UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Models and Methods for Network Function Virtualization (NFV) Architectures

THÈSE

Soutenance le 19 Mars 2019

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Meihui GAO

Composition du jury

M. Bernard Fortz	**Rapporteur**	**Professeur, Université Libre de Bruxelles**
M. Luigi De Giovanni	**Rapporteur**	**Professeur, Università degli studi di Padova**
M. Stefano Secci	**Examinateur**	**Professeur, Conservatoire National des Arts et Métiers**
M. Eric Gourdin	**Examinateur**	**Doctorat, Orange Gardens**
M. Ye Qiong Song	**Directeur de Thèse**	**Professeur, Université de Lorraine**
Mme Bernardetta Addis	**Co-directeur de Thèse**	**MCF, Université de Lorraine**

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Acknowledgement

First of all, I would like to thank the universe for giving me such an amazing experience in life.

I would like to thank my parents and my brother for their unconditional love. I will always be thankful that you have always been supporting me.

I would also like to thank my boyfriend Shuguo for his companionship and encouragement during all these years of study.

I would like to thank my advisors, Prof. Bernardetta Addis and Prof. Ye-qiong Song, who have always been incredibly available when needed. Thanks so much for their disponibility, their patience and their deep knowledge in the domain of network optimization that were very precious all along the working of this thesis. They helped me to grow professionally and I learnt many skills from them that will be of great assistance in my future career. The discussions and suggestions inspired me to do my best to accomplish my goals.

I would like to express my special thanks to Prof. Giuliana Carello who has guided me during my PhD study. Without her help and advice, I would not be able to successfully finish my thesis. Also, I am really thankful to Prof. Stefano Secci for his technical advises and help. Without his help, I probably would have missed the opportunity of doing this PhD.

Finally, I would like to thank the friends and colleagues in LORIA and INRIA for creating such a pleasant and enjoyable working environment. I am grateful to all of you for your support and encouragements as well as the fun times we have shared together. Last, special thanks to the MADYNES team for the equipment assistance.

To my family

Contents

Glossary **1**

1 General Introduction **5**

1.1 General Technical Background 6

1.2 Problem Statement 9

1.3 Thesis Goals and Contributions 10

1.4 Thesis Organization 11

2 State of the Art **13**

2.1 Basic Features of NFV Service Chaining 14

2.2 Similar Problems 14

2.2.1 Virtual Machine Placement (VMP) 15

2.2.2 Virtual Network Embedding (VNE) 16

2.3 Possible Variants Tackled in Literature and A Classification Guideline 17

2.3.1 Deployment Environment 18

2.3.2 System Performance Metrics 20

2.3.3 Main Optimization Objectives and Constraints 20

2.3.4 Existing Modeling Strategies 24

2.3.5 A Classification of the Approaches 26

2.4 Overview of Related Work 28

3 The Application Problem **33**

3.1 Problem Features Overview 33

3.2 Problem Statement and Mathematical Formulation 34

3.2.1 Basic Placement and Routing (PR) Model 36

3.2.2 Virtual Network Function (VNF) Forwarding Latency 40

3.2.3 Bit-rate Compression/Decompression 41

3.2.4 Multi-objective Math-heuristic Resolution 45
3.3 Computational Results 46
3.3.1 Results of PR Algorithm under Different Cases of Demand Distribution and Different VNF Forwarding Latency Profiles 46
3.3.2 PR vs. VNE Based Approaches 62
3.4 Further Model Refinements 64
3.5 Conclusion 67

4 The Theoretical Problem 69
4.1 Problem Description 70
4.2 Problem Complexity 70
4.3 Problem Properties 77
4.4 Problem Formulations 79
4.4.1 Relation between the Two Formulations 82
4.4.2 Additional Inequalities 93
4.4.3 Extended Split Path (SP) Model for Multiple Services 94
4.5 Computational Results 97
4.5.1 Test Instances 97
4.5.2 Continuous Relaxation Results on Small-medium Size Instances 98
4.5.3 Full model (ILP/MILP) Results on Small-medium Size Instances 102
4.5.4 Scalability on Large Size Instances 105
4.5.5 Impact of Proposition 4.3.1 106
4.5.6 Articulation Point Based Preprocessing 108
4.6 Conclusion 108

5 ILP-based Constructive Approaches 111
5.1 ILP-based Constructive Methods 111
5.1.1 SP Formulation with Bin-Packing-like Improved Bound . . . 111
5.1.2 Three-Step Bin Packing Based Fix-and-Solve Method 113
5.1.3 Fix-and-Check Method 116
5.2 Computational Results 120
5.2.1 Results on Small-medium Size Instances 121
5.3 Conclusion 123

6 ILP-based Heuristic Approaches **125**

6.1 Overview of *k-opt* Technique and *k-opt* Models . . . 126

6.1.1 *k-opt neighborhood search* Technique . . . 126

6.1.2 *k-opt* Models . . . 127

6.2 Computing An Initial Solution . . . 128

6.3 Improving Local Search (LS) Based Heuristics . . . 131

6.4 Fix-and-Optimize Heuristics . . . 131

6.5 Computational Results . . . 133

6.5.1 Results of Approaches for Obtaining An Initial Solution . . . 133

6.5.2 Results of Heuristic Methods . . . 136

6.6 Conclusion . . . 142

7 General Conclusion **143**

7.1 Summary of Contributions . . . 143

7.2 Future Research Directions . . . 144

Résumé de la Thèse **145**

List of Publications **153**

List of Figures **155**

List of Tables **157**

Bibliography **161**

Appendices

A

Father Node Formulation

B

Column Generation Approach

Glossary

BPP : Bin Packing Problem
CapEx : Capital Expenditures
CPE : Customer Premises Equipment
DPDK : Data Plane Development Kit
DPI : Deep Packet Inspection
EPC : Evolved Packet Core
ETSI : European Telecommunications Standards Institute
HTTP : Hypertext Transfer Protocol
IETF : Internet Engineering Task Force
ILP : Integer Linear Programming
IP : Internet Protocol
ISP : Internet Service Provider
LS : Local Search
MANO : NFV Management and Orchestration
MIP : Mixed Integer Programming
MILP : Mixed Integer Linear Programming
N-PoP : Network Point of Presence
NFV : Network Function Virtualization
NFVI : NFV Infrastructure
NFVI-PoP : NFV Infrastructure Point of Presence
NS : Neighborhood Search
OpEx : Operational Expenditures
PR : Placement and Routing formulation/modeling
QoS : Quality of Service
RAN : Radio Access Network
SDOs : Standards Developing Organizations
SDN : Software Defined Networking
SFC : Service Function Chaining
SLA : Service Level Agreement
SP : Split Path formulation/modeling
TE : Traffic Engineering
VMP : Virtual Machine Placement Problem
VNE : Virtual Network Embedding
VNF : Virtual Network Function
VNF-PR : Virtual Network Function Placement and Routing problem

VNF-PR$_{\mathbf{SP}}$: Virtual Network Function Placement and Routing problem with Simple Path routing

VPN : Virtual Private Network

Abstract

Models and Methods for Network Function Virtualization (NFV) Architectures

Due to the exponential growth of service demands, telecommunication networks are populated with a large and increasing variety of proprietary hardware appliances, and this leads to an increase in the cost and the complexity of the network management. To overcome this issue, the NFV paradigm is proposed, which allows dynamically allocating the Virtual Network Functions (VNFs) and therefore obtaining flexible network services provision, thus reducing the capital and operating costs.

In this thesis, we focus on the VNF Placement and Routing (VNF-PR) problem, which aims to find the location of the VNFs to allocate optimally resources to serve the demands. From an optimization point of view, the problem can be modeled as the combination of a facility location problem (for the VNF location and server dimensioning) and a network design problem (for the demands routing). Both problems are widely studied in the literature, but their combination represents, to the best of our knowledge, a new challenge. We start working on a realistic VNF-PR problem to understand the impact of different policies on the overall network management cost and performance. To this end, we extend the work in [1] by considering more realistic features and constraints of NFV infrastructures and we propose a linear programming model and a math-heuristic to solve it. In order to better understand the problem structure and its properties, in the second part of our work, we focus on the theoretical study of the problem by extracting a simplified, yet significant variant. We provide results on the computational complexity under different graph topology and capacity cases. Then, we propose two mathematical programming formulations and we test them on a common testbed with more than 100 different test instances under different capacity settings. Finally, we address the scalability issue by proposing ILP-based constructive methods and heuristics to efficiently deal with large size instances (with up to 60 nodes and 1800 demands). We show that our proposed heuristics can efficiently solve medium size instances (with up to 30 nodes and 1000 demands) of challenging capacity cases and provide feasible solutions for large size instances of the most difficult capacity cases, for which the models cannot find any solution even with a significant computational time.

Keywords: Network Function Virtualization (NFV), Resource Allocation, Operations Research

Résumé

Modèles et méthodes d'optimisation pour architecture NFV (Network Function Virtualization)

Avec la croissance exponentielle des demandes de service, les opérateurs ont déployé de nombreux équipements, et par conséquent, la gestion du réseau est devenue de plus en plus difficile et coûteuse. La virtualisation des fonctions réseau (NFV) a été proposée comme un nouveau paradigme pour réduire les coûts liés à l'acquisition et à la maintenance pour les réseaux de télécommunications.

Dans ce travail de thèse, nous nous intéressons aux problèmes du chaînage des fonctions virtuelles (VNFs) qui combinent des décisions de localisation des VNFs et de routage des demandes. D'un point de vue d'optimisation, ce problème est une combinaison des problèmes de localisation (pour la partie d'installation des VNFs) et de conception de réseaux (pour la partie de routage). Ces deux problèmes ont été largement étudié dans la littérature. Cependant, leur combinaison représente des divers challenges en termes de modélisation et de résolution. Dans la première partie de cette thèse, nous considérons une version réaliste du problème du chaînage des VNFs (VNF-PR) afin de comprendre l'impact des différents aspects sur les coûts et les performances de gestion du réseau. Dans ce but, nous étendons le travail dans [1] en considérant des caractéristiques et des contraintes plus réalistes des infrastructures NFV et nous proposons un modèle de programmation linéaire et une heuristique mathématique pour le résoudre. Dans le but de mieux comprendre la structure du problème et ses propriétés, la deuxième partie de la thèse est orientée vers l'étude théorique du problème, où nous avons étudié une version compacte du problème du chaînage des VNFs. Nous fournissons des résultats sur la complexité de calcul sous divers cas de topologie et de capacité. Ensuite, nous proposons deux modèles et nous les testons sur un testbed avec plus de 100 instances différentes avec différents cas de capacité. Au final, nous abordons la scalabilité du problème en proposant des méthodes constructives et des méthodes heuristiques basées sur la programmation linéaire entière pour traiter efficacement des instances de taille grande (jusqu'à 60 nœuds et 1800 demandes). Nous montrons que les heuristiques proposées sont capables de résoudre efficacement des instances de taille moyenne (avec jusqu'à 30 noeuds et 1 000 demandes) de cas de capacité difficiles et de trouver de bonnes solutions pour les instances dures, où le modèle ne peut fournir aucune solution avec un temps de calcul limité.

Mots-clés: Virtualisation des Fonctions Réseaux (NFV), Allocation des Ressources, Recherche Opérationelle

Chapter 1

General Introduction

Contents

1.1 General Technical Background 6
1.2 Problem Statement 9
1.3 Thesis Goals and Contributions 10
1.4 Thesis Organization 11

Current telecommunication network services rely on proprietary appliances and different network devices that are diverse and dedicated-built. This situation, coupled with the increasing diffusion (innovation) of network applications, induces the so called network ossification problem. As a result, the operation of service additions and network upgrades becomes extremely expensive. To give an example, in the current networks, a service includes a set of hardware dedicated network appliances which offer block functions such as firewall, load balancing, Deep Packet Inspection (DPI), Intrusion Detection System, etc., to support the required networking processing and applications. When a new networking service requirement emerges, new hardware devices must be installed and chained, which is extremely time consuming and costly in terms of capital expenditures. This kind of networking service deployment requires dedicated plan of networking updates, which additionally incurs in high operating expenditures.

Virtualization technologies are expected to address these challenges by proposing solutions to flexibly and efficiently design, deploy, and manage network services [2]. After about ten years of fundamental research, the virtualization of network functions is becoming a reality thanks to huge investments being made by telecommunication providers, cloud providers and vendors. The breaking point sits in 2012, when calls for experimentation and deployment of what was coined as "Network Functions Virtualization (NFV)" [3] lead to the creation of an NFV industry research group at the European Telecommunications Standards Institute (ETSI) [4]. Since then, applied researches and developments have accelerated investments, hence preliminary prototypes were demonstrated and deployed (leading to commercialization in some cases) since late 2014 [5].

With NFV, the attention of network virtualization research is now focusing on key aspects of NFV systems that were either not considered relevant or not conceived before industry effort at Standards Developing Organizations (SDOs). A central role is played by the NFV service chaining [6] provisioning, i.e., the problem of allowing a traffic flow passing through a pre-computed or dynamically computed list of Virtual Network Function (VNF) nodes, possibly accounting for the fact that VNF nodes can be placed at, and migrated across, virtualization clusters as a function of demand assignment to existing VNF chains or sub-chains.

In this context, this thesis addresses the NFV service chaining as an optimization problem. In the reminder, we first introduce the technical background, then we provide an overview on the optimization problems tackled and we summarize the study goals and thesis contributions.

1.1 General Technical Background

Network Function Virtualization

The main idea of NFV consists in decoupling network functions from the physical network equipment that they have been run on. Consequently, NFV allows software based network functions to run over high volume industry standard physical servers, switches and storage, which could be located in data centers, Points of Presence (PoPs), network nodes and in the end user premises. Therefore, NFV allows avoiding the purchase and installation of specialized hardware when new network service requests come.

ETSI is de-facto the reference SDO for the NFV high-level functional architecture specification. High-level means that its identified role is the specification of the main functional blocks, their architecture and inter-relationship, whose implementation elements could then be precisely addressed by other SDOs. ETSI specifies three components [7] for the NFV architecture: Virtual Network Functions (VNFs); NFV Infrastructure (NFVI), including the elements needed to run VNFs such as the hypervisor node and the virtualization clusters; MANO (Management and Orchestration), handling the operations needed to run, migrate, optimize VNF nodes and chains, possibly in coordination with transport network orchestrators. Figure 1.1 illustrates the components and their relationships.

The NFVI is the combination of both software (e.g. virtualization hypervisor) and hardware (e.g., servers, storage and network) resources that build an NFV environment for the deployment of VNFs. Such infrastructure can be seen as a distributed set of VNF nodes, which is also known as NFVI-PoPs. Each NFVI-PoP is an abstracted physical location (e.g., data center, network node, or end user premise) of the network infrastructure which has a limited computational capacity to host the VNFs.

VNFs are software implementation of network functions (e.g., firewall and DPI) running over the NFVI. A single VNF may be composed of multiple internal components that run individually over different virtual resources such as VMs (Virtual Machines). In telecommunication networks, a network service is composed of one or

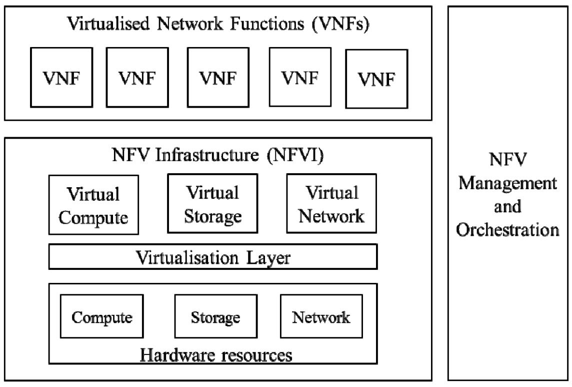


Figure 1.1 – General NFV architecture proposed by the ETSI

more network functions. In the case of NFV, the network functions that make up the network service are implemented by VNFs on virtual resources. Therefore, the behavior of the network service in NFV environment is dependent on that of the constituent VNFs.

Finally, the NFV MANO provides the functionality required for the provisioning of VNFs and all the related operations, such as the placement and instantiation of VNFs to better meet user's demands, the configuration of the VNFs to share them among active demands while meeting common Traffic Engineering (TE) objectives in IP (Internet Protocol) transport networks as well as novel NFV efficiency goals such as the minimization of the number of VNF instances to install.

Service Function Chaining

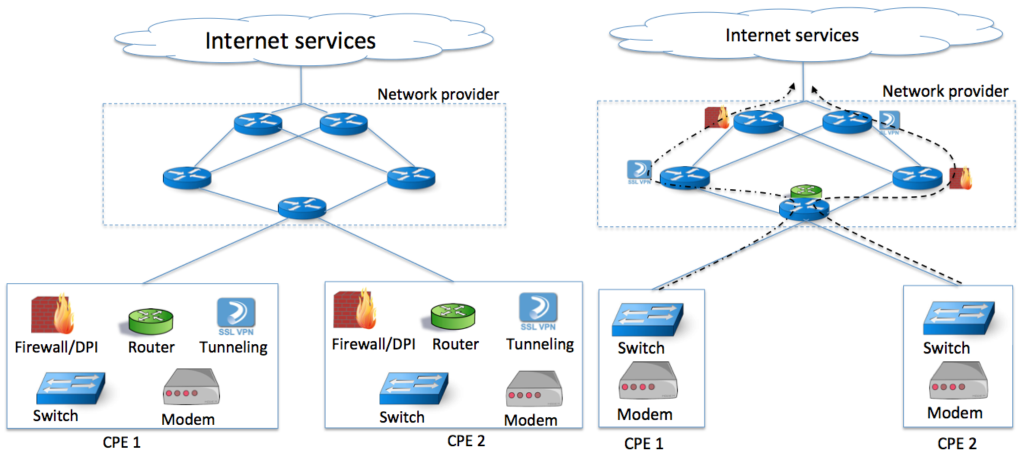
In most networks, the provisioning of end-to-end network services often requires various service functions including traditional network service functions (e.g., firewalls), as well as application-specific features such as HTTP (Hypertext Transfer Protocol) header manipulation. The delivery of network services is realized when the demands pass through several service functions. Typically, a network service is constructed as an ordered or partially ordered sequence of service functions [8], and this construction is known as Service Function Chaining (SFC) [9].

On the current network service deployments, service function chains tightly depend on the physical underlying infrastructure. Such dependency imposes many constraints on network function delivery, such as:

- configuration complexity. Due to high dependency on physical network infrastructure, modifications on deployed service function chains need reconfiguration on those chains, which leads to additional operational expenditures and slows down the delivery of new services;
- limited ability to utilize infrastructure resources. Due to the increasing changes of traffic pattern, the hardware based SFC deployment may become less efficient to manage network resources by using the dedicated networking plans.

NFV service chaining problem

The deployment of network services in NFV environments is known as the NFV service chaining [6] provisioning. Thanks to its virtualization nature, NFV service chaining present great potential for overcoming the ossification problem of current network by virtualizing network functions to provide flexibility and agility to the management (e.g., create, remove, scale) of VNFs and the chaining of these VNFs via NFV MANO. For example, the virtual Customer Premises Equipment (vCPE) [10] can simplify the network service delivery by means of virtualized individual network functions placed at network provider locations, which allows replacing large number of dedicated hardware devices deployed at each customer side (e.g., CPE1 and CPE2 in Figure 1.2a) by software-based functions running on common servers at aggregation locations (Figure 1.2b). With traditional CPE, deployment of new services can be time-consuming and expensive. While by virtualizing CPE, network providers can reduce Capital Expenditure (CapEx) and Operational Expenditure (OpEx) and speed service delivery by configuring and managing the shared software-based functions and providing new services on demand. There are also other promising use-cases like the virtualization of the Evolved Packet Core (EPC) cluster in cellular core networks [11, 12], and the virtualization of cellular base stations [13].



(a) Traditional CPE (b) vCPE

Figure 1.2 – Traditional Customer Premises Equipment (CPE) compared to virtualized CPE (vCPE) with VNF chaining.

A key feature for the effective success of NFV service chaining is the cost-efficient placement of VNFs and routing of service demands to pass through the required VNF nodes. Therefore, the optimization of NFV service chaining has received much attention from both industry and academia. There is a significant number of works on the problem with respect to, for instance, resource efficiency [14], or competitive goals [1]. Besides the common NFVI-PoP context, the NFV service chaining problem has also been addressed in specific contexts, such as wireless networks [15] [16], mobile core networks [17] and optical networks [18]. For tackling these problems, most of the approaches rely on heuristic algorithms. Besides, game theoretic approaches are also considered in the literature: in [19] authors propose a heuristic method based on routing games; in [20] authors propose a distributed dynamic pricing approach to allocate demands to already placed VNF instances, with convex congestion functions for both links and VNFs to control congestion; authors in [21] use the non-cooperative game theory to propose a distributed and privacy-preserving algorithm to solve the service chain composition problem. Despite recent research activity in the field, the problem of NFV service chaining is relatively new and various research questions remain open. A recent study [22] evaluates some of them highlighting that they may come at a “Revenue/Cost” ratio of 50%, i.e., twice as many resources were consumed than demands realized by heuristic approaches, which suggests that there is significant optimization potential to achieve in the area.

1.2 Problem Statement

The NFV service chaining optimization problem can be schematically described as follows: an NFVI network is given and represented by a graph $G(N, A)$, where N is the set of NFVI nodes and A the set of links between nodes. Each node $i \in N$ can host a limited number of VNF instances, and each link $(i, j) \in A$ can allow a limited quantity of flow to pass by. We consider a set of traffic demands D, each demand $k \in D$ is characterized by a source $o_k \in N$, a destination $t_k \in N$, a nominal bandwidth d_k, and a set of VNF types that provide the network services required by the demand. The problem is to decide on which nodes to install the VNF instances and to route the demands from the source node to the destination node traversing the nodes that host their required VNF instances, so that the utilization of the overall network resources is optimized. Many constraints can be taken into account, such as the VNF order constraint (e.g, the VNFs to be traversed firstly, secondly, ..., lastly by the demand may be specified), the VNF sharing constraint (e.g., some VNF instances can be shared by multiple demands while some others cannot), the node/VNF capacity constraint, the link capacity constraint, the VNF forwarding latency (i.e., the processing time that a VNF instance uses) constraints, thus leading to different versions of the problem.

Given the context above, the work presented in this thesis is articulated as follows. In the first part, we consider a very detailed problem. The core part, the **Virtual Network Function Placement and Routing (VNF-PR)** problem, is to find the optimal placement of VNF instances over NFVI nodes and the optimal routing for demands and their assignment to VNF nodes, so that both network-level (i.e., traffic flow) and NFVI-level (i.e., network infrastructure) resources utilization are minimized. Specific features and constraints of NFV infrastructures are considered, such as link capacity constraints, NFVI node capacity constraints, VNF flow compression/decompression constraints (i.e., VNF instances may change the bit-rate), VNF forwarding latency constraints, and VNF chain (total or partial) order constraints for each demand (e.g., to deploy the network services, the demands may have to pass by some VNF instances before passing by other VNF instance, thus leading to different requirements of VNF sequences, which is known as the VNF chain order constraints). For such problem, an MILP (Mixed Integer Linear Programming) and a math-heuristic are proposed and various scenarios analysed.

In the second part of the work, we focus on the problem complexity, properties and formulations' comparison. In order to study the problem structure, we reduce the VNF-PR problem to a simplified, yet significant variant: a single type of VNF is considered and any node can be equipped with a single VNF instance (i.e., node capacity and VNF capacity are confounded); each demand requires a single service and must be routed on a simple path (i.e., demands are allowed to pass by a node at most once); VNF instances and links are capacitated and the objective is to minimize the number of installed VNF instances. This version of problem is denoted as the **Virtual Network Function Placement and Routing with Simple Path (VNF-PR$_{\mathbf{SP}}$)** problem.

1.3 Thesis Goals and Contributions

NFV service chaining related problems are very actual and relevant, many papers have been published on such topics especially in the telecommunication literature. However, there is still significant optimization potential to achieve. Despite the large number of papers, it is difficult to find a comparison among the proposed approaches as many different versions of the problem have been considered and the proposed approaches are tailored on them.

In this context, the goal of this thesis is to study the problem structure, investigate problem complexity and properties, analyze and compare the most promising formulation strategies on a common test bed, and finally to devise efficient methods able to scale with large size problems. To summarize, the study of this thesis involves mainly four goals:

1. formalize the VNF-PR and VNF-PR$_{\text{SP}}$ optimization problems;
2. investigate the problem structure, complexity and properties;
3. compare the most promising formulation strategies proposed in the literature on a common test bed and propose proper mathematical formulation for the considered problem;
4. design efficient and scalable methods in order to compute timely solutions of good quality to the VNF-PR$_{\text{SP}}$ problem.

Achievements of the aforementioned goals finally bring about the following main contributions of this thesis:

1. on the Virtual Network Function Placement and Routing (VNF-PR) problem.
    - we formalize a realistic VNF-PR problem and we propose a linear programming formulation that is able to accommodate specific features and constraints of NFV infrastructures, such as bit-rate changes;

- we design a math-heuristic able to scale with multiple objectives step by step and large instances;
- we conduct extensive tests and we draw conclusions on the trade-off achievable between classical TE and NFV infrastructure efficiency goals, evaluating both Internet access and Virutal Private Network (VPN) demands;
- we provide possible refinements of the model to meet specific requirements, such as VNF affinity and anti-affinity rules, VNF isolation, etc.

2. on the Virtual Network Function Placement and Routing with Simple Path (VNF-PR$_{\text{SP}}$ problem) problem.

   - we study the problem complexity and prove the NP-completeness nature of the VNF-PR$_{\text{SP}}$ problem;
   - we investigate the problem properties and prove that: the single VNF type case is equivalent to the multiple VNF types one under certain conditions; an instance of the service must be installed on particular nodes under certain conditions;
   - we develop problem property and complexity based algorithms that are able to help in speeding up the computational time;
   - we generate a large common test bed and we compare two most promising formulations proposed in the literature both theoretically and computationally. We further develop and evaluate several valid inequalities to improve the problem formulations;
   - we extend the formulations to address more general versions of the problem with multiple VNF types and multiple orders;
   - we provide several strategies for obtaining quickly an initial feasible solution for instances such that the model cannot find any feasible solution with large computational time;
   - we address the scalability of the problem by proposing ILP-based heuristics. We demonstrate that our proposed methods can solve efficiently medium size instances of challenging capacity cases and provide feasible solutions for large size instances of the most difficult capacity cases. We also provide insights for further improving the methods.

1.4 Thesis Organization

The remainder of this thesis is organized as follows.

- Chapter 2 presents the main concepts for fully understanding the optimization problems tackled and approaches proposed in this thesis. We review possible variants of the NFV service chaining problem tackled in the literature. A classification guideline for the state of the art is also proposed. Further, the relation of the NFV service chaining problem with the optimization literature is discussed;

- Chapter 3 defines formally a realistic version of the Virtual Network Function Placement and Routing (VNF-PR) problem and proposes a mathematical formulation and a math-heuristic to solve the considered problem. Detailed analysis on realistic scenarios are reported in terms of VNF placement, total delay, etc;
- Chapter 4 defines formally the Virtual Network Function Placement and Routing with Simple Path (VNF-PR$_{\text{SP}}$) problem and provides a thorough complexity and properties analysis. A comparison of two most promising formulations proposed in the literature is studied both theoretically and computationally;
- Chapter 5 and 6 propose ILP-based constructive methods and heuristics for the VNF-PR$_{\text{SP}}$ problem to allow solving large size instances;
- Chapter 7 presents the final considerations and ideas for future work.
- Appendices A and B describe two additional methods that we have implemented during the thesis. As they performed not as well as the methods presented in Chapters 3 to 6, we decided to present them only shortly and in an appendix.

Chapter 2

State of the Art

Contents

2.1 Basic Features of NFV Service Chaining 14
2.2 Similar Problems 14
2.2.1 Virtual Machine Placement (VMP) 15
2.2.2 Virtual Network Embedding (VNE) 16
2.3 Possible Variants Tackled in Literature and A Classification Guideline 17
2.3.1 Deployment Environment 18
2.3.2 System Performance Metrics 20
2.3.3 Main Optimization Objectives and Constraints 20
2.3.4 Existing Modeling Strategies 24
2.3.5 A Classification of the Approaches 26
2.4 Overview of Related Work 28

As mentioned previously, the NFV service chaining problem is one of the most challenging aspects in NFV systems. This problem represents big challenges for many reasons. First, NFV is designed to spread VNFs over the deployment network infrastructure. Therefore, depending on the type of the deployment network, the way the placement and chaining VNFs is done changes. Second, different types of network services ask for different network performance metrics, and the performance (e.g., latency and throughput) is affected by the way the service is deployed. Therefore, new network services may further impose additional constraints on how VNFs are chained and deployed onto physical servers. Third, NFV is expected to reduce CapEx and OpEx, thus, how to utilize efficiently various network resources should also be considered in the deployment of SFC in NFV.

Many different versions of the problem have been studied in the literature considering different optimization goals and constraints. Nevertheless, basic features representing essential components of the problem can be found in all the versions.

In this chapter, we present the basic features of the NFV service chaining problem distinguishing it from other resources allocation problems and we review some most prominent approaches in the literature. In section 2.1 we provide an overview on basic features of the NFV service chaining problem and we discuss the relation with the optimization literature. In section 2.2, we discuss the similarity and the difference between NFV service chaining and another two well-studied resources allocation problems. In section 2.3, we provide a classification guideline for the state of the art of the NFV service chaining problem, considering possible variants tackled in the literature. In section 2.4 we review some of the most prominent approaches regarding service deployment in the context of NFV environment.

2.1 Basic Features of NFV Service Chaining

The basic objective of the NFV service chaining problems is to optimize the design of NFV service chaining system, where final decision includes 1) the location of VNF instances; 2) the assignments of demands to the selected VNF instances; and 3) the end-to-end routing of traffic demands. From an optimization point of view, the NFV service chaining problem shares features with network design problems (for the service demand routing part) and with facility location problems (for the VNF location and the network resources dimensioning):

- demand routing of NFV service chaining and network design.
  Typically, network design involves dimensioning nodes and links and routing demands. In VNF-PR, service demands must be routed on the network from their source node to their destination node, and be served by a set of VNF instances. In this sense, the routing part of VNF-PR is similar to the demand routing part of network design;

- network resource allocation of NFV service chaining and facility location.
  Facility location includes locating facilities (and in case dimensioning them) and serving users (namely connecting users to facility). Similarly in VNF-PR, VNFs correspond to facilities and demands to users. If we consider the VNF-PR problem at a higher level of detail (i.e., with virtual machines), the two-level facility location takes part. In this sense, the network resource allocation aspect of VNF-PR and facility location are similar, since allocation or hierarchical allocation decisions must be made in a client-server manner.

Both network design and facility location problems are widely studied in the literature, but the combination described by the NFV service chaining problem represents a new challenge. Although the combined facility location and network design problem has been studied in optimization literature [23], the NFV service chaining problem specificities are still not explored in the optimization literature.

2.2 Similar Problems

In this section, we present two well-studied resources allocation problems in network virtualization, discussing the similarities and difference with respect to the NFV service chaining problem.

2.2.1 Virtual Machine Placement (VMP)

VMs are the key component of cloud computing and they provide virtual resources such as CPU, memory, storage, and network interfaces in the same way as physical resources do. Different from physical appliances, a VM instance can be dynamically created, scaled up, and migrated to other locations on demand. This improves the availability and scalability for dynamic cloud infrastructures. The process of selecting which VMs should be placed on which physical devices (e.g., servers and data centers) is known as the Virtual Machine Placement (VMP) problem [24].

In general, the problem of VMP can be divided into two tasks (as shown in Figure 2.1): the first is the admission of new requests for the VM provisioning and the placement of the accepted VMs on hosts, and the second is the optimization of the VMP by VM migration (move the VMs to other hosts from their initial VM placement) process. The underlying optimization problem is in the majority of cases a Bin-Packing Problem, which is NP-hard [25].

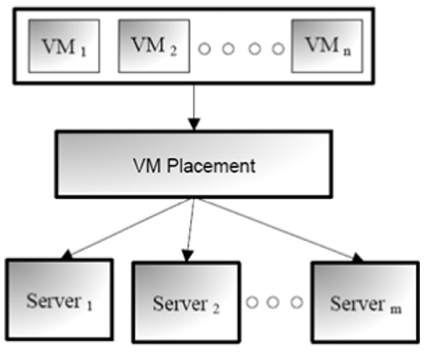


Figure 2.1 – VMP (source from [24])

NFV service chaining and VMP have the similarity that they both try to place virtual entities on potential locations, which can be considered as a facility location problem. However, existing works in the VMP are not suitable for the placement of VNFs for the reasons described by [26], e.g., the problem of VMP is node-centric (VMs being many and small endpoints), while the problem of VNFs placement is network-centric (VNFs being few and large middlepoint). Furthermore, VMP is generally the process of selecting the potential positions for the virtual entities, whereas NFV service chaining has to further take into consideration the traffic path generation passing through virtual entities by order.

More precisely,

- NFV service chaining places the VNFs on potential locations and generates routing path for each demand (i.e., demands must be steered to traverse through ordered VNFs), while VMP places VMs on potential locations;
- the topology abstraction may be different in the two problems, since VMP infrastructure is in general homogeneous as VMs are hosted in data centers,

whereas the infrastructure of NFV is heterogeneous and can involve optical network [27], wireless [15], and multi-providers [28], [29], etc.

2.2.2 Virtual Network Embedding (VNE)

Virtual networks are the primary entity in network virtualization. A virtual network is a combination of virtual nodes and virtual links on top of a substrate/physical network. Virtual nodes are interconnected through virtual links. By virtualizing both node and link resources of a substrate network, multiple virtual network topologies with can be created and co-hosted on the same physical hardware. The problem of embedding virtual networks in a substrate network is one of the main resource allocation challenge in network virtualization and is usually referred to as the Virtual Network Embedding (VNE) problem [30].

In general, the VNE problem can be divided in two sub-problems (as shown in Figure 2.2): *Virtual Node Mapping* where virtual nodes have to be allocated in physical nodes and *Virtual Links Mapping* where virtual links connecting the virtual nodes have to be mapped to paths connecting the corresponding nodes in the substrate network. Solving the VNE problem is NP-hard [30], as it is related to the multi-way separator problem.

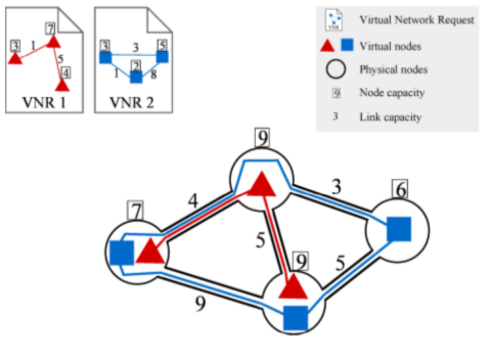


Figure 2.2 – VME (source from [30])

Special version of the NFV service chaining problem that considers fixed order of VNFs, can be seen as a special case of VNE in the sense that the virtual networks to be embedded in VNE reduce to linear graphs (e.g., a straight line) whose both extreme nodes have a fixed location. However, NFV service chaining problem is different from VNE, for instance, in case of partial orders and affinity/anti-affinity rules [31] (which tell the operator to keep virtual entities together or separated), NFV service chaining problem cannot be treated as a special case of VNE as VNE needs to know the fixed order of VNFs for embedding the virtual nodes and virtual links into the substrate network.

More precisely,

- VNE only works with the fixed order of virtual nodes, while NFV service chaining also can deal with partial order and no order cases;
- in NFV service chaining, a VNF can be shared by multiple demands, while in VNE, different virtual networks are typically independent, i.e., a flow of a virtual network does not traverse through the virtual nodes of another virtual network.

2.3 Possible Variants Tackled in Literature and A Classification Guideline

In this section, we propose a classification of the works on the NFV service chaining problem based on their deployment environments, system performance metrics, and modeling strategies.

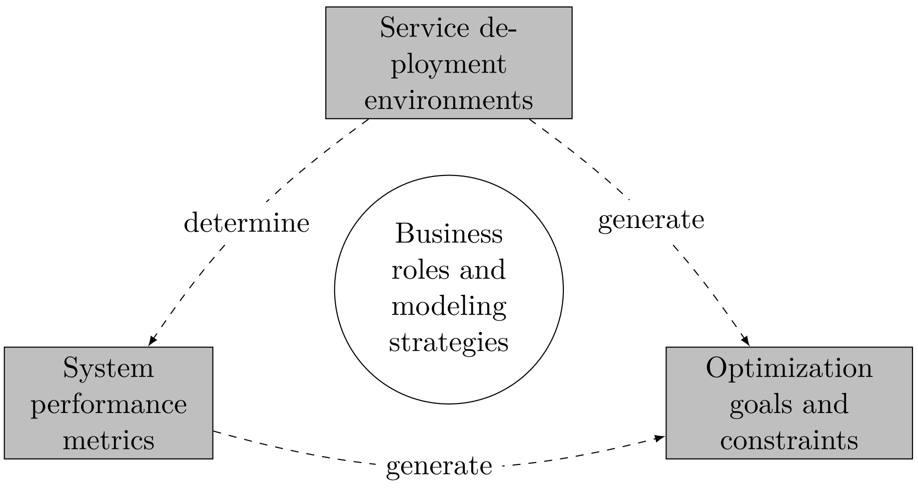

Service deployment environments
determine
generate
Business roles and modeling strategies
System performance metrics
Optimization goals and constraints
generate

Figure 2.3 – Variants of NFV service chaining problems

Although NFV is a novel concept, the problem of NFV service chaining have been tackled by many works in the literature considering various optimization objectives and constraints due to various use cases of the NFV technology. In order to review different approaches, we propose a classification mainly focusing on their underlying optimization problems, and on the proposed solution methods. For the sake of clarity, we illustrate in Figure 2.3 the principal connections among optimization, service deployment contexts, and specific system performance requirements. The optimization goals and constraints are mainly generated by the service deployment environments and system performance metrics, that depend on the NFV business model.

In fact, the deployment of the a service chain depends on the network functions that are included in chain. And the deployment of the network functions depends on the type of network. For example, in wireless networks it is necessary to deal with radio resources [16], thus in this context, the dimensioning of radio resources is likely to be considered in the NFV service chaining problem. Therefore, the deployment environments may generate optimization goals and constraints. Furthermore, the network environment may determine certain system performance metrics, and the system performance metrics may further introduce additional optimization goals and constraints. For instance, there are strict delay budgets among communicating data and control plane elements (e.g., Service Gateway and Packet Data Network Gateway) in mobile core network [17], which may introduce latency constraints to the VNF-PR problem.

Therefore, we classify the works on the NFV service chaining problem into different categories taking into considerations together the service deployment environment (Section 2.3.1), performance metrics (Section 2.3.2) and optimization goals and constraints (Section 2.3.3).

2.3.1 Deployment Environment

We mainly identify six deployment environments (see Table 2.1).

Environments	Main focus and/or quality indicator
General NFVI network	OpEx and CapEx costs, Computational resource utilization
Data Center (DC) network	Energy efficiency
Cloud network	Multi-domains (e.g., multi-DCs) infrastructure
Wireless and mobile edge network	Mobility, Radio resource utilization, Analog/digital modulation/demodulation
Mobile core network	Strict delay budget
Optical network	Spectrum resource utilization, Electronic and optical conversions

Table 2.1 – Summary of NFV service chaining deployment environments with main focus

Apart from general goals (e.g., costs minimization), specific ones may be introduced in different deployment environments, such as decreasing access latency (by placing network functions and certain services close to end-users) in mobile core network [17], and green networking (energy efficiency) in DCs [32]. We present in the following some of the main focuses in the state of the art of the NFV service chaining problem in each environment.

- *general NFVI network.* A general NFVI network can be represented by NFVI Points of Presence (NFVI-PoP), which means that a node can be any device in the network, such as a computational node, a router, an SDN (Software Defined Networking) controller, an access point, a network switch or any other type of device even a DC network. The main focus of the works considering a general NFVI network is to reduce the CapEx and OpEx costs of network service deployment;

- *data center network.* The goal of data center structure is to interconnect racks of servers with high-speed. Therefore, the basic data center network architecture is the 3-tier tree architecture, which can provide high speed forwarding. A key problem of the data center provider is the energy efficiency. Furthermore, when QoS (Quality of Service) requirements (e.g., service availability requirement) and the SLA (Service Level Agreement) violation (e.g., traffic delay requirement) are considered, the problem become more challenging since these goals are in competition with energy efficiency;

- *cloud network.* A cloud network is usually a multi-providers network where each provider is an independent domain consisting of a set of interconnected DCs. Cloud providers usually provide an abstraction of network topology as they do not possess full knowledge of the physical network topology in each domain. A key problem for cloud providers is how to conduct an effective service provision according to various domains, satisfying the bandwidth requirement of each service while saving as much cloud resource as possible as the cloud provider has to pay for the rented physical resources;

- *wireless network.* One of the main features of wireless network (and mobile edge network) environments is the mobility of their nodes. As a consequence, dynamic approaches that consider nodes mobility as a trigger for VNF-PR reconfiguration may be required in wireless networks. In this context the term capacity of NFVI is not related only to pure computational resources, such as number of CPU cores and memory, but, it refers also to packet forwarding, radio processing capabilities and analog/digital modulation/demodulation. Due to their stochastic nature, available bandwidth is a time–varying quantity in wireless networks. Channel fading, and also the distribution of end–users, can greatly influence the network performance. For example, users at the center of the cell will be, in general, able to use more efficient modulations and coding schemes thus achieving higher throughput for a fixed amount of radio resources than users at the edges of the cell. As a result, when the bandwidth–based provisioning model is employed, also the actual channel conditions experienced by the end–users must be taken into account [15];

- *mobile core network.* A mobile network comprises the Radio Access Network (RAN, also know as mobile edge network or wireless network) and the cellular core, known as the EPC (Evolved Packet Core). The RAN and EPC are typically connected via an optical transport network. The RAN mainly contains the base stations which provide radio access to the users. While the EPC consists of a range of data and control plane elements, responsible for routing, session establishment, mobility management, etc. The VNF-PR in mobile network should consider jointly the load balancing and latency, since there are strict delay budgets among communicating data and control plane elements [17];

- *optical network.* With the bandwidth in fibers, optical network provides a reliable infrastructure to support efficient high-throughput traffic provisioning [33]. However, in order to provide a network service via optical network, the VNF-PR needs to consider optimizing the utilization of spectrum resources in addition, thus special cost of electronic and optical conversions may occur [32].

2.3.2 System Performance Metrics

We mainly identify three categories of system performance metrics for the NFV service chaining problem (see Table 2.2). As discussed previously, system performance metrics can reflect the service deployment environments and they may affect the optimization focus of the resulting NFV service chaining problem. Moreover, these metrics are necessary to evaluate the efficiency of an optimization approach. Therefore, besides the service deployment environments, we also take into account the most used system performance metrics to provide the classification of the works on the NFV service chaining problem. We propose three categories of performance metrics. In what follows, we link these performance metrics with the optimization objectives and constraints considered in the works on the NFV service chaining problem.

Metric	Description
Resource consumption metrics	
Energy consumption	describes the number of active servers/physical machines
Resource utilization	describes the used quantity of computational resources
Economical metrics	
Acceptance ratio	describes the ratio between number of accepted traffic requests and total number of all the traffic requests
Cost	describes the cost of certain spent substrate resources
Revenue	describes the benefit of all accepted traffic requests
QoS and SLA metrics	
Delay	describes the time a packet/flow needs to travel across its path
Packet loss	describes network congestion conditions
Path length	describes the number of substrate links that are used by a traffic path
Reliability	describes the ability to provide and maintain an acceptable level of service in case of failures
Throughput	describes the data rate achievable between the nodes

Table 2.2 – Summary of system performance metrics with short descriptions

2.3.3 Main Optimization Objectives and Constraints

In this section, we associate the requirements of the system performance metrics mentioned above with the main optimization objectives and constraints studied in the literature. Moreover, for the sake of simplicity, we propose unique labels to represent the objectives and constraints (see the second column of Tables 2.3 and 2.4) for the classification of works on the NFV service chaining problem.

2.3.3.1 Main Optimization Objectives of NFV Service Chaining

In this part, we present two optimization objective groups according to the deployment environment and system performance metrics discussed in Section 2.3.1 and Section 2.3.2. Then for each objective group, we describe the objectives that are often pursued by existing approaches. A summary of the main optimization objectives of NFV service chaining is presented in Table 2.3.

Optimization objective	Label
Economical goals	
Number of admitted demands maximization	CST1
Node (or site opening) cost minimization	CST2
Link bandwidth (or path) cost minimization	CST3
Server (or energy) cost minimization	CST4
VNF cost minimization	CST5
SLA and QoS goals	
Availability maximization	SQG1
Delay minimization	SQG2
Link congestion minimization	SQG3
Maximum network traffic minimization	SQG4
SLA violation minimization	SQG5

Table 2.3 – Main optimization objectives and the corresponding labels

We mainly classify two groups of optimization objectives for the NFV service chaining problem:

- *economical goals*: Many works in the literature consider economical goals as optimization objective. In general, the economical goal can be expressed directly into two ways: the minimization of resources costs (e.g., cost metric and energy consumption metric) and the maximization of revenue profit (e.g., *revenue* metric). The latter is always measured by the number of admitted (served) demands (e.g, *acceptance ratio* metric). And for the former, according to the number of resources considered in the objective function, can be further divided into two groups: single resource cost minimization (e.g., the minimization of node cost or VNF instances cost) and multiple resource cost minimization (e.g., the minimization of both VNF cost and bandwidth cost). As multiple resources costs can be considered as the combinations of single resource cost, we classify five single objective functions: number of admitted demands maximization (acceptance ratio); node (site opening) cost minimization; link bandwidth (path) cost minimization; server (energy) cost minimization; VNFs cost minimization. Objective functions considering multiple resources are denoted by the combination of defined goals in one joint objective function. For example, in the classification Table 2.5, *2,3* in the *CST* column represents the joint nodes and links cost objective function;

- *SLA and QoS goals*: Meeting SLA/QoS requirements is one of the essential factors considered by many approaches in the literature. In this group, we mainly classify the following five objectives: availability maximization (e.g., *reliability* metric); delay minimization; link congestion minimization (i.e., used bandwidth minimization or remaining bandwidth maximization which may indicate the *packet loss* metric); maximum network traffic minimization (i.e., load balancing that may indicate the *throughput* metric); SLA violation minimization.

2.3.3.2 Main Optimization Constraints of NFV Service Chaining

Besides different objective functions discussed above, the NFV service chaining problem version varies also due to different constraints that are considered. In this part, we present three groups of optimization constraints that are usually considered in the existing work. A summary of the main optimization constraints is shown in Table 2.4:

Optimization constraint	Label
Network model constraints	
Bit-rate changes	NMC1
Limited link capacity	NMC2
Limited node capacity	NMC3
Limited server/VM capacity	NMC4
Limited VNF instance capacity	NMC5
Link latency	NMC6
VNF node sharing requirements	NMC7
VNF processing and forwarding latency	NMC8
Service demand model constraints	
Off-line	DMC1
On-line	DMC2
Schedule	DMC3
Partial ordered VNFs chain requirement	DMC4
Total ordered VNFs chain requirement	DMC5
Unordered VNFs chain requirement	DMC6
Unsplittable flow	DMC7
SLA and QoS constraints	
Load balancing	SQC1
Link availability	SQC2
Service availability	SQC3
Simple path routing	SQC4
Maximum allowed end-to-end latency	SQC5

Table 2.4 – Main optimization constraints and the corresponding labels

- *network model constraints*: Since there are multiple network environment as discussed in Section 2.3.1, papers that focus on different network environment may have a different level of network abstraction and consider different kind of network resources. For instance, in a general NFVI network, a single NFVI-PoP may represent a data center. In this case, node capacity limit may be considered, whereas the server capacity limit may be neglected. We mainly list eight optimization constraints referring to the network model:

1. bit-rate changes. Network functions may modify the traversing network flows in different way, for instance, firewalls may drop certain packets, leading to out-coming flows with a lower data rate than in-coming flows, while a video optimizer may change the encoding of the video, resulting in a higher data rate;
2. limited link capacity. Since high utilization of link bandwidth may cause network congestion, it is important to limit the occupation of link bandwidth;
3. limited node capacity (resource utilization);
4. limited server/VM capacity (energy efficiency);
5. limited VNF instance capacity (resource utilization);
6. link latency;
7. VNF node sharing requirements (e.g., VNF-server affinity or VNF-VNF anti-affinity requirements);
8. VNF processing and forwarding latency.

- *service demand model constraints*: A service demand in NFV service chaining is usually represented by a source node, a destination node and a sequence of VNFs (i.e., a VNF chain). Different types of service may introduce additional constraints on the NFV service chaining problem. For example, a demand asking for VoIP service needs to count on low delays, therefore a maximum allowed end-to-end latency may be imposed on the problem. In this group, we present seven optimization constraints that appear frequently in the existing work on NFV service chaining problem:

1. off-line (i.e., all the demands are given at the beginning);
2. on-line (i.e., batch of demands arrives dynamically);
3. schedule (i.e., demands are assigned to network resources according to the time-varying status of the resources).

The order of traversing the VNFs in a chain also requires special attention. As each VNF in the chain can modify the data rate of the flows, different chaining options can have different impact on the network traffic (e.g., latency), on the utilization of network resources, or on system performance. Therefore, we distinguish three chain order requirements:

4. partially ordered VNFs chain requirement;

5. totally ordered VNFs chain requirement;
6. unordered VNFs chain requirement;
7. unsplittable flow (i.e., a demand is not allowed being served by multiple VNF chains).

- *SLA and QoS constraints*: As discussed in the previous part 2.3.3.1, meeting SLA/QoS requirements in the network is considered as an essential factor to provision network services for many approaches in the literature. In this group, we mainly classify five optimization constraints:

  1. load balancing (e.g., *delay* metric);
  2. link availability (e.g., *link reliability* metric);
  3. service availability (e.g., *service reliability* metric);
  4. simple path routing (e.g., *path length* metric, *resource utilization* metric);
  5. maximum allowed end-to-end latency (e.g., *delay* metric).

2.3.4 Existing Modeling Strategies

In this section we provide an overview of the existing modeling strategies. Based on a business model for NFV [2] and impact of SDN/NFV on business models [34], we identify five main business roles/actors (the same actor may play several roles at the same time): *Infrastructure Provider (InP)*, *VNF/VM/Server Provider (VNFP)*, *Telecommunication Service Provider (TSP)*, *Network Operator* and *End User*. An example of the NFV business roles/actors and their relationships is provided in Figure 2.4.

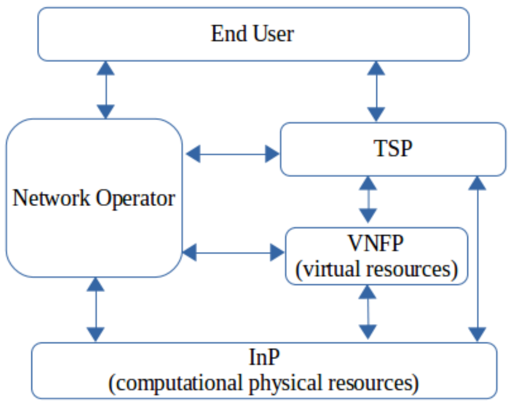


Figure 2.4 – An example of NFV business model

- InPs offer the other actors access to the computational physical resources in form of DCs and physical networks. It is on top of these resources that virtual resources may be provisioned and leased through programming interfaces to the other actors.

- VNFPs lease resources and run their VNFs, VMs or servers on top of the NFVI. They can provide these functions to potential users such as TSPs or Network Operators.
- TSPs may lease resources from one or more InPs for running VNFs. They can determine the chaining of these functions to create services for end users. In some cases, TSPs may sub-lease their virtual resources to other TSPs, and they take the role of an InP. And in some other cases, TSPs may lease VNFs from VNFPs to create services.
- Network Operators can also deploy and manage the end-to-end services running on resources from multiple InPs for the end users. Compared to TSPs, Network Operators is more likely to provide customized services asked by the end users. While TSP is more likely to create their own services and then allocate the services to the clients.
- Finally, End Users consume the services offered by the TSPs or Network Operators. And they may connect to multiple TSPs for different services.

As both TSPs and network operators may receive the service demands from the end users, they are the business roles concerned by the NFV service chaining problem. But they may have different perspectives: a network operator may offer and operate its NFVI with a resource-oriented model strategy, while a service provider may adopt a service-oriented model strategy. Therefore, they may adopt different modeling strategies to formulate the NFV service chaining problem:

- from a TSP point of view, NFV service chaining may be a two-level embedding problem: on the one hand, TSPs have to define network services, which are abstracted as VNF chains (i.e., described by VNF Forwarding Graphs) composed of virtual functions and virtual links that need to be implemented (i.e., embedded on top of the physical networks); on the other hand, TSPs have to allocate network service requests to the corresponding services (the implemented VNF chains). As a result, a two-level embedding strategy may be proposed to model the NFV service chaining problem.
- from a network provider perspective, a network service request can be modeled as follows: traffic (stream of packets) originating from a source is routed to a VNF instance, where the packets are processed and forwarded to a target. Thus the deployment of the network services is realized by flows that are routed from the source node to the destination node passing through the required VNF nodes. Therefore, there is no network service abstraction, the service deployment schema is the end-to-end traffic routing path for the demand/flow. We denote this modeling view with the VNF placement and demand routing strategy.

To the best of our knowledge, early works on the optimization of VNF chains deployment model the NFV service chaining problem as a VNE problem (i.e., using the embedding strategy). A virtual network is built for each demand representing the demand together with the chain of VNFs to serve it. The VNF instances location and demand routing are then defined according to the VNE mapping solution. For instance, in [35], authors introduce a VNE based Mixed Integer Programming (MIP) formulation; in [36] a VNE based Integer Linear Programming (ILP) formulation is proposed along with a dynamic programming based heuristic to deal with large size instances; [37] proposes a VNE based representation of the problem and focuses on the online version of the VNF chaining problem. The problem of embedding VNF chains is addressed also in [38], where demands may be rejected and the subset of accepted demands must be maximized while limiting the number of service chains considered. In [14], the authors propose an ILP model based on the mapping of VNF chains on a physical network, although without naming it explicitly. Precomputed paths for mapping are used. The ILP model is used as a step in a heuristic procedure based on a dichotomic search on the number of located VNFs.

However, as already discussed, the NFV service chaining problem differs from the general VNE problem and, as a consequence, the VNE representation paradigm is not always suitable for representing VNF problems (e.g., when the VNF chain is no ordered, or just partially ordered), nor it captures all the features of VNF problems (e.g., the possibility of sharing VNF resources among different demands. See [1] and [31] for more details). Therefore, the VNE based representation is possible only if the order of VNFs in the chain is fixed and fully determined. As a consequence, the virtual entity embedding strategy cannot always be applied to model the NFV service chaining problem, and the VNE complexity results cannot be just extended to the NFV service chaining problem as they are. Indeed, our work in [39] shows that the NFV service chaining problem is easier than the VNE in some particular case and for some network topologies.

Recently, the VNF placement and demand routing modeling perspective, which is not related to the VNE representation paradigm, has been adopted by a handful of papers in the networking literature and exploits the fact that the NFV service chaining problem shares features with both facility location and network design problems. In [1], authors provide an Mixed Integer Linear Programming (MILP) formulation accounting for facility location and demand routing with a generic number of services and no fixed order in the chain, taking into account different latency regimes and traffic compression properties. Their work investigates the trade-off between a legacy traffic engineering related goal (namely maximum link utilization) and a goal combining traffic engineering and Network Function Virtualization (namely, the sequential optimization of the traffic engineering goal and of the VNF installation cost). A model similar to [1] is proposed in [31], where additional constraints are added to take into account the incompatibility of certain VNFs and thus imposing that they are located in different nodes.

2.3.5 A Classification of the Approaches

Table 2.5 presents a problem features based classification of the main approaches on the NFV service chaining problem reviewed in this manuscript. Column 1 is the cited approach, and Column 2 shows the deployment context (Section 2.3.1). Columns 3-4 present the two objective categories and columns 5-7 present the three constraint categories previously introduced. Note that the numbers in Columns 3-7 point to the objectives/constraints labeled in Table 2.3 and Table 2.4. For example, number *5* in Column 3 (i.e., Column *CST*) refers to the *VNF cost minimization* objective labeled by *CST5* in Table 2.3. Column 8 is the formulation strategy: *E* represents *embedding*, and *P&R* represents *VNF placement and demand routing.* Columns 9-10 summarize the solution method. We further emphasize the importance of number of test scenarios in the last column, the reason of showing the number of test scenarios is discussed in the following with the scalability issue.

Ref.	Context	Objectives		Constraints			Strategies and methods			# Test
		CST	S&Q	NMC	DMC	SQC	Strategy	Exact	Heuristic	Scenarios
[1]	NFVI-PoP	5	4	1-8	1,6-7	4-5	P&R	MILP	Dichotomy	4
[31]	NFVI-PoP	3,5		2-3,5-7	1,4-5,7	4-5	P&R	ILP	LP relaxation	1
[36]	NFVI-PoP	3-5		2,4-7	1-2,5,7	5	E	ILP	DP	2
[40]	NFVI-PoP	3-5	5	2,4-7	1-2,5,7		E	ILP	DP	3
[26]	NFVI-PoP	2-3,5		2-3,5,7	1,7		P&R	ILP	Greedy	17
[17]	Mobile core	3-4		2-4,6	1,5,7	5	E	MILP	LP relaxation	1
[41]	DC	3,5		2-3,5	1-3,7		P&R	MILP	Greedy	1
[42]	DC	3,5		2-3,5,7	1-2,5		P&R	MIP	Local search	1
[43]	DC	1		2-5	1-2,5,7		E	MILP	DP	36
[14]	NFVI-PoP	5		2-3,5-8	1,5,7	5	E	ILP	Dichotomy	90
[44]	NFVI-PoP	5		2-3,5-8	1,5,7	5	E	ILP	VNS	1
[45]	NFVI-PoP	2	2-3	1-3,5-7	1,4-7	5	E	MIQCP	Greedy	1
[46]	NFVI-PoP	4		2-7	1,5,7	4-5	E	ILP	-	1
[15]	Wireless	2-3,5		2-3	1,7		E	ILP	Greedy	3
[16]	Wireless	3,5		2-3	1,7		E	ILP	Greedy	3
[47]	NFVI-PoP	5	4	1-8	1,4-5,7	4-5	P&R	MILP	Dichotomy	8
[48]	NFVI-PoP	5		2,5,7	1,7	4	P&R	ILP	-	144
[49]	NFVI-PoP	5		2,5,7	1,7	4	P&R	ILP	Local search	132

Table 2.5 – A classification of the approaches on the NFV service chaining problem proposed in the literature (DP refers to dynamic programming, VNS refers to variable neighborhood search.

Most of the approaches in the literature tend to design scalable optimization methods to solve the NFV service chaining problem. This is because, on the one hand, the NFV service chaining optimization problem is hard to solve; on the other hand, a scalable networking solution is necessary to achieve the full benefits of NFV [50], since NFV is expected to overcome the ossification issue of the current telecommunication network by automatically and dynamically deploying and removing VNFs to match changing traffic.

However, few of the work evaluates the proposed scalable solutions by using different test scenarios to justify the efficiency of the methods. According to the results of our investigation into the problem complexity and property, as well as the results of extensive experimental tests under different scenarios, we find that the efficiency to deal with the scalability is connected not only with the network size, the number of traffic demands and the number of VNFs, but it is influenced also greatly by the structure of tested network topology, the feature of considered resources (e.g., capacity) and the feature of demands distribution. Authors in [26] also observe the phenomenon that the size and the complexity of the problem instance are tightly linked with structure of the graph (e.g., a big number of links may offer more possible routes), according to the experimental results of both ILP and heuristic methods.

Besides the graph structure, we find by extensive experimental tests that, under certain capacity limits, even an ILP model can solve efficiently the problem (e.g., within 10 seconds to find an optimal solution for the VNF-PR problem with 27 nodes and 702 demands). Therefore, we believe that it is necessary to evaluate the approaches on multiple different scenarios (i.e., different graph structures, different demands distributions and different resource capacity limits) to avoid such easy cases. To this end, we report the number of test scenarios in the last column of Table 2.5, which is calculated as follows: *the number of test scenarios = the number of different graph structures ∗ the number of different resource capacity settings ∗ the number of different demand distribution case-studies.*

2.4 Overview of Related Work

The research works contributing to the NFV service chaining problem can be generally divided into two classes based on the considered objective function:

- the objective function of the first class is to maximize the number of admitted demands. Approaches of this class, such as works in [38] [43] [51], focus on maximizing the served demands by optimally allocating network resources. Generally speaking, it is easy to find a feasible solution with this objective function, e.g., an intuitive solution is to admit one single demand;
- the objective of the second class is to minimize a certain cost/utilization while meeting the requests of all the demands. In fact, solving the problem of this class is more challenging as finding a feasible solution (where all the demands need to be served) can be difficult (if the problem is not impossible to solve). There is already a certain number of studies in this scope and our works in this thesis also fall into this class. According to the criteria considered in the objective function, the approaches can be further divided into two types:
  - approaches that consider one single objective function (e.g., NFVI node cost minimization, path cost minimization or VNF cost minimization), such as works in [14, 44–46, 48, 49];
  - approaches that consider one combination of multiple goals (e.g., the minimization of both server utilization and link utilization), such as works in [1, 15–17, 26, 31, 36, 40–42, 47, 52].

Works in [45, 46] are among the first ones to address the NFV service chaining problem by formalizing it as an optimization problem. Authors in [46] decouple the legacy VNE problem into two embedding problems: VMs embedding and service requests embedding, where service requests are embedded on service chains, and VMs on physical networks. Each service request has specific requirements as notably an end-to-end latency requirement. However, the proposed model can only deal with totally ordered VNF chain requests. Authors in [45] deal with a NFV service chaining problem with totally and partially ordered VNF chain requests. They consider

3 different optimization goals separately and perform a Pareto set analysis to investigate the possible trade-offs between different optimization objectives. However, in the proposed two-step solution process, the partially ordered VNF chain requests are firstly formalized and built by using a context-free language, i.e., for each request, a VNF graph is defined by explicit orders of VNF nodes and in-between the virtual links with explicit bandwidth requirement are computed before solving the problem, then the predefined VNF graphs are mapped onto the substrate network by using a Mixed Integer Quadratically Constrained Program (MIQCP). Therefore, the final solutions found by this approach are not guaranteed to be optimal as the partially ordered VNF chains are pre-computed.

In both approaches [46] and [45], the proposed methods are evaluated on a single small scenario (e.g., one case-study of resource capacities on one graph with 12 nodes), whereas according to our study in this thesis, it is necessary to consider multiple test scenarios (i.e., with different resource capacity limits, different graph structures, and different demand distributions) to evaluate the proposals' performance. In fact, we proved in [39] that finding a feasible solution on a general graph with resource capacity limits is NP-hard, e.g., a feasible solution may be hard to provide if the capacity constraint is very strict. However, an optimal solution may be easy to provide if the capacity constraint is very loose or the graph is a particular one, e.g., a fully connected graph. Most of the approaches in the literature consider single test scenario, and some consider scenarios with variation of nodes for the scalability test, only few consider various scenarios with different resource settings, such as in [14, 26, 47–49].

Authors in [14] introduce a general ILP model based on the mapping of VNF chains on the physical network, taking into consideration the end-to-end delay and resource constraints. The objective function of their model aims at minimizing the number of VNF instances mapped on the infrastructure, which has the most significant and direct impact on the network provider's costs. Then the authors go one step further in [44] and address the scalability of the problem to solve large instances. They propose a novel fix-and-optimize approach which combines the model defined in [14] and Variable neighborhood Search (VNS). However, the evaluation of the proposed heuristic methods in [44] considers only scenarios with single resource capacity, where an initial feasible solution can be found fast by solving the ILP model dropping the objective function. In this thesis, we consider various test scenarios and the computational results show that for some case-studies, it is even difficult to provide a feasible solution within reasonable computational time (e.g., 3600s).

In [26] the specific DPI VNF node placement and traffic routing problem is targeted and modeled as an adaptation of the multi-commodity flow problem model. The authors focus on the case of single VNF type, where an operator has to run DPI functions monitoring each flow on its own NFV infrastructure, for accounting or cyber-security purpose. They provide a general ILP formulation of the vDPI deployment minimizing the overall cost (nodes and links) as viewed by the network operator. A graph centrality-based greedy algorithm is also proposed to minimize the number of nodes upon which DPI VNFs are activated: at each step a new DPI VNF is located in the node that has the highest centrality until all the traffic flows are served or all nodes have a vDPI. They apply both the ILP and the heuristic to larger random networks of up to 100 nodes and based on the flat and Barabasi models. Their results show that the network structure and the costs strongly influence time performance.

Authors in [41] also consider single VNF type. They formulate the static VNF chains embedding problem considering the trade-off between host and bandwidth resources consumption and elasticity. However, they assume that each demand consists of one unit of traffic transmitted from its source to a VNF instance and delivered to its target, and requires a unit of VNF resource to be served, which simplifies a lot the problem. In order to cope with on-demand version of the problem, the authors propose a consolidation algorithm called Simple Lazy Facility Location (SLFL) that optimizes the placement of the VNF instances in response to on-demand workload. In another study [42], the authors consider distributed service deployment (i.e., they assume that a service request flow can be served by multiple VNF chains, thus by multiple instance of a same VNF type), proposing an MIP model and a local search heuristic called *Kariz* for multiple VNF instances placement to provide decomposition-based approach for the placement of VNF chains. However, they adopt demand rejection model in the heuristic method to maximize the provider's revenue based on the number of accepted CPU and bandwidth resources.

In [1], the authors evaluate the mutual impact of multiple goals for a realistic NFV service chaining problem. They investigate the trade-off between legacy TE goal (namely maximum link utilization) and a combined TE-NFV goal (namely, the sequential optimization of the TE goal and of the VNF installing cost). They provide an MILP formulation to deal with the case of NFV service chaining problem with multiple VNF types and unordered chains, where different latency regimes and traffic compression properties are taken into account to capture and investigate the practical feature of traffic flow compression and decompression (bit-rate changes) that could be imposed by the VNFs along the traffic route. A model similar to [1] is proposed in [31], where additional constraints are added to take into account the incompatibility of certain VNFs and thus imposing that they are located in different nodes. An ILP model is proposed to solve small instance of the NFV service chaining problem, minimizing the total financial cost of demands routing and VNFs installation to serve all the demands. The authors also adapt their ILP model to allow demand rejection decisions, and they propose a linear relaxation based heuristic approach to cope with large instances. Our proposal in [47] extend the work in [1]. We focus on providing a more generic formulation to the NFV service chaining problem. In addition to different latency regimes, different optimization goals, and traffic compression properties, we further take into account different demand distribution scenarios. We perform a sensitivity analysis to put in evidence the effect of relaxing the TE objective with respect to the NFV optimal cost objective, with the purpose to further evaluate the trade-off between the two objectives. Furthermore, a dichotomy based math-heuristic is presented to speed up the solution phase and a numerical comparison with a *VNE based* model approach is proposed.

Online and schedule versions of the NFV service chaining problem also have been studied in the literature. The authors in [36] tend to solve a NFV service chaining problem aiming at optimizing network operational cost and resource utilization. In addition, they consider a penalty cost to be paid to the customer for the service level objective (SLO) violations. They propose an ILP and a dynamic programming heuristic to solve the online VNF placement by running the Viterbi algorithm for a multi-state directed graph with associated costs built by the authors. An extension of the work [36] is presented in [40]. In [53], the authors present a formulation of the NFV service chaining scheduling problem and they provide a novel primal-dual decomposition using column generation that solves exactly a relaxed version of the problem and can serve as a benchmark approach.

The NFV service chaining problem has also been addressed in specific contexts, such as wireless local area networks and mobile core networks. In [15], the authors formalize the NFV service chaining problem for wireless networks as an ILP problem for small networks and they propose a scalable VNFs placement heuristic called WiNE (Wireless Network Embedding) to deal with large instances. Besides the common computational resources (i.e., CPU, memory and storage), the authors also introduce the radio resources at each NFVI nodes (radio-enabled nodes) to consider the specific wireless network features (e.g., virtualization resource of 802.11 devices). The work intends to minimize the links and nodes utilization to increase the accepted service chain requests in wireless networks. Afterwards, the authors extend their work in [16], investigating the VNF placement and scheduling problems in the context of wireless networks under the hypothesis that the service requests have already been specified in terms of VNF-FGs by the mobile virtual network operators (MVNOs). However, in their work, a service demand is considered as a chain composed of VNF nodes, therefore, the VNF instances are not shared among service demands, each demand has its own VNF instance to be mapped to the substrate network. In [17], the authors tackle the problem of VNF chains placement onto the mobile core network. An MILP model is introduced for the optimal placement of VNF chains to serve all the requests, considering network resource capacity constraints and special 3GPP standards-related constraints, such as delay budgets between EPC components (i.e., each virtual link) and strict order requirements of VNFs. The objective is to efficiently map the VNF chains onto the substrate network while jointly minimizing the maximum server utilization and the maximum link utilization with respect to the delay budgets between each EPC elements. However, as the relation between link and server usage varies with the system status, the selection of weights should be adapted to the status, which has a big impact on the problem resolution.

To conclude, most of the approaches in the literature tackle application related problems and focus on developing heuristic methods to solve the problem within a reasonable computational time, and, to the best of our knowledge, the problem theoretical properties have not been studied so far. However, the problem is interesting also from an optimization point of view. The NFV service chaining problem shares features with network design problems (for the demand routing part) and with facility location problems (for the VNF location and the server dimensioning part). Both problems are widely studied in the literature, but although the combined facility location and network design problem has been studied in optimization literature [54], the NFV service chaining problem peculiarities are still not explored in the optimization literature and provide a challenging topic. Further, the problems addressed in the literature differ in the constraints, in the technical features and in the assumption considered. Thus, the approaches proposed in different papers are tailored on specific versions of the problem and heuristics or formulations have not been compared on a common data set. In this thesis, besides the practical work in [47], we also work on the problem resolution from the theoretical point of view. We provide a thorough complexity and properties analysis in [39] of a particular version of the NFV service chaining problem with respect to some design considerations. Then in [48], we propose and compare two different formulations that are representative of each modeling perspectives presented in the literature. Furthermore, we propose ILP based exact and heuristic methods in [49] to improve the problem resolution in comparison with the mathematical model, and we carry out extensive experimental tests to evaluate our proposals with different scenarios (i.e., more than 20 graph structures with 6 capacity case-study and 2 types of VNFs chain model). Moreover, we explore other optimization methods to deal with our problem: we propose a Farther Node (FN) formulation (see Appendix A) and we implement a column generation based method (see Appendix B).

Chapter 3

The Application Problem

Contents

3.1 Problem Features Overview 33
3.2 Problem Statement and Mathematical Formulation . . . 34
3.2.1 Basic Placement and Routing (PR) Model 36
3.2.2 Virtual Network Function (VNF) Forwarding Latency . . . 40
3.2.3 Bit-rate Compression/Decompression 41
3.2.4 Multi-objective Math-heuristic Resolution 45
3.3 Computational Results 46
3.3.1 Results of PR Algorithm under Different Cases of Demand Distribution and Different VNF Forwarding Latency Profiles 46
3.3.2 PR vs. VNE Based Approaches 62
3.4 Further Model Refinements 64
3.5 Conclusion 67

In this chapter, we focus on a realistic NFV service chaining problem[1], proposing a mathematical formulation to solve the considered problem making to optimality under reasonable execution time targets. Section 3.1 gives an overview of the tackled NFV service chaining problem. Section 3.2 formally defines the problem and presents the proposed MILP formulation. Section 3.3 gives analysis and discussion of optimization results. Section 3.4 presents the possible customization alternatives for the proposed formulation.

3.1 Problem Features Overview

In the NFV service chaining [6] provisioning, key aspects that are worth being mentioned (and often neglected in the proposed solution strategies) are the:

- ingress/egress bit-rate variations at VNFs, due to specific VNF operations (such as compression as with a firewall function or an egress tunneling function, or such as decompression as in an ingress tunneling function);

[1]This chapter is based on the publication [47].

- VNF processing and forwarding latency as an orchestration parameter. It can indeed be exponential with the traffic load on the VNF, or constant up to a maximum board if computation offloading solutions, such as direct memory access bypassing the hypervisor (as done with Intel/6WIND Data-Plane Development Kit [55]), or similar other 'fastpath' solutions are present.

We could not identify a work in the state of the art jointly taking the above aspects all together into account. In this context, our contribution is as follows:

- we define and formulate via mathematical programming the VNF-PR optimization problem, including compression/decompression constraints and two forwarding latency regimes (with and without fastpath), under both TE and NFV objectives; our mathematical programming model is the first one in the literature taking into account explicitly all these key aspects together;

- we compare our Placement and Routing (PR) approach to the legacy VNE based approach, qualitatively and quantitatively; to the best of our knowledge, we are the first to propose such formulation comparison;

- we design a model tailored math-heuristic approach allowing us to run experiments also for large instances of the problem within an acceptable execution time;

- we evaluate our solution by extensive simulations. We draw considerations on NFV deployment strategies.

3.2 Problem Statement and Mathematical Formulation

The network is represented by a graph $G(N, A)$, where N is the set of switching nodes, and A represents the possible directional connections between nodes. The router $i \in N$ and its associated NFVI cluster are represented by the same node; this choice allows to keep the size of the graph limited and reduces the computational effort. We represent with $N_v \subset N$ the set of nodes N disposing of NFVI server clusters. We consider a set of demands D, each demand $k \in D$ is characterized by a source o_k, a destination t_k, a nominal bandwidth d_k (statistically representative for demand k), and a sequence of VNFs of different types, that must serve the demand (and therefore must be traversed by the demand). For each VNF a single VM is reserved, therefore we can equivalently speak of allocating a VM or a VNF on an NFVI node, meaning that we are reserving the necessary resources (e.g., CPU, RAM) to host a VM running a VNF.

The VNF-PR optimization problem is to find:

- the optimal placement of VNF instances over NFVI nodes;

- the optimal routing for demands and their assignment to VNF node chains.

- subject to:

- link capacity constraints;
- NFVI node capacity constraints;
- VNF flow compression/decompression constraints;
- VNF forwarding latency constraints;
- VNF node sharing constraints;
- VNF chain (total or partial) order for each demand.

The optimization objective should contain both network-level and NFVI-level performance metrics. In our network model, we propose as network-level metric a classical TE metric, i.e., the maximum link utilization and as NFVI-level metric a measure of the overall allocated computing resources. Both objectives are - in a sequential way - minimized in the optimization model.

Furthermore, we assume that:

- multiple VNF instances of the same type (i.e., same functionality) can be allocated on the same node. Each VNF instance can serve multiple demands (this level of granularity allows to model the forwarding latency introduced by VNF instances), but each demand cannot split its flow on multiple VNF instances of the same type;

- the VNF computing resource consumption can be expressed in terms of live memory (e.g., RAM) and Computing Processing Units (CPUs), yet the model shall be versatile enough to integrate other computing resources;

- latency introduced by a VNF instance can follow one among the two following regimes (as represented in Fig. 3.1):

  - *Standard*: VNFs buffer traffic at input and output virtual and physical network interfaces such that the forwarding latency can be considered as a convex piece-wise linear function of the aggregate bit-rate at the VNF, due to increased buffer utilization and packet loss as the bit-rate grows as shown in [55, 56]. This is the case of default VNFs functioning with standard kernel and hypervisor buffers and sockets;
  - *Fastpath*: VNFs use optimally dimensioned and relatively small buffers, and decrease the number of times packets are copied in memory, so that the forwarding latency is constant up to a maximum aggregate bit-rate after which packets are dropped (e.g., this happens for Intel/6WIND DPDK fastpath solutions [55]).

  Fig. 3.1 gives examples of forwarding latency profiles for the two cases;

- for each demand and NFVI node, only one compression/decompression VNF can be installed. This allows us to keep the execution time of the optimization algorithm at acceptable levels, without reducing excessively the VNF placement alternatives. This assumption can be relaxed at the cost of working on an extended graph, and therefore increasing the computational time of the algorithm.

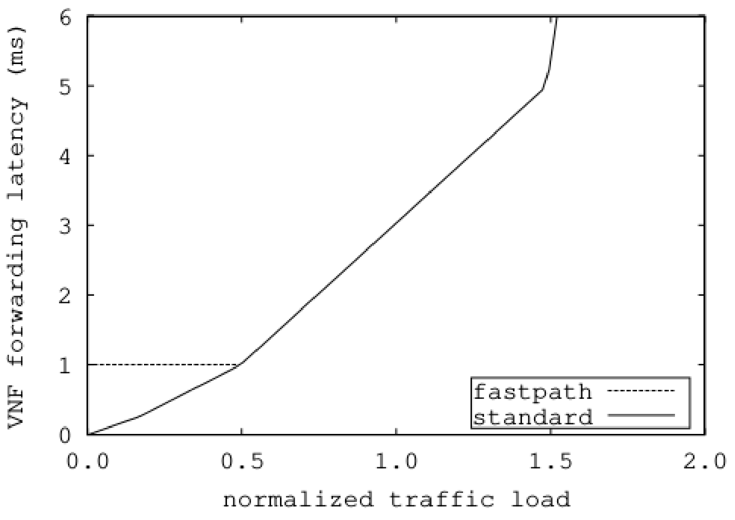


Figure 3.1 – Example of VNF forwarding latency profiles.

In the following, we first introduce a basic model that does not take into account latency limitations and compression/decompression features. The reason of this choice is twofold. First, it allows a clearer explanation of the model and a step by step introduction of the technicalities that allow us to keep the model linear; we recall that already without these two features, the model is a combination of a network design and a facility location. Second, as the model proved to be difficult to solve at optimality using a state of the art solver, we design a sequential procedure to reduce the solution computational time. In particular, instead of solving the overall model from the beginning, we solve a sequence of problems where the model details (latency, compression/decompression) are added step by step (see Section 3.2.4 for a detailed description of the procedure). Therefore, this presentation allows to put in evidence the peculiarities of each model component.

3.2.1 Basic Placement and Routing (PR) Model

Table 3.1[2] and 3.2 report the mathematical notation used in the following Mixed Integer Linear Programming (MILP) that represents the basic formulation of the VNF-PR problem.

We use four families of binary variables. The first family is used to represent the path used by each demand:

- x_{ij}^k represents the link utilization for demand k;

The other three families of binary variables are used to represent the VNF instantiation and the assignment of demands to VNF instances. Each VNF instance is

[2]The notation used is slightly different from the publication [47] to have an uniform notation along the manuscript.

[3]the maximum number of instances of VNF type f on a node i is limited by node capacity, therefore without loss of generality, we can set $c_i^f = \lfloor \min_{r \in R} \frac{\Gamma_{ir}}{rr_r} \rfloor$

Sets	
N	all nodes
$N_v \subseteq N$	nodes equipped with an NFVI cluster
$A \subseteq N \times N$	all arcs (links)
D	demands
R	resource types (CPU, RAM, ...)
F	VNF types
Parameters	
network parameters	
u_{ij}	link capacity
Γ_{ir}	capacity of node $i \in N_v$ in terms of resource $r \in R$
demand parameters	
o_k	origin of demand $k \in D$
t_k	destination of demand $k \in D$
d_k	nominal bandwidth of demand $k \in D$
m_k^f	1 if demand $k \in D$ requests VNF of type $f \in F$
s_k^f	order coefficient for VNF f requested by demand k
VNF/VM parameters	
rr_r	demand of resource $r \in R$ for a VM
c_i^f	maximum number[3]of instances of VNF f on node i

Table 3.1 – Parameter notations for base model.

Binary variables	
x_{ij}^k	1 if arc (i,j) is used by demand $k \in D$
z_{ik}^{fn}	1 if demand $k \in D$ uses the n-th instance of VNF of type $f \in F$ placed on node $i \in N_v$
y_i^{fn}	1 if the n-th instance of VNF of type f is assigned to node $i \in N_v$
Continuous variables	
$U \geq 0$	maximum allowed link utilization
$\pi_{ik} \geq 0$	label of position of node i in the path used by demand k

Table 3.2 – Variable notations for base model.

represented by a triple (i, f, n), indicating the n-th instance of a VNF of type f on node i;

- y_i^{fn} represents the allocation of the instance n of a VNF of type f on a node i;
- z_{ik}^{fn} represents the assignment of a demand k to instance n of a VNF of type f (multiple demands can be assigned to the same VNF instance);

The continuous variable U is used to represent the maximum link utilization in the network, that is the maximum fraction of link capacity that is used by the current solution. Continuous variable π_{ik} is used to represent the position of node i in the path used to route demand k. This family of variables is necessary to impose (total or partial) order in the VNF chain. As mentioned before, we consider two objective functions:

- TE goal: minimize the maximum network link utilization:

$$\min U \tag{3.1}$$

- NFV goal: minimize number of cores (CPU) used by the instantiated VNFs:

$$\min \sum_{i \in N_v} \sum_{f \in F} \sum_{n \in 1..c_i^f} rr_{CPU}\ y_i^{fn} \tag{3.2}$$

The former objective allows taking into consideration the inherent fluctuations related to Internet traffic and therefore minimizing the risk of sudden bottleneck on network links in the case bandwidth demands deviate from the expected values. Therefore, minimizing the fraction of link capacity that can be used, indirectly allows controlling the network congestion. The latter assumes the fact that today the first modular cost in virtualization servers, especially in terms of energy consumption and monetary cost, is the CPU. The reason is that, in common VM templates, the RAM resource is positively correlated to the CPU resource, considering CPU as reference cost indicator can indirectly imply also the consideration of the RAM consumption. We now present the constraints.

Single path flow balance constraints:

$$\sum_{j:(i,j) \in A} x_{ij}^k - \sum_{j:(j,i) \in A} x_{ji}^k = \begin{cases} 1 \text{ if } i = o_k \\ -1 \text{ if } i = t_k \\ 0 \text{ otherwise} \end{cases} \quad \forall k \in D, \forall i \in N \tag{3.3}$$

Utilization rate constraints:

$$\sum_{k \in D} d_k x_{ij}^k \leq U u_{ij} \quad \forall (i,j) \in A \tag{3.4}$$

Node resource capacity (VNF utilization) constraints:

$$\sum_{f \in F} \sum_{n \in 1..c_i^f} rr_r y_i^{fn} \leq \Gamma_{ir} \quad \forall i \in N_v, r \in R \tag{3.5}$$

Each demand uses exactly one VNF of each required type:

$$\sum_{i \in N_v} \sum_{n \in 1..c_i^f} z_{ik}^{fn} = 1 \quad \forall k \in D, f \in F : m_k^f = 1 \tag{3.6}$$

Constraints (3.7)-(3.8) are consistency constraints among binary variables. A VNF can be used only if present, for a given node:

$$z_{ik}^{fn} \leq y_i^{fn} \quad \forall k \in D, i \in N_v, f \in F, n \in 1..c_i^f \tag{3.7}$$

If a demand does not pass by a VNF, it cannot use it:

$$z_{ik}^{fn} \leq \sum_{j:(j,i) \in A} x_{ji}^k \quad \forall k \in D, i \in N_v, f \in F : m_k^f = 1 \tag{3.8}$$

Finally, we introduce constraints to avoid unfeasible routing and to impose the VNF chain order:

- Preventing the formation of isolated cycles:

$$\pi_{jk} \geq \pi_{ik} + x_{ij}^k - |N_v|(1 - x_{ij}^k) \quad \forall k \in D, (i,j) \in A \tag{3.9}$$

- Imposing an order for virtual functions:

$$\begin{aligned} \pi_{jk} \geq \quad & \pi_{ik} - (|N_v| + 1)(2 - \textstyle\sum_{n \in 1..c_i^{f_1}} z_{ik}^{f_1 n} - \sum_{n \in 1..c_i^{f_2}} z_{ik}^{f_2 n}) \quad \forall k \in D, \\ & \forall i, j \in N_v, f_1, f_2 \in F : s_k^{f_2} \geq s_k^{f_1} \end{aligned} \tag{3.10}$$

If we consider flow balance constraints (3.3) and link capacity constraints (3.4), for each demand, a selection of arcs forming a path plus an isolated cycle can be a feasible solution. In pure routing problems, these solutions are equivalent to the solution where routing variables along the cycle are removed and only the one along the path are kept. In fact, both constraints (3.3) and constraints (3.4) will be valid for this new solution. Our problem integrates routing features within a facility location problem, therefore such solutions cannot always be transformed in a simple path simply removing the cycle. In fact, if a facility (VNF) used by the demand is located on the cycle, removing the cycle will produce an unfeasible solution. Therefore, it is necessary to remove such solutions directly in the model, to this aim we introduced constraints (3.9), inspired by traveling salesman problems tour elimination constraints [57]. Variable π_{ik} represents the order of node i along the path serving demand k, therefore if arc (i, j) exists, then π_{jk} will be at least π_{ik} plus 1. On the other side, if arc (i, j) does not exist ($x_{ij}^k = 0$) then the constraint is not active: π_{ik} is always smaller than $|N_v|$, as a path can contain at most all the nodes in the graph. In this way, only solutions containing simple paths are allowed. These variables are also used in Equation (3.10) to allow imposing an order on VNFs along the route of a demand k. They impose that if demand k uses VNF f_1 located on node i and its VNF successor f_2 ($s_k^{f_2} \geq s_k^{f_1}$) that is located on node j, then in the routing path of demand k node i must be precede node j.

3.2.2 Virtual Network Function (VNF) Forwarding Latency

Parameters	
L	maximum allowed latency for a demand
λ_{ij}	latency introduced by link $(i,j) \in A$
standard latency model	
$g^f_j(b)$	j-th component of the linearized latency function for VNF f and aggregated bandwidth b
n_g	number of piece-wise components of lin. latency function
fastpath latency model	
$\bar{l}^f$	latency introduced by VNF f
B^f_{max}	maximum allowed bandwidth to traverse VNF f

Variables	
$l^f_{ik} \geq 0$	latency that demand $k \in D$ incurs using VNF f on node i of type $f \in F$ hosted by node $i \in N_v$

Table 3.3 – Notations to model forwarding latency.

Table 3.3 reports the mathematical notation used to model the VNF forwarding latency. We impose that for each demand $k \in D$ a maximum latency L is allowed, to guarantee some level of QoS. Latency depends on two components: link latency, represented by a parameter λ_{ij} for each arc (i,j), and VNF latency. VNF latency depends on the used model of latency (*standard* or *fastpath*). To keep the notation as uniform as possible, we introduce an additional variable l^f_{ik} to represent the latency experienced by demand k traversing VNF f located on node i. Therefore we get a set of constraints common to both models limiting the overall latency:

$$\sum_{(i,j)\in A} \lambda_{ij} x^k_{ij} + \sum_{i\in N_v} \sum_{f\in F} l^f_{ik} \leq L \quad \forall k \in D \tag{3.11}$$

A set of constraints depending on the chosen latency model allows to calculate the value of variable l^f_{ik}.

- *Standard*: the latency introduced on demand k for using VNF f depends on the overall traffic traversing the VNF (its own and the one of other demands). Let us call $g^f_j(\cdot)$ the j-th component of the piece-wise linearization of the latency function for VNF of type f, then we get:

$$l^f_{ik} \geq g^f_j(\sum_{b\in D} d_k z^{fn}_{ib}) - L(1 - z^{fn}_{ik})$$
$$\forall k \in D, i \in N_v, f \in F, n \in 1..c^f_i, j \in 1..n_g \tag{3.12}$$

  We can observe that constraint (3.12) is active only when the demand uses instance n of VNF f on node i ($z^{fn}_{ik} = 1$). Otherwise, as overall latency is limited by L (constraint (3.11)), any term $g^f_j(\cdot)$ must be smaller than L, and therefore, the constraint is redundant ($l_{ik} \geq 0$). We can observe that, even if in the *standard* latency model there is no theoretical limit on the allowed bandwidth, constraint (3.11) – limiting the overall latency for a single demand (VNFs forwarding latency plus propagation delay) – imposes an implicit limit on the maximum attainable bandwidth for a single VNF.

- *Fastpath*: the latency is fixed, but a limit in the total traffic that a VNF can support is imposed. Therefore we get the following two sets of constraints:

$$l_{ik}^{f} = \bar{l}^{f} \quad \forall k \in D, i \in N_v, f \in F \tag{3.13}$$

$$\sum_{k \in D} d_k z_{ik}^{fn} \leq B_{max}^{f} \quad \forall i \in N_v, f \in F, n \in 1..c_i^f \tag{3.14}$$

We can observe that constraints (3.13) can be substituted directly in constraints (3.11). Constraints (3.14) impose that the total bandwidth traversing the n-th VNF instance of type f on node i is limited by the maximum amount of bandwidth that the VNF instance can support before rejecting a demand. Therefore, in our solutions no demand is discarded due to the fastpath mechanism.

3.2.3 Bit-rate Compression/Decompression

To introduce the possibility of compressing/decompressing flows for some VNFs, we need some modifications to our model description. Table 3.4 reports the notations used to model the flow variations. We introduce a compression/decompression parameter μ_f for each type of VNFs, $\mu_f > 1$ means that a decompression is performed by VNF f. When a demand pass through a VNF with $\mu_f \neq 1$, its bandwidth changes, therefore knowing just the routing (x variables) is not enough to determine the overall flow along an arc. For this reason we introduce variable ϕ_{ij}^{k} that represents explicitly the flow on arc (i, j) for demand k. Another consequence is that the classical flow balance equations are not valid anymore. To extend the model without introducing an excess of complexity, we work under the assumption that given a node i, and a demand k, such demand uses at most a VNF f with a factor of compression/decompression ($\mu_f \neq 1$).

Sets	
N_a	access nodes
N'_a	duplication of access nodes, where demands are located

Parameters	
μ_f	compression/decompression factor for VNF $f \in F$
d_k^{min}	minimal bandwidth of demand $k \in D$
d_k^{max}	maximal bandwidth of demand $k \in D$
M_i	maximum traffic volume that can be switched by node i

Variables	
$\phi_{ij}^k \geq 0$	flow for demand $k \in D$ on arc (i, j)
$\psi_{ik}^{fn} \geq 0$	flow for demand $k \in D$ entering node i and using instance n of VNF $f \in F$

Table 3.4 – Notations to model bit-rate variations.

We work on an extended graph to distinguish between access nodes (origin/destination nodes) and NFVI nodes (remind that with the basic model we collapsed NFVI nodes on router nodes). Each access node i is duplicated in a node i' as shown in Fig. 3.2.

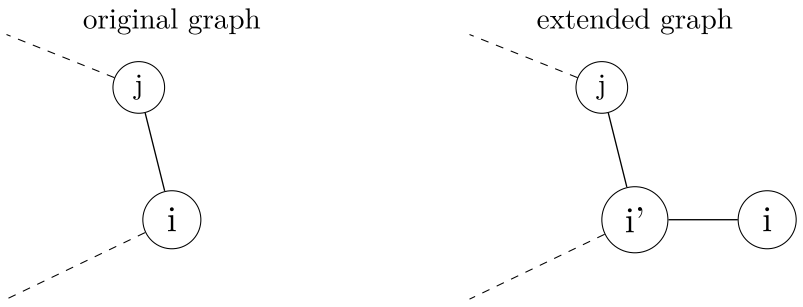


Figure 3.2 – Duplication of an access node i.

Arc (i, i') will be added and all arcs (i, j) originating from access node i will be transformed in arcs (i', j). Therefore, the routing functionality is on node i and the NFVI functionality can be allocated on node i'. Furthermore, we add variable ϕ_{ik}^{fn}, that represents the flow of demand k entering node i and using the instance n of the VNF of type f. If a demand passes through a VNF with a factor of compression/decompression μ_f, then the out-flow of the node is proportional to the in-flow:

$$\sum_{j \in N:(i,j) \in A} \phi_{ij}^k = \mu_f \sum_{j \in N:(j,i) \in A} \phi_{ji}^k$$

or equivalently:

$$\sum_{j \in N:(i,j) \in A} \phi_{ij}^k - \sum_{j \in N:(j,i) \in A} \phi_{ji}^k = \sum_{j \in N:(j,i) \in A} (\mu_f - 1)\phi_{ji}^k$$

This equation is valid only if demand k uses an instance n of VNF f on given node i (remind that latency depends on the bandwidth passing through a single instance).

Therefore, to obtain a valid equation, we have to write:

$$\sum_{j\in N:(i,j)\in A}\phi_{ij}^{k}-\sum_{j\in N:(j,i)\in A}\phi_{ji}^{k}=$$
$$\sum_{j\in N:(j,i)\in A}\phi_{ji}^{k}\sum_{n\in 1..c_i^f}(\mu_f-1)z_{ik}^{fn}$$

when $\sum_{n\in 1..c_i^f}(\mu_f-1)z_{ik}^{fn}=0$ the constraint states that the in-flow and out-flow are the same, that is, if no VNF is traversed, the flow remains unchanged. The same result is obtained for all VNF f such that $\mu_f=1$ (no compression/decompression). To the aim of linearizing this constraint, we introduced variable ψ_{ik}^{fn} (still non-linear representation):

$$\psi_{ik}^{fn}=(\sum_{j\in N:(j,i)\in A}\phi_{ji}^{k})z_{ik}^{fn}$$

The constraints can be linearized using Equations (3.19)-(3.21), with the parameter M_i equal to $\sum_{(j,i)\in A}\gamma_{ji}$, which represents the maximum quantity of flow that can enter node i. If $(\mu_f-1)z_{ik}^{fn}=1$ then ψ_{ik}^{fn} representing the flow of demand k entering node i and passing through the instance n of the VNF f (constraint (3.19)-(3.20)), otherwise it is zero (constraint (3.21)). It is now possible to present the new flow balance constraints for access nodes that must be added to the basic VNF-PR model:

$$\sum_{j\in N:(i,j)\in A}\phi_{ij}^{k}-\sum_{j\in N:(j,i)\in A}\phi_{ji}^{k}=$$
$$=\begin{cases} d^k & \text{if} \quad i=o_k \\ 0 & \text{otherwise} \\ -d^k\prod\limits_{f\in F:\mathrm{m}_k^f=1}\mu_f & \text{if} \quad i=t_k \end{cases} \quad \forall k\in D, i\in N_a \tag{3.15}$$

Flow and compression/decompression balance for NFVI nodes and for each demand:

$$\sum_{j\in N:(i,j)\in A}\phi_{ij}^{k}-\sum_{j\in N:(j,i)\in A}\phi_{ji}^{k}=$$
$$\sum_{f\in F,n\in 1..c_i^f}(\mu_f-1)\psi_{ik}^{fn} \qquad \forall k\in D, i\in N_v \tag{3.16}$$

Coherence between path and flow variables:

$$\phi_{ij}^{k}\leq d_k^{max}x_{ij}^{k} \quad \forall k\in D,(i,j)\in A \tag{3.17}$$
$$\phi_{ij}^{k}\geq d_k^{min}x_{ij}^{k} \quad \forall k\in D,(i,j)\in A \tag{3.18}$$

VNF compression/decompression linearization constraints:

$$\psi_{ik}^{fn} \leq \sum_{j \in N:(j,i) \in A} \phi_{ji}^{k} + M_i(1 - z_{ik}^{fn})$$

$$\forall k \in D, i \in N_v, f \in F, n \in 1..c_i^f \tag{3.19}$$

$$\psi_{ik}^{fn} \geq \sum_{j \in N:(j,i) \in A} \phi_{ji}^{k} + M_i(1 - z_{ik}^{fn})$$

$$\forall k \in D, i \in N_v, f \in F, n \in 1..c_i^f \tag{3.20}$$

$$\psi_{ik}^{fn} \leq M_i z_{ik}^{fn}$$

$$\forall k \in D, i \in N_v, f \in F, n \in 1..c_i^f \tag{3.21}$$

One compression/decompression VNF per node and demand:

$$\sum_{f \in F} \sum_{n \in 1..c_i^f : \mu_f \neq 1} z_{ik}^{fn} \leq 1 \quad \forall k \in D, \forall i \in N_v \tag{3.22}$$

Eq. (3.15) represents the flow balance for the access nodes. At destination node the quantity of flows is set equal to the demand multiplied for all factors of compression of all the demanded VNFs. Eq. (3.16) represents the flow balance for a given node that has the possibility of hosting VNFs (NFVI). Eq. (3.17)-(3.18) allow to connect variables x and ϕ, in such a way that only and only if arc (i, j) is used by demand k, that is $x_{ij}^k = 1$, then variable ϕ can be different from zero. As the demand passes through VNF that can compress or decompress the flow, then we can determine upper and lower bound for the demand that are: $d_k^{max} = d_k \prod_{f \in F:\mu_f \geq 1}$ and $d_k^{min} = d_k \prod_{f \in F:\mu_f \leq 1}$ (to avoid these parameters being zero when it does not exist any VNF with $\mu_f \geq 1$ (only decompression), and $\mu_f \leq 1$, respectively, the calculation of the parameter can be modified in $d_k^{max} = d_k \max\{1, \prod_{f \in F:\mu_f \geq 1}\}$ and $d_k^{min} = d_k \min\{1, \max\{0, \prod_{f \in F:\mu_f \geq 1}\}\}$, respectively). Variables x are still necessary to impose the isolated cycles elimination and the order in the VNF chain. The utilization rate constraints must be modified as follows:

$$\sum_{k \in D} \phi_{ij}^k \leq U u_{ij} \quad \forall (i, j) \in A \tag{3.23}$$

To take into account the combined effect of compression/decompression and VNF latency some modification are needed.

For the *standard* model, constraints (3.12) are modified as follows:

$$l_{ik}^f \geq g_j^f \left(\sum_{b \in D} \psi_{ib}^{fn}\right) - L(1 - z_{ik}^{fn})$$

$$\forall k \in D, i \in N_v, f \in F, n \in 1..c_i^f, j \in 1..n_g \tag{3.24}$$

For the *fastpath* model, constraints (3.14) are modified as follows:

$$\sum_{k \in D} \psi_{ik}^{fn} \leq B_{max}^f \quad \forall i \in N_v, f \in F, n \in 1..c_i^f \tag{3.25}$$

	Constraints		
Features	routing/location	latency profile	bit-rate
basic	(3.3), (3.4), (3.5)-(3.10)		
basic-lat	(3.3), (3.4), (3.5)-(3.10)	(3.11), [(3.12) vs (3.13)-(3.14)]	
basic-lat-cd	(3.3), (3.23), (3.5)-(3.10)	(3.11), [(3.12) vs (3.13)-(3.14)]	(3.15)-(3.22)

Table 3.5 – Applicable constraints to VNF-PR problem variations.

In Table 3.5 we summarize the different models. In the first column a short name is used to refer to each model, in the second column constraints necessary to model routing, location and resource capacity are reported. In the third and forth columns we report latency and compression/decompression constraints.

3.2.4 Multi-objective Math-heuristic Resolution

We face a multi-objective problem: minimizing the maximum link utilization, which reflects the service provider oriented vision to improve the user quality of experience (strictly related to link congestion, especially for real-time services) and minimizing the virtualization infrastructure cost evaluated as the number of required CPUs at the NFVI level, which reflects the aims of the NFVI provider. Such a multi-objective approach makes especially sense when the NFVI provider is a different entity than the ISP. These two objectives are in competition; in fact, to obtain a low utilization, a large number of VNFs must be allocated.

We decided to prioritize the objectives: first we minimize the maximal link utilization (U), and then the NFV cost (total number of used CPU). We refer to this as the TE-NFV objective. In practice, we perform a first optimization step to find the best solution accordingly to maximal link utilization ($U^\star$), and then, keeping the best value found in the first step as a parameter (i.e. adding the constraint $U \leq U^\star$), we minimize the second objective (NFV cost). In fact, for a given optimal value of the first step, different possible configurations are available to the second step, and a large primary cost reduction can be achieved by this second step without losing with respect to the primary objective (maximum link utilization).

In order to understand the impact of imposing a maximal link utilization constraint on the NFV cost, we decided to study the sensitivity of the second step of optimization on the optimal value $U^\star$. Therefore, we re-optimize the second objective relaxing the constraint on the maximum link utilization by a parameter α, i.e. we used constraint $U \leq \alpha + U^\star$ instead of $U \leq U^\star$. We increase α step by step, until the value of the NFV cost does not reduce anymore. This value corresponds to first minimize the NFV cost and then the maximum link utilization cost (NFV-TE).

From preliminary tests, we observed that optimizing the complete model is very expensive, and that computational time can be significantly reduced performing a sequence of optimization starting from a basic model to the complete one. The result of each step is used as a starting point for the following one, a so-called *warm-start*, that allows to reduce computational time and/or produce better solutions or gaps (when optimization is stopped before reaching the optimal solution). To be more precise, the sequence of models we optimize is first the basic one (only demand routing, VNF location and capacities are considered), then basic-lat (latency is added) and finally basic-lat-cd (latency and compression/decompression are added), see Table 3.5 for the complete description of the models and equations involved.

The most challenging model from an optimization point of view is the last one, basic-lat-cd. For this reason, we need to provide a feasible starting solution (warm start) for this step. To this aim, the previous step, optimizing basic-lat model, must be done with some slight modification: the compression/decompression feature changes the quantity of flow that traverses the graph, therefore to guarantee that the solution of the second step is feasible for the last one, it is necessary to route a worst case quantity of flow, given by the case that all the VNFs with decompression are already applied to the demand flow (of course, this worst case can be improved considering the order of VNFs, when it is known in advance).

The NFV objective function results to be computationally more challenging than the TE one. Therefore, for obtaining the optimal solution of the NFV goal, a bisection procedure is used on the number of allocated CPUs to guarantee solution optimality, even when in a single step the solver is not able to guarantee it: that is, at each bisection step, if a feasible solution is found, the interval of the possible number of CPUs is updated by replacing the greater end-point with the found solution, and if no feasible solution exists then it is updated by replacing the smaller end-point with the current number of CPUs. From numerical experiments, we observed that, for our problem, determining that an instance is unfeasible (fixing a maximum number of CPUs) using the TE objective is computationally less challenging than solving to optimality the model using the NFV objective function. For this reason, the speed up obtained using the bisection procedure allows us to solve to optimality, or obtain a solution with a small gap on a larger number of CPUs than solving directly the NFV objective problem.

3.3 Computational Results

Computational results are divided in two main parts: first, in Section 3.3.1, we show results using our PR algorithm under different choices of demand distribution and different VNF forwarding latency models and values. Then, in Section 3.3.2, we present comparison results between our PR algorithm and a VNE based algorithm.

3.3.1 Results of PR Algorithm under Different Cases of Demand Distribution and Different VNF Forwarding Latency Profiles

In this section, we report experiments on our PR model and algorithm under different scenarios. We first present the parameter setting and then the analysis of the results.

3.3.1.1 Test Settings

We adopt the three-tier topology represented in Fig. 3.3 for computational evaluation. Each edge node is connected to two aggregation nodes, each aggregation node is connected to two core nodes, and core nodes are fully meshed. We believe that this topology gives a good abstraction to represent the current vision on NFV deployment strategies: mobile edge facilities are represented by edge nodes, point-of-presence by aggregation nodes and data-centers by core nodes. Furthermore, the highly symmetric graph topology produces two main effects: it allows to better analyze the VNF distribution and the effects of latency limits and, on the other hand, produce a very challenging instance for the optimization phase, allowing to test the model in a stressful condition. As for the VNFs, we consider three VNF template types per traffic demand: one with a compression behavior, one with a decompression behavior and a third with no compression/decompression. For the sake of illustration, we name each of these templates with a realistic VNF type name: a 'Firewall' VNF for the compression (as it blocks part of the incoming traffic/packets), a 'Deep Packet Inspection (DPI)' VNF for the case with no compression and no decompression, and a 'Tunneling ingress VNF' for the decompression (as headers are added to packet in the entry end-point). For the latter VNF, the assumption we do is that the egress tunneling is done elsewhere in the Internet or in the access border side. We considered a strict order for the VNFs chain: Firewall VNF first, then DPI VNF, and finally Tunneling ingress VNF. Each VNF instance has to reserve its own VM, and we consider one single VM template requiring 1 CPU and 16 GB of RAM.

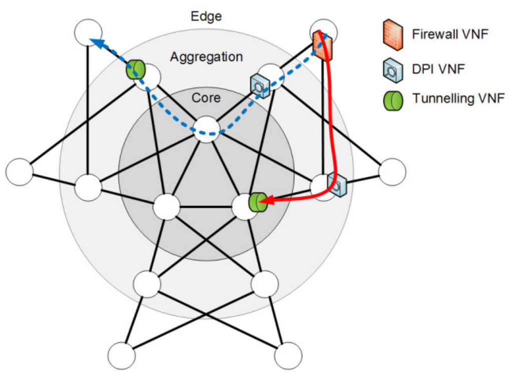


Figure 3.3 – Adopted network topology and PR solution example.

Table 3.6 presents the evaluation settings of the network resources. NFVI nodes are dimensioned with an increasing capacity from edge to core: 3 CPUs and 40 GB RAM at each edge node, 5 CPUs and 80 GB RAM at each aggregation node and 10 CPUs and 160 GB RAM at core nodes. The physical links are also dimensioned with different capacity to represent realistic settings: the aggregation links are dimensioned so that there is a risk of link saturation if the traffic distribution is not optimized (i.e. link utilization may higher than 100% when there are 3 or more demands passing by. The limit of number of demands also depend on the case of demand distribution, see Table 3.7 for detailed traffic settings), while the core links are dimensioned such that there is a very low bottleneck risk. Link latencies are set as follows to cope for the different geographical scopes: $1ms$ for edge links, $3ms$ for aggregation links, and $5ms$ for core links.

NFVI node settings		
NFVI node	CPU unit	RAM (GB)
Edge nodes	3	40
Aggregation nodes	5	80
core nodes nodes	10	160
Physical link settings		
Physical link	Bandwidth	Latency (ms)
Edge links	1	1
Aggregation links	0.5	3
core nodes links	1	5

Table 3.6 – Test settings of network resources.

We run our tests using two different case-studies for the demand distribution: Internet and Virtual Private Network (VPN). In the Internet case-study (e.g., the flow in red on Fig. 3.3), the traffic demands are sent by each edge node (e.g., end user) to each core node (e.g., data center), and from each core node to reach each edge node, which means that in this case both edge nodes and core nodes are access nodes (i.e., where demands are generated); while under VPN case-study (e.g., the flow in blue on Fig. 3.3), the edge nodes send traffic requests to each other, which means that the set of edge nodes corresponds to the set of access nodes. The total number of traffic demands is different for the two case-studies (36 for Internet and 30 for VPN), but we kept constant the average total traffic volume (sum of demands) in the network for the sake of comparison. As shown in Table 3.7, the demand quantity is randomly generated with uniform distribution of the required bandwidth volume in a given interval $[a, b]$, in such a way that edge demands cannot create any bottleneck at the edge links, i.e., $a = 0.11$ and $b = 0.14$ in the Internet case-study, $a = 0.13$ and $b = 0.17$ in the VPN case-study. These values allow us to keep the total traffic volume at the same level for the two case-studies. In order to have more significant results than one single demand instance, we generate 10 random demand instances (the practical averaged total traffic volume for both case-studies is around 4.3) for each case-study.

Study-case	$\|D\|$	a	b	Averaged total amount of traffic volume
Internet	36	0.11	0.14	$\|D\| * \frac{a+b}{2} = 4.5$
VPN	30	0.13	0.17	$\|D\| * \frac{a+b}{2} = 4.5$

Table 3.7 – Test settings of traffic demands.

We run tests for both Internet and VPN case-studies under standard as well as fastpath latency profiles, VNF processing latencies being set as in Fig. 3.1. The two profiles differ for their behaviors with respect to the bandwidth: fastpath has a fixed latency and it rejects demands beyond a given threshold, while the standard profile can accept a larger amount of demands at the cost of a larger latency. For both profiles, the same end-to-end latency is assigned to the bandwidth corresponding to the fastpath threshold. This allows to better compare the behavior of the two profiles. With fastpath we expect to aggregate demands up to the threshold (in the limits of the granularity of the demands and the routing possibilities), whereas with the standard profile we expect that a larger latency can be accepted (as long as we stay in the overall latency demand threshold) to reduce the number of VNF instances. In addition, we consider two levels of end-to-end latency bound (L): the strict and loose values ($15ms$ and $20ms$, resp). To be precise, we run tests with both strict and loose latency bounds for the following combinations of scenarios:

- TE goal and TE-NFV goal both with:
  - Internet demand profile:
    - standard forwarding regime
    - fastpath forwarding regime
  - VPN demand profile
    - standard forwarding regime
    - fastpath forwarding regime

For a total of 16 different cases (each of them repeated for 10 random generated demand instances).

The model is implemented and solved using AMPL [58] and CPLEX [59] 12.6.3.0. The execution time is limited to 600s for each basic TE optimization phase (i.e., model *basic* and *basic-lat*) and 800s for the complete TE phase (namely model *basic-lat-cd*), as well as for each step of the dichotomic search of the NFV optimization phase.

3.3.1.2 General Observations

In this part, we introduce general considerations from a computational point of view, discussing the quality of the results in terms of optimality and gap of the solutions. In Section 3.3.1.3 to 3.3.1.6, we provide detailed analysis of the structure and properties of the solution in terms of network and system indicators.

Table 3.8 presents the average, minimum and maximum results of the complete TE objective stage (using a time limit of 800s) for Internet and VPN case-studies considering the 10 random instances. We observe that for both Internet and VPN cases, the results of the complete TE objective stage of all the 10 test instances are stable: around 0.45 in Internet case and 0.60 in VPN case. The obtained results have an average optimality gap of 15%. In some instances the optimality of the solution is certified (i.e., optimality gap is 0%) and in the worst case the optimality gap is 25%. These optimality gaps may appear large, but we need to observe that the optimality gap is only an upper bound to the distance from the optimal value, and in some cases, even if the current solution shows large gap, it can be the optimal one (this can be due to a poor continuous relaxation, problem quite common when binary variables are present). Therefore, to further investigate this aspect, we have

Standard	Internet		VPN	
	$L = 15ms$	$L = 20ms$	$L = 15ms$	$L = 20ms$
Average	0.45	0.45	0.61	0.59
Maximum	0.49	0.49	0.63	0.61
Minimum	0.41	0.41	0.59	0.58

Fastpath	Internet		VPN	
	$L = 15ms$	$L = 20ms$	$L = 15ms$	$L = 20ms$
Average	0.44	0.45	0.60	0.59
Maximum	0.49	0.49	0.62	0.62
Minimum	0.41	0.41	0.58	0.57

Table 3.8 – Averaged, minimum and maximum values of the TE objective for Internet and VPN case-studies.

performed some additional tests with a longer time limit (2 hours), and we could certify that the solutions found with a smaller (800s) time limit were optimal.

As for the NFV objective stage, it is hard to reach the optimum within the time limit of 800s. Moreover, we observe that the results depend on the case-study: with the VPN case, under both standard and fastpath latency profiles, we obtain a lower optimality gap and a smaller variation of it than with the Internet case. A possible explanation is the increased number of traffic demands with the Internet case-study, which seems to significantly impact on the computational effort.

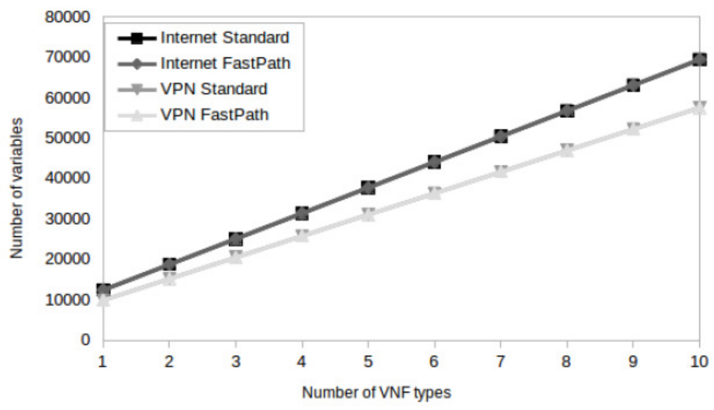
Fig. 3.4 illustrates the problem size with regard to the number of VNF types of both VPN and Internet case-studies. As shown in Table 3.5, different model stages require different sets of variables and constraints, and the most expensive model stage in terms of number of constraints is the basic-lat-cd. In order to show how the proposed models scale with the number of VNFs, we report in Fig. 3.4 the dependency between the number of VNF types and the problem size (in terms of number of variables and number of constraints) of the most expensive model (basic-lat-cd model). Fig. 3.4 (a) shows that the number of variables increases linearly with the number of VNF types, and Fig. 3.4 (b) shows that the number of constraints increases nonlinearly but smoothly with the number of VNF types.

In the following, we present the analysis of the solutions behavior. We compare the two different demand case-studies (i.e. Internet and VPN) with two points of view: i) what happens when we consider the NFV cost in the objective function instead of TE (Section 3.3.1.3 and Section 3.3.1.4), and ii) what happens when we make stronger the bound on the end-to-end latency (Section 3.3.1.5). Then we also compare the behavior with respect to the latency profiles (Section 3.3.1.6).

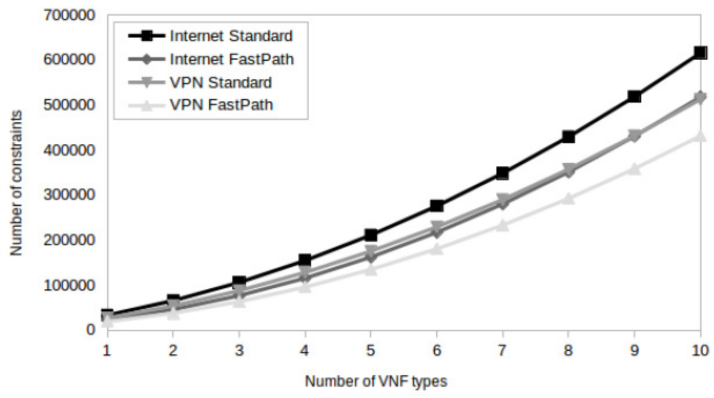
3.3.1.3 TE vs. TE-NFV Objective

We analyze the difference between the results with the TE objective and the results with the composite TE-NFV objective.

- NFVI cost (Fig. 3.5 and Table 3.9): Table 3.9 shows the averaged total NFVI cost (i.e., the number of used CPU) obtained using the TE objective and the



(a) Number of variables

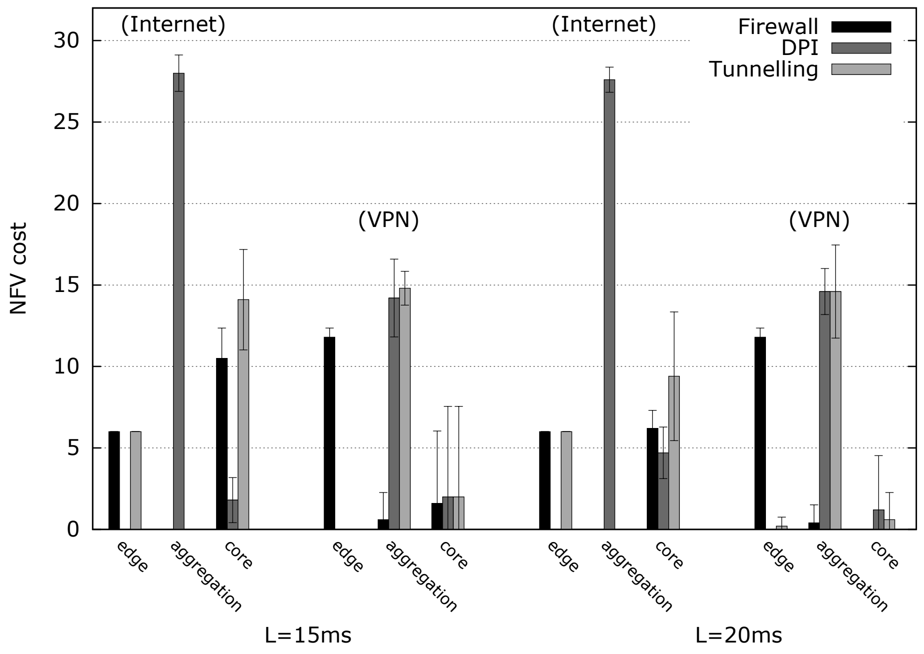


(b) Number of constraints

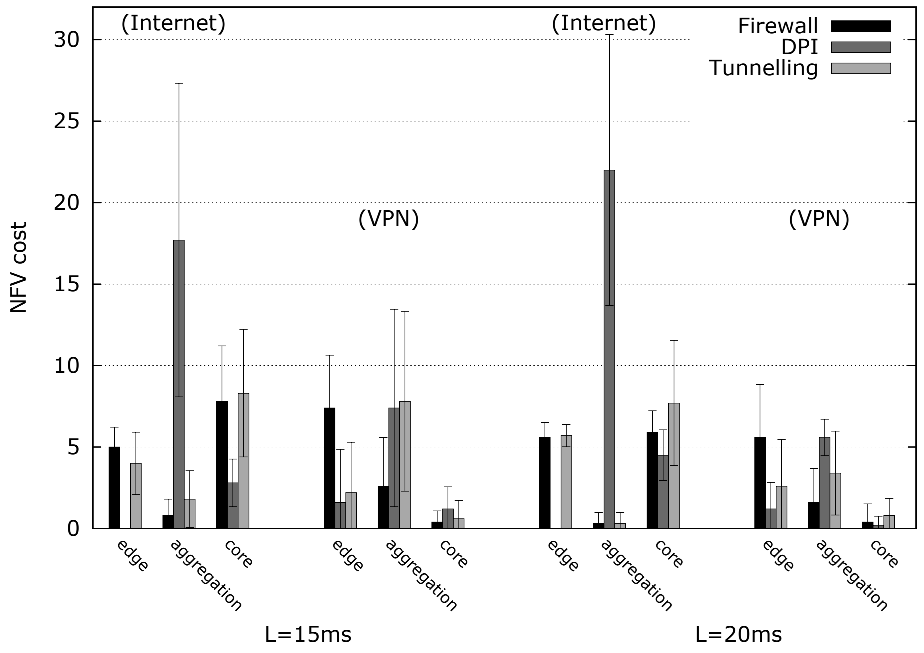
Figure 3.4 – Dependency between the number of VNF types and the problem size.

Standard	Internet			VPN		
	TE	TE-NFV	Reduction (%)	TE	TE-NFV	Reduction (%)
$L = 15ms$	66.4	48.2	26.86	46.1	31	31.08
$L = 20ms$	59.9	52	13.05	43.2	20.9	58.66

Table 3.9 – Averaged results of NFVI cost of the TE objective and the TE-NFV objective under Internet and VPN case-studies, with the reduction ratio of cost from TE to TE-NFV (standard case).



(a) TE objective.



(b) TE-NFV objective.

Figure 3.5 – VNF node distribution across NFVI levels (standard case).

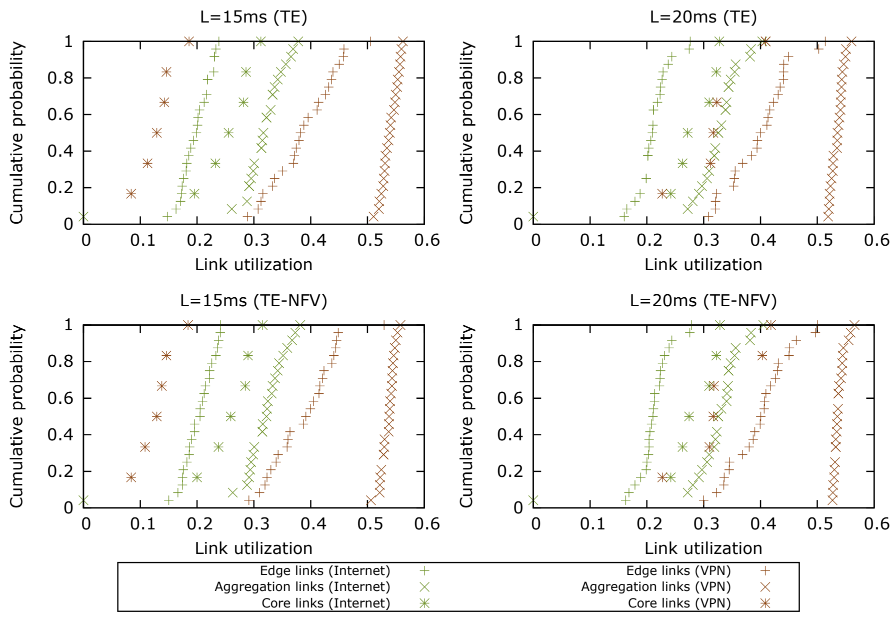


Figure 3.6 – Link utilization empirical CDFs (standard case).

TE-NFV objective for both Internet and VPN cases following the standard forwarding latency profile. In the fourth (respectively seventh) column the percentage reduction in the total NFVI cost is reported. As expected, the value is reduced, but it is worth to notice that the reduction is quite significant for both Internet and VPN traffic models. Moreover, the cost reduction with VPN is more significant than with Internet, especially with a loose bound on the end-to-end latency ($L = 20ms$), the total cost was reduced by 58.66% in average. As we discussed in Section 3.3.1.2, a possible explanation is the increased number of traffic demands in the Internet case, which leads to possibly lower quality solutions. The variation of solutions for Internet case can also be observed in Fig. 3.5, where the VNF node distribution (i.e., the number of used CPU by each VNF type across NFVI edge, aggregation and core levels) is illustrated for both Internet and VPN cases with a confidence interval of 95%. In plot (b) of Fig. 3.5 (with TE-NFV objective), the number of VNF instances varies greatly, whereas in plot (a) (with TE objective), the number of VNF instances varies gently. Although there is big variation in the solutions with the TE-NFV objective, and even if the solutions are not optimal, the averaged total NFVI cost is reduced greatly with a better deployment of VNF distribution. This result shows that considering the NFV cost in the TE objective can reduce significantly the resource consumption even without reaching the optimality.

- Link utilization (Fig. 3.6): the link utilization is not significantly affected by including the NFV cost minimization in the optimization goal. As illustrated in Fig. 3.6, in both cases with the TE goal, the aggregation links get most used. Furthermore, Internet case uses less the edge links, while VPN case uses less the core links. When taking into account the NFV cost in the objective, this behavior remains the same in both cases. This result shows that the TE-NFV goal can reduce the resource consumption without affecting the link utilization.

- VNF forwarding latency (Fig. 3.7): with both Internet and VPN demands, it increases passing from the TE goal to the TE-NFV one. This suggests that adopting the TE-NFV goal allows a higher level of VNF sharing for both latency bound situations.

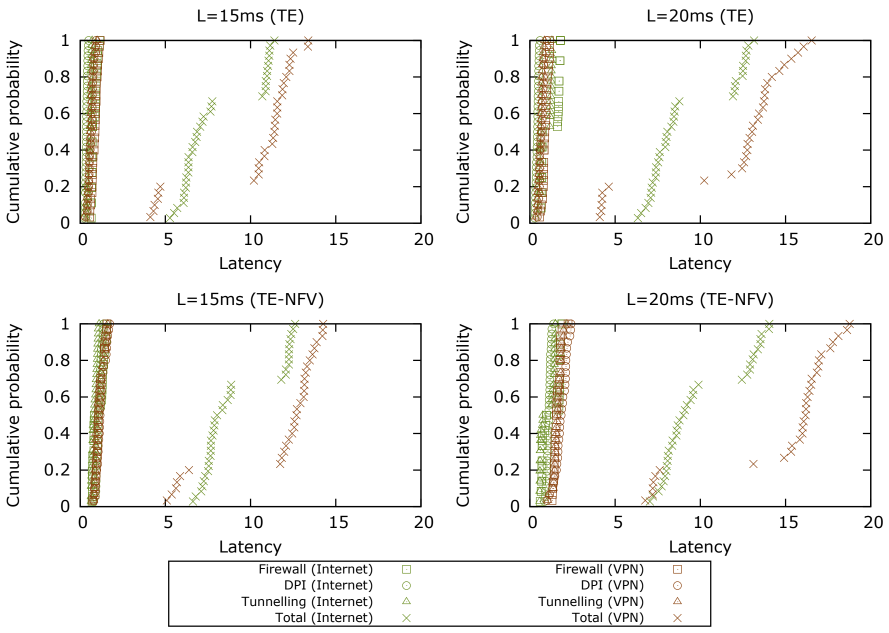


Figure 3.7 – Empirical CDFs of latency components (standard case).

3.3.1.4 Relaxing the TE Constraint - Sensitivity to Maximum Link Utilization

We perform a sensitivity analysis to put in evidence the effect of relaxing the TE objective with respect to the NFV optimal cost, with the goal to further put in evidence the trade-off between the two objectives.

With the TE-NFV objective, even if the VNF allocation cost is minimized, a minimum maximum link utilization is guaranteed. What we want to analyze is the impact of the TE bound on the NFV cost objective optimization. To this aim, starting from the TE optimal value, we perform a series of optimization steps of the NFV cost objective function, allowing this bound to be relaxed, increasingly.

Table 3.10 shows the NFV cost (average of 10 random instances) under different limit of maximal link utilization (U). We calculate the NFV cost under $U = U^{\star} + \alpha$,

Instance	$\alpha = 0$	$\alpha = 0.2$	$\alpha = 0.4$
	L=15ms		
Standard Internet	48.2	25.8	24.9
Fastpath Internet	37.125	31	31
Standard VPN	31	28.8	28.6
Fastpath VPN	39.1	37.7	37.7
	L=20ms		
Standard Internet	52	23.7	23.1
Fastpath Internet	38.7	34.8	31.2
Standard VPN	20.9	20.4	20
Fastpath VPN	34.9	34.8	33.8

Table 3.10 – NFV cost for different TE goal relaxation levels, with TE-NFV optimization.

with α varying from 0 to 0.4. For both Internet and VPN cases, when $\alpha = 0$, the TE bound (U) used for the TE-NFV phase is the $U^\star$ found in TE phase; when $\alpha = 0.4$, the U used for the TE-NFV phase is around 1 (i.e., link saturation reached). The results show that a loose TE bound (link utilization) allows a better TE-NFV solution. For most cases, there is almost no reduction (or no reduction at all) from $\alpha = 0.2$ to $\alpha = 0.4$, which suggests that there exists a ceiling between the TE objective and TE-NFV objective: we can get better utilization of NFV resources (i.e., TE-NFV objective) by allowing relaxed link utilization limit (i.e., TE objective), however this is not always true when we reach the ceiling (e.g., other limits like VM capacity also impact NFV cost). As for case FastPath Internet $L = 20ms$, there is a reduction of NFVI cost with α increasing from 0.2 to 0.4. We can observe that for the same case-study with $L = 15$ the best objective found is 31 (both for $\alpha = 0.2$ and 0.4), therefore we can attribute this change in behaviour to the non-optimality of the solution in the case $L = 20$, rather than to a different behaviour of the system (we remind the reader that the problem is computationally very challenging, and we imposed a short time limit).

3.3.1.5 Sensitivity to the Latency Bound

We analyze the impact of the VNF chain latency bound (L) on the results.

- NFVI cost (Fig. 3.5 and Table 3.9): as shown in Table 3.9, the averaged total cost is reduced with VPN demands under both optimization goals with a loose latency bound, especially with the TE-NFV goal. This happens because, with a loose latency bound, the traffic can pass by the links with high latency (e.g., core links) to share more VNFs. On the contrary, there is a small cost increase with Internet demands under TE-NFV goal. Analyzing in a more detailed way the results, we observe that for the Internet case-study, the solver (CPLEX) has more difficulties to reduce the gap. We can deduce that making the latency bound weaker makes the location problem component (locating VNF and the NFV goal) predominant with respect to the routing one, in fact, if the latency

Fastpath	Internet			VPN		
	TE	TE-NFV	Reduction (%)	TE	TE-NFV	Reduction (%)
$L = 15ms$	60.0	39.8	33.56	50.5	40	18.97
$L = 20ms$	59.9	38.7	37.83	47.5	35.9	21.50

Table 3.11 – Averaged results of NFVI cost of the TE objective and the TE-NFV objective under Internet and VPN case-studies, with the reduction ratio of cost from TE to TE-NFV (fastpath case).

bound is large enough the routing problem is not constrained anymore, and more routing solutions are available; this allows more freedom in the location part, and probably increases its combinatorial structure, and therefore it makes the solution of the problem computationally more challenging. This is also confirmed by a noticeable variability in the results, with a smaller averaged cost reduction as shown in Table 3.9 passing from strict latency (with a cost reduction of 26.86%) to loose latency bounds (with a cost reduction of 13.05%). Moreover, we can see that with VPN demands, there is a higher dependency to the latency bound than with Internet demands; this happens because in general it is farther to send traffic demands from edge node to edge node than from edge/core node to core/edge node, i.e., the end-to-end forwarding path of VPN demands is in general longer than that of Internet demands, which leads to a higher dependency to the latency bound.

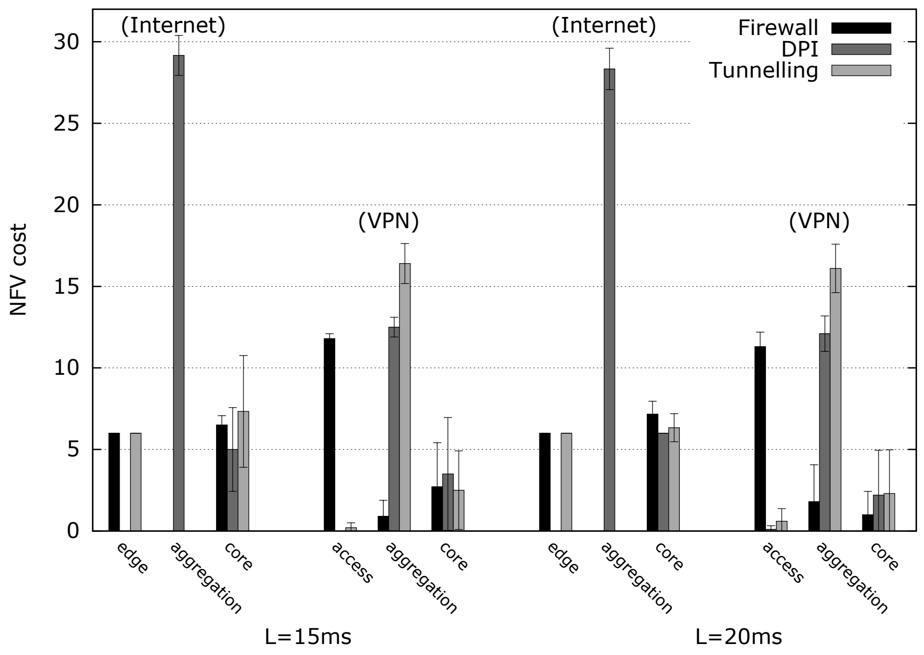
- Link utilization (Fig. 3.6): in support of the above-mentioned analysis, we can remark that under the loose latency bound, the core links get more utilized with VPN demand,; while with Internet demands, the link utilization remains almost the same. This indicates that VPN case is more sensible to the latency bound.
- VNF forwarding latency (Fig. 3.7 and Table 3.12): the same observation can be obtained by looking at end-to-end latency components. As shown in Table 3.12, the averaged total latency of VPN is always greater than Internet. Moreover, as shown in Fig. 3.7, the latency of each VNF and the total latency become longer with VPN demands with loose latency bound.

These observations confirm the importance of the bound for VNF chaining and placement decisions.

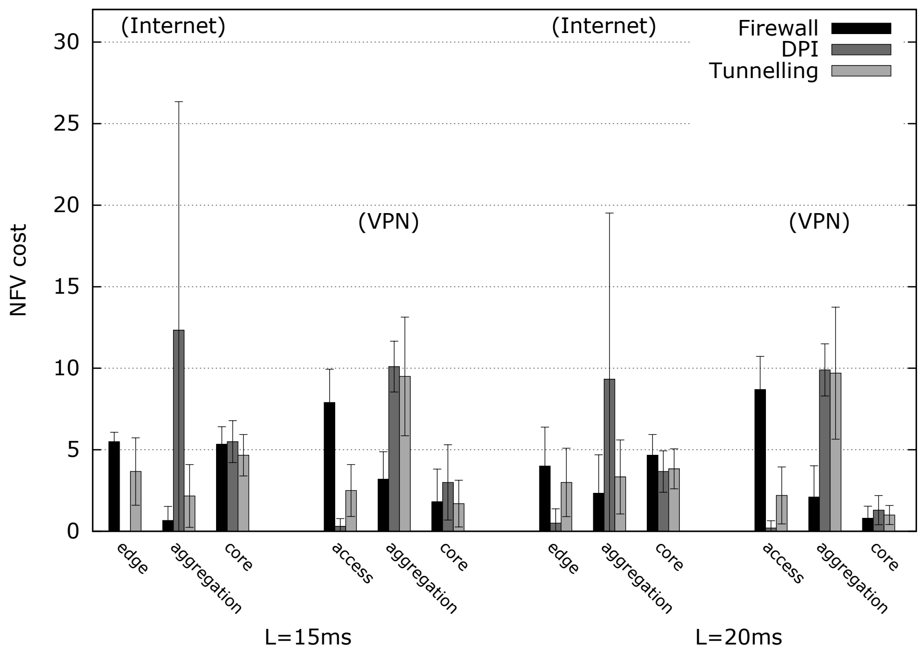
3.3.1.6 Standard vs. Fastpath VNF Switching

We now compare the results with the standard VNF forwarding latency profile to those with the fastpath profile.

- NFVI cost (Fig. 3.5 vs. Fig. 3.8, and Table 3.9 vs. Table 3.11): under the TE-NFV goal, the fastpath VNF forwarding is more expensive than the standard forwarding with VPN demands, especially with a loose bound on the end-to-end latency ($L = 20ms$), the averaged total NFVI cost is 35.9 under the fastpath profile and 20.9 under the standard profile; while it is the opposite with Internet demands, for example, with the loose latency bound of $20ms$, the averaged total NFVI cost of the TE-NFV goal is 52 under the standard profile, while it reduces to 38.7 under the fastpath profile. This happens because of the maximum traffic bound that is set under the fastpath case and that is not set for the standard case (which however brings to a higher end-to-end latency as confirmed in the last item hereafter), therefore, more VNF instances need to be deployed, which in turn increase the total cost.



(a) TE objective.



(b) TE-NFV objective.

Figure 3.8 – VNF node distribution across NFVI levels (fastpath case).

- Link utilization (Fig. 3.6 vs. Fig. 3.9) : the only difference between the link utilization under the standard and the fastpath profiles is remarkable for the case of VPN: the core links get less used with the strict latency bound of 15ms under the standard profile, while they are not used at all under the fastpath profile with the strict latency bound. However, there is no clear difference for both forwarding profiles when taking into account the NFV cost in the objective. Moreover, the behavior of the models is similar under the two forwarding profiles, when passing from strict latency limit to the loose one: under both the standard and the fastpath profiles, the core links get more utilized with VPN demands, while with Internet demands, the link utilization remains stable.

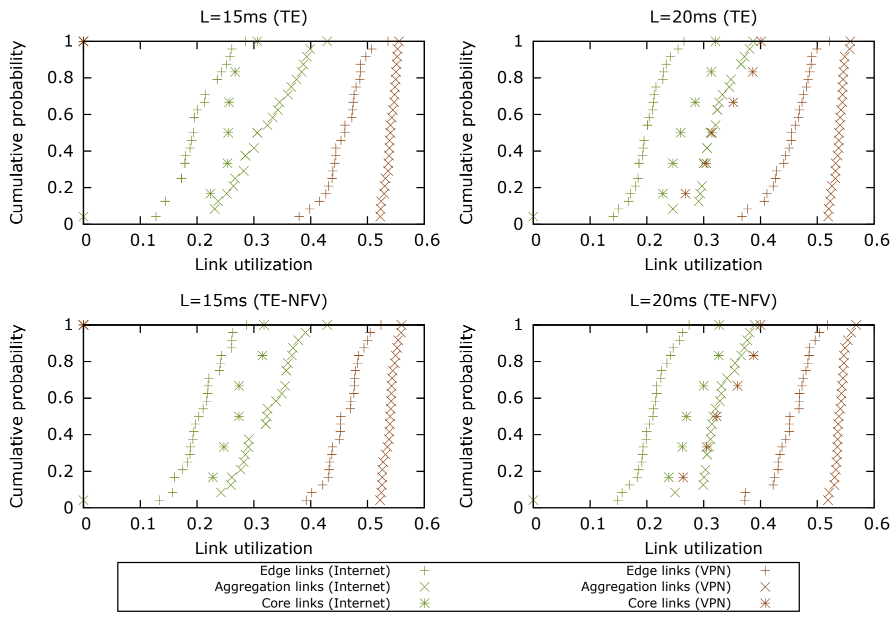


Figure 3.9 – Link utilization empirical CDFs (fastpath case).

- VNF forwarding latency (Fig. 3.7 vs. Fig. 3.10): as discussed previously, the VNF forwarding latency and the total latency under the standard profile increase passing from the TE goal to the TE-NFV goal, but this is not the case under the fastpath profile: the total end-to-end latency under the fastpath profile remains almost the same passing from the TE goal to the TE-NFV goal. For example, as shown in Table 3.12, the averaged total latency is the same for the TE and the TE-NFV goal in the case of VPN demands with both strict and loose latency bounds and also in the case of Internet demands with loose latency bound. This is because of the maximum traffic bound set under the fastpath case but not under the standard case. Therefore the standard model can allow a higher demand aggregation on a single VNF. More precisely, it can support a larger bandwidth than the fastpath model while introducing a higher VNF forwarding latency. As a result, VNFs are better shared under the standard case, and especially with the loose latency bound. This also can be observed by detailed results on latency components as shown in Table 3.12. For instance, under the standard profile with the TE-NFV goal, the averaged forwarding latency on each VNF instance of VPN is more than $1ms$ under both strict and loose latency bound. Furthermore, under the loose latency bound ($L = 20ms$), the averaged forwarding latency on each VNF instance is more than $1.5ms$. This result also confirms the results reported in Table 3.9: the total cost of TE-NFV goal has been reduced by 58.66% in average, which shows that VNFs are better shared with the TE-NFV goal and under the standard case.

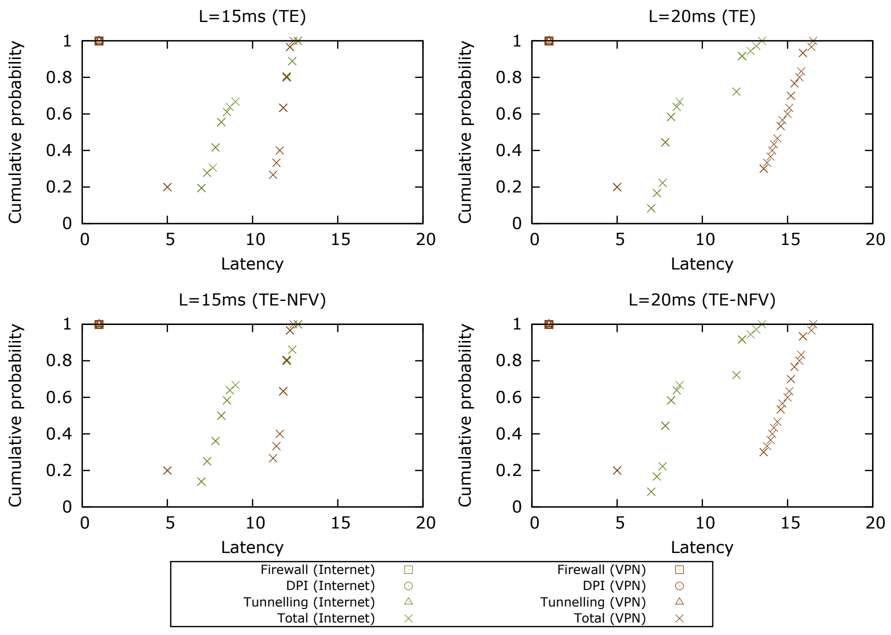


Figure 3.10 – Empirical CDFs of latency components (fastpath case).

To conclude, we discussed in this section the experimental results of our VNF-PR algorithm based on a large number of tests of different scenarios. The results show that the TE-NFV goal allows a significant higher level of VNF sharing with both the strict and the loose latency bounds. As a result, considering the NFV cost in the objective function reduces greatly the resource consumption (in terms of number of CPUs) without affecting the link utilization. Furthermore, by extensive tests of relaxing the TE objective with respect to the NFV cost, we find that there exists a ceiling between the TE objective and the TE-NFV objective: we can get better utilization of NFV resources by allowing relaxed link utilization limit, however this is not always true when we reach the ceiling. As for the TE-NFV goal with both the standard and the fastpath forwarding profiles, the total cost depends on the traffic model: the standard forwarding is more expensive than the fastpath with Internet demands, while it is the opposite with VPN demands. However, VNFs are better shared under the standard case for both Internet and VPN demands with the loose

Internet study-case under TE objective with L=15ms												
	Standard						FastPath					
	VNF F	VNF D	VNF T	ALL VNFs	Path	Total	VNF F	VNF D	VNF T	All VNFs	Path	Total
avg	0.82	0.38	0.57	1.76	6.27	8.03	1.00	1.00	1.00	3.00	6.25	9.25
min	0.65	0.24	0.36	1.27	4.00	5.27	1.00	1.00	1.00	3.00	4.00	7.00
max	1.15	0.50	0.70	2.15	9.60	11.40	1.00	1.00	1.00	3.00	9.67	12.67

Internet study-case under TE objective with L=20ms												
	Standard						FastPath					
	VNF F	VNF D	VNF T	ALL VNFs	Path	Total	VNF F	VNF D	VNF T	All VNFs	Path	Total
avg	1.21	0.44	0.92	2.56	6.68	9.24	1.00	1.00	1.00	3.00	6.37	9.37
min	0.66	0.29	0.59	2.17	4.00	6.34	1.00	1.00	1.00	3.00	4.00	7.00
max	1.79	0.59	1.29	2.98	10.80	13.16	1.00	1.00	1.00	3.00	10.50	13.50

Internet study-case under TE-NFV objective with L=15ms												
	Standard						FastPath					
	VNF F	VNF D	VNF T	ALL VNFs	Path	Total	VNF F	VNF D	VNF T	All VNFs	Path	Total
avg	1.07	1.09	0.84	3.01	6.27	9.28	1.00	1.00	1.00	3.00	6.34	9.34
min	0.70	0.79	0.64	2.53	4.00	6.62	1.00	1.00	1.00	3.00	4.00	7.00
max	1.55	1.38	1.11	3.46	9.60	12.62	1.00	1.00	1.00	3.00	9.67	12.67

Internet study-case under TE-NFV objective with L=20ms												
	Standard						FastPath					
	VNF F	VNF D	VNF T	ALL VNFs	Path	Total	VNF F	VNF D	VNF T	All VNFs	Path	Total
avg	1.26	1.11	1.01	3.38	6.68	10.06	1.00	1.00	1.00	3.00	6.37	9.37
min	0.60	0.86	0.57	2.99	4.00	7.04	1.00	1.00	1.00	3.00	4.00	7.00
max	1.87	1.39	1.50	3.75	10.80	14.05	1.00	1.00	1.00	3.00	10.50	13.50

VPN study-case under TE objective with L=15ms												
	Standard						FastPath					
	VNF F	VNF D	VNF T	ALL VNFs	Path	Total	VNF F	VNF D	VNF T	All VNFs	Path	Total
avg	0.86	0.64	0.61	2.12	8.07	10.18	1.00	1.00	1.00	3.00	7.48	10.48
min	0.56	0.32	0.22	1.45	2.00	4.12	1.00	1.00	1.00	3.00	2.00	5.00
max	1.18	0.93	1.00	2.71	11.67	13.39	1.00	1.00	1.00	3.00	9.40	12.40

VPN study-case under TE objective with L=20ms												
	Standard						FastPath					
	VNF F	VNF D	VNF T	ALL VNFs	Path	Total	VNF F	VNF D	VNF T	All VNFs	Path	Total
avg	0.90	0.68	0.61	2.19	9.58	11.77	1.00	1.00	1.00	3.00	9.95	12.95
min	0.57	0.43	0.33	1.67	2.00	4.10	1.00	1.00	1.00	3.00	2.00	5.00
max	1.15	0.99	0.85	2.86	14.17	16.53	1.00	1.00	1.00	3.00	13.50	16.50

VPN study-case under TE-NFV objective with L=15ms												
	Standard						FastPath					
	VNF F	VNF D	VNF T	ALL VNFs	Path	Total	VNF F	VNF D	VNF T	All VNFs	Path	Total
avg	1.16	1.21	1.11	3.48	8.07	11.55	1.00	1.00	1.00	3.00	7.48	10.48
min	0.74	0.80	0.80	2.55	2.00	5.09	1.00	1.00	1.00	3.00	2.00	5.00
max	1.58	1.74	1.63	4.59	11.67	14.26	1.00	1.00	1.00	3.00	9.40	12.40

VPN study-case under TE-NFV objective with L=20ms												
	Standard						FastPath					
	VNF F	VNF D	VNF T	ALL VNFs	Path	Total	VNF F	VNF D	VNF T	All VNFs	Path	Total
avg	1.59	1.84	1.62	5.06	9.58	14.63	1.00	1.00	1.00	3.00	9.95	12.95
min	1.30	1.01	1.06	4.29	2.00	6.76	1.00	1.00	1.00	3.00	2.00	5.00
max	2.24	2.41	2.17	5.68	14.17	18.76	1.00	1.00	1.00	3.00	13.50	16.50

Table 3.12 – Averaged, maximum and minimum latency components of both the standard and the fastpath cases (F represents 'Firewall', D is 'DPI' and T is 'Tunneling ingress').

latency bound.

3.3.2 PR vs. VNE Based Approaches

In this section, we compare our PR approach with the VNE Based ('VNE-B') approach, already discussed in Chapter 2, using the algorithm from [36], which is open sourced by the authors[4].

In [36], a VNE-B modeling approach is proposed for a generic VNF orchestration problem: each traffic demand is considered as a virtual graph (i.e., $G(N, L)$ in [36], where N is the set of traffic nodes, i.e., switches or VNFs, and L denotes the links between them) to be embedded in the substrate graph represented by switches/routers and NFVI nodes. The mapping of virtualized traffic demands' path onto a physical network is realized by embedding VNFs on physical servers and establishing path for virtual links. The objective considered is the minimization of the overall OpEx (operational expenditure) cost: VNF deployment cost, energy cost, traffic forwarding cost and an additional penalty to take into account Service Level Objective (SLO) violations. A weighted sum of the four aforementioned costs is considered as optimization objective. Authors proposed an ILP model, and they present two different problems, a static one - where demands are known in advance - and a dynamic one - where demands arrive in an online fashion. In our comparison, we focus on the static version of the problem and its proposed solution approach; it is based on a procedure that solves a sequence of ILPs, where, for each iteration, the number of VNFs is limited and the execution time is limited as well. ILP executions are solved with CPLEX (using the callable library).

For the sake of comparison, we adapt our original VNF-PR model to the hypothesis used in [36]; we list in the following the simplifications and adaptations to our model in order to use the same parameters used in the VNE-B approach:

- We reduced our objective to a single objective: minimizing the overall network operational cost, using the same parameters of the VNE-B approach.
- We considered only the *fastpath* latency regime, i.e., we fixed the VNF forwarding latency.
- We discarded compression/decompression aspects, i.e., we adopt our 'basic-lat' model.
- As the VNE-B approach uses VNF templates, we associated a VNE template to each VNF type according to VNF requested number of vCPUs; for example, a template with capacity of 4 CPUs is associated to the VNF requesting 4 CPUs.
- We added the penalty parameter for each traffic request to take into account SLO violations.

[4]The source code is at `https://github.com/srcvirus/middlebox-placement`.

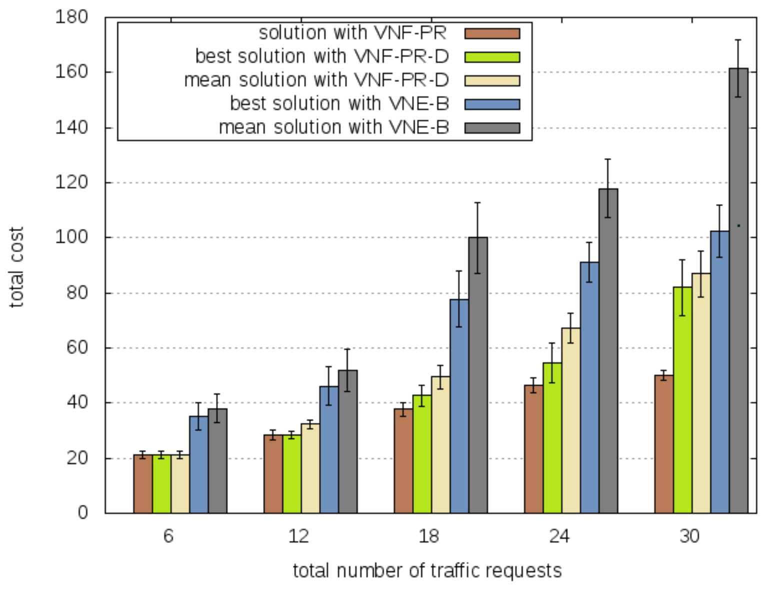


Figure 3.11 – Comparison between PR and VNE based algorithms in terms of objective function.

Furthermore, similarly to the NFV cost optimization phase of our PR approach described in Section 3.2.4, authors in [36] use a dichotomy method on the number of VNFs. In order to understand the impact of adding the dichotomy method and to have a fair comparison, we tested two solving strategies for our model:

- PR algorithm: the adapted PR model solved directly without the dichotomy procedure.
- PR-D algorithm: the adapted PR model solved with the dichotomy procedure. For the sake of comparison, we integrate our adapted PR model into the same procedure used in [36] (i.e., replacing their model by ours in the dichotomy procedure)

As for the comparison test, the network settings are taken directly from the simulation setup of [36]: adopting the Internet2 topology (12 switches and 15 links), setting the same physical link and NFVI server capacities, using the same VNF specification, VNF requests sequence, etc., and adopting the same cost data. As for the traffic data, we created 5 groups of tests with different sets of traffic requests (6, 12, 18, 24 and 30); for each group, we randomly selected, from the traffic matrix set of [36], 10 matrices. Then we tested these groups of data with the three methods (i.e., VNE-B, PR and PR-D). Our algorithms are implemented in AMPL, and the VNE-B algorithm is implemented in C++ using CPLEX for ILP resolution. CPLEX 12.5.1 was used for these tests. Fig. 3.11 reports a comparison between our PR solution and the VNE-B solution in terms of global cost, each group of bars represents results for different number of traffic requests. Note that PR is an exact method that finds always the optimal solution, while PR-D and VNE-B use the dichotomy process, so that they may provide sub-optimal solutions. Moreover, the dichotomy process provides a set of feasible solutions and not a single one. Therefore, Fig. 3.11 illustrates 5 bars representing the solutions of different algorithms. More precisely, we report the optimal solution for PR, and we report the best solution as well as the average value of the feasible solutions for the VNE-B algorithm and the PR-D algorithm. As a result, the absolute optimality gap for sub-optimal solutions is shown by the difference between the red bar (optimal solution) and each other bar. We observe that PR-D is able to find better solutions compared to the VNE-B algorithm, especially when the amount of traffic demands is small, where PR-D can always find optimal solutions; whereas the VNE-B algorithm is always able to find a feasible solution very fast (i.e., within 10 seconds vs. up to one hour for our method), the result can be of very low quality, in fact it does not provide optimal solution even for the test with 6 demands.

3.4 Further Model Refinements

The model we provided above can be possibly refined and customized to meet specific requirements. We list in the following the possible variants as well as the corresponding modeling variations.

- *VNF affinity and anti-affinity rules:* due to the privacy, reliability or other reasons, a provider may want impose rules on the placement of certain types of VNF: be placed or not placed on certain servers, be grouped or not grouped together, etc. Such specific VNF placement rules are called affinity and/or anti-affinity rules [60]. To extend our model to take them into account, the simplest way is to introduce a new variable representing the presence of a certain type of VNF f on a given node i (we remind the reader that our model allows to have multiple copies of the same type of VNF on the same node). Let us call this variable v_i^f, it will be equal to one if a VNF of type f is located on node i. To make these variables consistent with already defined variables y_i^{fn}, we need to add:

$$\sum_{n \in 1..c_i^f} y_i^{nf} \leq c_i^f v_i^f \quad \forall i \in N_v, f \in F$$

  More precisely, common affinity/anti-affinity rules are:

  - VNF-VNF affinity rules: if two VNFs communicate frequently and should share a host node, we may want to keep the VNFs together in order to reduce traffic across the networks and improve the traffic efficiency. Let $\text{AffVV}_{f_1 f_2}$ be a parameter equal to one if f_1 and f_2 should share the same node. Then:

$$v_i^{f_1} = v_i^{f_2} \quad \forall (f_1, f_2) : \text{AffVV}_{f_1 f_2} = 1$$

– VNF-Server affinity rules: certain intrusion prevention VNFs should reside in the network edges to guard against worms, viruses, denial-of-service (DoS) traffic and directed attacks. Let AffVS_i^f be a parameter equal to one if f should be installed on i. Then:

$$v_i^f = 1 \quad \forall (i,f) : \mathrm{AffVS}_i^f = 1$$

or restricted to a subset of nodes $S \in N_v$:

$$\sum_{i \in S} v_i^f = 1 \quad \forall (i,f) : \mathrm{AffVS}_i^f = 1$$

– VNF-VNF anti-affinity rules: it may be required to install multiple instances of the same VNF onto multiple servers in order to improve VNF reliability against failures. Let AAff_f be the anti-affinity parameter; we then impose that at least $nbMin$ nodes host the VNF:

$$\sum_{n \in N_s} v_i^f \geq nbMin \quad \forall f : \mathrm{AAff}_f = 1$$

if different VNFs cannot be co-located, let $\mathrm{AAffVV}_{f_1 f_2}$ be the anti-affinity parameter and impose:

$$v_i^{f_1} + v_i^{f_2} \leq 1 \quad \forall (f_1, f_2) : \mathrm{AAffVV}_{f_1 f_2} = 1$$

– VNF-Server anti-affinity rules: it may be required to avoid resource-hungry VNFs residing in certain cost-critical servers. Let AAffVS_i^f be the anti-affinity parameter and impose:

$$v_i^f = 0 \quad \forall (i,f) : \mathrm{AAffVS}_i^f = 1$$

We can observe that all the constraints that set some variables to one or zero, just reduce the number of variables; therefore we can expect that such constraints do not increase the computing time. A slightly different condition can be imposed for sharing a VNF among different demands; we refer to that as VNF isolation.

- *VNF isolation*: if the same VNF cannot be shared between two specific demands, we can add constraints to impose this condition. It is sufficient to introduce an incompatibility parameter $inc_{k_1 k_2}$, equal to one if demand k_1 must be isolated from demand k_2; then we need to add:

$$z_{ik_1}^{fn} + z_{ik_2}^{fn} \leq 1 \qquad \forall i \in N_v, f \in F, \\ n \in 1..c_i^f, k_1, k_2 \in D$$

- *Multiple comp./dec. VNFs per NFVI node*: to make the presentation simpler, we assumed that in each NFVI node there is at most one VNF that can compress/decompress a flow, i.e. with a factor of compression $\mu_f \neq 1$. This assumption can be relaxed using an extended graph in which each node that can host a VNF (N_v) is expanded in multiple copies, one for each type of VNF that can be allocated in the node. Otherwise, we can represent all possible combinations of different VNFs allocated to the same node, and adding additional binary variables to represent which combination is chosen.

- *VNF partial chain ordering*: we can observe that partial order can be imposed with the same form of constraints used for total ordering (3.10), just limiting their number to existing precedence conditions. It is sufficient to introduce a constraint for each couple of VNFs that has a precedence relation. More formally, for each demand k, we can introduce a directed acyclic graph $O_k(V_k, P_k)$, where nodes V represent the set of VNFs that must serve the demand ($V = \{i \in F : m_k^f = 1\}$), and arcs P represent the order relation between such VNFs, that is an arc $(i, j) \in P$ if VNF j must be used after VNF i. Then, constraints (3.10) can be rewritten as:

$$\pi_{jk} \geq \pi_{ik} - (|N_v| + 1)(2 - \sum_{n \in 1..c_i^{f_1}} z_{ik}^{f_1 n} - \sum_{n \in 1..c_i^{f_2}} z_{ik}^{f_2 n})$$
$$\forall k \in D, \forall i, j \in N_v, f_1, f_2 \in V_k : (f_1, f_2) \in P_k$$

- *Additional computing constraints*: it can be easily included by tuning existing parameters, as far as computing resource requests can be expressed in an additive way (e.g., for storage).

- *Load balancing*: in the current model, each demand can use a single VNF for each type. The model can be extended to allow per-VNF load balancing. If the load balancing is local to an NFVI node , the change in the model is small, in fact it is simply necessary to have some continuous variables taking into account the quantity of demand associated to each VNF. If the load balancing can be between different clusters, then it is necessary to extend the model allowing multiple paths for each demand. However such an extension is expected to largely increase the execution time.

- *Different VM templates*: for the sake of simplicity, differently from [1], we presented the model considering a one-to-one correspondence between VNF and VM templates (single template). Nevertheless, multiple VM templates can be considered in the model at the price of increasing of one dimension/index all variables indexed on the VNF identifiers.

- *Core router as a VNF*: if the core routing function is also virtualized, i.e., if the NFVI node and the network router can be considered as a single physical node that runs the core routing function, processing the aggregate traffic independently of the demand, as a VNF, then we need to add a term proportional to InFlow plus OutFlow to (3.5):

$$\begin{aligned}
&\sum_{k\in D}\sum_{f\in F}\sum_{n\in 1..c_i^f} rr_r y_i^{fn} \\
&+\sum_{k\in D}\sum_{j:(i,j)\in A} d_k x_{ij}^k \\
&+\sum_{k\in D}\sum_{j:(j,i)\in A} d_k x_{ji}^k \le \Gamma_{ir} \quad \forall i \in N_v, r \in R
\end{aligned}$$

If bit-rate compression/decompression is considered, constraint (3.5) must be modified as follows:

$$\begin{aligned}
&\sum_{k\in D}\sum_{f\in F}\sum_{n\in 1..c_i^f} rr_r y_i^{fn} \\
&+\sum_{k\in D}\sum_{j:(i,j)\in A} \phi_{ij}^k \\
&+\sum_{k\in D}\sum_{j:(j,i)\in A} \phi_{ji}^k \le \Gamma_{ir} \quad \forall i \in N_v, r \in R
\end{aligned}$$

3.5 Conclusion

In this chapter, we proposed a PR model and a math-heuristic for the VNF-PR problem. Our model takes into consideration realistic NFV features, such as specific VNF forwarding modes (standard and fastpath modes), VNF chain ordering constraints and flow bit-rate variations constraints, etc. We also studied how additional properties being discussed for NFV systems can be integrated in our proposed formulation. We ran extensive tests to evaluate our model on a three-tier topology representing an ISP topology. The computational results show that the combined TE-NFV objective can significantly reduce the number of VNFs in the network compared to the TE objective with almost no impact on the link utilization and on the latency. Moreover, we observed that, besides different optimization objectives (TE and TE-NFV), different distributions of traffic demand (Internet and VPN case-studies) and different VNF types (in terms of function on the bit-rate) could lead to different placements of VNF nodes and different VNF chaining paths. Then we further compared our PR approach to the classical VNE based approach often proposed in the literature for the VNF-PR problem. The experimental results show that PR is more stable and close-to-optimum than the VNE based solution, but the VNE based model is in general faster than PR. However, VNE based formulation is not as general as PR, e.g., VNE based formulation cannot solve the VNF-PR problem with partial or un-ordered chains. Therefore, further investigation on the problem properties and formulations are needed.

Chapter 4

The Theoretical Problem

Contents

4.1 Problem Description **70**

4.2 Problem Complexity **70**

4.3 Problem Properties **77**

4.4 Problem Formulations **79**

4.4.1 Relation between the Two Formulations 82

4.4.2 Additional Inequalities 93

4.4.3 Extended Split Path (SP) Model for Multiple Services . . . 94

4.5 Computational Results **97**

4.5.1 Test Instances 97

4.5.2 Continuous Relaxation Results on Small-medium Size Instances 98

4.5.3 Full model (ILP/MILP) Results on Small-medium Size Instances 102

4.5.4 Scalability on Large Size Instances 105

4.5.5 Impact of Proposition 4.3.1 106

4.5.6 Articulation Point Based Preprocessing 108

4.6 Conclusion **108**

In the previous chapter, we studied a realistic VNF-PR problem, such study suggests that further investigation on the problem formulations and properties are needed. The properties of a basic version of VNF-PR problem are investigated in this chapter with the goal of deriving insights that help in the problem solution. To so do, we focus on a simplified, yet significant variant: a single type of VNF is considered and any node can be equipped with a single capacitated VNF instance. Links are capacitated and the objective is to minimize the number of installed VNF instances. Each demand must be served by an instance of the VNF and routed on one simple path. The problem is denoted as Virtual Network Function Placement and Routing with Simple Path routing (VNF-PR$_{\text{SP}}$).

The remaining of this chapter is organized as follows. In Section 4.1 we report the full problem statement of VNF-PR$_{\text{SP}}$. In Section 4.2 and Section 4.3 we investigate the problem complexity and properties. In Section 4.4 we introduce two formulations and describe their properties and relation. In Section 4.5 we present and analyze a computational comparison of the two formulations for the VNF-PR$_{\text{SP}}$ problem.

4.1 Problem Description

For the sake of clarity and to allow reading this chapter quite independent from the previous one, we report here the full problem statement.

In the VNF-PR$_{\text{SP}}$ problem, the network is represented by a graph $G(N, A)$, where N represents the set of nodes and A represents the set of capacitated arcs (or links). Let u denote the arc capacity. An instance of the VNF can be installed on each node in N and can serve a limited amount of demand q. The network demands are represented by the set D: each demand $k \in D$ is characterized by a source (origin) node o_k a destination (terminal) node t_k, and a demand amount d_k. Each demand must be served (pass through) an instance of the VNF (service), but a demand can pass through a node without using a VNF installed on it. Demands cannot be split and must be routed on simple paths, i.e. the demand is not allowed to deviate from its path to "search" the VNF. The optimization goal is to minimize the number of installed VNF instances.

4.2 Problem Complexity

In this section, we study the problem complexity. We consider a version of the problem where only a subset of nodes, N_F, can host a service instance. The problem is NP-complete even with only one type of capacity (either link or service capacity): in [61] the problem is proved to be difficult if nodes are capacitated (the same approach can be used for the service capacity case); in our work of this thesis, we extend the result[5] for the case where only link capacity is considered, for the uncapacitated case and for some particular graph structures.

First we prove that even the uncapacitated version of VNF-PR$_{\text{SP}}$ (UVNF-PR$_{\text{SP}}$), namely without link and service capacity, is $\mathcal{NP}$-complete. Then, since its feasibility version is polynomial, we investigate whether adding one capacity at a time (either link capacity or service capacity) makes the feasibility problem difficult. Finally, we investigate the complexity on particular graph topologies: namely, full mesh, ring and tree-like.

Theorem 4.2.1. The UVNF-PR$_{\text{SP}}$ is $\mathcal{NP}$-complete.

[5]The main results can be found in publication [39]. In this article, the complexity is studied for the case where N_F is a proper subset of N. But for some study-cases, results can extend to the case of $N_F = N$. Therefore, we further extend our proofs in this manuscript with remarks for case of $N_F = N$.

Proof. We prove it by reduction from the Set Covering (SC) problem: given a set of elements $\mathcal{U} = \{1, 2, ..., n\}$, a collection of subsets of $\mathcal{U}$, $\mathcal{S} = S_1, \ldots, S_m$, and a threshold κ, SC asks for finding a subset of $\mathcal{S}$ that covers all the elements in $\mathcal{U}$ whose cardinality is smaller or equal to κ. Every instance of the SC can be reduced to an instance of the UVNF-PR$_{\text{SP}}$ as follows:

- the graph $G(N, A)$ is built as a tripartite graph:
  - $N = \{U_1, S, U_2\}$. The set U_1 (U_2, respectively) contains a node j_1 (j_2, respectively) for each element $j \in \mathcal{U}$. The set S contains a node for each subset S_i of $\mathcal{S}$;
  - two arcs (j_1, S_i) and (S_i, j_2) exist if $j \in \mathcal{U}$ can be covered by the subset S_i;
- only the nodes of the set S can host a service, i.e. $N_F = S$;
- for each element $j \in \mathcal{U}$ a demand is generated with source node $j_1 \in U_1$ and destination node $j_2 \in U_2$.

If there exists a solution of the UVNF-PR$_{\text{SP}}$ problem that uses at most κ service instances, then there exists a solution of the SC that uses at most κ subsets: if a demand is served by a service, the corresponding element is covered by the corresponding set, which is selected. □

Remark 4.2.1. The result of Theorem 4.2.1 holds also for the case where all the nodes can host a service ($N_F = N$). Let us keep the same graph structure and consider a demand d. Either the demand d is served by a node representing a covering set (a node in $\mathcal{S}$) or it is served by its origin (or destination). In the former case, the corresponding set is selected. In the latter, the service can serve only the demand d itself. If the instance is feasible, there must be at least one set that covers d. However, none of the sets that cover the demand d have been selected in the solution (otherwise, the origin or destination service would not have been needed). We can obtain a solution of value κ by replacing the origin (or destination) service with one of the sets covering the demand d.

Instead, the feasibility version of the UVNF-PR$_{\text{SP}}$ is polynomially solvable.

Theorem 4.2.2. The feasibility version of the UVNF-PR$_{\text{SP}}$ is polynomial.

Proof. As service instances are uncapacitated, each node in N_F can serve all the demands simultaneously. As arcs are uncapacitated, each demand can be routed independently of the others. Thus, we consider each demand and each node in N_F and we check if there exists a simple path for the demand which passes through the considered node (it can be done solving a suitable max flow problem with lower bound on arcs to represent the need of passing through a given node service). If it is so, then the demand is served, otherwise the instance is unfeasible. □

Remark 4.2.2. The proof of Theorem 4.2.2 works even if $N_F = N$.

We now analyze the impact of adding one type of capacity on the complexity of the feasibility version. It turns out that both the problem with only Service capacity (Sc-VNF-PR$_{\mathrm{SP}}$) and the problem with only Link capacity (Lc-VNF-PR$_{\mathrm{SP}}$) are $\mathcal{NP}$-complete, even in their feasibility version.

Theorem 4.2.3. The feasibility version of Sc-VNF-PR$_{\mathrm{SP}}$ is $\mathcal{NP}$-complete.

Proof. We prove it by reduction from the Bin Packing Problem (BPP):

Consider a set of items I, each with an integer weight, and a set of bins of capacity C. Given a threshold κ, there exists an assignment of all the items to the bins, such that at most κ bins are used?

Every instance of the BPP can be reduced to an instance of the Sc-VNF-PR$_{\mathrm{SP}}$ as follows:

- the graph $G(N, A)$ is built as a tripartite graph:
  - $N = \{I_1, N_F, I_2\}$. The set I_1 (I_2 respectively) contains a node i_1 (i_2, respectively) for each item $i \in I$. N_F has cardinality κ;
  - an arc connects each node in I_1 to each node in N_F and each node in N_F to each node in I_2;
- for each item $i \in I$ a demand is generated with origin i_1, destination i_2 and amount of demand equal to the item weight;
- the service capacity is equal to the bin capacity C.

If there exists a feasible solution of the instance of Sc-VNF-PR$_{\mathrm{SP}}$ then there exists a solution of the BPP that uses at most κ bins and vice versa. Indeed, routing the demand $i_1 - i_2$ on path i_1-b-i_2 and therefore assigning the demand to the VNF $b \in N_F$ corresponds to assigning the item i to the bin b. □

Theorem 4.2.4. The feasibility version of Lc-VNF-PR$_{\mathrm{SP}}$ is $\mathcal{NP}$-complete.

Proof. We prove it by reduction from the feasibility version of the Unsplittable Multicommodity Flow Problem (UMFP): given a graph $G = (N, A)$ with a capacity for each arc and a set of unsplittable commodities, each with a source, a destination and an amount of flow, the problem asks for finding one path for each commodity such that the overall flow on each arc does not exceed the arc capacity. Every instance of the UMFP can be reduced to an instance of the Lc-VNF-PR$_{\mathrm{SP}}$ as follows:

- the graph $G(N, A)$ and the set of demands are the same as in the UMFP;
- the set N_F is the set of nodes that are origin of at least one demand.

A service is installed on each node in N_F and all the demands originating from a node are assigned to it. If there exists a feasible routing for the Lc-VNF-PR$_{\mathrm{SP}}$ there exists a feasible solution for the UMFP and vice versa. □

Remark 4.2.3. The proof of Theorem 4.2.4 works even if $N_F = N$.

We now consider some graph topologies, namely full-mesh, ring and tree-like.

Theorem 4.2.5. The UVNF-PR$_{\text{SP}}$ is polynomially solvable on completely connected graphs.

Proof. As services are uncapacitated, one is sufficient to serve all the demands and it can be arbitrarily located on node i. As links are uncapacitated and the graph is fully connected, each demand can be routed on the path *source-i-destination*. $\square$

Remark 4.2.4. The proof of Theorem 4.2.5 works if $N_F = N$ (in fact, the hypothesis of subset N_F is not used).

Remark 4.2.5. If service capacity is considered, however, even the feasibility version of the problem (Sc-VNF-PR$_{\text{SP}}$) is $\mathcal{NP}$-complete. This can be proved by slightly modifying the reduction from BPP we presented for general graphs.

Definition 4.2.1. A ring topology is a graph in which each node is connected with exactly two other nodes, in such a way that a closed symmetric loop is formed.

Theorem 4.2.6. The UVNF-PR$_{\text{SP}}$ is polynomially solvable on ring topologies.

Proof. Any node can be chosen to install a service instance and labeled as node 1. Then, all other nodes are labeled with an increasing integer index in clockwise direction starting from node 1. For each demand there exist two paths (clockwise and anticlockwise). Given a demand, if the label of the origin node is smaller than the label of the destination node, then the demand is routed on the anticlockwise path, otherwise it is routed on the clockwise one. Thus each demand passes through node 1. $\square$

Remark 4.2.6. The proof of Theorem 4.2.6 works if $N_F = N$ (the hypothesis of N_F is not used).

Theorem 4.2.7. The feasibility version of Sc-VNF-PR$_{\text{SP}}$ is $\mathcal{NP}$-complete on ring topologies.

Proof. We prove it by reduction from the BPP. Every instance of BPP can be reduced to Sc-VNF-PR$_{\text{SP}}$ on a ring as follows:

- the set of nodes is divided into three subsets: I_1, N_F, I_2. $|I_1| = |I_2| = |I|$, for each item i there exists a pair of nodes i^s, i^t such that $i^s \in I_1, i^t \in I_2$, $|N_F| = \kappa$. Nodes are connected in a ring, as shown in Figure 4.1;
- service capacity is equal to bin capacity;
- for each item $i \in I$ there exists a demand from i^s to i^t, with demand amount equal to the item weight.

$\square$

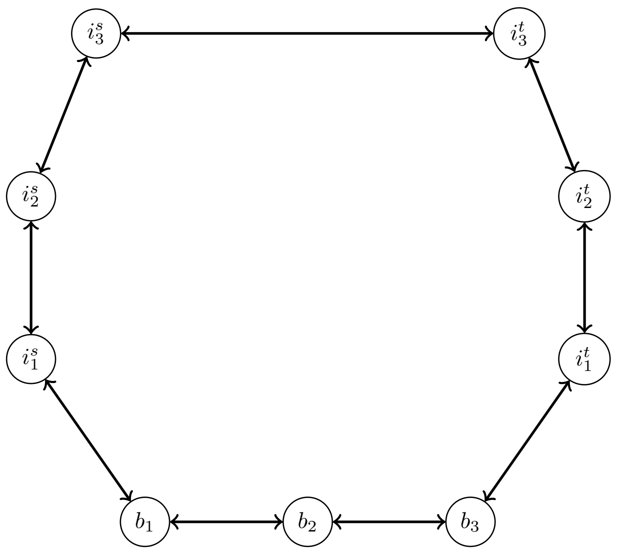


Figure 4.1 – Ring graph for reduction from BPP for the Sc-VNF-PR$_{\text{SP}}$ problem

Remark 4.2.7. The proof of Theorem 4.2.7 does not work if $N_F = N$. However, the reduction from BPP can still be used for the optimality case (where each node represents a bin).

Let us now consider a rooted tree where each edge is replaced by the corresponding pair of arcs. In the following we refer to such graph as a *tree-like topology*.

Theorem 4.2.8. The UVNF-PR$_{\text{SP}}$ is polynomially solvable on tree-like topologies if $N_F = N$.

Proof. We observe that a demand originating (or ending) in a leaf node i has to pass through its adjacent node j to reach (or to be reached by) other nodes. Thus a solution where a service is installed on the node j is always not worse than a solution where the service is installed on the leaf node i. As a consequence, a demand originating from a leaf node i (respectively, ending in i) can be represented by a demand originating (respectively, ending) from the only node adjacent to i.

The solution procedure starts with an empty set of service instances and iteratively collapses leaves on their adjacent nodes installing a service as close to the root as possible. The following three steps are repeated until all demands are served:

1. an unserved demand is considered whose origin (respectively, destination) is a leaf and its origin (respectively, destination) is moved to its adjacent node;
2. leaves that are not origin nor destination of demands anymore are removed shrinking the graph;
3. when an unserved demand has both origin and destination on the same node, the demand is assigned to the node itself, which is equipped with a service instance, if it does not already host it.

As each demand is checked at most twice in performing step (1) (origin and/or destination) and the worst case scenario is the linear graph, the overall complexity is $O(|N| \times |D|)$.

$\square$

Remark 4.2.8. Lc-VNF-PR$_{\text{SP}}$ is polynomial on tree-like topologies if $N_F = N$. Indeed, in a tree-like topology, there is only one simple path for each demand and thus, either there exists a feasible solution, and above procedure generates it, or the obtained routing is unfeasible, proving that the instance is not feasible.

In Tables 4.1-4.4 we summarize our current complexity case-studies on different topology structures (cases with normal texts are the complexity cases proved in this thesis, cases with texts in bold report the extensions of complexity proof as future works, and cases with ? means further investigations are needed)

Cap	$N_F \subset N$		$N_F = N$	
	Splittable D	Unsplittable D	Splittable D	Unsplittable D
Uncap feas	**from unsplit**	$\mathcal{P}$	**from** $N_F \subset N$	**from** $N_F \subset N$
Link cap feas	?	from UMFP	?	**from UMFP**
Service cap feas	?	from BPP	?	?
Uncap opt	**from SC**	from SC	?	?
Link cap opt	**from uncap**	**from feas**	?	**from feas**
Service cap opt	**from uncap**	**from feas**	?	**from BPP**

Table 4.1 – Summary of complexity on a general graph

Cap	$N_F \subset N$		$N_F = N$	
	Splittable D	Unsplittable D	Splittable D	Unsplittable D
Uncap feas	**from Uncap opt**	from Uncap opt	**from Uncap opt**	**from Uncap opt**
Link cap feas	?	?	?	?
Service cap feas	?	from BPP	?	?
Uncap opt	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$
Link cap opt	?	?	?	?
Service cap opt	?	**from feas**	?	**from BPP**

Table 4.2 – Summary of complexity on a full mesh graph

Cap	$N_F \subset N$		$N_F = N$	
	Splittable D	Unsplittable D	Splittable D	Unsplittable D
Uncap feas	**from Uncap opt**	from Uncap opt	**from Uncap opt**	**from Uncap opt**
Link cap feas	?	?	?	?
Service cap feas	?	from BPP	?	?
Uncap opt	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$
Link cap opt	?	?	?	?
Service cap opt	?	**from feas**	?	**from BPP**

Table 4.3 – Summary of complexity on a ring graph

Cap	$N_F \subset N$		$N_F = N$	
	Splittable D	Unsplittable D	Splittable D	Unsplittable D
Uncap feas	?	?	**from unsplit**	from Uncap opt
Link cap feas	?	?	**from unsplit**	$\mathcal{P}$
Service cap feas	**from unsplit**	**from uncap opt**	?	?
Uncap opt	?	?	**from unsplit**	$\mathcal{P}$
Link cap opt	?	?	?	?
Service cap opt	**from unsplit**	**from BPP**	?	?

Table 4.4 – Summary of complexity on a tree-like graph (where each node has one single adjacent/parent node)

4.3 Problem Properties

In this section, we investigate some properties of the VNF-PR$_{\text{SP}}$ problem. In order to guide the reading, this section is organized as follows. We first give a description and provide the full proof of each property, then we briefly summarize them at the end of this section.

We now present the first property studied. We assume that one VNF type is available, however this assumption is not so restrictive. Indeed, the problem where all the demands ask for the same sequence (same types of VNF and same order) is equivalent to the case with a single VNF type, if VNFs capacity is uniform and a node can host an instance of each VNF type.

Proposition 4.3.1. Let us consider two problems $\mathcal{P}_1$ and $\mathcal{P}_2$ that share the same features of VNF-PR$_{\text{SP}}$ but for the cardinality of the required chain of services: in $\mathcal{P}_1$ each demand requires the same unique service a, while in $\mathcal{P}_2$ a set of VNF types T is given, each VNF type in T has the same capacity of service a and all the demands require the same sequence of services, both in terms of service types and order.

If there exists an optimal solution for problem $\mathcal{P}_1$, then an optimal solution for $\mathcal{P}_2$ can be derived by installing all the required services in the optimal sites selected by the solution of $\mathcal{P}_1$ and routing the demands according to the optimal solution of $\mathcal{P}_1$.

Proof. An optimal solution for $\mathcal{P}_1$ is given, that is to say a set of locations for the instances of service a such that all the demands can be routed from their source node to their destination node passing through (at least) an instance of the service a, and respecting link capacity, service capacity and simple path constraints.

Now, let us consider the problem $\mathcal{P}_2$. By installing an instance of each service type in T on the same nodes where the instances of service a are located, and by keeping the same demand-service instance assignment as in the $\mathcal{P}_1$ solution, we obtain a feasible solution for $\mathcal{P}_2$. In fact installing several VNF type instances on one node does not affect neither the link capacity constraints nor the routing constraints. Furthermore, as the VNF types have the same capacity, if a service a instance installed in a given node i can serve all the demands assigned to it, then any service instance installed on node i can serve the same set of demands.

As any feasible solution of $\mathcal{P}_1$ can be extended to a feasible solution of $\mathcal{P}_2$ with the same number of service instances per service type, if the $\mathcal{P}_1$ solution location is optimal then it is optimal also for $\mathcal{P}_2$.

□

We can observe that under the hypothesis that the set of required service types is exactly the same, the two problems are equivalent even if they account for different service orders (we recall that it is always possible to schedule the services allocated on the same sever in any order, and that in the application problem this order is determined by a suitable scheduling algorithm performed at the server/cloud level).

We now present the second problem property: a lower bound on the number of VNF instances (and thus on the objective function) can be obtained if the graph contains at least one *articulation point*. Let us recall some definitions.

Definition 4.3.1. A graph is *biconnected* if removing any node the graph remains connected, i.e., there is a path between every pair of vertices.

Definition 4.3.2. Given a graph $G(N, A)$, a *biconnected component* is a maximal induced biconnected subgraph of G.

Biconnected components are connected to each other at shared vertices called *cut vertices* or *articulation points*.

Definition 4.3.3. An *articulation point* of a graph G is a vertex v such that $G \setminus v$ has more connected components than G.

The following property can be stated.

Proposition 4.3.2. Let us consider a VNF-PR$_{\text{SP}}$ problem defined on a graph G containing biconnected components and suppose that there exists at least one demand whose origin and destination nodes are both in the same biconnected component. Further, suppose that such biconnected component has only one articulation point. Let us refer to such biconnected components as *biconnected components with internal demand and single articulation point.*

Then, it is necessary to install a service inside each biconnected component with internal demand and single articulation point. Furthermore, if the instance is feasible, there exists an optimal solution where an instance of the service is located on each articulation point belonging to a biconnected component with internal demand and single articulation point.

Proof. Let us consider a biconnected component, a demand with both endpoints within it and a node i outside it. If the demand is served by a service installed on node i then its routing must pass first through the articulation point to reach the service and then it must pass again through the same articulation point to complete its routing, thus violating the simple path routing constraint. Thus, in a feasible solution, a service must be installed within the biconnected component to serve the demands whose source and destination belong to the biconnected component itself. Similarly, a demand with both endpoints outside the biconnected component cannot be served by a node in the biconnected component, if it is not the articulation point. Thus a solution where a service is installed in the articulation point is always at least as good as one in which the service is in the biconnected component but not in the articulation point. Therefore, the number of biconnected components with internal demand and single articulation point is a lower bound for the minimum number of installed service instances to the VNF-PR$_{\text{SP}}$ problem. □

Thanks to Property 4.3.2, it is possible to determine a lower bound combining the number and structure of the biconnected components with the source and destination of the demands. Furthermore, a partial solution can be built, installing services on articulation points. A pre-processing can be devised, which aims at

1. detecting the number of biconnected components with internal demand and single articulation point, thereby installing one service on each of their articulation points;

2. forbidding the assignment of a demand to the VNFs which are out of the biconnected component the demand belongs to.

To conclude, in this section we proved that:

- the single service case is equivalent to the multiple services one, when the services have all the same capacity and the required type and order of services is the same for all the demands;

- an instance of the service must be installed on the articulation points that belong to biconnected components with internal demands and single articulation point.

4.4 Problem Formulations

In this section we present two formulations: the Split Path (SP) formulation and the Placement and Routing (PR) one.

The first one, SP, is based on the decomposition of the path of each demand into several sub-paths, each associated with a service instance serving the demand. A similar model is presented in [26] (also here a single VNF type is considered, but no simple path assumption is considered). The second one, PR, is directly inspired by network routing problem and facility location problem: a set of variables and constraints represent the demand routing and a set of variables and constraints represent the facility location-allocation part, connecting constraints are used to couple the two sub-problems. It can be considered as the adapted version of the formulation presented in [1] to our VNF-PR$_{\text{SP}}$ problem.

In Table 4.5 the notation is summarized and the variables used by the two models are reported. As some variables are common to both models and some are model dependent, in the last column we report the model in which the parameter/variable is used.

Notation		Model
	Sets	
N	set of nodes	both
A	set of arcs	both
D	set of demands	both
	Capacities (Links and Services)	
u	arc capacity	both
q	VNF instance capacity	both
	Demand parameters	
o_k	origin of demand $k \in D$	both
t_k	destination of demand $k \in D$	both
d_k	amount of demand $k \in D$	both
	Variables common to both models (binary)	
y_i	1 if a VNF instance is located on node $i \in N$	both
z_i^k	1 if demand $k \in D$ uses the VNF instance on node $i \in N$	both
	Routing variables (binary)	
x_{ij}^k	1 if arc $(i,j) \in A$ is used by demand $k \in D$	PR
x_{ij}^{k1}	1 if arc $(i,j) \in A$ is used by demand k on sub-path 1	SP
x_{ij}^{k2}	1 if arc $(i,j) \in A$ is used by demand k on sub-path 2	SP
	TSP-like labeling variables (continuous non-negative)	
π_i^k	position of node $i \in N$ in the path used by demand $k \in D$	PR

Table 4.5 – Mathematical notation

In both models, binary variable y_i represents the location of an instance of the VNF on node $i \in N$ and binary variable z_i^k represents the assignment of demand k to the instance of the VNF located on node i. The two models differ in the way the routing is modeled. In both, binary variables are used, that are equal to one if an arc $(i,j) \in A$ is used by a demand $k \in D$. In PR, these variables are x_{ij}^k. In SP, the path is explicitly divided into two sub-paths: the first from the origin o_k to the service node (described by variables x_{ij}^{k1}) and the second from the service node to the destination t_k (described by variables x_{ij}^{k2}).

As we want to highlight the common points and differences between the two models, we present them in parallel, starting from the common part.

$$\min \textstyle\sum_{i \in N} y_i \tag{4.1}$$
$$\textstyle\sum_{i \in N} z_i^k = 1 \qquad \forall k \in D \tag{4.2}$$
$$z_i^k \le y_i \qquad \forall k \in D, i \in N \tag{4.3}$$
$$\textstyle\sum_{k \in D} d_k z_i^k \le q \qquad \forall i \in N \tag{4.4}$$

The objective function (4.1) minimizes the sum of the opened services (i.e., instances of the VNF). Constraints (4.2) impose that each demand is assigned to exactly one instance of the service. Inequalities (4.3) guarantee that if no VNF instance is installed on a node, then no demand can be assigned to it. Constraints (4.4) impose that each instance of the VNF can serve a maximum quantity q.

The link capacity constraints are similar for the two models:

SP:

$$\sum_{k\in D} d_k(x_{ij}^{k1} + x_{ij}^{k2}) \leq u \qquad \forall (i,j) \in A \tag{4.5}$$

PR:

$$\sum_{k\in D} d_k x_{ij}^{k} \leq u \qquad \forall (i,j) \in A \tag{4.6}$$

We now present the constraints characterizing each formulation. The main difference is in the way the routing is managed, and, as a consequence, in how the models deal with the coherence between service assignment and routing. In short, in the SP formulation routing and assignment are implied by modified flow balance constraints, while in the PR formulation the routing is implied by the classical flow balance constraints and the consistency between assignment and routing is implied by coherence constraints and isolated loop elimination constraints.

Routing and assignment are modeled as follows:

SP:

$$\sum_{j:(i,j)\in A} x_{ij}^{k1} - \sum_{j:(j,i)\in A} x_{ji}^{k1} = \begin{cases} 1 - z_i^k & \text{if } i = o_k \\ -z_i^k & \text{otherwise} \end{cases} \quad \forall k \in D,\ i \in N \tag{4.7}$$

$$\sum_{j:(i,j)\in A} x_{ij}^{k2} - \sum_{j:(j,i)\in A} x_{ji}^{k2} = \begin{cases} z_i^k - 1 & \text{if } i = t_k \\ z_i^k & \text{otherwise} \end{cases} \quad \forall k \in D,\ i \in N \tag{4.8}$$

$$\sum_{j:(j,i)\in A} (x_{ji}^{k1} + x_{ji}^{k2}) \ \leq 1 \ \forall k \in D, i \in N \tag{4.9}$$

$$\sum_{j:(i,j)\in A} (x_{ij}^{k1} + x_{ij}^{k2}) \ \leq 1 \ \forall k \in D, i \in N \tag{4.10}$$

Each demand is routed on two sub-paths: from the source node to the VNF node (i.e., the node where the selected service instance is installed) (equations (4.7)), then from the VNF node to the destination node (equations (4.8)). These two constraints impose that the routing of the demand passes through the VNF instance assigned to the demand itself, therefore ensure that the assignment is consistent.

Simple path routing[6] is imposed by constraints (4.9) and (4.10): for each demand, at most one of the sub-paths can pass through a node.

[6]Isolated cycles with no service can be part of a feasible solution. Such cycles can be removed obtaining a cycle-free equivalent feasible solution.

PR:

$$\sum_{j:(i,j)\in A} x_{ij}^k - \sum_{j:(j,i)\in A} x_{ji}^k = \begin{cases} 1 \text{ if } i = o_k \\ -1 \text{ if } i = t_k \\ 0 \text{ otherwise} \end{cases} \quad \forall k \in D,\ i \in N \tag{4.11}$$

$$z_i^k \ \le \sum_{(j,i)\in A} x_{ji}^k \qquad \forall k \in D,\ i \in N \setminus \{o_k\} \tag{4.12}$$

$$\pi_j^k \ \ge \ \pi_i^k + x_{ij}^k - |N|\,(1 - x_{ij}^k) \qquad \forall k \in D,\ (i,j) \in A \tag{4.13}$$

Each demand is routed from its source to its destination with the classical flow balance constraints (4.11) and it is forced to pass through the VNF instance to which it is assigned by constraints (4.12), that impose that a demand k can be assigned to a service located on node i only if the routing path of the demand passes through the given node. The simple path and the elimination of isolated cycles are enforced using the TSP-like labeling variables π and constraints (4.13): continuous variables π_i^k represents the position of node i in the routing path of demand k.

Both models can be straightforwardly generalized to take into account multiple service types, even hosted on different nodes. However, when a partial (or no) order is asked, the SP formulation needs a new set of variables to partially decouple the path-splitting and service order allocation. Further, the number of sub-paths, and of the related variables, increases with the increasing number of services in the chain. Instead, PR model can be simply generalized to deal with any imposed order (full, partial, none). In fact, variables π can be used, together with additional constraints, to impose a given full or partial order of the services along the demand path (see Section 3.4 in Chapter 3).

4.4.1 Relation between the Two Formulations

In this section we first show that the SP formulation produces a continuous relaxation bound that is always not worse than the one produced by PR, then we provide some examples where the SP formulation provides better continuous relaxation bound than PR.

The feasible region of SP can be partitioned into two subsets: in the first subset no isolated cycles exist while in the second one isolated cycles are present, but they cannot host a service (see Remark 4.4.1). Any solution of the second subset has an equivalent in the first one (see Proposition 4.4.2) and we prove that any solution of the first subset can be mapped into a solution of PR (see Theorem 4.4.1). Thus a solution of SP either is feasible also for PR or is equivalent to one that is. Instead, there exist solutions of PR that are not feasible for SP and for which the bound provided by SP is strictly better than the one provided by PR (see Proposition 4.4.3).

For any demand k, the flow in a feasible continuous solution can be divided into flow on simple paths and flow on cycles. Let us denote with P_k the set of simple paths and with C_k the set of cycles. We can distinguish between two types of cycles (see Figure 4.2): cycles sharing some nodes with a simple path (Figure 4.2a) and isolated cycles (Figure 4.2b).

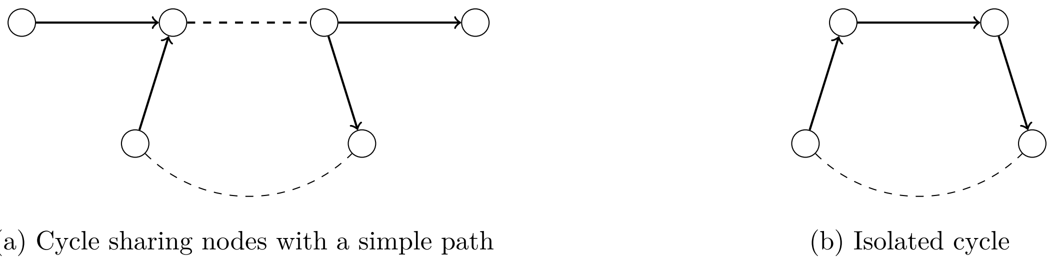

(a) Cycle sharing nodes with a simple path (b) Isolated cycle

Figure 4.2 – Examples of cycle sharing nodes and of isolated cycle

Remark 4.4.1. SP forbids demands to be served, even partially, by a VNF instance installed on an isolated cycle (notice that, PR accepts solutions with an isolated cycle and a partial service installed on it, but routing variable values (on the cycle) cannot be greater than $\frac{|N|}{m+|N|}$ (where m is the length of the cycle) due to the isolated loop elimination constraints (4.13)). For example, let us consider an isolated cycle of two nodes A and B such that the demand is partially served by a node A ($z^k_A = \alpha, \alpha < 1$). Summing up flow balancing constraints (4.7) leads to an inconsistency $0 = -\alpha$ (analogously for subpath 2).

As a consequence, only three cases occur:

Remark 4.4.2. Consider a feasible solution of the continuous relaxation of SP (x^1, x^2, y, z), a demand k and a cycle induced by such solution. Let us denote with N_c the set of nodes of the cycle c that are not shared with any simple path: $N_c = \{i \in c \setminus P_k\}$. Due to flow balance constraints (4.7)-(4.8), we have only three possible cases for all nodes belonging to N_c:

1. if a (partial) service instance is located on any node belonging to N_c, the cycle is induced by variables of both semi-paths, and $x^{k1}_i = x^{k2}_i$
2. if no service is located on any node belonging to N_c, then
   2.1) the cycle is induced by only one type of semi-path variables
   2.2) the cycle can be decomposed in two super-imposed cycles, each of them generated by only one type of semi-path variables.

Thus we can observe that:

Proposition 4.4.1. Let us consider a feasible solution (x^1, x^2, y, z), a demand k and the path-cycle decomposition $\{P_k, C_k\}$ of its routing. Let us suppose that a cycle $c \in C_k$ exists that shares at least one node with a simple path $p \in P_k$.

If a (partial) service is located on a node belonging to a cycle $c \in C_k$, but not to the path ($\{i \in c \setminus P_k\}$), the routing variables inducing the path and the cycle are fractional and their value is less than or equal to $\frac{1}{2}$, due to constraints (4.7)-(4.8) and (4.9)-(4.10).

Proposition 4.4.2. Any solution such that no service instance is installed on a node belonging to a cycle, but not to a simple path, can be transformed in an equivalent solution (in term of cost, location and assignment) removing the cycles without service instances installed and the corresponding flow from the routing.

Theorem 4.4.1. Any solution of the continuous relaxation of SP such that no service instance is installed on a node belonging to a cycle, but not to a simple path, can be mapped into an equivalent solution of PR in terms of routing, location and assignment and therefore cost.

Proof. Location variables y_i and assignment variables z_i^k, the objective function and constraints (4.2), (4.3) and (4.4) are common to both formulations. SP routing variables can be easily mapped into PR ones as follows:

$$x_{ij}^k = x_{ij}^{k1} + x_{ij}^{k2} \tag{4.14}$$

Thanks to constraints (4.9) and (4.10) the mapping guarantees also that $x_{ij}^k \in [0, 1]$. Further, link capacity constraints of SP (4.5) imply the link capacity constraint of PR (4.6). Routing constraints of the SP formulation imply the routing constraints for the PR formulation. In fact, by summing equations (4.7)-(4.8) and using the mapping (4.14), we obtain constraints (4.11).

Now we show that SP routing constraints for the semi-path (4.7) imply routing-location connecting constraints of PR (4.12).
Let us consider the case $i \in N, i \neq o_k$:

$$\sum_{j:(i,j)\in A} x_{ij}^{k1} - \sum_{j:(j,i)\in A} x_{ji}^{k1} \;=\; -z_i^k$$

reordering terms, we obtain:

$$\sum_{j:(i,j)\in A} x_{ij}^{k1} + \; z_i^k \;=\; \sum_{j:(j,i)\in A} x_{ji}^{k1}$$

As $\sum_{j:(i,j)\in A} x_{ij}^{k1} \geq 0$, we can derive:

$$z_i^k \;\leq\; \sum_{j:(j,i)\in A} x_{ji}^{k1}$$

Adding $\sum_{j:(i,j)\in A} x_{ij}^{k2}$ at the right side, as this term is always not negative, the inequality still holds:

$$z_i^k \leq \sum_{j:(j,i)\in A} (x_{ji}^{k1} + x_{ij}^{k2})$$

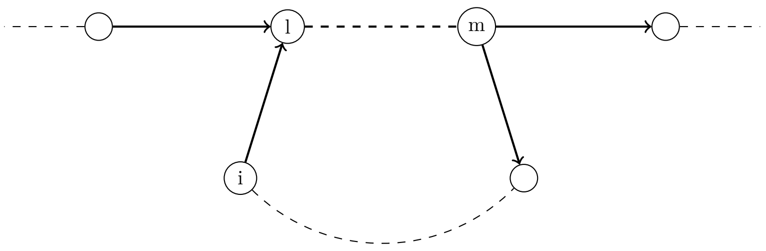


Figure 4.3 – An example of a cycle sharing some nodes with a simple path

Then, using the routing variables mapping (4.14), we verify constraint (4.12):

$$z_i^k \leq \sum_{j:(j,i)\in A} x_{ji}^k.$$

Finally, suitable values for π variables must be derived. Let us consider a demand k and its routing. We can build an induced graph as follows:

$$G^k(N_k, A_k)$$

where $(i,j) \in A_k$ if $x_{ij}^{k1} + x_{ij}^{k2} > 0$ and $\exists p \in P_k : (i,j) \in p$, i.e. the arc belongs to at least a simple path. A node $\hat{\imath}$ belongs to N_k if there exists an arc $((\hat{\imath}, j)$ or $(j, \hat{\imath}))$ in A_k. For any arc $(i,j) \in G^k$ we define the following cost $c_{ij} = x_{ij}^{k1} + x_{ij}^{k2} = x_{ij}^k$.
As G^k does not contain arcs that belong only to cycles in C^k, the obtained graph is acyclic, thus we can define π_j^k as the longest path from node o_k to node j:

- $\pi_{o_k}^k = 0$
- $\pi_j^k = \max_{i:(i,\hat{\jmath})\in A^k}\{\pi_i^k + x_{ij}^k\}$

Such values obviously satisfy constraints (4.13) for the nodes belonging to the path.

The nodes that belong to a cycle and to a simple path (nodes from l to m in Figure 4.3) have already been assigned a suitable value π. We now need to determine a value of π for the nodes on the cycle N_C that have been removed.

Let us consider the two nodes l and m, corresponding to the smallest (π_l^k) and largest (π_m^k) value of variables π on the cycle N_C, respectively. For all the nodes $i \in N_c$, we impose:

$$\pi_i^k = \frac{\pi_m^k + \pi_l^k}{2}$$

Now, we prove that such value of π always satisfy constraints (4.13).
We can observe that $x_{ij} \leq 1$ in any feasible solution, therefore the values of π are bounded by $|N| - 1$. As for the arcs in $A \setminus A^k$, we have two cases.

1. Arcs belonging to the cycle but not a path.
   Due to Property 4.4.1, we can infer that for any arc belonging to the cycle but not the path, we have $x^k_{ij} \leq \frac{1}{2}$. Therefore, the term:

   $$x^k_{ij} - |N| \, (1 - x^k_{ij})$$

   is always non positive as $|N| \geq 2$. Therefore for any two nodes in the cycle, as they have the same value of π, the constraint is valid.

2. Arcs belonging to the cycle and incident both in the path and the cycle, i.e. with one extreme in l or m.
   Let consider the arc of the cycle entering node l: (i, l) (see Figure 4.3), we need to prove that:

   $$\pi^k_l \geq \pi^k_i + x^k_{il} - |N|(1 - x^k_{il}), \tag{4.15}$$

   Let us denote with n the number of arcs in the path between node l and node m ($n \leq |N| - 1$), and with $\beta \leq 1$ the value of the associated routing variable on the path, then we have that:

   $$\pi^k_m = \pi^k_l + \beta n$$

   and we obtain:

   $$\pi^k_i = \frac{\pi^k_m + \pi^k_l}{2} = \pi^k_l + \frac{\beta n}{2} \leq \pi^k_l + \frac{(|N| - 1)}{2}$$

   Thus, denoting with γ the right-hand side of equation (4.15), we get:

   $$\gamma \leq \pi^k_l + \frac{(|N| - 1)}{2} + x^k_{il} - |N|(1 - x^k_{il})$$

   and, rearranging the terms,:

   $$\gamma \leq \pi^k_l + \left(x^k_{il} - \frac{1}{2} \right) (|N| + 1)$$

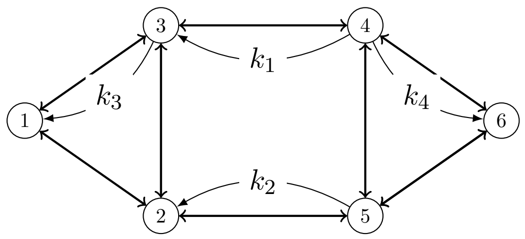
   and using $x^k_{il} \leq \frac{1}{2}$ (see Proposition 4.4.1):

   $$\pi^k_l \geq \gamma$$

   A similar argument can be used to prove that the constraint is valid for the link belonging to the cycle and exiting the other extreme node m.

□

We now prove that PR bound is strictly worse than the one of SP, by showing an instance in which the bound provided by SP is stricter than the one provided by PR. Further, we present some features that make a solution unfeasible for the SP continuous relaxation but are acceptable for the PR continuous relaxation.



(a) Topology and demands distribution.

k	$o_k \to t_k$	d_k
1	$4 \to 3$	**5**
2	$5 \to 2$	**5**
3	$3 \to 1$	$\forall d_3 \in (0,5]$
4	$4 \to 6$	$\forall d_4 \in (0,5]$

(b) Demands.

Figure 4.4 – The SP relaxation bound is strictly better than the one of PR: a numerical example

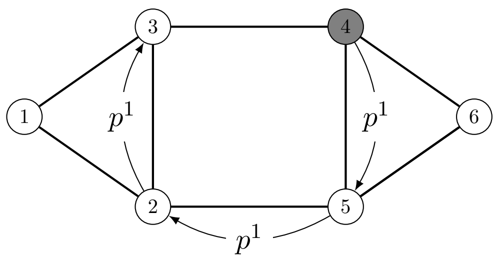
Proposition 4.4.3. The bound provided by PR is strictly worse than the one provided by SP.

Proof. Let us consider the symmetric graph in Figure 4.4a (in the following figures, each pair of the symmetric arcs is represented by the corresponding edge). The services are uncapacitated, and the link capacity is 5. There are 4 demands, whose characteristics are listed in Table 4.4b. The continuous relaxation provided by SP is equal to 2, while the continuous relaxation provided by PR is 1, for any value of the demands k_3 and k_4 in $(0, 5]$.

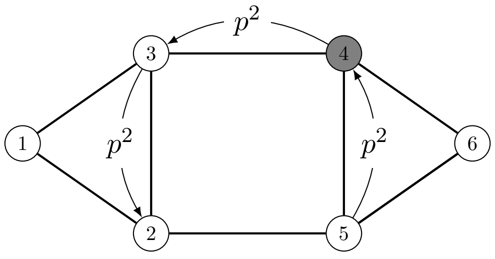
demand	paths	f_p	node (i)	z_i^k
k_1	$p^1 : \mathbf{4} \to 5 \to 2 \to 3$	1	4	1
k_2	$p^2 : 5 \to \mathbf{4} \to 3 \to 2$	1	4	1
k_3	$p^3 : \mathbf{3} \to 1$	1	3	1
k_4	$p^4 : \mathbf{4} \to 6$	1	4	1
Installed services				
$y_3 = 1 \quad y_4 = 1$				

Table 4.6 – Routing and assignment/location in the continuous relaxation of SP.

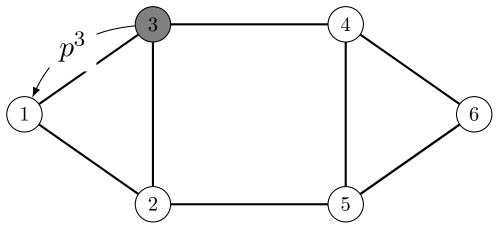
We report the details of the solution of SP formulation in Table 4.6 and Fig. 4.5, while the details of the solution of PR in Fig. 4.6 and Table 4.7: the assignment of services instances to nodes is reported in bold in the tables and in gray in the figures. As for the SP formulation, one service is located on node 4 and one on node 3. Demands k_1, k_2 and k_4 are served by the service located on node 4, while demand k_3 is served by the service located on node 3. As for the solution of PR, half service is installed on node 4 and the other half on node 5; each demand uses both services.



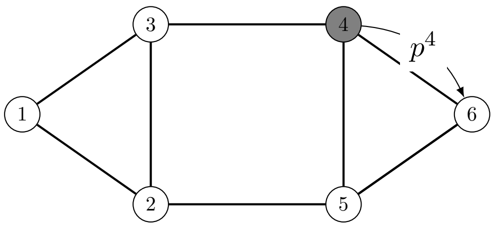
(a) Routing of demand k_1.



(b) Routing of demand k_2.

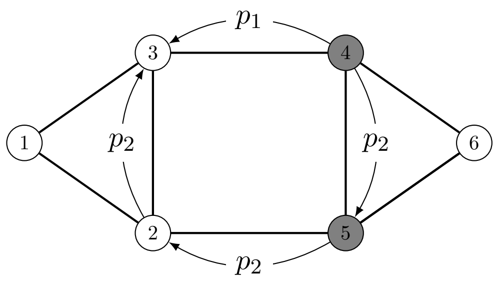


(c) Routing of demand k_3.

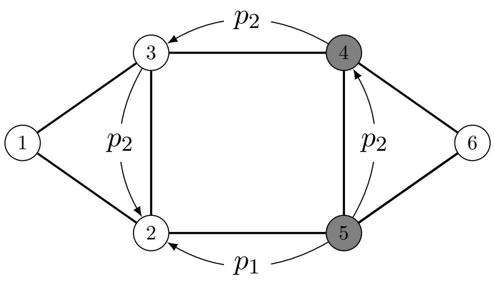


(d) Routing of demand k_4.

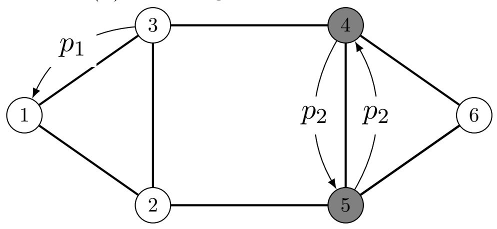
Figure 4.5 – The routing provided by the continuous relaxation of SP



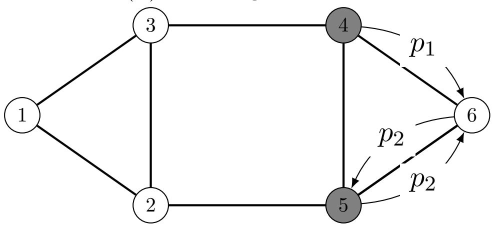
(a) Routing of demand k_1



(b) Routing of demand k_2



(c) Routing of demand k_3



(d) Routing of demand k_4

Figure 4.6 – The routing solution of continuous relaxation of the PR

□

The difference between the two bounds is due to the fact that in the continuous relaxation of the PR formulation some routing solutions are feasible, while they are

demand	paths	f_p	node (i)	z_i^k
k_1	$p_1 : \mathbf{4} \rightarrow 3$	0.5	4	0.5
	$p_2 : 4 \rightarrow \mathbf{5} \rightarrow 2 \rightarrow 3$	0.5	5	0.5
k_2	$p_1 : \mathbf{5} \rightarrow 2$	0.5	5	0.5
	$p_2 : 5 \rightarrow \mathbf{4} \rightarrow 3 \rightarrow 2$	0.5	4	0.5
k_3	$p_1 : 3 \rightarrow 1$	1	-	
	$p_2 : 5 \rightarrow \mathbf{4} \rightarrow \mathbf{5}$	0.5	4	0.5
			5	0.5
k_4	$p_1 : \mathbf{4} \rightarrow 6$	1	4	0.5
	$p_2 : 5 \rightarrow 6 \rightarrow \mathbf{5}$	0.5	5	0.5
Installed services				
$y_4 = 0.5 \quad y_5 = 0.5$				

Table 4.7 – Routing and assignment/location solution in the continuous relaxation of PR.

not for the SP formulation, because of the direct coupling of assignment and routing determined by constraints (4.7)-(4.8) present in SP. Indeed, the example above is a representative of a family of solutions that are feasible for PR but not for SP.

Remark 4.4.3. A solution where a source-destination path does not pass through any service node (see path p_1 for demand k_3 in 4.4.3) and an isolated loop hosts a (partial) service (see path p_2 for demand k_3 in 4.4.3) is feasible for PR but not for SP.

Such solutions may be profitable when the capacity of a cut is small and some demands cannot reach the services installed on the other side of the cut itself: using a service on an isolated cycle is then the best option for PR. In the example described in Table 4.4b demands k_1 and k_2 saturate the cut $\{4, 5, 6\}, \{1, 2, 3\}$ forcing SP to install a service in each side of the cut to serve also demands k_3 and k_4. Instead, PR does not open two services, thus providing a weaker bound. The bound provided by SP is stricter even if the amount of flow of demands k_1 and/or k_2 decreases and thus the cut is not fully saturated, as long as the residual capacity on the cut $\{4, 5, 6\}, \{1, 2, 3\}$ is not enough to allow the demand k_3 to pass fully through it and back (or symmetrically demand k_4 for the reversed cut). A feasible solution is then selected by PR as described in Fig. 4.7, where the demand is half routed on the path and is completely served by partially using the two services located in node i and j. Such solution is instead unfeasible for SP (the larger the fraction of demand k_3 that can pass the cut, the smaller the gap between the two continuous relaxation bounds), as proved by the following proposition.

Proposition 4.4.4. Let us consider a feasible solution for the continuous relaxation of SP formulation where a fraction of a demand k is routed on a path p_k. Let us consider the services located along this path and the corresponding assignment variables z_i^k, and suppose that such services are not used by demand k along other

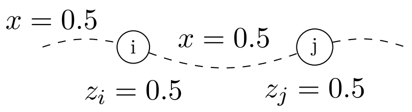


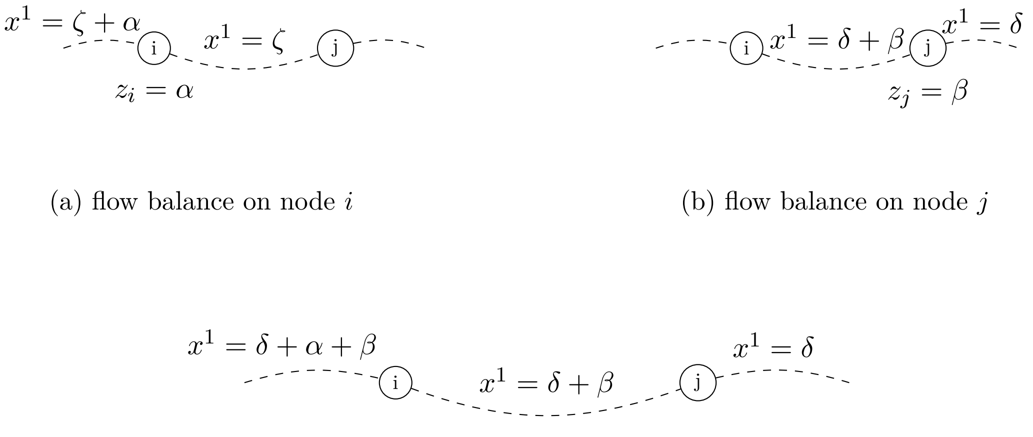
Figure 4.7 – Example of unfeasible flow acceptable for PR

paths. Then:

$$x_{lm}^{k1} \geq \sum_{i \in p_k : i = \mathrm{succ}(lm)} z_i^k \tag{4.16}$$

where with succ(lm) we represent all the nodes that appear in the path p_k not before arc (l, m) (node m is considered belonging to succ(lm)).

Proof. This property is a direct consequence of the modified flow balancing constraints (4.7)-(4.8). See Figure 4.8 for a schematic illustration of this property. □



(a) flow balance on node i

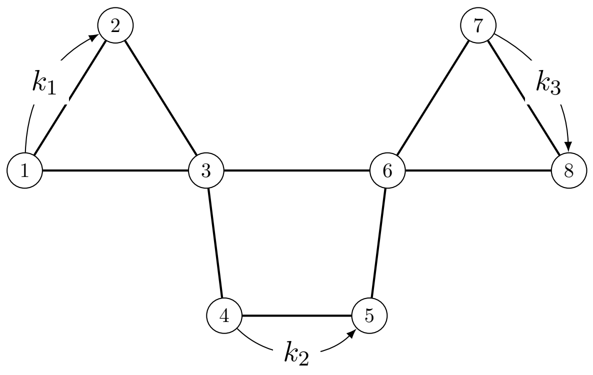
(b) flow balance on node j

(c) resulting feasible values for x^1

Figure 4.8 – Relation between routing and assignment variables in the SP formulation.

4.4.1.1 Impact of the Biconnected Components

The network topology has an impact on the quality of the bounds produced by the two formulations. In fact, the SP formulation can produce a tighter bound in presence of biconnected components, even in the uncapacitated case, due to the simple path assumption.



(a) Topology and demands distribution.

k	$o_k \to t_k$	d_k
1	$1 \to 2$	1
2	$4 \to 5$	1
3	$7 \to 8$	1

(b) Demands.

Figure 4.9 – The SP relaxation bound is strictly better than the one of PR: a numerical example with biconnected components

Let us consider the graph in Fig. 4.9a, and a set of three demands described in Table 4.9b. Service and link capacities are unbounded. The graph contains two articulation points (nodes 3 and 6) and 3 biconnected components:

$$BC_1 = \{1, 2, 3\}, \quad BC_2 = \{3, 4, 5, 6\}, \quad BC_3 = \{6, 7, 8\}.$$

The continuous relaxation bound obtained by the SP formulation for this instance is $\frac{4}{3}$, the one obtained by the PR formulation is the trivial bound 1.

demand	paths	fraction f_p	node (i)	z_i^k
k_1	$p_1^1 : 1 \to \mathbf{2}$	0.5	2	0.5
	$p_2^1 : 1 \to 3 \to 6 \to \mathbf{8} \to 6 \to 3 \to 2$	0.5	8	0.5
k_2	$p_1^2 : 4 \to \mathbf{5}$	1/3	5	1/3
	$p_2^2 : 4 \to 3 \to \mathbf{2} \to 3 \to 6 \to 5$	1/3	2	1/3
	$p_3^2 : 4 \to 3 \to 6 \to \mathbf{8} \to 6 \to 5$	1/3	8	1/3
k_3	$p_1^3 : 7 \to \mathbf{8}$	0.5	8	0.5
	$p_2^3 : 7 \to 6 \to 3 \to \mathbf{2} \to 3 \to 6 \to 8$	0.5	2	0.5
Installed services				
$y_2 = 0.5 \quad y_5 = 1/3 \quad y_8 = 0.5$				

Table 4.8 – Routing and assignment/location in the continuous relaxation of SP for the biconnected components case example.

In Table 4.8, a solution (for both formulations several equivalent solutions exist, both for location and routing. For the sake of clarity, we chose to show a “compact” solution for both formulations privileging symmetric, less fractional solutions, with short routings and a reduced number of isolated cycles) for the SP formulation is reported. For each path, the node where the (partially) used service is located is reported in bold. Partial services are installed on nodes 2, 5 and 8. Half for nodes 2 and 8 and one third for node 5. Demand k_1 (k_3, respectively) is served by the half service installed on node 2 belonging to its connected component BC_1 (node 8 in BC_3 respectively) and by half service installed on node 8 in connected component BC_3(node 2 in BC_1, respectively). Demand k_2 is split on three paths, and each of them is served by a (partial) service in a different connected component.

In Table 4.9 a solution for the PR formulation is reported. Each demand is routed on the direct arc connecting its origin to its destination, thus satisfying the flow balance constraints (4.11). To reach the service, both demand k_2 and k_3 use two isolated cycles each in the connected component BC_1. It is worth to notice that a single cycle would not be enough to satisfy both the coherence constraints (4.12) and the TSP-like cycle elimination constraints (4.13) (only fractional solutions can have cycles).

demand	paths	fraction f_p	node (i)	z_i^k
k_1	$p^1 : \mathbf{1} \to 2$	1	1	1
k_2	$p_1^2 : \mathbf{1} \to 3 \to 1$	0.5	1	0.5
	$p_2^2 : \mathbf{1} \to 2 \to 1$	0.5	1	0.5
	$p_3^2 : 4 \to 5$	1	-	
k_3	$p_1^3 : \mathbf{1} \to 3 \to 1$	0.5	1	0.5
	$p_2^3 : \mathbf{1} \to 2 \to 1$	0.5	1	0.5
	$p_3^3 : 7 \to 8$	1	-	
Installed services				
$y_1 = 1$				

Table 4.9 – Routing and assignment/location in the continuous relaxation of PR for the biconnected components case example.

4.4.2 Additional Inequalities

Some additional inequalities can be introduced to enrich both formulations. We can rewrite the service capacity constraint (4.4): the total demand that can be served by the service located on the node i is limited not only by the service capacity but also by the overall capacity of the links incident in the node (excepted the demands served in the origin node i). Symmetrically, the overall capacity of the outgoing links imposes a limit on the amount of demand served in a node. We can calculate the maximal demand that can be served by a node as follow:

$$\bar{q_i} = \min\left\{q, \max\left\{\sum_{(i,j)\in A} u + \sum_{k\in D:t_k=i} d_k, \sum_{(j,i)\in A} u + \sum_{k\in D:o_k=i} d_k\right\}\right\} \tag{4.17}$$

Therefore, constraint (4.4) can be replaced by:

$$\sum_{k\in D} d_k\ z_i^k \ \leq\ \bar{q_i}\ \forall i \in N \tag{4.18}$$

Both constraint (4.4) and constraint(4.18) can be strengthened using the service location variable y, obtaining:

$$\sum_{k\in D} d_k\ z_i^k \ \leq\ q\ y_i\ \forall i \in N \tag{4.19}$$

and

$$\sum_{k\in D} d_k\ z_i^k \ \leq\ \bar{q_i}\ y_i\ \forall i \in N \tag{4.20}$$

respectively.

When the capacity of a single VNF instance is not large enough to serve all the demands, a bound on the number of needed VNF instances can be calculated and introduced in a valid inequality (VI):

$$\sum_{i\in N} y_i \ \geq\ \left\lceil \frac{\sum_{k\in D} d_k}{q} \right\rceil\ \forall i \in N \tag{4.21}$$

Additionally, VIs inspired by cover inequalities can be added. As preliminary tests showed that adding all cover inequalities was not effective, we decide to select only the *max-minimal cover*. We define the max-minimal cover C_1 with respect to the service capacity as the minimal cover, i.e.

- $\sum_{k\in C_1} d_k > q$
- $C_1 \setminus k$ is not a cover for any $k \in C_1$

such that $|C_1|$ is maximal among all possible covers, namely the maximum number of demands that can be served by one service. We can find C_1 as follows: we order the set of demands in increasing order of demand amount, and then select them until the total capacity is reached. Then we introduce the following constraints:

$$\sum_{k\in D} z_i^k \ \leq \ |C_1| - 1 \ \forall i \in N \tag{4.22}$$

We can define in the same way a *max-minimal cover* C_2 for link utilization, changing the total demand condition as follows $\sum_{k\in C_2} d_k > u$. Thus, we get the following VIs:

For PR:

$$\sum_{k\in D} x_{ij}^k \ \leq \ |C_2| - 1 \quad \forall (i,j) \in A \tag{4.23}$$

For SP:

$$\sum_{k\in D} (x_{ij}^{k1} + x_{ij}^{k2}) \ \leq \ |C_2| - 1 \quad \forall (i,j) \in A \tag{4.24}$$

4.4.3 Extended Split Path (SP) Model for Multiple Services

We now consider to have multiple services and that each demand can ask for a different number of services among the available ones. We need to introduce the following notation:

- F: set of VNFs types
- n_k: number of services asked by demand k
- VNF_f^k: indicator parameters, equal to 1 if demand k asks for service of type f

Furthermore, if the chain order is given, we need to define:

- $f^k(s) : 1..n_k \rightarrow F$: integer indicator map, type of service at position s for demand k, 0 if the service is not requested

Decision variables must be extended accordingly:

- $y_{if} \in \{0,1\}$ if service $f \in F$ is located on node $i \in N$
- $z_{if}^k \in \{0,1\}$ if demand $k \in D$ uses service $f \in F$ on node $i \in N$
- $x_{ij}^{ks} \in \{0,1\}$ if arc $(i,j) \in A$ is used by demand $k \in D$, sub-path $s \in 1..n_k + 1$

The resulting extended model is:

$$\min \sum_{i\in N} \sum_{f\in F} y_{if} \tag{4.25}$$

$$\sum_{i \in N} z^k_{if} \;=\; 1 \qquad \forall k \in D,\; f \in F : VNF^k_f = 1 \tag{4.26}$$

$$z^k_{if} \;\leq\; y_{if} \qquad \forall k \in D,\; i \in N,\; f \in F \tag{4.27}$$

$$\sum_{k \in D} \sum_{s \in 1..n_k+1} d_k\, x^{ks}_{ij} \;\leq\; u \qquad \forall (i,j) \in A \tag{4.28}$$

$$\sum_{k \in D : VNF^k_f = 1} d_k\, z^k_{if} \;\leq\; q_f \qquad \forall i \in N,\; f \in F \tag{4.29}$$

$$\textstyle\sum_{j:(i,j)\in A} x^{k,s}_{ij} - \sum_{j:(j,i)\in A} x^{k,s}_{ji} \;= z^k_{i,f^k(s-1)} \;-\; z^k_{i,f^k(s)} \qquad \forall k \in D,\; i \in N,\; s \in 2..n_k \tag{4.30}$$

$$\textstyle\sum_{j:(i,j)\in A} x^{k,1}_{ij} - \sum_{j:(j,i)\in A} x^{k,1}_{ji} = \begin{cases} 1 - z^k_{i,f^k(1)} & \text{if } i = o_k \\ -z^k_{i,f^k(1)} & \text{otherwise} \end{cases} \qquad \forall k \in D,\; i \in N \tag{4.31}$$

$$\textstyle\sum_{j:(i,j)\in A} x^{k,n_k+1}_{ij} - \sum_{j:(j,i)\in A} x^{k,n_k+1}_{ji} \;=\; \begin{cases} z^k_{i,f^k(n_k)} - 1 & \text{if } i = t_k \\ z^k_{i,f^k(n_k)} & \text{otherwise} \end{cases} \qquad \forall k \in D,\; i \in N \tag{4.32}$$

$$\sum_{s \in 1..n_k+1} \sum_{j:(j,i)\in A} x^{ks}_{ji} \leq 1 \qquad \forall k \in D, i \in N \tag{4.33}$$

$$\sum_{s \in 1..n_k+1} \sum_{j:(i,j)\in A} x^{ks}_{ij} \leq 1 \qquad \forall k \in D, i \in N \tag{4.34}$$

To extend the model for the case with unordered services, it is necessary to decouple the sub-paths description and the services assignment to nodes (originally both represented by variable z). We keep variables z to represent the location of services, while the new variables w will be used to describe the flow balance for sub-paths:

- $w^{ks}_i \in \{0,1\}$ if demand $k \in D$ uses the $s-th$ service on node $i \in N$

Only the flow balancing constraints (Eq. 4.35-Eq.4.37) are affected by this change:

$$\textstyle\sum_{j:(i,j)\in A} x^{k,s}_{ij} - \sum_{j:(j,i)\in A} x^{k,s}_{ji} \;= w^{k,s-1}_i \;-\; w^{k,s}_i \qquad \forall k \in D,\; i \in N,\; s \in 2..n_k \tag{4.35}$$

$$\textstyle\sum_{j:(i,j)\in A} x_{ij}^{k,1} - \sum_{j:(j,i)\in A} x_{ji}^{k,1} = \begin{cases} 1 - w_i^{k1} & \text{if } i = o_k \\ -w_i^{k1} & \text{otherwise} \end{cases}$$
$$\forall k \in D,\ i \in N \tag{4.36}$$

$$\textstyle\sum_{j:(i,j)\in A} x_{ij}^{k,n_k+1} - \sum_{j:(j,i)\in A} x_{ji}^{k,n_k+1} = \begin{cases} w_i^{k,n_k} - 1 & \text{if } i = t_k \\ w_i^{k,n_k} & \text{otherwise} \end{cases}$$
$$\forall k \in D,\ i \in N \tag{4.37}$$

and additional consistency constraints 4.38 must be added to link z and w variables:

$$\sum_{f\in F:VNF_f^k=1} z_{if}^k = \sum_{s\in 1..n_k+1} w_i^{k,s} \qquad \forall k \in D,\ i \in N \tag{4.38}$$

4.5 Computational Results

We performed computational tests on the SP and PR formulations so as to

- compare the performance of the two formulations (Section 4.5.2 and Section 4.5.3)
- evaluate the scalability of the two formulations (Section 4.5.4)
- evaluate the impact of Property 4.3.1 (Section 4.5.5)
- evaluate the impact of the articulation point based preprocessing (Section 4.5.6)

4.5.1 Test Instances

We have generated a test bed based on 24 instances from the SNDLib [62]. For each instance (topology and set of demands), we have generated different capacity profiles to analyze the impact of the VNF and link capacity:

- as for the links, two levels of capacity have been generated: *low* and *high*. The high capacity is such that all the demands can be routed on a single link, thus leading to uncapacitated link instances. The low capacity is computed as the minimum capacity such that a feasible routing exists, neglecting the services;
- as for the services, three levels of capacity are considered: *low*, *medium* and *high*. The high capacity is computed so as to guarantee that all the demands can be served by a single VNF (service uncapacitated instances). Low VNF capacity is twice the total amount of the demands divided by the number of nodes, that is, we need to install a VNF in at least half of the nodes (for lower values many instances were not feasible due to the network topologies). Medium capacity is the average between high and low.

In the following, we denote with h, m and l the *high*, *medium* and *low* capacity level respectively. For example instances marked with l_h are the ones where service capacity assumes the lowest value and the link the highest (therefore uncapacitated with respect of links). The obtained 144 instances features are summarized in Table 4.10, where each row refers to a network topology. Columns two to four report the network features from SNDLib (number of nodes, number of links and number of demands). Column five gives the sum of the demand amount based on the SNDLib values. Such value corresponds to the high capacity value both for links and services. The last three columns give the remaining values of capacity of VNFs and links: the first two report the medium and low capacity values for the VNFs and the last one reports the low capacity value for the links. The first 16 network topologies, from “di-yuan” to “norway” have less than 30 nodes and are used in the formulations comparison; 6 topologies, from “india35” to “germany50” have between 30 and 50 nodes are used to assess formulation scalability; the last two topologies “ta2” and “zib54” are added with france to assess the impact of adding the articulation point based pre-processing.

Data from SNDLib					Capacity		
Network	$\|N\|$	$\|A\|$	$\|D\|$	$\sum_{k \in D} d_k$	Service		Link
				(high cap)	medium	low	low
Small-medium size instances							
di-yuan	11	42	22	53	31	9	5
pdh	11	34	24	4621	2730	840	384
polska	12	18	66	9943	5800	1657	995
sun	27	51	67	476	255	35	53
dfn-bwin	10	45	90	548388	329032	109677	55916
nobel-us	14	21	91	5420	3097	774	486
nobel-germany	17	26	121	660	368	77	74
abilene	12	15	132	3000002	1750001	500000	829282
atlanta	15	22	210	136726	77478	18230	19404
newyork	16	49	240	1774	997	221	66
france	25	45	300	99830	53908	7986	9413
nobel-eu	28	41	378	1898	1016	135	214
ta1	24	51	396	10127249	5485593	843937	819678
geant	22	36	462	2999992	1636359	272726	359868
janos-us	26	42	650	80000	43076	6153	7624
norway	27	51	702	5348	2872	396	358
Large size instances							
india35	35	80	595	3292	1740	188	121
cost266	37	57	1332	679598	358166	36735	53562
giul39	39	86	1471	7366	3871	377	363
janos-us-ca	39	61	1482	2032274	1068246	104219	180471
pioro40	40	89	780	115953	60875	5797	7609
germany50	50	88	662	2365	1229	94	123
Biconnected components instances							
zib54	54	80	1501	12230	6341	484	528
ta2	65	108	1869	31419014	16192876	966738	1311190

Table 4.10 – Test-instance details

Models are implemented in AMPL and instances are solved with IBM ILOG CPLEX (version 12.7.1.0 is used for tests in Section 4.5.2, Section 4.5.3 and Section 4.5.6, and version 12.8.0.0 for tests in Section 4.5.4 and Section 4.5.5) on an Intel Xeon, CPU E5-1620 v2 (4 cores), 3.7 GHz with 32 GB of RAM. A time limit of 3600s and a tree memory limit of 3000 MB are set, but for solving the large instances, the time limit has been extended to 7200s.

4.5.2 Continuous Relaxation Results on Small-medium Size Instances

We first compare the results of the two formulations on the continuous relaxation, then we evaluate the impact of the additional inequalities described in Section 4.4.

Summarized results on the continuous relaxations of the two formulations are reported in Table 4.11. The instances are grouped based on the capacity levels and aggregated values over the 16 topologies (from "di-yuan" to "norway") are reported. As we proved in Section 4.3, the SP formulation produces a bound that is never worse than the PR formulation, therefore, (in the first group of columns) we report the improvement obtained by the SP continuous relaxation upon the PR one. In the second column the number of instances where SP obtains a better bound than PR is given. The third and fourth columns report the average and maximum percentage of the improvement, calculated as $100\% * \frac{CR_{SP}-CR_{PR}}{CR_{PR}}$, where CR_{SP} and CR_{PR} denote the continuous relaxation of SP and PR, respectively. The average value is computed based on the instances where SP improves upon PR. The last four columns report the average and the maximum computational times (in seconds) for the two formulations. The best average computational time is reported in bold.

As expected, SP formulation always provides a continuous relaxation bound that is better than or equal to the one provided by PR. When the link capacity is high SP improves upon PR just in one or two instances, but for such instances the improvement in the bound can be quite significant (up to around 150% for the l_h case) and the average and maximum computational times are smaller (at least one third less). When link capacity is low, the number of instances where the SP formulation produces a better bound increases greatly (11-12 cases over 16) and the average and maximum improvement is more significant (up to 350%). The improvement is obtained at the price of a longer computational time, and SP cannot find a feasible solution within the time limit (3600s) for one instance (marked with * in Table 4.11), "norway" l_l, while PR does.

cap	Improvement			Computational times (s)			
	SP vs. PR			SP		PR	
	#	avg	max	avg	max	avg	max
h_h	1	100.00	100.00	**0.77**	4.13	5.23	46.79
h_l	12	143.00	349.38	136.67	1502.60	**83.13**	698.42
m_h	2	62.00	100.00	**1.35**	7.89	7.24	61.48
m_l	12	143.00	349.38	116.32	1240.33	**96.27**	513.98
l_h	2	109.7	152.66	**10.41**	52.35	29.42	134.35
l_l*	11	131.92	270.54	138.41	1153.76	**111.91**	875.50

Table 4.11 – Comparison between the continuous relaxations.

We run tests on the additional inequalities introduced in Section 4.4. In Table 4.12, we report summarized qualitative results. In the first column we report the equation corresponding to the VI. Constraints (4.20) are enhanced versions of the VNF capacity constraint, so they replace the VNF capacity constraint (4.4). In the second and the third columns the change obtained by adding each VI, in terms of bound and computational time, is qualitatively described.

VI	Bound	Runtime
Eq.(4.20)	almost always better	almost always increased
Eq.(4.21)	almost always better	almost always decreased
Eq.(4.22)	always unchanged	unchanged/increased
Eq.(4.23) (PR)	always unchanged	unchanged/increased
Eq.(4.24) (SP)	always unchanged	almost always unch./decr.

Table 4.12 – Adding valid inequalities to continuous relaxation: qualitative behavior.

In the following, we discuss in detail the most effective additional inequalities in terms of bound improvement and/or computational time reduction, i.e. the enhanced capacity equation (4.20) (in the following VI1) and the lower bound on the minimum number of VNFs: equation (4.21) (in the following VI2). We consider them singularly and together (VI1+2).

We compare the continuous relaxation bound obtained by the two formulations with the best upper bound known to evaluate its overall quality. The comparison is reported in Table 4.13. Results are reported in aggregated form, grouping instances based on the capacity level, for the case with noVI, and with VI1, VI2 and VI1+2. For each case, three columns are reported: the number of instances where the continuous relaxation bound is equal to the best UB and the average and maximum gap for the cases where they are not equal (calculated as $\frac{\text{best\_UB}-\text{CR\_LB}}{\text{best\_UB}}$). No integer solution can be found for the instance norway l_l, therefore in this case we cannot calculate any gap with the best UB case. By the way, SP cannot even find a CR value for such instance: indeed the instance has the greatest number of demands, thus suggesting that SP may experience some issues with the increasing number of demands.

SP continuous relaxation is equal to the best upper bound in all the uncapacitated instances (h_h). PR continuous relaxation is equal to the best upper bound in all the h_h instances but one, france, where an articulation point is present. In this case, as we proved in Section 4.4.1, PR is weaker than SP. We can infer that for the uncapacitated case, we cannot expect an improvement in terms of bound from the VIs, and the only possible improvement can be in terms of computational time, if any.

Adding VI1 does not increase the number of cases where the CR bound is equal to the best UB, but reduces in general the gap for the cases where they are not equal. Adding VI2 has a larger impact both for the SP and PR formulation: increasing significantly the number of instances where the bound is equal to the optimal value. This is not surprising, since VI2 imposes a lower bound on the number of installed VNFs, that is also a lower bound on the objective function and in the high capacity link case such bound is often tight. Combining VI1 and VI2 reduces further the gap for the cases where the CR bound and the best UB do not coincide.

Table 4.14 shows the average computational times for the continuous relaxation of SP and the PR formulations with and without the addition of VIs. The smallest computational time for each capacity case is reported in bold. The VI2 in general provides very good computational times, with few exceptions. Instead, VI1 always increases the computational times of both formulations, without improving the bound (which already coincides with the integer optimal solution).

Finally, we compare the CR of SP and PR when VIs are added. Figure 4.10 reports the percentage difference between the two continuous relaxation bounds for each VI (calculated as $100\% * \frac{CR_{SP_{VI_i}} - CR_{PR_{VI_i}}}{CR_{SP_{VI_i}}}$). As expected VI2 reduces the difference between the two continuous relaxations only when the service capacity is medium or low, while it provides the naive bound (at least one service) when the service capacity is high. If the service capacity is medium and low, adding VIs

Continuous relaxation bound for the SP formulation												
cap	noVI			VI1			VI2			VI1+2		
	#	gap		#	gap		#	gap		#	gap	
	opt	avg	max	opt	avg	max	opt	avg	max		avg	max
h_h	16	-	-	16	-	-	16	-	-	16	-	-
h_l	2	35.35	50.00	2	33.21	48.93	2	35.35	50.00	2	33.21	48.93
m_h	1	49.2	50.00	1	10.94	16.67	16	-	-	16	-	-
m_l	0	37.79	50.00	0	20.01	42.79	7	28.9	45.03	7	28.53	42.79
l_h	0	87.57	93.33	0	8.08	16.67	16	-	-	16	-	-
l_l	0	78.96	91.67	0	8.37	16.67	15	-	-	15	-	-

Continuous relaxation bound for the PR formulation												
cap	noVI			VI1			VI2			VI1+2		
	#	gap		#	gap		#	gap		#	gap	
	opt	avg	max	opt	avg	max	opt	avg	max		avg	max
h_h	15	50.00	50.00	15	50.00	50.00	15	50.00	50.00	15	50.00	50.00
h_l	2	61.82	80.00	2	49.19	60.60	2	61.82	80.00	2	49.19	60.60
m_h	0	50.00	50.00	0	10.72	16.67	16	-	-	16	-	-
m_l	0	60.00	80.00	0	27.71	59.54	7	43.33	60.00	7	42.81	59.54
l_h	0	88.90	93.33	0	8.08	16.67	16	-	-	16	-	-
l_l	0	88.67	93.33	0	8.37	16.67	15	-	-	15	-	-

Table 4.13 – Comparison of the CR bound with respect of the best UB.

Computational times (s) for the SP formulation continuous relaxation								
cap	noVI		VI1		VI2		VI1+2	
	avg	max	avg	max	avg	max	avg	max
h_h	**0.77**	4.13	0.85	4.64	4.35	26.02	7.4	78.91
h_l	136.67	1502.6	137.83	1603.9	**73.04**	792.81	106.95	1268.05
m_h	**1.35**	7.89	20.62	130.63	5.03	36.63	7.16	56.01
m_l	116.32	1240.33	140.31	1555.4	**64.9**	759.71	76.57	805.44
l_h	10.41	52.35	39.13	278.51	3.37	31.44	**2.86**	19.66
l_l	354.69	3598.88	71.92	597.26	**6.55**	75.62	7.82	78.92

Computational times (s) for the PR formulation continuous relaxation								
cap	noVI		VI1		VI2		VI1+2	
	avg	max	avg	max	avg	max	avg	max
h_h	**5.23**	46.79	9.96	82.29	8.87	85.5	12.14	131.3
h_l	83.13	698.42	163.46	2002.48	66.01	515.51	**30.68**	293.53
m_h	7.24	61.48	43	285.57	**7.00**	51.12	18.11	148.33
m_l	96.27	513.98	143.75	2008.5	**33.88**	273.42	108.31	1125.63
l_h	29.42	134.35	39.46	297.89	**0.79**	4.08	1.19	10.46
l_l	111.91	875.5	61.92	483.14	6.54	72.25	**5.6**	60.22

Table 4.14 – Computational time with and without VIs for SP and PR formulations.

reduces the difference between the continuous relaxations. We can observe that adding VIs reduces the computational time (see Table 4.14) of PR while improving its bound (see Table 4.13), the performance of PR is then similar to the SP one (see Figure 4.10). Nevertheless, SP performs better than PR as SP provides better bounds in m_l case.

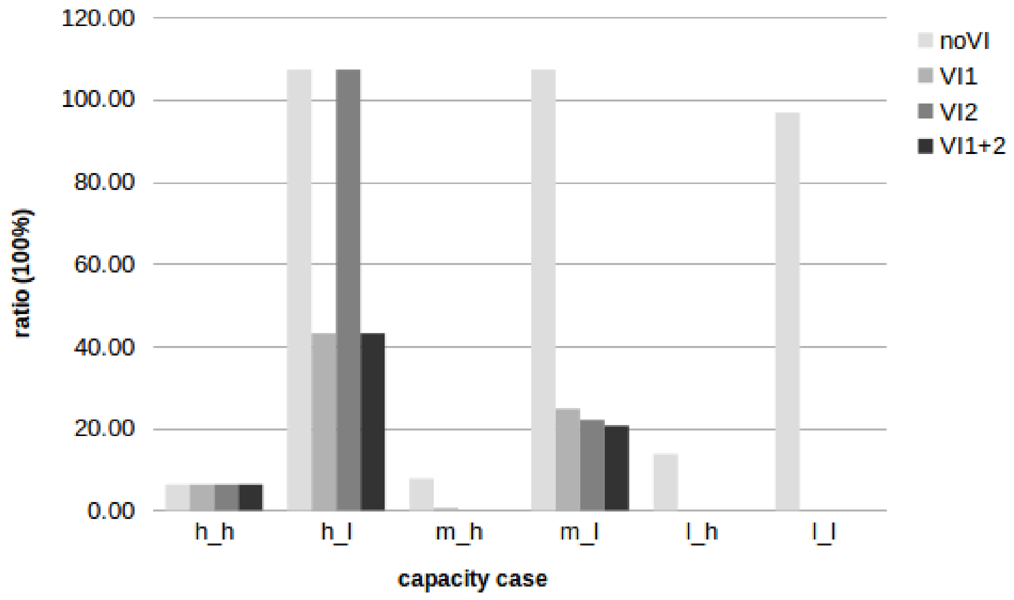
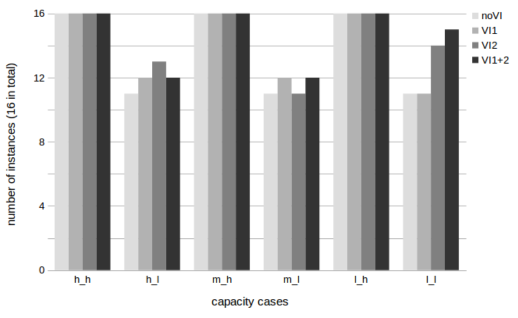


Figure 4.10 – Average percentage difference between SP model and PR model

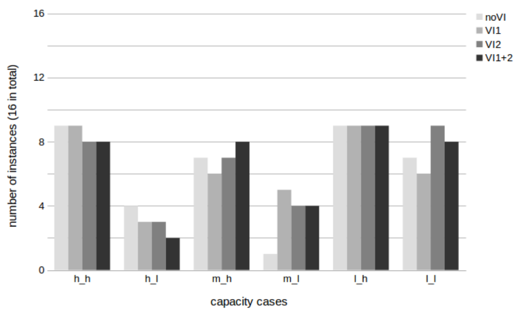
4.5.3 Full model (ILP/MILP) Results on Small-medium Size Instances

Let us now compare the behavior of the two formulations when solving the integer problem. In Figures 4.11a - 4.11c (and respectively Figures 4.11b - 4.11d), the number of optimal and integer feasible solutions found by the SP (respectively PR) formulation are reported.

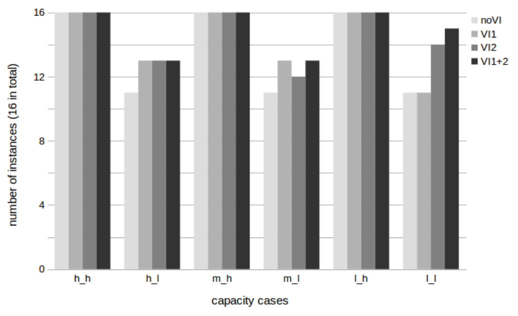
Results show that SP outperforms PR in all the capacity cases, whatever additional inequalities are applied. Indeed, in the high link capacity cases, SP can solve to optimality all the instances with any additional inequalites choice, while PR can solve at most 25 out of 48 instances (the best performance is obtained when no additional inequality is used or both of them are added). In the low link capacity cases, SP can solve more than 30 instances while PR can solve 12 instances with no additional inequalities, 16 instances using only inequality VI2 and 14 instances in the other two cases. SP seems to find the optimal solution almost every time it can find a feasible one: indeed, it is not able to prove the optimality of the feasible solution found only in very few instances with h_l and m_l capacity settings. When SP cannot prove optimality, the gap is still reasonable (at most around 27%



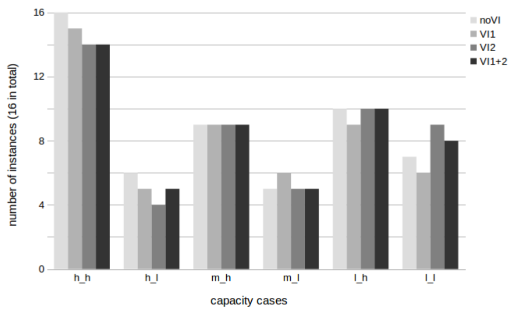
(a) Number of optimal solutions SP.



(b) Number of optimal solutions PR .



(c) Number of feasible solutions SP.



(d) Number of feasible solutions PR .

Figure 4.11 – Number of optimal and integer feasible solutions found

for "janos_us" instance with h_l and m_l capacities). The number of instances where a feasible but not optimal solution is found is very small also for PR. It seems therefore that finding a feasible solution is in a sense as challenging as proving its optimality.

The addition of inequalities improves the number of optimal or feasible solutions found by SP (or leaves them unchanged). Further, it almost always reduces the computational time needed to find the optimal solutions. In general, the impact of VI1 is more evident when the link capacity is low and the service capacity is high. This is reasonable, because with such capacity setting the multicommodity nature of the problem emerges and the capacity of incident links is the one that limits the access to services. Symmetrically, when the service capacity is low, the VI2 shows its effectiveness more clearly. Adding both additional inequalities seems to produce the best results, reducing in almost all cases the total computational time. The impact of adding inequalities on PR is not as clear: it may slightly worsen the performance of PR (for example in the h_l case the number of optimal solutions is reduced).

In Table 4.15 we report about the UB and the LB of each combination of formulation and VIs w.r.t. the best known. Each group of columns refers to a VI case. The following data are reported: in the first column, the number of cases where the bound is equal to the best known, in the second column, the number of cases where the bound cannot be found and in the third column, the average gap with respect to the best known bound (when the gap is not null). The SP formulation with the VI1+2 shows the best performance among all possible variants, producing the best bounds (UB and LB) in the largest number of instances.

SP formulations comparison vs best known UB												
cap	noVI			VI1			VI2			VI1+2		
	best	none	gap	best	none	gap	best	none	gap	best	none	gap
h_h	16	0	-	16	0	-	16	0	-	16	0	-
h_l	11	5	-	13	3	-	13	3	-	13	3	-
m_h	16	0	-	16	0	-	16	0	-	16	0	-
m_l	11	5	-	13	3	-	12	4	-	13	3	-
l_h	16	0	-	16	0	-	16	0	-	16	0	-
l_l	11	5	-	11	5	-	14	2	-	15	1	-

PR formulations comparison vs best known UB												
cap	noVI			VI1			VI2			VI1+2		
	best	none	gap	best	none	gap	best	none	gap	best	none	gap
h_h	9	0	7.00	9	1	4.5	8	2	5.50	8	2	5.00
h_l	4	10	4.50	3	11	3.00	3	12	3.00	2	11	2.00
m_h	7	7	2.50	6	7	2.33	7	7	5.00	8	7	9.00
m_l	2	11	1.33	5	10	1.00	4	11	4.00	4	11	1.00
l_h	9	6	2.00	9	7	-	9	6	1.00	9	6	4.00
l_l	7	9	-	6	10	-	9	7	-	8	8	-

SP formulations comparison vs best known LB												
cap	noVI			VI1			VI2			VI1+2		
	best	none	gap	best	none	gap	best	none	gap	best	none	gap
h_h	16	0	-	16	0	-	16	0	-	16	0	-
h_l	14	0	0.98	14	0	0.54	14	0	0.47	15	0	1.07
m_h	16	0	-	16	0	-	16	0	-	16	0	-
m_l	13	0	0.62	13	0	0.11	15	0	0.33	15	0	0.01
l_h	16	0	-	16	0	-	16	0	-	16	0	-
l_l	12	1	0.80	12	0	0.70	16	1	-	16	0	-

PR formulations comparison vs best known LB												
cap	noVI			VI1			VI2			VI1+2		
	best	none	gap	best	none	gap	best	none	gap	best	none	gap
h_h	15	0	1.00	15	0	1.00	15	0	1.00	15	0	1.00
h_l	6	0	2.22	6	0	1.40	5	0	2.11	4	0	1.26
m_h	8	0	0.37	9	0	0.15	16	0	-	16	0	-
m_l	2	0	1.31	6	0	1.12	7	0	1.57	7	0	1.29
l_h	11	0	0.77	12	0	0.71	16	0	-	16	0	-
l_l	9	0	0.70	8	0	0.60	16	0	-	16	0	-

Table 4.15 – Comparison of UB and LB found for the different formulations (with and without VIs) with respect to the best known UB and LB.

SP outperforms PR also as far as the computational times are concerned. In order to analyze in detail the computational times, we further divide the small-medium size instances into two groups: from “di-yuan” to “france” (up to 300 demands) and from “nobel-eu” to “norway” (more than 300 demands). In fact, instances of the first group can be solved both by SP and PR, while, for instances of the second group, the PR formulation cannot find a feasible solution for all the capacity settings, but h_h.

cap	SP				PR			
	noVI	VI1	VI2	VI1+2	noVI	VI1	VI2	VI1+2
h_h	0.60	0.66	1.16	1.16	860.54	859.12	1156.42	1170.26
h_l	1222.72	1235.26	845.12	870.11	2635.54	3016.69	2854.70	2777.77
m_h	5.50	2.78	1.31	1.07	1712.37	1582.70	1210.71	794.53
m_l	850.41	1289.08	1082.63	800.37	3292.24	2436.50	2371.35	2643.45
l_h	35.80	10.59	3.70	1.52	722.36	431.68	665.01	675.03
l_l	1049.62	1201.86	609.36	487.96	1344.28	1015.80	787.24	841.75

Table 4.16 – Average computational times (in seconds) of SP and PR on the first group of small-medium instances.

In Table 4.16, average computational times (in seconds) on the first group of small-medium size instances are reported for each capacity setting. For this first group of instances, when the link capacity is high, SP can be up to 3 orders of magnitude faster than PR. Low link capacity instances seem to be more challenging both for SP and PR: indeed the increase of SP computational times passing from high link capacity to low link capacity is larger than the one of PR. However, SP remains always significantly faster than PR.

In Table 4.17 the average computational times only of the SP formulation on the second group of small-medium instances are reported (as PR always reaches the time limit for these instances without providing any feasible solution).

cap	SP			
	noVI	VI1	VI2	VI1+2
h_h	6.54	6.69	18.12	19.19
h_l	2199.63	1849.57	1425.26	2226.85
m_h	1111.19	197.95	167.03	31.36
m_l	3265.52	1631.53	1597.95	2171.11
l_h	305.35	148.77	14.74	13.17
l_l	2235.18	2378.76	83.77	109.91

Table 4.17 – Average computational times (in seconds) for of SP on the second group of small-medium size instances.

The increasing number of demands causes an increase of the SP computational times: as for the first group, its computational time depends both on the capacity settings (the high link capacity cases being faster) and on the choice of added inequalities.

4.5.4 Scalability on Large Size Instances

We have considered 6 network topologies, from “cost266” to “pioro40”, that have from 35 to 50 nodes and from about 600 to about 1500 demands, to assess the scalability of the formulations. In these tests the time limit is increased to 7200s. Both the additional inequalities (i.e., VI1+2) have been added.

PR is not able to provide even an integer feasible solution for any of the considered instances, no matter the capacity settings. For some capacity settings, namely h_l and m_l, SP is not able to provide any feasible solution within the time limit, either. Therefore, in Table 4.18 the computational times of SP for the six considered instances are reported, for the capacity settings h_h, m_h, l_h, l_l. SP can solve all of them but one instance with m_l setting, for which a feasible solution is found, and two with l_l, for which not even a feasible solution can be found.

Increasing the network size and the number of demands increases of course the computational times, but SP proves to suffer significantly less than PR. As observed for the small-medium size instances, high link capacity instances are easier than low link capacity ones. The average computational times of SP for the high link capacity cases are acceptable, spreading from about 230 seconds up to about 20 minutes. When both services and links have low capacity the average computational time is about 40 minutes.

In general, when SP can find a feasible integer solution it can also prove its optimality (a similar behavior has been observed for the small-medium size instances). In the m_h giul39 instance, the only one where a feasible, but not optimal, solution is provided, the gap from the best bound is significant (the best bound is 2 whereas the provided solution is 15).

SP	h_h	m_h	l_h	l_l
cost266	501.82	333.79	182.8	TL
germany50	151.97	694.57	70.39	265.93
giul39	975.67	TL*	415.86	2989.96
india35	119.65	65.14	39.62	3286.9
janos-us-ca	885.48	3718.73	610.05	TL
pioro40	302.33	1434.99	83.22	2950.12

Table 4.18 – Computational times of SP formulation for large size instances (TL* means that the time limit has been reached and a feasible solution is found, while TL means that not even a feasible solution has been found within the time limit).

4.5.5 Impact of Proposition 4.3.1

In order to assess the impact of Proposition 4.3.1 we have generated, for three topologies (where, according to the previous test results, both SP and PR can find the optimal solution in h_h case): “abilene”, “atlanta” and “dfn-bwin”, instances with three types of VNF service instead of one. Table 4.19 reports about the computational time needed to solve the single VNF type instances and the three VNF types ones: for each instance, the percentage increase of the computational time when the number of services increases is given, computed as $\frac{CPUtime_3 - CPUtime_1}{CPUtime_1}$, where $CPUtime_\kappa$ is the computational time with κ services (a negative increase represents a reduction of the computational time when tackling a higher number of service types). PR is not able to solve to optimality any instance. When an instance cannot be solved four cases may occur: PR is not able to solve neither the one service type instance nor the three service types one ("A"); PR can solve to optimality the single service type case, but provide only a feasible solution for the three service types case ("B"); PR can solve to optimality the single service type case, but cannot find a feasible solution for the three service types case ("C"); PR can find a feasible solution for the single service type case, but cannot for the three service types case ("D").

	h_h	h_l	m_h	m_l	l_h	l_l
			SP formulation			
abilene	82	138	80	415	38	**-52**
atlanta	136	147	337	**-17**	56	**-89**
dfn-bwin	226	**-1**	174	91	68	94
			PR formulation			
abilene	7027	C	6800	C	12159	145558
atlanta	C	A	D	A	C	C
dfn-bwin	39	D	B	D	4455	C

Table 4.19 – Time increase (in percentage) due to the increase of the number of VNF types (from 1 to 3) (in bold the cases where the overall computational time decreases). "A" denotes the instances for which a feasible solution within the given time limit cannot be found for both one and three service types; "B" denotes the instances where the optimum is found for the single service type but only a feasible solution can be found for the three types case; "C" denotes the instances for which the single service type case can be solved to optimality but no feasible solution is found for the three service type case; "D" denotes the instances for which a feasible solution can be found for the single service type case but no feasible solution is found for the three service types case.

Results show that the property is far from irrelevant, as it allows, in general, to dramatically reduce the computational time. Although for a few instances the computational times of SP formulation may reduce, in general, if the property is neglected the computational times increase up to two orders of magnitude as far as SP is concerned, and up to 5 orders of magnitude as far as PR is concerned. Further, if the property is not applied, the number of instances that PR can solve reduces significantly: 7 out of 15 instances (almost half of them) that PR can solve to optimality with one service type cannot be solved with three service types ("B" and "C") and for all but one no feasible solution can be found with three service types. For further three instances, PR cannot find a feasible solution when the number of service types increases.

It is worth noting that, even if the property is neglected, SP suffers significantly less for the increase of the number of services, although with SP formulation the number of variables depends on the number of service types in each chain.

4.5.6 Articulation Point Based Preprocessing

We tested the articulation point preprocessing on three network topologies of SNDLib which contain at least one articulation point: “france”, “ta2” and “zib54”. In Table 4.20 the computational time variation obtained adding the preprocessing is reported for the capacity cases where an optimal solution can be found for all the three instances by the SP formulation (results are not reported for the PR, as it fails in solving almost all the instances even with preprocessing).

The preprocessing is quite fast (few seconds) and its addition reduces almost always the computational time: thanks to the preprocessing SP solves to optimality, despite their big size, instances for which no feasible solution can be found without preprocessing (“zib54”for the l_l capacity case). Nevertheless, in some capacity cases (all the h_l and m_l cases) no feasible solution can be found even using the preprocessing. Furthermore, the preprocessing is not enough to allow PR to find a solution in most of the cases (the only exception being the france network, where the solution can be certified for the uncapacitated case). To conclude, as it needs negligible computational time and is beneficial for some instances, although not for all, we believe that the articulation point based preprocessing is worth performing.

cap	h_h	m_h	l_h
france	-32.0	-26.0	**153.0**
ta2	-73.0	-93.0	-6.0
zib54	-37.0	-84.0	-36.0

Table 4.20 – Time reduction (in percentage) obtained by using articulation point preprocessing (in bold the only case where the overall computational time increases) for the SP formulation.

4.6 Conclusion

In this chapter, we formally defined the VNF-PR$_{\text{SP}}$ problem and studied some problem complexity and properties. Furthermore, we compared the most promising formulation strategies (i.e., SP and PR) and analyzed their performance on a common test bed. The first modeling strategy (SP) is based on splitting each demand path into two sub-paths, one connecting the source with the service, the second one connecting the service with the destination; the second one (PR) uses arc flow variables and forbids cycles exploiting node labels. Enriched formulations were also proposed and their impact evaluated. We proved that the continuous relaxation of SP always provides a bound not worse than the one of PR. As expected from the theoretical results, computational tests show that stand alone SP improves upon PR, therefore, SP benefits more from the VIs and seems to suffer less from the increasing number of demands. In addition, we observed that, for some hard and large size instances, either SP is able to find an optimal solution or it is not even able to provide a feasible solution. Thus, efficient methods are needed and the problem of finding a feasible solution is worth investigating.

Chapter 5

ILP-based Constructive Approaches

Contents

5.1 ILP-based Constructive Methods 111

5.1.1 SP Formulation with Bin-Packing-like Improved Bound . . 111

5.1.2 Three-Step Bin Packing Based Fix-and-Solve Method . . . 113

5.1.3 Fix-and-Check Method 116

5.2 Computational Results 120

5.2.1 Results on Small-medium Size Instances 121

5.3 Conclusion 123

In previous chapters, we formally defined the VNF-PR$_{\mathrm{SP}}$ problem and provided a thorough complexity and properties analysis. Furthermore, we proposed two formulations that proved capable of coping with small and medium size instances. Computational results of the proposed models showed that small and medium size instances of the VNF-PR$_{\mathrm{SP}}$ problem can be solved efficiently using the CPLEX solver. To further improve the resolution, we develop several constructive approaches that combine mathematical programming with fast search engines (i.e., by fixing partial solutions).

The remaining of this chapter is organized as follows. In Section 5.1 we propose two ILP-based constructive methods for VNF-PR$_{\mathrm{SP}}$. In Section 5.2 we present and analyze the results of computational experiments of the proposed methods.

5.1 ILP-based Constructive Methods

In this section we propose two ILP-based constructive methods. In brief, the main idea of both methods is to partially fix the solution and complete it by solving an ILP model, which is an improved SP formulation.

5.1.1 SP Formulation with Bin-Packing-like Improved Bound

The SP formulation with Bin-Packing-like improved bound is based on SP formulation with VI1+2, as it shows the best performance among all possible variants (see Section 4.5). The improvement takes inspiration from the problem features of VNF-PR. As discussed in section 2.1, our problem shares features with network design problems (for the demand routing) and with facility location problems (for the VNF location). If we discard the routing part, computing the minimum number of VNF instances turns out to be a bin packing problem whose optimal solution always provides a lower bound of the VNF-PR$_{\text{SP}}$.

Compared with the calculated lower bound used in VI2 (equation 4.21), the lower bound provided by the BPP (integer) solution is always not worse than the former, as BPP solution considers the demand assignment in addition to service capacity. Moreover, the lower bound provided by the BPP solution can be further improved if we consider in the BPP the overall capacity of links incident in the node as in VI1, i.e., using the computed service capacity $\bar{q}_i$ (the calculated maximal demand amount that can be served by a node) instead of q. Therefore, in order to compute such lower bound, we solve the following BPP-like sub-problem of the VNF-PR$_{\text{SP}}$ problem:

BPP-like.

$$\min \sum_{i \in N} y_i \tag{5.1}$$

$$s.t. \sum_{i \in N} z_i^k = 1 \qquad \forall k \in D \tag{5.2}$$

$$\sum_{k \in D} d_k \; z_i^k \leq \bar{q}_i \; y_i \qquad \forall i \in N \tag{5.3}$$

The BPP-like objective (5.1) is to minimize the total number of used bins (VNF instances). Constraints (5.2) ensure that each item (demand) is packed into exactly one bin (VNF instance), while constraints (5.3) impose that the capacity of any used bin is not exceeded. Different from the typical BPP formulation, in our BPP-like formulation, we consider different capacity $\bar{q}_i$ for each bin i.

Solving the BPP-like sub-problem is in general fast (less than 1 second for small-medium size instances) on the considered instances. The optimal solution of the BPP-like is then used as a cut-off lower bound (denoted with LB_{bpp}) in the formulation:

SP with improved bound (SPI):

$$obj. \; (4.1)$$
$$s.t. \; (4.2), (4.3), (4.5), (4.7) - (4.10), (4.20)$$
$$\sum_{i \in N} y_i \; \geq \; LB_{bpp} \qquad \forall i \in N \tag{5.4}$$

Similar to the original SP formulation, the objective (4.1) of SPI is to minimize the total number of used VNF instances. Constraints (4.2), (4.3), (4.5), and (4.7)-(4.10) are taken from the original SP formulation to formulate the VNF-PR$_{\text{SP}}$ problem. Constraints (4.20) is the VI1 introduced previously to strengthen the service capacity formulation. SPI differs from SP with VI1+2 for that it uses the lower bound provided by the BPP-like solution to force the number of selected VNF instances to b at least equal to the optimal number of bins.

5.1.2 Three-Step Bin Packing Based Fix-and-Solve Method

In this section, we introduce the Three-Step Bin Packing based Fix-and-Solve (TS-BP-FS) method for the VNF-PR$_{\text{SP}}$ problem.

We first compute the optimal solution of the BPP-like to obtain the input parameters. Besides the problem solution that is used as the lower bound in SPI, the BPP-like provides also *clusters* (namely, groups of demands that use the same VNF instance) that are used in TS-BP-FS. Then, we perform three steps, in each a model is solved where additional constraints are added so as to partially fix the solution. In brief, in Step1, we solve the VNF-PR$_{\text{SP}}$ problem fixing placement and assignment variables as in the optimal solution of BPP. If a feasible routing solution is found, the procedure stops. The solution found is an optimal solution for VNF-PR$_{\text{SP}}$, as it uses the number of VNF instances provided by the optimal solution of the BPP-like which provides a lower bound for the VNF-PR$_{\text{SP}}$ problem. Otherwise, in Step2 we solve VNF-PR$_{\text{SP}}$ keeping the assignment part of the BPP-like solution (clusters) while allowing chaining VNF instance locations. If a feasible routing solution is found, we stop the procedure. The solution found is still an optimal solution for VNF-PR$_{\text{SP}}$, as the number of installed VNF instances is still equal to the BPP-like. Otherwise, in Step3 we allow assigning the demands of one *cluster* to two VNF instances. As the number of installed instances maybe twice the lower bound, the solution provided in Step3 is not optimal (however it has a guaranteed error of 100%).

Notation		Step
Sets		
C	set of *Clusters*	1,2,3
D_c	set of demands that belongs to cluster $c \in C$	1,2,3
Parameters		
LB_{bpp}	the optimal solution of the BPP-like	1,2,3
vnf_c	the VNF node that is used by cluster $c \in C$	1
Variables (binary)		
w_i^c	1 if cluster $c \in C$ uses the VNF instance on node $i \in N$	3

Table 5.1 – Additional mathematical notation used in method TS-BP-FS.

In Table 5.1 the additional notation used in TS-BP-FS is summarized and in which step the additional parameters/variables are used are reported. (basic mathematical notation of SP formulation is reported in Table 4.5)

We now describe the procedure more into detail.

1. **Step 1.**

   The pure routing problem is solved while keeping location of VNF instances and assignment of the BPP-like solution:

SPI$_{\text{rout}}$:

$$\begin{aligned}
&obj.\ (4.1)\\
&s.t.\ (4.2),(4.3),(4.5),(4.7)-(4.10),(4.20)\\
&s.t.\ (5.4)\\
&z_i^k \ = \ 1 \qquad \forall c \in C, k \in D_c, i \in N : vnf_c = i \qquad (5.5)
\end{aligned}$$

SPI$_{\text{rout}}$[7] is based on SPI formulation. Additional constraints (5.5) are introduced to guarantee that each demand is assigned to a VNF instance according to the BPP-like solution.

If SPI$_{\text{rout}}$ finds a feasible routing, it is then an optimal one. While SP$_{\text{rout}}$ being not feasible can be due to different reasons:

- the total number of VNF instances is not enough to serve all the demands
- the total number of VNF instances is enough, but the selected location and/or assignments are not feasible

Thus in Step2 and Step3, we allow changing more the BPP-like solution to "correct" these two problems.

2. **Step 2.**

In this step, we allow changing the location of the VNF instances, but we keep the same number of VNF instances and the same clusters as in the BPP-like solution. We allow the demands of a cluster to use a VNF instance different from the one selected by the BPP-like solution. However, the demands of a cluster must select the same VNF instance.

SPI$_{\text{ass}}$:

$$\begin{aligned}
&obj.\ (4.1)\\
&s.t.\ (4.2),(4.3),(4.5),(4.7)-(4.10),(4.20)\\
&s.t.\ (5.4)\\
&z_i^{k1} \ = \ z_i^{k2} \qquad \forall c \in C, k1 \in D_c, k2 \in D_c, i \in N \qquad (5.6)
\end{aligned}$$

Additional constraints (5.6) impose that the demands of one cluster are assigned to the same VNF instance.

If SPI$_{\text{ass}}$ finds a feasible solution, it is an optimal solution of the VNF-PR$_{\text{SP}}$ problem. Otherwise, either the total number of VNF instances is not enough,

[7]When we fix the location and assignment in the solution, objective function (4.1) and constraints (4.2), (4.3) and (5.4) are in fact redundant. However, preliminary tests on small-medium size instances show that the SPI$_{\text{rout}}$ formulation with these redundant constraints performs a little faster than the variants without, therefore, we decide to keep these constraints in SPI$_{\text{rout}}$.

	Objective function and basic constraints		Additional constraints		
Models	obj	routing/location	fix location	change location	split assignment
SP-VI1+2	(4.1)	(4.2),(4.3),(4.5), [(4.7)-(4.10)],(4.20),(4.21)			
SPI	(4.1)	(4.2),(4.3),(4.5), [(4.7)-(4.10)],(4.20),(5.4)			
SPI$_{\text{rout}}$	(4.1)	(4.2),(4.3),(4.5), [(4.7)-(4.10)],(4.20),(5.4)	(5.5)		
SPI$_{\text{ass}}$	(4.1)	(4.2),(4.3),(4.5), [(4.7)-(4.10)],(4.20),(5.4)		(5.6)	
SPI$_{\text{split}}$	(4.1)	(4.2),(4.3),(4.5), [(4.7)-(4.10)],(4.20),(5.4)			(5.7),(5.8)

Table 5.2 – Applicable constraints to the models used in TS-BP-FS.

or the number of VNF instances is correct but the selected clusters are not feasible.

3. **Step 3.**

   In this step, we give more flexibility to the model by allowing the demands in one cluster to use two VNF instances. More precisely, we allow changing the location of the VNF instances and using up to twice the number of VNF instances with respect to the optimal solution provided by BPP-like (i.e., $2 * LB_{bpp}$).

   SPI$_{\text{split}}$:

$$\begin{aligned}
&obj.\ (4.1)\\
&s.t.\ (4.2),(4.3),(4.5),(4.7)-(4.10),(4.20)\\
&s.t.\ (5.4)\\
&\sum_{i\in N} w_i^c \ \leq\ 2 \qquad \forall c \in C && (5.7)\\
&z_i^k \ \leq\ w_i^c \qquad \forall c \in C, k \in D_c, i \in N && (5.8)
\end{aligned}$$

   Additional constraints (5.7) allow each cluster to use two VNF instances, and (5.8) ensure that if a cluster does not use the VNF instance on a node, then no demand of this cluster can be assigned to this node.

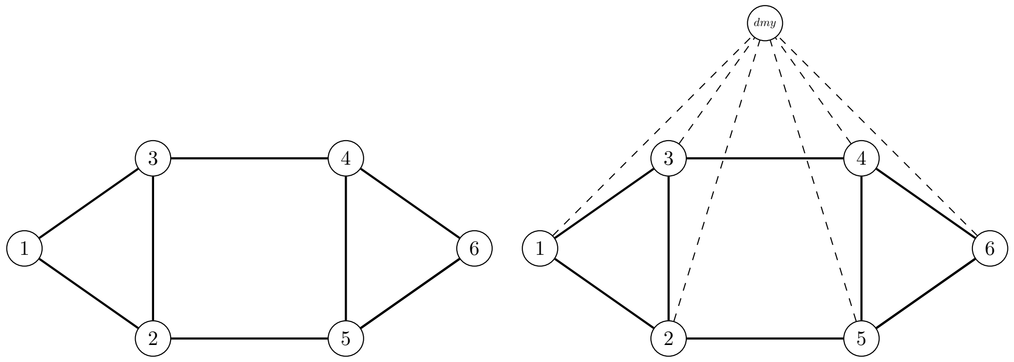
   If SPI$_{\text{split}}$ finds a feasible solution with the used number of VNFs instances equal to LB_{bpp}, then it is an optimal solution for the VNF-PR$_{\text{SP}}$ problem. Otherwise, any feasible solution found by SPI$_{\text{split}}$ with the used number of VNFs instances that is greater than LB_{bpp}, provides an upper bound, and the optimality gap is guaranteed to be not worse than twice of the optimal solution.

The models used in each step of TS-BP-FS are reported in Table 5.2. If a feasible solution is found, the procedure stops. The first two steps lead to an optimal solution, whereas the last one always leads to a solution with a gap guarantee of twice the optimal solution.

5.1.3 Fix-and-Check Method

In this section, we introduce the second constructive method. In short, the main idea of this method is to *fix* the number of installed VNF instances and *check* if a feasible solution could be found. If a proposed number of VNF instances is not enough to serve all the demands, we increase this number by one and continue the fix and check procedure.

In this fix-and-check (FC) method, in order to be able to check if a number of VNF instances is enough to serve all the demands, we propose to solve the problem of *maximizing* the number of served *demands*, the VNF-PR$_{\text{SP}}{}^{\text{MD}}$ problem.



(a) Example graph. (b) Extended graph of the example.

Figure 5.1 – The graph extension operated by model SP$^{\text{MD}}$.

VNF-PR$_{\text{SP}}{}^{\text{MD}}$ differs from VNF-PR$_{\text{SP}}$ for the objective function that is to maximize the number of served demands. To allow finding always a feasible solution, the network graph is extended with a dummy node *dmy* allowing installing a VNF instance that can serve all the demands not served elsewhere. Figure 5.1 shows an example of this extended graph, the uncapacitated dummy node is added to the original graph, and an uncapacitated arc is added from every demand source node to the node *dmy* and from the node *dmy* to every demand destination node, so that any demand can be assigned to the node *dmy* and routed on these uncapacitated arcs. The problem goal is to maximize the demands that are served by any VNF node but *dmy*. The resulting graph is denoted as $G^{\text{MD}}(N^{\text{MD}}, A^{\text{MD}})$, where $N^{\text{MD}} = N \cup dmy$ and A^{MD} is the union of the original arcs A and the arcs incident in the node *dmy*. Then the location, assignment and routing variables are defined on the resulting graph. To solve the VNF-PR$_{\text{SP}}{}^{\text{MD}}$ problem, we solve the following model which adopts the split-path formulation strategy and uses similar constraints as in the SP formulation with VI1 (equation (4.20)):

SP$^{\text{MD}}$-VI1.

$$\max \sum_{k \in D, i \in N} z_i^k \tag{5.9}$$

$$s.t. \sum_{i \in N:(i,dmy) \in A^{\text{MD}}} (x_{i,dmy}^{k1} + x_{i,dmy}^{k2}) \leq z_{dmy}^k \quad \forall k \in D \tag{5.10}$$

$Eq.$ (4.3), (4.5), (4.9), (4.10), (4.20) are defined on the original graph $G(N, A)$

$Eq.$ (4.2), (4.7), (4.8) are defined on the extended graph $G^{\text{MD}}(N^{\text{MD}}, A^{\text{MD}})$

The objective function (5.9) maximizes the number of demands that use the real VNF instance. Constraints (5.10) are introduced to forbid the served demands passing by the dummy links if they are not using the VNF instance located on the dummy node.

In brief, the FC procedure is a loop algorithm including three main steps. In each iteration of FC, the number of installed VNF instances is fixed and three steps are executed one after another. Each step fixes partially the solution and solves the ILP model to complete the solution. The flexibility of the model is increased from one step to another: at the beginning of each iteration, the BPP-like is first solved with a minimum number of bins (i.e., the fixed number of VNF instances) to provide the clusters (i.e., VNF location and demand assignment solution). In Step1, we solve the problem fixing placement and assignment variables as in the optimal solution of BPP-like. If all the demands can be served, the procedure stops. The fixed number of installed VNF instances is a feasible solution of the VNF-PR$_{\text{SP}}$ problem. Otherwise, in Step2 we only keep the location part of the BPP-like solution while allowing changing the assignment of the demands. If all the demands can be served, a feasible solution of the VNF-PR$_{\text{SP}}$ problem is found, and the procedure stops. Otherwise, in Step3 we allow free location and assignment, but we fix the number of installed VNF instances. If all the demands can be served, we stop the procedure. Otherwise, we increase the number of VNF instances by one and go to the next iteration.

Notation		Method
	Sets	
B	set of installed VNF instances	FC
C	set of clusters	both
D_c	set of demands that belong to cluster $c \in C$	both
	Parameters	
LB_{fix}	the number of VNF instances to open, 1 by default	FC
LB_{bpp}	the optimal solution found by BP	both
vnf_c	the VNF node that is used by cluster $c \in C$	both

Table 5.3 – Additional mathematical notation used in method FC.

In Table 5.3 the additional notation used in FC is presented. Since FC shares some same mathematical notation as in the method TS-BP-FS, in the last column we report the method where the set/parameter is used.

We now describe the loop algorithm of FC more into detail. A comprehensive view of the method is shown in Algorithm 1.

Since in each iteration of FC the number of VNF is fixed, to provide locations and assignments to start from, a BPP-like problem with a lower bound on the number of bins is solved:

The bounded BPP-like (BPP$_{\text{bd}}$-like):

$$\begin{aligned} &obj.\ (5.1)\\ &s.t.\ (5.2), (5.3)\\ &\sum_{i \in N} y_i \ \geq \ LB_{fix} \end{aligned} \tag{5.11}$$

Additional constraints (5.11) are introduced to impose the number of VNF instances to install to be at least LB_{fix}.

Let LB_{bpp} be the number of VNF instances provided by BPP$_{\text{bd}}$-like (LB_{bpp} equals to LB_{fix}, except for the first iteration). The clusters are built according to the assignment provided by the BPP$_{\text{bd}}$-like solution. The selected VNF locations are kept in the set B, and described by parameters vnf_c for each cluster $c \in C$.

1. **Step 1.**

   The goal of Step1 is to verify if a feasible routing can be found for all the demands while fixing the location solution as in the BPP$_{\text{bd}}$-like solution. Each demand can only use either the dummy VNF instance or the one that is allocated to it in the BPP$_{\text{bd}}$-like solution. To this end, the following model is solved:

   SP$_{\text{rout}}{}^{\text{MD}}$-VI1:

   $$\begin{aligned} &obj.\ (5.9)\\ &s.t.\ (5.10)\\ &s.t.\ (4.3), (4.5), (4.9), (4.10), (4.20) \text{ are defined on the original graph } G(N, A)\\ &s.t.\ (4.2), (4.7), (4.8) \text{ are defined on the extended graph } G^{\text{MD}}(N^{\text{MD}}, A^{\text{MD}}) \end{aligned}$$

   $$y_i \ = \begin{cases} 1 \text{ if } i \in B \\ 0 \text{ otherwise} \end{cases} \quad \forall i \in N \tag{5.12}$$

   $$z_i^k \ = \ 0 \quad \forall c \in C, k \in D_c, i \in N : vnf_c \neq i \tag{5.13}$$

   SP$_{\text{rout}}{}^{\text{MD}}$-VI1 is based on SP$^{\text{MD}}$-VI1. Additional constraints (5.12) and (5.13) are introduced to fix partially the solution: constraints (5.12) fix the location of VNF instances as in the BPP$_{\text{bd}}$-like solution, while constraints (5.13) fix the assignment solution by forbidding the demand to use any other VNF instance (except for the dummy one) that is not assigned to the demand's cluster.

If the optimal solution found by $\text{SP}_{\text{rout}}{}^{\text{MD}}$-VI1 equals to the total number of demands, then the fixed number of VNF instances is a feasible solution to the VNF-PR_{SP} problem.

2. **Step 2.**

   In this step, we check if the fixed number of VNF instances is sufficient to serve all the demands by allowing the demands to be assigned to any open VNF instances regardless the clusters. To do so, we remove the additional constraints (5.13) from $\text{SP}_{\text{rout}}{}^{\text{MD}}$-VI1, and we solve the following model:

   $\text{SP}_{\text{fixloc}}{}^{\text{MD}}$-VI1:

   $$\begin{aligned}
   &\textit{obj.}\ (5.9)\\
   &\textit{s.t.}\ (5.10)\\
   &\textit{s.t.}\ (4.3),(4.5),(4.9),(4.10),(4.20) \text{ are defined on the original graph } G(N,A)\\
   &\textit{s.t.}\ (4.2),(4.7),(4.8) \text{ are defined on the extended graph } G^{\text{MD}}(N^{\text{MD}},A^{\text{MD}})\\
   &\textit{s.t.}\ (5.12)
   \end{aligned}$$

   If $\text{SP}_{\text{fixloc}}{}^{\text{MD}}$-VI1 finds a feasible routing such that all the demands are served by real VNF instances, a feasible solution of VNF-PR_{SP} is then provided and we stop the procedure. Otherwise, either the number of VNF instances is not enough to serve all the demands, or the number is sufficient but the location is not feasible.

3. **Step 3.**

   In this step, we fix the number of VNF but allow changing the location of VNF instances by solving the following model:

   $\text{SP}_{\text{fixnum}}{}^{\text{MD}}$-VI1:

   $$\begin{aligned}
   &\textit{obj.}\ (5.9)\\
   &\textit{s.t.}\ (5.10)\\
   &\textit{s.t.}\ (4.3),(4.5),(4.9),(4.10),(4.20) \text{ are defined on the original graph } G(N,A)\\
   &\textit{s.t.}\ (4.2),(4.7),(4.8) \text{ are defined on the extended graph } G^{\text{MD}}(N^{\text{MD}},A^{\text{MD}})\\
   &\sum_{i\in N} y_i \;=\; LB_{fix} \qquad (5.14)
   \end{aligned}$$

   If the number of served demands in the optimal solution found by $\text{SP}_{\text{fixnum}}{}^{\text{MD}}$-VI1 equals to the total number of demands, a feasible solution of VNF-PR_{SP}

is built. Otherwise, if the number of served demands in the optimal solution is smaller than the number of demands, the given number of VNF instances is a lower bound for the VNF-PR$_{\mathrm{SP}}$ problem. In this case, we increase the number of VNF instances and we repeat the main steps, until we find a feasible solution for the VNF-PR$_{\mathrm{SP}}$ problem or the number of VNF instances reaches the number of physical nodes.

	Basic features		Additional constraints		
Models	obj	constraints	fix location	change assignment	fix number
SP$^{\mathrm{MD}}$-VI1	(5.9)	(4.2), (4.3), (4.5), (4.7)-(4.10), (4.20), (5.10)			
SP$_{\mathrm{rout}}{}^{\mathrm{MD}}$-VI1	(5.9)	(4.2), (4.3), (4.5), (4.7)-(4.10), (4.20), (5.10)	(5.12), (5.13)		
SP$_{\mathrm{fixloc}}{}^{\mathrm{MD}}$-VI1	(5.9)	(4.2), (4.3), (4.5), (4.7)-(4.10), (4.20), (5.10)		(5.12)	
SP$_{\mathrm{fixnum}}{}^{\mathrm{MD}}$-VI1	(5.9)	(4.2), (4.3), (4.5), (4.7)-(4.10), (4.20), (5.10)			(5.14)

Table 5.4 – Applicable constraints to models used in FC.

In Table 5.4 the models used in each step are reported. The purpose of the method FC is to check if a proposed number of VNF instances can serve all the demands and increase the number if not. The check is performed iteratively in three steps after finding the BP solution with at least the given number of VNF instances: first a feasible routing is searched with fixed location and assignment solution. Then if a feasible routing cannot be found, the VNF locations are kept but a new assignment and routing is searched. Finally, only the number of VNFs is fixed.

In order to improve the performance of the method and iterate quickly, in its implementation (see Algorithm 1), we solve the relaxation problem of the corresponding model before solving the integer one. As the relaxation provides an upper bound to the optimal integer solution, if the relaxation solution is smaller than the total number of demands, we can move to the next step. Furthermore, we first solve the relaxation problem of SP$_{\mathrm{fixnum}}{}^{\mathrm{MD}}$-VI1, since if the relaxation solution of SP$_{\mathrm{fixnum}}{}^{\mathrm{MD}}$-VI1 is smaller than the total number of demands, we can increase the number of installed VNF instances by one and move to the next iteration.

5.2 Computational Results

For the sake of comparison and completeness, the computational tests for the proposed constructive methods are run on the small-medium size instances of the test bed generated previously (Table 4.10 in Chapter 4) with a time limit of 3600s.

Algorithm 1 Overview of the method *FC*

```
1: let $LB_{fix} \leftarrow 1$
2: while time limit is not exceeded & $LB_{fix} \leq |N|$ do
3:     procedure THE BOUNDED BPP-LIKE
4:         solve $BPP_{bd}$-like
5:         initialize sets $B$, $C$, $D_c$ and parameters $vnf_c$
6:     procedure FIX-AND-CHECK
7:         solve the relaxation of ${SP_{fixnum}}^{MD}$-VI1
8:         let $res \leftarrow$ the optimal solution
9:         if $res + 0.01 \geq |D|$ then
10:            solve the relaxation of ${SP_{rout}}^{MD}$-VI1
11:            let $res \leftarrow$ the optimal solution
12:            if $res + 0.01 \geq |D|$ then
13:                solve ${SP_{rout}}^{MD}$-VI1
14:                let $res \leftarrow$ the optimal solution
15:                if problem solved & $res = |D|$ then
16:                    stop: a feasible solution to VNF-$PR_{SP}$ is found
17:            solve the relaxation of ${SP_{fixloc}}^{MD}$-VI1
18:            let $res \leftarrow$ the optimal solution
19:            if $res + 0.01 \geq |D|$ then
20:                solve ${SP_{fixloc}}^{MD}$-VI1
21:                let $res \leftarrow$ the optimal solution
22:                if problem solved & $res = |D|$ then
23:                    stop: a feasible solution to VNF-$PR_{SP}$ is found
24:            solve ${SP_{fixnum}}^{MD}$-VI1
25:            let $res \leftarrow$ the optimal solution
26:            if problem solved & $res = |D|$ then
27:                stop: a feasible solution to VNF-$PR_{SP}$ is found
28:     let $LB_{fix} = LB_{fix} + 1$
```

5.2.1 Results on Small-medium Size Instances

We first present the results of TS-BP-FS in comparison with the SP-VI1+2 formulation as a reference. In Table 5.5, the number of optimal solutions found by TS-BP-FS and SP-VI1+2 on the small-medium size instances as well as the average computational times are reported for each capacity setting. The first column indicates the capacity case. Column 2 report the number of optimal solutions found by SP formulation with VI1+2 while column 6 reports it for the TS-BP-FS. Columns 3-5 summarize the number of optimal solution found in each step of method TS-BP-FS. Columns 7-8 report the average computational times when the optimallity is proved by both SP-VI1+2 and TS-BP-FS (the cases where the time limit is reached are not considered). In column 9 we report the number of instances that TS-BP-FS uses less computational time than SP-VI1+2 to find the optimal solution and in column 10 we report the average reduction of time of these instances.

We observe that, for each capacity setting, when TS-BP-FS is able to find the optimal solution, SP-VI1+2 also finds the optimal solution, while the opposite is not true. For example, low link capacity instances are challenging for SP-VI1+2, but TS-BP-FS suffers more than SP-VI1+2: SP-VI1+2 solves 38 instances out of 48 instances to optimality while TS-BP-FS can only solve 24 instances. As for

cap	# opt					Avg computational times (s)		Time reduction w.r.t. SP-VI1+2	
	SP-VI1+2	TS-BP-FS							
		Step1	Step2	Step3	Total	SP-VI1+2	TS-BP-FS	#	% avg
h_h	16	16	0	0	16	6.5	1.4	16	62.34
h_l*	11	1	1	5	7	150.70	97.36	5	58.14
m_h	16	16	0	0	16	8.74	1.64	16	60.79
m_l*	12	3	2	2	7	246.42	15.42	5	68.91
l_h	16	16	0	0	16	6.24	1.56	16	69.36
l_l	15	3	3	4	10	33.74	422.70	4	53.97

Table 5.5 – Number of optimal solutions found by TS-BP-FS and SP-VI1+2 and the average computational times used by each for test on small-medium size instances with time limit of 3600s (* the instance janos-us is not counted in the number of optimal solutions found by SP-VI1+2 in both h_l and m_l cases, the solution found is in fact an optimal solution, but the optimality is not proved within time limit).

the computational times in low link capacity cases, TS-BP-FS only reduces the computational time of SP-VI1+2 for 14 instances among 24 instances. But TS-BP-FS performs well when the link capacity is high: compared with SP-VI1+2, TS-BP-FS finds optimal solution with less computational times for all the 48 instances in high link capacity cases, and the average time reduction is at least 60%.

Both SP-VI1+2 and TS-BP-FS can find the optimal solution with very short computational time (several seconds) for all the small-medium size instances of high link capacity cases. Therefore, we decide to eliminate the tests on small-medium size instances of high link capacity cases for the rest of tests on constructive and heuristic methods.

We now report the results of the FC method. As observed previously, the low link capacity instances are challenging for the model: either SP-VI1+2 is able to find an optimal solution with large computational time or it is not even able to provide a feasible solution. Thus, we decide to compare FC with SP-VI1+2 from two aspects: for the instances that SP-VI1+2 can provide an optimal solution with large computational time, we check if FC can reduce the computational time; for those that SP-VI1+2 fails to provide a feasible solution, we check if FC can find a feasible solution.

In Table 5.6, the number of optimal solutions found by FC and SP-VI1+2 on small-medium size instances as well as the average times are reported for each low link capacity setting. The first column is the capacity case. Columns 2-4 summarize the number of optimal solutions found by SP-VI1+2 and FC and both of them. Columns 5-6 report the average computational times when the optimality is proved by both SP-VI1+2 and FC. Column 7 reports the number of instances that FC can solve to optimality while SP-VI1+2 cannot provide any feasible solutions. Columns 8 summarizes the number of instances where FC uses less computational time than SP-VI1+2 to find the optimal solution (counting instances where both methods can find the optimal solution) and column 9 reports the average reduction of time for these instances. On the contrary, Columns 10 and 11 summarizes the number of instances and the average augmentation of time in cases where FC uses more computational time than SP-VI1+2 to find the optimal solution.

cap	# opt			Avg computational times (s)		Performance w.r.t SP-VI1+2				
	SP-VI1+2	FC	Both	SP-VI1+2	FC	Impr # opt	Time reduction #	Time reduction % avg	Time augmentation #	Time augmentation % avg
h_l*	11	11	10	497.10	350.02	1	6	46.90	4	28.53
m_l*	12	11	11	491.33	492.04	0	3	79.58	8	414.29
l_l	15	13	13	63.61	242.38	0	5	40.73	8	283.60

Table 5.6 – Number of optimal solutions found by FC and SP-VI1+2 and the average computational times used by each for test on small-medium size and low link capacity instances with time limit of 3600s (* the instance janos-us is not counted in the number of optimal solutions found by SP-VI1+2 in both h_l and m_l cases, the solution found is in fact an optimal solution, but the optimality is not proved within time limit).

In h_l case, for the 10 instances[8] that both SP-VI1+2 and FC can solve to optimality, the average computational time used by FC is smaller than SP-VI1+2. Moreover, FC uses less computational time than SP-VI1+2 for 6 instances with the averaged time reduction of 46%. Then for the instances that SP-VI1+2 cannot provide any feasible solution, FC can find an optimal solution for only one instance (polska). But there is one instance (nobel-us) that SP-VI1+2 can find an optimal solution whereas FC cannot provide any feasible solution within time limit of 3600s. Therefore, the behaviour of method FC is similar to that of SP-VI1+2 in case h_l. Then the performance of FC becomes worse than SP-VI1+2 in m_l and l_l cases: FC finds optimal solution for 3 less instances than SP-VI1+2 does. Moreover, for all the instances that FC can solve to optimality, SP-VI1+2 is able to find an optimal solution also. As for the average computational time for solving these instances, FC uses more computational time than SP-VI1+2 in the majority of cases.

5.3 Conclusion

In this chapter, we presented two ILP-based constructive methods to deal with the VNF-PR$_{SP}$ problem. Both of them are based on ILP model. The TS-BP-FS method allows us to search quickly a feasible routing with a set of models by fixing location and/or assignment solutions provided by the BPP-like solution. This method works well when the BPP-like sub-problem dominates the VNF-PR$_{SP}$ problem, e.g., the optimal solution of VNF-PR$_{SP}$ is exact the number of VNF instances provided by the BP solution. However, this method cannot find efficiently feasible solutions in cases where the routing part dominates the VNF-PR$_{SP}$ problem (e.g., low link capacity cases). To handle this, we came up with the FC method. FC allows us to check if a

[8]in this case, both SP-VI1+2 and FC finds optimal solution for 11 instances, but 10 instances in common.

proposed number of installed VNF instances is a feasible solution to VNF-PR$_{\text{SP}}$ by searching the maximum number of demands it can serve. The check is performed iteratively by solving a set of models with partially fixed solution. FC is able to find a feasible solution for the network instances under low link capacity cases. However, the computational time is expensive with hard and large size instances. Therefore, the problem of finding a feasible solution is worth investigating.

Chapter 6

ILP-based Heuristic Approaches

Contents

6.1 Overview of *k-opt* Technique and *k-opt* Models 126
6.1.1 *k-opt neighborhood search* Technique 126
6.1.2 *k-opt* Models 127
6.2 Computing An Initial Solution 128
6.3 Improving Local Search (LS) Based Heuristics 131
6.4 Fix-and-Optimize Heuristics 131
6.5 Computational Results 133
6.5.1 Results of Approaches for Obtaining An Initial Solution . . 133
6.5.2 Results of Heuristic Methods 136
6.6 Conclusion 142

In this chapter, we present several heuristic approaches to deal with large size instances of the problem. As discussed previously, it is too time-consuming both for the models and the proposed constructive methods to solve to optimality large size instances, and sometimes they have difficulty even in providing a feasible solution. Therefore, we propose heuristic methods to tackle the large size instances within reasonable computational time.

The remaining of this chapter is organized as follows. In Section 6.1 we provide an overview of the *k-opt neighborhood search* technique and we describe three resulting models that consider different neighborhoods for our problems. Then in Section 6.2 and 6.3 we present several heuristic approaches that are based on the resulting models. In Section 6.4 we propose two fix-and-solve heuristic approaches. In Section 6.5 we present and analyze the results of computational tests of the proposed approaches.

6.1 Overview of *k-opt* Technique and *k-opt* Models

In this section, we introduce the *k-opt neighborhood search* technique (Section 6.1.1) and we propose three *k-opt* neighborhoods for our problems (Section 6.1.2).

6.1.1 *k-opt neighborhood search* Technique

In brief, *k-opt neighborhood search* presented in [63] is in the spirit of local search meta-heuristics, but the neighborhoods are defined as the sets of solutions that differ in at most k variable values from the current solution: in the MIP model a linear inequality called *local branching constraint* is added that imposes a distance of changes with respect to a reference solution. The framework uses a generic MIP solver to explore effectively the defined neighborhoods.

Let the generic MIP problem with binary variables be a minimization problem of the following form

$$\begin{aligned}
&min\ c^T x\\
&s.t.\ Ax \geq b\\
&x_j \in \{0,1\} \quad \forall j \in \mathbb{B} \neq \emptyset\\
&x_j \geq 0, \quad \text{integer} \quad \forall j \in \mathbb{G}\\
&x_j \geq 0 \quad \forall j \in \mathbb{C}
\end{aligned}$$

where $\mathbb{B} \neq \emptyset$ is the index set of the binary variables, while the possible empty sets $\mathbb{G}$ and $\mathbb{C}$ are the integer and the continuous variables, respectively. Given a feasible reference solution $\bar{x}$, let $\bar{S} := \{j \in \mathbb{B} : \bar{x}_j = 1\}$ denote the binary support of $\bar{x}$. For a given positive integer parameter k, the *k-opt neighbourhood* of $\bar{x}$ is defined as the set of the feasible solutions satisfying the additional *local branching constraint*:

$$\sum_{j \in \bar{S}} (1 - x_j) + \sum_{j \in \mathbb{B} \setminus \bar{S}} x_j \leq k \tag{6.1}$$

Constraint (6.1) imposes a maximum Hamming distance of k among feasible neighbors of $\bar{x}$, i.e., in a neighbor solution, only k binary variables are allowed to flip their value with respect to $\bar{x}$.

6.1.2 *k-opt* Models

In the previous sections, we proposed the SPI formulation (an improved SP formulation in Section 5.1.1) to solve directly the VNF-PR$_{\mathrm{SP}}$ problem. We also proposed the SP$_{\mathrm{fixnum}}{}^{\mathrm{MD}}$-VI1 formulation to solve the VNF-PR$_{\mathrm{SP}}{}^{\mathrm{MD}}$ problem (in Section 5.1.3) with a fixed number of installed VNF instances to check if all the demands can be served. If the fixed number of installed VNF instances can serve all the demands in the VNF-PR$_{\mathrm{SP}}{}^{\mathrm{MD}}$ problem, a feasible solution for the VNF-PR$_{\mathrm{SP}}$ problem is found. We now use the SPI formulation and the SP$_{\mathrm{fixnum}}{}^{\mathrm{MD}}$-VI1 formulation within the *k-opt neighborhood search* framework.

Indeed, two families of binary variables can be involved in the *local branching constraints*, the location variables y and the assignment variables z:

- if the location variables are considered, the *local branching constraint* is as follows:

$$\sum_{i \in \bar{S}_f} (1 - y_i) + \sum_{i \in N \setminus \bar{S}_f} y_i \leq k_f \tag{6.2}$$

  where $\bar{S}_f$ is the set of selected locations and it limits the number of location variables that can change their values;

- if the demand-VNF assignment variables are considered, the *local branching constraint* is as follows:

$$\sum_{(k,i) \in \bar{S}_d} (1 - z_i^k) + \sum_{(k,i) \in (D,N) \setminus \bar{S}_d} z_i^k \leq k_d \tag{6.3}$$

  where $\bar{S}_d$ represents the selected demand-VNF pair.

Resulting model	Formulation used	Additional constraints
SPI-NS$_{\mathrm{loc}}$	SPI	Eq. (6.2)
SPI-NS$_{\mathrm{locass}}$	SPI	Eq. (6.2), (6.3)
SP$_{\mathrm{fixnum}}{}^{\mathrm{MD}}$-VI1-NS$_{\mathrm{ass}}$	SP$_{\mathrm{fixnum}}{}^{\mathrm{MD}}$-VI1	Eq. (6.3)

Table 6.1 – Resulting models of the SPI and SP$_{\mathrm{fixnum}}{}^{\mathrm{MD}}$-VI1 formulations combined with the additional *local branching constraint.*

By combining the formulations and the additional *local branching constraints* we obtain three models, which will be solved within different *k-opt neighborhood search* based heuristics both for computing a feasible solution and for improving it. The obtained models are reported in Table 6.1. We want to point out that the choice of the values k_f and k_d is crucial for the algorithm performances. We conducted some preliminary tests to choose them, but a full analysis on their impact will be addressed in future work.

6.2 Computing An Initial Solution

In this section, we present four approaches for obtaining an initial feasible solution for the VNF-PR$_{\mathrm{SP}}$ problem. In Table 6.2 we summarize the models implemented in each approach.

Approach	Models solved
RFR	SPI-NS$_{\mathrm{loc}}$
SC-RFR	SP$_{\mathrm{fixnum}}{}^{\mathrm{MD}}$-VI1, SPI-NS$_{\mathrm{loc}}$
SC-ANS	SP$_{\mathrm{fixnum}}{}^{\mathrm{MD}}$-VI1, SP$_{\mathrm{fixnum}}{}^{\mathrm{MD}}$-VI1-NS$_{\mathrm{ass}}$
SC-ANSI	SP$_{\mathrm{fixnum}}{}^{\mathrm{MD}}$-VI1, SP$_{\mathrm{fixnum}}{}^{\mathrm{MD}}$-VI1-NS$_{\mathrm{ass}}$

Table 6.2 – Models used in each approach for obtaining an initial feasible solution for VNF-PR$_{\mathrm{SP}}$.

The first approach starts opening a VNF at each node. Then SPI-NS$_{\mathrm{loc}}$ is solved to verify if a feasible assignment can be found. Although the model is solved only once, the *local branching constraint* is added to reduce the feasible region so as to speed up the solution process. This Reduced Feasible Region (RFR) however leads to solutions where the number of opened VNF is at least $|N| - k_f$. The approach is summarized in Algorithm 2.

Algorithm 2 Overview of RFR procedure

```
1: procedure RFR
2:     let $\bar{S}_f \leftarrow N$
3:     solve SPI-NS$_{\mathrm{loc}}$
4:     if problem solved then
5:         stop with initial feasible solution found
6:     else
7:         stop without any feasible solution found
```

In RFR, if SPI-NS$_{\mathrm{loc}}$ is solved within the given time limit, an initial feasible solution of our problem is provided. Otherwise, no feasible solution is provided.

Algorithm 3 Overview of SC-RFR approach

```
1: let $LB_{fix} \leftarrow \min(2 * LB_{bpp}, |N|)$
2: execute procedure SC
3: if initial feasible solution provided by SC then
4:     stop with initial feasible solution found
5: else
6:     execute procedure RFR
```

An alternative approach SC-RFR has been developed with the aim of providing a smaller number of used VNF. The approach selects a number of VNF and checks if a solution can be found with the selected number by solving the SP$_{\mathrm{fixnum}}{}^{\mathrm{MD}}$-VI1 formulation (Procedure SC). The number of VNFs to be checked is selected as follows: initially we consider $LB_{fix} = min(|N, 2 * LB_{bpp})$ VNFs; then we solve the continuous relaxation of $\text{SP}_{\text{fixnum}}{}^{\text{MD}}$-VI1: if a feasible solution is found, namely all the demands are served, the number of VNFs is then selected to be checked; otherwise, LB_{fix} is increased and the continuous relaxation is solved again until either a feasible continuous solution is found or all the VNFs are opened. The approach is sketched in Algorithm 3 and the select-and-check (SC) procedure is in Algorithm 4.

Algorithm 4 Overview of SC procedure

```
 1: procedure SC
 2:     while time limit is not exceeded & $LB_{fix} \leq |N|$ do
 3:         solve relaxation of $\text{SP}_{\text{fixnum}}{}^{\text{MD}}$-VI1
 4:         let $res \leftarrow$ the solution
 5:         if problem solved & $res = |D|$ then
 6:             get out of the loop
 7:         else
 8:             let $LB_{fix} = LB_{fix} + 1$
 9:     solve $\text{SP}_{\text{fixnum}}{}^{\text{MD}}$-VI1
10:     let $res \leftarrow$ the solution
11:     if problem solved & $res = |D|$ then
12:         stop with initial feasible solution found
13:     else
14:         stop without any feasible solution found
```

Both the above described approaches may end without providing a feasible solution. Indeed, in both, the $\text{SPI-NS}_{\text{loc}}$ model (that searches a feasible solution) is solved only once. To overcome such drawback, we propose to apply a Local Search (LS) to look for a feasible solution when SC does not provide any. The approach starts as SC-RFR, but instead of executing the RFR procedure which may stop without returning a feasible solution (if $\text{SPI-NS}_{\text{loc}}$ cannot be solved), it tries to find a feasible solution through a *k-opt neighborhood* based local search procedure that solves the $\text{SP}_{\text{fixnum}}{}^{\text{MD}}\text{-VI1-NS}_{\text{ass}}$ model to maximize the number of served demands. We denote this procedure with the Assignment Neighborhood Search (ANS) procedure, where a VNF instance is located at each node and the $\text{SP}_{\text{fixnum}}{}^{\text{MD}}\text{-VI1-NS}_{\text{ass}}$ model is solved iteratively until a feasible solution is found (namely all the demands are served). An overview of the SC-ANS approach is described in Algorithm 5 (the ANS procedure is shown in Algorithm 6).

Algorithm 5 Overview of SC-ANS approach

```
1: let $LB_{fix} \leftarrow \min(2 * LB_{bpp}, |N|)$
2: execute procedure SC
3: if initial feasible solution provided by SC then
4:     stop with initial feasible solution found
5: else
6:     let $LB_{fix} \leftarrow |N|$
7:     let $\bar{S}_d \leftarrow \emptyset$
8:     let bestSolution $\leftarrow 0$
9:     execute procedure ANS
```

Algorithm 6 Overview of ANS procedure

```
 1: procedure ANS
 2:     while total time limit is not exceeded do
 3:         solve $SP_{\text{fixnum}}{}^{MD}$-VI1-$NS_{\text{ass}}$
 4:         let $res \leftarrow$ the solution
 5:         if problem solved & $res = |D|$ then
 6:             stop with initial feasible solution found
 7:         if problem solved & bestSolution $< res < |D|$ then
 8:             let $\bar{S}_d \leftarrow$ current assignment solution
 9:             let bestSolution $\leftarrow res$
10:         else
11:             stop without any feasible solution found
```

In a feasible solution for the VNF-PR_{SP} problem provided by SC-ANS, the number of installed VNF instances is always equal to $|N|$. In order to provide an initial feasible solution that uses less VNF instances, we propose an improved ANS procedure, where $SP_{\text{fixnum}}{}^{MD}$-VI1-$NS_{\text{ass}}$ is solved repeatedly starting with LB_{fix} equal to the value found by SC, and the value is increased one by one until all the demands are served by the real VNF instances. An overview of the SC-ANSI (SC-ANS improved) approach is shown in Algorithm 7 (the ANSI procedure is described in Algorithm 8).

Algorithm 7 Overview of SC-ANSI

```
1: let $LB_{fix} \leftarrow \min(2 * LB_{bpp}, |N|)$
2: execute procedure SC
3: if initial feasible solution provided by SC then
4:     stop with initial feasible solution found
5: else
6:     let $\bar{S}_d \leftarrow \emptyset$
7:     let bestSolution $\leftarrow 0$
8:     execute procedure ANSI
```

Algorithm 8 Overview of ANSI procedure

```
 1: procedure ANSI
 2:     while total time limit is not exceeded & $LB_{fix} \leq |N|$ do
 3:         solve $SP_{\text{fixnum}}{}^{MD}$-VI1-$NS_{\text{ass}}$
 4:         let $res \leftarrow$ the solution
 5:         if problem solved & $res = |D|$ then
 6:             stop with initial feasible solution found
 7:         if problem solved & bestSolution $< res < |D|$ then
 8:             let $\bar{S}_d \leftarrow$ current assignment solution
 9:             let bestSolution $\leftarrow res$
10:         else
11:             let $LB_{fix} = LB_{fix} + 1$
12:         if $LB_{fix} > |N|$ then
13:             stop without any feasible solution found
```

6.3 Improving Local Search (LS) Based Heuristics

To improve the solution, we propose two *k-opt neighborhood search* based steps that operates either on location variables (we denote it with NS_{loc}, where the SPI-NS_{loc} model is solved) or on both location and assignment variables (we denote it with $\text{NS}_{\text{locass}}$, where the SPI-$\text{NS}_{\text{locass}}$ model is solved). Combining the two improving local search steps with the initial solution approaches, we developed eight heuristics methods, which are listed in Table 6.3. In the second and third columns we report the initial solution approach and the improving step considered in each combination.

Methods	Initial solution approach	Improving step
m1	RFR	NS_{loc}
m1Bis	RFR	$\text{NS}_{\text{locass}}$
m2	SC-RFR	NS_{loc}
m2Bis	SC-RFR	$\text{NS}_{\text{locass}}$
m3	SC-ANS	NS_{loc}
m3Bis	SC-ANS	$\text{NS}_{\text{locass}}$
m4	SC-ANSI	NS_{loc}
m4Bis	SC-ANSI	$\text{NS}_{\text{locass}}$

Table 6.3 – Summary of improving heuristic approaches.

6.4 Fix-and-Optimize Heuristics

In this section, we present two fix-and-optimize (FO) heuristic methods.
Our first FO method uses a bisection algorithm on the number of installed VNF instances. As discussed previously in Section 3.2.4, solving the TE objective with a bisection procedure on the number of VNF instances allows us to obtain better results (e.g., solve to optimality, or obtain a solution with smaller gap) than solving directly the NFV objective problem. Inspired by this result, we decide to implement a bisection based FO method, where the $\text{SP}_{\text{fixnum}}{}^{\text{MD}}$-VI1 model is solved iteratively within a bisection framework to check if all the demands can be served by the selected number of VNF instances. An overview of this method is shown in Algorithm 9.

In Algorithm 9, the interval of the possible number of installed VNF instances is denoted by interval $[Int_a, Int_b]$. Endpoint Int_a is initialized as the optimal solution of the BPP-like (namely a lower bound of VNF-PR_{SP}) and endpoint Int_b is initialized as the number of physical nodes. At each iteration, the number of installed VNF instances is set to be the midpoint (if the number is not integer, we round down it to an integer value) of the interval, and $\text{SP}_{\text{fixnum}}{}^{\text{MD}}$-VI1 is solved with the obtained number. If all the demands can be served by the given number of VNF instances, the interval is updated by replacing the value of Int_b by the value of LB_{fix}. Otherwise, the interval is updated by replacing the value of Int_a by the value of LB_{fix}. In addition, in order to check quickly if a given number of VNF instances can serve all the demands, we solve the relaxation of $\text{SP}_{\text{fixnum}}{}^{\text{MD}}$-VI1 before solving the integer one. Since it takes less time to solve the relaxation problem, if the given number of VNF instances cannot serve all the demands, the interval is then updated

Algorithm 9 Overview of bisection based FO method

```
 1: solve BPP-like
 2: let $Int_a \leftarrow LB_{bpp}$
 3: let $Int_b \leftarrow |N|$
 4: while time limit is not exceeded & $Int_b \neq Int_a$ do
 5:     let $LB_{fix} \leftarrow \lfloor \frac{Int_a+Int_b}{2} \rfloor$
 6:     solve relaxation of $\mathrm{SP_{fixnum}}^{\mathrm{MD}}$-VI1
 7:     let $res \leftarrow$ the solution
 8:     if problem solved & $res = |D|$ then
 9:         solve $\mathrm{SP_{fixnum}}^{\mathrm{MD}}$-VI1
10:         let $res \leftarrow$ the solution
11:         if problem solved & $res = |D|$ then
12:             let $Int_b \leftarrow LB_{fix}$
13:         else
14:             let $Int_a \leftarrow LB_{fix}$
15:     else
16:         let $Int_a \leftarrow LB_{fix}$
```

and the algorithm may converge rapidly.

The second FO method fixes the number as well as the location of VNF instances and solves the $\mathrm{SP_{fixnum}}^{\mathrm{MD}}$-VI1 model to check the problem feasibility. We start with locating VNF instances on all the nodes. At each iteration, we close randomly one instance on a node and we solve $\mathrm{SP_{fixnum}}^{\mathrm{MD}}$-VI1 to check if all the demands can be served by real VNF instances. If all the demands are served, the instance is closed ans the procedure is repeated. Otherwise, we switch the VNF instance to close and solve $\mathrm{SP_{fixnum}}^{\mathrm{MD}}$-VI1 to check the feasibility, until all the possible locations have been traversed.

6.5 Computational Results

For the sake of comparison and completeness, we performed computational tests on the proposed heuristic methods with the same test bed generated previously (Chapter 4). The 132 instances features are summarized before in Table 4.10. For the convenience of reading, we report the generated instances in Table 6.4, and we re-group the instances according to the solution quality provided by the model SP-VI1+2 to present the computational tests of heuristic methods. Similar to the tests on constructive methods, the results of the full model SP-VI1+2 are used as a reference for the comparison of heuristic methods.

Data from SNDLib					Capacity		
Network	$\|N\|$	$\|A\|$	$\|D\|$	$\sum_{k \in D} d_k$ (high cap)	Service medium	Service low	Link low
Small-medium size instances							
di-yuan	11	42	22	53	31	9	5
pdh	11	34	24	4621	2730	840	384
sun	27	51	67	476	255	35	53
dfn-bwin	10	45	90	548388	329032	109677	55916
nobel-germany	17	26	121	660	368	77	74
abilene	12	15	132	3000002	1750001	500000	829282
ta1	24	51	396	10127249	5485593	843937	819678
atlanta	15	22	210	136726	77478	18230	19404
nobel-us	14	21	91	5420	3097	774	486
nobel-eu	28	41	378	1898	1016	135	214
geant	22	36	462	2999992	1636359	272726	359868
janos-us	26	42	650	80000	43076	6153	7624
polska	12	18	66	9943	5800	1657	995
newyork	16	49	240	1774	997	221	66
france	25	45	300	99830	53908	7986	9413
norway	27	51	702	5348	2872	396	358
Large size instances							
india35	35	80	595	3292	1740	188	121
cost266	37	57	1332	679598	358166	36735	53562
giul39	39	86	1471	7366	3871	377	363
janos-us-ca	39	61	1482	2032274	1068246	104219	180471
pioro40	40	89	780	115953	60875	5797	7609
germany50	50	88	662	2365	1229	94	123

Table 6.4 – Test-instance details

6.5.1 Results of Approaches for Obtaining An Initial Solution

Since SP-VI1+2 can be fast solved for all the small-medium size instances in l_l case (on average 150s), we decide to test the performance of the four approaches only on instances in h_l and m_l cases. Moreover, even in h_l and m_l cases, SP-VI1+2 can always solve to optimality quickly the network topologies of the first group (the average computational time of the 14 instances is less than 100s). Therefore, we further eliminate these instances, and we run test only on the second and third group of small-medium size network topologies (from “atlanta” to “norway”) for the approaches. Hamming distances k_f (on location variables) of $\lceil |N|/10 \rceil$, k_d (on assignment variables) of $\lceil |D|/2 \rceil$ and a time limit of 600s are set for the test of the four approaches.

Since SP-VI1+2 is always able to find a solution but with large computational time for the instances in the second group, for these instances, we evaluate and compare the initial feasible solutions obtained by each approach with respect to model solution (see Table 6.5). Computational times are reported in Table 6.6. Results are reported for each instance of each capacity case. For both tables, Column 2 refers to SP-VI1+2 (with the time limit of 3600s). In Table 6.5, two columns are reported for each approach: the initial feasible solution obtained with a time limit of 600s and the absolute gap with respect to the SP-VI1+2 solution, i.e., the optimal solution[9].

Instance	Solution and the gap w.r.t SP-VI1+2								
	SP-VI1+2	RFR		SC-RFR		SC-ANS		SC-ANSI	
	sol	sol	gap	sol	gap	sol	gap	sol	gap
h_l case									
atlanta	3	14	11	4	1	4	1	4	1
nobel-us	4	13	9	4	0	4	0	4	0
nobel-eu	3	26	23	4	1	4	1	4	1
geant	1	20	19	2	1	2	1	2	1
janos-us	5*	24	19	24	19	26	21	-	-
m_l case									
atlanta	3	14	11	4	1	4	1	4	1
nobel-us	4	13	9	4	0	4	0	4	0
nobel-eu	3	26	23	4	1	4	1	4	1
geant	2	20	18	4	2	4	2	4	2
janos-us	5*	24	19	24	19	26	21	-	-

Table 6.5 – Initial feasible solution obtained by each approach with a time limit of 600s and the gap with respect to SP-VI1+2 solution with a time limit of 3600s for test on the second group of small-medium size network instances (- means no feasible solution found within time limit. * means optimality not proved within time limit).

We observe that RFR is the fastest approach: for all the 10 instances, it uses less than 30s (up to 3 orders of magnitude faster than SP-VI1+2) to provide an initial solution. However, the gap with the optimal solution is huge. In fact, as mentioned before, the initial solution provided by RFR depends on the value of k: the number of installed VNF instances is at best $|N| - k$. Approaches SC-RFR, SC-ANS and SC-ANSI perform similarly in solving typologies “atlanta”, “nobel-us”, “nobel-eu” and “geant” in both h_l and m_l cases. They provide a same initial feasible solution with a similar computational time. Compared with SP-VI1+2, the

[9]as mentioned before, the solution of instance janos-us of both h_l and m_l cases is in fact an optimal solution, but the optimality is not proved within time limit

	Computational times				
Instance	SP-VI1+2	RFR	SC-RFR	SC-ANS	SC-ANSI
h_l case					
atlanta	1215.05	3.04	14.61	15.21	14.64
nobel-us	580.25	5.48	32.98	33.61	32.83
nobel-eu	2688.76	4.33	90.28	91.45	90.64
geant	473.64	9.59	28.07	27.04	26.10
janos-us	3600.00	16.98	522.26	534.09	600.00
m_l case					
atlanta	3004.51	20.16	8.88	10.06	9.11
nobel-us	283.58	5.60	47.70	47.97	44.27
nobel-eu	380.32	4.74	58.36	44.82	44.60
geant	1648.92	25.31	17.46	20.11	17.70
janos-us	3600.00	17.87	524.47	543.29	600.00

Table 6.6 – Computational times used by each approach and the SP-VI1+2 model for test on the second group of small-medium size network instances.

three approaches provide a close-to-optimal initial feasible solution (the worst gap is 2) with a decrease of 2 orders of magnitude in the computational time. As described previously, the three approaches are integrated with the SC procedure, which allows solving $\text{SP}_{\text{fixnum}}{}^{\text{MD}}$ with a fixed number of VNF instances (i.e., LB_{fix}). If all the demands can be served by the given number, an initial feasible solution for VNF-SP_{SP} is found. This is the case of the four topologies (from atlanta to geant), for which the initial feasible solution is found by the SC procedure in the three approaches. Thus, the three approaches have a similar behaviour.

Things change when SC cannot provide any feasible solution, and this is the case of janos-us. In this case, SC-RFR usually spends more time than RFR for providing a same initial feasible solution (if found), as after running the SC procedure, SC-RFR behaves in the same way as RFR. Different from SC-RFR, SC-ANS tries to find a feasible solution by using all the services after the SC procedure. Therefore, if SC-ANS finds a feasible solution, the number of opened services is exactly the number of nodes (i.e., 26 nodes in janos-us topology). While in SC-ANSI, the given number of opened services is checked and increased step by step. Therefore, SC-ANSI may not provide any feasible solution with short computational time. But if more time is given (or the rest of time is enough after the procedure finishes), this strategy may provide the best solution among all the four strategies, which is confirmed by the results reported in Table 6.7 for the third group of small-medium size instances.

Here both h_l and m_l cases are challenging for the model: SP-VI1+2 finds optimal solution only for instance "polska" in the m_l case, and it cannot find any feasible solution for the other instances within a time limit of 3600s. Therefore, we compare the initial feasible solution obtained by each approach with the best one among them (see Table 6.7). For each approach, both the initial feasible solution obtained with a time limit of 600s and the computational times are reported, the best values are in bold. Columns 10-13 summarize the metrics for the comparison: column 10 reports the rounded lower bound found by SP-VI1+2, column 11 reports the best solution obtained, and column 12 reports the smallest time used for obtaining an initial feasible solution.

Instance	Solution and the computational times								Metric		
	RFR		SC-RFR		SC-ANS		SC-ANSI		SP-VI1+2	Best all	
	sol	time	sol	time	sol	time	sol	time	$\lceil LB \rceil$	sol	time
h_l case											
polska	**11**	**153.73**	**11**	335.80	-	TL	-	TL	4	11	153.73
newyork	-	TL	-	TL	**16**	**266.05**	-	TL	3	16	266.05
france	-	TL	-	TL	-	TL	-	TL	4	-	TL
norway	**25**	**346.22**	-	TL	-	TL	-	TL	5	25	346.22
m_l case											
polska	11	**378.32**	11	530.92	-	TL	**5**	TL	4	5	378.32
newyork	15	**107.79**	-	TL	16	215.58	**4**	544.48	3	4	107.79
france	-	TL	-	TL	-	TL	-	TL	4	-	TL
norway	**25**	**360.67**	-	TL	-	TL	-	TL	5	25	360.67

Table 6.7 – Initial feasible solution obtained and the computational time used by each approach for test on the third group of small-medium size instances. TL represents the time limit of 600s (- means no feasible solution found within time limit).

In terms of number of initial feasible solutions found, RFR is the best among the four approaches: it provides initial feasible solution for 5 out of 8 instances. Furthermore, it is the only strategy that can provide an initial feasible solution for topology "norway" in both h_l and m_l cases within 600s, while the last three approaches provide initial feasible solution only for 2 instances by each. Moreover, it is not the SC procedure who provides the initial feasible solution for the 2 instances that are succeeded by the last three approaches. For "polska" of both h_l and m_l case, SC-RFR finds the same solution as RFR since SC-RFR behaves in the same way as RFR after the SC procedure. As for "newyork" topology, it is worth noting that in h_l case, SC-ANS is the only one that obtains an initial feasible solution (with services all opened). This suggests that the strategy of finding a feasible solution by using all the services is worth to investigate when dealing with instances that are difficult to solve in short computational times. As mentioned previously, if given more time, SC-ANSI may provide the best initial solution. This is confirmed by the results of case m_l: it finds 5 for "polska" and 4 for "newyork" as initial feasible solution, for which the gap with the optimal solution is at most 1.

6.5.2 Results of Heuristic Methods

In brief, tests for heuristic methods are run on three groups of instances:

- with the second group of small-medium size instances and a time limit of 600s (including the time for the initial solution), we compare the proposed methods with the SP-VI1+2 model.

- with the third group of small-medium size instances and a time limit of 600s, we evaluate the solution improvement with respect to the initial feasible solution obtained.

- with the large size instances and a time limit of 7200s, we further study the scalability of the proposed methods.

For all the instances, only h_l and m_l capacity cases are considered as they are the most challenging cases for the model. Similar to tests of approaches for obtaining an initial feasible solution, Hamming distances k_f (on location variables) of $\lceil |N|/10 \rceil$ and k_d (on assignment variables) of $\lceil |D|/2 \rceil$ are set.

We now compare the solution provided by each heuristic method with the solution provided by SP-VI1+2 (see Table 6.8). Column 2 reports the solution provided by the SP-VI1+2 model with a time limit of 3600s. Columns 3-12 report the solution provided by the ten heuristic methods with a time limit of 600s: columns 3-10 report the solution provided by the eight local search heuristic methods, and columns 11-12 report the solution provided by the two FO methods. Aggregated results are reported for each method in the last three rows, i.e., the number of optimal solutions found, the average absolute gap with the optimal solution and the maximal gap with the optimal solution.

	Solution obtained by each method										
Approach		RFR		SC-RFR		SC-ANS		SC-ANSI			
Step		NS$_{\text{loc}}$	NS$_{\text{locass}}$	NS$_{\text{loc}}$	NS$_{\text{locass}}$	NS$_{\text{loc}}$	NS$_{\text{locass}}$	NS$_{\text{loc}}$	NS$_{\text{locass}}$		
Instance	SP-VI1+2	m1	m1Bis	m2	m2Bis	m3	m3Bis	m4	m4Bis	m5	m6
h_l case											
atlanta	3	4	4	3	4	3	4	3	4	3	5
nobel-us	4	4	4	4	4	4	4	4	4	4	4
nobel-eu	3	3	3	3	3	3	3	3	3	4	3
geant	1	1	1	1	2	1	2	1	2	3	3
janos-us	5	6	7	14	14	16	16	-	-	8	12
m_l case											
atlanta	3	4	4	3	4	3	4	3	4	3	6
nobel-us	4	4	5	4	4	4	4	4	4	4	4
nobel-eu	3	3	3	3	4	3	4	3	4	3	3
geant	2	2	3	2	2	2	2	2	2	2	3
janos-us	5	6	7	12	12	16	10	-	-	8	12
aggregated results of 10 instances											
# opt	10	6	4	8	4	8	4	8	4	6	4
avg gap	0	1	1.3	8	3.3	11	3.3	0	1	2.5	3.7
max gap	0	1	2	9	9	11	11	0	1	3	7

Table 6.8 – Solution obtained by the SP-VI1+2 model with a time limit of 3600s and each heuristic method with a time limit of 600s for test on the second group of small-medium size network instances. The gap is computed only for feasible not optimal solutions. We denote the bisection based FO method (Algorithm 9) with m5, and the other FO method with m6 (- means no feasible solution found within time limit).

Here we present a classification among the proposed heuristic methods according to the number of optimal solution found.

1. methods m2, m3 and m4 are the best as they provide optimal solution for 8 out of 10 instances. Moreover, they perform similarly in solving instances atlanta, nobel-us, nobel-eu and geant in both h_l and m_l cases. This is because they all use the $\mathrm{NS}_{\mathrm{loc}}$ strategy to improve a same initial feasible solution found by the same procedure in their stages for obtaining an initial feasible solution (analysis in Section 6.5.1). However, for the two instances of janos-us in h_l and m_l cases, m2 and m3 provide a feasible solution with a large gap, and m4 cannot provide any feasible solution;

2. after we have m1 and m5 that can find optimal solution for 6 out of 10 instances. For the rest 4 instances, they provide a feasible solution with an average gap of 1 for m1 and 2.5 for m5. It seems that m1 can provide better solutions than m5;

3. then m1Bis, m2Bis, m3Bis, m4Bis and m6 all find optimal solution for 4 out of 10 instances. For the remaining 6 instances, all of them can provide a feasible solution, except for m4Bis. However, the average gap of m4Bis solution is the smallest (i.e., 1), while for the others at least 3.

When we consider the computational times (in Table 6.9) used by each method, m1Bis, m2Bis, m3Bis, m4Bis and m5 are the fast ones, their average computational time is around 150s for the 10 instances. m1, m2, m3 and m4 requires about 50s more. While m6 needs at least twice the computational times of the others.

	Solution obtained by each method										
Approach		RFR		SC-RFR		SC-ANS		SC-ANSI			
Step		$\mathrm{NS}_{\mathrm{loc}}$	$\mathrm{NS}_{\mathrm{locass}}$	$\mathrm{NS}_{\mathrm{loc}}$	$\mathrm{NS}_{\mathrm{locass}}$	$\mathrm{NS}_{\mathrm{loc}}$	$\mathrm{NS}_{\mathrm{locass}}$	$\mathrm{NS}_{\mathrm{loc}}$	$\mathrm{NS}_{\mathrm{locass}}$		
Instance	SP-VI1+2	m1	m1Bis	m2	m2Bis	m3	m3Bis	m4	m4Bis	m5	m6
h_l case											
atlanta	1215.05	43.14	42.33	18.26	14.61	18.75	14.65	18.11	14.70	41.84	275.79
nobel-us	580.25	112.44	25.81	32.25	32.98	33.61	31.96	32.83	32.88	34.62	508.22
nobel-eu	2688.76	197.59	73.58	178.56	104.78	462.00	98.74	456.46	103.42	139.96	413.50
geant	473.64	109.37	92.32	50.15	28.07	52.92	25.71	50.80	26.34	125.92	200.22
janos-us	3600.00	362.85	418.05	600.00	586.90	583.56	590.04	600.00	600.00	200.76	508.396
m_l case											
atlanta	3004.51	50.87	39.95	44.75	8.88	83.30	8.78	75.77	8.96	51.24	507.484
nobel-us	283.58	112.75	16.84	43.78	47.70	47.97	43.55	44.27	44.33	97.00	525.77
nobel-eu	380.32	211.87	75.23	164.28	58.36	220.43	44.44	223.25	45.94	247.17	407.15
geant	1648.92	271.86	141.21	94.62	25.12	58.02	37.08	51.07	37.38	344.41	414.32
janos-us	3600.00	319.67	427.26	600.00	600.00	583.56	593.66	600.00	600.00	189.20	474.92
average computational time considering time limit											
avg time	1,747.50	179.24	135.26	182.94	151.27	214.41	148.86	215.26	151.39	147.21	423.58

Table 6.9 – Computational times used by the model SP-VI1+2 with time limit of 3600s and each heuristic methods with a time limit of 600s for test on the second group of small-medium size network instances.

When we take into consideration both solution quality and computational time for tests on the second group of small-medium size instances, the performance of m6 is the worst one since it finds less optimal solution with large computational time. Although m5 solution is not the best, it is in general very fast. As for the local search methods, it seems that the NS_{loc} step (used in method m1, m2, m3 and m4) is able to find better solution than $\text{NS}_{\text{locass}}$ (used in method m1Bis, m2Bis, m3Bis and m4Bis) with a bit more computational time. $\text{NS}_{\text{locass}}$ based methods take less computational time since the neighbourhood of the reference solution is restricted by both k_f and k_d. For these instances, even the time limit is big enough, $\text{NS}_{\text{locass}}$ is not able to find as good solutions as NS_{loc}. This may be due to the fact that the value of k_d is chosen too small to contain better solutions than the current one. However, janos-us in m_l is an exception: m3 (i.e., with NS_{loc}) provides a solution of 16 while m3Bis (with $\text{NS}_{\text{locass}}$) finds 10. This shows that the solution of NS_{loc} is not necessary better than $\text{NS}_{\text{locass}}$, since with smaller neighbourhood we can iterate faster (if better solution found) to improve current solution. But it is difficult to determine which search strategy is better, the reason is that, the choice of value k can influence strongly the results.

Solution obtained by each method with a time limit of 600s										
Approach	RFR		SC-RFR		SC-ANS		SC-ANSI			
Step	NS_{loc}	$\text{NS}_{\text{locass}}$	NS_{loc}	$\text{NS}_{\text{locass}}$	NS_{loc}	$\text{NS}_{\text{locass}}$	NS_{loc}	$\text{NS}_{\text{locass}}$		
Instance	m1	m1Bis	m2	m2Bis	m3	m3Bis	m4	m4Bis	m5	m6
h_l case										
polska	5	5	5	5	-	-	-	-	-	-
newyork	-	-	-	-	10	8	-	-	-	-
france	-	-	-	-	-	-	-	-	13	24
norway	19	19	-	-	-	-	-	-	15	24
m_l case										
polska	7	6	7	6	-	-	5	5	-	-
newyork	14	7	-	-	7	7	4	4	-	-
france	-	-	-	-	-	-	-	-	-	-
norway	19	19	-	-	-	-	-	-	15	24
# reduced services with respect to their initial feasible solution obtained										
h_l case										
polska	6	6	6	6	-	-	-	-	-	-
newyork	-	-	-	-	6	8	-	-	-	-
france	-	-	-	-	-	-	-	-	0	0
norway	6	6	-	-	-	-	-	-	0	2
m_l case										
polska	4	5	4	5	-	-	0	0	-	-
newyork	1	8	-	-	9	9	0	0	-	-
france	-	-	-	-	-	-	-	-	-	-
norway	6	6	-	-	-	-	-	-	0	2
average # reduced services when there is an improvement										
avg	4.6	6.2	5	5.5	7.5	8.5				2

Table 6.10 – Solution and the improvement w.r.t initial feasible solution obtained by each heuristic method for tests on the third group of small-medium size instances. - means no feasible solution found within the time limit of 600s.

We further evaluate the proposed heuristic methods with tests on the third group of small-medium size instances of both h_l and m_l cases. Table 6.10 reports the results of each heuristic methods. As before, we use a time limit of 600s. Rows 16-25 report the difference between the final solution obtained within 600s and the initial feasible solution provided, and the last row reports the average reduction in the number of services. In Table 6.11 the computational times are given. As presented previously (Table 6.7), it is challenging to provide an initial feasible solution for these instances within the time limit of 600s. Thus, the number of instances in which a method can find a solution is exactly the number of instances where the approach can provide an initial solution. Therefore, we only compare the improvement with respect to the initial feasible solution.

We observe that methods m4, m4Bis and m5 are not improving the initial feasible solution once obtained. Moreover, m5 use less than 300s to provide an initial feasible solution, but it needs longer time to improve the current solution. While m4 and m4Bis use more than 540s to find an initial feasible solution, the remaining available time for improving the current solution is less than 60s. Method m6 can find solution for the difficult instances “france”, “norway” of h_l and “norway” of m_l, but it is less efficient for providing good solution with short time. As m6 tries to close the opened VNF instances one by one, it provides an initial feasible whose number of services is always $N-1$ and it takes a lot of time to improve the solution. As for methods m2, m2Bis, m3 and m3Bis, we observe a half reduction (i.e., 5 VNF instances) with respect to the initial feasible solution within less than 200s (Table 6.7 vs. Table 6.11). Moreover, $\mathrm{NS}_{\mathrm{locass}}$ (used in methods m2Bis and m3Bis) seems work more efficiently than $\mathrm{NS}_{\mathrm{loc}}$ (used in methods m2 and m3), as it finds better solutions than $\mathrm{NS}_{\mathrm{loc}}$. This also can be confirmed by the results of m1 and m1Bis, where the average improvement of m1Bis is 6.2 services while m1 is 4.6 services. Moreover, m1Bis is able to provide a much better solution (which is 7 services) for “newyork” of m_l case than m1 (14 services), for which the difference of their improvement is 7 services.

Approach Step	Computational times used by each method									
	RFR		SC-RFR		SC-ANS		SC-ANSI			
	$\mathrm{NS}_{\mathrm{loc}}$	$\mathrm{NS}_{\mathrm{locass}}$	$\mathrm{NS}_{\mathrm{loc}}$	$\mathrm{NS}_{\mathrm{locass}}$	$\mathrm{NS}_{\mathrm{loc}}$	$\mathrm{NS}_{\mathrm{locass}}$	$\mathrm{NS}_{\mathrm{loc}}$	$\mathrm{NS}_{\mathrm{locass}}$		
	m1	m1Bis	m2	m2Bis	m3	m3Bis	m4	m4Bis	m5	m6
h_l case										
polska	203.90	395.12	381.37	579.29	TL	TL	TL	TL	TL	TL
newyork	TL	TL	TL	TL	443.98	328.60	TL	TL	TL	TL
france	TL	TL	TL	TL	TL	TL	TL	TL	80.99	189.16
norway	520.42	573.74	TL	TL	TL	TL	TL	TL	283.46	552.79
m_l case										
polska	423.62	452.72	597.96	TL	TL	TL	TL	TL	TL	TL
newyork	140.93	410.63	TL	TL	448.00	318.54	544.48	545.92	TL	TL
france	TL	TL	TL	TL	TL	TL	TL	TL	TL	TL
norway	521.39	596.07	TL	TL	TL	TL	TL	TL	279.66	537.42

Table 6.11 – Computational times used by each heuristic method for tests on the third group of small-medium size instances. TL represents the time limit of 600s.

Approach Step	$\lceil LB \rceil$ SP-VI1+2	Solution provided by each method							
		RFR		SC-RFR		SC-ANS		SC-ANSI	
		NS_{loc}	NS_{locass}	NS_{loc}	NS_{locass}	NS_{loc}	NS_{locass}	NS_{loc}	NS_{locass}
		m1	m1Bis	m2	m2Bis	m3	m3Bis	m4	m4Bis
h_l case									
india35	4	-	-	-	-	17	14	10	9
cost266	3	-	-	-	-	19	13	19	13
giul39	3	18	36	36	36	-	-	-	-
janos-us-ca	3	-	-	-	-	30	30	36	30
pioro40	3	-	-	-	-	40	40	40	40
germany50	4	6	6	-	-	-	-	-	-
m_l case									
india35	4	-	-	-	-	11	7	14	9
cost266	3	-	-	-	-	19	13	19	13
giul39	-	18	36	36	36	-	-	-	-
janos-us-ca	3	-	-	33	27	30	36	30	36
pioro40	3	-	-	-	-	40	40	40	40
germany50	4	6	6	-	-	-	-	-	-

Table 6.12 – Solution provided by each NS heuristic method for tests on large size instances with a time limit of 7200s. (- means no feasible solution found within the time limit.)

Table 6.12 reports the results of tests on large instances (from india35 to germany50). In column 2 we present the rounded lower bound found by SP-VI1+2 with the time limit of 7200s. As before, columns 3-10 report the solution obtained by each local search heuristic method.

In terms of number of instances for which a method can provide a feasible solution, methods m3, m3Bis, m4 and m4Bis are able to find solution for 8 out of 12 instances. m1 and m1Bis can only find solution for 4 instances, while m2 and m2Bis for 3 instances. When we look at the results provided by m1, m1Bis, m2 and m2Bis, we find that for the instances where m1 and m1Bis provide a feasible solution while m2 and m2Bis do not ("germany50"), the reason is that when the time limit of 7200 is reached, m2 and m2Bis are still executing the SC procedure for finding an initial feasible solution. In fact, the SC procedure solves the relaxation of $SP_{fixnum}{}^{MD}$-VI1 with a time limit of 180s in a loop to find a number of VNF instances to fix in the integer problem. When the instance is challenging, the solver cannot even solve the relaxation in 180s. Therefore, when the number of node is bigger than 40, the SC procedure (i.e., used in m2, m2Bis, m3, m3Bis, m4 and m4Bis) may spend all the time (40*180=7200s).

When we compare the results provided by steps NS_{loc} (unsed in methods mX) and NS_{locass} (used in methods mXBis), we can get the same conclusion as with results on the third group of small-medium size instances: NS_{locass} seems to work more efficiently than NS_{loc}. To be exact, NS_{locass} can provide a no worse solution than NS_{loc} for almost all the solved instances. The only exception is topology giul39, where NS_{loc} finds 18 while NS_{locass} finds 36. However, result details show that this is due to the fact that the value of k_d is too small to contain a better solution, m1Bis stops as it cannot improve the current solution. Therefore, $\text{NS}_{\text{locass}}$ is in general more efficient than NS_{loc}, but when it get stuck, we would need to increase k_d (e.g., to $|D|$) to allow it to get out of the local optimal solution.

6.6 Conclusion

In this chapter we proposed four approaches for computing an initial feasible solution for the $\text{VNF-PR}_{\text{SP}}$ problem and we developed ten heuristic methods for efficiently solving the problem. Computational results show that our proposed methods can find good solutions for most of the test instances for which the model cannot provide any feasible solution. Moreover, we observe that the instances with strict capacity limits (i.e, the m_l case) are not necessarily harder to solve than the instances with looser capacity limits (i.e., h_l). In fact, it is sometimes easier to find solutions for instances of m_l case than for instances of h_l case (e.g., "newyork" of m_l and h_l in Table 6.10), since the solver may take more time to explore the larger solution region of h_l case, as a results, no feasible solution is provided within the timelimit.

As for the performance of the approaches for computing an initial feasible solution, in general, the RFR approach is able to provide quickly an initial feasible solution but the gap with the optimal solution is big, whereas SC-ANSI may provide a close-to-optimal initial feasible solution but it needs more computational time. Compared with RFR, SC-RFR integrates the SC procedure at the beginning of algorithm to check the problem feasibility by solving ${\text{SP}_{\text{fixnum}}}^{\text{MD}}$-VI1 model with a given number of services. Therefore, it may provide a better solution at the cost of larger computational time. Then in SC-ANS, if the SC procedure fails to find an initial feasible solution, it tries to find a feasible assignment and routing solution with services all opened.

The results of the proposed heuristic methods show that the $\text{NS}_{\text{locass}}$ improving step is to be preferred to NS_{loc}, but it is necessary to enlarge its neighborhood (namely the parameter k_d) to allow moving out of local optimal solutions.

After analyzing the results, we come up with several ideas to further improve the performance of our proposed heuristic methods: (1) increase k_d to $|D|$ when the local search gets stuck in the $\text{NS}_{\text{locass}}$ step; (2) impose a time limit (e.g. one third of the total time limit) for the SC procedure in the approaches for obtaining an initial feasible solution; (3) use a bisection algorithm on the number of VNF instances (as in method m5) to accelerate the SC procedure for searching for the number of VNF instances to install (instead of increasing one by on or opening a service on all the nodes). Preliminary results along these lines are encouraging.

Chapter 7

General Conclusion

Contents

7.1 Summary of Contributions 143
7.2 Future Research Directions 144

7.1 Summary of Contributions

In this thesis, we focused on a key problem in NFV, i.e., the NFV service chaining: we are given a network where some nodes are connected with computational servers and a set of demands asking for network services composed of a sequence of VNF instances, VNF instances need to be installed on servers and the demands must be routed in such a way that each demand accesses the requested VNFs. More formally, we considered the VNF-PR (VNF placement and routing) problem that can be schematically described as follows: the NFVI network is represented by a graph $G(N, A)$, where N is the set of NFVI nodes and A the set of links between nodes, each node $i \in N$ can host a limited number of VNF instances, and each link $(i, j) \in A$ can allow a limited quantity of flow to pass by. We are given a set of traffic demands D, each demand $k \in D$ is characterized by a source $o_k \in N$, a destination $t_k \in N$, a nominal bandwidth d_k, and a set of VNF types that provide the network services required by the demand. The problem is to decide on which nodes to install the VNF instances and to route the demands from the source node to the destination node traversing the nodes that host their required VNF instances, so that the utilization of the overall network resources are optimized.

We first tackled the VNF-PR problem considering realistic features, such as flow compression/decompression due to VNFs, routing and VNF forwarding delay, etc. We proposed a linear programming formulation and a math-heuristic for efficiently solving the problem with multiple objectives and large instances. Extensive experiments allowed us to draw conclusions on the trade-off achievable between classical TE (i.e., the minimization of maximum link utilization) and NFVI efficiency (measured as the minimization of CPU usage on servers/nodes) goals.

In the second part of our work, we tackled the VNF-PR$_{\text{SP}}$ problem that represents the core features of a general VNF-PR problem and where a same single VNF instance is asked by each demand. First, we studied the problem complexity and proved that the problem is in its general form NP-complete. We further investigate the problem properties and proved that:

- the single service case is equivalent to the multiple services one, when the services have the same capacity and the sequence of services is the same for all the demands (and node capacity is not limited);
- an VNF instance must be installed on the articulation points that belong to biconnected components with internal demands.

Based on the study of the problem complexity and properties, we developed preprocessing algorithms that are able to help in speeding up the computational time to solve the formulations. Furthermore, we generated a large common test bed based on instances from SNDLib and we used it to compare the most promising formulation strategies, namely PR (Placement and Routing) and SP (Split Path) both theoretically and computationally. We proved the the continuous relaxation of SP always provides a bound not worse than the one of PR, which is confirmed by the computational results. Afterwards, we extended the formulations to address more general versions of the problem: the multi-services and multi-sequences cases and the capacitated node one.

Finally, we developed ILP based exact and heuristic methods for the VNF-PR$_{\text{SP}}$ problem. We also proposed several strategies for quickly obtaining an initial feasible solution for hard instances. Computational results show that the proposed heuristic methods can solve efficiently medium size instance of challenging capacity cases to optimum and provide feasible solutions for most of the large size instances of the most difficult cases.

7.2 Future Research Directions

In this thesis, we mainly focus on the VNF-PR$_{\text{SP}}$ problem, thoroughly studying the problem complexity and properties, analysing different formulations, and proposing efficient heuristic methods. As NFV is a recent network paradigm still in its maturing phase, there are many problem variants that we could not consider. Natural extensions of our work can be:

1. complete the study of the problem complexity;
2. evaluate and compare the extended versions of the formulations considering the multi-services and the multi-sequences cases (and the capacitated node case);
3. improve the proposed heuristic methods following the guidelines presented at the end of Chapter 6;
4. extend our proposed heuristic methods to deal with more realistic VNF-PR versions, such as the detailed version studied in Chapter 3 and/or the on-line case.

Résumé de la Thèse

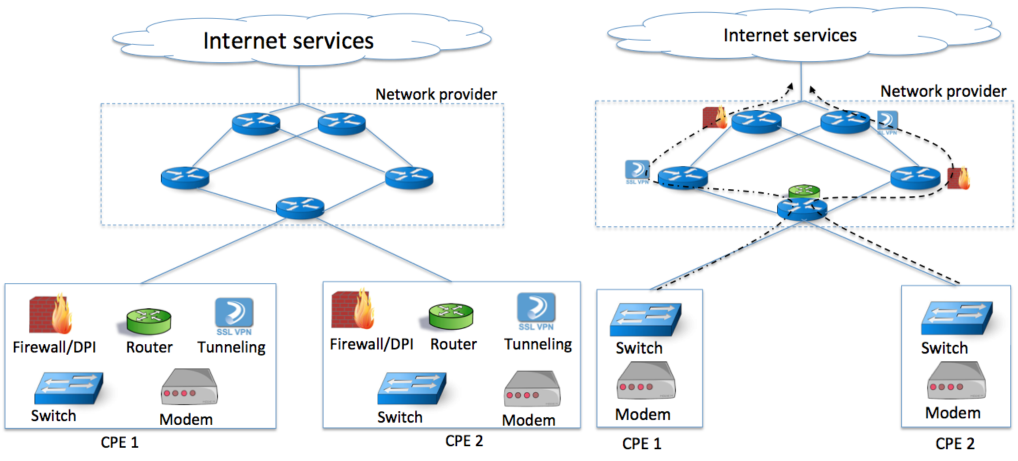
Contexte

Les réseaux des opérateurs se composent d'équipements dédiés ayant des fonctions diverses (pare-feu, routage, etc.) pour fournir aux clients des services réseaux. Avec l'évolution des services de télécommunications et des trafics de données multimédia, les opérateurs ont déployé de nombreux équipements dans le but d'augmenter la capacité et les fonctionnalités des réseaux. La mise en place et le déploiement des nouveaux services réseaux sont devenus de plus en plus difficiles et coûteux, et par conséquent, l'allocation efficace des ressources d'un réseau est devenue un enjeu majeur lors de la panification du réseau.

La virtualisation des fonctions réseau (Network Function Virtualization, NFV) a été proposée comme un nouveau paradigme pour réduire les coûts liés à l'acquisition et à la maintenance pour les réseaux de télécommunications. Le principe de NFV se traduit par l'extrait des fonctions réseaux des équipements dédiés pour les faire exécuter dans un environnement virtualisé. Ceci permet d'éteindre des serveurs virtuels pour économiser de l'énergie, en migrant les fonctions réseau virtuelles à la demande ou automatiquement en fonction de l'état du réseau afin d'améliorer les performances. Ceci permet également de créer des services réseau logiques appelés chaînes de service NFV en fonction de la demande des clients. Plus précisément, ces chaînes de service NFV sont des fonctions réseau virtualisées (Virtual Network Functions, VNFs) interconnectant dans un ordre spécifique par des liens virtualisés, et c'est grâce à cela que le déploiement des services réseaux peut se faire par le chaînage des VNFs. L'objectif de ce dernier est de déterminer de façon optimale l'ensemble de localisation (dans le réseau physique) et d'affectation des VNFs et d'acheminement des demandes de service réseau considérant plusieurs contraintes (par exemple, respecter le délai de service). Par conséquent, le paradigme NFV offre une gestion et un contrôle optimisé de l'infrastructure physique de façon flexible en termes d'automatisation et d'évolutivité.

Figure 1 présente un exemple d'un cas d'usage de NFV, dans lequel des chaînes des VNFs sont créées et déployées en utilisant des VNFs installés et gérés dans le côté de l'opérateur (Figure 1b) pour fournir les services demandés par des clients (c'est-à-dire, affecter les demandes aux VNFs installés et les router de bout en bout).

Dans ce travail de thèse, nous nous intéressons aux problèmes du chaînage des VNFs qui combinent des décisions de localisation, de dimensionnement et de routage, qui sont particulièrement difficiles à résoudre en pratique. L'objectif de cette thèse



(a) CPE traditionel (b) CPE virtualisé

Figure 1 – CPE (Customer Premises Equipment) traditionel et CPE virtualisé.

est de proposer des modèles d'optimisation et des méthodes de résolution pour résoudre efficacement ces problèmes, c'est-à-dire déterminer la bonne localisation des VNFs et l'affectation des VNFs ainsi que le routage des demandes de client afin d'allouer des ressources de façon efficace et finalement réduire le coût des infrastructures. Nous allons présenter par la suite les problématiques traitées dans cette thèse ainsi que nos contributions principales, et nous fournissons également un résumé dans le Tableau 1.

Problématiques et contributions principales

Dans la première partie de cette thèse, nous considérons une version réaliste du problème du chaînage des VNFs tenant en compte de plusieurs caractéristiques complexes du système NFV: les capacités limitées au niveau des nœuds et des serveurs ainsi que des liens, l'ordre spécifique des VNFs des types différents et la variation du flux de demande causé par des traitements de fonction réseau, la latence d'acheminement de demande de bout en bout. Aucun modèle proposé dans la littérature ne considère toutes ces caractéristiques réunies. Notre travail vise à proposer un modèle de programmation linéaire à nombre entier et une méthode de solution associée permettant d'analyser plusieurs cas d'étude. Plus précisément, la modélisation doit (1) localiser et allouer les VNFs qui coexistent sur une même infrastructure physique et (2) acheminer et servir les demandes de client de bout en bout considérant plusieurs contraintes réelles. Les objectifs sont d'une part le partage équilibré de la bande passante mesuré par l'utilisation maximale des liens et d'autre part la réduction du coût des infrastructures NFV en termes de nombre de CPUs utilisés. Pour ce faire, nous proposons un modèle générique appelé PR (Placement and Routing) prenant en compte les aspects réels mentionnés précédemment, ainsi qu'une heuristique mathématique pour aider à la résolution. L'idée principale du modèle PR est de localiser

Problème	VNF-PR	VNF-PR$_{\text{SP}}$
Orientation	télécommunications	optimizations
Chapitres	Chapitre 3	Chapitres 4-6
Hypothèses	-multiple types de VNF, -capacité limitée (node, VNF et lien), -flot de demande indivisible, -chemin de routage sans boucle, -ordre des VNFs fixe, -latence, -variation de flot	-1 seul type de VNF, -capacité limitée (VNF et lien), -flot de demande indivisible, -chemin de routage sans boucle
Focus	-avatages de chaînage des VNFs, -impacts de chaînage des VNFs	-problème complexité et propriétés, -méthode de résolution
Modèles	-PR, -variantes de PR par rajout de contraintes	-PR et SP, -variantes de PR avec inégalités valides, -variantes de SP avec inégalités valides, -variantes de SP avec borne améliorée, -variantes de SP pour version MD
Méthodes	-math-heuristique -dichotomie	-méthodes constructives: TS-BP-FS et FC, -méthodes heuristiques basées sur recherche local: RFR, SC-RFR, SC-ANS et SC-ANSI, -méthodes heuristiques en fixant partiellement la solution: FO

Table 1 – Résumé des problématiques et contributions principales.

des VNFs et router des demandes de bout en bout passant les VNFs demandées (pour être servies) d'après l'ordre spécifique des VNFs. À travers cette modélisation, nous pouvons partager efficacement les ressources réseaux entre plusieurs demandes. Cette formulation peut être étendue pour gérer une grande variété de paramètres clés.

La question critique de ce problème VNF-PR (VNF Placement and Routing) est comment réduire le coût des infrastructures sans affecter la performance du réseau, car il est nécessaire de consolider des VNFs pour réduire le coût des ressources, cependant la consolidation de flux peut augmenter le risque de saturation en bande passante. Pour mesurer l'impact du chaînage des VNFs sur utilisation de bande passante et déterminer comment profiter des avantages du système NFV, nous résoudrons tout d'abord le problème de la minimisation de l'utilisation maximale des liens, ensuite, nous résoudrons le problème qui minimise le coût des infrastructures NFV gardant l'utilisation maximale des liens comme une borne de bande passante à respecter par rajout d'une contrainte dans le modèle.

Nous évaluons les performances de notre solution sous divers scénarios avec un large ensemble de paramètres réels. Le résultat principal montre que le déploiement de service avec le chaînage des VNFs permet de réduire le coût des infrastructures NFV sans causer des problèmes de performances de réseau en termes de la latence des demandes et de l'utilisation des liens. Par ailleurs, l'ensemble de nos tests a mis en évidence que la recherche de la solution optimale du modèle PR est très couteuse en termes de temps de calcul. En plus de ces constats, l'intérêt grandissant de la communauté des télécommunications pour disposer des solutions efficaces pour ce problème nous ont conduits au développement de la suite du travail de cette thèse.

En effet, la deuxième partie de la thèse s'intéresse à étudier des propriétés du problème d'optimisation et proposer des méthodes de résolution efficaces. Dans ce but, nous avons étudié une version compacte du problème du chaînage des VNFs, dans lequel nous gardons un unique type de VNF et nous considérons seulement des contraintes sur la capacité des nœuds et des liens. L'objectif est de trouver la bonne localisation des VNFs et l'acheminement sans boucle des demandes minimisant le nombre de VNFs installées. D'un point de vue d'optimisation, ce problème est une combinaison des problèmes de localisation (pour la partie d'installation des VNFs) et de conception de réseaux (pour la partie de routage). Bien que ces deux problèmes aient été largement étudié dans la littérature, leur combinaison représente des divers défis en termes de modélisation et de résolution. Cette partie de notre travail est donc orientée vers la communauté d'optimisation pour apporter une meilleure compréhension du problème afin de proposer de bonnes méthodes de solution.

Dans cette phase, nous commençons par des études sur la complexité et les propriétés du problème que nous avons appéllé VNF-PR$_{\text{SP}}$ (VNF Placement and Routing on Simple Path). Nous démontrons que le problème étudié est NP-difficile. Ensuite, nous démontrons que (1) en matière de localisation et affectation des VNFs, le cas de type unique de VNF est équivalent au cas de types multiples de VNF lorsque les VNFs ont toutes la même capacité et type, ainsi que l'ordre de VNFs requis sont le même pour toutes les demandes (Proposition 4.3.1); (2) sous hypothèse de routage en chemin sans boucle, une instance de VNF doit être installée sur les points d'articulation appartenant à des composants bi-connectés qui ont des demandes internes et un unique point d'articulation (Proposition 4.3.2). Grâce à la propriété 4.3.2, il est possible de déterminer une borne inférieure sur le nombre de VNFs à installer en considérant le nombre et la structure des composants bi-connectés avec des informations des demandes. De plus, une partie de solution de VNF-PR$_{\text{SP}}$ peut être construite en installant des VNFs sur des points d'articulation. Par conséquent, nous développons un algorithme de pré-processing basé sur cette propriété pour faciliter la résolution. Plus précisément, cet algorithme vise à (1) trouver le nombre de composants bi-connectés avec demande interne et un unique point d'articulation, puis installer une instance de VNF sur chacun de leurs points d'articulation; (2) éviter l'affectation d'une demande aux VNFs qui sont en dehors du composant bi-connecté auquel la demande appartient.

Pour fournir des solutions exactes du problème VNF-PR$_{\text{SP}}$, nous proposons et comparons deux modèles (PR et SP) basés sur les stratégies de formulation les plus prometteuses dans la littérature. Comme présentée précédemment, la stratégie du modèle PR est d'utiliser des variables de flux d'arc pour router les demandes et les chemins avec boucles sont enlevées en exploitant les étiquettes de nœud. L'idée principale du modèle SP est de diviser chaque chemin de demande en deux sous-chemins: l'un connecte la source et le VNF nœud, l'autre connecte le VNF nœud et la destination. Nous comparons théoriquement et numériquement ces deux modèles. Nous démontrons que la relaxation continue de SP fournit toujours une meilleure borne que celle de PR. Les résultats des tests numériques confirment que SP a une meilleure performance que PR.

Quand un réseau a un nombre croissant de nœuds et demandes, la résolution du modèle de la programmation linéaire à nombre entier devient problématique (les temps de calcul et/ou l'utilisation de la mémoire sont très élevés). C'est pour cela que la troisième partie de la thèse se focalise sur des méthodes d'optimisation performantes pour atteindre rapidement de bonnes solutions pour des instances de taille grande afin de pouvoire concrétiser les avantages de NFV lié à l'échelle et l'efficacité.

Dans cet objectif, nous avons implémenté dans un premier temps deux approches constructives basées sur la programmation linéaire. L'idée principale de la première approche (appelée TS-BP-FS, Three-Step Bin Packing based Fix-and-Solve) est de rechercher rapidement l'acheminement faisable pour toutes les demandes en résolvant l'ensemble de variantes du modèle SP fixant la localisation et / ou l'affectation des VNFs fournies par la solution optimale d'un problème Bin Packing (BPP) associé. Cette méthode fonctionne bien lorsque la partie de localisation et affectation (du type BPP) domine le problème VNF-PR$_{\mathrm{SP}}$. Toutefois, elle ne permet pas de trouver des solutions de manière efficace dans les cas où la partie de routage domine le problème. Pour gérer cela, nous implémentons la deuxième approche (appelée FC, Fix-and-Check) qui nous permet de vérifier si un nombre de VNFs proposé est une solution faisable pour VNF-PR$_{\mathrm{SP}}$ en cherchant le nombre maximal de demandes qu'il peut servir. Plus précisément, la vérification est effectuée de manière itérative en résolvant un ensemble de variantes du modèle SP fixant partiellement la solution. Les résultats numériques montrent que toutes les deux méthodes sont capables de trouver une solution pour les instances de taille moyenne du problème. Cependant, le temps de calcul est parfois trop long pour des instances dures, par exemple, lorsque la capacité des liens est très faible.

Dans le but de fournir une solution dans un temps de calcul acceptable pour des instances dures, nous avons proposé et développé, dans un deuxième temps, des heuristiques basées sur la recherche locale. Tout d'abord, nous apportons quatre approches pour trouver une solution initiale pour le problème VNF-PR$_{\mathrm{SP}}$. La première approche appelée RFR (Reduced Feasible Region) consiste à chercher la solution dans une région faisable réduite par rajout de la contrainte de branchement local afin d'accélérer la résolution. La deuxième approche appelée SC-RFR qui a pour le but d'améliorer l'approche RFR en résolvant tout d'abord la procédure SC (Select-and-Check). Toutes les deux approches ci-dessus peuvent se terminer sans aucune solution trouvée. En effet, dans les deux méthodes, le modèle associé est résolu qu'une seule fois. Pour pallier cet inconvénient, nous proposons d'appliquer une recherche locale dans la troisième (appelée SC-ANS, Assignment Neighborhood Search) et la quatrième (appelée SC-ANSI, c'est-à-dire SC-ANS improved) approche pour trouver une solution initiale. C'est-à-dire, au lieu d'exécuter la procédure RFR, nous résoudrons la procédure ANS ou ANSI qui maximise de manière itérative le nombre de demandes servies jusqu'à ce qu'une solution faisable soit trouvée. Une fois une solution initiale trouvée, nous proposons deux méthodes basées sur la recherche du voisinage pour améliorer la solution de façon itérative.

Afin d'évaluer la performance des méthodes proposées, nous utilisons de divers cas de capacité dans un testbed très étendu. Les tests effectués nous ont permis de montrer que les méthodes proposées sont capables de trouver de bonnes solutions (parfois optimales) dans un temps de calcul raisonnable pour les instances dures où le modèle ne peut fournir aucune solution avec un temps de calcul limité.

Structure de la thèse

Le manuscrit est structuré en sept chapitres. Le Chapitre 1 présente l'introduction générale avec la motivation de cette thèse.

Le Chapitre 2 présente une introduction de NFV en général et le chaînage des fonctions de service dans l'infrastructure NFV en particulier. Il fournit également au lecteur les éléments nécessaires pour comprendre le reste du manuscrit. Nous proposons une classification de l'état de l'art et nous révisons des différentes versions du problème d'optimisation du chaînage des VNFs qui ont été traitées dans la littérature. En plus, nous discutons la relation entre le problème du chaînage des VNFs et des problèmes d'optimisation classiques qui représentent certaines caractéristiques communes.

Le Chapitre 3 présente l'étude sur des aspects pratiques du problème du chaînage des VNFs. Nous étudions et évaluons des impacts de chaînage des VNFs sur le réseau de télécommunications en considérant des paramètres de réseau importants, par exemple, le délai de traitement de VNF, la latence de propagation de demande et le changement de flux de demande. Dans cet objectif, nous définissons une version réelle du problème appelé VNF-PR (localisation et routage des VNFs) et nous proposons une formulation mathématique et une math-heuristique pour la résoudre. Les résultats détaillés sont rapportés avec analyses en matière de localisation des VNFs et de la latence des demandes ainsi que l'utilisation des liens physiques.

Le Chapitre 4 présente l'étude théorique sur le problème du chaînage des VNFs. Nous procédons à étudier la structure et des caractéristiques du problème. Dans cet objectif, nous définissons une version fondamentale (VNF-PR$_{\text{SP}}$) du problème et nous fournissons une analyse approfondie de la complexité et des propriétés. Ensuite, une comparaison entre deux stratégies de formulation prometteuses dans la littérature a été faite à la fois théoriquement et pratiquement. Les résultats numériques sont présentés et analysés à la fin du chapitre.

Les Chapitres 5 et 6 présentent et décrivent des approches proposées pour résoudre efficacement le problème VNF-PR$_{\text{SP}}$, y compris des méthodes basées sur la programmation mathématique et des méthodes constructives (pour résoudre efficacement des instances de taille moyenne) ainsi que des heuristiques basées sur la recherche locale (pour résoudre efficacement des instances de taille grande). Ensuite, nous évaluons la performance des méthodes proposées sous un testbed étendu.

Le Chapitre 7 conclut la thèse par une synthèse de nos contributions et discute des perspectives et extensions possibles de ce travail.

List of Publications

International Journal

- Meihui Gao, Bernardetta Addis, Mathieu Bouet, and Stefano Secci, “Optimal Orchestration of Virtual Network Functions,“ on *Computer Networks*, Volume 142, Pages 108-127, September 2018.

International conference

- Bernardetta Addis, Giuliana Carello, and Meihui Gao, “On the complexity of a Virtual Network Function Placement and Routing problem,“ on *Electronic notes in discrete mathematics, Special issue of joint EURO/ALIO International Conference 2018 on Applied Combinatorial Optimization*, Bologna, Italy, June 2018.

Presentation

- Meihui Gao, Francesca De Bettin, Giuliana Carello, Bernardetta Addis, “Formulations and complexity of the network function virtualization service chaining problem (abstract),“ on *8th International Network Optimization Conference*, Lisboa, Portugal, 2017.

Working paper

- Bernardetta Addis, Giuliana Carello, and Meihui Gao, “On a Virtual Network Functions Placement and Routing problem: ILP-based exact and heuristic methods,“ in preparation, 2019.

- Bernardetta Addis, Giuliana Carello, and Meihui Gao, “On a Virtual Network Functions Placement and Routing problem: some properties and a two formulations' comparison,“ under revision on *Networks*, 2018.

- Bernardetta Addis, Giuliana Carello, Francesca De Bettin, and Meihui Gao, “On a Virtual Network Function Placement and Routing problem: properties and formulations,“ on HAL, 2017.

List of Figures

1.1 General NFV architecture proposed by the ETSI 7
1.2 Traditional Customer Premises Equipment (CPE) compared to virtualized CPE (vCPE) with VNF chaining. 8

2.1 VMP (source from [24]) 15
2.2 VME (source from [30]) 16
2.3 Variants of NFV service chaining problems 17
2.4 An example of NFV business model 24

3.1 Example of VNF forwarding latency profiles. 36
3.2 Duplication of an access node i. 42
3.3 Adopted network topology and PR solution example. 47
3.4 Dependency between the number of VNF types and the problem size. 51
3.5 VNF node distribution across NFVI levels (standard case). 52
3.6 Link utilization empirical CDFs (standard case). 53
3.7 Empirical CDFs of latency components (standard case). 54
3.8 VNF node distribution across NFVI levels (fastpath case). 58
3.9 Link utilization empirical CDFs (fastpath case). 59
3.10 Empirical CDFs of latency components (fastpath case). 60
3.11 Comparison between PR and VNE based algorithms in terms of objective function. 63

4.1 Ring graph for reduction from BPP for the Sc-VNF-PR$_{\text{SP}}$ problem . 74
4.2 Examples of cycle sharing nodes and of isolated cycle 83
4.3 An example of a cycle sharing some nodes with a simple path 85
4.4 The SP relaxation bound is strictly better than the one of PR: a numerical example 87
4.5 The routing provided by the continuous relaxation of SP 88
4.6 The routing solution of continuous relaxation of the PR 88
4.7 Example of unfeasible flow acceptable for PR 90
4.8 Relation between routing and assignment variables in the SP formulation. 90
4.9 The SP relaxation bound is strictly better than the one of PR: a numerical example with biconnected components 91
4.10 Average percentage difference between SP model and PR model . . . 102

4.11 Number of optimal and integer feasible solutions found 103
5.1 The graph extension operated by model SP$^{\mathrm{MD}}$. 116
1 CPE (Customer Premises Equipment) traditionel et CPE virtualisé. 146

List of Tables

2.1 Summary of NFV service chaining deployment environments with main focus . . . 18
2.2 Summary of system performance metrics with short descriptions . . 20
2.3 Main optimization objectives and the corresponding labels . . . 21
2.4 Main optimization constraints and the corresponding labels . . . 22
2.5 A classification of the approaches on the NFV service chaining problem proposed in the literature (DP refers to dynamic programming, VNS refers to variable neighborhood search. . . . 27

3.1 Parameter notations for base model. . . . 37
3.2 Variable notations for base model. . . . 37
3.3 Notations to model forwarding latency. . . . 40
3.4 Notations to model bit-rate variations. . . . 42
3.5 Applicable constraints to VNF-PR problem variations. . . . 45
3.6 Test settings of network resources. . . . 48
3.7 Test settings of traffic demands. . . . 48
3.8 Averaged, minimum and maximum values of the TE objective for Internet and VPN case-studies. . . . 50
3.9 Averaged results of NFVI cost of the TE objective and the TE-NFV objective under Internet and VPN case-studies, with the reduction ratio of cost from TE to TE-NFV (standard case). . . . 51
3.10 NFV cost for different TE goal relaxation levels, with TE-NFV optimization. . . . 55
3.11 Averaged results of NFVI cost of the TE objective and the TE-NFV objective under Internet and VPN case-studies, with the reduction ratio of cost from TE to TE-NFV (fastpath case). . . . 56
3.12 Averaged, maximum and minimum latency components of both the standard and the fastpath cases (F represents ‘Firewall’, D is ‘DPI’ and T is ‘Tunneling ingress’). . . . 61

4.1 Summary of complexity on a general graph . . . 75
4.2 Summary of complexity on a full mesh graph . . . 75
4.3 Summary of complexity on a ring graph . . . 76
4.4 Summary of complexity on a tree-like graph (where each node has one single adjacent/parent node) . . . 76

4.5 Mathematical notation 80
4.6 Routing and assignment/location in the continuous relaxation of SP. 87
4.7 Routing and assignment/location solution in the continuous relaxation of PR. 89
4.8 Routing and assignment/location in the continuous relaxation of SP for the biconnected components case example. 91
4.9 Routing and assignment/location in the continuous relaxation of PR for the biconnected components case example. 92
4.10 Test-instance details 98
4.11 Comparison between the continuous relaxations. 99
4.12 Adding valid inequalities to continuous relaxation: qualitative behavior. 99
4.13 Comparison of the CR bound with respect of the best UB. 101
4.14 Computational time with and without VIs for SP and PR formulations.101
4.15 Comparison of UB and LB found for the different formulations (with and without VIs) with respect to the best known UB and LB. 104
4.16 Average computational times (in seconds) of SP and PR on the first group of small-medium instances. 105
4.17 Average computational times (in seconds) for of SP on the second group of small-medium size instances. 105
4.18 Computational times of SP formulation for large size instances (TL* means that the time limit has been reached and a feasible solution is found, while TL means that not even a feasible solution has been found within the time limit). 106
4.19 Time increase (in percentage) due to the increase of the number of VNF types (from 1 to 3) (in bold the cases where the overall computational time decreases). "A" denotes the instances for which a feasible solution within the given time limit cannot be found for both one and three service types; "B" denotes the instances where the optimum is found for the single service type but only a feasible solution can be found for the three types case; "C" denotes the instances for which the single service type case can be solved to optimality but no feasible solution is found for the three service type case; "D" denotes the instances for which a feasible solution can be found for the single service type case but no feasible solution is found for the three service types case. 107
4.20 Time reduction (in percentage) obtained by using articulation point preprocessing (in bold the only case where the overall computational time increases) for the SP formulation. 108

5.1 Additional mathematical notation used in method TS-BP-FS. 113
5.2 Applicable constraints to the models used in TS-BP-FS. 115
5.3 Additional mathematical notation used in method FC. 117
5.4 Applicable constraints to models used in FC. 120

5.5 Number of optimal solutions found by TS-BP-FS and SP-VI1+2 and the average computational times used by each for test on small-medium size instances with time limit of 3600s (* the instance janos-us is not counted in the number of optimal solutions found by SP-VI1+2 in both h_l and m_l cases, the solution found is in fact an optimal solution, but the optimality is not proved within time limit). 122
5.6 Number of optimal solutions found by FC and SP-VI1+2 and the average computational times used by each for test on small-medium size and low link capacity instances with time limit of 3600s (* the instance janos-us is not counted in the number of optimal solutions found by SP-VI1+2 in both h_l and m_l cases, the solution found is in fact an optimal solution, but the optimality is not proved within time limit). 123

6.1 Resulting models of the SPI and $SP_{\text{fixnum}}{}^{MD}$-VI1 formulations combined with the additional *local branching constraint*. 127
6.2 Models used in each approach for obtaining an initial feasible solution for VNF-PR_{SP}. 128
6.3 Summary of improving heuristic approaches. 131
6.4 Test-instance details 133
6.5 Initial feasible solution obtained by each approach with a time limit of 600s and the gap with respect to SP-VI1+2 solution with a time limit of 3600s for test on the second group of small-medium size network instances (- means no feasible solution found within time limit. * means optimality not proved within time limit). 134
6.6 Computational times used by each approach and the SP-VI1+2 model for test on the second group of small-medium size network instances. 135
6.7 Initial feasible solution obtained and the computational time used by each approach for test on the third group of small-medium size instances. TL represents the time limit of 600s (- means no feasible solution found within time limit). 136
6.8 Solution obtained by the SP-VI1+2 model with a time limit of 3600s and each heuristic method with a time limit of 600s for test on the second group of small-medium size network instances. The gap is computed only for feasible not optimal solutions. We denote the bisection based FO method (Algorithm 9) with m5, and the other FO method with m6 (- means no feasible solution found within time limit). 137
6.9 Computational times used by the model SP-VI1+2 with time limit of 3600s and each heuristic methods with a time limit of 600s for test on the second group of small-medium size network instances. 138
6.10 Solution and the improvement w.r.t initial feasible solution obtained by each heuristic method for tests on the third group of small-medium size instances. - means no feasible solution found within the time limit of 600s. 139

6.11 Computational times used by each heuristic method for tests on the third group of small-medium size instances. TL represents the time limit of 600s. 140
6.12 Solution provided by each NS heuristic method for tests on large size instances with a time limit of 7200s. (- means no feasible solution found within the time limit.) 141

1 Résumé des problématiques et contributions principales. 147

B.1 Mathematical notation used in the CG approach 172
B.2 Primal-dual (constraint-variable) table 174

Bibliography

[1] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, Oct 2015, pp. 171–177.

[2] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, Firstquarter 2016.

[3] M. Chiosi and et al., "Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action," in *SDN and OpenFlow World Congress*, 2012.

[4] ETSI, "Network Functions Virtualization - Introductory White Paper," October 2012. [Online]. Available: https://portal.etsi.org/NFV/NFV_White_Paper.pdf

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

[6] M. Boucadair, C. Jacquenet, Y. Jiang, R. Parker, and K. Naito, "Requirements for service function chaining (SFC)," *IETF Internet-Draft draft-boucadair-sfc-requirements-06*, 2015.

[7] "Network function virtualisation NFV: Architectural framework," https://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf, October 2013.

[8] P. Quinn and T. Nadeau, "Service function chaining (SFC) architecture," https://tools.ietf.org/html/rfc7665, Tech. Rep., 2015.

[9] ——, "Problem statement for service function chaining," https://tools.ietf.org/html/rfc7498, Tech. Rep., 2015.

[10] "Network function virtualisation NFV: Use cases," https://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf, October 2013.

[11] A. Basta, W. Kellerer, M. Hoffmann, H. J. Morper, and K. Hoffmann, "Applying NFV and SDN to LTE mobile core gateways, the functions placement problem," in *Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, &#38; Challenges*, ser. AllThingsCellular '14. New York, NY, USA: ACM, 2014, pp. 33–38. [Online]. Available: http://doi.acm.org/10.1145/2627585.2627592

[12] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC)," *IEEE Network*, vol. 28, no. 6, pp. 18–26, Nov 2014.

[13] J. Wu, Z. Zhang, Y. Hong, and Y. Wen, "Cloud radio access network (C-RAN): a primer," *IEEE Network*, vol. 29, no. 1, pp. 35–41, Jan 2015.

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

[15] R. Riggio, A. Bradai, T. Rasheed, J. Schulz-Zander, S. Kuklinski, and T. Ahmed, "Virtual network functions orchestration in wireless networks," in *2015 11th International Conference on Network and Service Management (CNSM)*, Nov 2015, pp. 108–116.

[16] R. Riggio, A. Bradai, D. Harutyunyan, T. Rasheed, and T. Ahmed, "Scheduling wireless virtual networks functions," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 240–252, June 2016.

[17] D. Dietrich, C. Papagianni, P. Papadimitriou, and J. S. Baras, "Network function placement on virtualized cellular cores," in *2017 9th International Conference on Communication Systems and Networks (COMSNETS)*, Jan 2017, pp. 259–266.

[18] J. F. Riera, X. Hesselbach, E. Escalona, J. A. García-Espín, and E. Grasa, "On the complex scheduling formulation of virtual network functions over optical networks," in *2014 16th International Conference on Transparent Optical Networks (ICTON)*, July 2014, pp. 1–5.

[19] M. Obadia, J. L. Rougier, L. Iannone, V. Conan, and M. Brouet, "Revisiting NFV orchestration with routing games," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2016, pp. 107–113.

[20] J. Elias, F. Martignon, S. Paris, and J. Wang, "Efficient orchestration mechanisms for congestion mitigation in NFV: Models and algorithms," *IEEE Transactions on Services Computing*, vol. 10, no. 4, pp. 534–546, July 2017.

[21] S. D'Oro, L. Galluccio, S. Palazzo, and G. Schembra, "Exploiting congestion games to achieve distributed service chaining in NFV networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 2, pp. 407–420, Feb 2017.

[22] A. Fischer and H. de Meer, "Generating virtual network embedding problems with guaranteed solutions," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 504–517, Sept 2016.

[23] I. Contreras and E. Fernández, "General network design: A unified view of combined location and network design problems," *European Journal of Operational Research*, vol. 219, no. 3, pp. 680 – 697, 2012, feature Clusters. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221711009969

[24] M. Masdari, S. S. Nabavi, and V. Ahmadi, "An overview of virtual machine placement schemes in cloud computing," *Journal of Network and Computer Applications*, vol. 66, pp. 106 – 127, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804516000291

[25] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations.* New York, NY, USA: John Wiley & Sons, Inc., 1990.

[26] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of vDPI functions in NFV infrastructures," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, April 2015, pp. 1–9.

[27] M. Zeng, W. Fang, and Z. Zhu, "Orchestrating tree-type VNF forwarding graphs in inter-DC elastic optical networks," *Journal of Lightwave Technology*, vol. 34, no. 14, pp. 3330–3341, July 2016.

[28] A. Abujoda and P. Papadimitriou, "DistNSE: Distributed network service embedding across multiple providers," in *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*, Jan 2016, pp. 1–8.

[29] J. Soares and S. Sargento, "Optimizing the embedding of virtualized cloud network infrastructures across multiple domains," in *2015 IEEE International Conference on Communications (ICC)*, June 2015, pp. 442–447.

[30] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1888–1906, Fourth 2013.

[31] Z. Allybokus, N. Perrot, J. Leguay, L. Maggi, and E. Gourdin, "Virtual function placement for service chaining with partial orders and anti-affinity rules," *Networks*, vol. 71, no. 2, pp. 97–106. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.21768

[32] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, “Network function placement for NFV chaining in packet/optical datacenters,” *Journal of Lightwave Technology*, vol. 33, no. 8, pp. 1565–1570, April 2015.

[33] C. Develder, M. D. Leenheer, B. Dhoedt, M. Pickavet, D. Colle, F. D. Turck, and P. Demeester, “Optical networks for grid and cloud computing applications,” *Proceedings of the IEEE*, vol. 100, no. 5, pp. 1149–1167, May 2012.

[34] D. Lopez, A. Reid, A. Manzalini, and M.-P. Odini, “Impact of SDN/NFV on business models,” https://sdn.ieee.org/newsletter/january-2016/impact-of-sdn-nfv-on-business-models, January 2016.

[35] R. Guerzoni, R. Trivisonno, I. Vaishnavi, Z. Despotovic, A. Hecker, S. Beker, and D. Soldani, “A novel approach to virtual networks embedding for SDN management and orchestration,” in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–7.

[36] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, “On orchestrating virtual network functions,” in *2015 11th International Conference on Network and Service Management (CNSM)*, Nov 2015, pp. 50–56.

[37] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and S. Davy, “Design and evaluation of algorithms for mapping and scheduling of virtual network functions,” in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, April 2015, pp. 1–9.

[38] T. Lukovszki and S. Schmid, “Online admission control and embedding of service chains,” in *Post-Proceedings of the 22Nd International Colloquium on Structural Information and Communication Complexity - Volume 9439*, ser. SIROCCO 2015. New York, NY, USA: Springer-Verlag New York, Inc., 2015, pp. 104–118. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-25258-2_8

[39] B. Addis, G. Carello, and M. Gao, “On the complexity of a virtual network function placement and routing problem,” in *ENDM - Alio Euro 2018 special issue*, 2018.

[40] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, “Orchestrating virtualized network functions,” *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 725–739, Dec 2016.

[41] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, “Elastic virtual network function placement,” in *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, Oct 2015, pp. 255–260.

[42] M. Ghaznavi, N. Shahriar, S. Kamali, R. Ahmed, and R. Boutaba, “Distributed service function chaining,” *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2479–2489, Nov 2017.

[43] T. W. Kuo, B. H. Liou, K. C. J. Lin, and M. J. Tsai, “Deploying chains of virtual network functions: On the relation between link and server usage,” in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, April 2016, pp. 1–9.

[44] M. C. Luizelli, W. L. da Costa Cordeiro, L. S. Buriol, and L. P. Gaspary, “A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining,” *Computer Communications*, vol. 102, pp. 67 – 77, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366416305485

[45] S. Mehraghdam, M. Keller, and H. Karl, “Specifying and placing chains of virtual network functions,” in *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, Oct 2014, pp. 7–13.

[46] H. Moens and F. D. Turck, “VNF-P: A model for efficient placement of virtualized network functions,” in *10th International Conference on Network and Service Management (CNSM) and Workshop*, Nov 2014, pp. 418–423.

[47] M. Gao, B. Addis, M. Bouet, and S. Secci, “Optimal orchestration of virtual network functions,” *Computer Networks*, vol. 142, pp. 108 – 127, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128618303578

[48] B. Addis, G. Carello, and M. Gao, “On a virtual network functions placement and routing problem: some properties and a two formulations’ comparison,” *under revision on Networks*, 2018.

[49] ——, “On a virtual network functions placement and routing problem: ILP-based exact and heuristic methods,” *working paper*, 2019.

[50] “Network function virtualisation NFV: Terminology for main concepts in nfv,” https://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf, January 2018.

[51] T. Kuo, B. Liou, K. C. Lin, and M. Tsai, “Deploying chains of virtual network functions: On the relation between link and server usage,” *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1562–1576, Aug 2018.

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

[53] H. A. Alameddine, S. Sebbah, and C. Assi, “On the interplay between network function mapping and scheduling in VNF-based networks: A column generation approach,” *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 860–874, Dec 2017.

[54] I. Contreras and E. Fernández, “General network design: A unified view of combined location and network design problems,” *European Journal of Operational Research*, vol. 219, no. 3, pp. 680 – 697, 2012, Feature Clusters.

[55] Intel, “Impact of the Intel Data Plane Development Kit (Intel DPDK) on packet throughput in virtualized network elements,” White Paper, 2009.

[56] ——, “Network Function Virtualization Packet Processing Performance of Virtualized Platforms with Linux* and Intel Architecture,” Technical Report, Oct. 2013.

[57] C. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity.* Mineola, NY: Dover, 1998.

[58] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A mathematical programming language.*

[59] I. I. Cplex, “V12. 1: User’s manual for CPLEX,” *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.

[60] N. Bouten, M. Claeys, R. Mijumbi, J. Famaey, S. Latré, and J. Serrat, “Semantic validation of affinity constrained service function chain requests,” in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, June 2016, pp. 202–210.

[61] M. C. Luizelli, D. Raz, Y. Sa’ar, and J. Yallouz, “The actual cost of software switching for NFV chaining,” in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, May 2017, pp. 335–343.

[62] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, “SNDlib 1.0 – Survivable Network Design library,” *Networks*, vol. 55, no. 3, pp. 276–286, May 2010. [Online]. Available: http://dx.doi.org/10.1002/net.v55:3

[63] M. Fischetti and A. Lodi, “Local branching,” *Mathematical Programming*, vol. 98, no. 1, pp. 23–47, Sep 2003. [Online]. Available: https://doi.org/10.1007/s10107-003-0395-5

Appendix A

Father Node Formulation

We present another possible formulation for the VNF-PR$_{\mathrm{SP}}$ problem. Since the demands are assumed to be routed on simple path, each node on the routing path of a demand has an antecedent (except for the source node) and a successor (except for the destination node). For this, we introduce binary variables p^k_{ij}, equals to 1 if node $i \in N$ is an antecedent node of node $j \in N$ on the routing path of demand $k \in D$. We denote this formulation with Father Node (FN) model.

We have in the FN model the same objective function (A.1) as in PR and SP, and similar constraints (A.2) (flow balance), (A.3) (link capacity), (A.4) (VNF capaciry), (A.5) (single demand-VNF assignment), (A.6) (coherence use-installation of VNF).
Specific FN constraints: (A.7) and (A.8) are to force demand flow pass by the used VNF. Constraint (A.9) and (A.10) are to fix positions of source node and destination node for each demand. Constraint (A.11) is to link the node position (variable p) and the path decision (variable x). Constraint (A.12) prevents up-down paths. Constraint (A.13) tells if a node i is the antecedent node of node j in demand k's path. Constraints (A.14) and (A.15) are to enforce the antecedent nodes' constraint.

Prevent isolated cycles : with constraint (A.11), (A.12) and (A.13), we can prevent isolated cycles. If we have a cycle in the path (variable x), we then know with constraint (A.11) (variable p): node 1 is antecedent to VNF, VNF is antecedent to node 2, and node 2 is antecedent to node 1. And according to constraint (A.13), if node 1 is antecedent to VNF and VNF is antecedent to node 2, then node 1 is antecedent to node 2 (i.e., $p^k_{12} \geq p^k_{1vnf} + p^k_{vnf2} - 1$), and this is not allowed by constraint (A.12): $p^k_{12} + p^k_{21} \leq 1$, which says that both p^k_{12} and p^k_{21} equal to 1 is not allowed.

$$\min \sum_{i \in N} \sum_{f \in F} y_{if} \tag{A.1}$$

$$\text{s.t.} \sum_{j:(i,j)\in A} x_{ij}^k - \sum_{j:(j,i)\in A} x_{ji}^k = \begin{cases} 1 & \text{if } i = o_k \\ -1 & \text{if } i = t_k \\ 0 & \text{otherwise} \end{cases} \qquad \forall k \in D,\ i \in N \tag{A.2}$$

$$\sum_{k \in D} d_k\ x_{ij}^k \ \leq\ u_{ij} \qquad \forall (i,j) \in A \tag{A.3}$$

$$\sum_{k \in D} d_k\ z_{if}^k \ \leq\ qu_f \qquad \forall i \in N, f \in F \tag{A.4}$$

$$\sum_{i \in N} z_{if}^k \ =\ 1 \qquad \forall k \in D, f \in F : vfr_f^k = 1 \tag{A.5}$$

$$z_{if}^k \ \leq\ y_{if} \qquad \forall k \in D,\ i \in N, f \in F \tag{A.6}$$

$$p_{o_k i}^k \geq z_i^k \qquad \forall k \in D, f \in F, i \in N : i \neq o_k, vfr_f^k = 1 \tag{A.7}$$

$$p_{it_k}^k \geq z_i^k \qquad \forall k \in D, f \in F, i \in N : i \neq t_k, vfr_f^k = 1 \tag{A.8}$$

$$p_{io_k}^k = 0 \qquad \forall k \in D, i \in N : i \neq o_k \tag{A.9}$$

$$p_{t_k i}^k = 0 \qquad \forall k \in D, i \in N : i \neq t_k \tag{A.10}$$

$$p_{ij}^k \geq\ x_{ij}^k \qquad \forall k \in D, (i,j) \in A \tag{A.11}$$

$$p_{ij}^k + p_{ji}^k \leq 1 \qquad \forall k \in D, i \in N, j \in N : i \neq j \tag{A.12}$$

$$p_{ij}^k \geq \begin{cases} 1 & \text{if } i = o_k, j = t_k \\ p_{mj}^k + p_{im}^k - 1 & \text{otherwise} \end{cases} \qquad \begin{array}{r} \forall k \in D, (m,j) \in E, i \in S : \\ i \neq j, i \neq t_k, j \neq o_k \end{array} \tag{A.13}$$

$$p_{ij}^k \leq \sum_{(i,m)\in E} x_{im}^k \qquad \forall j \in N, i \in N : i \neq j \tag{A.14}$$

$$p_{ij}^k \leq \sum_{(m,j)\in E} x_{mj}^k \qquad \forall j \in N, i \in N, i \neq j \tag{A.15}$$

$$p_{ij}^k \geq z_{if_1}^k + z_{jf_2}^k - 1 \qquad \begin{array}{r} \forall j \in N, i \in N, f_1 \in F, f_2 \in F : i \neq j, \\ vfr_{f_1}^k = 1, vfr_{f_2}^k = 1, order_{f_2^k} \geq order_{f_1}^k \end{array} \tag{A.16}$$

Appendix B

Column Generation Approach

B.0.0.1 Introduction of column generation method

As in some large scale linear programs it is not possible to enumerate all the variables, the main idea of Column Generation is to consider only a subset of variables (i.e., *columns*) and then iteratively add the ones which improves the objective value. The variables to be added are chosen exploiting duality properties.

In the procedure of Column Generation, the problem is split into two sub-problems: a **Master** problem and a **Pricing** problem. The master problem is the original linear program, but restricted to a subset of variables. Its dual is used to identify columns with negative reduced costs, hence the ones which improve the objective function when entering the basis. When solved, the Master problem passes a revised set of cost coefficients (*prices*) associated with its dual optimal solution to the Pricing problem. The Pricing problem uses these prices to generate and determine the column that improves the solution of the Master problem. Both sub-problems keep on exchanging information until an optimal solution or a stopping condition is met.

Let the optimization ILP problem be a minimization problem of the following form

$$\min \sum_{i\in N} c_i x_i$$

$$s.t. \sum_{i\in N} a_i x_i \geq b$$

$$x_i \in \{0,1\}, \quad \forall i \in N$$

Then the restricted master problem is

$$\min \sum_{i\in N'\subset N} c_i x_i$$

$$s.t. \sum_{i\in N'\subset N} a_i x_i \geq b$$

$$x_i \geq 0, \quad \forall i \in N' \subset N$$

which is initialized with a subset N' of feasible solutions that we refer to by *columns* and which satisfy all of its constraints. Its dual problem is

$$\max\ b^T y$$

$$s.t.\ ya_i \leq c_i, \quad \forall i \in N' \tag{B.1}$$

$$y \geq 0$$

The primal variable with the highest negative *reduced costs* is the one which violates constraint(B.1) the most. Thus in the pricing problem, the evolving problem is

$$\min\ (c_i - ya_i), \quad \forall i \in N$$

Indexes of N correspond to entities which can be described through the feasible domain Ω of an optimization problem. The resulting pricing problem becomes the optimization problem

$$\min\ (c\ (z_k) - ya\ (z_k))$$

$$z_k \in \{0,1\}, \quad \forall k \in \Omega$$

When a negative solution is found, the variable x_i and its coefficient column (c_i, a_i) are added to the restricted mater problem. The master problem is solved again and possibly produces an improved objective function.
This procedure is repeated until no more variables with negative reduced costs can be found (or a stopping condition is met) in the pricing problem. Then the solution is optimal for the LP master problem.

B.0.0.2 An alternative path formulation of SP model for VNF-PR$_{\mathbf{SP}}$ problem

Before presenting the alternative formulation, the modeling of the path construction is explained and the choice for the decision variables justified.

The feasible routes of demands used in formulation SP are called *arc routes.* There exists an alternative formulation that uses a binary variable r_p for each $(s-t)-$ routing path p. An $(s-t)-$ routing path is a sequence of arcs a_1, ... a_n that starts at s and ends at t, such that no node is visited twice by the path (i.e., *simple path*). Let P_{s_k,f_k} be the set of all $(s-f)-$ routing paths (i.e., the simple paths from source node to the service node) of demand $k \in D$, and let P_{f_k,t_k} be the set of all $(f-t)-$ routing paths (i.e., the simple paths from the service node to the source node) of demand $k \in D$. Because of the strategic nature of the SP formulation (i.e., the decomposition of end-to-end routing path into two sub-paths, associated with a VNF instance serving the demand), variables

$$r_{p_1}, \forall p_1 \in P_{s_k,f_k}$$

and

$$\sum_{(i,j)\in A} x_{ij}^{k1}$$

express the same first half of the total end-to-end routing decision of demand $k \in D$. Similarly, variables

$$r_{p_2}, \forall p_2 \in P_{f_k,t_k}$$

and

$$\sum_{(i,j)\in A} x_{ij}^{k2}$$

express the same remaining half of the total end-to-end routing decision of $k \in D$.

In Table B.1 the notation used for the problem is summarized. Furthermore, the variables used by the arc routing formulation and the path routing formulation of SP model are reported. As some sets, parameters and variables are common to both formulations and some are formulation dependent, in the last column we report in which formulation the set/parameter/variable is used.

Notation		Models
Sets		
N	set of nodes	both
A	set of arcs	both
D	set of demands	both
P_{s_k,f_k}	set of simple paths from source node to service node of demand $k \in D$	alternative
P_{f_k,t_k}	set of simple paths from service node to terminal node of demand $k \in D$	alternative
P	set of simple paths: $P = \bigcup_{k \in D} (P_{s_k,f_k} \cup P_{f_k,t_k})$	alternative
Network parameters		
u	arc capacity	both
q	service capacity	both
Demand parameters		
s_k	origin of demand $k \in D$	both
t_k	destination of demand $k \in D$	both
d_k	bandwidth of demand $k \in D$	both
f_k	node position of service that serves demand $k \in D$, with $f_k \in N$	alternative
Path parameters		
src_p	origin of path $p \in P$	alternative
dst_p	destination of path $p \in P$	alternative
srv_p	demand that path $p \in P$ serves	alternative
Binary variables		
y_i	1 if a service is located on node $i \in N$	both
z_i^k	1 if demand $k \in D$ uses the service on node $i \in N$	both
Model-depending variables		
x_{ij}^{k1}	1 if arc $(i,j) \in A$ is used by demand k on sub-path 1	initial
x_{ij}^{k2}	1 if arc $(i,j) \in A$ is used by demand k on sub-path 2	initial
r_p	1 if path $p \in P$ is used	alternative

Table B.1 – Mathematical notation used in the CG approach

The alternative *path routing* formulation of SP Model is

$$\min \sum_{i \in N} y_i$$

$$s.t. \sum_{i \in N} z_i^k = 1 \qquad \forall k \in D \tag{B.2}$$

$$z_i^k \leq y_i \qquad \forall k \in D, i \in N \tag{B.3}$$

$$\sum_{k \in D} d_k z_i^k \leq q \qquad \forall i \in N \tag{B.4}$$

$$\sum_{\substack{p \in P_{s_k, f_k}: \\ dst_p = i}} r_p \; = \; z_i^k \qquad \forall k \in D, i \in N : i \neq s_k \tag{B.5}$$

$$\sum_{\substack{p \in P_{f_k, t_k}: \\ src_p = i}} r_p \; = z_i^k \qquad \forall k \in D, i \in N : i \neq t_k \tag{B.6}$$

$$\sum_{k \in D} \sum_{\substack{p_1 \in P_{s_k, f_k}: \\ (i,j) \in p_1}} d_k r_{p_1} + \sum_{k \in D} \sum_{\substack{p_2 \in P_{f_k, t_k}: \\ (i;j) \in p_2}} d_k r_{p_2} \leq u \qquad \forall (i,j) \in A \tag{B.7}$$

$$\sum_{j:(i,j) \in A} \sum_{\substack{p1 \in P_{s_k, f_k}: \\ (i,j) \in p1}} r_{p1} + \sum_{j:(i,j) \in A} \sum_{\substack{p2 \in P_{f_k, t_k}: \\ (i,j) \in p2}} r_{p2} \leq 1 \qquad \forall k \in D, i \in N \tag{B.8}$$

$$\sum_{j:(j,i) \in A} \sum_{\substack{p1 \in P_{s_k, f_k}: \\ (j,i) \in p1}} r_{p1} + \sum_{j:(j,i) \in A} \sum_{\substack{p2 \in P_{f_k, t_k}: \\ (j,i) \in p2}} r_{p2} \leq 1 \qquad \forall k \in D, i \in N \tag{B.9}$$

$$y_i \in \{0,1\} \qquad \forall i \in N$$

$$z_i^k \in \{0,1\} \qquad \forall k \in D, i \in N$$

$$r_p \in \{0,1\} \qquad \forall p \in P$$

Eq. (B.2), (B.3) and (B.4) are taken from the initial SP formulation. Eq. (B.2) imposes that each demand is assigned to exactly one instance of the service. Eq. (B.3) guarantees that if there is no VNF instance is installed, then it cannot be assigned to any demand. Eq. (B.4) imposes that each instance of the VNF can serve a maximum quantity of demand given by its capacity q.

Eq. (B.5) to (B.9) impose a routing path for each demand, which passes through the requested VNF node. The idea of the *path routing* is still 'SP', i.e., each demand is routed in two sub-paths: one sub-path from the source node to the VNF node (Eq. (B.5)) then the other sub-path from the VNF node to the destination node (Eq. (B.6)). The link capacity constraints is introduced by Eq. (B.7), and the simple path solution (i.e., each demand can pass through a node at most one) is guaranteed by Eq. (B.8) and (B.9).

Note that it is possible to have a path formulation that considers the VIs.

The related dual variables of the *path formulation* are presented in Table B.2.

Primal constraint	Corresponding dual variable
(B.2)	β_k free, $\forall k \in D$
(B.3)	$\alpha_i^k \geq 0, \ \forall k \in D, i \in N$
(B.4)	$\theta_i \leq 0, \ \forall i \in N$
(B.5)	$\lambda_{1,i}^k$ free, $\forall k \in D, i \in N$
(B.6)	$\lambda_{2,i}^k$ free, $\forall k \in D, i \in N$
(B.7)	$\pi_{ij} \leq 0, \ \forall (i,j) \in A$
(B.8)	$\sigma_{1,i}^k \leq 0, \ \forall k \in D, i \in N$
(B.9)	$\sigma_{2,i}^k \leq 0, \ \forall k \in D, i \in N$

Table B.2 – Primal-dual (constraint-variable) table

Given the master problem, and its dual variables, we can find the dual constraints corresponding to the primal variables r_p :

For the first half paths $p \in P_{s,f}$, with $P_{s,f} = \bigcup_{k \in D} P_{s_k,f_k}$:

$$\sum_{i \in N} \sum_{\substack{k \in D: \\ p \in P_{s_k,f_k}}} \sum_{\substack{j:(j,i) \in A: \\ (j,i)=last(p)}} \lambda_{1,i}^k + \sum_{\substack{(i,j) \in A: \\ (i,j) \in p}} d_k \pi_{ij}$$
$$+ \sum_{i \in N} \sum_{\substack{k \in D: \\ p \in P_{s_k,f_k}}} \sum_{\substack{j:(i,j) \in A: \\ (i,j) \in p}} \sigma_{1,i}^k + \sum_{i \in N} \sum_{\substack{k \in D: \\ p \in P_{s_k,f_k}}} \sum_{\substack{j:(j,i) \in A: \\ (j,i) \in p}} \sigma_{2,i}^k \leq 0$$

For the remaining half paths $p \in P_{f,t}$, with $P_{f,t} = \bigcup_{k \in D} P_{f_k,t_k}$:

$$\sum_{i \in n} \sum_{\substack{k \in D: \\ p \in P_{f_k,t_k}}} \sum_{\substack{j:(i,j) \in A: \\ (i,j)=first(p)}} \lambda_{2,i}^k + \sum_{\substack{k \in D: \\ p \in P_{f_k,t_k}}} \sum_{\substack{(i,j) \in A: \\ (i,j) \in p}} d_k \pi_{ij}$$
$$+ \sum_{i \in n} \sum_{\substack{k \in D: \\ p \in P_{f_k,t_k}}} \sum_{\substack{j:(i,j) \in A: \\ (i,j) \in p}} \sigma_{1,i}^k + \sum_{i \in n} \sum_{\substack{k \in D: \\ p \in P_{f_k,t_k}}} \sum_{\substack{j:(j,i) \in A: \\ (j,i) \in p}} \sigma_{2,i}^k \leq 0$$

As each path serves exactly one demand, we can polish the above dual constraints by removing the void columns in the constraints. Now we use the parameter srv_p to represent the demand $k \in D : k = srv_p$ that the path $p \in P$ serves. Then the dual constraints become:

For each first half path $p \in P_{s,f}$, with $P_{s,f} = \bigcup_{k \in D} P_{s_k,f_k}$:

$$\lambda_{1,dst_p}^{srv_p} + \sum_{\substack{(i,j) \in A: \\ (i,j) \in p}} d_{srv_p} \pi_{ij} + \sum_{i \in N} \sum_{\substack{j:(i,j) \in A: \\ (i,j) \in p}} \sigma_{1,i}^{srv_p} + \sum_{i \in N} \sum_{\substack{j:(j,i) \in A: \\ (j,i) \in p}} \sigma_{2,i}^{srv_p} \leq 0 \quad \text{(B.10)}$$

For the remaining half paths $p \in P_{f,t}$, with $P_{f,t} = \bigcup_{k \in D} P_{f_k,t_k}$:

$$\lambda_{2,src_p}^{srv_p} + \sum_{\substack{(i,j) \in A: \\ (i,j) \in p}} d_{srv_p} \pi_{ij} + \sum_{i \in N} \sum_{\substack{j:(i,j) \in A: \\ (i,j) \in p}} \sigma_{1,i}^{srv_p} + \sum_{i \in n} \sum_{\substack{j:(j,i) \in A: \\ (j,i) \in p}} \sigma_{2,i}^{srv_p} \leq 0 \quad \text{(B.11)}$$

For each first half path $p \in P_{s,f}$, with $P_{s,f} = \bigcup_{k \in D} P_{s_k,f_k}$:

$$\lambda_{1,dst_p}^{srv_p} + \sum_{\substack{(i,j)\in A: \\ (i,j)\in p}} (d_{srv_p}\pi_{ij} + \sigma_{1,i}^{srv_p} + \sigma_{2,j}^{srv_p}) \leq 0 \tag{B.12}$$

For the rest half paths $p \in P_{f,t}$, with $P_{f,t} = \bigcup_{k \in D} P_{f_k,t_k}$:

$$\lambda_{2,src_p}^{srv_p} + \sum_{\substack{(i,j)\in A: \\ (i,j)\in p}} (d_{srv_p}\pi_{ij} + \sigma_{1,i}^{srv_p} + \sigma_{2,j}^{srv_p}) \leq 0 \tag{B.13}$$

B.0.0.3 Decomposition strategy

As mentioned in Section B.0.0.1, the Column Generation solves very large linear programs by considering a small subset of the variables at once, based on a primal-dual decomposition procedure. For the Master problem, we have to relax the *path routing* formulation and consider a startset of paths. For the Pricing sub-problem, we have to evaluate the sub-paths for each demand according to the dual constraints Eq. (B.10) and (B.11). Furthermore, an efficient method to generate a feasible start-set of paths has to be designed.

Master problem.
The restricted master problem should be a LP and should consider only a subset of paths. Hence the decision variables of *path routing* formulation of the VNF-SP$_{\text{SP}}$ problem should be relaxed:

$$\begin{aligned} y_i &\in [0,1] && \forall i \in N \\ z_i^k &\in [0,1] && \forall k \in D, i \in N \\ r_p &\in [0,1] && \forall p \in P \end{aligned}$$

Then the restricted mater problem and its dual variables:

$$
\begin{aligned}
\min \ & \sum_{i\in N} y_i && && \\
s.t.\ & z_i^k \le y_i && \forall k\in D, i\in N && \alpha_i^k \ge 0\\
& \sum_{i\in N} z_i^k = 1 && \forall k\in D && \beta_k\ free\\
& \sum_{k\in D} d_k z_i^k \le q && \forall i\in N && \theta_i \le 0\\
& \sum_{\substack{p\in P_{s_k,f_k}:\\ dst_p=i}} r_p = z_i^k && \forall k\in D, i\in N : i\neq s_k && \lambda_{1,i}^k\ free\\
& \sum_{\substack{p\in P_{f_k,t_k}:\\ src_p=i}} r_p = z_i^k && \forall k\in D, i\in N : i\neq t_k && \lambda_{2,ji}^k\ free\\
& \sum_{k\in D}\sum_{\substack{p_1\in P_{s_k,f_k}:\\ (i,j)\in p_1}} d_k r_{p_1} + \sum_{k\in D}\sum_{\substack{p_2\in P_{f_k,t_k}:\\ (i;j)\in p_2}} d_k r_{p_2} \le u && \forall (i,j)\in A && \pi_{ij}\le 0\\
& \sum_{j:(i,j)\in A}\sum_{\substack{p1\in P_{s_k,f_k}:\\ (i,j)\in p1}} r_{p1} + \sum_{j:(i,j)\in A}\sum_{\substack{p2\in P_{f_k,t_k}:\\ (i,j)\in p2}} r_{p2} \le 1 && \forall k\in D, i\in N && \sigma_{1,i}^k \le 0\\
& \sum_{j:(j,i)\in A}\sum_{\substack{p1\in P_{s_k,f_k}:\\ (j,i)\in p1}} r_{p1} + \sum_{j:(j,i)\in A}\sum_{\substack{p2\in P_{f_k,t_k}:\\ (j,i)\in p2}} r_{p2} \le 1 && \forall k\in D, i\in N && \sigma_{2,i}^k \le 0
\end{aligned}
$$

Pricing problem.
The aim of the pricing problem is to find a $(s-t)-$ routing path for each demand, which violates both constraints (B.12) and (B.13). As we divide the end-to-end routing path into two independent sub-paths for each demand, we can build separately the two sub-paths. For each demand $k\in D$ and each service node $f_k \in N$, the goal is thus to find two sub-paths of k using the service on node f_k that violates (B.12) and (B.13).
The objective of the pricing sub-problems are:

Pricing sub-problem 1: find a path $p\in P_{s_k,f_k}$ for demand $k\in D$ such that

$$
-\sum_{(i,j)\in A} (d_{srv_p}\pi_{ij} \ + \ \sigma_{1,i}^k \ + \ \sigma_{2,j}^k) < \lambda_{1,f_k} \tag{B.14}
$$

Pricing sub-problem 2: find a path $p\in P_{f_k,t_k}$ for demand $k\in D$ such that

$$
-\sum_{(i,j)\in A} (d_{srv_p}\pi_{ij} \ + \ \sigma_{1,i}^k \ + \ \sigma_{2,j}^k) < \lambda_{2,f_k} \tag{B.15}
$$

Therefore, for each demand $k\in D$, we look for the shortest sub-path from the source s_k to the service node f_k and the shortest sub-path from the service node f_k

to the destination t_k on a graph on which the cost of arc (i, j) is

$$-(d_{srv_p}\pi_{ij} \ + \ \sigma^k_{1,i} \ + \ \sigma^k_{2,j})$$

- If in the Pricing sub-problem 1 we find a shortest sub-path of demand d which is less than λ_{1,f_k}, i.e., the dual constraint is violated, we add this shortest path to the restricted master problem.
- Similarly, if in the Pricing sub-problem 2 we find a shortest sub-path which is less than λ_{2,f_k}, we also add this path to the restricted master problem.
- Otherwise, i.e., if for all the demands, the shortest path found by Pricing sub-problem 1 is greater than λ_{1,f_k}, and the shortest path found by Pricing sub-problem 2 is greater than λ_{2,f_k}, then the solution of the RMP (restricted master problem) is optimal for the original problem.

In order to generate such path, we use the following decision variables in the pricing problems:

- In Pricing sub-problem 1, $x^1_{i,j} \in \{0, 1\}$ if arc $(i, j) \in A$ is used by the first half path $p \in P_{s_k,f_k}$
- In Pricing sub-problem 2, $x^2_{i,j} \in \{0, 1\}$ if arc $(i, j) \in A$ is used by the rest half path $p \in P_{f_k,t_k}$

And the constraints which impose the routing paths in each pricing sub-problem are:

Pricing sub-problem 1

$$\sum_{j:(i,j)\in A} x^1_{ij} - \sum_{j:(j,i)\in A} x^1_{ji} = \begin{cases} 1 & \text{if } i = s_k, \text{and } i \neq f_k \\ -1 & \text{if } i = f_k \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in N \qquad \text{(B.16)}$$

Pricing sub-problem 2

$$\sum_{j:(i,j)\in A} x^2_{ij} - \sum_{j:(j,i)\in A} x^2_{j,i} = \begin{cases} -1 & \text{if } i = t_k, \text{and } i \neq f_k \\ 1 & \text{if } i = f_k \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in N \qquad \text{(B.17)}$$