THÈSE

pour obtenir le grade de

Docteur de l'Université Sorbonne Paris Nord

Discipline : "Ingénierie Informatique"

présentée et soutenue publiquement par

IKHELEF Issam Abdeldjalil

le 19 Janvier 2024

Optimisation de placement et chaînage de fonctions réseaux selon le paradigme SDN/NFV

Directeur de thèse : **Pr. Ken CHEN**

Co-encadrant de thèse : **Dr. Mohand Yazid SAIDI**

JURY

Samia BOUZEFRANE,	Professeur, CNAM (Paris)	Présidente du Jury
Fen ZHOU,	Maître de conférences HDR, Université d'Avignon	Rapporteur
Yassine HADJADJ AOUL,	Professeur, Université de Rennes	Rapporteur
Nadjib AIT SAADI,	Professeur, Université Paris-Saclay	Examinateur
Saadi BOUDJIT,	Maître de conférences HDR, USPN	Examinateur
Ken CHEN,	Professeur, USPN	Directeur
Mohand Yazid SAIDI,	Maître de conférences, USPN	Co-encadrant

Thesis

Submitted for the degree of Doctor of Philosophy of

Sorbonne Paris Nord University

Specialization : "Computer Engineering"

presented and defended by

IKHELEF Issam Abdeldjalil

January 19, 2024

Optimization of placement and chaining of network functions according to the SDN/NFV paradigm

Supervisor : **Pr. Ken CHEN**

co-supervisor : **Dr. Mohand Yazid SAIDI**

Committee

Samia BOUZEFRANE,	Professor, CNAM (Paris)	Chairwoman
Fen ZHOU,	Associate professor, University of Avignon	Reviewer
Yassine HADJADJ-AOUL,	Professor, University of Rennes	Reviewer
Nadjib AIT SAADI,	Professor, University of Paris-Saclay	Examiner
Saadi BOUDJIT,	Associate professor, University of Sorbonne Paris Nord	Examiner
Ken CHEN,	Professor, University of Sorbonne Paris Nord	Supervisor
Mohand Yazid SAIDI,	Associate professor, University of Sorbonne Paris Nord	Co-Supervisor

L2TI – EA3043
École doctorale Galilée – Université Sorbonne Paris Nord
99, avenue Jean-Baptiste Clément

93430 Villetaneuse.

Résumé

Actuellement, le secteur des réseaux connaît un intérêt croissant pour deux technologies majeures : la virtualisation des fonctions réseau (NFV) et les réseaux définis par logiciel (SDN). NFV et SDN offrent des approches novatrices en matière de gestion des réseaux, en apportant une grande flexibilité et programmabilité.

Le réseau défini par logiciel (SDN) révolutionne la gestion des réseaux en dissociant le plan de contrôle du plan de données. Cette séparation de plans permet un contrôle centralisé et programmable, remplaçant le contrôle traditionnel basé sur le matériel physique par un contrôle basé sur des logiciels. Le SDN peut créer des réseaux virtuels ou gérer efficacement des réseaux matériels existants à l'aide de logiciels offrant ainsi une grande flexibilité et adaptabilité.

La virtualisation des fonctions réseau (NFV), quant à elle, vise à réduire les coûts et à accélérer le déploiement des services pour les différents opérateurs de réseau. Cela est réalisé en dissociant les fonctions réseau (NF) telles que les pare-feu ou le chiffrement du matériel dédié et en les virtualisant sur des serveurs standards. La NFV permet de regrouper plusieurs fonctions sur un seul serveur physique, ce qui permet d'économiser les coûts et réduire les interventions sur site. Au lieu de déployer de nouveaux matériels, les fournisseurs de services (opérateurs) peuvent activer des machines virtuelles (VM) pour exécuter des fonctions réseau spécifiques (VNFs). Par exemple, le chiffrement du réseau peut être réalisé en déployant un logiciel de chiffrement sur un serveur standard ou un commutateur existant, ce qui élimine la nécessité de déployer de nouveaux matériels.

Le placement et le chainage des VNFs posent un problème complexe connu pour être NP-difficile. Il s'agit de déterminer la séquence optimale de chaînes de fonctions de service (SFC) tout en dirigeant efficacement le trafic à travers les différents VNFs. L'objectif principal de la thèse est de développer un algorithme qui optimise le coût de placement et chainage des VNFs dans le réseau, en tenant compte de diverses contraintes telles que l'ordre des VNFs, la capacité de traitement et la capacité de bande passante.

L'algorithme vise à minimiser les coûts d'allocation globaux tout en garantissant un routage efficace des flux à travers les VNFs.

En exploitant les capacités du NFV et du SDN, la thèse vise à contribuer à l'avancement de la virtualisation des réseaux et des réseaux définis par logiciel (SDN). Les algorithmes proposés permettront aux opérateurs de réseau de prendre des décisions éclairées concernant le placement des VNFs, ce qui se traduira par une amélioration de l'efficacité du réseau, une réduction des coûts et une amélioration de la fourniture de services. La thèse vise à permettre aux opérateurs de réseau de gérer efficacement leurs réseaux, d'allouer les ressources de manière optimale et de fournir des services avec une agilité et une rentabilité accrues.

Mots-clés : Virtualisation des Fonctions Réseau, Réseau Défini par Logiciel, Chaine des Fonctions de Service, Allocation des Ressources, Optimisation, Fonctions Réseau Virtuelles, Déploiement de Services, Optimisation des Coûts.

Abstract

The current landscape of networking has witnessed a growing interest in two significant technologies: Network Function Virtualization (NFV) and Software-Defined Networking (SDN). These technologies offer novel approaches to network management, bringing flexibility and programmability to the forefront.

Software-Defined Networking (SDN) revolutionizes the network management by decoupling the control plane from the data plane. This separation of plans allows for centralized control and programmability, replacing traditional hardware-centric control with software-based control. SDN can create virtual networks or efficiently control traditional hardware networks through software, offering enhanced flexibility and adaptability.

Network Function Virtualization (NFV), on the other hand, focuses on cost reduction and service deployment acceleration for network operators. It achieves this by decoupling network functions (NF), such as firewall or encryption, from dedicated hardware and virtualizing them on standard servers. NFV enables the consolidation of multiple functions onto a single physical server, resulting in cost savings and minimized field interventions. Adding new network functions becomes more streamlined, requiring the activation of virtual machines (VM) rather than deploying additional hardware across the entire network.

The research thesis aims to tackle the NP-hard problem of optimal VNFs placement and chaining within a NFV network. This problem involves determining the most efficient sequence of service function chains (SFC) while directing flows through the VNFs. The primary objective is to develop an algorithm that achieves a cost optimal placement and chaining of VNFs, considering various constraints such as chaining order, processing capacity, and bandwidth capacity. The algorithm seeks to minimize the overall allocation costs while ensuring effective flow routing through the VNFs.

By leveraging the capabilities of NFV and SDN, the thesis aims to contribute to the advancement of network virtualization and software-defined networking. The proposed algorithms will empower network operators to make informed decisions regarding VNFs placement, leading to improved network efficiency, cost reduction, and enhanced service provisioning. Ultimately, the thesis aims to enable network operators to effectively manage their networks, allocate resources optimally and deliver services with increased agility and cost-effectiveness.

Keywords: Network Function Virtualization, Software-Defined Networking, Service Function Chaining, Resource Allocation, Optimization, Virtual Network Functions, Service Deployment.

Acknowledgments

It will be very difficult for me to thank everyone because it is thanks to the help of many people that I was able to complete this thesis.
I would like to express my gratitude to my supervisors Pr. Ken CHEN and Dr. Mohand Yazid SAIDI, for their supports and for the continuous flow of intellectual and moral assistance. I would like to thank them for the encouragement and the valuable feedback to improve my research quality. Pr. Yassine HADJADJ AOUL and Dr. Fen ZHOU did me the honor of being reviewers for my thesis. Their remarks allowed me to consider my work under another angle. For all this I thank them.

I would like to express my gratitude and my thanks to Mme Isabelle BARBOTIN and Dr Saadi BOUDJIT for their support. I will always be grateful. I am grateful to Pr Samia BOUZEFRANE and Pr Nadjib AIT SAADI for agreeing to review my thesis and attend my thesis defense. I would also like to thank Dr John Warwicker Alasdair, researcher of KIT, who hosted me for a month in his laboratory in Germany. It is thanks to him that I was able to happily reconcile theoretical and applied research during this thesis. Thanks to the University of Sorbonne Paris Nord for awarding me a Dissertation Completion Fellowship, providing me with the financial means to complete this project. Finally, I would like to thank everyone who contributed to this thesis, directly or indirectly. -

Contents

List of Tables

List of Figures

Abbreviations

SDN	***S***oftware Defined Networking
NFV	***N***etwork Function Virtualisation
NFV-I	***N***etwork Function Virtualisation Infrastucture
VNF	***V***irtual ***N***etwork Function
VNF-PC	***V***irtual ***N***etwork Function Placement Chaining
MANO	***M***anagement ***An***d Orchestration
SFC	Service Function Chain
API	Application Programming Interface
CPU	Central Pprocessing Unit
DC	Data Centers
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
POP	Points Of Presence
ILP	Integer Linear Programming
MILP	Mixed Integer Linear Programming
NAT	Network Address Translation
VM	Virtual Machine
QoS	Quality of Service
CAPEX	***C***apital Expenditure
OPEX	***O***perational Expenditure
ETSI	***E***uropean Telecommunications Standards Institute
WLAN	Wireless Local Area Network
NS	Network Service

VNF-FG	***V***irtualized **N**etwork **F**unctions **F**orwarding **G**raph
DMTF	***D***istributed **N**anagement **T**ask **F**orce

Chapter 1

Introduction

Walk in sandals

until wisdom gives you shoes.

Avicenna

Abstract

This chapter introduces the research topic of Virtual Network Function Placement and Chaining (VNF-PC) in the context of Network Function Virtualization (NFV) and Software-Defined Networking (SDN). It discusses the objective of optimizing resource allocation and minimizing costs while considering constraints. The contributions of the thesis are summarized. The thesis is organized into six chapters, each addressing different aspects of the VNF-PC problem and proposing innovative solutions.

Chapter content

1.1 Introduction

In this chapter, we highlight the main issues and motivations involved in network services according to the NFV/SDN paradigms. Subsequently, we will discuss the main characteristics of this emerging technology. Likewise, the chapter discusses the limitations of the technology and presents the background and challenges facing the technology. The objective of this synthesis is to give a concept and to help the understanding of our work presented in the following chapters.

Network function virtualization (NFV) and the software defined network (SDN) have become nowadays, interesting topics in the networking industry. The combination of these two new paradigms can overcome the limitations of traditional networks and clouds by improving their dynamic networking capabilities.

Nowadays, service Providers go beyond simply providing network connectivity (internet) to their clients. They also provide additional services and network functions such as network address translation (NAT), firewall, domain name service (DNS), etc. To deploy these network function, providers traditionally use proprietary hardware at the customer's premises.

This methods provides to the providers additional revenue, but deploying multiple proprietary hardware is expensive and makes the upgrades difficult because each time a new network function is added to a certain service, a team is required to move to the customer's premises in order to install the new dedicated hardware device. As a result, service providers have begun to explore ways to reduce costs and accelerate deployments through the Network Functions Virtualization (NFV) [1].

Network functions virtualization (NFV) involves virtualizing network services and functions currently made available by dedicated, proprietary hardware. NFV reduces proprietary hardware amount required to launch and exploit network services.

Exploring Software-Defined Network (SDN) technologies and the foundations of network virtualization, NFV helps IT professionals and providers to adapt a modern hybrid network architecture.

Its objective is to separate the network functions from the hardware equipment dedicated to them, such as routers, firewalls and load-balancers, to host the services they provide on virtual machines (VM). Several use cases of Network Function Virtualization are discussed in [2].

Software Defined Networking (SDN) is one of the most important architectures that allows pure software management of the network, which may require re-configurations or re-policing from time to time [3]. For this, the control plane implemented as standard in the hardware components or the control logic is abstracted from the hardware; in this context, we also speak of the hardware intelligence, which is nothing more than its specific operating software (firmware). In simple terms, the SDN concept corresponds to the separation of the infrastructure and its configuration.

The data plane, on the other hand, remains a part of the individual network devices (i.e. all the routers, switches and firewalls integrated into the network). However, its task with SDN is exclusively to transmit the packets, which is why it requires little computing power. This has the advantage, among other things, that the devices do not need elaborate firmware and are generally much cheaper than other network concepts [4].

The task field of the abstract control plane, which is responsible for the correct data traffic in the SDN architecture and therefore has to perform all the relevant analyses, is considerably more complex. However, detached from hardware and implemented in centralized software, it is highly programmable in a software-defined network and therefore much more flexible in terms of network administration than other architectures.

Knowing that providers will no longer need to purchase dedicated network equipment, this approach takes on its full value when building a service chain. Indeed, since it will be possible to increase the capacity of servers by software, providers will no longer need to over-equip their data centers, which will reduce both capital expenditure (capex) and operational expenditure (opex) [5].

For example, if the execution of an application on a VM requires more bandwidth, the administrator can move the corresponding VM to another physical server or make available on the initial server another virtual machine which will absorb part of load. This flexibility will allow providers to respond more agilely to changing business goals and changing network service needs [6].

1.2 Research Problem and Objectives

In NFV, a service is defined as a chain composed of Virtual Network Functions (VNFs) named Service Function Chain (SFC). The allocation process of resources from servers to services, called VNF Placement and chaining (VNF-PC), is one of the most difficult tasks in NFV. Dynamic nature of SFC arrivals and departures makes the problem of VNF-PC even more difficult [7].

For ease of understanding, we consider the example of Figure 1.1 where a network infrastructure (NFV-I) composed of 5 servers, Service provider (Source) and 2 Routers (Targets) is shown.

Each server of the NFV-I has different types of resources, each server with its unit cost and limited capacity. The NFV-I servers are connected with virtual links (mapped on physical paths) whose unit costs and capacities are negotiated and fixed for each couple of servers.

The bottom of Figure 1.1 shows an example of a Service Function Chain comprising Virtual Network Functions (VNFs) to be placed in the NFV-I. To provision the SFC, all its VNFs should be placed on the NFV-I servers with enough resources. The NFV-I

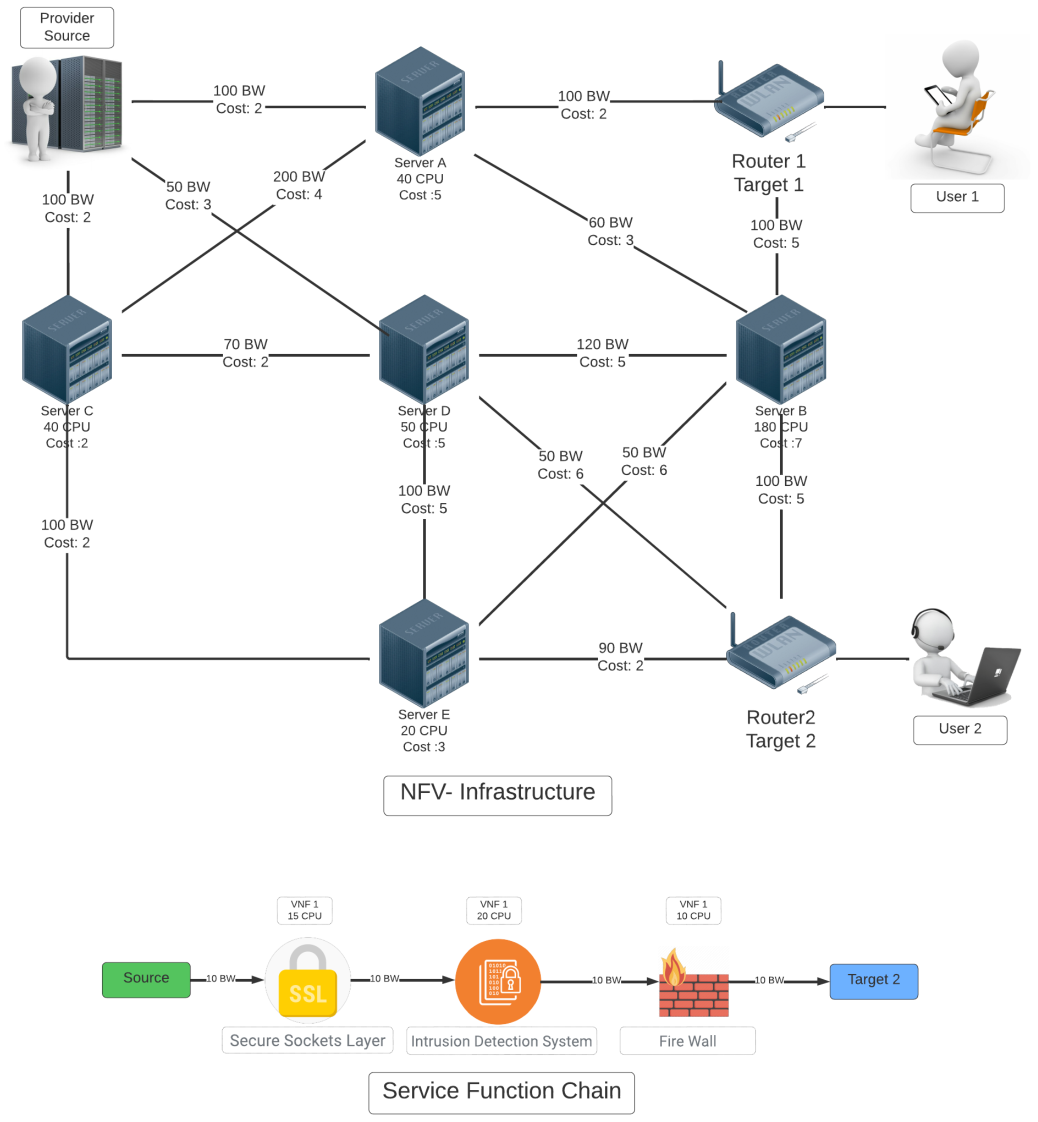


Figure 1.1: *An example of NFV-I topology.*

should also provide sufficient resources for the links that connect between the VNFs. For example, let us consider the SFC of Figure 1.1 composed of 3 VNFs. We assume that *VNF1* requires 15 CPUs, *VNF2* requires 20 CPUs and 10 CPUs for *VNF3*. The aim is to find the optimal placement and chaining of these three VNFs on the NFV-Infrastructure shown in Figure 1.1. The solution must ensure that the total SFC allocation cost is minimized while respecting the capacity constraints on the NFV-I components.. Therefore, *VNF1* and *VNF2* will be placed on *Server C* and *VNF3* will be placed on *Server E* as shown in Figure 1.2.

The problem of VNF placement and chaining is known to be NP-Hard [8]. It consists of placing a sequence of service function chains (SFC) while directing the flows of through the VNFs. The objective of this paper is to provide a K-multi constrained shortest path-based heuristic (K-MSPH) that addresses the generic problem of VNF-PC using nodes sharing capability enabled by the NFV paradigm.

1.2.1 Scope of the thesis

The purpose of this work is to solve the VNF placement and chaining problem in the context of Network Functions Virtualization (NFV), focusing on optimizing the overall

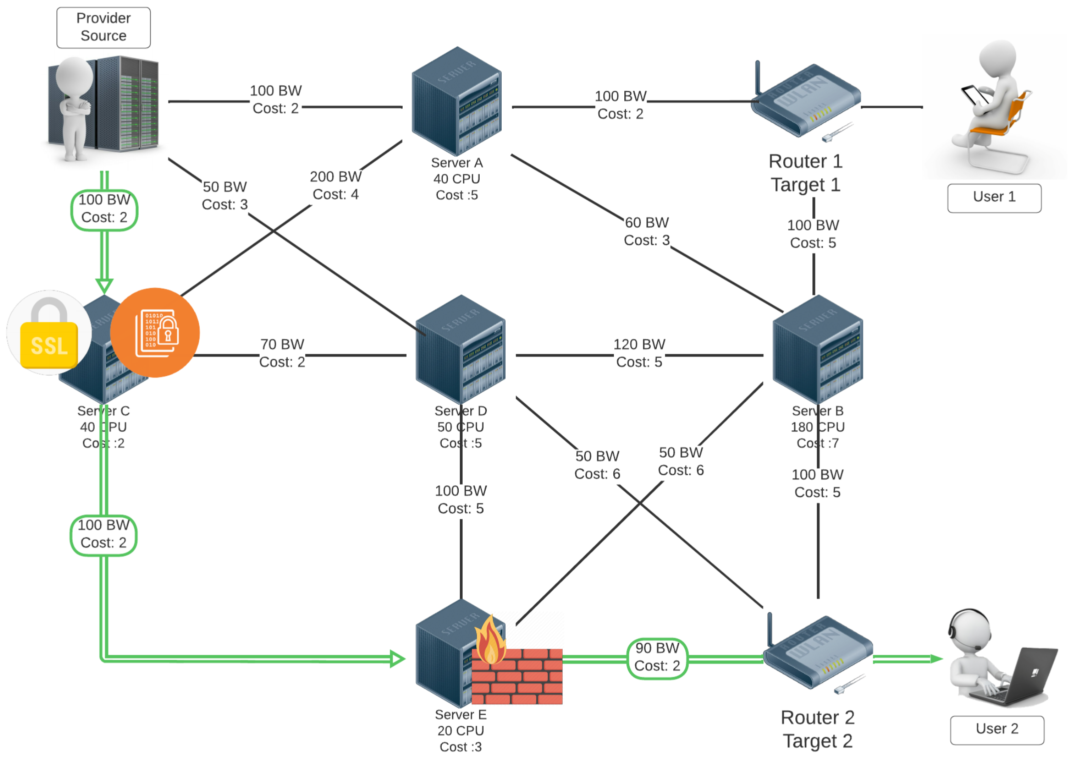


Figure 1.2: *Optimal placement of the SFC.*

allocation cost and reducing the rejection rate. The scope of this thesis is to develop efficient algorithms and mechanisms for placing and chaining virtual network functions (VNFs) over a virtualized infrastructure built on commodity equipment. The research aims to achieve the following goals:

i. SFC Placement: The primary focus is to develop efficient algorithms and methodologies for the optimal placement of VNFs within the virtualized infrastructure. This involves determining the most appropriate physical servers and network resources to host each VNF instance, taking into account factors such as resource availability and network connectivity.

ii. Cost Optimization: The main objective is to minimize the total allocation costs associated with the deployment of VNFs. This involves considering various cost factors, such as resource usage, network bandwidth, and infrastructure provision. The research will explore optimization techniques and algorithms to determine optimal placement and chaining of VNF instances that minimize overall costs while meeting service requirements.

iii. Reducing rejection rate: Another important aspect is to reduce rejection rates of service requests due to resource constraints. When demand for services exceeds available resources, service requests may be rejected, resulting in a degraded user experience. The research will focus on developing strategies to minimize rejection rates by dynamically managing the allocation of VNF instances.

iv. Scalability and Efficiency: As NFV environments increase in scale and complexity, it becomes essential to develop scalable and efficient algorithms for VNF placement and chaining. The research will explore techniques for managing large-scale deployments, dynamic resource allocation, and efficient management of VNF instances.

1.3 Research Contributions

This thesis makes several significant contributions to the field of resource allocation optimization in Network Function Virtualization (NFV) environments. The primary objective of the thesis is to optimize the placement and chaining of Virtual Network Functions (VNFs) within NFV infrastructures to achieve efficient resource utilization and cost minimization. One of the most important contributions is the introduction of an innovative approach using integer linear programming (ILP) models. The model takes into account factors such as computational capabilities and bandwidth requirements to optimize resource allocation. By formulating the problem as an ILP, this work proposes a novel framework for finding optimal solutions that minimize the costs associated with Service Function Chain (SFC) allocation. This approach brings new perspectives to existing research on VNF placement and chaining.

Additionally, the thesis explores a relaxed version of the VNF-PC problem in unloaded network scenarios. Leveraging the assumption of abundant resources available on both nodes and links, a shortest path algorithm is proposed. This algorithmic approach combines principles from shortest path algorithms and graph transformation techniques to address resource allocation and cost minimization objectives.
Furthermore, this thesis also addresses a specific variant of the VNF-PC problem in network scenarios with abundant link resources but limited node resources. To address this challenge, a genetic algorithm is proposed that combines the knapsack optimization principle with a genetic algorithm. By iteratively refining candidate solutions using genetic operators, the algorithm explores various potential configurations, ultimately minimizing the overall placement cost. The paper, published in LCN 2022 ([9]), provides a valuable tool for optimizing resource allocation in situations with abundant available resources.
Moreover, when extending the study of the ILP using Bender decomposition, the research led to the publication of a paper in APNOMS 2023 [10]. This paper delves into the sophisticated realm of combinatorial Benders decomposition, systematically partitioning the problem into a master problem and a linear sub-problem. It also explores the formulation of combinatorial cuts, pivotal elements that pinpoint infeasibilities within the sub-problem and propel the search for feasible solutions.
Moreover, the thesis tackles the full generic version of the VNF-PC problem, considering limited node and link resources. To address this challenge, a multi-constrained algorithm is adopted that takes into account multiple constraints such as computational capabilities and bandwidth requirements. This algorithmic approach, published in ICC 2023 [11], systematically explores potential solutions that satisfy all constraints while minimizing the overall placement cost. By considering the various constraints, the algorithm is able to achieve efficient resource allocation in NFV environments.

Moreover, building upon the work presented in the ICC paper [11], we have extended and refined our approach in the journal paper [12]. This new paper provides a more comprehensive exploration of the Constrained Shortest Path-base Heuristic in multi-Partite Graph (CSPH-PG). Our research in this extension demonstrates the scalability and efficiency of CSPH-PG, particularly in the context of large NFV deployments and scenarios with high traffic loads. It's important to note that this work is submitted to the Journal of Network and Computer Applications and is undergoing the peer review procedure.

Taken together, these contributions significantly advance the understanding and techniques for optimal resource allocation in NFV environments. By proposing novel frameworks and algorithmic approaches and solving different versions of the VNF-PC problem, this thesis provides valuable insights and tools to improve the performance, efficiency, and cost-effectiveness of network operations.

1.4 Thesis Organization

This thesis is organized into six main chapters. In addition to this chapter introducing the context, thesis objectives and contributions, the manuscript is organized as follows:

- **Chapter 2 :** This chapter begins by providing essential contextual information on the thesis topic, offering an introduction to two key concepts in modern networking: Network Function Virtualization (NFV) and Software-Defined Networking (SDN). NFV refers to the virtualization of network functions, such as firewalls, routers, and load balancers, decoupling them from dedicated hardware appliances and running them as software instances on standard servers. SDN, on the other hand, is an architectural approach that separates the network's control plane from the data plane, enabling centralized control and programmability of network resources.

  Furthermore, the chapter presents an overview of the current state of research in resource allocation problems within the context of NFV. Resource allocation is a critical aspect of NFV deployments as it involves efficiently mapping virtual network functions onto physical resources, such as servers, storage, and network links. The chapter explores various resource allocation strategies, including heuristic algorithms, optimization models, and machine learning techniques, highlighting their strengths and limitations.

Additionally, the chapter sheds light on the networking challenges that arise in cloud environments, where NFV and SDN are often deployed. Cloud environments are characterized by a dynamic and multi-tenant nature, with diverse workloads and varying resource demands. This introduces complexities in terms of scalability, performance, security, and resource management. The chapter discusses these challenges and their implications on resource allocation and overall network performance in cloud environments.

By providing this comprehensive overview, the chapter establishes a solid foundation for the subsequent discussions on the specific variation of the VNF-PC problem, focusing on networks with limited node resources but abundant link resources. Understanding the context of NFV, SDN, resource allocation challenges, and cloud networking sets the stage for addressing the targeted problem effectively and proposing innovative solutions.

- **Chapter 3 :** This chapter focuses on tackling the issue of VNF (Virtual Network Function) placement and chaining, with the primary goal of minimizing SFC (Service Function Chaining) allocation costs and reducing the rejection of SFC requests in the face of increasing demand and traffic load. To address this challenge, an ILP (Integer Linear Programming) model is proposed, offering a novel approach to optimize the placement and chaining of VNFs.

  The ILP model aims to optimize the allocation of resources by considering factors such as computational capabilities, bandwidth requirement. By formulating the problem as an ILP, it becomes possible to find an optimal solution that minimizes the costs associated with SFC allocation and ensures efficient utilization of available resources.

  This proposed ILP model presents a fresh perspective on addressing the VNF placement and chaining problem, offering a potential solution to the challenges posed by rising demand and traffic load. By effectively optimizing resource allocation, it becomes feasible to mitigate the rejection of SFC requests and enhance the overall performance of the network.

  Expanding our arsenal of optimization strategies, we delve into the realm of combinatorial Benders decomposition, a sophisticated approach that resonates with the intricacies of the VNF-PC problem. This technique involves the systematic partitioning of the problem into a master problem and a linear sub-problem. Additionally, we delve into the formulation of combinatorial cuts, pivotal elements that pinpoint infeasibilities within the sub-problem and propel the search for feasible solutions.

- **Chapter 4 :** This chapter extends the study by considering the VNF-PC problem in an unloaded network. An unloaded network refers to a network environment where there is an abundance of resources available on nodes and links, allowing for the provisioning of SFCs (Service Function Chains) using any available NFV-I (NFV Infrastructure) components. In such scenarios, traditional approaches may not effectively leverage the available resources, leading to sub-optimal solutions. In this particular version of the VNF-PC problem, the assumption is made that link resources have large and sufficient capacities. The main objective is to minimize the overall placement cost, which encompasses the costs associated with utilizing node and link resources for the provisioning of SFCs.

  The inclusion of the VNF-PC problem in an unloaded network in this chapter expands the scope of the research, addressing a practical scenario where network resources may not be fully utilized.

- **Chapter 5 :** In this chapter, we focus on addressing a specific variation of the VNF-PC (Virtual Network Function Placement and Chaining) problem, which targets a network with limited node resources but abundant link resources.

  To address this particular version of the VNF-PC problem, we propose a knapsack-based genetic algorithm. This algorithmic approach combines knapsack optimization principles with genetic algorithms to optimize resource allocation and minimize costs.

  The knapsack-based genetic algorithm is inspired from the classical knapsack problem (KP), where items with different weights and values are selected to maximize the total value while respecting a weight constraint. In the context of the VNF-PC problem, we adapt these principles to determine the optimal selection and placement of VNFs considering the limited resources of the network nodes.

  By leveraging a genetic algorithm, our approach incorporates genetic operators such as selection, crossover, and mutation. These operators iteratively refine the population of candidate solutions, allowing us to explore a diverse range of potential configurations and identify promising placements that minimize the overall cost.

Integrating the knapsack-based genetic algorithm into the problem-solving process enhances the efficiency and effectiveness of resource allocation and cost minimization objectives. This approach provides a systematic and evolutionary framework for finding optimized solutions, particularly in networks with limited node resources but abundant link resources.

Through the utilization of this novel algorithmic approach, this section makes a valuable contribution to the field of VNF-PC problem solving in networks with limited node resources and abundant link resources.

- **Chapter 6 :** This chapter addresses the full generic version of the VNF-PC (Virtual Network Function Placement and Chaining) problem, encompassing the placement and chaining of VNFs in a network environment. Our objective is to optimize resource allocation and minimize the overall placement cost, considering various constraints and requirements.

  We consider a scenario where both the node and link resources within the network are limited. Our goal is to efficiently allocate these resources to minimize the placement and chaining cost, which includes the costs associated with utilizing nodes and links for SFC (Service Function Chain) provisioning.

  To address this challenge, we employ a multi-constrained algorithm (KMSPH) tailored for the VNF-PC problem. This algorithmic approach takes into account multiple constraints, such as computational capabilities and bandwidth requirements to optimize the placement and chaining of VNFs.

  By leveraging the multi-constraint algorithm, we can systematically explore potential solutions that satisfy all constraints while minimizing the overall placement cost. This allows for efficient resource allocation, given the abundant resources available within the network.

  By addressing the full generic version of the VNF-PC problem and using the multi-constraint algorithm, we contribute to the field of resource allocation optimization in NFV environments.

- **Chapter 7 :** This chapter presents a multifaceted contribution to the domain of

Virtual Network Function (VNF) placement and chaining. Firstly, it conducts an extensive examination of the VNF placement and chaining problem within an unloaded network (over-resourced network), offering a comprehensive insight into its complexities.

Secondly, it introduces the constrained shortest paths algorithm, a pivotal tool for tackling the VNF-PC problem. This algorithm optimizes the allocation cost of Service Function Chains (SFCs) provisioning. Moreover, this chapter serves as an extension and refinement of the previous chapter 6 approach, offering a more comprehensive exploration of the Constrained Shortest Path-base Heuristic in multi-Partite Graph (CSPH-PG).

It addresses the challenges presented by various NFV-Infrastructure (NFV-I) sizes and diverse traffic demands. Highlighting its scalability and efficiency, the chapter emphasizes the effectiveness of the CSPH-PG heuristic for larger NFV-Is and extensive SFC scenarios, making it a suitable choice for managing high traffic loads.

- **Chapter 8 :** The concluding section of this thesis summarizes the main findings and contributions while also highlighting potential future research directions and improvements to our work.

Chapter 2

State of the Art

The thinkers of the past believed that the mind's discovery of the truth is not a strange thing, but the strange thing is its inability to discover it.

Ibn Khaldun

Abstract

This chapter discusses current state of the art related to Network Functions Virtualization (NFV) and Software Defined Network (SDN). It also present a taxonomy of existing research that addresses the challenge of VNF placement and chaining.

Chapter content

2.1 Introduction

In this chapter, we begin by introducing the concept of Network Function Virtualization (NFV) and provide an overview of the NFV reference architecture, including its various components. We then delve into the topic of Service Function Chaining (SFC), discussing its significance in the context of NFV and how it enables the sequential routing of network functions.

Additionally, we explore Software-Defined Networking (SDN) and its architectural framework, highlighting its role in increasing network programmability and flexibility. To provide a comprehensive understanding of the NFV environment, we present a taxonomy of existing studies that address VNF placement and chaining challenges. The purpose of this literature review is to provide readers with the knowledge and insight necessary to understand the architecture and deployment of service function chains within NFV infrastructures.

2.2 Background of NFV

2.2.1 What is NFV ?

NFV (Network Functions Virtualization) is an architectural concept and framework that aims to virtualize and consolidate network functions onto standard hardware infrastructure. It is an evolution in the telecommunications industry that seeks to replace traditional, specialized hardware appliances with software-based virtualized network functions (VNFs) running on commodity servers, storage, and switches (Figure 2.1 illustrates the NFV principle) [13].

In NFV, network functions such as routing, firewalling, load balancing, and intrusion detection are implemented as software instances that can be dynamically deployed, scaled, and managed in a cloud-like environment. By virtualizing these functions, NFV offers flexibility, scalability, and cost efficiency by leveraging standard IT hardware and virtualization technologies [14].

Key aspects and benefits of NFV include:

i. **Virtualization**: NFV leverages server virtualization technologies, such as hypervisors, to run multiple VNFs on shared hardware, enabling efficient resource utilization and consolidation.

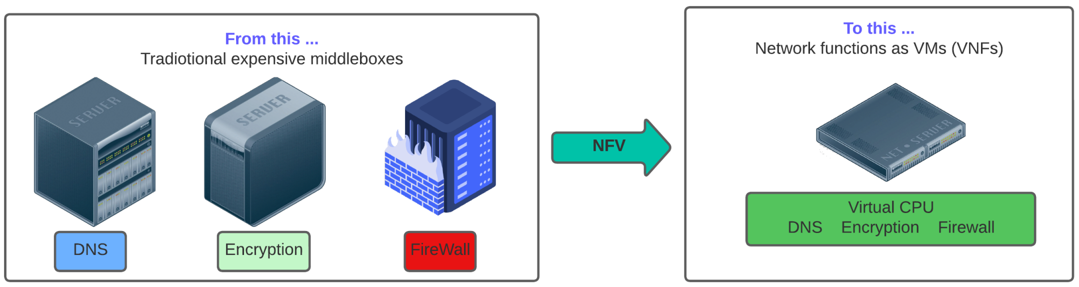


Figure 2.1: *Network Function Virtualization (NFV).*

ii. **Agility and Scalability**: VNFs can be quickly deployed, scaled up or down, and moved to different hardware resources as network demands change. This improves agility in deploying and scaling services.

iii. **Cost Efficiency**: By using off-the-shelf hardware, NFV reduces the capital and operating costs associated with dedicated network devices. It also allows service providers to use hardware resources more efficiently.

iv. **Service Innovation and Time-to-Market**: NFV allows service providers to introduce new services more quickly by leveraging software-based VNFs and automated orchestration, enabling faster time-to-market.

v. **Network Optimization and Management**: NFV provides centralized management and orchestration of network functions, making it easier to monitor, configure and manage network resources.

NFV is standardized and promoted by organizations such as the European Telecommunications Standards Institute (ETSI). These standards define the architectural principles, interfaces, and management frameworks to ensure interoperability and compatibility across different NFV deployments [13].

Overall, NFV represents a shift from dedicated, proprietary hardware appliances to flexible, scalable, software-driven network architectures. It enables service providers to innovate, optimize their networks and deliver services more efficiently and dynamically.

2.2.2 NFV framework

Figure 2.2 provides a high-level overview of the NFV framework, which encompasses three key domains of operation:

- **NFV Infrastructure (NFVI):** The NFVI comprises a collection of hardware and

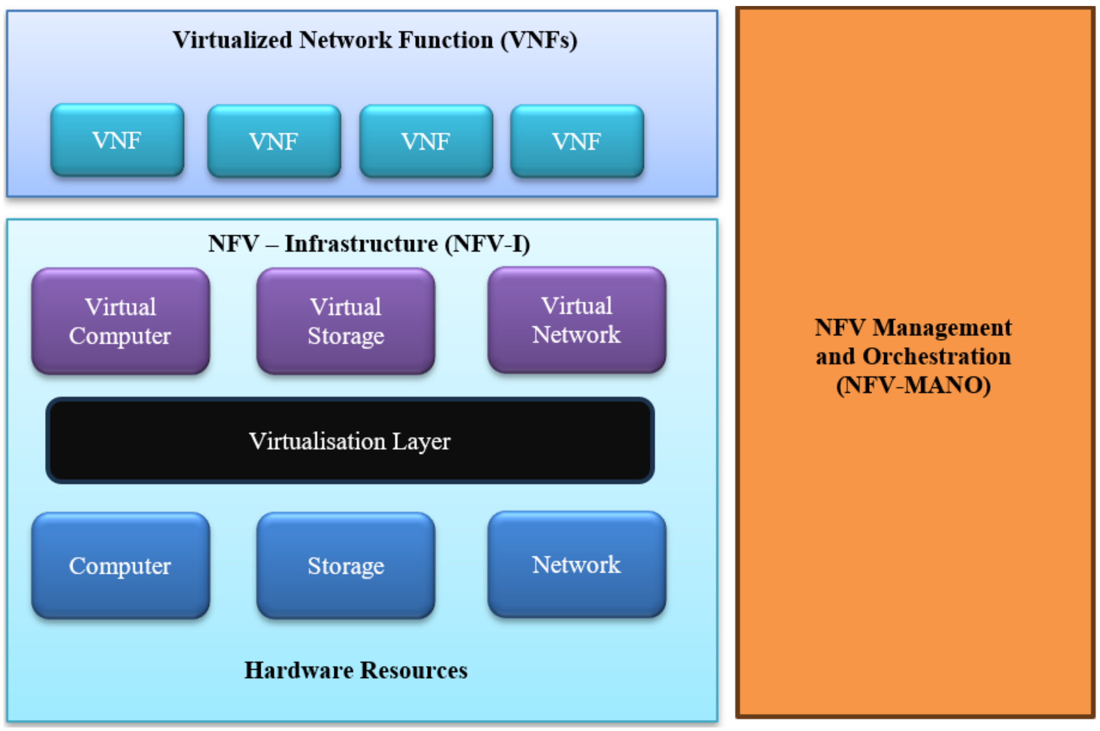


Figure 2.2: *High-level view of the NFV framework.*

software components that constitute the underlying environment for deploying, managing, and executing NFV services. It encompasses various elements, including hardware resources (such as RAM, CPU), storage hardware (e.g., disk storage, Network Attached Storage), and network hardware (such as switches and routers). Additionally, a virtualization layer is employed to abstract the underlying hardware resources and facilitate the support of multiple Virtual Network Functions (VNFs) by effectively sharing resources. Various open-source and proprietary solutions, such as VMWare, Xen, and KVM, are available for implementing this virtualization layer. Furthermore, the virtual infrastructure component encompasses virtual compute (Virtual Machines or VMs), virtual storage, and virtual links [15].

- **Virtualized Network Functions (VNFs):** VNFs represent the virtualized instances of distinct network functions, implemented as software, that operate within the NFVI. These VNFs encapsulate the functionalities traditionally performed by dedicated network hardware appliances, such as routing, firewalling, load balancing, and intrusion detection. By virtualizing these functions, NFV introduces greater flexibility, scalability, and cost efficiency by leveraging standard IT hardware and virtualization technologies [14].

- **NFV Management and Orchestration (MANO):** MANO involves the deployment, operation, and lifecycle management of both the physical and software resources necessary to support infrastructure virtualization. It also encompasses the management of VNF lifecycles. Within the NFV framework, MANO plays a crucial role in managing the overall virtualization-specific tasks and ensuring the effective orchestration of resources and services [13].

This high-level representation of the NFV framework provides an overview of its underlying components and their interrelationships. NFVI, VNF, and MANO collectively enable the virtualization and efficient management of network functions, enabling improved network flexibility, scalability, and cost-effectiveness.

2.2.3 NFV principles

Three fundamental principles in NFV contribute to the creation of practical Network Services (NS) [16], [17]:

i. **Service Chaining:** NS requires the execution of a sequence of network functions, known as Service Function Chaining (SFC) or Virtualized Network Functions Forwarding Graph (VNF-FG). This involves establishing specific connectivity and sequencing between VNFs. The service chaining ensures that network traffic flows through the required set of network functions in a predefined order.

ii. **VNF Embedding:** VNF embedding refers to the process of deploying VNFs on physical servers to form SFCs. It encompasses both node embedding, which involves mapping VNFs to the appropriate physical server resources, and link embedding, which establishes connectivity between embedded VNFs. The orchestrator, responsible for managing and orchestrating software resources and virtualized hardware infrastructure, plays a crucial role in the embedding stage of VNF.

iii. **Distributed Architecture:** A VNF may consist of multiple VNF components (VNFCs), each responsible for implementing a subset of the overall functionality. These VNFCs can be deployed as separate instances, providing scalability and redundancy. In a distributed architecture, VNFC instances can be deployed across multiple hosts, enabling load balancing, fault tolerance, and efficient resource utilization.

By adhering to these principles, NFV enables the creation of flexible and scalable network services. Service chaining ensures proper sequencing and connectivity between VNFs, while VNF integration ensures their efficient deployment on physical servers. The distributed architecture enables the deployment of VNFC instances across multiple hosts, promoting scalability and resiliency in the delivery of network services.

2.2.4 Service Function Chaining

An essential component of network architecture is the Service Function Chain (SFC), which represents an ordered or partially ordered set of Virtualized Network Functions (VNFs)[18]. The ordering constraints within an SFC dictate how packets, frames, or flows should be processed after undergoing classification. It is worth noting that the SFC architecture permits the replication of SFCs across multiple branches and allows flexibility in the order of applying service functions. The term "service chain" is often used interchangeably with "service function chain" [19].

The objective of placement and chaining algorithms is to determine the optimal placement of specific VNFs within the network. Two fundamental types of chains can be distinguished: linear and bifurcated. In the case of linear chains, a VNF may be shared across multiple SFC requests. Conversely, in bifurcated chains, the same VNF can be deployed on different Physical Nodes (PNs) to handle diverse flows. This flexibility in VNF deployment enables the creation of various SFCs tailored to specific applications and flow requirements.

2.3 Background of SDN

2.3.1 What is SDN ?

SDN, which stands for Software-Defined Networking, is an architectural approach that aims to separate the control plane and data plane functionalities in network devices. In traditional networks, switches and routers perform both control plane functions (making forwarding decisions) and data plane functions (actual traffic forwarding) [20]. In SDN, the control plane is abstracted and centralized in a dedicated software entity known as the SDN controller (see figure 2.3). The SDN controller serves as the central management point of the network, monitoring and controlling the behavior of network devices.

By separating the date plane from the control plane, SDN allows more flexibility and programmability in network management. It allows network administrators to dynamically control and configure the network, making it easier to adapt to changing traffic patterns and optimize network performance.

2.3.2 SDN framework

SDN, or Software-Defined Networking, is an innovative networking approach that revolves around several core principles [21]:

i. Centralized Control: SDN introduces a centralized control point called the SDN controller. This controller offers administrators a unified view and control of the entire network, enabling efficient management and configuration of network devices from a single location.

ii. Programmability: SDN empowers network administrators to programmatically control and configure the network using open APIs. This programmability simplifies network management tasks and allows for automation, resulting in greater operational efficiency.

iii. Separation of Control and Data Planes: SDN decouples the control plane from the data plane, providing more flexibility and agility in network management. With the control logic centralized in the SDN controller, network devices can focus solely on forwarding data, making the network more scalable and adaptable.

iv. Network Virtualization: SDN facilitates network virtualization by creating virtual network overlays that can operate independently from the underlying physical infrastructure. This virtualization capability allows for easier provisioning, management, and isolation of network resources, leading to improved resource utilization and enhanced network efficiency.

SDN offers numerous benefits to network environments. It enhances network agility, allowing for quick adaptation to changing requirements. The scalability of SDN enables efficient resource utilization, optimizing network performance. With programmability, administrators can automate tasks, reducing manual effort and increasing operational efficiency. The centralized control and management simplify network administration, leading to easier troubleshooting and configuration. Ultimately, SDN revolutionizes traditional networking by introducing a software-centric approach that brings greater flexibility, control, and efficiency to networks [22].

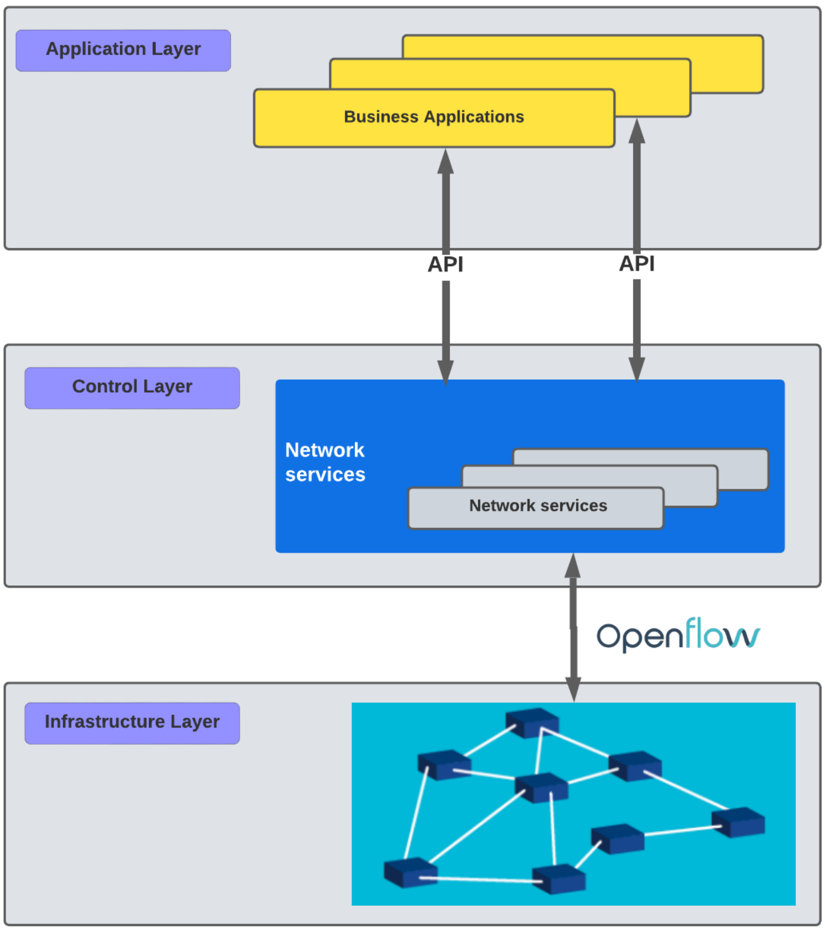


Figure 2.3: *Open Networking Foundation (ONF)/SDN architecture.*

2.3.3 Integrating NFV with other technologies

In recent years, there has been a significant focus on integrating Network Function Virtualization (NFV) with other cutting-edge technologies, including Software-Defined Networking (SDN), Cloud computing, and 5G [23], [1]. This integration has captured the attention of both the academic research community and industry due to the numerous benefits it offers.

The integration of NFV with SDN and Cloud computing is particularly advantageous because each technology brings unique features and approaches to address the challenges faced by today's and future networks [24]. NFV enables the abstraction of network functions by virtualizing them, as endorsed by the European Telecommunications Standards Institute (ETSI) [25]. SDN, on the other hand, provides network abstraction through centralized control and programmability, as supported by the Open Networking Foundation (ONF) [26]. Cloud computing offers computational abstraction by providing a shared pool of configurable computing resources, such as networks, servers, storage, applications, and services, as specified by the Distributed Management Task Force (DMTF) [27]. The core feature of Cloud computing is abstraction, allowing users and developers to be shielded from the underlying technical complexities.

To better understand the relationships between NFV, SDN, and Cloud computing, refer to Figure 2.4. While these technologies are complementary to each other, they can also function independently or be combined to create a powerful network architecture [4]. Deploying them together offers a range of advantages, including enhanced agility, cost reduction, dynamism, automation, and efficient resource scaling.

By integrating NFV with SDN, networks can benefit from the flexibility and programmability of SDN controllers, enabling efficient management and orchestration of virtualized network functions. Additionally, NFV leverages the scalability and resource allocation capabilities of Cloud computing, allowing for dynamic and on-demand provisioning of network resources. The combination of NFV, SDN, and Cloud computing not only improves network efficiency and resource utilization but also facilitates the rapid deployment of new services and applications.

Furthermore, the integration of NFV, SDN, and Cloud computing is highly relevant in the context of emerging technologies like 5G. The flexible and scalable nature of NFV and Cloud computing, along with the centralized control and management capabilities of SDN, can effectively support the diverse requirements of 5G networks, such as low latency, high bandwidth, and efficient network slicing.

In conclusion, the integration of NFV with SDN and Cloud computing presents a promising approach to overcome the challenges of modern networks. The combination of these technologies provides a holistic solution, offering agility, cost-effectiveness, automation, and resource scalability. As networks continue to evolve, leveraging the benefits of NFV, SDN, and Cloud computing in a synergistic manner will play a vital role in building efficient, adaptive, and future-proof network architectures.

2.4 Related work

Over the past few years, there has been a growing interest in the problem of VNF-PC (Virtual Network Function Placement and Chaining), leading to significant contributions from the research community. This problem is known to be more challenging than traditional virtual network embedding problems, as it involves the sharing of resources

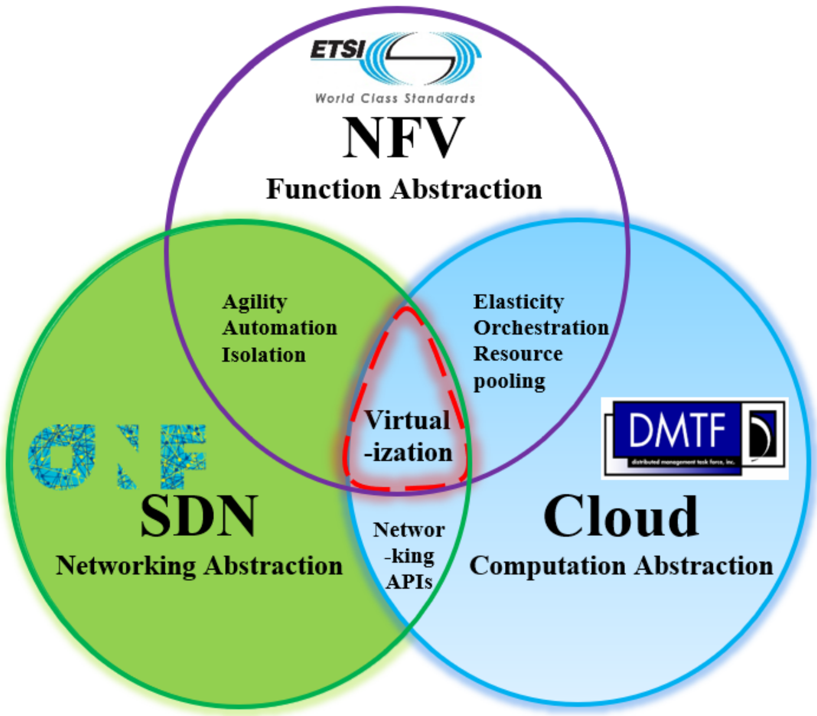


Figure 2.4: *Relationships between NFV, SDN and cloud computing [28].*

among multiple requests. Previous studies, such as [29–31], have highlighted the complexity of this problem[32], specifically emphasized the resource sharing aspect.

Researchers have approached the VNF-PC problem using two main categories of solution approaches. The first category consists of Mathematical Programming and Heuristic Approaches, which aim to find optimal or near-optimal solutions through mathematical modeling or intelligent algorithms. The second category comprises Robust Optimization Approaches, which focus on designing solutions that are resilient to uncertainties and variations in the network environment.

By exploring these different solution approaches, researchers have made significant progress in addressing the challenges posed by the VNF-PC problem.

2.4.1 Mathematical Programming and Heuristic Approaches

The VNF-PC problem has seen various solution approaches, including ILP and MILP models, which can suffer from combinatorial explosion due to their complexity. To address this, researchers have also explored heuristic approaches as alternatives.

In their work, [32] introduced a dynamic programming based heuristic for solving the VNF-PC problem. Their approach achieves solutions that are approximately 1.3 times the optimal solution obtained from the ILP model they presented, while being significantly faster, up to 3,000 times.

Another approach by [33] formulated the VNF-PC problem as a path-based Integer Linear Program (ILP) model, aiming to maximize service provider profits. They employed the column generation method to solve the problem. However, the time efficiency of this proposal remains a challenge, particularly when dealing with large-scale network scenarios.

In the study conducted by [34], the authors proposed a model for VNF chaining and placement. They developed an algorithm formulated as a mixed integer linear program (MILP) to address the problem. The MILP model considered various factors, including VNF chain ordering constraints, specific VNF forwarding modes (standard and fast-path modes), as well as variations in flow bit-rates.

On a related note, [35] investigated the energy-aware service function placement problem for SFC in data centers. They formulated the problem as a binary integer programming (BIP) model. Additionally, they proposed a novel SFC placement algorithm called Merge-RD, aiming to minimize energy consumption in data centers. The algorithm took into account computing and bandwidth resource constraints, as well as power models for servers and switches.

In the research paper [36], the authors propose the use of near-optimal approximation algorithms for VNF placement. These algorithms guarantee a placement solution with theoretically proven performance, specifically targeting the objective of minimizing the overall network cost.

In another study, [37] address the VNF-PC problem by formulating it as a MILP model. The objective is to determine the optimal placement of VNFs and routing of flows while minimizing resource utilization. Additionally, the authors propose a heuristic solution that solves the problem incrementally, allowing for support of a large number of SFC flows and enabling the handling of incoming flows without impacting existing ones.

In the research paper [38], the authors propose a model for formalizing the chaining of network functions using a context-free language. They introduce a mapping approach based on a Mixed Integer Quadratically Constrained Program (MIQCP) to determine the optimal placement of VNFs and their chaining while considering network resource limitations and VNF requirements.

On a related note, [39] present a heuristic approach aimed at guiding the ILP solver towards feasible and near-optimal solutions for the VNF-PC problem. This heuristic technique enhances the efficiency of the solver, albeit with a slight impact on solution quality. By utilizing this approach, the solver can provide solutions that are both practical and close to the optimal solution for the problem at hand.

In the research paper [40], the authors address the VNF-PC problem specifically for real-time applications. They formulate the problem as an ILP and propose a heuristic solution to overcome scalability issues associated with larger network instances.

Furthermore, in the work presented by [41], a suitable NFV network model for Internet Service Providers (ISPs) operations is defined. They introduce the generic VNF chain routing optimization problem and develop a mixed integer linear programming formulation to address it effectively.

The research presented in [42] introduces an SDN-NFV infrastructure designed to address the VNF placement and resource allocation (VNFPRA) problem in wireless MEC networks. The primary objective of this work is to minimize the overall placement and resource costs. To achieve this, the authors propose two algorithms: an optimal solution formulated as a MILP model and a genetic-based heuristic solution. The proposed solution is then compared with the Random-Fit Placement Algorithm (RFPA) and the First-Fit Placement Algorithm (FFPA) to evaluate its performance.

In a different study, [43] tackle the VNF-PC problem by formulating it as an optimization problem using the particle swarm optimization algorithm (PSO). Their aim is to minimize the paths average propagation delay, the number of used servers and the links average utilization while satisfying network demands and constraints. The proposed approach aims to achieve a balance between performance optimization and resource utilization in the VNF-PC problem.

In the study presented in [44], the authors focus on the cost-effective provision of Service Function Chains (SFCs) with arbitrary topologies in a multi-domain elastic optical network. The network consists of both private and public domains. The objective of the research is to minimize the total resource cost associated with the Virtual Network Function Placement and Chaining (VNF-PC) problem.

To meet this objective, the authors formulate the problem in the form of an integer linear programming (ILP) model. The ILP model takes into account the specific characteristics of the multi-domain elastic optical network and aims to find an optimal solution that minimizes the total resource cost for VNF-PC.

In addition to the ILP formulation, the authors propose the utilization of the minimal k-cut problem to design two time-efficient heuristics. These heuristics provide alternative solution approaches that prioritize computational efficiency while still aiming to achieve a cost-effective provisioning of SFCs in the multi-domain elastic optical network.

By combining the ILP formulation and the heuristics based on the minimal k-cut problem, [44] aims to provide effective and efficient solutions for the VNF-PC problem in a multi-domain elastic optical network. The research considers the cost aspect of the problem and provides different approaches to optimize resource utilization and reduce overall provisioning costs.

2.4.2 Robust Optimisation Approaches

Indeed, recent works on the VNF-PC problem have recognized the presence of uncertainties in real-world scenarios and have addressed them by incorporating robust optimization (RO) techniques. Unlike the approaches assuming perfect knowledge of all parameters, RO-based methods aim to provide solutions that are resilient to uncertainties and variations in the network environment.

By considering uncertainties, such as changes in traffic demands, resource availability, or link failures, RO-based approaches for the VNF-PC problem seek to find solutions that can adapt to these uncertainties while still satisfying the desired performance objectives. These approaches typically involve formulating optimization models that incorporate uncertainty sets and constraints, allowing for more robust decision-making and resilient network design.

By considering uncertainties in the VNF-PC problem, researchers aim to enhance the practical applicability and effectiveness of their solutions in real-world network scenarios where parameter variations and uncertainties are inevitable. RO approaches have indeed shown success in addressing similar problems like virtual network embedding [45]. However, it was the work of [46] that first applied robust optimization techniques specifically to the VNF-PC problem.

In their study, [46] focused on modeling uncertainties in the VNF-PC problem by considering demand fluctuations in the network components. They limited the number of parameters allowed to deviate from their nominal values, taking into account the practical constraints of the problem. By incorporating robust optimization, they aimed to find solutions that are resilient to uncertainties and can maintain performance even under varying conditions.

By pioneering the application of robust optimization to the VNF-PC problem, [46] contributed to the advancement of research in handling uncertainties in this specific context. Their work opened up possibilities for developing more robust and adaptive approaches for VNF-PC, enabling network operators to handle uncertainties effectively and achieve reliable network performance.

The research presented in [47] extends the traditional view of network services as fixed chains of functions by addressing both the composition and placement of Service Function Chains (SFCs). The paper introduces an under-specified structure for network composed services, allowing for the flexibility to change the order of SFC functions without impacting the functionality of the service. To represent this flexibility, the authors propose a YANG data model.

In addition to the proposed data model, [47] presents a heuristic algorithm for the placement of a combination of services. The algorithm aims to find suitable placements for service components along shortest paths in the network that have sufficient resource capacity to accommodate the required services. By leveraging this heuristic algorithm, the paper aims to optimize the placement of services within the network infrastructure, considering both the functional requirements and the resource constraints.

Overall, the work in [47] provides a holistic approach that considers both the composition and placement of SFCs, introducing flexibility in the service structure and proposing a heuristic algorithm for effective placement decision-making.

In the research paper [48], the authors propose an approach based on the Markov approximate method to tackle the VNF-PC problem. The method involves starting with a random feasible solution and iteratively transforming the current solution to another feasible solution based on the states of the substrate network. The process continues until the steady-state distribution of the Markov chain is reached.

However, the authors note that this method has a drawback in terms of convergence time, particularly when dealing with larger network sizes. The optimization problem's large state space contributes to the lengthy convergence process required by the Markov approximate method. This limitation highlights the need for more efficient algorithms to address the VNF-PC problem in scenarios involving larger network infrastructures.

In the research presented in [49], the authors focus on addressing the application module placement and task scheduling in a heterogeneous "edge-cloud" network environment. Their objective is to establish a mapping scheme between application modules and the underlying resource equipment. To achieve this, they propose a dynamic task processing heuristic algorithm that takes into account both tolerant task latency and system power consumption as key factors.

On a different note, [50] proposes a resource allocation algorithm for the VNF-PC problem, utilizing genetic algorithms (GAs). The algorithm consists of two stages: the initial placement of Virtual Network Functions (VNFs) and the subsequent scaling of VNFs to accommodate changes in traffic demand. However, similar to the Markov method mentioned earlier, the GA-based solution in [50] also faces the challenge of a lengthy convergence process due to the nature of crossover and mutation operations inherent in genetic algorithms. Both studies highlight the need for more efficient optimization algorithms to solve the VNF-PC problem, as the convergence time of traditional methods can be a limiting factor, especially when dealing with larger networks or complex scenarios. Finding ways to reduce convergence time while maintaining solution quality remains an important area of research in this field.

In the research presented in [51], the authors address the challenge of high utilization or fragmentation levels in a Network Function Virtualization Infrastructure (NFV-I), which can limit the feasible solution space for Service Function Chain (SFC) embedding methods. To overcome this issue, they propose a solution approach called SFC graph transformation (SFC-GT). They formulate the SFC-GT as a multi-objective optimization problem and develop a mixed-integer linear program (MILP) to solve it. The objective is to explore the potential of expanding the SFC graph before its embedding, thereby mitigating the constraints imposed by high utilization or fragmentation levels.

On the other hand, in [52], the authors consider both the VNF placement problem and policy-aware traffic steering to maximize the number of served flows. They decompose the problem into a master problem, which involves VNF placement, and a sub-problem, which focuses on policy-aware routing for each flow. The proposed model can be applied to both online and offline scenarios, addressing the optimization of VNF placement and traffic steering with consideration for policy-based requirements.

Both studies contribute to addressing different aspects of the VNF-PC problem by formulating it as an optimization problem and proposing algorithms to find optimal or near-optimal solutions. These approaches aim to improve the efficiency and effectiveness of VNF placement and traffic steering, considering various constraints and objectives.

In the research presented in [53], the authors focus on the problem of decomposing and embedding network services, specifically Network Service Chains (NSCs). They propose two algorithms to address this problem and map NSCs onto the network infrastructure, while considering the possibility of decomposing network functions using Network Function Virtualization (NFV) techniques.

The first algorithm presented is based on Integer Linear Programming (ILP). It formulates the problem as an optimization model to minimize the mapping cost, taking into account the requirements of the NSCs and the capabilities of the network infrastructure. The ILP algorithm aims to make an optimal selection of the network function decomposition to achieve the minimum mapping cost.

To address the scalability issues associated with the ILP formulation, the authors propose a heuristic algorithm as a second approach. The heuristic algorithm provides a practical solution that alleviates the computational complexity of the ILP algorithm while aiming to minimize the mapping cost. By employing this heuristic algorithm, the authors aim to achieve a good compromise between computational efficiency and solution quality.

Overall, the work in [53] presents two novel algorithms, an ILP-based approach and a heuristic, for the decomposition and embedding of NSCs in the network infrastructure. These algorithms provide different strategies to optimize the mapping cost, taking into consideration the requirements of NSCs and the capabilities of the network functions.

In their recent work, [54] present a unified framework for maximizing network throughput while ensuring that the end-to-end delay requirements of accepted requests are met. The goal is to accept as many requests as possible while optimizing resource consumption and reducing network operational costs.

To address this objective, the authors propose an integer linear programming (ILP) solution for the problem, especially when the size of the substrate network is small. The ILP formulation considers two different techniques for scaling VNF instances: horizontal scaling and vertical scaling.

The horizontal scaling technique involves migrating existing VNF instances from their current locations to new locations, allowing these instances to be shared by multiple requests. This approach aims to reduce resource consumption and operational costs by maximizing the utilization of existing VNF instances.

On the other hand, the vertical scaling technique focuses on instantiating new VNF instances to meet the demands of new requests. This is done when it becomes more expensive to share existing VNF instances or when the end-to-end delay requirements of currently executing requests are at risk of being violated.

By jointly exploring these two scaling techniques, [54] aim to optimize resource allocation and enhance network performance. Their unified framework provides a comprehensive approach to maximize network throughput while considering both resource efficiency and end-to-end delay requirements.

Overall, the work by [54] contributes to the field by presenting a unified framework for VNF instance scaling and proposing an ILP-based solution for optimizing network throughput while meeting end-to-end delay requirements.

[55] focuses on deployment of virtual network functions in a highly dynamic network where virtual nodes and virtual links are created and destroyed depending on the traffic volumes, the service requests or high-level goals such as reduction in energy consumption. [55] gives an architecture based on an orchestrator that ensures the automatic placement of the VNF and the allocation of network services on them, supported by a monitoring system that collects and reports on the behaviour of the network resources.

In their work, [56] address the challenge of routing and Virtual Network Function (VNF) deployment in the context of network function virtualization. They propose an optimization model that aims to minimize the maximum index of used frequency slots, the number of initialized VNFs and the number of used frequency slots. This model takes into account the dependency among different VNFs. To tackle the service chain mapping problem in dynamic virtual networks, the authors introduce a novel algorithm called PDQN-VNFSC. This algorithm combines prediction algorithms with Deep Q-Network (DQN) to optimize the real-time mapping process of virtual network service chains. The authors formulate the mapping problem as a partial observable Markov decision process and employ global and long-term benefits for optimization. By leveraging an offline learning and online deployment decision framework, the algorithm effectively maps the service chain of virtual network functions.

The work in [57] address the problem of VNF-PC from an original perspective: it propose an advanced and dynamic pricing algorithm for pricing the requested substrate resources. The aim of the proposed algorithm is to increase the infrastructure provider's revenue on the basis of historical data, current infrastructure utilization levels and the pricing of competitors.

In the work by [58], the authors propose an approach for efficient and proactive resource provisioning in the context of the VNF-PC problem. They employ a linear programming model (ILP) with randomized rounding to find a near-optimal solution.

The proactive algorithm developed by [58] aims to dynamically allocate resources in a way that maximizes the acceptance of requests while ensuring timing guarantees. By formulating the problem as an ILP and using randomized rounding techniques, the algorithm can effectively allocate resources to accommodate as many requests as possible within the given constraints.

This approach introduces a proactive strategy to resource provisioning, allowing for efficient and timely allocation of resources in the VNF-PC problem. By leveraging ILP and randomized rounding, the algorithm strikes a balance between optimization and practical feasibility, enabling the system to handle a high volume of requests while meeting timing requirements.

The VNF-PC problem is known to be NP-Hard, which means that finding exact solutions can be computationally expensive and time-consuming [8]. As a result, heuristic algorithms are commonly used to tackle the problem by providing faster but approximate solutions.

Heuristic algorithms offer a scalable approach to the VNF-PC problem, allowing for more efficient handling of larger problem instances. While they may sacrifice solution quality compared to exact methods, they can still provide reasonably good solutions within a reasonable time frame.

However, despite the advancements in heuristic algorithms, there are still important aspects that require further study in effectively managing and allocating resource usage in NFV-I. This work aims to address these aspects by making contributions in the areas of VNF-PC, mathematical modeling, and heuristic optimization algorithms.

The contributions of this thesis involve developing novel approaches to tackle different versions the VNF-PC problem, including the formulation of mathematical models that capture the complexities of resource allocation and optimization. Additionally, the work proposes heuristic optimization algorithms that can efficiently find near-optimal solutions in a timely manner.

By focusing on these aspects, this work aims to enhance our understanding of the VNF-PC problem and provide practical solutions that balance solution quality and computational efficiency.

2.5 Conclusion

In contrast to the existing approaches discussed in this section, the primary contribution of this thesis lies in the introduction of novel and efficient heuristic approaches to address the VNF-PC problem. The key objective of these heuristics is to enhance scalability while simultaneously aiming to achieve optimal solutions. By focusing on developing innovative algorithms, this thesis aims to overcome the limitations of previous methods and provide more effective solutions for the VNF-PC problem.

The proposed heuristics leverage advanced optimization techniques to efficiently explore the solution space and identify near-optimal configurations. These approaches take into account various constraints and requirements of the VNF-PC problem, such as computational capabilities, bandwidth requirements, and overall placement costs. By integrating these factors into the design of the heuristics, the aim is to strike a balance between solution quality and computational efficiency.

One notable aspect of the heuristics developed in this thesis is their emphasis on scalability. The goal is to address the challenges posed by large-scale processing networks and service function chains, enabling efficient resource allocation and minimizing the overall placement cost. Through careful algorithm design and optimization, the proposed heuristics aim to handle complex and dynamic scenarios, accommodating the increasing demands and traffic loads within NFV environments.

Moreover, the focus on finding exactly optimal solutions distinguishes this thesis from previous approaches. While heuristics are typically employed to approximate solutions within a reasonable time frame, the heuristics proposed in this thesis strive to achieve solutions that are as close to the true optimum as possible. By leveraging advanced optimization techniques and intelligent search strategies, the aim is to improve the quality of solutions and provide more accurate representations of optimal configurations for the VNF-PC problem.

Overall, the novelty of this thesis lies in the introduction of efficient heuristics that address different versions of the VNF-PC problem with a focus on improved scalability and the pursuit of exactly optimal solutions. By combining advanced optimization techniques, intelligent search strategies, and consideration of various constraints, the proposed heuristics aim to contribute to the field by providing more effective and accurate solutions for the resource allocation optimization in NFV environments.

Chapter 3

Defining and Modeling the VNF Placement and Chaining Problem: A Mathematical Approach

I have never seen a teacher better than time, nor a learner worse than a human being.

Al-Khwarizmi

Abstract

This chapter focuses on optimizing the allocation of Virtual Network Functions (VNFs) and their interconnections within Network Function Virtualization (NFV). It presents a mathematical formulation using Integer Linear Programming (ILP) to minimize placement and bandwidth costs. The chapter introduces the concept of combinatorial Benders decomposition as an efficient solution approach.

[9] Issam Abdeldjalil Ikhelef et al. "A Knapsack-based Optimization Algorithm for VNF Placement and Chaining Problem". In: *2022 IEEE 47th Conference on Local Computer Networks (LCN)*. 2022, pp. 430–437. DOI: 10.1109/LCN53696.2022.9843566

[10] Issam Abdeldjalil Ikhelef et al. "Efficient Decomposition-Based Methods for Optimal VNF Placement and Chaining". In: *2023 24st Asia-Pacific Network Operations and Management Symposium (APNOMS)*. 2023, pp. 89–94

Chapter content

3.1 Introduction

In this chapter, our attention turns to the intricate challenge of the VNF placement and chaining problem within the broader landscape of Network Function Virtualization (NFV). This intricate problem revolves around the meticulous determination of the most optimal allocation of Virtual Network Functions (VNFs) and the arrangement of their interconnected sequences, referred to as Service Function Chains (SFCs), within the confines of an NFV-Infrastructure (NFV-I). The core aim is to effectively minimize the cumulative cost associated with both VNF placements and the allocation of bandwidth resources.

In our pursuit of a viable solution, we introduce a comprehensive mathematical formulation that draws upon the power of Integer Linear Programming (ILP) as the driving force behind our optimization technique. This formulation intricately incorporates decision variables cast in the mold of binary constructs, meticulously representing the assignment of VNFs to specific nodes as well as the embedding of VNF-links within the infrastructure. The keystone of this formulation resides in the definition of the objective function, masterfully tailored to minimize the collective cost incurred by the intricate processes of VNF placement and bandwidth resource allocation.

Expanding our arsenal of optimization strategies, we delve deeper into the area of combinatorial Benders decomposition, a sophisticated approach that resonates with the intricacies of the VNF-PC problem. This technique involves the systematic partitioning of the problem into a main problem, called Master problem, and a linear sub-problem. Additionally, we delve deeper into the formulation of combinatorial cuts, essential elements that identify infeasibilities within the sub-problem and propel the search for feasible solutions.

3.2 VNFs placement and chaining problem

3.2.1 Problem description

In this section, we present the VNF placement and chaining (VNF-PC) problem. After explaining the context of the problem, we define and explain the variables used to formulate the problem as an ILP optimization problem.

To make it easier to understand, let's consider the example of Figure 3.1 which represents an NFV-Infrastructure (NFV-I) composed of three node servers (Server1, 2 and 3) and five switches (Switch A, Switch B, Switch C, Switch S and Switch T), where Switch S and Switch T symbolically represent the sources S and targets T of the SFC. Each server has a certain capacity of CPU to host VNFs and its own pricing policy. The servers are interconnected by substrate virtual links. Each link has a given bandwidth cost and capacity. The substrate virtual links are virtual links embedded on substrate paths that interconnect end nodes supporting substrate servers.

The Figure 3.2 shows an SFC composed of two VNFs to be placed and chained in the NFV-I. Each virtual network function (VNF) requires a number of CPU resource to run properly. The optimal placement and chaining of the SFC of the figure 3.2 is highlighted by the green lines in figure 3.3: *VNF1* will be placed in *Server* S_1 and *VNF2* will be placed on *Server* S_3 and path S-A-C-T (green arrows in Figure 3.3) determines the flow routing for the SFC. The total allocation cost for this solution is calculated as 360 units (30×(3+1+4)+20×4+40×1=360), which is the sum of the allocated resource for links usage and VNFs according to the specified bandwidth and CPU requirements, and their associated costs.

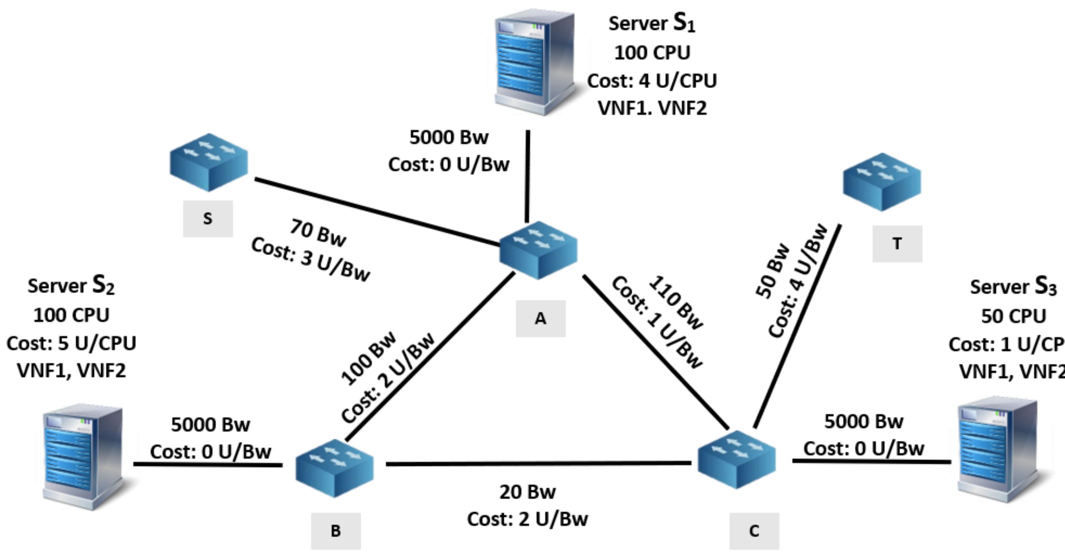


Figure 3.1: *An example of NFV-I topology.*

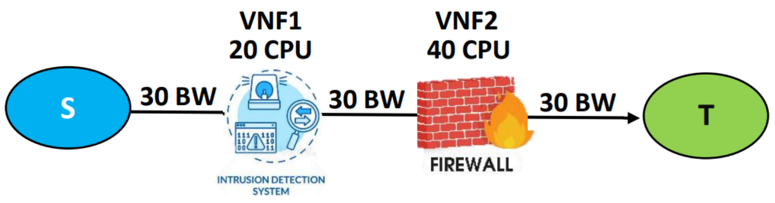


Figure 3.2: *An example of a Service Function Chain (SFC).*

3.2.2 Complexity Implications

The NP-hardness of VNF-PC implies that finding an exact solution to this problem is computationally intractable within polynomial time. As a consequence, any algorithm attempting to solve the VNF-PC problem optimally must contend with an inherent level of computational complexity [8].

This result has profound implications for the field of network function virtualization, emphasizing the need for approximation algorithms, heuristics, and practical strategies to address VNF-PC in real-world scenarios. While the problem's complexity poses challenges, it also motivates researchers and practitioners to explore innovative algorithmic approaches that balance computational efficiency with solution quality [59].

3.2.3 Proposals

In the context of the VNF-PC problem, we first propose an Integer Linear Programming (ILP) solution to address it.

The ILP solution formulates the VNF-PC problem as an optimization problem using linear equations and inequalities with integer variables. The objective function and constraints are defined based on the problem requirements, such as resource usage. The ILP solver then searches for the optimal values of the variables that minimize or maximize the objective function while satisfying the constraints. By leveraging the power of mathematical programming techniques, the ILP solution provides an optimal solution to the VNF-PC problem.

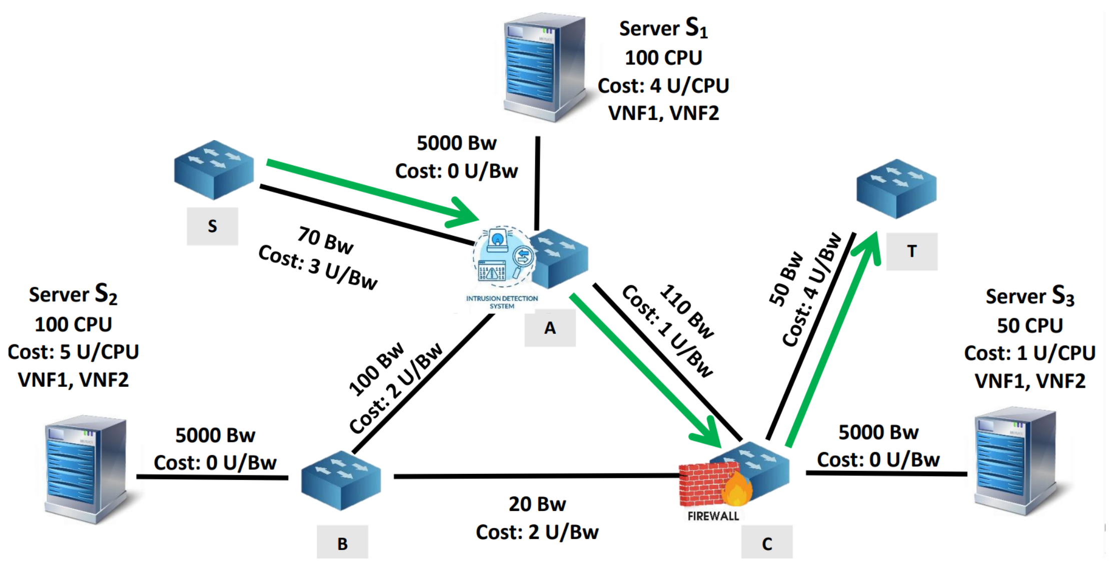


Figure 3.3: *SFC optimal placement.*

3.2.4 Mathematical Modelling of VNF-PC problem

The NFV substrate network is modeled by an undirected weighted graph, noted $G_S = (V_S, E_S)$ where V_S is the set of nodes corresponding to servers and switches of the NVF-I, and E_S is the set of substrate virtual links.

Each node in the network is associated with a list of its available resources: memory, CPU, etc.

Each type of resource $r \in R$ is identified by its index number where $R = \{0, 1, 2, 3, ...\}$.

For a given resource r of a node U, its capacities and costs are denoted by respectively C_U^r and α_U^r. Similarly, for each substrate virtual link $l_S = (l_S^a, l_S^b)$, its bandwidth capacity is denoted by $C_{l_S}^{Bwd}$, and its bandwidth cost by β_{l_S}.

Each Service Function Chain (SFC) is also modeled by a graph interconnecting a *source* node (represented by U_S, the switch on its left side), k VNFs and a *target* node (represented by U_T, the switch on its right side), as shown in Figure 3.2. A SFC is represented by a graph $G_f(U_S, U_T) = (V_f, E_f)$, where E_f is the set of virtual links that connect the VNFs and V_f contains all the VNFs of the service chain plus the nodes B_s and B_t which represent, symbolically, on V_f the two NFV-I switches U_S and U_T, respectively.

Each VNF $v \in V_f$ has a demand for resource $r \in R$, denoted D_v^r. Each virtual link $l_v \in E_f$ has a demand for bandwidth denoted $D_{l_v}^{Bwd}$.

The traffic flow coming from the source U_S is processed through the SFC before being delivered to the target node U_T.

For each virtual link $l_v \in E_f$, its end (edges) nodes are designated by l_v^a and l_v^b.

We propose, hereafter, the ILP I formulation of the VNF-PC problem:

3.2.4.1 Decision variables

$$x_U^v = \begin{cases} 1, & \textit{if the VNF}\, v \in V_f \textit{ is assigned to the node}\, U \in V_S; \\ 0, & \textit{otherwise} \end{cases}$$

$$y_{l_S}^{l_v} = \begin{cases} 1, & \textit{if the SFC virtual link } l_v \in E_f \textit{ is embedded on the substrate virtual link}\, l_S \in E_S; \\ 0, & \textit{otherwise} \end{cases}$$

Table 3.1 summarizes the parameters and the used variables.

Notation	Description
G_S	Weighted graph modeling the NFV-I substrate network
V_S	Set of nodes corresponding to servers and switches of the NVF-I
E_S	Set of substrate virtual links
R	Set of NFV-I resources
C_U^r	Capacity of node U in resource r
α_U^r	Cost of resource r on node U
$C_{l_S}^{Bwd}$	Bandwidth capacity of substrate virtual link l_S
β_{l_S}	Bandwidth cost on the substrate virtual link l_S
l_S^a, l_S^b	Edges nodes of the substrate virtual link l_S
G_f	Weighted graph modeling the SFC
V_f	Set of VNFs
E_f	Set of virtual links that connect the VNFs
U_S	Source node on NFV-I
U_T	Target node on NFV-I
B_s	Source node on SFC
B_t	Target node on SFC
D_v^r	Capacity of VNF v in resource r
$D_{l_v}^{Bwd}$	Demand of link l_v for bandwidth
l_v^a, l_v^b	Edges nodes of the virtual link l_v
x_U^v	binary variable equal to 1 if the VNF $v \in V_f$ is assigned to the node $U \in V_S$, 0 otherwise.
$y_{l_S}^{l_v}$	binary variable describes whether the virtual link $l_v \in E_f$ is embedded on the substrate virtual link $l_S \in E_S$ or not.

Table 3.1: *Main notations*

3.2.4.2 Objective Function

Our objective is to minimize the combined cost of VNF allocation (A) and bandwidth resource allocation (B):

$$Z = \min A + B \tag{3.1}$$

Where:

- A represents the overall cost of placing VNFs on NFV infrastructure nodes:

$$A = \sum_{r \in R} \sum_{v \in V_f} \sum_{U \in V_S} x_U^v \times D_v^r \times \alpha_U^r \tag{3.2}$$

- B corresponds to the cost of allocating bandwidth resources:

$$B = \sum_{l_v \in E_f} \sum_{l_S \in E_S} y_{l_S}^{l_v} \times D_{l_v}^{Bwd} \times \beta_{l_S} \tag{3.3}$$

Here, α_U^r and β_{l_S} denote the unitary cost of resource r and bandwidth, respectively.

3.2.4.3 Constraints

The problem is subject to the following constraints:

$$\sum_{v \in V_f} x_U^v \times D_v^r \leq C_U^r \quad \forall U \in V_S \quad \forall r \in R \tag{3.4}$$

$$\sum_{l_v \in E_f} y_{l_S}^{l_v} \times D_{l_v}^{Bwd} \leq C_{l_S}^{Bwd} \quad \forall \quad l_S \in E_S \tag{3.5}$$

$$\sum_{U \in V_S} x_U^v = 1 \quad \forall \quad v \in V_f \tag{3.6}$$

$$\sum_{l_S \in E_S} y_{l_S}^{l_v} \leq 1 \quad \forall \quad l_v \in E_f \tag{3.7}$$

$$x_{U_S}^{B_s} = 1 \quad x_{U_T}^{B_t} = 1 \tag{3.8}$$

$$x_{l_S^a}^{l_v^a} + x_{l_S^b}^{l_v^b} - y_{l_S}^{l_v} \leq 1 \quad \forall l_v \in E_f \quad \forall l_S \in E_S \tag{3.9}$$

$$x_{l_S^a}^{l_v^b} + x_{l_S^b}^{l_v^a} - y_{l_S}^{l_v} \leq 1 \quad \forall l_v \in E_f \quad \forall l_S \in E_S \tag{3.10}$$

Where:

- Constraint (3.4) ensures that the demand D_v^r of VNF v for resource r is within the capacity C_U^r of substrate node U.
- Constraint (3.5) guarantees that the total allocated bandwidth for any substrate virtual link l_S is within its capacity $C_{l_S}^{Bwd}$.
- Constraint (3.6) ensures that each VNF v in the service function chain (SFC) is placed on a single substrate node.
- Constraint (3.7) ensures that each virtual link l_v is mapped to at most one substrate virtual link l_S.
- Constraint (3.8) associates the virtual source node B_s with the substrate source node U_S, and similarly for the target nodes.

- Constraints (3.9) and (3.10) handle the mapping of the end nodes of virtual link l_v to substrate nodes l_s^a and l_s^b.

3.2.5 Ordered VNF-PC

We further consider an *ordered* variant of ILP I formulation, in which an ordering is imposed on the nodes of the NFV-I. Then, the VNFs of the SFC must be placed preserving the same ordering (i.e., the label of the node VNF_k is assigned to must be smaller than all subsequent labels VNF_m, where $m > k$). This variant ensures the VNFs are considered in a specific order with regards to a given reference point (see e.g., [60]).

Alongside ILP I formulation, we include the following constraint on the binary x_U^v variables $(\forall U \in V_S; \forall v \in V_f)$:

$$x_U^v \leq \sum_{b \in V_S : b > U} x_b^v \qquad \forall U \in V_S; v \in V_f.$$

It is known that including constraints on the binary variables of a MILP can lead to faster solve times [61]; hence, we expect the ordered VNF-PC to exhibit such properties.

3.3 Model Decomposition using Combinatorial Benders Cuts

In Section 2.4, we discussed different approaches to solving the VNF-PC problem. While heuristics can provide quick solutions, exact approaches have shown the potential to find high-quality solutions efficiently. We don't need to compromise solution quality for efficiency, especially considering the significant impact of sub-optimal solutions in real-world scenarios.

In this context, we investigate the utilization of Combinatorial Benders Decomposition (CBD) as introduced by Codato and Fischetti [62]. The Benders' decomposition technique, initially developed in 1962 by Benders [63], is well-documented in various references on large-scale optimization and stochastic programming. A comprehensive explanation of this method can be found in the book by Benders [64].

The main idea behind Benders decomposition is to decompose a complex optimization problem into smaller, more manageable sub-problems. These sub-problems are usually formulated as linear programs and are designed to capture some aspect of the original problem. Benders decomposition is particularly useful when sub-problems can be solved efficiently and their solutions can help improve the overall solution of the original

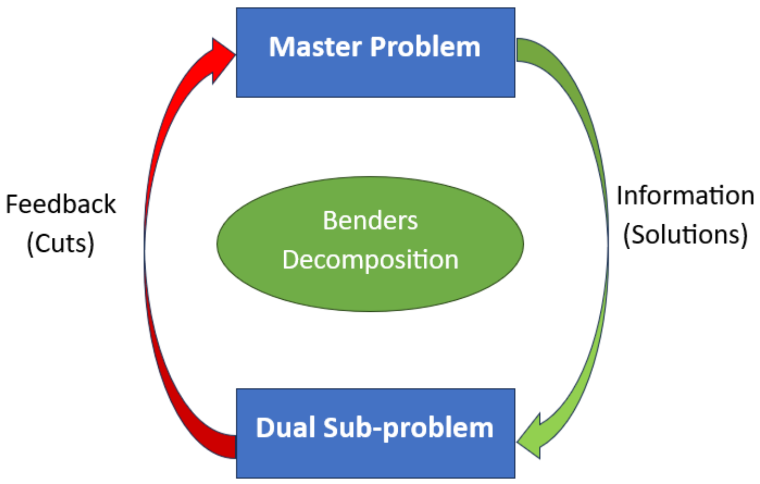


Figure 3.4: *Schematic representation of Benders decomposition method [66].*

problem. (See Figure 3.4).

The decomposition process involves solving a *Master* problem and a series of *Sub-problems* iteratively. The master problem typically deals with a subset of the original variables and constraints and aims to find an initial feasible solution. Then, in each iteration, a sub-problem is formulated and solved to generate a "cut" or constraint that is added to the master problem. These cuts help to tighten the feasible region of the master problem, bringing it closer to the optimal solution [65].

3.3.1 Master Problem

The master problem aims to find feasible solutions for the binary variables x_U^v ($\forall U \in V_s; v \in V_f$) and treats $y_{l_S}^{l_v}$ ($\forall l_S \in E_S; l_v \in E_f$) as binary for decomposition purposes.

MASTER:

$$\min Z := A + B \qquad (2a)$$

$$\text{where} \quad A := \sum_{r\in R}\sum_{v\in V_f}\sum_{U\in V_s} x_U^v \cdot D_v^r \cdot \alpha_U^r$$

$$B := \sum_{l_v\in E_f}\sum_{l_S\in E_S} y_{l_S}^{l_v} \cdot D_{l_v}^{Bwd} \cdot \beta_{l_S}$$

$$\text{s.t.} \quad \sum_{U\in V_s} x_U^v = 1 \qquad \forall v \in V_f \qquad (2b)$$

$$x_{l_S^a}^{l_v^a} + x_{l_S^b}^{l_v^b} - y_{l_S}^{l_v} \le 1 \qquad \forall l_v \in E_f; l_S \in E_S \qquad (2c)$$

$$x_{l_S^a}^{l_v^b} + x_{l_S^b}^{l_v^a} - y_{l_S}^{l_v} \le 1 \qquad \forall l_v \in E_f; l_S \in E_S \qquad (2d)$$

$$x_U^v \in \{0,1\} \qquad \forall U \in V_s; v \in V_f \qquad (2e)$$

$$y_{l_S}^{l_v} \in [0,1] \qquad \forall l_S \in E_s; l_v \in E_f \qquad (2f)$$

The master problem solution provides binary variable assignments. Let $\hat{x}_U^v$ ($\forall U \in V_s; v \in V_f$) and $\hat{y}_{l_S}^{l_v}$ ($\forall l_S \in E_S; l_v \in E_f$) represent fixed variable values obtained from solving the master problem.

Note that at this step, the y variables are not required to be binary, since their value is fixed to 0 or 1 in constraints (3.9)-(3.10).

3.3.2 Sub-problem

The sub-problem checks the feasibility of the fixed binary variables from the master problem.

<u>Sub</u>$(\hat{x}, \hat{y})$:

$$\begin{aligned}
\min \quad & \xi \in \mathbb{R}^+ && && \text{(3a)} \\
\text{s.t.} \quad & \sum_{v \in V_f} \hat{x}_U^v \cdot D_v^r \le C_U^r && \forall U \in V_s; r \in R && \text{(3b)} \\
& \sum_{l_v \in E_f} \hat{y}_{l_S}^{l_v} \cdot D_{l_v}^{Bwd} \le C_{l_S}^{Bwd} && \forall l_S \in E_S && \text{(3c)}
\end{aligned}$$

If the sub-problem is feasible, the binary variable assignment from the master problem is optimal. If not, the infeasible binary variables are identified, guiding the search towards feasible solutions. This is achieved through combinatorial cuts introduced to the master problem.

This process of iteratively solving the master problem and using the sub-problem to guide variable assignments forms the basis of CBD. This approach can effectively handle large-scale MILP problems with numerous binary variables. The formulation and decomposition process can lead to better solution quality while maintaining computational efficiency.

3.3.3 Formulating the Combinatorial Cuts

If the sub-problem (<u>Sub</u>$(\hat{x}, \hat{y})$) is found to be infeasible, the aim is to identify the irreducible infeasible subsystem (IIS) within the sub-problem. The IIS comprises a subset of the sub-problem's constraints, such that removing any of them would render the problem feasible. The uniqueness of IISs is not guaranteed; indeed, for a given problem, the number of IISs can be exponential in the system's size [67, 68]. Thus, developing effective methods to find IISs that lead to potent combinatorial cuts can enhance the efficiency of the CBD approach (as exemplified in [69]).

The occurrence of an infeasibility within the IIS can be attributed to constraints (3b)-(3c), contingent on the fixed binary variable values. Consequently, it becomes evident that one of the binary variables within these constraints must be altered to achieve a feasible solution.

If a constraint of the form (3b) exists within the IIS, the over-resourced node can be precisely identified. Specifically, given a node $U^\star \in V_s$, it can be concluded that the assignment of VNFs to that node is infeasible. The cut then signifies that at least one of the VNFs ($v \in V_f$) assigned to node $U^\star$ must be repositioned. Formally, when a set $N \subseteq V_s$ is formed to identify nodes causing infeasibility (i.e., constraints appearing in the IIS), the following term contributes to the combinatorial cut:

$$T_1 := \sum_{n^\star \in N} \sum_{k \in V_f : \hat{x}^k_{n^\star} = 1} 1 - x^k_n \geq 0. \tag{3.11}$$

In cases where a constraint of the form (3c) is present within the IIS, the identification of over-resourced links is approached in a similar manner. For a given link $l_S{}^\star \in E_s$, it becomes apparent that assigning VNFs to the nodes composing that virtual link ($l_v = (l^a_v, l^b_v) \in E_f$) leads to infeasibility. As a result, the necessity arises for at least one of the two nodes to be reassigned to eliminate the infeasibility. The assignment of nodes can be derived from constraints (2c) and (2d) in the master problem. With $L \subseteq E_s$ representing the set of links causing infeasibility (i.e., constraints appearing in the IIS), each link $l_S{}^\star \in L$ consists of two nodes that satisfy either $\hat{x}^{e^a}_{l^a_S{}^\star} + \hat{x}^{e^b}_{l^b_S{}^\star} - y^{l_v}_{l_S{}^\star} = 1$ (referred to as event E_1 for brevity) or $\hat{x}^{e^b}_{l^a_S{}^\star} + \hat{x}^{e^a}_{l^b_S{}^\star} - y^{l_v}_{l_S{}^\star} = 1$ (referred to as event E_2). Given that the SFC is symmetric regarding bandwidth demands, it follows that one of the two nodes composing the link must be altered. Consequently, the following constraint contributes to the combinatorial cut:

$$\begin{aligned} T_2 &:= \sum_{l_S{}^\star \in L} \sum_{l_v \in E_f : \hat{y}^{l_v}_{l_S{}^\star} = 1} 1 - y^{l_v}_{l_S} \geq 0 \\ \implies T_2 &:= \sum_{l_S{}^\star \in L} \sum_{l_v \in E_f : E_1 || E_2} 2 - x^{l^a_v}_{l^a_S} - x^{l^b_v}_{l^b_S} \geq 0. \end{aligned} \tag{3.12}$$

Combining these two terms results in the combinatorial cut presented formally as:

$$T_1 + T_2 \geq 0. \tag{3.13}$$

The combinatorial cut signifies that modifying at least one fixed binary variable within the IIS (i.e., those causing the infeasibility) is essential to achieve a better solution. If the sub-problem is found to be infeasible, this cut is integrated into the master problem. Consequently, the master problem is solved while adhering to this cut, yielding a new set of fixed binary variable values. This iterative process persists until the optimal solution is attained (i.e., when the sub-problem is feasible).

The CBD process for addressing the VNF-PC problem is formalized in Algorithm1.

3.3.4 Evaluation of Non-Viable Solution Nodes

The combinatorial Benders decomposition (CBD) approach, as discussed in the preceding section, generates combinatorial cuts as expressed in equation 3.13 when the master problem (2) has been solved to its optimal solution. This occurs when a feasible configuration of fixed binary variables x_U^v and pseudo-binary variables $y_{l_S}^{l_v}$ is achieved.

Furthermore, we can extract solution insights from non-feasible solutions of the master problem, where the binary variables might assume non-binary values. Before the CBD branch-and-cut process proceeds from these non-viable solutions, it's possible to assess whether such branching will result in infeasibility. If this is anticipated, we can tailor the branching at these unviable nodes to circumvent further examination of already identified infeasible scenarios. This integration augments the CBD approach with the capability to swiftly identify a significant number of infeasible solutions.

Let's consider a scenario at a given node $U^\star \in V_s$, where $\sum_{v \in V_f} x_{U^\star}^v \cdot D_v > C_{U^\star}^r$, and the values of $x_{U^\star}^v$ at this node lie within the interval $[0, 1]$. While such a solution is infeasible within the master problem due to non-binary values of the x_U^v variables, we can still glean valuable insights into the feasibility of the current branch within the branch-and-bound tree.

To address this, a pragmatic branching heuristic is incorporated within the decomposition approach. Specifically, we design the branching direction to divert away from nodes that are prone to infeasibility, as such paths are more likely to lead to infeasible solutions.
In essence, we establish a new branch where all non-zero binary variables are set to 0. This deliberate maneuver aims to facilitate the exploration of unexplored territories within the search space. Given our overall understanding of the VNF-PC problem, we recognize that the current branch is more predisposed to encountering infeasibilities.

Therefore, by venturing into unexplored regions, we increase the likelihood of discovering a feasible solution rather than persisting on the current path.

Empirical observations, presented in Section 3.3.5, suggest that the count of non-zero variables for a given node seldom exceeds $|V_f|-4$.

Algorithm 1 CBD for the VNF-PC Problem

```
1:  Input:
    NFV:
2:  Cost ($\alpha_U^r$) and capacity ($C_U^r$) for each node $n \in V_s$;
    Cost ($\beta_{l_S}$) and capacity ($C_{l_S}^{Bwd}$) of each link $l_S \in E_S$.
    SFC:
3:  Demand ($D_v$) for each VNF $v \in V_f$;
    Bandwidth demand ($D_{l_v}^{Bwd}$) for each virtual link $l_v \in E_f$.
4:  Output:
    $x_U^v$: Optimal (ordered) assignment of VNFs $v \in V_f$ to nodes $U \in V_s$.
5:  Algorithm:
6:  Initialise $t = 0$;
7:  Solve master problem (2) to optimality to give $(\hat{x}^t, \hat{y}^t)$;
8:  Solve Sub$(\hat{x}^t, \hat{y}^t)$;
9:  While sub-problem is infeasible
10:     $t = t + 1$;
11:     Find IISs of sub-problem (3) and formulate (multiple) combinatorial cuts (4);
12:     Add combinatorial cut (4) to the master problem;
13:     Solve master problem (2) subject to added combinatorial cuts to give new
    solution $(\hat{x}^t, \hat{y}^t)$;
14:     Solve Sub$(\hat{x}^t, \hat{y}^t)$;
15: End While
16: Return $(\hat{x}^t, \hat{y}^t)$.
```

3.3.5 Performance of CBD for VNF-PC

To evaluate the effectiveness of the augmented Combinatorial Benders Decomposition (CBD) approach, along with the inclusion of the greedy branching heuristic, for solving the VNF-PC problem, we implemented the model using C++ and integrated it with IBM ILOG-CPLEX version 20.1.0. The experiments were conducted on a machine with an Intel 3.00 GHz processor and 16 GB of RAM. Refer to Section 3.3.3 for insights into the formulation of combinatorial cuts.

3.3.5.1 Simulation Setup

The simulation setup for testing the CBD approach for VNF-PC, in comparison with the monolithic mixed-integer model as presented in ILP I formulation, involves two NFV infrastructures: one with 30 nodes and the other with 100 nodes. In both cases, we consider a complete graph of links ($|E_s| = |V_s| \times (|V_s|-1)/2$), focusing solely on CPU resources.

For the NFV:

- CPU costs and capacities for each node $U \in V_s$ are randomly chosen: $\alpha_U^r \in [5, 20]$, $C_U^r \in [20, 120]$.
- Bandwidth costs and capacities for each link $l_S \in E_S$ are randomly chosen: $\beta_{l_S} \in [5, 20]$, $C_{l_S}^{Bwd} = 1000$.

For the SFC:

- Size of each SFC is determined: $|V_f| \in [2, 15]$.
- CPU demand of each VNF $v \in V_f$ is randomly selected: $D_v \in [10, 20]$.
- Bandwidth demand of each link $l_v \in E_f$ is randomly chosen: $D_{l_v}^{Bwd} \in [2, 5]$.

Comparative outcomes are provided for both NFV infrastructures with varying SFC sizes ($|V_f| \in [2, 15]$).

An instance is considered infeasible for a node $U \in V_s$ if its capacity surpasses C_U^r, implying node overload. If at least one node in an instance is overloaded for any distribution of VNFs to nodes, the instance lacks a feasible solution. Additionally, we consider that each link can be overloaded ($C_{l_S}^{Bwd} = 1000$), though we set this high value to focus solely on overloaded nodes.

The range of possible CPU capacities C_U^r simulates real-world network environments, where resource availability at nodes might already be allocated for other SFCs. This facet grants realistic insights into the performance of the proposed approach.

Two key simulation metrics are addressed:

- **Solution Availability:** Determines if placement and chaining of the SFC are feasible.
- **Runtime:** Measures the algorithm's time taken to attain an optimal placement and chaining solution, if available.

For solution availability, each tested instance yielded a feasible optimal solution, benefiting from sufficiently large NFV infrastructures. For runtime assessment, an average across 20 instances was computed, with values for fixed variables being randomly drawn for each instance.

3.3.5.2 Simulation Results

We first examine the standard CBD approach for the VNF-PC problem. Given the large bandwidth capacity of links ($C_{l_S}^{Bwd} = 1000$), infeasibilities in the master problem arise predominantly from VNF allocation to nodes. Consequently, combinatorial cuts primarily adopt the form of T_1 (see Section 3.3.3), impacting solely the binary variables x_U^v (where $U \in V_s$, $v \in V_f$). Considering that these binary variables are governed by knapsack-style constraint (2b) in the master problem, the resulting combinatorial cuts only eliminate a small subset of solutions.

Hence, the incorporation of the greedy branching heuristic, as elucidated in Section 3.3.4, is indispensable to discard a larger set of infeasible solutions. This holds true even for non-integer solutions at non-viable points in the master problem.

Runtime results are graphically presented in Figure 3.5 (for $|V_s| = 30$) and Figure 3.6 (for $|V_s| = 100$), employing a logarithmic time scale. The standard CBD approach is denoted as "CBD," while the combined approach with the greedy branching heuristic is labeled as "CBD_B."

Both figures reflect that while the unmodified CBD approach is relatively inefficient, often slower than the monolithic MILP approach for larger SFC sizes, the incorporation of the branching heuristic yields noteworthy runtime improvements. Since the proposed approach weeds out infeasible solutions even at non-integer values, the subsequent traversal of the branching tree becomes more efficient.

Figure 3.5 showcases that the collaborative effect of the branching heuristic with the CBD approach performs on par with the MILP approach and occasionally outperforms it slightly. In Figure 3.6, the benefit of employing the branching heuristic becomes evident as the SFC size exceeds 7, as the approach consistently surpasses the MILP approach in runtime efficiency. Notably, for larger instances, the presented decomposition approach proves to be particularly effective as the SFC size increases. For smaller SFCs, both methods efficiently solve the problem within 10 seconds.

In Figure 3.7, we further see comparative results between the MILP and the decomposition approach with the greedy branching heuristic for the ordered VNF-PC variant. We know from Section 3.2.5 that the inclusion of the ordering constraints leads to faster solve times. Hence, we only compare for $|V_s| = 100$ and for $|V_f|$ between 5 and 15.

In this case, the faster performance of the combined decomposition approach is clear for larger SFC sizes. The ordering restrictions on the binary variables mean the cuts

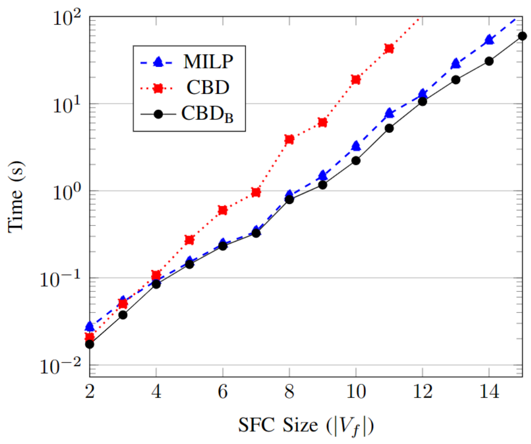


Figure 3.5: *Runtime for the three approaches:* $|V_s| = 30$.

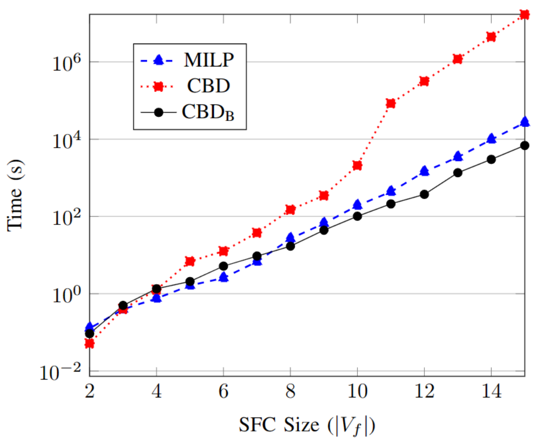


Figure 3.6: *Runtime for the three approaches:* $|V_s| = 100$.

formed from the combinatorial Benders decomposition are stronger, and remove many infeasible solutions in each iteration.

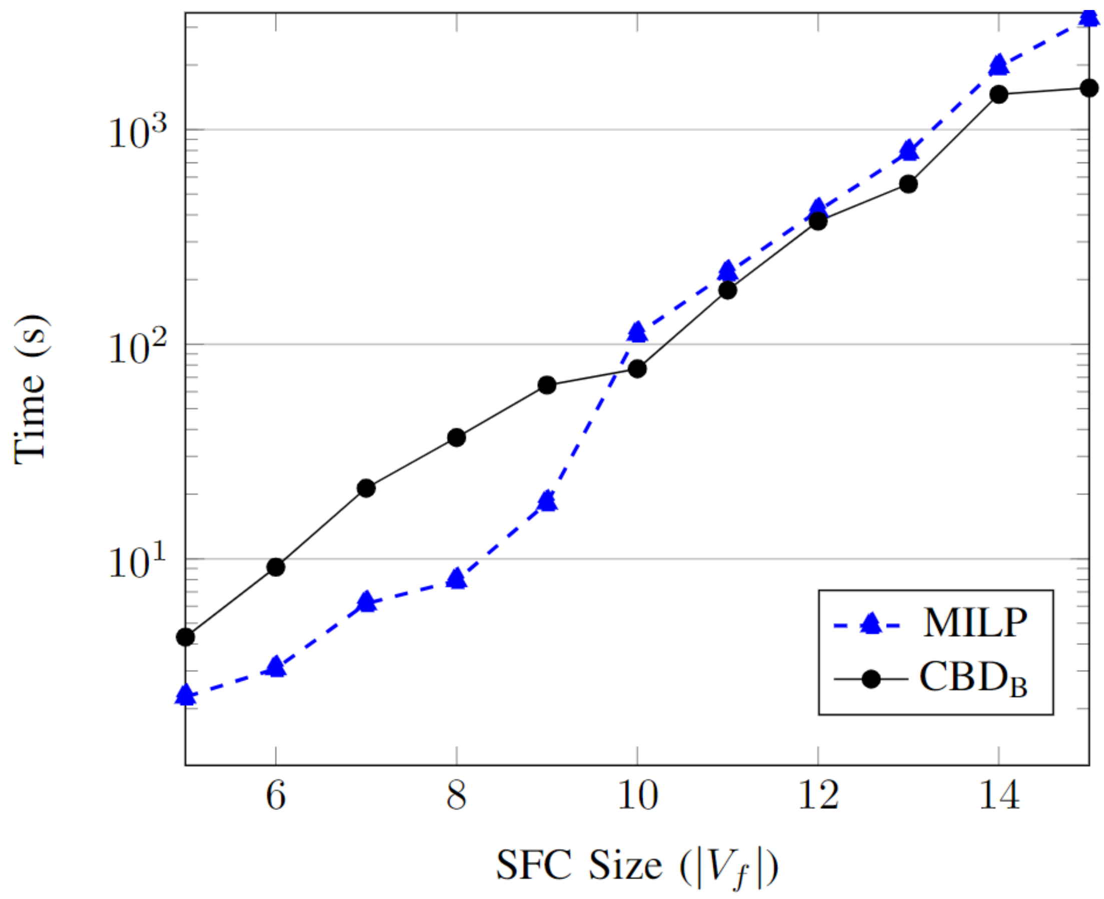


Figure 3.7: *Runtime for the two approaches for ordered VNF-PC:*$|V_s| = 100$.

3.4 Conclusion

The given chapter discusses the VNF placement and chaining problem in the context of Network Function Virtualization (NFV). The chapter introduces a mathematical formulation of the problem using Integer Linear Programming (ILP) as the optimization technique. The VNF placement and chaining problem involve determining the optimal allocation of VNFs and their interconnection order, known as Service Function Chains (SFCs). The goal is to find the most cost-effective placement and routing of VNFs in the NFV infrastructure. The chapter presents a formal representation of the problem and its constraints, serving as a foundation for developing efficient optimization algorithms and approaches. The problem formulation includes decision variables for VNF-node assignments and VNF-link embeddings. The objective function aims to minimize the total cost of VNF placement and bandwidth resource allocation.

Additionally, the chapter presents a complementary section on the application of Combinatorial Benders Decomposition (CBD) to enhance the efficiency of solving the VNF-PC problem. This approach leverages the power of IIS identification and combinatorial cuts to improve the solution process, contributing to a deeper understanding of the problem and its potential solutions. The integration of CBD into the discussion enriches the chapter's exploration of advanced techniques in tackling complex NFV optimization challenges.

Furthermore, we introduce an ordered variant of the ILP formulation, imposing constraints on the placement of VNFs according to a specified order. This ordered VNF-PC variant enhances the algorithmic efficiency by introducing constraints on binary variables. Such enhancements are expected to lead to faster solution times. This chapter concludes by highlighting the significance of the discussed techniques and approaches in addressing the challenges of VNF placement and chaining within NFV, underscoring their potential for optimizing resource allocation and cost-effectiveness in NFV infrastructure.

Chapter 4

VNF placement and chaining problem in unloaded network

Seek knowledge from the cradle to the grave.

Ibn al-Haytham

Abstract

In this chapter, we explore a relaxed version of the VNF placement and chaining problem in an unloaded network environment. We assume abundant resources on nodes and links within the NFV infrastructure, aiming to minimize the overall placement cost associated with Service Function Chains (SFCs). By disregarding resource constraints, we analyze cost optimization aspects of VNF placement, providing valuable insights into cost-efficient strategies for NFV deployments in resource-rich networks.

[11] Issam Abdeldjalil Ikhelef et al. “Multi-Constrained Routing-based Heuristic for VNF Placement and Chaining”. In: *ICC 2023 - IEEE International Conference on Communications*. 2023

Chapter content

4.1 Introduction

In this chapter, we delve into a relaxed version of the VNF placement and chaining problem, specifically focusing on an unloaded network environment. In an unloaded network, there exists an abundance of resources available on both nodes and links within the NFV infrastructure (NFV-I), enabling the provisioning of Service Function Chains (SFCs) using any components.

In this relaxed scenario, we make a critical assumption that resource capacities on nodes and links are significantly large, more than enough to accommodate the demands of the SFCs. Within this context, our primary objective is to minimize the overall placement cost associated with the deployment and execution of SFCs. This cost encompasses the resource expenses incurred on nodes and links utilized during the provisioning process.

By exploring the placement and chaining of VNFs in an unloaded network with ample resources, we can analyze and optimize cost-related aspects of placement decisions, such as resource utilization efficiency and overall cost minimization. The goal is to achieve cost-effective SFC provisioning in an environment where resource constraints are not a limiting factor.

This relaxed version of the VNF-PC problem offers valuable insights into cost optimization in the placement process while disregarding resource limitations. These insights serve as a foundation for understanding the cost implications of different placement strategies and can guide decision-making in NFV-I design and deployment.

4.2 VNF placement and chaining problem in an unloaded network

In this section, the focus is on a relaxed version of the VNF-PC problem, specifically in the context of an unloaded network. An unloaded network refers to a network where there is an abundance of resources available on nodes and links, allowing for the provisioning of Service Function Chains (SFCs) using any components of the NFV infrastructure (NFV-I).

In this relaxed version, an important assumption is made that the resource capacities of both nodes and links are significantly large and sufficient to accommodate the demands of the SFCs. With this assumption, the objective is to minimize the overall placement cost associated with provisioning the SFCs. This cost includes the resource costs of the nodes and links that are utilized for the deployment and execution of the SFCs.

By considering an unloaded network with abundant resources, it becomes possible to explore the placement and chaining of VNFs without being constrained by resource limitations. This allows for the analysis of cost aspects associated with the placement decisions, such as the efficient utilization of resources and the optimization of overall placement cost. By minimizing the placement cost, the goal is to achieve cost-effective provisioning of SFCs in an environment where resource availability is not a limiting factor.

The relaxed version of the VNF-PC problem provides valuable insights into the cost optimization aspects of the placement process while disregarding resource constraints. This analysis serves as a basis for understanding the cost implications of different placement strategies and can help inform decision-making in the design and deployment of NFV-I components.

By studying the relaxed version of the problem, researchers and practitioners can gain a better understanding of the cost-related considerations involved in VNF placement. This understanding can guide the development of more efficient and cost-effective placement algorithms and policies, which can ultimately contribute to the optimization of NFV deployments in networks with abundant resources.

4.2.1 NFV-I and SFC modeling and problem definition

In this subsection, we focus on the modeling and problem definition of the NFV-I and SFC in the context of the relaxed version of the VNF-PC problem.

As presented in section 3.2.4, the substrate network NFV-I is modeled as an undirected weighted graph, denoted as $G_S = (V_S, E_S)$, where V_S represents the set of nodes corresponding to NFV-I servers and switches, and E_S represents the set of virtual links.

To simplify the modeling, we assume that the CPU is the only resource available for substrate nodes. Each substrate node U is associated with a CPU cost and a residual capacity, denoted respectively as α_U and C_U. In the case of an NFV unloaded network, where there is an abundance of resources, we assume that the capacity of each node is infinite, i.e., $\forall U \in V_S : C_U = \infty$.

Similarly, each virtual link $l_S = (l_S^a, l_S^b)$ in the substrate network has a residual bandwidth capacity denoted as $C_{l_S}^{Bwd}$ and a bandwidth cost denoted as β_{l_S}. In the context of an NFV unloaded network, where resources are abundant, the bandwidth capacities on links are assumed to be infinite.

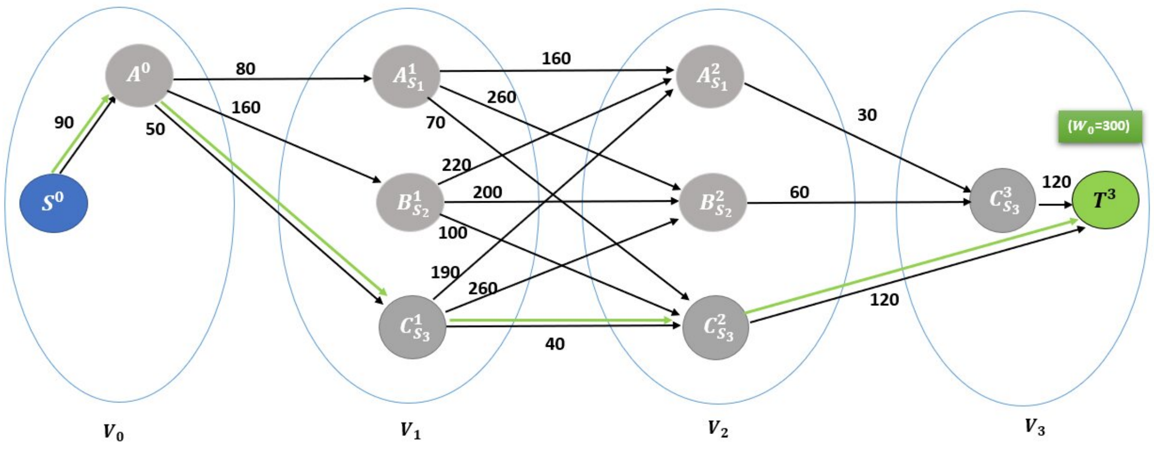


Figure 4.1: *SFC placement and chaining using Shortest Paths Algorithm.*

The chain of virtual functions, known as the Service Function Chain (SFC), is modeled as a graph $G_f = (U_{Src}, U_{Trg}, V_f, E_f)$. The source node of the SFC is denoted as U_{Src}, which is provisioned by the NFV-I switch Src, and the target node is denoted as U_{Trg}, provisioned by the NFV-I switch Trg. The SFC graph interconnects the different VNFs that compose the SFC, with V_f representing the set of VNFs and E_f representing the set of links between the VNFs. Each VNF and link in G_f have specific resource demands to ensure the proper functioning of the associated service. The demand of VNF VNF_i is denoted as d_i, and the demand of a link (VNF_i, VNF_{i+1}) in the SFC is denoted as b_{i+1}.

To simplify notation, we use the notation VNF_0 to represent the source node U_{Src} and $VNF_{|V_f|+1}$ to represent the target node U_{Trg}.

To facilitate comprehension, we will begin by elucidating the graph transformation process in the following subsection (Section 4.2.2). This transformation simplifies the resolution of the VNF-PC problem's relaxed version, which assumes ample and extensive NFV-I resources. Subsequently, in Chapter 6, we will introduce and elucidate the algorithm designed to address the generalized version of the VNF-PC problem. This algorithm incorporates the constraints pertaining to node and link resources into the transformed graph, enabling a comprehensive solution.

4.2.2 Graph transformation to solve the relaxed version of VNF placement and chaining problem

To tackle the relaxed version of the VNF-PC problem, we introduce a novel k-partite directed graph denoted as $G_p = (V_p, E_p)$, which is derived from the NFV-I and SFC graphs. This transformation process is illustrated in Figure 4.1 and encompasses the following key steps:

1. For each VNF$_i$ in the SFC, we establish a corresponding set V_i $(0 < i \leq k)$ comprising server nodes capable of hosting that particular VNF. In the case of an SFC with k VNFs, k sets $(V_i,\ 0 < i \leq k)$ are defined, along with $V_0 = \{src_0\}$ and $V_{k+1} = \{t_{k+1}\}$.

2. We create connections between nodes u_i in V_i $(0 \leq i \leq k)$ and nodes v_{i+1} in V_{i+1} $(u_i \neq v_{i+1})$ if and only if the corresponding edge $(u,\, v) \in E_s$ exists in the original graphs. To compute the cost $\theta_{(u_i,v_{i+1})}$ of the link $(u_i,\, v_{i+1})$, we use the following equation:

$$\theta_{(u_i,v_{i+1})} = \alpha_v \times d_{i+1} + \beta_{(u,v)} \times b_{i+1} \tag{4.1}$$

Here, $\theta_{(u_i,v_{i+1})}$ encompasses the cost of mapping VNF$_i$ and VNF$_{i+1}$ onto server nodes u and v, as well as the cost of the path between servers u and v used to accommodate the data flow between VNF$_i$ and VNF$_{i+1}$.

In Figure 4.1, we provide a visual representation of the resulting 3-partite graph derived from the NFV-I (Figure 3.1) and SFC (Figure 3.2). This example demonstrates the process as follows:

- Assuming that all the nodes can support the SFC VNFs, we define $V_{i\in\{0,1,2,3\}}$ as follows:

$V_0 = \{S^0, A^0\}$, $V_1 = \{A^1_{S_1}, B^1_{S_2}, C^1_{S_3}\}$
$V_1 = \{A^2_{S_1}, B^2_{S_2}, C^2_{S_3}\}$, $V_3 = \{C^3_{S_3}, T^3\}$

- Given that switch A connects to switch B and C and server S_1, we introduce the following links into the 3-partite graph:

$(S^0,\, A^0)$, $(A^0,\, A^1_{S_1})$, $(A^0,\, B^1_{S_2})$, and $(A^0,\, C^1_{S_3})$.

Similarly, we incorporate the following links into the 3-partite graph:

$(A^1_{S_1},\, B^2_{S_2})$, $(A^1_{S_1},\, A^2_{S_1})$, $(A^1_{S_1},\, C^2_{S_3})$, $(B^1_{S_2},\, B^2_{S_2})$, $(B^1_{S_2},\, A^2_{S_1})$, $(B^1_{S_2},\, C^2_{S_3})$, $(C^1_{S_3},\, B^2_{S_2})$, $(C^1_{S_3},\, A^2_{S_1})$, $(C^1_{S_3},\, C^2_{S_3})$, $(A^2_{S_1},\, C^3_{S_3})$, $(B^3_{S_2},\, C^3_{S_3})$, $(C^2_{S_3},\, T^3)$ and $(C^3_{S_3},\, T^3)$.

- We compute the link costs within the 3-partite graph using equation 4.1, and these values are represented as W_0 in Figure 4.1.

Upon completing the graph transformation, the optimal mapping of the VNFs is achieved by identifying the shortest path connecting the NFV-I source node S^0 to the NFV-I target node T^{k+1} within the k-partite graph. To illustrate this process using the example in Figures 3.1 and 3.2, the optimal mapping is determined by the shortest path (S^0, A^0, $C^1_{S_3}$, $C^2_{S_3}$, T^3). Specifically, *VNF*$_1$ and *VNF*$_2$ are assigned to *server* S_3 . The overall cost of this solution amounts to 300 as calculated below:

- $w_0(S^0, A^0)= \theta_{(S^0,A^0)} = 3 \times 30 = 90.$
- $w_0(A^0, C^1_{S_3}) = \theta_{(A^0,C^1_{S_3})} = 1 \times 30 + 1 \times 20 = 50.$
- $w_0(C^1_{S_3}, C^2_{S_3}) = \theta_{(C^1_{S_3},C^2_{S_3})} = 1 \times 40 = 40.$
- $w_0(C^2_{S_3}, T^3)) = \theta_{(C^2_{S_3},T^3))} = 4 \times 30 = 120.$

Lemma 4.1. *The optimal solution for the relaxed version of the VNFPC problem corresponds to the shortest path* $\pi = (src_0, a_1, b_2, c_3, .., t_{k+1})$ *in the corresponding k-partite graph. The VNFs VNF*$_1$*, VNF*$_2$*, VNF*$_3$*, .. are placed on the nodes a, b, c, ..*

Proof. The validity of the preceding lemma is evident and can be verified by contradiction. Assuming the existence of a VNF placement with a lower cost than that obtained with the shortest path in the k-partite graph, it is straightforward to demonstrate that such an assumption leads to a contradiction. (Proof complete) □

4.3 Conclusion

This chapter has delved into the relaxed version of the VNF placement and chaining problem within unloaded network conditions, where resource constraints are not a limiting factor.

By focusing on cost optimization aspects and disregarding resource limitations, we've gained valuable insights into efficient VNF placement strategies. These insights lay the groundwork for cost-effective NFV deployments in resource-rich network environments. The transformation of NFV-I and SFC into a k-partite graph provides a powerful approach for analyzing and optimizing VNF placement and chaining, as supported by the presented lemma.

This understanding of cost-related considerations can guide the development of more efficient placement algorithms and policies, ultimately contributing to the optimization of VNF deployments.

Chapter 5

VNF placement and chaining problem in high bandwidth network

Il va falloir choisir, dans un avenir plus ou moins proche, entre le suicide collectif ou l'utilisation intelligente des conquêtes scientifiques.

Albert Camus

Abstract

In this chapter, we address the VNF-PC problem in networks with limited node resources but abundant link resources. We propose a knapsack-based genetic algorithm that combines knapsack optimization principles with genetic algorithms to optimize resource allocation and minimize costs. Benefiting from the work [9], the objective is to minimize the overall placement cost, considering ample resource capacities for nodes and links.

[9] Issam Abdeldjalil Ikhelef et al. "A Knapsack-based Optimization Algorithm for VNF Placement and Chaining Problem". In: *2022 IEEE 47th Conference on Local Computer Networks (LCN)*. 2022, pp. 430–437. DOI: 10.1109/LCN53696.2022.9843566

Chapter content

5.1 Introduction

In this chapter, our focus is on a specific variation of the VNF-PC problem, which pertains to networks with limited node resources but abundant link resources. To address this challenge, we propose a knapsack-based genetic algorithm that combines knapsack optimization principles with genetic algorithms to optimize resource allocation and minimize costs.

Inspired by the classical knapsack problem, our algorithmic approach selects and places VNFs based on their weights and values to maximize overall value while adhering to node resource constraints. By employing a genetic algorithm, we incorporate genetic operators like selection, crossover, and mutation to iteratively refine candidate solutions. This allows us to explore diverse configurations and identify placements that minimize costs effectively.

The integration of the knapsack-based genetic algorithm enhances resource allocation efficiency and cost minimization objectives. It provides a systematic and evolutionary framework for finding optimized solutions, specifically tailored to networks with limited node resources and abundant link resources.

By introducing this novel algorithmic approach, this section contributes significantly to solving the VNF-PC problem in networks characterized by limited node resources and abundant link resources.

5.2 VNF placement and chaining problem in high band- width network

As presented in section 3.2.4, we model the NFV-I as an undirected weighted graph, noted $G_S = (V_S, E_S)$ where V_S is the set of nodes corresponding to NVF-I servers and switches, and E_S represents the set of virtual links.

For ease of understanding, we suppose that the CPU is the only resource available for substrate nodes. Each substrate node U is associated with a CPU cost and a residual capacity denoted respectively α_U and C_U. For NFV unloaded network, we assume that: $\forall U \in V_S : C_U = \infty$.

Similarly, each virtual link $l_S = (l_S^a, l_S^b)$ has a residual bandwidth capacity denoted $C_{l_S}^{Bwd}$, and a bandwidth cost denoted β_{l_S}. For NFV unloaded network, the bandwidth capacities on links are assumed infinite.

The chain of virtual functions (SFC) is also modeled by a graph $G_f = (U_{Src}, U_{Trg}, V_f, E_f)$, interconnecting its source node denoted U_{Src} (that is provisioned by the NFV-I switch

Src), to its target node denoted U_{Trg} (that is provisioned by the NFV-I switch Trg) by a path crossing the different VNFs that compose the SFC (see Figure 3.2). V_f corresponds to the set of VNFs while the set of links interconnecting the VNFs is denoted E_f. VNFs and links of G_f require certain amounts of resources to ensure the proper functioning of the service associated with the SFC.
The VNF demand of VNF_i is denoted d_i whereas the demand of a link (VNF_i, VNF_{i+1}) in the SFC corresponds to b_{i+1}.

To facilitate understanding and simplify notations, we denote by VNF_0 and $VNF_{|V_f|+1}$ respectively the source node U_{Src} and the the target node U_{Trg}.

In this section, we address the relaxed version of the VNF Placement and Chaining problem (RVNF-PC) by assuming abundant bandwidth on links and limited resources on nodes. Specifically, we focus on the CPU resource to simplify the problem, removing the $r \in R$ variables from the general model.

To facilitate understanding, we first solve RVNF-PC under the assumption of negligible bandwidth costs, which implies that bandwidth is superabundant and inexpensive. We then generalize our results by considering non-negligible bandwidth costs.

For an NFV Infrastructure (NFV-I) with negligible link cost, the objective function of the Integer Linear Programming (ILP) formulation I (Equation 3.1) reduces to the term A, which represents the overall cost of placing VNF instances:

$$Z = \text{Minimize} \quad A = \text{Minimize} \sum_{U \in V_S} \sum_{v \in V_f} x_U^v \times D_v \times \alpha_U \tag{5.1}$$

Here, α_U denotes the unitary CPU cost on node $U \in V_S$. The following constraints are enforced:

$$\sum_{v \in V_f} x_U^v \times D_v \leq C_U \qquad \forall U \in V_S \tag{5.2}$$

$$\sum_{U \in V_S} x_U^v = 1 \qquad \forall v \in V_f \tag{5.3}$$

Constraint (5.2) guarantees that the CPU resource demand D_v is satisfied within the residual capacity C_U of each node of the NFV-I. Constraint (5.3) guarantees that each VNF $v \in V_f$ is placed in a single node of the NFV-I.

We can show that the solution to this problem corresponds to a variant of the Multiple Knapsack Problem (MKP), where all objects must be placed in the knapsacks.

This can be proven by deriving a new ILP formulation II from ILP I, which is essentially the same as the one used to solve the multiple knapsack problem. This transformation involves modifying the objective function (5.1) as follows:

$$\text{Maximize} \quad C(x) = \text{Maximize} \sum_{U \in V_S} \sum_{V \in V_f} x_U^v (\eta - D_v \times \alpha_U) \tag{5.4}$$

Here, η is a high constant satisfying:

$$\eta \gg \sum_{U \in V_S} \sum_{v \in V_f} D_v \times \alpha_U$$

We note that the objective functions (5.1) and (5.4) are equivalent, as shown by:

$$\begin{aligned} \sum_{v \in V_f} \sum_{U \in V_S} \eta \times x_U^v &= \eta \times \sum_{v \in V_f} \left(\sum_{U \in V_S} x_U^v \right) \\ &= \eta \times \sum_{v \in V_f} (1) = \eta \times |V_f| \end{aligned} \tag{5.5}$$

Thus, minimizing the objective function (5.4) is equivalent to maximizing the non-constant part of the objective (equivalent to objective function (5.1)). In other words:

$$\text{Maximize} \quad - \sum_{U \in V_S} \sum_{v \in V_f} x_U^V \times D_v \times \alpha_U \tag{5.6}$$

Note that ILP II can be transformed into the generic version of the Multiple Knapsack Problem by relaxing constraint 5.3 (i.e., $\sum_{U \in V_S} x_U^v \leq 1, \forall v \in V_f$). In this case, ILP I is solvable if and only if the solution of ILP II satisfies constraint 5.3.

In terms of interpretation, we can say that the RVNF-PC problem can be transformed into an instance of the Multiple Knapsack Problem (see Figure 5.1), where VNFs correspond to the objects and the servers correspond to the knapsacks. Specifically:

- The set of nodes in the NFV-I is modeled as the set of knapsacks, denoted by $M = \{1, ..., m\}$.
- The Service Function Chain (SFC) composed of multiple VNFs is modeled as the set of objects, denoted by $N = \{1, ..., n\}$.

- Each VNF $v_i \in V_f$ has a CPU resource demand denoted by D_v, equivalent to the weight W_i of the corresponding object i in the MKP.
- Placing VNF $v_i \in V_f$ on node $U_j \in V_S$ incurs a cost of $(\eta - D_v \times \alpha_U)$, equivalent to the profit P_i^j of placing object i in knapsack j.
- Each node $U_j \in V_S$ has a maximum CPU resource capacity denoted by C_{U_j}, which corresponds to the constant capacity of knapsack j denoted by C_j in the MKP.
- The objective is to minimize the total allocation cost of VNFs, which is equivalent to maximizing the profit (or negative costs) of the placed objects.

In the case where the costs of interconnecting paths between VNF servers are equal to or much higher than the CPU costs, the SFC optimal placement and chaining with equal bandwidth demands can be obtained by solving the NP-hard knapsack problem when all CPU costs are identical.

5.3 Knapsack Problem (KP)

The classical knapsack problem (KP) is a combinatorial optimization problem that involves a single constraint on the solution objects. It is considered a challenging problem due to its complexity and classification as an NP-hard problem. In real-life scenarios, the knapsack problem can take various forms. For instance, imagine a mountaineer

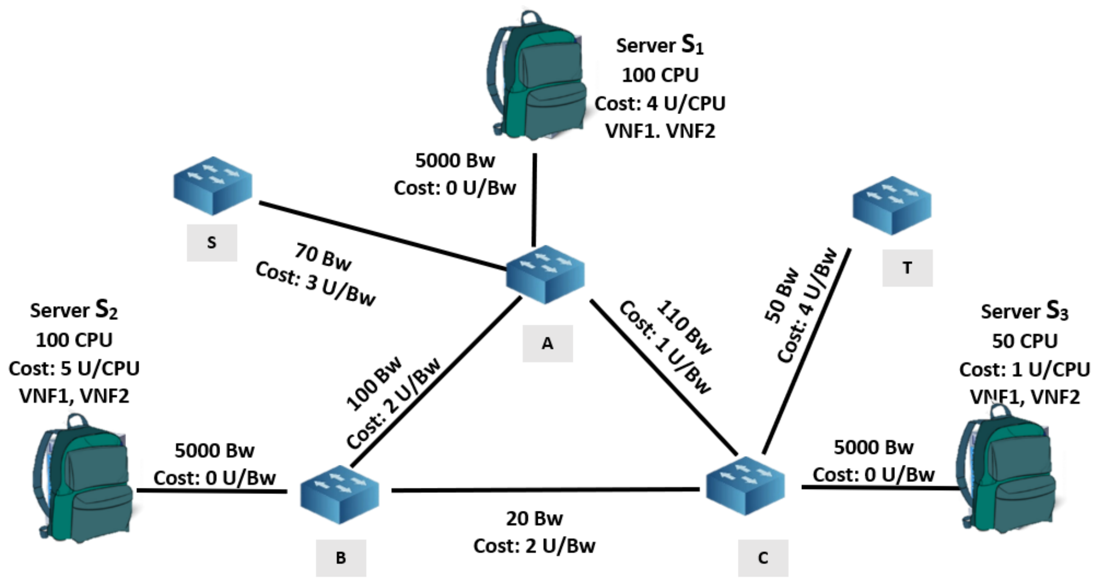


Figure 5.1: *MKP adaption of VNF-PC.*

who carries a knapsack and has a list of objects that they can potentially include in the knapsack. Each object provides a certain level of comfort to the mountaineer while occupying a specific amount of space. However, the capacity of the knapsack is limited. Therefore, the mountaineer's objective is to maximize comfort while ensuring that the total capacity of the knapsack is not exceeded. Another example is the investment problem, where there is a fixed budget and multiple projects to choose from. Each project is characterized by a profit and an investment cost. The optimal investment decision can be determined by solving a knapsack-type problem [70]. In general, the knapsack problem involves filling a bag with a subset of objects, considering their weights and profits, while satisfying two conditions:

- The total weight of the selected subset must not exceed the capacity of the knapsack.
- The profit generated by the selected subset should be maximized.

The knapsack problem has many applications and often serves as a subproblem in solving other problems. It has been extensively studied by researchers and various formulations and solution techniques have been proposed.

5.3.1 The Multidimensional Knapsack Problem in 0-1 Variables (MKP)

The Multidimensional Knapsack Problem in 0-1 Variables (MKP), also known as the multi-constrained knapsack or multiple knapsack problem, is a generalization of the KP problem where there are more than one capacity constraint.

The MKP exhibits distinct characteristics compared to other knapsack problems. Firstly, the associated matrix in MKP is typically dense. Secondly, obtaining a feasible solution for MKP is relatively straightforward. By setting all variables of MKP to 0, a solution with a value of 0 can be obtained. Several researchers have studied the MKP, including [71] and [72].

The Multiple Knapsack Problem (MKP) falls under the category of NP-hard problems, which means that finding an optimal solution can be computationally challenging and often requires significant computational resources when using traditional numerical methods [73]. When the number of capacity constraints, denoted as m, is equal to 1, the MKP reduces to the knapsack problem. The MKP can be modeled as follows:

MKP:

$$\max Z = \sum_{j=1}^{n} x_j \cdot p_j$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{j=1}^{n} x_i \cdot w_{ij} \leq c_i \quad && \text{for} \quad i = 1, \ldots, m. \\ & x_j \in \{0, 1\} && \text{for} \quad j = 1, \ldots, n. \end{aligned}$$

with p_j, w_{ij}, c_i positive integers, (for j = 1, . . . , n and for i = 1, . . . , m) The instances of the MKP which exist in the literature contain only very few constraints. However, their resolution remains quite difficult for optimization software. For example, instances that include 10 constraints and 500 variables are not solved optimally in a reasonable time by optimization software.

5.4 Genetic algorithms based meta-heuristic

5.4.1 Definition of Genetic algorithms

A genetic algorithm (GA) is a powerful and versatile search and optimization technique based on the principles of natural selection and genetics. It is a type of evolutionary algorithm that mimics the process of biological evolution to find approximate solutions to complex optimization problems [74].

The main idea behind a genetic algorithm is to model the problem-solving process as an evolutionary process, where a population of potential solutions evolves over successive generations to improve the quality of the solutions. This population of individuals represents a pool of candidate solutions, often encoded as strings of binary digits or other data structures [75]. Genetic algorithms (GAs) are optimization algorithms inspired by the principles of natural selection and evolution. They operate on a population of individuals, each representing a potential solution to the problem at hand. The operations in a genetic algorithm include:

i. **Initialization:** A population of individuals is randomly generated as the initial set of solutions. These individuals are usually encoded as binary digits strings or other representations suitable for the problem domain.

ii. **Fitness Evaluation:** Each individual's fitness is evaluated, which represents its quality or suitability as a solution. The fitness function measures how well an individual solves the problem and determines its likelihood of being selected for reproduction.

iii. **Selection:** Individuals with higher fitness values are more likely to be selected for reproduction. The selection process, often implemented through mechanisms like roulette wheel selection or tournament selection, favors individuals with better fitness and promotes the propagation of their genetic material.

iv. **Crossover:** Crossover, also known as recombination, involves combining genetic information from two parent individuals to create offspring. This operation mimics genetic recombination in natural reproduction. Crossover points are chosen in the parent individuals, and portions of their genetic material are exchanged to create new offspring.

v. **Mutation:** Mutation introduces small random changes in the genetic material of individuals. It helps to maintain diversity in the population and prevents the algorithm from converging too quickly to a suboptimal solution. Randomly selected positions in an individual's genetic representation are altered with low probability.

vi. **Offspring Generation:** The offspring generated through crossover and mutation operations replace a portion of the existing population. This allows for exploration of new areas of the solution space while retaining promising individuals.

vii. **Termination Criteria:** The algorithm continues to iterate through the selection, crossover, and mutation operations until a termination condition is met. Common termination criteria include reaching a maximum number of iterations, achieving a satisfactory fitness level, or stagnation of improvement over successive generations.

By iteratively applying these operations, genetic algorithms explore the solution space and adaptively search for optimal or near-optimal solutions. The genetic information of fitter individuals is propagated to future generations, leading to the evolution of the population towards better solutions [76].

To tackle the NP-hard RVNF-PC problem, we propose the use of genetic algorithms (GA), known for their effectiveness in solving various knapsack problem variants. GAs enable us to find near-optimal solutions from a set of feasible solutions by employing a fitness function. The GA operates as follows.

5.4.2 Initial Population

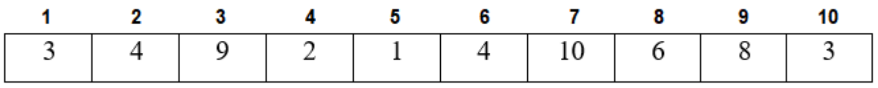
We begin with an initial population that promotes diversification and generates new solutions through crossover and mutation operations. Selecting an initial population comprising individuals with diverse and varied genes is crucial for exploring a wide range of promising regions within the solution space. Including individuals with higher-quality genes in the initial population often leads to swift convergence towards the best solutions. In our approach, the following processes is used to generate the initial population:

- **Random Generation:** For each SFC of n VNFs, a list of n randomly selected servers is generated and evaluated based on the fitness function.

- **Constrained Shortest Paths:** For an SFC composed of n VNFs, a shortest path of n links connecting the SFC's source to a neighbor of the SFC's destination is determined. The path calculation stage verifies the constraints, ensuring that only paths satisfying the constraints are added to the initial population. We include zero-cost reflexive links with unlimited capacities in the NFV-I to facilitate the deployment of multiple successive VNFs on the same server. To guarantee an adequate individuals number in the initial population, random solutions generated using the previous process are also included.

5.4.3 Coding

The coding process transforms the real data of the problem into a format suitable for GAs. In our approach, we employ a coding scheme using natural numbers to represent server indices. Each VNF placement solution is encoded as an individual represented by an array, where the indices correspond to the identifiers of the VNFs and the cell contents represent the servers on which the VNFs are deployed. For example, an SFC with 10 VNFs and an NFV-I with 10 servers would have a solution coding as shown in Figure 5.2, where *VNF1* and *VNF10* are placed on *Server 3*, while *VNF3* is placed on *Server 9*.



1	2	3	4	5	6	7	8	9	10
3	4	9	2	1	4	10	6	8	3

Figure 5.2: *Chromosome coding example*

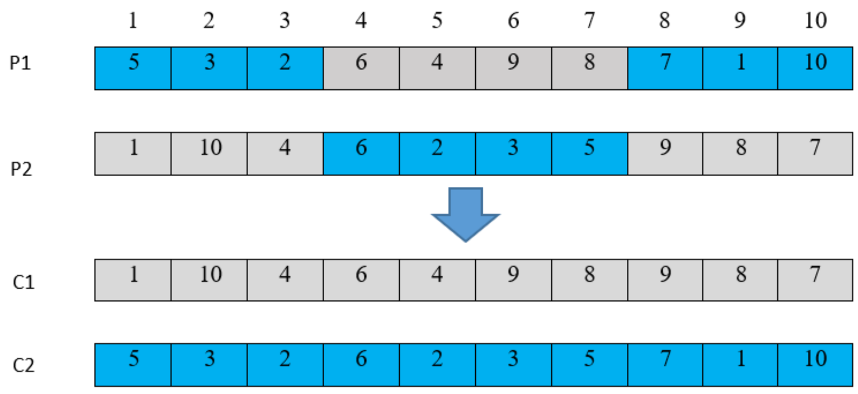


Figure 5.3: *Crossover example*

5.4.4 Fitness Function

The fitness function evaluates the quality of individuals and guides the breeding process. Individuals with higher fitness values are more likely to be selected for breeding. In our GAs, the fitness is determined as follows:

- For feasible solutions:

$$fitness = cost^{-(1+\frac{index}{nb\_generations})}$$

  Here, *index* represents the number of times the current population is regenerated (ranging from 1 to $nb\_generations$), $nb\_generations$ denotes the total number of population regenerations, and $cost = A + B$ is determined based on equations (3.2) and (3.3). This fitness function increases the probability that an individual's genes are derived from the best individuals (i.e., those with the smallest costs) over time.

- For non-feasible solutions: To prevent the algorithm from converging prematurely without exploring promising regions in the solution space, particularly when the initial population does not contain any feasible solutions, very small fitness values are assigned to non-feasible solutions. For breeding, non-feasible solutions that violate fewer constraints are preferred. In our proposal, we iterate through an individual's genes in the same order and count the number of times ($nb\_violate$) constraints are violated:

$$fitness = \frac{\epsilon}{nb\_violate}$$

  Here, ϵ is a very small constant.

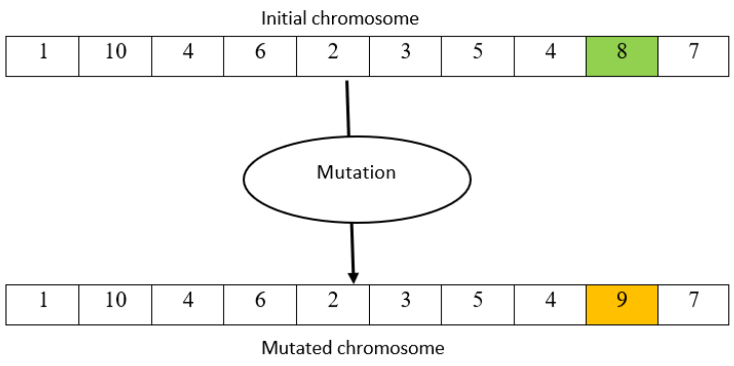


Figure 5.4: *Mutation example*

5.4.5 Genetic Operators

The population reproduction process generates a new population $i+1$ from the previous population i. This process involves selection, crossover, and mutation operations:

- **Selection:** The selection process determines which individuals are more likely to produce the best results. Our algorithms utilize two selection methods:

   i. tournament selection, where parents and children are randomly paired, and the best individuals are selected in pairs,

   ii. selection of the best individuals, where the next population is composed of individuals with the highest fitness.

- **Crossover:** Crossover is a reproductive process that allows the exchange of genetic information between individuals. Using two parents it produces one or two children.
  In our proposal, parents are selected using the *Roulette Wheel Selection*, where the probability of choosing an individual for breeding in the next generation is proportional to its fitness. For generating children, we use 2-point crossover, where two points are randomly selected among the genes. Figure 5.3 illustrates an example of crossover, where genes between the two crossover points in parents P1 and P2 are exchanged to produce two new children, C1 and C2.

- **Mutation:** The mutation process randomly modifies the value of a component in an individual with a certain probability. In our solution, we randomly select a gene and replace the server identifier S with another randomly selected server identifier S' from the list of servers. Figure 5.4 illustrates an example of mutation.

5.5 Performance Evaluation

This section provides an overview of the compared algorithms, simulation environment, performance metrics, and simulation results.

5.5.1 Compared Algorithms

The simulation evaluates the following algorithms :

- **GA-RB**: Genetic Algorithm with random generation of the initial population and selection of the next generation based on the best individuals among children and parents.
- **GA-RT**: Genetic Algorithm with random generation of the initial population and selection of the next generation through a 2-to-2 tournament between an individual from different parents and an individual from the children. Pairing of parents and children is done randomly.
- **GA-CB**: Similar to GA-RB, but the initial population consists of Shortest Constrained Paths to the destinations' neighbors. Additional randomly generated mapping solutions are included to ensure an adequate number of individuals in the population.
- **GA-CT**: Similar to GA-RT, but the initial population consists of Shortest Constrained Paths to the destinations' neighbors.
- **CSP**: Constrained Shortest Paths between the sources and destinations based on cost. Constraints are checked during the computation to ensure that the determined paths satisfy them. Each node is associated with a constraint vector, and a modified version of the Ford-Bellman algorithm is used to find the best paths satisfying the constraints.
- **ES**: Exhaustive Search to determine the solution with the minimum cost.

The population size is set to 100 individuals, and the mutation probability is 0.01. The simulation time is 10^5 units.

5.5.2 Simulation Environment

SFC requests follow a Poisson process with an arrival rate of λ requests per time unit. The lifetime of each request follows a uniform distribution $U(10, 20)$. The simulation is performed on two NFV-Infrastructures: one with 30 nodes and the other with 100 nodes. The available CPU capacity per physical node is randomly selected within the range of 20 to 120 units, and the available bandwidth capacity per link is set to 1000 units. The requested CPU capacity of each VNF is randomly chosen from the range of (10, 20), and the requested bandwidths are randomly selected from the interval (2, 5). The resource costs α and β are randomly generated within the range of (5, 20).

5.5.3 Performance Metrics

This subsection defines the simulation metrics used for performance evaluation and comparison.

- **Mean SFC Cost (MSC):** The Mean SFC Cost (MSC) represents the average cost of successfully placed SFCs. It is calculated as the ratio of the total cost of CPU and bandwidth resources allocated for SFCs to the total number of accepted SFCs.
- **Ratio of Accepted Requests (RA):** The Ratio of Accepted Requests (RA) is the ratio of the number of accepted SFC requests to the total number of received SFC requests.

5.5.4 Simulation Results

The simulation compares the proposed algorithms to ES and CSP for different sizes of SFCs and NFV-Is. The first experiment involves small SFCs and NFV-Is, while the second experiment includes medium and large SFCs and NFV-Is.

In the first experiment, the performance of the four variants of GAs is compared to ES and CSP for SFCs with varying numbers of VNFs ($n \in (2,6)$) and an NFV-I with 30 nodes. The results, shown in Figure 5.5, indicate that the compared algorithms have similar performance. Although CSP exhibits slightly lower RA compared to GAs and ES for SFCs with 6 or 5 VNFs (Figure 5.5a), the differences are negligible and indistinguishable for other scenarios.

While ES guarantees optimal solutions, the probability of CSP to determine solutions decreases as the SFC sizes increase. CSP maintains only one path per node during computations, which reduces the probability of finding the best solutions as the SFC sizes and path lengths increase. For example, CSP returns optimal solutions for SFCs with 2 VNFs.

To further examine the behavior of the algorithms, the second experiment explores various network loads using an NFV-I with 30 nodes and randomly generated SFCs with 6 VNFs. The results in Figure 5.6 demonstrate that the GA variants have performance similar to ES, which is slightly better than CSP in terms of RA. As the network load increases, the ratio of accepted requests decreases, and CSP exhibits performance close to that of ES and GAs. This can be attributed to CSP's limited exploration of promising regions in the solution space and its tendency to reuse the same path segments, potentially leading to network congestion and decreased solution determination for future requests.

Regarding MSC, the compared algorithms achieve similar results, although the determined solutions may differ. GAs construct solutions by combining path segments of the best solutions, similar to CSP which maintains one best path segment per node. Consequently, both algorithms yield nearly optimal solutions, although CSP may not explore certain promising parts of the solution space.

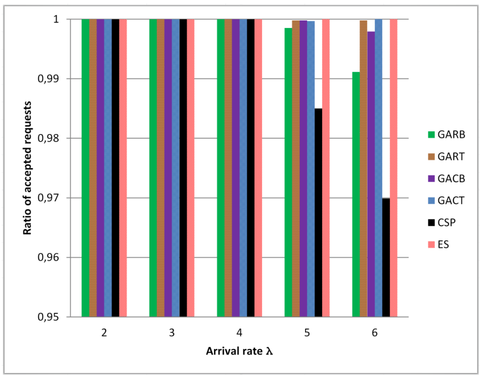
The first two experiments demonstrate that GAs and ES have comparable performance, slightly outperforming CSP, particularly in terms of RA. To provide a comprehensive comparison and gain further insights into the behavior of the GA variants, a third experiment is conducted with larger SFCs (size in (10, 20)) and NFV-Is consisting of 100 nodes. The results, presented in Figure 5.7, indicate that the GA variants GACB and GACT outperform the other algorithms, including CSP, GARB, and GART.

In terms of RA (Figure 5.7a), all GA variants surpass CSP, indicating that diversification and increased utilization of path segments improve RA to some extent.

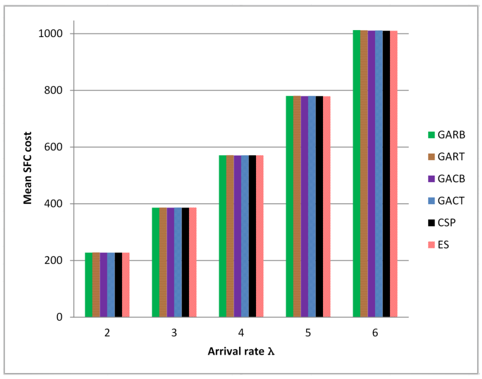
Regarding MSC, Figure 5.7b shows that the GA variants GACB and GACT achieve lower mean SFC costs compared to the other algorithms.

While the GA variants with random paths yield slightly higher SFC costs compared to CSP due to the small population size and limited computation time, it is important to note that CSP rejects larger SFCs, resulting in a lower RA compared to GAs.

The choice of the initial population has a significant impact on the performance of GAs, while the selection process for the next population has negligible effects on the overall performance. Additionally, incorporating constrained shortest paths in the initial population can accelerate convergence and improve performance.

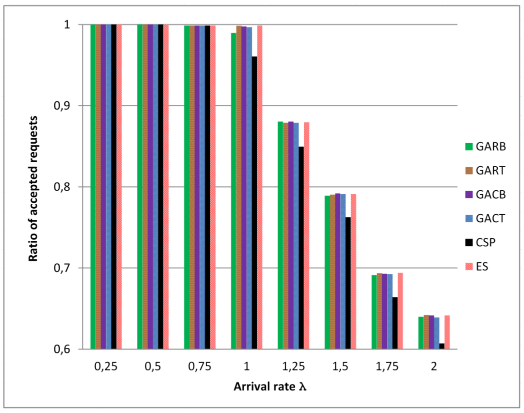


(a) *Ratio of acceptation*

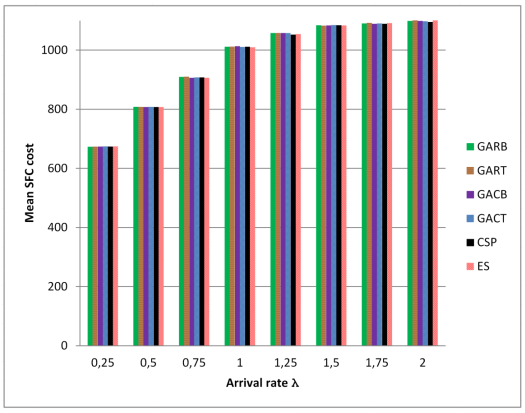


(b) *Mean SFC cost*

Figure 5.5: *Comparison results for different arrival rates and small NFV-Infrastructure and SFC sizes*

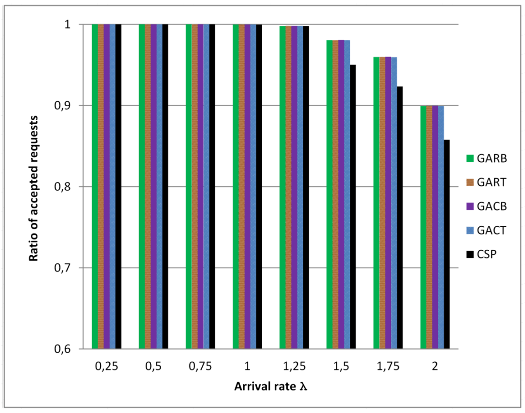


(a) *Ratio of acceptation*

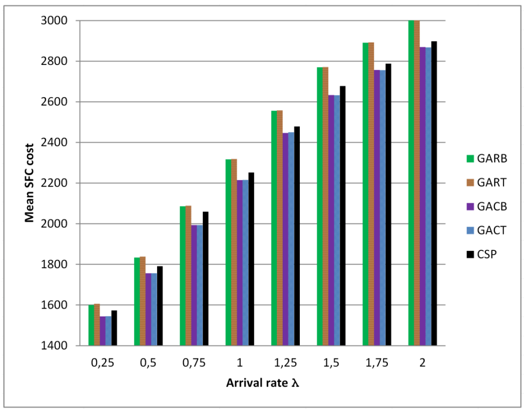


(b) *Mean SFC cost*

Figure 5.6: *Comparison results for various network loads and medium NFV-Infrastructure size*



(a) *Ratio of acceptation*



(b) *Mean SFC cost*

Figure 5.7: *Comparison results for various network loads and large NFV-Infrastructure size*

5.6 Conclusion

In this chapter, we proposed a Knapsack-based algorithm utilizing Genetic Algorithm (GA) for solving the VNF-PC in NFV-Infrastructures (NFV-Is). The performance of our proposed algorithm was compared with other existing algorithms, including Exhaustive Search (ES), Constrained Shortest Paths (CSP), and different variants of GAs.

The simulation experiments were conducted with varying sizes of SFCs and NFV-Is. The results demonstrated that our proposed Knapsack-based algorithm using GA, along with ES and GAs, outperformed CSP in terms of the Ratio of Accepted Requests (RA) for small and medium-sized SFCs. This indicated that our algorithm, which incorporated the Knapsack problem formulation into the GA framework, effectively determined suitable placements for SFCs while considering resource constraints.

Furthermore, the Knapsack-based algorithm demonstrated comparable performance to ES and GAs in terms of Mean SFC Cost (MSC). It produced nearly optimal solutions, considering the trade-off between solution quality and computational complexity. The algorithm's performance was particularly promising for larger SFCs and NFV-Is, where it achieved lower mean SFC costs compared to the other algorithms, including CSP.

The incorporation of the Knapsack problem formulation within the GA framework allowed for efficient exploration of the solution space, leading to improved resource allocation and better RA. The genetic operators, such as selection, crossover, and mutation, facilitated the evolution of solutions, enabling the algorithm to find near-optimal placements for SFCs.

Overall, our proposed Knapsack-based algorithm using GA demonstrated its effectiveness in solving the VNF-PC problem in NFV-Is. It provided a robust and efficient approach, considering resource constraints and achieving high RA with competitive MSC. Further research could focus on optimizing the algorithm's parameters, exploring different selection mechanisms, and investigating the algorithm's performance under diverse scenarios.

In conclusion, the proposed Knapsack-based algorithm using GA offers a promising solution for VNF-PC in NFV-Is. Its ability to efficiently allocate resources and achieve high RA makes it a valuable approach in the field of network function virtualization.

Chapter 6

VNF placement and chaining problem in its generic version

Science without wisdom is like a lottery. We risk losing everything we invested.

Albert Einstein

Abstract

This Chapter tackles the comprehensive version of VNF-PC, considering limited resources and employing a multi-constrained algorithm for efficient resource allocation. Our research contributes to optimizing resource allocation in NFV environments, considering broader network scenarios.

[11] Issam Abdeldjalil Ikhelef et al. “Multi-Constrained Routing-based Heuristic for VNF Placement and Chaining”. In: *ICC 2023 - IEEE International Conference on Communications*. 2023

Chapter content

6.1 Introduction

This chapter tackles the comprehensive version of the VNF-PC problem, benefiting from the work [11] considering limited node and link resources in a broader network environment. Our approach employs a multi-constrained algorithm to efficiently allocate resources for SFC provisioning. By considering computational and bandwidth requirements, we ensure resource optimization while minimizing costs.

Our research contributes to the optimization of resource allocation in NFV environments, encompassing the full scope of the VNF-PC problem. Through insights gained from both the unloaded network scenario and the broader network environment, we aim to enhance the efficiency and cost-effectiveness of resource allocation in NFV deployments.

6.2 Multi-constrained routing to solve the VNF placement and chaining problem

6.2.1 Definition of Multi-constrained routing algorithm

Multi-constrained routing algorithms are computational techniques designed to address the challenges of routing in networks where multiple constraints need to be considered simultaneously. In traditional routing algorithms, paths are determined based on a single metric, such as the shortest path or minimum hop count. However, in many real-world scenarios, networks have diverse requirements that extend beyond a single metric. These requirements can include factors such as bandwidth, delay, cost, reliability, security, and quality of service (QoS) [77].

The objective of multi-constrained routing algorithms is to find optimal or near-optimal paths in the network that satisfy a combination of these constraints. The algorithms aim to strike a balance between different constraints while taking into account the network topology, available resources, and the specific requirements of the application or service being supported.

One common approach to multi-constrained routing is the use of heuristic-based algorithms. These algorithms employ rule-based or iterative methods to find feasible paths that satisfy the given constraints. Heuristic algorithms leverage domain-specific knowledge or predefined metrics to guide the routing decisions. They may explore different paths, evaluate their feasibility based on the constraints, and iteratively refine the routing choices until a suitable solution is found. While heuristic algorithms may not guarantee optimality, they provide practical and efficient solutions for routing in complex networks.

Another approach to multi-constrained routing is the use of mathematical programming models. These models formulate the routing problem as an optimization problem, where the objective is to find the optimal solution that satisfies the constraints. Mathematical techniques, such as linear programming, integer programming, or mixed-integer programming, are employed to solve these models. These methods can handle multiple constraints and provide optimal or near-optimal solutions, but they may be computationally intensive and may not scale well for large networks.

Evolutionary algorithms, such as genetic algorithms, have also been applied to multi-constrained routing problems [78], [79]. These algorithms simulate the process of natural evolution, where a population of candidate solutions evolves over generations through mechanisms such as selection, crossover, and mutation. By iteratively exploring and evaluating different solutions, genetic algorithms search for the best set of paths that satisfy the given constraints. They offer a flexible and adaptive approach to finding near-optimal solutions in complex network environments.

Multi-constrained routing algorithms have applications in various fields such as telecommunications, transport networks, wireless sensor networks, and cloud computing. These algorithms play an important role in ensuring efficient and reliable communication, taking into account various constraints and requirements of the network. By providing optimal or near-optimal routing solutions, multi-constrained routing algorithms enable networks to effectively use available resources, optimize performance, and meet the specific needs of applications and services [80].

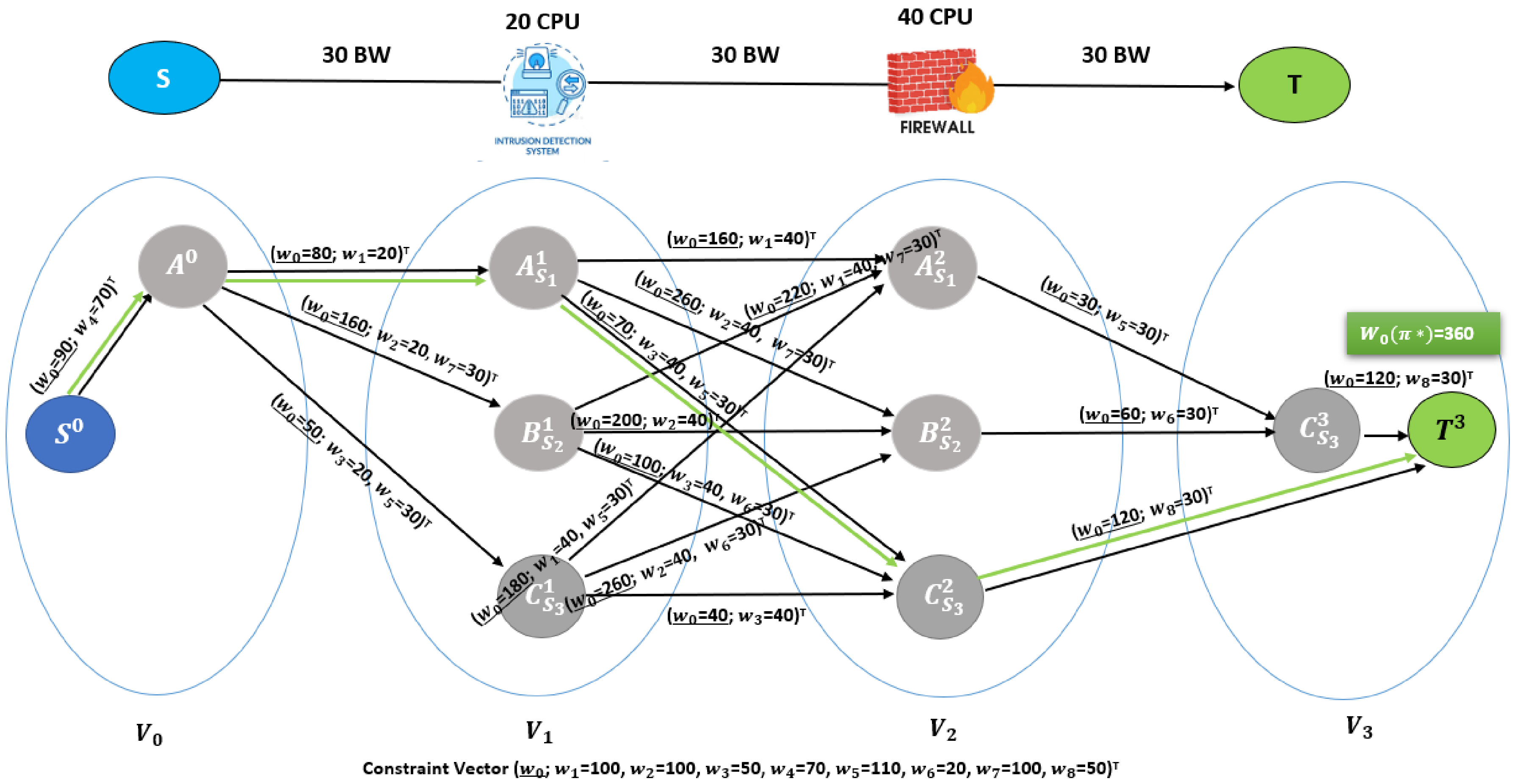

20 CPU
40 CPU
S
30 BW
INTRUSION DETECTION SYSTEM
30 BW
FIREWALL
30 BW
T
A^0
S^0
$A^1_{S_1}$
$B^1_{S_2}$
$C^1_{S_3}$
$A^2_{S_1}$
$B^2_{S_2}$
$C^2_{S_3}$
$C^3_{S_3}$
T^3
$W_0(\pi *)$=360
$(w_0=90; w_4=70)^T$
$(w_0=80; w_1=20)^T$
$(w_0=160; w_2=20, w_7=30)^T$
$(w_0=50; w_3=20, w_5=30)^T$
$(w_0=160; w_1=40)^T$
$(w_0=260; w_2=40, w_7=30)^T$
$(w_0=70; w_3=40, w_5=30)^T$
$(w_0=220; w_1=40, w_7=30)^T$
$(w_0=200; w_2=40)^T$
$(w_0=100; w_3=40, w_6=30)^T$
$(w_0=180; w_1=40, w_5=30)^T$
$(w_0=260; w_2=40, w_6=30)^T$
$(w_0=40; w_3=40)^T$
$(w_0=30; w_5=30)^T$
$(w_0=60; w_6=30)^T$
$(w_0=120; w_8=30)^T$
$(w_0=120; w_8=30)^T$
V_0
V_1
V_2
V_3
Constraint Vector $(w_0; w_1=100, w_2=100, w_3=50, w_4=70, w_5=110, w_6=20, w_7=100, w_8=50)^T$

Figure 6.1: *Multi-constrained routing for SFC placement and chaining.*

6.2.2 Graph transformation and application of the Multi-constrained routing to the VNF-PC problem

To enhance clarity, let's first revisit the graph transformation discussed in Section 4.2.2 of the chapter 4.
In this subsection, we will introduce and provide a detailed explanation of the algorithm developed to address the generalized version of the VNF-PC problem. This algorithm incorporates constraints related to both node and link resources into the transformed graph, thereby providing a comprehensive solution.

This approach is designed to offer a systematic, step-by-step explanation to facilitate a better understanding of the resolution process. In the generic version of the VNF-PC problem, it is essential to validate node and link constraints alongside optimizing the overall cost. To achieve this objective, we propose the following steps:

i. We extend and enhance the k-partite graph G_p defined in the section 4.2.2.

ii. We store all the nodes $u \in V_s$ in an array and define the function $I(u)$ to return the index i $(1 \leq i \leq |V_s|)$ of node u in the array.

iii. We store all the links $l_s \in E_s$ in an array and define the function $I'(l_s)$ to return the index i $(|V_s| < i \leq |V_s| + |E_s|)$ of link l_s in the array.

   For the example in Figure 3.1, the indexes can be set as follows:

   $I(S_1) = 1$
   $I(S_2) = 2$
   $I(S_3) = 3$
   $I'((\mathrm{S}, A)) = 4$
   $I'((A, C)) = 5$
   $I'((C, B)) = 6$
   $I'((B, A)) = 7$
   $I'((C, \mathrm{T})) = 8$

iv. We associate each link (u_i, v_{i+1}) in the k-partite graph with a vector $\overrightarrow{W}_{(u_i, v_{i+1})} =$

$(w_0; w_1, .., w_{|V_s|+|E_s|})^T$ as follows:

$$w_0 = \theta_{(u_i,v_{i+1})}$$
$$\forall 1 \leq j \leq |V_s| : w_j = \begin{cases} d_{i+1} & \text{if } j = I(v) \\ 0 & \text{otherwise} \end{cases}$$
$$\forall |V_s| < j \leq |V_s| + |E_s| : w_j = \begin{cases} b_{i+1} & \text{if } j = I'((u,v)) \\ 0 & \text{otherwise} \end{cases}$$

This way, each link (u_i, v_{i+1}) is associated with three non-nil weights: (1) w_0 corresponding to the cost $\theta_{(u_i,v_{i+1})}$, (2) $w_{I(v)}$ corresponding to the resources required by VNF$_{i+1}$, and (3) $w_{I'((u,v))}$ corresponding to the resources required by the link connecting VNF$_i$ and VNF$_{i+1}$ in the SFC. All other weights are zero.

v. We associate each path $\pi \in G_p$ with a vector $\overrightarrow{W}_\pi$ determined as follows:

$$\overrightarrow{W}_\pi = \sum_{l_p \in \pi} \overrightarrow{W}_{l_p}$$

A path π satisfies the node constraints if and only if it fulfills the following inequality:

$$\forall 1 \leq i \leq |V_s| : \quad w_i(\pi) \leq C_{I^{-1}(i)}$$

where I^{-1} is the inverse function of I.

A path π satisfies the link constraints if and only if it meets the following inequality:

$$\forall |V_s| < i \leq |V_s| + |E_s| : \quad w_i(\pi) \leq B_{I'^{-1}(i)}$$

where I'^{-1} is the inverse function of I'.

Hence, we say that a path π satisfies all the constraints if it complies with both the link and node constraints.

vi. We determine the *optimal multi-constrained path* π^* in G_p that minimizes the overall cost w_0 and satisfies the link and node constraints (see Figure 6.1). The NFV-I nodes traversed by the path π^* in G_p provide the optimal solution to the VNF-PC problem.

Lemma 6.1. *The optimal solution to the VNF-PC problem is determined by the optimal multi-constrained path, which not only minimizes the overall cost but also satisfies the link and node constraints within the corresponding k-partite graph.*

Proof. The correctness of this lemma is straightforward and can be demonstrated through a proof by contradiction. Let's assume the existence of a VNF placement that satisfies the constraints and has a lower cost than the one obtained using the multi-constrained shortest path in the k-partite graph. It is then evident that such an assumption leads to a contradiction.

Therefore, we can confidently assert that the optimal solution to the VNF-PC problem is indeed achieved through the multi-constrained shortest path in the k-partite graph, as it simultaneously optimizes the cost while adhering to the link and node constraints. □

It is noteworthy to mention that we use the term "optimal multi-constrained path" to refer to a path that optimizes one metric (w_0) while satisfying a non-constant number of constraints (w_i with $1 \leq i \leq |V_s| + |E_s|$). Traditionally, multi-constrained routing problems involve a constant number of metrics/constraints.

These results underscore the significance of our findings, demonstrating that the VNF-PC problem can be addressed using a variation of the well-known multi-constrained routing problem. This implies that the multitude of heuristics and algorithms developed for solving the multi-constrained routing problem [81][82] can be leveraged to solve the VNF-PC problem effectively.

6.3 K-multi constrained shortest path-based heuristic (KMSPH) to solve VNF-PC problem

Here, we propose a novel heuristic to efficiently solve the VNF-PC problem in polynomial time. Our heuristic is based on the following:

Algorithm 2 Pseudo code of K-multi constrained shortest path-based heuristic (KMSPH)

```
    Input: $K$ constant indicating the maximum number of paths that can be stored on
    each node
    Output: a path verifying the constraints and connecting the source node $src$ with
    the destination node $t$ in NFV-I. By assumption, $VNF_0$ is mapped on $src$, and
    $VNF_{|V_f|+1}$ is mapped on $t$.
 1: $\forall u \in V$ and $\forall 0 < i \leq |V_f|+1 :\ count(u,i) = 1$
 2: $path(src_0)(1)$={}: the first path to $src_0$ is empty. This path indicates that VNF$_0$
    runs on $src$.
 3: queue.$add(src, 0, \{\})$        ▷ A tuple $(u,i,j)$ identifies node $u_i \in V_i$ (in the k-partite
    graph) and path $path(u_i)(j)$ which is the $j^{th}$ path from the source node $src$ to $u_i$.
 4: While queue.$empty()$=false do
 5:     $(u, i, j)$ = queue.$delete\_min()$
 6:     If $i = |V_f|+1$ then return $path(u_{|V_f|+1})(j)$
 7:     Else
 8:       For each  $v \in$ adjacent $(u) \cup \{u\}$:
                                  $v.offers(VNF_{i+1})$ do
 9:         If $w_{I(v)}(path(u_i)(j)) + d_{i+1} \leq C_v$ and
              $w_{I'((u,v))}(path(u_i)(j) + b_{i+1} \leq B_{(u,v)}$  then
10:           $\pi = path(u_i)(j) + (u,v)$
11:           If $count\,(v,i+1) < K$  then
12:             $path(v_{i+1})(count(v,i+1)) = \pi$
13:             queue.$add((v,i+1,count(v,i+1)))$
14:             $count(v,i+1) = count(v,i+1)+1$
15:           Else
16:             $ind = ind\_high\_cost(Paths(v_{i+1}))$
17:             If $w_0(\pi) < w_0(path(v_{i+1})(ind))$ then
18:               $path(v_{i+1})(ind) = \pi$
19:               queue.$update(v,i+1,ind)$
20:             End If
21:           End If
22:         End If
23:       End For
24:     End If
25: End While
```

i. K-shortest paths: The solution to the VNF-PC problem is given by the shortest path which verifies the node and link constraints. This means that the optimal solution can be obtained by calculating the K-shortest paths with a sufficiently large value of K (K $\geq$ 1).

ii. Limiting the number of stored paths on each node: On each node of the k-partite graph, at most K paths are stored and used for path computation. Such limitation of the stored paths number guarantees a polynomial time complexity but does not ensure the determination of existing or optimal solutions.

iii. Constraints verification at each step of path computation: only the paths verifying the constraints are stored and used.

iv. No use of path dominance: unlike the most algorithms which solve the multi-constrained routing problem (for instance, [81][82]), our heuristic does not use path dominance. Indeed, except the paths traversing the same links in the k-partite graph, no path can dominate another path (recall that each link corresponds to a different constraint and note that each path contains the same number of links).

The pseudo code of our KMSPH is provided by Algorithm 2 which aims to determine the best path connecting in the k-partite graph a source node src_0 to a target node t_{k+1}, verifying all resource constraints and reducing cost.

At initialization, Algorithm 2 assigns at step 1 a counter for each node in the k-partite graph. The counter guarantees that at most K paths are stored in each node. An empty path is also created and associated with the source node src_0 at step 2.

The tuple consisting of the source node, the index of the supported VNF (equal to 0 for the source node), and the empty path $path(src_0)(1)$ is then inserted into a priority queue at step 3 of the algorithm. Obviously, the tuple storing the path with the lowest cost has the highest priority.

As long as the priority queue is not empty, a tuple (u, i, j) identifying the next lowest cost path is removed at step 5 and is explored in the block of instructions from 6 to 24. If the 2 first components of the removed tuple correspond respectively to the target node t and $|V_f|+1$, the destination is reached and the path $path(t_{|V_f|+1}(j)$ is returned.

Otherwise, all the neighbors nodes v to u which support VNF$_{i+1}$ are explored (step 8) by adding the link (u,v) to the path $path(u_i)(j)$ (i.e., $\pi = path(u_i)(j) + (u,v)$). If the resulting path π satisfies the node and link resource constraints (step 9), the path π is kept (instruction 10) and becomes a candidate for storage. More precisely, path π is stored for node v_{i+1} if its number of stored paths is less than K (lines 11-14). Otherwise, path π will replace the highest cost path stored for node v_{i+1} if its cost is also larger than that of π (lines 15-21). The queue is updated to reflect this path replacement in statement 18 (cost decrease of the replaced path).

The complexity of KMSPH can be obtained as follows: Instruction 1 is performed in $O(|V_f|.|V_s|)$. Instructions 2 and 3 are performed in $O(1)$. Instructions 4 to 25 inside the do while loop can be executed $K.|V_s|.|V_f|+2$ times at most. By implementing *queue* with a Fibonacci heap, instruction 5 is accomplished in $O(\log(K|V_s|.|V_f|))$. The 'for loop' of statement 8 is invoked $K.(2|E_s|+V_s).(|V_f|)$ to process the nodes between *src* and t. The condition in line 9 is checked in $O(|V_f|)$, and instruction 16 runs in $O(K)$. The other instructions 11 to 14 and 17 to 19 hold in $O(1)$.

Therefore, the complexity of KMSPH is:

$$O(K|V_f|.|V_s|.\log(K|V_s|.|V_f|)+K^2(|V_s|+|E_s|).|V_f|^2)$$

As $\log(|V_s|)<|V_s|$ and $\log(K)<K$, the complexity of KMSPH can be reduced to:

$$O(K.|V_f|.|V_s|.\log(|V_s|)+K^2.(|V_s|+|E_s|).|V_f|^2)$$

Note that for unbounded values of K (i.e., $K \geq |V_s|^{|V_f|-1}$), KMSPH solves the VNF-PC problem exactly. For constant values of K, the complexity becomes polynomial and equal to:

$$O(|V_f|.|V_s|.\log(|V_s|)+|V_f|^2.|V_s|+|V_f|.|E_s|)$$

6.4 Performance Evaluation

The higher the number of paths stored on each node, the better the quality of the solutions determined by KMSPH. The objective of our simulations is to study the impact of the values of K on the quality of the solutions. To do this, we compared 5 instances of KMSPH obtained by setting K to the following values: 1, 5, 50, 300, and ∞. We recall that with $K=\infty$, KMSPH gives exact solutions at the expense of a very long running time.

In our first tests, we used SFC and NFV-I of small sizes to ensure the determination of the optimal solutions, which will be compared to the solutions determined for $K \leq 300$.

In our latest tests, we used larger SFCs and NFV-I. The objective is to experimentally measure the impact of the increase of K on the quality of the determined solutions and to estimate the convergence time.

6.4.1 Simulation Environment and Scenarios

In our simulation, SFC requests arrive according to a Poisson process with λ requests/-time unit, and the lifetime of each request follows a uniform distribution $U(10,20)$.

Table 6.1: *Ratio of Accepted requests (1st scenario)*

λ	K = 1	K = 5	K = 50	K = 300	Optimum
0.2	100%	100%	100%	100%	100%
0.4	99.92%	99.92%	99.95%	99.95%	99.95%
0.6	97.55%	97.70%	97.77%	97.77%	97.77%
0.8	91.33%	91.75%	91.89%	91.89%	91.89%
1	82.37%	83.21%	83.50%	83.51%	83.51%

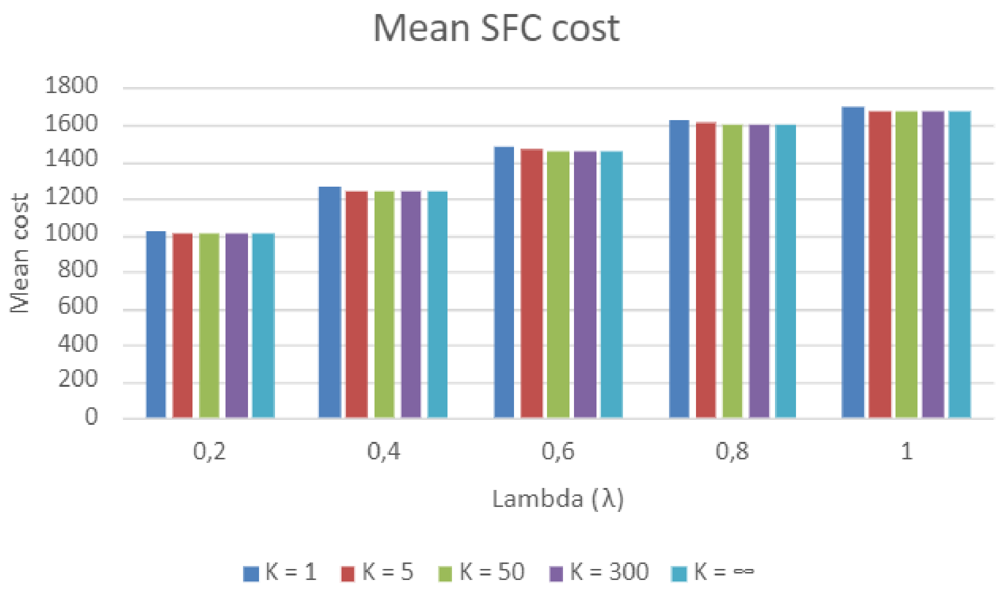


Figure 6.2: *Mean SFC costs (1st scenario)*

Three scenarios are considered for the performance comparison:

- **Scenario 1**: with λ evolving from 0.2 to 1 (with step = 0.2). The SFC size is equal to 8, and the NFV-Infrastructure is composed of 15 nodes and 105 links.
- **Scenario 2**: with $\lambda = 1$, the NFV-I is the same as Scenario 1, but the SFC sizes change from 4 to 8 with a step of 1.
- **Scenario 3**: a very large NFV-I is used with the number of nodes equals 50 and the number of links equals 1225. SFC sizes scale from 6 to 12 VNFs.

The available CPU capacity per physical node is randomly drawn within the range of 50 to 250 units, and the available bandwidth capacity per link is randomly drawn within the range of 10 to 50 units. The requested CPU capacity of each VNF is drawn randomly in (10, 20) with random requested bandwidths in the interval (5, 10). The resource costs α and β are randomly generated in the range of (5, 20). The simulation time for the three scenarios is 100,000 units.

Table 6.2: *Ratio of Accepted requests (2nd scenario)*

SFC size	K = 1	K = 5	K = 50	K = 300	Optimum
4	100 %	100 %	100 %	100 %	100 %
5	99,18 %	99,25 %	99,26 %	99,26 %	99,26 %
6	95,88 %	96,16 %	96,17 %	96,17 %	96,17 %
7	89,76 %	90,50 %	90,61 %	90,61 %	90,61 %
8	82,25 %	83,05 %	83,27 %	83,27 %	83,27 %

6.4.2 Performance Metrics

For our study, we computed the two metrics defined below:

- Ratio of Accepted Requests (RA): It represents the ratio between the number of accepted SFC requests and the total number of received SFC requests.
- Mean SFC Cost (MSC): It determines the mean cost of SFCs placed successfully. It's obtained by calculating the ratio between the total cost of CPU and bandwidth resources allocated for SFCs and the total number of accepted SFCs.

6.4.3 Simulation Results

In the first scenario, the results indicate that all the compared methods have similar acceptance rates, as shown in Table 1, when dealing with low network loads ($\lambda = 0.2$ and $\lambda = 0.4$). This similarity can be attributed to the fact that, under conditions of ample resources, the shortest path algorithm performs optimally, leading to nearly identical results across the methods.

However, as the network load increases, the performance difference becomes apparent, especially for KMSPH instances with $K \leq 50$. In particular, a larger value of K means a larger proportion of requests are accepted. This suggests that increasing K leads to improved acceptance rate, especially under high network load.

In summary, the performance differences between methods become more prominent as network loads intensify, and KMSPH with higher K values outperforms its counterparts by accepting a greater proportion of SFC requests. Regarding the mean SFC cost metric, the observations from Figure 6.2 reveal that, except for KMSPH with $K = 1$, all the compared methods exhibit similar performance. Despite the potentially large number of paths stored for each node, which can reach up to 15^7 for scenario 1, the combined utilization of (1) the k-partite graph to diversify path structures (by storing K different paths $(src,\ v_i)$ for each node $v_i \in V_i$) and (2) the resource constraint

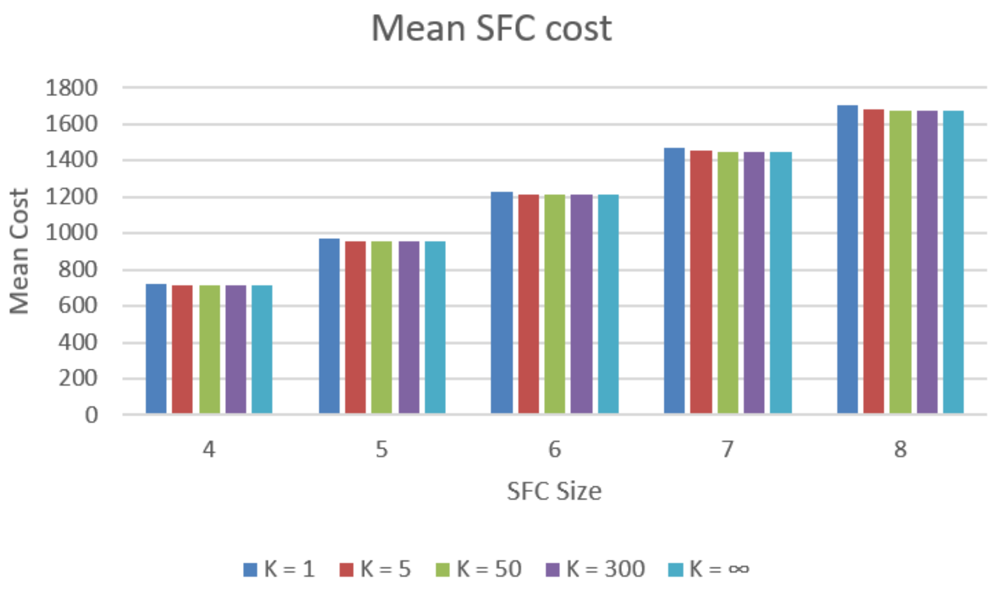


Figure 6.3: *Mean SFC costs (2nd scenario)*

Table 6.3: *Ratio of Accepted requests (3rd scenario)*

λ	K = 1	K = 5	K = 50	K = 300
1	100 %	100 %	100 %	100 %
2	100 %	100 %	100 %	100 %
3	97,82 %	97,98 %	98,02 %	98,05 %
4	83,03 %	83,86 %	84,31 %	84,63 %
5	68,68 %	69,98 %	70,97 %	71,19 %

verification that removes path segments not conforming to the constraints allows for the determination of solutions that closely approximate the optima.

Scenario 2, which investigates the impact of SFC size on KMSPH's performance, is characterized by results presented in Table 6.2 and Figure 6.3. These results indicate that as SFC sizes increase, the differences in performance among the compared methods remain limited and relatively stable. The disparities are often minimal and hardly discernible when K exceeds 50.

In summary, the mean SFC cost metric shows that KMSPH with K values greater than 1 performs similarly to other methods, with only slight differences in performance, even when SFC sizes increase.

Despite the exponential growth in the total number of paths resulting from the increase in SFC size, which expands the solution space substantially, the utilization of K-constrained shortest paths within the k-partite graph effectively facilitates the exploration of diverse regions within the feasible solution space. This exploration process

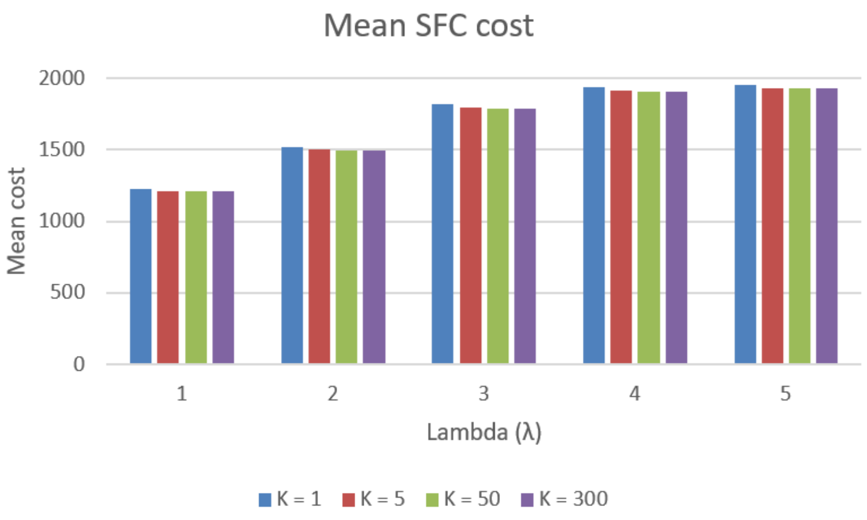


Figure 6.4: *Mean SFC costs (3rd scenario)*

frequently leads to solutions that closely approximate the optima. Consequently, in your simulation, deploying KMSPH with a value of $K = 300$ enables the attainment of performance that is remarkably close to the optimal solutions.

In the third scenario, the results clearly demonstrate that KMSPH instances rapidly converge towards solutions that are very close to each other as the value of K increases. This convergence is evident in Table 6.3, where the disparity between the KMSPH instances diminishes considerably as K grows. For instance, the gap between the ratios of accepted requests obtained with instances of $K = 50$ and $K = 300$ is quite small, and it's notably smaller than the difference between instances with $K = 1$ and $K = 5$. The same behavior is observed in figure 6.4, where the average SFC costs of different KMSPH instances are very close to each other. As the value of K increases, the difference in average SFC costs decreases. This behavior highlights the effectiveness of KMSPH in providing consistent near-optimal solutions as K increases.

To further quantify the differences in mean SFC costs between various instances of KMSPH with K values less than 300 and the instance with $K = 300$, we calculated the normalized difference and present it in Table 6.4. This metric, denoted as *ND* $(K = x, K = 300)$, for the KMSPH instance with $K = x$ is computed as follows:

$$ND(K = x, K = 300) = \frac{MSC(K = x) - MSC(K = 300)}{MSC(K = 300)} \times 100$$

The results in Table 6.4 clearly show that the normalized difference decreases rapidly

Table 6.4: *Normalized difference (3rd scenario)*

λ	ND (K=1, K=300)	ND (K=5, K=300)	ND (K=50, K=300)
1	1,18 %	0,26 %	0,03 %
2	1,48 %	0,36 %	0,05 %
3	1,75 %	0,46 %	0,07 %
4	1,37 %	0,16 %	-0,13 %
5	0,94 %	-0,14 %	-0,01 %

as K increases. For instance, in the case where $K = 50$, the normalized difference is very small and almost negligible. In some instances, it may even be negative due to the acceptance of more requests with higher costs for $K = 300$.

These results from the third scenario show that as K increases, the KMSPH heuristic rapidly converges to a solution that is very close to the optimal solution. This property highlights the usefulness and effectiveness of KMSPH, especially for solutions close to the optimal solution.

6.5 Conclusion

In this chapter, we focus on the VNF Placement and Chaining (VNF-PC) problem within the context of Network Function Virtualization (NFV) environments. We address the general version of the problem and demonstrate that it can be effectively solved using multi-constrained routing techniques.

We explored the transformation of this problem into a graph-based representation, facilitating the application of the K-Constrained Multi-Shortest Paths Heuristic (KMSPH) algorithm. Through a systematic evaluation across various scenarios, we discovered that KMSPH, guided by the parameter K, adeptly adapts to changing network conditions, exhibiting remarkable performance by consistently converging toward near-optimal solutions.

Notably, KMSPH's ability to efficiently handle diverse SFC sizes, maintain solution diversity, and optimize resource utilization underscores its potential as a valuable tool for network operators seeking robust solutions for VNF placement and chaining within NFV infrastructures.

CHAPTER 7

Exploring Constrained Shortest Paths Routing for VNF Placement and Chaining in NFV-I

Des chercheurs qui cherchent, on en trouve. Mais des chercheurs qui trouvent, on en cherche.

Charles de Gaulle

Abstract

This chapter delves into Virtual Network Function (VNF) placement and chaining. We present an extended Constrained Shortest Path-base Heuristic in multi-Partite Graph (CSPH-PG), exploring its effectiveness in diverse NFV scenarios. The chapter details the VNF placement problem in unloaded networks and introduces the constrained shortest paths algorithm, a core component of the CSPH-PG approach. Based on ongoing research [12], this work demonstrates the CSPH-PG's scalability and efficiency, particularly for large NFV deployments and high traffic loads.

[12] Issam Abdeldjalil Ikhelef et al. “Constrained Routing in Multi-Partite Graph to Solve VNF Placement and Chaining Problem”. Subbmitted to Journal of Network and Computer Applications - 2023

Chapter content

7.1 Introduction

In this chapter, we delve into the intricate realm of Virtual Network Function (VNF) placement and chaining. Our overarching goal is to minimize the total allocation cost for incoming Service Function Chains (SFCs), especially in the face of surging demand and heightened traffic loads within NFV-Is.

This work serves as an extension and refinement of the approach presented in the previous chapter 6, offering a more comprehensive exploration of the Constrained Shortest Path-base Heuristic in multi-Partite Graph (CSPH-PG). Our objective remains the same: to provide efficient solutions to the VNF-PC problem while addressing the unique challenges posed by varying NFV-I sizes and traffic demands.

In this chapter, we meticulously detail the VNF placement and chaining problem in the context of an unloaded network, providing a solid foundation for our subsequent discussions. Then, we introduce the constrained shortest paths algorithm, a pivotal component in our quest to solve the VNF-PC problem. This algorithm forms the core of our heuristic approach and plays a crucial role in optimizing the allocation of resources and achieving cost-effective SFC provisioning.

Furthermore, it's worth noting that this chapter is the result of a submission to a prestigious journal [12].

7.2 VNFs placement and chaining problem in an over-resourced NFV-I

To facilitate comprehension, we begin by addressing a simplified variant of the VNF placement and chaining problem known as the RVNF-PC problem, before tackling the more complex generic VNF-PC problem. In this section, we operate within the context of an over-resourced NFV-I, where the available resources on all substrate components significantly exceed the demands of the Service Function Chains (SFCs).

In this relaxed version of the VNF placement and chaining problem, our objective is to minimize the total placement cost associated with provisioning the SFCs, with the sole constraint being flow continuity. Clearly, this overall cost is determined by the resources allocated to the nodes and links used for deploying and executing the SFCs.

7.2.1 Graph transformation to solve RVNF-PR problem

For ease of understanding, we only consider one node resource (e.g. CPU) as well as one link resource (bandwidth). Similarly, to simplify the notations, we denote by v_i the

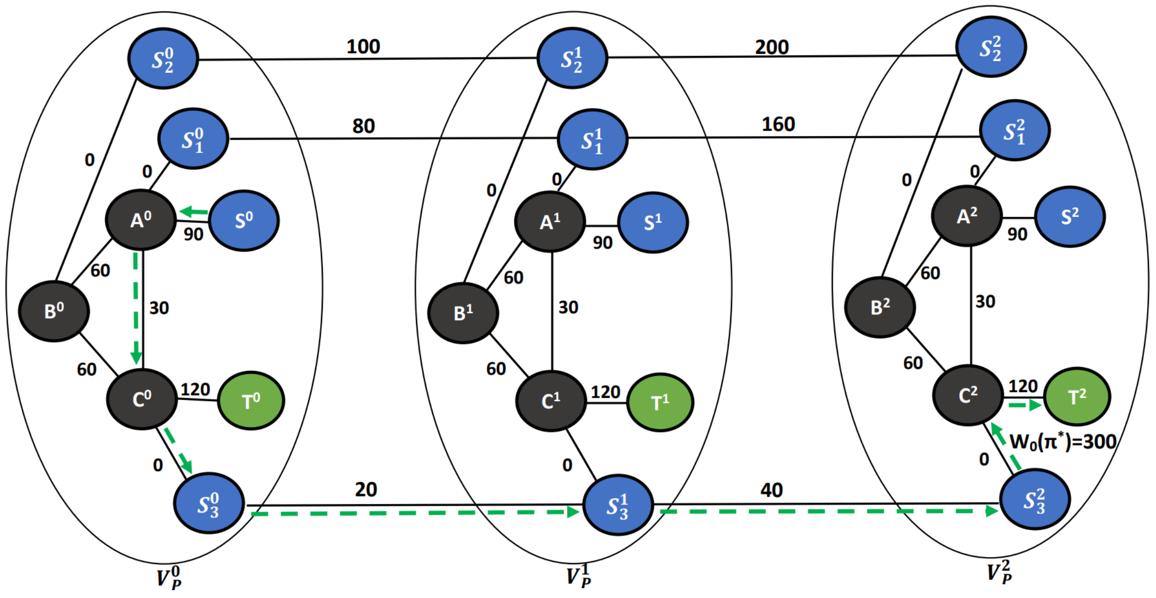


Figure 7.1: *Construction of multi-partite graph to solve relaxed version of VNF-PC*

i^{th} VNF in the SFC, starting with index 0 and ending with index $|\mathcal{V}_f|-1$. In this way, the source and target nodes of the SFC respectively correspond to v_0 and $v_{|\mathcal{V}_f|-1}$

To solve RVNF-PC problem, we first derive a new multi-partite oriented graph $\mathcal{G}_p = (\mathcal{V}_p, \mathcal{E}_p)$ from the NFV-I and SFC graphs as follows (for a good understanding of the transformation see Figure 7.1):

- $\mathcal{V}_p = \bigcup_{i \in (0, |\mathcal{V}_f|-2)} \mathcal{V}_p^i$ so that:

$$\forall i \in (0, |\mathcal{V}_f|-2) : (U^i \in \mathcal{V}_p^i \iff U \in \mathcal{V}_s)$$

  In other word, the multi-partite graph is composed of $k = |V_f| - 2$ different components (sets of nodes), each one contains $|\mathcal{V}_s|$ nodes.

- Only node servers U^i which can embed VNF v_i are connected with node servers U^{i-1} with links $l_p = (U^{i-1}, U^i)$ whose cost w_0^l are determined as follows:

$$w_0^{l_p} = w_0^{(U_{i-1}, U_i)} = d_{v_i}^1 \times \alpha_{U, v_i}^1 \tag{7.1}$$

- Two nodes U^i, $V^i \in \mathcal{V}_p^i$ are interconnected with a link $l_p = (U_i, V_i)$ iff their corresponding nodes U, $V \in \mathcal{V}_s$ are interconnected with a substrate link. In this way, the cost $w_0^{(U_i,V_i)}$ of link $(U_i,\, V_i)$ corresponds to:

$$w_0^{l_p} = w_0^{(U_i,V_i)} = d^{bw}_{(v_i,v_{i+1})} \times \beta_{(U,V)} \tag{7.2}$$

In Figure 7.1, the 3-partite graph obtained from the transformation of the NFV-I of Figure 3.1 and SFC of Figure 3.2 is shown. It is determined as follows:

- We determine the components V_p^i $(i \in \{0,1,2\})$:
  $V_p^0 = \{S^0, A^0, B^0, C^0, S_1^0, S_2^0, S_3^0, T^0\}$,
  $V_p^1 = \{S^1, A^1, B^1, C^1, S_1^1, S_2^1, S_3^1, T^1\}$,
  $V_p^2 = \{S^2, A^2, B^2, C^2, S_1^2, S_2^2, S_3^2, T^2\}$.
- We add to the 3-partite graph the following intra-component links:
  (S^0,A^0), (A^0,C^0), (A^0,B^0), (B^0,C^0), (C^0,T^0), (A^0,S_1^0), (B^0,S_2^0), (C^0,S_3^0),
  (S^1,A^1), (A^1,C^1), (A^1,B^1), (B^1,C^1), (C^1,T^1), (A^1,S_1^1), (B^1,S_2^1), (C^1,S_3^1),
  (S^2,A^2), (A^2,C^2), (A^2,B^2), (B^2,C^2), (C^2,T^2), (A^1,S_1^1), (B^1,S_2^1), (C^1,S_3^1).
- We add to the 3-partite graph the following inter-component links:
  (S_1^0,S_1^1), (S_2^0,S_2^1) and (S_3^0,S_3^1),
  (S_1^1,S_1^2), (S_2^1,S_2^2) and (S_3^1,S_3^2).
- According to the equations 7.1 and 7.2, we can deduce the link costs in the 3-partite graph.

By adding reflexive links to NFV-I, we will be able to encode any solution to VNF-PC by a substrate non elementary path connecting the source to the destination and crossing as many reflexive links as there are VNFs in the SFC to be provisioned.
In Figure 3.3, substrate non elementary path (S,A,C,S_3,S_3,S_3,C,T) encodes one and only one solution to VNF-PC problem where the flow coming from the source node S crosses switches A and C before its is processed by VNFs 1 and 2 on server S_3 (to each reflexive link corresponds a VNF). The flow is then transmitted to the target switch T via switch C.

It is easy to see that substrate non elementary paths can also be encoded by paths in the multi-partite graph (inter-component links become reflexive links whereas the intra-component paths are transformed to substrate subpaths interconnecting successive reflexive links). In Figure 7.1 for instance, the multi-partite path $(S^0, A^0, C^0, S_3^0, S_3^1, S_3^2, C^2, T^2)$

can be transformed (vice-versa) to the substrate non elementary path $(S, A, C, S_3, S_3, S_3, C, T)$ which crosses node S_3 three times and node C twice.

As a result, VNF-PC can be transformed to an equivalent path search problem in a multi-partite graph. Lemmas 7.1 and 7.2 formalize the equivalence between VNF-PC problem and shortest path search problem in a multi-partite graph.

Lemma 7.1. *The optimal solution to the relaxed version of VNF placement and chaining problem is given by the shortest path π that interconnects S_s^0 and $T_s^{|\mathcal{V}_f|-2}$ in the corresponding multi-partite graph. The first node U^i traversed by π in each set of nodes $\mathcal{V}_p^i$ $(0 < i \leq |\mathcal{E}_f| - 2)$ determines the embedding substrate node U for VNF v_i.*

Proof. The correctness of the previous lemma is trivial and can be proved by contradiction. We assume the existence of a RVNF-PC solution that has a lower cost than that obtained with the shortest path in the multi-partite graph and then we show that such an assumption leads to a contradiction. Indeed, by definition, the path encoding the solution to RVNF-PC cannot be lower than the shortest path in the k-prtite graph.

Similarly, we assume the existence of a path interconnecting the source and target nodes in the multi-partite graph with a smaller cost than that corresponding to the optimal solution to RVNF-PC problem, then show that such an assumption leads to a contradiction. Indeed, as any path interconnecting the source and target nodes in the multi-partite graph also encodes a feasible solution to RVNF-PC, its cost should be lower or equal to that corresponding to the optimal solution to RVNF-PC. □

For our example in Figures 3.1 and 3.2, we obtain the optimal solution to RVNF-PC by determining the shortest path that connects the substrate source node S^0 to the substrate target node T^2 in the 3-partite graph shown in Figure 7.1. More specifically, the optimal mapping of the SFC is shown by the green arrows in Figure 7.1 $(S^0, A^0, C^0, S3^0, S3^1, S3^2, C^2, T^2)$ where both $VNF1$ and $VNF2$ should be placed in server S_3. The overall cost of this SFC mapping is equal to 300 since:
$w_0^{(S^0,A^0)} = 30 \times 3 = 90$,
$w_0^{(A^0,C^0)} = 30 \times 1 = 30$,
$w_0^{(C^0,S_3^0)} = 0$,
$w_0^{(S_3^0,S_3^1)} = 20 \times 1 = 20$,
$w_0^{(S_3^1,S_3^2)} = 40 \times 1 = 40$,

$w_0^{(S_3^2,C^2)} = 0,$
$w_0^{(C^2,T^2)} = 30 \times 4 = 120.$

7.2.1.1 Complexity

Our algorithm solves RVNF-PC in polynomial time. By applying Dijkstra's algorithm on the multi-partite graph, we obtain the worst-case time complexity of our algorithm which corresponds to:

$$\mathcal{O}(|\mathcal{V}_f||\mathcal{E}_s| + |\mathcal{V}_f||\mathcal{V}_s| + |\mathcal{V}_f||\mathcal{V}_s|\log(|\mathcal{V}_f||\mathcal{V}_s|)) = \mathcal{O}(|\mathcal{V}_f||\mathcal{E}_s| + |\mathcal{V}_f||\mathcal{V}_s|\log(|\mathcal{V}_f||\mathcal{V}_s|))$$

7.3 Constrained Shortest Paths to solve generic version of VNF-PC

In this section, we address the generic VNF-PC problem, which involves optimizing the cost of SFC embedding while considering limited resources on nodes and links in an NFV infrastructure. More clearly, we assume here that the CPU and bandwidth capacities are finite and restrained. To tackle these challenges, we extend the previous algorithm to take into account the capacity constraints when computing the paths.

More specifically, we first define a vector of constraints $\overrightarrow{C}$ which include all the server node and link capacity constraints. For instance, the vector constraints corresponding to Figure 3.1 is as follows:
$\overrightarrow{C} = (-; C^1_{S_1} = 100, C^1_{S_2} = 100, C^1_{S_3} = 50, C^{bw}_{SA} = 70, C^{bw}_{AB} = 100, C^{bw}_{AC} = 110, C^{bw}_{BC} = 20, C^{bw}_{CT} = 50)$.

The three servers S_1, S_2 and S_3 are respectively associated with the following constraints $C^1_{S_1} = 100$, $C^1_{S_2} = 100$ and $C^1_{S_3} = 50$ whereas the links (S,A), (A,B), (A,C), (B,C), (C,T) are respectively associated with the five last constraints $C^{bw}_{SA} = 70$, $C^{bw}_{AB} = 100$, $C^{bw}_{AC} = 110$, $C^{bw}_{BC} = 20$ and $C^{bw}_{CT} = 50)$. Note that neither the switches nor the links with unlimited bandwidth capacities are associated with constraints because we assumed that their resources are sufficient to accommodate the SFCs.

Second, we enriched the multi-partite graph with new weights allowing constraint checking during the path computation. More specifically, we respectively assign to each substrate server node U and each substrate link $l_s = (U,V)$ in NFV-I weight metrics w_U and w_{UV}. Besides, we propose to set the values of the weight metrics in the multi-partite graph as follows:

- The first weight w_0 of each link corresponds to its cost. It is determined according to equations (7.1) and (7.2).

- For an inter-component link (U_i, U_{i+1}) in the multi-partite graph, the value of its corresponding weight metric w_U is equal to the demand $d^1_{v_{i+1}}$ of VNF v_{i+1}.
- For an intra-component link (U_i, V_i) in the multi-partite graph, the value of its corresponding weight metric w_{UV} is equal to the demand $d^{bw}_{l_f}$ where l_f is the SFC link connecting VNF v_i and VNF v_{i+1}.

As a result, each link l_p in the multi-partite graph can be associated with a weight vector $\overrightarrow{W}_{l_p}$ of $1 + r \times |\mathcal{V}_s| + |\mathcal{E}_s|$ components with:

- at most, $r+1$ components in the weight vector can be non-nil for inter-component links which represent substrate server nodes,
- at most, 2 components in the weight vector can be non-nil for intra-component links which represent substrate links.

In Figure 7.2 where the links are labeled with their corresponding vectors, we deliberately omit to represent the nil vector components. As we associated one resource (cpu resource) to the server nodes, their vectors will include at most 2 non-nil components. For instance, the weight vectors of inter-component links (S^0_1, S^1_1) (S^1_3, S^2_3) are determined as follows:
$\overrightarrow{W}_{(S^0_1,S^1_1)} = (w_0 = 80; w^1_{S_1} = 20, w^1_{S_2} = 0, w^1_{S_3} = 0, w^{bw}_{SA} = 0, w^{bw}_{AB} = 0, w^{bw}_{AC} = 0, w^{bw}_{BC} = 0, w^{bw}_{CT} = 0)$
$\overrightarrow{W}_{(S^1_3,S^2_3)} = (w_0 = 40; w^1_{S_1} = 0, w^1_{S_2} = 0, w^1_{S_3} = 40, w^{bw}_{SA} = 0, w^{bw}_{AB} = 0, w^{bw}_{AC} = 0, w^{bw}_{BC} = 0, w^{bw}_{CT} = 0)$

In a similar way, we determine the weight vectors of intra-component links (A^0, C^0) and (C^2, T^2):
$\overrightarrow{W}_{(A^0,B^0)} = (w_0 = 60; w^1_{S_1} = 0, w^1_{S_2} = 0, w^1_{S_3} = 0, w^{bw}_{SA} = 0, w^{bw}_{AB} = 30, w^{bw}_{AC} = 0, w^{bw}_{BC} = 0, w^{bw}_{CT} = 0)$
$\overrightarrow{W}_{(C^2,T^2)} = (w_0 = 120; w^1_{S_1} = 0, w^1_{S_2} = 0, w^1_{S_3} = 0, w^{bw}_{SA} = 0, w^{bw}_{AB} = 0, w^{bw}_{AC} = 0, w^{bw}_{BC} = 0, w^{bw}_{CT} = 30)$

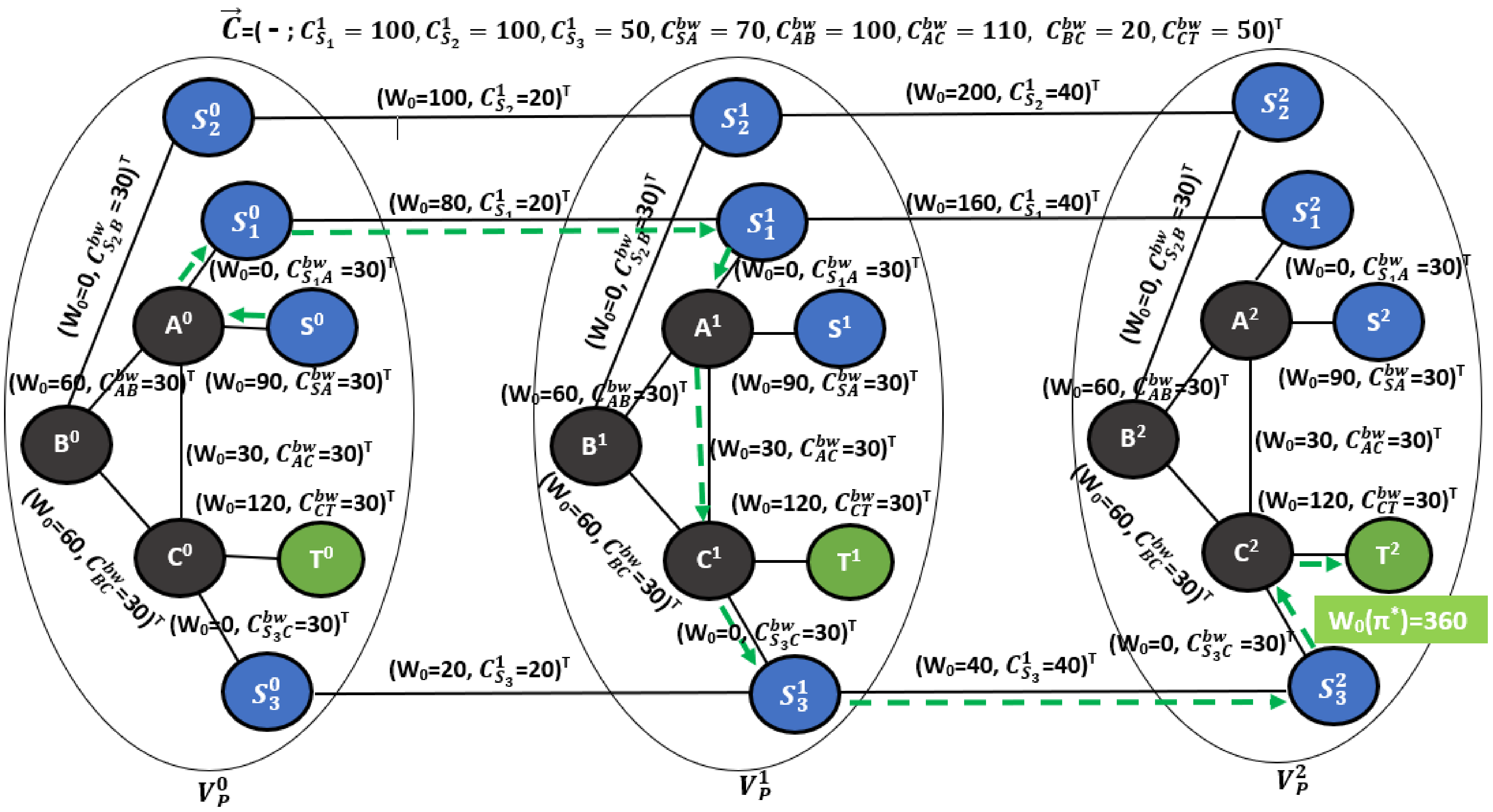


Figure 7.2: *Multi-partite graph with link weight vectors.*

As each feasible solution to the VNF placement and chaining problem can be encoded by one and only one path connecting S_0 to $T_{|\mathcal{V}_f|-2}$ in the multi-partite graph, we translate the objective and constraints of the SFC problem to a path computation problem as follows:

- First, we define the weight $\overrightarrow{W}(\pi)$ of a path π as the sum of the vectors of its links, i.e.:

$$\overrightarrow{W}(\pi) = \sum_{l_p \in \mathcal{E}_p} \overrightarrow{W}_{l_p}$$

- Second, we determine the optimal solution to the VNF placement and chaining problem by determining a path which: (1) connects S_0 to $T_{|\mathcal{V}_f|-2}$ in the multi-partite graph, (2) minimizes the weight metric w_0 and (3) verifies the constraint $\overrightarrow{W}(\pi) \leq \overrightarrow{C}$.

We recall that $\overrightarrow{W}(\pi) \leq \overrightarrow{C}$ iff $\forall i > 0 : w_i(\pi) \leq c_i$ where $w_i(\pi)$ and c_i correspond respectively to the i^{th} components of $\overrightarrow{W}(\pi)$ and $\overrightarrow{C}$.

As a result, a path π in G_p satisfies the node constraints if it verifies the following inequality:

$$\forall U \in \mathcal{V}_s : \quad w_U(\pi) \leq C_U$$

Similarly, a path π in G_p satisfies the link constraints if it verifies the following inequality:

$$\forall l_s \in \mathcal{E}_s : \quad w_{l_s}(\pi) \leq C_{l_s}$$

To find the optimal solution to VNF-PC problem, we determine the constrained shortest path π^* in G_p that minimizes the allocation cost w_0 while satisfying the link and node capacity constraints (see Figure 7.2). The corresponding substrate non elementary path give us the optimal routing and placement of the VNFs.

We support the optimality of this assertion with the following lemma:

Lemma 7.2. *The optimal solution to VNF-PC problem is given by the constrained shortest path π that minimizes the overall cost and satisfies the link and node capacity constraints while connecting the nodes S_s^0 and $T_s^{|\mathcal{V}_f|-2}$ in the corresponding multi-partite graph. The first node U^i crossed by π in each set of nodes $\mathcal{V}_p^i$ $(0 < i \leq |\mathcal{E}_f| - 2)$ determines the embedding substrate node U for VNF v_i.*

Proof. The correctness of the lemma can be proven by contradiction. Assuming the existence of a VNF placement satisfying the constraints and having a lower cost than the constrained shortest path in the multi-partite graph leads to a contradiction since the latter also encodes a least-cost solution to VNF-PC. Similarly, assuming a constrained shortest path interconnecting S_s^0 and $T_s^{|\mathcal{V}_f|-2}$, we can easily prove that this path encodes a valid and least-cost solution to VNF-PC problem, leading to a contradiction. $\square$

It is worth noting that we use the expression *optimal constrained shortest path* to refer to a path verifying a varying number of constraints (w_i with $1 \leq K \leq |\mathcal{V}_s| + |\mathcal{E}_s|$). This deviates from the conventional constrained shortest paths problem, which typically involves a fixed number of metrics or constraints.

Our results highlight the significance of the equivalence between the VNF-PC problem and the well-known multi-constraint routing problem. This equivalence allows us to leverage and adapt existing heuristics and algorithms developed for constrained shortest paths to effectively solve VNF-PC problem.
Given the NP-hard nature of the problem [83], we propose below a novel heuristic accelerating the computation by reducing the search areas in the solution space. More precisely, by limiting the number of paths to be stored on each node, we show that constrained shortest paths still allows determining near-optimal solutions to VNF-PC.

7.3.1 Constrained Shortest Path-base Heuristic in multi-Partite Graph (CSPH-PG) to solve VNF-PC problem

In this subsection, we introduce a novel heuristic approach aimed at efficiently solving the VNF-PC problem within a polynomial time complexity. Our heuristic is based on the following:

Algorithm 3 Pseudo code of Constrained Shortest Path-base Heuristic in multi-Partite Graph (CSPH-PG)

```
Input: $K$ constant indicating the maximum number of paths that can be stored on each node, $G_f$ and NFV-I
Output: a path verifying the constraints and connecting the source node $S_s$ to the destination node $T_s$ in NFV-I. By assumption, $VNF\ v_0$ is mapped on $S_s$ whereas $VNF\ v_{|\mathcal{V}_f|-2}$ is mapped on $T_s$.
1: $path(S_s)(0)(1) \leftarrow \{\}$ ▷ the first path to $S_s^0$ in the multi-partite graph is empty. This path indicates that $VNF\ v_0$ runs on $S_s$.
2: $\forall U \in \mathcal{V}_s$ and $\forall 0 < i \leq |\mathcal{V}_f| - 2 :\ count(U,i) \leftarrow 0$
3: $queue.add(S_s, 0, 1)$ ▷ Each tuple $(U,i,j)$ in $queue$ identifies node $U_i \in V_p^i$ (in the multi-partite graph) and its $j^{th}$ stored path $path(U)(i)(j)$.
4: While queue.$empty()$=false do
5:   $(U, i, j) \leftarrow queue.delete\_min()$
6:   If $i = |\mathcal{V}_f| - 2$ and $U = T_s$ then
7:     return $path(U)(|\mathcal{V}_f| - 2)(j)$
8:   End If
9:   If $i \neq |\mathcal{V}_f| - 2$ and $offers\ (U,\ v_{i+1})$ and
         $w_U(path(U)(i)(j)) + d^1_{v_{i+1}} \leq C_U$ then
10:      $\pi = path(U)(i)(j) + (U,U)$
11:      If $count\,(U, i+1) < K$
12:        $count(U, i+1) \leftarrow count(U, i+1) + 1$
13:        $path(U)(i+1)(count(U, i+1)) \leftarrow \pi$
14:        $queue.add(U, i+1, count(U, i+1))$
15:      Else
16:        $d \leftarrow index\_highest\_cost(path(U)(i+1))$
17:        If $w_0(\pi) < w_0(path(U)(i+1)(d))$
18:          $path(U)(i+1)(d) \leftarrow \pi$
19:          $queue.update(U, i+1, d)$
20:        End If
21:      End If
22:  End If
23:  $bw \leftarrow d^{bw}_{(v_i, v_{i+1})}$
24:  For each $V \in$ adjacent $(U, \mathcal{G}_s)$ do
25:    If $loop\_free\ (path(U)(i)(j) + (U,V)$ and $w_{UV}(path(U)(i)(j)) + bw \leq C_{UV}$ then
26:      $\pi \leftarrow path(U)(i)(j) + (U,V)$
27:      If $count\,(V,i) < K$ then
28:        $count(V,i) \leftarrow count(V,i) + 1$
29:        $path(V)(i)(count(V,i)) \leftarrow \pi$
30:        $queue.add(V, i, count(V,i))$
31:      Else
32:        $d \leftarrow index\_highest\_cost(path(V)(i))$
33:        If $w_0(\pi) < w_0(path(V)(i)(d))$
34:          $path(V)(i)(d) \leftarrow \pi$
35:          $queue.update(V, i, d)$
36:        End If
37:      End If
38:    End If
39:  End For
40: End While
41: return $\{\}$
```

i. K-shortest paths: The solution to VNF-PC problem is given by the shortest path which satisfies the node and link constraints in the multi-partite graph. Which means that the optimal solution can be obtained by calculating the K shortest paths with K sufficiently large. We recall that the computation of the K-shortest paths requires the storage of up to K paths on each node.

ii. Selection of the path to be stored: An effective heuristic should not only find a trade-off between the desired quality of solutions and the computation time but it should also better explore the solution space by focusing more on the promising areas. In our proposal, we have set K to a constant and chose to store on each node the best K paths minimizing our objective which corresponds to the SFC cost embedding. We note that it is possible to optimize others objectives such as parameters related to load balancing ($\max W_i(\pi)/C_i$) or a combination of several metrics.

iii. Constraint verification at each path computation step: For efficiency, only paths satisfying the constraints are stored and explored.

iv. Loop detection always limited to a single partition in the multi-partite graph.

v. No use of path dominance: Unlike most algorithms that solve the constrained shortest paths problem, our heuristic does not use path dominance for effective computations. Indeed, except for the paths traversing the same links in the multi-partite graph, no path can dominate another one (because each link and each node are associated with distinct components in the weight vector).

Algorithm 3 summarizes the steps of our Constrained Shortest Path-base Heuristic in multi-Partite Graph (CSPH-PG). This latter is based on a modified version of Dijkstra's algorithm in which up to K best paths verifying the constraints are explored on each node.

In our algorithm, each node U^i in the multi-partite graph (represented by an array $U(i)$) is associated with a list of paths $path(U)(i)$ where the j^{th} path corresponds to $path(U)(i)(j)$. Like Dijkstra's Algorithm, our heuristic starts by adding an empty path to the source node S_s^0 (instruction 1). The number of stored paths on each node is also initialized to zero in line 2 before adding the tuple $(S_s, 0, 1)$ to the heap identified by the variable *queue* (line 3). As described in the algorithm, we used a tuple (U, i, j) to identify one node U^i in the multi-partite graph and one stored path $path(U)(i)(j)$ in the list $paths(U)(i)$.

In line 5, our algorithm removes from *queue* the tuple (U, i, j) that is associated with the least costly path. If the tuple identifies the target $T_s^{|\mathcal{V}_f|-2}$, the algorithm stops by returning the optimal path $path(T_s)(|\mathcal{V}_f|-2)(j)$, otherwise the adjacent nodes to U are explored. More specifically, for a server node U that is capable to run VNF v_{i+1} (i.e., *offers* (U, v_{i+1}) is true), a new path $path(U)(i)(j)+(U,U)$ and its corresponding tuple are generated and respectively added to the list $path(U, i+1)$ and *queue* if the constraint associated with the reflexive link (U,U) is verified (lines from 11 to 14). Obviously, if the list of paths $path(U)(i+1)$ was already full then the new path will replace the most expensive path both in the list and in *queue* (lines from 15 to 22).

In the next step of our algorithm, the adjacent links (U,V) to node U in $\mathcal{G}s$ are explored (lines from 24 to 39). If the resulting path $path(U)(i)(j)+(U,V)$ is loop free and it verifies the constraint C_{UV} (line 25), the new path and its corresponding tuple are respectively added to list $path(V)(i)$ and heap *queue*. As explained previously, for a full path list $path(V)(i)$, the new path will replace the most expensive path both in the list and in *queue* (lines from 31 to 38).

Finally, steps from 4 to 40 are repeated until an optimal path is determined (line 10) or the heap *queue* is empty. In this last case, an empty path indicating that the algorithm doesn't find a solution is returned (line 41).

7.3.1.1 Complexity

The complexity of the Constrained Shortest Path-base Heuristic in multi-Partite Graph (CSPH-PG) can be analyzed as follows:

i. Instructions 1 and 3 are performed in $\mathcal{O}$ (1) whereas instruction 2 is accomplished in $\mathcal{O}$ $(|\mathcal{V}_f|)$.

ii. Instructions inside the while loop (line 4) are executed $K(|\mathcal{V}_f|-1)|\mathcal{V}_s|$ times at most.
When *queue* is implemented with Fibonacci heap, instruction 5 is performed in $\mathcal{O}$ $(\log(K|\mathcal{V}_f||\mathcal{V}_s|))$.
Like implementations of the Dijkstras's algorithm, the path concatenation can be performed in $\mathcal{O}(1)$ by simply pointing the successor of each link. As a result, instructions 6 and 7 can be performed in $\mathcal{O}$ (1).

Complexity of the statement in line 9 is dominated by the capacity constraint verification which holds in $\mathcal{O}\ (|\mathcal{V}_f||\mathcal{V}_s|)$.

Complexity of the bloc of instructions from 10 to 21 is dominated by the if statement in line 17. It corresponds to $\mathcal{O}\ (|\mathcal{V}_f||\mathcal{V}_s|$

To summarize, lines from 5 to 22 can be executed with a complexity of $\mathcal{O}\ |(\mathcal{V}_f||\mathcal{V}_s|+\log K)$

iii. Instructions inside the *for loop* (line 24) are executed $K|\mathcal{V}_f|(2|\mathcal{E}_s|)$ times at most. The procedure *loop_free* only treats path nodes in $\mathcal{V}_p^i$ so it holds in $\mathcal{O}\ (|\mathcal{V}_s|)$. The capacity constraint verification on link (U,V) can be performed in $\mathcal{O}\ (|\mathcal{V}_f||\mathcal{V}_s|)$. The complexity of the bloc of instructions from 26 to 38 is $\mathcal{O}\ (|\mathcal{V}_f||\mathcal{V}_s|)$.

iv. The overall complexity is given below:
$\mathcal{O}\ (K(|\mathcal{V}_f|-1)|\mathcal{V}_s|\,(|\mathcal{V}_f||\mathcal{V}_s|+\log K)+K|\mathcal{V}_f|(2|\mathcal{E}_s|)\,(|\mathcal{V}_f||\mathcal{V}_s|) = \mathcal{O}\ (K|\mathcal{V}_f|^2|\mathcal{E}_s||\mathcal{V}_s|+(K\log K)|\mathcal{V}_f||\mathcal{V}_s|)$
For a constant number of stored path in each node, the resulting complexity is:

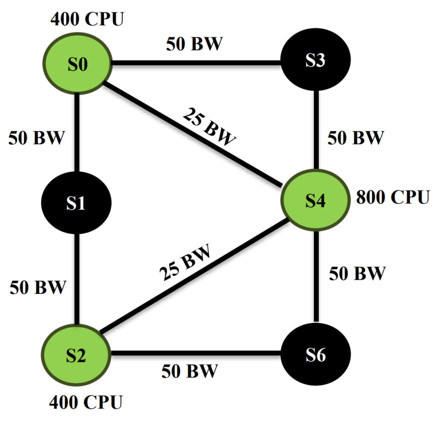
$$\mathcal{O}(|\mathcal{V}_f|^2|\mathcal{E}_s||\mathcal{V}_s|)$$

7.4 Performance Evaluation

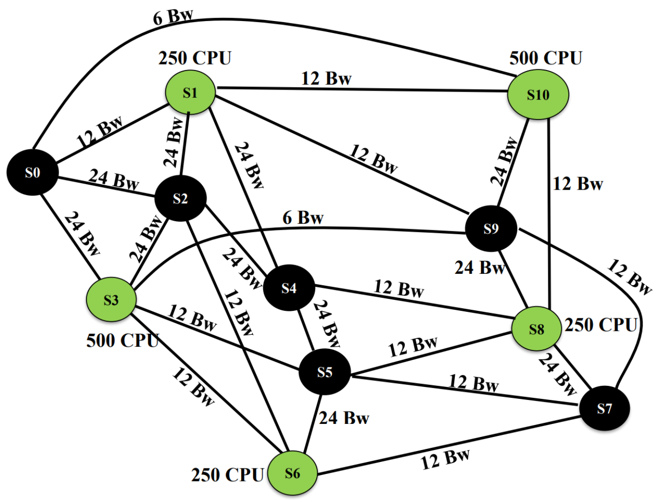
Through these experiments, we want to show the efficiency of our heuristic and its ability to determine near-optimal solutions with finite and small values of K (K is the maximum number of paths on each node). Thus, we deliberately chose to compare the performance of our heuristic with the algorithm giving optimum. The experiment has another objective consisting in evaluating the convergence speed towards the optimum of our heuristic with the increase of K.

We implemented a simulation environment where the arrival and lifetime of the SFC requests follow respectively a Poisson process (with different means λ) and uniform law ($\mathcal{U}(10,20)$). Each experiment lasts for 100,100 units and depicted metrics correspond to mean values over the last 100,000 units to avoid side effects due to start-up period.

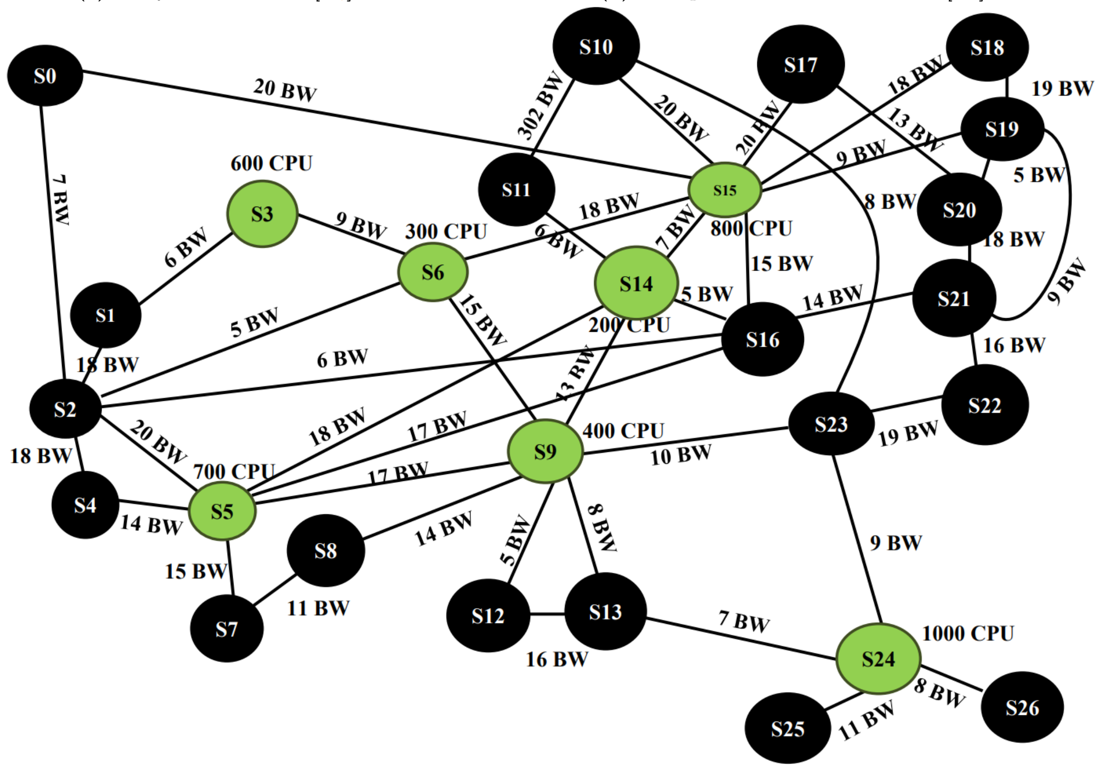
Below, we describe in more details the experiment environment, metrics and scenarios.



(a) *Artificial Network [84]*



(b) *European Network Cost 239 [85]*



(c) *US Longhaul network topology [86]*

Figure 7.3: *NFV Infrastructures*

7.4.1 Simulation Environment and scenarios

For our experiments, we used the 3 networks (NFV-Is) depicted in Figure 7.3: (1) a small artificial network that is generated from a network in [84], (2) European network [85] (Cost 239) of middle size that is very connected, and (3) US Longhaul network [86] which is a large network of intermediate connectivity. The first topology allows us to determine the minimal values of K permitting to obtain optimal solution whereas we used the two last networks to measure the convergence speed and capacity of our heuristic to scale. We note that the labels in Figure 7.3 correspond to the resource capacities of links and nodes of the corresponding networks.

Similarly, we used various sizes of SFCs to evaluate the performance and ability of our heuristic to effectively provision SFCs of different sizes, especially the larger ones. The CPU demands of VNFs are uniformly distributed in interval $(10, 20)$ whereas the bandwidth demands of SFC links are chosen in interval $(1, 6)$.

Accordingly, we considered 3 main experiments, each one with different scenarios as described below:

7.4.2 Experiments and Scenarios

i. **Experiment 1:** The NFV-Infrastructure used is the artificial network shown in Figure 7.3a [84]. This network consists in 6 switches where only 3 among them (gray switches) are directly connected to server nodes.

   (a) **Scenario 1.1:** To account for different network loads, we vary λ from 0.2 to 1 (with a step of 0.2). The SFC size is fixed at 6.

   (b) **Scenario 1.2:** The SFC sizes vary from 2 to 6 with a step of 1 whereas the arrival rate λ is fixed and equal to 1. The objective of this sub-scenario is to measure the impact of SFC size increase on the performance of our heuristic.

ii. **Experiment 2:** Two NFV-Infrastructures (European Cost239 Network and the US Long Haul Network) are used in this scenario. The main objective of this scenario is to show that our heuristic quickly converges with the augmentation of K.

   (a) **Scenario 2.1:** λ evolves from 0.2 to 1 (with a step of 0.2) on the Cost239 infrastructure, and from 1 to 4 (with a step of 0.5) on the Long Haul infrastructure. The SFC size is fixed at 6.

   (b) **Scenario 2.2:** On the Cost239 infrastructure, the SFC size varies from 2 to 12 with a step of 1 while λ is fixed at 1. On the Long Haul infrastructure, the SFC size varies from 2 to 12 with a step of 1 whereas λ is fixed at 2.

7.4.3 Metrics

We chose 4 quality metrics to measure the performance of our proposal:

Table 7.1: *Ratio of accepted requests on Artificial Network (SFC size = 6)*

λ	K = 1	K = 10	K = 100	K = 1000	Optimum
0.2	99,85%	99,85%	99,85%	99,85%	99,85%
0.4	95,88%	96,48%	96,53%	96,53%	96,53%
0.6	90,76%	92,13%	92,30%	92,32%	92,32%
0.8	83,39%	85,43%	85,72%	85,72%	85,72%
1	76,96%	79,33%	79,74%	79,77%	79,78%

i. **Ratio of accepted requests (RA)**: This metric determines the proportion of SFC requests which are accepted. It is calculated by dividing the number of accepted requests by the total number of received requests. Generally, service providers are required to provide a very high acceptance rate to avoid penalties.

ii. **Average Cost of accepted SFCs (AC):** The AC metric measures the average cost of the SFCs successfully placed. It is determined by dividing the total mapping cost of accepted SFCs by their total number. We recall that both the node and link resource are considered to determine the mapping cost (c. f., equations (7.1) and (7.2)). Obviously, the lower the AC, the more efficient the resource utilization.

iii. **Derivative Function (DF)**: The DF metric is a crucial indicator of convergence, providing insight into the stability of the solution. It is calculated using the formula:

$$\delta(K_1, K_2) = \frac{Mean\ Cost(K_2) - Mean\ Cost(K_1)}{K_2 - K_1}$$

where Mean Cost (K_2) and Mean Cost (K_1) respectively correspond to the mean costs obtained for K $=K_2$ and K $=K_1$.

iv. **Path Length of SFCs (PL)** :The PL metric determines the average length of the substrate paths connecting source and target servers. Generally, the greater the path length, the greater the number of paths explored and the higher the execution time.

7.4.4 Experiment Results

7.4.4.1 Experiment 1

Tables 7.1, 7.2 and Figures 7.4, 7.5 depict the results obtained on the artificial small networks. As expected and regardless of values of K, the higher the arrival rates and

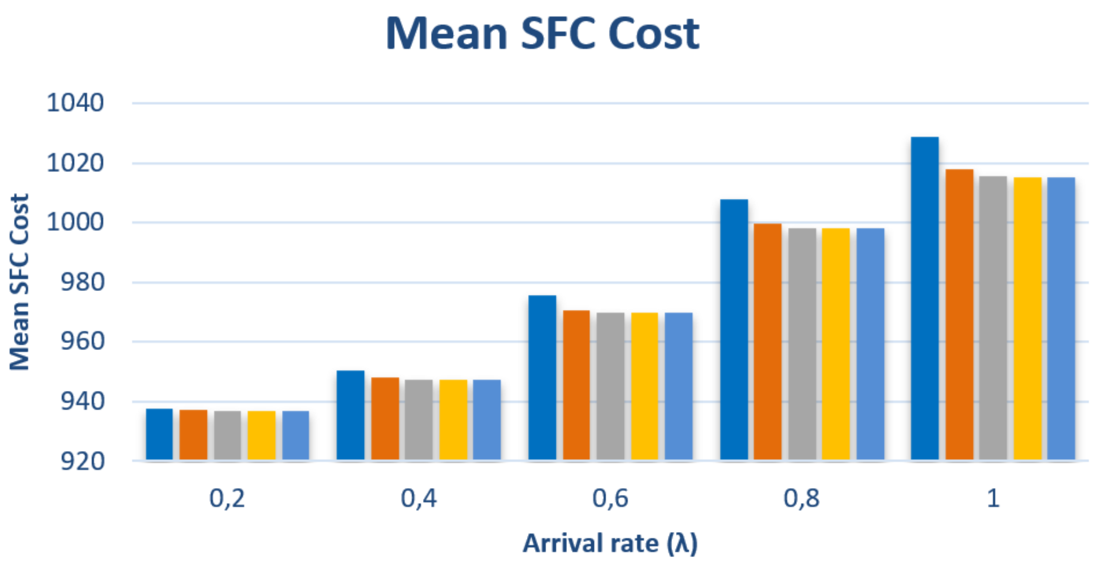


Figure 7.4: *AC for various loads on Artificial Network*

Table 7.2: *Ratio of accepted requests on Artificial Network (λ=1)*

SFC Size	K=1	K=10	K=100	K=1000	Optimum
2	92,85%	93,23%	93,23%	93,23%	93,23%
3	87,59%	88,68%	88,75%	88,75%	88,75%
4	85,47%	87,03%	87,19%	87,20%	87,20%
5	84,98%	86,57%	86,85%	86,86%	86,86%
6	76,96%	79,33%	79,74%	79,77%	79,78%

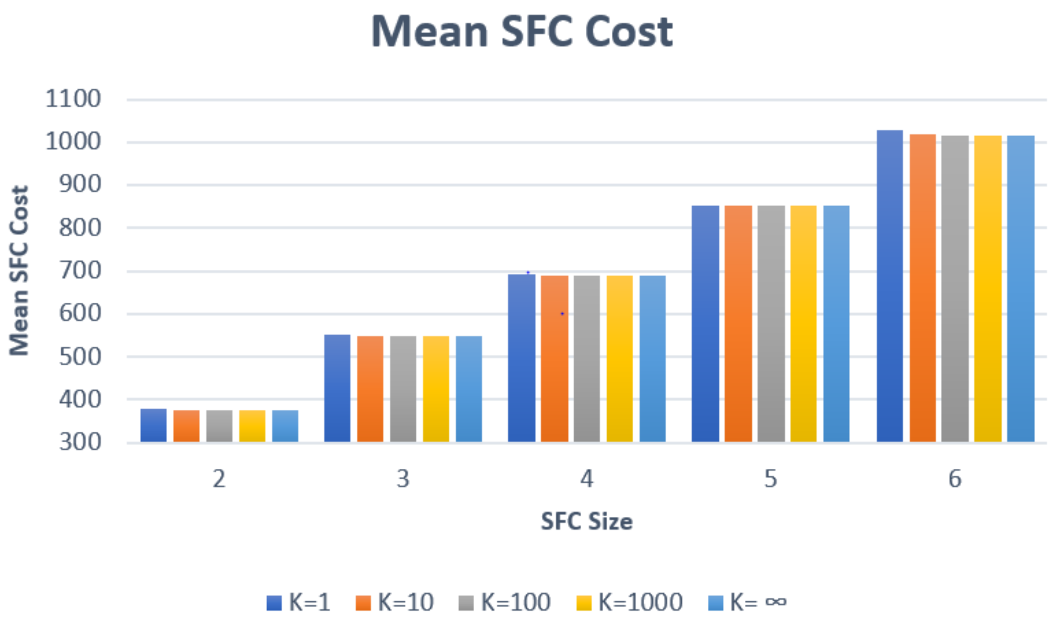


Figure 7.5: *AC for different SFC sizes on Artificial Network*

Table 7.3: *Minimum values of K for optimal solutions on Artificial Network (SFC size = 6)*

λ	K=1	K $\in$ (1,2)	K $\in$ (1,9)	K $\in$ (1,99)
0.2	94,7%	99,23%	99,89%	100%
0.4	88,74%	89,47%	99,1%	99,95%
0.6	80,92%	90,93%	97,83%	99,82%
0.8	74,46%	86,51%	96,63%	99,78%
1	70,75%	83,17%	95,36%	99,61%

Table 7.4: *Minimum values of K for optimal solutions on Artificial Network (λ = 6)*

SFC size	K=1	K $\in$ (1,2)	K $\in$ (1,9)	K $\in$ (1,99)
2	94,14%	99,17%	99,99%	100%
3	87,68%	95,1%	99,28%	99,99%
4	85,4%	93,74%	98,86%	99,96%
5	84,52%	93,15%	98,5%	99,94%
6	70,75%	83,17 %	95,35%	99,61%

SFC sizes, the lower the ratios of accepted requests and the higher the averages of SFC costs.

Even for high loads of networks, Table 7.1 and Figure 7.4 clearly show that the performance of our heuristic quickly converge to the optimum with the increase of K. Clearly, the performance of our heuristic reaches near-optimal solution as soon as K increases and reaches the value of 10.

Similarly, we observe the same behaviour as that described previously in Table 7.2 and Figure 7.5 which show the experiment results for SFC sizes evolving from 2 to 6.

To better understand the previous results, we determined for each experiment scenario the minimal values of K permitting to obtain optimal solutions. As shown in Tables 7.3 and 7.4, the most of the values of K permitting to obtain optimal solutions are very small. For instance, for $K < 10$, our heuristic determines optimum solutions in more than 95% of cases.

In fact, for high and practical ratios of accepted requests (e.g. $RA > 95\%$), the number

Table 7.5: *Ratio of accepted requests on Cost239 (λ evolves from 0.2 to 1)*

λ	K = 1	K = 10	K = 100	K = 1000
0.2	100,00%	100,00%	100,00%	100,00%
0.4	99,20%	99,33%	99,33%	99,33%
0.6	94,82%	95,34%	95,38%	95,38%
0.8	87,60%	88,52%	88,83%	88,88%
1	80,70%	82,18%	82,42%	82,51%

of paths to be explored in order to determine feasible solutions is low compared to those obtained with intermediate network loads. For instance, the storage of only one path allows to determine the optimum for very low network loads as stated by Lemma 7.1. Thanks to our heuristic which privileges and first explores the best paths in terms of cost, the storage of a small number of paths is often sufficient to connect the source and target servers, and thus determine an optimal solution.

7.4.4.2 Experiment 2

In Experiment 2, we conducted a comprehensive performance analysis for Scenario 2, focusing on two distinct network types: the Cost239 network [85] and the Long Haul network [86]. The former is a network of medium size whereas the last one is large.

Table 7.5 and Table 7.6 provided valuable insights into the acceptance rate. As λ increased, there was a noticeable decrease in the ratio of accepted requests, suggesting potential congestion or resource constraints.

Additionally, it's crucial to emphasize the impact of the parameter K on the acceptance rate. The higher is K, the lower are the improvements obtained by increasing K. As a results, for quick computation of good quality path, the increase of K should be reserved to the small values.

For instance, in our simulations, augmenting K beyond $K = 10$ does not yield significant improvements in the acceptance rate. Beyond $K = 10$, we observe significant diminishing returns in terms of enhancements in the acceptance rate.

In practical scenarios, the heuristic demonstrated rapid convergence to near-optimal acceptance rates, even with relatively small values of K ($K < 10$). This highlights the influence of K on the acceptance rate and computational time.

Turning to graphical representations, Figure 7.6 provided additional insights. The left graph explored the relationship between λ and SFC costs, with a clear upward trend as λ increased, indicating higher request arrival rates led to increased costs. This pattern held across various values of K. Conversely, the right graph of Figure 7.6 delved into mean SFC path lengths concerning varying values of λ and K. Here, we observed a slight increase in mean SFC path lengths as λ increased from 0.2 to 1, suggesting longer paths for service function chains with higher request arrival rates. The impact of K on path lengths is clearly perceptible.

Analyzing Figure 7.7, which presented mean SFC costs and mean SFC path lengths, we noted an upward trend in mean SFC costs as λ increased from 1 to 4 across all values of K. However, there was small variation in mean SFC costs as K increased, especially for high values of K, highlighting K's substantial influence on costs. Similarly, the right graph of Figure 7.7, which showcased mean SFC path lengths, indicated a slight increase as λ increased from 1 to 4. Again, K played a small and noticeable role in determining the lengths of service function chains.

Figures 7.8 and 7.9 provided further insights, illustrating mean SFC path lengths and mean SFC costs at a constant request arrival rate ($\lambda = 1$ for Cost and $\lambda = 2$ for Long Haul) while varying the number of paths stored at each node (K). The right graphs of Figures 7.8 and 7.9 demonstrated that as K increased, a small and noticeable trend of decreasing mean SFC path lengths emerged, suggesting that a larger pool of paths improved the heuristic's ability to discover shorter, more efficient paths. This emphasized the trade-off between computational resources and solution quality when selecting an appropriate K value.

The left graphs of Figure 7.8 and 7.9 explored the relationship between SFC size and mean SFC costs, revealing an upward trend as the SFC size increased. This implied that larger service function chains generally incurred higher associated costs. However, increasing the number of paths stored at each node (K) tended to decrease mean SFC costs, indicating that a larger path pool facilitated the discovery of more cost-efficient solutions.

It is important to emphasize that in these two scenario cases we face the challenge of defining a single optimal value of K, mainly due to the large size of the NFV infrastructure and SFC. Our approach focused on a detailed examination of the heuristic's behavior across different metrics. To achieve this, we used a differential function (DF), as shown in figure 7.10 and figure 7.11. This tool provides valuable insight into how adjusting K affects performance metrics and allows you to make informed decisions regarding the selection of an appropriate K.

The derivative function (DF) results have unveiled an intriguing pattern. As illustrated in Figure 7.10 and Figure 7.11, we have made notable observations concerning the choice of K. For small values of K ($K < 10$), we observe a swift convergence towards near-optimal solutions. However, as K increases beyond this threshold, the improvements in performance become very low and imperceptible. This observation underscores the pivotal role played by K in enhancing the performance of our heuristic, particularly to determine the best values which reduce the computations while guaranteeing satisfying

Table 7.6: *Ratio of accepted requests on Long Haul (λ evolves from 1 to 4)*

λ	K = 1	K = 10	K = 100	K = 1000
1	99,90%	99,90%	99,90%	99,90%
1.5	99,18%	99,20%	99,21%	99,21%
2	95,89%	96,18%	96,28%	96,34%
2.5	88,27%	89,17%	89,53%	89,75%
3	79,12%	80,67%	81,41%	81,92%
3.5	69,76%	72,00%	73,02%	73,57%
4	61,38%	64,01%	65,17%	65,96%

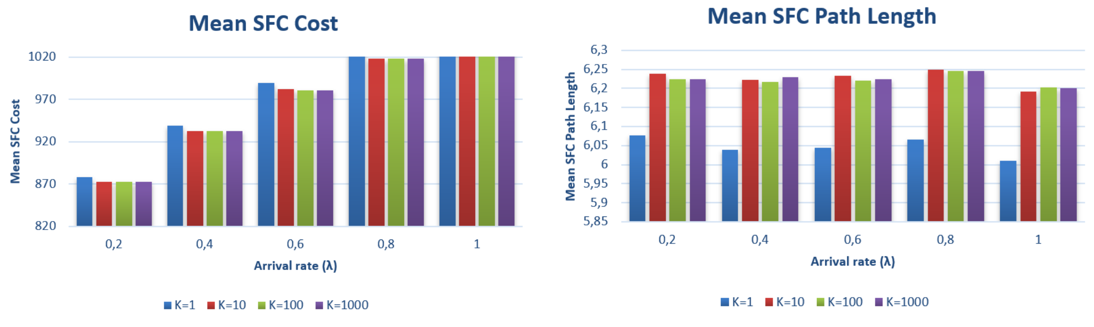


Figure 7.6: *Results of the 2.1 scenario on Cost239*

solution quality.

It's worth noting that these findings align with the outcomes reported in Table 7.4, where we determined that K = 10 is sufficient to determine the best solutions. Indeed, the DF exhibits rapid growth before reaching the K value of 10, followed by slower and imperceptible increase of the DF after the point $(\delta(10, 100))$. Performance stabilizes effectively starting from K = 10. Furthermore, for the small network topology [84], it's noteworthy that K = 1 often allowed us to find the optimal solution in more than 70% of cases, and this stability in solution quality remained consistent even for the larger network, as shown in our simulation.

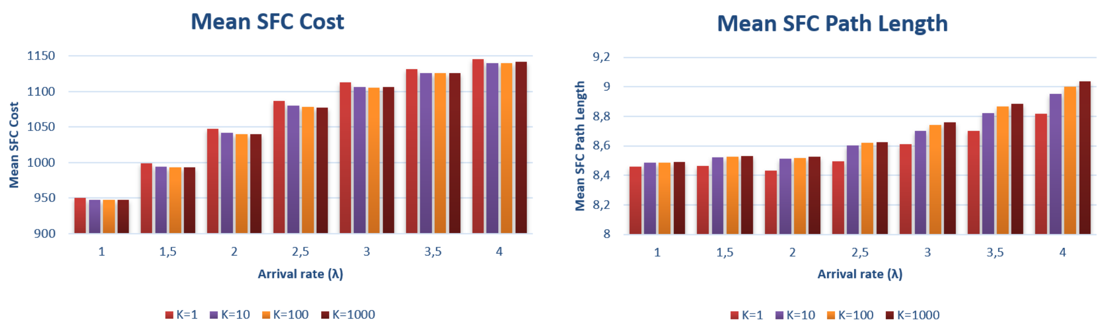


Figure 7.7: *Results of the 2.1 scenario on Long Haul*

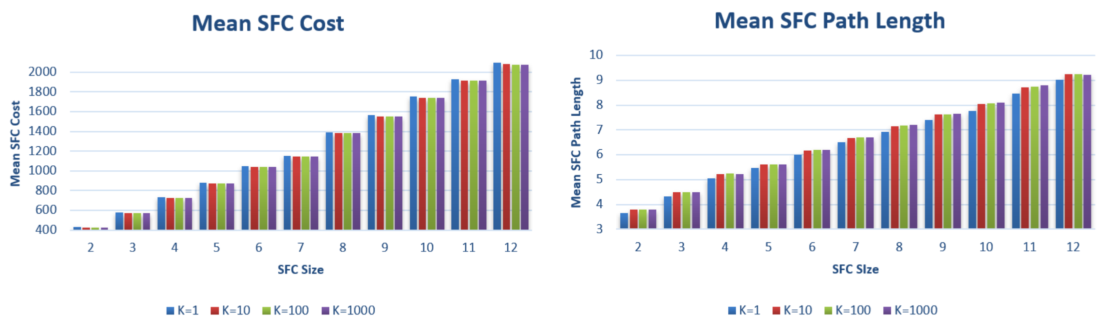


Figure 7.8: *Results of the scenario 2.2 on Cost239 - Lambda=1*

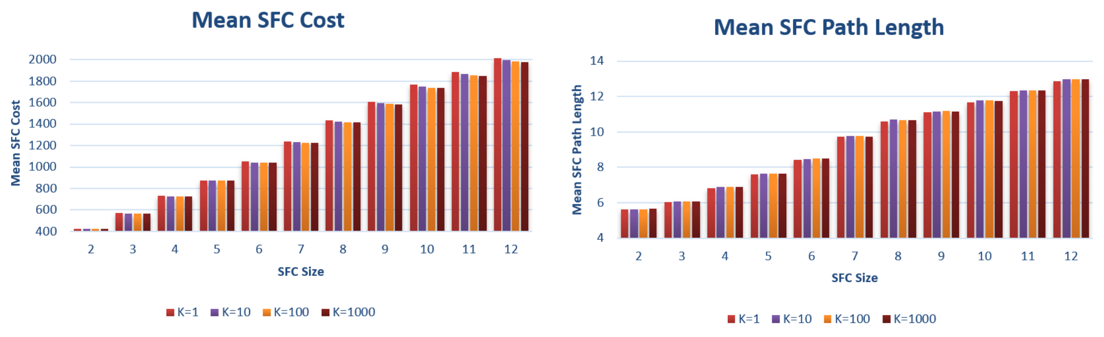


Figure 7.9: *Results of the scenario 2.2 on Long Haul - Lambda=2*

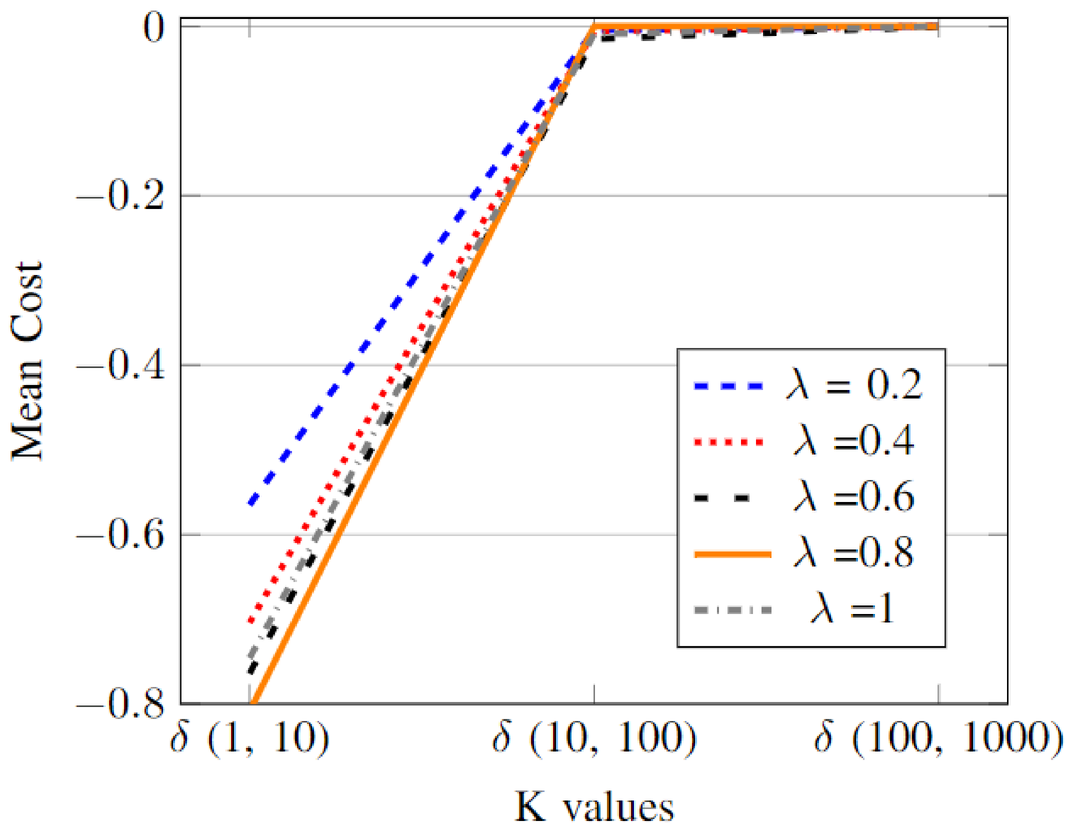


Figure 7.10: *Slope of the mean cost for different K and lambdas values (Cost infrastructure)*

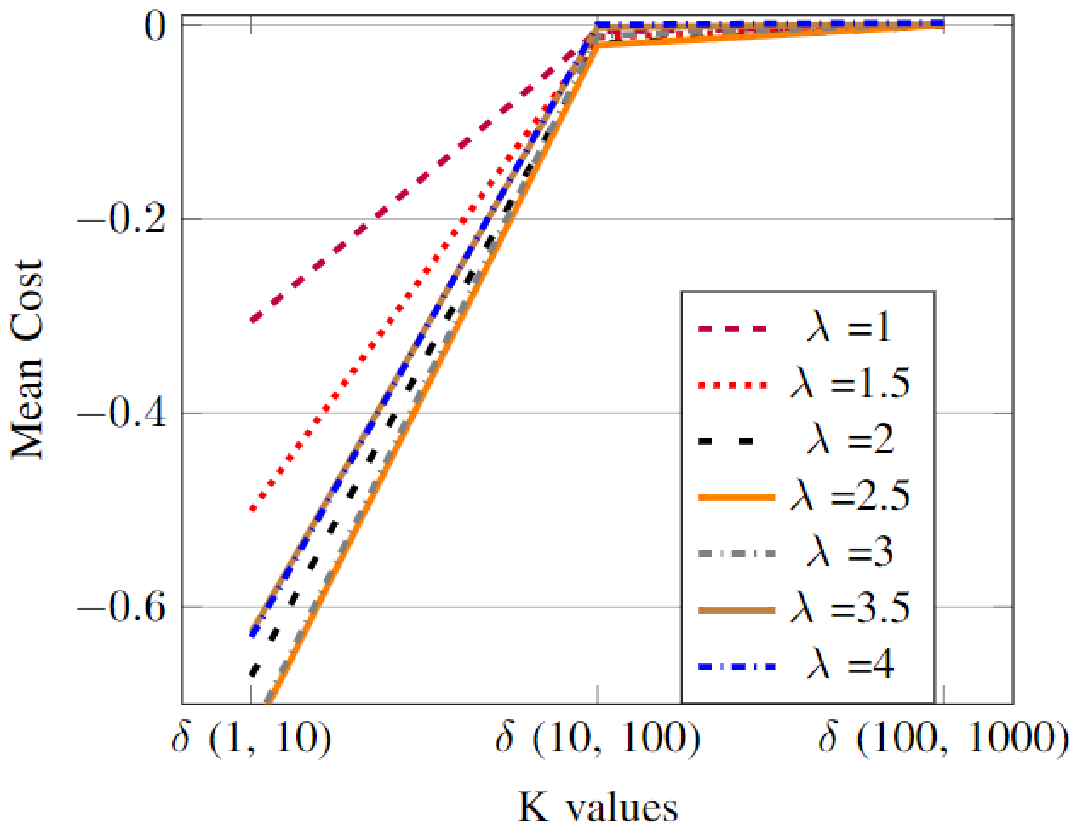


Figure 7.11: *Slope of the mean cost for different K and lambdas values (Long Haul infrastructure)*

7.5 Conclusion

In this chapter, we have embarked on a comprehensive exploration of the Virtual Network Function (VNF) placement and chaining problem, aiming to minimize the allocation costs associated with incoming Service Function Chains (SFCs), especially in the context of heightened demand and traffic loads within NFV-Infrastructures (NFV-Is).

Our approach in this chapter has revolved around the in-depth analysis of the Constrained Shortest Path-base Heuristic in multi-Partite Graph (CSPH-PG). This strategic choice stems from our emphasis on scalability and efficiency, particularly in larger NFV-Is and extensive SFC scenarios.

By addressing the core concepts of VNF placement and chaining problem in unloaded networks, we provided a solid foundation for subsequent discussions. The introduction of the constrained shortest path algorithm was crucial in aiming to solve the VNF-PC problem. This algorithm forms the backbone of the heuristic approach and enables cost-effective SFC deployment.

This chapter represents an extension and refinement of our previous work of the Chapter 5, aiming to provide efficient solutions to the VNF-PC problem while accommodating the diverse challenges posed by varying NFV-I sizes and traffic demands.

Our experimental results and evaluation revealed the effectiveness and scalability of the proposed heuristic. They have demonstrated its ability to minimize allocation costs for incoming SFCs, making it particularly well-suited for large-scale NFV-I scenarios.

CHAPTER 8

Conclusions and Future Research Directions

This most beautiful system of the sun, planets, and comets, could only proceed from the counsel and dominion of an intelligent being.

Isaac Newton

Abstract

This chapter summarizes the research work and contributions of the thesis on VNF placement and chaining problem. The chapter then outlines some directions for future research.

Chapter content

8.1 Introduction

In this last chapter, we conclude the thesis by summarizing our main contributions and presenting some future research directions and perspectives. Fulfilling the promises of Network Functions Virtualization (NFV) technology requires addressing new challenges. The main challenge of this thesis is to design an efficient VNF placement and chaining algorithm over a virtualized infrastructure while meeting special requirements such as optimizing the total allocation cost and respecting the resources availability of the NFV-I. Below is a brief summary of our contribution.

8.2 Summary of contributions

In conclusion, this thesis has made significant contributions to the field of resource allocation optimization in Network Function Virtualization (NFV) environments. By addressing the VNF placement and chaining problem, the objective was to optimize resource utilization and minimize costs and cost-effectiveness of network operations.

Chapter 2 served as a crucial introduction, providing essential contextual information on NFV and SDN. NFV, the virtualization of network functions, and SDN, the separation of the network's control and data planes, are fundamental concepts in modern networking. The chapter explored resource allocation strategies and highlighted the challenges encountered in cloud environments where NFV and SDN are commonly deployed. By gaining a comprehensive understanding of these concepts and challenges, the subsequent chapters were able to propose innovative approaches and algorithms for resource allocation optimization.

Chapter 3 focused on the VNF placement and chaining problem, aiming to minimize SFC allocation costs. The chapter introduced an Integer Linear Programming (ILP) model as a novel approach to optimize the placement and chaining of Virtual Network Functions (VNFs). By formulating the problem as an ILP, it became possible to find an optimal solution that minimizes the costs associated with SFC allocation and ensures efficient utilization of available resources. Furthermore, this chapter includes a supplementary section that explores the utilization of Combinatorial Benders Decomposition (CBD) to augment the effectiveness of VNF-PC problem-solving. CBD harnesses the capabilities of identifying Irreducible Infeasible Sets (IIS) and implementing combinatorial cuts, thereby enhancing the efficiency of the solution process. This addition leads to a more profound comprehension of the problem and its prospective resolutions. The incorporation of CBD enriches the chapter's examination of advanced techniques for addressing intricate NFV optimization challenges.

Chapter 4 extended the study to the unloaded network scenario, where an abundance of resources is available on nodes and links. To address this scenario, a graph transformation and Shortest Path algorithm were proposed, effectively allocating resources and minimizing costs. This extension provided valuable insights into resource allocation strategies in unloaded networks and opened up possibilities for optimizing resource utilization even in scenarios with abundant resources.

Chapter 5 tackled a specific variation of the VNF placement and chaining problem in networks characterized by limited node resources but abundant link resources. To address this challenge, a knapsack-based genetic algorithm was proposed. This algorithmic approach combined the principles of knapsack optimization and genetic algorithms to optimize resource allocation and minimize costs. By iteratively refining candidate solutions using genetic operators like selection, crossover, and mutation, the algorithm explored a diverse set of potential configurations, ultimately minimizing the overall placement cost. This contribution provided a valuable tool for optimizing resource allocation in scenarios where node resources are limited, but link resources are abundant.

Chapter 6 addressed the full generic version of the VNF placement and chaining problem, considering limited node and link resources. To tackle this challenge, a multi-constrained routing algorithm was employed. This algorithmic approach took into account multiple constraints, such as computational capabilities and bandwidth requirements, to optimize the placement and chaining of VNFs. By systematically exploring potential solutions that satisfied all constraints while minimizing the overall placement cost, the algorithm enabled efficient resource allocation in NFV environments. This contribution was particularly significant as it considered the complexities arising from limited node and link resources, offering insights into optimizing resource allocation under various constraints.

Chapter 7 makes several significant contributions to the field of Virtual Network Function (VNF) placement and chaining. Firstly, it conducts an in-depth analysis of the VNF placement and chaining problem within an unloaded network, providing a comprehensive understanding of its intricacies. Secondly, it introduces the constrained shortest paths algorithm, a pivotal tool for solving the VNF-PC problem enabling cost-effective provisioning of Service Function Chains (SFCs). Furthermore, this work represents an extension and refinement of the previous chapter's approach, offering a more comprehensive exploration of the Constrained Shortest Path-base Heuristic in multi-Partite Graph (CSPH-PG) and addressing various challenges associated with different NFV-Infrastructure (NFV-I) sizes and traffic demands. Emphasizing scalability and efficiency, the chapter highlights the CSPH-PG heuristic's effectiveness for larger NFV-Is and extensive SFC scenarios, making it well-suited for handling high traffic loads. Ultimately, this chapter equips network engineers and researchers with a versatile toolkit for addressing VNF placement and chaining challenges, enhancing their ability to optimize resource allocation and minimize allocation costs in modern network infrastructures.

In summary, this thesis provided valuable insights, novel frameworks, and algorithmic approaches for resource allocation optimization in NFV environments. By addressing various versions of the VNF placement and chaining problem and considering different resource scenarios, the thesis contributed to the understanding and techniques for optimizing resource allocation.

8.3 Future Research Directions

In terms of future perspectives, our thesis opens up several potential directions for further research and development of algorithms and mechanisms. Some suggested areas for future work include:

i. **Enhancing Performance Evaluation**: To gain deeper insights into the performance of the proposed algorithms and heuristics, it is recommended to conduct more comprehensive simulations. By exploring various network conditions and conducting exhaustive evaluations, we can better understand the limitations and effectiveness of the proposed solutions.

ii. **Resilient VNF-PC**: Another interesting avenue for research is the exploration of resilient SFC placement. This involves considering the possibility of SFC placement failures within the NFV infrastructure and developing strategies to mitigate such failures. By addressing the resilience aspect, we can enhance the robustness and reliability of SFC provisioning.

iii. **Integrating Traffic Predictions**: Integrating traffic prediction techniques into SFC placement can be highly beneficial. By leveraging predictive models to anticipate future demands in terms of SFC flows, it becomes possible to maintain network stability and optimize resource allocation proactively. This area offers opportunities to improve overall network performance and user satisfaction.

iv. **Integrating Machine Learning for Fault Prediction**: The integration of machine learning technology into fault prediction can significantly enhance serviceability. By collecting information on network failures over time, it becomes possible to identify the most vulnerable network elements, such as VNFs or Points of Presence (PoPs). By incorporating availability-aware placement and chaining algorithms, it becomes feasible to predict and prevent future failures, minimizing service interruptions.

These future research directions have the potential to further advance the field of VNF placement and chaining, optimizing resource allocation, and improving the overall performance and reliability of NFV environments.

Publications

Based on the research work presented in this thesis, some papers have been published in international conferences:

i. **International Conferences:**

1. Issam Abdeldjalil Ikhelef, Saidi Mohand Yazid, Li Shuopeng, and Chen Ken, “A Knapsack-based Optimization Algorithm for VNF Placement and Chaining Problem”. In: 2022 IEEE 47th Conference on Local Computer Networks (LCN).
2. Issam Abdeldjalil Ikhelef, Saidi Mohand Yazid, Li Shuopeng, and Chen Ken, “Multi-Constrained Routing-based Heuristic for VNF Placement and Chaining”. In: ICC 2023 - IEEE International Conference on Communications.
3. Issam Abdeldjalil Ikhelef, Warwicker John Alasdair, Rebennack Steffen, Saidi Mohand Yazid, and Chen Ken, “Efficient Decomposition-Based Methods for Optimal VNF Placement and Chaining”. In: 2023 24st Asia-Pacific Network Operations and Management Symposium (APNOMS).

Submissions

Based on the research work presented in this thesis, a journal paper has been submitted to an international journal:

i. **International Journal:**

   1. Issam Abdeldjalil Ikhelef, Saidi Mohand Yazid, Li Shuopeng, and Chen Ken, "Constrained Routing in Multi-Partite Graph to Solve VNF Placement and Chaining Problem." Submitted to the Journal of Network and Computer Applications - 2023.

Bibliography

[1] Omar Houidi. "Algorithms for Virtual Network Functions chaining". PhD thesis. June 2020 *Cited on pages 3, 23.*

[2] Bo Han et al. "Network function virtualization: Challenges and opportunities for innovations". In: *IEEE Communications Magazine* 53.2 (2015), pp. 90–97. DOI: 10.1109/MCOM.2015.7045396 *Cited on page 3.*

[3] Fei Hu, Qi Hao, and Ke Bao. "A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation". In: *IEEE Communications Surveys Tutorials* 16.4 (2014), pp. 2181–2206. DOI: 10.1109/COMST.2014.2326417 *Cited on page 4.*

[4] Karamjeet Kaur, Veenu Mangat, and Krishan Kumar. "A comprehensive survey of service function chain provisioning approaches in SDN and NFV architecture". In: *Comput. Sci. Rev.* 38 (2020), p. 100298 *Cited on pages 4, 24.*

[5] Bo Yi et al. "A comprehensive survey of Network Function Virtualization". In: *Comput. Networks* 133 (2018), pp. 212–262 *Cited on page 4.*

[6] Sang Il Kim and Hwa-sung Kim. "A VNF Placement Method based on VNF Characteristics". In: *2021 International Conference on Information Networking (ICOIN)* (2021), pp. 864–869 *Cited on page 4.*

[7] Deval Bhamare et al. "A survey on service function chaining". In: *J. Netw. Comput. Appl.* 75 (2016), pp. 138–155 *Cited on page 4.*

[8] Md. Faizul Bari et al. "On orchestrating virtual network functions". In: *2015 11th International Conference on Network and Service Management (CNSM)* (2015), pp. 50–56 *Cited on pages 6, 33, 39.*

[9] Issam Abdeldjalil Ikhelef et al. "A Knapsack-based Optimization Algorithm for VNF Placement and Chaining Problem". In: *2022 IEEE 47th Conference on Local*

Computer Networks (LCN). 2022, pp. 430–437. DOI: 10.1109/LCN53696.2022.9843566 *Cited on pages 8, 35, 63.*

[10] Issam Abdeldjalil Ikhelef et al. “Efficient Decomposition-Based Methods for Optimal VNF Placement and Chaining”. In: *2023 24st Asia-Pacific Network Operations and Management Symposium (APNOMS)*. 2023, pp. 89–94 *Cited on pages 8, 35.*

[11] Issam Abdeldjalil Ikhelef et al. “Multi-Constrained Routing-based Heuristic for VNF Placement and Chaining”. In: *ICC 2023 - IEEE International Conference on Communications*. 2023 *Cited on pages 8, 9, 55, 81, 83.*

[12] Issam Abdeldjalil Ikhelef et al. “Constrained Routing in Multi-Partite Graph to Solve VNF Placement and Chaining Problem”. Subbmitted to Journal of Network and Computer Applications - 2023 *Cited on pages 9, 99, 101.*

[13] “The European Telecommunications Standards Institute”. In: () *Cited on pages 17, 18, 20.*

[14] Rashid Mijumbi et al. “Network Function Virtualization: State-of-the-Art and Research Challenges”. In: *IEEE Communications Surveys Tutorials* 18.1 (2016), pp. 236–262. DOI: 10.1109/COMST.2015.2477041 *Cited on pages 17, 19.*

[15] Jose Luis Herrera et al. “NFVI virtualization layer: A survey”. In: *Journal of Network and Computer Applications* 82 (2017), pp. 23–36. DOI: 10.1016/j.jnca.2016.09.008 *Cited on page 19.*

[16] ETSI Industry Specification Group (ISG) Network Functions Virtualization (NFV). “Network Functions Virtualization (NFV); Architectural Framework”. In: *ETSI GS NFV 002* (2013) *Cited on page 20.*

[17] Nick McKeown et al. “OpenFlow: Enabling Innovation in Campus Networks”. In: *ACM SIGCOMM Computer Communication Review* 38.2 (2008), pp. 69–74 *Cited on page 20.*

[18] Paul Quinn and Thomas Nadeau. “Service Function Chaining (SFC) Architecture”. In: *RFC Editor* (2016). RFC 7665 *Cited on page 21.*

[19] P. Quinn. *Internet Engineering Task Force (IETF)*. 2015 *Cited on page 21.*

[20] Mohammad Usman Khan et al. “Software-Defined Networking: A Taxonomy, Survey, and Research Issues”. In: *Journal of Network and Computer Applications* 135 (2019), pp. 62–84. DOI: 10.1016/j.jnca.2019.02.020 *Cited on page 21.*

[21] John Smith and Sarah Johnson. “Software-Defined Networking: Principles and Benefits”. In: *Networking Review* 10.2 (2023), pp. 45–58. DOI: 10.1234/netrev.2023.10.2.45 *Cited on page 22.*

[22] Bruno Astuto Nunes et al. “A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks”. In: *IEEE Communications Surveys & Tutorials* 16.3 (2014), pp. 1617–1634. DOI: 10.1109/COMST.2014.2337766 *Cited on page 22.*

[23] Sherif Abdelwahab et al. “Network function virtualization in 5g”. In: *IEEE Communications Magazine* 54.4 (2016), pp. 84–91 *Cited on page 23.*

[24] Michel S. Bonfim, Kelvin Lopes Dias, and Stenio F. L. Fernandes. “Integrated NFV/SDN architectures: A systematic literature review”. In: *CoRR* abs/1801.01516 (2018) *Cited on page 23.*

[25] ETSI. *Network Functions Virtualisation (NFV)*. https://www.etsi.org/technologies/nfv. Accessed: <insert date> *Cited on page 23.*

[26] ONF. *Software Defined Standards*. https://www.opennetworking.org/software-defined-standards/overview/. Accessed: <insert date> *Cited on page 23.*

[27] DMTF. *Standards and Technology*. https://www.dmtf.org/standards *Cited on page 24.*

[28] Rashid Mijumbi et al. “Network function virtualization: State-of-the-art and research challenges”. In: *IEEE Communications Surveys and Tutorials* 18.1 (2016), pp. 236–262 *Cited on page 25.*

[29] NM Mosharaf Kabir Chowdhury and Raouf Boutaba. “A survey of network virtualization”. In: *Computer Networks* 54.5 (2010), pp. 862–876 *Cited on page 25.*

[30] NM Mosharaf Kabir Chowdhury and Raouf Boutaba. “Network virtualization: state of the art and research challenges”. In: *IEEE Communications magazine* 47.7 (2009), pp. 20–26 *Cited on page 25.*

[31] Md Faizul Bari et al. “Data center network virtualization: A survey”. In: *IEEE communications surveys & tutorials* 15.2 (2012), pp. 909–928 *Cited on page 25.*

[32] Md. Faizul Bari et al. “On orchestrating virtual network functions”. In: Nov. 2015, pp. 50–56 *Cited on page 25.*

[33] Junjie Liu et al. “On Dynamic Service Function Chain Deployment and Readjustment”. In: *IEEE Transactions on Network and Service Management* 14.3 (2017), pp. 543–553. DOI: `10.1109/TNSM.2017.2711610` *Cited on page 26.*

[34] Meihui Gao et al. “Optimal orchestration of virtual network functions”. In: *Computer Networks* 142 (2018), pp. 108–127. ISSN: 1389-1286. DOI: `https://doi.org/10.1016/j.comnet.2018.06.006` *Cited on page 26.*

[35] Ke Yang, Hong Zhang, and Peilin Hong. “Energy-Aware Service Function Placement for Service Function Chaining in Data Centers”. In: *2016 IEEE Global Communications Conference (GLOBECOM)* (2016), pp. 1–6 *Cited on page 26.*

[36] Rami Cohen et al. “Near optimal placement of virtual network functions”. In: *2015 IEEE Conference on Computer Communications (INFOCOM)* (2015), pp. 1346–1354 *Cited on page 26.*

[37] Ali Mohammadkhan et al. “Virtual function placement and traffic steering in flexible and dynamic software defined networks”. In: *The 21st IEEE International Workshop on Local and Metropolitan Area Networks* (2015), pp. 1–6 *Cited on page 26.*

[38] Sevil Mehraghdam, Matthias Keller, and H. Karl. “Specifying and placing chains of virtual network functions”. In: *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)* (2014), pp. 7–13 *Cited on page 26.*

[39] Marcelo Caggiani Luizelli et al. “Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions”. In: *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)* (2015), pp. 98–106 *Cited on page 26.*

[40] Abdelhamid Alleg et al. “Virtual Network Functions Placement and Chaining for real-time applications”. In: *2017 IEEE 22nd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)* (2017), pp. 1–6 *Cited on page 27.*

[41] Bernardetta Addis et al. “Virtual Network Functions Placement and Routing Optimization”. In: Oct. 2015, pp. 171–177. DOI: 10.1109/CloudNet.2015.7335301 *Cited on page 27.*

[42] Nahida Kiran et al. “VNF Placement and Resource Allocation in SDN/NFV-Enabled MEC Networks”. In: *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. 2020, pp. 1–6 *Cited on page 27.*

[43] Samane Asgari et al. “Performance-aware placement and chaining scheme for virtualized network functions: a particle swarm optimization approach”. In: *ArXiv* abs/2105.05248 (2021) *Cited on page 27.*

[44] Yixiang Wang et al. “Cost-Efficient Virtual Network Function Graph (vNFG) Provisioning in Multidomain Elastic Optical Networks”. In: *Journal of Lightwave Technology* 35.13 (2017), pp. 2712–2723. DOI: 10.1109/JLT.2017.2700229 *Cited on pages 27, 28.*

[45] Stefano Coniglio, Arie MCA Koster, and Martin Tieves. “Virtual network embedding under uncertainty: Exact and heuristic approaches”. In: *2015 11th International Conference on the Design of Reliable Communication Networks (DRCN)*. IEEE. 2015, pp. 1–8 *Cited on page 28.*

[46] Antonio Marotta and Andreas Kassler. “A Power Efficient and Robust Virtual Network Functions Placement Problem”. In: *2016 28th International Teletraffic Congress (ITC 28)* 01 (2016), pp. 331–339 *Cited on pages 28, 29.*

[47] Sevil Dräxler and Holger Karl. “Specification, composition, and placement of network services with flexible structures”. In: *International Journal of Network Management* 27.2 (2017). e1963 nem.1963, e1963 *Cited on page 29.*

[48] Chuan Pham et al. “Traffic-Aware and Energy-Efficient vNF Placement for Service Chaining: Joint Sampling and Matching Approach”. In: *IEEE Transactions on Services Computing* 13.1 (2020), pp. 172–185. DOI: 10.1109/TSC.2017.2671867 *Cited on page 29.*

[49] Juan Fang and Aonan Ma. “IoT Application Modules Placement and Dynamic Task Processing in Edge-Cloud Computing”. In: *IEEE Internet of Things Journal* 8 (2021), pp. 12771–12781 *Cited on page 29.*

[50] Windhya Rankothge et al. “Optimizing Resource Allocation for Virtualized Network Functions in a Cloud Center Using Genetic Algorithms”. In: *IEEE Transactions on Network and Service Management* 14.2 (2017), pp. 343–356. DOI: 10.1109/TNSM.2017.2686979 *Cited on page 30.*

[51] Angelos Pentelas and Panagiotis Papadimitriou. “Service Function Chain Graph Transformation for Enhanced Resource Efficiency in NFV”. In: May 2021. DOI: 10.23919/IFIPNetworking52078.2021.9472854 *Cited on page 30.*

[52] Sara Ayoubi, Samir Sebbah, and Chadi M. Assi. “A Cut-and-Solve Based Approach for the VNF Assignment Problem”. In: 2017 *Cited on page 30.*

[53] Sahel Sahhaf et al. “Network service chaining with optimized network function embedding supporting service decompositions”. In: *Computer Networks* 93 (2015). Cloud Networking and Communications II, pp. 492–505. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2015.09.035 *Cited on page 31.*

[54] Meitian Huang et al. “Maximizing Throughput of Delay-Sensitive NFV-Enabled Request Admissions via Virtualized Network Function Placement”. In: *IEEE Transactions on Cloud Computing* 9.4 (2021), pp. 1535–1548. DOI: 10.1109/TCC.2019.2915835 *Cited on pages 31, 32.*

[55] Stuart Clayman et al. “The dynamic placement of virtual network functions”. In: *2014 IEEE Network Operations and Management Symposium (NOMS)* (2014), pp. 1–9 *Cited on page 32.*

[56] Xingsi Xue et al. “Deep Reinforcement Learning-Based Algorithm for VNF-SC Deployment”. In: *Security and Communication Networks* 2021 (2021), p. 7398206. ISSN: 1939-0114. DOI: 10.1155/2021/7398206 *Cited on page 32.*

[57] Bram Naudts et al. “A dynamic pricing algorithm for a network of virtual resources”. In: *2016 IEEE NetSoft Conference and Workshops (NetSoft)*. 2016, pp. 328–335. DOI: 10.1109/NETSOFT.2016.7502429 *Cited on page 32.*

[58] Yang Li, Linh Thi Xuan Phan, and Boon Thau Loo. “Network functions virtualization with soft real-time guarantees”. In: *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. 2016, pp. 1–9. DOI: 10.1109/INFOCOM.2016.7524563 *Cited on page 33.*

[59] Konstantinos V. Katsaros, George Xylomenos, and George C. Polyzos. “Virtual network functions: On the elasticity of software defined network services”. In: *IEEE INFOCOM 2015 - IEEE Conference on Computer Communications*. 2015, pp. 848–856 *Cited on page 39.*

[60] Zaid Allybokus et al. “Virtual function placement for service chaining with partial orders and anti-affinity rules”. In: *Networks* 71.2 (2018), pp. 97–106 *Cited on page 43.*

[61] Steffen Rebennack and Vitaliy Krasko. “Piecewise Linear Function Fitting via Mixed-Integer Linear Programming”. In: *INFORMS Journal on Computing* 32.2 (2020), pp. 507–530 *Cited on page 43.*

[62] Gianni Codato and Matteo Fischetti. “Combinatorial Benders Cuts for Mixed-Integer Linear Programming”. In: *Operations Research* 54 (2006), pp. 756–766 *Cited on page 43.*

[63] J. F. Benders. “Partitioning procedures for solving mixed-variables programming problems”. In: *Numerische Mathematik* 4.1 (1962), pp. 238–252. ISSN: 0945-3245. DOI: 10.1007/BF01386316 *Cited on page 43.*

[64] *Introduction to linear optimization*. 1997 *Cited on page 43.*

[65] J. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer-Verlag, 1997 *Cited on page 44.*

[66] Ragheb Rahmaniani et al. “The Benders Decomposition Algorithm: A Literature Review”. In: *European Journal of Operational Research* 259 (June 2017), pp. 801–817. DOI: 10.1016/j.ejor.2016.12.005 *Cited on page 44.*

[67] Nilotpal Chakravarti. “Some results concerning post-infeasibility analysis”. In: *European Journal of Operational Research* 73.1 (1994), pp. 139–143 *Cited on page 45.*

[68] Marc Pfetsch. “The maximum feasible subsystem problem and vertex-facet incidences of polyhedra”. PhD thesis. Technische Universität Berlin, 2003 *Cited on page 45.*

[69] John Alasdair Warwicker and Steffen Rebennack. "Generating Optimal Robust Continuous Piecewise Linear Regression with Outliers Through Combinatorial Benders Decomposition". In: *IISE Transactions* 0.ja (2022), pp. 1–27. DOI: 10.1080/24725854.2022.2107249. eprint: https://doi.org/10.1080/24725854.2022.2107249 *Cited on page 45.*

[70] Silvano Martello and Paolo Toth. "Knapsack Problems: Algorithms and Computer Implementations". In: *Wiley-Interscience Series in Discrete Mathematics and Optimization* (1990) *Cited on page 69.*

[71] Fred W. Glover and Gary A. Kochenberger. "Critical Event Tabu Search for Multidimensional Knapsack Problems". In: 1996 *Cited on page 69.*

[72] Saïd Hanafi and Arnaud Freville. "An efficient tabu search approach for the 0–1 multidimensional knapsack problem". In: *European Journal of Operational Research* 106.2 (1998), pp. 659–675. ISSN: 0377-2217. DOI: https://doi.org/10.1016/S0377-2217(97)00296-8 *Cited on page 69.*

[73] Stefka Fidanova. "Hybrid Ant Colony Optimization Algorithm for Multiple Knapsack Problem". In: *2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*. 2020, pp. 1–5. DOI: 10.1109/ICRAIE51050.2020.9358351 *Cited on page 69.*

[74] Chunyan Li et al. "A novel parallel hybrid genetic algorithm with decomposition for large-scale optimization". In: *Information Sciences* 563 (2021), pp. 33–55 *Cited on page 70.*

[75] David E. Goldberg. "Genetic algorithms in search, optimization, and machine learning". In: *Addison-Wesley Professional* (1989) *Cited on page 70.*

[76] Kalyanmoy Deb et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *International conference on parallel problem solving from nature*. Springer. 2002, pp. 849–858 *Cited on page 71.*

[77] Mingjie Liu, Hui Wang, and Yansheng Zhang. "Multi-Constrained Routing Algorithm Based on Artificial Bee Colony Optimization in Wireless Sensor Networks". In: *2020 International Conference on Computer, Information and Telecommunication Systems (CITS)*. IEEE. 2020, pp. 1–5. DOI: 10.1109/CITS49685.2020.9209533 *Cited on page 83.*

[78] John Smith and Lisa Johnson. "Multi-Constrained Routing Algorithm for Network Optimization". In: *Proceedings of the International Conference on Networking*. ACM. 2020, pp. 123–134 *Cited on page 84.*

[79] Anuj Puri and Stavros Tripakis. "Algorithms for the Multi-constrained Routing Problem". In: *Algorithm Theory — SWAT 2002*. Ed. by Martti Penttonen and Erik Meineche Schmidt. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 338–347 *Cited on page 84.*

[80] Husam Shakhatreh, Mohsen Guizani, and Abdallah Shami. "A Survey of Multi-Constrained Routing Protocols in Wireless Sensor Networks". In: *Sensors* 19.11 (2019), p. 2540. DOI: 10.3390/s19112540 *Cited on page 84.*

[81] Piet Van Mieghem, Hans De Neve, and Fernando Kuipers. "Hop-by-hop quality of service routing". In: *Comput. Netw.* 37.3-4 (2001), pp. 407–423. ISSN: 1389-1286 *Cited on pages 88, 90.*

[82] Hans De Neve and Piet Van Mieghem. "TAMCRA: a Tunable Accuracy Multiple Constraints Routing algorithm". In: *Comput. Commun.* 23.7 (2000), pp. 667–679 *Cited on pages 88, 90.*

[83] Wissal Attaoui et al. *VNF and Container Placement: Recent Advances and Future Trends*. 2022. arXiv: 2204.00178 [cs.NI] *Cited on page 109.*

[84] Yijun Xiong and L.G. Mason. "Restoration strategies and spare capacity requirements in self-healing ATM networks". In: *Networking, IEEE/ACM Transactions on* 7 (Mar. 1999), pp. 98–110. DOI: 10.1109/90.759330 *Cited on pages 114, 115, 121.*

[85] Peter Batchelor et al. "Study on the Implementation of Optical Transparent Transport Networks in the European Environment—Results of the Research Project COST 239". In: *Photonic Network Communication* 2 (Mar. 2000), pp. 15–32. DOI: 10.1023/A:1010050906938 *Cited on pages 114, 119.*

[86] Ramakrishnan Durairajan et al. "InterTubes: A Study of the US Long-Haul Fiber-Optic Infrastructure". In: *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. SIGCOMM '15. London, United Kingdom: Association for Computing Machinery, 2015, pp. 565–578. ISBN: 9781450335423. DOI: 10.1145/2785956.2787499 *Cited on pages 114, 119.*