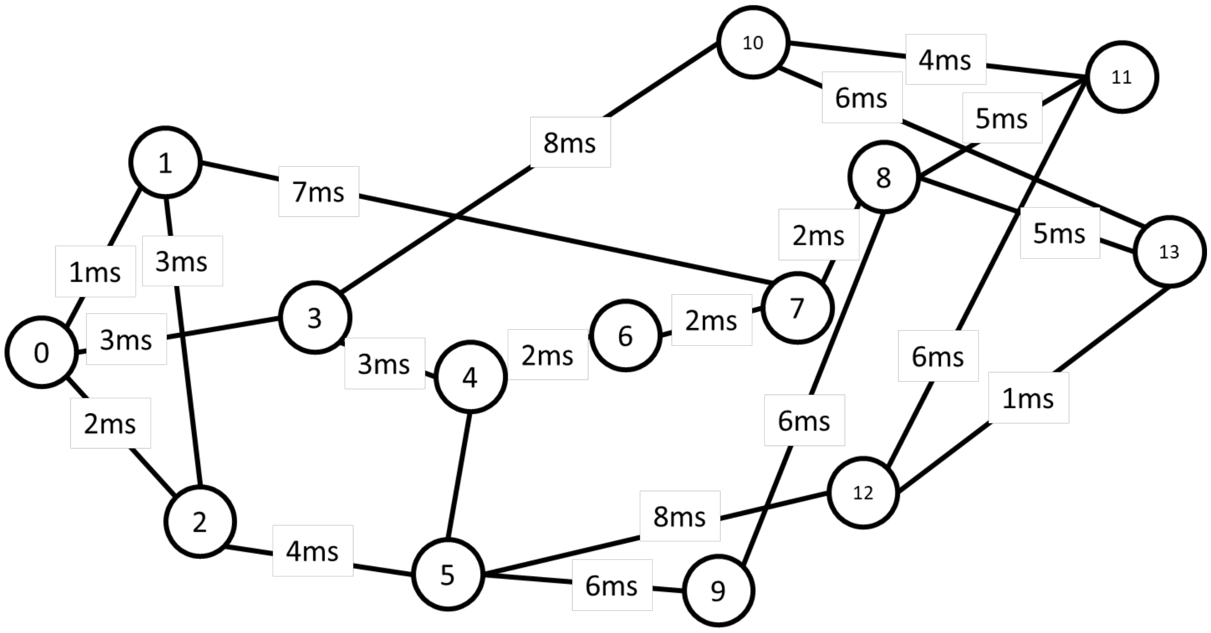

Figure 4-3 NSFNET network model.

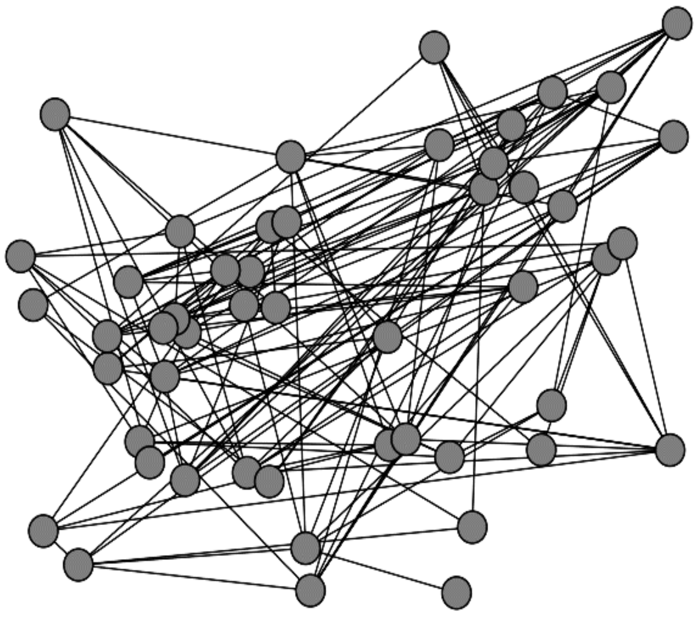

Figure 4-4 Random graph network model.

4.5.3. Evaluation on performance of end-to-end delay

The end-to-end delay of SFC is the delay of substrate path from the src to the dst of an SFC for service delivery. We evaluate our proposed DP based algorithm with comparison of other five algorithms described above on different substrate network topologies. Specifically, KS-1 is the k-shortest paths algorithm with $k = 1$, and KS-10 is the k-shortest paths algorithm with $k = 10$. BC is the betweenness centrality algorithm.

To evaluate how each algorithm could achieve a minimized end-to-end delay, we first consider that the network resources are sufficient for each algorithm. We vary the number of SFC requests from 100 to 500 and vary the number of VNFs in an SFC from 2 to 5. For each SFC request, the src and dst of an SFC are randomly selected from the substrate network. We evaluate the impact of number of VNFs as well as the impact of size of network on end-to-end delay by varying the number of VNFs in an SFC and number of substrate nodes. For each number of VNFs, we repeat 100-time experiments and in each time experiment, the src and dst of SFC are randomly selected from the substrate network.

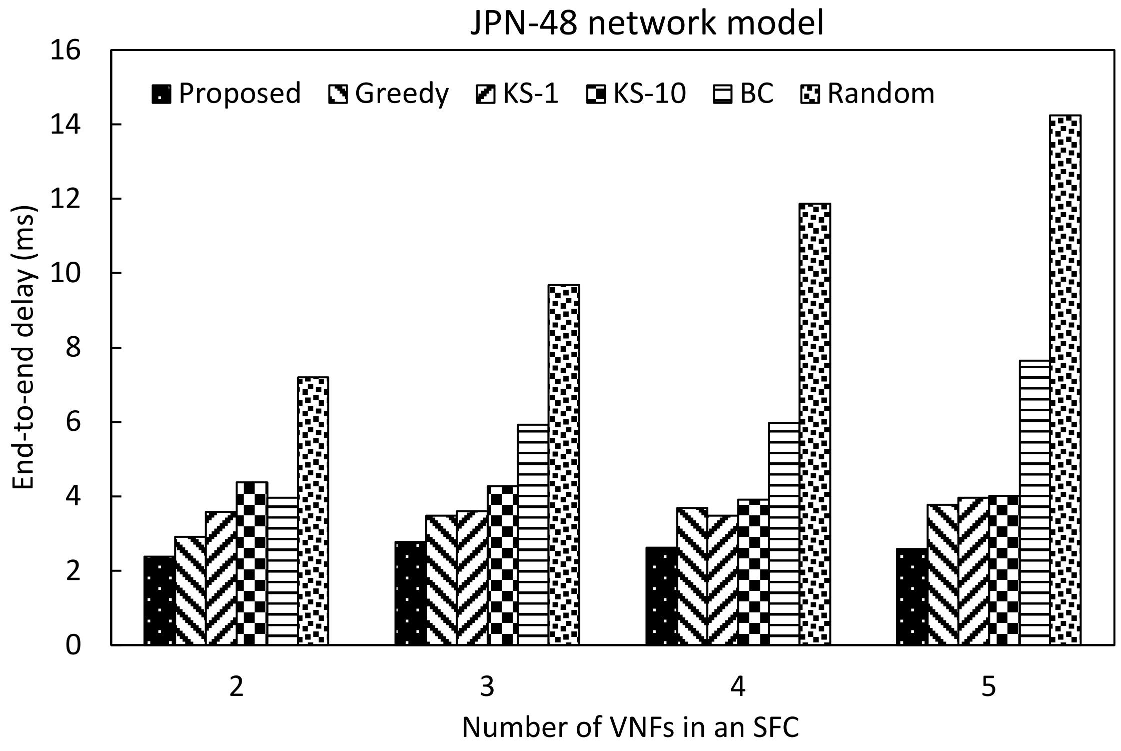

Figure 4-5 End-to-end delay on the JPN 48 network model.

a) End-to-end delay performance on JPN48 network model

Figure 4-5 shows the results of performance in terms of end-to-end delay with various number of VNFs on the JPN 48 substrate network model. The results reflect that our proposed algorithm obtains a relatively low end-to-end delay in comparison of other algorithms at each number of VNFs.

Specifically, the end-to-end delay of our proposed algorithm is $2.37ms$ when the number of VNF is 2. In comparison, the end-to-end delay are $2.91ms, 3.58ms, 4.37ms, 3{,}96ms$, and $7.20ms$, for Greedy, KS-1, KS-10, BC and Random algorithms respectively. In addition, the end-to-end delay of our proposed algorithm is 2.58 when the number of VNF is 5, whereas the end-to-end delay are $3.78ms, 3.97ms, 4.00ms, 7{,}65ms$, and $14.24ms$ for Greedy, KS-1, KS-10, BC and Random algorithms respectively.

The results demonstrate that our proposed algorithm is superior to all other algorithms. Our algorithm reduces the end-to-end delay with a maximum 67.08% and 81.95% comparing with random algorithm at the number of VNFs 2 and 5, respectively.

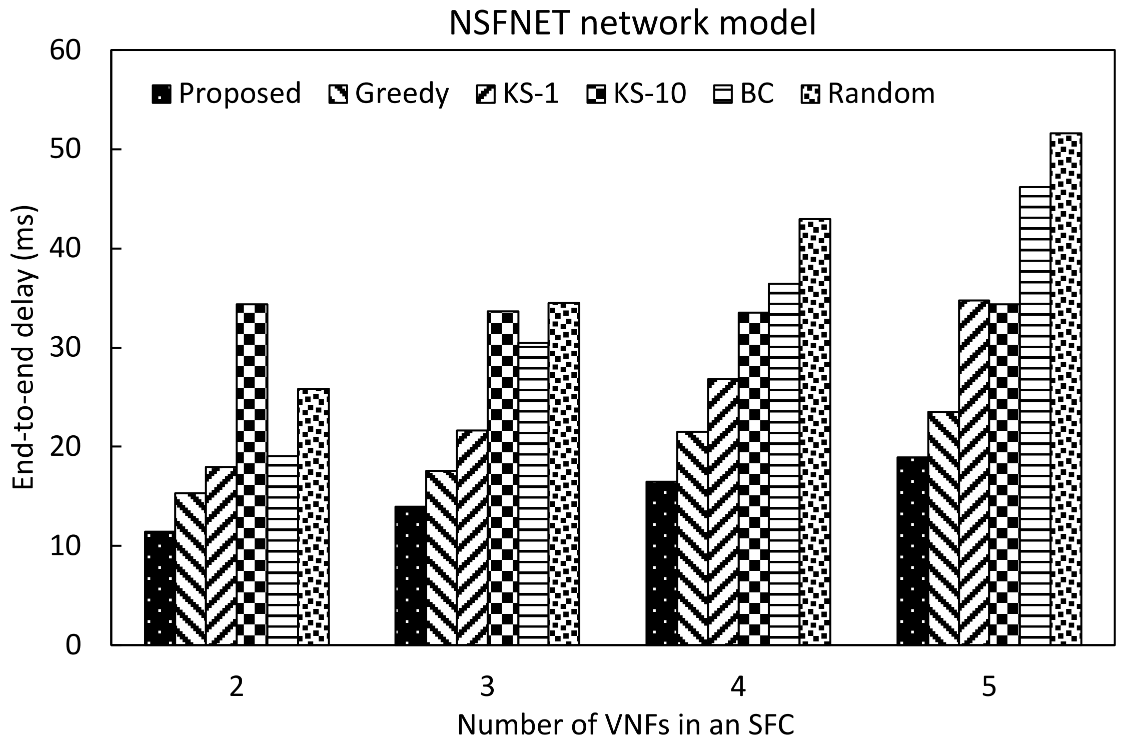

Figure 4-6 End-to-end delay on NSFNET network model.

b) End-to-end delay performance on NSFNET network model

Figure 4-6 shows the results of performance in terms of end-to-end delay with various number of VNFs on the NSFNET substrate network model. Our proposed algorithm achieves a better result in terms of low end-to-end delay.

The end-to-end delay of our proposed algorithm is $11.47ms$ at the number of VNFs of an SFC 2, whereas the end-to-end delay are $15.31ms, 17.96ms, 34.35ms, 19.03ms$, and $25.81ms$ for Greedy, KS-1, KS-10, BC and Random algorithms respectively. In addition, the end-to-end delay of our proposed algorithm is 18.91 when the number of VNF is 5, whereas the end-to-end delay are $23.50ms, 34.75ms, 34.39ms, 46.17ms$, and $51.62ms$ for Greedy, KS-1, KS-10, BC and Random algorithms respectively.

The results reflect that our algorithm has a better performance than all other algorithms in reduction of end-to-end delay with a maximum 55.57% and 63.37% at the number of VNFs 2 and 5, respectively.

c) End-to-end delay performance on Random Graph network model

Figure 4-7 through Figure 4-10 illustrate the performance of end-to-end delay on four different random substrate network models with number of nodes 100, 200 at edge connectivity probability 0.05 and 0.2 similar to the parameter setting in [127]. Note that a higher connectivity probability leads to a graph with larger number of edges.

Comparing four figures, the results reflect that our proposed algorithm outperforms other algorithms on all network models at different number of VNFs in an SFC.

Specifically, the end-to-end delay of our proposed algorithm is $7.90ms, 8.61ms, 9.60ms,$ and $10.83ms$ on the random graph model with 100 nodes and 0.05 edge connectivity probability, whereas, the end-to-end delay of our algorithm is $4.56ms, 5.60ms, 6.56ms,$ and $7.54ms$ on the random graph model with 100 nodes at 0.2 edge connectivity probability with respect to the number of VNFs in an SFC at 2, 3, 4, and 5, respectively.

Similarly, the end-to-end delay of our proposed algorithm is $6.58ms, 7.24ms, 8.08ms,$ and $9.13ms$ on the random graph model with 200 nodes and 0.05 edge connectivity probability, whereas, the end-to-end delay of our algorithm is $3.99ms, 4.80ms, 5.72ms,$ and $6.67ms$ on the random graph model with 200 nodes at 0.2 edge connectivity probability with respect to the number of VNFs of an SFC at 2, 3, 4, and 5, respectively.

It can be found that the end-to-end delay is smaller when the edge connectivity is larger on both 100 and 200 nodes random graph. This is reasonable since a larger value of edge connectivity probability leads to a larger number of edges in the graph, which could increase the degree of free to find a path with a shorter end-to-end delay between two nodes. However, the number of nodes of random graph at a fixed edge connectivity probability does not affect the end-to-end delay a lot. This is because that since the edge connectivity probability is fixed, as the number of nodes increases, the number of edges increases accordingly, which does not affect the density of the graph and the distance of two nodes.

By comparing with Random Algorithm, our algorithm achieves $64.26\%{\sim}75.75\%$ and $60.20\%{\sim}67.14\%$ reduction in terms of end-to-end delay on the random graph model with 100 nodes at 0.05 edge connectivity probability and 0.2 edge connectivity probability, respectively. Besides, our algorithm achieves $62.28\%{\sim}74.17\%$ and $59.72\%{\sim}66.42\%$ reduction in terms of end-to-end delay on the random graph model with 200 nodes and 0.05 edge connectivity probability and 0.2 edge connectivity probability, respectively.

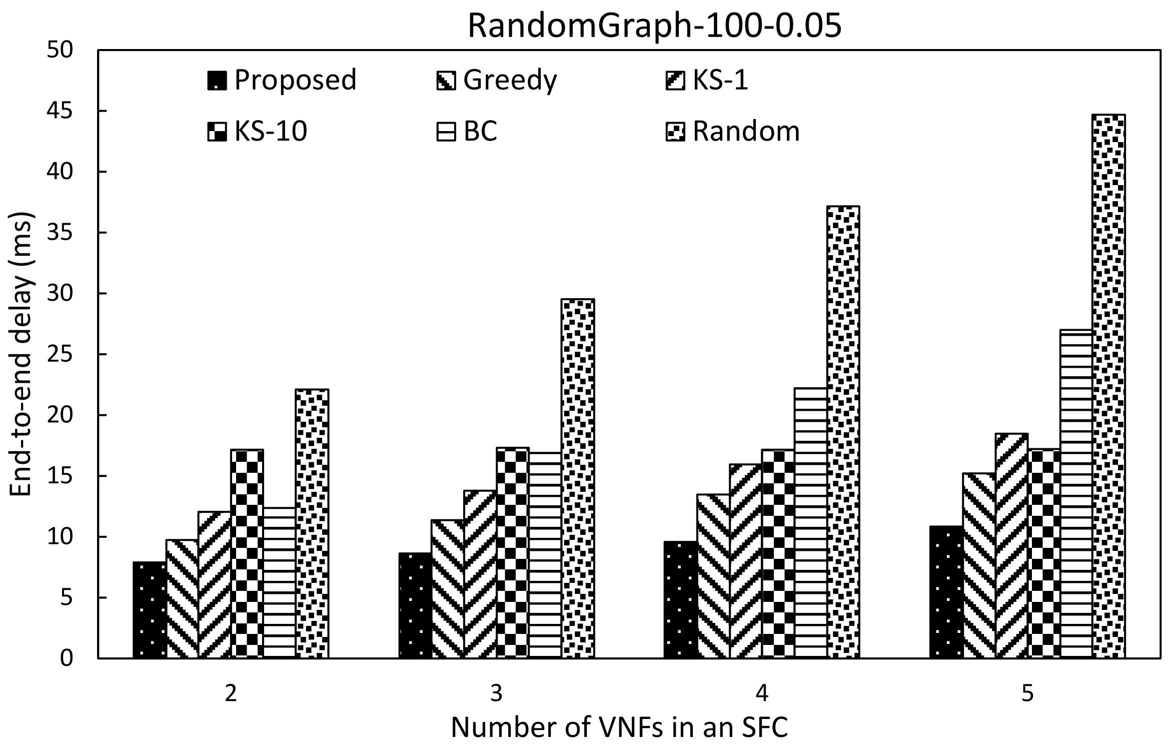

Figure 4-7 End-to-end delay on Random Graph network model.

(100 nodes, 0.05 edge connectivity probability)

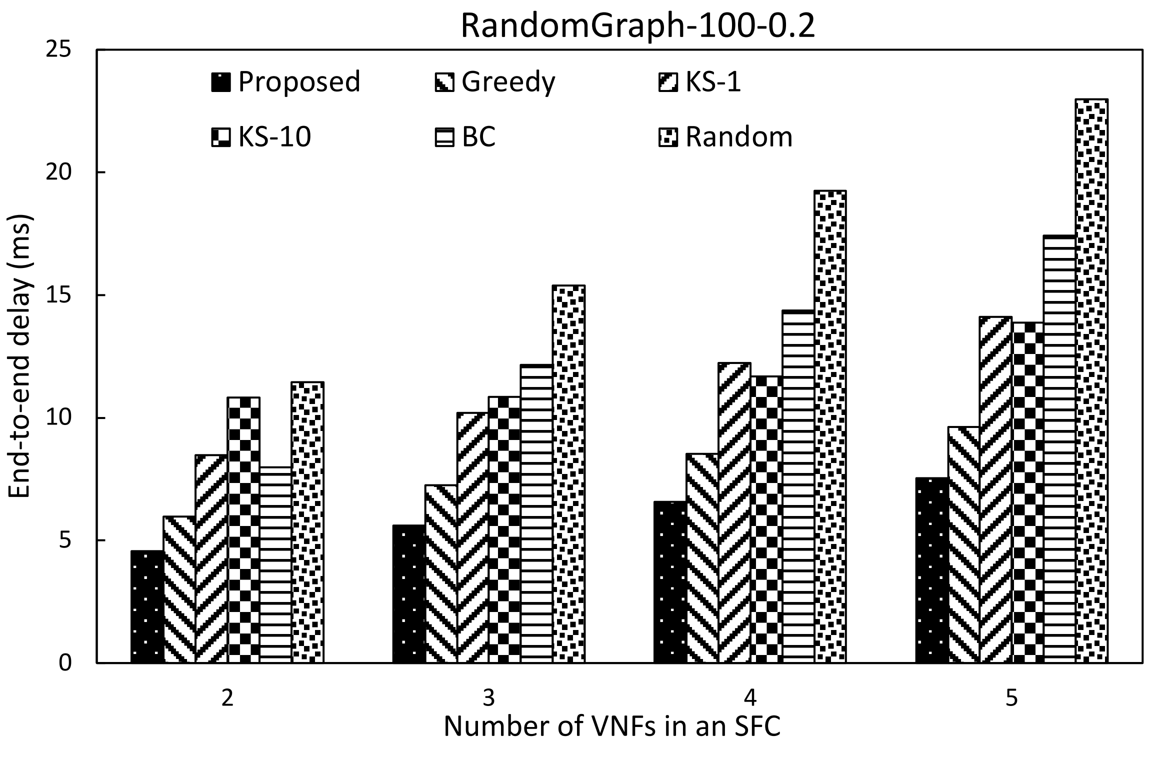

Figure 4-8 End-to-end delay on Random Graph network model.

(100 nodes, 0.2 edge connectivity probability)

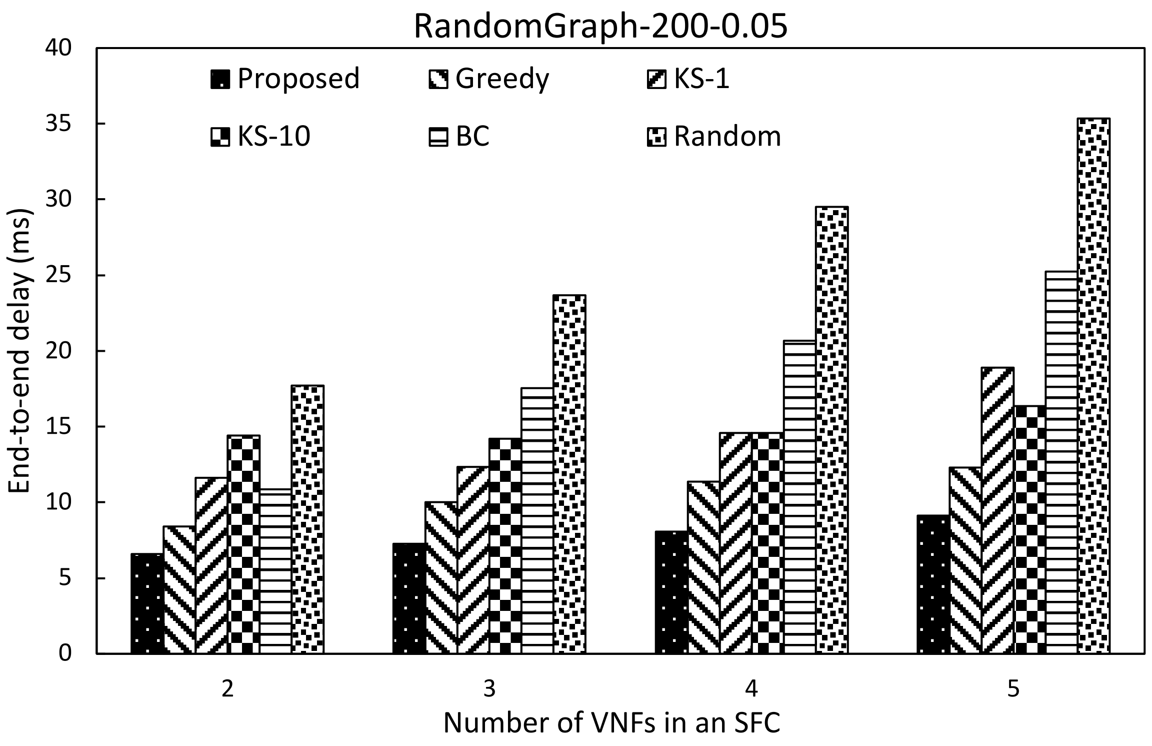

Figure 4-9 End-to-end delay on Random Graph network model.

(200 nodes, 0.05 edge connectivity probability)

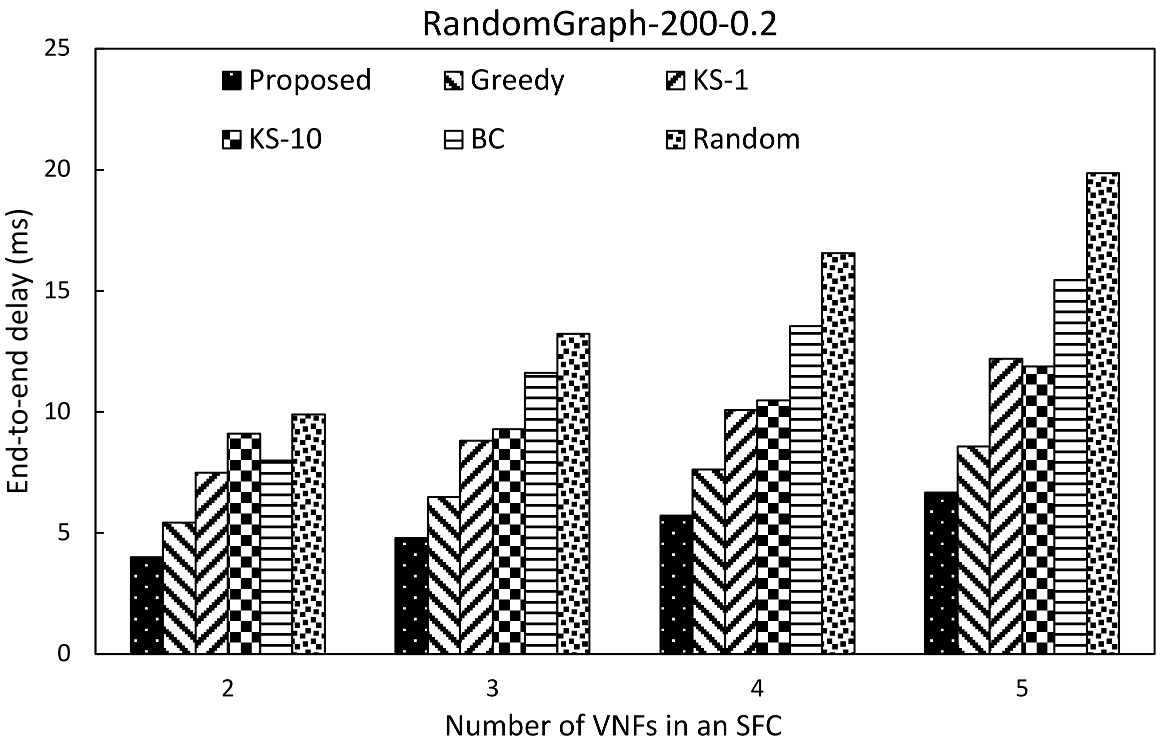

Figure 4-10 End-to-end delay on Random Graph network model.

(200 nodes, 0.2 edge connectivity probability)

4.5.4. Evaluation on performance of acceptance rate

The acceptance rate is the ratio of number of SFC requests that have been successfully deployed on the substrate, with total number of SFC requests. A higher acceptance rate indicates that the network can receive larger number of SFC requests which potentially utilizes the network resource efficiently and obtains a higher revenue to network operator.

We evaluate our proposed DP based algorithm with comparison of other five algorithms described above on different substrate network topologies. Specifically, KS-1 is the k-shortest paths algorithm with k=1, and KS-10 is the k-shortest paths algorithm with k=10. BC is the betweenness centrality algorithm.

To evaluate the performance in terms of acceptance rate, we vary the number of SFC requests from 100 to 500 and vary the number of VNFs in an SFC from 2 to 5. For each SFC request, the src and dst of an SFC are randomly selected from the substrate network. The amount of CPU requirement for one VNF is randomly chosen from (5, 10), and the amount of bandwidth requirement for one SFC is randomly chosen from (5, 10), respectively. We evaluate our algorithm in comparison with other algorithms on different substrate network models. The CPU capacity of each node in substrate network is set as 100 units, and the bandwidth capacity of each edge in substrate network is set as 1000 units, respectively.

a) Acceptance rate on JPN48 network model

Figure 4-11 through Figure 4-14 show the performance results in terms of acceptance rate with respect to the number of SFC requests and the number of VNFs on JPN48 network model. The results reflect that our proposed algorithm can outperform other algorithms with a higher acceptance rate.

Specifically, at the number of VNFs 2, our algorithm can accept all SFC requests when the number of SFC requests is 100 and 200. In comparison, the acceptance rate at the number of SFCs 200 is $90\%, 85\%, 84\%, 79\%,$ and 58% for Greedy, SK-1, SK-10, BC and Random algorithms, respectively.

Meanwhile, the acceptance rate decreases, as the number of SFCs increases. The acceptance rate is 64% when the number of SFC is 500 for our proposed algorithm, whereas the acceptance rate is no more than 55% for all other algorithms, and the acceptance rate for random algorithm is only 32% which is only half of our proposed algorithm.

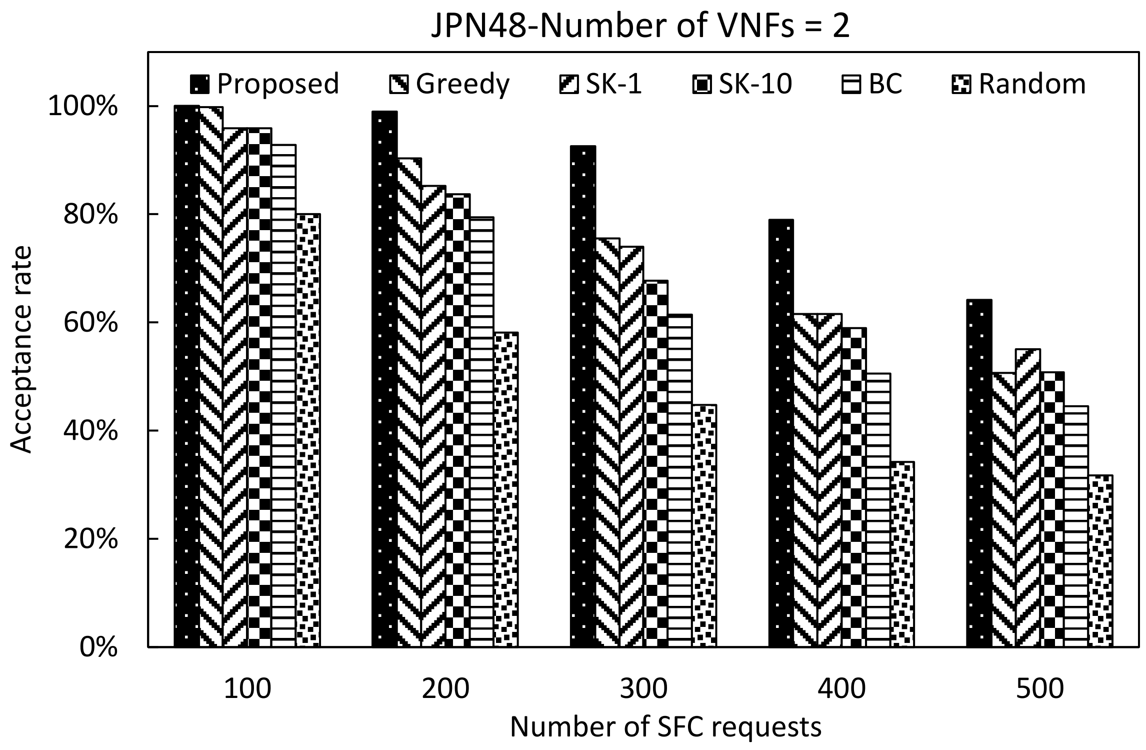

Figure 4-11 Acceptance rate on JPN48 network model. (#VNF=2)

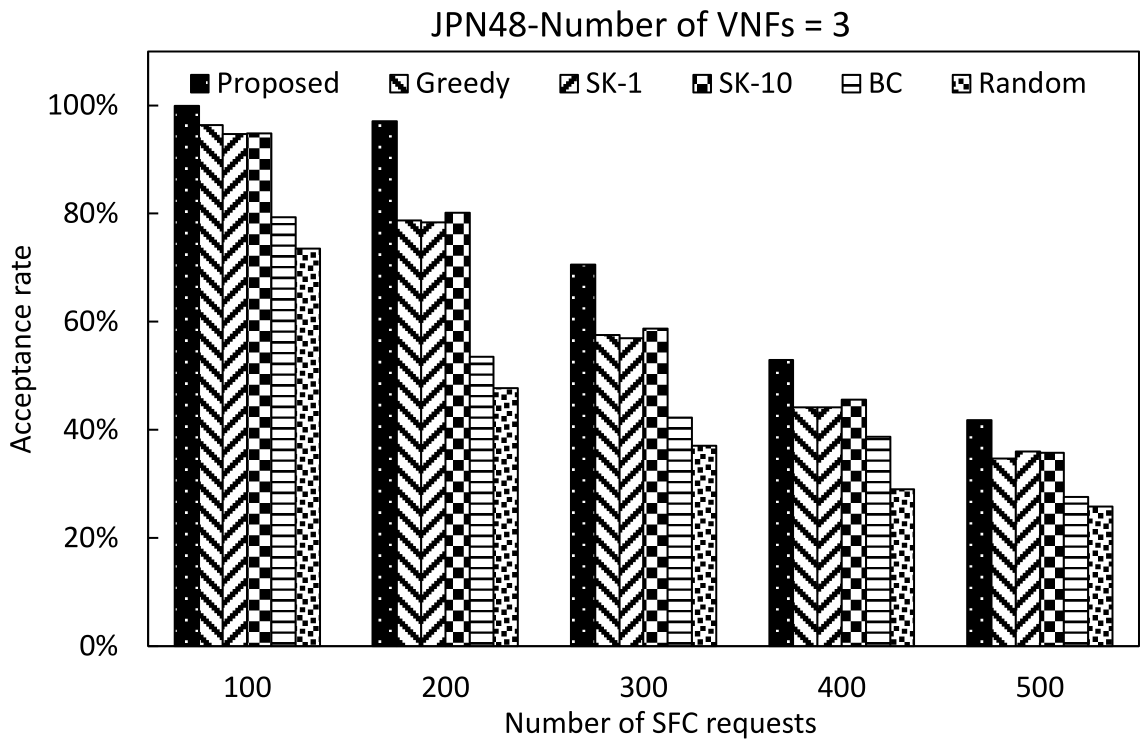

Figure 4-12 Acceptance rate on JPN48 network model. (#VNF=3)

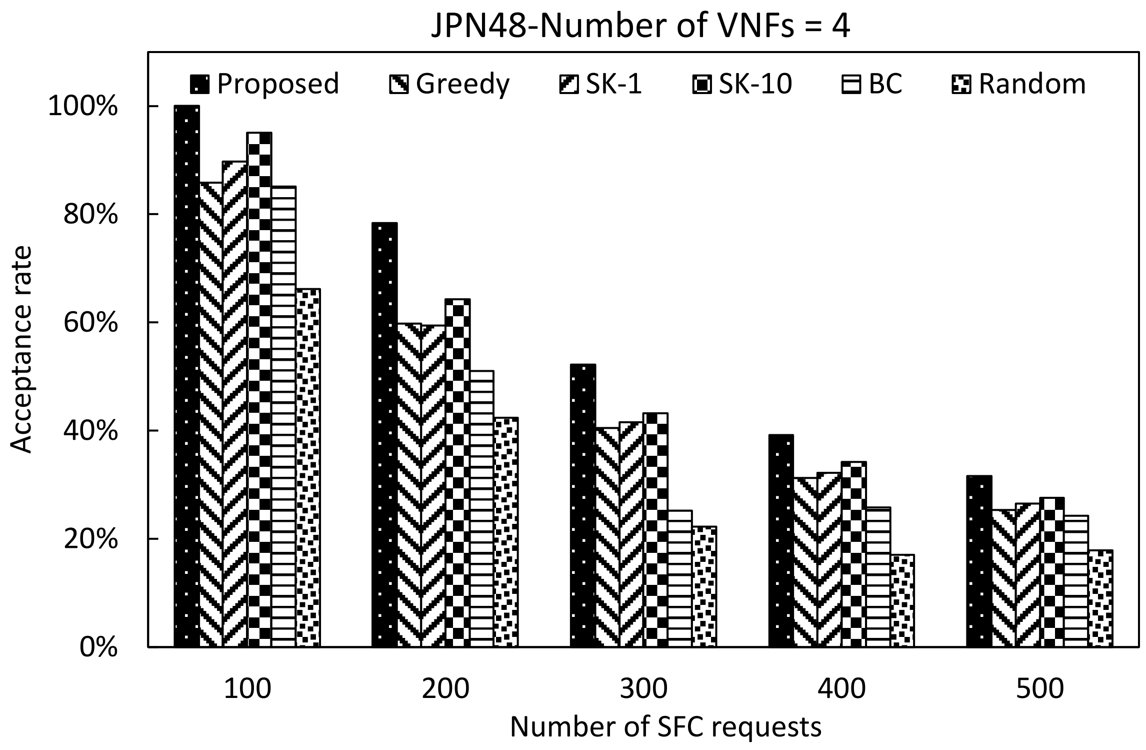

Figure 4-13 Acceptance rate on JPN48 network model. (#VNF=4)

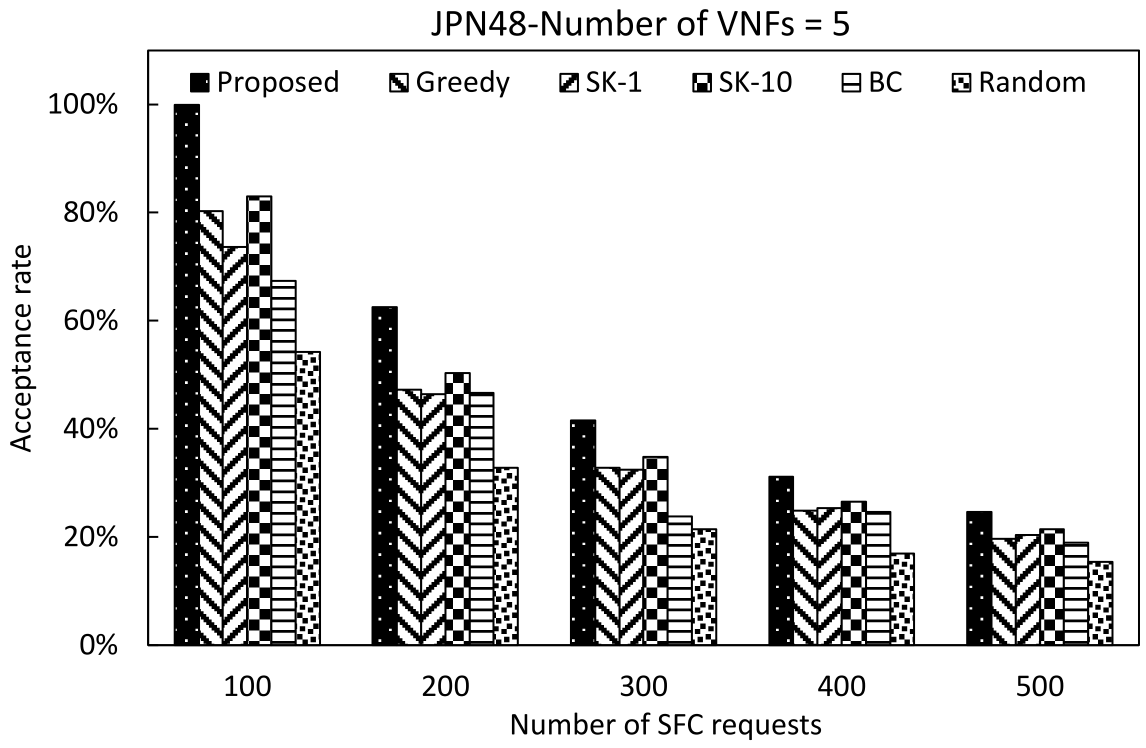

Figure 4-14 Acceptance rate on JPN48 network model. (#VNF=5)

The result of acceptance rate at the number of VNFs-3,4,5 also show a similar trend as the number of VNFs 2. Moreover, at the number of VNFs 5, our proposed algorithm can still achieve a 100% acceptance rate when the number of SFCs is 100, whereas other algorithms are all below 80%. In addition, at the number of SFCs 500, which indicates a high work load, acceptance rates of all algorithms are falling below 30% due to limited resources, however, our algorithm achieves a 25% acceptance rates which is higher than other algorithms that 20%, 20%, 21%, 19% and 15% for Greedy, SK-1, SK-10, BC and Random algorithms, respectively.

b) Acceptance rate on NSFNET network model

We next evaluate the performance of acceptance rate on our algorithms on NSFNET network model. Figure 4-15 through Figure 4-18 show the results of the evaluation at different number of VNFs in an SFC. Compared with the JPN48 network model, which consists 48 number of nodes, the NSFNET network model has a smaller of 14 number of nodes, which indicates less amount of total resources. This results in a relatively lower value of acceptance rate compared with that in JPN48 network model, with the same parameter setting, like length of an SFC or number of SFC requests.

The acceptance rate cannot reach 100% even at the number of SFCs 100 with a shorter length of SFC for all algorithms as shown at number of VNFs 2. However, even in this small network model, our proposed algorithm can achieve a relatively high acceptance rate. Our algorithm achieves an acceptance rate 95%, 63%, 46% and 37% at number of SFCs 100 with various number of VNFs 2, 3, 4, and 5, respectively, whereas all other algorithms achieve acceptance rate lower than 87%, 60%, 43%, and 33%, respectively, from which we can see that our algorithms is superior to other algorithms at most 8% in terms of acceptance rate. When the number of SFCs is 500 at various number of VNFs, which indicates the network source is extremely limited, our algorithm performs higher or no lower than other algorithms in terms of acceptance rate as shown in Figure 4-15 through Figure 4-18.

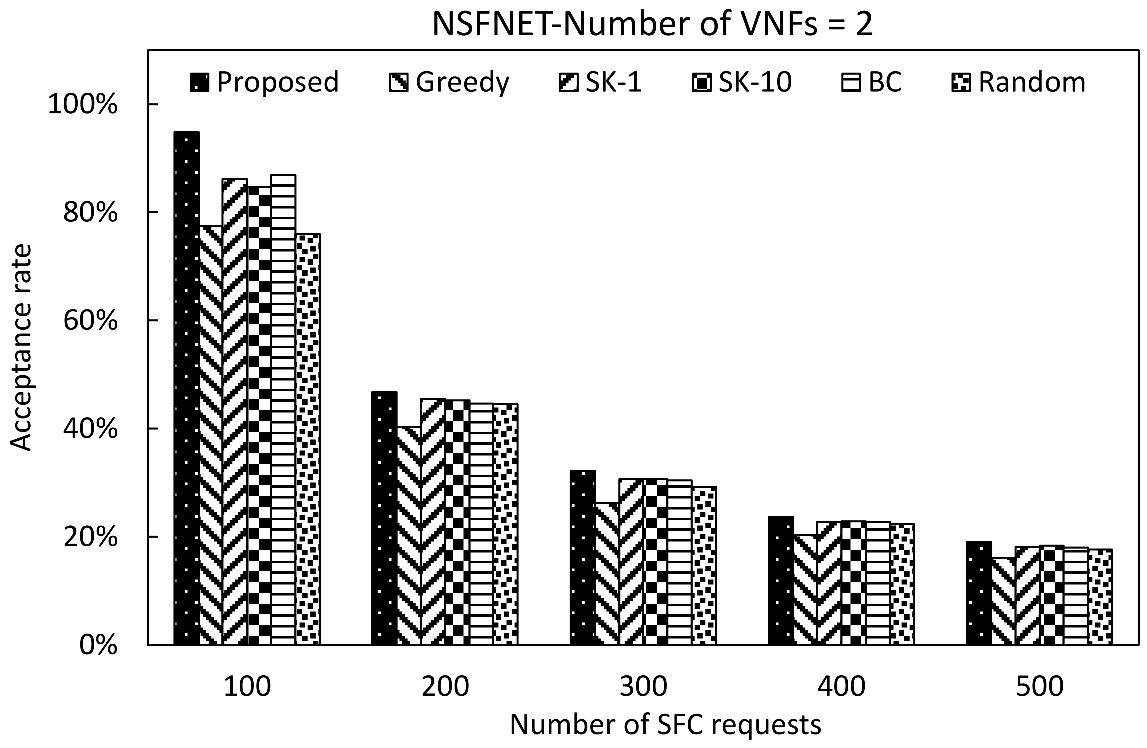

Figure 4-15 Acceptance rate on NSFNET network model. (#VNF=2)

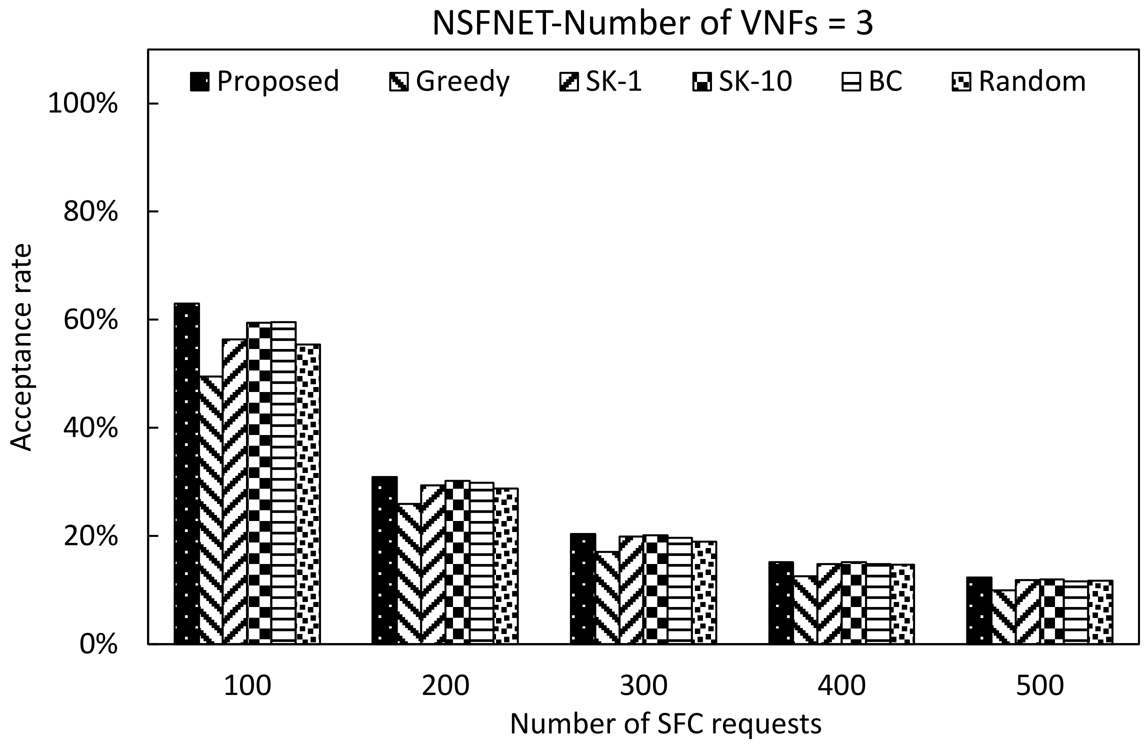

Figure 4-16 Acceptance rate on NSFNET network model. (#VNF=3)

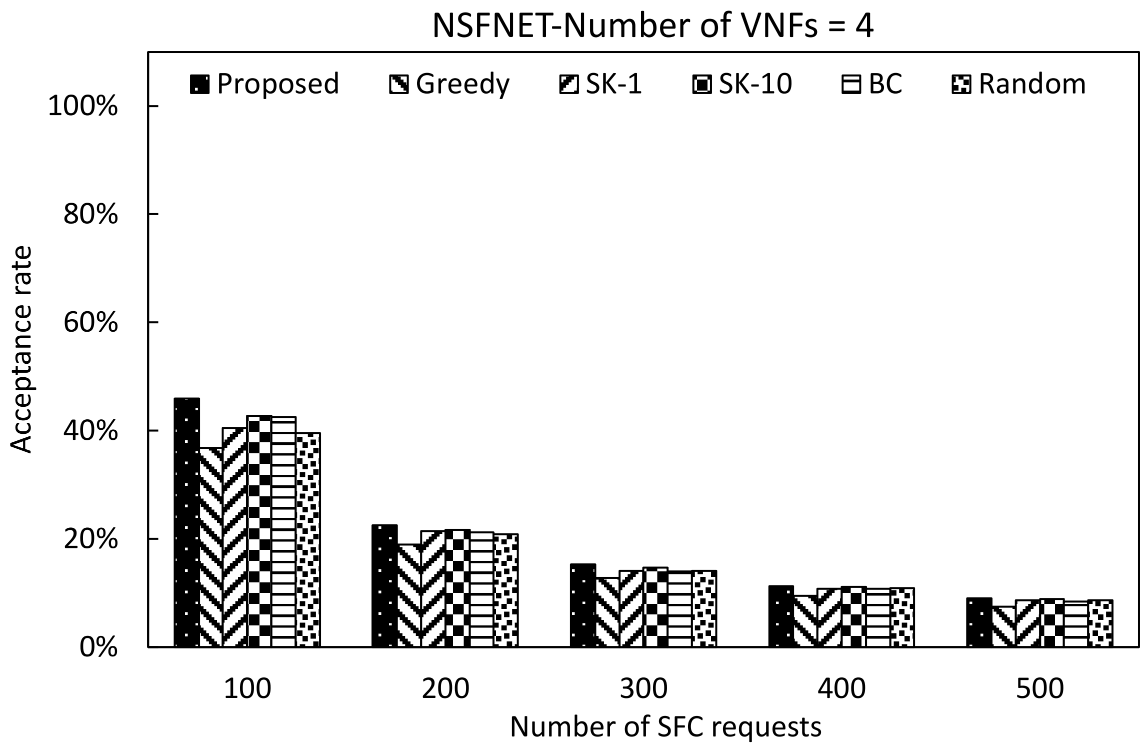

Figure 4-17 Acceptance rate on NSFNET network model. (#VNF=4)

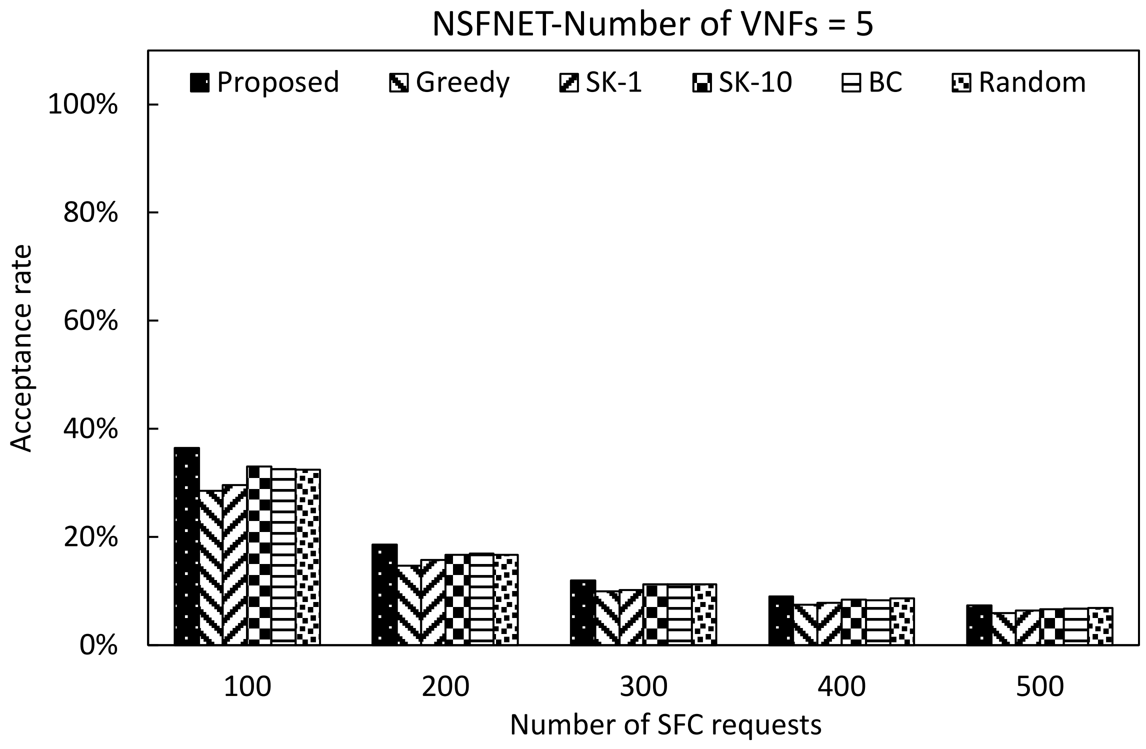

Figure 4-18 Acceptance rate on NSFNET network model. (#VNF=5)

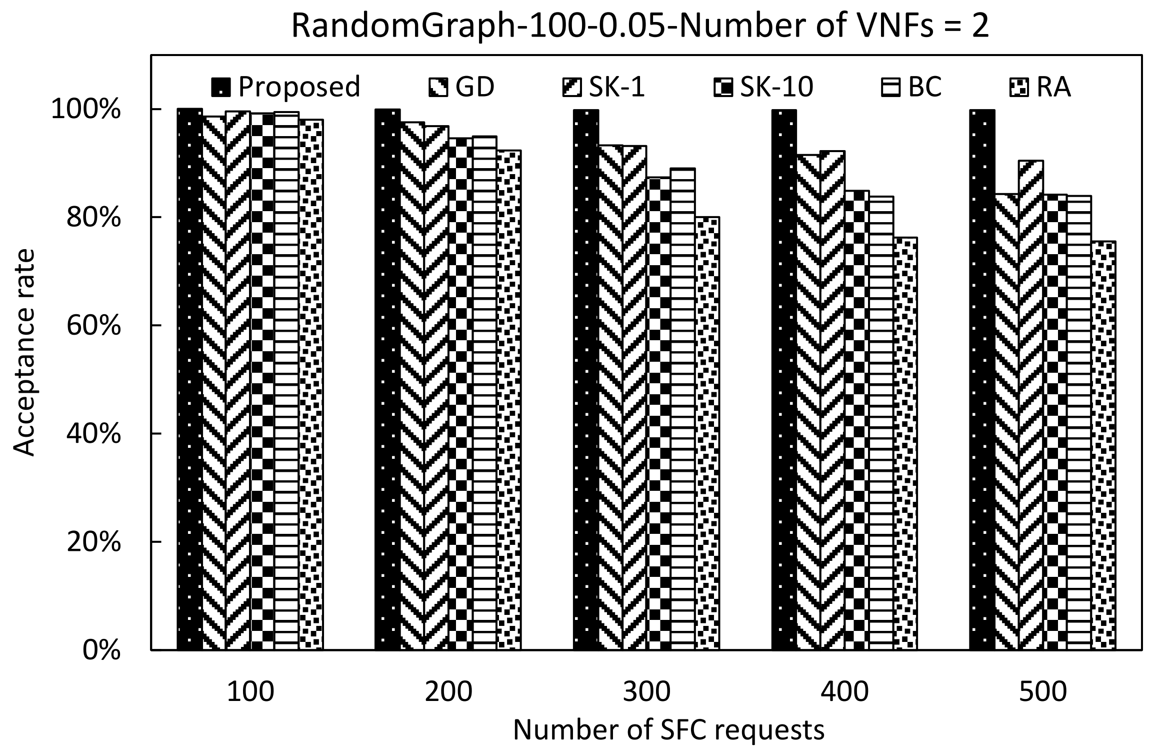

Figure 4-19 Acceptance rate on Random Graph network model. (#VNF=2)

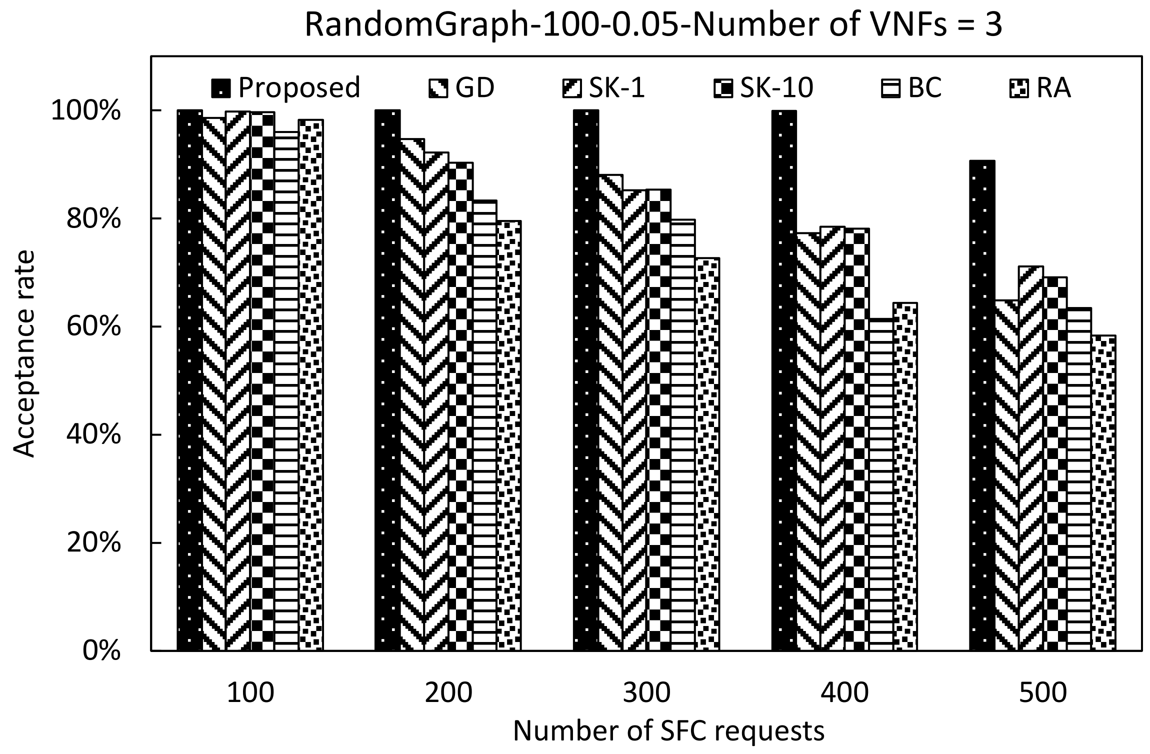

Figure 4-20 Acceptance rate on Random Graph network model. (#VNF=3)

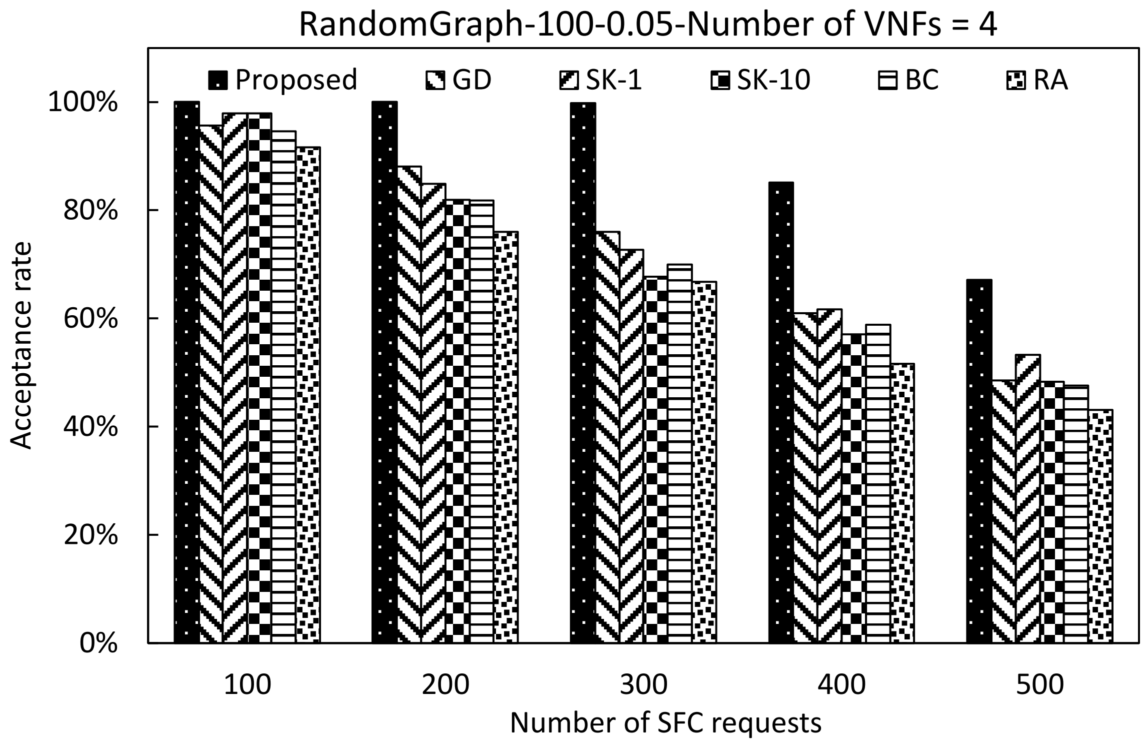

Figure 4-21 Acceptance rate on Random Graph network model. (#VNF=4)

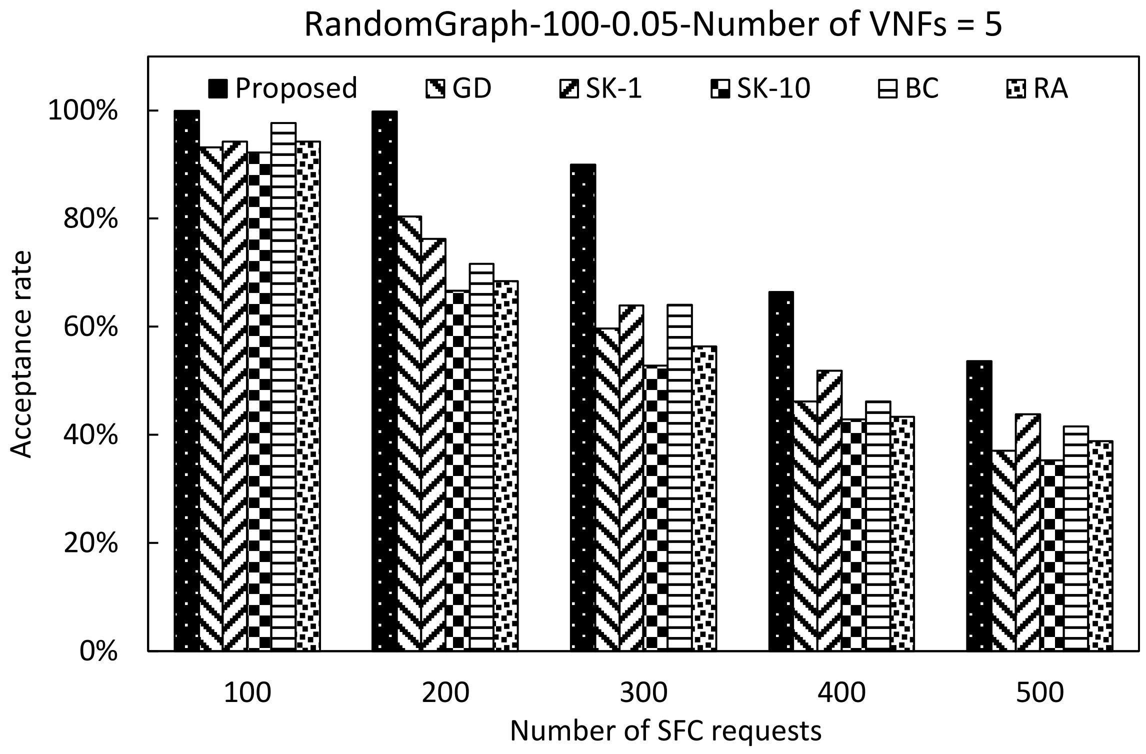

Figure 4-22 Acceptance rate on Random Graph network model. (#VNF=5)

c) Acceptance rate on Random Graph network model

Figure 4-19 through Figure 4-22 shows acceptance rate at various number of VNFs on the random network model consisting 100 number of nodes at edge connectivity probability 0.05. At the number of VNFs 2, on both random network models, our proposed algorithm can acceptance almost all the SFC requests from 100 to 500 number of SFCs. The acceptance rate of our proposed algorithm starts decreasing at 400 number of SFCs when number of VNFs is 3, at 300 number of SFCs when number of VNFs is 4, and at 200 number of SFCs when number of VNFs is 5, respectively. This is reasonable since an SFC with a larger number of VNFs requires more resources. As larger number of SFCs arrive, more SFCs are rejected due to limited resources. By comparing with Random Algorithm, our proposed algorithm can accept at most $24\%, 36\%, 34\%$, and 34% more SFC requests on the random network model at edge connectivity probability 0.05.

4.6. Discussion

4.6.1. Discussion on end-to-end delay performance

Previous sections show that our algorithm could find a shorter end-to-end delay path than other algorithms. The logic behind it is reasonable. Every time our algorithm deploys a VNF, it can find a shortest path from the src of SFC. By storing the intermediate results, our algorithm does not need to recalculate the path from very beginning src node but utilize the intermediate results. As all the substrate network nodes are exanimated, our algorithm utilizes the global information to achieve a relatively shorter end-to-end delay. Greedy algorithm deploys a VNF on a substrate node who has a shortest path to the previous VNF rather than to the src of SFC. k-shortest path could find a shortest path from src to dst of SFC, but it does not consider the number of nodes thus has to break some links to branch the link to other nodes, which increases the end-to-end delay. Betweenness centrality algorithm deploys VNFs on nodes with high value of betweenness centrality, but it could overload the nodes and other SFC cannot use it.

4.6.2. Discussion on acceptance rate

Previous sections show that our algorithm has a better performance than other algorithms in terms of acceptance rate. We can see that random algorithm always has the lowest acceptance rate, because random algorithm places VNFs of SFCs arbitrarily without considering optimizing resource consumptions. Greedy and k-shortest path algorithms achieve a higher acceptance rate, as they consider finding a path with a minimum delay. However, a shortest path may not have enough substrate nodes or sufficient CPU resources on the nodes to host VNFs of SFCs, which lowers the value of acceptance rate. In a smaller network model, the greedy algorithm even performs a relatively low acceptance rate, because greedy algorithm rejects an SFC request if any link or node have not sufficient resources to host the SFC. BC algorithms could achieve a relatively shorter delay path than random algorithm, but as it always deploys the VNFs on nodes who have a higher betweenness centrality value, some nodes with higher BC value could get a higher CPU workload and reject the SFC request.

4.7. Conclusion

In this chapter, we analyzed the necessities and difficulties to deploy SFCs with a desired QoS requirement in terms of end-to-end delay of SFCs. As stated in section 4.3.2, designing this algorithm is different due to complex requirements and constraints of SFCs in NFV. The algorithm must satisfy the computation and bandwidth resource constraints when deploying SFCs on the NFV system. The algorithm must keep the order of VNFs of SFCs when routing traffic through the SFC.

To tackle this problem, we presented an SFC placement algorithm with satisfying CPU and bandwidth resource constraints, keeping the order of VNFs and achieving a desired minimum end-to-end delay of SFC. The proposed algorithm is designed based on dynamic programming technique, which obtains a global information of the network by recording the intermediate information when searching the solution. We described the details of the algorithm.

We also conducted experimental studies on different substrate network models in comparison with other algorithms in literature. The results show that our proposed algorithm performs better than other algorithms in terms of both end-to-end delay and acceptance rate. This indicates that our algorithm can achieve a relatively better QoS guarantee with efficient network resource utilization.

Specifically, our algorithms achieved a maximum 81.95% reduction of end-to-end delay comparing with random algorithm at the number of VNFs 5 on JPN48 network model. On the other network model, our algorithms achieved more than 50% reduction of end-to-end delay comparing with random algorithms. In addition, our algorithm has a good performance with more or no less acceptance rate than other algorithms on different network models. These results indicate our algorithm can achieve a better QoS requirement and efficient resource usage.

Chapter 5

5. Distributed Service Function Chain Placement Algorithm

5.1. Introduction

Network function virtualization (NFV) separates the software implementation of network function from hardware by leveraging virtualization technologies. In the context of NFV, network services are composed of a series of ordered virtual network functions (VNFs) that are deployed on the network function virtualization infrastructure (NFVI). This separation way is different from the traditional network functions that is integrated into a dedicated hardware and deployed in a network operator's server room or on-premise of custom. This paradigm of NFV improves the capital and operational efficiencies of network, increases the flexibility and scalability of network, accelerates the speed of service innovation, and reduces the power usage of hardware [6].

Because of the advantages of NFV, many network operators are planning to integrate NFV technology into their network [18]. However, the network operators must guarantee the performance of the network service in terms of service quality while benefitting from the NFV. The white paper in [132] summaries NFV service quality metrics associated with quality criterion such as speed, accuracy, and reliability, and service/lifecycle category such as orchestration, operation. For instances, packet delay, packet loss ratio, and network outage are belonged to speed, accuracy, and reliability criterion of virtual network operation category of service metric, respectively. Moreover, each NFV service metric can impair the VNF user service quality in terms of provisioning latency, scheduling latency, packet delay, etc.

One of the important metrics is anti-affinity rules, as placement policy compliance metrics [132]. The anti-affinity rules describe the relationship between virtual machine and hosts [133], that two VMs should be deployed distributedly on different hosts. ETSI resilience requirement white paper [134] points out that the distributed deployment of VNFs should be considered to prevent from failure of VNFs. More specifically, distributed deployment of SFC can improve network service robustness by avoiding VNFs to sharing same physical resources [136].

Moreover, according to the IDC survey [137] and white paper [138] of IBM, it costs $100, 000 per hour if an infrastructure failed, and $500,000 to $1million per hour if a critical application failed in organization such as Fortune 1000 company. Therefore, a long recovery time is not tolerable for both user and provider. Nevertheless, if two VNFs are deployed in a same host, once this host is failure due to an incident such as power outages, software problems, misconfiguration or disaster, both VNFs cannot perform function correctly. Consequently, this failure increases the recovery time to reestablish the end-to-end network service delivery, as it requires restarting the VNFs that in the same failure substrate nodes or reconfigure the VNFs. All these show that distributed deployment of network service is critical for preventing the network service from single-point failure and helps to restore services quickly to users affected by an incident [139].

In addition, VNFs in an SFC should be placed following a specific order to process traffic flow correctly. Consolidating multiple VNFs on a same substrate node would leads the traffic flow travelling back-and-forth in the substrate network, thus increase the risk of traffic loops in the substrate network.

Furthermore, a centralized deployment of VNFs can lead to an unbalanced network resources utilization and result in the overloading of some hosts for providing sufficient computation resources. The consequences of overloading increase risks of a node failure under high resource usage loads [100], cause a service level agreement (SLA) violations or decrease the performance of VNFs [140].

An example of network performance degradation in terms of packet loss is described in [19] due to collocation of VNFs of an SFC in a same host. Two VNFs of SFC consume 100 MIPS for processing normal traffic, and 200 MIPS for processing peak traffic. The probability to reach peak traffic is 30% for VNFs. If both VNFs of an SFC are deployed in a same server with capacity of 400 MIPS, the probability of both VNFs are reaching peak traffic is 30%, and once both VNFs are reaching peak traffic, it will need all capacity of 400 MIPS of host for processing traffic, which will result in packet loss. However, the probability is only 9% for two VNFs that reach peak traffic and consume all capacity of 400 MIPS of host if these two VNFs are belonged to different SFCs. Therefore, researchers in [19] explicitly claimed that VNFs of the same SFC should not be deployed on the same server.

Motivated by these concerns, in this chapter, we address the problem of deploying VNFs of an SFC in a distributed manner with satisfying distributed deployment (anti-affinity) requirement while keeping a desired end-to-end delay of SFCs. More specifically, VNFs belonged to the same SFCs should not be deployed on the same substrate network node to avoid network issues mentioned above, and the end-to-end delay of substrate path of an SFC should be minimized simultaneously. We also consider that one type of VNF can be deployed on only a certain number of candidate substrate network nodes. This differs to the scenario in previous chapter in which one type of VNF can be deployed on all substrate network nodes. This scenario is practical when considering placement policy compliance metrics. As described in [134], some VNFs should not be deployed on some certain substrate network nodes. For example, some service providers prefer to deploying VNFs in a certain region because of legislative restrictions or high cost of some substrate network nodes. Another reason that this scenario is practical is that some VNF instances may have been already deployed on the substrate nodes, and it could be better to select one instance to host the VNF in an SFC rather than installing a new VNF instance into a new server, which may increase the power consumption or extra license fee as described in Chapter 2.

However, it is challenging to distributed deploy VNFs while keeping a desired end-to-end delay. The first challenge is that the VNF deployment scheme must decide which VNF a substrate network node should host, if this substrate network node could host more than one VNFs of an SFC. The second challenge is that while the VNF deployment scheme making the decision of which VNFs a substrate node should host, it must guarantee a relative lower end-to-end delay for SFC to achieve a desired QoS. The third challenge is that the VNF deployment scheme must keep the order of VNFs of SFC for successful network service delivery. It can be obvious that varies deployment strategies would result a very different results of deployment of SFCs in terms of distributed deployment requirement, end-to-end delay, and the order of VNFs of an SFC, and it is obvious that random deployment of VNFs cannot achieve these requirements.

To address this problem, we utilize a modified layered graph algorithm to realize the distributed deployment of VNFs in an SFC while satisfying distributed deployment requirement. The layered graph algorithm is an easy way and a proper tool to find a shortest path between src and dst nodes of SFC, while guaranteeing the order of VNFs simultaneously. By leveraging the advantages of layered graph, we apply it to solve the distributed deployment of VNFs problem to jointly guarantee distributed deployment requirement and a minimum end-to-end delay of SFC path.

5.2. Related work

Most of current researches on the deployment of SFCs consider constraints on network capacity and computing resources. However, location constraints like affinity and anti-affinity are critical metrics for measuring QoS of network services. Bouten et al. in [141] analyzed the affinity and anti-affinity constraints in the placement of VNFs from the aspect of efficiency, resilience, legislation, privacy, and economic. Authors proposed a set of affinity and anti-affinity constraints on both deployment of virtual functions and virtual edges. Jacobs et al. in [142] proposed a solution for measurement of two VNFs' affinity and anti-affinity. The criterion used for measurement are classified into two types, static criteria, such as minimum CPU and memory, and dynamic criteria such as CPU usage, package loss and latency. For those VNFs that have not use any network resources, the authors proposed an affinity prediction model by leveraging artificial neural network in [143]. Zhou et al. in [19] observed that VNFs of an SFC should be placed onto different physical servers to guarantee performance and scalability of the SFC. The authors solve this distributed deployment problem by utilizing bipartite matching algorithm with maximizing vertical scalability and minimizing packet loss rate. Zhu et al. in [144] proposed a cost-efficient VNF placement strategy for IoT network. The authors find an effective placement policy for each configuration of VNF and guarantee the availability and confidentiality by leveraging affinity and anti-affinity placement rules while achieving a low cost. Allybokus et al. in [136] formulate an ILP model for VNF placement with a partial order constraints and anti-affinity rules by considering reducing total financial cost. The authors also proposed a heuristic algorithm based on a linear relaxation. However, few of them consider the anti-affinity rules with a minimizing end-to-end delay of an SFC.

The layered graph approach has been applied to solve the SFC deployment problem by many previous studies. We leverage the advantages of layered graph and apply it to solve the distributed deployment of VNFs problem to jointly guarantee the anti-affinity rules and a minimum end-to-end delay of SFC path. Dwaraki et al. [111] uses layered graphs [145] to place SFC with a minimized network delay in an adaptive way. Huin et al. [146] modeled two ILP formulations and utilizes layered graph to investigate the tradeoff between bandwidth demands and number of substrate nodes that can possibly host VNFs. Tomassilli et al. [112] transformed SFC placement problem into a set cover problem based on layered graph concept to reduce overall placement cost with correct order of VNFs in SFC requests. Authors also designed a greedy algorithm as well as an optimal algorithm specific to tree topologies. Sallam et al. in [147] modified the layered graph to reduce the computation time by utilizing directed graph to remove the nodes without incoming edges. The authors proved the correctness of the layered algorithm. The authors also formulated an ILP model to solve the maximum flow problem with SFC constraints. Gu et al. in [148] also utilized layered graph to joint optimize the delay and resource allocation for SFC orchestration. However, these works did not take the anti-affinity into account. In this thesis, we take anti-affinity rules into account. We consider anti-affinity rules as distributed deployment of SFC. Specifically, we consider satisfying distributed deployment requirement which is defined as to place VNFs of an SFC onto different substrate network nodes.

5.3. System model

In this section, we first introduce necessity of distributed SFC deployment requirement and then state the preliminary assumption on the system model. Finally, we describe the substrate network and SFC model in which the notations are used in the following sections.

5.3.1. Necessity of distributed SFC deployment requirement

Distributed SFC deployment in this thesis is defined as to deploy VNFs of an SFC to different substrate network nodes. This is necessary in some cases as shown in the following:

Case 1: Overload substrate network node resource

Consider an SFC composed of two VNFs. Each VNF needs 300 units CPU resources to process normal traffic flow. CPU resource requirement of VNFs increases as the traffic flow increases. Suppose the probability to reach peak traffic of the SFC is 30%, and VNFs require 600 units resource to process peak traffic. Also suppose a substrate node have 1000 units available CPU resource capacity. If both VNFs are deployed on the substrate network, then, there will be 30% probability for the SFC overloading the substrate network node, as the summation of CPU requirement of two VNFs is 1200 whereas the substrate node has only 1000 units available CPU resources.

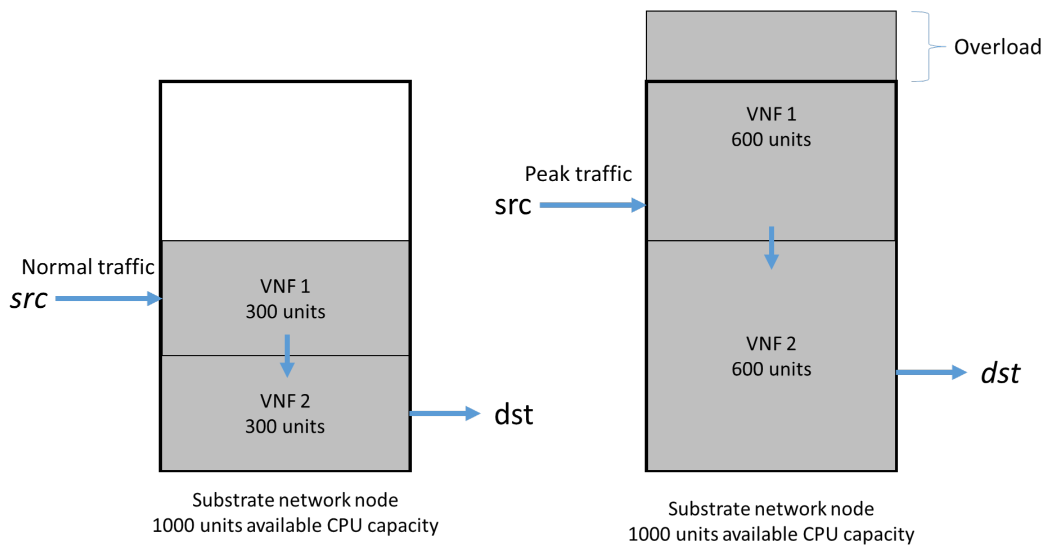

Figure 5-1 Case 1: Two VNFs are placed on one substrate node.

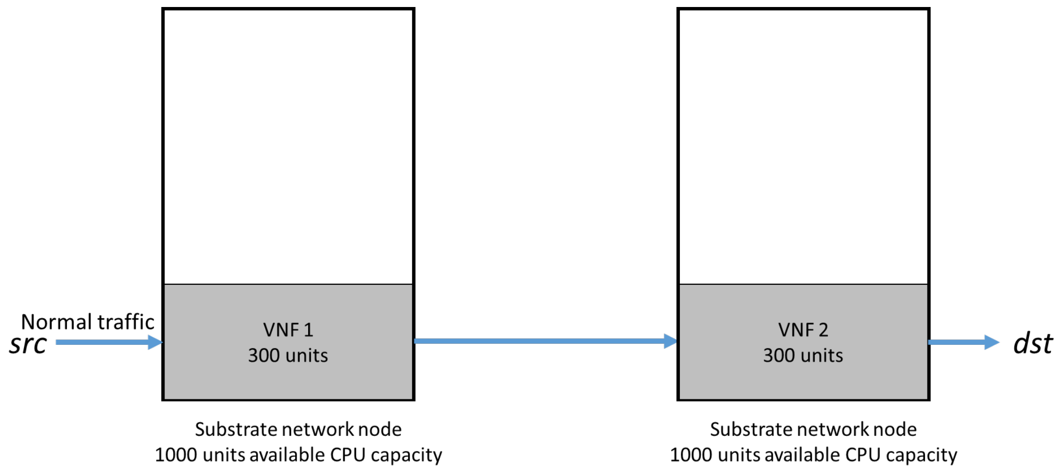

Figure 5-2 Case 1: Two VNFs are placed on two substrate nodes. (normal traffic)

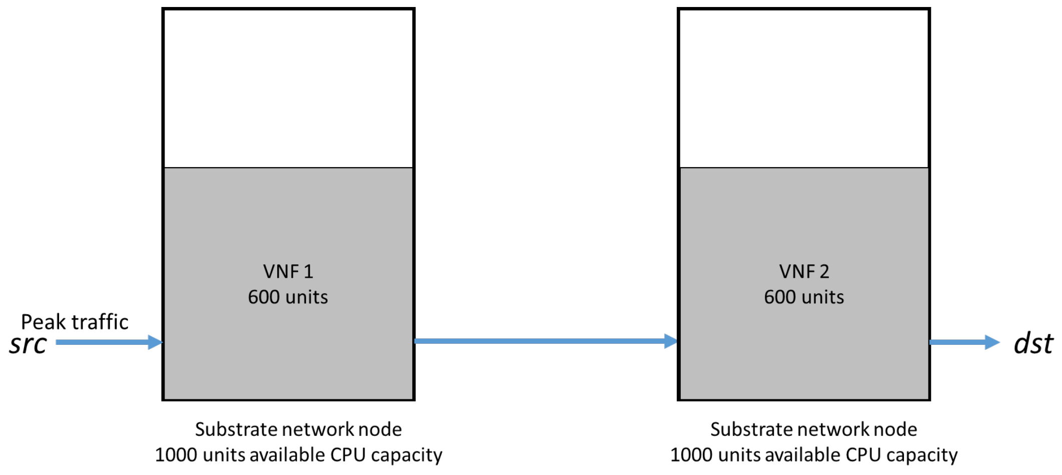

Figure 5-3 Case 1: Two VNFs are placed on two substrate nodes. (peak traffic)

Case 2: Risk of traffic loop

Consider an SFC with 3 VNFs. As shown in the following figures, if VNF1 and VNF3 are collocated on a same server, the traffic will travel back and forth between VNF1, VNF2 and VNF3, which would increase the risk of traffic loop.

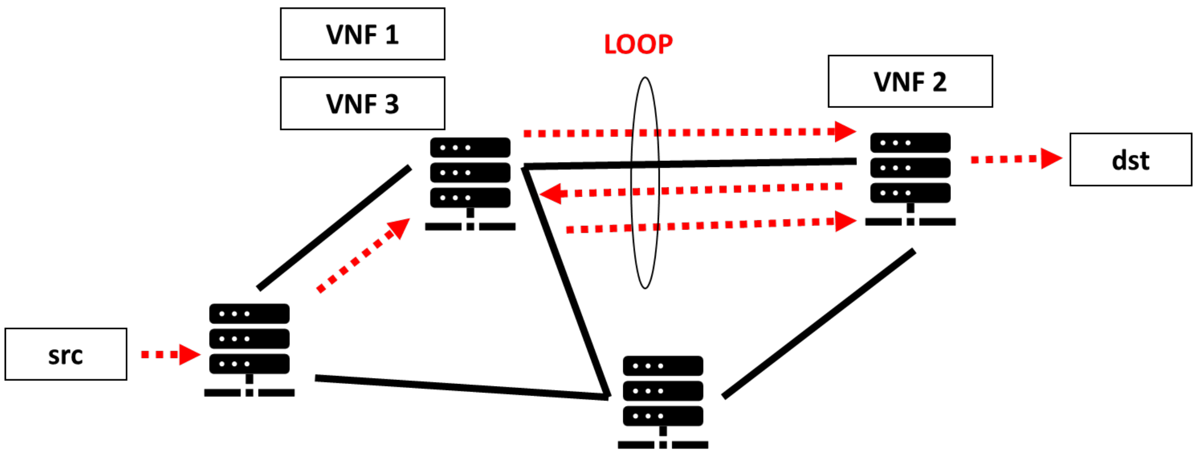

Figure 5-4 Case 2: Two VNFs are placed on a same substrate node.

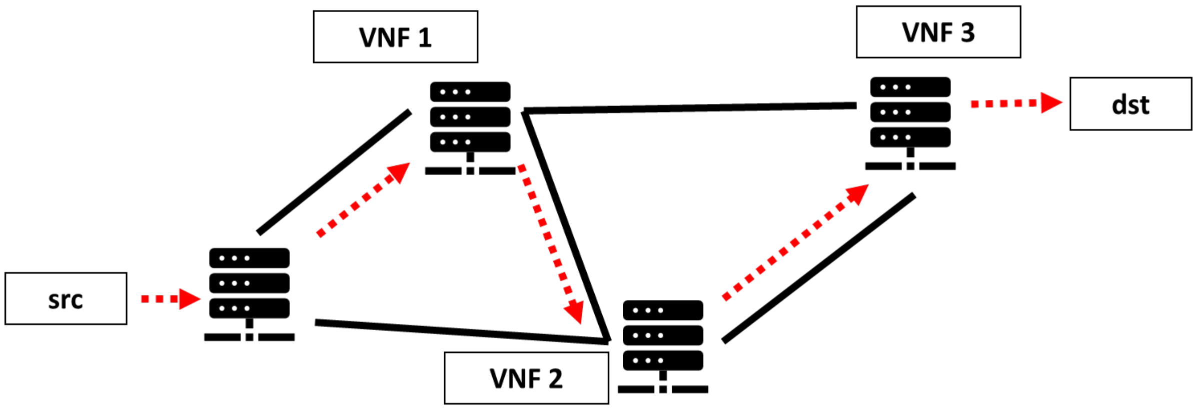

Figure 5-5 Case 2: Two VNFs are placed on two substrate nodes.

Case 3: Long time to recovery from failures

When a VNF fails, the MANO must find other location to recover this VNF, or the whole SFC cannot function. This requires restarting the VNF, reconfigure the VNF, and reroute the traffic through the VNF. If more than 2 VNFs are deployed on a same substrate network node, once this substrate network node fails, multiple VNFs in this node must be recovered, which will take a long time comparing to recovering only one VNF. Thus, distributed VNFs onto different substrate network nodes would reduce the time to recovery.

5.3.2. Preliminary assumption

In this chapter, we assume that the CPU and bandwidth resources are always sufficient for the deployment of SFC. We consider the edges of substrate network have some delay, and we try to find an optimal placement strategy that satisfies both the distributed deployment requirement and determines a path that has minimum end-to-end delay for SFC.

We consider that it is a distributed deployment requirement violation or anti-affinity rule violation if more than one VNFs of an SFC are placed on a same substrate network. We consider that some instances of VNFs have been already installed on the substrate network differing from the previous chapters, i.e., the VNFs of an SFC can be only deployed on a certain limited number of substrate network nodes. We call a substrate network node is a candidate node to one type of VNF if it can host this type of VNF.

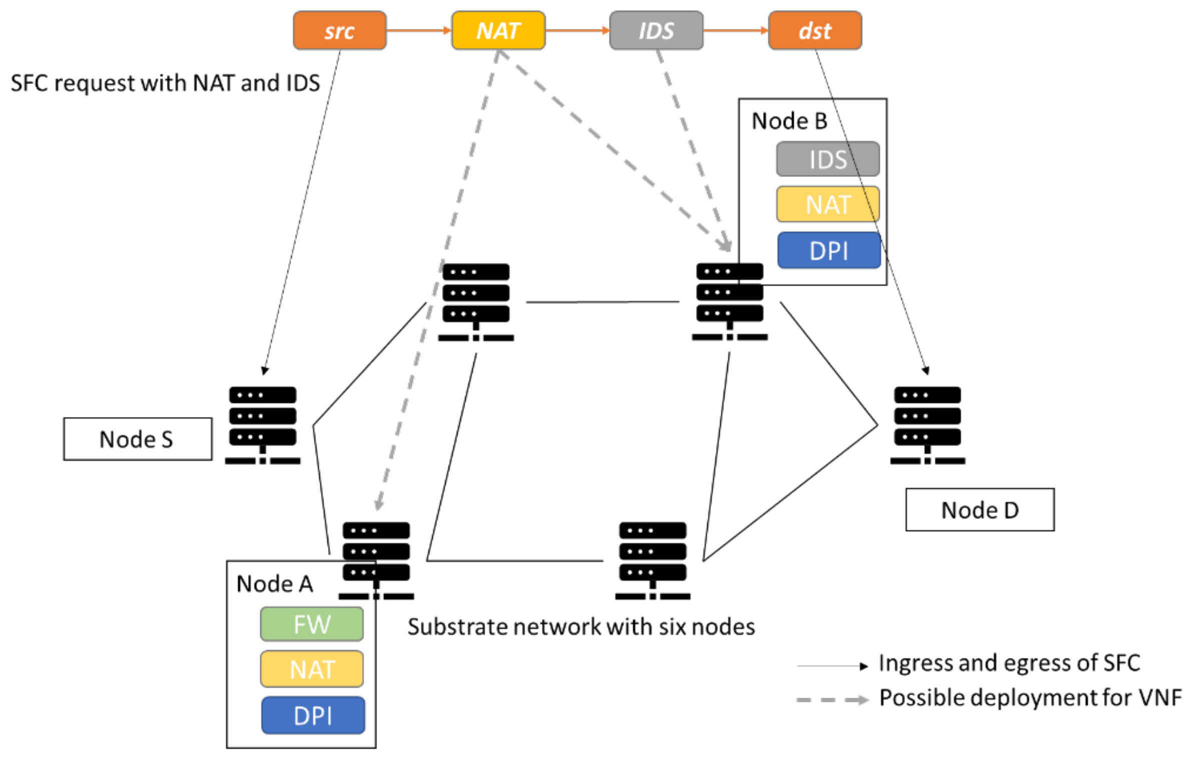
src
NAT
IDS
dst
SFC request with NAT and IDS
Node B
IDS
NAT
DPI
Node S
Node D
Node A
FW
NAT
DPI
Substrate network with six nodes
Ingress and egress of SFC
Possible deployment for VNF

Figure 5-6 An example of preinstalled VNFs on substrate networks.

(FW, NAT and DPI are preinstalled in substrate node A; IDS, NAT, and DPI are preinstalled on substrate node B. NAT in SFC request can be only placed on Node A and B, and IDS can be placed only on Node B.)

Figure 5-6 shows an example of this assumption. The substrate network consists six nodes. The bottom-left node is node A, on which the instances of FW, NAT, and IDS have been preinstalled. The top-right node is node B, on which the instances of IDS, NAT, and DPI have been preinstalled. This indicates that substrate node A is a candidate node for hosting VNF FW, NAT and DPI, and substrate node B is a candidate node for hosting IDS, NAT and DPI. Suppose an SFC request with two VNFs, NAT and IDS, then this SFC can be deployed in a path of S→A→B→D, or S→B→B→D. where S and D are *src* and *dst* of SFC. However, notice that the later placement is violated the distributed deployment requirement, as both NAT and IDS in the SFC are placed on the same substrate node B.

5.3.3. Substrate network and SFC request model notations

Notations below represent the substrate, SFC, VNF, etc. These notations are used in the following sections.

$G = (V, E)$: an undirected graph represents the substrate network.

V: the set of substrate network nodes. Specially, $|V|$ represents the number of nodes in the substrate network.

E: the set of substrate network edges.

F: set of virtual network functions (VNFs) $f \in F$.

d_{ij}: edge latency between node i and node j, where $(i, j) \in E$.

S: service function chain that contains a series of ordered VNFs $\{f_1, f_2, \dots, f_i, \dots, f_t\}$, where f_i is the ith VNF in S and t is the total number of VNFs in S.

R: service function chain request which contains a service function chain S and source node v_s and destination v_d node (i.e., $R = \{v_s, v_d, S\}$).

N_f: set of candidate nodes for hosting VNF f. Specially, $|N_f|$ represents the number of substrate network nodes for hosting VNF f.

5.4. Layered graph algorithm

In this section, we introduce an SFC placement scheme based on layered graph approach, and then discuss the insufficiency of layered graph algorithm for distributed placement of VNFs.

5.4.1. Layered graph algorithm description

The layered graph algorithm provides a simple but powerful approach to find the shortest path between src and dst of an SFC, while keeping the order of VNFs of an SFC. The layered graph algorithm receives a substrate network topology graph G, and an SFC request S consisting k VNFs. The procedure of layered graph algorithm is shown in below.

Step 1. Create k copies of substrate network graph, where k is the number of VNFs in the SFC request. Each copy is exactly same as G's topology. We denote G^0 as the original substrate network graph and G^i as the ith copy network graph.

Step 2. Connect substrate nodes in each neighboring network graphs G^{i-1} and G^i vertically if they can host ith VNF of the SFC.

Step 3. The original substrate graph G^0, t copies of substrate network graphs $G^1 \sim G^k$, and edges between these layered networks compose a new graph, denoted as G^{whole}.

Step 4. Find a shortest path between the ingress node in G^0 and the egress node in G^{whole}. Each transition node on the shortest path connecting $i-1$ and i layers is the selected substrate node to host the ith VNF of the SFC.

Figure 5-7 illustrates the layered graph approach for deployment of VNFs. In this example, G^0 represents the original substrate network topology. An SFC contains two types of VNFs, VNF1, and VNF2. The src and dst of the SFC are substrate network nodes S0 and D0, respectively.

B0 and F0 are candidate substrate nodes for hosting VNF1, i.e., the type of VNF1 instance has been pre-installed on B0, F0. B0, C0 and E0 are candidate substrate nodes for hosting VNF2, i.e., the type of VNF2 instance has been pre-installed on B0, C0 and E0.

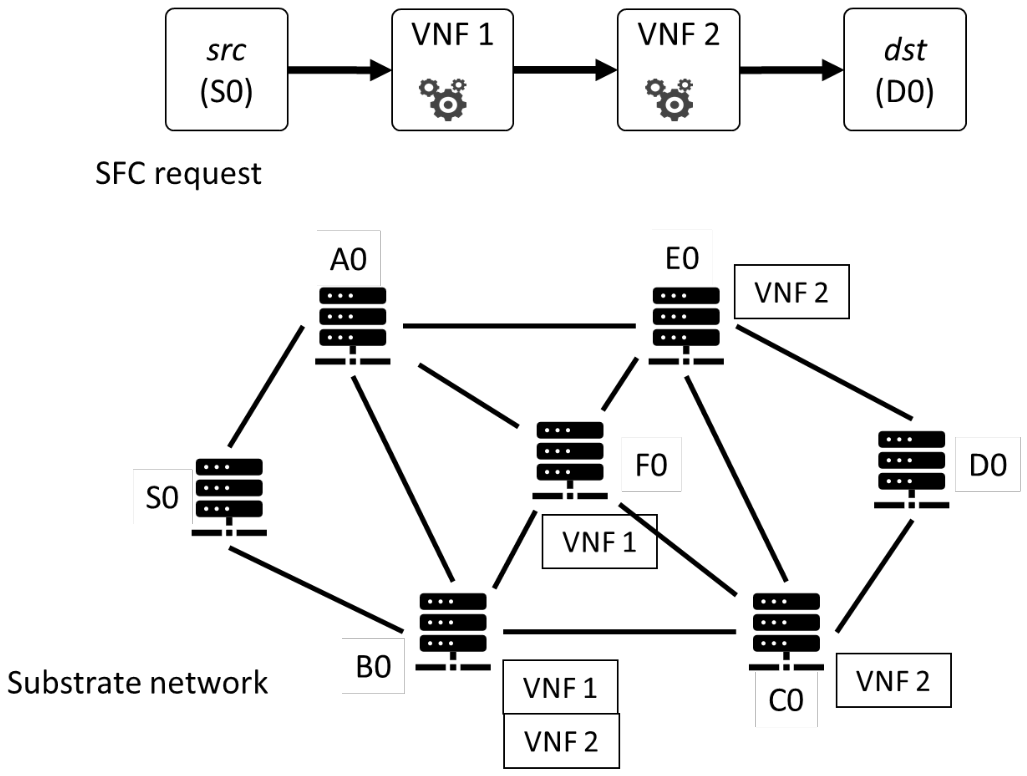

Figure 5-7 Layered graph approach: An example of SFC request and substrate network with preinstalled VNF instances.

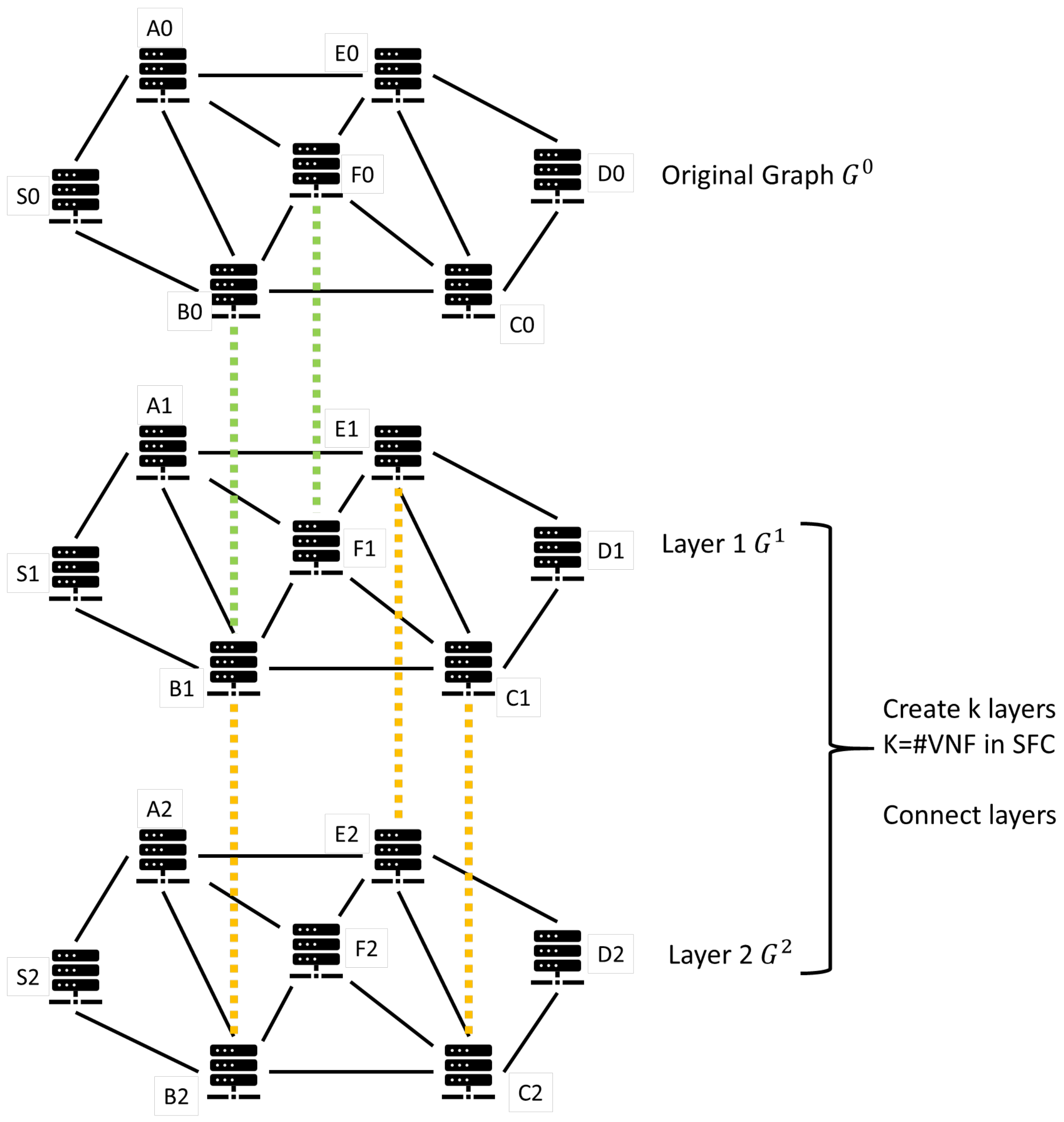

Figure 5-8 Layered graph approach step 1 & 2: create and connect k layers of substrate network.

The layered graph approach firstly constructs two new graphs G^1 and G^2, which are exactly same as G^0, and connects these layers as shown in Figure 5-8. It connects B0 to B1, and F0 to F1 as shown by the green dash lines, because B0 and F0 can host VNF 1, and it connects B1 to B2, C1 to C2, and E1 to E2 as shown by yellow dash lines in the figure, because B0, C0, and E0 can host VNF 2.

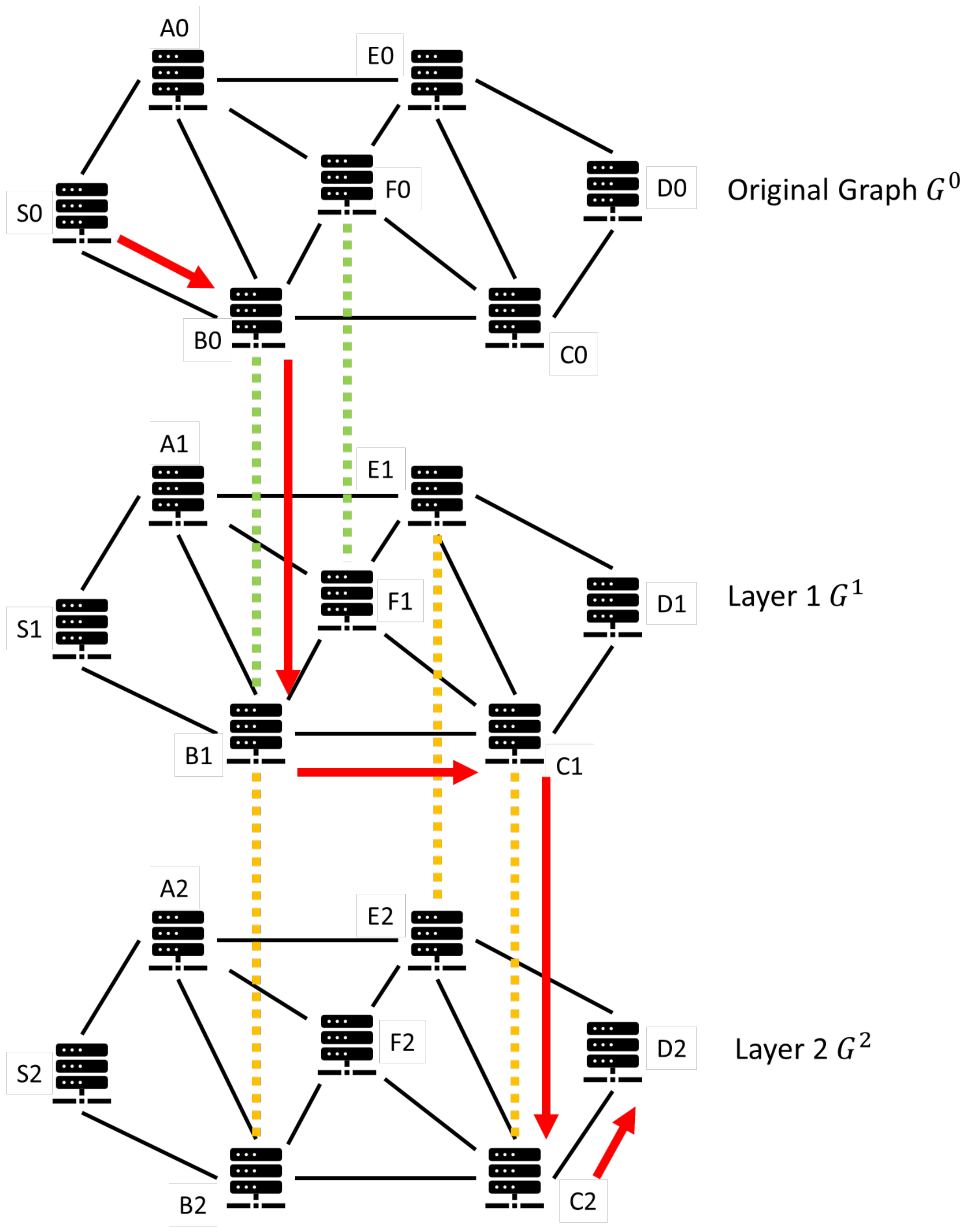

Figure 5-9 Layered graph approach step 3 & 4: compose whole graph and find shortest path.

After created and connected layers, layered graph approach tries to find a shortest path between S0 and D2, by taking edge delay as weight, as shown by red arrow lines in Figure 5-9. Since B0 and C1 are transition nodes along the shortest path from G^0 to G^1 and G^1 to G^2, B and C are selected to host VNF1 and VNF2 of SFC, respectively. Therefore, the deployment path in the original graph is (S0 → B0 → C0 → D0).

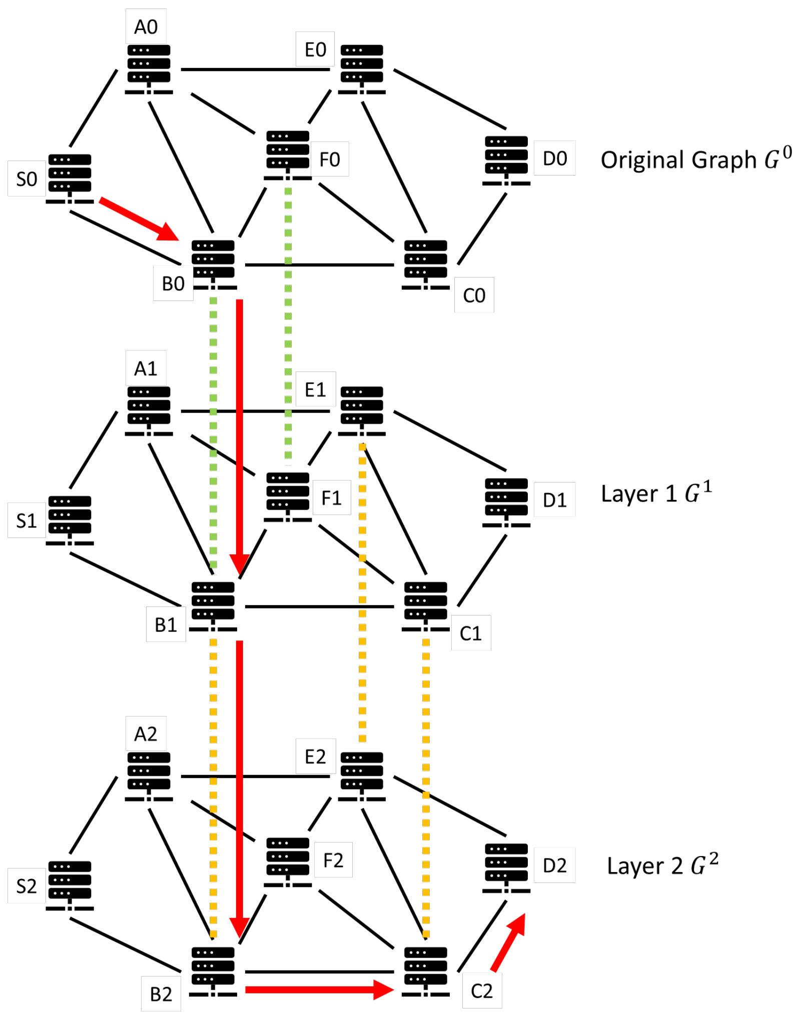

Figure 5-10 Insufficiency of layered graph algorithm.

5.4.2. Insufficiency of layered graph algorithm for distributed deployment of VNFs

The insufficiency of layered graph algorithm is that this algorithm cannot guarantee the distributed deployment requirement of an SFC request. It is highly possible that the shortest path traverse through a substrate network node several times to avoid a longer path in the last step of the algorithm.

Figure 5-10 shows an example. The SFC deployment path could be ($S0 \rightarrow B0 \rightarrow B0 \rightarrow D0$), where VNF1 and VNF2 are collocated in the same substrate network node B0, and this violates the distributed deployment requirement.

As described in the previous sections, this collocated deployment of VNFs causes overload resource utilization, risk to forming loops and long recovery time. Thus, distributed SFC placement is necessary to ensure the performance of network services.

5.5. Distributed SFC placement algorithm

This section describes our proposed distributed SFC placement algorithm. This SFC placement algorithm is based on the layered graph approach. This algorithm is aiming to place VNFs of an SFC onto different substrate network nodes, so that prevents SFCs from potential performance degradation.

To complete this goal, the placement algorithm avoids to connecting two or more layers via a same substrate node. If one substrate node connects more than one layers, the algorithm should remain only one layers for the node and break all other layers. As shown in Figure 5-7, both VNF 1 and VNF 2 can be placed on substrate B, which will lead a high possibility of collocation of VNF1 and VNF2 on node B. To avoid this collocation, the algorithm should remain only one connection and break others among the connection of B0 and B1, and connection of B1 and B2. Thus, for each substrate node, the algorithm needs to examine all layers to find all connections of this node, and then make a decision of which connection should be remained for this node. This ensures that one substrate node could only hosts one VNF of an SFC.

Since one VNF can be placed on multiple candidate substrate nodes, the algorithm must select an appropriate substrate node among candidate nodes to host this VNF. Namely, if one substrate node connects multiple layers, the algorithm must select one connection of two neighbor layers and break all other connections. As each time a connection of two layers is eliminated, the chance is reduced to calculate a shorter path for an SFC. In the worst case, all connections of a node would be eliminated without any design. Then, this node cannot host any VNF.

In our algorithm, to decide which VNF a node should host, we first get the value of $|N_f|$, the number of candidate substrate nodes that could host a VNF f, as defined in Section 5.3. The algorithm finds the VNF whose value of $|N_f|$ is smallest among all VNFs of the SFC by comparing values of $|N_f|$ of all f in SFCs. Once the VNF with the smallest $|N_f|$ is found, the substrate node becomes a candidate node only for this VNF of this SFC.

For example, considering an SFC with two VNFs, VNF 1 and VNF 2. Substrate nodes {B, F} are candidate nodes to host VNF 1, and substrate nodes {B, C, E} are candidate nodes to host VNF 2. To avoid VNF 1 and VNF 2 being placed on substrate node B, our algorithm decides node B to be the candidate node to host VNF 1, and remove B from VNF 2's candidate node set. This is because VNF 1 has a smaller number of candidate substrate nodes than VNF 2.

Algorithm 3 shows the node selection algorithm described in above contents. The inputs of Algorithm 3 are N and V, where $N = \{N_f | \forall f \in S\}$ is a set of N_f for all VNFs, and V is the node set in substrate network. The output of Algorithm 3 is a new set of N_f which are the candidate nodes selected to host VNF f.

From lines 3 to 11, Algorithm 3 iterates to find suitable substrate network nodes. For each substrate node v, the algorithm first initializes a set F_v. Inside F_v are VNFs that node v can host. The algorithm fills F_v with VNFs that node v can host by examining all N_f as shown from lines 5 to 9. At line 10, the algorithm fills L_F with F_v. L_F is a set containing sets of VNFs which can be placed on a common candidate substrate node.

From lines 12 to 30, the algorithm remains only one VNF and removes all other VNFs of an SFC. The remained VNF is the one that could be placed on this substrate node v. At lines 13 and 14, the algorithm creates two variables min_f and min_N with initial value $None$ and a large enough integer (say Max_Int). min_f tracks the VNF of F_v that has the least number of candidate substrate nodes, and min_N records the number of candidate substrate nodes to host min_f.

At line 15, the algorithm examines whether substrate node v is able to host more than one VNFs. If substrate node v cannot host more than one VNFs, the algorithm does nothing and continues to iterate the next F_v, because in this condition, at most one VNF is able to be placed on this node, which satisfies distributed deployment requirement. From line 16, it indicates that this substrate node v is a candidate node for multiple VNFs. The algorithm then iterates and check each VNF f whether it is the VNF with minimum $|N_f|$ among all VNFs in F_v.

If VNF f is the VNF with minimum $|N_f|$, the algorithm set min_N as this $|N_f|$ and removes substrate node v from N_{min_f} at line 20. Then the algorithm updates min_f by f at line 22. If VNF f is not the VNF with minimum $|N_f|$ at line 17, the algorithm removes the substrate node v from $|N_f|$ at line 25, which indicates that substrate node v is no longer the candidate node for hosting VNF f. Here the variable min_f in lines 19 and 20 is the VNF with minimum value of $|N_f|$ found previously. The reason to remove v from $|N_{min_f}|$ at line 20 is that VNF f has not been the VNF that has the minimum $|N_f|$, as the algorithm has found a new VNF. Therefore, in line 20, the substrate node v should not host the VNF because it is not the VNF that has the minimum $|N_f|$ among F_v.

Next, the algorithm creates a new layered graph by utilizing new set of candidate nodes for hosting VNFs and calculates the shortest path in terms of delay, as shown in Algorithm 4.

Algorithm 4 receives R, G, and N as inputs. R is an SFC request. G is graph of original substrate network. N is a new set of nodes for VNFs that $N = \{N_f | \forall f \in S\}$ is generated by Algorithm 3.

Algorithm 4 creates $|R.S| + 1$ copies of G. G is graph of original substrate network. $R.S$ is the SFC of request R. $|R.S|$ is the length of the SFC $R.S$. At line 2, G_{new} is a variable storing the new graph composed of G and copies of G. From lines 3 to 7, the algorithm connects layers in G_{new} according to N obtained from Algorithm 3. At lines 8 and 9, Algorithm 4 assigns $R.v_s$ in G^0 to v_s^0 as the source node of the SFC, and assigns $R.v_d$ in $G^{|R.S|}$ to $v_d^{|R.S|}$ as the destination node of the SFC. Finally, the algorithm finds the shortest path between v_s^0 and $v_d^{|R.S|}$ and returns the path at lines 10 and 11.

Note that in some situations, our algorithm (and also all other algorithms) cannot find a solution to distributed place VNFs on different substrate nodes, because there is no such solution. For example, in a scenario of three VNFs $f1$, $f2$, and $f3$, and $N_{f1} = \{A, B\}, N_{f2} = \{A, B\}$, $N_{f3} = \{A, B\}$, which means $f1$ can be placed on either A or B, and so forth, then VNFs $f1$, $f2$, and $f3$ cannot be placed onto different substrate nodes, and node A or B will always host more than one VNF.

In this section, we solve the SFC placement with satisfaction of distributed requirement, which is merely touched by previous works. we described a distributed SFC placement algorithm based on a modified layered graph approach. Our proposed algorithm can also achieve a relatively low end-to-end delay and guarantee order of VNFs of an SFC. Note that the computation complexity for VNFs stays at almost the same level comparing to original layered graph either in distributed or centralized placement of VNFs.

Algorithm 3 Node selection

```
1:  **Procedure** NodeSelection($N$, $V$)
2:      $L_F \leftarrow []$
3:      **for** each $v$ in $V$ **do**
4:          $F_v \leftarrow []$
5:          **for** each $N_f$ in $N$ **do**
6:              **if** $v$ in $N_f$ **then**
7:                  insert $f$ into $F_v$
8:              **end if**
9:          **end for**
10:         insert $F_v$ to $L_F$
11:     **end for**
12:     **for** each $F_v$ in $L_F$ **do**
13:         $min_f \leftarrow None$
14:         $min_N \leftarrow Max\_Int$
15:         **if** $|F_v| > 1$ **then**
16:             **for** each $f$ in $F_v$ **do**
17:                 **if** $min_N > |N_f|$ **then**
18:                     $min_N \leftarrow |N_f|$
19:                     **if** $min_f \neq None$ **and** $|N_{min_f}| > 1$ **then**
20:                         delete $v$ from $N_{min_f}$
21:                     **end if**
22:                     $min_f \leftarrow f$
23:                 **else**
24:                     **if** $|N_f| > 1$ **then**
25:                         delete $v$ from $N_f$
26:                     **end if**
27:                 **end if**
28:             **end for**
29:         **end if**
30:     **end for**
31:     **return** $N$
32: **end procedure**
```

Algorithm 4 Layered graph based routing

```
1:  **Procedure** LinkRouting($R$, $G$, $N$)
2:      $G_{new} \leftarrow |R.S| + 1$ copies of $G$ stored in a new graph
3:      **for** $G^i$ in $G_{new}$ **do**
4:          **for** $nodes$ in $N_{f_{i+1}}$ **do**
5:              connect $nodes$ in $G^i$ and $nodes$ in $G^{i+1}$
6:          **end for**
7:      **end for**
8:      $v_s^0 \leftarrow R.v_s$ in $G^0$
9:      $v_d^{|R.S|} \leftarrow R.v_d$ in $G^{|R.S|}$
10:     $p \leftarrow$ shortest path in $G_{new}$ between $v_s^0$ and $v_d^{|R.S|}$
11:     **return** $p$
12: **end procedure**
```

5.6. Evaluation and analysis

In this section, we compare our proposed algorithm with other algorithms in literature. Specifically, the based layered graph algorithm, a bipartite matching algorithm utilized in the work [19] and a random deployment algorithm. We wrote a simulator in Python language. We evaluate the performances of these algorithms in terms of distributed deployment requirement satisfaction rate, and end-to-end delay of an SFC. The metric of ant-affinity requirement satisfaction rate indicates whether the algorithm can distributed deploy VNFs of an SFC in such way that no more than two VNFs are collocated on a same substrate network node. And the metric of end-to-end delay of an SFC shows how the algorithm could guarantee a desired QoS of network service. The topology used for evaluation is a random graph with 100 nodes and 0.05 edge connectivity probability.

5.6.1. Algorithms for comparison

In this section, we introduce algorithms in literature for comparison with our proposed algorithm. We compare our algorithm with the non-modified layered graph algorithm, the bipartite matching algorithm, and the random algorithm.

a) **Bipartite matching algorithm[19]**

Bipartite graph: A bipartite graph (bigraph) is a graph whose vertices can be partitioned into two disjoint sets such that no two vertices from the same set are adjacent [149]. Figure 5-11 shows an example of bipartite graph. The graph can be divided into two vertices sets {a, b, c, d, e} and {1, 2, 3}.

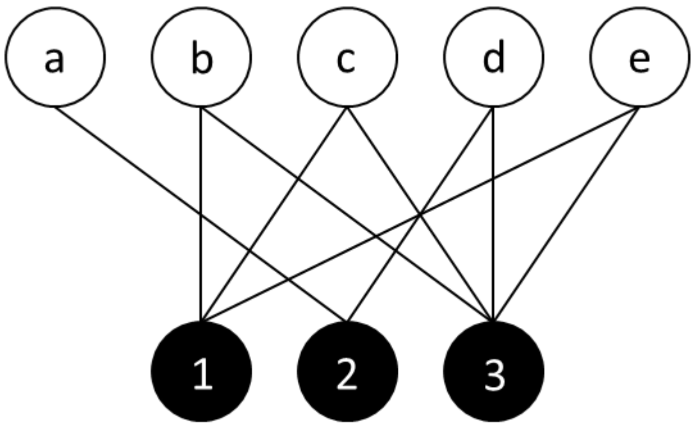

Figure 5-11 An example of bipartite graph.

Bipartite matching: a matching in a graph is to find a sub graph in which no two edges share a common vertex, and a maximum matching is a matching that finds as many edges as possible. Figure 5-12 shows an example of matching of Figure 5-11. In this figure, node 1 is matching to node c, node 2 is matching to d, and node 3 is matching to b, respectively.

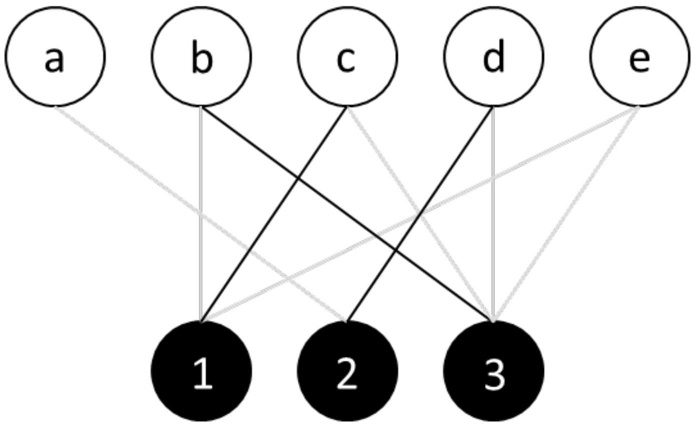

Figure 5-12 A matching corresponding to Figure 5-11.

Distributed deployment of VNFs by bipartite matching: A distributed deployment of VNFs can be transferred into a bipartite maximum matching problem. In particular, the VNFs of an SFC and the substrate network nodes together construct a bipartite graph. The VNFs of an SFC is belonged to one vertex set, and the substrate network nodes is belonged another vertex set. The result of the bipartite matching problem is the solution of the deployment of VNFs. For example, vertices 1, 2 and 3 in Figure 5-11 can represent an SFC request with three VNFs, vnf-1, vnf-2 and vnf-3. Vertices a, b, c, d and e can represent substrate network with five nodes. Nodes b, c, and e are candidate nodes for hosting vnf-1, nodes a and d are candidate nodes for hosting vnf-2, and nodes b, c, e and e are candidate nodes for hosting vnf-3, respectively. As the goal of distributed deployment of VNFs is to find a scheme that no more than two VNFs are deployed on a same substrate node, which indicates that no two edges share a common substrate node. The bipartite matching problem can be solved by well-known augmenting path method. Figure 5-12 shows a solution of the deployment, that vnf-1 is placed on substrate node c, vnf-2 is placed on substrate node d, and vnf-3 is placed on substrate node b, respectively. In the experiments, we utilize the maximum matching algorithms in NetworkX [150][151]. After found substrate nodes for each VNFs, we connect the substrate nodes from src to dst of the SFC through each substrate nodes by shortest path method.

b) Random algorithm

This algorithm randomly chooses a substrate node from the candidate substrate network node set of a type of VNF. For instance, consider an SFC composed of two VNFs, vnf-1, and vnf-2. Suppose vnf-1 can be deployed on substrate network nodes {a, b, c} and vnf-2 can be deployed on substrate network nodes {b, c, d}. Then, this algorithm randomly chooses a node from {a, b, c} for hosting vnf-1, and randomly chooses a node from {b, c, d} for hosting vnf-2.

5.6.2. Substrate network model for evaluation

In this section, we introduce the network model used in the experiments. We conducted our evaluations on two network models, JPN 48, and Random Graph network model.

a) **JPN 48 network model:** JPN 48 is a network model developed based on publicly available information of Japan nationwide transportation network [124][125]. The network model contains 48 nodes and 82 edges. The nodes are placed on major cities in Japan. The model has been utilized for network research in Japan [126][127][128]. Figure 4-2 shows the topology of JPN 48. The number in the figure represents a city of Japan, e.g., node number 10 is Sapporo, node number 20 is Aomori, and node number 131 is Tokyo. We set the delay of each edge according to the real distance between two nodes. For instance, the delay between 10 and 20 is $1.5884322213336\ ms$, which is calculated as $476.2\ km/299792458(km/s) * 1000$, where 476.2km is the distance between city Sapporo (10) and city Aomori (20), and $299792458km/s$ is the speed of light. 1000 is used to adjust the unit from second to millisecond.

b) **Random Graph network model:** To evaluate and compare our proposed algorithm on a larger, more complex but general substrate network without dependent on any specific topology, we introduce random graph model as the underlaying substrate network topology. Random graph model [109][110] is an intersection of graph theory and probability theory. There are two ways to generates a random graph, and we leverage the $G(n, p)$ model to generate a random graph topology, where n is the number of nodes in the graph and the p is the probability that whether two nodes in graph is connected. Random Graph model has been utilized for evaluation algorithms such as in

[78][111][112][127][131]. In this experiment, we generate a random graph of 100 nodes with 0.05 edge connective probability. The delay of each edge is randomly chosen from [1, 10] in milliseconds unit.

We evaluate the performance of algorithms in terms of distributed deployment requirement satisfaction and end-to-end delay. As mentioned above, distributed deployment requirement is satisfied if no more than two VNFs are deployed on a same substrate network node, or the requirement is violated, and end-to-end delay is a metric that how well the algorithm can guarantee a desired QoS of network service.

To utilize maximum matching algorithm, the bipartite graph must be a connected graph as described in [151], or it will raise an *AmbiguousSolution* exception when run the algorithm, which indicates that more than on valid solution is possible. To avoid this exception, we assume half number of the substrate network nodes can host any type of VNFs and the other half number of the substrate nodes can arbitrarily host any type of VNFs. This scenario is practice, for example, some of devices in the substrate network can host all types of the VNFs, whereas some other devices can only host a certain type of VNFs due to interoperability, regulation, or license issues. We repeat the experiment 100 times and collect the times that distributed deployment requirement is satisfied by each algorithm.

5.6.3. Performance on JPN48 network model

Figure 5-13 and Figure 5-14 shows the results of performances in terms of distributed deployment satisfaction rate and end-to-end delay of SFCs, respectively, with respect to the length of an SFC. Specifically, proposed (triangle marker), BM (cross marker), LG (rhombus marker), and RA (square marker) represent our proposed algorithm, bipartite matching algorithm, layered graph algorithm, and random algorithm, respectively.

The results show that both our proposed algorithm and the bipartite matching algorithm (BM) could perform 100% distributed deployment satisfaction rate, which indicate all VNFs in an SFC are deployed on the substrate network nodes in such a way that no more than two VNFs are collocated on a same substrate network node. The random algorithm (RA) performs a 98% distributed deployment satisfaction rate at the length of an SFC 2, and then the rate decreases as the length of an SFC increases. This is reasonable as the longer an SFC is, the higher probability that two VNFs could be randomly deployed on a same substrate network node. The non-modified layered graph (LG) algorithm first achieves 26% of distributed deployment satisfaction rate at the length of an SFC 2, and then decreases to values around 0%, at the length of an SFC 2, 4, and 5. This is because layered graph only considers to find a deployment solution with a minimum path for an SFC, without considering the collocation issues.

Meanwhile, Figure 5-14 shows that our proposed algorithm can outperform the bipartite matching algorithm in terms of end-to-end delay which is a critical network metrics of QoS. Specifically, our algorithm is superior to the bipartite matching algorithm with 46%, 38%, 27%, and 22% reduction in terms of end-to-end delay of SFCs at various length of an SFC 2, 3, 4, and 5, respectively. With comparing the random algorithm, our algorithm outperforms with 37%, 36, 38%, and 32% reduction in terms of end-to-end delay of SFCs at various length of an SFC 2, 3, 4, and 5, respectively, while our algorithms can simultaneously achieve a 100% distributed deployment satisfaction rate at each length of an SFC. By comparing with the non-modified layered graph algorithm, our proposed algorithm improved at most 100% performance in terms of distributed deployment satisfaction.

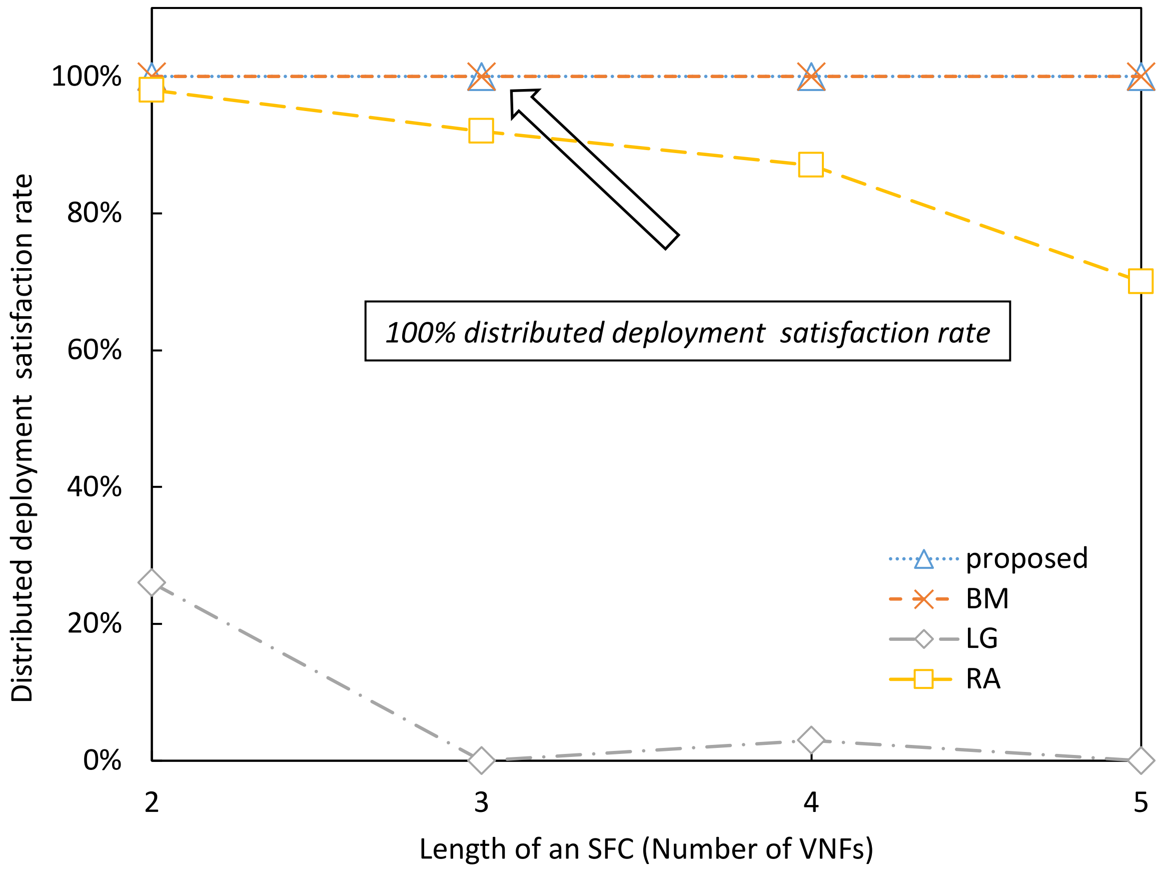

Figure 5-13 Distributed deployment satisfaction rate on JPN48 network model.

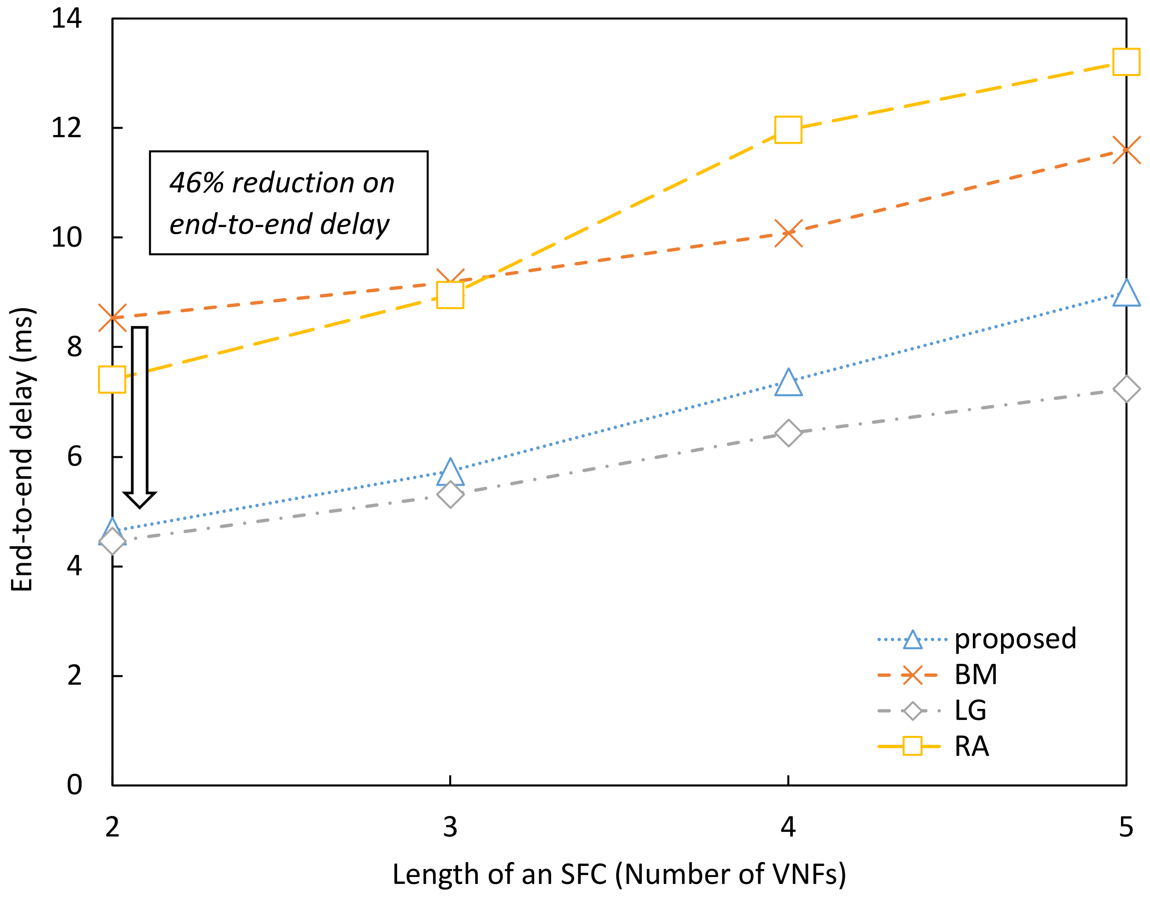

Figure 5-14 End-to-end delay of SFCs on JPN48 network model.

5.6.4. Performance on the random graph network model

Figure 5-15 and Figure 5-16 show the results of performances in terms of distributed deployment satisfaction rate and end-to-end delay of SFCs, respectively, with respect to the length of an SFC. Specifically, proposed (triangle marker), BM (cross marker), LG (rhombus marker), and RA (square marker) represent our proposed algorithm, bipartite matching algorithm, layered graph algorithm, and random algorithm, respectively.

It can be observed that both our proposed algorithm and bipartite matching algorithm (BM) can achieve a 100% distributed deployment satisfaction rate. The random algorithm (RA) could perform a high satisfaction rate, but the rate decreases as the number of VNFs in an SFC request increases. This is reasonable since the more VNFs in an SFC requests, the higher probability that one node can be randomly chosen to host more than one VNF of an SFC. The non-modified layered graph (LG) algorithm performs almost 0% satisfaction rate since this algorithm always finds a shortest path without considering distributed deployment of VNFs.

Meanwhile, as shown in the Figure 5-16, our proposed algorithm can achieve a better performance in terms of end-to-end delay of SFC. Specifically, our proposed algorithm reduced the end-to-end delay around 57%, 58%, 60%, and 59% comparing with the bipartite matching algorithm at the length of an SFC 2, 3, 4, and 5, respectively, while keeping a same 100% distributed deployment satisfaction rate, simultaneously. By comparing with random algorithm, our proposed algorithm achieved around 56%, 59%, 60%, and 58% reduction on the end-to-end delay with various of length of SFC, while our algorithm outperforms at most 8% higher than random algorithm in terms of distributed deployment satisfaction rate. By comparing with the non-modified layered graph algorithm, our proposed algorithm can achieve 100% performance in terms of distributed deployment satisfaction.

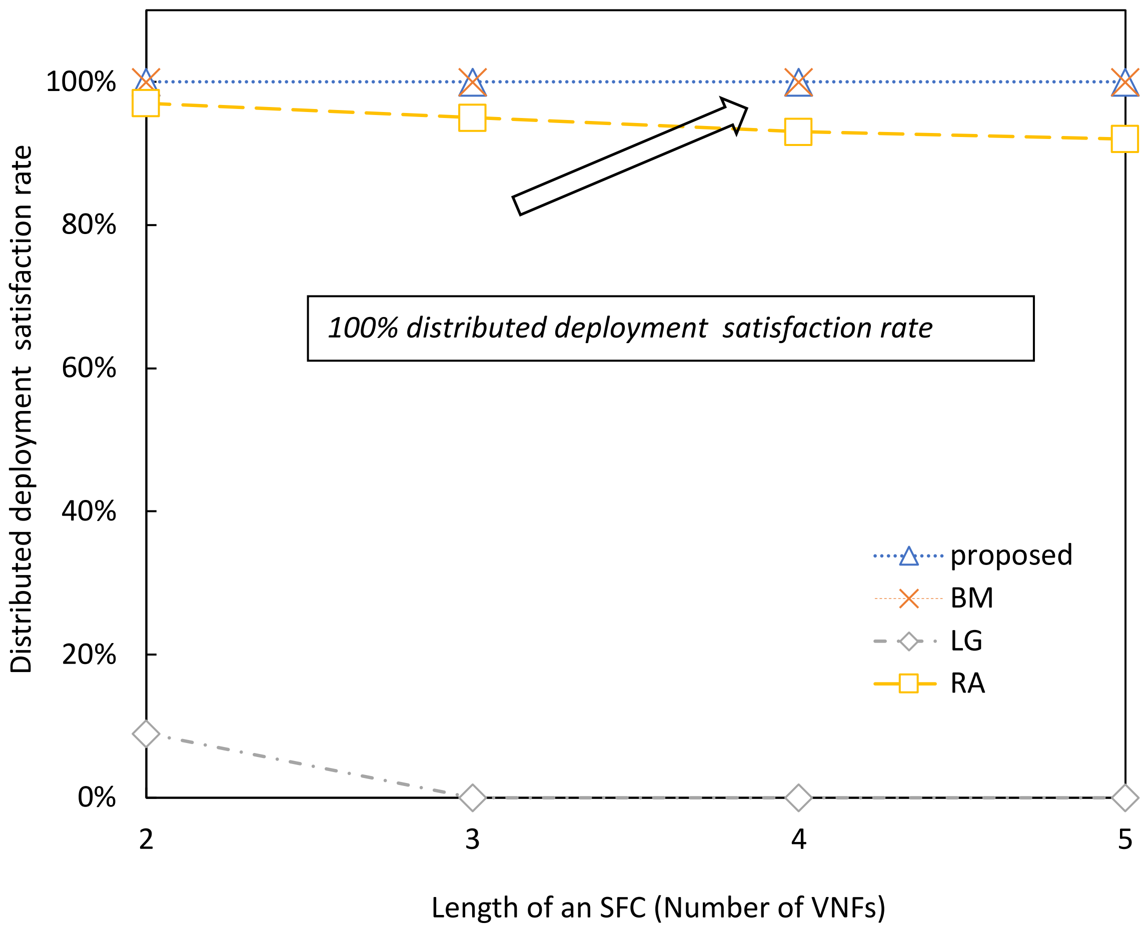

Figure 5-15 Distributed deployment satisfaction rate on random graph network model.

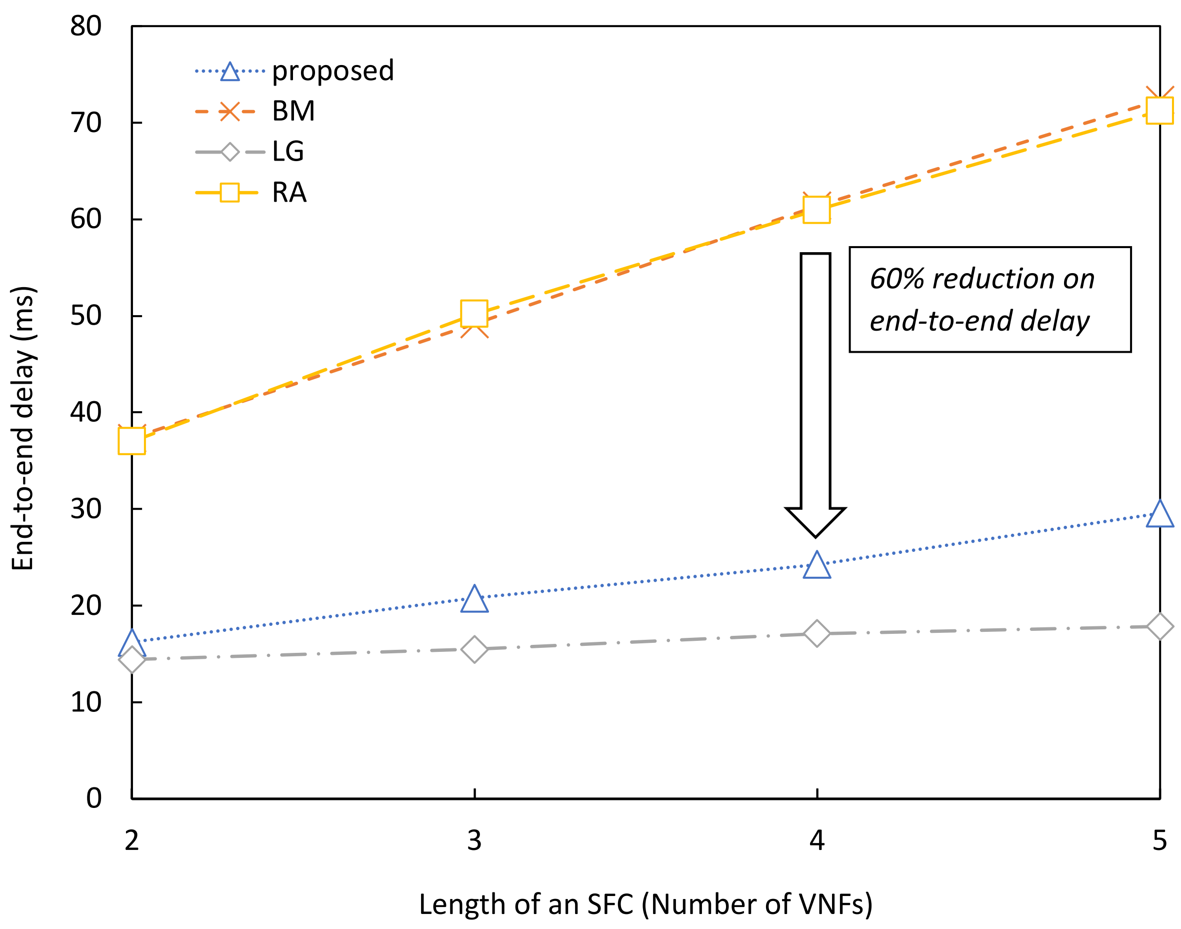

Figure 5-16 End-to-end delay of SFCs on random graph network model.

5.7. Extensive evaluation and analysis

5.7.1. Simulation setup

We evaluate our proposed algorithm in terms of distributed deployment requirement satisfaction, end-to-end delay, and computation cost. The simulator used in the evaluation is written in Python language. The simulator was executed on an Intel-4600M 2.90GHz CPU Ubuntu 16.04 server with 8 GB memory. We conduct experiments on different network topologies generated on the basis of *Erdős-Rényi* graph[109][110]. The types of VNFs of an SFC were different to each other.

5.7.2. Performance of distributed deployment requirement satisfaction

We first evaluate the performance of our proposed algorithm in terms of distributed deployment satisfaction. In this context, a deployment satisfied distributed deployment requirement if there is no more than two VNFs collocated on a same substrate network node.

We evaluated our proposed algorithm on various substrate networks whose topologies are generated by random graph, also known as *Erdős-Rényi* graphs. Similar to prior works [78][111][112], the numbers of substrate network nodes were varied from 50 to 100, and the edge connectivity probabilities between two nodes were 1.0, 0.5, and 0.1. The edge connectivity probability value of 1.0 indicates the network is a fully connected mesh network. The edge connectivity probability value of 0.5 indicated the substrate network is a densely connected network, such as a partial mesh network. The edge connectivity probability of 0.1 indicated the substrate network is sparsely connected with only a few links, such as Germany50 [112][113], which is composed of 50 nodes and 88 links.

The number of VNFs of each SFC was varied from 5 to 20. $|N_f|$ of each VNF was varied from 1 to $|V| - 2$. Nodes in N_f were selected randomly from the substrate network except source and destination nodes of the SFC. $|N_f|$ was identical to each VNF of an SFC, namely, $|N_f|$ was a constant value for all f. As $|V|$ is the number of substrate network nodes, $|V| - 2$ indicated that the source and destination nodes of an SFC could not host VNFs.

The number of nodes in the substrate network was varied from 50 to 100. In each setting, the algorithm placed SFC requests by increasing $|N_f|$ from 1 to $|V| - 2$. We collected statistics in terms of the number of times that distributed placement requirement was not satisfied during these $|V| - 2$ times experiments. The distributed placement requirement was not satisfied if more than one VNF were placed on the same substrate network node.

Table 5-1 Number of times distributed deployment requirement was unsatisfied. (edge connectivity probability=1.0)

Number of VNFs	**50 nodes**	**60 nodes**	**70 nodes**	**80 nodes**	**90 nodes**	**100 nodes**
5 VNFs	0	0	0	0	0	0
10 VNFs	0	1	0	0	0	0
15 VNFs	1	1	0	1	1	0
20 VNFs	2	1	0	1	1	1

Table 5-2 Number of times distributed deployment requirement was unsatisfied. (edge connectivity probability=0.5)

Number of VNFs	**50 nodes**	**60 nodes**	**70 nodes**	**80 nodes**	**90 nodes**	**100 nodes**
5 VNFs	0	0	0	0	0	0
10 VNFs	0	0	0	1	0	1
15 VNFs	1	0	1	1	1	0
20 VNFs	2	2	1	1	1	1

Table 5-3 Number of times distributed deployment requirement was unsatisfied. (edge connectivity probability=0.1)

Number of VNFs	**50 nodes**	**60 nodes**	**70 nodes**	**80 nodes**	**90 nodes**	**100 nodes**
5 VNFs	0	0	1	0	0	0
10 VNFs	1	1	0	1	0	1
15 VNFs	1	0	1	0	0	1
20 VNFs	2	2	1	1	1	1

Table 5-1, Table 5-2, and Table 5-3 summarize results of evaluation in terms of the number of times that distributed deployment requirement was unsatisfied. The number in the table indicates how many times distributed deployment requirement was unsatisfied. For example, in Table 1, the value “2” on the cross cell of 20 VNFs and 60 nodes means it was twice that the distributed deployment requirement was unsatisfied in 1000 times experiments. Therefore, a zero value in the table indicates there was zero time that the distributed deployment requirement was unsatisfied, that is, all VNFs were deployed on different substrate network nodes in all experiments.

From above tables, most of the values were 0, indicates that our algorithm satisfied well the distributed deployment requirement of SFCs. Furthermore, by carefully check, we found that the non-zero values in the above tables only happened at value of $|N_f|$ 1 or 2. This is because in the parameter setting, $|N_f|$ was same to all VNFs, and the candidate nodes in N_f were chosen randomly from the substrate network nodes. For example, for any VNF in the SFC, suppose $|N_f|$ is 1, which indicates each VNF has only one candidate node. As the candidate node is selected randomly from substrate network, there is possibility that more than one VNFs' candidate nodes are same node in substrate network. In this case, it is inevitable to collocate these VNFs on a same substrate node.

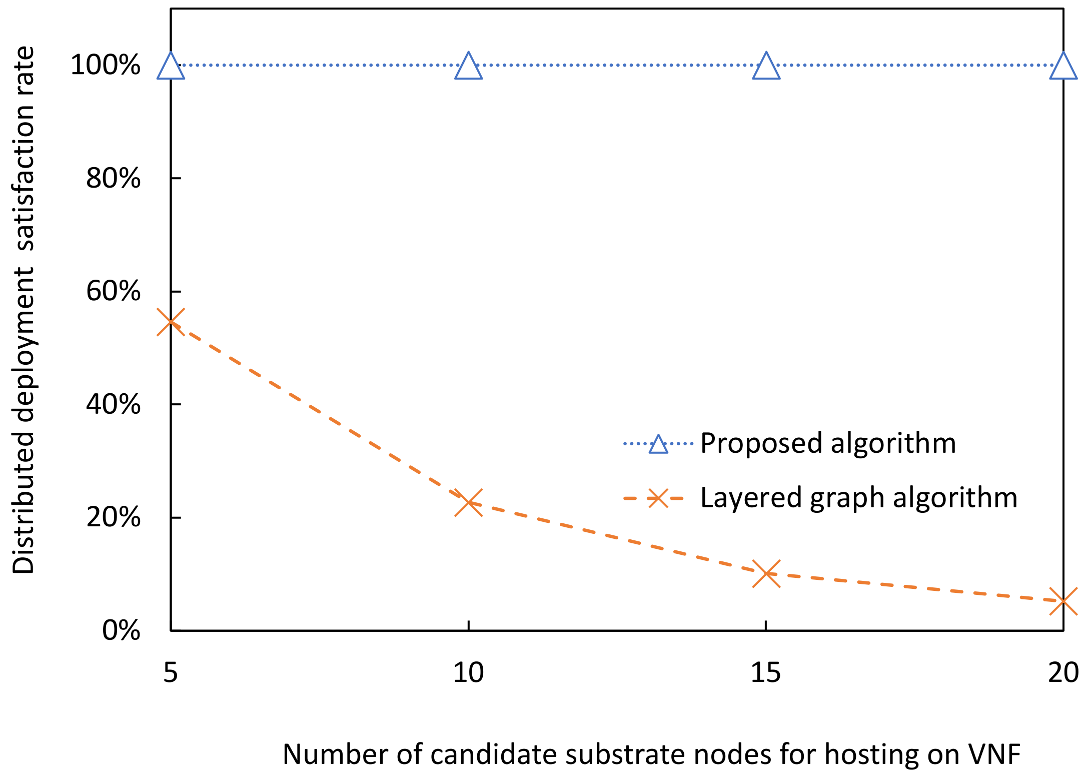

Figure 5-17 Performance of distributed deployment satisfaction.
(100 nodes, edge connectivity probability 0.5)

To evaluate the performance of the distributed deployment requirement was satisfied in different $|N_f|$, we generated a substrate network with 100 nodes at edge connectivity probability 0.5. We fixed the length of an SFC at 5, that an SFC is composed of 5 VNFs. We conducted the evaluation experiment 1000 times. In each time, the number of candidate nodes for hosting one VNF $|N_f|$ was set as 5, 10, 15, and 20. We compared our proposed algorithm with a base algorithm, specified in related work [18] and [27]. The base algorithm is to place VNFs by utilizing layered graph approach but not considering distributed placement of VNFs.

Figure 5-17 show the results of the comparison in terms of distributed deployment satisfaction rate. Our proposed algorithm can satisfy the distributed deployment requirement 100% (1000 out of 1000 times) at each value of $|N_f|$, whereas the base algorithm can only satisfy the distributed deployment requirement 54.6% (454 out of 1000 times) and 5.2% (52 out of 1000 times) at value of $|N_f|$ 5 and 20, respectively. The satisfaction rate decreases as the value of $|N_f|$ increases in the based layered graph approach, because at a higher value of $|N_f|$, more VNFs could be placed on a substrate node than that at a lower value of $|N_f|$.

5.7.3. Performance of end-to-end delay of SFC

To evaluate end-to-end delay performance, we set edge latency to a value selected uniformly between 1 to 10 milliseconds. We suppose substrate network node computation capacity and substrate network edge bandwidth capacity are sufficient to host SFC. The time to process a VNF on a substrate node are ignored in evaluations. We conducted the evaluations on substrate networks whose topologies are random graphs with 50 and 100 nodes at edge connectivity probability 1.0, 0.5, and 0.1.

Figure 5-18 and Figure 5-19 illustrate SFCs' end-to-end delay on substrate networks with 100 nodes at edge connectivity probability 1.0 and 0.1. The x-axis is the number of candidate substrate nodes for hosting a VNF. The y-axis is the results of SFCs' end-to-end delay. Curves in each graph represent the end-to-end delay of SFCs at numbers of VNFs 5, 10, 15, and 20.

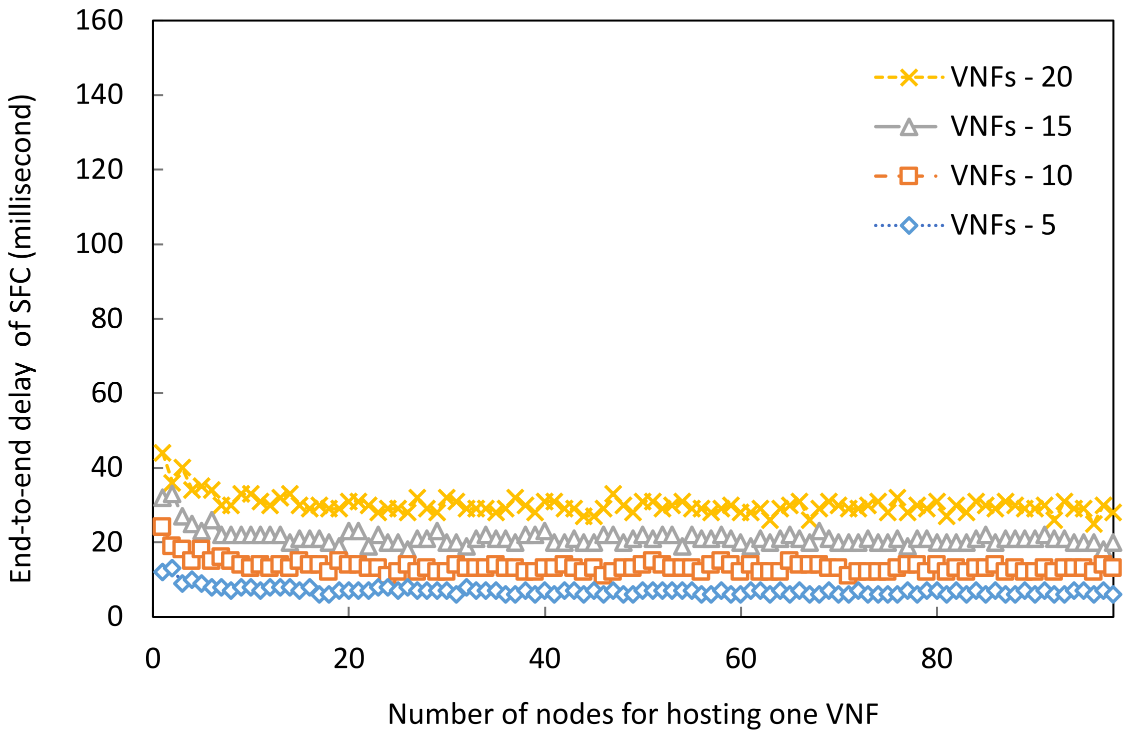

Figure 5-18 Performance of end-to-end delay of SFC.

(100 nodes, edge connectivity probability 1.0)

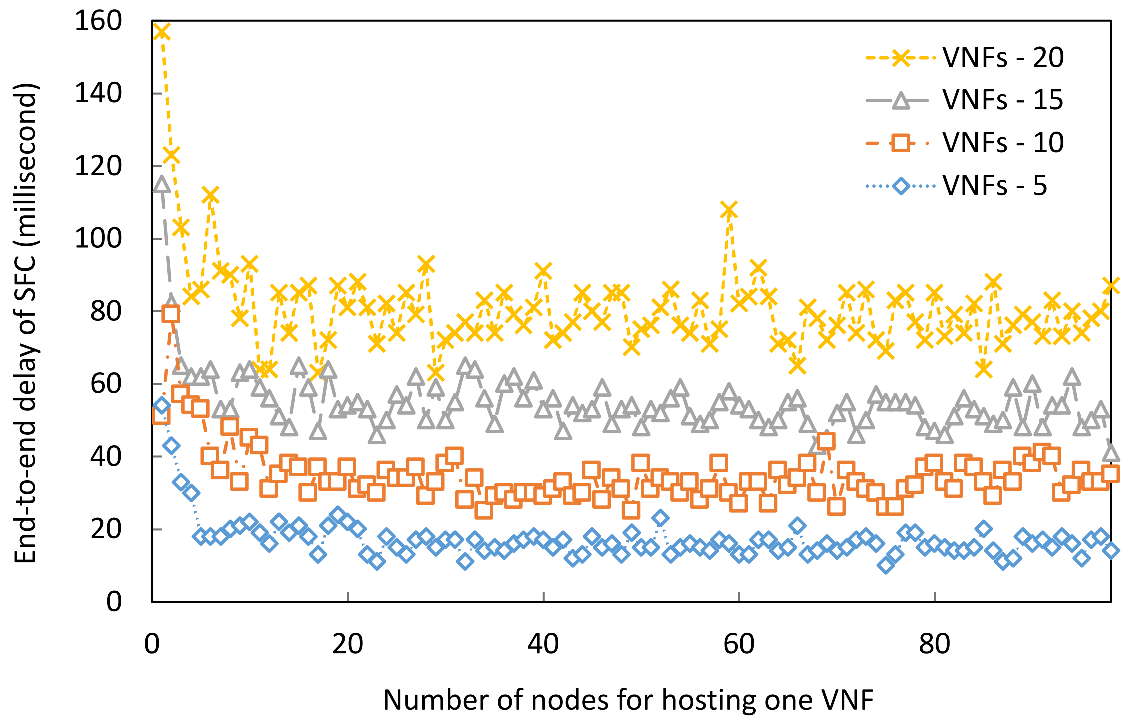

Figure 5-19 Performance of end-to-end delay of SFC.

(100 nodes, edge connectivity probability 0.1).

As the number of VNFs of an SFC increases, the end-to-end delay of the SFC increased proportionately in both figures. This is because distributed placement of VNFs led a longer substrate path for an SFC.

Figure 5-18 and Figure 5-19 also show that end-to-end delay of SFCs decreased as the number of substrate nodes and edge connectivity probability increase. The reason for this is that more nodes and higher edge connectivity probability increases the degree of free to find a shorter latency substrate path.

At each specific length of SFC, we evaluated our algorithm by varying $|N_f|$. Figure 5-18 and Figure 5-19 show that when $|N_f|$ was at low value, as $|N_f|$ increased, end-to-end delay of SFCs decreased. Finally, end-to-end delay of SFCs did not decrease with $|N_f|$ increased, but remain stable. This is because at a low value of $|N_f|$ the substrate nodes for hosting SFC are few and scattered in the substrate network, which results in a larger end-to-end delay.

Figure 5-20 shows the average end-to-end delay of SFCs in substrate network. We conducted the evaluation on a substrate network with 100 substrate nodes at 0.5 edge connectivity probability. The length of an SFC request was five. $|N_f|$ was set to values of 5, 10, 15, and 20. At each $|N_f|$ value, the experiment was conducted 100 times. The results show that the average of end-to-end delay of SFC was 11.287ms by using our algorithm against 10.292ms by using the base algorithm. As number of candidate substrate nodes for one VNF got higher, our algorithm results in a bit longer latency in comparison of base algorithm. This is reasonable since our algorithm distributes VNFs on different substrate network nodes, which results in a longer end-to-end delay path.

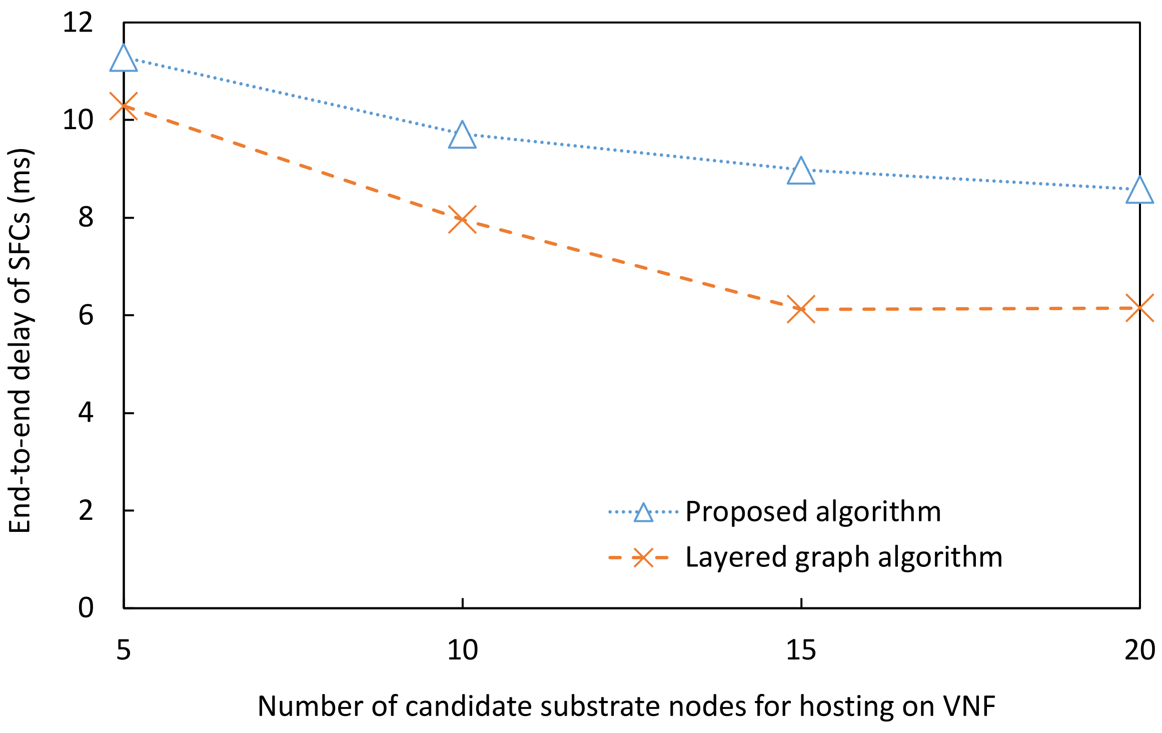

Figure 5-20 Performance of average end-to-end delay of SFC.
(100 nodes, edge connectivity probability 0.5)

5.7.4. Performance of computation time

To evaluate the performance of computation time on our algorithm, we generate substrate network topologies with the number of nodes 50 and 100, at a fixed edge connectivity probability 0.5. A VNF have 5 candidate substrate nodes ($|N_f|$ is fixed at 5). The length of an SFC is 5, 10, 15 and 20. We conducted the evaluation experiment 100 times at each length of SFC and on each substrate network topologies. We collected computation time and draw the results.

Figure 5-21 and Figure 5-22 illustrate the cumulative distribution function (CDF) of computation time of Algorithm 3. The computation time was longer when an SFC consisted larger number of VNFs. Also, the time was longer when the substrate network contains more substrate nodes. This is because either a longer SFC or a larger substrate network would increase the computation time. However, in all topologies, the algorithm performs less than 1 millisecond to obtain the results which is acceptable.

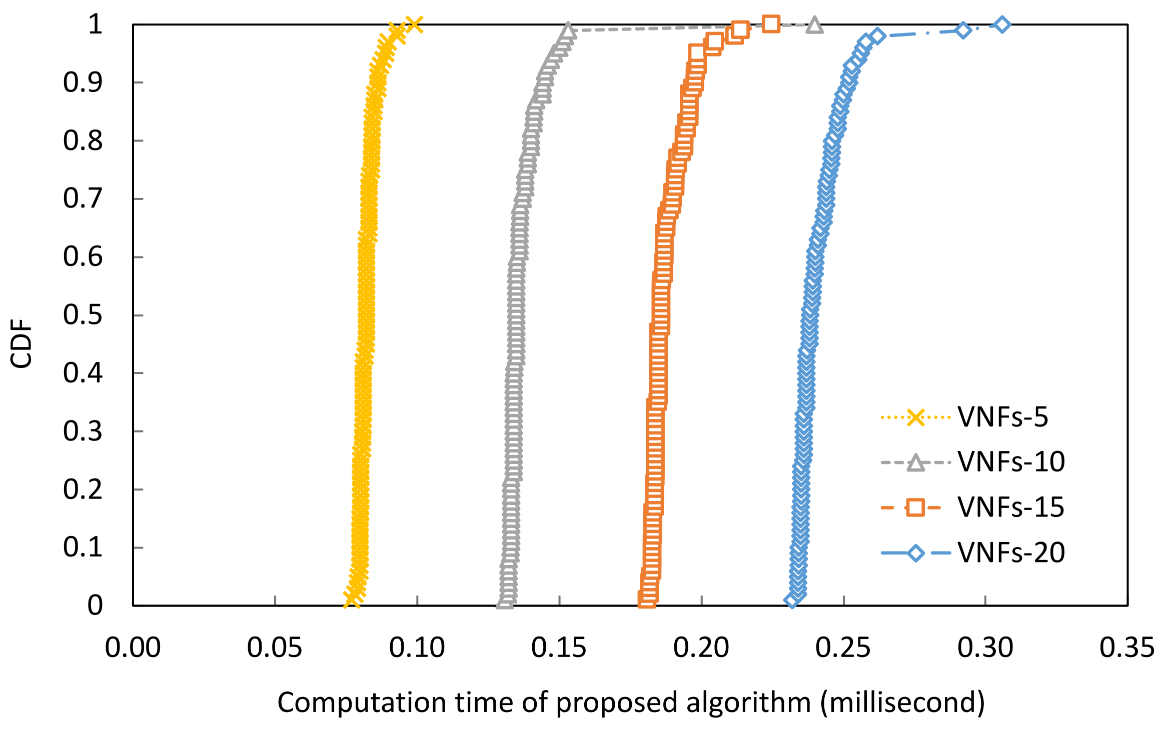

Figure 5-21 Performance of computation time of proposed algorithm.
(100 nodes, edge connectivity probability 0.5)

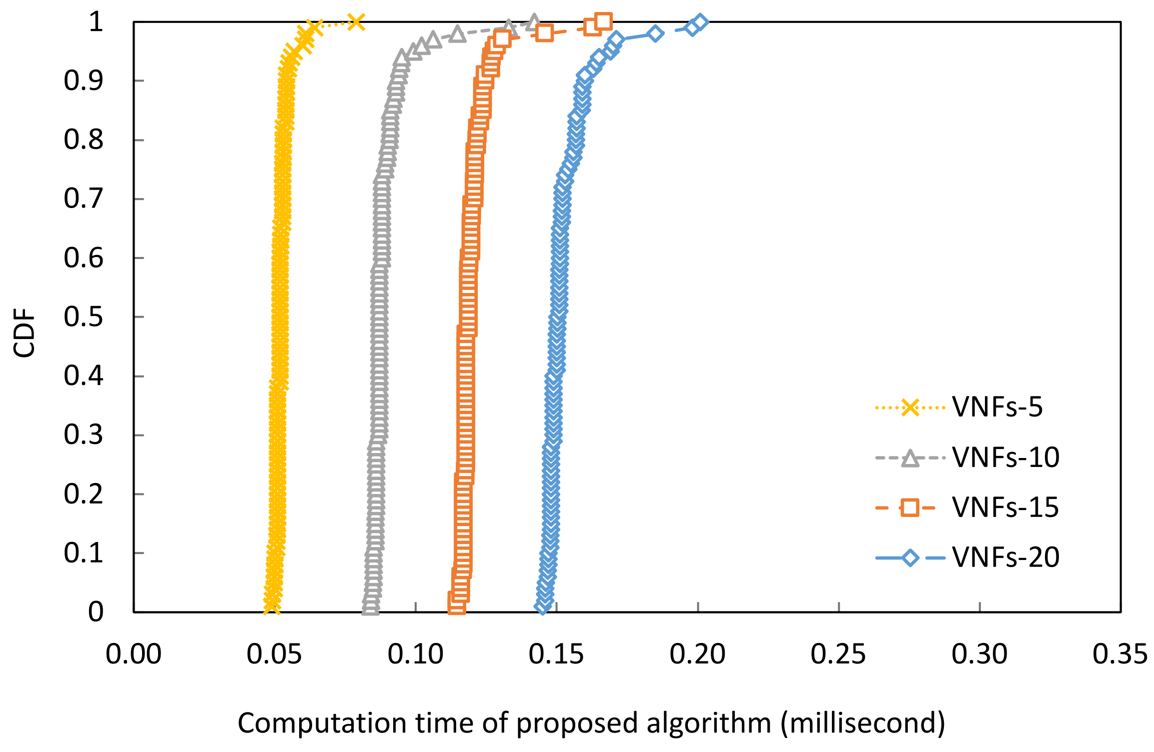

Figure 5-22 Performance of computation time of proposed algorithm.
(50 nodes, edge connectivity probability 0.5)

5.8. Discussion

5.8.1. Discussion on the necessity of distributed SFC deployment

In section 5.3.1, we have described the necessity of distributed SFC deployment. We showed that distributed SFC deployment could potentially avoid substrate network node overload, reduce risk of traffic loop and shorten time to recovery. We proposed an algorithm to satisfy the distributed SFC deployment requirement.

Although our algorithm can satisfy the requirement well, some deep investigations could be conducted. For example, how much load a distributed SFC deployment algorithm can release to the substrate network, how extent the risk of traffic loop can be reduce, and how much the recovery time would be shortened when failure of the substrate network node happened. We consider these as future works.

5.8.2. Discussion on large number of SFC requests

As shown in previous sections, our algorithm could obtain a higher distribution requirement satisfaction rate while maintaining a shorter end-to-end delay for an SFC. To achieve this, we modified the layered graph approach by breaking connections between layers of graph. This is efficient for one SFC, but it must consider the case of large number of SFC requests. In such case, multiple SFCs with same types of VNFs but different src and dst would be deployed on several same server nodes which pre-install these VNFs. As src and dst in SFC requests are different, solutions to some of these SFC requests may not travel through optimal paths. Although distribution requirement satisfaction is the main purpose of this chapter, an optimal path (shorter end-to-end delay path) should be achieved as much as possible.

Regarding to this consideration, there may be several approaches to solve this problem. One of such approach could be to try to break the connections between layers based on some other metrics such as degree of a node or betweenness centrality value of a node. These metrics are topology property of substrate network thus can be helpful to find a solution with awareness of substrate network topology. Another approach could be to calculate the shortest path without firstly breaking the connections. After found the path, it could migrate VNFs that have been deployed on the same node to other substrate nodes one by one in order to satisfy the distribution requirement. As the solution is derived from the shortest path, the end-to-end delay could be relatively shorter than no any design.

Besides, an algorithm could leverage deployment information on substrate network such as number of VNFs that has been deployed on one node, or resource consumption of nodes, to balance the usage of resources among substrate network nodes. We consider these approaches as future work as they should be carefully designed and evaluated.

5.9. Conclusion

In this chapter, we analyzed the necessity for distributed deployment of VNFs of an SFC in terms of distributed deployment requirement. Distributed deployment of VNFs of an SFC is critical for guaranteeing network services performance. As mentioned in the previous sections, a distributed deployment can shorten recovery time, balance load usage, avoid loops, etc. However, current researches in literature focus more on deployment of VNFs with satisfying resource constraints than placement policy such as distributed deployment requirement in this chapter. As shown in previous sections, only considering QoS such as end-to-end delay of SFCs will result in a distributed deployment requirement violation. The deployment algorithm must consider both distributed deployment requirement and the end-to-end delay of SFCs.

To solve this problem, we borrow this advantage of the layered graph, modify it to applicable to deploy VNFs of an SFC with satisfying the distributed deployment requirement and simultaneously obtaining a minimum end-to-end delay of an SFC. We conducted numerical evaluations on different substrate network topologies. The results of evaluations show that our proposed distributed SFC placement algorithm performs well in ensuring distributed deployment requirement with an acceptable computation time cost. We conducted extensive experiments to compare our algorithm against other algorithms in literature. The results show that our algorithm can 100% satisfy the distributed deployment requirement superior to the non-modified layered graph algorithm which can merely satisfy the distributed deployment requirement. Moreover, our proposed algorithm can outperform the bipartite matching algorithm and random algorithm in terms of end-to-end delay of SFCs, while keeping an equal or higher distributed deployment satisfaction rate comparing with these two algorithms. Specifically, our proposed algorithm achieves a maximum 46% and 60% reduction of end-to-end delay by comparing with bipartite matching algorithm on JPN48 and random graph network models, respectively. our proposed algorithm achieves a maximum 38% and 60% reduction of end-to-end delay by comparing with random algorithm on JPN48 and random graph network models, respectively.

Chapter 6

6. Conclusions and Future Work

6.1. Conclusions

SFC deployment is a key technology to realize network service in NFV. In this thesis, we focused on the problem of service function chain deployment with satisfying particular requirements. Specially, we consider the cost requirement, end-to-end delay requirement, and distributed deployment requirement.

To deploy SFC with a minimized cost, in chapter 3, we formulated a mathematic model with objective of minimizing setup costs and operational costs. With a realistic definition of setup cost and operation cost, our model can reduce cost up to 30% in average. Moreover, our model also considers guaranteeing the end-to-end delay deadline, which is a critical requirement to SFCs, such as real-time applications. Our model can satisfy 100% end-to-end delay deadline requirements of SFC requests.

In chapter 4, we analyzed that the end-to-end delay of an SFC is one of critical QoS metrics for network services. A shorter end-to-end delay can significantly improve the user's experiences. Motivated by this, we addressed the problem of SFC deployment with shorter end-to-end delay requirement. We proposed a delay aware SFC deployment algorithm by leveraging dynamic programming technique. The results of experiments show that our algorithms achieved a maximum 81.95% reduction of end-to-end delay comparing with randomly deployment of SFC on JPN48 network model, and more than 50% reduction of end-to-end delay on the other network model. Besides, our algorithm achieves a good performance in terms of acceptance rate. The short end-to-end delay and high acceptance rate indicate that our algorithm can achieve a better QoS guarantee and efficient network resource utilization.

In chapter 5, we perceived that consolidating VNFs of an SFC onto a substrate network node will potentially lead to overload resource usage, traffic loops, long recovery time, etc. Motivated by this observation, we addressed the problem of SFC deployment with distributed deployment requirement. Distributed deployment requirement avoids to deploying multiple VNFs of an SFC onto a same substrate network, so that reduce the risk of degradation. We designed a distributed SFC deployment algorithm with distributed deployment requirement by leveraging advantages of layered graph approach. The results of experiments show that our proposed algorithm can 100% satisfy the distributed deployment requirements on different network model. Besides, our algorithm can achieve a maximum 46% and 60% reduction of end-to-end delay by comparing with bipartite matching algorithm on JPN48 and random graph network models, respectively. Our proposed algorithm achieves a maximum 38% and 60% reduction of end-to-end delay by comparing with random algorithm on JPN48 and random graph network models, respectively. The 100% satisfaction of distributed deployment requirements and high reduction of end-to-end delay indicates that our algorithm can perform a reliable network service and a better QoS guarantee.

6.2. Future works

This thesis has studied the problem of resource allocation to SFCs, formulated a mathematic model and proposed several algorithms for solving this problem from different aspects. To gain all benefit from NFV, there are still many works to consider. We provide some directions for future works.

SFC at MEC: Multi-access edge computing (MEC) allows applications deployed at the edge of network where is closed to the end users. By leveraging this feature, MEC enables an ultra-low latency and high bandwidth environment and many new applications and business are emerging in MEC, such as connected autonomous cars, video analytics, Internet of Things (IoT), and augmented reality (AR). By incorporation of NFV, the MEC can benefit flexibility and efficiency in network service deployment. However, issues such as mobility are still remained to research.

Machine learning for SFC: The future network such as 5G are expected to provide various services such as enhanced Mobile Broadband (eMBB), massive IoT connections(mMTC), ultra-reliable low latency communications (uRLLC), with dynamic traffic load, and various QoS requirements. Thus, it needs a fast and dynamic resource allocation mechanism for future networks to adapt to different the demand of network services. Machine learning is considered as a promising technique for tackling such problem. By exploiting past experiences, machine learning can improve the future performance of the whole network system. As described in this survey [152], machine learning technique has already been utilized in many aspects of network, such as traffic classification, routing optimization, QoS/QoE prediction, security and resource management. Therefore, it is an interesting topic to study how to incorporate machine learning technique to improve the performance of network service chain with maximized utilization of resource and minimized cost of a substrate network. For example, real time network resources consumption could be predicted by machine learning techniques such as reinforcement learning and long short-term memory model (LSTM), and on basis of the prediction, the resources can be efficiently allocated.

Interdomain SFC deployment: Internet consists multiple domains operated by many network operators. Each network operator has different policies and strategies. A network service may travel through multiple domains for end-to-end service delivery. In such context, the deployment of SFCs will be very difficult as it must deal with different network operators among which some information such as topology, management policies cannot be shared [153]. To address interdomain SFC deployment, two types approaches are envisioned, centralized approach and distributed approach, as described in [154] and [155][156], respectively. However, as more network operators are considering to integrating NFV into their networks, interdomain SFC deployment will be more challenge and deserved to be researched more deeply.

Bibliography

[1] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang, "An untold story of middleboxes in cellular networks," ACMSIGCOMM Comput. Commun. Rev., vol. 41, no. 4, pp. 374–385, 2011.

[2] Tom Barbette, "Architecture for programmable network infrastructure", PhD thesis, July, 2018.

[3] T. Benson, A. Akella, and A. Shaikh, "Demystifying configuration challenges and trade-offs in network-based ISP services," ACM SIGCOMM Computer Communication Review, 41(4), 302.

[4] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, "Design and Implementation of a Consolidated Middlebox Architecture," Nsdi, p. 24, 2012.

[5] J. Sherry et al., "Making middleboxes someone else's problem: network processing as a cloud service," ACM SIGCOMM Computer Communication Review 42.4 (2012): 13-24.

[6] R. Guerzoni et al., "Network functions virtualisation: An introduction, benefits, enablers, challenges & call for action," in Proc. SDNOpenFlow World Congr., 2012, pp. 1–16.

[7] B. Han, V. Gopalakrishnan, L. Ji and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," in IEEE Communications Magazine, vol. 53, no. 2, pp. 90-97, Feb. 2015.

[8] ETSI, Network Functions Virtualization (NFV); Architectural Framework, V1.2.1, Dec. 2014

[9] M. Veeraraghavan, T. Sato, M. Buchanan, R. Rahimi, S. Okamoto, and N. Yamanaka, "Network function virtualization: A survey," IEICE Trans. Commun., vol. E100.B, no. 11, pp. 1978–1991, 2017.

[10] 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.

[11] "ETSI group specifications on network function virtualization. 1st phase documents," ETSI Ind. Spec. Group (ISG) Netw. Functions Virtual- isation (NFV), Sophia-Antipolis Cedex, France, Jan. 2015. [Online].

[12] 3rd Generation Partnership Project (3GPP). Management of Mobile Networks that Include Virtualized Network Functions (MANO). [Online]. Available: http://www.3gpp.org/news-events/3gpp-news/1738-sa5_nfv_study

[13] "The Internet Engineering Task Force (IETF) Service Function Chaining (SFC) Working Group (WG)," 2019. [Online]. Available at: https:// datatracker.ietf.org/wg/sfc/charter/

[14] "Alliance for telecommunications industry solutions" 2019. [Online]. Available: https://www.atis.org/

[15] "The Broadband Forum," 2019. [Online]. Available: https://www.broadband-forum.org/

[16] "Open Platform for NFV (OPNFV)," 2019. [Online]. Available: https://www.opnfv.org/

[17] "Zero-time Orchestration, Operations and Management (ZOOM)," Tele- Manage. Forum, Morristown, NJ, USA, Tech. Rep., Aug. 2014.

[18] Infonetics Research, "Survey: SDN and NFV Moving from Lab to Field Trials; Operators Name Top NFV Use Case," 2014. [Online]. Available: https://www.businesswire.com/news/home/20140403006434/en/Infonetics-Survey-SDN-NFV-Moving- Lab-Field.

[19] W. Zhou, Y. Yang, M. Xu and H. Chen, "Accommodating Dynamic Traffic Immediately: A VNF Placement Approach," ICC 2019 - 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 2019, pp. 1-6.

[20] M. Ghaznavi, N. Shahriar, S. Kamali, R. Ahmed and R. Boutaba, "Distributed Service Function Chaining," in IEEE Journal on Selected Areas in Communications, vol. 35, no. 11, pp. 2479-2489, Nov. 2017.

[21] J. Gil Herrera and J. F. Botero, "Resource Allocation in NFV: A Comprehensive Survey," in IEEE Transactions on Network and Service Management, vol. 13, no. 3, pp. 518-532, Sept. 2016.

[22] ITU-T, Rec. E.800 (09/2008), "Definitions of terms related to quality of service", Sept, 2008.

[23] K. U. Rehman Laghari and K. Connelly, "Toward total quality of experience: A QoE model in a communication ecosystem," in IEEE Communications Magazine, vol. 50, no. 4, pp. 58-65, April 2012.

[24] M. Khokhar, T. Ehlinger, C. Barakat, "From Network Traffic Measurements to QoE for Internet Video," IFIP Networking Conference, Varsovie, Poland, May 2019, pp. 1-9.

[25] V. Sekar, S. Ratnasamy, M. K. Reiter, N. Egi, and G. Shi, "The middlebox manifesto: Enabling innovation in middlebox deployment," in Proc.ACM HotNets-X, Cambridge, MA, USA, 2011, pp. 1–6.

[26] RFC 3234 Middleboxes: Taxonomy and Issues, Feb. 2002.

[27] D. A. Joseph, A. Tavakoli, and I. Stoica, "A policy-aware switching layer for data centers," in Proc. ACM SIGCOMM, Seattle, WA, USA, 2008, pp. 51–62.

[28] P. Quinn, and T. Nadeau, "Problem statement for service function chaining," IETF RFC 7498 (2015).

[29] ETSI GS NFV 003: "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV".

[30] Network Functions Virtualisation (NFV); Infrastructure Overview.

[31] NFVI PoP Network Topology: Problem Statement. [Online]. Available: http://www.watersprings.org/pub/id/draft-bagnulo-nfvrg-topology-01.html#ETSI_GS_NFV-INF_001

[32] ETSI, Network Functions Virtualisation (NFV); Management and Orchestration, V1.1.1, Dec. 2014.

[33] ETSI, "Network Functions Virtualisation – White Paper on NFV priorities for 5G," the NFV#17 Plenary meeting, Bilbao, Spain, Feb. 21st, 2017

[34] ETSI, "NFV: Use Cases", ETSI White Paper, 2013.

[35] ETSI, "NFV: Network Operator Perspectives on Industry Progress", ETSI White Paper, 2014.

[36] ETSI, "NFV: Terminology for Main Concepts in NFV", ETSI White Paper, 2014.

[37] ETSI, "Network Functions Virtualisation – White Paper on NFV priorities for 5G", ETSI White Paper, 2017.

[38] J. Halpern, C. Pignataro, "Service Function Chaining (SFC) Architecture," 2015. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-sfc-architecture/

[39] Z. A. Qazi, C. C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying middlebox policy enforcement using SDN," Comput. Commun. Rev., vol. 43, no. 4, pp. 27–38, 2013.

[40] M. Arregoces and M. Portolani. Data Center Fundamentals. Cisco Press, 2003.

[41] D. Zheng, C. Peng, E. Guler, G. Luo, L. Tian and X. Cao, "Hybrid Service Chain Deployment in Networks with Unique Function," ICC 2019 - 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 2019, pp. 1-6.

[42] H. Moens and F. D. Turck, "VNF-P: A model for efficient placement of virtualized network functions," 10th International Conference on Network and Service Management (CNSM) and Workshop, Rio de Janeiro, 2014, pp. 418-423.

[43] H. Huang, S. Guo, J. Wu and J. Li, "Service Chaining for Hybrid Network Function," in IEEE Transactions on Cloud Computing, vol. 7, no. 4, pp. 1082-1094, 1 Oct.-Dec. 2019.

[44] C. Mouradian, S. Kianpisheh and R. H. Glitho, "Application Component Placement in NFV-based Hybrid Cloud/Fog Systems," 2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), Washington, DC, 2018, pp. 25-30.

[45] S. N. Afrasiabi, S. Kianpisheh, C. Mouradian, R. H. Glitho and A. Moghe, "Application Components Migration in NFV-based Hybrid Cloud/Fog Systems," 2019 IEEE

International Symposium on Local and Metropolitan Area Networks (LANMAN), Paris, France, 2019, pp. 1-6.

[46] Q. Zhang, F. Liu and C. Zeng, "Adaptive Interference-Aware VNF Placement for Service-Customized 5G Network Slices," IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, Paris, France, 2019, pp. 2449-2457.

[47] V. Reddy, G. Garg, B. R. Tamma and F. A. Antony, "Interference Aware Network Function Selection Algorithm for Next Generation Networks," 2019 IEEE Conference on Network Softwarization (NetSoft), Paris, France, 2019, pp. 54-59.

[48] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," LISA'99: Proceedings of the 13th USENIX conference on System administration, pp. 229-238, November 1999.

[49] Snort, [Online]. Available: https://www.snort.org/.

[50] PktStat, [Online]. Available: https://linux.die.net/man/1/pktstat/.

[51] C. Zeng, F. Liu, S. Chen, W. Jiang and M. Li, "Demystifying the Performance Interference of Co-Located Virtual Network Functions," IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, Honolulu, HI, 2018, pp. 765-773.

[52] R. S. Kannan, A. Jain, M. A. Laurenzano, L. Tang and J. Mars, "Proctor: Detecting and Investigating Interference in Shared Datacenters," 2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Belfast, 2018, pp. 76-86.

[53] G. Moualla, T. Turletti and D. Saucez, “An Availability-Aware SFC Placement Algorithm for Fat-Tree Data Centers,” 2018 IEEE 7th International Conference on Cloud Networking (CloudNet), Tokyo, 2018, pp. 1-4.

[54] S. Herker, X. An, W. Kiess, S. Beker and A. Kirstaedter, “Data-Center Architecture Impacts on Virtualized Network Functions Service Chain Embedding with High Availability Requirements,” 2015 IEEE Globecom Workshops (GC Wkshps), San Diego, CA, 2015, pp. 1-7.

[55] J. Kong et al., "Guaranteed-Availability Network Function Virtualization with Network Protection and VNF Replication," GLOBECOM 2017 - 2017 IEEE Global Communications Conference, Singapore, 2017, pp. 1-6.

[56] L. Yala, P. A. Frangoudis and A. Ksentini, "Latency and Availability Driven VNF Placement in a MEC-NFV Environment," 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 2018, pp. 1-7.

[57] J. Fan et al., "A Framework for Provisioning Availability of NFV in Data Center Networks," in IEEE Journal on Selected Areas in Communications, vol. 36, no. 10, pp. 2246-2259, Oct. 2018.

[58] N. ISG, "Network functions virtualisation (nfv)-virtualisation requirements", ETSI Technical Report, 2013.

[59] Z. Xu, X. Zhang, S. Yu and J. Zhang, "Energy-Efficient Virtual Network Function Placement in Telecom Networks," 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, 2018, pp. 1-7.

[60] D. Zheng, C. Peng, X. Liao, G. Luo, L. Tian and X. Cao, "Service Function Chaining and Embedding with Spanning Closed Walk," 2019 IEEE 20th International Conference on High Performance Switching and Routing (HPSR), Xi'An, China, 2019, pp. 1-5.

[61] K. Mebarkia and Z. Zsóka, "Service traffic engineering: Avoiding link overloads in service chains," in Journal of Communications and Networks, vol. 21, no. 1, pp. 69-80, Feb. 2019.

[62] G. Garg, V. Reddy, A. Antony Franklin and B. R. Tamma, "DAVIS: A Delay-Aware VNF Selection Algorithm for Service Function Chaining," 2019 11th International Conference on Communication Systems & Networks (COMSNETS), Bengaluru, India, 2019, pp. 436-439.

[63] S. Vassilaras et al., "The Algorithmic Aspects of Network Slicing," in IEEE Communications Magazine, vol. 55, no. 8, pp. 112-119, Aug. 2017.

[64] Y. Bi, C. Colman-Meixner, R. Wang, F. Meng, R. Nejabati and D. Simeonidou, "Resource Allocation for Ultra-Low Latency Virtual Network Services in Hierarchical 5G Network," ICC 2019 - 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 2019, pp. 1-7.

[65] D. Harutyunyan, N. Shahriar, R. Boutaba and R. Riggio, "Latency-Aware Service Function Chain Placement in 5G Mobile Networks," 2019 IEEE Conference on Network Softwarization (NetSoft), Paris, France, 2019, pp. 133-141.

[66] H. A. Alameddine, C. Assi, M. H. Kamal Tushar and J. Y. Yu, "Low-Latency Service Schedule Orchestration in NFV-based Networks," 2019 IEEE Conference on Network Softwarization (NetSoft), Paris, France, 2019, pp. 378-386.

[67] M. Zinkevich, "Online Convex Programming and Generalized Infinitesimal Gradient Ascent", Proc. of International Conference on Machine Learning (ICML), 2003.

[68] S. Shalev-Shwartz, "Online Learning and Online Convex Optimization", Journal of Foundations and Trends in Machine Learning, vol. 4, no. 2, pp. 107-194, 2012.

[69] N. Chen, A. Agarwal, A. Wierman, S. Barman, L.L.H. Andrew, "Online Convex Optimization Using Predictions", Proc. of ACM SIGMETRICS, 2015.

[70] Y. Jia, C. Wu, Z. Li, F. Le and A. Liu, "Online Scaling of NFV Service Chains Across Geo-Distributed Datacenters," in IEEE/ACM Transactions on Networking, vol. 26, no. 2, pp. 699-710, April 2018.

[71] X. Zhang, C. Wu, Z. Li and F. C. M. Lau, "Proactive VNF provisioning with multi-timescale cloud resources: Fusing online learning and online optimization," IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, Atlanta, GA, 2017, pp. 1-9.

[72] H. Kim, D. Lee, S. Jeong, H. Choi, J. Yoo and J. W. Hong, "Machine Learning-Based Method for Prediction of Virtual Network Function Resource Demands," 2019 IEEE Conference on Network Softwarization (NetSoft), Paris, France, 2019, pp. 405-413.

[73] P. T. A. Quang, Y. Hadjadj-Aoul and A. Outtagarts, "A Deep Reinforcement Learning Approach for VNF Forwarding Graph Embedding," in IEEE Transactions on Network and Service Management, vol. 16, no. 4, pp. 1318-1331, Dec. 2019.

[74] IP Multimedia Subsystem, 2019. [Online]. Available: https://en.wikipedia.org/wiki/IP_Multimedia_Subsystem

[75] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," ACM SIGCOMM Computer Communication Review 38, no. 2 (2008): 17–29.

[76] X. Cheng et al., "Virtual network embedding through topology-aware node ranking," ACM SIGCOMM Computer Communication Review 41, no. 2 (2011): 38–47.

[77] N.M.M.K. Chowdhury, M.R. Rahman, and R. Boutaba, "ViNEYard: virtual network embedding algorithms with coordinated node and link mapping," IEEE/ACM Trans. Netw. 20 (1) (2012) 206–219.

[78] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," In IEEE INFOCOM, 2014.

[79] "Network functions virtualisation (NFV): Network operator perspectives on industry progress," ETSI, Tech. Rep., 2014.

[80] "AT&T vision alignment challenge technology survey," AT&T, Tech. Rep., 2013. [Online]. Available: https://www.att.com/Common/about_us/pdf/AT&T%20Domain%202.0%20Vision%20White%20Paper.pdf

[81] "Network functions virtualization," Hewlett Packard, Tech. Rep., 2014. [Online] Available: http://www.hp.com/hpinfo/newsroom/press_kits/2014/MWC/White_Paper_NFV.pdf

[82] “Value-added services and service chaining: Deployment considerations and challenges,” Sandvine, Tech. Rep., 2014.

[83] G. Xilouris et al., "T-NOVA: A marketplace for virtualized network functions," 2014 European Conference on Networks and Communications (EuCNC), Bologna, 2014, pp. 1-5.

[84] M. Schöller, M. Stiemerling, A. Ripke and R. Bless, "Resilient deployment of virtual network functions," 2013 5th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Almaty, 2013, pp. 208-214.

[85] S. M. Sinha, Mathematical Programming: Theory and Methods, Elsevier Science & Technology Books, 2006.

[86] T. Lin, Z. Zhou, M. Tornatore and B. Mukherjee, "Optimal Network Function Virtualization Realizing End-to-End Requests," 2015 IEEE Global Communications Conference (GLOBECOM), San Diego, CA, 2015, pp. 1-6.

[87] M. Zeng, W. Fang and Z. Zhu, "Orchestrating Tree-Type VNF Forwarding Graphs in Inter-DC Elastic Optical Networks," in Journal of Lightwave Technology, vol. 34, no. 14, pp. 3330-3341, 15 July15, 2016.

[88] B. Addis, D. Belabed, M. Bouet and S. Secci, "Virtual network functions placement and routing optimization," 2015 IEEE 4th International Conference on Cloud Networking (CloudNet), Niagara Falls, ON, 2015, pp. 171-177.

[89] S. Mehraghdam, M. Keller and H. Karl, "Specifying and placing chains of virtual network functions," 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet), Luxembourg, 2014, pp. 7-13.

[90] M. Savi, M. Tornatore and G. Verticale, "Impact of processing costs on service chain placement in network functions virtualization," 2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN), San Francisco, CA, 2015, pp. 191-197.

[91] M. M. Tajiki, S. Salsano, L. Chiaraviglio, M. Shojafar and B. Akbari, "Joint Energy Efficient and QoS-Aware Path Allocation and VNF Placement for Service Function Chaining," in IEEE Transactions on Network and Service Management, vol. 16, no. 1, pp. 374-388, March 2019.

[92] I. Jang, S. Choo, M. Kim, S. Pack and M. Shin, "Optimal network resource utilization in service function chaining," 2016 IEEE NetSoft Conference and Workshops (NetSoft), Seoul, 2016, pp. 11-14.

[93] 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," 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, 2015, pp. 98-106.

[94] A. Gupta, M. F. Habib, P. Chowdhury, M. Tornatore and B. Mukherjee, "On service chaining using Virtual Network Functions in Network-enabled Cloud systems," 2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS), Kolkata, 2015, pp. 1-3.

[95] IBM ILOG CPLEX Optimization Studio v12.8, [Online]. Available: https://www.ibm.com/products/ilog-cplex-optimization-studio.

[96] R. B. Miller, "Response time in man-computer conversational transactions," Proc. AFIPS Fall Joint Computer Conference, Vol. 33, pp 267-277, 1968.

[97] J. Nielsen, "Response Times: The Three Important Limits," excerpt from Usability Engineering, Morgan Kaufmann, San Francisco, 1994.

[98] S. Cheshire "Latency and the Quest for Interactivity," November 1996. [Online]. Available: http://www.stuartcheshire.org/papers/LatencyQuest.html.

[99] "Amazon - Every 100ms delay costs 1% of sales." [Online]. Available: http://tinyurl.com/k635quz.

[100] J. Lee et al., "Application-driven bandwidth guarantees in datacenters," ACM SIGCOMM Computer Communication Review. Vol. 44. No. 4. ACM, 2014.

[101] ITU-T Rec. G.1010(2001), "End-user multimedia QoS categories," November 2001.

[102] P. Agyapong, M. Iwamura, D. Staehle et al., "Design considerations for a 5G network architecture", IEEE Communications Magazine, vol. 52, no. 11, pp. 65-75, 2014.

[103] ITU-R Rec. M.2083-0 (2015), "IMT Vision — Framework and Overall Objectives of the Future Development of IMT for 2020 and Beyond," Sept. 2015.

[104] J. Liu, Y. Li, Y. Zhang, L. Su and D. Jin, "Improve Service Chaining Performance with Optimized Middlebox Placement," in IEEE Transactions on Services Computing, vol. 10, no. 4, pp. 560-573, 1 July-Aug. 2017.

[105] M. Xia, M. Shirazipour, Y. Zhang, H. Green and A. Takacs, "Network Function Placement for NFV Chaining in Packet/Optical Datacenters," in Journal of Lightwave Technology, vol. 33, no. 8, pp. 1565-1570, 15 April15, 2015.

[106] T. H. Cormen, "Introduction to algorithms," MIT press, 2009.

[107] Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart, "Exploring network structure, dynamics, and function using NetworkX", in Proceedings of the 7th Python in Science

Conference (SciPy2008), Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11–15, Aug 2008

[108] Python, [Online]. Available: https://www.python.org/

[109] P. Erdo˝s and A. Re´nyi, “On random graphs,” Publ. Math., 6, 290, 1959.

[110] E.N. Gilbert, “Random graphs,” Ann. Math. Stat., vol.30, no.4, pp.1141–1144, 1959.

[111] A. Dwaraki and T. Wolf, “Adaptive service-chain Routing for virtual network functions in software-defined networks,” in Workshop on Hot topics in Middleboxes and Network Function Virtualization (HotMIddlebox), 2016, pp. 32–37.

[112] A. Tomassilli, F. Giroire, N. Huin and S. Pérennes, “Provably Efficient Algorithms for Placement of Service Function Chains with Ordering Constraints,” IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, Honolulu, HI, 2018, pp. 774-782.

[113] S. Orlowski, R. Wessa˙ly, M. Pio´ro, and A. Tomaszewski, “Sndlib 1.0—survivable network design library,” Networks, vol.55, no.3, pp.276–286, 2010.

[114] S. Yang, F. Li, S. Trajanovski, X. Chen, Y. Wang and X. Fu, "Delay-Aware Virtual Network Function Placement and Routing in Edge Clouds," in IEEE Transactions on Mobile Computing. 2019, (To be appear)

[115] L. Qu, C. Assi, K. Shaban and M. J. Khabbaz, "A Reliability-Aware Network Service Chain Provisioning with Delay Guarantees in NFV-Enabled Enterprise Datacenter Networks," in IEEE Transactions on Network and Service Management, vol. 14, no. 3, pp. 554-568, Sept. 2017.

[116] H. Hawilo, M. Jammal and A. Shami, "Network Function Virtualization-Aware Orchestrator for Service Function Chaining Placement in the Cloud," in IEEE Journal on Selected Areas in Communications, vol. 37, no. 3, pp. 643-655, March 2019.

[117] W. Guan, X. Wen, L. Wang, Z. Lu and Y. Shen, "A Service-Oriented Deployment Policy of End-to-End Network Slicing Based on Complex Network Theory," in IEEE Access, vol. 6, pp. 19691-19701, 2018.

[118] A. Bavelas, “A mathematical model for group structure. Human organization,” Applied Anthropology, vol. 7, no. 3, pp. 16–30, 1948.

[119] L. C. Freeman, “A Set of Measures of Centrality Based on Betweenness,” Sociometry 40, no. 1 (1977): 35-41.

[120] U. Brandes, “A Faster Algorithm for Betweenness Centrality,” Journal of Mathematical Sociology 25(2):163-177, 2001.

[121] U. Brandes, “On Variants of Shortest-Path Betweenness Centrality and their Generic Computation,” Social Networks 30(2):136-145, 2008.

[122] U. Brandes and C. Pich, “Centrality Estimation in Large Networks,” International Journal of Bifurcation and Chaos 17(7):2303-2318, 2007.

[123] F. Carpio, S. Dhahri and A. Jukan, "VNF placement with replication for Loac balancing in NFV networks," 2017 IEEE International Conference on Communications (ICC), Paris, 2017, pp. 1-6.

[124] Japan Photonic Network Model. [Online]. Available: http://www.ieice.org/cs/pn/jpn/jpnm.html

[125] S. I. Arakawa et al., “Topological characteristic of japan photonic network model,” IEICE Tech. Rep., vol. 113, no. 91, pp. 7–12, 2013.

[126] Y. Kishi, N. Kitsuwan, H. Ito, B. C. Chatterjee and E. Oki, "Modulation-Adaptive Link-Disjoint Path Selection Model for 1 + 1 Protected Elastic Optical Networks," in IEEE Access, vol. 7, pp. 25422-25437, 2019.

[127] K. Tabota, and T. Tachibana, “Greedy-Based VNF Placement Algorithm for Dynamic Multipath Service Chaining,” IEICE Transactions on Communications, 2019, vol. E102.B, Issue 3, pp 429-438.

[128] S. Fujisawa, B. Yatabe, H. Takeshita, T. Oguma and A. Tajima, "Reliability enhancement by a joint optimization of elastic optical path allocation methods," 2017 Opto-Electronics and Communications Conference (OECC) and Photonics Global Conference (PGC), Singapore, 2017, pp. 1-3.

[129] A Brief History of NSF and the Internet, [Online]. Available: https://www.nsf.gov/news/news_summ.jsp?cntn_id=103050

[130] J. Liu, W. Lu, F. Zhou, P. Lu and Z. Zhu, "On Dynamic Service Function Chain Deployment and Readjustment," in IEEE Transactions on Network and Service Management, vol. 14, no. 3, pp. 543-553, Sept. 2017.

[131] Q. Sun, P. Lu, W. Lu and Z. Zhu, "Forecast-Assisted NFV Service Chain Deployment Based on Affiliation-Aware vNF Placement," 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, 2016, pp. 1-6.

[132] ETSI, GS NFV-INF 010 v1.1.1, Network Functions Virtualization: Service Quality Metrics, Dec, 2014.

[133] Affinity and Anti-Affinity Rules, [Online]. Available: https://www.cisco.com/c/en/us/td/docs/net_mgmt/elastic_services_controller/5-0/user/guide/Cisco-Elastic-Services-Controller-User-Guide-5-0/affinity_antiaffinity.pdf

[134] N. Bouten, M. Claeys, R. Mijumbi, J. Famaey, S. Latré and J. Serrat, "Semantic validation of affinity constrained service function chain requests," 2016 IEEE NetSoft Conference and Workshops (NetSoft), Seoul, 2016, pp. 202-210.

[135] ETSI GS NFV-REL 001 v1.1.1, Network Functions Virtualization (NFV): Resiliency Requirements, Jan, 2015.

[136] Z. Allybokus, N. Perrot, J. Leguay, L. Maggi, and E. Gourdin, "Virtual function placement for service chaining with partial orders and anti-affinity rules," NETWORKS, 71: 97-106, 2018.

[137] IDC, "DevOps and the Cost of Downtime: Fortune 1000 Best Practice Metrics Quantified." Stephen Elliot. December 2014, IDC

[138] IBM, Resiliency in the Cloud, [Online]. Available: https://www.ibm.com/downloads/cas/AVY5QYG0

[139] "Why distribution matters in NFV," White Paper, Alcatel-Lucent, Aug. 2014. [Online]. Available: https://www.tmcnet.com/tmc/whitepapers/documents/whitepapers/2014/10462-why-distribution-matters-nfv.pdf

[140] W. A. S. P. Abeysiriwardhana, J. Wijekoon and H. Nishi, "Optimized Service Function Path Selection for IoT Devices Using Virtual Network Function Performance Data," 2019 International Conference on Information Networking (ICOIN), Kuala Lumpur, Malaysia, 2019, pp. 165-170.

[141] N. Bouten, R. Mijumbi, J. Serrat, J. Famaey, S. Latré and F. De Turck, "Semantically Enhanced Mapping Algorithm for Affinity-Constrained Service Function Chain Requests," in IEEE Transactions on Network and Service Management, vol. 14, no. 2, pp. 317-331, June 2017.

[142] A. S. Jacobs, R. L. do Santos, M. F. Franco, E. J. Scheid, R. J. Pfitscher and L. Z. Granville, "Affinity measurement for NFV-enabled networks: A criteria-based approach," 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, 2017, pp. 125-133.

[143] A. S. Jacobs, R. J. Pfitscher, R. L. d. Santos, M. F. Franco, E. J. Scheid and L. Z. Granville, "Artificial neural network model to predict affinity for virtual network functions," NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, Taipei, 2018, pp. 1-9.

[144] H. Zhu and C. Huang, "Cost-Efficient VNF Placement Strategy for IoT Networks with Availability Assurance," 2017 IEEE 86th Vehicular Technology Conference (VTC-Fall), Toronto, ON, 2017, pp. 1-5.

[145] S. Choi, J. Turner, and T. Wolf, “Configuring sessions in programmable networks,” Computer Networks 41.2 (2003): 269-284.

[146] N. Huin, B. Jaumard and F. Giroire, "Optimal Network Service Chain Provisioning," in IEEE/ACM Transactions on Networking, vol. 26, no. 3, pp. 1320-1333, June 2018.

[147] G. Sallam, G. R. Gupta, B. Li and B. Ji, "Shortest Path and Maximum Flow Problems Under Service Function Chaining Constraints," IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, Honolulu, HI, 2018, pp. 2132-2140.

[148] Y. GU, Y. DING, Y. HU, "Joint Optimization of Delay Guarantees and Resource Allocation for Service Function Chaining," IEICE Transactions on Information and Systems, 2019, Volume E102.D, Issue 12, Pages 2611-2614.

[149] A. S. Asratian, T. M. J. Denley, R. Hggkvist, “Bipartite Graphs and Their Applications,” U.K., Cambridge: Cambridge Univ. Press, 1998.

[150] J. E. Hopcroft and R. M. Karp, "An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs" SIAM Journal of Computing 2.4 (1973), pp. 225-231.

[151] Source code for maximum matching, [Online]. Available: https://networkx.github.io/documentation/stable/_modules/networkx/algorithms/bipartite/matching.html

[152] J. Xie et al., "A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): Research Issues and Challenges," in IEEE Communications Surveys & Tutorials, vol. 21, no. 1, pp. 393-430, Firstquarter 2019.

[153] G. Sun, Y. Li, D. Liao and V. Chang, "Service Function Chain Orchestration Across Multiple Domains: A Full Mesh Aggregation Approach," in IEEE Transactions on Network and Service Management, vol. 15, no. 3, pp. 1175-1191, Sept. 2018.

[154] D. Dietrich, A. Abujoda, A. Rizk, P. Papadimitriou, "Multi-provider service chain embedding with Nestor", IEEE Trans. Netw. Service Manag., vol. 14, no. 1, pp. 91-105, Mar. 2017.

[155] A. Abujoda, P. Papadimitriou, "DistNSE: Distributed network service embedding across multiple providers", Proc. Int. Conf. Commun. Syst. Netw., pp. 1-8, 2016.

[156] Q. Zhang, X. Wang, I. Kim, P. Palacharla, T. Ikeuchi, "Vertex-centric computation of service function chains in multi-domain networks", Proc. IEEE Netsoft Conf. Workshops, pp. 211-218, 2016.

Appendix A Setup Cost Settings

In this appendix, we show the cost parameter settings of evaluations in section 3.5.3.

Figure A. 1 shows the setup cost setting at different number of SFC requests for evaluation in section 3.5.3. The x-axis is the number of SFC requests, and the y-axis is the setup cost to process VNFs. In the evaluation, we use four types of VNFs. In our proposed model, we randomly select four values from 5 to 10 to set the setup cost for each type of VNFs, and in previous model, we randomly select one value from 5 to 10 for all types of VNFs.

At each number of SFC requests, we select a new set of setup costs to types of VNFs and conduct the evaluation. The square marks show the setup costs at each number of SFC requests in previous model. As the previous model assumes that all types of VNFs have a same setup cost, the square marks have no variability at each number of SFC requests.

The triangle marks in the figure are the mean of setup costs to types of VNFs at each number of SFC requests. The error bars on the triangle marks are standard deviations of setup costs. The error bars indicate the variability of setup costs in our proposed model, as our model considers the setup costs are different to each type of VNFs.

However, it must be careful when select different setup cost values to different types of VNFs in such random way for our model, as it could select four identical values which is not the case for our model. Some approaches could avoid to this situation. One of such approach is to drop the identical values and select a new set values of setup costs.

As the sample size is small, we also show all data in Table A. 1 under the figures.

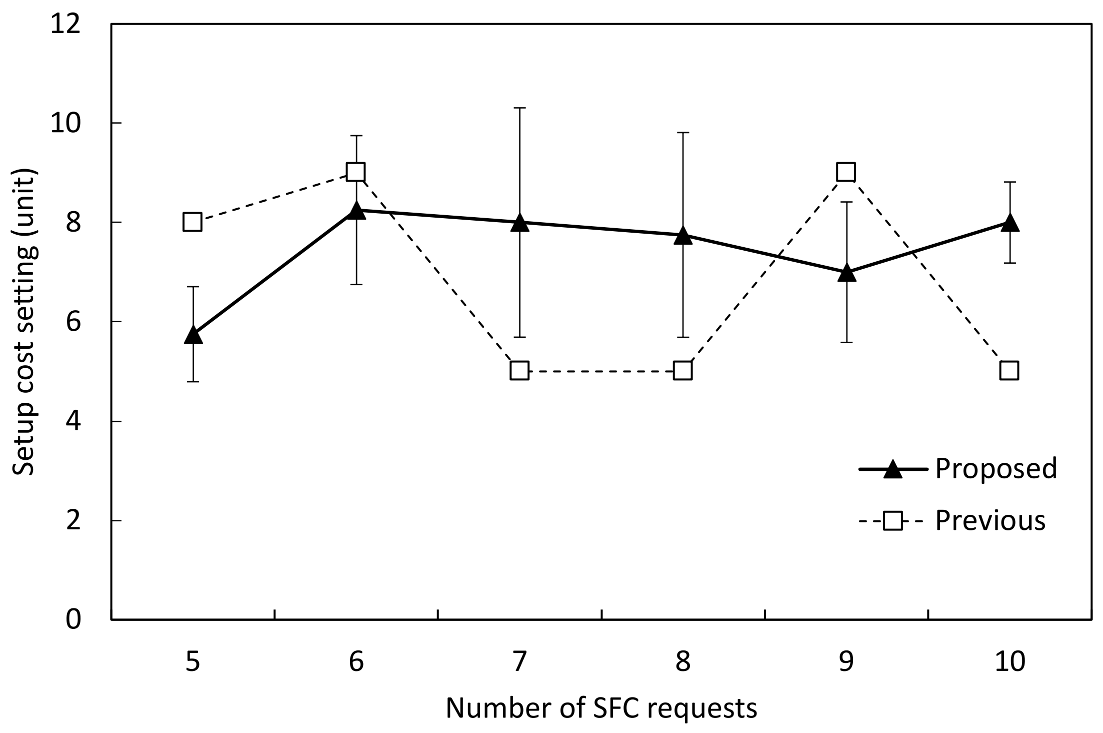

Figure A. 1 Setup cost setting at different number of SFC requests

Table A. 1 Data of setup cost settings at different number of SFC requests.

#SFC requests	**Models**	**Setup cost for types of VNF (unit)**				**Mean**	**Deviation**
		VNF1	VNF2	VNF3	VNF4		
5	Proposed	7	5	6	5	5.75	0.957427
	Previous	8				-	-
6	Proposed	10	9	7	7	8.25	1.5
	Previous	9				-	-
7	Proposed	10	6	10	6	8	2.309401
	Previous	5				-	-
8	Proposed	10	6	9	6	7.75	2.061553
	Previous	5				-	-
9	Proposed	6	6	7	9	7	1.414214
	Previous	9				-	-
10	Proposed	8	9	8	7	8	0.816497
	Previous	5				-	-

Appendix B Operational Cost Settings

Figure B. 1 through Figure B. 6 show the operational cost settings at each number of SFC requests for evaluation of cost reduction in section 3.5.3. The x-axis is the node index of the substrate network and the y-axis is the operational cost setting. For each substrate node (index), we randomly choose four value from 5 to 10 for operational cost of different types of VNFs in our proposed model, and randomly choose one value from 5 to 10 for operational cost of all types of VNFs in previous model.

The square marks show the operational costs to process a unit of VNFs in previous model. As previous model assumes one substrate node process all types of VNFs at same costs, the setting of operational cost is identical to all types of VNFs for a node index.

The triangle marks show the operational costs to process a unit of VNFs in our proposed model. The error bars on the triangle marks are the standard deviations of operational costs at different substrate nodes. The error bars indicate the variability of operational costs, as our model considers that a substrate node processes different types of VNFs at different operational costs. In both previous model and our model, the operational costs are different values to different substrate nodes.

However, it must be careful when select different operational cost values to different types of VNFs in such random way for our model, as it could select four identical values to four types of VNFs for all substrate nodes which is not the case for our model. Some approaches could avoid to this situation. One of such approach is to drop the identical values and select a new set values of operational costs. Note that same values of operational costs for a few number of substrate nodes could be kept instead of dropped, as shown in the figure of #SFC requests = 9, index 1. But it should be better to evaluate how such cases could affect the overall costs.

As the sample size is small, we also show all data from Table B. 1 through Table B. 6 below each figure.

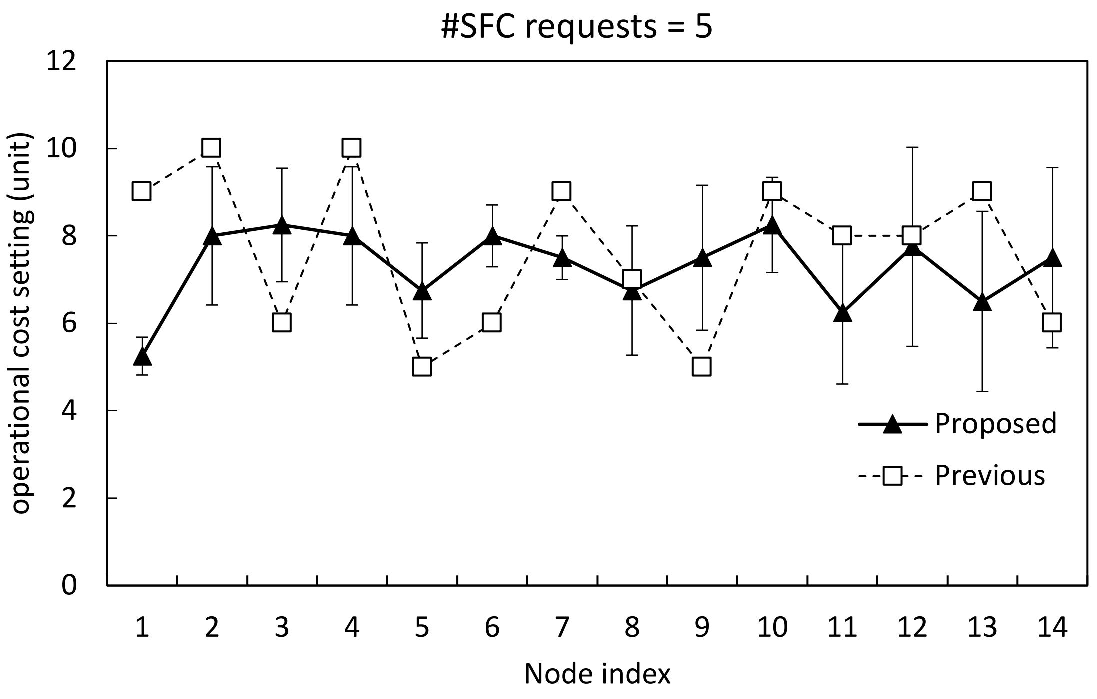

Figure B. 1 Operational cost setting at number of SFC requests 5

Table B. 1 Data of operational cost settings at number of SFC requests 5.

	Proposed						Previous
Node index	VNF1	VNF2	VNF3	VNF4	Mean	Deviation	All VNFs
1	5	6	5	5	5.25	0.433012702	9
2	6	9	10	7	8	1.58113883	10
3	10	7	9	7	8.25	1.299038106	6
4	9	7	6	10	8	1.58113883	10
5	7	7	5	8	6.75	1.089724736	5
6	8	9	7	8	8	0.707106781	6
7	8	8	7	7	7.5	0.5	9
8	7	9	5	6	6.75	1.479019946	7
9	6	10	8	6	7.5	1.658312395	5
10	7	8	8	10	8.25	1.089724736	9
11	5	9	5	6	6.25	1.639359631	8
12	10	5	10	6	7.75	2.277608395	8
13	5	6	10	5	6.5	2.061552813	9
14	9	10	6	5	7.5	2.061552813	6

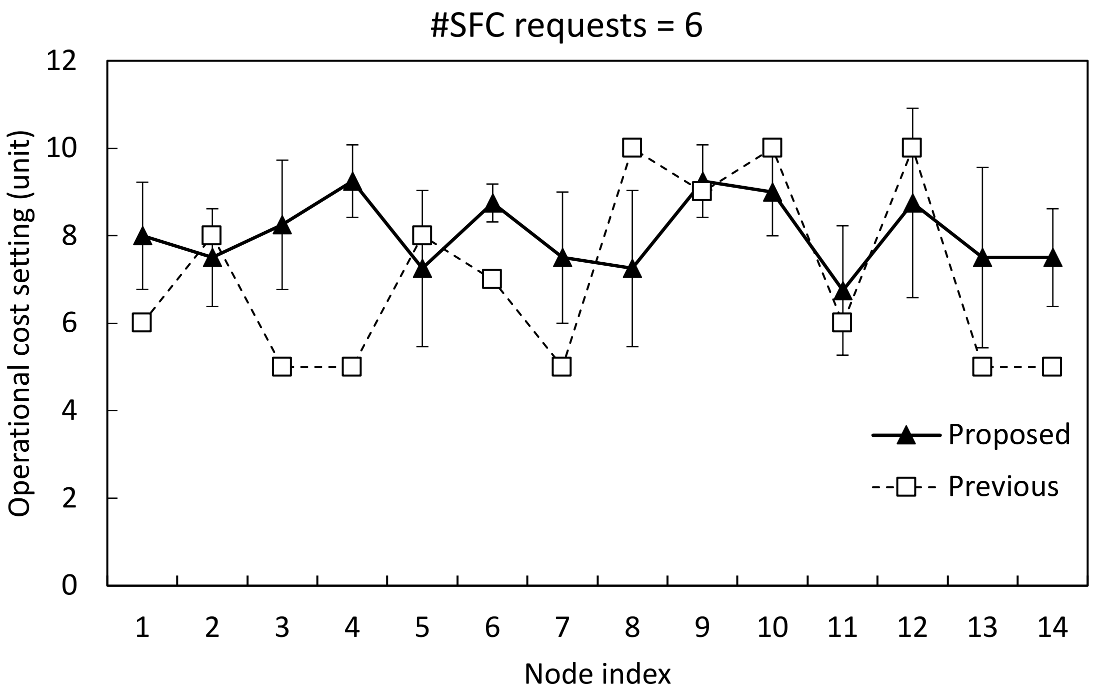

Figure B. 2 Operational cost setting at number of SFC requests 6.

Table B. 2 Data of operational cost settings at number of SFC requests 6.

	Proposed						Previous
Node index	VNF1	VNF2	VNF3	VNF4	Mean	Deviation	All VNFs
1	9	6	8	9	8	1.224745	6
2	7	9	8	6	7.5	1.118034	8
3	8	10	9	6	8.25	1.47902	5
4	8	10	9	10	9.25	0.829156	5
5	5	9	6	9	7.25	1.785357	8
6	8	9	9	9	8.75	0.433013	7
7	8	5	9	8	7.5	1.5	5
8	10	7	5	7	7.25	1.785357	10
9	9	10	8	10	9.25	0.829156	9
10	10	10	8	8	9	1	10
11	7	9	6	5	6.75	1.47902	6
12	10	5	10	10	8.75	2.165064	10
13	5	10	9	6	7.5	2.061553	5
14	7	8	9	6	7.5	1.118034	5

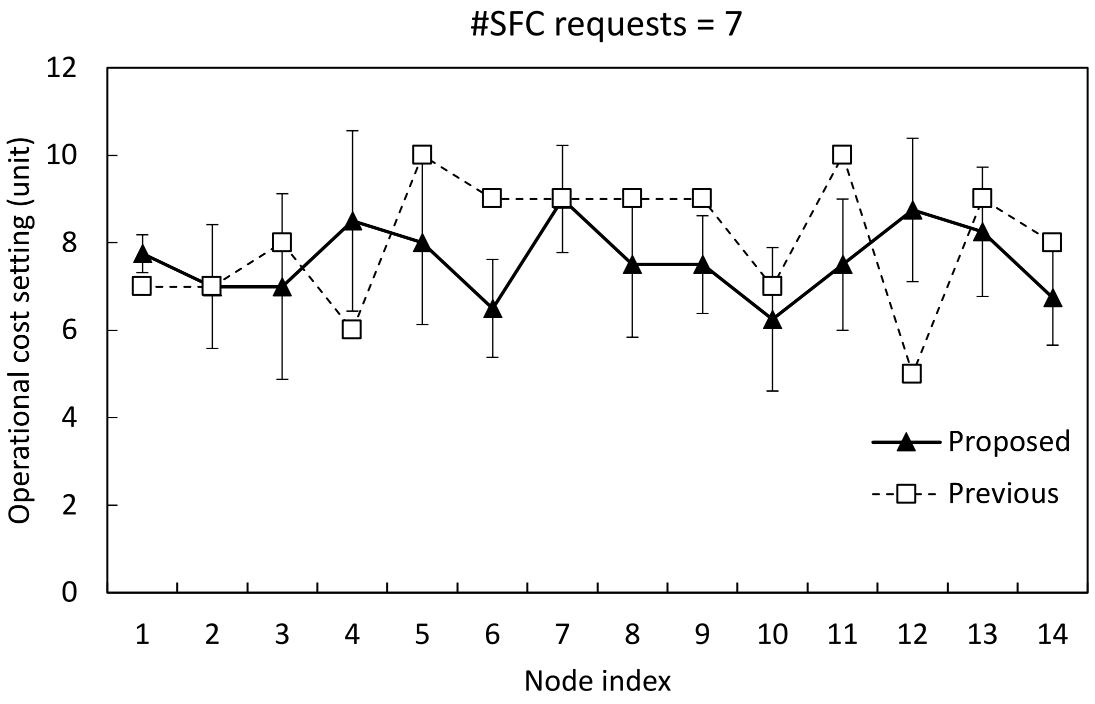

Figure B. 3 Operational cost setting at number of SFC requests 7.

Table B. 3 Data of operational cost settings at number of SFC requests 7.

	Proposed						**Previous**
Node index	VNF1	VNF2	VNF3	VNF4	Mean	Deviation	All VNFs
1	8	8	8	7	7.75	0.433013	7
2	7	5	7	9	7	1.414214	7
3	5	10	8	5	7	2.12132	8
4	5	9	10	10	8.5	2.061553	6
5	5	10	8	9	8	1.870829	10
6	6	7	5	8	6.5	1.118034	9
7	7	9	10	10	9	1.224745	9
8	9	7	5	9	7.5	1.658312	9
9	7	6	8	9	7.5	1.118034	9
10	9	5	6	5	6.25	1.63936	7
11	7	6	10	7	7.5	1.5	10
12	10	9	10	6	8.75	1.63936	5
13	6	10	9	8	8.25	1.47902	9
14	7	5	7	8	6.75	1.089725	8

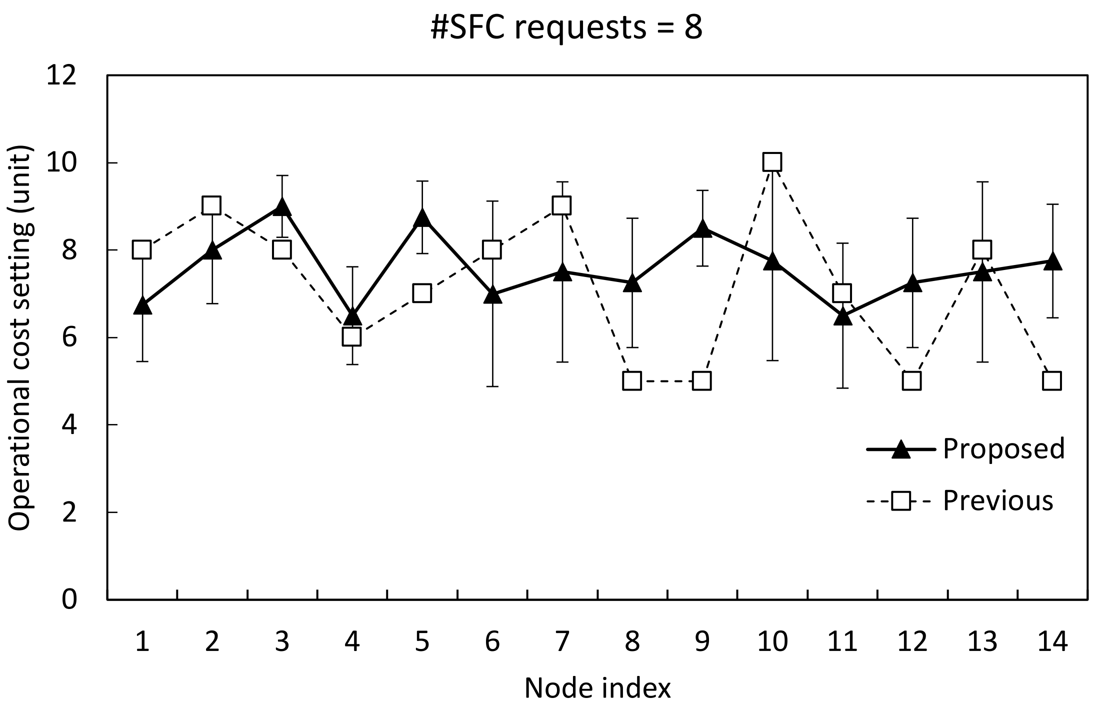

Figure B. 4 Operational cost setting at number of SFC requests 8.

Table B. 4 Data of operational cost settings at number of SFC requests 8.

	Proposed						Previous
Node index	VNF1	VNF2	VNF3	VNF4	Mean	Deviation	All VNFs
1	9	6	6	6	6.75	1.299038	8
2	9	8	9	6	8	1.224745	9
3	8	10	9	9	9	0.707107	8
4	5	8	7	6	6.5	1.118034	6
5	8	9	10	8	8.75	0.829156	7
6	5	5	8	10	7	2.12132	8
7	10	5	9	6	7.5	2.061553	9
8	9	7	5	8	7.25	1.47902	5
9	8	8	10	8	8.5	0.866025	5
10	6	10	5	10	7.75	2.277608	10
11	9	5	7	5	6.5	1.658312	7
12	7	5	9	8	7.25	1.47902	5
13	10	5	9	6	7.5	2.061553	8
14	7	7	7	10	7.75	1.299038	5

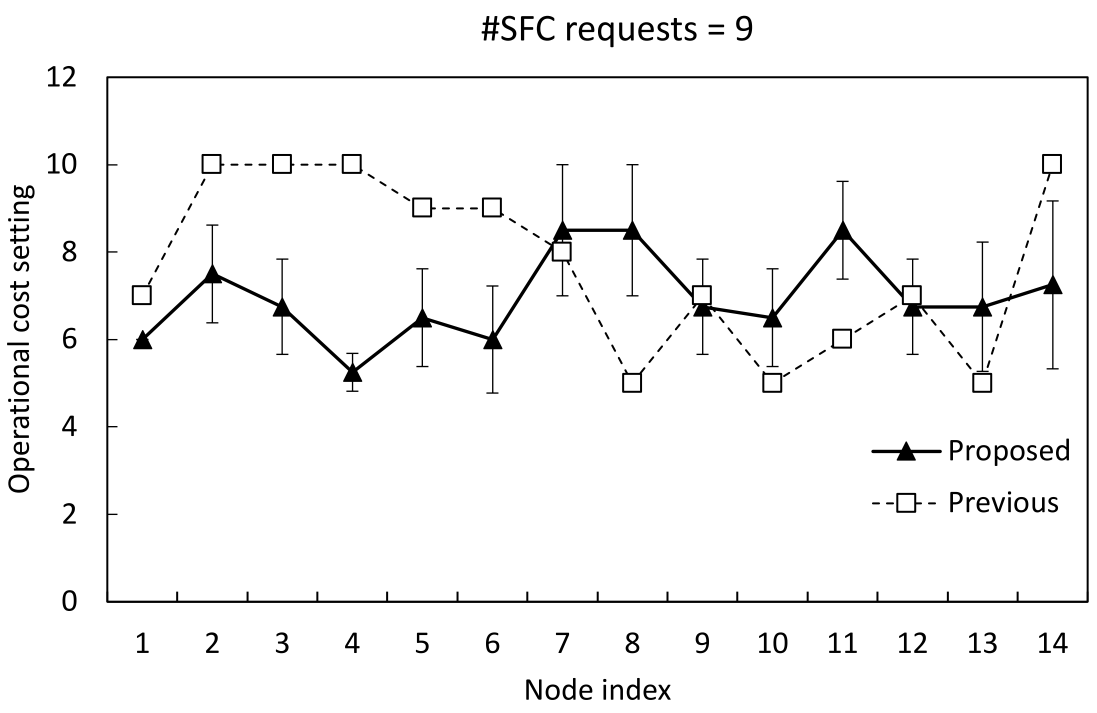

Figure B. 5 Operational cost setting at number of SFC requests 9.

Table B. 5 Data of operational cost settings at number of SFC requests 9.

	Proposed						Previous
Node index	VNF1	VNF2	VNF3	VNF4	Mean	Deviation	All VNFs
1	6	6	6	6	6	0	7
2	6	8	9	7	7.5	1.118034	10
3	7	8	7	5	6.75	1.089725	10
4	5	5	6	5	5.25	0.433013	10
5	7	5	6	8	6.5	1.118034	9
6	5	8	6	5	6	1.224745	9
7	9	6	9	10	8.5	1.5	8
8	6	9	9	10	8.5	1.5	5
9	8	7	7	5	6.75	1.089725	7
10	5	7	8	6	6.5	1.118034	5
11	8	10	9	7	8.5	1.118034	6
12	7	8	7	5	6.75	1.089725	7
13	7	6	9	5	6.75	1.47902	5
14	8	10	6	5	7.25	1.920286	10

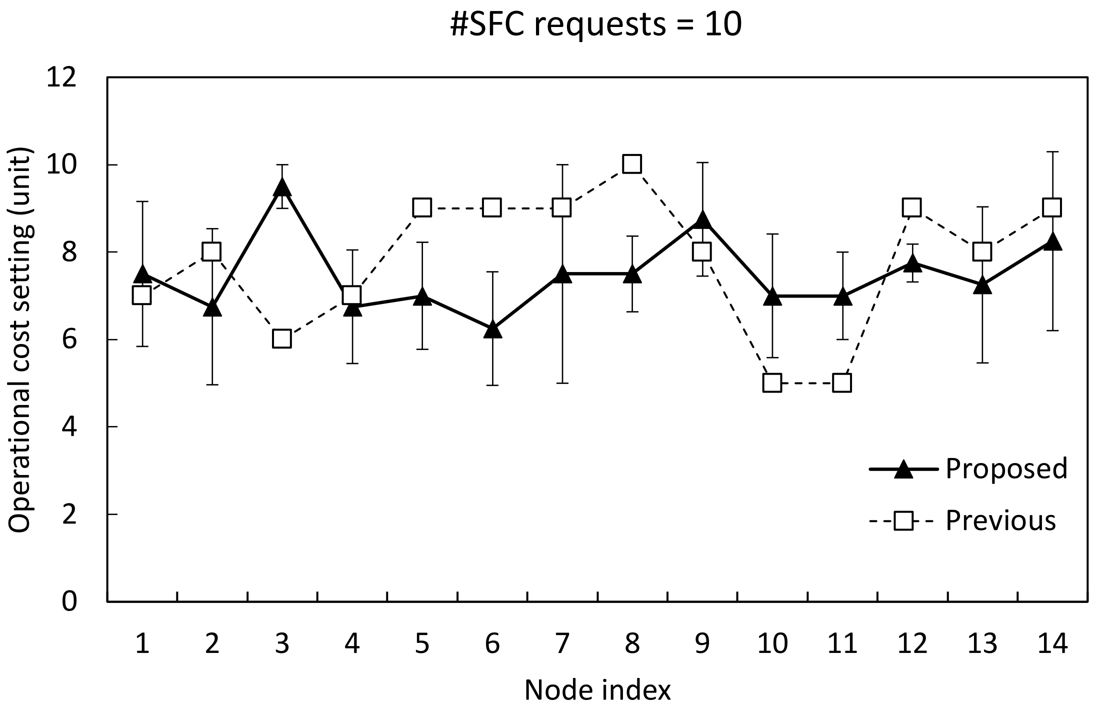

Figure B. 6 Operational cost setting at number of SFC requests 10.

Table B. 6 Data of operational cost settings at number of SFC requests 10.

	Proposed						**Previous**
Node index	VNF1	VNF2	VNF3	VNF4	Mean	Deviation	All VNFs
1	9	5	9	7	7.5	1.658312	7
2	5	9	8	5	6.75	1.785357	8
3	10	9	10	9	9.5	0.5	6
4	5	8	6	8	6.75	1.299038	7
5	8	5	7	8	7	1.224745	9
6	8	5	5	7	6.25	1.299038	9
7	10	10	5	5	7.5	2.5	9
8	7	7	9	7	7.5	0.866025	10
9	7	8	10	10	8.75	1.299038	8
10	5	9	7	7	7	1.414214	5
11	8	6	6	8	7	1	5
12	8	7	8	8	7.75	0.433013	9
13	7	7	5	10	7.25	1.785357	8
14	8	10	5	10	8.25	2.046338	9

List of Abbreviations

5G	the Fifth Generation (5G) Mobile Communication
AR	Argument Reality
ARPANET	The Advanced Research Projects Agency Network
CAPEX	Capital Expenditure
CPE	Customer Premise Equipment
CPU	Central Processing Unit
DoS	Denial of Service
DP	Dynamic Programming
DPI	Deep Packet Inspector
eMBB	Enhanced Mobile Broadband
ETSI	European Telecommunications Standards Institute
IDS	Intrusion Detection System
ILP	Integer Linear Programming
IoT	Internet of Things
IPS	Intrusion Prevention System
ISP	Internet Service Provider
ITU	International Telecommunication Union
JPN	Japan Photonic Network
LB	Load Balancer
LSTM	Long Short-term Memory Model
MANO	Management and Orchetstration
MEC	Multi-access Edge Computing
MILP	Mixed Integer Linear Programming
MIP	Mixed Integer Programming
MIPS	Millions of Instructions Per Second
mMTC	Massive Machine Type Communication
NAT	Network Address Translation

NF	Network Functions
NFV	Network Function Virtualization
NFVI	Network function Virtualization Infrastructure
NSFNET	National Science Foundation Network
OPEX	Operational Expenditure
QoE	Quality of Experience
QoS	Quality of Services
SDN	Software-Defined Networking
SFC	Service Function Chain
SLA	Service Level Agreement
SSL	Secure Sockets Layer
URLLC	Ultra-Reliable and Low Latency
VNE	Virtual Network Embedding
VNF	Virtual Network Function
WAN	Wide Area Network

Acknowledgments

I am honored to acknowledge and express my deeply appreciation to people who supported me to accomplish this great achievement.

First, and foremost, I would like to express my deep and sincere gratitude my supervisor, Associate Professor Ved Prasad Kafle, his constant guidance, instructions and encouragement during my Ph.D study years in the University of Electro-Communications and at National Institute of Information and Communications Technology (NICT). He is a dedicated, professional and caring supervisor. He always supports me whenever I needed his help. Thank you very much.

I would like to acknowledge Professor Toshihiko Kato and Associate Professor Satoshi Ohzahata for their great help in giving me instructions in all aspects of my research.

I would like to express my sincere appreciation and acknowledgement to Professor Hiroshi Yoshiura, Associate Professor Mitsugu Iwamoto, and Associate Professor Celimuge Wu. Their questions, comments and suggestions during my research presentations were very helpful in improving my research and completing this thesis.

Special thanks to JT Asia Scholarship Foundation for their financial support during this research. Many thanks go to my laboratory colleagues, especially Timothy Girry Kale, Jin Lin, and Phetlasy Sornxayya for their advice and kindness.

I would like to specially thank Zhiqi Zhang who is always supporting me.

Finally, I would like to thank my parents for their love, support and encouragement all the time.

Author Biography

Yansen XU received a B.E. degree from Department of Electronic Information Engineering in Jilin University, China, and an M.S. degree from Graduate School of Information Network System, Department of Information Network Systems, the University of Electro-Communications, Japan, in 2011 and 2016, respectively. Since 2016, he is working towards his Ph. D degree at The University of Electro-Communications, Japan.

He had shortly studied at Shandong University and Chinese Academy of Sciences (CAS) as a visiting student in 2008 and 2014, respectively. He worked in ZTE Cooperation in 2011 for the development of wireless network systems.

His research interests include network function virtualization (NFV), software-defined networking (SDN), and information centric networking (ICN). He is a student member of IEICE and IEEE.

List of Publications

Journal Paper

[1] Y. Xu and V. P. Kafle, “A Mathematical Model and Dynamic Programming based Scheme for Service Function Chain Placement in NFV,” in IEICE TRANSACTIONS on Information and Systems, Special Section on the Architectures, Protocols, and Applications for the Future Internet, Volume E102.D, Issue 5, pp. 942-951, May 2019.

(Related to the content of Chapter 3 and 4)

[2] Y. Xu and V. P. Kafle, “An Availability-Enhanced Service Function Chain Placement Scheme in Network Function Virtualization,” Journal of Sensor and Actuator Networks, vol. 8, no. 2:34, pp. 1-16, Jun. 2019.

(Related to the content of Chapter 5)

International Conference

[1] Y. Xu and V. P. Kafle, “A Delay-Aware Service Function Chain Placement Scheme Based on Dynamic Programming,” 2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), Washington, DC, 25-27 Jun. 2018, pp. 110-111.

(Related to the content of Chapter 4)

[2] Y. Xu and V. P. Kafle, “Optimal Service Function Chain Placement Modeling for Minimizing Setup and Operation Cost,” 2018 IEEE 7th International Conference on Cloud Networking (CloudNet), Tokyo, 22-24 Oct. 2018, pp. 1-3.

(Related to the content of Chapter 3)

[3] Y. Xu and V. P. Kafle, “Reliable service function chain provisioning in software-defined networking,” 2017 13th International Conference on Network and Service Management (CNSM), ManSDNNFV Workshop, Tokyo, 26-30 Nov. 2017, pp. 1-4.

(Related to the content of Chapter 5)

IEICE General Conference

[1] Y. Xu and V. P. Kafle, “Reinforcement Learning based Service Function Chain Scaling in Network Function Virtualization,” IEICE General Conference, 2019.

[2] Y. Xu and V. P. Kafle, “A Dynamic Programming Based Scheme for Service Function Chain Placement,” IEICE General Conference, 2018.

[3] Y. Xu and V. P. Kafle, “A Mathematical Model for Virtual Network Embedding in Varying Bandwidth Environment,” IEICE General Conference, 2017.

IEICE Society Conference

[1] Y. Xu and V. P. Kafle, “Minimizing Setup and Operation Cost for Service Function Chain Placement,” IEICE Society Conference, 2018.

[2] Y. Xu and V. P. Kafle, “Node Degree Based Reliable Service Function Chain Provisioning in Software-Defined Networking,” IEICE Society Conference, 2017.