UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Vers les Réseaux de Nouvelle Génération avec SDN et NFV

Andrea Tomassilli

Laboratoire d'informatique, Signaux et Système de Sophia Antipolis

Présentée en vue de l'obtention du grade de docteur en Informatique **d'**Université Côte d'Azur
Dirigée par : Frédéric Giroire / Stéphane Pérennes
Soutenue le : 24 Juin 2019

Devant le jury, composé de :
Mathieu Bouet, *Ingénieur, Thales*
Michele Flammini, *Professeur, Università degli Studi dell'Aquila*
Frédéric Giroire, *Chargé de recherche, CNRS*
Brigitte Jaumard, *Professeur, Concordia University*
Stéphane Pérennes, *Directeur de recherche, CNRS*
Stefano Secci, *Professeur, Cnam*
Thierry Turletti, *Directeur de recherche, Inria*

Acknowledgements

This Thesis would not have been possible without the guidance of my PhD advisors Frédéric Giroire and Stéphane Pérennes. Thank you for the continued support during my PhD study, for the great patience, for your faith and enthusiasm, and for having shared your experiences with me. I was very fortunate to work with you.

I would like to also express my deep and warm acknowledgements to Mathieu Bouet and Stefano Secci, who both thoroughly reviewed my Thesis, and evaluated my work. Thank you for your time and effort taken to review this report and for the valuable comments. You contributed to improve this revised version of the manuscript.

I would also like to express my sincere gratitude to all the members of the examination committee: Michele Flammini, Brigitte Jaumard, and Thierry Turletti. Thank you for having accepted my request to evaluate my Thesis and research work.

I'm thankful to all the members of the COATI research group for making me feel welcomed from the very first moment. The three years spent with you made this journey much more fun. Special thanks to Fionn for the careful English reviews made on my drafts over these years.

Also, I would like to thank the teams I had the opportunity to visit and collaborate with during these 3 years. In particular, the Computer Science and Software Engineering Department of Concordia University, the RealOpt team of Inria Bordeaux, the Network Protocols & Systems Research Team of Nokia Bell-Labs, and the DIANA team of Inria Sophia Antipolis. I've learned a lot from each and every one of you.

Thanks to all my friends who were with me during these years and for the great times we spent together, the laughs, the chats, and the parties.

To conclude, I would like to thank all my family for having supported me in every possible way.

Résumé

Les progrès récents dans le domaine des réseaux, tels que les réseaux logiciel (SDN) et la virtualisation des fonctions réseaux (NFV), modifient la façon dont les opérateurs de réseaux déploient et gèrent les services Internet.
D'une part, SDN introduit un contrôleur logiquement centralisé avec une vue globale de l'état du réseau. D'autre part, NFV permet le découplage complet des fonctions réseaux des appareils propriétaires et les exécute en tant qu'applications logicielles sur des serveurs génériques. De cette façon, les opérateurs de réseaux peuvent déployer dynamiquement des fonctions réseaux virtuelles (VNF).
SDN et NFV, tous deux séparément, offrent aux opérateurs de nouvelles opportunités pour réduire les coûts, améliorer la flexibilité et le passage à l'échelle des réseaux et réduire les délais de mise sur le marché des nouveaux services et applications. De plus, le modèle de routage centralisé du SDN, associé à la possibilité d'instancier les VNF à la demande, peut ouvrir la voie à une gestion encore plus efficace des ressources réseaux. Par exemple, un réseau SDN/NFV peut simplifier le déploiement des chaînes de fonctions de services (SFC) en rendant le processus plus facile et moins coûteux.
Dans cette thèse, notre objectif était d'examiner comment tirer parti des avantages potentiels de combiner SDN et NFV. En particulier, nous avons étudié les nouvelles possibilités offertes en matière de conception de réseau, de résilience et d'économies d'énergie, ainsi que les nouveaux problèmes qui surgissent dans ce nouveau contexte, comme l'emplacement optimal des fonctions réseaux.
Nous montrons qu'une symbiose entre le SDN et le NFV peut améliorer la performance des réseaux et réduire considérablement les dépenses d'investissement (CapEx) et les dépenses opérationnelles (OpEx) du réseau.

Abstract

Recent advances in networks such as Software Defined Networking (SDN) and Network Function Virtualization (NFV) are changing the way network operators deploy and manage Internet services.
On one hand, SDN introduces a logically centralized controller with a global view of the network state. On the other hand, NFV enables the complete decoupling of network functions from proprietary appliances and runs them as software applications on general–purpose servers. In such a way, network operators can dynamically deploy Virtual Network Functions (VNFs).
SDN and NFV, both separately, bring to network operators new opportunities for reducing costs, enhancing network flexibility and scalability, and shortening the time-to-market of new applications and services.
Moreover, the centralized routing model of SDN jointly with the possibility of instantiating VNFs on–demand may open the way for an even more efficient operation and resource management of networks .
For instance, an SDN/NFV-enabled network may simplify the Service Function Chain (SFC) deployment and provisioning by making the process easier and cheaper.
In this study, we aim at investigating how to leverage both SDN and NFV in order to exploit their potential benefits. We took steps to address the new opportunities offered in terms of network design, network resilience, and energy savings, and the new problems that arise in this new context, such as the optimal network function placement in the network.
We show that a symbiosis between SDN and NFV can improve network performance and significantly reduce the network's Capital Expenditure (CapEx) and Operational Expenditure (OpEx).

Contents

List of Abbreviations

ATM	Asynchronous Transfer Mode
API	Application Programming Interface
CAPeX	Capital Expenditure
CG	Column Generation
CLI	Command Line Interface
CPU	Central Processing Unit
DPI	Deep Packet Inspection
EON	Elastical Optical Network
FM	Flow Manager
FW	Firewall
ICT	Information and Communication Technology
IDS	Intrusion Detection System
ILP	Integer Linear Program
IP	Internet Protocol
ISP	Internet Service Provider
LB	Load Balancer
LP	Linear Program
MILP	Mixed Integer Linear Program
MPLS	Multiprotocol Label Switching
NAT	Network Address Translator
NFV	Network Function Virtualization
ONF	Open Networking Foundation
OPeX	Operational Expenditure
QoE	Quality of Experience

QoS	Quality of Service
RSA	Routing and Spectrum Assignment
RSMA	Routing, Spectrum, and Modulation Assignment
SDN	Software Defined Networking
SFC	Service Function Chain
SFP	Service Function Path
SLA	Service Level Agreement
VM	Virtual Machine
VNF	Virtual Network Function
WAN	Wide Area Network

Chapter 1
Introduction

The last decade has seen the development of new paradigms to pave the way for a more flexible, open, and economical networking. In this context, *Software Defined Networking* (SDN) and *Network Function Virtualization* (NFV) are two of the more promising technologies for the Next-Generation Network.

SDN aims at simplifying network management by decoupling the control plane from the data plane. Network intelligence is logically centralized in an SDN controller that maintains a global view of the network state. SDN offers, to network operators, better ways to manage and configure their networks. For instance, network devices do not have to be configured individually in a command line interface (CLI) environment and do not have to be changed manually in response to new network conditions [KF13]. Forwarding decisions are rather taken in a single (logical) location, called the *controller*, with a complete knowledge of the network state. Another advantage concerns the ability to introduce new ideas and to easily implement and test new protocols that are hard to deploy in the so-called legacy networks. This offers new opportunities in terms of better usage of network resources, such as the available bandwidth, so as to maximize the operator's profit.

With the NFV paradigm, network functions (e.g., a firewall, a load balancer, and a content filtering) can be implemented in software and executed on generic-purpose servers located in small cloud nodes. Virtual Network Functions (VNFs) can be instantiated and scaled on–demand without the need of installing new equipment. The goal is to shift from specialized hardware appliances to commoditized hardware in order to deal with the major problems of today's enterprise middlebox infrastructure, such as cost, capacity rigidity, management complexity, and failures [She+12].

Both paradigms are penetrating the industry in a big way due to their numerous advantages in terms of cost, flexibility, and energy efficiency. In 2013, Google announced the use of SDN to interconnect its data centers across the planet [Jai+13]. By utilizing this technology, Google was able to achieve several benefits, including an efficient network management, easier and faster

innovation cycles of networks and services, a better network utilization and a reduction of both OPEX (Operating Expenditure) and CAPEX (Capital Expenditure).

Motivated by these results, the combined application of both SDN and NFV is strongly stimulating the interest of Internet Service Providers and Network Operators. For instance, AT&T, the leading US Telecom Operator, has set as a goal the virtualization of 75% of its network by 2020 [Mar17]. Moreover, Orange introduced an SDN offering coverage for 75 countries, designed to help companies instantly provision branch offices with Virtual Network Functions (VNFs) [Wor17], and Huawei in the last years deployed 560 SDN/NFV commercial projects around the world [Hua17].

This list is not exhaustive. Several industrial and academic laboratories are exploring how to maximize SDN and NFV benefits.

The aim of this thesis is to further investigate how to take advantage of the full benefits of these technologies.

In this chapter, we first introduce the context in which this thesis takes place and the research problems that inspired our work. We then highlight our contributions and conclude this chapter with an outline of the remainder of this thesis.

1.1 Software Defined Networks

Computer networks consist of a large number of network devices such as routers, switches, and many types of middleboxes. Router and switches run complex control software that is typically closed and proprietary [McK+08]. They are configured individually using low-level and specific commands which vary across vendors and may even vary across different products of the same vendor. As a result, network management is challenging and failure-prone [Nun+14]. This leads to an increased complexity, slower innovation, and to additional costs of running a network, in terms of both CAPEX and OPEX [FRZ14; Pfa+09].

SDN tries to simplify the network management and make the deployment of new services easier by separating the *control plane* from the *data forwarding plane*. Network elements (i.e., the data plane) become simple forwarding devices, and decisions of how the traffic must be handled are taken in a logically centralized controller responsible of generating the routing tables. See Fig. 1.1 for a comparison with legacy networks.

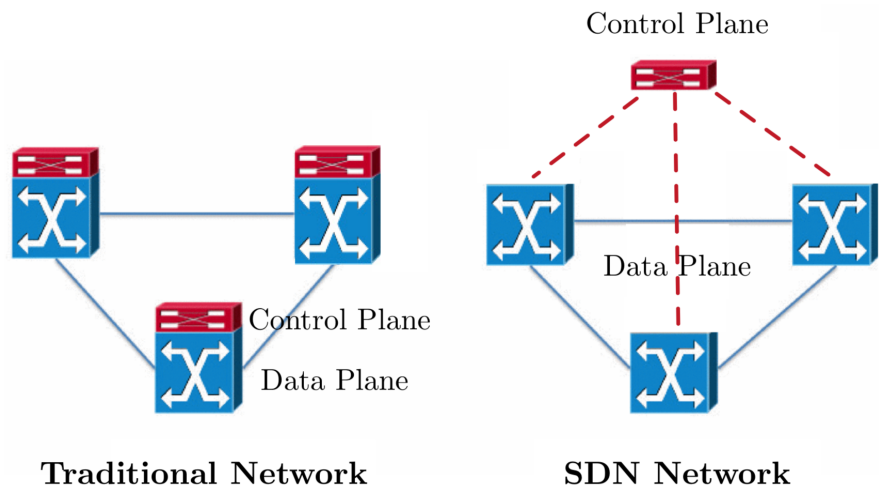


Figure 1.1: SDN decouples the control plane from the data forwarding plane.

The centralization of the control logic leads to several benefits. For instance, network policies become simpler and less failure-prone as they are not changed with low-level device specific configurations, but instead, using high level languages and software components [Kre+15].

Moreover, the controller, with a global view of the network state, can detect changes (e.g., network failures and link loads) and automatically react, thus maintaining high-level policies intact.

Finally, traffic engineering mechanisms can be much more efficiently implemented with respect to a legacy network approach (e.g., IP, ATM, and MPLS). This is thanks to both the possibility to easily retrieve global network information and to program network elements dynamically and proactively without having to handle them individually [Aky+14]. For example, Google has shown their ability to achieve nearly 100% of link utilization using an OpenFlow WAN controller [Jai+13].

1.1.1 SDN Architecture

An SDN architecture can be described as a composition of 3 main layers: *Data Plane*, *Control Plane*, and *Management Plane*, as illustrated in Fig. 1.2.

The *Data plane* is a set of software or handware networking elements such as routers, switches, and middleboxes. Physical devices consist of highly ef-

ficient and programmable packet forwarding devices without any software to take autonomous decisions. Indeed, traffic is forwarded according to decisions that the control plane makes.

The *Control plane* elements are represented by a single logical entity, the controller, which exercises direct control over the data plane using an Application Programming Interface (API), which defines the information exchange between the two planes. The controller provides abstractions, services, and common APIs to developers. Examples of functionalities are statistics about the network state, network topology information, distribution of network configurations, and device discovery [Kre+15]. Thus, a developer does not need anymore to care about low-level details of data distribution of networking elements.

A centralized controller manages all the forwarding devices in the network and implements all control plane logic in a single location. It may be enough to manage a small network. Examples of centralized controllers include Beacon [Eri13], Ryu [Tel12], OpenDayLight [Med+14], and Maestro [CCN10]. However, a single controller represents a single point of failure and it may not be sufficient to manage, in a resilient way, the data plane network elements. Thus, when a node fails, another node should take over the tasks of the failed node. A distributed controller can be either a centralized cluster or a physically distributed set of nodes. Examples of distributed controllers are Onix [Kop+10], ONOS [Ber+14], and DISCO [PBL14].

The *Management plane* can be defined as the network brain. It implements the control functions that will be translated by the controller into commands to be installed in the data plane. It includes applications that allow network operators to develop their high-level policies of network. SDN applications can be grouped in to 5 categories [Kre+15]: traffic engineering (such as load balancing), mobility and wireless, measurement, security, and data center networking.

In order to enable information exchange between the three layers, NorthBound and SouthBound interfaces are defined between controller/applications and controller/network devices (see, e.g., Fig. 1.2).

The *SouthBound API* defines the means of communication between the control and data planes and specifies how the SDN controller instructs the switches regarding how they should behave. OpenFlow [McK+08] is the most popular protocol promoted by the Open Networking Foundation (ONF) [Fou], an organization founded by Deutsche Telekom, Facebook, Google, Microsoft, Verizon, and Yahoo! in 2011, to promote the implementation of

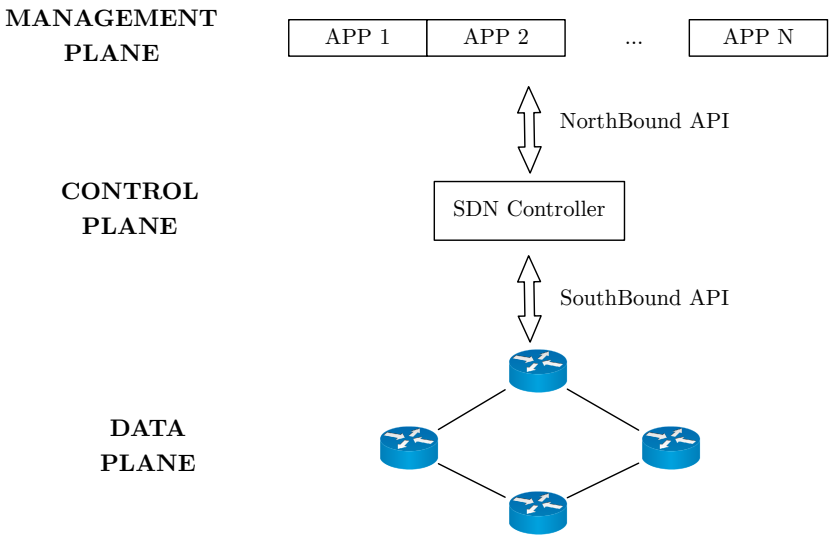


Figure 1.2: SDN Architecture with the interactions between layers

SDN.

OpenFlow was originally deployed in academic campus networks with the goal being to allow researchers to run experiments in production networks. It is a Layer 2 protocol which provides a way to program the flow table of switches and routers over an SSL or TCP/IP connection. Currently, many vendors and network device manufacturers support OpenFlow. The list includes Cisco, Huawei, HP, NEC, IBM, and many others [Aky+14; LKR14].

An OpenFlow switch has one or more flow routing tables consisting of flow entries and defining how the packet belonging to a certain flow will be processed. The controller can control traffic paths in the network by revising, adding, and deleting entries from the flow tables of the switches.
Flow entries consist of:

- *Matching Rules*: set of rules used to match incoming packets. Matching starts at the first flow table and may continue to additional flow tables of the pipeline. Matching can be done either on the packet header fields (e.g., Ethernet source address and IPv4 destination address) or it can also be performed against the ingress port, the metadata field, and other pipeline fields.

- *Counters*: to collect statistics about a particular flow, such as the num-

ber of packets and bytes for each flow, and the time since the last packet matched the flow.

- *Actions*: an action defines how to handle a matching packet. Three basic actions are: forward this flow's packets to a given port, encapsulate and forward this flow's packets to a controller, and drop the packet.

After a packet arrives to an OpenFlow Switch, there are two possibilities. If a match is found, then the switch executes the appropriate set of instructions associated with the specific flow entry. If a match is not found, then the action taken would be defined by the table-miss flow entry, defining the actions to be taken when a match does not occur. Examples of actions that may be taken in this case are packet drop, try to match another flow table, or send the packet to the controller over the OpenFlow channel in order to determine the action to be taken. After a decision has been taken, a new rule may be sent to the device in order to make it able to handle future packets of the same kind.

The *NorthBound API* connects the control layer and the application layer. Its role consists in providing a high-level API between applications/services and the network infrastructure. Conversely, from the SouthBound interface, which has OpenFlow as open source protocol, NorthBound lacks such protocol standards. Defining a common NorthBound API is a critical task as the requirement of each networking application can vary [PST18]. For instance, a security application has different requirements with respect to a routing application. As a consequence, existing controllers such as Onix and OpenDaylight propose and define their own NorthBound APIs.

With a global view of the network state that centralizes network management tasks, the controller can create optimized routes to forward traffic. As a consequence, routing can be optimized and networks can be made more efficient. This opens the way for opportunities for more efficient network resource utilization by adapting the routing configuration over time. On the other hand, SDN brings several challenges in terms of security, availability, scalability, survivability, and costs [LKR14], that must be taken into consideration in order to exploit its full benefits.

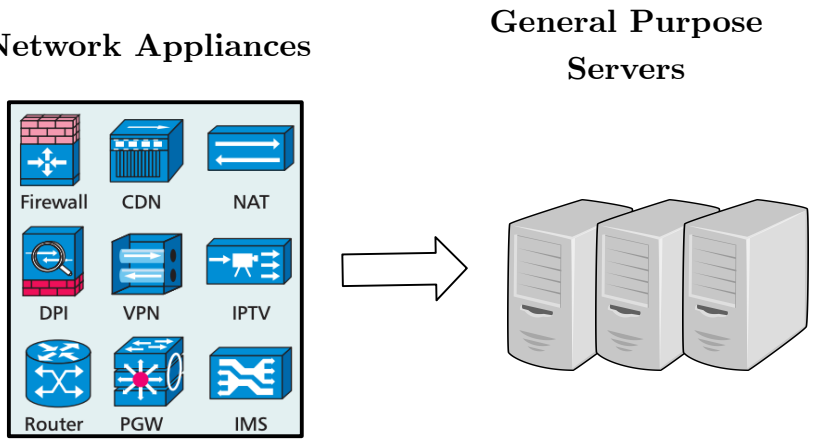


Figure 1.3: NFV moves network functions from dedicated hardware to general purpose servers.

1.2 Network Function Virtualization

Networks include many kinds of equipment. From routers and switches to middleboxes, such as Proxies, Firewalls (FW), Intrusion Detection Systems (IDS), Load Balancers (LB), Network Address Translators (NAT), WAN Optimizer, and Flow Monitor (FM). The number of middleboxes in an enterprise network is comparable with the number of routers and switches [SRA12]. They are necessary as they offer a wide range of benefits, in terms of security, performance, and cost.

Typically, a network function is implemented as a specialized hardware device. This approach has several drawbacks. For instance, middleboxes are expensive, require specialized technicians to be managed, do not allow to add new functionalities, are energy-hungry, and have short lifecycles [HB16]. As shown in [She+12], the cost of middleboxes for large networks may even go beyond a million dollars over a 5 year range.

The high cost of middleboxes does not only depend on the cost of acquisition, but it also depends on the management of a heterogeneous set of devices. Indeed, they require a large management team with expertise to perform tasks such as upgrades, monitoring and diagnostics, and configuration [She+12]. In addition, the personnel need training in order to be able to deal with them.

NFV aims at changing the way operators design, deploy, and manage their network infrastructures. Its goal is to deal with the huge amount of specialized hardware devices deployed in the operators' networks and with the high cost that derives from them.

With the NFV paradigm, network functions are not executed by closed and proprietary appliances anymore, but instead run as software on top of COTS (commercial off-the-shelf) equipment (i.e., industry standard servers, storage, and switches). By decoupling network functions from their underlying hardware appliances, NFV provides a more flexible provisioning of software based network functionalities. Indeed, virtual functions can be instantiated on demand without the installation of new equipment.

For example, a network operator may run an open source software-based firewall on an x86 platform in a virtual machine [Han+15].

NFV opens the way to new possibilities with respect to legacy networks [Mij+16]. For instance, some of the new opportunities introduced by NFV are as follows.

- the possibility of decoupling software from hardware. This allows to separate development and maintenance for both software and hardware.
- a more flexible network function deployment. In such a way, network operators can deploy new network services in a fast way over the same physical platform.
- dynamic resource allocation (scaling up and down) as the network functionality is decoupled from the hardware.

Thus, NFV is a promising approach for network providers and service operators which brings several benefits such as reduction of the CAPEX and OPEX costs, better flexibility of management, resources scaling, and service agility [Ngu+17]. It also brings several challenges in terms of performance, manageability, reliability, and security [Han+15].

With the advent of 5G networks, NFV needs to address some of the design challenges like optimizing resource provisioning of the VNFs for cost and energy efficiency, ensuring performance guarantees of VNF operations, ensuring coexistence of VNFs with non-virtualized network functions, and mobilizing and scaling VNFs between hardware resources [Abd+16].

Figure 1.4: Example of a Service Function Chain [HB16].

1.3 Service Function Chaining

Network virtualization does not require SDN. Similarly, SDN does not imply network virtualization. However, they are complementary to each other and share many properties. A symbiosis between network virtualization and SDN would help in addressing challenges in terms of resource management and intelligent service orchestration.

Software Defined - NFV Network may play an important role in *Service Function Chaining* (SFC). Network flows are often required to be processed by an ordered sequence of network functions [QN15]. For instance, an Intrusion Detection System may need to inspect the packet before compression or encryption are performed. Moreover, different customers can have different requirements in terms of the sequence of network functions to be performed [STV]. See, e.g., Fig. 1.4 for an example.

A Service Function Chain defines the required functions and the corresponding order that must be applied to the packet belonging to a specific data flow. SFC is an enabling technology for the flexible management of a service/application traffic and provides a solution for classifying flows and enforcing a certain policy according to the service requirements [Med+17].

In the current networks, a service chain includes a sequence of hardware dedicated network devices to support a specific application or service. When a new service is required, new hardware devices need to be deployed, installed, and connected. While doing this, computational and network capacity constraints, as well as the policy constraints should be considered [Bha+16]. This process is time-consuming, expensive, and error-prone [LC15].

Thanks to the dynamic function provisioning of NFV and the centralized control of SDN, a Software Defined-NFV Network is able to simplify the service chain deployment and provisioning by making the process easier and cheaper, enabling a flexible and dynamic deployment of network functions as well as a simplified middlebox traffic steering [Qaz+13]. Indeed, the SDN controller can easily provide and reconfigure service function chains in the network, without the need of changing any hardware, and thus reducing the

complexity of resource provisioning. With the centralized control allowed by SDN, the flow can be managed dynamically from end-to-end and the network functions can be installed only along paths for which and when they are necessary.

As a consequence, a Service Function Chain may be added or deleted dynamically, saving time and cost to the operators [Kum+15]. Indeed, in legacy networks, changing the locations of physical middleboxes as the network conditions change would be very costly and impractical.

The SFC problem brings new challenges in the network, one of which is in the context of resource allocation. Optimization models are needed for the SFC distribution and allocation in order to achieve optimal performance of the network, satisfy user demands, accomodate SFCs dynamically, and minimize network cost [Med+17]. Efficient algorithms are needed to determine on which nodes VNFs should be placed. This, with different objectives in mind, such as load balancing, CAPEX and OPEX reduction, energy savings, and ability to recover from failures [HIP15; HB16].

1.4 Research Challenges and Contributions

In this section, we summarize our contributions and put them into context. We first present the contributions made in the context of SDN/NFV-enabled networks, for then summarizing the ones made in the context of Data Center Scheduling and Elastical Optical Networks.

1.4.1 NFV Resource Allocation

Network operators should place VNFs when they will be used most effectively and least expensively [Han+15]. In the context of NFV, the same virtual function can be replicated and executed on several servers. It follows that a fundamental problem arising when dealing with chains of network functions is how to map these functions to nodes (servers) in the network while achieving a specific objective.

Different optimization strategies may be applied to deal with the NFV Resource Allocation Problem: exact solution and heuristics. Exact solutions are mainly based on ILP techniques (i.e., branch and bound and branch and price) and can efficiently deal with small instances [MD14; Lui+15; MKK14; Moh+15; Add+15]. When instance sizes are medium to large or when the

time to find a solution is crucial, such as in dynamic scenarios, an exact solution approach is not suitable. In this case, a heuristic approach able to find a good solution in a reasonable time is preferred. The family of heuristic solutions includes both greedy approaches to find a feasible solution [Rig+15; Moh+15; Add+15] and methods able to provide an approximation to the exact solution [Coh+15; San+17; Tom+18b; Fen+17; CWJ18; Ma+17; SJ19; Pou+19].

In [Tom+18b; Tom+18a], we address the problem of how to optimally place virtual functions within the physical network in order to satisfy the SFC requirements of all the network flows.

Our goal is to place network functions reducing the overall deployment or setup cost. The cost aims at reflecting the cost of having a virtual machine that runs a virtual function, such as license fees, network efficiency, or energy consumption [Oba+16].

Since the formulated problem is NP-Hard, we propose two algorithms that achieve a logarithmic approximation factor. For the special case of tree network topologies with only upstream and downstream flows, we devise an optimal algorithm. We demonstrate the cost effectiveness of our algorithms through extensive simulations.

1.4.2 Survivable SDN/NFV Networks

Network operators are responsible for ensuring that the network provides all the services that users are expecting, with the agreed Quality of Service (QoS). Hence, different factors need to be taken into account during network design and management in order to optimize both the cost and the performance.

However, the underlying network that connects all of these things has remained virtually unchanged. Demands of the exploding number of devices using the network are stretching its limits, and network failures such as (multiple) link or node failures may have a significant impact on the QoS experienced by the customers and lead to SLA (Service Level Agreement) violations. Consequently, resiliency needs to be strongly addressed while designing a network.

In [Tom+18c], we consider the problem of providing, for each demand, a primary and a link-disjoint backup path, under both dedicated and shared path protection schemes. Moreover, the problem also consists in *provisioning VNFs* in order to ensure that the *traversal order* of the network functions by

each path is respected. This adds a challenge to the classical version of the problem. Our goal is to minimize the bandwidth requirements while ensuring that the delays on primary and backup paths stay below SLAs.

We propose a scalable exact method to solve the problem of reliable service function chaining. The method is based on a decomposition model using column generation.

In [Tom+19], we consider a protection technique called *unrestricted flow reconfiguration*, also known as *global rerouting* [PM04]. In each of the possible failure situations, a new set of backup paths are defined, one for each demand. This makes this technique the most bandwidth-efficient protection method. However, this also means that each failure may give rise to a completely different routing for the demands. In a legacy network, it is extremely expensive and impractical to implement this technique due to the huge number of rules to install on the network devices.

We develop a scalable mathematical model that we handle using the Column Generation technique. We show the effectiveness of our methods and demonstrate the feasibility of our approach with an implementation in OpenDaylight.

1.4.3 Energy Aware SDN/NFV Networks

With the large yearly increase of Internet traffic and the growing concern of the public and governments towards greenhouse gas emissions, future networks will have to be more energy efficient. In [08], it is reported that the Information and Communication Technology (ICT) sector is responsible for between 2 and 10% of global energy consumption, of which 51% is attributed to the infrastructure of Telecommunication Networks and data centers.

Moreover, energy bills represent more then the 10% of telecom operators operational expenditure [Bel15]. With the emergence of techniques of NFV, the functions can now be executed by generic hardware instead of dedicated equipment. Coupled with the SDN paradigm, NFV brings a great flexibility to manage network flows. These new paradigms thus bear the opportunity for energy savings in networks.

In [Tom+16; Hui+18a], we explore the potential energy savings of using NFV for Service Function Chains. We consider the problem of reducing network energy consumption while placing network functions using generic hardware along the paths followed by flows. We propose a way of modeling this problem based on Integer Linear Programming (ILP). The ILP can op-

timally solve instances of small sizes. To handle instances of larger sizes, we thus propose and validate a heuristic algorithm and we formulate a Column Generation model.

1.4.4 Other Work

1.4.4.1 Path Protection in Elastic Optical Network

With a *flexible frequency grid*, Elastic Optical Networks (EONs) will support a more efficient usage of the spectrum resources. On the other hand, this efficiency may lead to even more disruptive effects of a failure on the number of involved connections with respect to traditional networks.

In [TJG18], we study the problem of providing path protection to the lightpaths against a single fiber failure event in the optical layer. Our optimization task is to minimize the spectrum requirements for the protection in the network. We develop a scalable exact mathematical model using column generation for both shared and dedicated path protection schemes. The model takes into account practical constraints such as the *modulation format*, *regenerators*, and *shared risk link groups*. We demonstrate the effectiveness of our model through extensive simulation on two real-world topologies of different sizes.

1.4.4.2 Data Center Scheduling with Network Tasks

Network transfers represent up to 50% of the completion time of classical jobs inside a data center [Cho+11; Tho+11]. Thus, network resources must be considered when placing jobs. In [Gir+19b], we propose a new scheduling framework, introducing network tasks that need to be executed on network machines alongside traditional (CPU) tasks. The model takes into account the competition between communications for the network resources, which is not considered in the formerly proposed scheduling models with communication. Network transfers inside a data center can be easily modeled in our framework. As we show, classical algorithms do not efficiently handle a limited amount of network bandwidth. We thus propose new provably efficient algorithms with the goal of minimizing the makespan in this framework. We show their efficiency and the importance of taking into consideration network capacity through extensive simulations on workflows built from Google data center traces.

1.5 Plan of the Thesis

Table 1.1 illustrates the organization and structure of the contributions in the Thesis.

Topic	Chapter
NFV Resource Allocation	3
Survivable SDN/NFV Networks	4, 5
Energy Aware SDN/NFV Networks	6
Data Center Scheduling	7
Path Protection in EONs	8

Table 1.1: Topic organization within the Thesis.

1.6 List of Publications

International Conferences

[1] Frédéric Giroire, Nicolas Huin, **Andrea Tomassilli**, Stéphane Pérennes. *When network matters: Data center scheduling with network tasks*, Proceedings of IEEE International Conference on Computer Communications `(IEEE INFOCOM 2019)`, Paris, France, April 2019.

[2] **Andrea Tomassilli**, Frédéric Giroire, Nicolas Huin, Stéphane Pérennes. *Provably Efficient Algorithms for Placement of Service Function Chains with Ordering Constraints*, Proceedings of IEEE International Conference on Computer Communications `(IEEE INFOCOM 2018)`, Honolulu, HI, USA, April 2018.

[3] **Andrea Tomassilli**, Brigitte Jaumard, Frédéric Giroire. *Path Protection in Optical Flexible Networks with Distance-adaptive Modulation Formats*, Proceedings of International Conference on Optical Network Design and Modeling `(ONDM 2018)`, Dublin, Ireland, May 2018

[4] **Andrea Tomassilli**, Nicolas Huin, Frédéric Giroire, Brigitte Jaumard. *Resource Requirements for Reliable Service Function Chaining*, Proceedings of IEEE International Conference on Communications `(IEEE ICC 2018)`, Kansas City, MO, USA, May 2018.

[5] Nicolas Huin, **Andrea Tomassilli**, Frédéric Giroire, Brigitte Jaumard. *Energy-Efficient Service Function Chain Provisioning*, International Network Optimization Conference (INOC 2017), Lisbon, Portugal, February 2017.

International Journals

[6] Nicolas Huin, **Andrea Tomassilli**, Frédéric Giroire, Brigitte Jaumard. *Energy-Efficient Service Function Chain Provisioning*, IEEE/OSA Journal of Optical Communications and Networking. Volume 10, number 3, pages 114-124, 2018

Posters at International Conferences

[7] **Andrea Tomassilli**, Giuseppe Di Lena, Frédéric Giroire, Issam Tahiri, Damien Saucez, Stéphane Pérennes, Thierry Turletti, Rusland Sadykov, Francois Vanderbeck, Chidung Lac. *Design of Survivable SDN/NFV-enabled Networks with Bandwidth-optimal Failure Recovery*, IFIP Networking (NETWORKING'19), Warsaw, Poland, May 2019.

[8] Adrien Gausseran, **Andrea Tomassilli**, Frédéric Giroire, Joanna Moulierac. *Don't Interrupt Me When You Reconfigure my Service Function Chains*, IFIP Networking (NETWORKING'19), Warsaw, Poland, May 2019.

National Conferences

[9] **Andrea Tomassilli**, Frédéric Giroire, Nicolas Huin, Stéphane Pérennes. *Algorithmes d'approximation pour le placement de chaînes de fonctions de services avec des contraintes d'ordre*, 20ème Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (ALGOTEL 2018), Roscoff, France, May 2018.

Research Reports

[10] Frédéric Giroire, Nicolas Huin, **Andrea Tomassilli**. *The Structured Way of Dealing with Heterogeneous Live Streaming Systems*, HAL-Inria RR-9070

[11] **Andrea Tomassilli**, Nicolas Huin, Frédéric Giroire, Brigitte Jaumard. *Energy-Efficient Service Chains with Network Function Virtualization*, HAL-Inria RR-8979

Submitted

[12] **Andrea Tomassilli**, Giuseppe Di Lena, Frédéric Giroire, Issam Tahiri, Damien Saucez, Stéphane Pérennes, Thierry Turletti, Rusland Sadykov, Francois Vanderbeck, Chidung Lac. *Design of Survivable SDN/NFV-enabled Networks with Bandwidth-optimal Failure Recovery.*

[13] Adrien Gausseran, **Andrea Tomassilli**, Frédéric Giroire, Joanna Moulierac. *Don't Interrupt Me When You Reconfigure my Service Function Chains.*

1.7 Collaboration

During the work of this thesis, I had the possibility to collaborate with other PhD Students and researchers in Sophia Antipolis and elsewhere to conduct research. A list that summarizes these collaborations, ordered in alphabetical order, is as follows.

Name	Affiliation
Zied Ben-Houidi	Nokia Bell-Labs, France
Giuseppe Di Lena	Orange Labs and Inria, France
Adrien Gausseran	Université Cote d'Azur, France
Frédéric Giroire	CNRS, France
Brigitte Jaumard	Concordia University, Canada
Nicolas Huin	Huawei, France
Chidung Lac	Orange Labs, France
Joanna Moulierac	Université Cote d'Azur, France
Stéphane Pérennes	CNRS, France
Rusland Sadykov	Inria, France
Damien Saucez	Inria, France
Issam Tahiri	Inria, France
Thierry Turletti	Inria, France
Francois Vanderbeck	University of Bordeaux 1, France

Also, a list of the research visits carried out during these years.

Hosting Institution	**Time Period**
Concordia University, Montreal, Canada	October-December 2017
Inria Bordeaux - Sud-Ouest, Bordeaux, France	March 2018
Nokia Bell-Labs, Paris, France	October-December 2018

References

[08] *SMART 2020 Enabling the low-carbon economy in the information age, http://www.smart2020.org/_assets/files/02_-Smart2020Report.pdf*. 2008 (cit. on pp. 12, 166).

[Abd+16] Sherif Abdelwahab, Bechir Hamdaoui, Mohsen Guizani, and Taieb Znati. "Network function virtualization in 5G". In: *IEEE Communications Magazine* 54.4 (2016), pp. 84–91 (cit. on p. 8).

[Add+15] Bernardetta Addis, Dallal Belabed, Mathieu Bouet, and Stefano Secci. "Virtual network functions placement and routing optimization". In: *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*. IEEE. 2015 (cit. on pp. 10, 11, 46).

[Aky+14] Ian F Akyildiz, Ahyoung Lee, Pu Wang, Min Luo, and Wu Chou. "A roadmap for traffic engineering in SDN-OpenFlow networks". In: *Computer Networks* 71 (2014), pp. 1–30 (cit. on pp. 3, 5).

[Bel15] Alcetel Lucent Bell Labs. *White Paper: Global What if Analyzer of NeTwork Energy ConsumpTion (GWATT). Bell labs application able to measure the impact of technologies like SDN & NFV on network energy consumption*. Murray Hill, NJ, USA, 2015 (cit. on pp. 12, 166).

[Ber+14] Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O'Connor, Pavlin Radoslavov, William Snow, et al. "ONOS: towards an open, distributed SDN OS". In: *Proceedings of the third workshop on Hot topics in software defined networking*. ACM. 2014, pp. 1–6 (cit. on pp. 4, 80).

[Bha+16] Deval Bhamare, Raj Jain, Mohammed Samaka, and Aiman Erbad. "A survey on service function chaining". In: *Journal of Network and Computer Applications* 75 (2016), pp. 138–155 (cit. on p. 9).

[CCN10] Zheng Cai, Alan L Cox, and TS Ng. *Maestro: A system for scalable openflow control*. Tech. rep. 2010 (cit. on p. 4).

[Cho+11] Mosharaf Chowdhury, Matei Zaharia, Justin Ma, Michael I Jordan, and Ion Stoica. "Managing data transfers in computer clusters with orchestra". In: *ACM SIGCOMM Computer Communication Review*. Vol. 41. 4. 2011 (cit. on pp. 13, 172, 174, 191).

[Coh+15] Rami Cohen, Liane Lewin-Eytan, Joseph Seffi Naor, and Danny Raz. "Near optimal placement of virtual network functions". In: *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE. 2015, pp. 1346–1354 (cit. on pp. 11, 46).

[CWJ18] Yang Chen, Jie Wu, and Bo Ji. "Virtual Network Function Deployment in Tree-structured Networks". In: *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. IEEE. 2018, pp. 132–142 (cit. on p. 11).

[Eri13] David Erickson. "The beacon openflow controller". In: *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM. 2013, pp. 13–18 (cit. on p. 4).

[Fen+17] Hao Feng, Jaime Llorca, Antonia M Tulino, Danny Raz, and Andreas F Molisch. "Approximation algorithms for the NFV service distribution problem". In: *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE. 2017, pp. 1–9 (cit. on p. 11).

[Fou] Open Networking Foundation. URL: https://www.opennetworking/about (cit. on p. 4).

[FRZ14] Nick Feamster, Jennifer Rexford, and Ellen Zegura. "The road to SDN: an intellectual history of programmable networks". In: *ACM SIGCOMM Computer Communication Review* 44.2 (2014), pp. 87–98 (cit. on p. 2).

[Gir+19b] Frédéric Giroire, Nicolas Huin, Andrea Tomassilli, and Stéphane Pérennes. "When network matters: Data center scheduling with network tasks". In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. 2019 (cit. on pp. 13, 171).

[Han+15] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. "Network function virtualization: Challenges and opportunities for innovations". In: *IEEE Communications Magazine* 53.2 (2015), pp. 90–97 (cit. on pp. 8, 10).

[HB16] Juliver Gil Herrera and Juan Felipe Botero. "Resource allocation in NFV: A comprehensive survey". In: *IEEE Transactions on Network and Service Management* 13.3 (2016), pp. 518–532 (cit. on pp. 7, 9, 10).

[HIP15] Enrique Hernandez-Valencia, Steven Izzo, and Beth Polonsky. "How will NFV/SDN transform service provider opex?" In: *IEEE Network* 29.3 (2015), pp. 60–67 (cit. on p. 10).

[Hua17] Huawei. *Huawei Releases SDN/NFV Commercial and Technological Innovations*. 2017. URL: http://www.huawei.com/en/press-events/news/2017/10/Huawei-SDN-NFV-Commercial-Technological-Innovations (cit. on p. 2).

[Hui+18a] N Huin, A Tomassilli, F Giroire, and B Jaumard. "Energy-efficient service function chain provisioning". In: *IEEE/OSA Journal of Optical Communications and Networking* 10.3 (2018), pp. 114–124 (cit. on pp. 12, 45, 136).

[Jai+13] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. "B4: Experience with a globally-deployed software defined WAN". In: *ACM SIGCOMM Computer Communication Review*. Vol. 43. 4. ACM. 2013 (cit. on pp. 1, 3).

[KF13] Hyojoon Kim and Nick Feamster. "Improving network management with software defined networking". In: *IEEE Communications Magazine* 51.2 (2013), pp. 114–119 (cit. on p. 1).

[Kop+10] Teemu Koponen, Martin Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan, Yuichiro Iwata, Hiroaki Inoue, Takayuki Hama, et al. "Onix: A distributed control platform for large-scale production networks." In: *OSDI*. Vol. 10. 2010, pp. 1–6 (cit. on p. 4).

[Kre+15] Diego Kreutz, Fernando MV Ramos, Paulo Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. "Software-defined networking: A comprehensive survey". In: *Proceedings of the IEEE* 103.1 (2015), pp. 14–76 (cit. on pp. 3, 4).

[Kum+15] S Kumar, M Tufail, S Majee, C Captari, and S Homma. "Service function chaining use cases in data centers". In: *IETF SFC WG* (2015) (cit. on p. 10).

[LC15] Yong Li and Min Chen. "Software-defined network function virtualization: A survey". In: *IEEE Access* 3 (2015), pp. 2542–2553 (cit. on p. 9).

[LKR14] Adrian Lara, Anisha Kolasani, and Byrav Ramamurthy. "Network innovation using openflow: A survey". In: *IEEE communications surveys & tutorials* 16.1 (2014), pp. 493–512 (cit. on pp. 5, 6).

[Lui+15] M. C. Luizelli, L. R. Bays, L.S. Buriol, M. P. Barcellos, and L. P. Gaspary. "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions". In: *IFIP/IEEE International Symposium on Integrated Network Management*. 2015 (cit. on pp. 10, 46).

[Ma+17] Wenrui Ma, Oscar Sandoval, Jonathan Beltran, Deng Pan, and Niki Pissinou. "Traffic aware placement of interdependent nfv middleboxes". In: *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE. 2017, pp. 1–9 (cit. on p. 11).

[Mar17] Sue Marek. *Update: AT&T's Stephens: More Than 40% of Network Functions Are Virtualized*. 2017. URL: https://www.sdxcentral.com/articles/news/atts-stephens-47-network-functions-virtualized/2017/07/ (cit. on p. 2).

[McK+08] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. "OpenFlow: enabling innovation in campus networks". In: *ACM SIGCOMM Computer Communication Review* 38.2 (2008), pp. 69–74 (cit. on pp. 2, 4).

[MD14] Hendrik Moens and Filip De Turck. "VNF-P: A model for efficient placement of virtualized network functions". In: *10th International Conference on Network and Service Management (CNSM) and Workshop*. IEEE. 2014, pp. 418–423 (cit. on p. 10).

[Med+14] Jan Medved, Robert Varga, Anton Tkacik, and Ken Gray. "Opendaylight: Towards a model-driven sdn controller architecture". In: *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*. IEEE. 2014, pp. 1–6 (cit. on p. 4).

[Med+17] Ahmed M Medhat, Tarik Taleb, Asma Elmangoush, Giuseppe A Carella, Stefan Covaci, and Thomas Magedanz. "Service function chaining in next generation networks: State of the art and research challenges". In: *IEEE Communications Magazine* 55.2 (2017), pp. 216–223 (cit. on pp. 9, 10).

[Mij+16] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. "Network function virtualization: State-of-the-art and research challenges". In: *IEEE Communications Surveys & Tutorials* 18.1 (2016), pp. 236–262 (cit. on p. 8).

[MKK14] Sevil Mehraghdam, Matthias Keller, and Holger Karl. "Specifying and placing chains of virtual network functions". In: *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. IEEE. 2014, pp. 7–13 (cit. on pp. 10, 45, 46).

[Moh+15] Ali Mohammadkhan, Sheida Ghapani, Guyue Liu, Wei Zhang, KK Ramakrishnan, and Timothy Wood. "Virtual function placement and traffic steering in flexible and dynamic software defined networks". In: *Local and Metropolitan Area Networks (LANMAN), 2015 IEEE International Workshop on*. IEEE. 2015, pp. 1–6 (cit. on pp. 10, 11, 46, 138).

[Ngu+17] Van-Giang Nguyen, Anna Brunstrom, Karl-Johan Grinnemo, and Javid Taheri. "SDN/NFV-based mobile packet core network architectures: A survey". In: *IEEE Communications Surveys & Tutorials* 19.3 (2017), pp. 1567–1602 (cit. on p. 8).

[Nun+14] Bruno Astuto A Nunes, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. "A survey of software-defined networking: Past, present, and future of programmable networks". In: *IEEE Communications Surveys & Tutorials* 16.3 (2014), pp. 1617–1634 (cit. on p. 2).

[Oba+16] Mathis Obadia, Jean-Louis Rougier, Luigi Iannone, Vania Conan, and Mathieu Brouet. "Revisiting NFV orchestration with routing games". In: *Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE Conference on*. IEEE. 2016, pp. 107–113 (cit. on pp. 11, 44).

[PBL14] Kévin Phemius, Mathieu Bouet, and Jérémie Leguay. "Disco: Distributed multi-domain sdn controllers". In: *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE. 2014, pp. 1–4 (cit. on p. 4).

[Pfa+09] Ben Pfaff, Justin Pettit, Keith Amidon, Martin Casado, Teemu Koponen, and Scott Shenker. "Extending networking into the virtualization layer." In: *Hotnets*. 2009 (cit. on p. 2).

[PM04] Michal Pióro and Deep Medhi. *Routing, flow, and capacity design in communication and computer networks*. Elsevier, 2004 (cit. on pp. 12, 80).

[Pou+19] Konstantinos Poularakis, Jaime Llorca, Antonia M Tulino, Ian Taylor, and Leandros Tassiulas. "Joint Service Placement and Request Routing in Multi-cell Mobile Edge Computing Networks". In: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. 2019 (cit. on p. 11).

[PST18] Manish Paliwal, Deepti Shrimankar, and Omprakash Tembhurne. "Controllers in SDN: A review report". In: *IEEE Access* 6 (2018), pp. 36256–36270 (cit. on p. 6).

[Qaz+13] Zafar Ayyub Qazi, Cheng-Chun Tu, Luis Chiang, Rui Miao, Vyas Sekar, and Minlan Yu. "SIMPLE-fying middlebox policy enforcement using SDN". In: *ACM SIGCOMM computer communication review*. Vol. 43. 4. ACM. 2013, pp. 27–38 (cit. on p. 9).

[QN15] Paul Quinn and Tom Nadeau. "Problem statement for service function chaining". In: (2015) (cit. on p. 9).

[Rig+15] R. Riggio, Abbas Bradai, Tinku Rasheed, Julius Schulz-Zander, Slawomir Kuklinski, and Toufik Ahmed. "Virtual network functions orchestration in wireless networks". In: *Intl. Conf. on Network and Service Management (CNSM)*. 2015, pp. 108–116 (cit. on pp. 11, 138).

[San+17] Yu Sang, Bo Ji, Gagan R Gupta, Xiaojiang Du, and Lin Ye. "Provably Efficient Algorithms for Joint Placement and Allocation of Virtual Network Functions". In: *Computer Communications (INFOCOM), 2017 IEEE Conference on*. IEEE. 2017 (cit. on pp. 11, 46, 47).

[She+12] Justine Sherry, Shaddi Hasan, Colin Scott, Arvind Krishnamurthy, Sylvia Ratnasamy, and Vyas Sekar. "Making middleboxes someone else's problem: network processing as a cloud service". In: *ACM SIGCOMM Computer Communication Review* 42.4 (2012), pp. 13–24 (cit. on pp. 1, 7).

[SJ19] Gamal Sallam and Bo Ji. "Joint Placement and Allocation of Virtual Network Functions with Budget and Capacity Constraints". In: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. 2019 (cit. on p. 11).

[SRA12] Justine Sherry, Sylvia Ratnasamy, and Justine Sherry At. "A survey of enterprise middlebox deployments". In: (2012) (cit. on p. 7).

[STV] Marco Savi, Massimo Tornatore, and Giacomo Verticale. "Impact of processing costs on service chain placement in network functions virtualization". In: *IEEE NFV-SDN 2015* (cit. on pp. 9, 49, 119).

[Tel12] Nippon Telegraph. *Telephone Corporation,"Ryu Network Operating System."*. 2012 (cit. on p. 4).

[Tho+11] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song. "Design and evaluation of a real-time url spam filtering service". In: *IEEE Symposium on Security and Privacy (SP)*. 2011, pp. 447–462 (cit. on pp. 13, 172).

[TJG18] A Tomassilli, B Jaumard, and F Giroire. "Path Protection in Optical Flexible Networks with Distance-adaptive Modulation Formats". In: *2018 International Conference on Optical Network Design and Modeling (ONDM)*. 2018 (cit. on pp. 13, 203).

[Tom+16] A Tomassilli, N Huin, F Giroire, and B Jaumard. *Energy-efficient service chains with network function virtualization*. 2016 (cit. on p. 12).

[Tom+18a] Andrea Tomassilli, F Giroire, N Huin, and S Pérennes. "Algorithmes d'approximation pour le placement de chaines de fonctions de services avec des contraintes d'ordre". In: *ALGOTEL 2018-20émes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*. 2018 (cit. on pp. 11, 43).

[Tom+18b] Andrea Tomassilli, Frédéric Giroire, Nicolas Huin, and Stéphane Pérennes. "Provably Efficient Algorithms for Placement of Service Function Chains with Ordering Constraints". In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 2018 (cit. on pp. 11, 43).

[Tom+18c] Andrea Tomassilli, Nicolas Huin, Frederic Giroire, and Brigitte Jaumard. "Resource requirements for reliable service function chaining". In: *2018 IEEE International Conference on Communications (ICC)*. IEEE. 2018, pp. 1–7 (cit. on pp. 11, 107).

[Tom+19] Andrea Tomassilli, Giuseppe Di Lena, Frédéric Giroire, Issam Tahiri, Damien Saucez, Stéphane Perennes, Thierry Turletti, Rusland Sadykov, Francois Vanderbeck, and Chidung Lac. "Poster: Design of Survivable SDN/NFV-enabled Networks with Bandwidth-optimal Failure Recovery". In: *Annex to the IFIP Networking 2019 Proceedings*. 2019 (cit. on p. 12).

[Wor17] Marcel van Wort. *SDN and NFV transforming the network: where do we go from here?* 2017. URL: https://www.orange-business.com/en/blogs/connecting-technology/networks/sdn-and-nfv-transforming-the-network-where-do-we-go-from-here (cit. on p. 2).

Chapter 2
Preliminaries

In this chapter, we introduce the preliminaries and tools that are used throughout the later chapters.

Different techniques are used throughout this thesis. They can be grouped into 2 main categories: *Decomposition techniques* and *Approximation Algorithms*. In what follows, these techniques are presented in more detail, describing their principles and benefits.

2.1 Linear Programming

A Linear Program (LP) is the problem of *minimizing* or *maximizing* a linear objective function subject to linear constraints, where the constraints may be equalities and/or inequalities.

A constraint can be defined as a condition on variables which restricts the values that they can take.

In general, we are given:

- a cost vector $c = (c_1, c_2, \ldots, c_n)^T \in \mathbb{R}^n$
- a vector $b = (b_1, b_2, \ldots, b_m)^T \in \mathbb{R}^m$
- a matrix $A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \cdots & \cdots & \ddots & \cdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} \in \mathbb{R}^{m \times n}$

Given a vector of variables $x = (x_1, x_2, \ldots, x_n)^T$, the aim consists in either minimizing or maximizing a linear cost function

$$c^T x = \sum_{i=1}^{n} c_i \cdot x_i$$

subject to a set of equality or inequality constraints such as $Ax \leq b$, $Ax \geq b$, and $Ax = b$.

In addition, we are given constraints on the values that the variables can take (e.g., non-negativity $x \geq 0$ and non-positivity $x \leq 0$). If all variables can take continuous values, then it is a Linear Problem (LP). Otherwise, if some or all of the variables are restricted to be integers, then we refer to it as a Mixed Integer Linear Problem (MILP) and an Integer Linear Problem (ILP), respectively.

The field of linear programming began in 1947 with the work of Dantzig. He proposed an algorithm for solving linear problems, namely *the simplex method* [Dan48]. The idea consists in starting from a vertex of the feasible region, then moving to an adjacent vertex if an improvement is possible, until the process reaches an optimal point. Indeed, if there is an optimal solution, then this solution is on a vertex of the feasible region

Nowdays, the simplex method is still an excellent and widely used method for solving general linear programs.

Even though the simplex method is fairly efficient in practice, it may require an exponential number of iterations. Indeed, in 1972, Klee and Minty [KM70] showed that, for certain linear programs, the simplex method will examine every vertex of the feasible region and their number can be exponential in the number of variables and constraints.

In 1979, Khachiyan proposed a new approach to linear programming, namely the *ellipsoid method*, proven to be a polynomial-time algorithm. Practical experience, however, was disappointing. In almost all cases, the simplex method was much faster than the ellipsoid method [LY+84].

Another polynomial time algorithm was proposed in 1984 by Karmarkar. In [Kha79], he introduced the *projective method*, which led to many other algorithms known as *interior point methods*. An interior point method algorithm which achieves the best known asymptotic running time is due to Ye [Ye91]. For additional details, the reader is referred to [LY+84; Tod02; Bix12].

Thus, linear programs can be solved efficiently in polynomial time. Unfortunately, this is not true for Integer Linear Programs (ILPs). Indeed, even if there are special cases of ILP problems for which we do have polynomial-time algorithms, Integer Linear Programming is NP-hard *in general*. Thus, no theoretically efficient ILP-solver is possible.

The most common method for solving ILP problems is called *Branch-and-Bound* [LD10]. The main idea consists in partitioning the set of feasible

solutions into smaller subsets of solutions and solving a problem for each subset. The process is repeated recursively by exploring promising areas by keeping track of the upper and lower bounds for the optimal solution. We refer to [LW66] for additional details about this technique.

Another approach to solve ILPs is the Cutting Plane method [Gom+58]. In this case, the strategy consists in solving a set of linear relaxations and iteratively adding constraints to the original problem with the goal of better approximating the convex hull of the feasible region around the optimal solution.

Both methods can be considered powerful tools to solve ILPs and are integrated in most of the ILP solvers, such as Cplex [CPL09], Gurobi [OPT14], SCIP [Ach04], GLPK [Mak15], and COIN-OR [Lou03].

Anyway, the cutting plane algorithms exhibit a slow convergence as the addition of too many cuts can lead to very large LPs that are hard to solve [Jün+09]. Also, the branch and bound method is slow as many useless nodes may be explored [Wol98].

2.1.1 Column Generation

Instead of solving the general problem, sometimes it may be useful to exploit its special structure, try to decompose it, and separately solve smaller and easier-to-solve parts of the problem to achieve the original optimal solution. Column generation has been shown to be useful and applicable to many problems. Examples are vehicle routing [DDS92; SS98], machine scheduling [CP99], graph coloring [MT96], cutting stock [Van+94], and Service Function Chain provisioning [HJG17b] problems.

We use it as a tool which can help to deal with the complexity of the class of routing problems we are going to consider in this Thesis.

Column generation consists in solving a linear program with only a subset of the variables present, and using the dual solution to generate new variables, which may improve the current optimal solution, until no such variables can be found. As in the simplex method, with column generation the aim is to find, at each iteration, one or more promising variables to enter the basis.

The process starts by defining a *restricted master problem* (RMP) with only a subset of the variables $x = (x_1, x_2, \ldots, x_n)$. The RMP may be initialized with artificial variables as well as a feasible solution computed using a heuristic, for example. By solving the master problem, we obtain a primal feasible solution and dual multipliers π, which will be used to find a new

variable x_{n+1} (if any) with negative or positive reduced cost, according to the case of a minimization or a maximization problem, respectively. The problem to be solved when finding a new variable is often referred to as a *pricing problem* (PP).

The solution of the subproblem provides either a certificate of optimality, or a new column that will be added to the master problem and that may potentially improve the value of the current solution x.

Note that solving the subproblem to optimality is only necessary to prove optimality of the general problem. Indeed, it could be enough to stop solving the subproblem as soon as a column with positive (maximization) or negative (minimization) reduced cost is found. The newly generated column is added to our RMP and the process is repeated until no improving column can be generated. In such a case, the optimal solution of the RMP is also optimal for the linear relaxation of the general problem.

However, there are several issues to be dealt with when using the Column generation algorithm [DDS06]. For instance, a problem consists in speeding up the convergence of the column generation algorithm. Indeed, convergence of the basic column generation procedure suffers from dual oscillations especially when the number of constraints is large. To improve the convergence and reduce the fluctuations in the dual variables, it may be useful using stabilization techniques (see e.g., [Pes+18; ADF04]).

Another problem consists in choosing the best column to enter the Master Problem, as for problems with degeneracy, the selected column may not be useful in improving the current optimal solution.

Finally, as the column generation solved the linear relation of the input problem, additional steps are necessary in order to find the optimal integral solution. Existing techniques include the Branch and Cut [HP85], Branch and Bound [LD10], and Branch and Price [Bar+98] algorithms.

The strategy we use to obtain a *good* integral solution is as follows. We first solve the linear relaxation of the general problem by the standard column generation procedure, and then obtain an ϵ-optimal integer solution for the general problem by solving exactly the ILP model associated with the last master problem. We refer the reader to [DL05; CC+83] for more details about this technique.

2.2 Complexity Theory

Let us start with some preliminaries to NP-Completeness. NP-Completeness is based on decision problems, where a decision problem is a problem with a yes/no answer.

The class P (Polynomial Time) contains decision problems which can be solved in polynomial time. This means that, given an instance of the problem, the answer yes/no can be decided in polynomial time (efficiently).

The class NP (Non-Deterministic Polynomial Time) contains decision problems which can be verified in polynomial time. That is, given a solution for an instance of the problem, the correctness of the solution can be checked in polynomial time. Thus, any problem which belongs to P also belongs to NP.

An NP Problem is said to be NP-complete if every NP problem can be reduced to it in polynomial time. That is, Q is NP-complete if, for every NP problem P, we can define a polynomial-time algorithm mapping an instance x of P to an instance y of Q with the following property: x has a *yes* answer if and only if y has a *yes* answer too. For example, determining whether a Boolean formula is satisfied (problem often referred to as *Boolean satisfiability problem*) is NP-complete. NP-complete problems are the hardest problems in NP.

Intuitively, NP-hard problems are at least as hard as any NP-complete problems. These problems are not required to be decision problems, but can be optimization problems in which the goal consists in optimizing some objective function. No polynomial-time algorithms are known for any NP-hard problem.

A large number of optimization problems are difficult to solve optimally and many of them have been shown to be NP-hard [GJ02; CKH95]. For these problems it is not possible to design polynomial-time algorithms able to compute an optimal solution for every possible instance of the problem, unless P=NP, and this is very unlikely to be true.

To deal with these problems, two commonly adopted approaches consist in using either *heuristics* or *approximation algorithms*. A heuristic produces a good solution but without any guarantee on the quality of the solution found. On the other hand, an approximation algorithm aims at finding a solution whose cost is as close as possible to the optimal solution, in polynomial time.

Consider a minimization problem P. An algorithm A is said to be an approximation algorithm with approximation ratio α if and only if, for every

instance of the problem, A gives a solution at most α times the optimal value for this instance in polynomial time. Conversely, if P is a maximization problem, then A must be able to compute, for each instance, a solution which is at least α times the optimal solution.

Given an optimization problem Π, then A is an α-approximate algorithm if and only if, for every instance x of Π, A returns a feasible solution such that

$$OPT \leq A(x) \leq \alpha \cdot OPT(x) \quad \text{(minimization case, } \alpha \geq 1)$$

or

$$\alpha \cdot OPT \leq A(x) \leq OPT(x) \quad \text{(maximization case, } \alpha \leq 1)$$

where $A(x)$ represents the solution returned by A with x as an instance and $OPT(x)$ the corresponding optimal solution.

Many different techniques have been used to develop approximation algorithms such as greedy, randomized, and linear programming based algorithms. [Vaz13] provides a thoughtful summary of the mainly used techniques.

In what follows, we describe two techniques that have been used in this thesis in more detail, namely greedy and linear programming, applying them on a classical problem: *the Vertex Cover Problem*.

The Vertex Cover Problem can be stated as follows.
Input: A graph $G = (V, E)$.
Output: A vertex cover for G, i.e., a subset of vertices $V' \subseteq V$ such that, for each $(u, v) \in E$, either $u \in V'$ or $v \in V'$.
Objective: Minimize the cardinality of the vertex cover, i.e., $|V'|$.

An example for a vertex cover in given in Fig. 2.1.

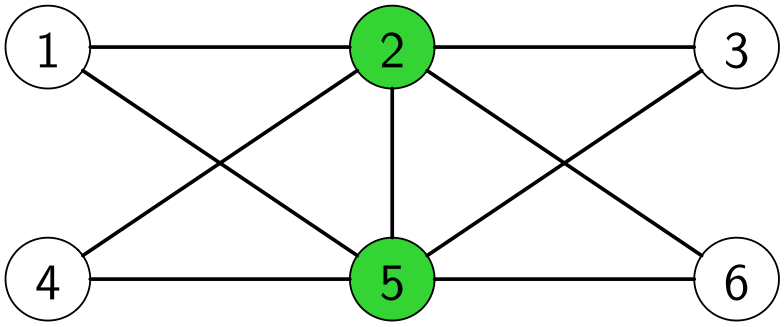


Figure 2.1: Example of Vertex Cover instance. An optimal vertex coloring is given by the nodes highlighted in green.

2.2.1 LP-Rounding

The Vertex Cover problem can be expressed as an ILP by using decision variables x_u for all $u \in V$ to indicate whether $u \in V'$.

V' is a vertex cover if and only if, for each $(u, v) \in E$, the constraint $x_u + x_v \geq 1$ is satisfied.

Thus, the vertex cover problem can be expressed with the following ILP.

Objective

$$\min \sum_{u \in V} x_u$$

Cover conditions

$$x_u + x_v \geq 1, \forall (u, v) \in E$$

Variables Domain

$$x_u \in \{0, 1\}, \forall\ u \in V$$

As discussed below, solving ILPs is NP-hard. Therefore, we make use of an LP to approximate the optimal solution. To this end, we relax the constraint $x_v \in \{0, 1\}$ to $x_v \geq 0, \forall v \in V$.

A linear programming formulation for Vertex Cover is:

Objective

$$\min \sum_{u \in V} x_u$$

Cover conditions

$$x_u + x_v \geq 1, \forall (u, v) \in E$$

Variables Domain

$$x_u \geq 0, \forall\ u \in V$$

We can solve the LP in polynomial time, but the solution may be fractional. In order to obtain an actual vertex cover, given an optimal fractional solution x^* for the LP, we apply the following rounding procedure.

Algorithm 1 FromFractionalToInteger

```
for each x*_u ∈ x* do
    if x*_u ≥ 0.5 then
        x̃_u ← 1
    else
        x̃_u ← 0
    end if
end for
return V' = {u ∈ V | x̃_u = 1}
```

V' is a vertex cover. Indeed, for each edge $x_u^* + x_v^* \geq 1$. Thus, at least one of u and v will be greater than $\frac{1}{2}$ and so, in the Vertex Cover. Also, the cost of V' is at most twice the optimum. This is because

$$OPT \leq |V'| = \sum_{u \in V} \tilde{x}_u \leq \sum_{u \in V} 2 \cdot x_u^* = 2 \cdot OPT(LP) \leq 2 \cdot OPT(ILP),$$

as the value of the fractional optimal solution must be less than or equal to the value of the optimal solution of the integer program.
Thus, this is a polynomial-time 2-approximate algorithm for the vertex cover problem.

2.2.2 Greedy

A greedy heuristic for Vertex Cover may apply the following procedure: pick repeatedly a non-covered edge, put both of its endpoints in the cover and remove all incident edges to the 2 endpoints. A pseudocode is as follows.

Algorithm 2 GreedyVertexCover

```
1: V' ← {}
2: A ← {}
3: while E ≠ ∅ do
4:     pick an edge (u, v) ∈ E
5:     V' ← V' ∪ {u, v}
6:     A ← A ∪ {(u, v)}
7:     remove all edges incident to either u or v from E
8: end while
9: return V'
```

The algorithm returns a valid vertex cover of G as every edge in E has at least one end-point in V'.
A represents the set of edges selected by the algorithm. We have that:

- Every edge in A contributes 2 vertices to V'. Thus, $|V'| = 2 \cdot |A|$.
- Every optimal vertex cover must include at least one endpoint of each edge in A. Thus, $|A| \leq OPT$.

Finally, we have:

$$|V'| = 2 \cdot |A| \leq 2 \cdot OPT$$

Thus, GreedyVertexCover is a 2-factor approximation algorithm for the Vertex Cover problem.

References

[Ach04] Tobias Achterberg. "SCIP-a framework to integrate constraint and mixed integer programming". In: (2004) (cit. on p. 29).

[ADF04] Hatem Ben Amor, Jacques Desrosiers, and Antonio Frangioni. *Stabilization in column generation*. Groupe d'études et de recherche en analyse des décisions, 2004 (cit. on p. 30).

[Bar+98] Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. "Branch-and-price: Column generation for solving huge integer programs". In: *Operations research* 46.3 (1998), pp. 316–329 (cit. on p. 30).

[Bix12] Robert E Bixby. "A brief history of linear and mixed-integer programming computation". In: *Documenta Mathematica* (2012), pp. 107–121 (cit. on p. 28).

[CC+83] Vasek Chvatal, Vaclav Chvatal, et al. *Linear programming*. Macmillan, 1983 (cit. on p. 30).

[CKH95] Pierluigi Crescenzi, Viggo Kann, and M Halldórsson. *A compendium of NP optimization problems*. 1995 (cit. on p. 31).

[CP99] Zhi-Long Chen and Warren B Powell. "Solving parallel machine scheduling problems by column generation". In: *INFORMS Journal on Computing* 11.1 (1999), pp. 78–94 (cit. on p. 29).

[CPL09] IBM ILOG CPLEX. "V12. 1: User's Manual for CPLEX". In: *International Business Machines Corporation* 46.53 (2009), p. 157 (cit. on p. 29).

[Dan48] George B Dantzig. "Programming in a linear structure". In: (1948) (cit. on p. 28).

[DDS06] Guy Desaulniers, Jacques Desrosiers, and Marius M Solomon. *Column generation*. Vol. 5. Springer Science & Business Media, 2006 (cit. on p. 30).

[DDS92] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. "A new optimization algorithm for the vehicle routing problem with time windows". In: *Operations research* 40.2 (1992), pp. 342–354 (cit. on p. 29).

[DL05] Jacques Desrosiers and Marco E Lübbecke. "A primer in column generation". In: *Column generation*. Springer, 2005, pp. 1–32 (cit. on p. 30).

[GJ02] Michael R Garey and David S Johnson. *Computers and intractability*. Vol. 29. wh freeman New York, 2002 (cit. on pp. 31, 212).

[Gom+58] Ralph E Gomory et al. "Outline of an algorithm for integer solutions to linear programs". In: *Bulletin of the American Mathematical society* 64.5 (1958), pp. 275–278 (cit. on p. 29).

[HJG17b] Nicolas Huin, Brigitte Jaumard, and Frédéric Giroire. "Optimization of network service chain provisioning". In: *2017 IEEE International Conference on Communications (ICC)*. IEEE. 2017, pp. 1–7 (cit. on p. 29).

[HP85] Karla Hoffman and Manfred Padberg. "LP-based combinatorial problem solving". In: *Annals of Operations Research* 4.1 (1985), pp. 145–194 (cit. on p. 30).

[Jün+09] Michael Jünger, Thomas M Liebling, Denis Naddef, George L Nemhauser, William R Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A Wolsey. *50 Years of integer programming 1958-2008: From the early years to the state-of-the-art*. Springer Science & Business Media, 2009 (cit. on p. 29).

[Kha79] Leonid G Khachiyan. "A polynomial algorithm in linear programming". In: *Doklady Academii Nauk SSSR*. Vol. 244. 1979, pp. 1093–1096 (cit. on p. 28).

[KM70] Victor Klee and George J Minty. *How good is the simplex algorithm*. Tech. rep. WASHINGTON UNIV SEATTLE DEPT OF MATHEMATICS, 1970 (cit. on p. 28).

[LD10] Ailsa H Land and Alison G Doig. "An automatic method for solving discrete programming problems". In: *50 Years of Integer Programming 1958-2008*. Springer, 2010, pp. 105–132 (cit. on pp. 28, 30).

[Lou03] Robin Lougee-Heimer. "The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community". In: *IBM Journal of Research and Development* 47.1 (2003), pp. 57–66 (cit. on p. 29).

[LW66] Eugene L Lawler and David E Wood. "Branch-and-bound methods: A survey". In: *Operations research* 14.4 (1966), pp. 699–719 (cit. on p. 29).

[LY+84] David G Luenberger, Yinyu Ye, et al. *Linear and nonlinear programming*. Vol. 2. Springer, 1984 (cit. on p. 28).

[Mak15] A Makhorin. *The GNU Linear Programming Kit (GLPK). GNU Software Foundation, 2000*. 2015 (cit. on p. 29).

[MT96] Anuj Mehrotra and Michael A Trick. "A column generation approach for graph coloring". In: *informs Journal on Computing* 8.4 (1996), pp. 344–354 (cit. on p. 29).

[OPT14] GUROBI OPTIMIZATION. "INC. Gurobi optimizer reference manual, 2015". In: *URL: http://www. gurobi. com* (2014) (cit. on p. 29).

[Pes+18] Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. "Automation and combination of linear-programming based stabilization techniques in column generation". In: *INFORMS Journal on Computing* (2018) (cit. on pp. 30, 89).

[SS98] Martin Savelsbergh and Marc Sol. "Drive: Dynamic routing of independent vehicles". In: *Operations Research* 46.4 (1998), pp. 474–490 (cit. on p. 29).

[Tod02] Michael J Todd. "The many facets of linear programming". In: *Mathematical Programming* 91.3 (2002), pp. 417–436 (cit. on p. 28).

[Van+94] Pamela H Vance, Cynthia Barnhart, Ellis L Johnson, and George L Nemhauser. "Solving binary cutting stock problems by column generation and branch-and-bound". In: *Computational optimization and applications* 3.2 (1994), pp. 111–130 (cit. on p. 29).

[Vaz13] Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013 (cit. on pp. 32, 59).

[Wol98] Laurence A Wolsey. *Integer programming*. Wiley, 1998 (cit. on p. 29).

[Ye91] Yinyu Ye. “An O (n 3 L) potential reduction algorithm for linear programming”. In: *Mathematical programming* 50.1-3 (1991), pp. 239–258 (cit. on p. 28).

Part I

NFV Resource Allocation

Chapter 3

Service Function Chains Placement

Contents

The content of this chapter is an extended version of [Tom+18b; Tom+18a].

3.1 Introduction

NFV gives network operators a great freedom to customize their networks and offers a chance to reduce both the capital expenditure and operational costs. Indeed, design choices such as the placement of the functions may have a significant impact on the overall expenditure. It follows that a fundamental problem arising when dealing with chains of network functions is how to map these functions to nodes (servers) in the network while achieving a specific objective. Objectives may differ depending on network's operator goal. Examples of possible objectives are the minimization of the number of used network nodes, the minimization of the network cost, the minimization of the total latency over all paths and the optimization of bandwidth.
We address the problem of how to optimally place virtual functions within the physical network in order to satisfy the SFC requirements of all the network flows. The network is specified by a set of nodes V and links E. The traffic is given as a set of demands $\mathcal{D}$. Each demand is associated with an ordered sequence of network functions that need to be performed to all the packets belonging to the same flow. We assume that the flow between each demand is completely processed at a single node for one function [Cas+10]. Our goal is to place network functions reducing the overall deployment or setup cost. The cost aims at reflecting the cost of having a virtual machine that runs a virtual function, such as license fees, network efficiency, or energy consumption [Oba+16]. In our framework, we consider a general cost function that depends on both the network node and the network function. We refer to this problem as the *SFC Placement Problem*.
In the case in which all the service chains consist of only one function, the problem is known to be equivalent to the Minimum Set Cover problem, as shown in [Cha+05]. This implies that the problem is NP-hard and that an algorithm cannot achieve a better approximation factor than $(1-\varepsilon)\ln|S|$ for any $\varepsilon > 0$, where S is the set of elements to be covered (unless P=NP) [DS]. No positive results are known when the lengths of the service function chains are larger than 1.
We demonstrate that also the generic case, in which the demands have order constraints on the network functions, also corresponds to a set cover instance. We show that the exponential (in $|V|$) number of sets in the instance can be reduced to a polynomial number (in $|V|$ and $|\mathcal{D}|$) by exploiting the structure of the specific type of set cover instances. It allows us to propose two efficient algorithms for the *SFC Placement Problem*. The first one is based on

LP rounding. The second one is a greedy algorithm. For both, we exploit the specific structure of the problem to achieve a short running time, i.e., polynomial also in the length of the largest chain. We show that both the algorithms achieve a solution of cost within a logarithmic factor of the optimal.
We then restrict our attention to tree network topologies. We first show that the problem is NP-hard even in this restricted case. Then, we investigate the scenario in which all the flows are either upstream or downstream flows. We devise an optimal algorithm for this particular case using the dynamic programming technique.
We implement our algorithms and compare their results with the optimal solutions obtained by a linear program. We show that the logarithmic approximation factor is only a worst case upper bound and that we can achieve solutions close to the optimal in most cases.
Although many works on VNF placement have been reported in the literature, no existing work provides algorithms with proven theoretical results for the placement of chains of VNFs with ordering constraints. Most of the solutions are ILP-based, lacking in scalability, or heuristic-based, with no approximation guarantees. To the best of our knowledge, *we are the first to propose a provably efficient algorithm to place chains of virtualized network functions within the network.*
The rest of this chapter is organized as follows. In Section 3.2, we review related works in more detail. In Section 3.3, we present the problem formulation. In Section 3.4, we first show that the *SFC Placement Problem* is equivalent to Set Cover even in the general case. We then present details and analysis of our placement algorithms. In Section 3.5, we propose our optimal algorithm for tree topologies. In Section 3.6, we evaluate our proposed algorithms. Conclusions are drawn in Section 3.7, together with open questions for future work.

3.2 Related Work

There have been some studies on how to place ordered chains of network functions within the network in the literature. Objectives may differ depending on network's operator goal. In particular, the optimization models try to deal with different objectives, such as number of used nodes [MKK14], cost [Bou+15], energy consumption [Hui+18a], bandwidth [HJG18a], and

end-to-end latency [MKK14].
Existing placement algorithms can be roughly classified into two categories: ILP-based and greedy-based. These approaches typically have no provable performance guarantees.
In [Lui+15], the authors address the problem of placing and chaining virtual network functions on physical infrastructures minimizing their number. They propose an Integer Linear Programming and a heuristic procedure. The work in [Kuo+16] studies the joint problem of VNF placement and path selection to better utilize the network. They consider the chaining constraints. Their goal is to maximize the total size of admitted demands. Authors in [MKK14] propose a VNF chaining placement formulated as a Mixed Integer Quadratically Constrained Program. They considered various objectives like minimizing the number of used nodes or the latency of the paths. In [Moh+15] and [Add+15], the authors provide both an ILP and a heuristic with resource utilization being their main focus.
The closest works to ours that study the placement of virtual functions as an optimization problem and provide theoretical results for the performance of the proposed algorithms are [Coh+15] and [San+17].
[Coh+15] addresses the problem of the placement of virtual functions within the physical network. Each demand has a set of required VNFs that need to be executed. The goal of the authors is to minimize the network cost, given by the setup cost of installing a function on a node and the connection cost that depends on the distance between the clients (i.e., the paths) and the nodes from which they get the service. They provide near-optimal approximation algorithms with theoretically proven performance. However, the execution order of the network functions is not considered in their model.
In [San+17], the authors focus their attention on the problem of optimal placement and allocation of VNFs to provide a service to all the flows of the network. The goal is to minimize the total number of network functions. In their model, flow routes are fixed, and one flow may be fractionally processed by the same network function at multiple nodes. However, they study the scenario of one single network function and leave the placement of virtual functions with chaining constraint as an open problem for future research.

$G = (V, E)$	digraph
$\mathcal{D}$	set of demands
$\mathcal{F}$	set of functions
`sfc(d)`	service chain of the demand $d \in \mathcal{D}$
`path(d)`	path associated with the demand $d \in \mathcal{D}$
$l(d)$	length of the path of the demand $d \in \mathcal{D}$
$s(d)$	length of the service chain of the demand $d \in \mathcal{D}$
$c(v, f)$	cost to install the function $f \in \mathcal{F}$ on the node $v \in V$

Table 3.1: Summary of the notations

3.3 System Model and Problem Formulation

We model the network as a digraph $G = (V, E)$. A demand $d \in \mathcal{D}$ is modeled by a couple composed of a path `path(d)` of length $l(d)$ and a service function chain `sfc(d)` of length $s(d)$. A path is a sequence of vertices in V. Similarly to [San+17], we consider the case of an operator which has already routed its demands and which now wants to optimize the placement of network functions. A service function chain is an ordered sequence of functions in $\mathcal{F}$, where $\mathcal{F}$ is the set of network functions. The flow associated with the demand should be processed by the network functions of its chain in the correct order. Each function $f \in \mathcal{F}$ has a setup cost which may depend on the nodes. We note $c(v, f)$ the setup cost of function f in node $v \in V$. In Table 3.1, we summarize the notations used in this chapter.

The problem we consider, referred to as SFC-Placement, is to find a *placement of network functions of minimum setup cost, satisfying the service chain constraints of all demands.* It can be stated as follows.

Input: A digraph $G = (V, E)$, a set of functions $\mathcal{F}$, and a collection $\mathcal{D}$ of demands. Each demand $d \in \mathcal{D}$ is associated with a path `path(d)` $\in V^*$ and to a sequence of functions `sfc(d)` $\in \mathcal{F}^*$. Lastly, a cost $c : V \times F \to c(v, f)$, defining the cost of setting up the function f in node v.

Output: A *function placement* that is a subset $\Pi \subset V \times F$ of function locations, such that, all demands of $\mathcal{D}$ are satisfied. We say that a demand $d \in \mathcal{D}$ associated with a path `path(d)` $= u_1, ..., u_{l(d)}$ and to a chain `sfc(d)` $= r_1, ..., r_{s(d)}$ is *satisfied by* Π, if there exists a sequence of indices $i_1 \leq ... \leq i_{s(d)}$, such that $(v_{i_j}, r_j) \in \Pi$, for $1 \leq j \leq s(d)$.

Objective: minimize $\sum_{(v,f)\in\Pi} c(v, f)$

3.3.1 Preliminaries: Single Function and Uniform Case

Single Function. We use the hitting set formulation of the MINIMUM-WEIGHT SET COVER PROBLEM (MIN-WSC), which is equivalent [ADP80]. The MINIMUM-WEIGHT HITTING SET PROBLEM (MIN-WHS) can be formally defined as follows:
Input: Collection C of subsets of a finite set S.
Output: A hitting set for C, i.e., a subset $S' \subseteq S$ such that S' contains at least one element from each subset in C.
Objective: Minimize the cost of the hitting set, i.e., $\sum_{x \in S'} c_x$.

When all the demands have a service function chain which consists of a single function, the problem can be directly mapped to an instance of MIN-WHS:
- the elements of S are the possible function locations, i.e., the vertices in V. Each element has cost $c(v)$.
- the sets in C correspond to the paths of the demands in $\mathcal{D}$.
For each path `path(d)`, the corresponding set is the set of all the nodes in the path, i.e., $\{u_1, ..., u_{l(d)}\}$.
The placement of minimum cost covering all demands thus corresponds to a minimum cost hitting set.
MIN-WHS is equivalent to Min-Weight Set Cover (MIN-WSC) [ADP80].
MIN-WSC asks, given a set S and a collection C of subsets of S such that $\bigcup_C = S$, to find the subcollection $C' \subseteq C$ whose union is S of minimum cost.
A Hitting Set Instance can be mapped to an equivalent Set Cover Instance. In fact, in the MIN-WHS formulation:
- the elements are the paths of the demands
- the sets correspond to the function location for node v. The set associated with v has cost $c(v)$ and it is the set of all paths containing v.
In the equivalent MIN-WSC formulation, the elements are the paths of the demands and the sets correspond to the function location for node v. The set associated with v has cost $c(v)$ and it is the set of all paths containing v.

The equivalence directly gives us an $H(|\mathcal{D}|)$-approximation using the greedy-algorithm for Set Cover [Chv79] on the positive side. On the negative side, it tells us that the *SFC Placement Problem* is hard to approximate within $\ln |\mathcal{D}|$ [AMS06].

Uniform Service Chains and Installation Costs. There is another case in which the service chain placement problem is immediately equivalent to a

general set cover problem. When all the demands require the same function chain $r_1, r_2, \dots r_k$ and when the installation cost does not depend on the location (i.e. $\forall\ v \in V, cost(v, f) = cost(f)$). In such a case, the function chain $r_1, r_2, \dots r_k$ can be seen as a single function with cost $\sum_{j=1}^{k} c(r_i)$ and the minimum cost placement can be obtained by placing all the functions in a minimum set of locations covering all the paths of the demands.

3.4 Approximation Algorithms for SFC-Placement

After discussing briefly the trivial subcase in which the service chains have length one, we show that the general problem can be modeled as a Set Cover Problem. The instances have an exponential (in $|V|$) number of sets at first. But, we show that this number can be reduced to a polynomial number (in $|V|$ and $|\mathcal{D}|$) by exploiting the specific structure of the problem. We then propose two algorithms with logarithmic (in $|V|$ and $|D|$) approximation factor. Note that the number of sets is still exponential in the maximum size of a service chain, $s_{\max}$, but this number is small in practice [STV] and can thus be considered constant in most scenarios. Finally, we discuss the specific structure of the sets to be covered to improve the efficiency of the algorithms.

3.4.1 Equivalence with Hitting Set

We now show that, even in the general case (with order), *SFC Placement Problem* is equivalent to Min-WHS (and so to Min-WSC). For each demand $d \in \mathcal{D}$, we denote with $l(d)$ and $s(d)$ the length of the associated path and chain respectively. Let `path(d)` $= u_1, u_2, ..., u_{l(d)}$ and assume that d requires the sequence of functions $sfc(P) = r_1, r_2, ..., r_{s(d)}$.

Given a demand d, we build an associated network $H(d)$.

Definition 1 (Associated Networks)**.** The network $H(d)$ associated with a demand d is built as follows:

- $H(d)$ has $s(d)$ layers $L_1, L_2, ..., L_{s(d)}$. Each layer contains $l(d)$ nodes corresponding to the nodes of `path(d)`. We note (u_i, j) the i-th node of layer j.
- There is an arc between the node (u, j) and the node $(v, j+1)$ if $u = v$ or if u precedes v in `path(d)`.

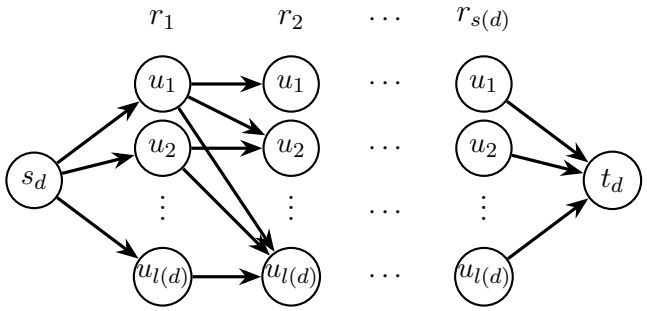


Figure 3.1: The associated network of a demand $d \in \mathcal{D}$ routed on a path $\texttt{path(d)} = u_1, u_2, ..., u_{l(d)}$ that requires a chain $\texttt{sfc(d)} = r_1, r_2, ..., r_{s(d)}$

- $H(d)$ has two other nodes, s_d and t_d. There is an arc between a node s_d and all the nodes of the first layer and an arc between all the nodes of the last layer and t_d.

See Figure 3.1 for an example. We then define the capacities to obtain the *capacitated network* $H(d, \Pi)$ associated with a demand d and a function placement Π:

- All arcs have infinite capacity.
- Each node has a capacity, and the capacity of the node u of layer i is 1 if $(u, r_i) \in \Pi$ and 0 otherwise.

Lemma 1. A demand $d \in \mathcal{D}$ is satisfied by Π if and only if there exists a feasible $st - path$ in the capacitated associated network $H(d, \Pi)$.

Proof. The intuition of the proof is that an $s_d t_d - path$ (or $st - path$ in short) in the layered graph contains exactly one node from each layer and defines where the flow associated with the demand is going to be processed by the required functions in the specified order. Each layer is associated with a function - the j^{th} layer corresponds to the j^{th} function of the function chain $\texttt{sfc(d)} = r_1, r_2, ..., r_{s(d)}$. Since node (u, j) is connected to $(v, j+1)$ if and only if u precedes v in the path $\texttt{path(d)}$, the sequence of functions is performed in the right order when travelling along the path.

Suppose there exists a feasible $st - path$, p. This means that there exists a set of indices $i_1, ..., i_{s(d)}$ such that $p = \{s, u_{i_1}, ..., u_{i_{s(d)}}, t\}$. This implies that the capacity of u_{i_j} is equal to one, i.e., $(u_{i_j}, r_j) \in \Pi$, for all $1 \leq j \leq s(d)$.

Since, in the associated network $H(d, \Pi)$, node (u, j) is connected to $(v, j+1)$ if and only if u precedes v in `path(d)`, we have that $i_1 \leq ... \leq i_{s(d)}$. Therefore all functions of `sfc(d)` are placed in the right order with respect to the nodes of `path(d)`, that is, d is satisfied by Π.

Suppose now that d is satisfied by Π. It means that there exists a set of indices $i_1 \leq ... \leq i_{s(d)}$, such that $(u_{i_j}, r_j) \in \Pi$ for all $1 \leq j \leq s(d)$. Nodes (u_{i_j}, j) of the associated network $H(d, \Pi)$ thus have capacity one. Moreover, there is an arc between (u_{i_j}, j) and $(u_{i_{j+1}}, j+1)$ as u_{i_j} precedes $u_{i_{j+1}}$ in `path(d)`. Hence, $\{s, (u_{i_1}, 1), ..., (u_{i_{s(d)}}, s(d)), t\}$ is a feasible $st-path$ in $H(d, \Pi)$. □

With this notion of associated network, we define the following problem,

Problem 1. HITTING-CUT-PROBLEM $(\mathcal{D}, c)$ is an instance of the Weighted Hitting Set problem where:

- the elements are the function locations (u, f), for all $u \in V$ and $f \in \mathcal{F}$. Its cost is $c(u, f)$.
- the subsets of the universe correspond to all the st-vertex-cuts of the associated networks $H(d)$ for all $d \in \mathcal{D}$.

The problem is thus to find the sub-collection S of elements (functions placement) hitting all the subsets (cuts) of the universe of minimum cost.

Proposition 1. HITTING-CUT-PROBLEM $(\mathcal{D}, c)$ is equivalent to SFC-PLACEMENT $(\mathcal{D}, c)$.

Proof. By construction, a solution S of HITTING-CUT-PROBLEM corresponds to a solution of SFC-PLACEMENT of same cost.

Let us show that S is feasible for HITTING-CUT-PROBLEM if and only if it is a feasible solution of SFC-PLACEMENT. The proof is direct using Menger's theorem for digraphs [Men27]. Consider a digraph and two vertices s and t not connected by an arc. The theorem states that the number of $st-paths$ in a digraph is equal to the minimum st-vertex cut.

Lemma 1 says that all the demands in $\mathcal{D}$ are satisfied by Π if there exists an $st-path$ in all the associated networks $H(d, \Pi)$ for each $d \in \mathcal{D}$. We thus have that all demands are satisfied if all $st-vertex-cuts$ of $H(P, \Pi)$ have a capacity larger or equal to one. Consider C an st-vertex cut. It is hit by S. This implies that in $H(d, \Pi)$, the capacity of the cut is larger than 1. This yields the proposition. □

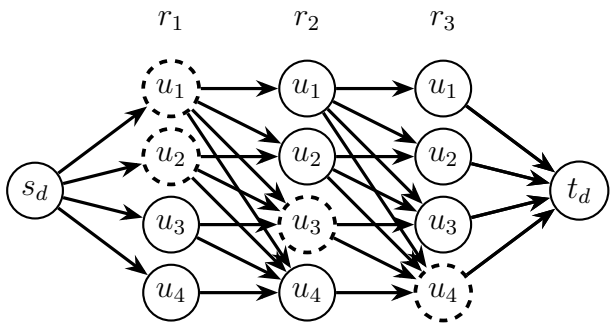


Figure 3.2: Example of a proper cut (dashed nodes in red) for the layered graph relative to a demand d associated with a path of length 4 and a chain of length 3.

Our problem is thus equivalent to a Hitting Set Problem, for which we know approximation algorithms. However, the number of *st*-vertex cuts is exponential in the number of vertices of the digraph. To derive a polynomial algorithm, we need to reduce the size of an instance of Cut-Hitting-Problem. To this end, we use the fact that checking only the *extremal* cuts is enough (An *extremal* cut is a cut that is not strictly included in another cut) and that, in our problem, the extremal cuts of the associated graphs have a specific shape that we call *proper st-cuts*. See Figure 3.2 for an example.

Definition 2. A proper *st*-cut of the associated graph $H(d)$ is a cut of the following form:

$$\{\underbrace{(u_1,1),...,(u_{j_1},1)}_{\text{layer 1}},\underbrace{(u_{j_1+1},2),...,(u_{j_1+j_2},2)}_{\text{layer 2}},...,$$

$$\underbrace{(u_{j_1+j_2+\cdots+j_{s(d)-1}+1},s(d)),...,(u_{l(d)=j_1+j_2+\cdots+j_{s(d)}},s(d))}_{\text{layer } s(d)}\}$$

for $j_1, j_2, ..., j_{s(d)} \geq 0$, such that $\sum_{i=1}^{s(d)} j_i = l(d)$.

Property 1. All the extremal cuts of the associated graphs are proper.

Proof. Let us first remark that, given a cut C in the associated graph, if from the source s it is possible to reach node (u_i, l), then node (u_{i+1}, l) can also be reached from the source. Similarly, if the sink t can be reached from

node (u_i, l), then the sink can also be reached from node (u_{i-1}, l).
Suppose that there exists an extremal cut C such that, for a layer l, C contains nodes u_i, u_{i+2} with $u_{i+1} \notin C$. Since by definition C is a cut, we have 2 possibilities:

- u_{i+1} at layer l cannot be reached by the source. Then, all the nodes u_j with $j \leq i+1$ in the layer $l-1$ cannot be reached, and so u_i is not reachable from the source. We can remove it from C and still get a cut. It follows that C is not an extremal cut (contradiction).

- u_{i+1} at layer l cannot reach the sink. In the same way, u_{i+2} cannot reach the sink. We can then remove u_{i+2} from C and still get a cut. C is not an extremal cut (contradiction).

□

Example 1. Consider a demand $D_{a,c}$ that requires the service function chain $\{f_1, f_2\}$. Suppose that the demand is routed on the path $P = \{a, b, c\}$. There are 4 proper cuts: $\{(a,2),(b,2),(c,2)\}$, $\{(a,1),(b,2),(c,2)\}$, $\{(a,1),(b,1),(c,2)\}$, $\{(a,1),(b,1),(c,1)\}$ corresponding respectively to $j_1 = 0, ..., l(d)$ from which we can derive the following 4 constraints:

$$x(a,f_2) + x(b,f_2) + x(c,f_2) \geq 1$$
$$x(a,f_1) + x(b,f_2) + x(c,f_2) \geq 1$$
$$x(a,f_1) + x(b,f_1) + x(c,f_2) \geq 1$$
$$x(a,f_1) + x(b,f_1) + x(c,f_1) \geq 1$$

We can thus define a new problem of smaller size.

Problem 2. We define the problem HITTING-PROPER-CUT-PROBLEM $(\mathcal{D}, c)$ as the same problem as HITTING-CUT-PROBLEM $(\mathcal{D}, c)$, except that the sets to be hit are only the *proper* st-vertex-cuts of the associated networks $H(d)$ for all $d \in \mathcal{D}$.

Proposition 2. The problem SFC-PLACEMENT $(\mathcal{D}, c)$ is equivalent to a Hitting Set Problem with $\sum_{d \in \mathcal{D}} \binom{l(d)+s(d)-1}{s(d)-1}$ sets as an input. If each demand requires at most $s_{\max}$ network functions and is associated with a path of length smaller than $l_{\max}$, then the size of the instance is at most $O(|\mathcal{D}| \cdot (l_{\max})^{s_{\max}-1})$.

Proof. The proposition follows from previous results. HITTING-PROPER-CUT-PROBLEM $(\mathcal{D}, c)$ is equivalent to HITTING-CUT-PROBLEM $(\mathcal{D}, c)$ as it is enough to consider extremal sets of the collection in a Hitting Set Problem and all extremal cuts are proper cuts. SFC-PLACEMENT $(\mathcal{D}, c)$ thus is equivalent to HITTING-PROPER-CUT-PROBLEM $(\mathcal{D}, c)$.

The size of the ground set of HITTING-PROPER-CUT-PROBLEM $(\mathcal{D}, c)$ is the number of proper cuts of all the associated networks. For each path P, the number of proper cuts of $H(P)$ is simply equal to $\binom{l(d)+s(d)-1}{s(d)-1}$.

Indeed, to obtain the indices $j_1, ..., j_{s(d)}$ defining a proper cut, it is sufficient to select $s(d)-1$ numbers between 0 and $l(d)+s(d)-1$. Without loss of generality, we call them $n_1 \leq ... \leq n_{s(d)-1}$. We then take $j_1 = n_1 - 1$, $j_i = n_i - n_{i-1} - 1$ for $2 \leq i \leq s(d)-1$, and $j_{s(d)} = (l(d) - s(d) - 1) - n_{s(d)-1}$. We have that $\sum_{i=1}^{l(d)} j_i = l(d)$, so the indices define a proper cut. There are $\binom{l(d)+s(d)-1}{s(d)-1}$ ways of choosing $s(d)-1$ elements in a set with $l(d)+s(d)-1$ elements. It yields the number of proper cuts. The size of the ground set is thus $\sum_{d\in\mathcal{D}} l(d)^{s(d)-1}$.

Last, we have $\binom{l(d)+s(d)-1}{s(d)-1} = O(l(d)^{s(d)-1})$. This gives that the number of proper cuts over all paths of the set of demands $\mathcal{D}$ is of the order $O(|\mathcal{D}|(l_{\max})^{s_{\max}-1})$. □

Proposition 2 leads us to two approximation algorithms, a greedy one presented in Section 3.4.2, and one using LP-rounding presented in Section 3.4.3.

3.4.2 Naive and Faster Greedy Algorithms

Naive Greedy Algorithm. The naive greedy algorithm is just the classic greedy algorithm for set cover [Chv79]. It consists of a main loop: while there are proper cuts not hit, it selects the function location with the smallest average cost per newly hit proper cut.

When the demands are routed on paths with length at most $l_{\max}$ and require at most $s_{\max}$ functions, the greedy algorithm achieves an approximation ratio equal to $H(\#\text{Proper Cuts}) = H(|\mathcal{D}|l_{\max}^{s_{\max}-1}) \sim \ln(|\mathcal{D}|) + (s_{\max} - 1)\ln(l_{\max})$ [Chv79], where $H(n)$ is the n-th harmonic number.

Problem for large chains. When the number of functions in the service chains is large, the greedy algorithm could become impractical if it is implemented naively. In fact, the greedy algorithm selects the function location with the smallest average cost per newly hit proper cut. In a naive implementation, it is necessary to generate explicitly all the proper cuts, and this

is not practical since, for a demand d, there may be $O(l_{\max}^{s_{\max}-1})$ of such cuts. Indeed, $l_{\max}$ is in the order of the network diameter. As an example, the network *Cogent* [Kni+11] that we consider in the experiments, has a diameter of 28. For a chain of length 10, we would have $\binom{37}{9}$ proper cuts. However, since the structure of the proper cuts is very specific, we can take advantage of it, providing a much faster greedy algorithm.

Faster greedy algorithm, SFCFastGreedy. The main idea of the faster greedy algorithm is to exploit the *specific structure* of the set cover problem and to avoid generating all proper cuts by showing it is enough to keep track of the *number of not hit proper cuts*. We show here that, by using dynamic programming, this number can be counted in time $O(|\mathcal{D}|l_{\max}^2 s_{\max})$ (instead of $O(|\mathcal{D}|l_{\max}^{s_{\max}})$).

Let us first introduce some notation. For a demand $d = (\texttt{path(d)}, \texttt{sfc(d)})$, a function placement Π can be seen as a matrix A_d with $l(d)$ rows and $s(d)$ columns and for which $A_d[i,j] = 1$ iff $(u_i, r_j) \in \Pi$. We note $A_d[i:j, k:l]$ the submatrix of A_d considering only the rows from i to j and the columns from k to l.

For a demand $d = (\texttt{path(d)}, \texttt{sfc(d)})$ and a function placement Π (or equivalently A_d), we note $N(d)$ the number of proper cuts not hit by A_d. It can be computed using the recursive function $N(r,c)$ defined below. We have $N(d) = N(l(d), s(d))$ with

$$N(r,c) = \mathbb{1}_{i^*(r,c)=0} + \sum_{j_c=0}^{r-i^*(r,c)} N(n-j_c, c-1), \text{if } c \geq 2$$
$$N(r,1) = \mathbb{1}_{i^*(r,c)=0}$$

where $i^*(r,c)$ is defined as follows. We consider the matrix $A_d[1:r, 1:c]$). We consider the ones placed in the last column of the matrix, column c. If there are none, $i^*(r,c) = 0$. Otherwise, $i^*(r,c)$ is the maximum index of such ones, that is, $i^*(r,c) = \max_{0 \leq i \leq l(d)}\{i, \text{ such that } A_d[i,c] = 1\}$.

The explanation of the formula is the following. We carry out a recursion on the columns of $A_d[1:r, 1:c]$. First, if $i^*(r,c) = 0$, the cut $\{(u_1, f_c), ..., (u_r, c)\}$ is not hit. We thus count $\mathbb{1}_{i^*(r,c)=0}$. We then consider all possible values of j_c for the proper cuts (recall that a proper cut is defined by a set of indices $j_1, ..., j_c$). For a not hit proper cut, $j_c \leq l(d) - i^*$. For a possible value of j_c, the number of corresponding not hit proper cuts is equal to the number of not hit proper cuts in the submatrix $A_d[1 : r - j_c, 1 : c-1]$ for a path of length $r - j_c$ and a chain of size $c-1$, that is, $N(r - j_c, c-1, A_d[1 : r - j_c, 1 : c-1])$.

$N(r, c)$ can be computed using dynamic programming, see the function NC of Algorithm 5. We use a table T with r rows and c columns to keep track of the partial results of the computation. Initially, $T(i, 1) = \mathbb{1}_{i^*(r,c)=0}$ for $1 \leq i \leq r$.

3.4.2.1 An example

Consider a demand d with $\texttt{sfc(d)} = f_1, f_2, f_3$ and $\texttt{path(d)} = u_1, u_2, u_3$. Let Π be a potential function placement. $\Pi = \{(u_1, f_1), (u_3, f_2), (u_2, f3)\}$, that is, f_1 is installed on u_1, f_2 on u_3, and f_3 on u_2. All the required functions are placed, but not in the right order. We show that, in this case, some proper cuts of the associated network $H(d, \Pi)$ are not hit. $H(d, \Pi)$ has $\binom{5}{2} = 10$ proper cuts as shown in Proposition 2. We compute here the number of not hit proper cuts from this set without generating them. The matrix A_d associated with the demand and the starting table T in Algorithm 4 would be the following:

$$A_d = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad T = \begin{bmatrix} 0 & - & - \\ 0 & - & - \\ 0 & - & - \end{bmatrix}$$

As $A_d[1, 1] = 1$, we have $i^*(3, 1) = 1 \neq 0$ (the cut $\{(u_1, 1), (u_2, 1), (u_3, 1)\}$ is hit). Similarly, $i^*(2, 1) = 1 \neq 0$ and $i^*(1, 1) = 1 \neq 0$. We thus initialize the first column of T with only zeroes.

In order to compute $T(3, 3)$ the following steps are necessary ($i^*(3, 3) = 1$):
$T(3, 3) = T(1, 2) + T(2, 2) + T(3, 2)$
$T(1, 2) = 1 + T(1, 1) = 1$
$T(2, 2) = T(2, 1) + T(1, 1) + 1 = 1$
$T(3, 2) = T(3, 1) = 0$
Since $T(3, 3) = 2$, we can derive that 2 proper cuts, out of the overall 10 proper cuts of $H(P, \Pi)$, are not hit. Note that this corresponds to the two proper cuts $\{(v_1, f_2), (v_2, f_2)(v_3, f_3)\}$ and $\{(v_1, f_2)(v_2, f_3)(v_3, f_3)\}$. This shows that the order of the functions is not valid. The related constraints are as follows. In red the constraints not satisfied.

$$
\begin{aligned}
x_{u1,f1} + x_{u2,f1} + x_{u3,f1} &\geq 1 \\
x_{u1,f2} + x_{u2,f2} + x_{u3,f3} &\geq 1 \\
x_{u1,f3} + x_{u2,f3} + x_{u3,f3} &\geq 1 \\
x_{u1,f1} + x_{u2,f2} + x_{u3,f2} &\geq 1 \\
x_{u1,f1} + x_{u2,f1} + x_{u3,f2} &\geq 1 \\
x_{u1,f1} + x_{u2,f3} + x_{u3,f3} &\geq 1 \\
x_{u1,f1} + x_{u2,f1} + x_{u3,f3} &\geq 1 \\
x_{u1,f2} + x_{u2,f3} + x_{u3,f3} &\geq 1 \\
x_{u1,f2} + x_{u2,f2} + x_{u3,f3} &\geq 1 \\
x_{u1,f1} + x_{u2,f2} + x_{u3,f3} &\geq 1
\end{aligned}
$$

From this approach, we can derive a faster algorithm with pseudo-code given in Algorithm 4.
At each iteration, the algorithm selects the pair (u, f) of minimum cost, i.e., with the smallest average cost per newly hit proper cut. In order to do this, it makes use of the function NC, calling it for each demand and for each pair $(u, f) \in \mathcal{V} \times \mathcal{F}$. The pair of minimum cost is added to the solution Π. Then, the number of remaining proper cuts to be hit is updated. This process is repeated until all the proper cuts are hit.
Algorithm Complexity. The number of iterations of the main loop of the algorithm is bounded by $|V||\mathcal{F}|$ as we install a function at each iteration. The complexity of the function $\mathrm{NC}(l(d), s(d), \Pi)$ is of the order $O(l(d)^2 s(d))$. It gives us a complexity of $O(l_{\max}^2 s_{\max} |V|^2 |\mathcal{F}|^2 |\mathcal{D}|)$, when a naive algorithm would be of order $O(l_{\max}^{s_{\max}} |V|^2 |\mathcal{F}|^2 |\mathcal{D}|)$, as it would generate all proper cuts.

3.4.3 An LP-Rounding Approach.

First formulation. The HITTING-PROPER-CUT-PROBLEM can be formulated as an ILP. For each node $u \in V$ and for each function $f \in \mathcal{F}$, we define the decision binary variable $x(u, f)$ that indicates whether the function f is

Algorithm 4 Cover Proper Cuts given all the demands

Input: set of demands $\mathcal{D}$
for each $d \in \mathcal{D}$ **do**
    $\textit{not-hit}[d] \leftarrow \binom{l(p)+s(P)-1}{s(P)-1}$
end for
$\Pi = \emptyset$
repeat
    $\textit{min\_cost} \leftarrow +\infty$
    $\textit{best\_sol} \leftarrow \textit{null}$
    $\textit{best\_not-hit} \leftarrow \textit{null}$
    for each $(u, f) \in \mathcal{V} \times \mathcal{F}$ **do**
        $\textit{newly\_hit} \leftarrow 0$
        $\Pi' \leftarrow \Pi \cup \{(u, f)\}$
        for each $d \in \mathcal{D}$ **do**
            $T = l(d) \times s(d)$ matrix of null
            for $1 \leq i \leq l(d)$ **do** ▷ initialization of T
                $T[i, 1] \leftarrow \mathbb{1}_{i^*(i,1)}$
            end for
            $\textit{new-not-hit}[d] \leftarrow \mathrm{NC}(l(d), s(d), \Pi')$
            $\textit{newly\_hit}$ += $\textit{not-hit}[d] - \textit{new-not-hit}[d]$
        end for
        $\textit{cost} \leftarrow \frac{\mathrm{cost}(u,f)}{\mathrm{newly\_hit}}$
        if $\textit{cost} < \textit{min\_cost}$ **then**
            $\textit{min\_cost} \leftarrow \textit{cost}$
            $\textit{best\_sol} \leftarrow (u, f)$
            $\textit{best\_not-hit} \leftarrow \textit{new-not-hit}$
        end if
    end for
    $\Pi = \Pi \cup \{\textit{best\_sol}\}$
    $\textit{not-hit} \leftarrow \textit{best\_not-hit}$
until $\textit{not-hit}[d] = 0$ for each $d \in \mathcal{D}$
Output: placement Π

Algorithm 5 Count proper cuts not hit given demand d and a function placement Π

```
1: function NC (row r, column c, Π)
2:     ▷ Recursive function used to count the number of proper cuts not hit
   given a demand d and a function placement Π
3:     if T[r, c] ≠ null then return T[r, c]
4:     end if
5:     result ← 0
6:     if i*(r, c) = 0 then result ← result + 1
7:     end if
8:     for 0 ≤ j ≤ n − i*(r, c) do
9:         result += NC(n − j, c − 1)
10:    end for
11:    T[r, c] ← result
12:    return result
13: end function
```

installed on node u ($x(u, f) = 1$ in this case). We get as global ILP:

Objective

$$\min \sum_{u \in V} \sum_{f \in \mathcal{F}} c_{u,f} \cdot x_{u,f}$$

Cover conditions

$$\forall d \in \mathcal{D}, \sum_{(u,f) \in C} x_{u,f} \geq 1, \forall\ C \text{ proper cut of } A(d)$$

We consider here the Set-Cover approximation through LP-Rounding. For each $u \in V$ and $f \in F$, we relax the ILP by replacing the constraints $x(u, f) \in \{0, 1\}$ by $0 \leq x(u, f) \leq 1$. The relaxed ILP can be solved in time polynomial in the number of constraints. Let x^* be an optimal solution to the LP relaxation. Each fractional variable $x^*(u, f)$ is rounded to 1 with probability $x^*(u, f)$. The problem is then solved again with the additional constraints given by the rounded variables.
The process is repeated iteratively until all the variables have values in $\{0, 1\}$. With this approach, we find a feasible solution with logarithmic approximation ratio in *expected* polynomial time (in the number of constraints) [Vaz13]. The number of constraints is the number of proper cuts, which is of the order $O(|\mathcal{D}| l_{\max}^{s_{\max}-1})$. It is thus polynomial in $|\mathcal{D}|$, the number of demands, but

exponential in $s_{\max}$, the maximum size of a service chain. As discussed, this number is small in practice, but it may still have a strong impact on the algorithm execution time. We propose a faster algorithm below.

Faster rounding algorithm, SFCFastRounding. The number of constraints of the first formulation is the number of proper cuts which is of the order $O(|\mathcal{D}|l_{\max}^{s_{\max}-1})$, exponential in $s_{\max}$. In fact, similarly as for the greedy algorithm, we can avoid generating explicitly all proper cuts. The idea is to use the formulation of the problem looking for a path in the associated networks $H(d,\Pi)$, as it is equivalent. We derive another ILP formulation. The binary decision variables are now of two kinds:
(i) Location or capacity variables. These variables are the same as in the first formulation: $x(u,f)$ indicates in the first formulation whether the function f is installed on node u. In the second formulation, it corresponds to the shared capacity of the node (u,f) of the associated networks.
(ii) Flow variables. For each demand $d \in \mathcal{D}$, we have a flow variable f_{uv}^d for each edge of the associated network $H(d)$.

The constraints are (i) node capacity constraints and (ii) flow conservation constraints. There are $O(|V| + s_{\max}l_{\max}|D|)$ constraints, a number which is now polynomial in $s_{\max}$.

Objective

$$\min \sum_{u\in V}\sum_{f\in\mathcal{F}} c_{u,f}\cdot x_{u,f}$$

Capacity constraints. $\forall u \in V, \forall f \in \mathcal{F}$,

$$\sum_{d\in\mathcal{D}}\sum_{vu\in E(H(d))} f_{vu} \leq x_{u,f},$$

Flow conservation constraints. $\forall d \in \mathcal{D}$,

$$\sum_{uv\in E(H(d))} f_{uv}^d = \sum_{vu\in E(H(d))} f_{vu}^d,\ \forall\ u \in V(H(d)) \setminus \{s_d, t_d\},$$

$$\sum_{s_dv\in E(H(d))} f_{s_dv}^d = 1$$

$$\sum_{vt_d\in E(H(d))} f_{vt_d}^d = 1$$

A solution of the second formulation corresponds to a solution of the first formulation of same cost (as finding paths in the associated networks is equivalent to covering the cuts, see Lemma 1). Therefore, the rounding can be carried out in the same way and leads to the same approximation factor.

To summarize, along with the fast greedy algorithm, SFCFastGreedy, we obtain a second approximation algorithm for SFC-Placement, called SFCFastRounding, with the same approximation factor $O(\ln(|\mathcal{D}|) + (s_{\max} - 1)\ln(l_{\max}))$. Its expected execution time is $O(M \ln M)$ with $M = |V| + s_{\max} l_{\max} |D|$.

3.5 Tree Topologies

In this section, we restrict our attention to tree logical network topologies. Note that the physical network itself can be of any shape, but the clients are communicating through a tree. The network architecture of today's data centers typically consists of a tree of routing and switching elements [ALV08]. Moreover, tree topologies are widely used, e.g., for Wireless Sensor Networks [Soh+00], and Content Delivery Networks [Yin+09]. We first prove that the *SFC Placement Problem* is NP-hard even on trees through a reduction from the *Vertex Cover Problem*. Then, for the special case in which all the flows are either upstream or downstream flows (i.e., flows are either going towards the tree root or towards the leaves), we devise an optimal algorithm, TreeSFCAlgo.

Theorem 1. The SFC Placement Problem is NP-hard even on a tree and in the case of a single network function.

Proof. Given a graph $G = (V, E)$ and a positive weight function $w : V \to \mathbb{R}^+$, a vertex cover of minimum weight is a subset $C \subseteq V$ such that $\forall\ (u, v) \in E, u \in V$ or $v \in V$ (or both) and $\sum_{u \in C} w(u)$ is minimized.
Let $\mathcal{I} = (G = (V, E), w)$ be an instance of Vertex Cover. We can create an instance $\mathcal{I}'$ of *SFC-tree Placement* by taking the digraph $T = (V \cup \{r\}, \{(u, r), \forall u \in V\} \cup \{(r, u), \forall u \in V\})$. For each $uv \in E$, we create a demand d with `path(d)` $= u, r, v$ and `sfc(d)` $= \{f\}$. The setup cost is $c(u, f) = w(u)$ for all $u \in V$, and $c(r) = \sum_{(u) \in V} w(u) + 1$ for the root of the tree. Note that with this choice of costs, the function f is never placed in the root in an optimal placement, as it is cheaper to place the function in all the other vertices of the tree. We thus have the following equivalence: There is a function placement that satisfies all the paths' requirements in the tree with cost at most $\leq c \iff$ G has a vertex cover of cost $\leq c$. The reduction can be done in polynomial time. It only requires scanning all the edges and creating the set of demands $\mathcal{D}$. Since Vertex Cover is NP-hard

to approximate within a factor of 1.36 [DS05], then the Placement Problem cannot be solved in polynomial time even on trees. □

We now provide a polynomial algorithm that computes the optimal solution in the upstream/downstream case. We present the algorithm in the upstream case, since downstream flows can be replaced by upstream flows, by reversing both the paths and the required function chains.

Main idea. We use dynamic programming in a bottom-up fashion. Given a sub-tree T_v rooted at v, we call a *partial solution*, a feasible function placement restricted to T_v. We also distinguish 3 kinds of paths: *internal-paths,* all vertices of the paths are inside T_v; *external-paths,* no vertex is in T_v; and *crossing-paths,* some but not all vertices are in T_v.

In fact, partial solutions can be encoded in a compact way. To see that, we look at how a partial solution s interacts with a global solution and we claim that:

a) s has to cover all the internal paths.

b) s has no impact on the external paths.

c) On each crossing-path, s provides some (potentially empty) prefix of the required function chain.

d) s induces some cost, namely the cost of the functions located inside T_v.

Since a) and b) are common to all partial solutions, a partial solution is fully characterized by (c) and its cost (d). Now to code c), remark that, instead of remembering for each external path what prefix is provided inside T_v, one may keep track of what suffix must be provided outside T_v. Now, since all paths are upstream, we may simply remember that some suffix s must be provided outside T_s at depth $\geq x$. We call this a *constraint*. The key element here is that, if two paths share the same suffix, one only needs to keep the one that stops at the largest depth.

Overall, this means that a partial solution can be encoded with a set of constraints, and its internal cost. So, our algorithm computes inductively for each subtree, the table containing, for each possible list of constraints, the minimum cost of a partial solution matching these constraints.

TreeSFCAlgo. Let us first introduce some notations and definitions, summarized in Table 3.2. We note $\texttt{depth}(u)$, the depth of a node u in the tree

Table 3.2: New definitions and notations.

$\mathcal{D}_u$	the set of demands s.t. `path(d)` starts at Node u
`src`(d)	source of the path `path(d)`
`dest`(d)	destination of the path `path(d)`
$\mathcal{C}$	set of distinct service chains
`suff`$(\mathcal{C})$	set of suffixes of service chains
`depth`(u)	depth of node $u \in V$ in the tree (source is at depth 1)
`deg`(u)	degree of node $u \in V$ in the tree (# children = `deg`-1)
constraint c	couple (chain suffix,destination d_s)
partial solution s	couple (set of constraints C_s,`cost`(s))
table S_u	set of partial solutions of node u

T (the tree root is at depth 1). Let $\mathcal{C}$ be the set of service chains (a chain per demand). We call `suff`$(\mathcal{C})$ the set of suffixes of elements of $\mathcal{C}$.

A *constraint* is a couple ($s \in$ `suff`$(\mathcal{C}), h \in \mathbb{N}$). A constraint positioned at node u means that the subchain s must be placed in parents of u with depth larger of equal to h. To each demand $d \in \mathcal{D}$ is associated the constraint (`sfc(d)`, `depth`(`dest`(d))), positioned at the node $src(d)$. This means that the chain `sfc(d)` has to be placed below node `dest`(p). Let C_1 and C_2 be two sets of constraints. Two operations may be done to a set of constraints, POP and MERGE.

- MERGE(C_1, C_2). The MERGE operation is a union with "suffixe uniqueness": if $(s, h_1) \in C_1$ and $(s, h_2) \in C_2$, then only $(s, \max(h_1, h_2))$ is present in MERGE(C_1, C_2), as this is the most stringent constraint.

- POP$(F \subseteq \mathcal{F}, C_1)$. We update every suffix σ of C_1 by removing from it the longest prefix made of functions present in F.

A *partial solution* at a node of the tree is encoded by a set of constraints and a cost. A *table* is a set of partial solutions. We note S_u, the table of node u.

- MERGE(S_1, S_2). Two tables S_1 and S_2 may be merged by building a partial solution z for each pair of partial solutions $x \in S_1$ and $y \in S_2$. The constraints of z are the MERGE of the constraints of x and y. The cost of z is just the sum of the costs of x and y. The pseudo-code of all functions and of the algorithm is given in Algorithm 6.

- MERGE$(S_1, ...S_n)$. n tables $S_1, ...S_n$, with $n > 2$, may be merged by doing a two-by-two merge in any order (by associativity of the MERGE function).

We now present our solution TREESFCALGO (pseudo-code in Algorithm 6). It considers the nodes one by one starting from the leaves and builds the tables of each node. S_u, the table of node u is created from intermediate tables $S_{\mathcal{D}_u}$, $S_{\texttt{children}(u)}$, and the tables of its children in the following way. For a node u, it first builds the table $S_{\mathcal{D}_u}$, corresponding to the demands whose paths start in u, using function BUILD_CONSTRAINTS$(\mathcal{D}_u)$. $S_{\mathcal{D}_u}$ contains a single solution of cost 0. The constraints of this solution are built in the following way. For each demand $d \in \mathcal{D}_u$, create the constraint $(\texttt{sfc(d)}, \texttt{depth}(\texttt{dest}(d)))$. Then, it does the MERGE of all the generated constraints. TREESFCALGO then builds $S_{\texttt{children}(u)}$ by merging $S_{\mathcal{D}_u}$ with the tables of its children. Lastly, using function ADD_NODE$(u, \mathcal{D}_u)$, it considers all possible function placements in u and, for each one of them, it considers all solutions in $S_{\texttt{children}(u)}$ and updates the constraints and cost if the placement is compatible with them, using the POP operation. Updating a constraint means removing the functions placed at node u from the suffix representing the chain functions which remain to be placed.

When the table of the root of T is computed, we can select the best solution. The last step of the algorithm is to reconstruct the solution by doing a second pass on the tree, starting from the root.

Time complexity. TREESFCALGO is doing a loop over the vertices of T:

- Complexity of BUILD_CONSTRAINTS. During the whole algorithm, we consider all demands of $\mathcal{D}$. For each demand d, we build the constraint $(\texttt{sfc(d)}, \texttt{depth}(\texttt{dest}(d)))$. Computing the depth of all nodes can be done beforehand with a single pass on the tree of cost $O(|V|)$. We then check the uniqueness of the constraint in $S_{\mathcal{D}_u}$. The test takes constant time using a hash table. We thus obtain an amortized complexity: $O(|V(T)| + |\mathcal{D}|)$.

- Complexity of MERGE: A table is a set of solutions. The size of a solution $x = (C_s, c)$ is given by its set of constraints. The number of constraints is limited by the number of possible suffixes of chains,

$s_{max}|\mathcal{C}|$, where s_{max} is the size of the longest chain. Thus, the memory to store a solution is of order $O(s_{max}|\mathcal{C}|)$. The size of a table is then limited by the number of possible sets of constraints and is thus of order $O(2^{s_{max}|\mathcal{C}|})$.

Merging two tables of size $O(2^{s_{max}|\mathcal{C}|})$, has a complexity of $O(2^{2s_{max}|\mathcal{C}|})$, as we consider each pair of elements. To merge the tables of u's children, as discussed and due to the associativity of the MERGE function, we can do it iteratively, leading to a complexity of $O(n2^{2s_{max}|\mathcal{C}|})$.

- Complexity of ADD_NODE: For each possible placement of functions of u ($2^{|\mathcal{F}|}$ potential placements), we consider each solution in $S(u)$ ($2^{s_{max}|\mathcal{C}|}$ potential solutions). For each solution, we update its set of constraints ($s_{max}|\mathcal{C}|$ potential constraints). The time to update a constraint: $O(s_{max})$, with s_{max} maxsize of a suffix. This leads to a complexity of $O(s_{max}^2|\mathcal{C}|2^{|\mathcal{F}|+s_{max}|\mathcal{C}|})$.

In summary, we get a complexity of $O(|\mathcal{D}|+|V|+|V|^2 2^{2s_{max}|\mathcal{C}|}+|V|s_{max}^2|\mathcal{C}|2^{|\mathcal{F}|+s_{max}|\mathcal{C}|})$. The number of functions $|\mathcal{F}|$ and the number of chains $|\mathcal{C}|$ are usually small in practice. They can thus be considered constant most of the time. The algorithm is thus *quadratic in the number of nodes of the tree and linear in the number of demands.*

Memory usage. The memory used during the algorithm is to keep the tables for all vertices, that is $O(|V|2^{s_{max}|\mathcal{C}|})$. The memory is thus linear in the number of vertices.

3.5.1 Special Case: Cost uniform over nodes

When the cost of setting up a function f is the same for each node of the graph ($\forall v, v' \in V, c(v, f) = c(v', f)$), the algorithm can be improved using the following lemma.

Lemma 2. There exists an optimal solution placing only functions on nodes which are destinations of a path.

Proof. Consider an optimal solution. We create a new solution in the following way. For each function f placed in a non-destination node u, we move it up in the tree towards the root to the first destination node v encountered. The set of demands satisfied by u is a subset of the set of demands satisfied by v. We thus built a feasible solution. The new solution has the same cost as the first one, as the number of placed functions is the same. □

Algorithm 6 TreeSFCAlgo

Input: T with root r
T'=T
while True **do**
    Consider a leaf u of T'
    $S_{\mathcal{D}_u} \leftarrow$ BUILD_CONSTRAINTS($\mathcal{D}_u$)
    $S_{\texttt{children}(u)} \leftarrow$ MERGE($S_{\mathcal{D}_u}, S_{v_1}, ..., S_{v_n}$), with $v_1, ..., v_n$ the children of u in T
    $S_u \leftarrow$ADD_NODE($u, S_{\texttt{children}(u)}$)
    $T' \leftarrow T' \setminus \{u\}$
end while
Output: return solution of S_r with minimum value.

---

function MERGE(S_1,S_2)
    ▷ Merging two tables S_1 and S_2
    S $\leftarrow \{\}$
    for each $x \leftarrow (C_x, cost_x) \in S_1$: **do**
        for each $y \leftarrow (C_y, cost_y) \in S_2$: **do**
            $C_z \leftarrow$MERGE(C_x, C_y)
            $cost_z \leftarrow cost_x + cost_y$
            if $(C_z, c) \notin S$ **then**
                S.append($C_z, cost_z$)
            else
                S.append(C_z, min($cost_z, c$))
            end if
        end for
    end for
    return S
end function

```
1: function BUILD_CONSTRAINTS(D_u ⊆ D)
2:     ▷ Building S_{D_u} from D_u, the set of demands with a path starting in
       u. For each chain s of a demand in D_u, we keep a constraint with s and
       the deepest destination of a path with the chain.
3:     C ← {}
4:     for each d ∈ D_u do
5:         C←MERGE(C,{(sfc(d), depth(dest(d)))})
6:     end for
7:     return S←{(C,0)}
8: end function
```

```
1: function ADD_NODE(u, S_{children(u)})
2:     ▷ Build S_u, the table of solutions of node u
3:     S_u ← {}
4:     for each r ⊆ F do                          ▷ functions installed on node u
5:         for each s ← (C_s, cost(s)) ∈ S_{children(u)} do
6:             if s is compatible with r (meaning if all constraints with level
       d are satisfied by r) then
7:                 C_s ←POP(r, C_s)                          ▷ update constraints of s
8:                 cost(s) ← cost(s) + Σ_{f∈r} c(u, f)              ▷ update cost
9:                 if (C_s, c) ∉ S then                          ▷ ensure uniqueness
10:                    S.append(C_s, cost(s))
11:                else
12:                    S.append(C_s, min(cost(s), c))
13:                end if
14:            end if
15:        end for
16:    end for
17:    return S
18: end function
```

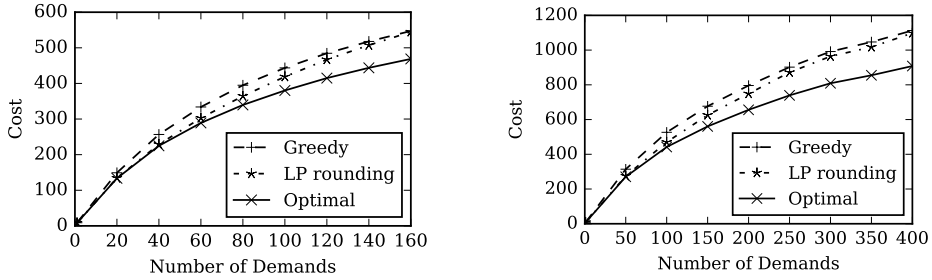
Tree contraction. Following Lemma 2, the first step of the algorithm is to contract the paths and the tree T by removing the non-destination nodes. We obtain a contracted tree, T^*, and a set of contracted paths, $\mathcal{P}^*$. Note now that all paths of $\mathcal{P}^*$ start at a destination node (either its own destination node or the destination of another path). To each destination node u, we associate the set of contracted paths starting in u, $\mathcal{P}_u$.

3.6 Experimental Study

In this section, we evaluate the performances of our proposed algorithms: SFCFastRounding and SFCFastGreedy, referred to as LP rounding and Greedy in the plots, respectively. We study how the total setup cost and the accuracy of our algorithms vary according to four different settings: (i) different path lengths, (ii) increasing number of demands, (iii) increasing length of the service function chains, and (iv) different network topologies. We compare the solutions computed by our algorithms with the optimal ones computed by solving an ILP using IBM ILOG CPLEX.
We show that the logarithmic approximation ratio is just a worst case upper bound and that our algorithms perform well in all the considered scenarios. In fact, the additional cost of the solutions computed by the two algorithms never exceeds 25% of the optimal one. Moreover, the LP rounding algorithm usually obtains a better ratio than the greedy one, but at a cost of a much higher processing time.

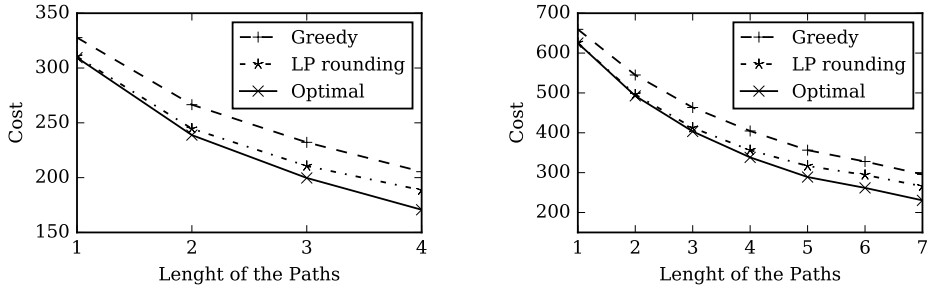
3.6.1 Data sets

We conduct experiments on two real-world topologies of different sizes: `InternetMCI` [Kni+11], (19 nodes and 33 links) and `germany50` [Orl+10b], (50 nodes and 88 links), and on random Erdős-Rényi graphs [Bol98]. We build our instances in the following way. The source and destination nodes of a demand are uniformly chosen at random from the set of vertices. The path of the demand is given by a shortest path between these two nodes and its chain is composed of 2 to 6 functions uniformly chosen at random from a set of 30 functions. Finally, the setup cost of a function on a node is uniformly chosen at random between 1 and 5.



(a) InternetMCI (b) germany50

Figure 3.3: Average setup cost as a function of the number of demands



(a) InternetMCI (b) germany50

Figure 3.4: Average setup cost as a function of the length of the paths

3.6.2 Number of demands

We first compare the performances of the algorithms in the case of an increasing number of demands. Results are given in Figure 3.3. In this scenario, we consider up to 160 demands for `InternetMCI` and up to 400 for `germany50`. As expected, we see that the setup cost increases with the number of demands, as the number of functions to be placed increases. However, the increase is sublinear. The reason is that, the more demands in a network, the higher the opportunity of sharing functions. The optimality ratio is at most 21% for both algorithms. The solution provided by the greedy algorithm differs from 7 to 15% from the optimal one for `InternetMCI` and from 10 to 21% for `germany50`. However, the LP rounding algorithm shows an interesting behavior. When the number of demands is small, it finds optimal solutions. As the number of demands increases, its accuracy deteriorates

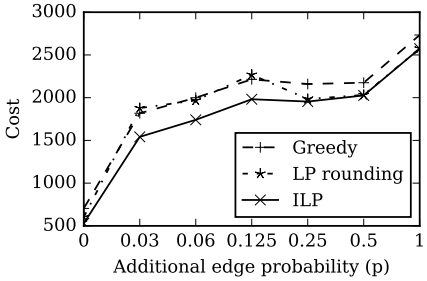


Figure 3.5: Average setup cost in random graphs as a function of the additional edge probability

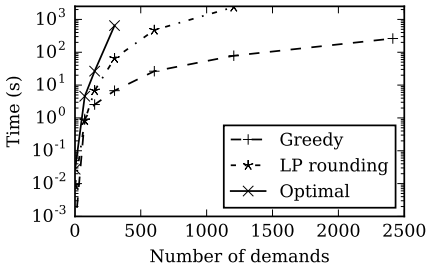


Figure 3.6: Average completion time as a function of the number of demands on Cogent

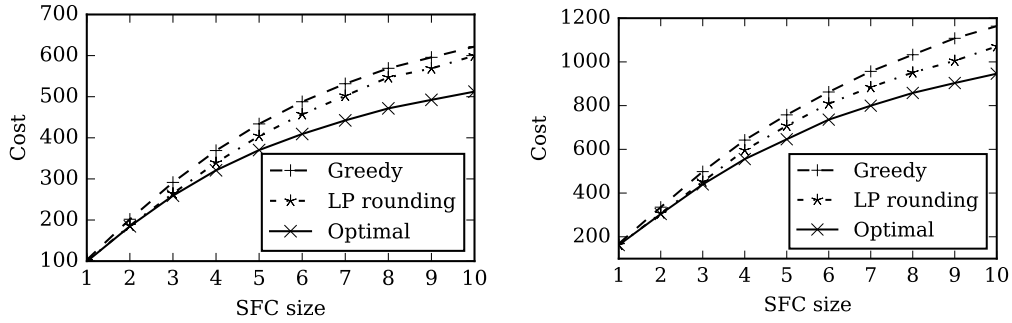
faster than the one of the greedy algorithm. For the highest number of demands, both algorithms exhibit similar performance.

3.6.3 Length of the paths

We now study the impact of the length of the paths. We only consider demands with pairs of nodes at equal distances, from 1 to 4 for `InternetMCI`, and from 1 to 7 for `germany50`. For each length, we consider 40 demands for `InternetMCI` and 75 demands for `germany50`. As we can observe in Figure 3.4, in both networks, the total setup cost strictly decreases when the length of the path increases. In fact, when paths are longer, the demands tend (in average) to share more nodes, reducing the number of required functions to satisfy all the demands and so the cost. For both topologies, the LP rounding algorithm performs better than the greedy one. For the rounding algorithm, the ratio to the optimal solution is smaller than 10% for `InternetMCI` and 15% for german50. The greedy algorithm presents a gap from the optimal solution between 6 ($l(d) = 1$) and 20% ($l(d) = 4$) for `InternetMCI` and between 5 ($l(d) = 1$) and 25% ($l(d) = 7$) for `germany50`.

3.6.4 Length of the chain

We now look at the impact of the service function chains' length on the algorithms' accuracies. In this experiment, we consider service function chains, composed of 1 to 10 functions. In total, we route 75 demands for `InternetMCI` and 150 for `germany50`. As shown in Figure 3.7, an increasing



(a) InternetMCI (b) germany50

Figure 3.7: Average setup cost with respect to the length of the service function chains

length of the chains impacts the performance of the algorithms negatively. In fact, for `InternetMCI`, the ratio between the solution computed by the LP Rounding algorithm and the optimal solution varies from 0.1% with a single function chain to 17% with 10 functions in the chain. For the greedy algorithm, it ranges from 4 to 21% for chains of length 1 and 10, respectively. We observe the same results for the `germany50` topology. The solution of the LP rounding algorithm varies from 0.1 to 13%, while the solution of the greedy algorithm is between 3 and 22%. Nevertheless, these results demonstrate satisfactory performance.

3.6.5 Network topology

We considered random graphs with 100 nodes and different number of edges. The goal is to test the accuracy of the algorithms for topologies with very different shapes, from a tree to a complete graph. We use here a connected variant of random Erdős-Rényi graphs. A graph is built as follows. We start from a random tree. An additional edge is present between two vertices u and v with probability p. For each experiment, we consider 400 random demands. We see, in Figure 3.5, that when the number of edges increases, the cost increases too. This is due to the fact that, when the number of edges increases, the average length of the shortest paths decreases. As discussed above, this reduces the opportunities of sharing. For small values of p, both algorithms have a similar accuracy. However, when $p \geq 0.25$, LP rounding

provides optimal results in these settings.

3.6.6 Processing time

To study the limits in terms of computing time of an LP-based approach, we tested the LP-rounding and greedy algorithms using a larger topology: `Cogent` [Kni+11] with 197 nodes and 245 links. The algorithms have been implemented in C++, and the experiments were conducted on an Intel Xeon E5520 with 24GB of RAM. In Figure 3.6, we show the impact of the number of demands on the execution time. We compare the time necessary to find the optimal solutions with an ILP with the time needed by our algorithms to return a solution. We set a maximum time limit of one hour for each experiment. For just 500 demands, the time to find an exact solution exceeds 1 hour. This implies that, for large instances, an optimal solution cannot be found using the ILP in a reasonable amount of time. Both algorithms can compute solutions for larger instances. However, the greedy algorithm is much faster. Indeed, it takes 78 seconds to find a placement for 1200 demands, while the LP rounding algorithm requires more than 40 minutes.

3.7 Conclusion

NFV is a novel approach for the deployment of network services that opens the way to a more efficient and flexible network management. Hence, placing network functions in a cost effective manner is an essential step toward the full adoption of the NFV paradigm.
In this chapter, we investigated the problem of placing VNFs to satisfy the ordering constraints of the flows with the goal of minimizing the total setup cost. Since the formulated problem is NP-Hard, we proposed two algorithms that achieve a logarithmic approximation factor. To the best of our knowledge, no approximation algorithms have been proposed for the SFC Placement Problem in the literature so far. For the special case of tree network topologies with only upstream and downstream flows, we devised an optimal algorithm. Numerical results are given and validate the cost effectiveness of our algorithms.

References

[Add+15] Bernardetta Addis, Dallal Belabed, Mathieu Bouet, and Stefano Secci. "Virtual network functions placement and routing optimization". In: *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*. IEEE. 2015 (cit. on pp. 10, 11, 46).

[ADP80] Giorgio Ausiello, Alessandro D'Atri, and Marco Protasi. "Structure preserving reductions among convex optimization problems". In: *Journal of Computer and System Sciences* 21.1 (1980), pp. 136–153 (cit. on pp. 48, 85).

[ALV08] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. "A scalable, commodity data center network architecture". In: *ACM SIGCOMM Computer Communication Review*. Vol. 38. 4. ACM. 2008, pp. 63–74 (cit. on p. 61).

[AMS06] N. Alon, D. Moshkovitz, and S. Safra. "Algorithmic Construction of Sets for K-restrictions". In: *ACM Trans. Algorithms* 2.2 (2006). ISSN: 1549-6325. DOI: 10.1145/1150334.1150336 (cit. on p. 48).

[Bol98] Béla Bollobás. "Random graphs". In: *Modern Graph Theory*. Springer, 1998, pp. 215–252 (cit. on p. 68).

[Bou+15] Mathieu Bouet, Jérémie Leguay, Théo Combe, and Vania Conan. "Cost-based placement of vDPI functions in NFV infrastructures". In: *International Journal of Network Management* 25.6 (2015), pp. 490–506 (cit. on p. 45).

[Cas+10] Martin Casado, Teemu Koponen, Rajiv Ramanathan, and Scott Shenker. "Virtualizing the network forwarding plane". In: *Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow*. ACM. 2010, p. 8 (cit. on p. 44).

[Cha+05] Claude Chaudet, Eric Fleury, Isabelle Guérin Lassous, Hervé Rivano, and Marie-Emilie Voge. "Optimal positioning of active and passive monitoring devices". In: *Proceedings of the 2005 ACM conference on Emerging network experiment and technology*. ACM. 2005, pp. 71–82 (cit. on p. 44).

[Chv79] V. Chvatal. "A greedy heuristic for the set-covering problem". In: *Mathematics of operations research* 4.3 (1979), pp. 233–235 (cit. on pp. 48, 54).

[Coh+15] Rami Cohen, Liane Lewin-Eytan, Joseph Seffi Naor, and Danny Raz. "Near optimal placement of virtual network functions". In: *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE. 2015, pp. 1346–1354 (cit. on pp. 11, 46).

[DS] Irit Dinur and David Steurer. "Analytical Approach to Parallel Repetition". In: *Proceedings ACM STOC 2014*. New York, New York. ISBN: 978-1-4503-2710-7 (cit. on pp. 44, 85).

[DS05] Irit Dinur and Samuel Safra. "On the hardness of approximating minimum vertex cover". In: *Annals of mathematics* (2005), pp. 439–485 (cit. on p. 62).

[HJG18a] N. Huin, B. Jaumard, and F. Giroire. "Optimal Network Service Chain Provisioning". In: *IEEE/ACM Transactions on Networking* 26.3 (June 2018), pp. 1320–1333. ISSN: 1063-6692. DOI: 10.1109/TNET.2018.2833815 (cit. on p. 45).

[Hui+18a] N Huin, A Tomassilli, F Giroire, and B Jaumard. "Energy-efficient service function chain provisioning". In: *IEEE/OSA Journal of Optical Communications and Networking* 10.3 (2018), pp. 114–124 (cit. on pp. 12, 45, 136).

[Kni+11] Simon Knight, Hung X Nguyen, Nick Falkner, Rhys Bowden, and Matthew Roughan. "The internet topology zoo". In: *IEEE Journal on Selected Areas in Communications* 29.9 (2011), pp. 1765–1775 (cit. on pp. 55, 68, 72).

[Kuo+16] Tung-Wei Kuo, Bang-Heng Liou, Kate Ching-Ju Lin, and Ming-Jer Tsai. "Deploying chains of virtual network functions: On the relation between link and server usage". In: *Computer Communications (INFOCOM), 2016 IEEE Conference on*. IEEE. 2016, pp. 1–9 (cit. on p. 46).

[Lui+15] M. C. Luizelli, L. R. Bays, L.S. Buriol, M. P. Barcellos, and L. P. Gaspary. "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions". In: *IFIP/IEEE International Symposium on Integrated Network Management*. 2015 (cit. on pp. 10, 46).

[Men27] Karl Menger. "Zur allgemeinen kurventheorie". In: *Fundamenta Mathematicae* 10.1 (1927), pp. 96–115 (cit. on p. 51).

[MKK14] Sevil Mehraghdam, Matthias Keller, and Holger Karl. "Specifying and placing chains of virtual network functions". In: *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. IEEE. 2014, pp. 7–13 (cit. on pp. 10, 45, 46).

[Moh+15] Ali Mohammadkhan, Sheida Ghapani, Guyue Liu, Wei Zhang, KK Ramakrishnan, and Timothy Wood. "Virtual function placement and traffic steering in flexible and dynamic software defined networks". In: *Local and Metropolitan Area Networks (LANMAN), 2015 IEEE International Workshop on*. IEEE. 2015, pp. 1–6 (cit. on pp. 10, 11, 46, 138).

[Oba+16] Mathis Obadia, Jean-Louis Rougier, Luigi Iannone, Vania Conan, and Mathieu Brouet. "Revisiting NFV orchestration with routing games". In: *Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE Conference on*. IEEE. 2016, pp. 107–113 (cit. on pp. 11, 44).

[Orl+10b] Sebastian Orlowski, Roland Wessäly, Michal Pióro, and Artur Tomaszewski. "SNDlib 1.0—Survivable network design library". In: *Networks* 55.3 (2010) (cit. on pp. 68, 95, 119, 213).

[San+17] Yu Sang, Bo Ji, Gagan R Gupta, Xiaojiang Du, and Lin Ye. "Provably Efficient Algorithms for Joint Placement and Allocation of Virtual Network Functions". In: *Computer Communications (INFOCOM), 2017 IEEE Conference on*. IEEE. 2017 (cit. on pp. 11, 46, 47).

[Soh+00] Katayoun Sohrabi, Jay Gao, Vishal Ailawadhi, and Gregory J Pottie. "Protocols for self-organization of a wireless sensor network". In: *IEEE personal communications* 7.5 (2000), pp. 16–27 (cit. on p. 61).

[STV] Marco Savi, Massimo Tornatore, and Giacomo Verticale. "Impact of processing costs on service chain placement in network functions virtualization". In: *IEEE NFV-SDN 2015* (cit. on pp. 9, 49, 119).

[Tom+18a] Andrea Tomassilli, F Giroire, N Huin, and S Pérennes. "Algorithmes d'approximation pour le placement de chaines de fonctions de services avec des contraintes d'ordre". In: *ALGOTEL 2018-20émes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*. 2018 (cit. on pp. 11, 43).

[Tom+18b] Andrea Tomassilli, Frédéric Giroire, Nicolas Huin, and Stéphane Pérennes. "Provably Efficient Algorithms for Placement of Service Function Chains with Ordering Constraints". In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 2018 (cit. on pp. 11, 43).

[Vaz13] Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013 (cit. on pp. 32, 59).

[Yin+09] Hao Yin, Xuening Liu, Tongyu Zhan, Vyas Sekar, Feng Qiu, Chuang Lin, Hui Zhang, and Bo Li. "Design and deployment of a hybrid CDN-P2P system for live video streaming: experiences with LiveSky". In: *Proceedings of the 17th ACM international conference on Multimedia*. ACM. 2009, pp. 25–34 (cit. on p. 61).

Part II

Survivable SDN/NFV Networks

Chapter 4

Bandwidth-optimal Failure Recovery with SDN

Contents

4.1 Introduction

Faults in the IP and optical layer tend to be correlated between them [KKV05]. Indeed, the failure of a component located on a common router, such as a linecard, or in the underlying optical infrastructure, such as a common fiber, may result in the consequential failure of multiple entities at the IP layer. *Shared Risk Link Groups* (SRLGs) allow to easily model this correlation, and also, they can represent different types of failures, such as single and multiple, nodes and links failures. An SRLG can be defined as a set of logical resources that share an underlying physical resource and that are likely to fail simultaneously.

We address in this chapter the problem of designing an SDN programmable network with NFV Infrastructure (NFVI)-enabled servers that provides SRLG-failure survivability.
A way to guarantee the recovery would consist in finding two SRLG-disjoint paths, a primary path and a protection path[1]. If a failure occurs on the primary path of a demand, the traffic is rerouted through the protection path. If, on one hand, this approach is easy to implement and to deploy, on the other hand, it leads to excessive resource requirements in terms of both VNFI Nodes and bandwidth cost and also may impact the number of potential future demands to be routed.

In this chapter, we consider a protection technique called *unrestricted flow reconfiguration*, also known as *global rerouting* [PM04]. In each of the possible failure situations, a new set of backup paths are defined, one for each demand. This makes this technique the most bandwidth-efficient protection method. However, this also means that each failure may give rise to a completely different routing for the demands. In a legacy network, it is extremely expensive and impractical to implement this technique due to the huge number of rules to install on the network devices.
With SDN, thanks to a centralized controller, this technique may be put in practice [VVK14], [Kem+12]. Indeed, SDN offers many potential benefits in terms of fast detection time [Sha+13] and rerouting [Ber+14] and highlight its ability to detect and recover from a failure within the sub 50 ms requirement [Niv+09]. In this chapter, we propose efficient methods to evaluate global rerouting both in theory and practice.

[1]This problem is often referred to as the *SRLG Diverse Routing Problem* and it has been shown to be NP-Complete [Hu03].

Global rerouting is the most bandwidth-efficient protection method. If, on one hand, it represents an opportunity for a better design of bandwidth-aware networks, on the other hand, it is necessary to provide efficient methods for its evaluation and comparison with other protection schemes with the goal being to understand the bandwidth costs savings opportunities.

Thus, our problem is to provide, for each demand, *a primary and a backup path for each SRLG failure scenario*, under the global rerouting protection schema, while ensuring that the required network functions will be performed on the packets in the order specified by its Service Function Chain.

The studied problem is a *dimensioning problem* for an ISP which has to define the needed equipment for its SDN/NFV-enabled network and want to minimize resource usage, while guaranteeing protection against an SRLG failure. Even though, at first glance, the problem may appear easy due to the absence of capacities constraints, we demonstrate that it is not the case. Indeed, we show that *even for a single demand* the problem is NP-Hard and inapproximable within $(1-\epsilon)\ln(|R|)$ for any $\epsilon > 0$ unless P=NP, where $|R|$ denotes the number of SRLG failure scenarios.

Our contributions can be summarized as follows.

- To the best of our knowledge, we are the first to provide two scalable exact methods to solve the problem of global rerouting in SDN/NFV-enabled networks.

- We also propose a fast 2-phase polynomial method. The first one consists in solving the fractional relaxation of the problem. The second phase is building an integral solution from the fractional one. It leads to an optimization problem we named Min Overflow Problem. We show that the problem is NP-complete, but that there exists a $(1+\frac{1}{e}+\varepsilon)$–approximation algorithm to solve it. We use this positive result to propose a fast implementation of the second phase.

- We analyze the impact of the number of VNF-enabled nodes in the network on the bandwidth requirements and on the delays of both primary and backup paths, comparing the proposed protection method against a classical dedicated path protection model.

- We demonstrate the applicability of our proposed protection method on a virtualized SDN testing environment studying metrics such as burden

on the network elements and time to reestablish the flows after a failure. We also discuss the technical choices to be taken into account by the network operator in order to put in practice our proposed technique.

The rest of this chapter is organized as follows. In Section 4.2, we discuss related work. In Section 4.3, we formally define the problem to be studied, as well as notations that will be used in this chapter. Section 4.4 develops the proposed optimization approaches. In Section 4.5, we validate our models by various numerical results on real world and randomly generated data instances, and in Section 4.6, we demonstrate the feasibility of our proposal on Mininet. Finally, we draw our conclusions in Section 4.7.

4.2 Related Work

The problem of providing network protection against failures has been widely investigated in the last decades, see e.g., [Xu+04; RMD05; FV00]. With the advent of SDN/NFV, there are more opportunities for network operators to create, deploy, and manage their networks more efficiently. Indeed, with SDN and its control–data decoupling, routing decisions can be done using a logically centralized approach. This paves the way for a broadening of perspective in terms of fault management [FM17].
They also show in [KCG] a way to create backup paths in such a way that the chances of congestion after a link failure are reduced.
Chu et al. [Chu+] consider a hybrid SDN network and propose a method to design the network in such a way that fast failure recovery from any single link failure is achieved. Their proposal consists in redirecting the traffic on the failed link from the routers to SDN switches through pre-configured IP tunnels. Next hops are pre-configured before the failures take place, and the set of candidate recovery paths for different affected destinations is chosen by the SDN controller in such a way that the maximal link utilization after redirecting the recovery traffic through these paths is minimized. Their optimization task is to minimize the number of SDN switches required.
Suchara et al. [Suc+] proposes a joint architecture for both failure recovery and traffic engineering. Their architecture uses multiple preconfigured paths between each pair of edge routers. In the event of a failure, the failover is made on the least congested path that ensures connectivity. Besides, Sgambelluri et al. [Sga+13] propose a controller–based fault recovery solution that uses OpenFlow's Fast Failover Group Tables to quickly select a preconfigured

backup path in case of link-failure.
Different from previous studies on failure recovery, we present a simple and bandwidth-efficient approach based on multiple backup paths to protect the network against SRLG failures where SDN switches are deployed.
The idea of using a set of pre-configured multiple backup network configurations is not new. For instance, in [Kva+; KCG] the authors propose a pre-configured proactive IP recovery schema that makes use of multiple routing backup configurations as a method for fast recovery. The main idea is to create a small set of backup routing configurations to be used in the case of a single link or node failure. The backup configuration used after a failure is selected according to the failure situation. Since the backup configurations are kept in the routers, it is necessary to reduce their number in order not to require the routers to store a significant amount of state information.
Herein, we take to the extreme the idea of multiple routing configurations by allowing a completely different routing in response to an SRLG failure situation. Different from the above works, our aim is to provide a bandwidth-efficient mechanism to design a reliable network. Besides guaranteeing the recovery, our proposed approach also takes into consideration the SFC requirement of the flows and allows to effectively study what is the right number of NFVI-enabled nodes, in terms of costs, and acceptable QoS levels.

4.3 Problem Statement and Notations

We model the network as an undirected graph $G = (V, E)$, where V represents the set of nodes and E the set of links. Each link represents two unidirectional links in opposite directions. We are given a set of SRLG events $\mathcal{R}$ that can incur link failures. Each $r \in \mathcal{R}$ consists of a set of links that share a common physical resource. We denote by $\mathcal{D}$ the set of demands. As we are solving a dimensioning problem, we assume prior full knowledge of traffic demands, i.e., traffic matrices are known beforehand. A demand $d \in \mathcal{D}$ is modeled by a quadruple (s_d, t_d, bw_d, C_d) with s_d the source, t_d the destination, C_d the ordered sequence of network functions that need to be performed to all the packets belonging to the flow of the demand, and bw_d the required units of bandwidth. We denote by $\ell(d)$ the length of the SFC for a demand d.
Network functions need to be executed on the so called NFVI nodes equipped with Commercial Off The Shelf (COTS) hardware. Not all the nodes are

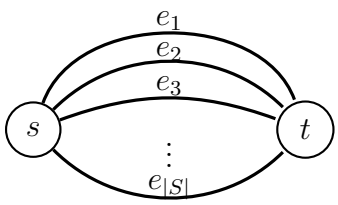


Figure 4.1: The multigraph resulting from the reduction

enabled to run virtual functions. We denote by $V^{\text{VNF}} \subseteq V$ the set of VNF-enabled nodes.
Moreover, we assume that an NFVI-enabled node can only run a subset of the network functions, as there may be constraints on their location in the network.
Given the network topology and the traffic rate of the demands to be supported, the purpose of the design problem is to precompute a set of paths to guarantee the recovery of all the demands in the event of an SRLG failure, while satisfying their SFC requirements. The considered optimization task is to *minimize the bandwidth cost in the network*.
For each demand $d \in \mathcal{D}$ we have to find a primary path and a protection one for each SRLG failure situation $r \in \mathcal{R}$, such that the total amount of bandwidth needed to guarantee the recovery in all the failure situations is minimized.

4.4 Optimization Approaches

We begin the section by proving hardness and inapproximability results for the Global Rerouting problem. Then, we introduce a layered model that is the basis of our proposed methods. The first optimization model that we present is a compact ILP formulation for the global rerouting schema. In order to overcome the scalability issues related to an ILP-based approach, we propose a scalable decomposition model which relies on the Column Generation technique.

Proposition 3. The Global Rerouting problem is NP-hard even for a single demand, and cannot be approximated within
$(1 - \epsilon) \ln(|R|)$ for any $\epsilon > 0$ unless P=NP, where $|R|$ denotes the number of failing scenarios.

Proof. We use a reduction from the Hitting Set Problem, which is defined as follows. We are given a collection C of subsets of a finite set S and the problem consists in finding a hitting set for C, i.e., a subset $S' \subseteq S$ such that S' contains at least one element from each subset in C of minimum cardinality. Given an instance $\mathcal{I} = (S, C)$ of Hitting Set, we can build an instance $\mathcal{I}' = (G, \mathcal{D}, \mathcal{R})$ of Global Rerouting in the following way. $G = (V, E)$ is a multigraph with $V = \{s, t\}$ and $E = \{e_i,\ i = 1, ..., |S|\}$. All the edges have s and t as endpoints.See Fig 4.1 for an example. For each $C' \subseteq C$, we add a failing scenario $r_{C'} = E \setminus C'$ to R, corresponding to edges that cannot be used in the failure situation r. Finally, we add to $\mathcal{D}$, a demand d with s and t as source and destination respectively, and with charge equal to 1. The goal now consists in finding a path for each of the failure scenarios $r \in \mathcal{R}$ minimizing the needed capacity to deploy. The total capacity needed to satisfy d in each of the failure situations is $\leq c \iff$ there exists a hitting set of cardinality $\leq c$. The proposition follows immediately from the fact that Hitting Set is NP-Hard [ADP80] and cannot be approximated within a factor of $\ln |S|$ [DS], unless P=NP. □

4.4.1 A layered network model

The traffic associated to each demand must be processed by an ordered sequence of network functions. Similarly to [HJG18b], we use a layered graph to model this constraint.

Let $G = (V, E)$ be a graph. We associate to each demand $d \in \mathcal{D}$ a layered graph $G^L(d) = (V', E')$. $G^L(d)$ is defined as follows. For each $u \in V$, V' contains the vertices $(u, 0), (u, 1), ..., (u, \ell(d))$. An edge $((u, i), (v, j))$ belongs to E' if and only if (1) $(u, v) \in E$ and $i = j$, or (2) u is a VNFI-enabled node, $u = v$, $j = i + 1$, and the j^{th} function of C_d is installed on u.

Given a demand d, let s_d and t_d be the source and the destination node, respectively. A path starting at vertex $(s_d, 0)$ and finishing at vertex $(t_d, \ell(d))$ of $G^L(d)$ defines (a) which edges of G are used to route the flow associated to the demand; and (b) on which VNFI-enabled nodes the traffic is processed by each of the requested network functions. We refer to a path in $G^L(d) = (V', E')$ as a Service Function Path (SFP). See Figure 4.2 for an example. For the rest of this chapter, the term *path* will refer to a *Service Path* in the layered network.

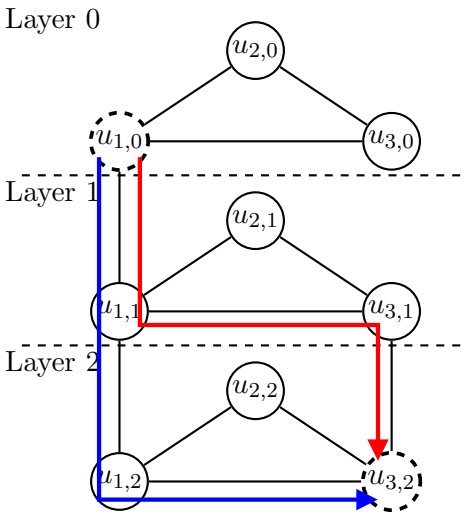


Figure 4.2: The layered network $G^L(d)$ associated with a demand d such that $s_d = u_1$, $t_d = u_3$, and $C_d = f_1, f_2$, with $G = (V, E)$ being a triangle network. We assume f_1 installed on Node u_1 and f_2 installed on Nodes u_1 and u_3. Source and destination nodes of $G^L(d)$ are $u_{1,0}$ and $u_{3,2}$, respectively. They are drawn with dashed lines. Two possible Service Paths that satisfy d are drawn in red and blue.

4.4.2 Compact ILP Formulation

A straightforward way to model our problem consists in using an ILP. The goal of the ILP is to find for each demand $d \in \mathcal{D}$ a Service path on the layered graph $G^L(d)$ for each SRLG event such that the total bandwidth required in the network is minimized. In order to take into account the no failure scenario, we add an SRLG associated with an empty set of links to $\mathcal{R}$. Thus, the SRLGs set is extended to $\mathcal{R} \cup \emptyset$.
Variables:
• $\varphi^{d,r}_{(ui,vj)} \in \{0, 1\}$, with $\varphi^{d,r}_{(ui,vj)} = 1$ if demand d uses link $((u, i), (v, j))$ of $G^L(d)$ in the SRLG failure event r.
• $x^r_{uv} \geq 0$ is the amount of bandwidth allocated on link (u, v) of G in the SRLG failure event r.
Objective: minimization of the bandwidth needed in the network in order to guarantee the recovery (i.e., considering for each link the maximum band-

width used between all the SRLG failure events).

$$\min \sum_{(u,v)\in E} \max_{r\in\mathcal{R}} x^r_{uv} \tag{4.1}$$

Flow Conservation: for each demand d, SRLG set $r \in \mathcal{R}$,

$$\sum_{(v,j)\in\omega((s_d,0))} \varphi^{d,r}_{(s_d 0,vj)} = \sum_{(v,j)\in\omega((t_d,\ell(d))} \varphi^{d,r}_{(t_d\ell(d),vj)} = 1 \tag{4.2}$$

$$\sum_{(v,j)\in\omega((u,i))} \varphi^{d,r}_{(ui,vj)} \leq 2 \qquad v \in V \setminus \{(s_d,0),(t_d,\ell(d))\} \tag{4.3}$$

$$\sum_{(v',j')\in\omega((u,i))\setminus\{(v,j)\}} \varphi^{d,r}_{(ui,v'j')} \geq \varphi^{d,r}_{(ui,vj)}$$
$$v \in V \setminus \{(s_d,0),(t_d,\ell(d))\}, \ell \in \omega((u,i)) \tag{4.4}$$

Unavailable links in an SRLG failure event: for each $r \in \mathcal{R}$,

$$\sum_{d\in\mathcal{D}} \sum_{(u,v)\in r} \sum_{k=0}^{\ell(d)} \varphi^{d,r}_{(uk,vk)} = 0. \tag{4.5}$$

Bandwidth utilization in an SRLG failure event: for each SRLG set $r \in \mathcal{R}$, link $(u,v) \in E$,

$$\sum_{d\in\mathcal{D}} bw_d \cdot \sum_{k=0}^{\ell(d)} \varphi^{d,r}_{(uk,vk)} \leq x^r_{uv}. \tag{4.6}$$

4.4.3 A Column Generation Approach

One can apply the Dantzig-Wolfe decomposition to the above compact formulation, to exploit its block structure per demand $d \in \mathcal{D}$. The resulting model takes the form of a path flow formulation. We denote by Π^r_d, the set of service paths for a demand d in the SRLG failure situation r. Each service path π is associated with an integer value $a^\pi_{uv} \geq 0$ telling the number of times link (u,v) is used in the service path π.
Variables:
• $y^{d,r}_\pi \geq 0$, where $y^{d,r}_\pi = 1$ if demand d uses path π as a service path in the SRLG failure event $r \in \mathcal{R}$.
• $x_{uv} \geq 0$, is the bandwidth allocated on link $(u,v) \in E$.

Objective: minimization of the required bandwidth

$$\min \sum_{(u,v)\in E} x_{uv} \tag{4.7}$$

One service path for each demand and SRLG failure event: for all $d \in \mathcal{D}$, $r \in \mathcal{R}$

$$\sum_{\pi\in\Pi_d^r} y_\pi^{d,r} \geq 1. \tag{4.8}$$

Bandwidth utilization: for all $(u,v) \in E$, $r \in \mathcal{R}$

$$x_{uv} \geq \sum_{d\in\mathcal{D}} \sum_{\pi\in\Pi_d^r} bw_d \cdot a_{uv}^\pi \cdot y_\pi^{d,r}. \tag{4.9}$$

Given its very large number of variables, column generation is an efficient technique to handle the above linear integer programming model. One starts with a limited set of variables in a so-called restricted master program (RMP). At each iteration, the RMP is solved. The dual values associated to the constraints are used to generate new paths with negative reduced cost and the associated variables are added to the RMP that may enable to improve the current solution. This process is repeated until no more columns can be added to the RMP, i.e., no more columns with negative reduced cost exist. We refer to [Chv83] for more details regarding this technique.
The pricing subproblem is solved independently for each demand d and SRLG failure event r and it returns a service path π. It consists in finding a minimum cost service path in the layered graph where the weight of a link is defined according to the dual values of the associated constraint.
Variables:
• $\varphi_{(ui,vj)} \in \{0,1\}$, where $\varphi_{(ui,vj)}^{d,r} = 1$ if the flow is forwarded on link $((u,i),(v,j))$ of $G^L(d)$.
Let $\alpha_\omega^{sd} \geq 0$ and $\beta_{uv}^r \geq 0$ be the dual values relative to Constraints (4.8) and (5.16), respectively. The service path reduced cost for a given demand d and an SRLG failure situation r can be written as:

$$\min -\alpha_r^d + bw_d \cdot \sum_{(u,v)\in E} \beta_{uv}^r \cdot \sum_{k=0}^{\ell(d)} \varphi_{(uk,vk)} \tag{4.10}$$

The first term is a constant for each request, and the second term corresponds to a summation over the links of the network. Therefore, we can solve the

pricing problem using the following objective function:

$$\min \sum_{(u,v)\in E} \beta^r_{uv} \cdot \sum_{k=0}^{\ell(d)} \varphi_{(uk,vk)}. \tag{4.11}$$

Thus, for each request and for each failure situation, the pricing subproblem corresponds to a weighted shortest-path problem in the layered graph. In a given SRLG failure situation r and for all the demands $d \in D$, the weight of a link $((u,i),(v,j))$ of $G^L(d)$ is defined to be β^r_{uv} if $i = j$, 0 otherwise. Either one of these paths leads to a negative reduced cost column, or the current master solution is optimal for the unrestricted program. In the former case, the new configurations found are then added iteratively to the RMP. In the second case, the solution of the linear relaxation of the RMP z^*_{LP} is optimal.

Convergence of the basic column generation procedure suffers from dual oscillations as the number of constraints (5.16) is large. To improve the convergence and reduce the fluctuations in the dual variables, we use a piecewise linear penalty function stabilization described in [Pes+18].

Associated to the optimal solution of the linear relaxation of the RMP, for each demand d and SRLG failure situation r, there is a set of service paths identified by all the variables $y^{d,r}_\pi$ with value greater than 0. These service paths guarantee the minimum cost in terms of required bandwidth to deploy to guarantee the recovery in the splittable flow case. However, if we restrict our attention to the unsplittable flow case, we have to select only one service path for each demand and SRLG failure situation. The problem now consists in making this choice by reducing the *overflow* introduced in the network.
One possible way consists in changing the domain of the variables in the last RMP from continuous to integer and use an ILP solver. We refer to this strategy as MasterILP.

4.4.4 Benders Decomposition Approach

Applying Benders Decomposition technique [Ben62] to our compact model consists in splitting the original problem variables into first stage link capacity assignments on one hand, and second stage routing decisions on the other hand. The master problem is in terms of the x_{uv} variables. It takes the following form.

Objective: minimization of the bandwidth used in the network

$$\min \sum_{(u,v)\in E} x_{uv} \tag{4.12}$$

Metric inequalities

$$\sum_{(u,v)\in E} \mu_{uv} \cdot \delta_{uv,r} \cdot x_{uv} \geq \sum_{d\in\mathcal{D}} \lambda_d(\mu) \cdot bw_d \quad \forall\ \mu \in \mathbb{R}+^E \tag{4.13}$$

where the latter constraints are known as metric inequalities. They can be separated in polynomial time by solving an LP. Hence, they can be handled in a lazy way by generating them dynamically, which allows to solve the problem using the cutting plane algorithm. These cuts are iteratively added to the master problem until the difference between the lower bound, corresponding to the solution of the master problem, and the upper bound, corresponding to the solution of the subproblems, falls under a fixed value ϵ.

Benders separation subproblem is solved given the link bandwidth vector x. This capacity assignment is globally feasible (for the splittable problem) if and only if for each vector $\mu = \{\mu_{uv} \geq 0 : (u,v) \in E\}$ and for each SRLG failure situation $r \in \mathcal{R}$, the inequality

$$\sum_{(u,v)\in E} \mu_{uv} \cdot \delta_{uv,r} \cdot x_{uv} \geq \sum_{d\in\mathcal{D}} \lambda_d(\mu) \cdot bw_d$$

holds, where $\delta_{uv,r} \in [0,1]$ is the available portion of link (u,v) under scenario r, and $\lambda_d(\mu)$ is the length of the shortest path for demand d with respect to link metrics μ.

Associated to the optimal solution of the Master problem, we have the optimal link capacities in the splittable flow case, as in the Column Generation case. The main difference relies in the fact that we do not have the selected paths. We thus have to find a path for each demand and failure situations trying to minimize the *overflow*, with respect to the solution found in the splittable flow case.

4.4.5 The Min-Overflow problem

As it is costly to solve (exactly) the integer version of the master program, to obtain a "good" integer solution, we could use another approach. That

is, we may start by efficiently compute a fractional solution to the linear relaxation of the problem (i.e., when flows are splittable) using either the Column Generation algorithm or the Benders Decomposition technique and then try to obtain a *good* integer solution to the problem (i.e., when flows are unsplittable) by minimizing the cost to pay in terms of additional capacity (i.e., the *overflow*) over all the scenarios.
We define overflow as the total amount of additional bandwidth to be allocated in the network in order to satisfy all the demands. One possible strategy to do that may consist in considering each scenario one at a time, and formulating a multicommodity flow problem as an ILP. The objective function consists in minimizing the overflow to be allocated in the network. We refer to this strategy as IterILP.
If on one hand, this strategy leads to good results, on the other hand, it may not scale well, since we have to solve an ILP for each SRLG failure scenario. Another strategy consists in using an algorithm to route the demands while minimizing the overflow.

The problem to be solved for an SRLG failure scenarios which we refer to as Min Overflow Problem can be stated as follows.
Input: A graph $G = (V, E)$, a collection $\mathcal{D}$ of demands, each associated with a source, a destination and the units of flows to be routed. Also, each demand is associated with a set of paths, corresponding to the fractional solution of the splittable flow version of the problem. Lastly, a capacity function $c^* : (u, v) \rightarrow c^*_{uv}$, according to the optimal capacities found solving the linear relaxation of the general problem.
Output: a path for each demand.
Objective: minimize the overflow, i.e., minimize $\sum_{(u,v)\in E} \frac{\tilde{c}(u,v)}{c^*_{uv}}$ with $\tilde{c}(u, v)$ defined as the maximum between c^*_{uv} and the capacity of the link (u, v) after having selected one path per demand.

Note that, contrary to the classical version of the problem, we do not have hard capacity constraints to respect while computing an integer routing. Herein, the goal is to route all the demands reducing the increase in terms of capacity over each of the links (i.e., the overflow) with respect to the *free given capacities* already available in the network.
In the following, we give two theoretical results about the possibility of efficiently approximating the problem. On the negative side, we show that the problem is APX-Hard, and cannot be approximated within a factor of $1 + \frac{3}{320}$. On the positive side, we devise an approximation algorithm with a

constant performance ratio.

Proposition 4. The MIN OVERFLOW PROBLEM is APX-hard (and so is NP-Hard) and cannot be approximated within a factor of $1 + \frac{3}{320}$, unless P=NP.

Proof. We use a reduction from MAX 3-SAT. Let $\mathcal{I}$ be an instance of MAX 3-SAT with n variables $V_i, 1 \leq i \leq n$ and m clauses $C_j, 1 \leq j \leq m$. We associate each boolean variable V_i to a demand d_i asking for one unit of flow from a source s_{d_i} to a destination t_{d_i} connected by two paths $P_0(V_i)$ and $P_1(V_i)$. Selecting $P_1(V_i)$ (respectively $P_0(V_i)$) correspond to assign to V_i the true (respectively false) value.
We associate each clause C to to an edge (u_C, v_C) and we build the paths in the following way. For each variable V_i, we consider all the set $C(V_i)$ with all the clauses in which V_i appears as positive literal. $C(V_i) = C_{i_1}, C_{i_2}, ..., C_{i_m}$ with $i_1 \leq i_2 \leq ... \leq i_m$. Then, $P_1(V_i) = s_{d_i}, (u_{i_1}, v_{i_1}), (u_{i_2}, v_{i_2}), ..., (u_{i_m}, v_{i_m}), t_{d_i}$.
In a similar way, we consider now all the clauses in which V_i appears as negative literal. $C(\overline{V_i}) = C_{\bar{i}_1}, C_{\bar{i}_2}, ..., C_{\bar{i}_m}$ with $\bar{i}_1 \leq \bar{i}_2 \leq ... \leq \bar{i}_m$. $P_1(V_i)$ is defined as $s_{d_i}, (u_{\bar{i}_1}, v_{\bar{i}_1}), (u_{\bar{i}_2}, v_{\bar{i}_2}), ..., (u_{\bar{i}_m}, v_{\bar{i}_m}), t_{d_i}$.
As we build paths in this way, the load of an edge (u_C, v_C) is equal to the number of literals in the clause C assigned to the false value. There are $\sum_{i=1}^{n}(2i_m + 1)(2\bar{i}_m + 1) = 6m + 2n$ edges in the construction, as $\sum_{i=1}^{n} |C(V_i)| + |C(\overline{V_i})| = 3m$, the numbers of literals in the formula. We now assign each edge a capacity 2. A fractional routing always exists. Indeed, routing one half of the the charge of each demand d_i on $P_0(V_i)$ and the half on $P_1(V_i)$ is feasible, since after identification an arc receives at most $3 \times \frac{1}{2} \leq 2$. The case of an integral flow is quite different, since, in such a case, only one between $P_0(x)$ or $P_1(x)$ can be chosen. Since the capacity of the edges is 2, the cost will be 2 on each identified edge $\iff$ the formula is satisfiable. This proves that the problem is NP-complete (as 3-SAT is NP-complete). Then, we derive an inapproximability result using the fact that it is NP-hard to satisfy more than $\frac{7}{8}$ of the clauses (even if the formula is satisfiable) [Hras01]. So, we may have to pay 3 on $m/8$ edges (even though the optimal is 2 on all edges). Since the initial cost is less than 2 times the number of edges, it is less than $2 \times (6m + 2n) = 12m + 4n$. We have $n \leq \frac{m}{3}$. So, it is NP-hard to decide if the cost is 1 or $\frac{(12+\frac{4}{3})m+\frac{m}{8}}{(12+\frac{4}{3})m} = 1 + \frac{3}{320}$. □

Proposition 5. The MIN OVERFLOW PROBLEM can be approximated

within a factor of $(1 + \frac{1}{\mathrm{e}}) + \epsilon$, for any $\epsilon > 0$.

Proof. Let c^*_{uv} be the optimal capacity of an edge (u, v) in the splittable flow case. After having computed a fractional flow, we have associated to each demand $d \in \mathcal{D}$ a set consisting of $n(d) \geq 1$ paths $\mathcal{P}_d = \{P_{d,i} : i = 1, ..., n(d)\}$. Each path $P_{d,i}$ is associated to a multiplier $0 \leq \lambda_{d,i} \leq 1$ such that $\sum_{i=1}^{n(d)} \lambda_{d,i} = 1$ which gives the amount of flow $\lambda_{d,i} \cdot bw_d$ routed on $P_{d,i}$. Let $\lambda_{d,i}(uv)$ be the fraction of flow routed on the edge (u, v) by a demand d. Note that for each edge (u, v) we have $\sum_{d\in\mathcal{D}} \sum_{i=1}^{n(d)} bw_d \cdot \lambda_{d,i}(uv) \leq c^*_{uv}$ since by hypothesis these capacities are feasible for the splittable flow case. In order to find an unsplittable solution, we use a rounding–based heuristic referred to as Randomized Rounding, which assigns to a demand d a path $P_{d,i}$ with probability $\lambda_{d,i}$. We consider now the impact in terms of load on an edge (u, v). Let f_{uv} be the flow on (u, v) at the end of the rounding procedure. Clearly, for each edge (u, v) $\mathbb{E}(f_{uv}) \leq c^*_{uv}$ holds. Let O_{uv} be the overflow on the edge (u, v) defined as $\max(0, f_{uv} - c^*_{uv})$. We denote by $\mathsf{P}_0(uv) = \mathsf{P}[f_{uv} = 0]$ the probability that the edge (u, v) is not used.

$$\mathbb{E}[O_{uv}] = \mathsf{P}_0(uv) \cdot 0 + (1 - \mathsf{P}_0(uv))\, \mathbb{E}[f_{uv}|f_{uv} > 0] - c^*_{uv} \tag{4.14}$$
$$= (1 - \mathsf{P}_0(uv))\, \mathbb{E}[f_{uv}|f_{uv} > 0] - c^*_{uv}(1 - \mathsf{P}_0(uv)) \tag{4.15}$$

Moreover,

$$\mathbb{E}[f_{uv}] = \mathsf{P}_0(uv) \cdot 0 + (1 - \mathsf{P}_0(uv))\, \mathbb{E}(f_{uv}|f_{uv} > 0) \tag{4.16}$$

$$\mathbb{E}[f_{uv}|f_{uv} > 0] = \frac{\mathbb{E}[f_{uv}]}{1 - \mathsf{P}_0(uv)} \tag{4.17}$$

We can therefore bound the expected overflow of a link (u, v).

$$\mathbb{E}[O_{uv}] = \mathbb{E}[f_{uv}] - c^*_{uv}(1 - \mathsf{P}_0(uv)) \tag{4.18}$$
$$= \mathsf{P}_0(uv)c^*_{uv} - (c^*_{uv} - \mathbb{E}[f_{uv}]) \leq \mathsf{P}_0(uv)c^*_{uv} \tag{4.19}$$

Let us now consider the probability $\mathsf{P}_0(uv)$ that an edge is not used after the randomized rounding. Given an edge (u, v), we define $\mathcal{P}_{uv}$ to be the paths that contain (u, v) as an edge.

$$\mathsf{P}_0(uv) = \prod_{P_{d,i}\in\mathcal{P}_{uv}} (1 - \lambda_{d,i}) \tag{4.20}$$

The probability for an edge not to be selected is maximized when all $\lambda_{d,i}$ are equal (i.e., $\lambda_{d,i} = \frac{1}{|\mathcal{P}_{uv}|} \; \forall \; \lambda_{d,i} \in \mathcal{P}_{uv}$). Thus,

$$\mathsf{P}_0(uv) = (1 - \frac{1}{\rho|\mathcal{P}_{uv}|})^{|\mathcal{P}_{uv}|} \tag{4.21}$$

where ρ is defined to be $\frac{\mathbb{E}[f_{uv}]}{c^*_{uv}}$. This gives an upper bound for the possible value of $\mathsf{P}_0(uv)$. Indeed,

$$\mathsf{P}_0(uv) \leq \lim_{n\to\infty}(1 - \frac{1}{\rho n})^n = \frac{1}{\mathrm{e}^\rho}. \tag{4.22}$$

The function is minimized with $\rho = 1$. We thus get

$$\mathbb{E}[O_{uv}] \leq \frac{1}{\mathrm{e}^\rho}c^*_{uv} - (c^*_{uv} - \mathbb{E}[f_{uv}]) \tag{4.23}$$

$$\leq c^*_{uv}(\frac{1}{\mathrm{e}^\rho} - (1-\rho)) \leq \frac{1}{\mathrm{e}}c^*_{uv} \approx 0.37c^*_{uv}. \tag{4.24}$$

Finally, the expected cost of the solution provided is

$$\mathbb{E}\left[\frac{\sum_{(u,v)\in E} O_{uv}}{\sum_{(u,v)\in E} c^*_{uv}}\right] = \frac{\sum_{(u,v)\in E}\mathbb{E}[O_{uv}]}{\sum_{(u,v)\in E} c^*_{uv}} \leq \frac{1}{\mathrm{e}} \approx 0.37. \tag{4.25}$$

By using the Markov inequality, the probability that the obtained solution has a cost larger than $1.37(1+\epsilon)$ is at most $\frac{1}{1+\epsilon}$. The overflow resulting from the execution of the randomized rounding can be checked in polynomial time. If the overflow exceeds the factor of $(1 + \frac{1}{\mathrm{e}}) + \epsilon$, another trial may be necessary in order to find a solution below this value. The number of trials depends on the chosen value for ϵ. For instance, if we set $\epsilon = \frac{1}{10}$, we need an average of 10 trials in order to find a solution with cost not greater than 1.507 ($= 1.37 + 0.137$) times the optimal fractional one. □

As just shown, the problem of minimizing the overflow can be approximated efficiently for a single scenario. The proposed schema consists in a randomized rounding to be performed according to the value of the splittable flow solution. We may extend Randomized Rounding to the case of multiple scenarios by simply solving the scenarios in an iterative fashion. At each iteration, an SRLG $r \in \mathcal{R}$ is considered. First, a fractional capacitated multicommodity flow is solved. Then, a $(1 + \frac{1}{\mathrm{e}} + \epsilon)$–approximated integer

solution is found using the RandomizedRounding procedure. The overflow introduced (if any) by the procedure is then added. We refer to this method as Iterative Randomized Rounding. See Algorithm 7 for the pseudo-code of our proposed algorithm.

Algorithm 7 Iterative Randomized Rounding

1: Solve the linear relaxation of the general problem
2: $\tilde{c} \leftarrow c^*$
3: **for each** $r \in \mathcal{R}$ **do**
4: (a) route the demands on $G' = (V, E \setminus r, \tilde{c})$ solving a fractional multicommodity flow problem
5: (b) use RandomizedRounding to find a $(1 + \frac{1}{e} + \epsilon)$–approximate integer routing
6: (c) update $\tilde{c}$ with the introduced overflow (if any)
7: **end for**
8: **return** $\tilde{c}$

4.5 Numerical Results

In this section, we evaluate the performances of our proposed algorithms on both real and synthetic instances. The compared methods are as follows. MasterILP, in which in the last RMP is solved as an ILP by setting the domain of the paths variables from fractional to binary. IterILP, in which each scenario is solved independently with an ILP that has, as a goal, the minimization of the overflow and IterRR, in which instead of using an ILP to minimize the overflow, we use a $(1 + \frac{1}{e} + \epsilon)$–approximation algorithm.

4.5.1 Data sets

We conduct experiments on three real-world topologies from SNDlib [Orl+10b]: `polska`, (12 nodes, 18 links, and 66 demands), `pdh` (11 nodes, 34 links, and 24 demands) and `nobel-germany` (17 nodes, 26 links, and 121 demands). For these networks, we use the given traffic matrices. No information is available about the SRLGs for these networks. Thus, the collection of network failures $\mathcal{R}$ for these instances contains single edge failures. We also conduct experiments on randomly generated instances of different sizes. We build our synthetic instances using a similar method to the one in [KKV05]. We

generate two networks in which we place nodes in a unit square. In each of them, we add links according to the Waxman model [Wax88]. The probability of having a link (u, v) is defined as $\alpha \exp^{\frac{-\text{dist}(u,v)}{\beta L}}$ where $\text{dist}(u, v)$ is the Euclidean distance from node u to node v, L is the maximum distance between two nodes and α, β are real parameters in the range $[0, 1]$. One of the two networks represents the logical IP network, i.e., IP routers and IP links while the other represents the underlying optical network, i.e., cross-connect and fibers. Each IP node is mapped to the closest optical cross-connect and each IP link (u, v) is mapped onto the shortest path between u and v in the physical network. All the IP links using the same physical link are associated to an SRLG. In addition, we add an SRLG for each undirected link.
Demands are generated using the model described in [FT02]. The model takes into consideration the distance factor $\exp^{\frac{-\text{dist}(u,v)}{2L}}$ between two nodes u and v. As a result, the load of the demands between close pairs of nodes is higher with respect to pairs of nodes far apart. Finally, the chain of each demand is composed of 3 to 6 functions uniformly chosen at random from a set of 10 functions. Each VNF-enabled node can run up to 6 network functions. Similarly as in [HJG18b], locations are chosen according to their betweenness centrality, an index of the importance of a node in the network: it is the fraction of all shortest paths between any two nodes that pass through a given node. Experiments have been conducted on an Intel Xeon E5520 with 24GB of RAM.

4.5.2 Limits of an ILP-based approach.

To study the limits in terms of computing time of an ILP-based approach, we tested our optimization models on a small random topology with 10 nodes, 16 links, and 26 SRLGs.
In Figure 4.3, we show the impact of the number of demands on the execution time. We compare the time necessary to find an optimal solution (on the left) and the value of the solution found (on the right) by the ILP and by our proposed methods. For each experiment, we set a maximum time limit of one hour. If the time limit is exceeded, the solution reported represents the best solution found so far.
For just 30 demands, the time needed by Cplex 12.8 to find an exact solution exceeds 1 hour. For large instances, an optimal solution cannot be found

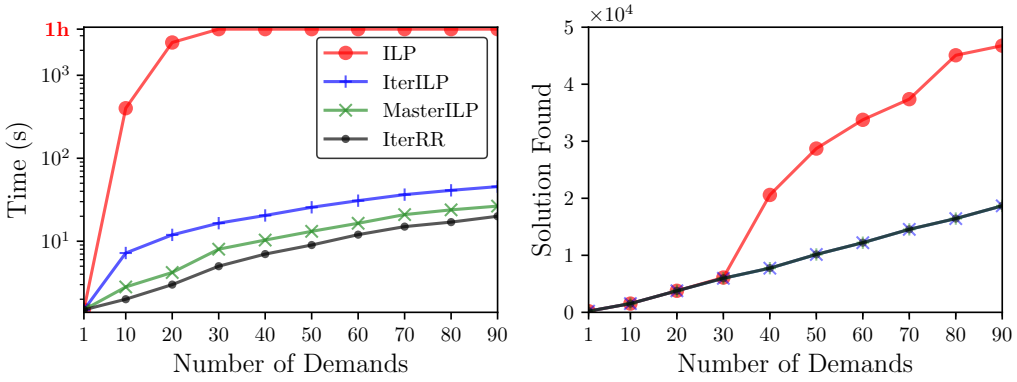


Figure 4.3: Time and Value of the solution found by the ILP and by our proposed methods as a function of the number of demands.

using an ILP approach in a reasonable amount of time. On the other hand, the proposed algorithms can compute solutions for larger instances fairly efficiently. Indeed, they only take 1 minute to solve the problem for 90 demands. As the considered network is small, the computed values tend to be close between them. Later in the discussion, using bigger networks we will highlight differences, limits, and advantages of the proposed approaches.

4.5.3 Performances of the optimization models

Table 4.1 summarizes the results of our proposed methods for the already presented 3 real networks and for 4 Waxman random networks. Networks are identified as `wxm_N` with `N` being the number of nodes. The number of demands is set to be $50, 100, 150$, and 200 for the $10, 20, 30$, and 40 nodes networks, respectively. Moreover, the number of resulting SRLGs for the Waxman random networks are $22, 40, 53$, and 70, respectively. The first column compares the Column Generation (ColGen) and the Benders Decomposition [Ben62] (Benders) techniques to find a fractional solution based on which the heuristics find an integer solution. The Column Generation techniques appears to be faster in finding the optimal solution z^*_{LP}. Indeed, on the largest considered network `wxm40` only takes 22 minutes to find an optimal solution, while Benders would require more than one hour. The remaining 3 columns refer to our optimization methods. For each method, we present both the time needed to find a solution $\tilde{z}_{ILP}$ as well as the ratio

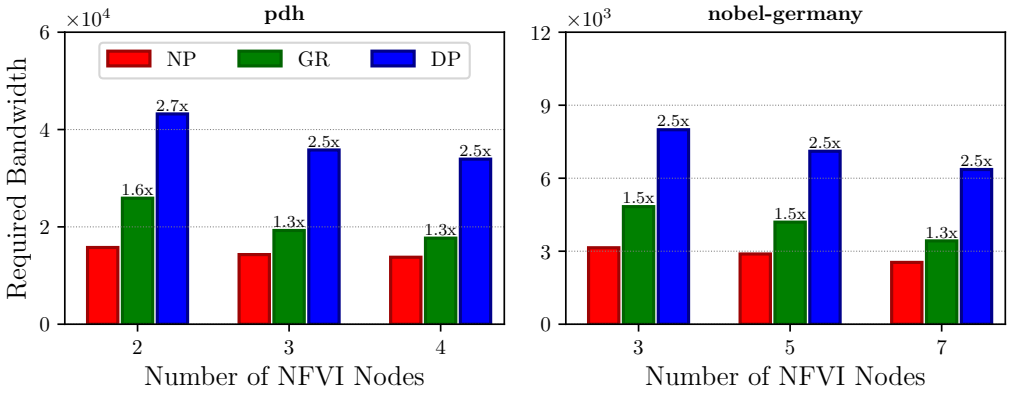


Figure 4.4: Bandwidth overhead comparison of the global rerouting (GR) and Dedicated Path Protection (DP) schemas with respect to the no-protection scenario (NP) for `pdh` and `nobel-germany` networks. Labels on top of the bars indicate the overhead with respect to the unprotected case.

$\epsilon = \frac{\tilde{z}_{ILP} - z^*_{LP}}{z^*_{LP}}$ with respect to the optimal fractional solution z^*_{LP}. ϵ gives an upper bound on the maximum overflow to pay in excess with respect to the optimal integer solution z^*_{ILP}, since the optimal integer solution may be larger than the fractional one. Both MasterILP and IterILP allow to find near–optimal solutions. As the size of the network increases, we begin to observe the limits of the IterILP approach, as it solves an ILP for each of the scenario. Although MasterILP demonstrates a better scalability and a very high accuracy, for larger networks we have a tradeoff between the time to find the solution and the quality of the solution found. Indeed, for `wxm40`, IterRR only takes 2 minutes to find a good solution with an accuracy of about 9%, while MasterILP requires 27 minutes to find a solution with an accuracy of 2.2%.

4.5.4 Varying Number of NFVI-enabled Nodes

NFVI nodes are expensive to both purchase and maintain (e.g., hardware, software licenses, energy consumption, and maintenance). If, on one hand, an over-provisioning corresponds to undue extra costs, on the other hand, under-provisioning may result in poor service to user and in Service Level Agreement (SLA) violations. It is thus necessary to find the right trade-off in terms of NFVI nodes in the network design phase.

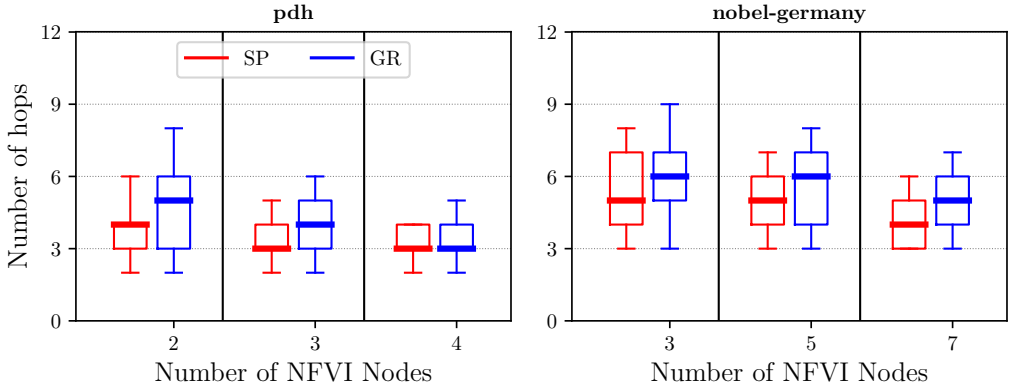


Figure 4.5: Hops distribution of the backup paths computed by the global rerouting schema (GR) compared with the shortest paths (SP) for `pdh` and `nobel-germany` networks. Boxes are defined by the first and third quartiles. Ends of the whiskers correspond to the first and ninth deciles.

Bandwidth overhead. In Figure 4.4, we compare the overhead in terms of bandwidth needed in the network by the global rerouting schema and Dedicated Path Protection with respect to the bandwidth needed in the unprotected case. For Dedicated Path Protection we compute, for each demand, two SRLG-disjoint paths, i.e., two paths such that no link on one path has a common risk with any link on the other path. In doing this, we set the bandwidth minimization as an optimization task. With an increasing number of VNFI nodes in the network, the required bandwidth decreases. However, the overhead with respect to the unprotected case tends to remain constant. Indeed, if with global rerouting we only need from 30 to 60% more bandwidth, with dedicated path protection we may need almost 3 times more bandwidth to guarantee the recovery.

Number of hops. In Figure 4.5, we show the impact of the number of NFVI nodes on the paths' number of hops distribution and compare them with the ones calculated using shortest paths on the layered network. As expected, we see that the number of hops decreases as the number of NFVI-enabled nodes increases. The reason is that, the more NFVI-nodes in a network, the higher the opportunity of easily finding closer NFVI-nodes which can perform some of the required network functions. Another result is that the length of the paths computed using our method are almost as good (in terms of number of hops) as the shortest paths.

Network	z^*_{LP}		**MasterILP**		**IterILP**		**IterRR**	
	ColGen	Benders	time	ϵ	time	ϵ	time	ϵ
pdh	22s	32s	11m	4%	1m	4.82%	40s	12.7%
polska	15s	18s	40s	0.22%	1m	0.1%	20s	1.4%
nb-germany	35s	1m	40s	0.17%	4m	0.06%	30s	3.2%
wxm10	10s	5s	50s	0.3%	40s	1%	10s	5.5%
wxm20	40s	2m	1m	0.6%	4m	0.6%	30s	2.7%
wxm30	3m	16m	6m	0.2%	21m	0.9%	1m	4.5%
wxm40	22m	>1h	27m	2.2%	>1h	-	2m	9.2%

Table 4.1: Numerical results for the proposed optimization models. First column refers to the time needed to find the optimal fractional solution z^*_{LP}. We set a maximum time limit of 1h. The other columns refer to the proposed methods to obtain an integer solution $\tilde{z}_{ILP}$. For each method, we show the additional time needed and the quality of the solution found, expressed as the ratio $\epsilon = \frac{\tilde{z}_{ILP} - z^*_{LP}}{z^*_{LP}}$.

4.5.5 Number of paths

In our considered protection schema, a demand may be rerouted on a different path in each of the possible SRLG failure situations. Even though our optimization models do not impose constraints on the number of distinct paths for a demand, the experimental results indicate that their number tends to be small in practice. In Figure 4.6, we show the distribution for the number of distinct paths of the demands for our considered networks. The number of distinct paths increases with the size of the network and tends to stay within the range $(5, 10)$ for most of them. For instance, for wxm40 we may have potentially 71 distinct paths to be used for a demand, one for each of the possible SRLG failure scenarios plus one for the no–failure case. As the results show, in such a case, 50% of the demands would use no more than 12 distinct paths.

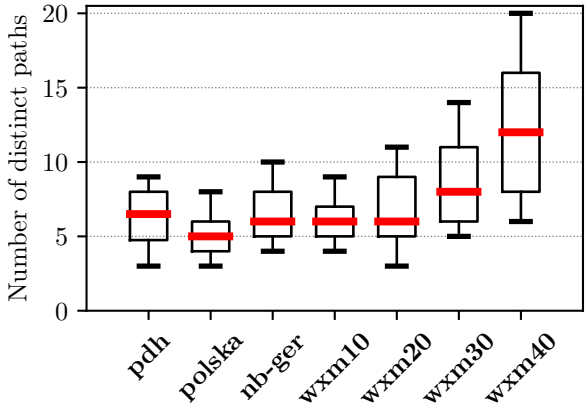


Figure 4.6: Distribution for the number of distinct paths for each demand. Boxes are defined by the first and third quartiles. Ends of the whiskers correspond to the first and ninth deciles. The median value is drawn in red.

4.6 Experimental evaluation

In this section, we discuss how to implement our global rerouting proposition with OpenFlow and evaluate it with the Mininet SDN emulator [LHM10]. Our evaluation in realistic conditions shows that implementation choices have a significant impact on the convergence time of protection mechanisms.

4.6.1 Implementation options

A first option to implement the protection scheme in OpenFlow is to let the OpenFlow controller fully update the flow tables on the switches upon failure. When the controller detects a failure, it sends the new flow tables to the impacted switches. This approach minimizes the memory usage on the switches but incurs high signaling overhead between the controller and the switches, and imposes the latter to install a full flow table at every network change. We refer to this option as *full* in the rest of the chapter. A variation of this option is to only send the changes to be performed on the flow tables to the switches to reduce the signaling load and the number of flow table updates on the switches. We name this option *delta*. Another option is to pre-install the flow tables for each SRLG failure scenario in the switches. When the controller sends a failure notification to a switch, the switch activates the

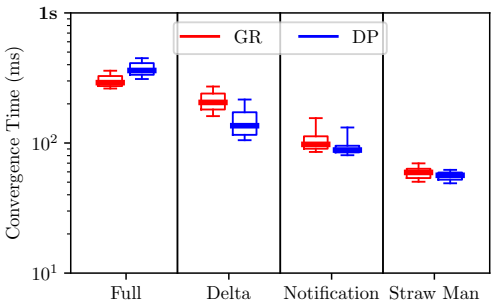


Figure 4.7: Convergence time comparison of various implementation options for Global Rerouting (GR) and Dedicated Path Protection (DP) for the `polska` network; $p = 10ms$.

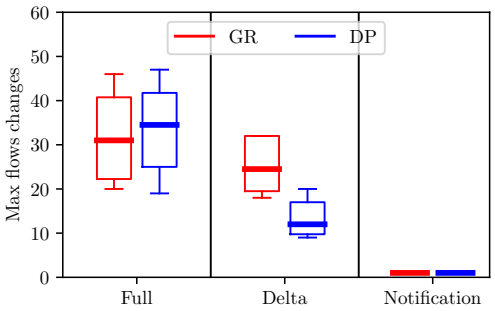


Figure 4.8: Comparison of the number of flow table changes of various implementation options for Global Rerouting (GR) and Dedicated Path Protection (DP) for the `polska` network.

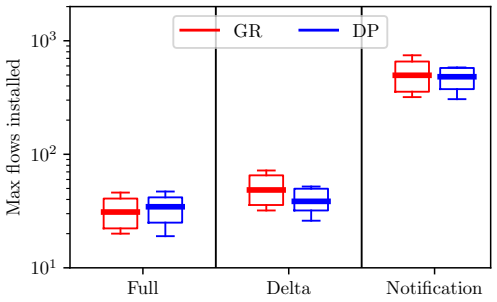


Figure 4.9: Comparison of the flow table sizes of various implementation options for Global Rerouting (GR) and Dedicated Path Protection (DP) for the `polska` network.

appropriate flow table in only one operation. This approach minimizes the signaling load and flow table changes but consumes more memory on the switches than the other options. This option is called *notification* in the rest of the chapter.

4.6.2 Experimental setup

Our experimental platform is a dual Intel Xeon E5506 CPU server with 48GB of RAM running Mininet 2.2.2 [LHM10] and the controller *OpenDaylight* Nitrogen [Med+] with OpenFlow 1.3.

We aim at understanding the impact of the technical choices on the convergence time in realistic operational scenarios. This is why we use the most popular OpenFlow controller, OpenDaylight, even though it focuses more on features than on performances. We implemented the routing logic as a network application orchestrator that communicates with the controller with the HTTP OpenDaylight Northbound API. This approach is recommended as it decouples the implementation of the logic from the implementation of the controller, at the cost of communication and abstraction overhead.

We also made a *straw man* implementation to assess the best possible performance one could have. It is equivalent to the full option but is implemented directly in Mininet with Open vSwitch commands. This solution is impractical and should only be considered as an ideal reference point.

Even though our implementation supports any type of network, because of the limited number of CPU cores on our emulation server, we evaluated only the `wxm10` and the `polska` networks. Due to space limitations, we only discuss the `polska` network as results are similar for both networks.

4.6.3 Convergence time

The *convergence time* is the span of time between a failure event and the moment in which all switches are updated to be in a state that circumvents the failure. To measure the convergence time, we continuously probe end-to-end paths with UDP datagrams. This method has resolution of $2 \cdot p$, where p is the probing period; in this chapter $p = 10\,\text{ms}$.

Fig. 4.7 shows the convergence time for our three OpenDaylight implementation options and the straw man. It compares our Global Rerouting (GR) protection scheme to the Dedicated Path Protection (DP) scheme.

The figure highlights the importance of implementation choices on the convergence time: the notification option significantly outperforms the other options. The straw man implementation also shows that the tools used to implement the protection scheme have a significant impact on the convergence time as, all things considered, our straw man is just a way of implementing the full option without a controller. Actually, on average, 68% of the convergence time in OpenDaylight implementations is caused by the usage of the Northbound API that incurs multiple marshalings and unmarshalings. It is worth to notice that all implementation options offer sub-second convergence time for the considered network and, in some situations, the straw man can even go below 50 ms.

Comparing Fig. 4.7 with Fig. 4.8 shows that there is a direct link between the number of changes to be performed on the switches and the convergence time. Fig. 4.8 reports, for each switch, the maximum number of flow table changes observed expressed in number of flow entries for the three OpenDaylight implementation options. The full option requires the highest number of changes and is the slowest, while the notification option needs the least number of changes and is the fastest. We see that dedicated path protection has longer convergence time than global rerouting when the full implementation is used. This is because with DP two SRLG-disjoints paths are always provided while GR only provides the paths of the current scenario. On the contrary, DP converges faster than GR with the delta implementation as less path changes are needed for DP than for GR. Finally, GR and DP reach the same level of performance when notifications are used. The slight difference in disfavor of GR comes from the switches themselves that must manipulate larger flow tables, which may cause more frequent L1 cache thrashing in our emulation environment.

4.6.4 Operational trade-offs

Based on the convergence time, one would recommend to deploy the notification option. However, the reduction of the convergence time comes at the cost of increasing flow table sizes on switches. Fig. 4.9 reports, for each switch, the maximum observed flow table size expressed in number of flow entries for the three OpenDaylight implementation options. The full option minimizes the number of entries as it only requires to have the flow table for the current routing case. The delta option consumes slightly more space than the full one as the flow table always contains the “no-failure” scenario

flow table and the additional flow entries needed to circumvent the current failure. Finally, the notification option has significantly larger flow tables (one order or magnitude more) as flow tables always contain all the potential failure scenarios in addition to the "no-failure" tables.

In addition, operators must chose the method to detect failures. In particular cases, only active probing methods can be used which are inherently slow, e.g., the minimum configurable time period for BFD on some Cisco interfaces is 50 ms [Cis], and then the time to re-route traffic becomes negligible compared to the time to detect the failure. The detection time is also constrained by the location of the controller in the network as the controller itself must detect the failure before applying the re-routing scheme to the switches. As failure detection is an orthogonal problem that must be tackled by all protection and recovery mechanisms, we considered the ideal case where failures are detected instantaneously so that the reader can focus on the actual costs of the re-routing scheme.

As the robustness of the controller is an orthogonal problem that must be treated by all SDN solutions and because it is already largely studied [Zha+18], it was not considered in our study.

4.7 Conclusion

In this chapter, we studied the ISP network dimensioning problem with protection against a Shared Risk Link Group failure. We considered a path-protection method based on a global rerouting strategy, which makes the protection method optimal in terms of bandwidth. We proposed algorithms to compute the backup paths for the demands which rely on the Column Generation technique. We validated them experimentally on real-world and on random generated instances. Finally, we showed the applicability of the global rerouting protection method thanks to SDN with a real implementation in the OpenDaylight controller. We also discussed important trade-offs between getting a lower convergence time and having a lower memory footprint on the switches.

Chapter 5

Path Protection for Service Function Chains

Contents

The content of this chapter has been published before in [Tom+18c].

5.1 Introduction

Network failures have been widely investigated (see, e.g., [GJN11],[Tur+10],[PJ13]). One of the key findings of [GJN11] is that links experience about an order of magnitude more failures than devices. Moreover, according to their analysis, low-cost commodity switches are highly reliable. This finding is also confirmed by [PJ13]. On the other hand, links are failure-prone. Indeed, in

a monitored network, each link experiences in average 16 failures per year, considering a five years period [Tur+10]. Despite this, most of the failures have very short duration. The majority of link failures are solved within 5 minutes (the median time to repair is 13s). Another finding is that link failures tend to be isolated. The short time to repair and the absence of a relation between link failures motivate us to focus our attention on the *single link failure* scenario.

Fault management techniques can be grouped into two categories: *restoration* and *protection*. Restoration is a reactive approach in which a backup path is computed and established after a failure. Protection is a proactive technique in which capacity on links is reserved during connection setup. Restoration schemes are more efficient in utilizing capacity, but, on the other side, protection schemes have a faster restoration time and offer a guaranteed recovery [SRM02]. Data plane restoration [Sha+11; Sta+11] and protection [Sha+13; Sga+13] have been both adressed in the context of SDN.

There are different protection schemes. In *dedicated protection*, some spare capacity is reserved for each backup path. This implies that the backup resources are used for at most one path. In *shared protection*, backup paths can share some link capacity if failures in their primary paths do not occur simultaneously. Thus, in shared protection, capacity is used more efficiently [ZS00]. However, dedicated protection is often used by network operators because of its simplicity. We thus study both protection schemes in this paper.

Each protection scheme may have two different recovery mechanisms: a local repair (i.e., link protection) or an end-to-end repair (i.e., path-protection). Link protection schemes reroute the traffic around the failing link. In path protection schemes, the traffic is rerouted on a link-disjoint backup path. In this case, the backup path would be used in all the failure situations that involve links of the primary path. Path-protection mechanisms have been shown to lead to better resource utilization compared to link protection [RM99; IMG98].

In this chapter, we consider the problem of *providing for each demand, a primary and a link-disjoint backup path, under both dedicated and shared protection schemes*. Moreover, the problem also consists in *provisioning VNFs* in order to ensure that the *traversal order* of the network functions by each path is respected. This adds a challenge to the classical version of the problem.

Our goal is to minimize the bandwidth requirements while ensuring that the delays on primary and backup paths stay below SLAs. Our contributions are as follows:

- To the best of our knowledge, we are the first to propose a *scalable exact method* to solve the problem of *reliable* service function chaining. The method is based on a decomposition model using column generation.
- The model allowed us to solve the problem with dedicated and shared protection schemes for networks with up to 1000 traffic requests.
- We also studied the *costs in terms of bandwidth and computation requirements* of both protection schemes and for networks of different sizes. When service function chaining is considered, dedicated protection requires three times more bandwidth and two times more processing than without protection. The ratios drop to 1.5 and 1.25 for shared protection.
- We additionally study the impact of the number of nodes of the network being able to host VNFs on the bandwidth requirements and on the delays of both primary and backup paths.

The paper is organized as follows. We study two different protection schemes: Dedicated Protection, in Section 5.4.1 and Shared Backup Path Protection, in Section 5.4.2. For each of them, we propose both a compact Integer Linear Program (ILP) formulation and a decomposition model. In Section 5.5, we compare the models and show the superiority of the decomposition models over the compact ILP formulations in terms of scalability. We then study the impact of design choices, such as the number of VNF nodes and the kind of protection, on the experienced delay by both the primary and backup paths of the demands, as well as the impact on the bandwidth requirements.

5.2 Related Work

The design of survivable networks has been widely studied in the network literature (see, e.g., [AA99],[RM99]). However, when dealing with NFV and SFCs, an additional challenge is to map network functions to nodes and to guarantee that the execution order of the network functions is respected in both primary and backup paths.

The problem of guaranteeing service continuity in Service Function Chain scenario has started to be investigated recently. Both *restoration* [Sou+17; LM15] and *protection* [Hma+17; Ye+16; BBS16] techniques have been investigated. In [Cas+17], the authors propose an approach to guarantee protection by replication. They consider the problem of placing VNFs on a NFV Infrastructure to satisfy the requests while guaranteeing high availability. They propose an ILP along with greedy heuristics to overcome the scalability issues of the ILP formulation.
In [Sou+17], the authors address VNF placement and chaining in the presence of physical link failures. The proposed algorithm makes use of a Monte-Carlo Tree Search algorithm to place VNFs and simultaneously steer traffic flows across them. When a link fails, the algorithm reactively re-maps the failed virtual links in other substrate paths. In Hmaity *et al.* [LM15], the authors consider the problem of recovering the traffic path after the failure of a network function. In their proposed solution, an alternative VNF is selected, in a greedy manner, to replace the failed one and then the communication is ensured by allocating a path between the new VNF and its neighbors.
In [Ye+16], the authors consider node and link failures and they propose a heuristic algorithm with the goal of meeting the client's reliability requirements. They propose two algorithms. The first one is based on dedicated protection and the second one on shared protection. In [Hma+17], the authors propose a compact ILP model in order to provide resiliency against single node, virtual link, and single node/single virtual link failure scenarios. They aim to reduce the number of VNF nodes used. The difference with our work is that the authors consider *link protection*, while we look into *path protection*, and their ILP models are not scalable.
[BBS16] discusses measures on how to backup resources in order to protect network services from failures. They consider both node and link failures and propose a resource allocation algorithm heuristic-based that aims at keeping the number of physical resources allocated to VNF chains low.
The *main difference* with our work is that we propose a scalable exact decomposition model to provide reliable service function chaining. (Other differences is that using path protection in order to minimize network bandwidth was also not considered in this setting.) Column generation techniques have been shown to be effective in dealing with both Service Function Chaining [HJG17a; Hui+18c] and failure protection [AV14]. In [HJG17a], the authors propose a decomposition modeling for the SFC Problem with the goal of optimizing the bandwidth. Through extensive numerical experiments, they

show that their model can solve exactly and in an efficient way the problem. We here extend their results to the case of unreliable networks in the case of single link failure scenario.

5.3 Problem and Notations

We model the network as a graph $G = (V, L)$, where V represents the set of nodes and L the set of links. A request is modeled as a quadruple (v_s, v_d, c, D^c_{sd}) with v_s the source, v_d the destination, $c = f^c_1, f^c_2, ..., f^c_{n_c}$ the sequence of VNFs that need to be performed with n_c the chain length, and D^c_{sd} the required units of bandwidth.
Each network function f has associated processing requirements per unit of bandwidth, denoted with Δ_f. Namely, given a request (v_s, v_d, c, D^c_{sd}), the number of cores needed to process the $i-th$ function of the chain c is equal to $D^c_{sd} \cdot \Delta_{f^c_i}$.
Different chains may have different maximum tolerated latency. For instance, the latency requirement of Video Streaming is less stringent than Online Gaming. Following a similar idea as in [Hma+17], we associate to each chain c a maximum tolerated delay, denoted as $\phi(c)$. Each network function f is associated with a processing latency per unit of bandwidth ρ^u_f, which also depends on the node in which the function is performed, and each link with a transmission and propagation latency λ_l.
Not all nodes may be enabled to run virtual functions. We denote by $V^{\text{VNF}} \subseteq V$ the set of VNF-enabled nodes equipped with Commercial Off The Shelf (COTS) hardware. We are given for each node $v \in V^{\text{VNF}}$ a capacity CAP_v, representing the amount of available resources, such as CPU, memory, and disk. Similarly, for each link $\ell \in L$, we are given the transport capacity CAP_ℓ.
The optimization task is to minimize the amount of bandwidth used in the network. At the same time, the problem consists in providing to each demand an edge disjoint backup path and to guarantee that the traversing order of the functions is respected in both primary and backup paths. Both node and link capacities must be respected, as well as the maximum tolerated latency for each request.

As in [HJG17a], to model the function ordering problem, we use a layered G^L graph with n_c+1 layers. We denote by $u(i)$ the copy of node u in layer i. The paths for demand D^c_{sd} starts from node $v_s(0)$ in layer 0 and ends at node

G	$= (V, L)$ optical (grid) network
V^{VNF}	$\subseteq V$ = subset of nodes which are enabled to host virtual network functions
$\mathcal{SD}$	Set of node pairs with some demand
D^c_{sd}	bandwidth demand from s to d for chain c
Δ_f	# required cores per bandwidth unit for function f
CAP_ℓ	transport capacity (bandwidth) of link ℓ
CAP_v	core capacity of node v
n^c	length (i.e., number of functions) of the chain c
f^i_c	the i^{th} function in chain c
$\phi(c)$	maximum tolerated delay for the chain c
λ_l	latency of physical link ℓ
ρ^f_u	processing time per bandwidth unit for function f on node u

Table 5.1: Notation

$v_d(n_c)$ in layer n_c. Layer i corresponds to nodes of the paths encountered after the i^{th} function of the service chain.
Using link $(u(i), v(i))$ on G^L, implies using link (u, v) on G. On the other hand, using link $(u(i), u(i+1))$ implies using the $(i+1) - th$ function of the chain at node u.

5.4 Optimization Models

We now present the optimization models for the dedicated and shared protection schemes, in Section 5.4.1 and 5.4.2 respectively. For each scheme, we present both a compact ILP formulation and a decomposition model.

5.4.1 Dedicated Protection

We consider here a dedicated protection scheme, also known as 1+1 protection. The capacity for the backup path is fully reserved. This is a method often used by operators in the case each demand is load balanced over both paths used at less than 50%. When a failure involving the primary path

happens, the traffic on the failing path is switched to the second path.

5.4.1.1 Model NFV_ILP_DP

In the case of dedicated protection, the problem consists in finding two *sd*-paths in G^L for each demand. Note that the paths have to be edge disjoint (i.e., if a path uses a link l for a layer of G^L then the other path cannot use l in any layer of G^L) but not node disjoint.
Variables:
• $\varphi_{\ell,p}^{sd,c,i}, \varphi_{\ell,b}^{sd,c,i} \in \{0,1\}$, where $\varphi_{l,p}^{sd,c,i} = 1$ ($\varphi_{l,b}^{sd,c,i} = 1$) if (v_s, v_d, c, D_{sd}^c) is provisioned on link ℓ for the primary (backup) path.
• $a_{v,p}^{sd,c,i}, a_{v,b}^{sd,c,i} \in \{0,1\}$ where $a_{v,p}^{sd,c,i} = 1$ ($a_{v,b}^{sd,c,i} = 1$) if f_{i+1}^c is installed on node v for the primary (backup) path. If $v \notin V^{VNF}$, $a_{v,1}^{sd,c,i}$ and $a_{v,2}^{sd,c,i}$ are set to 0.

Objective: minimization of the bandwidth used in the network

$$\min \quad \sum_{(v_s,v_d)\in\mathcal{SD}} \sum_{c\in C_{sd}} D_{sd}^c \sum_{\ell\in L} \sum_{i=0}^{n^c} (\varphi_{\ell,p}^{sd,c,i} + \varphi_{\ell,b}^{sd,c,i}) \tag{5.1}$$

Constraints: both primary and backup paths must satisfy the following constraints. They are written for the general case.

Flow Conservation: for all $(v_s, v_d) \in \mathcal{SD}$, $c \in C_{sd}$,

$$\sum_{\ell\in\omega^+(v)} \varphi_\ell^{sd,c,0} - \sum_{\ell\in\omega^-(v)} \varphi_\ell^{sd,c,0} + a_v^{sd,c,0} = \begin{cases} 1 \text{ if } v = v_s \\ 0 \text{ else} \end{cases} \tag{5.2}$$

$$\sum_{\ell\in\omega^+(v)} \varphi_\ell^{sd,c,n^c} - \sum_{\ell\in\omega^-(v)} \varphi_\ell^{sd,c,n^c} - a_v^{sd,c,n^c} = \begin{cases} -1 \text{ if } v = v_d \\ 0 \text{ else} \end{cases} \tag{5.3}$$

$$\sum_{\ell\in\omega^+(u)} \varphi_\ell^{sd,c,i} - \sum_{\ell\in\omega^-(u)} \varphi_\ell^{sd,c,i} + a_v^{sd,c,i} - a_v^{sd,c,i-1} = 0. \quad 0 < i < n_c \tag{5.4}$$

Link capacity: for all $\ell \in L$,

$$\sum_{(v_s,v_d)\in\mathcal{SD}} \sum_{c\in C_{sd}} D^c_{sd} \sum_{i=0}^{n^c} (\varphi^{sd,c,i}_{\ell,p} + \varphi^{sd,c,i}_{\ell,b}) \leq \text{CAP}_\ell. \tag{5.5}$$

Node capacity: for all $v \in V^{\text{VNF}}$,

$$\sum_{(v_s,v_d)\in\mathcal{SD}} \sum_{c\in C_{sd}} \sum_{i=0}^{n^c-1} D^c_{sd}\Delta_{f^c_i}(a^{sd,c,i}_{v,p} + a^{sd,c,i}_{v,b}) \leq \text{CAP}_u. \tag{5.6}$$

Latency: for all $(v_s, v_d) \in \mathcal{SD}$, $c \in C_{sd}$,

$$\sum_{\ell\in L}\sum_{i=0}^{n^c} \varphi^{sd,c,i}_{\ell}\lambda_\ell + \sum_{i=0}^{n^c-1} D^c_{sd}\,\rho^v_{f^c_i}\,a^{sd,c,i}_v \leq \phi(c). \tag{5.7}$$

In order to guarantee that the paths are edge disjoint, we add the following constraint. For all $(v_s, v_d) \in \mathcal{SD}, c \in C_{sd}, \ell \in L$,

$$\sum_{i=0}^{n^c} \varphi^{sd,c,i}_{\ell,p} + \sum_{i=0}^{n^c} \varphi^{sd,c,i}_{\ell,b} \leq 1. \tag{5.8}$$

5.4.1.2 Model NFV_CG_DP

We now propose a decomposition model for the dedicated protection scenario. Each configuration consists of a Service Path. A Service Path for a request (v_s, v_d, c, D^c_{sd}) is composed of: *(i)* a path, i.e., an ordered set of nodes from the source to the destination, and *(ii)* a set of locations for the VNFs in the SFC request. The goal of the Master Problem is thus to select a pair of configurations, i.e., Service Paths for each request.
The set of configurations must be chosen in such a way that: *(i)* each request is associated to a pair of edge-disjoint configurations; *(ii)* node and link capacities are respected and *(iii)* the overall required bandwidth is minimized.
• $\pi \in \Pi^c_{sd}$ is a service path from s to d. A service path is composed of a path and a set of node/function pairs (v, f) expressing that the function f is installed on node v.
• $a^f_{v,\pi} \in \{0, 1\}$, where $a^f_{v,\pi} = 1$ if f is installed on node v for service path $\pi \in \Pi^c_{sd}$ w.r.t sd, c.

• $\delta_\ell^\pi \in \{0,1\}$, where $\delta_\ell^\pi = 1$ if link ℓ belongs to path π.

Variables:
• $y_{\pi,p}^{sd,c}, y_{\pi,b}^{sd,c} \geq 0$, where $y_{\pi,p}^{sd,c} = 1$ ($y_{\pi,b}^{sd,c} = 1$) if the request from v_s to v_d for service chain c is forwarded through service path π for the primary (backup) path.

Objective

$$\min \sum_{(v_s,v_d)\in\mathcal{SD}} \sum_{c\in C_{sd}} \sum_{\pi\in\Pi_{sd}^c} D_{sd}^c \,\text{LEN}(\pi)\,(y_{\pi,p}^{sd,c} + y_{\pi,b}^{sd,c}) \tag{5.9}$$

One primary and one backup path per demand and per chain:

$$\sum_{\pi\in\Pi_{sd}^c} y_{\pi,p}^{sd,c} \geq 1. \tag{5.10a}$$

$$\sum_{\pi\in\Pi_{sd}^c} y_{\pi,b}^{sd,c} \geq 1. \tag{5.10b}$$

Edge disjoint primary and backup path per demand, per chain and per link:

$$\sum_{\pi\in\Pi_{sd}^c} \delta_\ell^\pi (y_{\pi,p}^{sd,c} + y_{\pi,b}^{sd,c}) \leq 1. \tag{5.11}$$

Link capacity: for all $\ell \in L$,

$$\sum_{(v_s,v_d)\in\mathcal{SD}} \sum_{c\in C_{sd}} \sum_{\pi\in\Pi_{sd}^c} D_{sd}^c\, \delta_\ell^\pi\, (y_{\pi,p}^{sd,c} + y_{\pi,b}^{sd,c}) \leq \text{CAP}_\ell. \tag{5.12}$$

Node capacity: for all $v \in V^{\text{VNF}}$,

$$\sum_{(v_s,v_d)\in\mathcal{SD}} \sum_{c\in C_{sd}} \sum_{f\in F_c} \sum_{\pi\in\Pi_{sd}^c} \Delta_f D_{sd}^c\, a_{v,\pi}^f\, (y_{\pi,p}^{sd,c} + y_{\pi,b}^{sd,c}) \leq \text{CAP}_v. \tag{5.13}$$

The role of the pricing problem is to generate a valid *Service Path* for a given request. Once again, the formulation uses the layered graph (G^{L}). We denote by $u^{(j)}$ the vector of dual variables of constraints (j) in the RMP. Note that these values are given as input to the pricing problem in the column generation solution process.
Variables:
• $a_v^i \in \{0,1\}$, where $a_{v_{f_i}} = 1$ if f_i^{st} is installed on node v.
• φ_ℓ^i, where $\varphi_\ell^i = 1$ if the flow forwarded on link ℓ on layer i.

For each request (v_s, v_d, c, D_{sd}^c), we use two pricing problems to generate a primary and a backup service path. A Service Path generated by the

pricing problems must respect constraints (6.22)-(5.7) of the model NFV_-ILP_DP presented before. The two paths are then added to the collection of service paths Π_{sd}. The only difference between the two sub-problems for each request relies in the objective function of the Pricing Problem. The objective function of the pricing problem for a primary path can be written as follows.

$$\min \quad D^c_{sd} \sum_{\ell \in L} \sum_{i=0}^{n_c} \varphi^i_\ell - u^{(10a)}_{sd,p} - \sum_{\ell \in L} \sum_{i=0}^{n_c} \varphi^i_\ell u^{(5.11)}_{sd} + \sum_{\ell \in L} u^{(5.12)}_\ell D^c_{sd} \sum_{i=0}^{n_c} \varphi^i_\ell + D^c_{sd} \sum_{v \in V} u^{(5.13)}_v \sum_{i=0}^{n_c} \Delta_f a_{vf_i} \quad (5.14)$$

5.4.2 Shared Protection

We now consider a shared protection scheme, also known as 1:1 protection. The capacities for the backup paths are reserved in case of a single link failure. In this case, the network resources may be shared among different failure scenarios. For each failure scenario, we guarantee that link and node resources are not exceeded.
We denote by Ω the set of all the possible failure situations. Since we are considering only single link failures, $\Omega = L \cup \emptyset$.

5.4.2.1 Model NFV_ILP_SP

In the shared protection case, the objective changes. Indeed, while in the dedicated protection case the required bandwidth depends on the length of the paths, this is no longer true here.
Let $x_\ell \geq 0$ be the bandwidth requirements of link $\ell \in L$.
The objective is thus:

$$\min \sum_{\ell \in L} x_\ell \quad (5.15)$$

In addition to the variables introduced in the dedicated protection scheme, we now define two new kinds of variables. Their goal is to tell us, given a failure situation ω which link the backup paths use and on which node a function will be performed.
Variables:
• $z^{sd,c}_{\ell,\omega} \in \{0,1\}$, where $z^{sd,c}_{\ell,\omega} = 1$ if the request uses link ℓ in the backup path

in the failure situation ω.
• $z^{sd,c}_{i,v,\omega} \in \{0,1\}$, where $z^{sd,c}_{i,v,\omega} = 1$ if the request uses function the i^{th} function of the chain C_{sd} on node v in the backup path in the failure situation ω.
We now describe the modified constraints, with respect to NFV_ILP_DP.
Link Capacity: for all $\ell \in L$, $\omega \in \Omega$

$$\sum_{(v_s,v_d)\in\mathcal{SD}} \sum_{c\in C_{sd}} D^c_{sd} \,(\sum_{i=0}^{n^c} \varphi^{sd,c,i}_{\ell} + z^{sd,c}_{\ell,\omega}) \leq x_\ell \leq \text{CAP}_\ell. \tag{5.16}$$

Node Capacity: for all $v \in V$, $v \in V^{\text{VNF}}$, $\omega \in \Omega$

$$\sum_{(v_s,v_d)\in\mathcal{SD}} \sum_{c\in C_{sd}} D^c_{sd} \sum_{i=0}^{n^c-1} \Delta_{f^c_i}(a^{sd,c,i}_v + z^{sd,c}_{i,v,\omega}) \leq \text{CAP}_v. \tag{5.17}$$

5.4.2.2 Model NFV_CG_SP

Following a similar idea as in [Sti+07], with each $\pi \in \Pi_{sd}$ we now represent a configuration as a service paths pairs (π_p, π_b) from s to d. The reason relies on the fact that, by using the same model as NFV_CG_DP, in order to ensure that node and link capacities are not exceeded in any failure situation $\omega \in \Omega$, we would need additional variables and constraints. This would lead to an increase in the size and consequently, to the complexity of the Reduced Master Problem. The price to pay for this choice is an increase in the complexity of the Pricing Problems, that would lead to a higher resolution time with respect to the dedicated protection case, as we will see in Section 5.5. More specifically, while the Pricing Problem in NFV_CG_DP reduces to be a Shortest Path Problem on the layered graph G^L, in this case solving the Pricing Problem is NP-Hard [Sti+07].
Each $\pi \in \Pi_{sd}$ is associated with a binary value SDN switch $^{\omega}_{\pi}$ telling if in failure situation ω the primary path cannot be used (i.e., if the failure involves a link that belongs to the primary path).
Variables:
• $y^{sd,c}_{\pi} \geq 0$, where $y^{sd,c}_{\pi} = 1$ if demand from v_s to v_d for service chain c uses $\pi = \{\pi_p, \pi_b\}$ as pair of service paths. • $x_\ell \geq 0$, is the bandwidth required on link $\ell \in L$.
Objective

$$\min \sum_{\ell\in L} x_\ell \tag{5.18}$$

Exactly one path pair per demand and per chain:

$$\sum_{\pi \in \Pi_{sd}^{c}} y_{\pi}^{sd,c} \geq 1. \tag{5.19}$$

Link capacity: for all $\ell \in L$ and failure situations ω:

$$\sum_{(v_s,v_d)\in\mathcal{SD}} \sum_{c\in C_{sd}} \sum_{\pi\in\Pi_{sd}^{c}} y_{\pi}^{sd,c} D_{sd}^{c} \\ (\delta_{\ell}^{\pi_p} + SDNswitch_{\pi}^{\omega}\, \delta_{\ell}^{\pi_b}) \leq x_{\ell} \leq \text{CAP}_{\ell}. \tag{5.20}$$

Node capacity: for all $v \in V^{VNF}$ and failure situations ω,

$$\sum_{(v_s,v_d)\in\mathcal{SD}} \sum_{c\in C_{sd}} \sum_{\pi\in\Pi_{sd}^{c}} \sum_{f\in F_c} y_{\pi}^{sd,c} \Delta_f D_{sd}^{c} \\ (a_{v,\pi_p}^{f} + SDNswitch_{\pi}^{\omega} a_{v,\pi_b}^{f}) \leq \text{CAP}_{v}. \tag{5.21}$$

In this case, the role of the Pricing Problem is to generate a pair of valid *Service Paths* for a given request. The path pair $\pi = (\pi_p, \pi_b)$ has to be link-disjoint but not node-disjoint. Given the layered graph (G^{L}), if one of the two paths uses link ℓ in some of the layers, the other path cannot use link ℓ in any of the layer of G^{L}.
We look at each iteration at the minimum cost path pair according to the dual values provided by the Restricted Master Problem. As in the dedicated protection scheme, the pricing problem is expressed as an ILP and solved independently for each demand and chain.
Variables:
• $a_{v,p}^{i}, a_{v,b}^{i} \in \{0,1\}$, where $a_{v,p}^{i} = 1$ ($a_{v,b}^{i} = 1$) if f_{i+1}^{c} is installed on node v in the primary (backup) path.
• $\varphi_{\ell,p}^{i}, \varphi_{\ell,b}^{i} \in \{0,1\}$, where $\varphi_{\ell,p}^{i} = 1$ ($\varphi_{\ell,b}^{i} = 1$) if the flow is forwarded on link ℓ on layer i in the primary (backup) path.
• $\gamma_{\ell,\omega} \in \{0,1\}$, where $\gamma_{\ell,\omega} = 1$ if the primary path needs to switch to the backup path in the failure situation ω and the backup path uses link ℓ.

$$\min \quad -u^{(5.19)} + \sum_{\ell\in L}\sum_{\omega\in\Omega} u_{\ell,\omega}^{(5.20)} \Big(\sum_{i=0}^{n_c-1} \varphi_{\ell,p}^{i} + \gamma_{\ell,\omega}\Big) \\ + D_{sd}^{c} \sum_{v\in V} u_{v}^{(5.21)} \sum_{i=0}^{n_c-1} \Delta_f (a_{v,p}^{i} + a_{v,b}^{i}) \tag{5.22}$$

Service Chain	Chained VNFs	% traffic
Web Service	NAT-FW-TM-WOC-IDPS	18.2%
VoIP	NAT-FW-TM-FW-NAT	11.8%
Video Streaming	NAT-FW-TM-VOC-IDPS	69.9%
Online Gaming	NAT-FW-VOC-WOC-IDPS	0.1%

Table 5.2: Service Chain Requirements [STV]

5.5 Experimental Study

In this section, we evaluate the performance of the four proposed models. We compare the time performance of the ILP models with their respective decomposition models. Moreover, we evaluate the trade-off between an efficient allocation of primary paths bandwidth and the total amount of required bandwidth needed to guarantee the protection.

5.5.1 Data Sets

We conduct experiments on three network topologies from SNDlib [Orl+10b]: `pdh` (11 nodes, 34 links), `geant` (22 nodes, 36 links) and `germany50` (50 nodes, 88 links). The number of requests varies according to the network size. All experiments are run on an Intel Xeon E5520 with 24GB of RAM. We consider 200 requests for `pdh`, 400 for `geant`, and 1000 for `germany50`. Network load is the same for all the networks and is set to 1 TB of data. The network traffic is divided into four common categories of traffic: Web Services, VoIP, Video Streaming and Online Gaming. Each traffic category is associated with a service function chain of five network functions. The traffic loads and the associated chains are given in Table 6.1.
For each network, we limit the nodes able to host VNFs. We consider different numbers and study the impact of the design choice on the delay and the required bandwidth. Nodes able to host VNFs are chosen according to their *betweenness centrality*, defined as the number of paths going through the node when considering the shortest paths between all pairs of nodes. It measures the relative importance of a node in a graph.

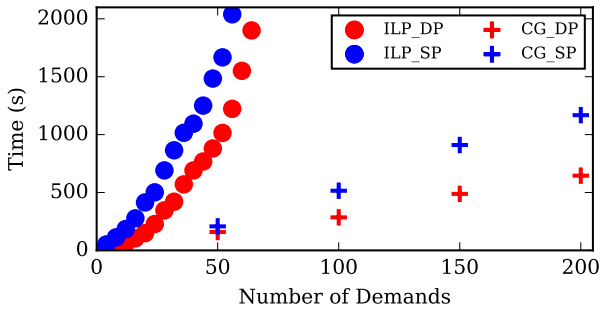


Figure 5.1: Execution time of the 4 models on the `pdh` network.

5.5.2 Compact ILPs vs. CG Models

In Figure 5.1, we compare the compact ILPs vs. the CG model for both dedicated and shared protection, on the `pdh` network. All nodes are assumed VNF enabled, and we consider an increasing number of demands from 4 to 200. The figure demonstrates the limits regarding the computing time of a compact ILP model. For 60 demands, the time needed to find an exact solution with the compact ILP model exceeds 30 min for shared protection and 25 min for dedicated protection. Hence, the compact ILP models are not suitable for large instances due to their limited scalability.
On the other hand, these results indicate that the decomposition models are fairly efficient. For 50 demands, 3 min are enough for both protection schemes. With larger values, the difference between the two decomposition models can be clearly seen. Indeed, for 200 demands `CG_DP` requires 11 min, while `CG_SP` takes 20 min, almost twice the time. This is due to the fact that the models for shared protection are more complex. Indeed, all failure scenarios have to be considered in the model, in order to share the backup bandwidth when possible.

5.5.3 Performance of CG Models

Table 9.2 summarizes the results of the decomposition models for dedicated and shared protection. We present the results for 3 different values of the number of VNF enabled nodes. Each traffic instance corresponds to 1 TB of traffic.
We provide the number of generated columns, the value of the ILP solution

Network	# traffic requests	# VNF nodes	# generated columns		$\tilde{z}_{\text{ILP}}$		ϵ	
			CG_DP	CG_SP	CG_DP	CG_SP	CG_DP	CG_SP
pdh	200	2	501	790	4,400	3,030.17	0	1.6×10^{-2}
		3	438	778	4,080	2,694.11	0	2.1×10^{-2}
		4	413	751	3,680	2,328.67	0	3.2×10^{-2}
geant	400	3	1,185	1,016	7,190	5,141.88	0	3.8×10^{-5}
		5	1,094	1,040	6,650	4,844.69	0	8.2×10^{-4}
		7	1,076	1,034	6,390	4,651.37	0	8.6×10^{-4}
germany50	1000	5	3,654	2,701	9,270	7,253.11	0	2.4×10^{-4}
		10	3,338	2,771	9,188	6,563.46	0	4.6×10^{-3}
		15	3,165	2,674	8,800	6,198.77	0	1.7×10^{-6}

Table 5.3: Numerical results for CG_DP and CG_SP

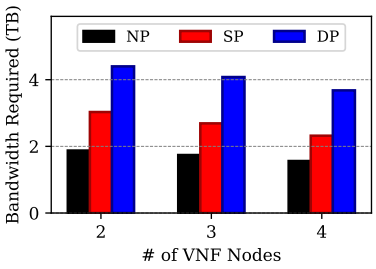
($\tilde{z}_{\text{ILP}}$) and the accuracy ε, defined as the ratio $(\tilde{z}_{\text{ILP}} - z_{\text{LP}})/z_{\text{LP}}$. For most of the instances, the value of the ILP solution coincides with the value of the linear relaxation (z_{LP}). In any case, the solution accuracy never exceeds 4%. The number of generated columns is similar in the two models. However, there is a fundamental difference in terms of complexity. We recall that, in the `CG_DP` model, a column corresponds to a service path, and that, in the master problem, we look for a pair of service paths for each demand. Conversely, in the `CG_SP` model, a column corresponds to a pair of service paths and then, for each demand, only one column is selected. Hence, the problem is very hard in the shared protection case with respect to the dedicated one.

Network	DP	SP
pdh	2	1.26
geant	2	1.28
Germany	2	1.38

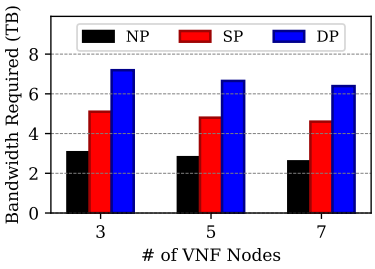
Table 5.4: Average ratio between the processing requirements of dedicated and shared protection over the processing requirements without protection.

5.5.4 Bandwidth and Processing Requirements

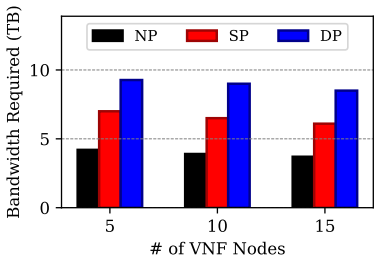
As expected, there is a relationship between the number of VNF nodes and the bandwidth needed for both the primary paths and the global protection. Indeed, a larger number of VNF nodes allows more flexibility to find shorter



(a) pdh



(b) geant



(c) germany50

Figure 5.2: Bandwidth requirements: no protection (NP) scheme vs. dedicated (DP) and shared (SP) protection schemes

paths.

VNFs nodes are expensive for both purchase and maintenance (e.g., hardware, software licenses, energy consumption, and maintenance). Thus, it is necessary to find the right trade-off between bandwidth and number of VNF nodes. For example, in `pdh`, using two nodes instead of four leads to an increase in the total required bandwidth of about 20% in the dedicated protection case and 30% in the shared protection case. Similar results are observed for the other networks. Even if the solution computed by the two protection schemes in terms of bandwidth used by primary paths is almost the same, there is a noticeable difference in terms of total bandwidth requirements. `CG_DP` requires on average about 40% more bandwidth than `CG_SP`. Similar results are found for the processing requirements. About 60% more processing units are required by the dedicated protection scheme than the shared one.

In Figure 5.2, we show the bandwidth requirements for the 3 networks without any protection strategy compared with the bandwidth requirements of

the dedicated and shared protection schemes. In the case of Dedicated Protection, we may need up to 3 times more bandwidth than the one needed if we do not consider protection. In the case of Shared Protection, the price to pay is less than twice. Note that we put a limit to the latency of the paths in order not to violate the SLA requirements. Hence, we expect the savings opportunities of the shared protection to be even larger in the general case.

5.5.5 Delay

In Figure 5.3 we show the delay of the primary and backup paths for the three networks in the case of dedicated and shared protection. In order to compute the link delays, we used the distances given by the geographical coordinates provided in SNDlib.
The delays of the primary paths tend to be close between the two different protection schemes with a maximum delay of 7.2, 9 and 18 ms for `pdh`, `geant`, and `germany50` respectively. The delay distributions of the primary paths slightly change when varying the number of allowed VNF nodes and tend to be homogeneous among them.
However, this is not true for the backup paths. In the dedicated protection case, paths are interested in using shorter paths in order to minimize the bandwidth requirements. In the shared protection case, this is not true. In fact, backup paths may find convenient to increase their lengths in order to share as much as possible and, thus, reduce the bandwidth requirements. This can be observed in the results. For example, the delay for a backup path in the dedicated protection case for `germany50` never exceeds 20 ms while it may go up to 40 ms in the shared protection case. Hence, particular attention should be paid to paths' latencies when considering shared path protection.

5.6 Conclusion

In this chapter, we provided exact methods to obtain *reliable Service Function Chains* against a single-link failure. We considered two different protection schemes, dedicated and shared path protection, providing for each of them a scalable decomposition ILP model. The models are very general and can be easily extended to the case of node-disjoint paths or to deal with multiple failures. We showed the limits of the ILP based approaches and the time efficiency of the decomposition models.

We implemented and evaluated the models on 3 network topologies with different sizes, studying the bandwidth requirements for the protection, as well as their latency robustness. We also studied the trade-off between the network bandwidth requirement to guarantee the protection and the number of VNF capable nodes.

References

[AA99] Murat Alanyali and Ender Ayanoglu. "Provisioning algorithms for WDM optical networks". In: *IEEE/ACM Transactions On Networking* 7.5 (1999) (cit. on p. 109).

[ADP80] Giorgio Ausiello, Alessandro D'Atri, and Marco Protasi. "Structure preserving reductions among convex optimization problems". In: *Journal of Computer and System Sciences* 21.1 (1980), pp. 136–153 (cit. on pp. 48, 85).

[AV14] YK Agarwal and Prahalad Venkateshan. "Survivable network design with shared-protection routing". In: *European Journal of Operational Research* 238.3 (2014), pp. 836–845 (cit. on p. 110).

[BBS16] Michael Till Beck, Juan Felipe Botero, and Kai Samelin. "Resilient allocation of service Function chains". In: *Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE Conference on*. IEEE. 2016 (cit. on p. 110).

[Ben62] Jacques F Benders. "Partitioning procedures for solving mixed-variables programming problems". In: *Numerische mathematik* 4.1 (1962), pp. 238–252 (cit. on pp. 89, 97).

[Ber+14] Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O'Connor, Pavlin Radoslavov, William Snow, et al. "ONOS: towards an open, distributed SDN OS". In: *Proceedings of the third workshop on Hot topics in software defined networking*. ACM. 2014, pp. 1–6 (cit. on pp. 4, 80).

[Cas+17] Marco Casazza, Pierre Fouilhoux, Mathieu Bouet, and Stefano Secci. "Securing virtual network function placement with high availability guarantees". In: *2017 IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE. 2017, pp. 1–9 (cit. on p. 110).

[Chu+] Cing-Yu Chu, Kang Xi, Min Luo, and H Jonathan Chao. "Congestion-aware single link failure recovery in hybrid SDN networks". In: *Proceedings of IEEE INFOCOM, 2015* (cit. on p. 82).

[Chv83] V. Chvatal. *Linear Programming*. Freeman, 1983 (cit. on pp. 88, 148).

[Cis] Cisco. *Bidirectional Forwarding Detection – Cisco*. https://www.cisco.com/c/en/us/td/docs/ios/12_0s/feature/guide/fs_bfd.html. Accessed: 2018-07 (cit. on p. 105).

[DS] Irit Dinur and David Steurer. "Analytical Approach to Parallel Repetition". In: *Proceedings ACM STOC 2014*. New York, New York. ISBN: 978-1-4503-2710-7 (cit. on pp. 44, 85).

[FM17] Paulo Fonseca and Edjard Mota. "A survey on fault management in software-defined networks". In: *IEEE Communications Surveys & Tutorials* (2017) (cit. on p. 82).

[FT02] Bernard Fortz and Mikkel Thorup. "Optimizing OSPF/IS-IS weights in a changing world". In: *IEEE journal on selected areas in communications* 20.4 (2002), pp. 756–767 (cit. on p. 96).

[FV00] Andrea Fumagalli and Luca Valcarenghi. "IP restoration vs. WDM protection: Is there an optimal choice?" In: *IEEE network* 14.6 (2000) (cit. on p. 82).

[GJN11] P. Gill, N. Jain, and N. Nagappan. "Understanding network failures in data centers: measurement, analysis, and implications". In: *ACM SIGCOMM Computer Communication Review*. Vol. 41. 4. 2011 (cit. on p. 107).

[Hras01] Johan Hrastad. "Some optimal inapproximability results". In: *Journal of the ACM (JACM)* 48.4 (2001), pp. 798–859 (cit. on p. 92).

[HJG17a] Nicolas Huin, Brigitte Jaumard, and Frédéric Giroire. "Optimization of Network Service Chain Provisioning". In: *IEEE International Conference on Communications 2017*. Paris, France, 2017 (cit. on pp. 110, 111).

[HJG18b] Nicolas Huin, Brigitte Jaumard, and Frédéric Giroire. "Optimal Network Service Chain Provisioning". In: *IEEE/ACM Transactions on Networking* (2018) (cit. on pp. 85, 96).

[Hma+17] Ali Hmaity, Marco Savi, Francesco Musumeci, Massimo Tornatore, and Achille Pattavina. "Protection strategies for virtual network functions placement and service chains provisioning". In: *Networks* (2017), pp. 1–15 (cit. on pp. 110, 111).

[Hu03] Jian Qiang Hu. "Diverse routing in optical mesh networks". In: *IEEE Transactions on Communications* 51.3 (2003), pp. 489–494 (cit. on p. 80).

[Hui+18c] Nicolas Huin, Andrea Tomassilli, Frédéric Giroire, and Brigitte Jaumard. "Energy-Efficient Service Function Chain Provisioning". In: *IEEE/OSA Journal of Optical Communications and Networking* 10.2 (2018) (cit. on p. 110).

[IMG98] R.R. Iraschko, M.H. MacGregor, and W.D. Grover. "Optimal capacity placement for path restoration in STM or ATM mesh-survivable networks". In: *IEEE/ACM Transactions on Networking* 6.3 (1998) (cit. on p. 108).

[KCG] Amund Kvalbein, Tarik Cicic, and Stein Gjessing. "Post-failure routing performance with multiple routing configurations". In: *Proceedings of IEEE INFOCOM, 2007* (cit. on pp. 82, 83).

[Kem+12] James Kempf, Elisa Bellagamba, András Kern, David Jocha, Attila Takács, and Pontus Sköldström. "Scalable fault management for OpenFlow". In: *Communications (ICC), 2012 IEEE international conference on*. IEEE. 2012, pp. 6606–6610 (cit. on p. 80).

[KKV05] Srikanth Kandula, Dina Katabi, and Jean-Philippe Vasseur. "Shrink: A tool for failure diagnosis in IP networks". In: *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*. ACM. 2005, pp. 173–178 (cit. on pp. 80, 95).

[Kva+] Amund Kvalbein, Audun Fosselie Hansen, Stein Gjessing, and Olav Lysne. "Fast IP network recovery using multiple routing configurations". In: *Proceedings of IEEE INFOCOM, 2006* (cit. on p. 83).

[LHM10] Bob Lantz, Brandon Heller, and Nick McKeown. "A Network in a Laptop: Rapid Prototyping for Software-defined Networks". In: *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. Hotnets-IX. Monterey, California: ACM, 2010, 19:1–19:6. ISBN: 978-1-4503-0409-2. DOI: 10.1145/1868447.1868466. URL: http://doi.acm.org/10.1145/1868447.1868466 (cit. on pp. 101, 103).

[LM15] S.-I. Lee and Myung M.-K. Shin. "A self-recovery scheme for service function chaining". In: *International Conference on Information and Communication Technology Convergence (ICTC)*. 2015, pp. 108–112 (cit. on p. 110).

[Med+] J. Medved, R. Varga, A. Tkacik, and K. Gray. "OpenDaylight: Towards a Model-Driven SDN Controller architecture". In: *Proceedings of IEEE WoWMoM 2014* (cit. on p. 103).

[Niv+09] B Niven-Jenkins, D Brungard, M Betts, N Sprecher, and S Ueno. *Requirements of an MPLS transport profile*. Tech. rep. 2009 (cit. on p. 80).

[Orl+10b] Sebastian Orlowski, Roland Wessäly, Michal Pióro, and Artur Tomaszewski. "SNDlib 1.0—Survivable network design library". In: *Networks* 55.3 (2010) (cit. on pp. 68, 95, 119, 213).

[Pes+18] Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. "Automation and combination of linear-programming based stabilization techniques in column generation". In: *INFORMS Journal on Computing* (2018) (cit. on pp. 30, 89).

[PJ13] R. Potharaju and N. Jain. "Demystifying the dark side of the middle: a field study of middlebox failures in datacenters". In: *Internet Measurement Conference*. 2013, pp. 9–22 (cit. on p. 107).

[PM04] Michal Pióro and Deep Medhi. *Routing, flow, and capacity design in communication and computer networks*. Elsevier, 2004 (cit. on pp. 12, 80).

[RM99] S. Ramamurthy and B. Mukherjee. "Survivable WDM mesh networks. Part I - protection". In: *Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM*. Vol. 2. 1999, pp. 744–751 (cit. on pp. 108, 109).

[RMD05] Smita Rai, Biswanath Mukherjee, and Omkar Deshpande. "IP resilience within an autonomous system: current approaches, challenges, and future directions". In: *IEEE Communications Magazine* 43.10 (2005), pp. 142–149 (cit. on p. 82).

[Sga+13] Andrea Sgambelluri, Alessio Giorgetti, Filippo Cugini, Francesco Paolucci, and Piero Castoldi. "OpenFlow-based segment protection in Ethernet networks". In: *Journal of Optical Communications and Networking* 5.9 (2013) (cit. on pp. 82, 108).

[Sha+11] Sachin Sharma, Dimitri Staessens, Didier Colle, Mario Pickavet, and Piet Demeester. "Enabling fast failure recovery in OpenFlow networks". In: *2011 8th International Workshop on the Design of Reliable Communication Networks (DRCN 2011)*. IEEE. 2011, pp. 164–171 (cit. on p. 108).

[Sha+13] Sachin Sharma, Dimitri Staessens, Didier Colle, Mario Pickavet, and Piet Demeester. "OpenFlow: Meeting carrier-grade recovery requirements". In: *Computer Communications* 36.6 (2013), pp. 656–665 (cit. on pp. 80, 108).

[Sou+17] O. Soualah, Marouen Mechtri, Chaima Ghribi, and Djamal Zeghlache. "A link failure recovery algorithm for Virtual Network Function chaining". In: *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. 2017 (cit. on p. 110).

[SRM02] Laxman Sahasrabuddhe, Senthil Ramamurthy, and Biswanath Mukherjee. "Fault management in IP-over-WDM networks: WDM protection versus IP restoration". In: *IEEE journal on selected areas in communications* 20.1 (2002), pp. 21–33 (cit. on pp. 108, 204, 206).

[Sta+11] Dimitri Staessens, Sachin Sharma, Didier Colle, Mario Pickavet, and Piet Demeester. "Software defined networking: Meeting carrier grade requirements". In: *18th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN)*. IEEE. 2011 (cit. on p. 108).

[Sti+07] Thomas Stidsen, Bjørn Petersen, Kasper Bonne Rasmussen, Simon Spoorendonk, Martin Zachariasen, Franz Rambach, and Moritz Kiese. "Optimal routing with single backup path protection". In: *International Network Optimization Conference (INOC)*. 2007 (cit. on p. 117).

[STV] Marco Savi, Massimo Tornatore, and Giacomo Verticale. "Impact of processing costs on service chain placement in network functions virtualization". In: *IEEE NFV-SDN 2015* (cit. on pp. 9, 49, 119).

[Suc+] Martin Suchara, Dahai Xu, Robert Doverspike, David Johnson, and Jennifer Rexford. "Network architecture for joint failure recovery and traffic engineering". In: *Proceedings of ACM SIGMETRICS 2011* (cit. on p. 82).

[Tom+18c] Andrea Tomassilli, Nicolas Huin, Frederic Giroire, and Brigitte Jaumard. "Resource requirements for reliable service function chaining". In: *2018 IEEE International Conference on Communications (ICC)*. IEEE. 2018, pp. 1–7 (cit. on pp. 11, 107).

[Tur+10] Daniel Turner, Kirill Levchenko, Alex C Snoeren, and Stefan Savage. "California fault lines: understanding the causes and impact of network failures". In: *ACM SIGCOMM Computer Communication Review*. Vol. 40. 4. ACM. 2010, pp. 315–326 (cit. on pp. 107, 108, 204).

[VVK14] Niels LM Van Adrichem, Benjamin J Van Asten, and Fernando A Kuipers. "Fast recovery in software-defined networks". In: *Software Defined Networks (EWSDN), 2014 Third European Workshop on*. IEEE. 2014, pp. 61–66 (cit. on p. 80).

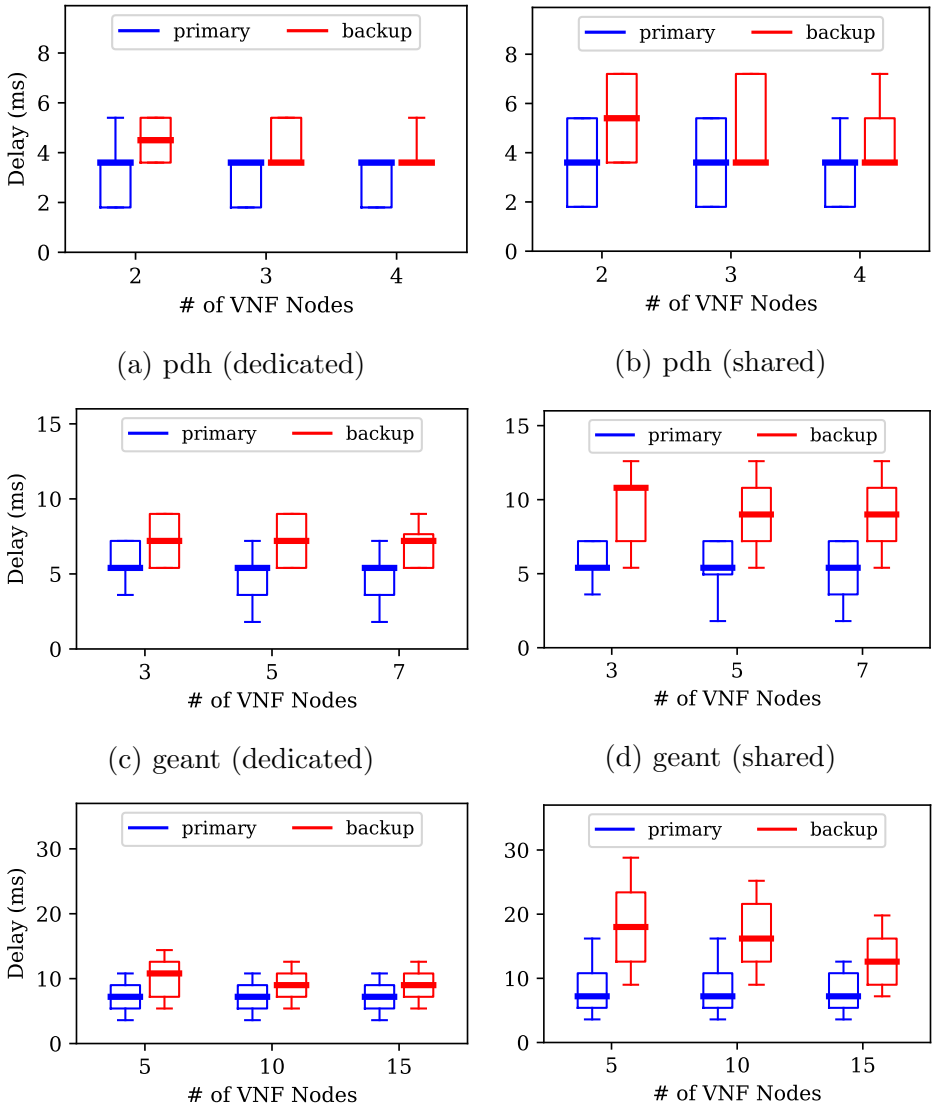
[Wax88] Bernard M Waxman. "Routing of multipoint connections". In: *IEEE journal on selected areas in communications* 6.9 (1988), pp. 1617–1622 (cit. on p. 96).

[Xu+04] Dahai Xu, Yizhi Xiong, Chunming Qiao, and Guangzhi Li. "Failure protection in layered networks with shared risk link groups". In: *IEEE network* (2004) (cit. on p. 82).

[Ye+16] Zilong Ye, Xiaojun Cao, Jianping Wang, Hongfang Yu, and Chunming Qiao. "Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization". In: *IEEE Network* 30.3 (2016) (cit. on p. 110).

[Zha+18] Yuan Zhang, Lin Cui, Wei Wang, and Yuxiang Zhang. "A Survey on Software Defined Networking with Multiple Controllers". In: *J. Netw. Comput. Appl.* 103.C (2018), pp. 101–118. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2017.11.015 (cit. on p. 105).

[ZS00] Dongyun Zhou and Suresh Subramaniam. "Survivability in optical networks". In: *IEEE network* 14.6 (2000), pp. 16–23 (cit. on pp. 108, 204, 206).



(a) pdh (dedicated)

(b) pdh (shared)

(c) geant (dedicated)

(d) geant (shared)

(e) germany50 (dedicated)

(f) germany50 (shared)

Figure 5.3: Primary and backup path delay distributions under the two protection schemes vs. the number of VNF nodes with 1TB offered load. Boxes are defined by the first and third quartiles. Ends of the whiskers correspond to the first and ninth deciles.

Part III

Energy Aware Routing

Chapter 6

Energy Efficient Service Function Chains

Contents

The content of this chapter has been published before in [Hui+18a].

6.1 Introduction

With the large yearly increase of Internet traffic and the growing concern of the public and governments towards greenhouse gas emissions, future networks will have to be more energy efficient [Mat+13]. In the past few years, this has been the focus of extensive research work [Ver+11; Le +13]. One of the classic methods to reduce the energy consumption of networks is to try to aggregate network traffic on a small number of network equipment in order to put to sleep the unused hardware. However, an additional challenge is given by the fact that today's traffic must pass through a certain number of network functions. Examples of network functions include deep packet inspection (DPI), firewall, load balancing, and WAN optimization. The network functions often need to be applied in a specific order, e.g., in a security scenario, the firewall has to be applied before carrying out a DPI, as the latter is more CPU intensive than the former. In this context, a *Service Function Chain (SFC)* is a list of network functions, that need to be applied to a flow in a particular order. These functions are carried out by specific hardware, which are installed at specific locations of the network. The paths followed by demands are thus very constrained, reducing the opportunities to aggregate traffic.

With the emergence of techniques of Network Function Virtualization (NFV), the functions can now be executed by generic hardware instead of dedicated equipment. Coupled with the Software Defined Network (SDN) paradigm, NFV brings a great flexibility to manage network flows. Indeed, with the centralized control allowed by SDN, the flow can be managed dynamically from end-to-end and the service functions can be installed only along paths for which and when they are necessary. These new paradigms thus bear the opportunity for energy savings in networks.

In this work, we explore the potential energy savings of using NFV for Service Function Chains. We consider the problem of reducing network energy consumption while placing service functions using generic hardware along

the paths followed by flows. A difficulty is that the network functions have to be executed in a specific order and can be repeated several time in the same chain.
In summary, the contributions of this work are the following

- We show how virtualization can be used to improve the energy efficiency of networks, when demands have to go through a chain of services. To the best of our knowledge, we are the first to propose such a method.

- We propose a way of modeling this problem based on Integer Linear Programming (ILP). The ILP can solve optimally instances of small sizes.

- To handle instances of larger sizes, we thus propose and validate a *heuristic algorithm*, GreenChains, and we formulate a Column Generation model to solve the EE-SFCP problem on large instances.

- We provide enhancements of the model with the use of cuts, as the our problem is a difficult optimization problem. As a matter of fact, it contains a sharp On-Off phenomena, as a network device consumes a large portion of its energy as soon as it is used, even if very lightly used. Cuts allow the reduction of the integrality gap.

- This allows us to carry out extensive simulations on networks of different sizes. We study three different scenarios: a *legacy scenario* which serves as baseline for comparison, a *hardware scenario* in which the routing can be changed dynamically by a centralized SDN controller, but in which network functions are executed by specific hardware, and finally, an *NFV scenario* in which the network functions are virtualized and can be placed dynamically. We show that from 22% to 62% of energy can be saved during the night while respecting the constraints of the service chains.

- Finally, we propose a latency analysis to evaluate the impact on delays of switching off some network elements to save energy.

The article is organized as follows. In Section 6.2, we review the current works on energy efficiency and Service Function Chaining. The problem is presented in Section 6.3 along with the power model and the layered graph model used in our mathematical formulations. We present in Sections 6.4, 6.5

and 6.6 the ILP formulation, GreenChains and column generation scheme, respectively. We then compare the models and assess their quality in Section 6.7.

6.2 Related Work

6.2.1 Service Chains

Several works study the problem of service function chain placement, but taking other metrics or other scenarios into account. Savi *et al.* [STV15] proposes a different ILP model to solve the problem. They study the impact of the positions of the network functions on the processing costs. Gupta *et al.* [Gup+15; Gup+17] explores the joint placement and routing of traffic in order to minimize the network bandwidth consumption. In Martini *et al.* [Mar+15], a layered ILP model close to the one we propose in the paper is proposed, but with latency minimization as optimization task. [Moh+15] explores the problem of joint optimization of maximum link, CPU core and maximum delay in the network while placing the VNFs. Last, Riggio *et al.* [Rig+15] considers a cloud environment in which the load has to be load-balanced in order to minimize the computation and the communication overheads. However, these works do not consider the problem of minimizing network energy consumption with a dynamic traffic.

Energy-Aware Routing. Several works have proposed algorithms to obtain energy aware routing, see e.g., Chiaraviglio *et al.* [CMN12]. However, these works are hard to be put in practice as operators of legacy networks are reluctant to change their network configurations.

6.2.2 SDN and Network Energy Efficiency

Since the pioneering work of Gupta et al. [GS03], a lot of researchers have considered the problem of energy efficiency of networks, see, e.g., [LMF11; Pha14; Gir+10] for backbone networks, [SLX10] for data center networks, [Mod+13] for content distribution, and [DGF10] for wireless networks. We refer to [Bol+10] for a comprehensive survey.

Recently, researchers have started to explore how the introduction of the SDN paradigm with a centralized control and a live report of metrology data may enable dynamic routing. In particular, it would allow the imple-

mentation of energy-aware routing algorithms, as discussed in Giroire *et al.* [GMP14] and later works [Hui+18b]. However, these papers did not consider the constraints of network functions. Some particular works considered some specific class of network functions, like compression [Gir+15], but not the general problem of ensuring that flows are treated by the network functions.

6.2.3 Network Virtualization and Network Energy Efficiency

Only two papers explore the potential of network virtualization for energy efficiency. In Bolla *et al.* [Bol+14], the authors present an extension of an open source software framework, the Distributed Router Open Platform (DROP), to enable a novel distributed paradigm for NFV. DROP includes sophisticated power management mechanisms, which are exposed by means of the Green Abstraction Layer. In [Mij15], authors estimate the energy savings that could result from the three main NFV use cases-Virtualized Evolved Packet Core, Virtualized Customer Premises Equipment and Virtualized Radio Access Network. However, both papers do not consider the constraints of service chains.

6.3 Statement of the Problem: SFC and VNF Placement

6.3.1 Notations.

We assume the network to be represented by a directed graph $G = (V, L)$, where V is the set of nodes (indexed by v), and L is the set of links (indexed by ℓ). Each node $u \in V$ has a set of computing, storage and network resources denoted by C_u to host network functions. Within this study, we assume that the resources are described by a given number of CPU cores.

Traffic is described by a set of requests D, in which each request d is defined by a 4-tuple (v_s, v_d, c, D^c_{sd}), where v_s is the source of the request, v_d its destination, D^c_{sd} its bandwidth requirement, and c the requested service chain. Indeed, each request d is associated with a given application, which is required to pass through a given SFC. Let F be the overall set of virtual

functions arising in the service chains, indexed by f, and C be the set of service chains, indexed by c. Each service chain c corresponds to a sequence of n_c functions $f_1^c, \ldots, f_i^c, \ldots, f_{n_c}^c$, where f_i^c denotes the ith function of chain c. Note that some functions may appear more than once in a given chain. Each virtual function f has its one resource requirement, and we denote by Δ_f the number (fraction) of cores required by the function f per bandwidth unit.

The *Energy Efficient Service Function Chain Provisioning (*EE-SFCP*)* problem consists in jointly provisioning a set D of requests coupled with service function chains C and placing virtual functions arising in the chains, in order to minimize the network energy consumption, subject to link and node capacities. Figure 6.1 shows examples of a Energy Efficient Service Function Chain provisioning problem. We have three requests and two types of services. Demand $D_1 = (A \rightsquigarrow D)$ requires 1 unit of bandwidth and the execution of a firewall (FW) and a packet inspector (IDPS). Flows $D_2 = (H \rightsquigarrow K)$ and $D_3 = (G \rightsquigarrow F)$ need 1 unit of bandwidth, and the execution of a firewall followed by a video optimizer (VOC). An instance of a firewall uses $\Delta_{\text{FW}} = 0.33$ core per unit of bandwidth, an instance of a video optimizer requires $\Delta_{\text{VOC}} = 1$ core, and an instance of a packet inspector uses $\Delta_{\text{FW}} = 2$ cores. Each link has a capacity of 3 units of bandwidth, and each node hosts 2 cores.

In Figure 6.1a, flows are routed according to the shortest path between their source and destination. We need to place a firewall function instance on three different nodes (namely A, G, and H) to cover the three demands. Flows D_2 and D_3 get their video optimizer on nodes F and I, respectively. The packet inspector of flow D_1 is installed on node B. In this last configuration, five links can be shut down ((A, E), (H, E), (E, F), (G, D), (G, K)) and node E can be put to sleep. A total of 7 cores are active (1 on nodes A, F, G, H, and I, and two on node B).

However, there exists another routing that minimizes the energy consumed by the network. It allows the shutdown of one more link and the reduction of the number of active cores by two units. Indeed, if we consider the routing given in Figure 6.1b, we can group all the firewall instances on node F. Since nodes only host 2 cores, we need to put one video optimizer instance on node F, in charge of D_2; D_3 is served by the instance on node G. We now only use 5 cores in total, and we can now shutdown links (A, B), (B, C), (C, D), (H, I), (I, J), and (J, K).

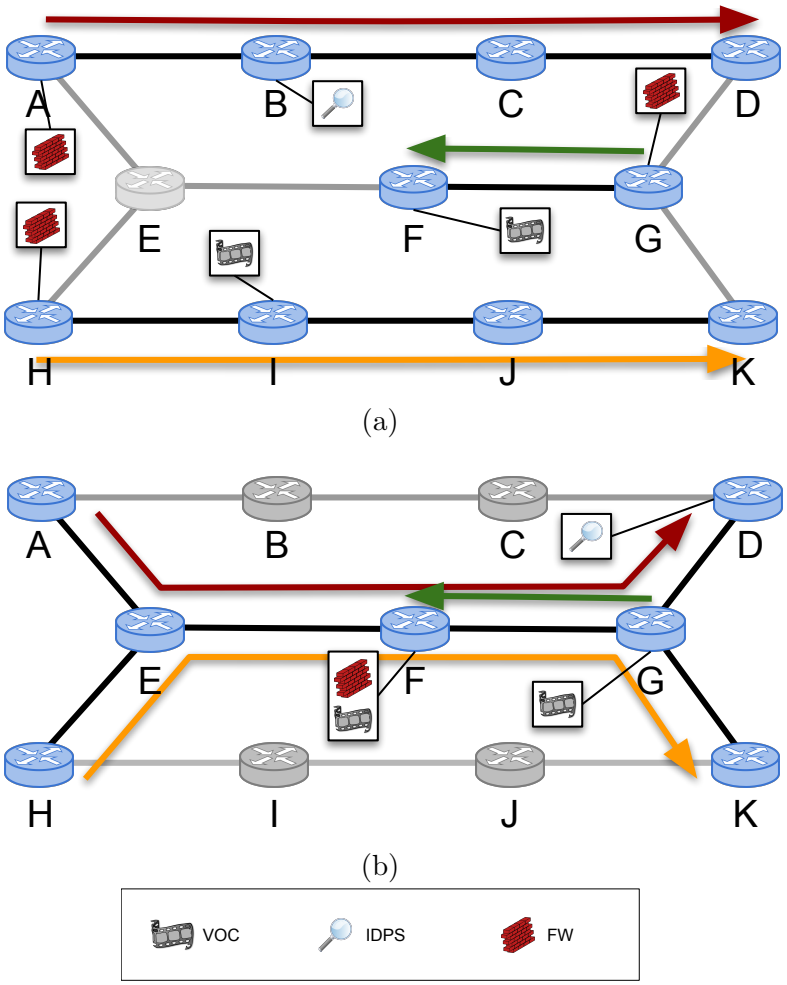


Figure 6.1: Example of Energy efficient Service Function Placement. Greyed links and nodes are inactive.

6.3.2 Power Model

Campaigns of measures of power consumption (see, e.g., [Cha+08]) show that a network device consumes a large amount of its power as soon as it is switched on and that the energy consumption does not depend much on the load. Following this observation, on/off power models have been proposed and studied. Later, researchers and hardware constructors have proposed more energy proportional hardware models [Nic+12]. To encompass those different models, we use a hybrid power model in which the power of an active link ℓ is expressed as

$$P_\ell = P_\ell^{\text{ON}} + \frac{\text{BW}_\ell}{C_\ell^{\text{LINK}}} P_\ell^{\text{MAX}},$$

where P_ℓ^{ON} represents the energy used when the link ℓ is switched on, BW_ℓ the bandwidth that is carried on ℓ, and P_ℓ^{MAX} the additional energy consumed by ℓ when it is fully capacitated, i.e., when the amount of carried bandwidth equals the transport capacity (C_ℓ^{LINK}) of link ℓ.
We assume that links can be put into sleep mode, by putting to sleep both endpoint interfaces. Two links in opposite direction between a pair of nodes are assumed to be in the same state (active or in sleep mode), as the send and receive elements of a unidirectional fiber are usually controlled by the same interface. Routers cannot be put into sleep mode, as there are the sources/destinations of network traffic. However, cores may be put into sleep mode and the power used by node v is equal to

$$P_v = P_v^{\text{UNIT}} \times \#\text{cores}$$

with P_v^{UNIT} being the energy consumption of a single core.

6.3.3 Layered Graph.

As in the previous chapters, to model the ordering constraints we use a layered graph G^{L} that is defined as follows. We add $\max_{c \in C} n_c$ layers to the original graph G and each layer contains a copy of G. For every node $u \in V$, let v^i be the corresponding node in the ith layer ($i = 0, 1, \ldots, n_c$). Every $(i-1, i)$ layer pair is connected by (v^{i-1}, v^i) links. Provisioning of a chain and node placement of its functions amounts to find a path from node v_s on the first layer, i.e. v_s^0, to node v_d on the n_cth layer, i.e., $v_d^{n_c}$. Placement of a function on a node is given by the endpoints of the link used to switch between layers.

6.4 Compact formulation

We now present the ILP formulation for the EE-SFCP problem. Let us first introduce the set of variables.

- $x_\ell \in \{0,1\}$ where $x_\ell = 1$ if link ℓ is active, 0 otherwise
- $f_{i\ell}^{sd,c} \in \{0,1\}$ where $f_{i\ell}^{sd,c} = 1$ if the flow for the request (v_s, v_d, c, D_{sd}^c) uses the link ℓ in layer i. We consider here un-splittable routing.
- $a_{iv}^{sd,c} \in \{0,1\}$ where $a_{iv}^{sd,c} = 1$ if the ith function of the chain c is executed on node v for the request (v_s, v_d, c, D_{sd}^c).
- $k_v \in \mathbb{N}$, number of CPU cores used in node v.
- $f_\ell \in \mathbb{R}$, flow passing through link (u, v). This variable is linked and is added to the ILP for clarity of the presentation.

The formulation is given as follows.
Objective

$$\min \sum_{\ell \in L} \left(P_\ell^{\text{ON}} \times x_\ell + P_\ell^{\text{MAX}} \times \frac{f_\ell}{C_\ell^{\text{LINK}}} \right) + \sum_{u \in V} P_v k_u \tag{6.1}$$

Flow Conservation

$$\sum_{\ell \in \omega^+(v)} f_{i\ell}^{sd,c} - \sum_{\ell \in \omega^-(v)} f_{i\ell}^{sd,c} + a_{iv}^{sd,c} - a_{i-1v}^{sd,c} = 0$$
$$(v_s, v_d) \in \mathcal{SD}, c \in C_{sd}, u \in V, 0 < i < n_c \tag{6.2}$$

$$\sum_{\ell \in \omega^+(v)} f_{0\ell}^{sd,c} - \sum_{\ell \in \omega^-(v)} f_{0\ell}^{sd,c} + a_{0v}^{sd,c} = \begin{cases} 1 \text{ if } u = v_s, \\ 0 \text{ else} \end{cases}$$
$$(v_s, v_d) \in \mathcal{SD}, c \in C_{sd}, u \in V \tag{6.3}$$

$$\sum_{\ell \in \omega^+(v)} f_{n_c\ell}^{sd,c} - \sum_{\ell \in \omega^-(v)} f_{n_c\ell}^{sd,c} - a_{n_c-1v}^{sd,c} = \begin{cases} -1 \text{ if } u = v_d, \\ 0 \quad \text{else} \end{cases}$$
$$(v_s, v_d) \in \mathcal{SD}, c \in C_{sd}, u \in V. \tag{6.4}$$

Link Capacity

$$f_\ell = \sum_{(v_s,v_d)\in\mathcal{SD}} \sum_{c\in C_{sd}} \sum_{i=0}^{n_c} D^c_{sd} \times f^{sd,c}_{i\ell} \leq C^{\text{LINK}}_\ell \times x_{uv} \qquad \ell \in A. \quad (6.5)$$

Number of CPU cores used

$$\sum_{(s,t)\in\mathcal{D}} \sum_{i=0}^{n_c-1} \left(\Delta_{f^c_i} D^c_{sd}\right) \times a^{sd,c}_{iv} \leq k_v \qquad u \in V. \quad (6.6)$$

Node Capacity

$$k_v \leq C^{\text{NODE}}_v \qquad u \in V. \quad (6.7)$$

6.5 Solving large Instances with GreenChains

As the ILP proposed in the previous section cannot provide solutions for large networks, we propose here an ILP-based heuristic algorithm called GreenChains to solve the

Problem 3. . The problem can be decomposed into three sub-problems.

- First, the *energy saving problem* tries to put into sleep mode as many links and cores as possible to decrease the energy consumption of the network.

- Second, the *routing problem* computes a path for each request, respecting the link capacity constraints.

- Last, the goal of the *service chain placement problem* is to find a placement of the NFV respecting the capacities of the nodes and the order defined by the service chains, according to the path computed for each request.

6.5.1 Energy Saving Module.

The goal of this module is to put links into sleep mode.

It first launches the routing module and then places the network functions on the requests' paths. If both modules succeed, it creates a list U of all links according to their usage (volume of traffic). It then chooses the less loaded link $\ell_{\min}$ as a candidate to be put in sleep mode. It now considers the graph $G' = (V, L \setminus \{\ell_{\min}\})$. It launches the routing and placement modules again. If they succeed, $\ell_{\min}$ is put in sleep mode. The list U is actualized with the new routing, as well as the less loaded link. If at least one of the two modules fails, GreenChains considers that $\ell_{\min}$ cannot be into sleep mode and the link is kept active for the final solution. The second element of U is then considered. The algorithm goes on till all links have been tried and set either as definitely in sleep mode or active. The goal of this module is to reduce the energy used by the links by putting in sleep mode as many links as possible.

6.5.2 Routing Module

We consider the requests one by one and compute a weighted shortest path on a residual graph for each one of them. To favor links with a lower load, the weight of the link in the residual graph is equal to the inverse of its residual capacity. When we assign a path to a request, we decrease the capacity of the residual graph by the amount of charge requested. Furthermore, when considering a new demand to be routed, we remove links with a residual capacity smaller than the demand.

6.5.3 Service Chain Placement Module.

This module is in charge of choosing the execution location of the chains functions. We propose the following ILP that aims at minimizing the total number of cores used.

Given a path $P_{sd,c}$ for every request (v_s, v_d, c, D^c_{sd}), we need to find the execution location of each function of the chain c. Each node of the path is indexed by i, i.e., $P^i_{sd,c}$ is the ith node of $P_{sd,c}$.

We introduce the following two sets of variables.

- $a^{sd,c}_{iv} \in \{0, 1\}$ where $a^{sd,c}_{i,v} = 1$ if f^c_i for request (v_s, v_d, c, D^c_{sd}) is executed on node v

- $k_v \in \mathbb{N}$, #cores used in node v.

The formulation is as follows.
Objective function

$$\min \sum_{u \in V} k_u. \tag{6.8}$$

Execution constraints

$$\sum_{v \in P_{sd,c}} a_{iv}^{sd,c} = 1 \quad (v_s, v_d) \in \mathcal{SD}, c \in C_{sd}, 1 \leq i \leq n_c. \tag{6.9}$$

Order constraints

$$a_{i,P_{sd,c}^k}^{sd,c} \leq \sum_{j=1}^{k} a_{i-1,P_{sd,c}^j}^{sd,c} \qquad (v_s, v_d) \in \mathcal{SD}, c \in C_{sd},$$
$$1 \leq k \leq \text{LEN}(P_{sd,c}), 1 \leq i \leq n_c. \tag{6.10}$$

Number of cores used

$$\sum_{(v_s,v_d) \in \mathcal{SD}} \sum_{c \in C} \sum_{i=1}^{n_c} \left(\Delta_{f_i^c} D_{sd}^c\right) \times a_{iv}^{sd,c} \leq k_v \qquad u \in V. \tag{6.11}$$

Node Capacity constraints

$$k_u \leq C_u \quad u \in V. \tag{6.12}$$

6.6 Decomposition Models

As the ILP does not scale, we propose a column generation scheme to help validate our heuristic for larger networks. We first present here a model using Column Generation, *CG-simple*. We then introduce two variants of the models, *CG-cut*, and *CG-cut+*. Indeed, problems dealing with energy-efficiency frequently lead to large integrality gap and bad precision. This is due to the On-Off phenomena of power models, which translates into large steps of the objective function. We thus try to improve the precision of the model by introducing different sets of constraints. We discuss the precision of the models in Section 6.7.3.

6.6.1 Column Generation Formulation

We propose a column generation formulation that relies on the concept of *Service Path*: each *Service Path* p is associated with a 4-uplet (v_s, v_d, c, D^c_{sd}) and defines: *(i)* a potential route for the request (v_s, v_d, c, D^c_{sd}) between v_s and v_d, *(ii)* node placement of the functions of chain c along the potential route. A route is described by parameters δ^p_ℓ, equal to the number of occurrences of the link ℓ in the path p. Node placement is given by a^p_{vi}, equal to 1 if the ith function of the chain c is located at node v, 0 otherwise. We denote by P^c_{sd} the overall set of *Service Path* for each request (v_s, v_d, c, D^c_{sd}).

We now define the set of variables. First set of decision variables: $x_\ell = 1$ if link ℓ is on (active), 0 otherwise. Note that links are powered off by pair, i.e., $x_{\ell=(v,v')} = x_{\ell'=(v',v)}$. Second set of decision variables: $y^p_d = 1$ if demand d is routed using configuration p, 0 otherwise. Integer variables: $k_v = \#$ required cores in node v.

The objective, i.e., the minimization of the energy, can be written

$$\min \quad \underbrace{\sum_{\ell \in L} P^{\text{ON}}_\ell x_\ell}_{\substack{\text{link switch} \\ \text{on energy}}} + \underbrace{\sum_{\ell \in L} \sum_{p \in P^c_{sd}} \delta^p_\ell \left(\sum_{d=(v_s,v_d,c) \in D} \frac{D^c_{sd}}{C^{\text{LINK}}_\ell} P^{\max}_\ell \right) y^p_d}_{\text{link bandwidth energy}} + \underbrace{\sum_{u \in V} P_v \, k_v}_{\substack{\text{node resource energy}}} \tag{6.13}$$

The constraint set decomposes into three sets of constraints.

One path per demand

$$\sum_{p \in P^c_{sd}} y^p_d = 1 \qquad (u_s, u_d) \in \mathcal{SD}, c \in C_{sd} \in D. \tag{6.14}$$

Link capacity

$$\sum_{d=(v_s,v_d,c) \in D} \sum_{p \in P^c_{sd}} D^c_{sd} \, \delta^p_\ell \, y^p_d \leq x_\ell \, C^{\text{LINK}}_\ell \qquad \ell \in L. \tag{6.15}$$

Node capacity

$$\sum_{d\in D}\sum_{p\in P^c_{sd}} D^c_{sd}(\sum_{i=1}^{n_c}\Delta_{f_i}a^p_{vf_i})y^p_d \le k_v \le C^{\text{NODE}}_v \qquad u \in V. \quad (6.16)$$

As we faced issues with large integrality gaps, we enhanced model (6.13)-(6.16) with different sets of cuts, through the next two models.

***CG-cut* model.** The first set of cuts in (6.17) states that, for each node, at least one incident link should always be on. Moreover, the second inequality given by Equation (6.18) enforces that at least $n-1$ links should be active to have a connected network (or different if not all-to-all).

$$\sum_{\ell\in\omega^+(v)} x_\ell \ge 1 \qquad u \in V \quad (6.17)$$

$$\sum_{\ell\in L} x_\ell \ge n-1. \quad (6.18)$$

***CG-cut+* model.** We further enhance the *CG-cut* model with:

$$x_\ell \ge \sum_{p\in P^c_{sd}} \gamma^p_\ell\, y^p_d \qquad \ell \in L, (u_s,u_d) \in \mathcal{SD}, c \in C_{sd} \quad (6.19)$$

where $\gamma^p_\ell = 1$ if the link ℓ belong the path p. Using (6.14), it follows that $\sum_{p\in P^c_{sd}} \gamma^p_\ell\, y^p_d \le 1$. It avoids the use of a big M formulation at the expense of a large number of constraints.

6.6.2 Solution Scheme

To solve the model of Section 6.6.1 efficiently, we need to recourse to column generation for solving the linear relaxation, and then to derive an ILP value, using the last restricted master problem. For additional details on linear programming and column generation schemes see, e.g., [Chv83].

There is a configuration generator, i.e., pricing problem, for each request (v_s, v_d, c, D^c_{sd}). Two sets of decision variables are required. First set is made of variables φ^i_ℓ such that $\varphi^i_\ell = 1$ if the provisioning of demand d uses link ℓ in layer i of the layered graph G^{L}, 0 otherwise. Second set contains variables a^i_v

such that $a_v^i = 1$ if the ith function (f_i^c) of chain c for request (v_s, v_d, c, D_{sd}^c) is placed on NFV node v, 0 otherwise. The formulation of the *Service Path* generator is given as follows.

$$\min -u_{sd}^{(6.14)} + \sum_{\ell \in L} \sum_{i=0}^{n_c} \varphi_\ell^i \times \left(P_\ell^{\max} \frac{D_{sd}^c}{C^{\text{LINK}}{}_\ell} + u_\ell^{(6.15)} D_{sd}^c \right) + \sum_{u \in V} \sum_{i=0}^{n_c - 1} a_v^i \times \left(u_v^{(6.16)} \Delta_{f_i} D_{sd}^c \right). \quad (6.20)$$

Path computation (flow conservation constraints):

$$\sum_{\ell \in \omega^+(v)} \varphi_\ell^i - \sum_{\ell \in \omega^-(v)} \varphi_\ell^i + a_v^i - a_v^{i-1} = 0 \qquad u \in V, 0 < i < n^c \quad (6.21)$$

$$\sum_{\ell \in \omega^+(v)} \varphi_\ell^0 - \sum_{\ell \in \omega^-(v)} \varphi_\ell^0 + a_v^0 = \begin{cases} 1 \text{ if } v = v_s \\ 0 \text{ else} \end{cases} \qquad u \in V \quad (6.22)$$

$$\sum_{\ell \in \omega^+(v)} \varphi_\ell^{n_c} - \sum_{\ell \in \omega^-(v)} \varphi_\ell^{n_c} - a_v^{n_c} = \begin{cases} -1 \text{ if } v = v_d \\ 0 \text{ else} \end{cases} \qquad u \in V. \quad (6.23)$$

$$\text{Link capacity:} \quad D_{sd}^c \sum_{i=0}^{n_c} \varphi_\ell^i \leq C_\ell^{\text{LINK}} \qquad \ell \in L. \quad (6.24)$$

$$\text{Node capacity:} \quad D_{sd}^c \sum_{i=0}^{n_c} \Delta_{f_i} a_v^i \leq C_v^{\text{NODE}} \quad u \in V. \quad (6.25)$$

6.6.2.1 Speeding up the Pricing Problem

The Pricing Problem corresponds to a constrained shortest path with negative weights on the layered graph, and we can use CPLEX to solve it. However, if we discard the capacity constraints, the problem becomes the *shortest path with negative weights* problem. It can be solved much faster than the original problem using the Bellman-Ford shortest path algorithm. Since we remove the capacity constraints, the set of solutions considered is a superset of the initial set of solutions. It is possible to find a path that might use more resources than available. In this case, we fall back the ILP solver to obtain a valid path. The weight of the inter-layer arcs is thus given by

$$w_{iv} = P_\ell^{\max} \frac{D_{sd}^c}{C^{\text{LINK}}{}_\ell} + u_\ell^{(6.15)} D_{sd}^c \qquad 0 \leq i < n_c, u \in V.$$

and the weight of intra-layer arcs by

$$w_{i\ell} = u_v^{(6.16)} \Delta_{f_i} D_{sd}^c \qquad 0 \leq i \leq n_c, \ell \in L.$$

6.6.2.2 Particularities of *CG-cut+*

By introducing the constraints (6.19) into the model, we also need to introduce a new set of variable γ_l into the Pricing Problem that indicates if the link ℓ is used in the path. The objective function becomes

$$\begin{aligned} \min \quad -u_{sd}^{(6.14)} + \sum_{\ell \in L} \sum_{i=0}^{n_c} \varphi_\ell^i \times \left(P_\ell^{\max} \frac{D_{sd}^c}{C^{\text{LINK}}{}_\ell} + u_\ell^{(6.15)} D_{sd}^c \right) \\ + \sum_{u \in V} \sum_{i=0}^{n_c} a_v^i \times \left(u_v^{(6.16)} \Delta_{f_i} D_{sd}^c \right) \\ + \sum_{\ell \in L} \gamma_l u_{sd,c,l}^{(6.19)}. \end{aligned} \tag{6.26}$$

and the link capacities constraints become

$$D_{sd}^c \sum_{i=0}^{n_c} \varphi_\ell^i \leq C_\ell^{\text{LINK}} \times \gamma_\ell \qquad \ell \in L. \tag{6.27}$$

Moreover, adding enhanced cuts creates negative cycles in the layered graph used for the Pricing Problem. We choose not to get rid of the cycles by

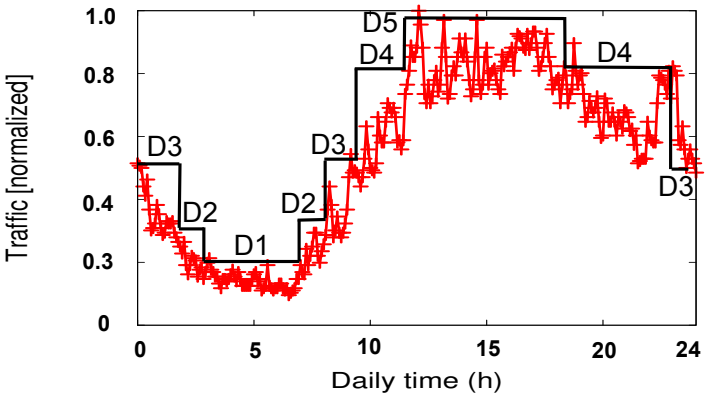


Figure 6.2: Normalized daily variation of traffic of a France Telecom network link and multi-period approximation

enumerating them all. Instead, we check if the solution provided by the solver contains any negative cycles. If that is the case, we add the corresponding constraints in the formulation and call the solver again. We repeat this process until the obtained solution no longer contains any negative cycles or the reduced cost is no longer negative. Removing the cycle has a negligible impact on the performance of the column generation scheme as it is executed only a few times at the start of the algorithm.

6.7 Numerical Experiments

In this section, we investigate the energy savings obtained by the Column Generation model. We compare the results with the ones of GreenChains heuristic algorithm. We first present the data sets we use for the experiments. We then take a look at the precision of the solutions obtained by the Column Generation model and GreenChains. We investigate different improvements of the model presented in Section 6.3. We then present the energy savings achieved for network topologies of different sizes. Last, we discuss the impact of the solutions on link usage and path lengths.

Service Chains	Chained VNFs	Rate	traffic (%)
Web Service	NAT-FW-TM-WOC-IDPS	100 kbps	18.2
VoIP	NAT-FW-TM-FW-NAT	64 kbps	11.8
Video Streaming	NAT-FW-TM-VOC-IDPS	4 Mbps	69.9
Online Gaming	NAT-FW-VOC-WOC-IDPS	50 kbps	0.1

Table 6.1: Service Chain Requirements [STV15]

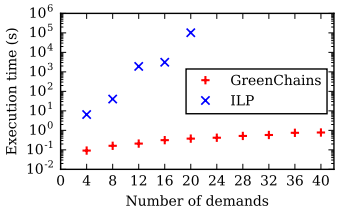
6.7.1 Data sets

In networks, each type of flows has to go through a different chain of network services. In our experiments, we consider four of the most frequent types of flows, as presented in Table 6.1: Video Streaming, Web Service, Voice-over-IP (VoIP), and Online Gaming. The traffic percentages are from [Net15]. For each one, we give the ordered set of functions required and the bandwidth used. In total, we have six different functions, and each function requires a different amount of cores to be executed.

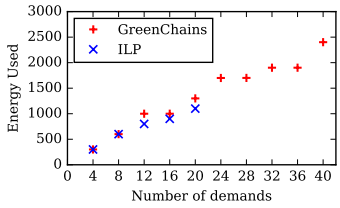
We tested the CG models and GreenChains on three topologies of different sizes from SNDlib [Orl+10a]: *pdh* (11 nodes and 64 directed links), *atlanta* (15 nodes and 44 directed links), and *germany50* (50 nodes and 176 directed links).

To obtain realistic looking traffic matrices, we generate, for each network, a set of demands from the traffic matrices provided in SNDlib: we divide each aggregate flow from a source to a destination into four demands corresponding to the four different types of traffic. We consider that the flows provided in the SNDlib data set represent aggregated flows of miscellaneous services. Thus, we can subdivided them into four different services. The original load of the flow is conserved, and each sub-flow load is given by the distribution of the last column of Table 6.1. For example, a flow with a charge of 1 is split into a Web Service, a VoIP, a Video Streaming and an Online Gaming sub-flows with a load of 0.182, 0.118, 0.699 and 0.001, respectively.

We tested the solution on a daily traffic to see how much energy can be saved during the day or at night. The variations of traffic come from a trace of a typical France Telecom link shown in Figure 6.2. Previous work



(a) Execution time



(b) Energy used

Figure 6.3: Comparison between the compact formulation and GreenChains

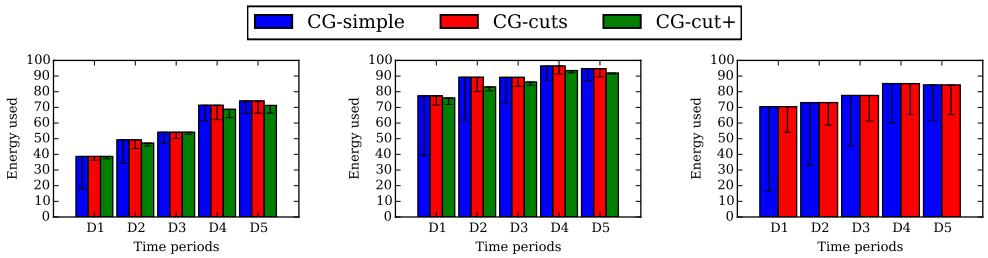
[Ara+16] indicates that using a small number of configurations during the day is enough to obtain most of the energy savings. In our case, we considered five different levels of traffic called D1 to D5. D1 represents the period with the lowest amount of traffic and D5 the one with the highest.

Traditionally, ISP networks use shortest path routing and operate their network with an overprovisioning factor of 2 or 3 [Fra+03; Iye+03], in order to be able to cope with failures and traffic growth. This means that links typically are used at only between 30 and 50% of their capacity. We set capacities accordingly at the beginning of the simulation. For each network, we solve the legacy scenario by routing requests on the shortest paths between each location of the service's functions. Each function location is chosen at random in the legacy scenario. We then choose the link capacities such that each link is at most used at 33% of its capacity. Finally, we considered equal values for the energy consumption of the links and nodes.

6.7.2 Compact formulation evaluation

We compare the results obtained by the heuristic algorithm, GreenChains, with the optimal results given by the integer linear program on a small network, *pdh*, with 11 nodes and 64 links. We consider instances with an increasing complexity: the number of demands varies from 4 to 40. Note that we consider multiples of 4 demands, as the traffic between a pair of nodes is divided into four different demands corresponds to different categories of traffic.

We compare the execution times of the ILP model and the algorithm in Figure 6.3a. The experiments are made on a Intel Xeon E5620 with 24GB



(a) pdh (b) atlanta (c) germany50

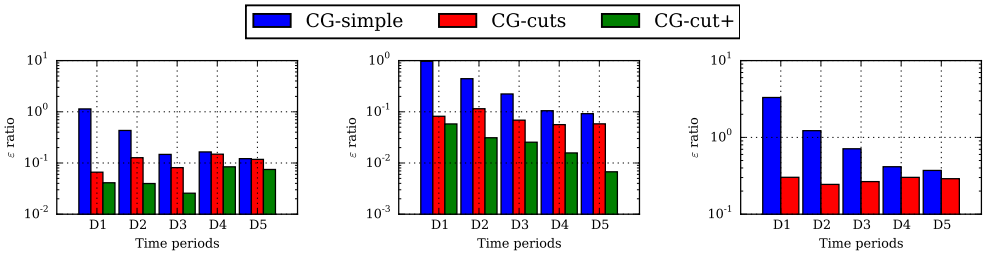
Figure 6.4: Performance of all three CG models on the (a) *pdh*, (b) *atlanta* and, (c) *germany50* network topologies. 100 represents the energy used by the legacy scenario.

of RAM. We see that the ILP model can be used to solve the problem with a reasonable amount of time for a maximum number of 16 demands. In this case, it takes around 45 minutes to return the optimal solution. The increase is exponential: for 20 demands, the execution time is almost 3 hours. On the other hand, GreenChains is much faster as it can find a solution in less than 1 second for 20 demands (0.38 s). It solves an instance with 40 demands in 0.78 s and the all-to-all instance (with 440 demands), considered in the following, in less than 7 s. We see that the ILP cannot be used in practice to solve instances with a large number of demands, and thus we use the GreenChains for the experiments on larger networks in the following.

The results regarding energy savings are given in Figure 6.3. GreenChains finds results within a precision ranging between 0% and 16% for the different number of demands. We consider this as good results given the difficulty of the EE-SFCP problem. Moreover, it means that the potential energy savings of using dynamic traffic and virtualization are in fact even greater than the one presented in the following.

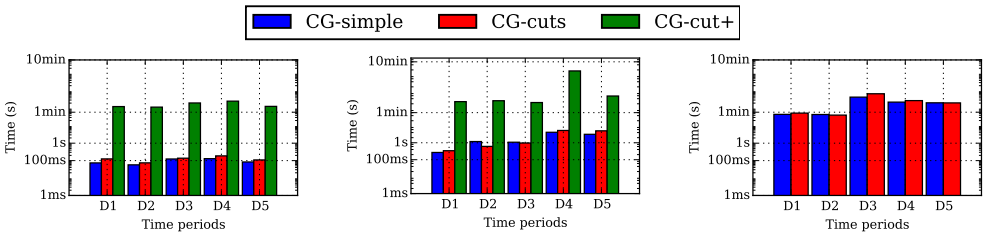
6.7.3 Quality of the Column Generation models

We now compare the performance of the three different CG models (*CG-simple*, *CG-cut*, and *CG-cut+*) with respect to their accuracy as given by $\varepsilon = (\tilde{z}_{\text{ILP}} - z^{\star}_{\text{LP}})/z^{\star}_{\text{LP}}$, where $z^{\star}_{\text{LP}}$ represents the optimal value of the relaxation of the Restricted Master Problem, and $\tilde{z}_{\text{ILP}}$ the integer solution obtained at the end of the column generation algorithm. In Figure 6.4, 6.5, and 6.6, we compare the solutions found by the three CG models for all three networks and for the



(a) pdh (b) atlanta (c) germany50

Figure 6.5: Accuracy, ε, of all three CG models on the (a) *pdh*, (b) *atlanta* and, (c) *germany50* network topologies



(a) pdh (b) atlanta (c) germany50

Figure 6.6: Execution times of all three CG models on the (a) *pdh*, (b) *atlanta* and, (c) *germany50* network topologies

5 different levels of traffic. We first observe in Figure 6.4, in which error bars represent the gap between the relaxed and integer solutions, that *CG-simple* and *CG-cut* provide similar solutions. However, ε dramatically varies, as shown in Figure 6.5. Cuts significantly improves ε: for *CG-simple*, it varies between 12% and 113% for pdh, 10% and 97% for atlanta, and 37% and 330% for germany50. For *CG-cut*, ε is between 7 to 15% for pdh, 6 and 12% for atlanta, and 24 and 30% for germany50. The ratio is further improved with *CG-cut+*: between 4 and 8% for pdh, 1 and 6% for atlanta. However, no solutions were found in a reasonable amount of time for the germany50 topology. As the energy savings are similar for the three models, it shows that the *three CG models provide rather accurate solutions, as confirmed by the solutions and accuracy of the* CG-cut *and* CG-cut+ *models.*

Finally, in Figure 6.6, we compare the execution times of the models. We observe that *CG-cut+* execution time (between 17 s and 5 h) is orders

of magnitude higher that the one of *CG-simple* (between 50 ms and 440 s) and *CG-cut* (between 70 ms and 670 s). This is greatly due to the fact that we speed up the resolution of the two previous model using the Bellman-Ford shortest path algorithm for the Pricing Problem. The second factor is that *CG-cut+*'s cuts slow the convergence time of the column generation drastically.

We now *focus on the* CG-cut *model*, as it offers the best compromise in terms of accuracy (w.r.t. *CG-simple* model) and computation time requirements (w.r.t. *CG-cut+* model) to solve large networks.

6.7.4 Energy Savings

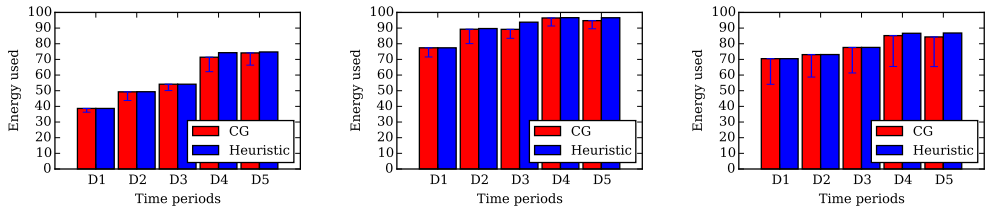
We now compare the energy savings obtained by GreenChains and *CG-cut*. We consider three scenarios in the experiments:

- *Legacy scenario.* This scenario corresponds to the one of a legacy network, whose operator does not try to reduce the energy consumption of its network. Its goal is to minimize the total bandwidth used while respecting the link capacity and the chain constraints. This scenario is used as a *baseline for comparison* for the energy-aware algorithms.

- *Hardware scenario.* The hardware scenario corresponds to one of an SDN (non-virtualized) network in which an operator tries to reduce its energy consumption by adapting the routing to the demands. In this scenario, the network functions are carried out by some specific hardware placed at given positions in the network.

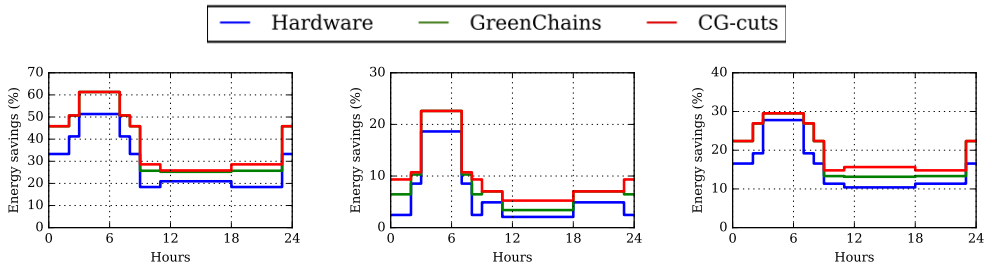
- *NFV scenario.* The NFV scenario is the one of a virtualized SDN network in which generic hardware nodes can execute any virtual network functions. This is the scenario solved by the solutions provided in Sections 6.4, 6.5 and 6.6.

We provide in Figure 6.7 the energy used for the five levels of demands for *pdh*, *atlanta*, and *germany50*. The values are normalized: 100 corresponds to the legacy scenario. We also present in Figure 6.8 the corresponding energy savings during the day. We see that we obtained important savings using virtualization: between 25 and 61% for *pdh*, 5 and 22% for *atlanta*, and 15 and 30% for *germany50*.



(a) pdh (b) atlanta (c) germany50

Figure 6.7: Energy used for GreenChains and the CG model on the three topologies.

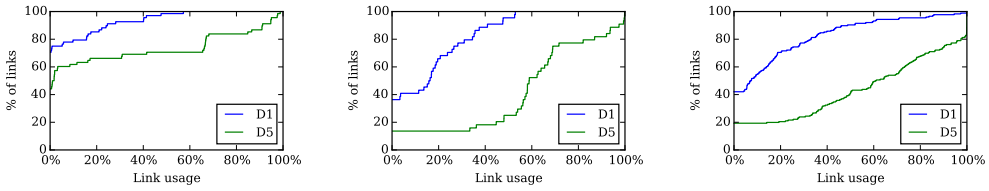


(a) pdh (b) atlanta (c) germany50

Figure 6.8: Saved energy for GreenChains for the three network topologies.

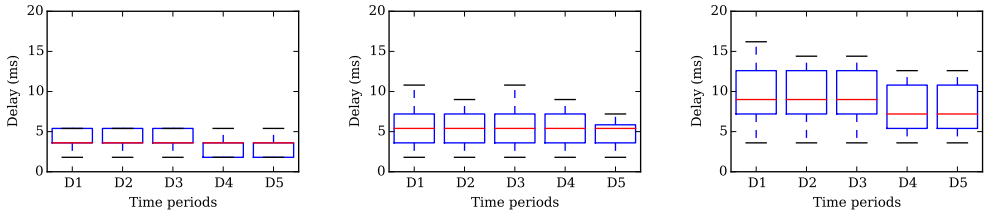
6.7.4.1 Validating GreenChains with *CG-cut*

We now compare the solutions provided by both GreenChains and *CG-cut* in Figure 6.7. Error bars on the *CG-cut* solutions represent the lower bounds given by $z^{\star}_{\text{LP}}$. For the lowest traffic periods (D1, D2 and in D3), both methods provide similar solutions for *pdh* and *germany50*. The CG model provides slightly better solutions when the traffic is higher, with a difference of 3 and 1% for *pdh*, of 5,and 2 for *atlanta*, and of 2 and 3% for *germany50* respectively in the D5 period. Observe that, even if CG only provides slight improvement of the heuristic's solutions, it shows (c.f. ε accuracy value) that the heuristic gives good results, regardless of the traffic period.



(a) pdh (b) atlanta (c) germany50

Figure 6.9: Link load for GreenChains for the three network topologies.



(a) pdh (b) atlanta (c) germany50

Figure 6.10: Delay in milliseconds for GreenChains for the three network topologies.

6.7.4.2 Link load

To reduce the amount of energy used by the network, we reroute some of the flows to be able to put links into sleep mode. This means that the remaining active links are more loaded. In Figure 6.9, we look at the link load given by GreenChains for the highest and lowest traffic periods. First, we see that, unsurprisingly, the percentage of links with no traffic is higher when the traffic is low, around 40% of the links for *atlanta* and *germany50*. When the network is at its highest utilization, it drops to around 15% for both networks. The *pdh* network, due to its higher link density, can have more links put into sleep mode. Indeed, between 44% and 71% of the links have no traffic. Moreover, at the lowest traffic period, no links are used at 100% for *pdh*, *atlanta* and are at most used up to 57%, 52% of their capacity, respectively. At rush hour, *pdh* and *atlanta* have at most links at 98 and 99% capacity while *germany50* has only one link at full capacity.

6.7.4.3 Impact on Delay

When some links are put into sleep mode, paths tend to become longer. However, we show in Figure 6.10 that the maximum delay of every path stays below the usual 50 ms latency value in Service Level Agreements: experienced delay is less than 5.4, 10.8 and 16.2 ms on *pdh*, *atlanta* and *germany50* respectively. Moreover, the median of the delay stays constant for *pdh*, *atlanta* at 3.6 and 5.4 ms, respectively. For *germany50*, it only increases from 7.2 for D5 (no link into sleep mode) to 9 ms for D1.

6.8 Conclusions

In this chapter, we investigated the potential of network virtualization to reduce the energy consumption of networks. We introduced a Column Generation model to solve the problem of minimizing network energy consumption while satisfying the SFC requirements. We also proposed GreenChains, an ILP-based heuristic that we validate using our Column Generation model. We then compared three different scenarios corresponding to a continuous deployment of the SDN and NFV paradigm for energy efficiency. We show that an operator, using SDN control, can save energy by choosing the paths of the flows dynamically according to the variations of demands during the day. Indeed, this allows the turning off of a large portion of network equipment. Indeed, compared to a legacy scenario, SDN can provide between 18 and 51% energy savings during the night. We also demonstrated that the deployment of VNF in an SDN network leads to additional energy savings between 4 and 12%. As a matter of fact, choosing dynamically the locations of network functions according to the variations of the demands allows a greater flexibility for the choice of the network paths and leads to the use of less network equipment.

References

[Ara+16] J. Araujo, F. Giroire, J. Moulierac, Y. Liu, and R. Modrzejewski. "Energy Efficient Content Distribution". In: *The Computer Journal* 59.2 (Feb. 2016), pp. 192–207 (cit. on p. 153).

[Bol+10] Raffaele Bolla, Roberto Bruschi, Franco Davoli, and Flavio Cucchietti. "Energy efficiency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures". In: *IEEE Communications Surveys & Tutorials* 13.2 (2010), pp. 223–244 (cit. on p. 138).

[Bol+14] R. Bolla, C. Lombardo, R. Bruschi, and S. Mangialardi. "DROPv2: energy efficiency through network function virtualization". In: *IEEE Network* 28.2 (2014), pp. 26–32 (cit. on p. 139).

[Cha+08] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright. "Power Awareness in Network Design and Routing". In: *Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM*. Apr. 2008, pp. 1130–1138 (cit. on p. 142).

[Chv83] V. Chvatal. *Linear Programming*. Freeman, 1983 (cit. on pp. 88, 148).

[CMN12] L. Chiaraviglio, M. Mellia, and F. Neri. "Minimizing ISP network energy cost: formulation and solutions". In: *IEEE/ACM Transactions on Networking (TON)* 20 (2 Apr. 2012), pp. 463–476 (cit. on p. 138).

[DGF10] Floriano De Rango, Francesca Guerriero, and Peppino Fazio. "Link-stability and energy aware routing protocol in distributed wireless networks". In: *IEEE Transactions on Parallel and Distributed systems* 23.4 (2010), pp. 713–726 (cit. on p. 138).

[Fra+03] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and S.C. Diot. "Packet-level traffic measurements from the Sprint IP backbone". In: *IEEE network* 17.6 (2003), pp. 6–16 (cit. on p. 153).

[Gir+10] Frédéric Giroire, Dorian Mazauric, Joanna Moulierac, and Brice Onfroy. "Minimizing routing energy consumption: from theoretical to practical results". In: *2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*. IEEE. 2010, pp. 252–259 (cit. on p. 138).

[Gir+15] F. Giroire, J. Moulierac, Truong Khoa Phan, and F. Roudaut. "Minimization of network power consumption with redundancy elimination". In: *Computer communications* 59 (2015), pp. 98–105 (cit. on p. 139).

[GMP14] F. Giroire, J. Moulierac, and K. Phan. "Optimizing Rule Placement in Software-Defined Networks for Energy-aware Routing". In: *IEEE Global Telecommunications Conference - GLOBECOM*. Austin, USA, Dec. 2014, pp. 2523–2529 (cit. on p. 139).

[GS03] Maruti Gupta and Suresh Singh. "Greening of the Internet". In: *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM. 2003, pp. 19–26 (cit. on p. 138).

[Gup+15] A. Gupta, M.F. Habib, P. Chowdhury, M. Tornatore, and B. Mukherjee. "On service chaining using virtual network functions in network-enabled cloud systems". In: *IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*. 2015, pp. 1–3 (cit. on p. 138).

[Gup+17] A. Gupta, B. Mukherjee, B. Jaumard, and M. Tornatore. "Service Chain (SC) Mapping with Multiple SC Instances in a Wide Area Network". In: *IEEE Global Telecommunications Conference - GLOBECOM*. 2017, pp. 1–6 (cit. on p. 138).

[Hui+18a] N Huin, A Tomassilli, F Giroire, and B Jaumard. "Energy-efficient service function chain provisioning". In: *IEEE/OSA Journal of Optical Communications and Networking* 10.3 (2018), pp. 114–124 (cit. on pp. 12, 45, 136).

[Hui+18b] Nicolas Huin, Myriana Rifai, Frédéric Giroire, Dino Lopez Pacheco, Guillaume Urvoy-Keller, and Joanna Moulierac. "Bringing energy aware routing closer to reality with SDN hybrid networks". In: *IEEE Transactions on Green Commu-*

nications and Networking 2.4 (2018), pp. 1128–1139 (cit. on p. 139).

[Iye+03] Sundar Iyer, Supratik Bhattacharyya, Nina Taft, and Christophe Diot. "An approach to alleviate link overload as observed on an IP backbone". In: *Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM*. Vol. 1. 2003, pp. 406–416 (cit. on p. 153).

[Le +13] Esther Le Rouzic, Edoardo Bonetto, Luca Chiaraviglio, Frederic Giroire, Filip Idzikowski, Felipe Jiménez, Christoph Lange, Julio Montalvo, Francesco Musumeci, Issam Tahiri, et al. "TREND towards more energy-efficient optical networks". In: *2013 17th International Conference on Optical Networking Design and Modeling (ONDM)*. IEEE. 2013, pp. 211–216 (cit. on p. 136).

[LMF11] L. Chiaraviglio, M. Mellia, and F. Neri. "Minimizing ISP Network Energy Cost: Formulation and Solutions". In: *IEEE/ACM Transactions on Networking* 20.2 (Apr. 2011), pp. 463–476 (cit. on p. 138).

[Mar+15] B. Martini, F. Paganelli, P. Cappanera, S. Turchi, and P. Castoldi. "Latency-aware composition of virtual functions in 5g". In: *NetSoft*. IEEE. 2015, pp. 1–6 (cit. on p. 138).

[Mat+13] Daisuke Matsubara, Takashi Egawa, Nozomu Nishinaga, Ved P Kafle, Myung-Ki Shin, and Alex Galis. "Toward future networks: a viewpoint from ITU-T". In: *IEEE Communications Magazine* 51.3 (2013), pp. 112–118 (cit. on p. 136).

[Mij15] Rashid Mijumbi. "On the Energy Efficiency Prospects of Network Function Virtualization". In: *CoRR* abs/1512.00215 (2015). URL: http://arxiv.org/abs/1512.00215 (cit. on p. 139).

[Mod+13] Remigiusz Modrzejewski, Luca Chiaraviglio, Issam Tahiri, Frederic Giroire, Esther Le Rouzic, Edoardo Bonetto, Francesco Musumeci, Roberto Gonzalez, and Carmen Guerrero. "Energy efficient content distribution in an ISP network". In: *2013 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2013, pp. 2859–2865 (cit. on p. 138).

[Moh+15] Ali Mohammadkhan, Sheida Ghapani, Guyue Liu, Wei Zhang, KK Ramakrishnan, and Timothy Wood. "Virtual function placement and traffic steering in flexible and dynamic software defined networks". In: *Local and Metropolitan Area Networks (LANMAN), 2015 IEEE International Workshop on*. IEEE. 2015, pp. 1–6 (cit. on pp. 10, 11, 46, 138).

[Net15] Index Cisco Visual Networking. "Cisco visual networking index: Forecast and methodology 2015-2020". In: *White paper, CISCO)* (2015) (cit. on p. 152).

[Nic+12] L. Niccolini, G. Iannaccone, S. Ratnasamy, J. Chandrashekar, and L. Rizzo. "Building a power-proportional software router". In: *USENIX Annual Technical Conference (USENIX ATC)*. Boston, MA, USA, 2012, pp. 89–100 (cit. on p. 142).

[Orl+10a] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. "SNDlib 1.0–Survivable Network Design Library". English. In: *Networks* 55.3 (2010), pp. 276–286. DOI: 10.1002/net.20371 (cit. on p. 152).

[Pha14] Truong Khoa Phan. "Design and management of networks with low power consumption". PhD thesis. Université Nice Sophia Antipolis, 2014 (cit. on p. 138).

[Rig+15] R. Riggio, Abbas Bradai, Tinku Rasheed, Julius Schulz-Zander, Slawomir Kuklinski, and Toufik Ahmed. "Virtual network functions orchestration in wireless networks". In: *Intl. Conf. on Network and Service Management (CNSM)*. 2015, pp. 108–116 (cit. on pp. 11, 138).

[SLX10] Yunfei Shang, Dan Li, and Mingwei Xu. "Energy-aware routing in data center network". In: *Proceedings of the first ACM SIGCOMM workshop on Green networking*. ACM. 2010, pp. 1–8 (cit. on p. 138).

[STV15] M. Savi, M. Tornatore, and G. Verticale. "Impact of Processing Costs on Service Chain Placement in Network Functions Virtualization". In: *IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*. Nov. 2015, pp. 191–197 (cit. on pp. 138, 152).

[Ver+11] Willem Vereecken, Ward Van Heddeghem, Margot Deruyck, Bart Puype, Bart Lannoo, Wout Joseph, Didier Colle, Luc Martens, and Piet Demeester. "Power consumption in telecommunication networks: overview and reduction strategies". In: *IEEE Communications Magazine* 49.6 (2011) (cit. on p. 136).

Chapter 7
Conclusion and Future Work

SDN and NFV have the potential to provide a solution to simplify management, enhance flexibility of the network, and reduce its costs. However, together with opportunities, they also bring new challenges to network operators that need to be explored.

One of these challenges deals with the optimal placement of VNFs in the network to satisfy the Service Function Chain requirements of the flows. Indeed, with a joint SDN-NFV paradigm, flows can be managed dynamically from end-to-end, and the network functions can be installed only along paths for which and when they are necessary, while, in a legacy network, a flexible control of network equipment would be very costly and impractical.

In Chapter 3, we investigated the problem of placing VNFs to satisfy the ordering constraints of the flows with the goal of minimizing the total setup cost. We proposed two algorithms that achieve a logarithmic approximation factor. In addition, for the special case of tree network topologies with only upstream and downstream flows, we devised an optimal algorithm. Our aim consists of proposing the first theoretical framework for studying the placement problem with ordering constraints.

However, a remaining unaddressed issue is considering flow rates and the accounting of practical constraints such as *soft capacities* on network functions or *hard capacities* on network nodes. An interesting future research direction may concern an investigation of the possibility of efficiently approximating these problems.

SDN and NFV also add additional constraints that make classical problems reconsiderable again. This is the case for the network protection problem, a problem that has been widely studied in the literature. We considered two different variants of the problem.

The first, in Chapter 4, is a network design problem. We consider a path-based protection scheme with a global rerouting strategy, in which, for each failure situation, we may have a new routing of all the demands. This makes the protection technique bandwidth-optimal. It is extremely expensive and

impractical to implement this technique on legacy networks due to the huge number of rules to install on the network devices. We assess the potential of this technique to be applied to SDN networks. To this end, we develop a scalable mathematical model that we handle using the Column Generation technique. We show the effectiveness of our proposed methods experimentally on real-world IP network topologies and on randomly generated instances. Also, with an implementation based on the OpenDaylight SDN controller with Mininet, we demonstrated the feasibility of the approach and showed that technical implementation choices may have a dramatic impact on the time needed to reestablish the flows after a failure takes place.

Then, in Chapter 5, we considered the problem of providing path protection against a single link failure under both dedicated and shared protection schemes. The problem also consists in provisioning VNFs in the required order, which adds a challenge to the classical version of the problem. We investigated the two different protection mechanisms and discussed their resource requirements, as well as the latency of their paths. For each mechanism, we developed a scalable exact mathematical model using column generation.

Finally, in Chapter 6, we studied the potentials of SDN and NFV in energy savings. Indeed, there is the need of reducing energy consumption from both environmental (the Information and Communication Technology (ICT) sector is responsible for between 2 and 10% of global energy consumption [08]) and economical (energy bills represent more then the 10% of telecom operators operational expenditure [Bel15]) reasons. With the centralized control allowed by SDN, the flows can be managed dynamically, and adapted according to the traffic conditions. We proposed both heuristics and ILP-based optimization models in order to be able to deal efficiently with the problem. We considered three different scenarios: a *legacy scenario* which represents legacy networks and which serves as a baseline for comparison, a *hardware scenario* in which the routing can be changed dynamically by a centralized SDN controller, but in which network functions are executed by specific hardware, and finally, an *NFV scenario* in which the network functions are virtualized and can be placed dynamically. We showed that we can save between 4 and 12% more energy using VNF than compared to using middleboxes.

In this thesis, we only addressed some of the challenges towards the Next-Generation Network. Many of them still need to be addressed to fully attain the benefits of the SDN and NFV paradigms. But overall, we believe that an SDN-NFV enabled network has the potential to boost NFV deployment

and support new efficient and cost-effective services.

References

[08] *SMART 2020 Enabling the low-carbon economy in the information age, http://www.smart2020.org/_assets/files/02_-Smart2020Report.pdf*. 2008 (cit. on pp. 12, 166).

[Bel15] Alcetel Lucent Bell Labs. *White Paper: Global What if Analyzer of NeTwork Energy ConsumpTion (GWATT). Bell labs application able to measure the impact of technologies like SDN & NFV on network energy consumption*. Murray Hill, NJ, USA, 2015 (cit. on pp. 12, 166).

Part IV

Appendix

Chapter 8

Data Center Scheduling with Network Tasks

Contents

The content of this chapter is an extended version of [Gir+19b].

8.1 Introduction

The increasing need for efficiently processing and analyzing huge amounts of data has led to data-oriented parallel computing solutions such as MapReduce [DG08], Dryad [Isa+07], CIEL [Mur+11], and Spark [Zah+12]. These solutions are based on input data partitioning over a number of parallel machines. Jobs are split up into finer–grained tasks, and partial results from the various stages of computation are then transferred through the network. Each stage requires all the outputs of the previous stage to be in place before moving to the next stage.

In this context, the network starts to become an increasingly significant bottleneck in the performance of parallel processing [Gre+09; Guo+08] and hence, an important resource to optimize in a data center. Indeed, decreasing the parallel communications' completion time may lead to completing the corresponding job faster [Cho+11; CZS14; Dog+14].

Today's most common applications spend a significant portion of their time in communications. As reported by [Cho+11], the communications accounted for 33% of total completion times of MapReduce jobs in Facebook's Hadoop cluster, and 42% for Monarch [Tho+11], an iterative MapReduce application in Spark identifying spam links on Twitter. The recent development of containers and micro-services [NS14] will amplify this trend. They further divide monolithic tasks into several subtasks, increasing the number of communications in the network.

Usually, when a job arrives, the orchestrator tries to optimize the data center resources and decide on which servers the job's tasks should be executed. Traditionally, this is done using scheduling algorithms which take into account properties of the server, such as CPU usage and memory utilization, and of the task, such as execution time, task deadline, and task activation time. The effects of the placement of the tasks on the network's resources are not usually taken into consideration. However, taking into account network resources when placing tasks is now of primary importance for a large number of applications to reduce the communication overhead.

Some scheduling models have been introduced to this end, such as *Scheduling with communication delays*, or *with communications costs*. If on one hand, they take into account communication delays, on the other hand, they do not consider the fact that network bandwidth might be limited and that the communications may compete for it, leading to an additional delay or cost when a large number of communications are performed at the same

time.

We thus introduce a *new scheduling framework* which takes into account the limited communication bandwidth. In this framework, traditional (CPU) tasks stand alongside new *network tasks*. As usual, (CPU) tasks have to be executed by servers, but network tasks have to be executed by *network machines*, aiming to model the limited network capacity. The originality and difficulty of this study, compared to scheduling with non-identical machines, lies in the fact that *network tasks may or may not be executed depending on the placement of the CPU tasks*.
Indeed, when placing two CPU tasks T_1 and T_2, we would incur a communication delay only in the case in which T_1 and T_2 are scheduled on two different CPU machines. In such a case, we would have a network task $T_{1\to 2}$ to schedule on one of the network machines.
Our contributions can be summarized as follows.

- We introduce a new scheduling framework to model communication delays when tasks are competing for a limited network bandwidth. The idea is to model communications with network tasks which have to be executed on network machines.

- We show that the problem of scheduling data center jobs while routing their communications can be modeled with our scheduling framework using a simple set of network machines.

- We then study a new problem, Scheduling With Network Tasks, with the goal of minimizing the makespan of a set of tasks. The problem is NP-complete and we show that the simple 3-approximation List Scheduling algorithm with communication delays may be as bad as the simple algorithm putting all tasks on a single machine when the network bandwidth is limited.

- We then propose a generalized version of the classical List Scheduling algorithm for our framework, called Generalized List Scheduling. We show that our algorithm is optimal on simple MapReduce workflows.

- We also introduce a two-phase algorithm, Partition. In the first phase, the algorithm partitions the tasks into the available machines. In the second phase, we schedule at which time and in which order the

tasks should be done. We provide approximation algorithms for the two phases.

- Finally, we evaluate the practical behavior of our algorithms.

We perform extensive simulations using workflows based on Google Trace statistics [RWH11]. We show that our algorithms are very efficient for scenarios in which network capacity has to be taken into account.
The rest of this paper is organized as follows. In Section 8.2, we review related works in more detail. We then formally introduce the new framework and scheduling problem formally in Section 8.3. We discuss the hardness of the problem in Section 8.4. We then propose two algorithms in Section 8.5 and we evaluate them in Section 8.6. Finally, we draw our conclusions in Section 8.7.

8.2 Related Work

Optimizing Data Center Communications. Recent works have started to address the problem of optimizing network activity in order to improve job performance.

In [Cho+11], the authors propose a centralized application-level mechanism to coordinate transfers in the shuffle stage of MapReduce jobs with the goal of reducing the average job completion time. Indeed, according to their measurements on MapReduce applications, up to 50% of the completion time may be consumed in the shuffle phase. To this end, the authors propose a method based on weighted fair sharing at the cluster level. They show that, with their approach, the shuffle phase duration can be reduced by a factor of 1.5.

A family of works is based on the coflow abstraction [CS12], that is, a collection of parallel flows belonging to the same job. Varys [CZS14] is a coordinated coflow scheduler designed with the goal of maintaining high network utilization and guaranteeing starvation freedom. [Cho+11] and [CZS14] are centralized coflow schedulers. A decentralized solution is presented in [Dog+14]. The authors design and implement Baraat, a decentralized task-aware scheduling system for data centers. The goal is to minimize task completion time. Their solution is based on scheduling network resources at the unit of a task. They show that FIFO–based schemes

perform well, allowing to reduce both the average and the tail task completion times.

In [CKL], the authors consider the problem of scheduling all three phases (i.e., Map, Shuffle, and Reduce) of the MapReduce process. They develop constant factor approximation algorithms to minimize the mean response time over all jobs.

Corral [Jal+15] is an offline planning algorithm with the goal of jointly optimizing the placement of data and compute, and improving the application latency. Corral performs network-aware task placement. Large shuffles are separated from each other to reduce network contention in the cluster and jobs are run across a small number of racks to their data locality.

In this paper, we introduce a *new theoretical framework* to address the problems considered in these works. In this framework, we define a new problem, Scheduling With Network Tasks, and we propose (provably) efficient algorithms to solve it.

Scheduling. The problem of the paper Scheduling With Network Tasks is related to different classic scheduling problems, see e.g., [CPW99] for a comprehensive survey.

Scheduling with *precedence constraints* or *list scheduling* was introduced in [Gra66]. The authors model the precedence with a directed acyclic graph (DAG), in which an arc D_{ij} between two tasks T_i and T_j means that task T_i has to be completely processed before T_j may begin its execution. The authors provide a $(2 - 1/m)$-approx of the problem. In the 90s, communications were introduced in the family of problems called *scheduling with communication delays.* At the end of a task, some data may be sent to other machines. A communication delay d_{ij} is paid to send data from machine i to machine j. The general problem of minimizing the makespan is still open (even with an infinite number of machines). However, when the communication delays are all the same for a given source ($d_i = d_{ij}$) and when a task can be duplicated on several machines to avoid some communication costs, there exists a 2-approximation algorithm [PY90]. When the communication costs are further simplified to be unitary, a 2-approximation exists without the additional hypothesis above [Ray87]. Note that a critical difficulty of scheduling with communication is that the communication delay may or may not be paid depending on the task placement. In the problem variant in which the communication is always paid, a 2-approximation for the general setting was presented in [MQS98]. This result can be extended to obtain a

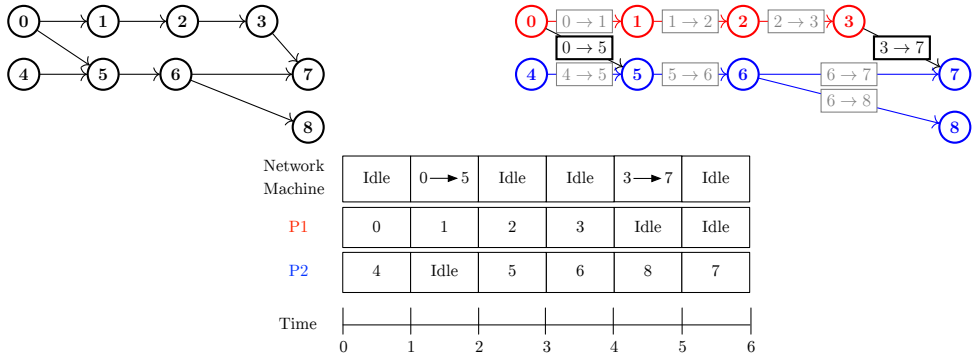


Figure 8.1: (Top Left) The dependency (di)graph of a job J with 9 (CPU) tasks. (Top Right) Modeling with network tasks indicated with rectangles. We provide a feasible schedule of job J. (Middle) Scheduled graph: (CPU) tasks executed by Machine 1 are in red, by Machine 2 in blue, carried out network tasks in black, not executed ones in gray. (Bottom) Timeline.

3-approximation algorithm for our generalized problem.
In all the above models, no network capacity is assumed. The model of this paper, on the contrary, *takes into account the competition between flows to access a limited amount of network bandwidth.*

8.3 A New Scheduling Framework

8.3.1 Problem and Example

We consider a set of jobs (often referred to as *workflows*) which have to be executed on m machines (also called processors or servers in the literature). A machine M_j has a processing speed S_j. Each job is made up of one or several tasks with dependency constraints between them. We denote by $\mathcal{T}$ the set of all the tasks (of all the jobs). The size of a task T_i is denoted by s_i. The completion time of a task T_i on the machine M_j is $c_{i,j} = s_i/S_j$.

The dependency between tasks in a job is expressed through a directed acyclic graph (DAG) in which an arc between tasks T_i and T_j means that T_i has to be completed before T_j may start. The set of jobs is thus modeled by a forest of DAGs.

When the task T_i has completed its execution, it may have computed data which is needed to execute task T_j. We model this by introducing a

network task $T_{i\to j}$ which has to be executed after task T_i and before task T_j. The size of $T_{i\to j}$ is denoted by $s_{i\to j}$. Network tasks have to be executed on a set of k network machines, which represent network links (or communication channels). The network machine N_ℓ has a capacity C_ℓ. The completion time of a network task $T_{i\to j}$ on the network machine N_ℓ is $c_{i\to j,\ell} = s_{i\to j}/C_\ell$.

We now define the new scheduling problem.

Problem 4 (Scheduling With Network Tasks)**.** Given a set of jobs $\mathcal{J}$ composed of tasks linked by a dependency digraph G and a set of network tasks $\mathcal{N}$, a set of m CPU machines, and k network machines, find the scheduling of $\mathcal{J}$ minimizing the makespan, that is, the maximum completion time of the jobs.

Example: Consider a system with 2 machines, M_1, M_2, of processing speeds $S_1 = S_2 = 1$, and one network machine N_1 of capacity $C_1 = 1$. We want to execute the job J with 9 tasks and the dependency digraph given in Fig. 8.1. We also provide the digraph with the potential network tasks to be executed. We set all task sizes to one in this example.

In Fig. 8.1, we provide a feasible schedule for job J. At time 0, tasks T_0 and T_4 are the only tasks which may be executed. They are placed on machines M_1 and M_2 respectively. Their completion time is 1 (size/processing speed). At time 2, T_1 can be executed on M_1, as its only predecessor, T_0, has been executed, and as its result is available in M_1. All predecessors of T_5 have been executed, but the task cannot be carried out, as the result of T_0 is in M_1 and the one of T_4 is in M_2. The result of T_0 is thus sent to M_2, i.e., the network task $T_{0\to 5}$ is executed by the network machine. It takes one time unit (size/capacity). The job completion time is thus 6.

8.3.2 Modeling Data Center Orchestration with Communication

We show here that we can model data center task orchestration and network resource allocation using our scheduling framework with a simple set of network machines.

Preliminary. Our framework directly models simple networks such as a set of machines connected via a bus by using a model with a single network machine or connected via an antenna (WiFi, 4G, ...) using a model with one network machine per channel. However, more complex networks can be

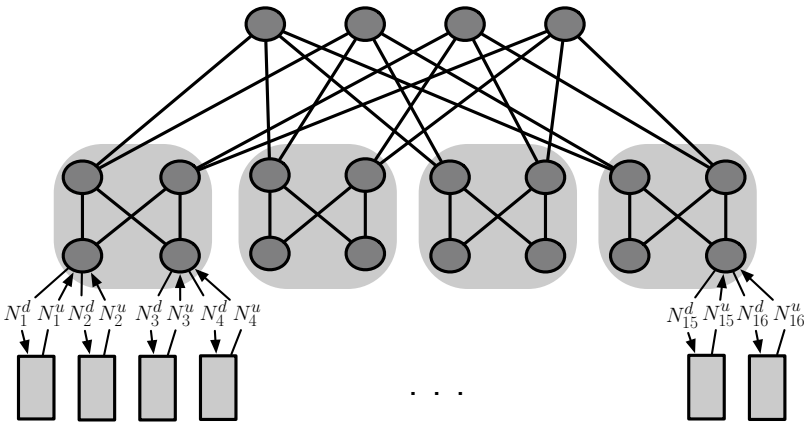


Figure 8.2: Modeling data center communications with Network machines for a 4-Fat Tree with 16 (racks of) servers.

represented.

Data center networks. The *simplicity of the model lies in the fact that only border links have to be modeled*. Indeed, data center architectures are built with large bisection bandwidth [CGC16]. Topologies such as Fat Trees or VL2 have full bisection bandwidth. In fact, they are permutation networks. It means that when the capacity is available at the border to send and receive a communication, it is always possible to find a path inside the network for the communication. Thus, only border links (i.e., links between the servers and the ToR switches) have to be taken into account.

We consider a data center with m servers, see an example in Fig. 8.2. In large data centers, what we refer to as servers are in fact a rack of servers. In this case, we only model inter-rack bandwidth, as within-rack bandwidth is usually 5 to 20 times larger than inter-rack bandwidth [Ahm+14]. Each server is modeled by a machine. However, we now introduce two network machines per link connecting a Top of Rack (ToR) switch to a server M_j, one for the download traffic, N_j^d, and the other for the upload traffic, N_j^u. When a network task is scheduled to be executed at time t between machines M_i and M_j, the task $T_{i \to j}$ is placed in both the upload network machine N_i^u of M_i and the download network machine N_j^d of M_j in the same time step t. The parallel execution of the task in both machines models the communication between the two machines.

In the following, we assume that the machines and the network machines

are identical. For the machines, this is a classical case considered in scheduling problems and is often true in practice in data centers. For the network machines, they are all representing links between servers and ToR switches and thus are often similar in data centers. Thus, $c_{ij} = c_i$ for all tasks $T_i \in \mathcal{T}$ and $c_{i\to j,\ell} = c_{i\to j}$ for all $T_{i\to j} \in \mathcal{N}$.

Also, less efficient networks can be modeled. In this case, it is enough to reduce the capacity of the network machines by a factor equal to $\frac{C}{O(m \log m)}$, with C the minimum multicut of the network, to ensure that paths exist, see e.g., [GVY93]. The model then gives a $\frac{C}{O(m \log m)}$-approximation of the makespan.

8.4 Hardness

We show here that Scheduling With Network Tasks is harder than scheduling with communication delays. Both problems are clearly NP-complete as they are generalizations of the problem of scheduling with precedence. However, there exists a simple greedy algorithm which has a 3-approximation factor when there are communication delays (but an infinite network capacity). We prove that this algorithm may be arbitrarily bad in our framework (by arbitrarily we mean $\Omega(m)$-approximate, i.e., the algorithm does not do significantly better than the simple algorithm putting all jobs on a single machine).

8.4.1 List-Scheduling

Next, we study the performance of the well-known "List Scheduling" algorithm which is 3-approximate in the case of infinite network capacity, i.e., b $= +\infty$ (see [Ray87]). Initially, we describe the algorithm and then we show that its approximation ratio is bad in the worst case, even when considering only unit time tasks.

List scheduler. The UET list scheduler algorithm [Ray87] provides a 2-approximation of the problem *scheduling with communication delays* when (CPU) task completion times and communication delays are unitary. We say that a task $T_j \in \mathcal{T}$ is available on Machine $M_p \in M$ during the time slot $(t-1, t]$ if it has no parent or if each of its parents has been completed either on machine M_i at time $< t-1$ or on machine $M_j \neq M_i$ at time $< t-2$. Note that, in this case, T_j can be feasibly executed by M_p during $(t-1, t]$.

Algorithm 8 Generalized UET List scheduler

```
1: U = 𝒯                                   ▷ U is the set of unprocessed tasks
2: t = 0                                                    ▷ t is the clock
3: while U ≠ ∅ do
4:     t = t + 1
5:     for p = 1, 2, ..., m do                          ▷ Iterate on machines
6:         Compute the set of available tasks A_{p,t}
7:         if A_{p,t} ≠ ∅ then
8:             min = {T' ∈ A_{p,t} | T' ⊏ T for all T ∈ A_{p,t}}
9:             Allocate to machine M_p the task min at time slot (t − 1, t]
10:        end if
11:    end for
12: end while
```

Initially, the algorithm computes a total order $\sqsubset$ of the tasks of $\mathcal{T}$ (containing the partial order defined by the jobs) which corresponds to a feasible schedule if all tasks are executed on a single machine. Then, it produces a schedule by proceeding time slot by time slot and machine by machine deciding a subset of available tasks to be executed during the slot $(t-1, t]$. The pseudo-code of the algorithm is provided in Algorithm 8.

Efficiency when bandwidth is limited. The generalized UET List scheduler provides an ordered list of tasks to be executed for each machine. We now consider the same schedule in the scenario in which bandwidth is limited. In this case, a task which was scheduled at time t by the list scheduler may have to wait and be scheduled later after all the necessary communications are done. Note that the execution of the algorithm defines a natural (partial) order on the network tasks when executed with limited bandwidth. The network tasks of a task T_j with $T_i \sqsubset T_j$ cannot be executed before all network tasks of T_i have been executed. The partial order can then be extended arbitrarily to a total order.

Theorem 2. List Scheduler is $\Omega(m)$-approximate when network bandwidth is limited even in the case of unitary costs.

Proof. We consider an instance of the problem with m machines and one job with m^2+m+1 tasks, where the DAG of precedence constraints G consists of 3 layers of nodes. The first layer contains the tasks $T_1, T_2, \ldots, T_{m^2}$, the second layer consists of the tasks $T_{m^2+1}, T_{m^2+2}, \ldots, T_{m^2+m}$ while the third layer

contains only the task T_{m^2+m+1}. Moreover, we have the following precedence relations: task T_{m^2+i} is preceded by tasks $T_{(i-1)m+1}, T_{(i-1)m+2}, \ldots, T_{im}$, for $1 \leq i \leq m$, and task T_{m^2+m+1} is preceded by tasks $T_{m^2+1}, T_{m^2+2}, \ldots, T_{m^2+m}$. There is a single ($k = 1$) network machine, N_1, of capacity $C_1 = 1$.

In the optimal schedule S^*, tasks $T_{(i-1)m+1}, T_{(i-1)m+2}, \ldots, T_{im}$ are executed by machine M_i followed by T_{m^2+i}, for $1 \leq i \leq m$. The task T_{m^2+m+1} is executed $m-1$ units of time after the completion of T_{m^2+1} on machine M_1. Only $m-1$ communications are performed during $(m+1, 2m]$ from $T_{m^2+2}, T_{m^2+3}, \ldots, T_{m^2+m}$ to T_{m^2+m+1}. The makespan of this schedule is $C(S^*) = 2m + 1$.

On the other hand, Algorithm 8 may choose an order scheduling tasks $T_{(i-1)m+1}, T_{(i-1)m+2}, \ldots, T_{im}$ simultaneously on machines $M_1, M_2, \ldots, M_m$, respectively, during the time slot $(i-1, i]$, for $1 \leq i \leq m$. The task T_{m^2+i} is executed by machine M_i. Lastly, the task T_{m^2+m+1} is executed by machine M_1. Globally, with Algorithm 8's schedule, $(m+1)(m-1)$ communications have to be done. $m(m-1)$ between tasks of Layers 1 and 2, and $(m-1)$ between tasks $T_{m^2+2}, ..., T_{m^2+m}$ and the task of Layer 3, T_{m^2+m+1}. No communication is done during the first and last time slot. During the other time slots, only one communication is performed. That is, the makespan computed by the algorithm is $C(S) = m^2 + 1$.

□

The main problem of Algorithm 8's schedule is that it performs a large number of communications compared to the optimal solution. The trivial algorithm which schedules all tasks on a single machine and produces a schedule with no communications is obviously m-approximate and Theorem 2 implies that List Scheduling is at least as bad as this trivial algorithm. It is thus of primary interest to find efficient algorithms to deal with limited bandwidth.

8.5 Algorithms

In this section, we propose two algorithms to solve the problem of Scheduling With Network Tasks. The first one is a generalization of the well known List Scheduling algorithm. The second one divides the problem into two subproblems. The first subproblem computes an assignment of the tasks to machines while minimizing the CPU and the networking work. The second subproblem computes a schedule for the tasks once the placement has been selected. We provide approximation algorithms for the two subproblems.

8.5.1 Generalized List Scheduling

We propose a new algorithm, Generalized List Scheduling (referred to as G-List), to solve our problem. To do so, we generalize the notion of an available task defined for the list scheduler algorithm [Ray87]. The goal is then to avoid carrying out useless network tasks. The main idea is two-fold: (1) Like classical greedy algorithms, we consider tasks and their possible assignments to machines. However, the same task may need different amounts of communications if assigned to different machines. We thus consider all the possible (available task, machine) couples at time t and sort them according to the number of required communications by the schedule. The algorithm thus places a task on a machine in which the most data is available if possible. (2) A task is placed on a machine at time t only if all its communications tasks can be done before time t. Otherwise, we delay its placement.

Description of the algorithm. A high level pseudo-code of G-List is provided in Algorithm 9. We define an *available task/machine-assignment* at time slot $(t-1, t]$ as a pair task/machine $(T \in \mathcal{T}, M)$ for which all preceding tasks of T have been completed before time $t-1$ and for which all needed communications with machines different than P can be scheduled before time $t-1$. At each time slot, G-List computes the set A of available task/machine-assignments. It then sorts the tasks in the set according to the amount of needed communications to be scheduled. While A is not empty, it schedules $(T_{\min}, M_{\min})$, the minimum element of the set. It then updates A by removing the task/machine-assignments with machine $M_{\min}$ and the ones whose needed communications cannot be scheduled any more.

Note that A is not computed and sorted from scratch at each iteration. Indeed, A can be updated using simple efficient algorithms and data structures. Moreover, when completion times of tasks are large, it is not necessary to iterate on all time slots. Several features are added to improve the algorithm:

(1) In case of ties, we place first the task whose out-tree has the longest branch, considering the sum of CPU and network tasks.

Indeed, the longest branch is a lower bound on the time to process the tasks depending on a task. It may be seen as a generalization of placing first tasks with longer processing times in the classic 4/3-approximation algorithm for scheduling [Gra66].

(2) The algorithm makes two passes: the first one considering the workflow

Algorithm 9 GENERALIZED LIST SCHEDULING

```
1: U = 𝒯                                   ▷ U is the set of unprocessed tasks
2: t = 0                                    ▷ t is the clock
3: while U ≠ ∅ do
4:     t = t + 1
5:     compute A the set of available task/machine-assignments.
6:     sort A according to number of needed communications
7:     while A ≠ ∅ do
8:         (T_min, M_min) = min(A)
9:         Allocate to machine M_min the task T_min at time slot (t − 1, t]
10:        Allocate needed network tasks for T_min to network machines in
   previous time slots
11:        Update A
12:    end while
13: end while
```

and the second one considering the "reverse workflow" in which an arc between task T_i and T_j is transformed into an arc between tasks T_j and T_i. Then, the best between the two passes is selected. The idea is that out-trees are optimized by the first pass and in-trees by the second pass, in the sense that tasks of the same subtrees are placed on the same machines if possible.

(3) For each job, we designate a longest branch as the master branch; all of its tasks are executed by a so called *master* machine. Then, before placing a task on a slave machine, we first test that it would not be faster to place it on the master machine when it will be free. That is, we only place a task on a slave machine if its completion time plus the time to send back the result to the master is smaller than the completion time of the master machine.

Discussion. Note that, when considering no dependency between tasks (and thus no communication), G-LIST boils down to the classical greedy scheduling algorithm which is a 4/3 approximation. When considering dependency and no communication, it reduces to *list scheduling* of [Gra66], and when considering dependencies and communication (but no bandwidth limit), to List Scheduler of [Ray87].

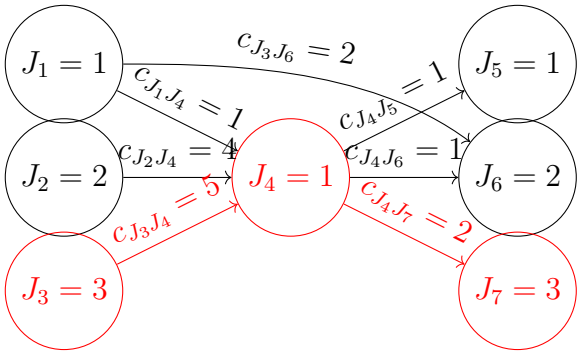


Figure 8.3: Example of a dependency graph . The master branch is highlighted in red.

Figure 8.4: Example of simple (left) and single-stage (right) MapReduce workflows with i map tasks and j reduce tasks.

8.5.2 Optimality on simple MapReduce Workflows

We consider the specific case of a simple MapReduce workflow, in which there is a single *Map* phase and a single *reduce* task. Formally, a simple MapReduce workflow is defined by a DAG with a source task s linked to $n-2$ tasks, $T_1, \ldots, T_{n-2}$, which are linked to a target task d, see Fig. 8.4. The completion times of tasks $T_1, \ldots, T_{n-2}$ are equal. Similarly, the communication times of tasks $T_{s \to T_i}$ for $1 \leq i \leq n-2$ are equal and the communication times of tasks $T_{T_i \to d}$ for $1 \leq i \leq n-2$ also. Note that simple MapReduce workflows are very frequent in data centers and that they also model simpler workflows (by setting to 0 some completion times) such as *Map* workflows and *Reduce* workflows which are also very common [Ren+13].

We prove here that G-List is optimal on a simple MapReduce Workflow.

Note that the problem is NP-complete if the processing times of tasks

$T_1, ..., T_{n-2}$ are different. Indeed, an instance of the k-PARTITION PROBLEM can be directly reduced to an instance of the problem, in which the numbers are the processing times of the tasks of a MapReduce workflow and for which the communication times are 0 and the capacity is infinite. The problem is NP-complete and no pseudo-polynomial algorithm exists to solve it when $k \geq 3$ [MD79].

Proposition 6. G-LIST is optimal on simple MapReduce workflows.

Proof. Let us compute the makespan of G-LIST on a simple MapReduce workflow. We note a (resp. b and c) the completion time of network tasks $T_{s \to T_i}$ (resp. of (CPU) task T_i and of network tasks $T_{T_i \to d}$) for $1 \leq i \leq n-1$.

G-LIST first selects any available branch (all branches are equivalent) of the workflow to be executed by a master machine. Note that without loss of generality we can always consider that the master machine executes both task s and d.

The makespan is thus given by the time at which the master machine finishes the last job d, denoted by t_m.

We denote by κ, the number of intermediate tasks (out of the $n-2$) carried out by slave machines. The master thus carries out task s, $n-2-\kappa$ intermediate tasks, and task d.

$$t_m = c_s + \max((n-2-\kappa)b, t_s) + c_t,$$

where t_s is the time at which the slave machine receiving the last job (we refer to this machine as the last slave machine in the following) has finished sending its last result. Indeed, the task t is done when the master has finished all the $(n-2-\kappa)$ intermediate tasks assigned to it, after a time $(n-2-\kappa)b$, and once the last data was received after a time t_s given by

$$t_s = t_f + \left(\left\lceil \frac{\kappa}{m-1} \right\rceil - 1 \right) t_I + b + t_\ell,$$

where t_f is the time at which the last slave machine receives its first data; the interjob time t_i is the time between the execution of two tasks; and t_ℓ is the time to send the result of the last job, starting from the end of its execution. We have $t_f = a(m-1)$ if $m-1$ divides κ, and $t_f = a(\kappa \mod (m-1))$ otherwise.

G-LIST sends a task to a slave machine only if it is faster to send its data, execute it, and get back its result, than executing it on the master machine.

Thus, G-List selects

$$\kappa = \arg\min_{\kappa}(\max(n-2-\kappa)b, t_s)).$$

The last step is to show that the makespan of G-List is optimal. Due to lack of space, the proof is not provided here, but it can be found in [Gir+19a].

- If $(n-2-\kappa)b \geq t_b$, the limiting resource is the CPU of the master machine. The makespan of G-List is optimal (i) as executing a job carried out by the master on a slave machine would increase the makespan; (ii) and, as the master machine takes the minimum time, $(n-2-\kappa)b$, necessary to carry out the $(n-\kappa-2)$ jobs assigned to it, as it always works during the whole execution of the algorithm.

We now consider that $(n-2-\kappa)b < t_b$. We also first consider the (most frequent) case in which the time to send the data of an intermediate task is larger than the time to send the results, that is, $a \geq c$. In this case, when the last slave machine has executed a job, the result of the previous machine has been sent, and it can then directly send its result. We thus have that

$$t_\ell = c$$

and that the interjob time t_i is

$$t_i = \max(b, a(m-1)).$$

We distinguish between two cases:

- if $b \geq a(m-1)$ (recall that we also have $a \geq c$), the limiting resource is the upload bandwidth of the master machine. We get

  $$t_s = a\kappa + b + c.$$

  The makespan of G-List is optimal, as t_s is the minimum time necessary for the last slave machine to send the results of the $\frac{\kappa}{m-1}$ tasks assigned to it. Indeed, $a\kappa$ is the minimum time to receive the data of the last job, as the master machine sends data continuously before. Then, the last slave machine executes the job in time b and sends it back in time c.

- if $b < a(m-1)$ (recall that we also have $a \geq c$), the limiting resource is the CPU of the slave machine. Thus,

$$t_s = t_f + \left\lceil \frac{\kappa}{m-1} \right\rceil b + c.$$

  Again, t_s is minimum. Indeed, the time to receive the first packet t_f is minimum, as the master machine always sends data in the network until κ jobs are sent to the slave machines. The time to carry out the $\frac{\kappa}{m-1}$ tasks is $\frac{\kappa}{m-1}b$, which is optimal. Lastly, the result of the last job is sent in time c.

Similarly, G-List is optimal in the last following cases.

- $a < c$ and $b \geq a(m-1)$, we have

$$t_s = a + b + \kappa c.$$

  The limiting resource is the download bandwidth of the master machine, which is always used.

- $a < c$ and $b < a(m-1)$, we get:

$$t_s = t_f + \left\lceil \frac{\kappa}{m-1} \right\rceil b + c.$$

  The limiting resource is the CPU of the slave machine, which is always used as soon as the data is received (in minimum time).

□

8.5.3 Partition

When the workflows are complex, the greedy algorithm may have difficulty in assigning the tasks to the available machines while minimizing the network load. To prevent this, an efficient method consists in first carrying out a partition of the tasks to be done by machines while minimizing the network tasks that would be necessary to be done. We call this first phase or subproblem the Partitioning to Schedule. For this subproblem, we provide an approximation algorithm with factor $O(\sqrt{\log n \log m})$, with n being the number of tasks and m the number of machines, which comes from the

best approximation factor for the $k-$balanced partitioning problem. When this preliminary phase is done, we just have to decide the order to process the tasks. We call this problem the SCHEDULING WHEN PLACED problem. We provide an algorithm (which is a generalization of Hu's algorithm to handle network tasks) which is a depth-approximation of the problem. Practical workflows have low depth, e.g., typically less than or equal to 4 for a MapReduce workflow. This leads to a constant factor approximation ratio in practice.

8.5.3.1 PARTITIONING TO SCHEDULE

To solve the problem, we use, as a subroutine, an algorithm to solve the classic k-balanced partitioning problem [Eve+99]. Given an integer $k \geq 2$ and a real $\nu \geq 1$, a (k, ν)-balanced partition of $G = (V, E)$ is a subset of the edges whose removal partitions the graph into at most k parts, where the sum of the vertex weights in each part is at most $\frac{\nu}{k} w(V)$. The (k, ν)-balanced partitioning problem with input $G = (V, E)$, k, and ν is to find a (k, ν)-balanced partition of G with minimum capacity, i.e., for which the sum of the weights of the arcs between parts is minimized. When $\nu = 2$, the problem is just called the k-balanced partitioning problem. Classic algorithms achieve an approximation factor of $O(\log n)$ to solve the problem [Eve+99; ST97]. The approximation algorithm with the best known approximation factor, $O(\sqrt{\log n \log k})$, is due to Krauthgamer et al. [KNS].

PARTITION works as follows. We consider the undirected version of the DAG of the workflow as input. The completion times of the network (resp. CPU) tasks correspond to the weights of the edges (resp. of the vertices). As we do not know in advance if the best partition for our problem is balanced (indeed, if the network delays are very long, it may be better to schedule all the tasks on a single machine), we systematically test different levels of balance.

The algorithm solves the k-balanced partitioning problem, for $1 \leq k \leq m$. It then outputs the best solution, that is, the one minimizing the sum of the weights of the arcs between parts divided by k (corresponding to the average work of the network machines) and the maximum partition size (corresponding to the work of the (CPU) machines).

In fact, first note that there exists an optimal partition using fewer than k machines when the maximum work over all machines is less or equal to $\frac{2n}{k} w(V)$. Indeed, if two machines have less than $\frac{n}{k} w(V)$ work to do, only

one among both machines may do all the tasks assigned to them, and the makespan may not be increased. Thus, only one machine may have less than $\frac{n}{k}w(V)$ work to do, and there may be only $k-1$ machines with more.

Theorem 3. PARTITION-ASSIGN provides a $O(\sqrt{\log n \log m})$ -approximation algorithm of the PARTITIONING TO SCHEDULE problem.

Proof. Let $S^* = W^*_{CPU} + W^*_N$ be an optimal solution of the PARTITIONING TO SCHEDULE problem, where W^*_{CPU} is the maximum work to be done on a machine and W^*_N is the network work.
There exists an integer k, with $1 \leq k \leq m$, such that $\frac{n}{k} \leq W^*_{CPU} \leq \frac{2n}{k}$. Indeed, $W^*_{CPU} \geq \frac{n}{m}$ as at least one machine of the m machines has to do more than $1/m$-th of the work and $W^*_{CPU} \leq n$, the total amount of work to be done.
Remark now that there exists an optimal partition using fewer than or exactly k machines when the maximum work over all machines is less than or equal to $\frac{2n}{k}w(V)$. Indeed, if two machines have less than $\frac{n}{k}w(V)$ work to do, only one among both machines may do all the tasks assigned to them, and the makespan may not be increased. Thus, only one machine may have less than $\frac{n}{k}w(V)$ work to do, and there may be only $k-1$ machines with more. Thus, without loss of generality, consider that S^* uses at most k machines.
Consider now the solution S_A provided by the k-balanced partitioning algorithm for this value of k. We have $S_A = \max\_part\_size + \text{cut\_weight}$, with cut_weight the capacity of its cut and $\max\_part\_size$ the maximum weight of a part.
On one hand, we have that cut_weight $\leq O(\sqrt{\log n \log k})W^*_N$. Indeed, S^* provides a solution of the k-balanced partitioning algorithm, as it uses at most k machines. On the other hand, we have that $\max\_part\_size \leq \frac{2n}{k}$. As $\frac{n}{k} \leq W^*_{CPU}$, we get that $\max\_part\_size \leq 2W^*_{CPU}$. This yields that $S_A \leq O(\sqrt{\log n \log m})S^*$. □

Algorithm 10 PARTITION

Input: Set of workflows G, m number of machines.
partitions[m] ▷ Solutions of m partitioning problems
for $k = 1, 2, \ldots, m$ **do** ▷ Iterate on number of processors
    partitions[k] ← Compute an approximate solution of the k-balanced partitioning problem for G.
end for
best_sol = $\min_k$(max_part_size($partitions[k]$) + cut_weight($partitions[k]$)/k)
return best_sol

As the algorithm of Krauthgamer et al. is based on semi–definite programming and has a long execution time, to solve the problem in practice we use the $O(\log n)$ approximation algorithm described in [ST97]. The main idea is to recursively partition the graph by solving, at each step, a Minimum Bisection Problem. We use the Kernighan and Lin heuristic algorithm [KL70] to solve bisection, leading to a time complexity of $O(mn^3 \log n)$.

8.5.3.2 SCHEDULING WHEN PLACED

Theorem 4. PARTITION-SCHEDULE provides a depth(W)-approximation algorithm of the SCHEDULING WHEN PLACED problem, where depth(W) is the depth of the workflow W to be scheduled.

Proof. PARTITION-SCHEDULE considers the tasks of the workflow layer by layer. It does not schedule a task of layer j if a task of layer i with $i < j$ can be scheduled.
Consider an optimal schedule S^* and let $C(S^*)$ be its makespan. We denote by $C(L_i)$ the time to process the tasks of layer i and by $C(L_i \to L_j)$ the time to process the network tasks between layers i and j. Clearly, $C(S^*) \geq C(L_i)$ for $1 \leq i \leq depth(W)$. Similarly, $C(S^*) \geq C(L_i \to L_{i+1})$ for $1 \leq i \leq depth(W) - 1$. Thus, the makespan of PARTITION-SCHEDULE, $c(A)$, is such that

$$c(A) \leq \sum_{i=1}^{depth(W)} C(L_i) + \sum_{i=1}^{depth(W)-1} C(L_i \to L_{i+1}).$$

We thus have $c(A) \leq (2\ depth(W) - 1)C(S^*)$. □

8.6 Experimental Evaluation

To validate our algorithm, we carried out some experiments using workflows built using statistics from the data center traces comprising 25 millions tasks released by Google [RWH11].
We compare the performances of our two proposed algorithms, G-List (its variants with and without selection of the master branch referred to as G-List-Master and G-List, respectively) and Partition, with the ones of List Scheduler [Ray87] which was proposed to handle communication delays, but which does not take into account the limited network capacity. We show the importance of taking into account the competition of tasks for bandwidth.

8.6.1 Trace

We extracted the distributions of the number of tasks per job and of the delay of computational tasks from the trace. The variances of the distributions are huge. Indeed, 75% of jobs have only 1 task, but these tasks only account for 20% of the total tasks. The average and maximum number of tasks of a workflow are 38 and 90,000, respectively. Also, the task completion time is heavy-tailed. The mean value is 28 minutes, but the longest task lasted 5 and a half days [LC12].

8.6.2 Network

The traces do not include statistics on network delays. This is why we tested different scenarios. To this end, we define the parameter ρ, which we refer to as *network factor* and which is the ratio between the average delay of a network task and the average delay of a CPU task. We then considered different values between 0% and 400% for ρ. We use $\rho = 0.5$ as the default value, as it corresponds to a scenario in which roughly 33% of the time is spent in network transfers [Cho+11].

8.6.3 Workflows

The dependencies between the tasks of a workflow are also not provided. We thus compare the algorithms using workflows of different types: simple MapReduce (defined in Section 8.5.2), 1-Stage MapReduce, and Random workflows. 1-Stage MapReduce workflows contain a map phase, a shuffle

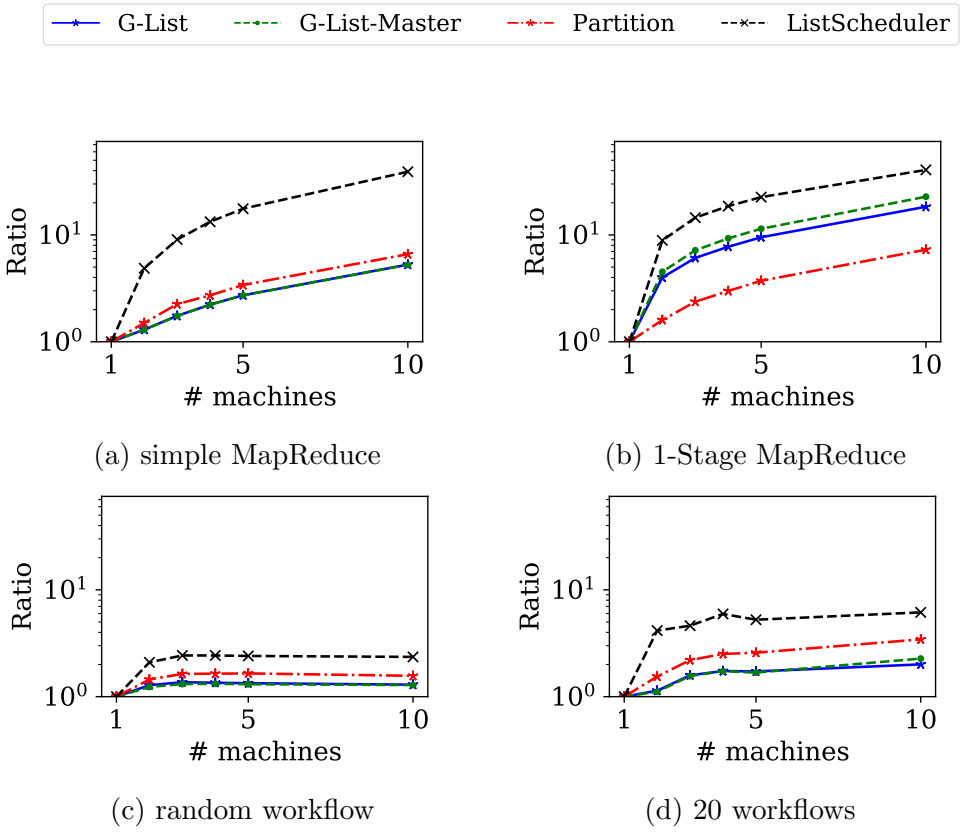
phase, and a reduce phase (see Fig. 8.4). For a given number of tasks, we randomly choose the proportion of tasks in the first layer and in the second layer. We then connect task s to all the tasks of the first layer, and all the tasks of the second layer to task d. Each task of the first layer is then connected to a task in the second layer. We then choose the edge density of the workflow, that is, a probability p that a given task in the second layer is dependent of a given task of the first layer. Random workflows are built in the following way: we order the tasks from T_1 to T_n. To avoid cycles, we only add an arc from T_i to T_j (with probability p) if $i < j$. We then check if the workflow is connected and accept it in this case, and generate another one if not. We tested different values for p.

8.6.4 Datasets

The datasets are then built in the following way. We choose a number of jobs to be executed. For each job, we choose its type of workflow randomly: simple MapReduce, 1-Stage MapReduce or Random workflow, with probabilities 20, 40, and 40%, respectively. It corresponds to a realistic distribution as at least 50% of applications in clusters can fit in the MapReduce paradigm [Ren+13]. We then draw its number of jobs randomly according to the distribution of the Google trace. The completion times of the (CPU) tasks (resp. of the network tasks) are then chosen according to the distribution of the Google trace (and multiplied by the network factor ρ). For each experiment, we average over 100 datasets.

8.6.5 Results

We compare the *makespan* of the schedules given by the different algorithms. We also study its sensitivity to different parameters such as the number of data center servers, number of tasks in a job, workflow edge density, and network factor. We provide two sets of results. The first ones are for a single workflow. The goal is to understand the efficiency of the algorithms for different types of workflows. The second ones are for sets of 20 random workflows of different types. Note that a single workflow may have up to 90,000 tasks. The goal is to see how efficient the algorithms are for a data center workflow. As the variance of the Google trace is very high, we present the results using a *normalized makespan metrics*, denoted as *ratio*. It is



(a) simple MapReduce

(b) 1-Stage MapReduce

(c) random workflow

(d) 20 workflows

Figure 8.5: Efficiency of the proposed algorithms as a function of the number of machines for different types of workflows.

defined as the ratio between the makespan provided by an algorithm and the best of two classical lower bounds: $\sum_{T_i \in \mathcal{T}} c_i/m$ and the completion time of the longest branch. This metric allows to normalize the makespan between workflows with very different task completion times. It also gives an idea of the cost of network communications as the lower bound does not take into account the completion time of network tasks.

8.6.5.1 Number of machines

We first vary the number of machines used to execute the workflows (Fig. 8.5). With one machine, all the algorithms have a ratio of 1 as all the tasks are executed on 1 machine and no network tasks need to be done. When the number of machines increases, the ratio increases as tasks are done on several machines in order to decrease the makespan. For simple MapReduce workflows, the ratio increases to 5 even though G-List is optimal. It means that this value corresponds to the cost of network communications (and not to a gap to optimality). Note that, for other types of workflows (1-Stage MapReduce, random), the ratio has similar or lower values, showing the efficiency of our algorithms.

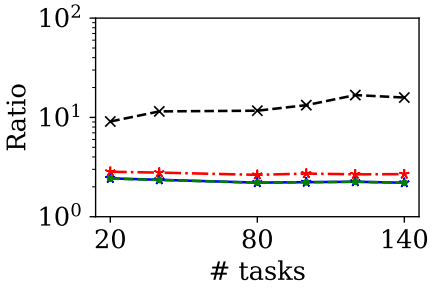
The gap between List Scheduler and our algorithms, G-List and Partition, also increases with the number of machines. This shows that our algorithms make much better use of the increased processing power available by optimizing the network communications. G-List provides the best solutions for simple MapReduce worfklows (Fig. 8.5a), as the algorithm is optimal for this type of workflow. However, Partition is also behaving well and provides close to optimal solutions. For 1-Stage MapReduce workflows (Fig. 8.5b), Partition is a lot more efficient than G-List as this type of workflow is more complex to schedule. On random workflows, both algorithms behave well. This is due to the fact that random workflows have DAGs with longer depths and that there are fewer possible scheduling combinations (see the following discussion for edge density). Indeed, the ratio is close to 1 in this case. On the sets of 20 workflows, the three algorithms behave similarly, with a small advantage for G-List-Master.

8.6.5.2 Size of the workflow

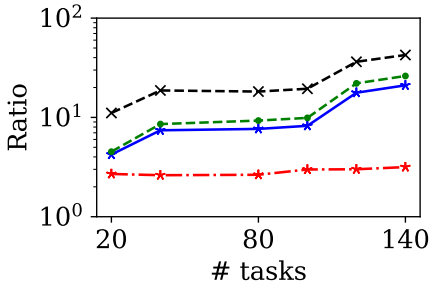
We observe similar results for this parameter (Fig. 8.6), G-List is a bit better on simple MapReduce, but Partition is significantly better on K2. For random workflows and the sets of 20 workflows, the algorithms perform similarly.

8.6.5.3 Edge density

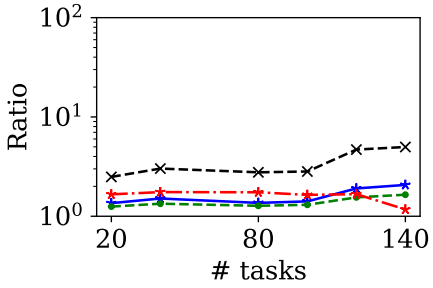
To understand for which kinds of workflows each algorithm is more efficient, we studied two parameters: edge density and network factor. We made the edge density vary from 0 to 1 (Fig. 8.7). With a small edge density, all



(a) simple MapReduce

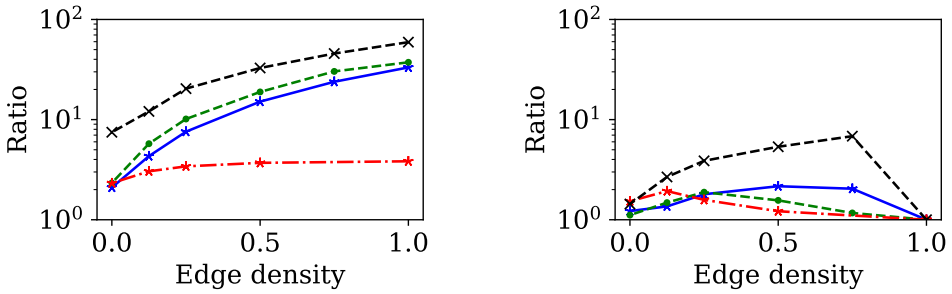


(b) 1-Stage MapReduce



(c) random workflow

Figure 8.6: #tasks per workflow.



(a) 1-Stage MapReduce (b) random workflows

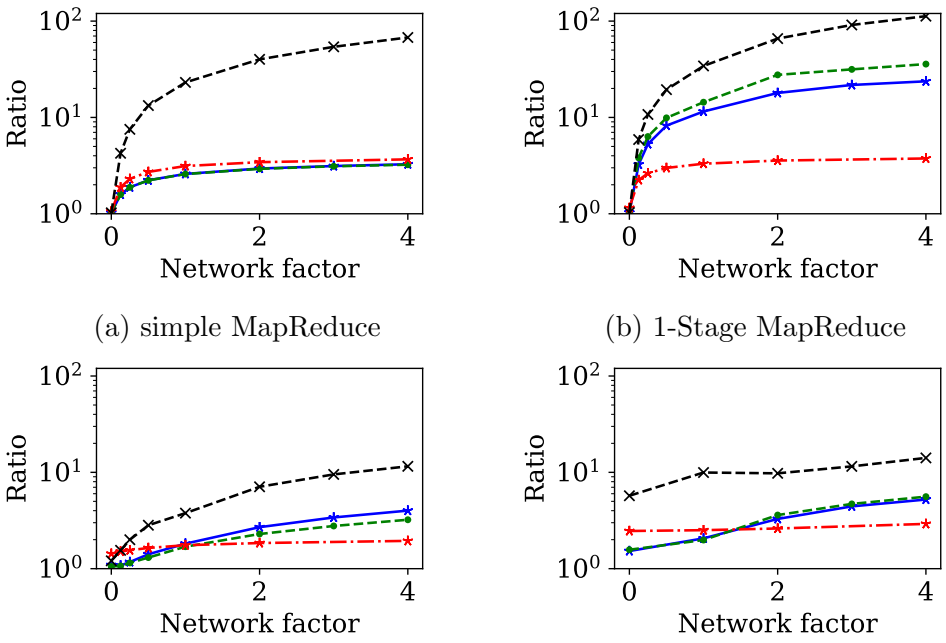
Figure 8.7: Efficiency of the proposed algorithms as a function of the workflow edge density.

algorithms behave well. The tasks are not very dependent on each other, and scheduling decisions are easy to take. When the edge density increases, scheduling becomes harder, and PARTITION behaves better as it considers the global structure of the dependency digraph, especially for 1-Stage MapReduce workflows (Fig. 8.7a). For random workflows (Fig. 8.7b), PARTITION is also better for edge densities higher than 0.2. However, all algorithms (including List Scheduler) behave well when the value of the edge density is 1. Indeed, in this case, there exists a complete order of the tasks, so all algorithms carry out the same schedule on a single machine. In general, random workflows tend to have a long branch. G-LIST-Master is executing all the tasks of this branch on the master machine and thus is the most effective for this type of workflow.

8.6.5.4 Network factor

We vary ρ from 0 to 4 (Fig. 8.8). When the network factor is zero, all algorithms are equivalent. Indeed, this corresponds to a scenario in which network capacity is not a limiting resource. In this case, only the CPU task placement has to be optimized. Then, when the completion times of the network tasks increase, our algorithms, as expected, perform better than List Scheduler. PARTITION is the most efficient for all except for simple MapReduce workflows.

To summarize, both algorithms behave well for different types of workflows and different sets of parameters. G-LIST is the best on simple MapRe-



(a) simple MapReduce

(b) 1-Stage MapReduce

(c) random workflow

(d) 20 workflows

Figure 8.8: Efficiency of the proposed algorithms for different network factors and types of workflows.

duce worfklows and its variant with Master branch is efficient on Random workflows. PARTITION, in general, is better when the workflows are more complex and when the network is a strong bottleneck. Data center operators should thus choose a solution based on their mix of applications and network capacity.

8.7 Conclusion

In this chapter, we proposed a new framework to model the orchestration of tasks in a datacenter for scenarios in which the network bandwidth is a limiting resource. We introduce a new problem, SCHEDULING WITH NETWORK TASKS, in which, along with traditional (CPU) tasks, network tasks have to be scheduled on network machines. We propose two algorithms to solve the problem, G-LIST and PARTITION, for which we derive some theoretical guarantees. We demonstrate their effectiveness using datasets built using statistics from Google data center traces [RWH11].
The paper focuses more on the theoretical side. An interesting future work may also concern the study of the practical behaviors of the algorithms on a testbed, comparing them with practical solutions proposed for data centers.

References

[Ahm+14] Faraz Ahmad, Srimat T Chakradhar, Anand Raghunathan, and TN Vijaykumar. "ShuffleWatcher: Shuffle-aware Scheduling in Multi-tenant MapReduce Clusters." In: *USENIX Annual Technical Conference*. 2014, pp. 1–12 (cit. on p. 178).

[CGC16] Tao Chen, Xiaofeng Gao, and Guihai Chen. "The features, hardware, and architectures of data center networks: A survey". In: *Journal of Parallel and Distributed Computing* 96 (2016), pp. 45–74 (cit. on p. 178).

[Cho+11] Mosharaf Chowdhury, Matei Zaharia, Justin Ma, Michael I Jordan, and Ion Stoica. "Managing data transfers in computer clusters with orchestra". In: *ACM SIGCOMM Computer Communication Review*. Vol. 41. 4. 2011 (cit. on pp. 13, 172, 174, 191).

[CKL] Fangfei Chen, Murali Kodialam, and TV Lakshman. "Joint scheduling of processing and shuffle phases in mapreduce systems". In: *IEEE INFOCOM 2012* (cit. on p. 175).

[CPW99] Bo Chen, Chris N Potts, and Gerhard J Woeginger. "A review of machine scheduling: Complexity, algorithms and approximability". In: *Handbook of combinatorial optimization*. Springer, 1999, pp. 1493–1641 (cit. on p. 175).

[CS12] Mosharaf Chowdhury and Ion Stoica. "Coflow: A networking abstraction for cluster applications". In: *ACM Workshop on Hot Topics in Networks*. 2012, pp. 31–36 (cit. on p. 174).

[CZS14] Mosharaf Chowdhury, Yuan Zhong, and Ion Stoica. "Efficient coflow scheduling with varys". In: *ACM SIGCOMM Computer Communication Review*. Vol. 44. 4. 2014, pp. 443–454 (cit. on pp. 172, 174).

[DG08] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters". In: *Communications of the ACM* 51.1 (2008) (cit. on p. 172).

[Dog+14] Fahad R Dogar, Thomas Karagiannis, Hitesh Ballani, and Antony Rowstron. "Decentralized task-aware scheduling for data center networks". In: *ACM SIGCOMM Computer Communication Review*. 2014, pp. 431–442 (cit. on pp. 172, 174).

[Eve+99] Guy Even, Joseph Naor, Satish Rao, and Baruch Schieber. "Fast approximate graph partitioning algorithms". In: *SIAM Journal on Computing* 28 (1999) (cit. on p. 188).

[Gir+19a] Frédéric Giroire, Nicolas Huin, Andrea Tomassilli, and Stéphane Pérennes. *When Network Matters: Data Center Scheduling with Network Tasks*. Tech. rep. Inria, Jan. 2019 (cit. on p. 186).

[Gir+19b] Frédéric Giroire, Nicolas Huin, Andrea Tomassilli, and Stéphane Pérennes. "When network matters: Data center scheduling with network tasks". In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. 2019 (cit. on pp. 13, 171).

[Gra66] Ronald L Graham. "Bounds for certain multiprocessing anomalies". In: *Bell System Technical Journal* 45.9 (1966), pp. 1563–1581 (cit. on pp. 175, 182, 183).

[Gre+09] Albert Greenberg, James R Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A Maltz, Parveen Patel, and Sudipta Sengupta. "VL2: a scalable and flexible data center network". In: *ACM SIGCOMM computer communication review*. Vol. 39. 4. 2009, pp. 51–62 (cit. on p. 172).

[Guo+08] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, and Songwu Lu. "Dcell: a scalable and fault-tolerant network structure for data centers". In: *ACM SIGCOMM Computer Communication Review*. Vol. 38. 4. 2008 (cit. on p. 172).

[GVY93] Naveen Garg, Vijay V Vazirani, and Mihalis Yannakakis. "Approximate max-flow min-(multi) cut theorems and their applications". In: *ACM symposium on Theory of computing*. 1993, pp. 698–707 (cit. on p. 179).

[Isa+07] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. "Dryad: distributed data-parallel programs from sequential building blocks". In: *ACM SIGOPS operating systems review*. Vol. 41. 3. ACM. 2007 (cit. on p. 172).

[Jal+15] Virajith Jalaparti, Peter Bodik, Ishai Menache, Sriram Rao, Konstantin Makarychev, and Matthew Caesar. "Network-aware scheduling for data-parallel jobs: Plan when you can". In: *ACM SIGCOMM Computer Communication Review*. Vol. 45. 4. 2015, pp. 407–420 (cit. on p. 175).

[KL70] Brian W Kernighan and Shen Lin. "An efficient heuristic procedure for partitioning graphs". In: *The Bell system technical journal* 49 (1970) (cit. on p. 190).

[KNS] Robert Krauthgamer, Joseph Naor, and Roy Schwartz. "Partitioning graphs into balanced components". In: *ACM-SIAM SODA 2009* (cit. on p. 188).

[LC12] Zitao Liu and Sangyeun Cho. "Characterizing machines and workloads on a Google cluster". In: *IEEE Parallel Processing Workshops (ICPPW)*. 2012 (cit. on p. 191).

[MD79] R Garey Michael and S Johnson David. "Computers and intractability: a guide to the theory of NP-completeness". In: *WH Free. Co., San Fr* (1979) (cit. on p. 185).

[MQS98] Alix Munier, Maurice Queyranne, and Andreas S. Schulz. "Approximation Bounds for a General Class of Precedence Constrained Parallel Machine Scheduling Problems". In: *Lecture Notes in Computer Science* (1998), pp. 367–382. ISSN: 0302-9743. DOI: 10.1007/3-540-69346-7_28 (cit. on p. 175).

[Mur+11] Derek G Murray, Malte Schwarzkopf, Christopher Smowton, Steven Smith, Anil Madhavapeddy, and Steven Hand. "CIEL: a universal execution engine for distributed data-flow computing". In: *Proc. 8th ACM/USENIX Symposium on Networked Systems Design and Implementation*. 2011, pp. 113–126 (cit. on p. 172).

[NS14] Dmitry Namiot and Manfred Sneps-Sneppe. "On micro-services architecture". In: *International Journal of Open Information Technologies* 2.9 (2014), pp. 24–27 (cit. on p. 172).

[PY90] Christos H Papadimitriou and Mihalis Yannakakis. "Towards an architecture-independent analysis of parallel algorithms". In: *SIAM journal on computing* 19.2 (1990), pp. 322–328 (cit. on p. 175).

[Ray87] Victor J Rayward-Smith. "UET scheduling with unit interprocessor communication delays". In: *Discrete Applied Mathematics* 18.1 (1987) (cit. on pp. 175, 179, 182, 183, 191).

[Ren+13] Kai Ren, YongChul Kwon, Magdalena Balazinska, and Bill Howe. "Hadoop's adolescence: an analysis of Hadoop usage in scientific workloads". In: *Proceedings of the VLDB Endowment* 6.10 (2013), pp. 853–864 (cit. on pp. 184, 192).

[RWH11] Charles Reiss, John Wilkes, and Joseph L Hellerstein. "Google cluster-usage traces: format+ schema". In: *Google Inc., White Paper* (2011), pp. 1–14 (cit. on pp. 174, 191, 198).

[ST97] Horst D Simon and Shang-Hua Teng. "How good is recursive bisection?" In: *SIAM Journal on Scientific Computing* 18.5 (1997), pp. 1436–1445 (cit. on pp. 188, 190).

[Tho+11] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song. "Design and evaluation of a real-time url spam filtering service". In: *IEEE Symposium on Security and Privacy (SP)*. 2011, pp. 447–462 (cit. on pp. 13, 172).

[Zah+12] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing". In: *USENIX conference on Networked Systems Design and Implementation*. 2012 (cit. on p. 172).

Chapter 9

Path Protection in Elastical Optical Networks

Contents

The content of this chapter has been published before in [TJG18].

9.1 Introduction

In an Elastic Optical Network (EON), data is distributed over a number of low data rate subcarriers without having to strictly follow the ITU-T fixed wavelength grid. In this way, with a data traffic more and more uncertain and heterogeneous, the spectrum resources can be used more efficiently and with a higher degree of flexibility [Jin+09].
With respect to a classical WDM network, EONs impose additional constraints on the structure of the optical path. Indeed, EONs require that contiguous frequency slots are allocated to each connection, which is also the main difference between the Routing and Spectrum Assignment (RSA) and

Routing and Wavelength Assignment (RWA) problems. Thus, the already proposed RWA methods are not suitable for EONs.
The RSA problem requires to find both an end-to-end optical path and a contiguous subset of frequency slots for each connection request.
Furthermore, EONs open up the possibility of exploiting multiple modulation formats for the different subcarriers. In such a way, the utilization efficiency could be further enhanced [Jin+10]. The problem of also determining a modulation format in addition to a routing path and a contiguous segment of spectrum is often referred to as the Routing, Modulation, and Spectrum Allocation (RMSA) problem. The problem is known to be NP-Hard even in the absence of modulation formats [KW11] and is challenging, even on small instances.
With the increasing efficiency in terms of resource usage, a link may accommodate a larger number of connections in EONs. Hence, the effects of a failure, such as a fiber cut, could be even more disruptive than in traditional networks. Network failures have been widely investigated (see e.g., [Tur+10; Ian+02]). In the results of [Tur+10], each link experienced, on average, 16 failures per year. If not well managed, a failure may correspond to loss of service to users and loss of revenue. It is thus necessary to provide protection against failures in order to guarantee continuity of service and no violation of SLA requirements. We focus our attention on the *single link failure* scenario, since they are the predominant form of failures in optical networks [RSM03].
Fault management techniques can be grouped into two categories: *restoration* and *protection*. In restoration, the network spare resources are used to reroute the connections affected by the failure. In protection, spare capacity is reserved in advance during connection setup. Restoration schemes use network spare resources more efficiently, but on the other hand, protection schemes have a faster restoration time and guarantee the recovery [SRM02]. We thus study the latter schema.
In *dedicated protection*, there is no spectrum resources sharing between backup lightpaths. Each frequency slot is used for at most one lightpath. In *shared protection*, backup spectrum resources can be shared among different lightpaths if they fail independently. If, on one hand, in shared protection, spectrum resources are used more efficiently [ZS00], on the other hand, in dedicated protection the recovery time is smaller. We thus study both protection schemes in this paper.
Another classification of the protection techniques can be made according to the recovery mechanisms. It could consist in a local repair (i.e., *link*

protection) or in an end-to-end repair (i.e., *path protection*). Link protection schemes reroute the traffic only around the failed link. Path protection schemes reroute the traffic through a backup path if a failure occurs on its working path. With path protection, network resources are used more efficiently [RSM03].
We consider the problem of *providing for each connection, a link-disjoint backup lightpath, under both dedicated and shared path protection schemes.* Our model also includes practical parameters such as the modulation format selection and the positions of regenerators. The modulation format of a lightpath adds a constraint on the maximum transmission distance, which may be extended by one or more regenerators if present in the route. One of the key concerns of the network operators' is the efficient utilization of the deployed network capacity [Jin+09]. Our optimization goal is thus the minimization of the spectrum requirements for the protection.
In this paper, we propose two models for both dedicated and shared path protection against a single link failure. Our resolution strategy is based on a decomposition model using the column generation technique. We show that this technique is effective in dealing with the RMSA problem.
Our contributions can be summarized as follows:

- To the best of our knowledge, we are the first to propose a *scalable exact method* to solve the problem of providing path protection against a single link failure in elastic optical networks. The method is based on a decomposition model using column generation.

- The model takes into account practical constraints, such as multiple modulation formats, regenerators, and shared risk link groups.

- We compare the shared and dedicated path protection models and evaluate the tradeoff between the resolution time and the effectiveness, in terms of bandwidth utilization.

- We additionally study the impact of the number of regenerators in the network on the bandwidth requirements and on the latencies of both primary and backup lightpaths.

The rest of this paper is organized as follows. In Section 9.2, we review related works in more detail. In Section 9.3, we formally state the problem addressed in this paper. In section 9.4, we describe our column-generation-based model

and show the subproblem to be solved in Section 9.5. In Section 9.6, we validate our model by various numerical results on two real world topologies of different sizes. Finally, we draw our conclusions in Section 9.7.

9.2 Related Work

The problem of providing protection against failures in WDM networks has been widely investigated in the literature, see e.g., [RSM03], [SRM02], [ZS00]. Nevertheless, not enough effort has been made in the context of EONs with multiple modulation formats and flexible spectrum allocation.

Dedicated path protection. The problem of off-line routing and spectrum allocation in flexible grid optical networks with dedicated path protection was studied in [KW12] and [Kli13b]. The optimization goal considered is to minimize the width of spectrum required in the network. In [KW12], the authors provide both an ILP formulation and a heuristic algorithm to solve the problem. In [Kli13b], an evolutionary algorithm metaheuristic is proposed with the aim to support the search for optimal solutions.

Shared path protection. Shared protection for EONs was considered in [Kli13a], [SWB14], and [WK13]. A genetic algorithm metaheuristic with the goal to provide near optimal solutions to the problem of finding a primary and a backup path for each demand is proposed in [Kli13a]. The closest works to ours are [SWB14] and [WK13]. The authors consider exact methods and propose ILP formulations for both dedicated and shared path protection, but with different optimization objectives. In [SWB14], the authors minimize both the required spare capacity and the maximum number of frequency slots used in the network. In [WK13], the objective is to minimize the width of spectrum required in the network. They propose an ILP formulation in which each demand has a set of candidate pairs of link disjoint routing paths. The ILP model is able to deal with small networks (up to 9 nodes and 26 links). For larger networks, they propose heuristic algorithms based on both jointly and separated assignment of lightpaths to the demands.

Model Scalability. Previous works highlight the fact that finding an optimal or a near-optimal solution to the problem of jointly computing both a primary and a backup path for each demand is a challenging task, even for networks of small sizes and for a small number of demands. For instance, in [WK13] the authors show the benefits in terms of computing time and

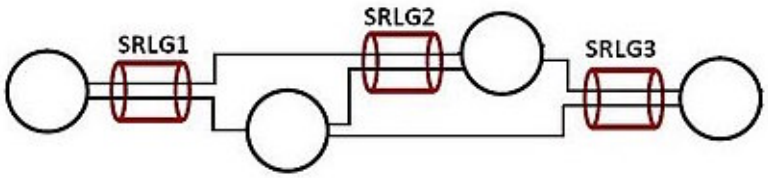


Figure 9.1: An example of SRLG Constraints

accuracy of computing the set of backup paths after the primary path allocation. In order to be able to deal with larger datasets, we adopt a two phase approach. First, we find a working path for each demand, then a backup path under both dedicated and shared protection schemes. We use the column generation technique as a solution approach, as results from [Rui+13] evidence the effectiveness of the column generation techniques in obtaining solutions for large instances of the RSA problem (but they do not consider protection against failures). Exact models proposed in the literature are only able to deal with small networks. On the contrary, our model is more scalable and we are able to solve instances with 24 nodes, 43 links, and 120 traffic requests. Moreover, we take into account regenerators and choices of modulation formats, which are not considered in the exact models of the literature.

9.3 Statement of the RMSA Protection Problem

The RMSA problem assumes an undirected graph $G = (V, L)$ with optical node set V and link set L. We denote by $\omega(v)$ the set of links adjacent to v, for $v \in V$. The bandwidth is slotted into a set S of frequency slots. The traffic is defined by a set K of requests where each request $k \in K$ has a source (s_k), a destination (d_k), and a spectrum demand D_k, expressed in terms of a number of frequency slots. The traffic is assumed to be symmetrical.

The provisioning of the primary lightpaths is given, and we are interested in finding both a dedicated and a shared path protection with minimum spectrum requirements, satisfying the spectrum contiguity and continuity constraints, as well as the following constraints:

- *Shared Risk Link Group (SRLG) constraints*, see Figure 9.1. Each SRLG constraint is defined by a set of links sharing a common resource, which affects all links in the set if the common resource fails. In the context of optical networks, it refers to a bundle of fiber links going through the same duct and that cannot be used simultaneously for primary and backup provisioning of the same demand. Let $\mathcal{F}$ be the set of all SRLG sets: $\mathcal{F} = \{F$: if ℓ and ℓ' both belong to F, then ℓ cannot be used in a path protecting ℓ' and vice versa $\}$.

- *Modulation constraints* The modulation format can be selected according to the traffic demand and the distance. We consider four modulation formats: BPSK (1 bit per symbol), QPSK (2 bits per symbol), 8QAM (3 bits per symbol), and 16QAM (4 bits per symbol) [CTV11]. For instance, if, for a demand k, we have a request of 250 Gb/s (i.e., $D_k = 20$ assuming the bandwidth of a subcarrier slot as 12.5 GHz), then with BPSK $D_k^{BSPK} = 20$ and with 16QAM, $D_k^{16QAM} = 5$. We consider the following maximum transmission distances: BPSK (9,600 Km), QPSK (4,800 Km), 8QAM (2,400 Km), and 16QAM (1,200 Km). These values are based on the experimental results reported in [Boc+]. Moreover, we assume that a subset of the nodes have regeneration capabilities. Indeed, decisions about the required equipment (i.e., transponders, regenerators, and switches) and its deployment are taken during the planning phase [Kre+14].

9.4 Path Protection Models

We propose two column generation models relying on lightpath configurations for both dedicated and shared path protection schemes. In the rest of the paper, the two models will be referred to, respectively, as CG_DP and CG_-SP.
A lightpath configuration, denoted by π, refers to a backup lightpath, i.e., a backup path, a spectrum slice with s as a starting frequency slot and a modulation format. Denote by Π the set of all possible backup lightpath configurations. Π is decomposed as follows:

$$\Pi = \bigcup_{k \in K} \Pi_k = \bigcup_{k \in K} \bigcup_{s \in S} \Pi_{ks},$$

where Π_k is the set of potentials lightpaths for provisioning request k, and Π_{ks} is the set of potential lightpaths for provisioning request k with a slot slice of width D_k^m according to the selected modulation format m such that s is a starting slot. Note that Π_k contains only *feasible backup lightpaths* for a demand k. We say that a backup lightpath is feasible for k if it does not contain any link in the same shared risk link group of some link of the primary lightpath for k. Each lightpath configuration, or lightpath for short, is denoted by π and is characterized by:
$b_{\ell s}^{\pi}$: indicates if slot s is used on link ℓ in the backup lightpath associated with π.
We assume that working lightpaths are known and described throughput the following parameter:
a_{ℓ}^{k}: indicates if the primary lightpath of request k goes through link ℓ.
The model uses the following decision variables:
$z_\pi = 1$ if lightpath $\pi \in \Pi$ is selected as a backup path, 0 otherwise.
$x_{\ell s} = 1$ if slot s is used on link ℓ in a backup path, 0 otherwise.
We denote with $\mathcal{L}S^{\text{B}}$ the pairs $(\ell, s) \mid \ell \in L, s \in S$ that can be used for protection, i.e., that are not used by the primary lightpaths.
The objective minimizes the spectrum requirements for the protection, and is written as follows:

$$\min \quad \sum_{(\ell,s)\in\mathcal{L}S^{\text{B}}} x_{\ell s} \tag{9.1}$$

Constraints are as follows:

$$\sum_{\pi\in\Pi_k} z_\pi \geq 1 \qquad k \in K \tag{9.2}$$

$$z_\pi \in \{0,1\} \qquad \pi \in \Pi \tag{9.3}$$

$$x_{\ell s} \in \{0,1\} \qquad \ell \in L, s \in S \tag{9.4}$$

Model CG_DP

$$\sum_{k\in K}\sum_{\pi\in\Pi_k} b_{\ell s}^{\pi} z_\pi \leq x_{\ell s} \qquad \ell \in L, s \in S, (\ell, s) \in \mathcal{L}S^{\text{B}} \tag{9.5}$$

Model CG_SP

$$\sum_{k\in K} a_{\ell'}^{k} \sum_{\pi\in\Pi_k} b_{\ell s}^{\pi} z_\pi \leq x_{\ell s} \qquad \ell, \ell' \in L, s \in S$$

$$\{\ell, \ell'\} \not\subseteq F : F \in \mathcal{F}, \ell \neq \ell', (\ell, s) \in \mathcal{L}S^{\text{B}} \tag{9.6}$$

Constraint (9.2) ensures that each request is protected. Constraints (9.5) and (9.6) make sure that each slot is never used more than once on each backup fiber link. The difference between the two models relies on these constraints. In the dedicated protection case, two working paths cannot have backup paths going through the same link ℓ and slot s. On the other hand, in the shared protection case, two working paths that are not sharing any link ℓ' can use protection paths going through the same link ℓ and slot s.

9.5 Solution Design

Given the huge number of variables/columns in the proposed model, we resort to the *Column Generation* method to solve its Linear Programming (LP) relaxation. This technique consists of decomposing the original problem into a restricted master problem - RMP - (i.e., model (9.1) - (9.6) with a very restricted number of variables) and one or several pricing problems - PPs. RMP and PPs are solved alternately. Solving RMP consists in selecting the best lightpaths, while solving one PP allows the generation of an improving potential lightpath, i.e., a lightpath such that, if added to the current RMP, improves the optimal value of its LP relaxation. The process continues until the optimality condition is satisfied, that is, the so-called reduced cost that defines the objective function of the pricing problems is non negative for all of them. An ε-optimal solution for the RSA problem is derived by solving exactly the ILP model associated with the last RMP.
Let K_σ denote the set of requests that have the potential to be protected by a lightpath starting at slot σ: $K_\sigma = \{k \in K : \sigma + D_k - 1 \leq |S|\}$. Let D_k^σ be the number of slots needed for request k in K_σ: $D_k^\sigma = D_k$ for $k \in K_\sigma$: $\sigma + D_k - 1 = |S|$ and $D_k^\sigma = D_k + 1$ for $k \in K_\sigma$: $\sigma + D_k - 1 < |S|$.
Each pricing problem is indexed by a demand k and a starting slot σ, and produces a single potential lightpath for protecting demand k, starting at slot σ.
Definitions of the decision variables are as follows:
$y_\ell = 1$ if link ℓ is used, 0 otherwise
$x_{\ell s}$ indicates if slot s is used on link ℓ or not.
We first describe the model for shared protection. Let $u_k^{(9.2)}$ and $u_{\ell\ell' s}^{(9.6)}$ be the values of the dual variables associated with constraints (9.2) and (9.6),

respectively. The pricing problem can be written as follows:

$$\min \quad 0 \quad - \quad u_k^{(9.2)} \quad - \quad \sum_{(s,\ell)\in S\times L} \quad \sum_{\substack{\ell'\in L:\\ \ell\neq\ell'}} u_{\ell\ell' s}^{(9.6)} \quad a_{\ell'}^k \quad x_{\ell s} \tag{9.7}$$

subject to:

$$\sum_{\ell\in\omega(s_k)} y_\ell = \sum_{\ell\in\omega(d_k)} y_\ell = 1 \tag{9.8}$$

$$\sum_{\ell\in\omega(v)} y_\ell \leq 2 \qquad\qquad v \in V \setminus \{s_k, d_k\} \tag{9.9}$$

$$\sum_{\ell'\in\omega(v)\setminus\{\ell\}} y_{\ell'} \geq y_\ell \quad v \in V \setminus \{s_k, d_k\}, \ell \in \omega(v) \tag{9.10}$$

$$\sum_{s=\sigma}^{\sigma+D_k^\sigma-1} x_{\ell s} = D_k^\sigma \, y_\ell \ell \in L \tag{9.11}$$

$$y_\ell, x_{\ell s} \in \{0,1\} \qquad \ell \in L, s \in S. \tag{9.12}$$

Constraints (9.8), (9.9) and (9.10) define the routing of the current request. Constraint (9.11) reserves a contiguous spectrum channel for the current request.
We observe that for each link ℓ:
$x_{\ell s} = y_\ell$ for $s \in \{\sigma, \ldots, \sigma + D_k^\sigma - 1\}$
$x_{\ell s} = 0$ for $s \notin \{\sigma, \ldots, \sigma + D_k^\sigma - 1\}$.
Therefore, the reduced cost can be rewritten:

$$\min \quad 0 - u_k^{(9.2)} - \sum_{\ell\in L} \left(\sum_{\substack{\ell'\in L:\\ \ell\neq\ell'}} \sum_{s=\sigma}^{\sigma+D_k^\sigma-1} u_{\ell\ell' s}^{(9.6)} \right) y_\ell.$$

The first term is a constant for each request, and the second term corresponds to a summation over the links of the network. Therefore, we can solve the pricing problem using the following objective function:

$$\min \quad -\sum_{\ell\in L} \left(\sum_{\substack{\ell'\in L:\\ \ell\neq\ell'}} \sum_{s=\sigma}^{\sigma+D_k^\sigma-1} u_{\ell\ell' s}^{(9.6)} \right) y_\ell.$$

where $u^{(9.6)}_{\ell\ell' s}$ are non-positive dual values. We conclude that, for each request k, the lightpath generator corresponds to a weighted shortest-path problem with link weight: $-\sum_{\ell' \in L:\ell \neq \ell'} \sum_{s=\sigma}^{\sigma+D^{\sigma}_{k}-1} u^{(9.6)}_{\ell\ell' s}$. As a result, the pricing problem when modulation and regenerators are not taken into account can be solved with a polynomial time algorithm, e.g., Dijkstra's algorithm.

In the dedicated protection case, the only difference lies in the objective function of the pricing problem, defined as:

$$\min \quad 0 - u^{(9.2)}_{k} - \sum_{(s,\ell)\in S\times L} u^{(9.5)}_{\ell s}\, x_{\ell s}$$

where $u^{(9.2)}_{k}$ and $u^{(9.5)}_{\ell s}$ are the values of the dual variables associated with constraints (9.2) and (9.5), respectively. As with the shared protection case, the problem can be reduced to finding a shortest path in a weighted graph.

Additional Modulation and Regenerators Constraints. However, if modulation is taken into account, we need to consider the maximum transmission distance constraint according to the considered modulation format. Also, a regenerator may extend the maximum reachable distance with respect to the chosen modulation format.

Each pricing problem is now indexed by a demand k, a starting slot σ, and a modulation format m, and produces a single potential lightpath for protecting demand k, starting at slot σ, if such a lightpath exists. In fact, some demands may not be satisfied, since the reachable distance is not long enough to reach the destination from the source, even in the presence of regenerators. Regenerators add an additional layer of complexity to the problem. Indeed, without regenerators, for a demand (s, t), we could only consider to solve the subproblem for the modulation formats whose transmission reach is greater or equal to the length of the shortest path between s and t. With the presence of regenerators, this consideration does not apply, since the transmission reach may be increased.

When considering modulation constraints and nodes with regenerator capabilities, the pricing problem becomes a *Minimum-Weight Path Problem with a constraint on the path length*. The Minimum-Weight Constrained Path Problem is proven to be NP-Hard [GJ02]. The problem has been widely studied and efficient algorithms have been proposed (see [ID05] for a survey on the subject).

Our solving strategy is described as follows. Pricing problems are solved us-

ing a modified version of the Label-setting algorithm for the Shortest Path Problem with Resource Constraints [ID05] based on the dynamic programming approach.
Given a weighted graph $G = (V, E)$, a demand (s, t), the maximum transmission distance according to the selected modulation format, and a set of nodes with regenerator capabilities $V_r \in V$, the algorithm starts from the trivial path $P = (s)$. It is then extended in all the feasible directions considering both the length of the links and the remaining transmission distance from the source s, which may have been increased because of the presence of one or more nodes in the set V_r in the considered path. For each path extension $P' \supset P$, a dominance algorithm is used in order to maintain only a Pareto-optimal set of paths or paths which can be extended to a Pareto-optimal one. When there are no more labels to be processed, the algorithm stops. A solution of minimum cost is selected from the set of all computed paths.

9.6 Numerical Results

In this section, we evaluate the accuracy and performance of the proposed models through simulation on two networks of different sizes and according to different types of metrics. The results indicate that our models perform well, with an accuracy better than 1% for CG_DP and 20% for CG_SP in the considered networks. We also compare the performance of the dedicated and shared protection schemes, and show the tradeoff between the time needed to find a solution to the problem in the two cases and the savings in terms of bandwidth overhead.

9.6.1 Data Sets

We conduct experiments on two network topologies: `nobel-US` (14 nodes, 21 links) from SNDlib [Orl+10b], and `USnet` (24 nodes, 43 links) from [Muk06]. For `nobel-US`, the length of each link is calculated using the GPS coordinates of the nodes, according to the Cosine-Haversine formula. We assume that there is one pair of bidirectional fibers on each link, and the available spectrum width of each fiber is set to be 2000 GHz. We set the bandwidth of a subcarrier slot to 12.5 GHz. We considered four modulation formats: BPSK (binary phase-shift keying), QPSK (quadrature phase-shift keying), 8QAM (8-quadrature amplitude modulation), and 16QAM (16-quadrature

Transmission Reach of BPSK (M=1)	9,600 Km
Transmission Reach of QPSK (M=2)	4,800 Km
Transmission Reach of 8-QAM (M=3)	2,400 Km
Transmission Reach of 16-QAM (M=4)	1,200 Km
Bandwidth of a frequency slot	12.5 GHz
Capacity of a frequency slot (with M=1)	12.5 Gb/s
Number of frequency slots per link	160

Table 9.1: Simulation Parameters

.

Network	# traffic requests	# slots primary lightpaths	# generated columns		z_{LP}		$\tilde{z}_{\text{ILP}}$	
			CG_DP	CG_SP	CG_DP	CG_SP	CG_DP	CG_SP
nobel-US	20	164	8,735	12,875	292	171.05	292	201
	40	273	15,190	21,744	546	237.1	546	290
	60	457	19,128	28,316	816	328.82	816	430
USnet	40	344	26,828	40,931	574	339.6	574	431
	80	856	39,514	67,936	1,278	557.37	1,278	713
	120	1138	46,938	80,495	1,790	835.55	1,790	1,021

Table 9.2: Numerical results for CG_DP and CG_SP.

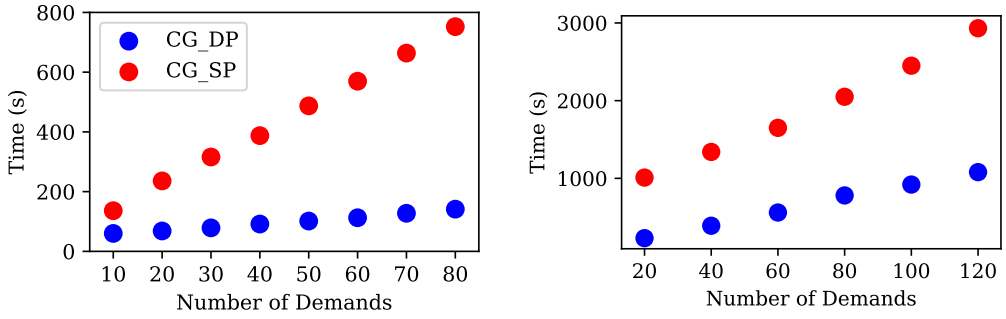
amplitude modulation). Similarly, as in [Zhu+13], we assume transmission distances of 9,600 km for BPSK (M = 1), 4,800 km for QPSK (M = 2), 2,400 km for 8QAM (M = 3), and 1,200 km for 16QAM (M = 4), where M denotes the number of bits per symbol. The number of considered nodes with regenerator capabilities is 5 for `nobel-US` and 10 for `USnet`. Locations are chosen according to the *betweenness centrality*, an index of the importance of an element in the network. It measures the extent to which a node lies on paths between other nodes. The length of each link is calculated using the GPS coordinates of the nodes, according to the Cosine-Haversine formula. Primary paths are computed with the objective of minimizing the total number of used frequency slots in the network. All experiments are run on an Intel Xeon E5520 with 24GB of RAM. The simulation parameters are summarized in Table 9.1.

9.6.2 Performance of CG Models

Table 9.2 summarizes the results of the two decomposition models for dedicated and shared protection on the two considered networks. We considered different numbers of demands. The load of each demand is randomly selected according to a uniform distribution within $50 - 200$ Gb/s.
A first difference can be observed in the number of generated columns, revealing the different level of complexity of the two models. This has an impact on the completion time, as can be observed in Figure 9.2. The large number of generated columns is also a consequence of our solving strategy. In fact, in order to accelerate the time needed to solve the RMP and to find an ILP solution to the last RMP, at each iteration, we remove nonbasic columns from the master problem according to their marginal cost. Thus, the number of iterations increases but, on the other hand, the time needed to find a solution decreases.
Another difference between the two models is the quality of the solution. CG_DP may require twice the number of frequency slots than CG_SP. This is a natural consequence of the different protection strategies. Moreover, the two models exhibit a different level of accuracy as expressed by the ratio of $(\tilde{z}_{\text{ILP}} - z_{\text{LP}})/z_{\text{LP}}$. In the case of CG_DP, it never exceeds 1%, while, for CG_SP, it may go up to 20%. The main reason for the difference in accuracy of the two models is the following. In CG_DP, to reduce the spectrum usage, the goal is to try to use short paths. This leads to fractional solutions with a small number of paths (and often a single one) for each demand. On the contrary, in CG_SP, the goal is to share backup paths as much as possible in order to reduce the value of the objective function. This leads to a large number of fractional paths per demand (sharing frequency slots with backup paths of several other demands) in the optimal fractional solution. The last RMP thus contains a large number of path variables with a nonzero value (often < 0.1) for each demand. Only one of them will be set to 1 per demand, when solving the last RMP as an ILP, leading to a larger gap.

9.6.3 Shared vs. Dedicated Path Protection

We now compare the performances of the two protection schemes. In Figure 9.3, we study the impact of the number of demands on the resources required by the two protection schemes. We keep the total traffic intensity constant and vary the number of demands. The traffic is set to be 10 Tbps

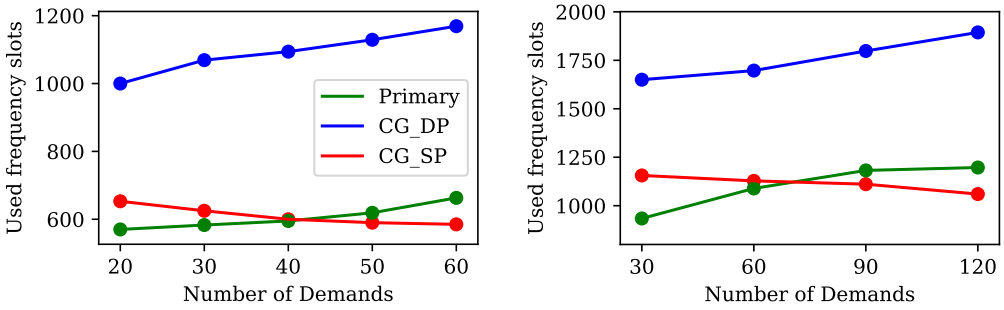


(a) nobel-US (b) USnet

Figure 9.2: Average completion time as a function of the number of demands

on `nobel-US` and 15 Tbps on `USnet`. As the results indicate, the two protection schemes exhibit a very different behavior. As the number of demands increases, the performance of the shared protection scheme, defined in terms of used frequency slots improves. On the other hand, both the primary lightpaths and the backup lightpaths computed according to the dedicated protection scheme, tend to require more resources as the number of demands becomes larger. This is not surprising, since an increasing number of demands improves the frequency slots' sharing opportunities of the lightpaths. In fact, in the shared protection scheme two link-disjoint primary lightpaths may share frequency slots in their backup paths. The benefits of shared over dedicated path protection is about 20% and 40% in the two networks according to the number of demands. Indeed, the benefits tend to increase with the number of considered demands. These results are similar to the ones reported by [SWB14] and [WK13].

Regenerators and Modulation Formats. Since, in optical networks, regenerators are costly, we are interested in evaluating the impact of the number of regenerators on the lightpaths. In Figures 9.4 and 9.5, we study the impact of the number of regenerators on the paths' latencies and on the spectrum requirements for the protection. We consider 50 demands for `nobel-US` and 100 demands on `USnet`. As the number of nodes with regeneration capabilities increases, from 0 to 10 for `nobel-US` and from 5 to 15 for `USnet` (Fig. 9.5), the spectrum requirements of the primary lightpaths and of the backup lightpaths decrease in both protection schemes. The reason is that a

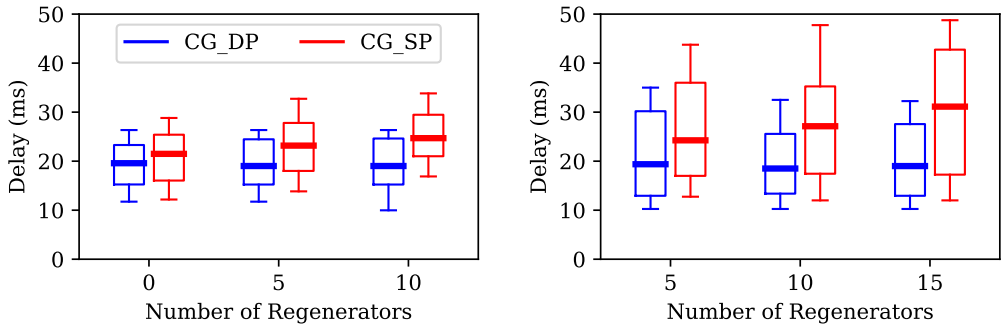


(a) nobel-US (b) USnet

Figure 9.3: Average number of frequency slots used as a function of the number of demands

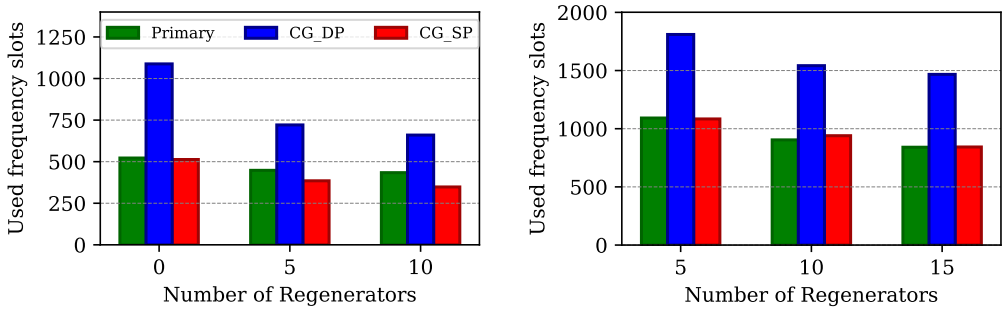
higher number of regenerators allows the lightpaths to use better modulation formats (in terms of bits per symbol) and consequently to use fewer resources. However, when considering lightpaths' latencies, the two protection schemes behave surprisingly in a strikingly different way. While, in the dedicated protection case, backup lightpaths' latencies tend to decrease, in the shared protection case, we observe the reverse phenomena. The explanation is the following. In dedicated protection, backup paths cannot be shared and, thus, the only means to reduce the number of used frequency slots is to use shorter paths. This is what happens when increasing the number of regenerators. Indeed, both primary and backup lightpaths need fewer resources, as they may now use more efficient modulation formats. This leads to increased spare capacity, allowing backup paths to use shorter routes. In shared protection, the situation is different. Indeed, there are *two* ways to reduce the spectrum usage: shorter paths as for DP, but also increased sharing of backup paths. The second way happens to be predominant in our experiments: regenerators allow better modulation formats and longer routes, leading to better sharing opportunities. As a consequence, the spectrum requirements are reduced, but this comes at the cost of increased lightpath lengths. However, the maximum delay of the backup paths in the shared protection case never exceeds 50 ms, the value often chosen as the maximum allowed delay for a route in networks [Gir+03]. As the results indicate, particular attention should be paid to lightpaths' latencies when considering shared path protection, in order not to violate the SLA requirements. Indeed, with the spectrum resources as optimization task, the possibility to share resources may lead to

longer paths at the expense of the delay. Note that we could also easily add a constraint in the pricing problem in order to consider only lightpaths under a certain delay requirement.



(a) nobel-US (b) USnet

Figure 9.4: Path delay distributions under the two protection schemes vs. the number of regenerators.



(a) nobel-US (b) USnet

Figure 9.5: Average number of frequency slots used as a function of the number of regenerators

9.7 Conclusion

In this chapter, we investigated the problem of providing path protection against a single link failure in elastic optical networks. We presented two decomposition models for both dedicated and shared path protection schemes

taking into consideration modulation, regenerators, and shared risk link group constraints. Through extensive simulation, we showed the effectiveness of our models in finding a solution in a reasonable amount of time. Moreover, we studied different metrics in order to compare the accuracy of those models, showing the tradeoff in terms of required bandwidth and latency with the time resources needed by the two protection schemes. Our future works include the further improvement of the model precision and scalability, in order to be able to deal with larger and more complex instances of the problem.

References

[Boc+] Adriana Bocoi, Matthias Schuster, Franz Rambach, Moritz Kiese, Christian-Alexander Bunge, and Bernhard Spinnler. "Reach-dependent capacity in optical networks enabled by OFDM". In: *Proc. Optical Fiber Communication (OFC), 2009*. IEEE (cit. on p. 208).

[CTV11] Konstantinos Christodoulopoulos, Ioannis Tomkos, and EA Varvarigos. "Elastic bandwidth allocation in flexible OFDM-based optical networks". In: *Journal of Lightwave Technology* 29.9 (2011), pp. 1354–1366 (cit. on p. 208).

[Gir+03] F. Giroire, A. Nucci, N. Taft, and C. Diot. "Increasing the robustness of IP backbones in the absence of optical level protection". In: *Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM*. Vol. 1. 2003, pp. 1–11 (cit. on p. 217).

[GJ02] Michael R Garey and David S Johnson. *Computers and intractability*. Vol. 29. wh freeman New York, 2002 (cit. on pp. 31, 212).

[Ian+02] Gianluca Iannaccone, Chen-nee Chuah, Richard Mortier, Supratik Bhattacharyya, and Christophe Diot. "Analysis of link failures in an IP backbone". In: *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. ACM. 2002, pp. 237–242 (cit. on p. 204).

[ID05] Stefan Irnich and Guy Desaulniers. "Shortest path problems with resource constraints". In: *Column generation* (2005), pp. 33–65 (cit. on pp. 212, 213).

[Jin+09] Masahiko Jinno, Hidehiko Takara, Bartlomiej Kozicki, Yukio Tsukishima, Yoshiaki Sone, and Shinji Matsuoka. "Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies". In: *IEEE Communications Magazine* 47.11 (2009) (cit. on pp. 203, 205).

[Jin+10] Masahiko Jinno, Bartlomiej Kozicki, Hidehiko Takara, Atsushi Watanabe, Yoshiaki Sone, Takafumi Tanaka, and Akira Hirano. "Distance-adaptive spectrum resource allocation in spectrum-sliced elastic optical path network [topics in optical communications]". In: *IEEE Communications Magazine* 48.8 (2010) (cit. on p. 204).

[Kli13a] Mirosław Klinkowski. "A genetic algorithm for solving RSA problem in elastic optical networks with dedicated path protection". In: *International Joint Conference CISIS'12-ICEUTE´ 12-SOCO´ 12 Special Sessions*. Springer. 2013, pp. 167–176 (cit. on p. 206).

[Kli13b] Mirosław Klinkowski. "An evolutionary algorithm approach for dedicated path protection problem in elastic optical networks". In: *Cybernetics and Systems* 44.6-7 (2013), pp. 589–605 (cit. on p. 206).

[Kre+14] Aristotelis Kretsis, Konstantinos Christodoulopoulos, Panagiotis Kokkinos, and Emmanouel Varvarigos. "Planning and operating flexible optical networks: Algorithmic issues and tools". In: *IEEE Communications Magazine* 52.1 (2014) (cit. on p. 208).

[KW11] Miroslaw Klinkowski and Krzysztof Walkowiak. "Routing and spectrum assignment in spectrum sliced elastic optical path network". In: *IEEE Communications Letters* 15.8 (2011), pp. 884–886 (cit. on p. 204).

[KW12] Mirosław Klinkowski and Krzysztof Walkowiak. "Offline RSA algorithms for elastic optical networks with dedicated path protection consideration". In: *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2012 4th International Congress on*. IEEE. 2012 (cit. on p. 206).

[Muk06] Biswanath Mukherjee. *Optical WDM networks*. Springer Science & Business Media, 2006 (cit. on p. 213).

[Orl+10b] Sebastian Orlowski, Roland Wessäly, Michal Pióro, and Artur Tomaszewski. "SNDlib 1.0—Survivable network design library". In: *Networks* 55.3 (2010) (cit. on pp. 68, 95, 119, 213).

[RSM03] S Ramamurthy, Laxman Sahasrabuddhe, and Biswanath Mukherjee. "Survivable WDM mesh networks". In: *Journal of Lightwave Technology* 21.4 (2003), p. 870 (cit. on pp. 204–206).

[Rui+13] Marc Ruiz, Michał Pióro, Mateusz Żotkiewicz, Mirosław Klinkowski, and Luis Velasco. "Column generation algorithm for RSA problems in flexgrid optical networks". In: *Photonic network communications* 26.2-3 (2013) (cit. on p. 207).

[SRM02] Laxman Sahasrabuddhe, Senthil Ramamurthy, and Biswanath Mukherjee. "Fault management in IP-over-WDM networks: WDM protection versus IP restoration". In: *IEEE journal on selected areas in communications* 20.1 (2002), pp. 21–33 (cit. on pp. 108, 204, 206).

[SWB14] Gangxiang Shen, Yue Wei, and Sanjay K Bose. "Optimal design for shared backup path protected elastic optical networks under single-link failure". In: *Journal of Optical Communications and Networking* 6.7 (2014) (cit. on pp. 206, 216).

[TJG18] A Tomassilli, B Jaumard, and F Giroire. "Path Protection in Optical Flexible Networks with Distance-adaptive Modulation Formats". In: *2018 International Conference on Optical Network Design and Modeling (ONDM)*. 2018 (cit. on pp. 13, 203).

[Tur+10] Daniel Turner, Kirill Levchenko, Alex C Snoeren, and Stefan Savage. "California fault lines: understanding the causes and impact of network failures". In: *ACM SIGCOMM Computer Communication Review*. Vol. 40. 4. ACM. 2010, pp. 315–326 (cit. on pp. 107, 108, 204).

[WK13] Krzysztof Walkowiak and Mirosław Klinkowski. "Shared backup path protection in elastic optical networks: Modeling and optimization". In: *Design of Reliable Communication Networks (DRCN), 2013 9th International Conference on the*. IEEE. 2013, pp. 187–194 (cit. on pp. 206, 216).

[Zhu+13] Zuqing Zhu, Wei Lu, Liang Zhang, and Nirwan Ansari. "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing". In: *Journal of Lightwave Technology* 31.1 (2013), pp. 15–22 (cit. on p. 214).

[ZS00] Dongyun Zhou and Suresh Subramaniam. “Survivability in optical networks”. In: *IEEE network* 14.6 (2000), pp. 16–23 (cit. on pp. 108, 204, 206).

Bibliography

[08] *SMART 2020 Enabling the low-carbon economy in the information age, http://www.smart2020.org/_assets/files/02_-Smart2020Report.pdf*. 2008 (cit. on pp. 12, 166).

[AA99] Murat Alanyali and Ender Ayanoglu. "Provisioning algorithms for WDM optical networks". In: *IEEE/ACM Transactions On Networking* 7.5 (1999) (cit. on p. 109).

[Abd+16] Sherif Abdelwahab, Bechir Hamdaoui, Mohsen Guizani, and Taieb Znati. "Network function virtualization in 5G". In: *IEEE Communications Magazine* 54.4 (2016), pp. 84–91 (cit. on p. 8).

[Ach04] Tobias Achterberg. "SCIP-a framework to integrate constraint and mixed integer programming". In: (2004) (cit. on p. 29).

[Add+15] Bernardetta Addis, Dallal Belabed, Mathieu Bouet, and Stefano Secci. "Virtual network functions placement and routing optimization". In: *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*. IEEE. 2015 (cit. on pp. 10, 11, 46).

[ADF04] Hatem Ben Amor, Jacques Desrosiers, and Antonio Frangioni. *Stabilization in column generation*. Groupe d'études et de recherche en analyse des décisions, 2004 (cit. on p. 30).

[ADP80] Giorgio Ausiello, Alessandro D'Atri, and Marco Protasi. "Structure preserving reductions among convex optimization problems". In: *Journal of Computer and System Sciences* 21.1 (1980), pp. 136–153 (cit. on pp. 48, 85).

[Ahm+14] Faraz Ahmad, Srimat T Chakradhar, Anand Raghunathan, and TN Vijaykumar. "ShuffleWatcher: Shuffle-aware Scheduling in Multi-tenant MapReduce Clusters." In: *USENIX Annual Technical Conference*. 2014, pp. 1–12 (cit. on p. 178).

[Aky+14] Ian F Akyildiz, Ahyoung Lee, Pu Wang, Min Luo, and Wu Chou. "A roadmap for traffic engineering in SDN-OpenFlow networks". In: *Computer Networks* 71 (2014), pp. 1–30 (cit. on pp. 3, 5).

[ALV08] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. "A scalable, commodity data center network architecture". In: *ACM SIGCOMM Computer Communication Review*. Vol. 38. 4. ACM. 2008, pp. 63–74 (cit. on p. 61).

[AMS06] N. Alon, D. Moshkovitz, and S. Safra. "Algorithmic Construction of Sets for K-restrictions". In: *ACM Trans. Algorithms* 2.2 (2006). ISSN: 1549-6325. DOI: 10.1145/1150334.1150336 (cit. on p. 48).

[Ara+16] J. Araujo, F. Giroire, J. Moulierac, Y. Liu, and R. Modrzejewski. "Energy Efficient Content Distribution". In: *The Computer Journal* 59.2 (Feb. 2016), pp. 192–207 (cit. on p. 153).

[AV14] YK Agarwal and Prahalad Venkateshan. "Survivable network design with shared-protection routing". In: *European Journal of Operational Research* 238.3 (2014), pp. 836–845 (cit. on p. 110).

[Bar+98] Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. "Branch-and-price: Column generation for solving huge integer programs". In: *Operations research* 46.3 (1998), pp. 316–329 (cit. on p. 30).

[BBS16] Michael Till Beck, Juan Felipe Botero, and Kai Samelin. "Resilient allocation of service Function chains". In: *Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE Conference on*. IEEE. 2016 (cit. on p. 110).

[Bel15] Alcetel Lucent Bell Labs. *White Paper: Global What if Analyzer of NeTwork Energy ConsumpTion (GWATT). Bell labs application able to measure the impact of technologies like SDN & NFV on network energy consumption*. Murray Hill, NJ, USA, 2015 (cit. on pp. 12, 166).

[Ben62] Jacques F Benders. "Partitioning procedures for solving mixed-variables programming problems". In: *Numerische mathematik* 4.1 (1962), pp. 238–252 (cit. on pp. 89, 97).

[Ber+14] Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O'Connor, Pavlin Radoslavov, William Snow, et al. "ONOS: towards an open, distributed SDN OS". In: *Proceedings of the third work-*

shop on Hot topics in software defined networking. ACM. 2014, pp. 1–6 (cit. on pp. 4, 80).

[Bha+16] Deval Bhamare, Raj Jain, Mohammed Samaka, and Aiman Erbad. "A survey on service function chaining". In: *Journal of Network and Computer Applications* 75 (2016), pp. 138–155 (cit. on p. 9).

[Bix12] Robert E Bixby. "A brief history of linear and mixed-integer programming computation". In: *Documenta Mathematica* (2012), pp. 107–121 (cit. on p. 28).

[Boc+] Adriana Bocoi, Matthias Schuster, Franz Rambach, Moritz Kiese, Christian-Alexander Bunge, and Bernhard Spinnler. "Reach-dependent capacity in optical networks enabled by OFDM". In: *Proc. Optical Fiber Communication (OFC), 2009*. IEEE (cit. on p. 208).

[Bol+10] Raffaele Bolla, Roberto Bruschi, Franco Davoli, and Flavio Cucchietti. "Energy efficiency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures". In: *IEEE Communications Surveys & Tutorials* 13.2 (2010), pp. 223–244 (cit. on p. 138).

[Bol+14] R. Bolla, C. Lombardo, R. Bruschi, and S. Mangialardi. "DROPv2: energy efficiency through network function virtualization". In: *IEEE Network* 28.2 (2014), pp. 26–32 (cit. on p. 139).

[Bol98] Béla Bollobás. "Random graphs". In: *Modern Graph Theory*. Springer, 1998, pp. 215–252 (cit. on p. 68).

[Bou+15] Mathieu Bouet, Jérémie Leguay, Théo Combe, and Vania Conan. "Cost-based placement of vDPI functions in NFV infrastructures". In: *International Journal of Network Management* 25.6 (2015), pp. 490–506 (cit. on p. 45).

[Cas+10] Martin Casado, Teemu Koponen, Rajiv Ramanathan, and Scott Shenker. "Virtualizing the network forwarding plane". In: *Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow*. ACM. 2010, p. 8 (cit. on p. 44).

[Cas+17] Marco Casazza, Pierre Fouilhoux, Mathieu Bouet, and Stefano Secci. "Securing virtual network function placement with high availability guarantees". In: *2017 IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE. 2017, pp. 1–9 (cit. on p. 110).

[CC+83] Vasek Chvatal, Vaclav Chvatal, et al. *Linear programming*. Macmillan, 1983 (cit. on p. 30).

[CCN10] Zheng Cai, Alan L Cox, and TS Ng. *Maestro: A system for scalable openflow control*. Tech. rep. 2010 (cit. on p. 4).

[CGC16] Tao Chen, Xiaofeng Gao, and Guihai Chen. "The features, hardware, and architectures of data center networks: A survey". In: *Journal of Parallel and Distributed Computing* 96 (2016), pp. 45–74 (cit. on p. 178).

[Cha+05] Claude Chaudet, Eric Fleury, Isabelle Guérin Lassous, Hervé Rivano, and Marie-Emilie Voge. "Optimal positioning of active and passive monitoring devices". In: *Proceedings of the 2005 ACM conference on Emerging network experiment and technology*. ACM. 2005, pp. 71–82 (cit. on p. 44).

[Cha+08] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright. "Power Awareness in Network Design and Routing". In: *Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM*. Apr. 2008, pp. 1130–1138 (cit. on p. 142).

[Cho+11] Mosharaf Chowdhury, Matei Zaharia, Justin Ma, Michael I Jordan, and Ion Stoica. "Managing data transfers in computer clusters with orchestra". In: *ACM SIGCOMM Computer Communication Review*. Vol. 41. 4. 2011 (cit. on pp. 13, 172, 174, 191).

[Chu+] Cing-Yu Chu, Kang Xi, Min Luo, and H Jonathan Chao. "Congestion-aware single link failure recovery in hybrid SDN networks". In: *Proceedings of IEEE INFOCOM, 2015* (cit. on p. 82).

[Chv79] V. Chvatal. "A greedy heuristic for the set-covering problem". In: *Mathematics of operations research* 4.3 (1979), pp. 233–235 (cit. on pp. 48, 54).

[Chv83] V. Chvatal. *Linear Programming*. Freeman, 1983 (cit. on pp. 88, 148).

[Cis] Cisco. *Bidirectional Forwarding Detection – Cisco*. https://www.cisco.com/c/en/us/td/docs/ios/12_0s/feature/guide/fs_bfd.html. Accessed: 2018-07 (cit. on p. 105).

[CKH95] Pierluigi Crescenzi, Viggo Kann, and M Halldórsson. *A compendium of NP optimization problems*. 1995 (cit. on p. 31).

[CKL] Fangfei Chen, Murali Kodialam, and TV Lakshman. "Joint scheduling of processing and shuffle phases in mapreduce systems". In: *IEEE INFOCOM 2012* (cit. on p. 175).

[CMN12] L. Chiaraviglio, M. Mellia, and F. Neri. "Minimizing ISP network energy cost: formulation and solutions". In: *IEEE/ACM Transactions on Networking (TON)* 20 (2 Apr. 2012), pp. 463–476 (cit. on p. 138).

[Coh+15] Rami Cohen, Liane Lewin-Eytan, Joseph Seffi Naor, and Danny Raz. "Near optimal placement of virtual network functions". In: *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE. 2015, pp. 1346–1354 (cit. on pp. 11, 46).

[CP99] Zhi-Long Chen and Warren B Powell. "Solving parallel machine scheduling problems by column generation". In: *INFORMS Journal on Computing* 11.1 (1999), pp. 78–94 (cit. on p. 29).

[CPL09] IBM ILOG CPLEX. "V12. 1: User's Manual for CPLEX". In: *International Business Machines Corporation* 46.53 (2009), p. 157 (cit. on p. 29).

[CPW99] Bo Chen, Chris N Potts, and Gerhard J Woeginger. "A review of machine scheduling: Complexity, algorithms and approximability". In: *Handbook of combinatorial optimization*. Springer, 1999, pp. 1493–1641 (cit. on p. 175).

[CS12] Mosharaf Chowdhury and Ion Stoica. "Coflow: A networking abstraction for cluster applications". In: *ACM Workshop on Hot Topics in Networks*. 2012, pp. 31–36 (cit. on p. 174).

[CTV11] Konstantinos Christodoulopoulos, Ioannis Tomkos, and EA Varvarigos. "Elastic bandwidth allocation in flexible OFDM-based optical networks". In: *Journal of Lightwave Technology* 29.9 (2011), pp. 1354–1366 (cit. on p. 208).

[CWJ18] Yang Chen, Jie Wu, and Bo Ji. "Virtual Network Function Deployment in Tree-structured Networks". In: *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. IEEE. 2018, pp. 132–142 (cit. on p. 11).

[CZS14] Mosharaf Chowdhury, Yuan Zhong, and Ion Stoica. "Efficient coflow scheduling with varys". In: *ACM SIGCOMM Computer Communication Review*. Vol. 44. 4. 2014, pp. 443–454 (cit. on pp. 172, 174).

[Dan48] George B Dantzig. "Programming in a linear structure". In: (1948) (cit. on p. 28).

[DDS06] Guy Desaulniers, Jacques Desrosiers, and Marius M Solomon. *Column generation*. Vol. 5. Springer Science & Business Media, 2006 (cit. on p. 30).

[DDS92] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. "A new optimization algorithm for the vehicle routing problem with time windows". In: *Operations research* 40.2 (1992), pp. 342–354 (cit. on p. 29).

[DG08] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters". In: *Communications of the ACM* 51.1 (2008) (cit. on p. 172).

[DGF10] Floriano De Rango, Francesca Guerriero, and Peppino Fazio. "Link-stability and energy aware routing protocol in distributed wireless networks". In: *IEEE Transactions on Parallel and Distributed systems* 23.4 (2010), pp. 713–726 (cit. on p. 138).

[DL05] Jacques Desrosiers and Marco E Lübbecke. "A primer in column generation". In: *Column generation*. Springer, 2005, pp. 1–32 (cit. on p. 30).

[Dog+14] Fahad R Dogar, Thomas Karagiannis, Hitesh Ballani, and Antony Rowstron. "Decentralized task-aware scheduling for data center networks". In: *ACM SIGCOMM Computer Communication Review*. 2014, pp. 431–442 (cit. on pp. 172, 174).

[DS] Irit Dinur and David Steurer. "Analytical Approach to Parallel Repetition". In: *Proceedings ACM STOC 2014*. New York, New York. ISBN: 978-1-4503-2710-7 (cit. on pp. 44, 85).

[DS05] Irit Dinur and Samuel Safra. "On the hardness of approximating minimum vertex cover". In: *Annals of mathematics* (2005), pp. 439–485 (cit. on p. 62).

[Eri13] David Erickson. "The beacon openflow controller". In: *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM. 2013, pp. 13–18 (cit. on p. 4).

[Eve+99] Guy Even, Joseph Naor, Satish Rao, and Baruch Schieber. "Fast approximate graph partitioning algorithms". In: *SIAM Journal on Computing* 28 (1999) (cit. on p. 188).

[Fen+17] Hao Feng, Jaime Llorca, Antonia M Tulino, Danny Raz, and Andreas F Molisch. "Approximation algorithms for the NFV service distribution problem". In: *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE. 2017, pp. 1–9 (cit. on p. 11).

[FM17] Paulo Fonseca and Edjard Mota. "A survey on fault management in software-defined networks". In: *IEEE Communications Surveys & Tutorials* (2017) (cit. on p. 82).

[Fou] Open Networking Foundation. URL: https://www.opennetworking/about (cit. on p. 4).

[Fra+03] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and S.C. Diot. "Packet-level traffic measurements from the Sprint IP backbone". In: *IEEE network* 17.6 (2003), pp. 6–16 (cit. on p. 153).

[FRZ14] Nick Feamster, Jennifer Rexford, and Ellen Zegura. "The road to SDN: an intellectual history of programmable networks". In: *ACM SIGCOMM Computer Communication Review* 44.2 (2014), pp. 87–98 (cit. on p. 2).

[FT02] Bernard Fortz and Mikkel Thorup. "Optimizing OSPF/IS-IS weights in a changing world". In: *IEEE journal on selected areas in communications* 20.4 (2002), pp. 756–767 (cit. on p. 96).

[FV00] Andrea Fumagalli and Luca Valcarenghi. "IP restoration vs. WDM protection: Is there an optimal choice?" In: *IEEE network* 14.6 (2000) (cit. on p. 82).

[Gir+03] F. Giroire, A. Nucci, N. Taft, and C. Diot. "Increasing the robustness of IP backbones in the absence of optical level protection". In: *Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM*. Vol. 1. 2003, pp. 1–11 (cit. on p. 217).

[Gir+10] Frédéric Giroire, Dorian Mazauric, Joanna Moulierac, and Brice Onfroy. "Minimizing routing energy consumption: from theoretical to practical results". In: *2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*. IEEE. 2010, pp. 252–259 (cit. on p. 138).

[Gir+15] F. Giroire, J. Moulierac, Truong Khoa Phan, and F. Roudaut. "Minimization of network power consumption with redundancy elimination". In: *Computer communications* 59 (2015), pp. 98–105 (cit. on p. 139).

[Gir+19a] Frédéric Giroire, Nicolas Huin, Andrea Tomassilli, and Stéphane Pérennes. *When Network Matters: Data Center Scheduling with Network Tasks*. Tech. rep. Inria, Jan. 2019 (cit. on p. 186).

[Gir+19b] Frédéric Giroire, Nicolas Huin, Andrea Tomassilli, and Stéphane Pérennes. "When network matters: Data center scheduling with network tasks". In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. 2019 (cit. on pp. 13, 171).

[GJ02] Michael R Garey and David S Johnson. *Computers and intractability*. Vol. 29. wh freeman New York, 2002 (cit. on pp. 31, 212).

[GJN11] P. Gill, N. Jain, and N. Nagappan. "Understanding network failures in data centers: measurement, analysis, and implications". In: *ACM SIGCOMM Computer Communication Review*. Vol. 41. 4. 2011 (cit. on p. 107).

[GMP14] F. Giroire, J. Moulierac, and K. Phan. "Optimizing Rule Placement in Software-Defined Networks for Energy-aware Routing". In: *IEEE Global Telecommunications Conference - GLOBECOM*. Austin, USA, Dec. 2014, pp. 2523–2529 (cit. on p. 139).

[Gom+58] Ralph E Gomory et al. "Outline of an algorithm for integer solutions to linear programs". In: *Bulletin of the American Mathematical society* 64.5 (1958), pp. 275–278 (cit. on p. 29).

[Gra66] Ronald L Graham. "Bounds for certain multiprocessing anomalies". In: *Bell System Technical Journal* 45.9 (1966), pp. 1563–1581 (cit. on pp. 175, 182, 183).

[Gre+09] Albert Greenberg, James R Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A Maltz, Parveen Patel, and Sudipta Sengupta. "VL2: a scalable and flexible data center network". In: *ACM SIGCOMM computer communication review*. Vol. 39. 4. 2009, pp. 51–62 (cit. on p. 172).

[GS03] Maruti Gupta and Suresh Singh. "Greening of the Internet". In: *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM. 2003, pp. 19–26 (cit. on p. 138).

[Guo+08] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, and Songwu Lu. "Dcell: a scalable and fault-tolerant network structure for data centers". In: *ACM SIGCOMM Computer Communication Review*. Vol. 38. 4. 2008 (cit. on p. 172).

[Gup+15] A. Gupta, M.F. Habib, P. Chowdhury, M. Tornatore, and B. Mukherjee. "On service chaining using virtual network functions in network-enabled cloud systems". In: *IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*. 2015, pp. 1–3 (cit. on p. 138).

[Gup+17] A. Gupta, B. Mukherjee, B. Jaumard, and M. Tornatore. "Service Chain (SC) Mapping with Multiple SC Instances in a Wide Area Network". In: *IEEE Global Telecommunications Conference - GLOBECOM*. 2017, pp. 1–6 (cit. on p. 138).

[GVY93] Naveen Garg, Vijay V Vazirani, and Mihalis Yannakakis. "Approximate max-flow min-(multi) cut theorems and their applications". In: *ACM symposium on Theory of computing*. 1993, pp. 698–707 (cit. on p. 179).

[Han+15] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. "Network function virtualization: Challenges and opportunities for innovations". In: *IEEE Communications Magazine* 53.2 (2015), pp. 90–97 (cit. on pp. 8, 10).

[Hras01] Johan Hrastad. "Some optimal inapproximability results". In: *Journal of the ACM (JACM)* 48.4 (2001), pp. 798–859 (cit. on p. 92).

[HB16] Juliver Gil Herrera and Juan Felipe Botero. "Resource allocation in NFV: A comprehensive survey". In: *IEEE Transactions on Network and Service Management* 13.3 (2016), pp. 518–532 (cit. on pp. 7, 9, 10).

[HIP15] Enrique Hernandez-Valencia, Steven Izzo, and Beth Polonsky. "How will NFV/SDN transform service provider opex?" In: *IEEE Network* 29.3 (2015), pp. 60–67 (cit. on p. 10).

[HJG17a] Nicolas Huin, Brigitte Jaumard, and Frédéric Giroire. "Optimization of Network Service Chain Provisioning". In: *IEEE International Conference on Communications 2017*. Paris, France, 2017 (cit. on pp. 110, 111).

[HJG17b] Nicolas Huin, Brigitte Jaumard, and Frédéric Giroire. "Optimization of network service chain provisioning". In: *2017 IEEE International Conference on Communications (ICC)*. IEEE. 2017, pp. 1–7 (cit. on p. 29).

[HJG18a] N. Huin, B. Jaumard, and F. Giroire. "Optimal Network Service Chain Provisioning". In: *IEEE/ACM Transactions on Networking* 26.3 (June 2018), pp. 1320–1333. ISSN: 1063-6692. DOI: 10.1109/TNET.2018.2833815 (cit. on p. 45).

[HJG18b] Nicolas Huin, Brigitte Jaumard, and Frédéric Giroire. "Optimal Network Service Chain Provisioning". In: *IEEE/ACM Transactions on Networking* (2018) (cit. on pp. 85, 96).

[Hma+17] Ali Hmaity, Marco Savi, Francesco Musumeci, Massimo Tornatore, and Achille Pattavina. "Protection strategies for virtual network functions placement and service chains provisioning". In: *Networks* (2017), pp. 1–15 (cit. on pp. 110, 111).

[HP85] Karla Hoffman and Manfred Padberg. "LP-based combinatorial problem solving". In: *Annals of Operations Research* 4.1 (1985), pp. 145–194 (cit. on p. 30).

[Hu03] Jian Qiang Hu. "Diverse routing in optical mesh networks". In: *IEEE Transactions on Communications* 51.3 (2003), pp. 489–494 (cit. on p. 80).

[Hua17] Huawei. *Huawei Releases SDN/NFV Commercial and Technological Innovations*. 2017. URL: http://www.huawei.com/en/press-events/news/2017/10/Huawei-SDN-NFV-Commercial-Technological-Innovations (cit. on p. 2).

[Hui+18a] N Huin, A Tomassilli, F Giroire, and B Jaumard. "Energy-efficient service function chain provisioning". In: *IEEE/OSA Journal of Optical Communications and Networking* 10.3 (2018), pp. 114–124 (cit. on pp. 12, 45, 136).

[Hui+18b] Nicolas Huin, Myriana Rifai, Frédéric Giroire, Dino Lopez Pacheco, Guillaume Urvoy-Keller, and Joanna Moulierac. "Bringing energy aware routing closer to reality with SDN hybrid networks". In: *IEEE Transactions on Green Communications and Networking* 2.4 (2018), pp. 1128–1139 (cit. on p. 139).

[Hui+18c] Nicolas Huin, Andrea Tomassilli, Frédéric Giroire, and Brigitte Jaumard. "Energy-Efficient Service Function Chain Provisioning". In: *IEEE/OSA Journal of Optical Communications and Networking* 10.2 (2018) (cit. on p. 110).

[Ian+02] Gianluca Iannaccone, Chen-nee Chuah, Richard Mortier, Supratik Bhattacharyya, and Christophe Diot. "Analysis of link failures in an IP backbone". In: *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. ACM. 2002, pp. 237–242 (cit. on p. 204).

[ID05] Stefan Irnich and Guy Desaulniers. "Shortest path problems with resource constraints". In: *Column generation* (2005), pp. 33–65 (cit. on pp. 212, 213).

[IMG98] R.R. Iraschko, M.H. MacGregor, and W.D. Grover. "Optimal capacity placement for path restoration in STM or ATM mesh-survivable networks". In: *IEEE/ACM Transactions on Networking* 6.3 (1998) (cit. on p. 108).

[Isa+07] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. "Dryad: distributed data-parallel programs from sequential building blocks". In: *ACM SIGOPS operating systems review*. Vol. 41. 3. ACM. 2007 (cit. on p. 172).

[Iye+03] Sundar Iyer, Supratik Bhattacharyya, Nina Taft, and Christophe Diot. "An approach to alleviate link overload as observed on an IP backbone". In: *Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM*. Vol. 1. 2003, pp. 406–416 (cit. on p. 153).

[Jai+13] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. "B4: Experience with a globally-deployed software defined WAN". In: *ACM SIGCOMM Computer Communication Review*. Vol. 43. 4. ACM. 2013 (cit. on pp. 1, 3).

[Jal+15] Virajith Jalaparti, Peter Bodik, Ishai Menache, Sriram Rao, Konstantin Makarychev, and Matthew Caesar. "Network-aware scheduling for data-parallel jobs: Plan when you can". In: *ACM SIGCOMM Computer Communication Review*. Vol. 45. 4. 2015, pp. 407–420 (cit. on p. 175).

[Jin+09] Masahiko Jinno, Hidehiko Takara, Bartlomiej Kozicki, Yukio Tsukishima, Yoshiaki Sone, and Shinji Matsuoka. "Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies". In: *IEEE Communications Magazine* 47.11 (2009) (cit. on pp. 203, 205).

[Jin+10] Masahiko Jinno, Bartlomiej Kozicki, Hidehiko Takara, Atsushi Watanabe, Yoshiaki Sone, Takafumi Tanaka, and Akira Hirano. "Distance-adaptive spectrum resource allocation in spectrum-sliced elastic optical path network [topics in optical communications]". In: *IEEE Communications Magazine* 48.8 (2010) (cit. on p. 204).

[Jün+09] Michael Jünger, Thomas M Liebling, Denis Naddef, George L Nemhauser, William R Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A Wolsey. *50 Years of integer programming 1958-2008: From the early years to the state-of-the-art*. Springer Science & Business Media, 2009 (cit. on p. 29).

[KCG] Amund Kvalbein, Tarik Cicic, and Stein Gjessing. "Post-failure routing performance with multiple routing configurations". In: *Proceedings of IEEE INFOCOM, 2007* (cit. on pp. 82, 83).

[Kem+12] James Kempf, Elisa Bellagamba, András Kern, David Jocha, Attila Takács, and Pontus Sköldström. "Scalable fault management for OpenFlow". In: *Communications (ICC), 2012 IEEE international conference on*. IEEE. 2012, pp. 6606–6610 (cit. on p. 80).

[KF13] Hyojoon Kim and Nick Feamster. "Improving network management with software defined networking". In: *IEEE Communications Magazine* 51.2 (2013), pp. 114–119 (cit. on p. 1).

[Kha79] Leonid G Khachiyan. "A polynomial algorithm in linear programming". In: *Doklady Academii Nauk SSSR*. Vol. 244. 1979, pp. 1093–1096 (cit. on p. 28).

[KKV05] Srikanth Kandula, Dina Katabi, and Jean-Philippe Vasseur. "Shrink: A tool for failure diagnosis in IP networks". In: *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*. ACM. 2005, pp. 173–178 (cit. on pp. 80, 95).

[KL70] Brian W Kernighan and Shen Lin. "An efficient heuristic procedure for partitioning graphs". In: *The Bell system technical journal* 49 (1970) (cit. on p. 190).

[Kli13a] Mirosław Klinkowski. "A genetic algorithm for solving RSA problem in elastic optical networks with dedicated path protection". In: *International Joint Conference CISIS'12-ICEUTE´ 12-SOCO´ 12 Special Sessions*. Springer. 2013, pp. 167–176 (cit. on p. 206).

[Kli13b] Mirosław Klinkowski. "An evolutionary algorithm approach for dedicated path protection problem in elastic optical networks". In: *Cybernetics and Systems* 44.6-7 (2013), pp. 589–605 (cit. on p. 206).

[KM70] Victor Klee and George J Minty. *How good is the simplex algorithm*. Tech. rep. WASHINGTON UNIV SEATTLE DEPT OF MATHEMATICS, 1970 (cit. on p. 28).

[Kni+11] Simon Knight, Hung X Nguyen, Nick Falkner, Rhys Bowden, and Matthew Roughan. "The internet topology zoo". In: *IEEE Journal on Selected Areas in Communications* 29.9 (2011), pp. 1765–1775 (cit. on pp. 55, 68, 72).

[KNS] Robert Krauthgamer, Joseph Naor, and Roy Schwartz. "Partitioning graphs into balanced components". In: *ACM-SIAM SODA 2009* (cit. on p. 188).

[Kop+10] Teemu Koponen, Martin Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan, Yuichiro Iwata, Hiroaki Inoue, Takayuki Hama, et al. "Onix: A distributed control platform for large-scale production networks." In: *OSDI*. Vol. 10. 2010, pp. 1–6 (cit. on p. 4).

[Kre+14] Aristotelis Kretsis, Konstantinos Christodoulopoulos, Panagiotis Kokkinos, and Emmanouel Varvarigos. "Planning and operating flexible optical networks: Algorithmic issues and tools". In: *IEEE Communications Magazine* 52.1 (2014) (cit. on p. 208).

[Kre+15] Diego Kreutz, Fernando MV Ramos, Paulo Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. "Software-defined networking: A comprehensive survey". In: *Proceedings of the IEEE* 103.1 (2015), pp. 14–76 (cit. on pp. 3, 4).

[Kum+15] S Kumar, M Tufail, S Majee, C Captari, and S Homma. "Service function chaining use cases in data centers". In: *IETF SFC WG* (2015) (cit. on p. 10).

[Kuo+16] Tung-Wei Kuo, Bang-Heng Liou, Kate Ching-Ju Lin, and Ming-Jer Tsai. "Deploying chains of virtual network functions: On the relation between link and server usage". In: *Computer Communications (INFOCOM), 2016 IEEE Conference on*. IEEE. 2016, pp. 1–9 (cit. on p. 46).

[Kva+] Amund Kvalbein, Audun Fosselie Hansen, Stein Gjessing, and Olav Lysne. "Fast IP network recovery using multiple routing configurations". In: *Proceedings of IEEE INFOCOM, 2006* (cit. on p. 83).

[KW11] Miroslaw Klinkowski and Krzysztof Walkowiak. "Routing and spectrum assignment in spectrum sliced elastic optical path network". In: *IEEE Communications Letters* 15.8 (2011), pp. 884–886 (cit. on p. 204).

[KW12] Mirosław Klinkowski and Krzysztof Walkowiak. "Offline RSA algorithms for elastic optical networks with dedicated path protection consideration". In: *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2012 4th International Congress on*. IEEE. 2012 (cit. on p. 206).

[LC12] Zitao Liu and Sangyeun Cho. "Characterizing machines and workloads on a Google cluster". In: *IEEE Parallel Processing Workshops (ICPPW)*. 2012 (cit. on p. 191).

[LC15] Yong Li and Min Chen. "Software-defined network function virtualization: A survey". In: *IEEE Access* 3 (2015), pp. 2542–2553 (cit. on p. 9).

[LD10] Ailsa H Land and Alison G Doig. "An automatic method for solving discrete programming problems". In: *50 Years of Integer Programming 1958-2008*. Springer, 2010, pp. 105–132 (cit. on pp. 28, 30).

[Le +13] Esther Le Rouzic, Edoardo Bonetto, Luca Chiaraviglio, Frederic Giroire, Filip Idzikowski, Felipe Jiménez, Christoph Lange, Julio Montalvo, Francesco Musumeci, Issam Tahiri, et al. "TREND towards more energy-efficient optical networks". In: *2013 17th International Conference on Optical Networking Design and Modeling (ONDM)*. IEEE. 2013, pp. 211–216 (cit. on p. 136).

[LHM10] Bob Lantz, Brandon Heller, and Nick McKeown. "A Network in a Laptop: Rapid Prototyping for Software-defined Networks". In: *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. Hotnets-IX. Monterey, California: ACM, 2010, 19:1–19:6. ISBN: 978-1-4503-0409-2. DOI:

10.1145/1868447.1868466. URL: http://doi.acm.org/10.1145/1868447.1868466 (cit. on pp. 101, 103).

[LKR14] Adrian Lara, Anisha Kolasani, and Byrav Ramamurthy. "Network innovation using openflow: A survey". In: *IEEE communications surveys & tutorials* 16.1 (2014), pp. 493–512 (cit. on pp. 5, 6).

[LM15] S.-I. Lee and Myung M.-K. Shin. "A self-recovery scheme for service function chaining". In: *International Conference on Information and Communication Technology Convergence (ICTC)*. 2015, pp. 108–112 (cit. on p. 110).

[LMF11] L. Chiaraviglio, M. Mellia, and F. Neri. "Minimizing ISP Network Energy Cost: Formulation and Solutions". In: *IEEE/ACM Transactions on Networking* 20.2 (Apr. 2011), pp. 463–476 (cit. on p. 138).

[Lou03] Robin Lougee-Heimer. "The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community". In: *IBM Journal of Research and Development* 47.1 (2003), pp. 57–66 (cit. on p. 29).

[Lui+15] M. C. Luizelli, L. R. Bays, L.S. Buriol, M. P. Barcellos, and L. P. Gaspary. "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions". In: *IFIP/IEEE International Symposium on Integrated Network Management*. 2015 (cit. on pp. 10, 46).

[LW66] Eugene L Lawler and David E Wood. "Branch-and-bound methods: A survey". In: *Operations research* 14.4 (1966), pp. 699–719 (cit. on p. 29).

[LY+84] David G Luenberger, Yinyu Ye, et al. *Linear and nonlinear programming*. Vol. 2. Springer, 1984 (cit. on p. 28).

[Ma+17] Wenrui Ma, Oscar Sandoval, Jonathan Beltran, Deng Pan, and Niki Pissinou. "Traffic aware placement of interdependent nfv middleboxes". In: *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE. 2017, pp. 1–9 (cit. on p. 11).

[Mak15] A Makhorin. *The GNU Linear Programming Kit (GLPK). GNU Software Foundation, 2000*. 2015 (cit. on p. 29).

[Mar+15] B. Martini, F. Paganelli, P. Cappanera, S. Turchi, and P. Castoldi. "Latency-aware composition of virtual functions in 5g". In: *NetSoft*. IEEE. 2015, pp. 1–6 (cit. on p. 138).

[Mar17] Sue Marek. *Update: AT&T's Stephens: More Than 40% of Network Functions Are Virtualized*. 2017. URL: https://www.sdxcentral.com/articles/news/atts-stephens-47-network-functions-virtualized/2017/07/ (cit. on p. 2).

[Mat+13] Daisuke Matsubara, Takashi Egawa, Nozomu Nishinaga, Ved P Kafle, Myung-Ki Shin, and Alex Galis. "Toward future networks: a viewpoint from ITU-T". In: *IEEE Communications Magazine* 51.3 (2013), pp. 112–118 (cit. on p. 136).

[McK+08] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. "OpenFlow: enabling innovation in campus networks". In: *ACM SIGCOMM Computer Communication Review* 38.2 (2008), pp. 69–74 (cit. on pp. 2, 4).

[MD14] Hendrik Moens and Filip De Turck. "VNF-P: A model for efficient placement of virtualized network functions". In: *10th International Conference on Network and Service Management (CNSM) and Workshop*. IEEE. 2014, pp. 418–423 (cit. on p. 10).

[MD79] R Garey Michael and S Johnson David. "Computers and intractability: a guide to the theory of NP-completeness". In: *WH Free. Co., San Fr* (1979) (cit. on p. 185).

[Med+] J. Medved, R. Varga, A. Tkacik, and K. Gray. "OpenDaylight: Towards a Model-Driven SDN Controller architecture". In: *Proceedings of IEEE WoWMoM 2014* (cit. on p. 103).

[Med+14] Jan Medved, Robert Varga, Anton Tkacik, and Ken Gray. "Opendaylight: Towards a model-driven sdn controller architecture". In: *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*. IEEE. 2014, pp. 1–6 (cit. on p. 4).

[Med+17] Ahmed M Medhat, Tarik Taleb, Asma Elmangoush, Giuseppe A Carella, Stefan Covaci, and Thomas Magedanz. "Service function chaining in next generation networks: State of the art and research challenges". In: *IEEE Communications Magazine* 55.2 (2017), pp. 216–223 (cit. on pp. 9, 10).

[Men27] Karl Menger. "Zur allgemeinen kurventheorie". In: *Fundamenta Mathematicae* 10.1 (1927), pp. 96–115 (cit. on p. 51).

[Mij+16] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. "Network function virtualization: State-of-the-art and research challenges". In: *IEEE Communications Surveys & Tutorials* 18.1 (2016), pp. 236–262 (cit. on p. 8).

[Mij15] Rashid Mijumbi. "On the Energy Efficiency Prospects of Network Function Virtualization". In: *CoRR* abs/1512.00215 (2015). URL: http://arxiv.org/abs/1512.00215 (cit. on p. 139).

[MKK14] Sevil Mehraghdam, Matthias Keller, and Holger Karl. "Specifying and placing chains of virtual network functions". In: *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. IEEE. 2014, pp. 7–13 (cit. on pp. 10, 45, 46).

[Mod+13] Remigiusz Modrzejewski, Luca Chiaraviglio, Issam Tahiri, Frederic Giroire, Esther Le Rouzic, Edoardo Bonetto, Francesco Musumeci, Roberto Gonzalez, and Carmen Guerrero. "Energy efficient content distribution in an ISP network". In: *2013 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2013, pp. 2859–2865 (cit. on p. 138).

[Moh+15] Ali Mohammadkhan, Sheida Ghapani, Guyue Liu, Wei Zhang, KK Ramakrishnan, and Timothy Wood. "Virtual function placement and traffic steering in flexible and dynamic software defined networks". In: *Local and Metropolitan Area Networks (LANMAN), 2015 IEEE International Workshop on*. IEEE. 2015, pp. 1–6 (cit. on pp. 10, 11, 46, 138).

[MQS98] Alix Munier, Maurice Queyranne, and Andreas S. Schulz. "Approximation Bounds for a General Class of Precedence Constrained Parallel Machine Scheduling Problems". In: *Lecture*

Notes in Computer Science (1998), pp. 367–382. ISSN: 0302-9743. DOI: 10.1007/3-540-69346-7_28 (cit. on p. 175).

[MT96] Anuj Mehrotra and Michael A Trick. "A column generation approach for graph coloring". In: *informs Journal on Computing* 8.4 (1996), pp. 344–354 (cit. on p. 29).

[Muk06] Biswanath Mukherjee. *Optical WDM networks*. Springer Science & Business Media, 2006 (cit. on p. 213).

[Mur+11] Derek G Murray, Malte Schwarzkopf, Christopher Smowton, Steven Smith, Anil Madhavapeddy, and Steven Hand. "CIEL: a universal execution engine for distributed data-flow computing". In: *Proc. 8th ACM/USENIX Symposium on Networked Systems Design and Implementation*. 2011, pp. 113–126 (cit. on p. 172).

[Net15] Index Cisco Visual Networking. "Cisco visual networking index: Forecast and methodology 2015-2020". In: *White paper, CISCO)* (2015) (cit. on p. 152).

[Ngu+17] Van-Giang Nguyen, Anna Brunstrom, Karl-Johan Grinnemo, and Javid Taheri. "SDN/NFV-based mobile packet core network architectures: A survey". In: *IEEE Communications Surveys & Tutorials* 19.3 (2017), pp. 1567–1602 (cit. on p. 8).

[Nic+12] L. Niccolini, G. Iannaccone, S. Ratnasamy, J. Chandrashekar, and L. Rizzo. "Building a power-proportional software router". In: *USENIX Annual Technical Conference (USENIX ATC)*. Boston, MA, USA, 2012, pp. 89–100 (cit. on p. 142).

[Niv+09] B Niven-Jenkins, D Brungard, M Betts, N Sprecher, and S Ueno. *Requirements of an MPLS transport profile*. Tech. rep. 2009 (cit. on p. 80).

[NS14] Dmitry Namiot and Manfred Sneps-Sneppe. "On micro-services architecture". In: *International Journal of Open Information Technologies* 2.9 (2014), pp. 24–27 (cit. on p. 172).

[Nun+14] Bruno Astuto A Nunes, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. "A survey of software-defined networking: Past, present, and future of programmable networks". In: *IEEE Communications Surveys & Tutorials* 16.3 (2014), pp. 1617–1634 (cit. on p. 2).

[Oba+16] Mathis Obadia, Jean-Louis Rougier, Luigi Iannone, Vania Conan, and Mathieu Brouet. "Revisiting NFV orchestration with routing games". In: *Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE Conference on*. IEEE. 2016, pp. 107–113 (cit. on pp. 11, 44).

[OPT14] GUROBI OPTIMIZATION. "INC. Gurobi optimizer reference manual, 2015". In: *URL: http://www. gurobi. com* (2014) (cit. on p. 29).

[Orl+10a] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. "SNDlib 1.0–Survivable Network Design Library". English. In: *Networks* 55.3 (2010), pp. 276–286. DOI: 10.1002/net.20371 (cit. on p. 152).

[Orl+10b] Sebastian Orlowski, Roland Wessäly, Michal Pióro, and Artur Tomaszewski. "SNDlib 1.0—Survivable network design library". In: *Networks* 55.3 (2010) (cit. on pp. 68, 95, 119, 213).

[PBL14] Kévin Phemius, Mathieu Bouet, and Jérémie Leguay. "Disco: Distributed multi-domain sdn controllers". In: *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE. 2014, pp. 1–4 (cit. on p. 4).

[Pes+18] Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. "Automation and combination of linear-programming based stabilization techniques in column generation". In: *INFORMS Journal on Computing* (2018) (cit. on pp. 30, 89).

[Pfa+09] Ben Pfaff, Justin Pettit, Keith Amidon, Martin Casado, Teemu Koponen, and Scott Shenker. "Extending networking into the virtualization layer." In: *Hotnets*. 2009 (cit. on p. 2).

[Pha14] Truong Khoa Phan. "Design and management of networks with low power consumption". PhD thesis. Université Nice Sophia Antipolis, 2014 (cit. on p. 138).

[PJ13] R. Potharaju and N. Jain. "Demystifying the dark side of the middle: a field study of middlebox failures in datacenters". In: *Internet Measurement Conference*. 2013, pp. 9–22 (cit. on p. 107).

[PM04] Michal Pióro and Deep Medhi. *Routing, flow, and capacity design in communication and computer networks*. Elsevier, 2004 (cit. on pp. 12, 80).

[Pou+19] Konstantinos Poularakis, Jaime Llorca, Antonia M Tulino, Ian Taylor, and Leandros Tassiulas. "Joint Service Placement and Request Routing in Multi-cell Mobile Edge Computing Networks". In: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. 2019 (cit. on p. 11).

[PST18] Manish Paliwal, Deepti Shrimankar, and Omprakash Tembhurne. "Controllers in SDN: A review report". In: *IEEE Access* 6 (2018), pp. 36256–36270 (cit. on p. 6).

[PY90] Christos H Papadimitriou and Mihalis Yannakakis. "Towards an architecture-independent analysis of parallel algorithms". In: *SIAM journal on computing* 19.2 (1990), pp. 322–328 (cit. on p. 175).

[Qaz+13] Zafar Ayyub Qazi, Cheng-Chun Tu, Luis Chiang, Rui Miao, Vyas Sekar, and Minlan Yu. "SIMPLE-fying middlebox policy enforcement using SDN". In: *ACM SIGCOMM computer communication review*. Vol. 43. 4. ACM. 2013, pp. 27–38 (cit. on p. 9).

[QN15] Paul Quinn and Tom Nadeau. "Problem statement for service function chaining". In: (2015) (cit. on p. 9).

[Ray87] Victor J Rayward-Smith. "UET scheduling with unit interprocessor communication delays". In: *Discrete Applied Mathematics* 18.1 (1987) (cit. on pp. 175, 179, 182, 183, 191).

[Ren+13] Kai Ren, YongChul Kwon, Magdalena Balazinska, and Bill Howe. "Hadoop's adolescence: an analysis of Hadoop usage in scientific workloads". In: *Proceedings of the VLDB Endowment* 6.10 (2013), pp. 853–864 (cit. on pp. 184, 192).

[Rig+15] R. Riggio, Abbas Bradai, Tinku Rasheed, Julius Schulz-Zander, Slawomir Kuklinski, and Toufik Ahmed. "Virtual network functions orchestration in wireless networks". In: *Intl. Conf. on Network and Service Management (CNSM)*. 2015, pp. 108–116 (cit. on pp. 11, 138).

[RM99] S. Ramamurthy and B. Mukherjee. "Survivable WDM mesh networks. Part I - protection". In: *Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM*. Vol. 2. 1999, pp. 744–751 (cit. on pp. 108, 109).

[RMD05] Smita Rai, Biswanath Mukherjee, and Omkar Deshpande. "IP resilience within an autonomous system: current approaches, challenges, and future directions". In: *IEEE Communications Magazine* 43.10 (2005), pp. 142–149 (cit. on p. 82).

[RSM03] S Ramamurthy, Laxman Sahasrabuddhe, and Biswanath Mukherjee. "Survivable WDM mesh networks". In: *Journal of Lightwave Technology* 21.4 (2003), p. 870 (cit. on pp. 204–206).

[Rui+13] Marc Ruiz, Michał Pióro, Mateusz Żotkiewicz, Mirosław Klinkowski, and Luis Velasco. "Column generation algorithm for RSA problems in flexgrid optical networks". In: *Photonic network communications* 26.2-3 (2013) (cit. on p. 207).

[RWH11] Charles Reiss, John Wilkes, and Joseph L Hellerstein. "Google cluster-usage traces: format+ schema". In: *Google Inc., White Paper* (2011), pp. 1–14 (cit. on pp. 174, 191, 198).

[San+17] Yu Sang, Bo Ji, Gagan R Gupta, Xiaojiang Du, and Lin Ye. "Provably Efficient Algorithms for Joint Placement and Allocation of Virtual Network Functions". In: *Computer Communications (INFOCOM), 2017 IEEE Conference on*. IEEE. 2017 (cit. on pp. 11, 46, 47).

[Sga+13] Andrea Sgambelluri, Alessio Giorgetti, Filippo Cugini, Francesco Paolucci, and Piero Castoldi. "OpenFlow-based segment protection in Ethernet networks". In: *Journal of Optical Communications and Networking* 5.9 (2013) (cit. on pp. 82, 108).

[Sha+11] Sachin Sharma, Dimitri Staessens, Didier Colle, Mario Pickavet, and Piet Demeester. "Enabling fast failure recovery in OpenFlow networks". In: *2011 8th International Workshop on the Design of Reliable Communication Networks (DRCN 2011)*. IEEE. 2011, pp. 164–171 (cit. on p. 108).

[Sha+13] Sachin Sharma, Dimitri Staessens, Didier Colle, Mario Pickavet, and Piet Demeester. "OpenFlow: Meeting carrier-grade recovery requirements". In: *Computer Communications* 36.6 (2013), pp. 656–665 (cit. on pp. 80, 108).

[She+12] Justine Sherry, Shaddi Hasan, Colin Scott, Arvind Krishnamurthy, Sylvia Ratnasamy, and Vyas Sekar. "Making middleboxes someone else's problem: network processing as a cloud service". In: *ACM SIGCOMM Computer Communication Review* 42.4 (2012), pp. 13–24 (cit. on pp. 1, 7).

[SJ19] Gamal Sallam and Bo Ji. "Joint Placement and Allocation of Virtual Network Functions with Budget and Capacity Constraints". In: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. 2019 (cit. on p. 11).

[SLX10] Yunfei Shang, Dan Li, and Mingwei Xu. "Energy-aware routing in data center network". In: *Proceedings of the first ACM SIGCOMM workshop on Green networking*. ACM. 2010, pp. 1–8 (cit. on p. 138).

[Soh+00] Katayoun Sohrabi, Jay Gao, Vishal Ailawadhi, and Gregory J Pottie. "Protocols for self-organization of a wireless sensor network". In: *IEEE personal communications* 7.5 (2000), pp. 16–27 (cit. on p. 61).

[Sou+17] O. Soualah, Marouen Mechtri, Chaima Ghribi, and Djamal Zeghlache. "A link failure recovery algorithm for Virtual Network Function chaining". In: *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. 2017 (cit. on p. 110).

[SRA12] Justine Sherry, Sylvia Ratnasamy, and Justine Sherry At. "A survey of enterprise middlebox deployments". In: (2012) (cit. on p. 7).

[SRM02] Laxman Sahasrabuddhe, Senthil Ramamurthy, and Biswanath Mukherjee. "Fault management in IP-over-WDM networks: WDM protection versus IP restoration". In: *IEEE journal on selected areas in communications* 20.1 (2002), pp. 21–33 (cit. on pp. 108, 204, 206).

[SS98] Martin Savelsbergh and Marc Sol. "Drive: Dynamic routing of independent vehicles". In: *Operations Research* 46.4 (1998), pp. 474–490 (cit. on p. 29).

[ST97] Horst D Simon and Shang-Hua Teng. "How good is recursive bisection?" In: *SIAM Journal on Scientific Computing* 18.5 (1997), pp. 1436–1445 (cit. on pp. 188, 190).

[Sta+11] Dimitri Staessens, Sachin Sharma, Didier Colle, Mario Pickavet, and Piet Demeester. "Software defined networking: Meeting carrier grade requirements". In: *18th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN)*. IEEE. 2011 (cit. on p. 108).

[Sti+07] Thomas Stidsen, Bjørn Petersen, Kasper Bonne Rasmussen, Simon Spoorendonk, Martin Zachariasen, Franz Rambach, and Moritz Kiese. "Optimal routing with single backup path protection". In: *International Network Optimization Conference (INOC)*. 2007 (cit. on p. 117).

[STV] Marco Savi, Massimo Tornatore, and Giacomo Verticale. "Impact of processing costs on service chain placement in network functions virtualization". In: *IEEE NFV-SDN 2015* (cit. on pp. 9, 49, 119).

[STV15] M. Savi, M. Tornatore, and G. Verticale. "Impact of Processing Costs on Service Chain Placement in Network Functions Virtualization". In: *IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*. Nov. 2015, pp. 191–197 (cit. on pp. 138, 152).

[Suc+] Martin Suchara, Dahai Xu, Robert Doverspike, David Johnson, and Jennifer Rexford. "Network architecture for joint failure recovery and traffic engineering". In: *Proceedings of ACM SIGMETRICS 2011* (cit. on p. 82).

[SWB14] Gangxiang Shen, Yue Wei, and Sanjay K Bose. "Optimal design for shared backup path protected elastic optical networks under single-link failure". In: *Journal of Optical Communications and Networking* 6.7 (2014) (cit. on pp. 206, 216).

[Tel12] Nippon Telegraph. *Telephone Corporation,"Ryu Network Operating System."*. 2012 (cit. on p. 4).

[Tho+11] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song. "Design and evaluation of a real-time url spam filtering service". In: *IEEE Symposium on Security and Privacy (SP)*. 2011, pp. 447–462 (cit. on pp. 13, 172).

[TJG18] A Tomassilli, B Jaumard, and F Giroire. "Path Protection in Optical Flexible Networks with Distance-adaptive Modulation Formats". In: *2018 International Conference on Optical Network Design and Modeling (ONDM)*. 2018 (cit. on pp. 13, 203).

[Tod02] Michael J Todd. "The many facets of linear programming". In: *Mathematical Programming* 91.3 (2002), pp. 417–436 (cit. on p. 28).

[Tom+16] A Tomassilli, N Huin, F Giroire, and B Jaumard. *Energy-efficient service chains with network function virtualization*. 2016 (cit. on p. 12).

[Tom+18a] Andrea Tomassilli, F Giroire, N Huin, and S Pérennes. "Algorithmes d'approximation pour le placement de chaines de fonctions de services avec des contraintes d'ordre". In: *ALGOTEL 2018-20émes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*. 2018 (cit. on pp. 11, 43).

[Tom+18b] Andrea Tomassilli, Frédéric Giroire, Nicolas Huin, and Stéphane Pérennes. "Provably Efficient Algorithms for Placement of Service Function Chains with Ordering Constraints". In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 2018 (cit. on pp. 11, 43).

[Tom+18c] Andrea Tomassilli, Nicolas Huin, Frederic Giroire, and Brigitte Jaumard. "Resource requirements for reliable service function chaining". In: *2018 IEEE International Conference on Communications (ICC)*. IEEE. 2018, pp. 1–7 (cit. on pp. 11, 107).

[Tom+19] Andrea Tomassilli, Giuseppe Di Lena, Frédéric Giroire, Issam Tahiri, Damien Saucez, Stéphane Perennes, Thierry Turletti, Rusland Sadykov, Francois Vanderbeck, and Chidung Lac. "Poster: Design of Survivable SDN/NFV-enabled Networks with Bandwidth-optimal Failure Recovery". In: *Annex to the IFIP Networking 2019 Proceedings*. 2019 (cit. on p. 12).

[Tur+10] Daniel Turner, Kirill Levchenko, Alex C Snoeren, and Stefan Savage. "California fault lines: understanding the causes and impact of network failures". In: *ACM SIGCOMM Computer Communication Review*. Vol. 40. 4. ACM. 2010, pp. 315–326 (cit. on pp. 107, 108, 204).

[Van+94] Pamela H Vance, Cynthia Barnhart, Ellis L Johnson, and George L Nemhauser. "Solving binary cutting stock problems by column generation and branch-and-bound". In: *Computational optimization and applications* 3.2 (1994), pp. 111–130 (cit. on p. 29).

[Vaz13] Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013 (cit. on pp. 32, 59).

[Ver+11] Willem Vereecken, Ward Van Heddeghem, Margot Deruyck, Bart Puype, Bart Lannoo, Wout Joseph, Didier Colle, Luc Martens, and Piet Demeester. "Power consumption in telecommunication networks: overview and reduction strategies". In: *IEEE Communications Magazine* 49.6 (2011) (cit. on p. 136).

[VVK14] Niels LM Van Adrichem, Benjamin J Van Asten, and Fernando A Kuipers. "Fast recovery in software-defined networks". In: *Software Defined Networks (EWSDN), 2014 Third European Workshop on*. IEEE. 2014, pp. 61–66 (cit. on p. 80).

[Wax88] Bernard M Waxman. "Routing of multipoint connections". In: *IEEE journal on selected areas in communications* 6.9 (1988), pp. 1617–1622 (cit. on p. 96).

[WK13] Krzysztof Walkowiak and Mirosław Klinkowski. "Shared backup path protection in elastic optical networks: Modeling and optimization". In: *Design of Reliable Communication Networks (DRCN), 2013 9th International Conference on the*. IEEE. 2013, pp. 187–194 (cit. on pp. 206, 216).

[Wol98] Laurence A Wolsey. *Integer programming*. Wiley, 1998 (cit. on p. 29).

[Wor17] Marcel van Wort. *SDN and NFV transforming the network: where do we go from here?* 2017. URL: https://www.orange-business.com/en/blogs/connecting-technology/

networks/sdn-and-nfv-transforming-the-network-where-do-we-go-from-here (cit. on p. 2).

[Xu+04] Dahai Xu, Yizhi Xiong, Chunming Qiao, and Guangzhi Li. "Failure protection in layered networks with shared risk link groups". In: *IEEE network* (2004) (cit. on p. 82).

[Ye+16] Zilong Ye, Xiaojun Cao, Jianping Wang, Hongfang Yu, and Chunming Qiao. "Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization". In: *IEEE Network* 30.3 (2016) (cit. on p. 110).

[Ye91] Yinyu Ye. "An O (n 3 L) potential reduction algorithm for linear programming". In: *Mathematical programming* 50.1-3 (1991), pp. 239–258 (cit. on p. 28).

[Yin+09] Hao Yin, Xuening Liu, Tongyu Zhan, Vyas Sekar, Feng Qiu, Chuang Lin, Hui Zhang, and Bo Li. "Design and deployment of a hybrid CDN-P2P system for live video streaming: experiences with LiveSky". In: *Proceedings of the 17th ACM international conference on Multimedia*. ACM. 2009, pp. 25–34 (cit. on p. 61).

[Zah+12] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing". In: *USENIX conference on Networked Systems Design and Implementation*. 2012 (cit. on p. 172).

[Zha+18] Yuan Zhang, Lin Cui, Wei Wang, and Yuxiang Zhang. "A Survey on Software Defined Networking with Multiple Controllers". In: *J. Netw. Comput. Appl.* 103.C (2018), pp. 101–118. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2017.11.015 (cit. on p. 105).

[Zhu+13] Zuqing Zhu, Wei Lu, Liang Zhang, and Nirwan Ansari. "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing". In: *Journal of Lightwave Technology* 31.1 (2013), pp. 15–22 (cit. on p. 214).

[ZS00] Dongyun Zhou and Suresh Subramaniam. “Survivability in optical networks”. In: *IEEE network* 14.6 (2000), pp. 16–23 (cit. on pp. 108, 204, 206).