NESTED COLUMN GENERATION FOR OPTICAL NETWORK OPTIMIZATION

Huy Duong

A thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Doctor of Philosophy (Computer Science)
Concordia University
Montréal, Québec, Canada

August 2020

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Huy Quang Duong

Entitled: Nested Column Generation for Optical Network Optimization

and submitted in partial fulfillment of the requirements for the degree of

Doctor Of Philosophy (Computer Science)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Rajamohan Ganesan

_____ External Examiner
Dr. Issmail El Hallaoui

_____ External to Program
Dr. Ivan Contreras

_____ Examiner
Dr. Tristan Glatard

_____ Examiner
Dr. Emad Shihab

_____ Thesis Co-Supervisor
Dr. Brigitte Jaumard

_____ Thesis Co-Supervisor
Dr. David Coudert

Approved by _____
Dr. Leila Kosseim, Graduate Program Director

July 27, 2020 _____
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Nested Column Generation for Optical Network Optimization

Huy Duong, Ph. D.
Concordia University, 2020

Defragmentation/reoptimization of Elastic Optical Networks (EONs) reallocates connections to achieve an improved system, e.g., reducing the total required spare capacity or transmission delay. EONs comprises multiple layers which have different functionalities and management. This thesis studies two main layers of EONs that are Logical Layer and Optical Layer. Although defragmentation/reoptimization has many techniques and strategies, because of practical-application requirements, this work only discusses make-before-break (MBB) technique to reduce capacity/spectrum usage at defragmentation events predetermined by time-driven manner.

There are two directions in terms of solution strategies. In the first direction, network operator solves the original problem of finding the optimal state that MBB rerouting sequences can reach. This direction is hard to solve because MBB condition makes the problem complicated. In the second direction, it decomposes the problem into two steps. The first step computes the optimal state (target state) without MBB condition. And the second step finds a rerouting sequence to bring current state to the target state as close as possible under MBB condition. This direction is a heuristic because there is no assurance that network can reach the target state. However, this direction is easier to model and solve than the first direction.

For both directions, this work proposes several heuristic algorithms and sub-optimal algorithms using column generation for (nested) decomposition mathematical models. Our proposed models and algorithms enlarge the scalability of data sets in literature.

Acknowledgments

I am grateful to all people who have followed and given me the motivation to go through many challenges and experiences in this wonderful journey. First, I want to express my deep gratitude to my supervisor, Dr. Brigitte Jaumard, for all her guidance and support throughout my entire program at Concordia University. She has always been engaged and helpful in my work and ready with feedback and comments that assisted me during my studies and research. I also sincerely appreciate the collaboration and contribution of Dr. David Coudert to my work, National Institute for Research in Computer Science and Control, France. He is consistently willing to give me valuable comments for my proposals. It is also my great pleasure to work with, Todd Morris, Ron Armolavicius and Petar Djukic, a genuinely knowledgeable group from Ciena Ottawa, Canada.

I would like to thank my parents and my younger sister. I would not have had this amazing opportunity to pursue my dreams if it was not for your continuing support, encouragement and love; and for always believing in me. I have deeply appreciated my colleagues, my friends, especially to Mahesh Bakshi, for the fantastic atmosphere and attitude in our office.

Final thanks are due to MITACS for granting my valuable internships, and Concordia University for the financial support through the International Tuition Fee Remission award.

Contribution of Authors

This dissertation is presented in a manuscript-style. It contains five articles that have been accepted for publication or under revision in different journals, or in preparation. They are presented here as follows.

- Published:

    1. B. Jaumard, H. Quang Duong, R. Armolavicius, T. Morris, and P. Djukic, "Efficient real-time scalable make-before-break network re-routing," *Journal of Optical Communications and Networking*, vol. 11,p. 52, 03 2019.

- Under revision:

    1. H. Duong, B. Jaumard, R. Armolavicius and D. Coudert, "Efficient make before break defragmentation," *IEEE/ACM Transactions on Networking*. Under review (last revision)

- Submitted:

    1. H. Duong, B. Jaumard, R. Armolavicius, T. Morris, and P. Djukic, "MBB Defragmentation with CD and CDC ROADMs in Elastic Optical Networks".
    2. H. Duong, B. Jaumard and D. Coudert, "A nested decomposition model for generalized MBB reoptimization".
    3. H. Duong and B. Jaumard, "A nested decomposition model for reliable NFV 5G network slicing".

The author of this thesis acted as the principal researcher with the corresponding duties of developing mathematical formulations, proofs, developing and implementing the algorithms and analysis of computational experiments along with the writing of all articles.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The world of telecommunications is now moving towards 5G, the fifth generation of telecommunication networks, and there are already discussions for beyond 5G, i.e. B5G. We expect the 5G technology to be a dramatic leap from the current 4G LTE networks. The additional features of 5G technology will offer advanced services, followed by the diversity of performance required. Especially, the concept of 5G is to use a single shared system that supports multiple tenants (customers). Since we expect network traffic to grow in an explosive and very dynamic way, providers are urgently seeking for efficient 5G network operating solutions. These solutions address the 5G challenges, e.g., ubiquitous connectivity, ultra-high data rates, extremely low latency, tremendous energy-saving, network function virtualization, software-defined networking [4, 5]. One of the important aspects of the network operation is resource fragmentation [6, 1], i.e., network resource re-optimization.

Several layers with different functionalities make up optical networks. Fragmentation may occur at any layer and can be defined as follows. Because of dynamic addition and termination of requests/connections, network provisioning is not always possible on shortest paths. Note that, to establish a path, all the links on the path must have enough available resources, in addition to potential continuity or contiguity constraints with respect to channel allocation. When we provision a request, we do so on the best available combination of route and spectrum/bandwidth resources. However, due to Service Level Agreements (SLAs), it is not interrupted and

rerouted as soon as a better path becomes available. The state, where connections are using longer paths than the shortest available ones, corresponds to what we call a fragmented network, and the process is called fragmentation.

However, later on, the state of the network changes because of the dynamic connection adds and drops, and fragmentation increases. Thus a global provisioning re-optimization is required to bring the connections to an optimal (or optimized) state. This process is called defragmentation (or reconfiguration). In this thesis, we propose novel mathematical models and algorithms to improve further defragmentation/reconfiguration operations. Our defragmentation problems and solutions are developed with technical and directional support of Ciena Corporation (our industrial partner).

1.2 Single-Layer and Cross-Layer Defragmentation/ Reoptimization

Logical and Optical Layer are two main layers of Optical Networks (ONs), and they are depicted in Figure 1. Besides, the Optical Layer usually comprises several proprietary domains belonging to different providers and using different technologies. Thus, these two layers work together as a transparent service. It means that Logical Layer operates on its logical networks independently of how Optical Layer manipulates actual lightpaths.

In 5G, communication among layers is simplified thanks to Software Defined Network (SDN), Network Function Virtualization (NFV) and Virtual Network Functions (VNFs). Thus logical networks are constructed and updated more easily. It also enables a paradigm that is network slicing. Network slicing refers to several virtual networks are built on the same infrastructure. Figure 2 shows an example of network slicing.

From this discussion about layer management, one can see that developing a comprehensive defragmentation/reoptimization scheme, taking into account both layers at once (cross-layer defragmentation), is complicated. Therefore, the idea of this thesis is first to study single-layer defragmentation/reoptimization problems (defragmentation with respect to one layer), then combine them into a cross-layer scheme.

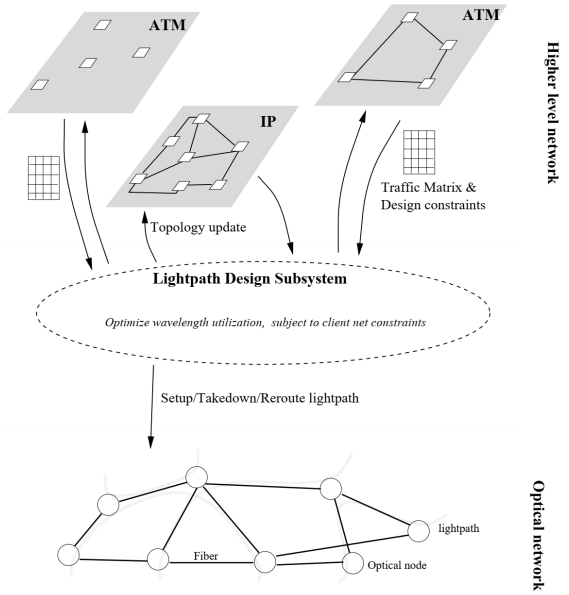


Figure 1: Overview of Optical Network's architecture [2]

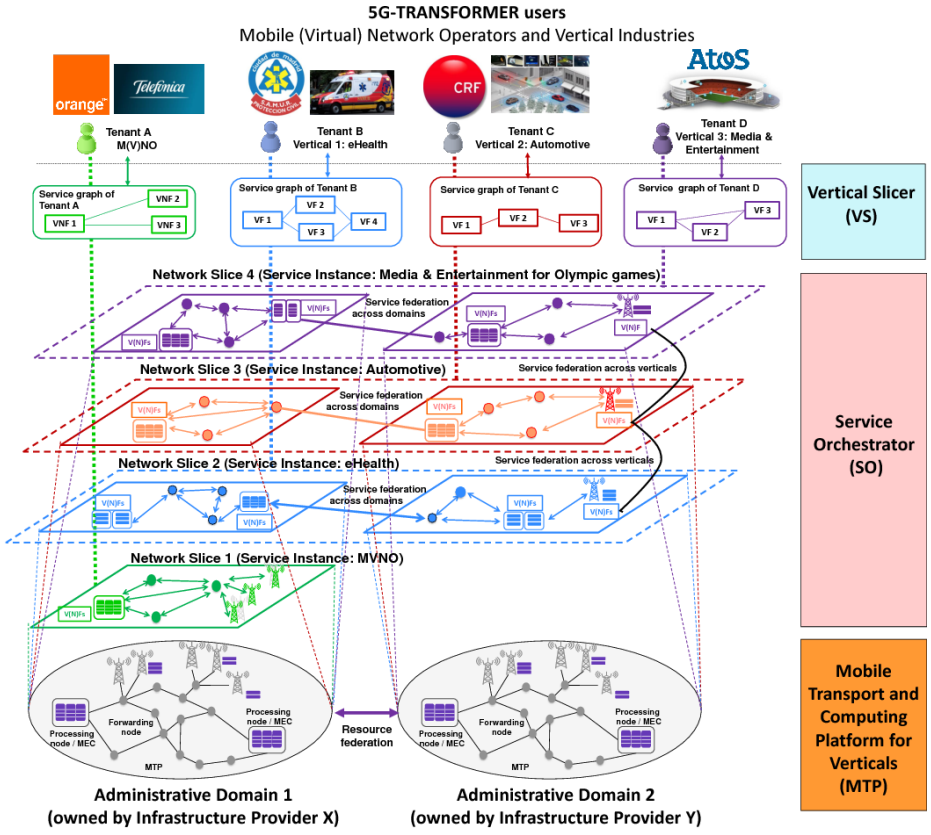


Figure 2: 5G Network architecture [3]

1.3 Defragmentaion/Reoptimization Paradigms and Strategies

In practice, a defragmentation solution has to guarantee Quality of Service (QoS) requirements while minimizing the spectrum/bandwidth usage. Indeed, when the network is well defragmented, it provides additional provisioning possibilities for the incoming connection request(s), as well as a longer period until the next defragmentation is triggered. Note that defragmentation events are usually triggered using a network load threshold or a time-driven criterion. Our industrial partner advises to only consider fixed-schedule defragmentation because service agreements do not allow service modification at any time. Usually, defragmentation/reconfiguration is only performed during night when traffic is lowest. Thus this thesis only focuses on time-driven defragmentation/reoptimization.

In terms of connection disruption, the defragmentation/reoptimization process can be classified into hitless or non-hitless paradigms [7]. The latter means a connection is temporarily interrupted before switching to a new route. Connection disruption degrades severely Quality of Service, thus it is not used in this thesis. On the other hand, hitless refers to techniques by which data transferring is continuous (or almost continuous) during rerouting.

At Optical Layer, several hitless techniques depend on the system's technology, e.g., push-pull, hop tuning and make before break. Push-pull and hop-tuning techniques allow the spectrum of a connection to move along the link's spectrum. However, they require special devices. On the other hand, the make-before-break paradigm establishes the new route of a connection, and transportation gradually switches to the new route. When there is no need for the old route, the connection is completely switched and the old route is torn down. This technique does not require special devices and ensures better service continuity. Our industrial partner also prefers a make-before-break paradigm. The make-before-break paradigm can also be used at Logical Layer. Therefore, this thesis focuses only on the make-before-break paradigm.

In terms of solution strategies, researchers have investigated this connection rerouting along two directions. The first one comprises two phases. The first phase computes the optimal state (target state) without MBB condition. The second phase finds a

rerouting sequence to bring current state as close as possible to the target state under MBB condition. The second direction is to compute the best provisioning that is reachable from the legacy provisioning by a sequence of connection reroutings with no disruption, i.e., MBB paradigm, but no target state is pre-computed.

Although the first direction is not desirable as the second direction, the first direction is easier to model and propose algorithms for each phase. Indeed, the first direction often lacks the computation of an optimized provisioning that is the closest one from the current provisioning, and then as a consequence, may take many reroutings in order to be reached. For instance, for wavelength assignment, there is an exponential number of them which are equivalent up to a permutation, while there are a very limited number of them that are close to a given provisioning. Besides, we notice that the optimal state without MBB is a lower bound of MBB reachable solution (in the case of minimum bandwidth requirement). As a consequence, if the second phase yields solution that is very close to the target state, then it is also very close to one of the best MBB reachable states. Thus, in this thesis, we investigate both directions.

1.4 Thesis's Plan

This thesis is composed of six more chapters, five of which correspond to manuscripts that have been either published or submitted for publication in international networking-related journals, except for the last one that is under preparation. Since this thesis has a manuscript-style, each chapter is self-contained and corresponds to a manuscript.

This thesis comprises three main parts. The first part focuses on Logical Layer reoptimization, the second one studies Optical Layer defragmentation, and the last part proposes a framework to optimally distribute resources over network slices in 5G networks. As discussed above, in given-target reoptimization, finding the optimal resource allocation is a necessary phase.

Chapter 2 proposes efficient heuristics and ε-optimal solutions for Logical Layer reoptimization problem where target state is given (first direction as defined in previous section) while Chapter 3 proposes an ε-optimal solution for true MBB reoptimization (second direction as in previous section). In these two chapters, the problem is

simplified to no parallel rerouting at any given time and no more than one rerouting per connection (non-multiple). In Chapter 4, parallel and multiple rerouting is considered.

Optical Layer defragmentation is studied in Chapter 5. Chapter 6 proposes a framework to optimally (with a small optimally gap) allocate network slices on the same physical infrastructure. Finally, Chapter 7 lays out conclusions and future research directions.

Note that, all ε-optimal solutions in this thesis are post evaluated, i.e., the gap is computed after solutions and bounds are obtained. As these obtained gaps are less than 5% for all proposed algorithms, there is no need to develop gap-guaranteed algorithm. In addition, for all decomposition models, solving the last master problem as an integer linear program was enough in order to reach a solution with a reasonable optimality gap, Therefore, we did not investigate diving heuristics and branch-and-price techniques as in, e.g., [8].

Chapter 2

Efficient real-time scalable make-before-break network re-routing

Dynamic traffic in optical networks leads to capacity fragmentation, which significantly reduces network performance. Consequently, non disruptive re-routing has attracted a lot of attention from network providers as it dramatically improves the amount of traffic that can be granted with a good quality of service. The recent development of flexible and software-defined networks is urgently calling for faster real-time reconfiguration algorithms at all network layers, while being highly efficient in terms of optimized reconfiguration of connections.

We propose a very efficient non disruptive bandwidth defragmentation heuristic (called MBB_DF_H) with a make-before-break (MBB) strategy at the user layer. In order to assess its accuracy, we also develop a near-optimal two-phase defragmentation process. Firstly, we formulate an exact optimization model for the decomposition scheme, in order to compute the best re-routing, i.e., with minimum bandwidth requirement. Secondly, we investigate whether the resulting re-routing can be reached with a MBB policy, if not, we exhibit the best possible MBB re-routing.

We conduct extensive numerical experiments on several data sets. We first run experiments on small to medium data sets so that the solution efficiency of the MBB_DF_H heuristic can be compared with the two-phase optimization models. Next, we focus on the real time scalability of the MBB_DF_H heuristic on large realistic data sets.

This work was published as: B. Jaumard, H. Quang Duong, R. Armolavicius,

T. Morris, and P. Djukic, "Efficient real-time scalablemake-before-break network re-routing," *Journal of Optical Communications and Networking*, vol. 11,p. 52, 03 2019.

2.1 Introduction

Dynamic traffic in optical networks leads to capacity fragmentation, which can reduce network efficiency. Specifically, in time division multiple access (TDM) networks based on the optical transport network (OTN) electrical switching protocol, dynamic arrivals and departures of end-to-end connections can lead to "stranded capacity". Stranded capacity consists of pockets of capacity available on individual links, which cannot be strung together as contiguous multi-link bandwidth for end-to-end paths. For the network operator, this issue manifests itself as either unnecessarily inflated capital expenses – adding capacity when it is not really needed, or causing reduced revenues – through blocking connection requests, which could otherwise be served with the same equipment.

One way to reduce stranded capacity in an optical network is to periodically optimize network usage by defragmenting the bandwidth. The goal of the defragmentation is to decrease total network usage, thus expanding the pockets of stranded capacity into contiguous bands of capacity available end-to-end. Generally speaking, the defragmentation process proceeds as a series of "re-routings" where an existing end-to-end network connection is re-routed on a new, better path. To prevent reduced quality-of-service (QoS) to client services, a re-routing should be done in make-before-break fashion (MBB), that is a new path for an existing connection is reserved, before the connection's traffic is electrically switched on this path and the old path is torn-down. It is also important that the number of re-routings is kept at a minimum so that the operator can optimize the network in its maintenance windows.

We note that the MBB can be accomplished by either: (1) reserving sufficient bandwidth throughout the entire new path and then switching the entire path at once, or (2) reserving the bandwidth on the new path only where the old path does not currently pass and then switching only the new segments of the new path, by re-routing each segment using MBB. The availability of each approach is dependent on the control plane capabilities. We know that at least some equipment vendors support the second approach and we assume throughout the paper that it is available, due

to its advantages – namely, a connection can be re-routed without having to reserve twice its capacity for the re-routing process.

Note that all defragmentation schemes, including make-before-break schemes, incur a "switching" disruption time. However, schemes that require connections to be torn down before moving them to a better path (break-before-make) incur longer disruption times. Thus, make-before-break is preferred to minimize disruptions

A key question for the optimization process is: what is the order in which connections should be moved so that the maximum amount of capacity is released, while moving the least number of connections?

Answering this question is a computationally hard problem, unlikely to yield a computationally scalable exact solution. So, it is necessary to devise effective heuristics, while being able to estimate the quality of their solutions. To this end, the contributions of this paper can be summarized as follows:

- A real-time highly scalable defragmentation heuristic, called MBB_DF_H, that aims to reduce the bandwidth requirements. Our heuristic uses a novel ranking criterion, called worst offender, which we found through extensive simulations to be better than many other heuristics and to perform close to optimum.

- In order to assess the quality of the MBB_DF_H heuristic, we develop original mathematical optimization models with a three-step approach.

  - A first optimization model, called DEFRAG, relying on a column generation (Dantzig-Wolfe) decomposition algorithm, called DF_ILP, which computes the best possible connection re-routing without worrying about MBB.
  - A second decomposition optimization model, called MBB_REACH, also relying on a column generation decomposition algorithm, called MBB_DF_ILP. Model MBB_REACH checks the reach-ability of a given connection re-routing with a Make-Before-Break process. Not only does it allow a good performance for a fair evaluation of the MBB_DF_H heuristic with the solutions of DF_ILP, but it also helps assess the difficulty of seamlessly reaching an optimized provisioning from a given current provisioning.
  - Finally, the two solutions are combined to deduce an ε-optimal solution, which can be used to assess the quality of the MBB_DF_H heuristic.

- Extensive simulations on real-network topologies show that the MBB_DF_H heuristic has an almost optimal performance on small to medium size instances. While the scalability of the optimization models have been greatly improved in comparison with the previous work [1], it is not yet possible to conduct comparative experiments on large data instances.

It is worth reflecting on what the defragmentation problem implies from a technological perspective. The introduction of the control plane technology, based on the Open Systems Routing Protocol (OSRP) [9] in conjunction with OTN has greatly simplified the instantiation of new OTN services. However, due to its distributed nature, this protocol is limited to finding shortest paths in the network. As the network becomes more dynamic, the shortest path protocol leads to stranded bandwidth, requiring periodic network optimization. At the same time, the network management systems (NMSs) are becoming more advanced and are able get a more holistic view of the network, its resource usage, as well as a way to re-route connections in a centralized fashion. This evolution of NMSs is now enabling network optimization, making practical network optimization algorithms an industry relevant problem.

We also note the applicability of the algorithms proposed in this paper in the context of packet networks using multiprotocol label switching (MPLS) and using a centralized path computation element (PCE) or a network controller in software-defined networking (SDN). In packet networks, the global concurrent optimization (GCO) is intended to be used to re-optimize the labeled-switched paths (LSPs) in order to improve network efficiency [10, 11]. The algorithms presented in this paper can be used in this context as is.

The paper is organized as follows. In the next section, we review the key references. A concise statement of the defragmentation problem in the user layer and the notations which are shared by all models and algorithms are introduced in Section 2.3. Section 2.4 describes a MBB defragmentation heuristic, called MBB_DF_H heuristic. In Section 2.5, we propose an exact decomposition model, called DF_ILP in order to identify the best re-routing in terms of minimum bandwidth requirement. The solution scheme allows the generation of an ε-optimal routing. We then investigate in Section 2.6 a mathematical model and an algorithm that check whether a given optimized routing can be reached with a MBB process. Extensive numerical results are presented in Section 2.7 where we first evaluate the accuracy of the solutions output

by the MBB_DF_H heuristic thanks to the ILP (Integer Linear Program) decomposition models (DF_ILP and MBB_REACH). Lastly, we demonstrate the real-time scalability of the heuristic on large realistic data instances.

2.2 Literature Review

We review the references for the defragmentation at the user layer, i.e., capacity deframentation, a problem which has been much less studied than defragmentation at the optical layer (i.e., either wavelength or spectrum defragmentation). We first discuss some generalities and why the graph theory tools used for wavelength/spectrum defragmentation do not apply for capacity defragmentation, and then review the previously proposed algorithms and models.

2.2.1 Defragmentation Strategies

Defragmentation mechanisms can be classified into reactive and proactive mechanisms. Reactive mechanisms are invoked when a request cannot be accommodated given the network-wide capacity allocation. In proactive mechanisms, the capacity allocation is performed in a manner that unused spectrum slots are proactively preserved for future use, such as the push-pull mechanism in spectrum defragmentation [12].

Several of the works on wavelength/spectrum defragmentation use the concept of dependency graph ([13, 14]) in order to identify the order in which the connections/requests can be re-routed as to maximize the number of re-routed requests, while attempting to minimize the bandwidth requirements. The dependency graph is such that each node is associated with a request and there is an arc between two requests if one connection needs to be re-routed before the other one in order to be able to achieve a make before break re-routing.

Such a graph cannot be reused in the context of capacity defragmentation as the position of the requests in the spectrum is not fixed, e.g., no fixed center frequency and fixed wavelength/slot spacing. As a consequence, wherever there is a unique way to resolve a conflict in the case of wavelength or slot defragmentation, there might be several solutions in the case of capacity defragmentation, i.e., re-route first connection k_1 or connection k_2 in order to be be able to re-route connection k_3 assuming

10G are needed to re-route k_3 while both k_1 and k_2 are 10G connections. A possible direction would be to use a so-called AND/OR graph [15]. However, AND/OR graphs are difficult to use due to the complexity of detecting circuits/cycles in them [16]. Another observation is that the dependency graph used for Layer 0 defragmentation takes into account the continuity/contiguity constraints of RWA/RSA problems, constraints which do not exist for Layer 3 defragmentation.

However, algorithms that do not use the concept of the dependency graph can be re-used (with some adaptations). They are reviewed in the next two sections.

2.2.2 Defragmentation with Known Optimized Re-routing

Most capacity defragmentation algorithms rely on a 2-phase process. The first phase is equivalent to the computation of an optimal network provisioning if we do not worry about the number of re-routings to reach it, which all proposed algorithms do. Such a problem has been widely studied since the beginning of network optimization, see, e.g., [17, 18, 19, 20].

The second phase deals with the best way to reach the optimum/optimized network provisioning. It has been studied with different objectives, i.e., minimize the number or the duration of the disruptions, or minimize the largest number of simultaneous disruptions, or minimize the maximum duration of a disruption. In agreement with the focus of this paper, we will focus on the minimum bandwidth requirement.

In the context of Layer 3, references for the first phase can be found in the literature, but generally for a purpose other than to re-optimize the bandwidth or spectrum usage, e.g., Maggi *et al.* [21] (finding a set of make-before-break compliant MPLS connection path changes to optimize a network's connection state while avoiding cases where make-before-break is infeasible due to capacity limitations through rate limiting some of the connections), Tajiki *et al.* [22] (reconfiguration of packet flows in a network triggered by congestion events or periodically, restricted to the processor interconnection network within a data center). Wang and Li [23] (IP/MPLS fast re-route speed up using precomputed restore paths as close to the failure point as possible).

2.2.3 MBB-guaranteed Reconfiguration

The strategy of MBB-guaranteed Reconfiguration is to re-route connections one by one under the condition that they can be carried out with a MBB condition [1, 24, 25]. While the latter direction guarantees a MBB reconfiguration, it can be at the expense of not getting an optimal routing.

While the above sections review the first direction, we present here several works along the second direction. Olinick and Rahman [25] proposed models which reconfigure the network to offer as much available space as possible for a new demand. However, these models are restricted as they only work with a set of precomputed paths. Jozsa and Makai [24] focus their attention on the existence of a MBB re-routing, assuming the initial (current) and the target (optimized) re-routing are given. They propose various heuristics, which are not very successful (around 50% of the cases for their data instances) at finding a MBB re-routing, even when such a re-routing exists. See also Józsa, Orincsay and Tamási [26] for an improved version of their heuristics.

Klopfenstein [1] attempted to guarantee MBB in a one step process: a move is carried out only when it is MBB feasible. This idea is similar to the one of Olinick and Rahman [25]. However, the proposed ILP is only able to solve toy examples.

2.3 Capacity Defragmentation: Statement of the Problem and Notations

2.3.1 Dynamic Granting & Routing and Capacity Defragmentation

On-line granting and denying of connection requests (call admission control) are periodically subject to a defragmentation of the network bandwidth, which re-routes admitted connections, as depicted in Fig. 3. The objective of this study is to investigate the alternation of the granting and routing phases with the defragmentation phases, their mutual impact, and their efficiency.

More precisely, we propose to evaluate the performance of a two-phase heuristic, which consists of: (i) the first (routing and granting) phase, referred to as G&R_H, in which requests are dynamically granted and routed if granted and (ii) the second (defragmentation) phase, called MBB_DF_H, which consists of a MBB re-routing

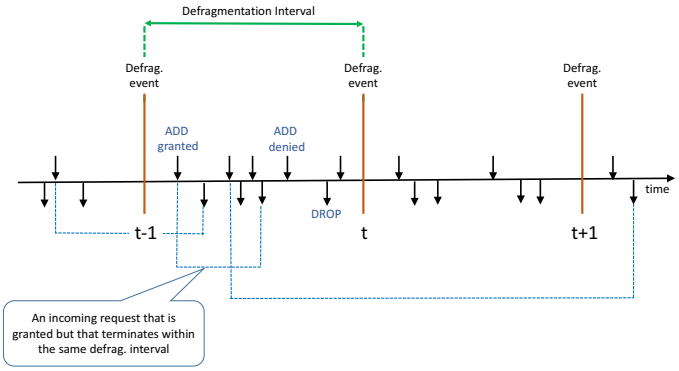


Figure 3: On-line Routing vs. Defragmentation

strategy, i.e., re-route as many requests as possible while minimizing the bandwidth usage. Details of both phases are described in Section 2.4.

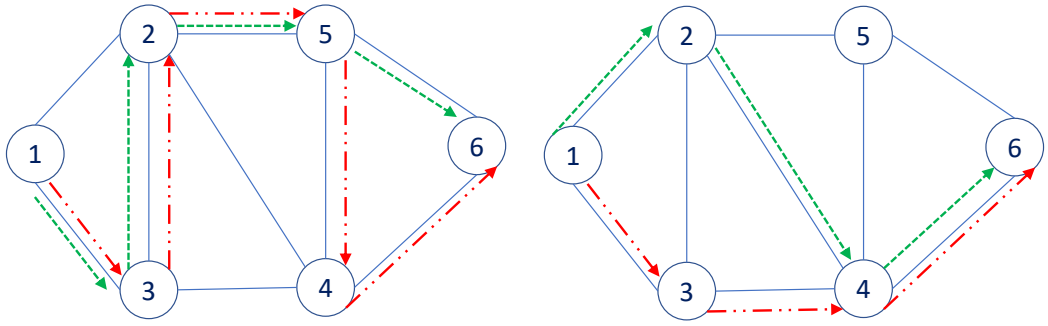
In order to evaluate the G&R_H heuristic, we will compare it against a simple on-line heuristic, called the G&RSP_H heuristic, which grants and routes the granted requests according to a simple shortest path strategy, which is described in Section 2.7.4.

In order to evaluate the performance of the MBB_DF_H heuristic, which alternates with either the G&R_H or the G&RSP_H heuristic, we next develop two optimization models. The first optimization model determines the best possible re-routing, in terms of bandwidth requirements, at the same defragmentation events as the MBB_DF_H heuristic. The second model, called MBB_REACH, checks whether the optimized re-routing solution of DF_ILP is reachable, if not, it provides the best MBB reachable re-routing, in terms of the number of re-routed connection requests.

In practice, we looked at several defragmentation intervals (i.e., 10, see Section 2.7 for the details) to see the impact of the granting process on the bandwidth gains and the blocking probabilities after a defragmentation event.

2.3.2 Make Before Break

The basic make-before-break (MBB) strategy aims to facilitate hitless re-routings. The connection is re-routed by first reserving bandwidth on the new route and then the traffic is transferred from the old route onto the new route. Note that the new route does not necessarily need to be link disjoint with the old one, see Fig. 4 for an



(a) Before Re-routing (b) After Re-routing

Figure 4: MBB Re-routing

illustration, where although the green routes are link disjoint, the red routes are not, but MBB can be achieved in both cases.

We assume that the control plane in the network is able to perform a partial make-before-break, where the shared segments in the original and the new routes are left in place. For example in Fig. 4, the control plane is smart enough to realize that the red path does not need to be fully re-rerouted. The control plane works from node 6 towards node 1 and leaves link 4-6 in place, reserves bandwidth between 3 and 4, re-routes sub-path 3-2-5-4 to sub-path 3-4 with MBB, and leaves link 1-3 in place. Only the segments which are not shared by the two routes are re-routed, and consequently partial MBB does not require double bandwidth on shared links for connection re-routing.

Now, if the new connection is kept totally independent of the existing one, they are bound to compete for the resources on the network segments which are common to both of them. Depending on the free resource availability, this competition can prevent an admission control to allow establishment of new routes. We assume that, in such a case, both connections, the new and the old ones, share the resources on common links. For instance, in the context of MPLS, RSVP-TE SE (shared explicit) reservation relate the new and old LSPs to each other to give them a sense of awareness that they are not totally independent of each other and they must share the resources on common segments.

2.3.3 Notations

We define all the mathematical notations we will use throughout the paper.

The network is represented by a directed multi-graph $G = (V, L)$ where L is the set of links (indexed by ℓ), each link being associated with a logical link (so, if there are two logical links from v to v', it leads to two links ℓ and ℓ') and V is the set of nodes (indexed by v). We denote by $\omega^+(v)$ and $\omega^-(v)$ the sets of outgoing and incoming links of node v, respectively. Let $\mathcal{SD} \subseteq V \times V$ be the set of node pairs with some requests/traffic, i.e., $(v_s, v_d) \in \mathcal{SD}$ if there exist requests from v_s to v_d.

We denote by C_ℓ the transport capacity of link ℓ.

Let T be the set of reconfiguration time events (indexed by t, when $t = 0$, it is initial state).

Let K_t be the set of requests (indexed by k) right before the time t defragmentation event occurs. It defines the set of legacy requests, i.e., on-going requests already granted and routed at the previous defragmentation event (at time $t-1$), minus the set of requests that terminated between times $t-1$ to t, plus the set of new incoming and granted requests between times $t-1$ to t, which are still on-going. For a given request k, let s_k be its source node, d_k its destination node, and b_k be the bandwidth requirement (expressed in, e.g., ODUflex units, see Table 1) and D_k be its duration.

Let $K_t^{sd} \subseteq K_t$ be the set of requests associated with node pair (v_s, v_d) at the defragmentation time t.

OTU	ODU	Marketing Rate	True Signal (OTU)	True Payload (OPU)
	0	1.25G	NA	1.238G/s
1	1	2.5G	2.666G/s	2.488G/s
2	2	10G	10.709G/s	9.953G/s
3	3	40G	43.018G/s	39.813G/s
4	4	100G	111.809G/s	104.794G/s

Table 1: OTN rates (ITU-T recommendation G.709, commonly called Optical Transport Network (OTN)

We need some additional notation with respect to paths. Let n_k^{SP} (resp. n_{sd}^{SP}) be the number of links in a shortest path from source to destination of request k (resp. the number of links in a shortest path from source v_s to destination v_d). We denote by P_k a set of routes that can be used to provision request k, P_{sd} a set of routes

that can be used to provision requests from v_s to v_d, P_k^{SHORTEST} the set of all possible shortest paths that can be used to provision request k, and P_{sd}^{SHORTEST} the set of all possible shortest paths that can be used to provision requests from v_s to v_d.

2.4 A Defragmentation Heuristic: MBB_DF_H

2.4.1 Defragmentation Context

The context is a connection-oriented network where setup and release requests arrive at random. Setup requests signal the duration of each connection and the required bandwidth. A routing path is defined to be a set of network links that support a connection: the length of the path is the number of links in the path. The network path computation algorithm attempts to find the shortest path whose links have sufficient free capacity to support the connection. If one is found, the connection is established and resources are allocated, otherwise the connection request is blocked and cleared from the network.

As the network evolves and free link bandwidth decreases, connections are established using paths that are the shortest possible at the time of setup but could still be long in comparison to what is possible in an empty network. Additionally, these paths may be rendered sub-optimal as connections are released and bandwidth is freed suggesting it may be beneficial to periodically move existing connections to shorter paths as resources become available. The goal is to decrease resource usage allowing more connections into the network.

2.4.2 MBB_DF_H Heuristic

We now describe the MBB_DF_H optimization algorithm that runs at fixed intervals and attempts to recover poorly allocated network resources by moving connections to better (shorter) paths. The optimization algorithm can be outlined as follows:

1. The algorithm is invoked periodically at fixed intervals (in the experiments in Section 2.7), or, in practice, is triggered by network conditions, or may be run during maintenance windows.

2. Each active connection k is assigned a weight, which represents the excess resources (computed as shown below) required by the connection using the current

path over what would be needed using the ideal path, i.e., the shortest possible one. We call this "worst offender" ranking.

3. The connections are sorted by decreasing weight and are assigned a sequence number that reflects this order. The smallest sequence numbers correspond to the "worst offenders" in terms of resource usage.

4. The algorithm then proceeds to simulate connection re-routings. Each connection is considered as a move candidate in sequence order so that the worst offenders are considered first. A connection under consideration is taken down, virtually, releasing allocated resources, and the routing algorithm is used to find a new path for the connection: if a new path is found, potentially reusing links in the existing paths, and if the length of the new path is less than that of the existing path then the connection is marked to be moved.

Before selecting the worst offender, we tried many different definitions of the weight. We discuss below some of the attempts and comparisons we made.

For a given connection k at the time of defragmentation, let B_k be its bandwidth requirement, D_k its (remaining) duration, h_k its number of links/hops in its current routing, $h_k^\star$ its number of links in an "ideal" routing (shortest path in an otherwise empty network).

We explored the following weights through simulation:

1. W_k = random (connections are selected uniformly for defragmentation until all have been considered)

2. $W_k = B_k \times h_k$

3. $W_k = B_k(h_k - h_k^\star)$ - worst offender (excluding duration)

4. $W_k = D_k B_k h_k$

5. $W_k = D_k B_k (h_k - h_k^\star)$: worst offender.

Cases 2 and 3 apply when the duration of a connection is not known at the time of establishment; cases 4 and 5 apply when the duration of a connection is known when it is established. The weights primarily affect the rate at which bandwidth recovery is achieved through incremental defragmentation since in all cases if all connections are defragmented (not always successfully) a similar degree of bandwidth recovery is achieved. In general case 1 produces the least desirable rate of bandwidth recovery, approximately linear in the number of connections defragmented. Cases 2 - 5 produced a geometric bandwidth recovery profile, with the first few connection defragmentations generating the greatest recovery. This behaviour allows the defragmentation process to be terminated early: the weights that produce the greatest initial rate of recovery minimize the amount of "work" needed to reach an acceptable degree of bandwidth recovery. The recovery rates for cases 2 and 3 (respectively 4 and 5) were similar with case 3 (respectively case 5) slightly higher due to the ability of the worst offender metrics to measure the excess resources required by the current connection over the ideal as opposed to just the total resources needed (which may in fact be close to the optimal resources required). Only case 5 was reported in the paper as it was assumed that connection durations were known and because this case tended to produce the most rapid incremental bandwidth recovery.

Algorithm MBB_DF_H lists the steps of the heuristic. During a defragmentation run, it is assumed that the set of connection requests (K) is fixed, admitting no new connection requests or releases. Each connection request k is characterized by:

- s_k, d_k, D_k, B_k, its source, destination, remaining duration, and bandwidth;
- L_k, the set of links associated with its routing with number of hops h_k;
- `#Moves`(k), the number of times connection k has been moved: when the algorithm starts, `#Moves`(k) is initialized to 0 for each connection k as shown in lines 1-2.

Each defragmentation pass begins by sorting the connection set by worst offender status (line 4). According to the definition of worst offender, the weight W_k of connection k is computed as

$$W_k = B_k D_k (h_k - h_k^\star), \tag{2.1}$$

where B_k is the bandwidth assigned to the connection, D_k is the remaining duration of the connection, h_k is the number of hops the connection is using right now, and $h_k^\star$

Algorithm 1 MBB_DF_H

```
1: for k ∈ K do
2:     #Moves(k) ← 0
3: for Passes = 1 To GroomingPasses do
4:     Sort the connections in K
5:     for k ∈ K do
6:         if #Moves(k) < ConnectionMoveLimit then
7:             if Not OnBestPath(k) then
8:                 NewPath ← FindPath(k)
9:                 if NewPath.Length(k) < Path.Length(k) then
10:                    Allocate(k, NewPath)
11:                    LeastHops = MinHops(k)
12:                    OnBestPath(k) ← (NewPath.Length(k) = LeastHops)
13:                    #Moves(k) ← #Moves(k) + 1
14:        Next k
15:    Passes → Passes + 1
```

denotes the number of hops that would be required by connection k using the shortest path in the empty network.

Sorting according to the worst offender allows the best move candidates to be considered earliest in the defragmentation pass minimizing the defragmentation effort by permitting an early exit of the pass after a `ConnectionMoveLimit` (line 6) is reached.

The defragmentation algorithm performs `GroomingPasses` defragmentation passes through the connection set K (line 3): consequently a connection k could be moved up to `GroomingPasses` times unless limited by `ConnectionMoveLimit` (line 6). If `ConnectionMoveLimit` = 1 each connection is moved at most once no matter how many passes are used. The idea of `GroomingPasses` is to give an additional chance for re-routing a given connection, once some other connections have been re-routed and their excess resources have been released.

Each pass considers each connection k in sort order and tries to find a shorter route `NewPath` (lines 8-9) using simple minimal hop routing. It should be noted that the procedures `FindPath` and `Allocate` are extensions of what would be used for

normal routing and connection setup in that they are able to reuse links that do not need to be moved to establish the new path and are able to ensure make-before-break connection setup. Finally, `MoveCount` keeps track of the total number of moves and can be used to limit the total number of moves permitted per defragmentation cycle.

2.5 An Exact Re-routing Model and a ε-optimal Solution Scheme

We propose here an exact mathematical re-routing model, called DF_ILP, aiming at minimizing the bandwidth requirement. It relies on a decomposition scheme with node pair configurations (Section 2.5.1), and is presented in Section 2.5.2. We discuss how to efficiently solve the DF_ILP model with a column generation solution scheme in Section 2.5.3.

2.5.1 Node Pair Configurations

A configuration, denoted by c, is associated with a given node pair, say (v_s, v_d). It is characterized by a set of paths from v_s to v_d and the set of requests routed on these paths, i.e., which request from K_t^{sd} is routed on which path $p \in P_{sd}$.

Let C be the overall set of configurations:

$$C = \bigcup_{(v_s,v_d)\in\mathcal{SD}} C_{sd},$$

where C_{sd} is the set of configurations associated with node pair (v_s, v_d).

A configuration $c \in C_{sd}$ is characterized by:

- $\delta_{k\ell}^c \in \{0, 1\}$, where $\delta_{k\ell}^c = 1$ if link ℓ is used for the route of request k in configuration c, 0 otherwise.
- Note that in the worst case, the routing is unchanged, and therefore, the DefragILP model has always a feasible routing for request k, assuming that $k \equiv (v_s, v_d) \in \mathcal{SD}$.

The DF_ILP model that will be described in the next section needs to choose a unique configuration per node pair $(v_s, v_d) \in \mathcal{SD}$ for all node pairs, the best one so as to grant the overall set of alive requests. See an illustration in Fig. 5.

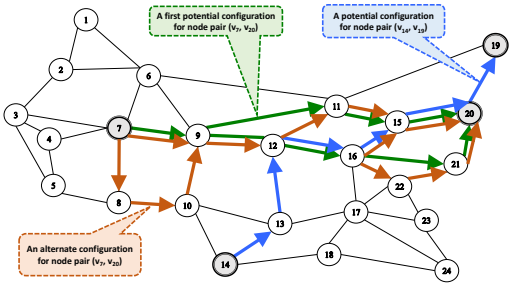


Figure 5: Examples of node pair configurations

2.5.2 DF_ILP Model

We now detail the DF_ILP model, an exact mathematical model, which aims at finding the best re-routing at a given defragmentation point. We consider the objective of minimizing the network capacity, i.e., sum of the link capacities over all the routes of the granted requests. The DF_ILP model requires only one set of variables: $z_c = 1$ if route configuration c is selected, 0 otherwise.

$$\min \quad \sum_{c \in C} \left(\sum_{k \in K_t} \sum_{\ell \in L} D_k \delta_{k\ell}^c \right) z_c \tag{2.2}$$

Constraints are written as follows.

- Capacity constraints.

$$\sum_{(v_s, v_d) \in \mathcal{SD}} \sum_{k \in K_t^{sd}} \sum_{c \in C_{sd}} D_k \delta_{k\ell}^c \, z_c \leq C_\ell \qquad \ell \in L \tag{2.3}$$

- Choose one configuration per node pair

$$\sum_{c \in C_{sd}} z_c = 1 \qquad (v_s, v_d) \in \mathcal{SD} : K_t^{sd} \neq \emptyset$$
$$\text{at least one legacy } v_s \rightsquigarrow v_d \tag{2.4}$$

- Domain of the variables

$$z_c \in \{0, 1\}. \tag{2.5}$$

2.5.3 Solution Process

As the number of variables in the DF_ILP model is exponential, we need to recourse to column generation techniques for solving its linear relaxation, and deduce an ILP ε-optimal solution or an optimal solution using branch-and-price methods [27].

We will use the following terminology, that is classical, in the decomposition method literature [27, 28].

Master Problem (MP): Problem (2.2) - (2.5) with all z_c variables.

Restricted Master Problem (RMP): Problem (2.2) - (2.5) with a very small set $C' \subseteq C$ of z_c variables.

Pricing Problem (PP): Configuration Generator Problem, also called the pricing problem, which either concludes that, among the non included variables (columns), none of them can contribute to a better LP solution with its addition in the current RMP, or exhibit a new column, called the improving column, which, if added to the current RMP, generates an improved LP value. In practice, there is no need to solve the pricing problem exactly as long as a heuristic solution generates an improving column, i.e., a column with a negative reduced cost [27]. When no more improving columns can be generated with a heuristic, we switch to an exact algorithm for solving the pricing problem in order to double check whether we can still find an improving column, or, otherwise, conclude that we have reached the optimal LP solution of MP.

The overall ILP solution scheme consists of several phases, which we examine in turn:

- Building an initial set of columns in order to initialize the column generation algorithm (Section 2.5.3).
- Solving the linear programming (LP) relaxation optimally $\rightsquigarrow z^{\star}_{\text{LP}}$ for its optimal value, using first a heuristic for solving the PP (with the use of a path formulation with a pre-computed set of paths/routes), and then an exact algorithm with the use of a link formulation (Section 2.5.3).
- Solving the ILP associated with the last restricted master problem $\rightsquigarrow \tilde{z}_{\text{ILP}}$, using a rounding off algorithm (Section 2.5.3). Note that $\tilde{z}_{\text{ILP}}$ defines an ε-optimal ILP value for the MP problem, where ε satisfies:

$$\varepsilon = \frac{\tilde{z}_{\text{ILP}} - z^{\star}_{\text{LP}}}{z^{\star}_{\text{LP}}}.$$

- Applying a post-processing algorithm in order to refine the ILP solution (Section 2.5.3).

The flowchart in Fig. 6 summarizes the solution process.

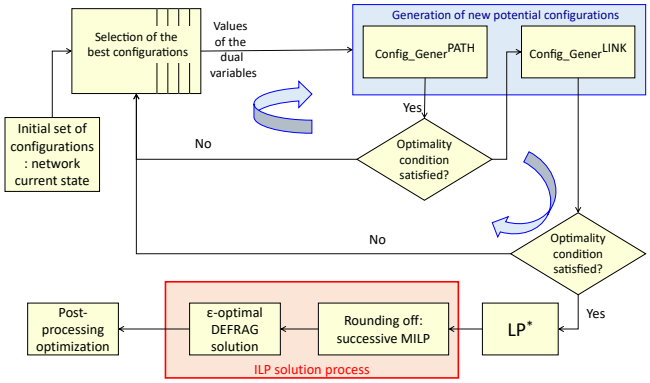


Figure 6: Flowchart of the Solution Process of the DF_ILP Model

Building an initial set of columns

In order to speed-up the solution of the LP relaxation of the DF_ILP, we generated an initial set of configurations. In order to overcome the feasibility issue, we ensure that the initial set of configurations (C^0) contains at least one valid node pair configuration for any pair $(v_s, v_d) \in \mathcal{SD} : K_t^{sd}$, hence satisfying Constraints (2.4). With respect to the transport capacity constraints (Constraints (2.3)), we add a variable β for measuring their violation, and minimize the objective until it reaches a zero value. If the generated set of paths satisfies the transport capacity constraints, we can directly move to the solution of the DF_ILP model with an initial set of columns (Section 2.5.3, otherwise we solve the DF_ILP_INIT model that is defined below. If it happens that we cannot reach a zero value for the objective of the DF_ILP_INIT model, then we conclude that there is no way to grant all the connection requests subject to the current transport capacities. In practice, note that such an initial set of configurations could be derived from the current network state. In this study, we assume that the first initial state of the network is provided, and in the subsequent defragmentation intervals, we use the solution of the last defragmentation.

The modified master problem of the DF_ILP model is called the DF_ILP_INIT model and written as follows, with $C = C^0$ at the outset.

Objective: minimize the bandwidth requirements, and if any, the transport capacity violations.

$$\min \quad \beta. \tag{2.6}$$

Constraints are written as follows.

- Capacity constraints.

$$\sum_{(v_s,v_d)\in\mathcal{SD}} \sum_{k\in K_t^{sd}} \sum_{c\in C_{sd}} D_k \delta_{k\ell}^c \, z_c \leq C_\ell + \beta \qquad \ell \in L \tag{2.7}$$

  They can be equivalently rewritten:

$$\sum_{k\in K_t} \sum_{c\in C} D_k \delta_{k\ell}^c \, z_c \leq C_\ell + \beta \qquad \ell \in L \tag{2.8}$$

- Constraints (2.4)-(2.5)

- Domain of the β variable

$$\beta \geq 0 \tag{2.9}$$

Solution of the DF_ILP_INIT model (2.6)-(2.9) uses the same solution process as the DF_ILP model, except that an optimal solution of the LP relaxation is reached much faster.

Solving the linear programming (LP) relaxation optimally

We use two different formulations for the pricing problem, an exact one with a link model, and a heuristic one, with a path model, which uses a set of pre-computed paths.

Link Formulation:
In this formulation, note that coefficients $\delta_{k\ell}^c$ of the DF_ILP (Master Problem) are determined by the values of the variables y_ℓ^k.

$$y_\ell^k = \begin{cases} 1 & \text{if configuration } c \text{ grants } k \text{ on } \ell \\ 0 & \text{otherwise} \end{cases}$$

We next describe the link and path formulations for building a configuration c. Note that index c is omitted to simplify the notation. Remember that the objective of the

pricing problem in a column generation formulation is defined by the reduced cost of the column generation variables of the master problem (see, e.g., [29] if not familiar with generalized linear programming).

Reduced cost of DF_ILP objective (variable z_c):

$$\min \quad \sum_{k \in K_t^{sd}} \sum_{\ell \in L} D_k y_\ell^k - \sum_{\ell \in L} \sum_{k \in K_t^{sd}} D_k u_\ell^{(2.3)} y_\ell^k - u_{sd}^{(2.4)} \tag{2.10}$$

Single path routing enforcement at the path endpoints of all requests:

$$\sum_{\ell \in \omega^+(s_k)} y_\ell^k = \sum_{\ell \in \omega^-(d_k)} y_\ell^k = 1 \qquad k \in K_t^{sd} \tag{2.11}$$

$$\sum_{\ell \in \omega^-(s_k)} y_\ell^k = \sum_{\ell \in \omega^+(d_k)} y_\ell^k = 0 \qquad k \in K_t^{sd} \tag{2.12}$$

At most one path going through each intermediate node:

$$\sum_{\ell \in \omega^+(v)} y_\ell^k = \sum_{\ell \in \omega^-(v)} y_\ell^k \leq 1 \qquad k \in K_t^{sd}, v \in V \setminus \{s_k, d_k\}. \tag{2.13}$$

Do not exceed transport capacity (or a given threshold as defined by a fraction of the link transport capacity):

$$\sum_{k \in K_t^{sd}} D_k y_\ell^k \leq C_\ell \qquad \ell \in L. \tag{2.14}$$

Such a constraint is not mandatory, however, it helps to prevent violation of the transport capacity constraints by the subset of paths used for a single node pair.

Domains of the variables:

$$y_\ell^k \in \{0, 1\} \qquad k \in K_t^{sd}, \ell \in L. \tag{2.15}$$

Path Formulation:

In this formulation, a set of paths are pre-computed, that is

$$P = \bigcup_{\{v_s, v_d\} \in \mathcal{SD}} P_{sd},$$

where P_{sd} is, e.g., the set of κ-shortest paths. In our experiments, we used P_{sd}^{SHORTEST}, the set of shortest paths using the geographical distances of the links. Then, for each

node pair, the configuration generator will choose one path $p \in P_{sd}$ to provision its demand.

Each path $p \in P$ is characterized by its set of links, using the parameter δ_ℓ^p such that $\delta_\ell^p = 1$ if path p contains link ℓ, 0 otherwise.

The ILP model defining the path formulation of the configuration generator uses a single set of variables: $x_k^p = 1$ if connection k is granted on path p, 0 otherwise.

Note that the path formulation is a compact version of the link formulation, thus the objectives and constraints can be easily derived using the following change of variables: $y_\ell^k = x_k^p \delta_\ell^p$.

Reduced cost of DF_ILP objective:

$$\min \sum_{k \in K_t^{sd}; \ell \in L; p \in P_{sd}} D_k \delta_\ell^p x_k^p (1 - u_\ell^{(2.3)}) - u_{sd}^{(2.4)} \tag{2.16}$$

Single path routing enforcement for all each request:

$$\sum_{p \in P_{sd}} x_k^p = 1 \qquad k \in K_t^{sd} \tag{2.17}$$

Do not exceed transport capacity (or a given threshold as defined by a fraction of the link transport capacity):

$$\sum_{k \in K_t^{sd}} D_k \delta_\ell^p x_k^p \leq C_\ell \qquad \ell \in L. \tag{2.18}$$

Domains of the variables:

$$x_k^p \in \{0, 1\} \qquad k \in K_t^{sd}, p \in P. \tag{2.19}$$

Deriving a First ILP Solution - Rounding Off Algorithm

We use a sequential rounding off algorithm in order to derive an ILP solution for the DF_ILP model, once the linear relaxation has been solved exactly with a column generation algorithm. This way, we get an upper bound ($\tilde{z}_{\text{ILP}}$) on the optimal ILP value ($z_{\text{ILP}}^\star$). Although it takes more computational time than solving the last RMP exactly with an ILP solver (e.g., Gurobi v7.0.1 [30]), it provides a more accurate upper bound, and therefore a more accurate ILP solution for the DF_ILP model.

The sequential rounding off algorithm consists of rounding off a small subset of variables at each step, and then re-optimizing the LP relaxation, hence generating additional configurations. The subset is selected as the set of variables with a fractional

value larger than a given threshold (e.g., .8), we set them to 1 and then re-optimize the resulting LP relaxation with the column generation algorithm. This is also called diving branch and price heuristic [8].

Improving the ILP Solution: A Heuristic

We observed that we could easily improve the ILP solution generated by the rounding off algorithm. The constraints of the configuration generators are such that we route all the connection requests involving the same node pair on the same route. However, sometimes, a fraction of those requests can be routed on shorter routes. Consequently, we use the following greedy heuristic. Consider each connection request in turn, in an arbitrary order, and check whether it can be re-routed on a shorter route. If yes, re-route the connection and iterate until no more re-routing on a shorter route is possible..

2.6 MBB Re-routing Reachability

In this section, we develop optimization models in order to check whether a given optimized provisioning can be reached from the current provisioning with a Make Before Break process. The first model, called MBB_REACH_COMPACT, is a compact one, but it is unfortunately not scalable. We therefore rewrite it using a column generation framework, which gives the MBB_REACH model. We develop two variants of both models. In the first one, the goal is to reach the best (minimum bandwidth requirement) network state assuming only MBB re-routing. In the second one, we discuss how to modify the MBB_REACH_COMPACT/MBB_REACH models if we aim to reach the best state of the network, even if it means allowing interruptions. All models use a t index in order to define the re-routing scheduling, i.e., the order in which the requests need to be re-routed in order to reach a MBB defragmentation, or a defragmentation with the minimum number of disruptions (or minimum disruption duration).

2.6.1 Compact Model MBB_REACH_COMPACT

We now describe the MBB_REACH_COMPACT model. It uses three sets of variables. $C_\ell^t \geq 0$, defines the transport capacity of link ℓ at round t of the re-routing order.

$\pi_k^t \in \{0,1\}$, indicates whether or not a given request k is the t^{th} re-routed one.
$x_k^{\text{MBB}} \in \{0,1\}$, identifies the requests that can be re-routed with a MBB process.

Parameters are defined as follows:
C_ℓ denotes the transport capacity of link ℓ.
$a_{k\ell}^0$ (resp. $a_{k\ell}^{\text{OPT}}$) defines the current (resp. optimized) provisioning/routing of request k: $a_{k\ell}^0 = 1$ (resp. $a_{k\ell}^{\text{OPT}} = 1$) if link ℓ is used in the routing, 0 otherwise.

The objective is to maximize the number of requests that can be MBB re-routed: if its optimal value is equal to the number of variables, then we obtain a MBB defragmentation, i.e., all requests can be seamlessly re-routed.

The objective is written as follows.

$$\min \sum_{\ell \in L} C_\ell^{|T|} \tag{2.20}$$

Set of constraints:

$$C_\ell^t \leq C_\ell \qquad \ell \in L, t \in T \tag{2.21}$$

$$C_\ell^t = C_\ell^{t-1} - \sum_{k \in K} b_k \left(a_{k\ell}^0 - a_{k\ell}^{\text{OPT}}\right) \pi_k^t \qquad \ell \in L, t \in T \tag{2.22}$$

$$x_k^{\text{MBB}} = \sum_{t \in T} \pi_k^t \qquad k \in K \tag{2.23}$$

$$C_\ell^t \geq 0 \qquad \ell \in L, t \in T \tag{2.24}$$

$$x_k^{\text{MBB}} \in \{0,1\} \qquad k \in K \tag{2.25}$$

$$\pi_k^t \in \{0,1\} \qquad t \in T, k \in K \tag{2.26}$$

Constraints (2.21) are the transport capacity: they make sure that they remain satisfied after each re-routing. Constraints (2.22) update the bandwidth usage on each link after each re-routing. Constraints (2.23) define the value of the variable x^{MBB}, i.e., whether request k has been MBB re-routed or not. The last three sets of constraints define the domains of the three sets of variables.

We now discuss the scalability of the MBB_REACH_COMPACT model (2.20) - (2.26). The most numerous variables are the π_k^t. We need $|T| \times |K|$ of them, i.e., $O(|K|^2)$, which makes the MBB_REACH_COMPACT model not very scalable as soon as the number of requests reaches 50. We next discuss a more efficient model, called MBB_REACH, based on a decomposition scheme.

2.6.2 Decomposition Model MBB_REACH

MBB_REACH model relies on the concept of ordering configuration γ clustered with respect to τ, in order to determine the τ^{th} re-routed connection. Let Γ be the set of possible ordering configurations, it is partitioned as follows

$$\Gamma = \bigcup_{\tau \in \{1..|K|\}} \Gamma_\tau,$$

where Γ_τ defines the set of potential configurations defining the τ^{th} re-routed connection. Each set Γ_τ is characterized by a vector π^γ such that:
$\pi^\gamma_k = 1$ if connection k is the τ^{th} re-routed one, 0 otherwise.

Model MBB_REACH introduces a new set of variables, z_γ, which are decision variables: $z_\gamma = 1$ if ordering configuration γ is selected, 0 otherwise. Besides, it reuses the first set of variables from model MBB_REACH_COMPACT, i.e., variables C^t_ℓ, and its objective is expressed identically. The set of variables x^{MBB}_k is not necessary anymore, note that $x^{\text{MBB}}_k = \sum_{\gamma \in \Gamma} \pi^\gamma_k \, z_\gamma$.

The set of constraints is expressed as follows:

$$\sum_{\gamma \in \Gamma_\tau} z_\gamma \leq 1 \qquad \tau \in \{1..|K|\} \tag{2.27}$$

$$C^\tau_\ell \leq C_\ell \qquad \ell \in L, \tau \in \{1..|K|\} \tag{2.28}$$

$$C^\tau_\ell = C^{\tau-1}_\ell - \sum_{k \in K} \sum_{\gamma \in \Gamma_\tau} b_k \pi^\gamma_k \, (a^0_{k\ell} - a^{\text{OPT}}_{k\ell}) z_\gamma \qquad \ell \in L, \tau \in \{1..|K|\} \tag{2.29}$$

$$\sum_{\gamma \in \Gamma} \pi^\gamma_k \, z_\gamma \leq 1 \qquad k \in K \tag{2.30}$$

$$C^\tau_\ell \geq 0 \qquad \ell \in L, \tau \in \{1..|K|\} \tag{2.31}$$

$$z_\gamma \in \{0, 1\} \qquad \gamma \in \Gamma. \tag{2.32}$$

Assuming we solve the linear relaxation of Model MBB_REACH with a column generation technique, then it is much more scalable than model MBB_REACH_COMPACT.

2.6.3 Reach of the Minimum Bandwidth Re-routing

Observe that in the above model (2.20) - (2.26), the underlying assumption is that the requests that cannot be MBB re-routed, are re-routed with disruption after all

the MBB re-routings with the aim of minimizing the duration of the disruptions. However, we could also consider performing the disruptions at the beginning of the defragmentation in order to free more bandwidth, and potentially conduct a larger number of MBB re-routings. In order to consider such a strategy, we need to rewrite constraints (2.21) as follows:

$$C_\ell^t \leq C_\ell + \sum_{k \in K} a_{k\ell}^0 (1 - x_k^{\text{MBB}}) \qquad \ell \in L, t \in T. \tag{2.33}$$

In addition, we must re-establish those *break before make* connections at the end of the process, and ensure the capacity constraints eventually, by introducing one more set of constraints:

$$C_\ell^{|K|+1} = C_\ell^{|K|} + \sum_{k \in K} a_{k\ell}^0 x_k^{\text{MBB}} \leq C_\ell \quad \ell \in L. \tag{2.34}$$

2.6.4 MBB_DF_ILP: Solution Process of Decomposition Model MBB_REACH

We first discuss the solution of the linear relaxation of Model MBB_REACH.

Each pricing problem, denoted by PP_τ, is associated with the selection of the re-routing position of a connection in the overall sequence of re-routings.

Let $u_\tau^{(2.27)} \geq 0$, $u_{\ell\tau}^{(2.29)} \gtreqless 0$ and $u_k^{(2.30)} \gtreqless 0$ be the values of the dual variables associated with constraints (2.27), (2.29) and (2.30), respectively.

The expression of the reduced cost, i.e., the objective of Pricing Problem PP_τ is as follows:

$$\begin{aligned} \min \qquad \overline{c}_{\text{PP}_t} = 0 - u_t^{(2.27)} - \sum_{k \in K} u_k^{(2.30)} \pi_k \\ - \sum_{\ell \in L} \sum_{k \in K} b_k u_{\ell t}^{(2.29)} \pi_k \, (a_{k\ell}^0 - a_{k\ell}^{\text{OPT}}). \end{aligned} \tag{2.35}$$

There is a unique constraint:

$$\sum_{k \in K} \pi_k = 1 \tag{2.36}$$

$$\pi_k \in \{0, 1\} \qquad k \in K. \tag{2.37}$$

Again, we can consider two variants of the MBB_REACH, either the above one, in which we assume the unavoidable disruptions to be made at the beginning of the

defragmentation, or the second one in which the durations of the disruptions are minimized.

In order to improve the ILP solution, we design a simple post-optimizing heuristic. Assume we obtain an ILP solution from the model with a set of columns (note that they are only a subset of the overall set of variables/configurations which have been used in the decomposition solution process), the network is then reconfigured using this solution. With the post-optimizing heuristic, we next go through unreconfigured connections, one by one. For each connection, we attempt to move those connections to their optimized paths. If it is allowed with a make before break mechanism, the heuristic will add one more variable, based on this re-routing, into the ILP model and re-optimize it. The post-optimizing process is repeated until we can generate no new configuration.

2.7 Numerical Results

We implemented the MBB_DF_H heuristic in C++ and ran it on a MacBookPro, Intel Core i7-4870HQ 2.50 GHz 1 processor, 4 cores, Memory 16384 MB 1600 MHz DDR3. For the solution of the DF_ILP and MBB_REACH, we also used a C++ program on a Linux computer with 773727 MB RAM and Intel Xeon E5-2687W v3 @ 3.10 GHz 2 processors, 20 cores, [1]. After describing the data sets, we first estimate the solution accuracy of the MBB_DF_H heuristic and then describe extensive experiments that validate its real time scalability.

2.7.1 Data Sets

We consider three networks: SMALL_NET, NATIONAL and METRO. The NATIONAL and METRO networks are based on real Ciena customer networks. The SMALL_NET network is derived from a Metro network in order to get a small network data instance. Actual NATIONAL and METRO network connections were used to construct traffic intensity matrices that we used in the experiments to dynamically generate realistic random connection states. Connection requests have Poisson arrivals based on the traffic matrix and random durations drawn from a common exponential distribution.

[1] Algorithms could not be run over the same computer for intellectual property reasons. Estimated speedup ratio between the two computers is 3 times.

Each connection has a Weibull distributed bandwidth with a coefficient of variation of 0.3. Time and bandwidth units are not specified and are not relevant as the only goal was to generate a targeted number of network connections for the defragmentation to operate on. Connections were routed on the shortest paths (in hop count) that had sufficient bandwidth. A "load factor" parameter was used to globally vary the connection arrival rates: the corresponding equilibrium connection states represent a range of congestion levels and blocking from light to heavy. Table 2 below displays the key attributes of the three test networks.

The SMALL_NET network has been used for the evaluation of the solution accuracy of the MBB_DF_H heuristic as it involves solving the optimization models DF_ILP and MBB_REACH. The two other networks were used to assess the real-time scalability of the MBB_DF_H heuristic.

	NATIONAL	METRO	SMALL_NET
Nodes	198	37	32
Links	1,018	268	250
Mean Node Indegree/Outdegree	5.14	7.24	7.81
Mean Connection Bandwidth	5	2	10
Mean Connection Duration	5	10	10
Equilibrium Connections (10% Loss target)	6,800 – 7,500	8,900	840

Table 2: Characteristics of the Data Sets

The node and link counts and topologies of the METRO and SMALL_NET networks are nearly identical by construction. The traffic matrix, mean connection bandwidth, and mean connection duration of the SMALL_NET network were selected to generate no more than 1,000 simultaneous connections. This ensured that the algorithms for solving the mathematical programming models that have been proposed could solve the data sets that arose with the available hardware and software. Typical numbers of equilibrium connections for each of the networks at a 10% connection request blocking rate are shown in Table 2. A range is given for the NATIONAL network

Load	average # of overall legacy requests		average # of granted requests, ending before the next nearest defragmentation point	
	HEURISTIC	DF_ILP	HEURISTIC	DF_ILP
0.5	777.2	777.5	305.3	304.7
0.6	894.6	902.9	339.2	341.3
0.7	951.4	952.8	369.3	368.1
0.8	971.1	985.3	378.9	387.9
0.9	993.3	1,006.0	406.9	414.8
1.0	1,015.4	1,020.6	423.6	428.4

Table 3: Comparison of Dynamic Re-routing Performance

connections due to network instabilities near the blocking target of 10%: the number of active connections can vary within this range even within a single simulation run accompanied by a wide variation in blocking, making the 10% blocking target difficult to achieve. If load is increased very slightly, blocking increases to close to 20%; if load is decreased slightly blocking can drop to less than 1%.

For each load factor, we considered 10 defragmentation events. Defragmentation was performed with a period of 1 mean connection duration, ensuring that sufficient connection requests and termination events occurred to produce comparably degraded pre-fragmentation connection states. The network connection states immediately prior to the defragmentation events as generated by the simulation model were also used as the starting states for defragmentation by the algorithms. This permitted a direct comparison of heuristic defragmentation results.

Last, we collect in Table 4 the names of the different models and algorithms we will compare in the sequel.

2.7.2 Comparison of the Dynamic Routing Phases: MBB_DF_H vs. DF_ILP

For each data set, all experimented schemes start with the same initial network configuration. In addition, when a connection request arrives, it will be granted if enough

Name	Model/Algorithm	Introduced in Section
G&RSP_H	Grant - Routing with first Shortest available Path	
G&R_H	Grant - Re-routing Heuristic	Section 2.7.4
DF_ILP	ILP Phase 1: Defragmentation	Section 2.5.2
		Solution in Section 2.5.3
MBB_DF_H	New MBB Defragmentation Heuristic	Section 2.4
MBB_REACH	ILP Phase 2: MBB Defragmentation Check Model	Section 2.5.2
MBB_DF_ILP	Algorithm for solving MBB_REACHmodel	Section 2.6.4
MBB_REACH_COMPACT	ILP Model of Klopfenstein [1]	Section 2.6.1

Table 4: List of Models and Algorithms

spare resources are available, on one of the shortest paths (note that there might exist several shortest paths). However, defragmentation solutions, DF_ILP models and MBB_DF_H heuristic, will produce different new network connection states, leading to significant differences in input data for each model at defragmentation points. The average numbers of requests of defragmentation points processed by the schemes are given in Table 3.

Firstly, we compare the pre- and post-grooming connection states to identify connections which are re-routed. Secondly, those connections are switched by MBB_REACH model in the best possible order in order to avoid any disruption. The average number of re-routed connections associated with the DF_ILP scheme is shown in the last column of Table 3.

2.7.3 Performance of the DF_ILP and MBB_REACH Models and their Solution Scheme

We first evaluate the efficiency of the solution process for the proposed re-routing optimization model and MBB_REACH with the number of generated columns, the accuracy achieved and the computational times. Results are reported in Tables 5 and 6. Each value corresponds to an average over 10 defragmentation points for each data instance.

The DF_ILP model shows very high accuracy (accuracy ε is at most equal to 3.5%). Computational times are within 1 or 2 minutes. While they are not yet real-time scalable, there are not out of reach in future work investigating the use of

Load	# generated columns	Accuracy (%)	Computing Times (Min.)
0.5	301.5	2.4	1.2
0.6	378.1	3.5	1.3
0.7	395.1	2.4	1.2
0.8	398.0	1.4	2.6
0.9	402.9	1.3	1.3
1.0	413.0	1.5	2.4

Table 5: Performance of the DF_ILP Model and Solution Process

heuristics to speed up the solution of the DF_ILP model.

The MBB_REACH model shows satisfactory accuracy (ε is at most equal to 6.3%). Computational times are however not as scalable as those of the DF_ILP model, as they can reach about 1 hour on the data instances with a load close to 1.

The last column of Table 6 shows the average percentage of connections moved successfully by a MBB strategy. In other words, those connections are each reconfigured to a new path without service interruption. We observe that almost all of required-reconfigured connections (at least 94.5%) are able to be satisfied under the MBB condition. The percentage is increasing as the load is increased: this can be explained by an increased spare bandwidth that is well managed by the optimization models, which can easily take full advantage of equivalent shortest paths.

Load	MBB_REACH model			G&RSP_H + DF_ILP
	# generated columns (average)	Accuracy (%)	Computing Times (min.)	MBB re-routing success (% wrt # re-routed requests)
0.5	1,074.9	5.7	1.6	80.6
0.6	1,305.5	6.3	10.2	85.2
0.7	1,429.4	5.0	38.8	86.9
0.8	1,895.9	4.9	63.3	86.5
0.9	1,809.8	4.8	61.0	87.5
1.0	1,555.1	5.1	76.4	90.1

Table 6: Performance of the MBB_REACH Model and Solution Process

Load	G&R_H + MBB_DF_H	G&RSP_H + MBB_DF_ILP	
	# MBB re-routed	# to be re-routed	# MBB re-route
0.5	51.8	78.7	63.1
0.6	144.1	168.2	143.2
0.7	203.4	233.2	202.4
0.8	224.1	261.5	225.9
0.9	248.4	275.6	241.0
1.0	258.0	278.7	250.6

Table 7: Comparison of the Number of Re-routing

	Blocking Probability (%)			Bandwidth Gain (%)			Computational Times (min.)		
	MBB Re-routing		MBBM	MBB Re-routing		MBBM	MBB Re-routing		MBBM
Load	MBB_DF_H	MBB_DF_ILP	DF_ILP	MBB_DF_H	MBB_DF_ILP	DF_ILP	MBB_DF_H	MBB_DF_ILP	DF_ILP
0.5	2.7	2.8	2.7	6.4	[7.4, 7.4 × 1.057]	[7.5, 7.5 × 1.024]	0.00038	2.7	1.1
0.6	7.7	6.8	6.6	15.8	[15.7, 15.7 × 1.063]	[15.7, 15.7 × 1.035]	0.00055	11.6	1.5
0.7	14.3	14.3	13.8	21.9	[22.6, 22.6 × 1.050]	[22.3, 22.3 × 1.024]	0.00100	40.0	1.4
0.8	22.8	21.5	21.6	23.0	[24.8, 24.8 × 1.049]	[25.7, 25.7 × 1.014]	0.00103	65.9	1.2
0.9	27.8	26.8	27.1	24.8	[26.1, 7.4 × 1.048]	[26.8, 26.8 × 1.013]	0.00087	62.3	1.6
1.0	33.0	32.5	32.0	24.6	[26.3, 26.3 × 1.057]	[26.8, 26.8 × 1.015]	0.00103	78.9	1.2

Table 8: Performance Comparison

2.7.4 Performance Evaluation G&R_H heuristic

In the next experiment, we compare the number of re-routings using the MBB_DF_H and then the two phase defragmentation process. Results are reported in Table 7.

The number of re-routings for reaching an MBB defragmentation is fairly similar (see the first and the third columns), while the number of re-routings for reaching the best possible re-routing, i.e., the one with minimum bandwidth requirement, is higher. However, the minimum bandwidth re-routing is usually not an MBB reachable re-routing.

2.7.5 Performance comparison of MBB_DF_H vs. DF_ILP & MBB_REACH

We compare here the performance of the defragmentation solutions output by the MBB_DF_H heuristic, and those derived from the solution of the DF_ILP model. We

use three different performance parameters:

- Bandwidth Gain
- Blocking Probability
- Computational Time.

For the bandwidth gain, we compare the bandwidth requirements before and after the defragmentation event. Note the the set of connection requests is not exactly the same for the MBB_DF_H heuristic and the DF_ILP solutions. Results are summarized in the first three columns of Table 8. Therein, we see that the bandwidth gain output by the solution of the DF_ILP model is slightly higher than the one of the MBB_DF_H heuristic, independently of the network load.

We need to make sure that the bandwidth gains are not met at the expense of the Grade of Service (GoS). For that reason, we propose to use a Blocking Probability (BP) estimator. It is computed as follows:

$$\mathrm{BP}_{\mathrm{DEFRAG\_\ INTERVAL}} = \frac{\#\text{ Denied requests}}{\text{Total }\#\text{ incoming requests}}, \tag{2.38}$$

where the total number of incoming requests includes those requests granted and then terminating inside the same defragmentation interval, see Fig. 3. The blocking probability that is next reported in Table 8 corresponds to the mean BP over all the defragmentation intervals, i.e.,

$$\mathrm{BP} = \frac{\sum_{\mathrm{DEFRAG\_INTERVAL}} \mathrm{BP}_{\mathrm{DEFRAG\_INTERVAL}}}{\#\text{ defrag. intervals}}. \tag{2.39}$$

We observe that the blocking probabilities of the DF_ILP model are all smaller than those of the MBB_DF_H heuristic, while the bandwidth gains are higher. Therefore, we can conclude that the better performance of the DF_ILP model is not due to a smaller number of processed requests during defragmentation events: on overage, the GoS is higher, and the bandwidth gains are also higher.

2.7.6 Real-Time Scalability of the MBB_DF_H Heuristic

The previous results have shown that the MBB_DF_H heuristic is able to practically reduce the bandwidth requirement close to the one of an ideal provisioning. It is

of great interest for network operators that it remains time scalable for larger data sets. Consequently, the MBB_DF_H heuristic was also applied on the METRO and the NATIONAL Networks. Results are reported in Table 9.

Experiments were conducted with quite high blocking probabilities, having in mind that, even though high blocking is not a desirable operating condition it is possible for it to arise in practice: for example, Table 2 shows a range of connections generated by simulation runs targeting a 10% blocking probability. A closer examination of a typical run shows that the simulated network exhibited unstable behavior, alternating between high blocking and low blocking states with a stable 10% target not being reachable in equilibrium. These types of network instabilities have been observed before (see Akinpelu [31]). Defragmentation is one means to limit the development of these instabilities and maintain low blocking with a relatively larger number of supported connections.

On those data instances, the MBB_DF_H heuristic consumes at most 2.66 seconds for the heaviest traffic load of the NATIONAL network. Additionally, it reduces the blocking probability significantly, about 3 times.

2.8 Conclusions

We fulfill the first objective of the study, which was to propose a defragmentation heuristic (MBB_DF_H), which is highly scalable. While many heuristics are often proposed, authors rarely worry about their time scalability, a stringent requirement for network operations. The second objective of the study was to evaluate the accuracy of the solutions output by the defragmention heuristic. We develop optimization ILP models, based on a decomposition scheme. We then observed that on small to medium data sets, the MBB_DF_H heuristic outputs were near optimal solutions. Enhancements of the ILP models are however required in order to validate that the accuracy remains very good for larger data instances.

	Before DF	After DF			CPU (ms)	
Load	Blocking rate	# Total connections	# Moved connections	Blocking rate	Routing + DF	Routing
Metro Network						
0.95	0.1584	10,054.22	551.45	0.007139	899.0	220.1
1.0	0.2124	10,292.59	1,020.97	0.028646	1,096.6	248.8
1.05	0.2604	10,283.01	1,316.38	0.066583	1,102.0	231.1
1.1	0.3003	10,253.63	1,530.51	0.100938	1,137.0	231.0
1.15	0.3326	10,177.08	1,652.82	0.136796	1,122.2	217.4
1.2	0.3604	10,110.58	1,741.08	0.171681	1,141.9	214.6
National Network						
0.80	0.0003	7,551.52	175.81	0.000188	436.5	236.1
0.85	0.0007	8,027.77	260.77	0.000404	654.7	350.4
0.9	0.1980	8,483.71	398.89	0.001127	962.9	511.5
0.95	0.2404	8,924.47	708.33	0.004102	1,544.36	811.3
1.00	0.2693	9,146.89	1,240.88	0.023884	2,514.55	1,253.6
1.05	0.2983	9.194.16	1,487.84	0.056600	2,336.24	1,149.3
1.10	0.3264	9,213.87	1,706.13	0.087514	2,332.08	1,114.8
1.15	0.3499	9,237.62	1,813.70	0.116580	2,637.22	1,240.4
1.2	0.3704	9,253.68	1,898.48	0.145731	2,656.14	1,236.0

Table 9: Computational Experiments on Large Data Sets

Chapter 3

Efficient make before break defragmentation

Optical multilayer optimization periodically reorganizes layer 0-1-2 network elements to handle both existing and dynamic traffic requirements in the most efficient manner. This delays the need for adding new resources in order to cope with the evolution of the traffic, thus saving CAPEX.

The focus of this paper is on Layer 2, i.e., on capacity reoptimization at the optical transport network (OTN) layer when routes (e.g., LSPs in MPLS networks) are making unnecessarily long detours to evade congestion. Reconfiguration into optimized routes can be achieved by re-defining the routes, one at a time, so that they use the vacant resources generated by the disappearance of services using part of a path that transits the congested section.

To maintain the Quality of Service, it is desirable to operate under a Make-Before-Break (MBB) paradigm, with the minimum number of reroutings. The challenge is to determine the best rerouting order while minimizing the bandwidth requirement. We propose an exact and scalable optimization model for computing a minimum bandwidth rerouting scheme subject to MBB in the OTN layer of an optical network. Numerical results show that we can successfully apply it on networks with up to 30 nodes, a very significant improvement with respect to the state of the art. We also provide some defragmentation analysis in terms of the bandwidth requirement vs. the number of reroutings.

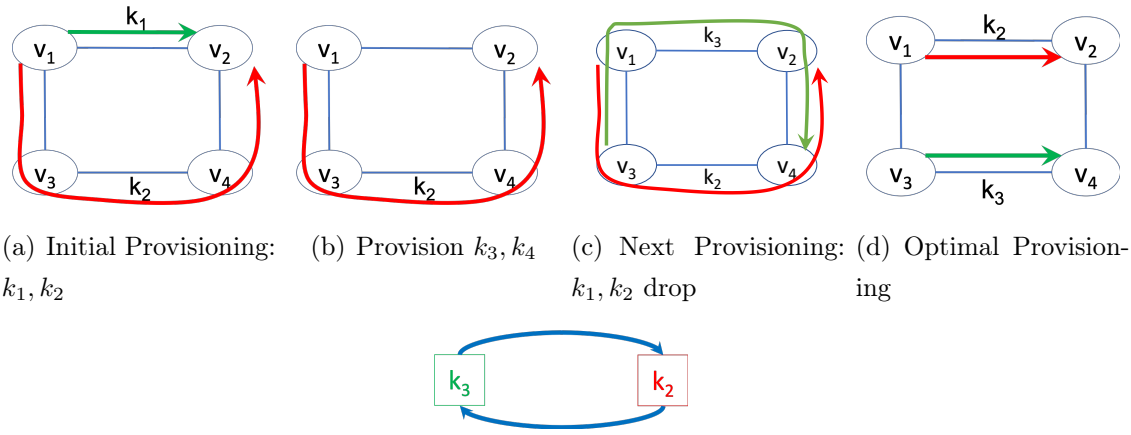
This paper is under revision as: H. Duong, B. Jaumard, R. Armolavicius and D.

Coudert, "Efficient make before break defragmentation," *IEEE/ACM Transactions on Networking*. Under review (last revision)

3.1 Introduction

Network reconfiguration is required in order to adapt to traffic changes, network failures, or new deployment of network resources. It occurs at the optical layer in order to make sure that the upper layer traffic, e.g., IP layer traffic, can be efficiently carried. In such a case, we deal with lightpath reconfigurations and the primary objective is to reduce disruptions to user traffic carried by existing lightpaths, measured by the number of disrupted lightpaths or the duration of lightpath disruptions [32]. Network reconfiguration may also appear in the logical layer, in order to attain a better resource utilization [24]. In heavily loaded networks, dynamic connection request addition and drop actions may result in a set of connection requests where some paths are not the shortest possible ones, leading to poor resource utilization compared to an optimal or at least an optimized state. Thus, global connections rerouting is proposed at certain time intervals (e.g., daily, weekly) to improve the network performance.

Researchers have investigated this connections rerouting along two directions. The first one consists in computing an optimized provisioning of the connection requests with respect to resource utilization, and then finding a sequence of connection rerouting operations in order to migrate from the current network provisioning to the optimized one with the minimum number of disruptions [33, 34, 35, 36, 14, 37]. These studies usually have the constraint that a connection request can be rerouted at most once (i.e., from legacy to optimized route). The existence of a strategy using only make-before-break (MBB) is not always possible due to the presence of dependency cycles. Consider the example in Figure 7 in which the state of Figure 7(c) results from add and drop requests of states represented in Figures 7(a) and 7(b). In order to reach the optimal provisioning of Figure 7(d), connection request k_2 needs to be rerouted before k_3 because a link of the new route of k_3 belongs to the current route of k_2, and for similar reasons, k_3 needs to be rerouted before k_2, as illustrated in Figure 7. In order to find rerouting strategies, authors have proposed to use the break-before-make (BBM) paradigm that allows for the temporary interruption of connection requests, and so for breaking dependency cycles [33, 34, 36, 14].



(a) Initial Provisioning: k_1, k_2

(b) Provision k_3, k_4

(c) Next Provisioning: k_1, k_2 drop

(d) Optimal Provisioning

(e) Dependency Graph

Figure 7: Optimal Provisioning is not MBB Reachable if all links and connections are with unit capacities and requirements, respectively

The idea of the second direction is to compute the best provisioning that is reachable from the legacy provisioning by a sequence of connection reroutings with no disruption, i.e., under the so-called MBB paradigm. While many studies have investigated the first direction, this second direction has received very little attention [1]. In this paper, we propose a scalable optimization model, called DEFRAG_SIM, for this NP-Complete optimization problem. Numerical results show that we improve very significantly against the state-of-the-art [1], enabling to solve instances on networks with up to 30 nodes.

The paper is organized as follows. We briefly review in Section 3.2 the papers related to reoptimization in the OTN layer, as well as the model of Klopfenstein [1], which is the only previously proposed optimization model for rerouting subject to MBB. We next describe in Section 3.3 our proposed decomposition model, called DEFRAG_SIM, which requires in practice a much smaller number of variables and constraints than the model of Klopfenstein [1]. In Section 3.4, we explain how to solve efficiently the proposed DEFRAG_SIM model with the DEFRAG_MBB algorithm, which contains a polynomial time algorithm for the generation of the rerouting configurations. In Section 3.6, we show how to use parallel reroutings for reducing the number of rerouting events or, in other words, the duration of each reoptimization event as output by DEFRAG_SIM (see Section 3.2.2 for a concise definition of rerouting/ reoptimization events). Numerical results are presented in Section 3.7.

Conclusions are drawn in the last section.

3.2 State of the Art

Note that we focus on Layer 2, while being aware that a lot of work has been recently made on Layer 0 in the context of flexible optical networks. But capacity reoptimization differs from spectrum defragmentation as there is no need to take care of continuity or contiguity constraints, and therefore we omit references related to Layer 0.

3.2.1 Literature Review

Several studies have been devoted to network reconfiguration with the minimum number of disruptions, following the strategy of migrating from a legacy ineffective provisioning to a given pre-computed optimized/optimal one. As a result, it usually prevents the existence of a strategy using only MBB due to the presence of dependency cycles as explained in the introduction. In order to find a rerouting strategy, authors have then proposed to use the Break-Before-Make (BBM) paradigm sparingly to allow temporary interruption of connection requests, and so to break dependency cycles. For instance, Jose and Somani [35] propose heuristics for minimizing the total number of BBMs used in the rerouting strategy, and Coudert *et al.* [33, 38] and Solano and Pióro [14] provide scalable exact algorithms to minimize the concurrent number of BBMs. Tradeoffs between these two conflicting objectives are investigated by Cohen *et al.* [34] and Solano [36].

To further reduce the total or concurrent number of BBMs, Kadohata *et al.* [37] propose to use spare wavelengths to reroute a connection request to a temporary route rather than using a BBM. For example, assume that the current connection k needs to be rerouted from path p to path p', but such a rerouting cannot be MBB due to resource dependence. Then one unavoidable BBM reroute is performed. However, using an intermediate reroute, it may be possible to reroute k under the MBB paradigm. For instance, assume that there exists a path p'' such that the reroutings from p to p'' and from p'' to p' satisfy the MBB condition. In other words, one BBM can be avoided at the expense of performing two MBBs.

The network reconfiguration problem has also been investigated in logical layers,

and in particular for MPLS [24, 39, 40] and SDN [41] networks, with the same constraints and objectives as above. Surprisingly, it has been found that deciding if the problem can be solved using MBB only can be done in polynomial time for Layer 0 [42], but this decision problem is NP-Complete for logical layers, i.e., for Layer 2 [40].

On the other hand, while many studies have investigated rerouting strategies both at the optical and the logical layers, very few studies have looked at rerouting subject to the MBB paradigm (i.e., joint computation of optimal provisioning and rerouting strategy subject to MBB, or, in other words, we perform a sequence of MBB reroutings to achieve an optimal provisioning in terms of minimum bandwidth requirement). Klopfenstein's study [1] is the only one proposing an optimization model, but unfortunately it is not scalable. Still, we recall it in Section 3.2.3 as an introduction to our decomposition model in Section 3.3.

3.2.2 Notations

We consider a network represented by a directed multi-graph $G = (V, L)$, where V is the set of nodes (indexed by v) and L is the set of links (indexed by ℓ). Different links may exist between two nodes in order to model different logical links, with e.g., different types of traffic. We denote $\omega^-(v)$ (resp. $\omega^+(v)$) the set of incident links incoming to (resp. outgoing from) node $v \in V$. Let C_ℓ denote the transport capacity of link ℓ.

Let K be the set of connection requests (indexed by k). Connection request $k \in K$ is characterized by its source s_k, its destination d_k, and its bandwidth requirement b_k.

In what follows, we call rerouting operation the action of rerouting a connection request $k \in K$, and rerouting event the action of either performing a single rerouting operation, or a set of parallel rerouting operations (see Section 3.6 for the details on the conditions under which we conduct parallel rerouting). A reoptimization event is an ordered sequence of rerouting events, and so of rerouting operations. Let T, indexed by t, be the set of rerouting events of a reoptimization event. Hence, t designates one rerouting event. Observe that under parallel rerouting, t designates the set of rerouting operations performed in parallel (again see Section 3.6 for the details).

3.2.3 Klopfenstein's Model (2008)

The model of Klopfenstein [1] consists in finding the best possible rerouting strategy, while guaranteeing it can be reached within a Make-Before-Break (MBB) policy.

Before developing further, we define the set of variables.

- $x_{k\ell}^t = 1$ if connection request $k \in K$ uses link $\ell \in L$ in its routing at step $t \in T$, 0 otherwise.

- $\pi_k^t = 1$ if connection request k is rerouted at step t, 0 otherwise.

Indeed, Klopfenstein [1] proposed a very general network resource utilization function subject to a parameter α and that can be written as follows:

$$\text{OBJ}_\alpha = \frac{1}{1-\alpha} \sum_{\ell \in L} \left(C_\ell - \underbrace{\sum_{k \in K} b_k x_{k\ell}^{|T|}}_{\text{link load}} \right)^{1-\alpha}, \tag{3.1}$$

In the sequel, we will adopt OBJ_0. The objective is then to maximize the overall spare capacity:

$$\max \qquad \left(\text{OBJ}_0 = \sum_{\ell \in L} C_\ell - \sum_{k \in K} b_k \left(\sum_{\ell \in L} x_{k\ell}^{|T|} \right) \right). \tag{3.2}$$

These objectives and variables are decided by the set of below constraints:

$$\sum_{\ell \in \omega^-(v)} x_{k\ell}^t - \sum_{\ell \in \omega^+(v)} x_{k\ell}^t = \begin{cases} -1 & \text{if } v = s_k \\ 1 & \text{if } v = d_k \\ 0 & \text{otherwise} \end{cases}$$

$$k \in K, v \in V, t \in T \tag{3.3}$$

$$\sum_{k \in K} b_k x_{k\ell}^t \leq C_\ell \qquad \ell \in L, t \in T \tag{3.4}$$

$$\sum_{k \in K} \pi_k^t \leq 1 \qquad t \in T \tag{3.5}$$

$$x_{k\ell}^t - x_{k\ell}^{t-1} \leq \pi_k^t \qquad k \in K, \ell \in L, t \in T \tag{3.6}$$

$$x_{k\ell}^t \in \{0,1\} \qquad k \in K, \ell \in L, t \in T \tag{3.7}$$

$$\pi_k^t \in \{0,1\} \qquad k \in K, t \in T. \tag{3.8}$$

Constraints (3.3) are flow constraints and are used in order to establish a route for each connection request, at each rerouting event. Due to the subsequent constraints, the set of paths at rerouting event t will differ by at most one path from the set of paths at rerouting event $t-1$. Constraints (3.4) enforce the transport capacity constraints. Constraints (3.5) impose that at most one connection request is rerouted per rerouting event. Constraints (3.6) are used to detect rerouted connection requests. More precisely, if the route of connection request k at rerouting event t uses a link ℓ that was not previously used to route that connection request (i.e., at time $t-1$), we have $x^t_{k\ell} = 1$, $x^{t-1}_{k\ell} = 0$, and so $\pi^t_k = 1$, which indicates that connection request k has been rerouted at rerouting event t. Now, if the route of connection request k is not changed, we have $x^t_{k\ell} = x^{t-1}_{k\ell}$ and $\pi^t_k \in \{0, 1\}$, in which case the value of π^t_k is forced to 0 by Constraints (3.5) if another request k' is rerouted at rerouting event t. Observe that the case $\pi^t_k = 0$, $x^t_{k\ell} = 0$ and $x^{t-1}_{k\ell} = 1$ cannot happen. Indeed, it would imply that the routing of connection request k at rerouting event $t-1$ is not simple (i.e., is not loop-free), which is prevented by the objective. Hence, when $\pi^t_k = 0$, we have $x^t_{k\ell} = x^{t-1}_{k\ell}$ and so the routing of connection request k is unchanged. The last two sets of constraints define the domain of the variables. Note that if the rerouted path has a link in common with the original one, there is no need to double the capacity reservation corresponding to the considered connection request [1].

The largest data instance on which experiments were conducted in [1] was on a network with 10 nodes and about 40 links with capacity C_ℓ. The author was not able to obtain optimal solutions for more than 5 rerouting operations within the time limit imposed (30 minutes).

3.3 A Decomposition Model: DEFRAG_SIM

In this section, we propose a decomposition model, called DEFRAG_SIM, based on a set of rerouting operations, where each rerouting operation proposes a potential MBB rerouting of a single connection request, i.e., a connection request for which there exists an alternate route with enough spare bandwidth for its routing. The model is parameterized by $|T|$, a bound on the number of rerouting operations. A solution of DEFRAG_SIM is an ordered sequence of at most $|T|$ rerouting operations leading to

the best provisioning reachable from the legacy provisioning. Observe that less than $|T|$ rerouting operations might be sufficient to reach that optimized provisioning. No external connection setup or release requests are granted between these rerouting operations. The objective is to minimize the bandwidth requirements of the best reachable MBB provisioning within at most T reroutings. Observe that there is no guarantee to reach the best provisioning with a monotonous sequence, i.e., such that the overall bandwidth requirement decreases after each single rerouting operation (of a connection request). It may happen that the overall bandwidth requirement increases after a given rerouting operation before decreasing again in order to reach the best reachable MBB provisioning.

Let P be the overall set of potential rerouting operations, with $P = \bigcup_{k \in K} \bigcup_{t \in T} P_k^t$, where P_k^t is the set of possible routes for connection request $k \in K$ at rerouting event $t \in T$.

The integer linear programming (ILP) formulation of DEFRAG_SIM uses the following binary variables:

- $z_{kp}^t = 1$ if route $p \in P_k^t$ is selected at rerouting event $t \in T$ for the rerouting of $k \in K$, 0 otherwise.
- $C_\ell^t =$ required bandwidth on link $\ell \in L$ at rerouting event $t \in T$.

It also uses the following parameters:

- $a_{k\ell}^0 = 1$ if link $\ell \in L$ is used in the initial routing of connection request $k \in K$, 0 otherwise.
- $C_\ell^0 = \sum_{k \in K} b_k a_{k\ell}^0 =$ initial bandwidth usage on link $\ell \in L$.
- $\delta_\ell^p = 1$ if path $p \in P$ uses link $\ell \in L$, 0 otherwise.

The objective of DEFRAG_SIM is to maximize the spare capacity

$$\max \sum_{\ell \in L} \left(C_\ell - C_\ell^{|T|} \right). \tag{3.9}$$

As $\sum_{\ell \in L} C_\ell$ is a constant value, the objective can be re-expressed as minimizing the bandwidth usage:

$$[\text{DEFRAG\_SIM}] \quad \min \sum_{\ell \in L} C_\ell^{|T|} \qquad \text{subject to} \quad (3.11) - (3.16).$$

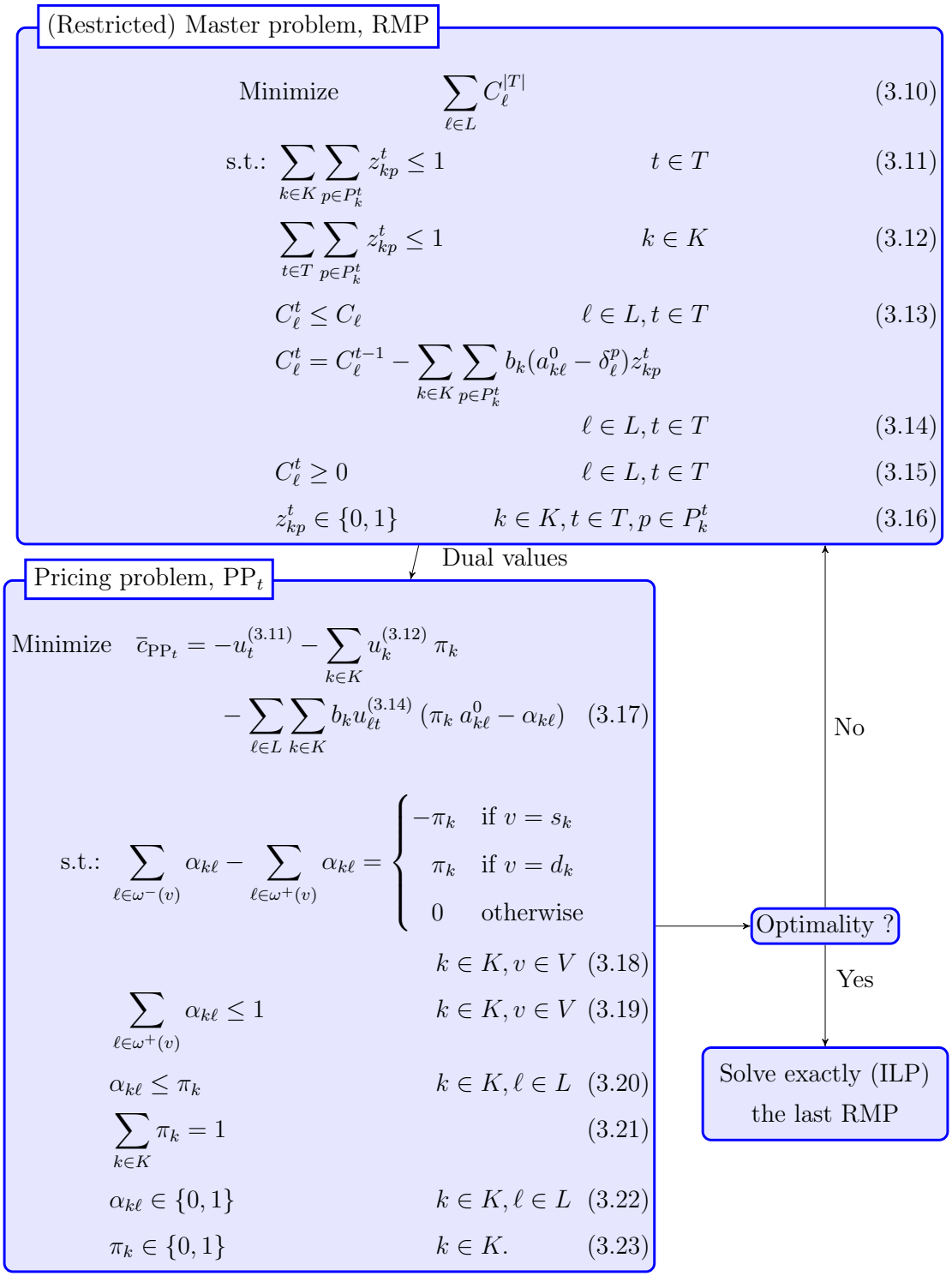


Figure 8: Flow chart of decomposition model DEFRAG_SIM

Constraints (3.11) in Figure 8 prevent the selection of more than one rerouting operation at each rerouting event. Note that $|T| \leq |K|$ is an upper bound on the number of rerouting operations as we cannot predict a priori the number of required MBB reroutings. Constraints (3.12) ensure that a connection request is rerouted at most once. Constraints (3.13) make sure that transport capacities are never exceeded at any rerouting event. Constraints (3.14) update the bandwidth usage on link ℓ at rerouting event t, taking into account the unique connection request that has been rerouted at t. Constraints (3.15)-(3.16) define the domain of the variables.

Since less that $|T|$ rerouting operations might be necessary to reach the best possible provisioning reachable from the legacy provisioning after at most $|T|$ rerouting operations, it might be desirable to ensure that rerouting operations are performed with consecutive indexes. This can be done adding Constraints (3.24) to the model (3.10)-(3.16). Indeed, Constraints (3.24) prevent performing a rerouting operation at rerouting event $t+1$ if no rerouting operation is performed at rerouting event t.

$$\sum_{k \in K} \sum_{p \in P_k^{t+1}} z_{kp}^{t+1} \leq \sum_{k \in K} \sum_{p \in P_k^t} z_{kp}^t \qquad t \in T. \tag{3.24}$$

3.4 Solution Process

3.4.1 Generic Process

The model (3.10)-(3.16) has an exponential number of variables, and therefore column generation [29] is required in order to efficiently solve its linear relaxation.

This technique consists of decomposing the original problem into a Restricted Master Problem (RMP), i.e., model (3.10)-(3.16) with a very restricted number of variables, and one or several pricing problems (PPs). In the particular case of model (3.10)-(3.16), we will show in the next section that the pricing problem can be decomposed into $|K| \times |T|$ independent smaller pricing problems, each denoted by PP_t^k. The RMP and the PP(s) are solved alternately. Solving the RMP consists in selecting the best connection reroutings, while solving the PPs allows for the generation of new columns, i.e., potential connection routes, and more precisely routes such that, if added to the current RMP, improve the optimal value of its linear relaxation. The process continues until the optimality condition is satisfied, that is, all the so-called

reduced costs that define the objective function of the pricing problems are positive (see [29] if not familiar with linear programming concepts). An ε-optimal solution is derived by solving exactly the ILP model associated with the last RMP, with ε defined as follows:

$$\varepsilon = \left(\tilde{z}_{\text{ILP}} - z^{\star}_{\text{LP}}\right) / z^{\star}_{\text{LP}}, \tag{3.25}$$

where $z^{\star}_{\text{LP}}$ and $\tilde{z}_{\text{ILP}}$ denote the optimal LP value and the optimal ILP value of the last RMP, respectively. The solution process is illustrated in the flowchart of Figure 9.

3.4.2 DEFRAG_SIM Algorithm

Pricing Problem PP_t

Let $u_t^{(3.11)} \leq 0$, $u_k^{(3.12)} \leq 0$ and $u_{\ell t}^{(3.14)} \gtreqless 0$ be the values of the dual variables associated with constraints (3.11), (3.12) and (3.14), respectively.

We use the following binary variables:

- $\pi_k = 1$ if $k \in K$ is selected for rerouting, 0 otherwise.

- $\alpha_{k\ell} = 1$ if $\pi_k = 1$ and the route of $k \in K$ uses link $\ell \in L$, 0 otherwise.

The goal of PP_t (Figure 8) is to select a unique request for potential rerouting, the one with a new route of minimum cost (Objective (3.17)). Constraints (3.18) take care of identifying the best possible route, using flow constraints, for the request that is rerouted, i.e., the unique request k such that $\pi_k = 1$. Constraints (3.19) make sure that we only output simple paths, with no loops. Constraints (3.20) make sure that routing variables $\alpha_{k\ell}$ are null if request k is not selected for rerouting during rerouting event t, i.e., the rerouting event associated with the PP_t pricing problem. Constraints (3.21) ensure that each PP_t selects exactly one connection for potential rerouting at rerouting event t. Constraints (3.22)-(3.23) define the domain of the variables.

Observe that we can decompose each PP_t into $|K|$ elementary pricing problems PP_t^k, each examining the option of rerouting request $k \in K$ at rerouting event t by setting $\pi_k = 1$ in PP_t. Then, the solution of PP_t is given by

$$\overline{c}_{\text{PP}_t} = \min_{k \in K} \{\overline{c}^k_{\text{PP}_t} : k \text{ is rerouted at rerouting event } t\}, \tag{3.26}$$

where $\overline{c}^k_{\text{PP}_t}$ denotes the reduced cost of PP_t^k, that we next describe.

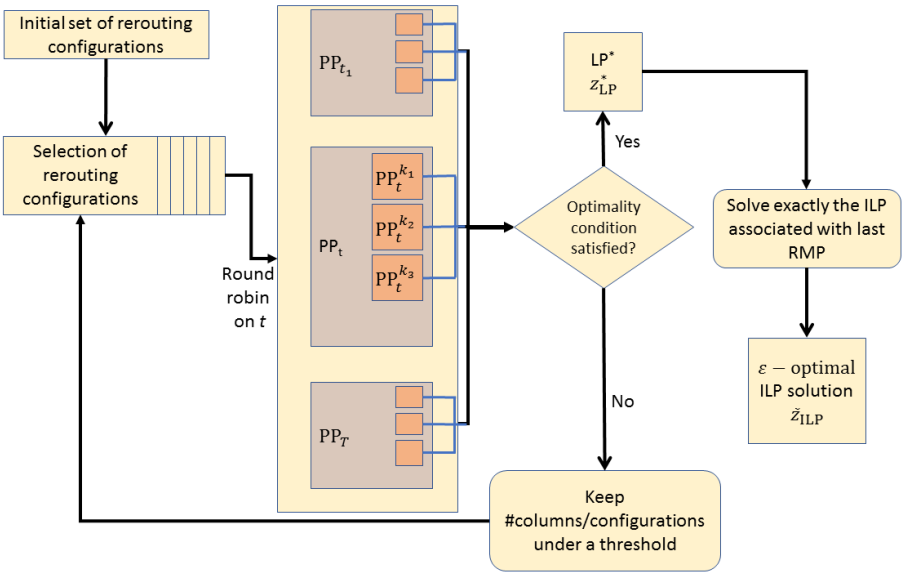


Figure 9: Flowchart of the Proposed Solution Scheme

Elementary Pricing Problem PP_t^k

In each elementary pricing problem PP_t^k, we examine the option of rerouting request k at rerouting event t by setting $\pi_k = 1$ in PP_t. Thus, the ILP formulation of PP_t^k is as follows.

$$\min \bar{c}^k_{\mathrm{PP}_t} = -u_t^{(3.11)} - u_k^{(3.12)} - \sum_{\ell \in L} b_k u_{\ell t}^{(3.14)} (a^0_{kl} - \alpha_{k\ell}) \tag{3.27}$$

$$\sum_{\ell \in \omega^-(v)} \alpha_{k\ell} - \sum_{\ell \in \omega^+(v)} \alpha_{k\ell} = \begin{cases} -1 & \text{if } v = s_k \\ 1 & \text{if } v = d_k \\ 0 & \text{otherwise} \end{cases}, \quad v \in V \tag{3.28}$$

$$\sum_{\ell \in \omega^+(v)} \alpha_{k\ell} \leq 1 \qquad v \in V \tag{3.29}$$

$$\alpha_{k\ell} \in \{0, 1\} \qquad k \in K, \ell \in L. \tag{3.30}$$

Now, observe that PP_t^k is a weighted shortest simple path problem in a graph with possibly negative weight cycles. This problem is NP-hard by a reduction from the longest simple path problem (see [43, Section 24.1] or [44, Chapter 5]), and so cannot be solved using only the Bellman-Ford-Moore (BFM) algorithm, even if it takes care of the negative weights. However, we can still use the BFM algorithm,

but with some additional tool. As we cannot enforce the simple path condition, the BFM algorithm may fail to output a path with a negative reduced cost, due to the discovery of a negative cycle. In such a case, we then recourse to the solution of the ILP formulation of PP_t^k ((3.27)-(3.30)), which includes constraints to enforce the simple path condition.

It is worth noting that calls to the BFM algorithm can be grouped by sources. So $|V|$ calls to BFM suffice to solve PP_t.

Theorem 1. *All pricing problems can be investigated with at most $O(|V|)$ runs of the BFM algorithm, leading to an $O\left(|L| \times (|K| + |V|^2) \times |T|\right)$ time complexity.*

Proof. Note that the reduced cost in (3.27) can be rewritten:

$$\overline{c}^k_{\mathrm{PP}_t} = \underbrace{-u_t^{(3.11)} - u_k^{(3.12)} - \sum_{\ell \in L} b_k u_{\ell t}^{(3.14)} a^0_{kl}}_{\text{constant}} + \sum_{\ell \in L} b_k u_{\ell t}^{(3.14)} \alpha_{k\ell}. \tag{3.31}$$

This entails that the solution of PP_t^k can be reduced to the solution of:

$$[\mathrm{PP}^{\text{generic}}_{tk}] \qquad \min \sum_{\ell \in L} u_{\ell t}^{(3.14)} \varphi_\ell \quad \text{subject to:} \quad (3.28) - (3.30).$$

Observe that problem $\mathrm{PP}^{\text{generic}}_{tk}$ is equivalent to a shortest simple path problem with negative weights, without any guarantee that it contains no negative cycles. Taking into account that *(i)* the coefficients of the objective function are independent of k, and *(ii)* the BFM algorithm can be easily modified in order to output a shortest path tree from a given source node, see, e.g., [43], (i.e., it computes all the (weighted) shortest paths from a given source node), we can then use $|V|$ calls of the BFM algorithm, one from each possible source node, for a given t. Then, for each connection k, we can compute $\overline{c}^k_{\mathrm{PP}_t}$ using (3.31) and check whether the reduced cost is negative, and, if so, generate a new potential rerouting. For a given t, computing $\overline{c}^k_{\mathrm{PP}_t}$ for all k can be done in $O(|K| \times |L|)$ time, once all $|V|$ BFM calls have been made, and each call to BFM requires time $O(|V| \times |L|)$, hence the overall complexity

$$O\left(|L| \times (|K| + |V|^2) \times |T|\right).$$

□

Solution Process

In order to be done efficiently, the resolution of the $|K|$ Elementary Pricing Problems (PP_t^k) for a given t require their grouping as seen in the previous paragraph. In the context of a column generation model with a large set of different pricing problems, the efficiency of the solution depends on the best combination of linear program re-optimization (i.e., solution of the current Restricted Master Problem), and the solution of the whole set or a subset of pricing problems. We consider the following three options.

1. Re-optimize the current RMP after solving all the elementary PP_t^k associated with a given t, and perform a round-robin on t. Re-optimization of the RMP (Restricted Master Problem) is performed with all the rerouting configurations with a negative reduced cost for a given t.

2. Solve all PP_t with the same set of dual values, and add all the new improving columns simultaneously. Again, either solve exactly each PP_t or stop their solution as soon as one PP_t^k has a negative reduced cost.

3. Solve all the elementary PP_t^k associated with a given t, and add to the RMP the rerouting associated with the smallest reduced cost. More specifically, with a given t, we perform a round-robin on k, then selected best reduced cost. This strategy focuses on k, instead of on t as in the two previous options.

Based on a careful analysis of the pros. and cons., we went on with the third option. Details are available in Algorithm 2. Therein, function CostBFM(k, t, RMP), using the Bellman–Ford–Moore algorithm, computes the weighted shortest path for connection k with weights defined by the dual values associated with the optimal solution of the current RMP. If we cannot find a shortest path due to a negative cycle (leveraging the CostBFM function), k is included into the set K^{NEG}. When there is no positive reduced cost found by the BFM algorithm and the set K^{NEG} is not empty, we try the exact ILP model (CostILP) for the connections in K^{NEG}, Lines 9-12.

Generation of an Initial Set of Columns

We use two complementary strategies to generate the initial set of columns. The first strategy is to identify the subset $K_I \subseteq K$ of connection requests that can be rerouted on a shortest path. So, none of the connection requests $k \in K_I$ is routed

Algorithm 2 Solution Process

Require: RMP_{LP}, Current Linear Relaxation of Restricted Master Problem (RMP)

repeat
  for $t \in T$ **do**
    $k' \leftarrow \arg\min_{k \in K}(\text{CostBFM}(k, t, \text{RMP}_{\text{LP}}))$
    $k' \leftarrow \arg\max_{k \in K}(\text{CostBFM}(k, t, \text{RMP}_{\text{LP}}))$
    if reduced cost of $k' < 0$ **then**
      add a new column to RMP_{LP}
      optimize RMP_{LP}
    else
      $k' \leftarrow \arg\min_{k \in K^{\text{NEG}}}(\text{CostILP}(k, t, \text{RMP}_{\text{LP}}))$
      if reduced cost of $k' < 0$ **then**
        add a new column to RMP_{LP}
        optimize RMP_{LP}
until no new column is found

on a shortest path in the legacy routing. Then, we arbitrarily select at most $|T|$ of these connection requests, and all of them if $|K_I| \leq |T|$, to form the initial set of columns. The first strategy is described in Algorithm 3. Therein, function Shortest_Avail_Path(G, k) finds the shortest path with current spare resources (it may be different from the shortest path without capacity constraints). Note that if the shortest path is sharing links with the current path, the resources on those links are not required twice. The second strategy is used when $|T|$ is large. Let T_1, T_2 and T_3 be pairwise disjoint sets of rerouting events such that $T = T_1 \cup T_2 \cup T_3$ and $|T| = |T_1|+|T_2|+|T_3|$. We use the first strategy to find a subset K_{I_1} of initial columns, with $K_{I_1} \leq |T_1|$. We then use K_{I_1} as initial columns for running the defrag_mbb algorithm with a bound $|T_1|$ on the number of rerouting events. We next obtain as an output a subset $K_1 \subseteq K$ of connection requests. We go on using the first strategy to find a subset K_{I_2} of initial columns, with $K_{I_2} \leq |T_2|$ and $K_1 \cap K_{I_2} = \emptyset$. Then, we use K_{I_2} to run the defrag_mbb algorithm with a bound $|T_2|$ on the number of rerouting events and the extra constraints that connection requests in K_1 cannot be rerouted (i.e., we set $\pi_k = 0$ for all $k \in K_1$). We obtain a subset $K_2 \subseteq K$ of connection requests such that $K_1 \cap K_2 = \emptyset$. We repeat the same procedure to find a subset $K_3 \subseteq K$ of connection requests such that $K_1 \cap K_3 = \emptyset$ and $K_2 \cap K_3 = \emptyset$. Finally, we

use $K_1 \cup K_2 \cup K_3$ as the initial set of columns for running the DEFRAG_MBB algorithm with the bound $|T|$ on the number of rerouting events. Observe that $|K_1 \cup K_2 \cup K_3|$ might be less than $|T|$ depending on the instance.

The number of subsets of rerouting events used in the second strategy can be adjusted depending on the value of T. In our experiments, we used only the first strategy when $|T| = 50$, the second strategy with two subsets of size 50 when $|T| = 100$, and the second strategy with three subsets of size 50 when $|T| = 150$. Reported computation times include the time needed for generating the initial columns and the resolution of the DEFRAG_MBB algorithm with 50 rerouting events. The second strategy is described in the Algorithm 4. In this algorithm, we demonstrate the case where T is divided into three smaller non-intersecting subsets.

Algorithm 3 Initial Set of Columns - Strategy 1

Require: G, current network state

Ensure: K_I, initial set of columns

```
1: K_I ← ∅
2: #reroutes ← 0
3: for k ∈ K do
4:     p^NEW ← SHORTEST_AVAIL_PATH(G, k)
5:     if p^NEW is shorter than k's current path then
6:         change k's current path to p^NEW
7:         K_I ← K_I ∪ {k}
8:         #reroutes ← #reroutes + 1
9:         update G's state                          ▷ route of k is changed
10:        if #reroutes = |T| then
11:            break
12: return K_I
```

3.5 Minimum Rerouting: DEFRAG_MIMO

In this section, we show how to modify the DEFRAG_SIM model in order to minimize the total number of rerouting operations required to obtain a solution satisfying a given bandwidth requirement (i.e., $BW^\star$ is used in this section). The DEFRAG_MIMO model, is formalized as follows.

Algorithm 4 Initial Set of Columns - Strategy 2

Require: G, current network state

Require: $T = T_1 \cup T_2 \cup T_3$, set of rerouting events

Ensure: K_I, initial set of columns

1: $K_I \leftarrow \emptyset$

2: $\#reroutes \leftarrow 0$

3: **for** i from 1 to 3 **do**

4: $\quad K_{I_i} \leftarrow$ initial set of columns using Strategy 1 with G, K, T_i

5: $\quad$ remove K_{I_i} from G

6: $\quad K \leftarrow K \setminus K_{I_i}$

7: $\quad K_I \leftarrow K_I \cup K_{I_i}$

8: **return** K_I

Master Problem

$$\min \quad \sum_{t \in T} \sum_{k \in K} \sum_{p \in P_k^t} z_{kp}^t \tag{3.32}$$

subject to Constraints (3.11)-(3.16) and:

$$\sum_{l \in L} C_l^{|T|} \leq BW^\star. \tag{3.33}$$

Constraint (3.33) ensures that the solution must be at least as good as the given bandwidth requirement $BW^\star$, and the Objective (3.32) is to minimize the total number of rerouting operations.

Observe that the DEFRAG_MIMO model might not admit a feasible solution if the given upper bound on the bandwidth requirement or on the number of rerouting events is too small. However, if these bounds are at least the objective value ($BW^\star$) obtained from the DEFRAG_SIM model with the same number $|T|$ of rerouting events, then the DEFRAG_MIMO model always admits a feasible solution (at least the solution of DEFRAG_SIM).

Pricing Problem

Let $u_t^{(3.11)} \leq 0$, $u_k^{(3.12)} \leq 0$ and $u_{\ell t}^{(3.14)} \gtreqless 0$ be the values of the dual variables associated with constraints (3.11), (3.12) and (3.14), respectively.

We use the following binary variables:

- $\pi_k = 1$ if $k \in K$ is selected for rerouting, 0 otherwise.

- $\alpha_{k\ell} = 1$ if $\pi_k = 1$ and the route of $k \in K$ uses link $\ell \in L$, 0 otherwise.

$$[\text{PP}^{\text{MIMO}}] \begin{cases} \min & \bar{c}_{\text{PP}_t} = 1 - u_t^{(3.11)} - \sum_{k \in K} u_k^{(3.12)} \pi_k \\ & \qquad - \sum_{\ell \in L} \sum_{k \in K} b_k u_{\ell t}^{(3.14)} (\pi_k a_{k\ell}^0 - \alpha_{k\ell}). \\ \text{Subject to} & \text{Constraints (3.18)-(3.23)} \end{cases}$$

In the new pricing problem PP^{MIMO}, the additional term (to the original pricing problem PP) is a constant, as the variables z_{kp}^t are introduced to the master objective function. Thus, this additional term does not change the pricing problem principle, i.e., the solution process of the DEFRAG_MBB algorithm can be applied completely to the DEFRAG_MIMO model.

3.6 Parallel Rerouting

Enabling parallel rerouting operations allows for reaching either a better provisioning within the specified bound on the number $|T|$ of rerouting events since more individual rerouting operations can be performed, or the same provisioning as DEFRAG_SIM within less rerouting events. In this section, we explore the latter advantage of parallel rerouting operations.

In the decomposition model DEFRAG_SIM, Constraints (3.11) restrict the number of rerouting operations per rerouting events to 1, although some rerouting operations could be safely done in parallel with respect to the MBB paradigm. Indeed, a subset $K' \subseteq K$ of the connection requests can be rerouted in parallel, or concurrently, at rerouting event t if the sum of their bandwidth requirements on the old and new routes does not exceed any link's capacity.

More formally, let $K_r \subseteq K$ be the subset of connection requests that are rerouted by DEFRAG_SIM, i.e., K_r contains each connection request $k \in K$ such that

$$\sum_{t \in T} \sum_{p \in P_k^t} z_{kp}^t = 1.$$

For convenience, we assume that $|K_r| = |T|$. Let $p_k \in \cup_{t \in T} P_k^t$ be the route selected for the rerouting of connection request $k \in K_r$. Recall that a connection request

can be rerouted at most once. Let B, indexed by rerouting event $t \in T$, be a set of bins. The bin B^t corresponds to a subset of the connection requests that can be safely rerouted in parallel at rerouting event t. Given a solution of DEFRAG_SIM, the problem of packing the connection rerouting operations into the minimum number of bins can be formalized as the ILP (3.34)-(3.42), using the following variables.

- $q_k^t = 1$ if $k \in K_r$ is packed into bin B^t, for some $t \in T$, 0 otherwise.

- $q^t = 1$ if bin B^t is used.

Observe that the rerouting operations of the connection requests in bin B^t must be performed before those of bin B^{t+1}. Hence, our problem combines a bin packing problem with a scheduling problem.

$$\text{Minimize} \qquad \sum_{t \in T} q^t \tag{3.34}$$

Subject to:

$$\sum_{t \in T} q_k^t = 1 \qquad k \in K_r \tag{3.35}$$

$$q_k^t \leq q^t \qquad k \in K_r, t \in T \tag{3.36}$$

$$q^{t+1} \leq q^t \qquad t \in T \tag{3.37}$$

$$C_\ell^t \leq C_\ell \qquad \ell \in L, t \in T \tag{3.38}$$

$$C_\ell^{t-1} + \sum_{k \in K_r} b_k \delta_\ell^{p_k} (1 - a_{k\ell}^0) q_k^t \leq C_\ell \quad \ell \in L, t \in T \tag{3.39}$$

$$C_\ell^t = C_\ell^{t-1} - \sum_{k \in K_r} b_k (a_{k\ell}^0 - \delta_\ell^{p_k}) q_k^t \quad \ell \in L, t \in T \tag{3.40}$$

$$q_k^t \in \{0, 1\} \qquad k \in K_r,\ t \in T \tag{3.41}$$

$$q^t \in \{0, 1\} \qquad t \in T \tag{3.42}$$

$$C_\ell^t \geq 0 \qquad \ell \in L,\ t \in T \tag{3.43}$$

The goal of this ILP is to minimize the number of bins (Objective (3.34)), hence minimizing the number of rerouting events needed to perform all parallel operations. Constraints (3.35) ensure that each connection request is packed into a single bin. Constraints (3.36) are used to identify used bins. Constraints (3.37) are used to break symmetries in the solution, ensuring that bin B^{t+1} can be used only if bin

B^t is used. Constraints (3.38)-(3.40) make sure that the capacity of a link is never exceeded. In particular, Constraints (3.39) ensure that the "make" part of the MBB operations of the connection requests in bin B^t respect the capacity constraints, i.e., there is enough capacity to establish all the new routes before releasing the capacity used by the legacy routes. Finally, Constraints (3.40) set the link capacities after the rerouting operations of rerouting event t are made. Constraints (3.41)-(3.43) define the domain of the variables.

The ILP formulation (3.34)-(3.43) is difficult to solve. Also, we now propose a simple greedy algorithm that packs rerouting operations into bins. Let $\sigma : T \to K_r$ be a mapping from rerouting events to connection requests, such that $\sigma(t)$ indicates the connection request that is rerouted at rerouting event t by DEFRAG_SIM. That is, $\sigma(t) = \sum_{k \in K_r} \sum_{p \in P_k^t} k \cdot z_{kp}^t$ since Constraints (3.11) ensure that, for each $t \in T$, a unique variable z_{kp}^t can be set to 1. We denote σ^{-1} the inverse mapping. So $\sigma^{-1}(k)$ is the rerouting event t at which connection request k is rerouted by DEFRAG_SIM.

Algorithm 5 Parallel Rerouting with respect to Original Order

Require: σ, mapping from rerouting events to connection requests
Require: T, set of rerouting events in original ordering
Ensure: B, bins containing requests rerouted in parallel

1: $B^1 \leftarrow \emptyset$ ▷ Initialize the first bin
2: $i \leftarrow 1$ ▷ Index of the current bin
3: **for** t from 1 to $|T|$ **do**
4: $k \leftarrow \sigma(t)$
5: **if** adding k to B^i violates Constraints (3.39) **then**
6: $i \leftarrow i + 1$
7: $B^i \leftarrow \emptyset$ ▷ Create a new bin
8: $B^i \leftarrow B^i \cup \{k\}$ ▷ Add k to current bin
9: **return** B

Algorithm 5 arranges the connection requests of K_r into bins, each bin corresponding to a set of connection requests that can safely be rerouted in parallel, and so fulfills Constraints (3.39). It proceeds as follows. After initializing the first bin, it considers the connection requests in the rerouting ordering given by DEFRAG_SIM (Lines 3-8). If the addition of connection request k to the current bin results in a violation of Constraints (3.39), then a new bin is created and k is added to it. Then

the algorithm considers the next connection requests until all connection requests have been placed into a bin.

Algorithm 5 ensures that if connection request k is placed into bin B^{t+1}, then it satisfies $\sigma^{-1}(k) > \sigma^{-1}(k')$ for all $k' \in B^t$. In other words, the connection requests in B^t are rerouted before those in B^{t+1}, which is consistent with the solution given by DEFRAG_SIM. Furthermore, the link capacity after the rerouting operations of bin B^t is exactly the same as the link capacity in the solution of DEFRAG_SIM after the rerouting of connection request $k = \sigma(\sum_{i=1}^{t} |B^i|)$. The number of bins created by Algorithm 5 depends on the solution given by DEFRAG_SIM. In the worse case, when no parallel rerouting operation is possible, it creates $|T|$ bins.

3.7 Numerical Results

3.7.1 Data Sets

We consider a network with 32 nodes and 250 directed links, which corresponds approximately to a Ciena customer network. Existing network connections were used to construct a traffic matrix input to a network simulator generating realistic random connection states. Connection requests had Poisson arrivals based on the traffic matrix and random durations drawn from a common exponential distribution. Each connection had a Weibull distributed bandwidth with a coefficient of variation of 0.3. Connections were routed on the shortest path (in hop count) that had sufficient bandwidth. A load factor parameter was used to globally vary the connection arrival rates: the corresponding equilibrium (after simulation start-up transients had disappeared) connection states represent a range of congestion levels from light (0.5) to heavy (1.0). For each load factor, we considered 10 reoptimization events. Reoptimization was performed with a period of 1 mean connection duration, ensuring that sufficient connection request arrival and termination events occurred in order to produce comparably degraded connection state. Characteristics of the data sets are described in Table 10, where for each load factor, we provide the average number of granted requests right before each reoptimization and the average number of connection requests that are not initially routed on a shortest path.

Table 10: Characteristics of the data sets

Load factor	Number of requests	# Req. not on shortest path
0.5	777.2	90.2
0.6	894.6	199.1
0.7	951.4	252.3
0.8	971.1	268.3
0.9	993.3	285.7
1.0	1,015.4	295.7

3.7.2 Comparison with the Model of Klopfenstein [1]

Table 11: Comparison with Klopfenstein: Traffic load 0.5, $|T| = 40$, number of connection requests = 250

Defrag. events	Bandwidth saving (%)		Accuracy ε (%)		# reroutings		Computing times (sec.) (limit = 1 hour)	
	DEFRAG_SIM	Klopfenstein [1]	DEFRAG_SIM	Klopfenstein [1]	DEFRAG_SIM	Klopfenstein [1]	DEFRAG_SIM	Klopfenstein [1]
210	5.7	5.7	0.00	0.00	16.0	21.0	14.7	2316.7
220	4.0	4.0	0.00	0.00	10.0	12.0	14.4	1020.8
230	3.4	2.9	0.00	0.55	9.0	8.0	13.4	**limit**
240	1.7	1.7	0.00	0.00	5.0	16.0	11.9	755.3
250	2.3	2.3	0.00	0.00	8.0	9.0	14.1	859.1
260	7.5	7.5	0.00	0.00	13.0	15.0	15.1	802.8
270	5.7	5.9	0.27	0.00	14.0	19.0	7.2	2249.1
280	3.0	3.0	0.00	0.00	9.0	11.0	13.9	895.1
290	2.3	2.3	0.00	0.00	6.0	13.0	13.9	479.1
300	7.1	4.5	0.00	2.81	17.0	11.0	11.6	**limit**
Average	4.3	4.0	0.03	0.34	10.7	13.5	13.0	1661.0
Ratio	1.1		0.08		0.8		0.0078 ($\approx 1/130$)	

We compared the performance of our model and algorithm with the model of Klopfenstein [1]. We use a dataset with a load factor of 0.5, $|T| = 40$ and for each reoptimization event, only 250 connections from the original connection set are taken into account. The differences are significant even for such small instances, as indicated by the results reported in Table 11. Therein, we report the results for each of the 10 reoptimization events. Since the Klopfenstein model can take enormous time to finish, we report the best solution found within a computation time limit of one hour. Note that due to these computation times, we were not able to perform comparisons on larger instances. Columns entitled DEFRAG_SIM correspond to the results obtained with the DEFRAG_SIM model and DEFRAG_MBB algorithm.

As expected, our model can be solved orders of magnitude faster than the compact ILP model of Klopfenstein [1], that is, about 130 times faster. Furthermore, the computation time limit of one hour was reached for two instances with the compact ILP model (reoptimization events indexed as 230 and 300). Moreover, our model yields on average a better accuracy (i.e., 0.03% versus 0.34%), because Klopfensteins's model was stopped by the time limitation twice, resulting in significant optimality gaps.

Obviously, when optimal solutions are obtained with both models, the bandwidth savings are equal. However, for the instances indexed 230 and 300 that were not optimally solved with the model of Klopfenstein, the solutions computed with our model offer better bandwidth savings. In addition, the solutions obtained with our model involve on average less rerouting operations. This can be explained by the facts that firstly, the minimization of the number of rerouting operations is part of none of the objective functions of the models, and that secondly, the model of Klopfenstein allows a connection request to be rerouted more than once.

3.7.3 Accuracy and Performance of the Reoptimization Solution

All statistics computed in this section correspond to averages over all the 10 reoptimization events performed for each load factor.

Table 12: Impact of the Initial Rerouting Configurations and Overall Number of Configurations

Load	# Initial potential reroutings			# Initial potential reroutings in the optimal solution						Overall # of generated potential reroutings		
	T = 50	T = 100	T = 150	T= 50		T = 100		T = 150		T = 50	T = 100	T = 150
0.5	43.3	68.1	70.0	18.5	(39.1%)	44.9	(63%)	42.0	(57.6%)	595.7	797.0	1401.0
0.6	50.0	99.0	145.4	15.3	(30.9%)	76.8	(77.4%)	112.8	(77.4%)	590.4	1157.7	1738.9
0.7	50.0	99.2	149.0	12.6	(25.4%)	63.9	(64.5%)	113.1	(75.8%)	546.0	1010.2	1408.2
0.8	50.0	98.6	148.5	14.1	(28.9%)	68.3	(69.4%)	110.2	(74.4%)	469.2	911.5	1348.5
0.9	50.0	99.1	149.1	12.6	(25.6%)	65.4	(66%)	122.8	(82.3%)	504.2	950.2	1283.4
1.0	50.0	99.5	149.5	12.1	(24.4%)	73.0	(73.7%)	120.5	(80.9%)	520.8	827.8	1207.8

Recall that we initialize the DEFRAG_MBB algorithm with at most $|T|$ potential rerouting operations (columns). When $|T| = 50$, we use the first strategy presented

in Section 3.4 for generating the initial set of columns. When $|T| = 100$ or $|T| = 150$, we use the second strategy. We have reported in Table 10 the average number of connection requests that are not routed on a shortest path in the legacy routing, and in Table 12 the average size of the initial sets of potential rerouting operations (initial columns) as produced by the first or second strategy. These sets are given as inputs to the DEFRAG_MBB algorithm. The relatively low number of initial potential reroutings for the instances with traffic load 0.5 is explained by the small number of connection requests (90.2 in average) that are not routed on a shortest path in the legacy routing.

Concerning the choice of $|T|$, which can be as large as the cardinality of the whole connection set, we conducted our experiments for values of $|T|$ of at most 150 in order to limit the computation time of our model to about one hour.

We now analyze the results reported in Table 12 on the efficiency of our strategies for generating the initial potential reroutings. Observe that the number of potential reroutings increases with the traffic load due to the increased number of routes that are not the shortest possible ones. For $|T| = 50$, where the initial potential reroutings are produced by the first strategy only, we observe in the middle set of columns of Table 12 that at most 39.1% of the initial potential reroutings appear in the final solutions. But, for $|T| = 100$ and $|T| = 150$, where we use the second strategy, at least 57.6% and up to 82.3% of these initial potential reroutings are part of the final solutions. This means that the time spent by the second strategy in the resolution of sub-problems results in very good choices of initial potential reroutings. Furthermore, it has a strong impact on the total number of generated potential reroutings, reported in the last set of columns of Table 12, and so on the number of times the pricing problem PP is solved. Indeed, the pricing problem generates a potential rerouting only if it is expected to improve the current solution of DEFRAG_MBB algorithm. Hence, when many initial potential reroutings are actually part of the final solution, fewer calls to the pricing problem are needed to solve the problem.

In Table 13, we report on the accuracy of the solutions. We first observe that the accuracy of computed solutions is always between 1% and 3%, which is satisfactory taking into account the additional computation time it would require for getting an optimal solution with a branch-and-price method [27], instead of the current solution process (Section 3.4). Furthermore, solutions obtained for larger values of $|T|$ have a

Table 13: Number of Reroutings and Accuracy of the Solutions

Load	Number of required rerouting			Accuracy (ε)			Computational times in minutes			ILP Pricing Problem computational times in seconds (number of solved ILPs)		
	T = 50	T = 100	T = 150	T = 50	T = 100	T = 150	T = 50	T = 100	T = 150	T = 50	T = 100	T = 150
0.5	47.2	71.2	72.9	1.2	1.7	1.5	1.5	7.1	16.1	0.0 (0.0)	0.0 (0.1)	0.0 (0.2)
0.6	49.4	99.1	145.6	2.0	1.6	1.6	2.0	20.4	89.7	0.3 (1.1)	0.2 (2.8)	3.6 (1.5)
0.7	49.6	99.0	149.1	2.4	1.3	1.2	1.8	20.0	65.9	2.0 (4.6)	0.7 (2.9)	2.2 (1.8)
0.8	48.7	98.4	148.1	2.7	1.6	1.2	1.7	15.8	63.2	1.5 (3.8)	2.0 (7.5)	2.1 (9.8)
0.9	49.1	99.0	149.1	2.2	1.3	1.2	2.0	16.8	64.3	0.4 (1.3)	0.6 (2.9)	9.7 (15.6)
1.0	49.5	99.0	148.9	2.3	1.4	1.0	2.2	14.3	45.9	1.8 (4.0)	4.0 (7.1)	3.7 (4.4)

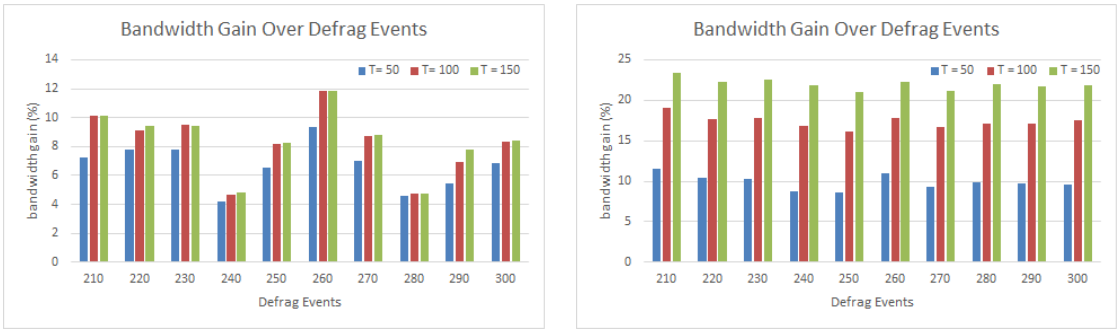
better (smaller) accuracy, except for load factor 0.5. This shows that the proposed solution process performs very well even on large instances.

Computation times, which include the generation of initial potential reroutings, are also quite reasonable, although too long for a real-time reoptimization operation. However, we expect to reduce them significantly in the near future with the addition of heuristics to speed up the solution process. Moreover, we observe that the overall computation time due to the resolution of the pricing problems using ILPs, which occur when the BFM algorithm fails to find a solution, is negligible. In the last part of Table 13, the main number is the averaged computing time spent on the ILP executions and the number in the parentheses is the averaged number of executions. We need less than 10 seconds (for at most 16 executions) for solving the ILPs of the pricing problems. This supports the effectiveness of the decomposition of PP into PP_t^k subproblems, each solved independently with the BFM algorithm.

3.7.4 Reoptimization Performance

We investigated the reduction of the overall bandwidth requirements after each reoptimization event. We have reported in Figure 10 the reduction after each reoptimization event for the two extreme load factors, i.e., 0.5 and 1.0. As already anticipated with the results of Table 13, enabling more than $|T| = 100$ rerouting operations for the instances with load factor 0.5 does not help further reducing the overall bandwidth usage. However, with load factor 1.0, enabling more rerouting operations allows for significant reduction of the bandwidth usage. More precisely, increasing $|T|$ from 50 to 100 leads to more than 5% gain, and increasing $|T|$ from 100 to 150 provides 5% additional bandwidth gain. An example of bandwidth usage evolution during a

sequence of rerouting operations is depicted in Figure 11 for the case of load factor 0.5 and $|T| = 50$. We observe that the bandwidth usage is essentially monotonically decreasing, although local increases of the bandwidth usage are possible with our model that only ensures that the final bandwidth usage is minimized.



(a) Load factor = 0.47 (b) Load factor = 1.0

Figure 10: Reduction (%) of bandwidth requirement

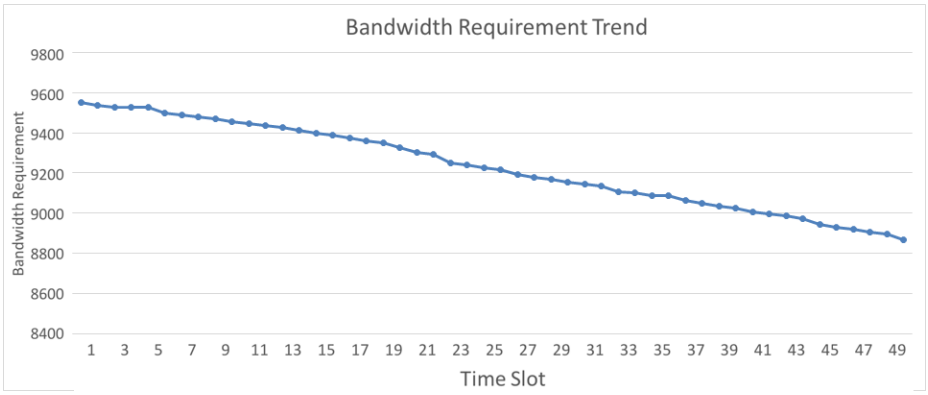


Figure 11: Trend of bandwidth usage after each rerouting operation on load 0.5 within $|T| = 50$.

3.7.5 Multiple Rerouting

In order to confirm the benefit of enabling multiple rerouting operations per connection requests, i.e., a connection is allowed to be rerouted more than once, we applied DEFRAG_SIM consecutively, taking as input the routing obtained in the previous call. More precisely, after first solving the problem with $|T| = 50$, we solve it again

with $|T| = 50$, starting from the routing R_1 computed by the first execution of DEFRAG_SIM, and then again producing R_3 taking as input the routing R_2 resulting from the second call.

In the first set of columns of Table 14, we ensure that a connection request is rerouted at most once. This is done by setting $\pi_k = 0$ for the connection requests that were previously rerouted. In the second set of columns of Table 14 we allow the reroute in R_2 (resp. R_3) of a connection request that has already been rerouted in R_1 (resp. R_1 or R_2). Hence, a connection request can be rerouted up to three times. We observe a small improvement in the bandwidth gain (around 0.1% for R_3) when enabling up to three reroutings per connection request.

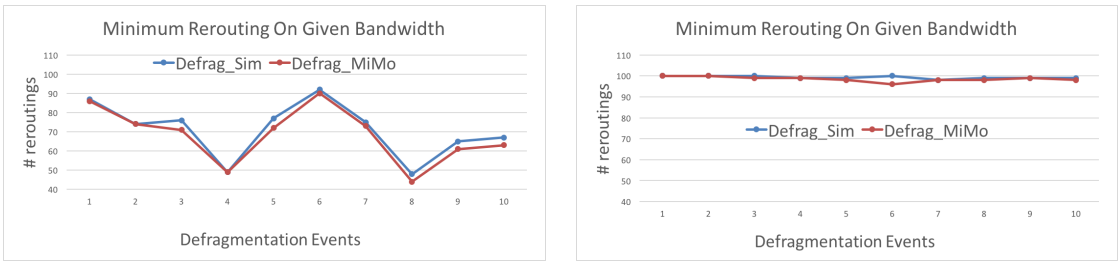
Table 14: Impact of Multiple Rerouting

Load	At most one rerouting per connection & t						Multiple rerouting per connection					
	# reroutings			bandwidth gain (%)			# reroutings			bandwidth gain (%)		
	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
0.5	47.2	20.9	1.9	6.7	8.0	8.2	47.2	21.0	2.5	6.7	8.1	8.2
0.6	49.4	49.6	46.4	9.4	14.9	18.4	49.6	49.3	46.3	9.4	14.8	18.5
0.7	49.6	49.6	49.8	9.8	16.6	21.0	49.1	49.7	49.8	9.8	16.6	21.1
0.8	48.7	49.9	49.9	9.9	16.8	21.5	48.7	49.5	49.6	9.9	16.8	21.4
0.9	49.1	50	50	10.1	17.2	22.1	49.4	49.6	50.0	10.1	17.2	22.2
1	49.5	50	50	9.9	17.1	21.9	49.7	49.8	50.0	9.9	17.1	21.9

3.7.6 Minimum Rerouting

The DEFRAG_MIMO model (Section 3.5) aims at minimizing the number of rerouting events to obtain a solution with specified quality (i.e., satisfying a given upper bound on the bandwidth usage). Figure 12 illustrates the results of DEFRAG_MIMO and DEFRAG_SIM for the instances with load factor 0.5 and 1.0, when $|T| = 100$. We observe that the reduction in the number of rerouting events offered by DEFRAG_MIMO over DEFRAG_SIM is small. With a 0.5 load factor, there are more connections which are being rerouted on their ultimate shortest path in terms of the number of links. In addition, although it is allowed to use up to $|T| = 100$ reroutes, it is usually not necessary to have so many before reaching an optimized or optimal rerouting (e.g., for events 4 and 8, only about half of the allowed number of reroutes are needed). With

a 1.0 load factor, i.e., in a competitive resource environment, there are many more connections that are routed over longer paths than the shortest possible. As a result, the solution must use more re-routings (more than 95 reroutings for all events). The reason is that the objective of minimizing the number of rerouting operations was implicitly involved into the DEFRAG_SIM model, since it is parameterized by $|T|$.



(a) Load factor = 0.5

(b) Load factor = 1.0

Figure 12: Number of Rerouting operations with DEFRAG_SIM and DEFRAG_MIMO

3.7.7 Parallel Rerouting

Table 15 demonstrates the efficiency of the parallel rerouting methodology (Section 3.6) in terms of the number of required rerouting events, i.e., a rerouting event will perform several rerouting operations in parallel provided that they do not violate the capacity limitation. Assuming that the duration of a rerouting event made of parallel rerouting operations is the same (or almost the same) as a single rerouting operation, applying parallel rerouting operations reduces by a factor 10 to 30 the cost of the reconfiguration procedure. In addition, the computing time for the DEFRAG_PAR model is remarkably less than for the DEFRAG_SIM model.

Table 15: Number of Parallel Rerouting Events

Load	Heuristic (Algorithm 5)			DEFRAG_PAR			DEFRAG_PAR Computational Time (min.)		
	T = 50	T = 100	T = 150	T = 50	T = 100	T = 150	T = 50	T = 100	T = 150
0.5	7.5	12.2	12.8	4.6	6.1	5.6	3.6	5.8	6.0
0.6	9.4	16.6	20.7	5.1	6.6	7.5	3.9	8.4	13.8
0.7	9.2	16.5	21.1	5.5	7.0	6.7	3.9	8.5	14.2
0.8	9.2	16.3	21.2	5.3	7.4	6.8	3.9	8.4	12.9
0.9	8.6	16.2	20.3	4.8	6.2	6.3	4.0	8.4	12.9
1.0	10.5	17.6	21.8	5.9	6.4	6.3	4.1	8.5	13.5

3.8 Conclusion

We have proposed a new model for progressive reoptimization, i.e., computing a sequence of make-before-break reroutings leading to the minimum bandwidth requirements. It corresponds to a huge improvement with respect to the previous model proposed by Klopfenstein [1] as we reach up to 150 reroutings in less than a few hours, while the model of [1] was only scalable for toy problems.

We plan to further study the proposed model so that it can handle the case of more than one rerouting per (selected) connection. In addition, we plan to study the adaptation of our optimization model to other seamless migrations, e.g., to requests in Content Delivery Networks (CDNs), [45].

Chapter 4

A nested decomposition model for generalized MBB reoptimization

Capacity fragmentation, incurred by dynamic traffic, reduces network efficiency. Reoptimization is a connection rerouting process improving system utilization. In the fifth telecommunication generation technology (5G), a physical network provision several virtual networks. So a virtual network is more dynamic and flexible in terms of operation. Although there are many objectives for the reoptimization of capacity networks, the most focused goal is to reduce as much as possible the total capacity used. Thus, in this work, we are using this aim.

This work takes into account multiple and parallel rerouting. To reduce the difficulty of the problem, studied works removes multiple and parallel conditions. This paper proposes decomposition mathematical models and column generation algorithms to solve this complicated problem. This proposal is not only more generalized but also reduced the fastest CPU time in [46] by about one order of magnitude.

This paper is in preparation.

4.1 Introduction

Capacity fragmentation is incurred by dynamic traffic and it reduces network efficiency. Reoptimization is a connection rerouting process by which system utilization is improved. In the context of the fifth telecommunication generation technology (5G), logical networks are separated from physical infrastructure, so they are more

dynamic and flexible in terms of operation. Although there are many objectives for reoptimization, for capacity networks, the most focused goal is to reduce as much as possible the whole capacity used. Thus in this work, we also use this objective.

To reroute a connection, the operator first finds a new route for this connection, then moves the connection from the current route to the new route. If the connection is switched after the current route is terminated, so there is a disruption of this connection and it is not recommended. Therefore the make-before-break (MBB) paradigm is widely used in practice. In MBB, before the current route is terminated, the new route is already established and transferring data of the connection. Once the whole data is transferred by the new route, the old route is freed. By this paradigm, a connection from the source to destination is not disrupted.

In [47, 46], although authors improved significantly the size of the solved instances in literature, they did not take into account parallel and multiple rerouting. Parallel rerouting allows several connections to be rerouted at the same time resulting in a shorter reoptimization duration. It is an important factor as the network cannot accept or terminate connections during reoptimization. On the other hand, multiple rerouting setting allows a connection can be rerouted more than once in a reoptimization process. This setting helps to reduce further capacity uses as it breaks difficult rerouting situation. This multiple rerouting introduces a huge difficulty to formulate the problem than one presented in [47, 46].

There are a few studies proposed such parallel and multiple reoptimization. Model in [1] can be considered as the state of the art for this problem. In this paper, we propose an optimization model that takes into account both parallel and multiple MBB rerouting for a capacity network. Several column generation algorithms are also presented to solve this model. Numerical results show that our algorithm is remarkably faster than the state of the art as in [1].

The paper is organized as follows. We briefly review in Section 4.2 the papers related to reoptimization in logical/capacity networks. We next describe in Section 4.3 our problem formally as a decomposition mathematical model. This model is solved by the column generation algorithm which is explained in Section 4.4. Numerical results are presented in Section 4.5. Conclusions are drawn in the last section

4.2 Literature Review

In this section, recent studies of reoptimization/defragmentation are presented. Note that Layer 2 connections do not have continuity and contiguity constraints, this section only discuss Layer 0 defragmentation works that are related to Layer 2 reoptimization in terms of strategy and mathematics.

Several studies have been devoted to network reconfiguration with the minimum number of disruptions, following the strategy of migrating from a legacy ineffective provisioning to a given pre-computed optimized/optimal one. As a result, it usually prevents the existence of a strategy using only MBB due to the presence of dependency cycles as explained in the introduction. In order to find a rerouting strategy, authors have then proposed to use the Break-Before-Make (BBM) paradigm sparingly to allow temporary interruption of connection requests, and so to break dependency cycles. For instance, Jose and Somani [35] propose heuristics for minimizing the total number of BBMs used in the rerouting strategy, and Coudert *et al.* [33, 38] and Solano and Pióro [14] provide scalable exact algorithms to minimize the concurrent number of BBMs. Tradeoffs between these two conflicting objectives are investigated by Cohen *et al.* [34] and Solano [36].

To further reduce the total or concurrent number of BBMs, Kadohata *et al.* [37] propose to use spare wavelengths to reroute a connection request to a temporary route rather than using a BBM. For example, assume that the current connection k needs to be rerouted from path p to path p', but such a rerouting cannot be MBB due to resource dependence. Then one unavoidable BBM reroute is performed. However, using an intermediate reroute, it may be possible to reroute k under the MBB paradigm. For instance, assume that there exists a path p'' such that the reroutings from p to p'' and from p'' to p' satisfy the MBB condition. In other words, one BBM can be avoided at the expense of performing two MBBs. This idea is similar to multiple rerouting for capacity reoptimization.

The idea of the second direction is to compute the best provisioning that is reachable from the legacy provisioning by a sequence of connection reroutings with no disruption, i.e., under the so-called MBB paradigm. While many studies have investigated the first direction, this second direction has received very little attention [1, 47].

4.3 Problem Statement

4.3.1 Notations

We consider a network represented by a directed multi-graph $G = (V, L)$, where V is the set of nodes (indexed by v) and L is the set of links (indexed by ℓ). Different links may exist between two nodes in order to model different logical links, with e.g., different types of traffic. We denote $\omega^-(v)$ (resp. $\omega^+(v)$) the set of incident links incoming to (resp. outgoing from) node $v \in V$. Let C_ℓ denote the transport capacity of link ℓ.

Let K be the set of connection requests (indexed by k). Connection request $k \in K$ is characterized by its source s_k, its destination d_k, and its bandwidth requirement b_k. In what follows, we call rerouting operation the action of rerouting a connection request $k \in K$, and rerouting event the action of either performing a single rerouting operation, or a set of parallel rerouting operations. A reoptimization event is an ordered sequence of rerouting events, and so of rerouting operations. Let $T = \{1, 2, ..., |T|\}$, be the set of rerouting event indices of a reoptimization event. Hence, $t \in T$ designates one rerouting event.

A life-line configuration, $\gamma \in \Gamma_k$, is a "life-line" of the connection k which is characterized by:

- λ_t^γ is 1 if k is rerouted at rerouting event t, 0 otherwise.
- $\alpha_{\ell t}^\gamma$ is 1 if k uses link ℓ at the end of rerouting event t.
- $\delta_{\ell t}^\gamma = \alpha_{\ell t-1}^\gamma - \alpha_{\ell t}^\gamma$ is the difference in usage of link ℓ between the begin of rerouting event t and its end.
- $x_{\ell t}^\gamma$ is 1 if k does not use link ℓ after rerouting event $t-1$ and uses link ℓ after rerouting event t.

Note that a connection can have as many life-lines as feasible rerouting, and a solution has to determine exactly one life line for each connection.

4.3.2 Multiple Parallel MBB Reoptimization Model

Assuming that all feasible life-line configurations are enumerated, the multiple parallel MBB reoptimization model (MUL_PAR_RO) is defined formally by following

decomposition mathematical formulation.

The integer linear programming (ILP) formulation of MUL_PAR_RO uses the following variables:

- z_γ is binary variable that is set to 1 if life-line γ is selected, 0 otherwise.
- C_ℓ^t = bandwidth requirement on link $\ell \in L$ after rerouting event $t \in T$.

It also uses the following parameters:

- Γ_k is the set of all feasible life-line configurations for connection $k \in K$.
- $a_{k\ell}^{\text{INIT}} = 1$ if link $\ell \in L$ is used in the initial routing of connection request $k \in K$, 0 otherwise.
- $C_\ell^{\text{INIT}} = \sum_{k \in K} b_k a_{k\ell}^{\text{INIT}}$ = initial bandwidth usage on link $\ell \in L$.
- R^{PAR} = limit on the number of parallel rerouting operations at a rerouting event.
- R^{MUL} = limit on the number of rerouting operations for connection k.

$$\text{Minimize} \qquad \sum_{\ell \in L} C_\ell^{|T|} \tag{4.1}$$

$$\text{s.t.:} \sum_{k \in K} \sum_{\gamma \in \Gamma_k} \lambda_t^\gamma z_\gamma \leq R^{\text{PAR}} \qquad t \in T \tag{4.2}$$

$$\sum_{\gamma \in \Gamma_k} z_\gamma \leq 1 \qquad k \in K \tag{4.3}$$

$$C_\ell^t \leq C_\ell \qquad \ell \in L, t \in T \tag{4.4}$$

$$C_\ell^{\text{INIT}} = \sum_{k \in K} b_k a_{k\ell}^{\text{INIT}} \qquad \ell \in L \tag{4.5}$$

$$C_\ell^{t-1} + \sum_{k \in K} \sum_{\gamma \in \Gamma_k} b_k \lambda_t^\gamma x_{\ell t}^\gamma z_\gamma \leq C_\ell \qquad \ell \in L, t \in T \tag{4.6}$$

$$C_\ell^{t-1} + \sum_{k \in K} \sum_{\gamma \in \Gamma_k} b_k \lambda_t^\gamma \delta_{\ell t}^\gamma z_\gamma \leq C_\ell^t \qquad \ell \in L, t \in T \tag{4.7}$$

$$C_\ell^t \geq 0 \qquad \ell \in L, t \in T \tag{4.8}$$

$$z_\gamma \in \{0, 1\} \qquad k \in K, \gamma \in \Gamma_k \tag{4.9}$$

The objective (4.1) is to minimize the capacity usage at the end of the reoptimization event. Constraints (4.2) prevent the selection of more than R^{PAR} rerouting operations at each rerouting event. Note that $|T| \leq |K|$ is an upper bound on the number of rerouting events as we cannot predict the number of required MBB rerouting events and operations. In case one does not want to restrict the number of parallel rerouting operations per rerouting event, R^{PAR} is set to infinite and constraints (4.2) is removed from the model. Constraints (4.3) ensure that a connection request has at most one life-line. Constraints (4.4) make sure that transport capacities are never exceeded after any rerouting event.

Constraints (4.5) specifies the initial bandwidth usage of each link. Constraints (4.6) ensure that the bandwidth which is needed on link ℓ for the "make" part does not exceed its capacity at rerouting event t. Note that if the old and new routes of a connection go through the same link, reserved capacity on that link is not duplicated. Constraints (4.7) update the bandwidth usage on link ℓ after rerouting event t. Constraints (4.8)-(4.9) define the domain of the variables.

This formulation has one z_γ variable per feasible life-line configuration of connection request $k \in K$. Hence, this formulation has an exponential number of variables. We explain in next section how to solve it using column generation.

4.4 Solution Process

In this section, two column generation algorithms are presented. The first one uses compact formulation for pricing problem, so it directly applies classical column generation algorithm. To accelerate the resolution of the first algorithm, the second one uses a decomposition model for the pricing problem and solves that model using column generation as well.

4.4.1 Compact Life Line Pricing Algorithm - CPP

The model (4.1)-(4.9) has an exponential number of variables, and therefore column generation [29] is required in order to efficiently solve its linear relaxation.

This technique consists of decomposing the original problem into a Restricted Master Problem (RMP), i.e., model (4.1)-(4.9) with a very restricted number of variables, and one or several pricing problems (PPs). In the particular case of model

(4.1)-(4.9), we will show in this section that the pricing problem can be decomposed into $|K|$ independent smaller pricing problems, each denoted by PP_k. The RMP and the PP(s) are solved alternately. Solving the RMP consists in selecting the best connection life-lines, while solving the PPs allows for the generation of new columns, i.e., potential connection life-line, and more precisely, a sequence of routes such that, if added to the current RMP, improves the optimal value of its linear relaxation.

The process continues until the optimality condition is satisfied, that is, all the so-called reduced costs that define the objective function of the pricing problems are positive (see [29] if not familiar with linear programming concepts). An ε-optimal solution is derived by solving exactly the ILP model associated with the last RMP, with ε defined as follows:

$$\varepsilon = \left(\tilde{\zeta}_{\text{ILP}} - \zeta^{\star}_{\text{LP}}\right) / \zeta^{\star}_{\text{LP}}, \tag{4.10}$$

where $\zeta^{\star}_{\text{LP}}$ and $\tilde{\zeta}_{\text{ILP}}$ denote the optimal LP value and the optimal ILP value of the last RMP, respectively.

The solution process is illustrated in the flowchart of Figure 13.

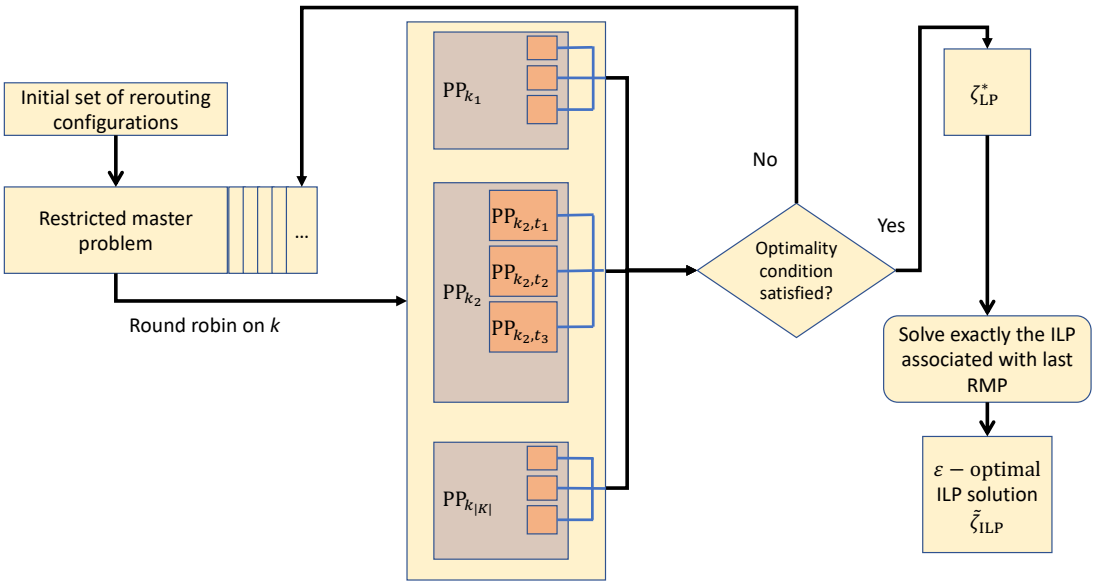


Figure 13: CPP Algorithm

The compact pricing problem model is described as follows.

Note that parameters of a life-line configuration become according variables in the pricing problem, thus their names are not changed.
Parameters:

- $u_t^{(4.2)} \leq 0$, $u_k^{(4.3)} \leq 0$, $u_{\ell t}^{(4.6)} \leq 0$, and $u_{\ell t}^{(4.7)} \leq 0$ are dual values from constraints (4.2), (4.3), (4.6) and (4.7) respectively.

Variables:

- $\alpha_{\ell t}$ is 1 if the routing of connection k uses link $\ell \in L$ at the end of rerouting event t, 0 otherwise.
- λ_t is 1 if t is a rerouting event, i.e. if a rerouting operation occurs, 0 otherwise.
- $x_{\ell t}$ is 1 if link $\ell \in L$ is used at the end of rerouting event t but was not used at $t-1$, 0 otherwise.
- $\delta_{\ell t}$ is 1 if link $\ell \in L$ is used at t but not at $t-1$, -1 if it was used at $t-1$ and is no longer used, and 0 if its usage is unchanged.

Objective (reduced cost):

$$\min \quad \overline{c}_{\mathrm{PP}_k} = -\sum_{t \in T} u_t^{(4.2)} \lambda_t - u_k^{(4.3)} - \sum_{\ell \in L} \sum_{t \in T} u_{\ell t}^{(4.6)} b_k \lambda_t x_{\ell t} - \sum_{\ell \in L} \sum_{t \in T} b_k u_{\ell t}^{(4.7)} \lambda_t \delta_{\ell t} \quad (4.11)$$

subject to:

$$\sum_{\ell \in \omega^-(v)} \alpha_{\ell t} - \sum_{\ell \in \omega^+(v)} \alpha_{\ell t} = \begin{cases} 1 & \text{if } v = s_k \\ -1 & \text{if } v = d_k \\ 0 & \text{otherwise} \end{cases}$$

$$t \in T, v \in V \tag{4.12}$$

$$\sum_{\ell \in \omega^+(v)} \alpha_{\ell t} \leq 1 \qquad t \in T, v \in V \tag{4.13}$$

$$\alpha_{\ell 0} = a_{k\ell}^{\text{INIT}} \qquad \ell \in L \tag{4.14}$$

$$\sum_{t \in t} \lambda_t \leq R^{\text{MUL}} \tag{4.15}$$

$$\lambda_t \geq \alpha_{\ell,t-1} - \alpha_{\ell,t} \qquad t \in T, \ell \in L \tag{4.16}$$

$$x_{\ell t} \geq \alpha_{\ell,t} - \alpha_{\ell,t-1} \qquad t \in T, \ell \in L \tag{4.17}$$

$$x_{\ell t} \leq \lambda_t \qquad t \in T, \ell \in L \tag{4.18}$$

$$\sum_{\ell \in L} x_{\ell t} \geq \lambda_t \qquad t \in T \tag{4.19}$$

$$\delta_{\ell t} = \alpha_{\ell,t} - \alpha_{\ell,t-1} \qquad t \in T, \ell \in L \tag{4.20}$$

$$\alpha_{\ell t} \in \{0, 1\} \qquad t \in T, \ell \in L \tag{4.21}$$

$$\lambda_t \in \{0, 1\} \qquad t \in T \tag{4.22}$$

$$x_{\ell t} \in \{0, 1\} \qquad t \in T, \ell \in L \tag{4.23}$$

$$\delta_{\ell t} \in \{-1, 0, 1\} \qquad t \in T, \ell \in L. \tag{4.24}$$

Note that the objective function (4.11) is quadratic. However, the first quadratic term $\lambda_t x_{\ell t}$ can can be equivalently rewritten $x_{\ell t}$ due to Constraints (4.18), while the second quadratic term $\lambda_t \delta_{\ell t}$ can be equivalently rewritten $\delta_{\ell t}$ due to combination of Constraints (4.16) and (4.20). Hence, we get reduced cost:

$$\overline{c}_{\text{PP}_k} = -\sum_{t \in T} u_t^{(4.2)} \lambda_t - u_k^{(4.3)} - \sum_{\ell \in L} \sum_{t \in T} u_{\ell t}^{(4.6)} b_k x_{\ell t} - \sum_{\ell \in L} \sum_{t \in T} b_k u_{\ell t}^{(4.7)} \delta_{\ell t}. \tag{4.11-a}$$

Model (4.12)-(4.24) describes the life-line of request k. Constraints (4.12) are flow conservation constraints defining a path for each connection after each rerouting operation, while avoiding loops along the path with constraints (4.13).

Note that this model may still create isolated loops, disconnected from the path. However, this issue will be solved by the nested decomposition scheme. Indeed, the second level decomposition is a set of path configurations. Thus, if path generation imposes the simple path condition, the decomposition model will not contain isolated loop, disconnected from a path.

Constraints (4.14) specify the links that are used in the initial routing of request k. Constraint (4.15) limits the number of rerouting operations on the life-line of connection k. Constraints (4.16)-(4.19) are used to identify rerouting operations and so rerouting events. More precisely, assume first that $\lambda_t = 0$. Then, Constraints (4.16) ensure that the links used at $t-1$ are still used at t, and Constraints (4.17)-(4.18) prevent from using new links at t. Hence, the routing at $t-1$ and t are the same. Now, suppose that a rerouting operation occurs at t, that is $\lambda_t = 1$. Constraints (4.19) ensure that at least one variable $x_{\ell t}$ is set to 1, thus enabling with Constraints (4.18) to use at t a link ℓ that was not used at $t-1$. Furthermore, Constraints (4.16) now enable to stop using at t some links that were used at $t-1$. On the other way around, if either a link is no longer used at t, or a link is used at t but was not used at $t-1$, Constraints (4.17)-(4.18) identify that a rerouting event occurs at t and set variable λ_t to 1. If no new link is used at t, Constraints (4.19) force variable λ_t to 0, Constraints (4.16) force to continue using at t the links that were used at $t-1$, and so the paths at $t-1$ and t are the same.

Constraints (4.20) encode in $\delta_{\ell t}$ the changes in links usage. Finally, Constraints (4.21)-(4.24) define the domains of the variables.

4.4.2 Decomposition Life Line Pricing Algorithm - DPP

As model (4.11)-(4.24) is an ILP, it can be reformulated as a decomposition model and solved by column generation algorithm. However, if the reformulated pricing problem is not solved exactly, the optimal condition cannot be verified as in CPP. In consequence, there is no guarantee that the obtained linear relaxation solution of (4.1)-(4.9) is a lower bound of the original problem. Fortunately, this lower bound for nested decomposition model is derived using simple computation applying Lagrangian relaxation as in [48, 49].

Upper Pricing $\mathbf{PP}_k$

A path configuration, $\pi \in \Pi_k^t$, is a simple path (i.e., without loops) of the connection k after the rerouting event t which is characterized by:

- a_ℓ is 1 if k uses link ℓ after the rerouting event t

Parameters:

- $u_t^{(4.2)} \leq 0$, $u_k^{(4.3)} \leq 0$, $u_{\ell t}^{(4.6)} \leq 0$, and $u_{\ell t}^{(4.7)} \leq 0$ are dual values from constraints (4.2), (4.3), (4.6) and (4.7) respectively.

Variables:

- y^π is 1 if the path configuration π is selected, 0 otherwise.

Objective (reduced cost):

$$\min \quad \overline{c}_{\mathrm{PP}_k} = -\sum_{t \in T} u_t^{(4.2)} \lambda^t - u_k^{(4.3)} - \sum_{\ell \in L} \sum_{t \in T} u_{t,\ell}^{(4.6)} b_k x_{\ell t} - \sum_{\ell \in L} \sum_{t \in T} b_k u_{\ell t}^{(4.7)} \left(\sum_{\pi \in \Pi_k^t} \alpha_\ell^\pi y^\pi - \sum_{\pi \in \Pi_k^{t-1}} \alpha_\ell^\pi y^\pi \right) \quad (4.25)$$

subject to:

$$\sum_{\pi \in \Pi_k^t} y^\pi = 1 \qquad t \in T \quad (4.26)$$

$$\sum_{t \in t} \lambda^t \leq R^{\mathrm{MUL}} \quad (4.27)$$

$$\sum_{\pi \in \Pi_k^t} \alpha_\ell^\pi y^\pi - \sum_{\pi \in \Pi_k^{t-1}} \alpha_\ell^\pi y^\pi \leq \lambda^t \qquad t \in T, \ell \in L \quad (4.28)$$

$$\sum_{\pi \in \Pi_k^t} \alpha_\ell^\pi y^\pi - \sum_{\pi \in \Pi_k^{t-1}} \alpha_\ell^\pi y^\pi \leq x_{\ell t} \qquad t \in T, \ell \in L \quad (4.29)$$

$$\lambda^t \in \{0, 1\} \qquad t \in T \quad (4.30)$$

$$x_{\ell t} \in \{0, 1\} \qquad t \in T, \ell \in L \quad (4.31)$$

$$y^\pi \in \{0, 1\} \qquad \pi \in \Pi. \quad (4.32)$$

Constraints (4.26) ensure that exactly one configuration (path) is selected per event. Constraint (4.27) limits the number of rerouting operations on the life-line

of connection k. Constraints (4.28) identify rerouting events. Indeed, if the path configurations at t and $t-1$ are different, there is at least one link $\ell \in L$ used at t that was not used at $t-1$, and so the left hand side of the inequality is 1. In addition, when $\lambda_t = 0$, the path configurations at t and $t-1$ are necessarily the same, otherwise contradicting that all considered path configurations are simple. Constraints (4.29) identify the links used at t and not at $t-1$. Finally, Constraints (4.30)-(4.32) define the domains of the variables.

Model (4.25)-(4.32) has one variable y^π per path configuration $\pi \in \Pi_k^t$, and so an exponential of variables. Therefore, we use column generation to solve it, and more precisely, we use pricing problem PP_k^t to generate path configurations for connection request $k \in K$ at event $t \in T$.

Pricing Problem PP_k^t

Parameters:

- $u_t^{(4.2)} \leq 0$, $u_k^{(4.3)} \leq 0$, $u_{t,l}^{(4.28)} \leq 0$, and $u_{t,l}^{(4.29)} \leq 0$ are dual values from constraints (4.2), (4.3), (4.6) and (4.7) respectively.

Variables:

- α_ℓ is 1 if link ℓ is used on this path, 0 otherwise.

$$\begin{aligned}\min \quad \bar{c}_{\text{PP}_k^t} = &-b_k u_{t,l}^{(4.7)} \alpha_\ell + b_k u_{t+1,l}^{(4.7)} \alpha_\ell \\ &- b_k u_{t,l}^{(4.28)} \alpha_\ell + b_k u_{t+1,l}^{(4.28)} \alpha_\ell \\ &- b_k u_{t,l}^{(4.29)} \alpha_\ell + b_k u_{t+1,l}^{(4.29)} \alpha_\ell \end{aligned} \tag{4.33}$$

$$\text{s.t.:} \sum_{\ell \in \omega^-(v)} a_\ell - \sum_{\ell \in \omega^+(v)} a_\ell = \begin{cases} -1 & \text{if } v = s_k \\ 1 & \text{if } v = d_k \\ 0 & \text{otherwise} \end{cases}$$

$$v \in V \tag{4.34}$$

$$\sum_{\ell \in \omega^+(v)} a_\ell \leq 1 \qquad v \in V \tag{4.35}$$

$$a_\ell \in \{0,1\} \qquad \ell \in L. \tag{4.36}$$

Note that Π_k^0 has exactly one configuration corresponding to current path of k. Constraints (4.34) are flow conservation constraints ensuring the establishment of a path from s_k to d_k. Constraints (4.35) prevent loops associated with a node. Constraints (4.36) define the domain of the variables.

Note that this is the weighted simple shortest path problem. To solve it, we first use an efficient combinatorial algorithm such as the Bellman-Ford algorithm. In case that algorithm detects a negative loop, and so does not return a path, we solve the more expensive ILP model (4.33)-(4.36).

Solution Process

Solution process for DPP is depicted in Figure 14.

Similarly to [48, 49], the bound $\overline{\zeta}_{\text{LP}}^{\tau}$ is computed as:

$$\overline{\zeta}_{\text{LP}}^{\tau} = u_{\tau} b + \sum_{k \in K} RC_{\text{LP}}^{\star} \left(\overline{c}_{\text{PP}_k} \right). \tag{4.37}$$

Note that the Lagrangian relaxation upper bound does not improve monotonically [50], thus, in order to derive the best possible upper bound, the algorithm must use:

$$\overline{\zeta}_{\text{LP}} = \min_{\tau} \overline{\zeta}_{\text{LP}}^{\tau} = \min_{\tau} \left\{ u_{\tau} b + \sum_{k \in K} RC_{\text{LP}}^{\star,\tau} \left(\overline{c}_{PP_k} \right) \right\}. \tag{4.38}$$

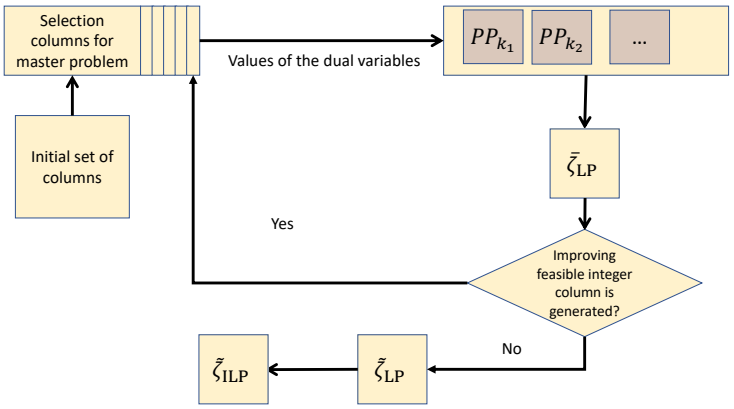
where τ represents a iteration of column generation and $RC_{\text{LP}}^{\star}(\overline{c}_{PP_k})$ is the optimal of linear-relaxed restricted-master-problem of pricing problem.

4.4.3 Solution Process of Non-Multiple (Single) Rerouting per Connection

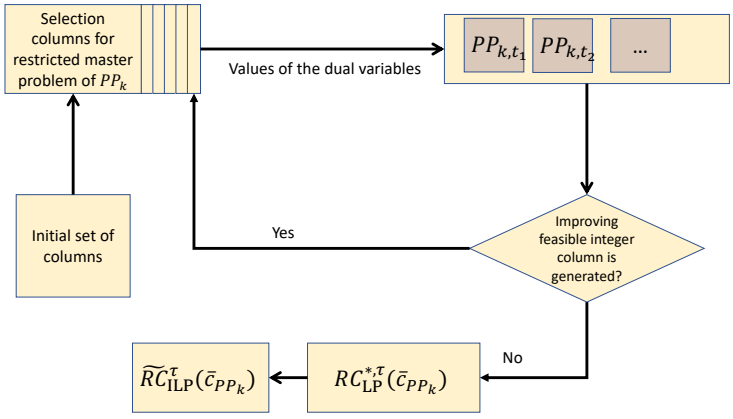
When $R^{\text{MUL}} = 1$ for all $k \in K$, each connection k can be rerouted at most once. In other words, at most one λ_t can be one. As in Huy *et al.* [46], we can easily decompose the compact pricing problem into a set of pricing problems $\text{PP}_{kt}^{\text{SR}}$ in which $\lambda_t = 1$. We then get:

$$\overline{c}_{\text{PP}_k^{\text{SR}}} = \min_{t \in T} \overline{c}_{\text{PP}_{kt}^{\text{SR}}} = -u_k^{(4.3)} - \max_{t \in T} \left(u_t^{(4.2)} + \sum_{\ell \in L} b_k u_{\ell t}^{(4.6)} x_{\ell t} + \sum_{\ell \in L} b_k u_{\ell t}^{(4.7)} \delta_{\ell t} \right).$$

Assuming $\lambda_t = 1$, Constraints (4.16) and (4.18) become redundant for the selected t, and can therefore be eliminated. The simplified pricing problem $\text{PP}_{kt}^{\text{SR}}$ with single rerouting per connection can be written as follows.



(a) Upper level flowchart



(b) Lower level flowchart

Figure 14: Decomposition Pricing Problem Algorithm

Parameters:

- $u_t^{(4.2)} \leq 0$, $u_k^{(4.3)} \leq 0$, $u_{\ell t}^{(4.6)} \leq 0$, and $u_{\ell t}^{(4.7)} \leq 0$ are dual values from constraints (4.2), (4.3), (4.6) and (4.7) respectively.

Variables:

- $\alpha_{\ell t}$ is 1 if the routing of connection k uses link $\ell \in L$ at the end of rerouting event t, 0 otherwise.
- λ_t is 1 if t is a rerouting event, i.e. if a rerouting operation occurs, 0 otherwise.
- $x_{\ell t}$ is 1 if link $\ell \in L$ is used at the end of rerouting event t but was not used at $t-1$, 0 otherwise.
- $\delta_{\ell t}$ is 1 if link $\ell \in L$ is used at t but not at $t-1$, -1 if it was used at $t-1$ and is no longer used, and 0 if its usage is unchanged.

Objective (reduced cost):

$$\begin{aligned} \min \quad \bar{c}_{\mathrm{PP}_{kt}^{\mathrm{SR}}} &= -u_t^{(4.2)} - u_k^{(4.3)} - \sum_{\ell \in L} b_k u_{\ell t}^{(4.6)} x_{\ell t} - \sum_{\ell \in L} b_k u_{\ell t}^{(4.7)} \left(\alpha_{\ell t} - a_{k\ell}^{\mathrm{INIT}}\right) \\ &= \underbrace{-u_t^{(4.2)} - u_k^{(4.3)} + \sum_{\ell \in L} b_k u_{\ell t}^{(4.7)} a_{k\ell}^{\mathrm{INIT}}}_{\text{constant}} - b_k \sum_{\ell \in L} (u_{\ell t}^{(4.6)} x_{\ell t} + u_{\ell t}^{(4.7)} \alpha_{\ell t}) \end{aligned} \quad (4.39)$$

subject to:

$$\sum_{\ell \in \omega^-(v)} \alpha_{\ell t} - \sum_{\ell \in \omega^+(v)} \alpha_{\ell t} = \begin{cases} -1 & \text{if } v = s_k \\ 1 & \text{if } v = d_k \\ 0 & \text{otherwise} \end{cases}$$

$$v \in V \tag{4.40}$$

$$\sum_{\ell \in \omega^+(v)} \alpha_{\ell t} \leq 1 \qquad v \in V \tag{4.41}$$

$$x_{\ell t} \geq \alpha_{\ell t} - 0 \qquad \ell \in L_k : \alpha_{\ell, t-1} = 0 \tag{4.42}$$

$$x_{\ell t} \geq \alpha_{\ell t} - 1 \qquad \ell \in L_k : \alpha_{\ell, t-1} = 1 \tag{4.43}$$

$$\sum_{\ell \in L} x_{\ell t} \geq 1 \tag{4.44}$$

$$\alpha_{\ell t} \in \{0, 1\} \qquad \ell \in L \tag{4.45}$$

$$x_{\ell t} \in \{0, 1\} \qquad \ell \in L \tag{4.46}$$

Proposition 1. *If we look for only negative optimal objective value (reduced cost), pricing problem* $\text{PP}_{kt}^{\text{SR}}$ *can be reduced to a shortest path problem with non negative weights.*

Proof. Firstly, we show that variables x can be eliminated. Let

$$u_{\ell t}^{(4.6)(4.7)} = \begin{cases} u_{\ell t}^{(4.7)} & \text{if } a_{k,\ell}^0 = 1 \\ u_{\ell t}^{(4.6)} + u_{\ell t}^{(4.7)} & \text{if } a_{k,\ell}^0 = 0 \end{cases}. \tag{4.47}$$

Consider the following model.

$$\min \quad \overline{cc}_{\text{PP}_{kt}^{\text{SR}}} = -u_t^{(4.2)} \lambda^t - u_k^{(4.3)} + \sum_{\ell \in L} b_k u_{\ell t}^{(4.7)} a_{k,\ell}^0 - \sum_{\ell \in L} u_{\ell t}^{(4.6)(4.7)} b_k \alpha_{\ell, t} \tag{4.48}$$

subject to Constraints (4.40)-(4.46)

We will prove that the model using Objective (4.48) is equivalent to the original pricing model of $\text{PP}_{kt}^{\text{SR}}$ using Objective (4.39). To do that, we need to prove the two following claims.

(i) The feasible regions of these models are equal. This claim is trivial because these problems have the same sets of constraints.

(ii) The optimal solutions of these models are equal. This claim will be proved if the two following statements are true:

(ii-a) $\overline{c}^{\star}_{\text{PP}^{\text{SR}}_{kt}} \geq \overline{cc}^{\star}_{\text{PP}^{\text{SR}}_{kt}}$;
(ii-b) $\exists(x', \alpha') \in \{(4.40) - (4.46)\} : \overline{cc}^{\star}_{\text{PP}^{\text{SR}}_{kt}} = \overline{c}_{\text{PP}^{\text{SR}}_{kt}}(x', \alpha')$.

To prove statement (ii-a), we must show that it is true for any valid assignment of the variables, that is:

$$\forall(x', \alpha') : \overline{c}_{\text{PP}^{\text{SR}}_{kt}}(x', \alpha') \geq \overline{cc}_{\text{PP}^{\text{SR}}_{kt}}(x', \alpha'). \tag{4.49}$$

Since $u^{(4.6)}_{\ell t} \leq 0$ and $u^{(4.7)}_{\ell t} \leq 0$, this statement holds if

$$\forall \ell \in L : -u^{(4.6)}_{\ell t} x'_{\ell t} - u^{(4.7)}_{\ell t} \alpha'_{\ell t} \geq -u^{(4.6)(4.7)}_{\ell t} \alpha'_{\ell t}$$

is true. It is equivalent to

$$\forall \ell \in L : u^{(4.6)}_{\ell t} x'_{\ell t} + u^{(4.7)}_{\ell t} \alpha'_{\ell t} \leq u^{(4.6)(4.7)}_{\ell t} \alpha'_{\ell t}$$

is true.

Indeed,

- If $\ell \in L$ is such that $a^0_{k,\ell} = 0$, we have by Constraints (4.42) that $x'_{\ell t} \geq \alpha'_{\ell t}$. Since $u^{(4.6)}_{\ell t} \leq 0$, we get $u^{(4.6)}_{\ell t} x'_{\ell t} + u^{(4.7)}_{\ell t} \alpha'_{\ell t} \leq u^{(4.6)}_{\ell t} \alpha'_{\ell t} + u^{(4.7)}_{\ell t} \alpha'_{\ell t}$, and Equation (4.47) sets $u^{(4.6)(4.7)}_{\ell t} = u^{(4.6)}_{\ell t} + u^{(4.7)}_{\ell t}$.

- If $\ell \in L$ is such that $a^0_{k,\ell} = 1$, we have by Constraints (4.43) that $x'_{\ell t} \geq \alpha'_{\ell t} - 1$. Since $x'_{\ell t} \in \{0, 1\}$ and $u^{(4.6)}_{\ell t} \leq 0$, we therefore have $u^{(4.6)}_{\ell t} x'_{\ell t} \leq 0$. Since Equation (4.47) sets $u^{(4.6)(4.7)}_{\ell t} = u^{(4.7)}_{\ell t}$, we can conclude that $u^{(4.6)}_{\ell t} x'_{\ell t} + u^{(4.7)}_{\ell t} \alpha'_{\ell t} \leq u^{(4.6)(4.7)}_{\ell t} \alpha'_{\ell t}$.

We now prove statement (ii-b). Let $x^\star$ and $\alpha^\star$ be the optimal solution of $\text{PP}^{\text{SR}}_{kt}$. We will show that with $\alpha' = \alpha^\star$, and x' such that

$$x'_{\ell t} = \begin{cases} 0 & \text{if } a^0_{k,\ell} = 1 \\ \alpha^\star_{\ell t} & \text{if } a^0_{k,\ell} = 0 \end{cases}, \tag{4.50}$$

we will have $\overline{cc}^{\star}_{\text{PP}^{\text{SR}}_{kt}} = \overline{cc}_{\text{PP}^{\text{SR}}_{kt}}(\alpha^\star, x^\star) = \overline{c}_{\text{PP}^{\text{SR}}_{kt}}(\alpha', x')$.

First, we need to show that α' and x' are a feasible solution of (4.40)-(4.46). Clearly, α' satisfies (4.40)-(4.41) because $\alpha' = \alpha^\star$. Furthermore, Constraints (4.42)-(4.43) are satisfied by α' and x' because they are hold in all cases of $\ell \in L$. Indeed, we have

- If $\ell \in L$ is such that $a^0_{k,\ell} = 0$, then we have $x'_{\ell t} = \alpha^\star_{\ell t} = \alpha'_{\ell t} \geq \alpha'_{\ell t} - 0$.
- If $\ell \in L$ is such that $a^0_{k,\ell} = 1$, then we have $x'_{\ell t} = 0 \geq \alpha'_{\ell t} - 1$ (as $\alpha'_{\ell t} \in \{0, 1\}$).

Note that we only consider the original pricing when $\overline{cc}^\star_{\mathrm{PP}^{\mathrm{SR}}_{kt}} \leq 0$. In that case, there has to exist one $\ell \in L : a^{\mathrm{INIT}}_{k,l} = 0$ such that $\alpha^\star_{lt} > 0$. Otherwise, $\alpha^\star_{\ell t} = a^{\mathrm{INIT}}_{k,\ell}$ for all $\ell \in L$ (the only feasible path in this case), then

$$\overline{cc}^\star_{\mathrm{PP}^{\mathrm{SR}}_{kt}} = -u^{(4.2)}_t \lambda^t - u^{(4.3)}_k > 0$$

$$\implies \overline{c}^\star_{\mathrm{PP}^{\mathrm{SR}}_{kt}} \geq \overline{cc}^\star_{\mathrm{PP}^{\mathrm{SR}}_{kt}} > 0,$$

this is contradict to assumption that $\overline{c}^\star_{\mathrm{PP}^{\mathrm{SR}}_{kt}} \leq 0$. Overall, it concludes that, in this case, Constraint (4.44) is satisfied by x'.

When $\overline{cc}^\star_{\mathrm{PP}^{\mathrm{SR}}_{kt}} > 0$, it implies $\overline{c}^\star_{\mathrm{PP}^{\mathrm{SR}}_{kt}} > 0$, so the pricing problem cannot generate improving configuration for the restricted master problem.

We now show that the objective values are equal. Note that $\overline{cc}_{\mathrm{PP}^{\mathrm{SR}}_{kt}}$ does not depend on $x^\star$, and so is computed with $\alpha^\star$ only. Now, we show that

$$\forall \ell \in L : u^{(4.6)(4.7)}_{\ell t} \alpha^\star_{\ell t} = u^{(4.6)}_{\ell t} x'_{\ell t} + u^{(4.7)}_{\ell t} \alpha'_{\ell t}. \tag{4.51}$$

This holds trivially by the combination of (4.47) and (4.50).

To this end, we observe that variables x can be removed from the simplified problem and it becomes the shortest path problem. □

Above proof can be described as follows. The optimal solution of the pricing problem can be a path that is the same with the current path or it has a link different from the current path's links. In the first case, it cannot give a negative optimal value, thus it is not interested. In the second case, x will be redundant, and removing x makes the pricing model the shortest path problem.

4.4.4 Formulation for Protection Scheme

Note that the pricing problem can define protection scheme related to single connection. Then this protection pricing problem can work with the master problem (4.1)-(4.9) as above algorithm. However, we keep the implementation of this protection feature for future works. Proposed protection pricing problem can be found at Section 4.7.

4.5 Numerical Results

For data sets and simulation description, they are taken completely from [46], and not repeated here.

4.5.1 Non-multiple Rerouting Algorithm

In Table 16, average solutions of different load factors, with $T = 20$ and $R^{\text{PAR}} = 20$,are reported.

First of all, we observe that the CPU times of non-multiple rerouting algorithm is about six times faster than algorithm in [46]. As we use much smaller number of rerouting events ($T = 20$ in this work, and $T = 50, 100, 150$ in [46]), symmetrical and tight possible configurations are reduced. In addition, non-multiple algorithm has lower gap (higher accuracy) than [46].

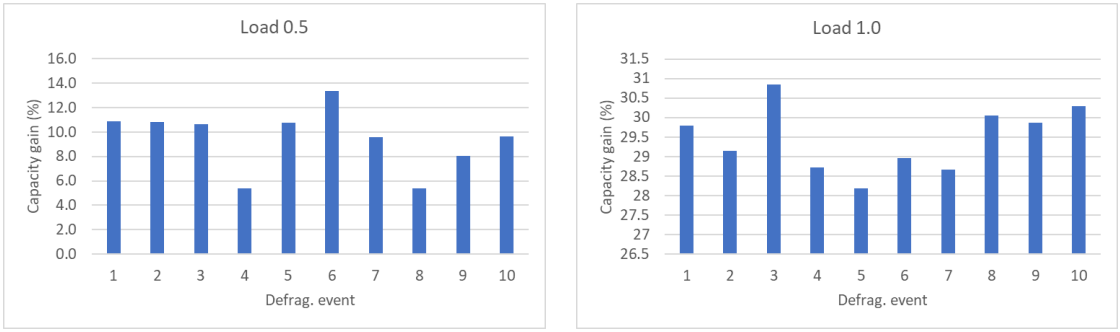
Table 16: Non-multiple Rerouting Algorithm's Solutions ($T = 20$, $R^{\text{PAR}} = 20$)

Load	# generated configs	# in-solution routings	gap (%)	CPU time (s)
0.5	3351.4	87.6	0.9	530.0
0.6	3424.9	192.2	0.7	415.1
0.7	4237.8	244.9	0.6	562.1
0.8	4254.6	256.4	0.7	561.3
0.9	4695.6	272.1	0.6	638.3
1	4756.4	283.9	0.5	693.3

With $T = 20$ and $R^{\text{PAR}} = 20$, the algorithm can reroute at most $T \times R^{\text{PAR}} = 400$ connections. Therefore the amounts of reduced capacity and bounds are better than ones obtained in [46] (with at most 150 rerouting operations). The reduced capacity percentages of the 0.5 and 1.0 load instances are reported in Figure 15.

4.5.2 Nested Column Generation Algorithm

Unfortunately, at the time of submission, the nested column generation algorithm is not working. We keep it as future works.



(a) Load factor = 0.5

(b) Load factor = 1.0

Figure 15: Reduction (%) of capacity requirement

4.6 Conclusions

We have proposed a new model for capacity defragmentation problem with respect to the previous model proposed by Duong et al. in [46]. We reach up to 400 reroutings in about 10 minutes, while the model of [46] was only scalable for 150 reroutings requiering about one hour. We plan to further study the proposed model so that it can handle the case of multiple reroutings per connection. In addition, we plan to study the adaptation of our optimization model to protection scheme.

4.7 Appendix

4.7.1 Protection Case: Decomposition Formulation of Connection Pricing Problem

A path configuration, $\pi \in \Pi_k^t$, is a path of the connection k after the rerouting event t which is characterized by:

- a_ℓ is 1 if k uses link ℓ after the rerouting event t

Variables:

- y_p^π is 1 if the path configuration π is selected as the primary path, 0 otherwise.
- y_b^π is 1 if the path configuration π is selected as the backup path, 0 otherwise.

$$\min \quad \overline{c}_{\mathrm{PP}_k} = -\sum_{t\in T} u_t^{(4.2)}\lambda^t - u_k^{(4.3)}\pi_k - \sum_{\ell\in L}\sum_{t\in T} u_{t,\ell}^{(4.6)} b_k \lambda^t x_\ell^t + \sum_{\ell\in L}\sum_{t\in T} b_k u_{\ell t}^{(4.7)}\, \delta_\ell^t \quad (4.52)$$

$$\text{s.t.:} \sum_{\pi\in\Pi_k^t} (y_p^\pi + y_b^\pi) = 2 * \lambda^t \qquad t \in T \quad (4.53)$$

$$\alpha_\ell^t = \sum_{\pi\in\Pi_k^t} a_\ell (y_p^\pi + y_b^\pi) \qquad t \in T \quad (4.54)$$

$$\sum_{t\in t} \lambda^t \le R^{\mathrm{M}} \quad (4.55)$$

$$\lambda^t \ge \alpha_\ell^t - \alpha_\ell^{t-1} \qquad t \in T, \ell \in L \quad (4.56)$$

$$x_\ell^t \ge \alpha_\ell^t - \alpha_\ell^{t-1} \qquad t \in T, \ell \in L \quad (4.57)$$

$$\delta_\ell^t = \alpha_\ell^t - \alpha_\ell^{t-1} \qquad t \in T, \ell \in L \quad (4.58)$$

$$\alpha_\ell^t \in \{0,1\} \qquad t \in T, \ell \in L \quad (4.59)$$

$$\lambda^t \in \{0,1\} \qquad t \in T. \quad (4.60)$$

$$x_\ell^t \in \{0,1\} \qquad t \in T, \ell \in L \quad (4.61)$$

$$\delta_\ell^t \in \{-1,0,1\} \qquad t \in T, \ell \in L \quad (4.62)$$

Pricing Problem PP_k^t

$$\min \quad \overline{c}_{\mathrm{PP}_k^t} \quad = \quad (4.63)$$

$$\text{s.t.:} \sum_{\ell\in\omega^-(v)} a_\ell - \sum_{\ell\in\omega^+(v)} a_\ell = \begin{cases} -1 & \text{if } v = s_k \\ 1 & \text{if } v = d_k \\ 0 & \text{otherwise} \end{cases}$$

$$t \in T, v \in V \quad (4.64)$$

$$\sum_{\ell\in\omega^+(v)} a_\ell \le 1 \qquad t \in T, v \in V \quad (4.65)$$

$$a_\ell \in \{0,1\} \qquad t \in T, \ell \in L \quad (4.66)$$

4.7.2 Revised Protection Case: Decomposition Formulation of Connection Pricing Problem

A pair of path configuration, $\pi, \pi' \in \Pi_k^t$, are two link disjoint paths for connection k after the rerouting event t which is characterized by:

- a_ℓ is 1 if k uses link ℓ after the rerouting event t (primary path)
- a'_ℓ is 1 if k uses link ℓ after the rerouting event t (backup path)

Variables:

- q_p^π is 1 if path configuration π is selected (primary & backup) path, 0 otherwise.

$$\begin{aligned}\min \quad \overline{c}_{\mathrm{PP}_k} = &-\sum_{t\in T} u_t^{(4.2)}\lambda^t - u_k^{(4.3)} \\ &- \sum_{\ell\in L}\sum_{t\in T} u_{t,\ell}^{(4.6)} b_k \lambda_t x_{\ell t} \\ &+ \sum_{\ell\in L}\sum_{t\in T} b_k u_{\ell t}^{(4.7)} \left(\alpha_\ell^t - \alpha_\ell^{t-1}\right) \quad (4.67)\end{aligned}$$

Quadratic term $\lambda_t x_{\ell t}$ in (4.52) can be replaced by $x_{\ell t}$ as ....

subject to:

$$\sum_{\pi\in\Pi_k^t} q^\pi \le \lambda^t \qquad t \in T \quad (4.68)$$

$$\alpha_\ell^t = \sum_{\pi\in\Pi_k^t} a_\ell^\pi q_\pi \qquad t \in T, \ell \in L \quad (4.69)$$

$$\sum_{t\in T} \lambda^t \le R^{\mathrm{M}} \quad (4.70)$$

$$\lambda^t \ge \alpha_\ell^t - \alpha_\ell^{t-1} \qquad t \in T, \ell \in L \quad (4.71)$$

$$x_\ell^t \ge \alpha_\ell^t - \alpha_\ell^{t-1} \qquad t \in T, \ell \in L \quad (4.72)$$

$$q^\pi \in \{0,1\} \qquad \pi \in \Pi_k^t, t \in T \quad (4.73)$$

$$\alpha_\ell^t \in \{0,1\} \qquad t \in T, \ell \in L \quad (4.74)$$

$$\lambda^t \in \{0,1\} \qquad t \in T \quad (4.75)$$

$$x_\ell^t \in \{0,1\} \qquad t \in T, \ell \in L. \quad (4.76)$$

Pricing Problem $\mathbf{PP}_k^t$

$$\min \quad \overline{c}_{\mathrm{PP}_k^t} = -u_t^{(4.68)} - \sum_{\ell \in L} u_t^{(4.69)} a_\ell \quad (4.77)$$

subject to:

$$\sum_{\ell \in \omega^-(v)} a_\ell - \sum_{\ell \in \omega^+(v)} a_\ell = \begin{cases} -2 & \text{if } v = s_k \\ 2 & \text{if } v = d_k \\ 0 & \text{otherwise} \end{cases}$$

$$t \in T, v \in V \quad (4.78)$$

$$a_\ell \in \{0,1\} \qquad t \in T, \ell \in L \quad (4.79)$$

Chapter 5

MBB Defragmentation with CD and CDC ROADMs in Elastic Optical Networks

CDC ROADM is going to be deployed widely in EONs as it removes the contentionless issue of CD ROADM. However, there are limited studies for this expensive device to prove it is significant. In addition, this evaluation has to take into account the defragmentation process because it is an important application of ROADM. In this paper, we first propose a novel defragmentation strategy in which spectrum usage minimizing and spectrum squeezing are combined. We also propose a column generation algorithm to significantly scale up exact solution in literature, to a network of 10 nodes, 32 links, 50 slots per link, and 250 connections. Experiments show that our novel strategy is better than the studied ones. It also asserts that CDC ROAM brings more advantages to the network.

This paper is in preparation.

5.1 Introduction

Telecommunication networks are constantly evolving in terms of quality, diversity and reliability. Optical Networks (ONs) exhibit its capability with connections requiring on-demand service, flexible-width bandwidth and reliability guarantee. Reconfigurable Optical Add Drop Multiplexer (ROADM) and Elastic Optical Networks

(EONs) are two remarkable developments for the next optical network generation. ROADM is a well-known device to automatically and remotely configure network provision by which it significantly accelerates the configuration time, while EONs introduce finer granularity to improve spectrum utilization.

EONs are focused to improve networks' capacity without bringing more physical infrastructure to the current systems. In the fixed grid Wavelength Division Multiplexing (WDM) system, connection occupies exactly one same wavelength for all links on its lightpath. Thus, if a connection is transferring less than the total wavelength capacity, the unused spectrum range is still unavailable to other connection requests on this path. Recently, vendors introduce new technology, so-called EONs, by which spectrum is used more efficiently. In EONs, the spectrum is divided into a finer grid compared to the fixed grid, i.e., each spectrum slot is 12.5 GHz or even 6.25 GHz [51]. Further, an operator can combine several consecutive spectrum slots to form a channel. Thus, a connection can be placed at any position and require diversity data rates. In addition, EONs are enhanced by the Orthogonal Frequency Division Multiplexing (OFDM) technique which helps bandwidth utilization more efficient [52, 53].

On the other hand, to introduce flexibility in connection routing and rerouting, ROADM is deployed and being improved. The main advantage of ROADM over Optical Add Drop Multiplexer is that it can be remotely configured by which network operation is significantly more flexible and accelerated. However, ROADM characteristics are improved in different generations. There are three main properties to define a ROADM generation which are colorless, directionless and contentionless (CDC) [54]. The latest ROADM possesses all these features and it is the so-called CDC ROADM. However most deployed ROADM networks are still colorless and directionless ROADM (CD ROADM) architecture.

In practice, a CDC ROADM node costs more than a CD ROADM node. Therefore, providers are in need of the evaluation of CDC ROADM performance to be confident in this deployment. Although CDC ROADM avoids contention blocking at an add/drop block consisting of many ports, several studies showed that in 50GHz fixed grid Optical Networks, the contention blocking only occurs when the network is really loaded [55]. Also, in the fixed 50GHz grid technology, the contention issue is less severe because of the fixed spectrum for all datarates architecture. As soon as

the number of local add/drop blocks of a node equal to its degree, there is no contention at that node. While in EONs, the contention issue is more complicated due to the various ranges of connection spectrum requirements. There is no such evaluation for CDC ROADM in EONs in literature. Furthermore, this evaluation must be accompanied by a defragmentation process because it is applied in a practical system.

The most frequently cited application for ROADMs is to enable optical layer remotely routing and defragmentation [7]. Thus the purpose of this work is to investigate the defragmentation performance of CD and CDC ROADMs. In practice, the defragmentation process must be planned and take place carefully, so the time-drive defragmentation manner is applied in this paper. Defragmentation is also desired to be non-disruptive, thus make-before-break (MBB) paradigm is the choice of our algorithms. MBB paradigm ensures that the new path of a connection is established and its data is gradually moved to the new path before the old path is torn down.

In literature, defragmentation attempts to reduce the required bandwidth or push down connections altitude. To our best of knowledge, there is no proposed work in literature taking into account these objectives at the same time. Thus we propose in this work novel heuristic algorithms which are minimizing both spectrum usage and spectrum altitude. For minimizing spectrum usage, we also present a scalable optimal solution to estimate proposed heuristics' performances. In addition, this optimal solution also enlarge twice data instance sizes of the state of the art in [56].

To perform simulations, we use a well-known United States topology with two network configurations: full CDC ROADM nodes and full CD ROADM nodes. For the latter, all the nodes are the same CD ROADM with 2 (3, or 4 respectively) ADD/DROP local blocks per node. Connection requests are generated in a random manner and routed by the first-fit routing scheme. In the first scenario, the network will not go through any defragmentation. While in the second scenario, it is defragmented in a time-driven manner. In this work, we show that a full CDC ROADM system reduces bandwidth blocking rate (in the best case) 2.5% better than a full CD ROADM one, with various defragmentation algorithms.

In contrast with fixed 50GHz grid technology, EONs enable various connection data rates (granularities), e.g., 100Gbps, 200Gbps, 400Gbps. In addition, the OFDM technique interprets these data rates into various ranges of required spectrum slots

per connection. To the best of our knowledge, the relation between connection granularities and defragmentation is not studied so far in the literature. Thus this work also proposes a simulation to evaluate the impact of connection granularities on different defragmentation strategies.

To summarize, in this paper, we want to answer the following questions:

- What are the performances of our novel proactive defragmentation strategies?
- Which CD ROADM configuration is similar to CDC ROADM?
- What are the behaviours of connection granularities with different defragmentation strategies?

The contributions of this paper are:

- Novel defragmentation strategies that minimizing both spectrum usage and spectrum altitude at the same time.
- An exact make-before-break EON defragmentation algorithm based on decomposition modelling and column generation algorithm. With this algorithm, we obtained optimal solutions for instances of 10 nodes, 32 links, 50 slots per links, 50 rerouting events limitation and about 250 connections. This is notably larger than the model proposed in [56].

This paper is organized as follows. Section 5.2 briefly presents literature of ROADM in EONs and defragmentation algorithms in EONs. We next describe in Section 5.3 the defragmentation problem statement. In Section 5.3, defragmentation problem is represented by an exact decomposition mathematical model. An column generation algorithm and three heuristic solutions are proposed in Section 5.4. In Section 5.5, numerical results are reported. Conclusions are drawn in the last section.

5.2 Related Works

As this work related to two distinct aspects of EONs that are ROADM architecture and defragmentation process, these aspects are reviewed one by one.

5.2.1 CD/CDC ROADM Architectures and Contention Issue

In this section, high-level architectures of currently developed CD ROADM and CDC ROADM are presented to explain different levels of contention. Then, several studies evaluating contention from the ROADM perspective are reviewed.

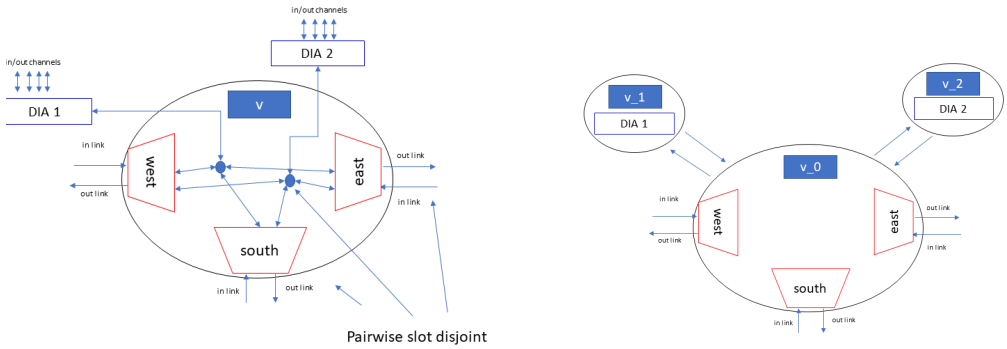
Several architectures for CD ROADM are presented in [57, 58]. As mentioned in these works, broadcast-and-select architecture is considered as the simplest and most affordable implementation. In practice, vendors utilize this architecture with their broadcasting/selecting designs so that a high number of transponders can be attached to each add/drop block. In our study, an isolated add/drop block is called a Direct Independent Access (DIA). The physical layout of a CD ROADM node is illustrated in Figure 16(a).

A DIA can implement as many add/drop ports as needed. As add/drop transponders of DIA are tunable, its colorless property is trivial. Besides, a DIA is connected to all directions, directionless is achieved. However, all added (dropped) connections associated with a DIA go through the same line before being filtered at ports, thus routed/rerouted connections sharing the same DIA block have to be non-overlapping regarding to spectrum. This restriction introduces some obstacles when routing/rerouting connections. This is also known as the ROADM contention issue.

To ease or reduce contention issue, more DIA blocks can be added. For examples, in Figure 16(a), there are 2 DIA blocks. So at most 2 added/dropped connections at this node can overlap their spectrum if they are not associated to the same DIA block.

Since DIA blocks work independent mutually, overlapping connections can use different DIA blocks. This workaround only alleviates the issue since the needed DIA blocks cannot be predicted. Also, it is preferred to have all DIA blocks filled evenly to reduce costs. Deploying additional DIA blocks with few active transponders harms profit in both short term and long term. To enable more overlapping connections, vendors introduced multicast devices that can take overlapping connections at different input ports, then guide them to output ports. Although these devices completely eliminate contention, their costs are remarkably higher than a CD ROADM node.

Evaluation and solution of the fixed grid ROADM contention were presented in several studies. In [59], the authors showed that the contention issue is negligible if less than 75% of local CD ROADM ports are used. While in [60], when a wavelength



(a) Physical CD ROADM (private communication from Ciena)

(b) Mathematical modelling of CD ROADM

Figure 16: CD ROADM Physical and CDC-Modelling Layout

can be shared by at most 2 connections, a full CD ROADM network performs as contentionless. To the best of our knowledge, there is no similar evaluation of EONs in open literature.

5.2.2 Defragmentation of EONs

In EONs, defragmentation is performed by different techniques and strategies. In consequence, they offer various conditions and efficiency. However, in this section, we only provide references relevant to our work and point out the novel aspects of our proposed strategy.

The defragmentation process is usually triggered when the network reaches some predefined thresholds, e.g., a request is blocked, it is called reactive defragmentation. Network reconfiguration can be also started in a fixed schedule, e.g., every midnight, this is called proactive defragmentation. With reactive defragmentation, it is hard to notify customers when the network is modified, thus it is not preferred in practice. On the other hand, proactive defragmentation allows more control of both customers and providers. In this section, works involving only reactive defragmentation are not discussed.

In terms of service interruption, they can be classified into hitless and non-hitless

defragmentation [7]. Since non-hitless techniques, e.g. break-before-make or re-planning, interrupt significantly connections, this work does not use it, and its literature is omitted. For hitless defragmentation, there are three most-used techniques, that are push-pull (PP), hop tuning (HT) and make-before-break (MBB). The first two techniques require special devices that can shift the connection spectrum along the link spectrum. On the other hand, MBB establishes the new route of a connection concurrently with its current route. MBB is simple and affordable, thus it is used by all our proposed algorithms.

In terms of spectrum, there are three prominent objectives: i) minimizing total spectrum (slots) usage, ii) minimizing the highest used spectrum index overall network and iii) minimizing the average altitude of spectrum usage. The objective i) determines the least required resources, while the objectives ii) and iii) corresponds to spectrum squeezing. In most studied works, the spectrum squeezing objective is used as it directly improves the continuity and contiguity of the spectrum. On the other hand, to the best of our knowledge, there is no proposed strategy and algorithm for spectrum-usage-minimizing defragmentation. Furthermore, these two objectives can be considered at once. Therefore, in our paper, we propose several spectrum-usage-minimizing and/or spectrum-altitude-minimizing algorithms and evaluate their performances.

For spectrum squeezing idea, In [56, 61], the authors evaluated spectral gain in EONs by push-pull, hop-tuning and re-planning (without MBB) strategies. Note that the last technique finds the first available slot block for rerouting connection, its new path may require higher the number of total required slots. It showed that the physical limitations of the push-pull technique obstruct its ability to effectively reconfigure demands, whereas hop tuning was generally much closer to the full re-planning benchmark which is the best strategy. In [62], the authors proposed a proactive defragmentation algorithm similar to a re-planning strategy where instead it squeezing connections to both low or high ends of the link spectrum (2-end re-planning). However, [62] implemented a simulation where a 2-end re-planning strategy is operated with the reactive defragmentation, thus the actual performance of this proactive strategy was not evaluated. When only proactive defragmentation is performed, experiments of [62] showed that hop-tunning strategy is the best proactive option in terms of blocking rate.

Several integer linear programming systems have been proposed for the push-pull, hop tuning and rerouting strategies in [7, 56]. To our best knowledge, there is no interge linear programming (ILP) model for the MBB in EONs yet. However, it is similar to re-planing models in [56]. Also, we can leverage the ILP models for Logical Layer [47, 46] as the initial ideas to develop MBB models of Optical Layer.

5.3 Defragmentation Problem in EONs

In this section, the defragmentation problem of EONs is described. Although the following description assuming that all nodes in the network are CDC ROADM, a mixed-ROADM network can also use following formulations by modelling a CD ROADM as a set of CDC ROADMs. In other words, if a node is not CDC ROADM, it is replaced by several artificial CDC ROADM nodes. The representation of CD ROADM node v in CDC-ROADM replacement is shown in Figure 16(b). With this modelling, the input network is considered as a full CDC ROADM configuration. This simplification allows us present mathematical models and algorithms for only CDC ROADM so that fewer complicated constraints and variables are introduced.

We first introduce notations used throughout this paper, then the formal mathematical model (master problem) of the defragmentation problem in EONs are presented in a decomposition method. Due to decomposition method, the pricing problem formulations are also presented to be used by column generation algorithm proposed in the next section.

5.3.1 Notations

We are given a physical network represented by a directed multi-graph $G = (V, L)$. Each link ℓ in L is associated with a directional fiber from ℓ_s to ℓ_d, with geometrical distance ℓ_Δ, and its capacity is defined by a set of frequency slots indexed by set $\{0, .., \ell_F - 1\}$ (around 400 assuming each frequency slot being 12.5 Ghz wide, and 1,000 assuming each frequency slot being 6.25 Ghz wide). Let $F^{\max} = \max_{\ell \in L} \ell_F$. Nodes are associated with ROADMs, and indexed by v. $\omega^-(v)$ and $\omega^+(v)$ are set of outgoing and incoming links respectively of node $v \in V$.

Due to the improvement of the OFDM technology, each data rate can be transferred with different bandwidth sizes depending on geometrical distance of routes.

In other words, the same data rate requires fewer frequency slots on a shorter path (geometrical distance). It is also called modulations where each modulation specifies for each data rate: the number of required frequency slots and the range of distance within that it can transfer. A practical specification of modulations is introduced in Table 17. Let M be the set of modulations. Each modulation $m \in M$ has:

- $\overline{\delta}(m)$ and $\underline{\delta}(m)$ as upper and lower reachable distances.
- $\lambda(m, r)$ is the number of required slots of this modulation for data rate r.

We are also given a set K of provisioned connections (set of connections). Each connection k is characterized by a data rate k_r from source k_s to destination k_d and $k_L \subseteq L$ is the link set of it path with modulation k_m. It also occupies $\lambda(k_m, k_r)$ frequency slots from slot index $k_{\underline{f}}$ to $k_{\overline{f}}$ for all $\ell \in k_L$. Let $k_\Delta = \sum_{\ell \in k_L} \ell_\Delta$ the geometrical distance of the connection k.

Assume that connections are arriving one at a time and the network is reconfigured at some pre-determined points in time manner (proactive defragmentation) during the whole process. At a defragmentation event, a network operator needs to find a sequence of MBB reroutings of current connections to free up the largest possible amount of spectrum slots in the network.

A defragmentation event is divided into rerouting events, each rerouting event is long enough to perform one rerouting operation by which one can reroute at most one lightpath. In addition, each rerouting operation has to be carried on under the MBB manner. Let $T = \{1, 2, ..., |T|\}$ be the index set of rerouting events of a defragmentation event. Assuming that network operator does not allow parallel rerouting operations at one rerouting event because it is easier monitor defragmentation process. Thus, it means that at most $|T|$ connections are rerouted.

For a given provisioned network, defragmenting the network at a defragmentation event is to minimize, i) the total required slots overall the network, and ii) the sum of connection starting slot indices. Since this is a multiple objective problem, the following mathematical model is explicitly and only dealing with the objective i). However, the objective ii) is still considered by heuristic algorithms.

5.3.2 Master Model

Let P_k^t be a set of all alternative paths to reroute a connection k at rerouting event t. Each configuration is characterized by a demand k, to be rerouted at rerouting event t, using modulation m, on the new lightpath p. For each $p \in P_k^t$, it represented by:

- $\alpha_{\ell,f}^p$ is 1 if p uses slot f on link ℓ after rerouting event t.

Note that, for a connection k, the MBB condition requires that a alternative path have to link disjoint to the original path, otherwise, their spectrum ranges have to be non-overlapping.

Parameters

- $a_{k,\ell,f}^0$ is 1 if initial connection $k \in K$ uses slot f on link ℓ in its route, 0 otherwise.

Variables

- $y_{\ell,f}^t$: binary variable that is set to 1 if slot f on link ℓ is used after reconfiguration event t by any connection.

- $z_{k,p}^t$: binary variable that is set to 1 if the solution selects configuration corresponding to path $p \in P_k^t$ for demand k at rerouting event t, using modulation m.

Objective

$$\text{Minimize} \qquad \sum_{\ell \in L} \sum_{f \in F} y_{\ell,f}^{|T|} \tag{5.1}$$

subject to:

$$\sum_{k\in K}\sum_{p\in P_k^t} z_{k,p}^t \leq 1 \qquad t \in T \tag{5.2}$$

$$\sum_{t\in T}\sum_{p\in P_k^t} z_{k,p}^t \leq 1 \qquad k \in K \tag{5.3}$$

$$y_{\ell,f}^0 = \sum_{k\in K} a_{k,\ell,f}^0 \qquad \ell \in L, f \in \ell_F \tag{5.4}$$

$$y_{\ell,f}^t \geq y_{\ell,f}^{t-1} - \sum_{k\in K}\sum_{p\in P_k^t} (a_{k,\ell,f}^0 - \alpha_{\ell,f}^p) z_{k,p}^t \qquad t \in T, \ell \in L, f \in \ell_F \tag{5.5}$$

$$\sum_{k\in K}\sum_{p\in P_k^{t+1}} z_{k,p}^{t+1} \leq \sum_{k\in K}\sum_{p\in P_k^t} z_{k,p}^t \qquad t \in T \tag{5.6}$$

$$y_{\ell,f}^t \in \{0,1\} \qquad t \in T, \ell \in L, f \in \ell_F \tag{5.7}$$

$$z_{k,p}^t \in \{0,1\} \qquad t \in T, k \in K, p \in P_k^t \tag{5.8}$$

Constraints (5.2) restrict at most one rerouting operation within a rerouting event. Each demand is reconfigured no more than once by constraints (5.3). During any reconfiguration, capacity feasibility is ensured by constraints (5.5). In order to eliminate symmetrical solutions, constraints (5.6) enforce reconfiguration will be performed in consecutive rerouting event. The last set of constraints (5.7) and (5.8) are variables' domain.

Path Modulation Pricing Problem

At first, we considered path pricing problem which would outputs a path together with a proper modulation. Such pricing problem is complex to present, as well as difficult to solve, as we need to introduce decision variables for the selection of the modulations. Fortunately, there is only one modulation is selected by path pricing problem, thus we can simplify this problem by decomposing it into "elementary" path modulation pricing problems. We therefore next present path modulation pricing problems that each outputs a path for a given modulation.

Parameters:

- k: connection related to this pricing.
- t: rerouting event.

- m: modulation.

Variables:

- α_f^ℓ is a binary variables that is set to 1 if slot f on link ℓ is used for this configuration, 0 otherwise,
- β_ℓ is a binary variable that is set to 1 if link ℓ has slots to be used in the configuration, 0 otherwise,
- x_f is a binary variable that is set to 1 if slot indexed f is used as the starting slot of this demand.

Objective:

$$\text{Minimize} \quad \overline{c}_{\text{PP}_k^{t,m}} = -u_t^{(5.2)} - u_k^{(5.3)} - u_t^{(5.6)} + u_{t-1}^{(5.6)} - \sum_{\ell \in L} \sum_{f \in F^\ell} u_{\ell,t,f}^{(5.5)} \left(a_{k,\ell,f}^0 - \alpha_f^\ell\right) \quad (5.9)$$

subject to:

$$\sum_{f \in F^{\max}} x_f = 1 \tag{5.10}$$

$$\beta_\ell + \sum_{f'=\max(0,f-\lambda(m,k_r)+1)}^{\min(f,\ell_F-\lambda(m,k_r))} x_{f'} \leq \alpha_\ell^f + 1 \qquad \ell \in L, f \in F^\ell \tag{5.11}$$

$$\alpha_\ell^f \leq \beta_\ell \qquad \ell \in L, f \in F^\ell \tag{5.12}$$

$$\alpha_\ell^f \leq \sum_{f'=\max(0,f-\lambda(m,k_r)+1)}^{\min(f,|F_\ell|-\lambda(m,k_r))} x_{f'} \qquad \ell \in L, f \in F^\ell \tag{5.13}$$

$$\sum_{f \in F^\ell} \alpha_f^\ell = \lambda(m,k_r)\beta_\ell \qquad \ell \in L \tag{5.14}$$

$$\sum_{\ell \in \omega^-(v)} \beta_\ell - \sum_{\ell \in \omega^+(v)} \beta_\ell = \begin{cases} -1 & \text{if } v = s_k \\ 1 & \text{if } v = d_k \\ 0 & \text{otherwise} \end{cases} \qquad v \in V \tag{5.15}$$

$$\sum_{\ell \in \omega^+(v)} \beta_\ell \leq 1 \qquad v \in V \tag{5.16}$$

$$\underline{\delta}(m) < \sum_{\ell \in L} \ell_\Delta \beta_\ell \leq \overline{\delta}(m) \tag{5.17}$$

$$\alpha_f^\ell \in \{0,1\} \qquad \ell \in L, f \in F \tag{5.18}$$

$$\beta_\ell \in \{0,1\} \qquad \ell \in L \tag{5.19}$$

$$x_f \in \{0,1\} \qquad f \in F. \tag{5.20}$$

Constraints (5.10) ensure that demand k has exactly one slot as its starting slot. Constraints (5.11), (5.12) and (5.13) make sure a slot in a link is selected if the link is chose and the slot is in the range from the selected starting slot. Constraints (5.14) guarantee that the set of selected slots is matched with the bandwidth requirement according to the modulation. These first three sets of constraints collectively guarantee the *contiguity* of selected slots.

Constraints (5.15) ensure the flow conditions to build a path from source to destination of the demand. Constraints (5.16) help avoid any node has a loop to itself. Since the path defined by these constraints is independent from the slot index, the set of slots is *continuous* on the path.

The restriction on the distance of selected modulation is guaranteed by Constraints

(5.17). The rest of constraint sets define variable domains.

Frequency Pricing Problem

Notice that for each pricing problem, only one starting slot is chosen. Thus the pricing problem can be solved by less than $F^{\max}$ simplified models, i.e., (5.9) and (5.11)-(5.20), with an unique x_f is fixed to 1. Therefore,

$$\overline{c}_{\mathrm{PP}_k^{t,m}} = \min_{f=0}^{F^{\max}-\lambda(m,k_r)} \left\{ \overline{c}_{\mathrm{PP}_{k,f}^{t,m}} \right\} \tag{5.21}$$

where,

$$\overline{c}_{\mathrm{PP}_{k,f}^{t,m}} = \min \Big\{ -u_t^{(5.2)} - u_k^{(5.3)} - u_t^{(5.6)} + u_{t-1}^{(5.6)} - \sum_{\ell \in L} \sum_{f \in F^\ell} u_{\ell,t,f}^{(5.5)} \, a^0_{k,\ell,f} + \sum_{\ell \in L} \beta_\ell \sum_{f'=f}^{f+\lambda(m,k_r)-1} u_{\ell,t,f}^{(5.5)} \Big\} \tag{5.22}$$

subject to

$$\sum_{\ell \in \omega^-(v)} \beta_\ell - \sum_{\ell \in \omega^+(v)} \beta_\ell = \begin{cases} -1 & \text{if } v = s_k \\ 1 & \text{if } v = d_k \\ 0 & \text{otherwise} \end{cases} \qquad v \in V \tag{5.23}$$

$$\sum_{\ell \in \omega^+(v)} \beta_\ell \leq 1 \qquad v \in V \tag{5.24}$$

$$\sum_{\ell \in L} \ell_\Delta \beta_\ell \leq \overline{\delta}(m) \tag{5.25}$$

$$\beta_\ell \in \{0,1\} \qquad \ell \in L. \tag{5.26}$$

If a starting slot is given to the model and constraints (5.25) are eliminated, the remaining problem is the shortest path problem on a positive weighted digraph. Therefore, the sub pricing problems are first solved by Dijkstra algorithm, if any of them satisfies the modulation distance limit constraints (5.25) and has a negative reduced cost, then it is added to the RMP as a new column. If there is no such a solution, the pricing problem is solved by the shortest path with resource constraint algorithm.

5.4 Spectrum Offender Algorithms

The general idea for all proposed heuristic solutions is to assign a priority value to each connection. Then connections are reconfigured to its new path according to these orders. Before going into the flows of the algorithms, we first define the important values which are used to prior connections during the defragmentation process.

5.4.1 Spectrum Offender Value Heuristics

Assume that a set of connections K is currently provisioned by network $G = (V, L)$, $G(K)$ represents the state of network G when provisioning K. Let $P_k(G(K'))$ be the set of all possible path to provision an connection request k when network G is provisioning a connection set K'. Then, the current number of used frequency slots for a connection k is $\lambda(k_m, k_r)|k_L|$. Let $p^\star$ is the best available path for connection $k \in K$ in current state, namely:

$$p^\star = \arg \min_{p \in P_k(G(K))} \{\lambda(p_m, k_r)|p_L| : \underline{\delta}(p_m) < p_\Delta \leq \overline{\delta}(p_m)\}. \tag{5.27}$$

Let $\underline{p}$ is the best available path for connection k in an empty network:

$$\underline{p} = \arg \min_{p \in P_k(G(\emptyset))} \{\lambda(p_m, k_r)|p_L| : \underline{\delta}(p_m) < p_\Delta \leq \overline{\delta}(p_m)\}. \tag{5.28}$$

In other words, $\lambda(p^\star_m, k_r)|p^\star_L|$ is the minimum number of frequency slots at the current state to to reprovision this connection, while $\lambda(\underline{p}_m, k_r)|\underline{p}_L|$ is the least frequency slots in an empty network.

At a given state of the network $G(K)$, the *immediate spectrum offender* value of the connection k is

$$\textsc{ImmediateSpectrumOffender}(\textsc{k}) = \lambda(k_m, k_r)|k_L| - \lambda(p^\star_m, k_r)|p^\star_L|, \tag{5.29}$$

where $p^\star$ is defined by 5.27, and the *complete spectrum offender* value of the connection k is

$$\textsc{CompleteSpectrumOffender}(\textsc{k}) = \lambda(k_m, k_r)|k_L| - \lambda(\underline{p}_m, k_r)|\underline{p}_L|, \tag{5.30}$$

where $\underline{p}$ is defined by 5.28.

To compute the immediate spectrum offender value of a connection, for a given modulation and starting slot, one must find the least hop paths from the source to the

destination of the connection. Algorithm 6 finds the least hop path according to each possible combination of a modulation and a starting slot of the connection. The best path is the one having the least number of total used slots. Note that, a connection is allowed to use a new modulation only when the new modulation's distance limit is better than the current one, i.e., shorter geometrical distance (it also implies that the number of slots per link is less).

Given an starting slot $\underline{f}$ and a modulation m, the least hop path problem for connection request k has a constraint on the path geometrical distance limit of $\delta(m)$, and it is solved by the Algorithm 7. This algorithm is a dynamic programming algorithm. For a connection request k, let variables $F[h][v]$ be the shortest geometrical distance among paths with h links (hops) from k_s to v. Assume that the values $F[1][u], F[2][u], .., F[h][v]$ were computed for all $v \in V$, $F[h+1]$ values can be computed by $F[h]$ values. Let $G(V, L, K \setminus k, \underline{f}, \overline{f})$ is the provisioning network whose links which have contiguous available slots from slot $\underline{f}$ to $\overline{f}$. Then, it is easy to figure out that:

$$F[v][h+1] = \min_{\ell \in G(E,V,K,\underline{f},\overline{f})} \{F[u][h] + \ell_\Delta : \ell_s = u \textbf{ and } \ell_d = v\}. \quad (5.31)$$

Therefore, when F is computed according to the increasing order of h, the first $F[k_d][h] < \overline{\delta}(m)$ is the least hops path. Note that this algorithm also offers the path whose the lowest starting slot, among the least slot paths.

The complexity of Algorithm 7 is $O(|V| \times |L|)$, then Algorithm 6 takes $O(4 \times F^{\max} \times |V| \times |L|)$ (assuming that there are four available modulations).

In following algorithms, `ConnectionMoveLimit` $= 1$ and `RoundLimit` is unlimited. We introduce these parameters in case we want to evaluate some variation of algorithms in future. For example, if a connection can reroute twice in a defragmentation event, then `ConnectionMoveLimit` $= 2$. If we want to reduce CPU time, `RoundLimit` can be set as a fixed number.

5.4.2 Immediate-Spectrum-Offender-Sorting Heuristic (ISO)

The ISO algorithm is presented in Algorithm 8. The idea is to reroute connections with larger immediate spectrum offender value to free more resource at the beginning

Algorithm 6 Immediate Spectrum Offender Computation

function ImmediateSpectrumOffender($G(V, L), K, k$)
  $m \leftarrow$ 16QAM
  $p^\star \leftarrow k$
  while $\overline{\delta}(m) \leq \overline{\delta}(k_m)$ **do**
    for $\underline{f} \leftarrow 0$ **to** $F^{\max} - \lambda(m, k_r)$ **do**
      $p \leftarrow$ LeastHopPath($G(V, L), K, k, m, \underline{f}$)
      if $p \neq null$ **and** $\lambda(m, p^\star_m)|p^\star_L| > \lambda(m, p_m)|p_L|$ **then**
        $p^\star \leftarrow p$
    $m \leftarrow$ next modulation
  return $(\lambda(m, p^\star_m)|p^\star_L| - \lambda(m, k_m)|k_L|, p^\star)$

of the defragmentation process. Thus, first, all the immediate spectrum offender values of the connections are computed. Then, the connections are sorted in the decreasing order by their offender values.

5.4.3 Complete Spectrum Offender Sort Heuristic (CSO)

This algorithm is presented in Algorithm 9. Firstly, all complete spectrum offender values of the connections are computed. Then, the connections are sorted in the decreasing order by their offender values. As the complete offender value is final, the sorting order is not recomputed after a round.

5.4.4 Spectrum Usage Sort Heuristic (SPU)

This algorithm is presented in Algorithm 10. The connections are sorted in the decreasing order of their spectrum usage values. As the spectrum usages are changed, the connections are reordered after every round.

5.4.5 Heuristics with PP and HT

Note that proposed so far do not exploit the spectrum squeezing method. There are two common ideas to squeeze down spectrum which are HT and PP strategies. The PP technique only allows continuous-spectrum to move while HT can assign a

Algorithm 7 Least Hop Path Algorithm

```
1: function LeastHopPath((G(V, L), K, k, m, f̲)
2:     f̄ ← f̲ + λ(m, k_r)
3:     path ← ∅
4:     for all u ∈ V and h = 0 → |V| − 1 do
5:         F[u][h] = ∞
6:         prev_link[u][h] ← null
7:     F[k_s][0] ← 0
8:     for h ← 1 to |V| − 1 do
9:         for ℓ ∈ G(V, L, K \ k, f̲, f̄) do
10:            if F[l_s][h − 1] ≤ δ̄(m) then
11:                if F[ℓ_d][h] > F[l_s][h − 1] + ℓ_Δ then
12:                    F[ℓ_d][h] ← F[ℓ_s][h − 1] + ℓ_Δ
13:                    prev_link[ℓ_d][h] ← ℓ
14:        if F[k_d][h] ≤ δ̄(m) then
15:            break
16:    if F[d][h] ≤ δ̄(m) then
17:        v ← k_d
18:        while v ≠ k_s do
19:            ℓ ← prev_link[v][h]
20:            path ← path ∪ ℓ
21:            v ← ℓ_s
22:            h ← h − 1
23:    return path
```

Algorithm 8 Immediate-Spectrum-Offender-Sorting Heuristic (ISO)

for $k \in K$ **do**
    $\texttt{\#Moves}(k) \leftarrow 0$
for $\texttt{move\_count} < |T|$ **do**
    for $k \in K$ **do**
        $(w[k], p^\star[k]) \leftarrow$ ImmediateSpectrumOffender$(G(V, L), K, k)$
    sort K in decreasing order by W
    $k \leftarrow K.top$
    while $\texttt{move\_count} < |T|$ **and** $w[k] > 0$ **and** $k \neq K.end()$ **do**
        if $\texttt{\#Moves}(k) < \texttt{ConnectionMoveLimit}$ **then**
            if k can be rerouted to the $p^\star[k]$ **and** $p^\star[k]$ is better than k_p **then**
                Update lightpath of k by $p^\star[k]$
                $\texttt{\#Moves}(k) \leftarrow \texttt{\#Moves}(k) + 1$
                $\texttt{move\_count} \leftarrow \texttt{move\_count} + 1$
        $k \leftarrow K.next()$

Algorithm 9 Complete Spectrum Offender Sort Heuristic

for $k \in K$ **do**
    $\texttt{\#Moves}(k) \leftarrow 0$
for $\texttt{round\_count} < \texttt{RoundLimit}$ **do**
    for $k \in K$ **do**
        $(w[k], \underline{p}[k]) \leftarrow$ ImmediateSpectrumOffender$(G(V, L), \emptyset, k)$
    sort K in decreasing order by w
    $k \leftarrow K.top$
    while $\texttt{move\_count} < |\mathrm{T}|$ **and** $w[k] > 0$ **and** $k \neq K.end()$ **do**
        if $\texttt{\#Moves}(k) < \texttt{ConnectionMoveLimit}$ **then**
            $p^\star \leftarrow$ ImmediateSpectrumOffender$(G(V, L), K \setminus k, k)$
            if $p^\star$ is better than the current path **then**
                Update lightpath of k by $p^\star$
                $\texttt{\#Moves}(k) \leftarrow \texttt{\#Moves}(k) + 1$
                $\texttt{move\_count} \leftarrow \texttt{move\_count} + 1$
        $k \leftarrow K.next(k)$
    $\texttt{round\_count} \leftarrow \texttt{round\_count} + 1$

Algorithm 10 Spectrum Usage Sort Heuristic

for $k \in K$ **do**
    $\texttt{\#Moves}(k) \leftarrow 0$
for $\texttt{round\_count} < \texttt{RoundLimit}$ **do**
    $w[k] \leftarrow k_{p_b} \times |k_{p_L}|$
    sort K in decreasing order by their spectrum usage values $w[k]$
    $k \leftarrow K.top$
    while $\texttt{move\_count} < |T|$ **and** $w[k] > 0$ **and** $k \neq K.end()$ **do**
        if $\texttt{\#Moves}(k) < \texttt{ConnectionMoveLimit}$ **then**
            $p^\star \leftarrow$ ImmediateSpectrumOffender$(G(V, L), K \setminus k, k)$
            if $p^\star$ is better than the current path **then**
                Update lightpath of k by $p^\star$
                $\texttt{\#Moves}(k) \leftarrow \texttt{\#Moves}(k) + 1$
                $\texttt{move\_count} \leftarrow \texttt{move\_count} + 1$
        $k \leftarrow K.next(k)$
    $\texttt{round\_count} \leftarrow \texttt{round\_count} + 1$

connection to any new position. To perform these techniques, connections are sorted in increasing order according to their current starting slots.

As these spectrum squeezing methods are used as post-processing of CSO, ISO SPU algorithms, the following algorithms are not presented in terms of pseudo-code to ease the readability of this paper. As a reminder, HT and PP used in this paper as strategies, not technologies, thus they are performed under the MBB condition. Thus, by combining three proposed algorithm with HT and PP, six new algorithms are created: CSO-HT, CSO-PP, SPU-HT, SPU-HT, CSO-HT, CSO-PP.

5.4.6 Column Generation Algorithm

The column generation solution process is presented in Algorithm 11. Note that, in this algorithm, costDijkstra$(t, k, m, \widetilde{RMP})$ is the function to solve the frequency pricing problem without distance constraints, while costILP$(t, k, m, \widetilde{RMP})$ use the labeling algorithm implemented by [63] to solve the exact problem.

When the column generation process finishes, the variables are set back to integer requirements. It is obvious that the number of variables and constraints are extremely

Algorithm 11 Solution Process

$\widetilde{RMP} \leftarrow$ Linear Relaxation of Restricted Master Problem with initial columns
repeat
    for $t \leftarrow 1$ **to** $|T|$ **do**
        for $k \in K$ **do**
            for $m \in M$ **with** $\overline{\delta}(m) \leq \overline{\delta}(k_m)$ **do**
                if costDijkstra$(t, k, m, \widetilde{RMP}) < 0$ **then**
                    add the new column to $\widetilde{RMP}$
                    solve $\widetilde{RMP}$
                else if $\exists f \in F_{\max} : PP_{k,f}^{t,m} \not\equiv$ costDijkstra$(f, t, k, m, \widetilde{RMP})$ **then**
                    $K^{\text{ILP}} \leftarrow K^{\text{ILP}} \cup (k, mod)$
    if no new column found **and** $K^{\text{ILP}} \neq \emptyset$ **then**
        for $t \leftarrow 1$ **to** $|T|$ **do**
            for $(k, m) \in K^{\text{ILP}}$ **do**
                while costILP$(t, k, m, \widetilde{RMP}) < 0$ **do**
                    add the new column to $\widetilde{RMP}$
                    solve $\widetilde{RMP}$
until no new column is found
$RMP \leftarrow$ ILP representation of $\widetilde{RMP}$
return ILP solution of RMP

huge for the ILP being solved. Fortunately, it is not necessary to keep all the variables and constraints in the ILP as the restricted master problem, assuming that the number of generated column is considerably smaller than the y variables in the restricted master problem.

On link ℓ and frequency slot f, assuming that there are two rerouting events $t_1 < t_2$ such that there is not generated column $z^t_{k,p}$ associated to ℓ and f (i.e., p use f and ℓ), and $t_1 < t < t2$. In that case, we can remove all the constraints (5.5), indexed by t, ℓ and f where $t_1 < t < t2$, and change the constraints related to t_2 as $y^{t_2}_{l,f} \geq y^{t_1}_{\ell,f} - \sum_{k\in K} \sum_{p\in P^{t_2}_k} (a^0_{k,\ell,f} - \alpha^p_{\ell,f}) z^{t_2}_{k,p}$. Thus, we can also remove the according variable $y^t_{l,f}$ where $t_1 < t < t2$.

Note that these pricing problems (5.22)-(5.26) may offer a alternative path overlapping with some spectrum slots on some links of the original connection. In addition, the master problem has no constraint to avoid this issue. Therefore, with this pricing models, the ILP solution of column generation algorithm is not a solution of the optimal MBB reachable defragmentation. However, the linear relaxation of the the master problem is still a lower bound of the original problem.

5.5 Numerical Results

5.5.1 Simulation Setting

Simulation are run on USA topology [64] having 24 nodes and 86 directed links. The spectrum for each link is slotted into 400 slots where each slot corresponds to a spectrum interval of width 12.5 GHz.

With this topology, four network configurations are produced:

- All nodes are CDC ROADM,
- All nodes are CD ROADM with 2-DIA blocks, 3-DIA blocks and 4-DIA blocks, respectively.

Each connection request requires one of three types of data rates that are 100 Gbps, 200 Gbps and 400 Gbps. The number of slots for a connection depends on different modulation format and the data rate. This specification is given in Table 17.

Data Rate (Gbps)	Modulation	#Slots	Distance (km)
100	BPSK	8	>4000
	QPSK	3	4000
	8QAM	2	1200
	16QAM	1	600
200	BPSK	16	>4000
	QPSK	6	4000
	8QAM	4	1200
	16QAM	3	600
400	BPSK	32	>4000
	QPSK	12	4000
	8QAM	8	1200
	16QAM	6	600

Table 17: Table of Modulations

For all simulations, a total of 300 time units are simulated from an empty network. Requests arrive in the network with a Poisson Distribution interval, with λ_{100}, λ_{200} and λ_{400} are expected interval time of requests of 100 Gbps, 200 Gbps and 400 Gbps respectively. It means that in a time unit, there is average $1/\lambda_{100}$ 100-Gbps requests arriving, and so on. Each connection request also has a Exponential Distribution with expected holding time $\lambda_h = 10(timeunits)$. The offered load of a simulation is:

$$Er = \left(\frac{100}{\lambda_{100}} + \frac{200}{\lambda_{200}} + \times\frac{400}{\lambda_{400}}\right) \times \lambda_h \quad (Gbps). \tag{5.32}$$

In other words, assuming all connection requests are being granted, the expected throughput of simulation at any time (after warming up period) is Er.

The source and destination nodes are randomly selected among the nodes of the network such that traffic started/ended at a node is propositional to its population.

All simulations are implemented by C++ language in a machine with Windows 10 operating system, processor AMD Ryzen Threadripper 2590X 16-Core 3.60 Ghz, and 128 GB of RAM. The optimizer is Gurobi version 8.1 [65].

5.5.2 Performance of Defragmentation Algorithms in static simulation

Performance of column generation algorithm is presented in Table 18. It shows that the proposed solution process for column generation is ε-optimal (with a gap of about 4%). Furthermore, the required CPU times are accelerated 10 times with double instance size in comparison with [56] (except for 3 out of 11 cases).

Table 18: T = 50, nodes = 10, links = 32, slots = 50

Time point	connections	gap (%)	time (s)	generated configs
100	250	4.0	743.4	24317
110	263	2.9	6072.0	35896
120	252	4.6	362.6	17924
130	251	2.0	53671.4	38407
140	263	4.3	453.6	21012
150	244	4.8	467.5	20243
160	250	8.0	596.9	22917
170	256	3.2	10633.9	36882
180	252	4.5	616.2	23991
190	252	5.8	559.8	20799
200	243	3.3	383.8	16078
Average	252.4	4.3	6778.3	25315.1

Optimality evaluation of proposed algorithms is presented in Table 19. The results show that our proposed heuristics for spectrum usage minimizing methods are closed to lower bound in these instances. Note that heuristic algorithms are offering better solutions than column generation's, it is due to we are not using heuristics solution as initial configurations for column generation solution process.

5.5.3 Impact of defragmentation process

The average bandwidth blocking rates (BBR) of 12 defragmentation strategies are presented in Table 20. The last column of this table also reports the BBR when

Table 19: Algorithm optimality evaluations

Time point	initial slots	CG		SPU		CSO		Lower bound
		sol.	gap (%)	sol.	gap (%)	sol.	gap (%)	
100	1112	1044	4.0	1016	1.2	1013	0.9	1004
110	1217	1120	2.9	1118	2.8	1118	2.8	1088
120	1143	1071	4.6	1030	0.6	1024	0.0	1024
130	1189	1081	2.0	1066	0.6	1066	0.6	1060
140	1205	1146	4.3	1132	3.0	1132	3.0	1099
150	1134	1080	4.8	1049	1.7	1049	1.7	1031
160	1153	1103	8.0	1069	4.7	1033	1.2	1021
170	1202	1099	3.2	1089	2.3	1092	2.5	1065
180	1193	1126	4.5	1107	2.8	1101	2.2	1077
190	1185	1132	5.8	1109	3.6	1097	2.5	1070
200	1139	1076	3.3	1063	2.0	1060	1.7	1042

simulation does not go through any defragmentation event. Each row corresponds to the simulation of an offered load which is shown at the first column of a row.

Firstly, when one of nine minimizing-spectrum-usage-related defragmentation strategies are performed, the BRR is always reduced (by at most 2.4% where 140 Tbps with ISO-HT). These strategies help the network has more available space for incoming requests. Among sorting strategies, they are showing a similar impact on defragmentation. On the other hand, the spectrum-squeezing strategies, HT and PP, have a little impact on the network. Although squeezed spectrum brings more continuous and contiguous space, this effect does not last long due to highly dynamic connection requests. To magnify the impact of HT and PP, the network must be defragmented more often, but the network's customers rarely accept too many modifications to their connections. HT and PP can even decrease the continuity and contiguity of the network's spectrum when connections are distributed at both ends of the spectrum range before defragmentation events. In such a case, HT and PP reduce the big contiguous block in the middle of the network's spectrum.

These results are also showing that adding push-pull spectrum squeezing does not

improve spectrum minimizing. There are 5 out of 18 data sets where minimizing-spectrum-usage-and-push-pull defragmentation strategy is worse than minimizing-spectrum-usage goal and 3 cases where it has little advantage. This effect is explained as follows. After the spectrum minimizing, connections are more evenly distributed along the link spectrum, then push-pull spectrum squeezing is very limited to move connections downward. Conversely, hop-tuning spectrum squeezing incorporates better with spectrum usage minimizing. There are only 3 cases where hop-tuning spectrum squeezing does not offer support. It is due to hop-tuning strategy has more freedom to squeezes spectrum, rather than push-pull.

Offered load (Tbps)	Overall blocking rate (%) (all CDC, mixed Gbps)											
	CSO-HT	CSO-PP	CSO	SPU-HT	SPU-PP	SPU	ISO-HT	ISO-PP	ISO	HT	PP	no_defrag
60	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
100	3.5	4.3	4.2	4.1	4.1	4.1	3.8	4.1	4.1	5.2	5.2	5.3
140	11.6	11.6	12.0	11.9	12.3	12.0	11.5	11.9	11.9	13.5	13.8	13.9
180	18.7	19.3	18.8	18.7	19.4	19.4	18.7	19.0	19.3	21.4	20.5	21.1
220	23.0	23.1	23.0	22.9	23.2	22.8	22.7	23.1	23.1	24.3	24.5	24.0
260	30.0	29.7	29.4	29.7	30.1	29.8	29.6	29.6	29.9	31.7	31.5	31.7

Table 20: Heuristic Performances

5.5.4 Impact of Contentionless ROADM

In this section, impact of contentionless property of ROADM is examined. Table 21 and 22 , for an offered load of 100 Tbps and 140 Tbps respectively, reports BBR of full 2-DIA, full 3-DIA and 4-DIA network configurations. First of all, a full 2-DIA configuration is having highest BBR in any with defragmentation or without defragmentation simulations. Since a node 2 DIA blocks accept at most two overlapping adding/dropping connections, it has the highest contention probability when routing and rerouting connections. When nodes have more DIA blocks or nodes upgraded by CDC ROADM, network's BBR is reduced with most defragmentation strategies. In addition, notice that simulations without defragmentation report tight BBRs of full CDC configuration and others. It means Contentionless ROADM has a significant impact on defragmentation strategies, rather than accepting connections. It is due to, firstly, defragmentation with CDC ROADM has more freedom to modify connections in terms of spectrum at add/drop side, and secondly, connection arriving and terminating between two defragmentation events are highly dynamic resulting contention

at links' side more than add/drop side.

Offered load (Tbps)	Blocking rate (%) (100 Tbps, mixed Gbps)											
	RI-CSO-HT	RI-CSO-PP	RI-CSO	RI-SPU-HT	RI-SPU-PP	RI-SPU	RI-ISO-HT	RI-ISO-PP	RI-ISO	HT	PP	no_defrag
all-CDC	3.5	4.3	4.2	4.1	4.1	4.1	3.8	4.1	4.1	5.2	5.2	5.3
4-DIA	4.3	4.5	4.4	4.1	4.3	4.5	4.2	4.6	4.7	5.0	5.4	5.2
3-DIA	4.6	4.5	4.7	4.5	4.8	4.4	4.3	4.7	4.4	5.1	5.2	5.1
2-DIA	5.4	5.6	5.4	5.6	5.9	5.7	5.3	5.7	5.5	6.1	6.2	6.4

Table 21: CDC ROADM vs CD ROADM, 100 Tbps

Offered load (Tbps)	Blocking rate (%) (140 Tbps, mixed Gbps)											
	RI-CSO-HT	RI-CSO-PP	RI-CSO	RI-SPU-HT	RI-SPU-PP	RI-SPU	RI-ISO-HT	RI-ISO-PP	RI-ISO	HT	PP	no_defrag
all-CDC	11.6	11.6	12.0	11.9	12.3	12.0	11.5	11.9	11.9	13.5	13.8	13.9
4-DIA	11.8	12.6	12.5	12.1	12.7	12.6	11.9	12.6	12.3	13.3	14.1	13.6
3-DIA	12.6	12.6	12.8	12.4	12.8	12.5	12.6	12.5	12.8	13.8	14.3	13.7
2-DIA	14.3	14.7	14.7	14.4	14.9	14.7	14.3	14.5	14.7	15.9	15.7	15.8

Table 22: CDC ROADM vs CD ROADM, 140 Tbps

5.5.5 Impact of Connection Granularities

In this section, we are going to examine presented defragmentation strategies to confirm that proposed algorithms performing independently from connection granularity. In other words, presented simulations so far have a fixed fractions of connection granularities, but it is not the case in practice, thus we want to ensure that obtained results are general. To do that, we set two extreme cases where all connection requests asking for only the smallest/biggest granularity. Table 23 and 24 reports the overall BBR of simulations in which connections are homogeneous rate of 100 Gbps or 400 Gbps. We see that these tables together reinforce the efficiency of proposed defragmentation strategies in different network conditions.

Offered load (Tbps)	Overall blocking rate (%) (all CDC, 100 Gbps)											
	RI-CSO-HT	RI-CSO-PP	RI-CSO	RI-SPU-HT	RI-SPU-PP	RI-SPU	RI-ISO-HT	RI-ISO-PP	RI-ISO	HT	PP	no_defrag
60	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
100	0.3	0.5	0.4	0.3	0.5	0.6	0.4	0.4	0.4	0.6	0.7	0.7
140	6.8	6.9	7.0	6.8	6.9	7.2	7.0	6.8	7.0	7.6	7.7	7.7
180	12.3	12.7	12.7	12.8	12.6	12.8	12.6	12.5	12.7	13.7	13.7	13.7
220	-	-	-	-	-	-	-	-	-	-	-	-
260	-	-	-	-	-	-	-	-	-	-	-	-

Table 23: Overal BBR of Homogeneous 100-Gbps Data Rate

Offered load (Tbps)	Overall blocking rate (%) (all CDC, 400 Gbps)											
	RI-CSO-HT	RI-CSO-PP	RI-CSO	RI-SPU-HT	RI-SPU-PP	RI-SPU	RI-ISO-HT	RI-ISO-PP	RI-ISO	HT	PP	no_defrag
60	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.3	0.3	0.1
100	5.2	5.0	5.6	4.7	5.1	5.9	5.2	4.9	5.3	6.2	6.7	6.6
140	11.6	12.1	12.1	11.9	12.6	11.9	11.9	11.9	11.9	13.6	13.3	13.7
180	18.2	17.1	17.1	17.3	17.2	16.9	17.5	17.6	17.6	18.3	18.8	18.8
220	-	-	-	-	-	-	-	-	-	-	-	-
260	-	-	-	-	-	-	-	-	-	-	-	-

Table 24: Overal BBR of Homogeneous 400-Gbps Data Rate

5.6 Conclusion

In this paper, impact of CDC ROADM and defragmentation in EONs are investigated. We proposed a novel defragmentation strategy in which spectrum usage minimizing and spectrum squeezing are combined. First, experiments show that our novel defragmentation is better than other separated strategy. Second, it also shows the advantage of CDC ROADM over 4-DIA CD ROADM when operation includes defragmentation process.

For the future work, we will work to improve optimal algorithm for larger instance sizes. In addition, this paper simplifies many practical factors, e.g., physical interference, so we will consider them in next studies.

Chapter 6

A Nested Decomposition Model for Reliable NFV 5G Network Slicing

With the 5th generation of mobile networking (5G) on our doorstep, optical network operators are reorganizing their network infrastructures so that they can deploy different topologies on the same physical infrastructure on demand. This new paradigm, called network slicing, together with network function virtualization (NFV), can be enabled by segmenting the physical resources based on the requirements of the application level.

In this paper, we investigate a nested decomposition scheme for the design of reliable 5G network slicing. It involves revisiting and improving the previously proposed column generation models, and adding in particular the computation of dual bounds with Lagrangian relaxation in order to assess the accuracy of the solutions.

Extensive computational results show that we can get ε-optimal reliable 5G slicing solutions with small ε (about 1% on average) in fairly reasonable computational times.

This paper is in preparation.

6.1 Introduction

The 5th generation of mobile networking (5G) is based on the key technologies of Software-Defined Networking (SDN) and Network Function Virtualization (NFV) in

order to offer multiple services with various performance requirements, e.g., low latency, high throughput, high reliability, or high security. SDN allows network operators to remotely (re)configure the physical network in order to reserve on demand networking resources. Virtual compute nodes (i.e., node with computing resources such as servers or a data center) can enable Virtual Network Functions (VNFs) running on top of general-purpose hardware, such as a cloud infrastructure.

Within the context of 5G networks, network slicing is an end-to-end logical network provisioned with a set of isolated virtual resources on a shared physical infrastructure. Slices are provided as different customized services to fulfill dynamic demands, with flexible resource allocations. In other words, a network slice is a self-contained network with its own virtual resources, topology, traffic flow, and provisioning rules. Network slicing is therefore a key feature of 5G networks, which allows the efficient resource share of a common physical infrastructure and consequently, reduces operators' network construction costs.

An interesting feature of SDN is its ability to process traffic while forwarding it, using "network functions" or "network services". The latter ones can implement header processing and payload processing functions, such as network address translation (NAT), firewall, or domain name system (DNS). They are VNFs and can be implemented in software on conventional processing systems (e.g., servers or data centers) that are co-located with networking equipment. The sequence of functions that need to be set up for a specific flow is referred to as a "service chain."

In this paper, we propose a 5G network slicing design model and algorithm, based on nested column generation. It aims at maximizing the number of granted slices while addressing the reliability requirements of network slices. In order to avoid the costly exact solutions of the sub-problems, we discuss how to compute bounds using Lagrangian relaxation, so that we can assess the accuracy of the output solutions.

The paper is organized as follows. Section 6.2 contains the literature review. Section 6.3 provides the detailed problem statement of the design of reliable 5G network slicing. An original nested decomposition model is proposed in Section 6.4. Algorithmic aspects are covered in Section 6.5. Numerical results are described in Section 6.6 and conclusions are drawn in the last section.

6.2 Literature Review

6.2.1 5G Network Slicing

Several papers and surveys have already appeared on 5G network slicing and described their various challenges and opportunities [66, 67]. Similarly, many studies and several surveys have been devoted to Network Function Virtualization (NFV), e.g., [68].

Very few studies look at the combination of reliable 5G slicing and NFV. Tang *et al.* [69] propose a MILP for 5G network slicing that maximizes the number of granted slices while minimizing their failure rate, without providing protection mechanisms.

Some authors looked at network slicing and NFV, more often in the wireless networks than in the wired optical ones. Challenges are discussed in, e.g., [70, 66].

Lin *et al.* [71] propose an exact algorithm using column generation aiming to minimize the total embedding cost in terms of spectrum cost and computation cost for a single virtual network request. Moreover, validation of the exact algorithm is made on a six node network. Large data instances are solved using a heuristic. Destounis *et al.* [72] also propose an exact column generation algorithm for network slicing without the NSF features. They solved data instances up to 200 nodes. Carella *et al.* [73] exemplified Network slicing as an addition to the current Cloud architecture and evaluated on a testbed architecture based on the Fraunhofer FOKUS and TU Berlinopen source Open Baton toolkit.

6.2.2 Nested Column Generation Decomposition

The idea of nested column generation is not new: several authors have already investigated it for various problems, e.g., Song [74] in logistics, Dohn and Mason [75] for staff rostering, Karabuk [76] for scheduling paratransit vehicles, Cóccola [77] for inventory-routing problem and Vanderbeck [48] for two-dimension cutting-stock.

However, most studies did not worry about assessing accurately the quality of the output solutions, except, e.g., [48, 78]. In [78], authors developed time consuming branch-and-price algorithms by which they could obtain optimal solutions. On the other hand, Vanderbeck proposed to leverage Lagrangian Relaxation to estimate bounds of a nested column generation algorithm [48], without solving exactly pricing problems. In this work, we reuse this methodology to evaluate our solutions.

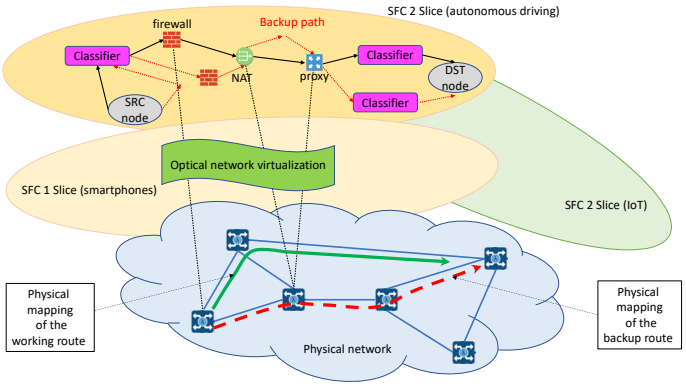


Figure 17: 5G Reliable Slicing

6.3 Problem Statement and Notations

6.3.1 Rel_5G_NFV Problem Statement

Consider a physical network G^{P} and a set K of connections, indexed by k. The Reliable 5G NFV Network Slicing (Rel_5G_NFV) problem consists of embedding/mapping the maximum number of slices onto the physical network while ensuring each slice is individually protected against any single link failure. We assume each slice is associated with a given application, that is characterized with the use of a single service function chain.

6.3.2 Notations

Physical Network. The physical network $G^{\text{P}} = (V^{\text{P}}, L^{\text{P}})$ is defined by its set of nodes V^{P}, indexed by v, set of links $\ell \in L^{\text{P}}$, with capacities $\text{CAP}_v \geq 0$ and $\text{CAP}_\ell \geq 0$ on both nodes and links, respectively.

5G Slicing. Each slice $S \in \mathcal{S}$ is associated with a virtual network $S = (V^S, L^S, \text{CAP}^S)$, which is defined by a set of virtual nodes V^S (indexed by v'), and virtual links L^S (indexed by ℓ'), with capacity requirements $\text{CAP}^S_{v'}$ and $\text{CAP}^S_{\ell'}$, respectively.

Virtual Networks. An embedding of S onto G^{P} consists of mapping:

- Each virtual node $v' \in V^S$ onto a physical node $v \in V^{\text{P}}$
- Each virtual link ℓ' onto a loop-free physical path, connecting two physical nodes u and v, to which the virtual nodes u' and v' have been mapped

- Each virtual "path" is protected by a virtual path, whose mapping is physical link-disjoint from the mapping of the first path.

A feasible embedding is an embedding in which all physical link and node capacity constraints are satisfied; that is, the sum of capacity demands of all virtual nodes embedded on a physical node is less than the capacity of this physical node, and the sum of the requests of all the virtual links going through a physical link does not exceed the capacity of this link.

In order to simplify the model and the algorithm, we work directly with the mapping of the virtual nodes/links, i.e., with physical nodes/links, without expressing explicitly the virtual links and nodes.

Service Function Chaining (SFC). Let F be the set of all services functions, indexed by f, and let C be the set of all service function chains, indexed by c. Any chain c is defined by an ordered sequence of n_c functions: $c = \{f_0, f_1, .., f_{n_c-1}\}$. The routing of any demand in a slice governed by SFC c must go through virtual compute nodes hosting the functions of c.

Application (Slice) Demand. Demands are provided for each slice S, with each slice being associated with one particular application, characterized by a given SFC c_S. We denote by K^{sd,c_S} the demand for node pair $(v_s, v_d) \in \mathcal{SD}_{c_S}$, i.e., with traffic in slice S, subject to the requirement of SFC c_S, and by $\Delta_{f_i}^{sd,c}$ the required computational resource of function f_i for demand K^{sd,c_S}.

6.4 A Nested Decomposition Scheme

We now present a nested decomposition scheme, in which at the upper layer of the decomposition, we select the slice configurations for each slice demand. Each slice configuration is defined by a virtual network as defined in Section 6.3.2, which satisfies the demand K^{c_S} associated with its required application and corresponding SFC c_S.

Let Γ, indexed by γ, be set of all possible slice configurations. Each slice configuration γ is characterized by a slice S and its assigned resources. Each slice configuration γ is characterized by its slice index S, its node assigned resources R_v^γ, and its link assigned resources B_ℓ^γ. We have $\Gamma = \bigcup_{c \in C} \Gamma_{c_S}$.

In order to simplify the notations, we will simply write c unless there is confusion.

6.4.1 Master Problem

Master problem maximizes the grade of service (GoS) subject to capacity constraints. It requires only one set of variables: $z_\gamma = 1$ if potential slice virtual network γ associated with c is selected, 0 otherwise, for $\gamma \in \Gamma_c$ and $c \in C$.

Objective:

$$\max \quad \sum_{c \in C} \sum_{\gamma \in \Gamma_c} \sum_{(s,d) \in \mathcal{SD}_c} K^{sd,c} \, z_\gamma \tag{6.1}$$

subject to:

$$\sum_{\gamma \in \Gamma_c} z_\gamma \leq 1 \qquad c \in C \tag{6.2}$$

$$\sum_{c \in C} \sum_{\gamma \in \Gamma_c} R_v^\gamma z_\gamma \leq \text{CAP}_v \qquad v \in V^{\text{P}} \tag{6.3}$$

$$\sum_{c \in C} \sum_{\gamma \in \Gamma_c} B_\ell^\gamma z_\gamma \leq \text{CAP}_\ell \qquad \ell \in L^{\text{P}} \tag{6.4}$$

$$z_\gamma \in \{0, 1\} \qquad \gamma \in \Gamma \tag{6.5}$$

Constraints (6.2) impose to select at most one virtual network (slice) for demand associated with $c \in C$. Constraints (6.3) enforce the compute node capabilities, while constraints (6.4) enforce the link transport capacities.

6.4.2 Slicing Pricing Problem (PP_{SLICE})

In order to be able to compute the required node and link resource for a given slice, the pricing problem, or equivalently, the slice configuration generator, needs to provision the demand K^c. We define the following parameters.

Parameters:

- $\pi \in \Pi$: a logical path that defines a service path with chain c from s to d. Note that a logical path may go through a given physical link several times due to the sequence of functions in c.
- $\Pi_{sd}^c \subseteq \Pi$: set of all potential paths for service chain c from s to d.
- $a_v^{i,\pi} = 1$ if, on path π, function f_i is hosted on physical node v, 0 otherwise.
- $\delta_\ell^\pi =$ number of times path π goes through link ℓ

- $x_\ell^\pi = 1$ if logical path π goes through physical link ℓ at least once, 0 otherwise.

Variables:

- $y_{\pi,p}^{sd,c} = 1$ if path π is the primary path to provision traffic from s to d, 0 otherwise.
- $y_{\pi,b}^{sd,c} = 1$ if path π is the backup path to provision traffic from s to d, 0 otherwise.

Objective:

$$\max \quad \mathrm{RC}_{\mathrm{PP}_{\mathrm{SLICE}}} = \sum_{(s,d)\in\mathcal{SD}} K^{sd,c} - u_c^{(6.2)} - \sum_{v\in V^{\mathrm{P}}} u_v^{(6.3)} \sum_{(s,d)\in\mathcal{SD}} \sum_{i=0}^{n_c-1} \sum_{\pi\in\Pi_{sd}^c} \Delta_{f_i}^{sd} a_v^{i,\pi} (y_{\pi,p}^{sd,c} + y_{\pi,b}^{sd,c}) - \sum_{\ell\in L^{\mathrm{P}}} u_\ell^{(6.4)} \sum_{(s,d)\in\mathcal{SD}} \sum_{\pi\in\Pi_{sd}^c} K^{sd,c} \delta_\ell^\pi (y_{\pi,p}^{sd,c} + y_{\pi,b}^{sd,c}) \quad (6.6)$$

Constraints:

One primary path per demand:

$$\sum_{\pi\in\Pi_{sd}^c} y_{\pi,p}^{sd,c} = 1 \qquad (v_s, v_d) \in \mathcal{SD} \quad (6.7)$$

One backup path per demand:

$$\sum_{\pi\in\Pi_{sd}^c} y_{\pi,b}^{sd,c} = 1 \qquad (v_s, v_d) \in \mathcal{SD}. \quad (6.8)$$

Link disjoint primary and backup paths:

$$\sum_{\pi\in\Pi_{sd}^c} x_\ell^\pi (y_{\pi,p}^{sd,c} + y_{\pi,b}^{sd,c}) \le 1 \qquad (v_s, v_d) \in \mathcal{SD}, \ell \in L^{\mathrm{P}}. \quad (6.9)$$

Link and node capacities:

$$(R_v =) \sum_{(v_s,v_d)\in\mathcal{SD}} \sum_{i=0}^{n_c-1} \sum_{\pi\in\Pi_{sd}^c} \Delta_{f_i}^{sd} a_v^{i,\pi} (y_{\pi,p}^{sd,c} + y_{\pi,b}^{sd,c}) \le \mathrm{CAP}_v \qquad v \in V^{\mathrm{P}} \quad (6.10)$$

$$(B_\ell =) \sum_{(v_s,v_d)\in\mathcal{SD}} \sum_{\pi\in\Pi_{sd}^c} K^{sd,c} \delta_\ell^\pi \, (y_{\pi,p}^{sd,c} + y_{\pi,b}^{sd,c}) \le \mathrm{CAP}_\ell \qquad \ell \in L^{\mathrm{P}}. \quad (6.11)$$

6.4.3 Path Pricing Problem (PP_{sd}): Service path for Demand from v_s to v_d

For a given $(v_s, v_d) \in \mathcal{SD}$, we look for the generation of a path π from v_s to v_d, which can improve the linear programming relaxation of PP_{SLICE}. The following formulations are developed for the primary path pricing problem. However, the backup path pricing problem is similar when we replace $u_{sd,p}^{(6.7)}$ by $u_{sd,b}^{(6.8)}$.

Variables:

- $x_\ell^\pi = 1$ if path π uses ℓ, 0 otherwise.
- $\delta_\ell^\pi =$ number of times path π goes through ℓ.
- $\varphi_\ell^{sd,c,i} = 1$ if, for service chain c, the path from v_s to v_d uses link ℓ to go from the location of function f_{i-1} to the location of function f_i, 0 otherwise. Note that, when $i = 0$, $\varphi_\ell^{sd,c,i}$ represents the path from the source to the first function, when $i = n_c$, it is the path from the last function to the destination.
- $a_v^i = 1$ if the ith function (f_i) of chain c is installed on node v, 0 otherwise.

Objective:

$$\max \quad \left(-\sum_{v \in V^{\text{P}}} u_v^{(6.3)} \sum_{i=0}^{n_c-1} \Delta_{f_i}^{sd} a_v^i - \sum_{\ell \in L^{\text{P}}} u_\ell^{(6.4)} K^{sd,c} \delta_\ell^\pi) \right) - u_{sd,p}^{(6.7)} - \sum_{\ell \in L^{\text{P}}} x_l^\pi u_{sd}^{(6.9)} - \sum_{v \in V} \sum_{i=0}^{n_c-1} \Delta_{f_i}^{sd} a_v^i u_v^{(6.10)} - \sum_{\ell \in L} u_\ell^{(6.11)} K^{sd,c} \delta_\ell^\pi \quad (6.12)$$

Constraints:

Aggregation of link usage:

$$\delta_\ell^\pi = \sum_{i=0}^{n_c} \varphi_\ell^{sd,c,i} \qquad \ell \in L^{\text{P}}. \quad (6.13)$$

Multiple usage of a link:

$$\varphi_\ell^i \leq x_\ell^\pi \qquad \ell \in L^{\text{P}}, i \in 0, .., n_c - 1. \quad (6.14)$$

This set of constraints ensures that x_ℓ keeps track of physical link ℓ if it is used by any logical link. Indeed, a link can be used multiple times by a given path, this set of

constraints result x_ℓ as used links, no matter how many times they are used. These variables play the role in the upper pricing where backup path and primary path must be disjoint.

Flow Conservation constraints

$$\sum_{\ell\in\omega^+(v)} \varphi_\ell^{sd,c,0} - \sum_{\ell\in\omega^- v} \varphi_\ell^{sd,c,0} + a_v^0 = \begin{cases} 1 \text{ if } v = v_s \\ 0 \text{ else} \end{cases} \qquad v \in V^{\mathrm{P}} \tag{6.15}$$

$$\sum_{\ell\in\omega^+(v)} \varphi_\ell^{sd,c,n_c} - \sum_{l\in w^- v} \varphi_\ell^{sd,c,n_c} - a_v^{n_c-1} = \begin{cases} -1 & \text{if } v = v_d \\ 0 & \text{else} \end{cases} \qquad v \in V^{\mathrm{P}} \tag{6.16}$$

$$\sum_{\ell\in\omega^+(v)} \varphi_\ell^{sd,c,i} - \sum_{\ell\in\omega^-(v)} \varphi_\ell^{sd,c,i} + a_v^i - a_v^{i-1} = 0 \qquad v \in V^{\mathrm{P}}, 0 < i < n_c. \tag{6.17}$$

Constraints (6.15) ensure that demand starts at the source node, then is transferred through a path to the location of first function (unless first function is located at the source node). Similarly, constraints (6.16) make sure that the demand is delivered to the destination after it is processed by the last function (unless the last function is installed at the destination node). From the location of function $i-1$ to the location of function i, constraints (6.17) define a path to connect them.

We next use constraints to eliminate the ineffective solutions and, as a consequence, those constraints help to improve the quality of the columns, i.e., slice configurations.

A unique node location for each function occurrence in the service chain:

$$\sum_{v\in V^{\mathrm{P}}} a_v^i = 1 \qquad i = 0, 1, \ldots, n_c. \tag{6.18}$$

Node capacity constraints:

$$\sum_{i=0}^{n_c-1} \Delta_{f_i} a_v^i \le \mathrm{CAP}_v \qquad v \in V^{\mathrm{P}} \tag{6.19}$$

Link capacity constraints:

$$\delta_\ell^\pi K^{sd,c} \le \mathrm{CAP}_\ell \qquad \ell \in L^{\mathrm{P}}. \tag{6.20}$$

Domain constraints:

$$x_\ell^\pi, \varphi_\ell^\pi, a_v^i \in \{0, 1\}; \qquad \delta_\ell^\pi \in \mathbb{Z}^+ \tag{6.21}$$

We will discuss in the next section how to solve efficiently the path pricing problems, without requiring the solution of ILP programs at each iteration of the column generation algorithm.

6.5 Nested Column Generation Algorithm

Column generation [79] is based on the fact that, in the simplex method, the solver does not need to simultaneously access all variables of the problem. In fact, a solver can start working only with the basis (a particular subset of the constrained variables), then use a reduced cost to choose the other variables to access, as needed. It is today a very well known and powerful technique [28, 80], while column generation modeling remains an art when the decomposition is not deduced from the application of the Dantzig-Wolfe decomposition.

We next provide the details of our nested column generation algorithm and how we estimated the accuracies of the resulting solutions.

6.5.1 Nested CG and ILP Solution

The conceptual column generation scheme alternates between solving a restriction of the original problem, usually called restricted master problem, and a column generation phase which is used to augment the set of variables/columns of the restricted master problem using a so-called pricing problem. Here, the pricing problem can be decomposed into $|\mathcal{S}|$ slice pricing subproblems.

In order to guarantee reaching an optimal LP solution, it is required to solve at least once the pricing problem. In this study, we propose to solve the slice pricing problem, indeed, the slicing pricing subproblems using again a column generation algorithm. As these last subproblems are Integer Linear Programs (ILPs), and as we did not develop any branch-and-price algorithms to solve them, they are never solved optimally, and therefore we need to derive a linear relaxation bound in order to get upper bounds, see next section for the details.

In any case, at both decomposition levels, we use the column generation algorithm as long as we can derive new improving columns. For integer solutions, when we cannot improve anymore the LP solution, we use an ILP solver on the current constraint matrix, i.e., the constraint matrix made of all the columns generated so

far, and deduce an ILP solution.

Flowcharts in Figure 18 summarize the algorithm. Accuracy of the output solutions is assessed with ε, which is defined as follows:

$$\varepsilon = \frac{\overline{z}_{\text{LP}} - \tilde{z}_{\text{ILP}}}{\tilde{z}_{\text{LP}}},$$

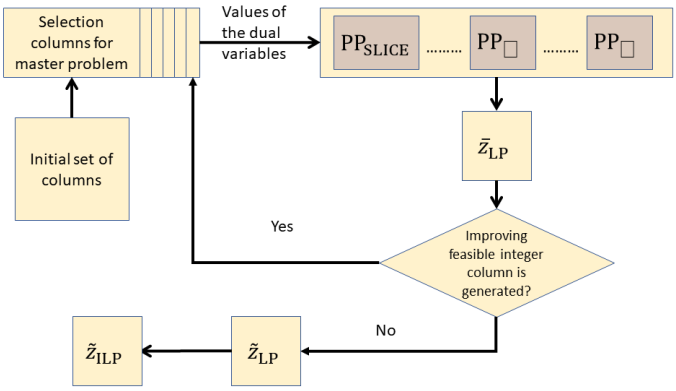
where $\overline{z}_{\text{LP}}$ is an upper bound of the LP solution of problem (6.1)-(6.5), whose calculation is developed in Section 6.5.2. $\tilde{z}_{\text{ILP}}$ is the best found ILP solution (hence a lower bound on the ILP solution), as derived by the solution of the ILP solver on the constraint matrix of (6.1)-(6.5) when no more improved column can be generated by the solution of the slice pricing problem (6.6)-(6.11).

In order to speed-up the solution of the path pricing subproblems, we first use a shortest path algorithm after noting that all the link costs are positive, taking into account the values of the dual variables. It is worth noting that the usage of a shortest path algorithm does not necessarily guarantee the generation of feasible lightpaths with respect to link and node capacities. However, those capacities are enforced in the slice pricing subproblems, and therefore taken care. When the path pricing subproblems are not able to generate improving paths (i.e., with a positive reduced cost), then we use an ILP solver to solve them, with the guarantee to satisfy all node and link capacities. More details about solution process of PP_{SLICE} are given in Section 6.5.3.

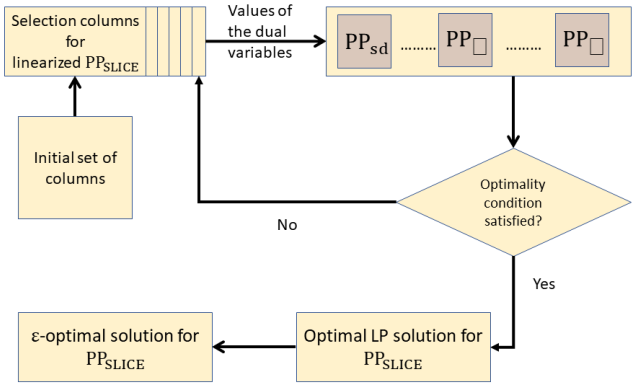
6.5.2 Solution Accuracy

The nested column generation framework allows the efficient exploitation of the substructures of a problem at the expense of a more difficult exact solution of the linear programming relaxation as it a priori requires the exact solution of the upper level pricing problem (here the slice PP_{SLICE} pricing problem), i.e., a branch-and-price algorithm. In order to overcome that difficulty, we propose to compute an upper bound on the objective (i.e., reduced cost) of the PP_{SLICE} problem, and then deduce an upper bound on the optimal LP solution of the Rel_5G_NFV master problem (6.1)-(6.5). It then allows the evaluation of the accuracy (gap) of output ILP solutions using the algorithm described in the previous section.

Consider the compact formulation associated with (6.1)-(6.5), i.e., the COMPACT model such that when applying a Dantzig-Wolfe decomposition to it, we derive model



(a) Upper level flowchart



(b) Lower level flowchart

Figure 18: Flowcharts

(6.1)-(6.5). Let

$$[\text{COMPACT}] \qquad \max\{cx : Ax \leq b, x \in X\}.$$

Using the Dantzig-Wolfe decomposition of Model COMPACT, the slicing pricing problem, PP_{SLICE}, can be written as follows:

$$\text{RC}^{\star}_{\text{PP}_{\text{SLICE}}} = \max\left\{\overline{c}\, x : x \in X^{\text{PRICING}}\right\}. \tag{6.22}$$

We simply write RC to shorten $\text{RC}_{\text{PP}_{\text{SLICE}}}$ when there is no ambiguity so that $\text{RC}^{\star}_{\text{PP}_{\text{SLICE}}} =$

$\text{RC}^\star$.

In Figure 19, we rank the relative positions of the various values that we discuss below. Question marks indicate values that are not computed accurately, and that are upper/lower bounded.

The Lagrangian relaxation of the COMPACT Model can be written:

$$\text{LR}(u) = \max_{x \in X} \left\{ L(u,x) = ub + \underbrace{(c - uA)x}_{\text{RC}(u,x)} \right\}. \tag{6.23}$$

Following Vanderbeck [48] and Pessoa *et al.* [50], a valid upper bound for the COMPACT problem can be computed using Lagrangian Relaxation (LR). At any iteration τ of the column generation algorithm, i.e., when we re-optimize the linear relaxation of the master problem (6.1)-(6.5), the optimal $x_{\text{RC}^\star}$ that maximizes $L(u_\tau, x_{\text{RC}^\star})$ can be written:

$$\begin{aligned} x_{\text{RC}^\star} &= \arg\max_{x \in X} L(u_\tau, x) \quad = \arg\max_{x \in X} \text{RC}(u_\tau, x) \\ &= \arg\max_{i \in I} \text{RC}(u_\tau, x^i) = \arg \max_{x \in X^{\text{PRICING}}} \text{RC}(u_\tau, x), \end{aligned}$$

where $x^i, i \in I$ denote the extreme points of X, see [81], Section II.3.6.

As $x_{\text{RC}^\star}$ is known only if we solve PP_{SLICE} exactly, we can bound it in order to get an upper bound, $\overline{z}_{\text{LP}}$, on the optimal value of the linear programming relaxation. Indeed, $\text{RC}^{\star,\tau}_{\text{ILP}} \leq \text{RC}^{\star,\tau}_{\text{LP}}$, where $\text{RC}^{\star,\tau}_{\text{LP}}$ is the optimal value of the LP relaxation of PP_{SLICE} at iteration τ of the column generation algorithm.

Consequently, $L(u_\tau, x_{\text{RC}^\star}) = u_\tau b + \text{RC}^{\star,\tau}_{\text{ILP}} \leq u_\tau b + \text{RC}^{\tau}_{\text{LP}} = \overline{z}^{\tau}_{\text{LP}}$.

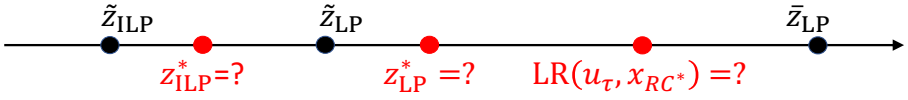


Figure 19: Ranking of the various LP, LR and ILP values.

At each iteration τ of the column generation algorithm, each pricing problem is decomposed into $|\mathcal{S}|$ elementary slice pricing problems of the type PP_{SLICE}. It implies:

$$\overline{z}^{\tau}_{\text{LP}} = u_\tau b + \sum_{S \in \mathcal{S}} \text{RC}^{\star}_{\text{LP}} \left(\text{PP}_{\text{SLICE}}(S) \right).$$

Note that the Lagrangian relaxation upper bound does not improve monotonically [50], thus, in order to derive the best possible upper bound, the algorithm must compute

$$\overline{z}_{\text{LP}} = \min_{\tau} \overline{z}^{\tau}_{\text{LP}} = \min_{\tau} \left\{ u_{\tau} b + \sum_{S \in \mathcal{S}} \text{RC}^{\star,\tau}_{\text{LP}} \left(\text{PP}_{\text{SLICE}}(S) \right) \right\}.$$

It remains possible to add several columns (i.e., slices) at a time (whose $\widetilde{\text{RC}}^{\tau}_{\text{ILP}} \left(\text{PP}_{\text{SLICE}}(S) \right) > 0$) to the master problem (6.1)-(6.5) in one iteration, as long as they are generated with the same set of dual values. Note that output ILP solutions of $\text{PP}_{\text{SLICE}}(S)$ are not guaranteed to be optimal, hence the notation $\widetilde{\text{RC}}$ to denote a heuristic solution of the slice pricing problem. Indeed, the algorithm has to go through all slice subproblems in each iteration to ensure the correctness of the Lagrangian bound.

6.5.3 Solution Process for PP_{SLICE}

Note that, when the linearized PP_{SLICE} pricing model is solved by column generation algorithm, any feasible solutions of the PP_{sd} pricing problem whose positive reduced costs can be integrated into the current restricted problem. Therefore, it is not necessary to solve exactly PP_{sd} model as long as an efficient heuristic can find a feasible solution with positive reduced cost.

If we remove the terms x^{π}_{ℓ} from the pricing problem PP_{sd}, namely

$$\begin{aligned} \max \quad & \left(- \sum_{v \in V^{\text{P}}} u^{(6.3)}_{v} \sum_{i=0}^{n_c - 1} \Delta^{sd}_{f_i} a^{i,\pi}_{v} - \sum_{\ell \in L^{\text{P}}} u^{(6.4)}_{\ell} K^{sd,c} \delta^{\pi}_{\ell} \right) - u^{(6.7)}_{sd,p} \\ & - \sum_{v \in V} \sum_{i=0}^{n_c - 1} \Delta^{sd}_{f_i} a^{i,\pi}_{v} u^{(6.10)}_{v} - \sum_{\ell \in L} u^{(6.11)}_{\ell} K^{sd,c} \delta^{\pi}_{\ell} \end{aligned} \quad (6.24)$$

subject to (6.13) - (6.18) and (6.21). This a simple shortest path problem in the layered graph, as shown in Section V-B in [82], with weight w^{i}_{ℓ} of the link ℓ at layer i is equal to

$$w^{i}_{\ell} = \sum_{\ell \in L} u^{(6.11)}_{\ell} K^{sd,c} \delta^{\pi}_{\ell} + \sum_{\ell \in L^{\text{P}}} u^{(6.4)}_{\ell} K^{sd,c} \delta^{\pi}_{\ell}, \quad (6.25)$$

and the weight of the link going from layer i to layer $i+1$ at node v is equal to

$$w^{i}_{v} = \sum_{v \in V^{\text{P}}} u^{(6.3)}_{v} \sum_{i=0}^{n_c - 1} \Delta^{sd}_{f_i} a^{i,\pi}_{v} + \sum_{v \in V} \sum_{i=0}^{n_c - 1} \Delta^{sd}_{f_i} a^{i,\pi}_{v} u^{(6.10)}_{v}. \quad (6.26)$$

Since the dual values of the constraints (6.11), (6.4), (6.10) and (6.3) cannot be negative, we use the Dijkstra algorithm to solve it. If link $\ell \in L^p$ appears in the found path by the Dijkstra algorithm, then it is added up to the actual reduced cost of the path. If the found path's reduced cost is positive, then it is given to the upper restricted problem. Otherwise, we will compute a upper bound (which is described below). Solving this pricing problem's ILP formulation is needed only when the upper bound is positive.

To compute a upper bound, we can use Dijkstra algorithm one more time to quickly estimate a PP_{sd} problem's upper bound. Namely, that is,

$$\max \quad -\sum_{\ell \in L^p} u_{sd}^{(6.9)} x_l^{\pi} \tag{6.27}$$

subject to (6.13) - (6.18) and (6.21). This problem can be easily reduced to a simple shortest path in the original graph with the weight w_ℓ of the link ℓ is

$$w_\ell^i = u_{sd,\ell}^{(6.9)} \quad (\geq 0). \tag{6.28}$$

The sum of solutions of these two subproblems is an upper bound for the original PP_{sd} pricing problem. If the upper bound is non-positive, then this pricing problem is skipped due to it is impossible to generate an improving column for the restricted PP_{SLICE} in this round. Otherwise, this pricing problem's ILP formulation is solved exactly.

To improve the solution's accuracy, for each round, after all the pricing problems are solved, we reconsider successful pricing problems to immediately generate link-disjoint paths according to the generated paths. For each pricing problem offered an improving primary/backup path, this path is then removed from the graph and the pricing problem is modified accordingly. If the modified pricing problem produces a backup/primary path with a positive reduced cost, then this path is also given to the restricted master problem. In other words, for each connection request, in each round, we try to generate a pair of disjoint paths for its primary and backup connections. If we only use a classical solution process, i.e., generate one path (primary or backup) for one connection request at each round, we observe that the accuracy is very low.

This low accuracy is due to the link-disjoint constraints (6.9) in the restricted problem PP_{SLICE} are easily satisfied when the variables are linear values, although the primary and backup paths are sharing links. For example, if there is only one

path π is generated as both primary and backup path for connection request sd, then $y_{\pi,p}^{sd,c} = y_{\pi,p}^{sd,c} = 0.5$ in the linearized problem. However, this case leads to an infeasible ILP formulation. In addition, this situation also forces the process early terminated as it cannot produce feasible slice configuration resulting in a lose upper bound. By avoiding directly this situation for each found path, the solutions are really closed to the bounds as we show in the experiment section.

6.6 Numerical Results

We implemented the model and algorithm described in the previous sections with a C++ program on a Linux computer with 773727 MB RAM and Intel Xeon E5-2687W v3 @ 3.10 GHz 2 processors, 20 cores. We first describe the data sets, and then we report on the performance of the algorithm.

6.6.1 Data Sets

We considered two topologies from SNDLib [83] and their characteristics are described in Table 25. We re-use the traffic matrix of [82] with four SFCs. In order to derive slice demand, for each original SFC in [82], we divided the overall traffic in 4 subsets, resulting into traffic demands for 16 slices. Transport capacities were set with the optimal solution when allowing only one NFV node.

Topologies	# nodes	# links	# connections per slice	# slices	Offered load
INTERNET2	10	34	90	16	1Tb
ATLANTA	15	44	210	16	1Tb
GERMANY	50	176	2450	16	1Tb

Table 25: Data sets

6.6.2 Model and Algorithm Efficiency and Accuracy

We conducted experiments with the same link transport capacities, and increased node capacities as we increase the number of NFV (compute) nodes. Corresponding

accuracies and computational times (seconds) are reported in Table 26 and 27. We observe that resulting accuracies are less than 3% except for 4 cases where the gap can reach up to 5.6%. Data Instances are easier to solve as the number of NFVs is increasing, and computational times are fairly reasonable taking inot account the accuracies and the complexity of the design problem of reliable 5G network slicing.

The last four columns of these two tables report the number of generated path configurations and solved path pricing problems. The noticeable differences between these numbers represent the remarkable efficiency of the heuristic algorithm to produce promising paths during the column generation process.

# NFV nodes	ϵ (%)	Generated configurations		Solved PP_{sd}	
		Heuristic	ILP	Heuristic	ILP
1	3.8	14584	28000	373000	65750
2	2.1	9696	16736	365608	67152
3	0.4	6992	5128	284296	7324
4	0.4	6440	2808	279700	4112
5	0.4	6208	2864	277064	4844
6	0.4	5760	976	267840	1784
7	0.4	5760	992	267840	1792
8	0.1	6000	4592	296400	11452
9	0.1	5760	728	267840	1372
10	0.1	5760	640	267840	1328
Average	0.8	7296.0	6346.4	294742.8	16691.0

Table 26: Nested CG performance - Internet2

6.6.3 Parallel Solution Processes

As the slice generators/path generators are independent of each other, the solution process can be accelerated by parallel programming. Observe that the number of available threads is unable to provide one thread for each path generator in each iteration of the master problem, we propose two parallel solution processes:

- Parallel slices: in each iteration, each slice generator is corresponding to one

# NFV nodes	ϵ (%)	Generated configurations		Solved PP_{sd}	
		Heuristic	ILP	Heuristic	ILP
1	5.7	31836	45166	1600732	94095
2	4.3	63770	117200	2172902	493070
3	2.9	24534	35900	1662711	91008
4	2.9	20312	22960	1574452	49168
5	2.9	18768	20336	1558660	38512
6	2.9	17504	19856	1564884	32420
7	0.0	19112	24504	1610940	58332
8	2.9	17280	16488	1545524	37928
9	0.0	17320	17864	1555496	30524
10	0.0	17080	19200	1562200	34412
11	0.0	17160	17784	1552972	34784
12	2.9	16568	19512	1556076	30720
13	0.0	16384	20336	1557700	30628
14	0.0	16400	15072	1540144	23296
15	0.0	16168	13984	1518928	22052
Average	1.8	22013.1	28410.8	1608954.7	73396.6

Table 27: Nested CG performance - Atlanta

thread. In other words, slice generators are solved in parallel.

- Parallel paths: in each iteration, for each slice generator, path generators of this slice generator are divided into parallel groups. In other words, slice generators are solved sequentially, each slice generator is then provided with a given number of parallel threads. Path generators of this slice are equally distributed into these threads.

In Table 28 and 29 we report the CPU time (in seconds) of these parallel solution process.

We observe that the first parallel process offers the most acceleration although it does not require the highest number of parallel threads. The second parallel solution process, although it can use more threads, involves a lot of parallel initiation, communication and termination, thus it is less efficient. Another explanation is that the slice generators require closely similar processing times while the path generators are highly different, e.g., a path between two adjacent nodes and a path between two far nodes. Therefore the wasted time waiting for all threads to finish are different between these two parallel processes.

For the Germany topology, because it is a huge data set to our algorithm, we are only able to solve two instances for the parallel slices process, as reported in Table 30. Note that, each solved PP_{sd} can produce two paths (one primary and one backup), because primary and backup PP_{sd} have the same formulation except a constant value. Thus, the number of generated path configurations by ILP (heuristic) can be at most double the number of solved PP_{sd} by ILP (heuristic).

6.6.4 Network Spectrum Usage

We investigated how the network spectrum is used when the number of nodes with compute capacities is increasing, i.e., when there are more network functions distributed all over the network. We provide the results for the ATLANTA topology in Figure 20.

Plots of Figure 20 show that it is more or less the same subset of links which are the most loaded, but their load vary with the number and location of the NFVs, and the increase of the overall network load when the number of NFVs is increasing. Sometimes we see a drop in the load of a link, which is explained by the increase and

# NFV	Parallel slices	Parallel paths			
nodes	(16 threads)	1 thread	30 threads	60 threads	90 threads
1	89.7	871.7	172.1	180.2	197.2
2	98.4	978.2	205.4	228.7	250.3
3	34.1	370.7	65.6	63.2	61.2
4	31.4	349.3	60.8	57.9	57.2
5	33.7	359.3	63.1	61.5	58.9
6	29.7	334.3	57.3	56.3	52.7
7	29.6	339.7	55.9	55.3	51.9
8	47.0	485.6	86.5	88.2	83.8
9	30.6	353.8	56.4	56.1	52.7
10	31.9	359.3	56.8	56.6	52.5

Table 28: Parallel Programming performance - Internet2

position of more NFVs. In conclusion, dimensioning of the link is very dependent on the number and location of the NFVs.

In terms of protection requirements, we investigate the amount of bandwidth reservation for backup paths against the resource allocated to the primary ones. In Figure 21, we report the bandwidth allocation for each instances of the Internet2 topology. It is showing that the backup paths require significantly more resource than the primary ones (about 150%). It confirms previous network provisioning studies that finding disjoint backup paths is more difficult than the primary ones.

We also investigated the throughput evolution when the number of NFV nodes increases and results are depicted in Figure 22 for the ATLANTA network. We observe that as soon as we reach four or five NFV nodes, then the throughput does not increase significantly anymore.

6.7 Conclusions

We designed a first efficient nested decomposition scheme for reliable 5G slicing. Future work will include several algorithmic enhancements such as new modeling of pricing problems and greedy heuristics to generate an initial solution (i.e., initial

# NFV nodes	Parallel slices (16 threads)	Parallel paths			
		1 thread	30 threads	60 threads	90 threads
1	297.5	3316.3	541.4	536.9	547.2
2	1127.0	8488.2	2418.1	2282.2	2603.4
3	467.5	4108.8	689.7	754.3	731.7
4	325.4	3255.8	535.5	524.1	546
5	286.1	3103.8	490.6	468.9	497.4
6	276.9	3095.2	473.3	449.6	477
7	362.8	3643.8	594.3	579	607.1
8	294.8	3256.6	489.8	472.7	490.3
9	317.8	3349.3	513.4	486.9	505.2
10	325.1	3480.3	529	504.7	509.8
11	330.2	3521.6	531.2	503.7	518.6
12	328.8	3536.4	528.1	493.7	510.7
13	323.8	3574.3	527.1	496.9	508.4
14	314.3	3530.5	505.9	480	500
15	309.7	3541.3	488.5	463.4	479.3

Table 29: Parallel Programming performance - Atlanta

# NFV nodes	ϵ (%)	CPU (s)	Generated configurations		Solved PP_{sd}	
			Heuristic	ILP	Heuristic	ILP
50	0.0	237558	216778	214970	194472228	135397
40	0.0	211231	194278	208240	194710738	131067

Table 30: Nested CG performance - Germany

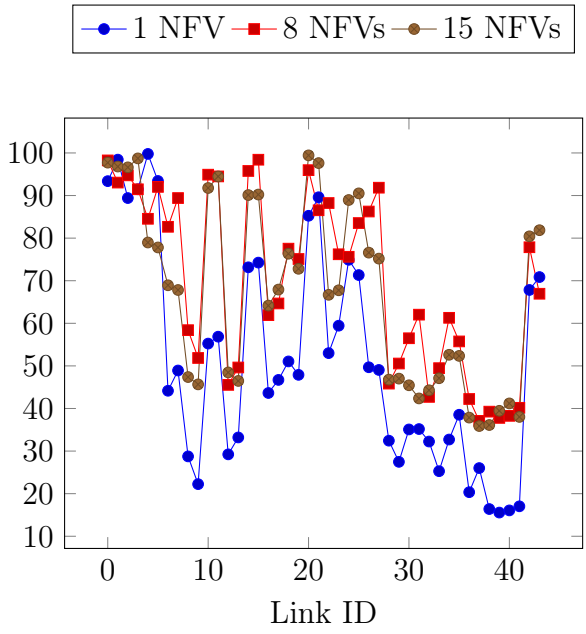


Figure 20: Physical Link Load - ATLANTA Topology

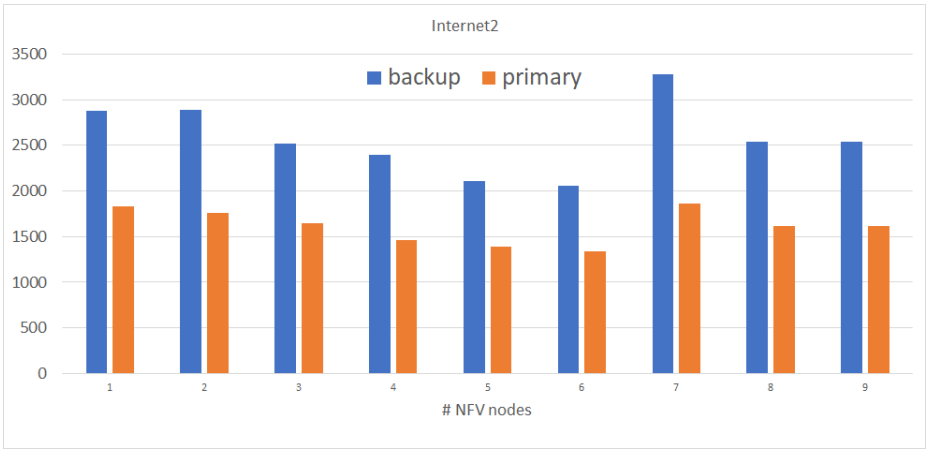


Figure 21: Bandwidth reservation for backup and primary paths

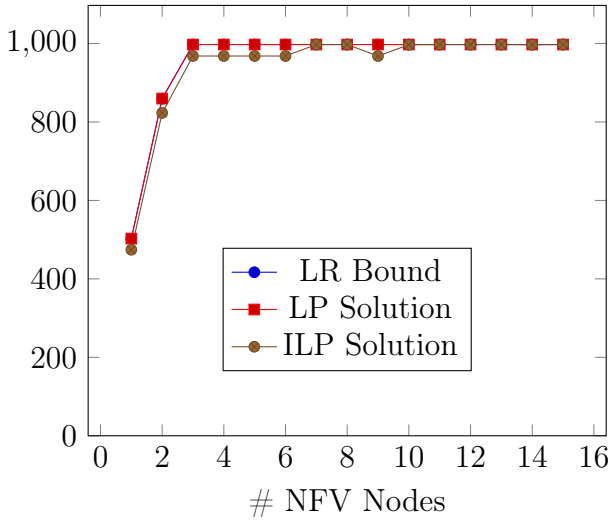


Figure 22: Throughput evolution with an increasing number of NFV nodes

columns at both decomposition levels).

Chapter 7

Conclusions and Future Work

7.1 Conclusions

To recap, in this thesis, we investigate three topics:

- Defragmentation at Logical Layer,
- Defragmentation at Optical Layer,
- 5G network slicing.

For defragmentation at Logical Layer, results in Chapter 2 confirm the usefulness of the defragmentation process at high layers of networks. Therein, the first mathematical model computes a minimum bandwidth provisioning and it is not guaranteed to be MBB reachable as the model does not take into account the sequence of reroutings. We next propose a second model that computes an MBB sequence that brings the network provisioning as close as possible to the minimum bandwidth network provisioning given by the first model. Observe that the resulting network provisioning of the second model is not necessarily the minimum bandwidth network provisioning that is MBB reachable. Indeed, not only there may be several minimum bandwidth network provisioning solutions (and we arbitrarily consider one of them) with different MBB reachability status, but there also may exist better MBB reachable network provisioning with minimum bandwidth requirements. Consequently, we address those shortcomings in the next chapter.

In Chapter 3, we proposed a mathematical model that computes the MBB reachable network provisioning with minimum bandwidth requirement. This modelling not

only allows the solution of larger instances of the state of the art but also confirms the high efficiency of the heuristics proposed in Chapter 2.

Although models in Chapters 2 and 3 offer a significant leap in comparison with the literature, they have some limitations as we do not consider parallel and multiple rerouting conditions. Thus, in Chapter 4, we develop enhanced and more complex models for the parallel-and-multiple MBB rerouting problem. Preliminary results show that parallel rerouting reduces by about ten times the algorithm's computational times and the number of required rerouting events (corresponding to the system's defragmentation real-time duration). However, it remains to implement the multiple rerouting feature to evaluate its impact on defragmentation, see future work in Section 7.2.

At Optical Layer, the defragmentation problem has more complicated conditions coming from physical devices and connections. In the networks of the current customers of Ciena, most of ROADM nodes are CD ROADM or CDC ROADM, while connections have contiguous and continuous constraints (i.e., constraints related to provisioning in elastic optical networks). In Chapter 5, results show that defragmentation in the optical layer offers less impact on the optimization of the spectrum usage if we trigger defragmentation in the same way as in the logical layer. Indeed, because of the requirements of the continuity and contiguity constraints, it is more difficult to optimize the bandwidth usage, and consequently, we need to defragment more often. Furthermore, we must investigate more thoroughly the characteristics of this layer in order to optimize its spectrum-usage and efficiently squeeze its spectrum. Besides, experimented 4-DIA CD ROADM configuration shows that we can provision a number of connections that is very close to the CDC ROADM configuration, so that, in practice, CDC ROADMs do not allow a signigicant increase of the throguhput. Network providers can use this result to select the right technology when they upgrade or deploy new network nodes.

The last part of this thesis is successful to propose an efficient nested decomposition scheme for the 5G network slicing problem. In terms of the decomposition method, it showed that the nested decomposition scheme does not lose the optimal perspective in this problem: thanks to a Lagrangian relaxation bound, we can assess the accuracy of the solutions, without developing a branch-and-price method. Regrading 5G virtual networks, the proposed framework can evaluate heuristics for

allocating resources with very dynamic virtual slice demands.

7.2 Future Work

For reoptimization at Logical Layer, we do not take into account protection schemes. Besides, a network may serve several customer service level agreements, so the network operator treats differently classified sets of connections. Regarding the size of practical systems, our proposed modelling still needs more investigation and improvement so that it can scale and evaluate meaningful sizes of instances. It also remains to implement multiple rerouting features to evaluate its impact on defragmentation.

For Optical Layer, experiments are conducted with randomly generated traffic and not networks with an optimized dimensioning. Indeed, benchmark data sets do not include optimized dimensioning with respect of the traffic matrices. However, in practice, network operators do optimize their link capacities, and the number of ports of their switching equipment. While this has usually no impact/drawback when, e.g., evaluating the performance of network provisioning algorithms, this may not be the case for evaluating the performance of defragmentation algorithms. Indeed, the mismatch between the network's dimensioning and traffic poses unavoidable blockings. As a consequence, the evaluation, so far, of defragmentation does not reflect completely its potential benefit. To be more accurate, we must study better the correlation between dimensioning and defragmentation in order to properly evaluate the efficiency of a given defragmentation strategy, i.e., both when to trigger it and how to defragment. More recently, very few work have been proposed for network dimensioning, so future work should first include network dimensioning for elastic optical networks, and then defragmentation under the assumption of a proper network dimensioning.

For the 5G network slicing problem, we can improve the proposed framework to deal with dynamic virtual network requests. For example, when a new virtual network request arrives or a virtual network needs more resources. In that case, the network operator must re-assign some resources of the current virtual slices, and this modification should be seamless to ensure the continuity of the current ongoing services.

Bibliography

[1] O. Klopfenstein, "Rerouting tunnels for MPLS network resource optimization," European Journal of Operational Research, vol. 188(1), pp. 293 – 312, 2008.

[2] O. Gerstel, P. Green, and R. Ramaswami, "Architecture for an optical network layer," IBM technical journal, 1996.

[3] A. de la Oliva, X. Li, X. P. Costa, C. J. Bernardos, P. Bertin, P. Iovanna, T. Deiss, J. Mangues, A. Mourad, C. E. Casetti, J. E. Gonzalez, and A. Azcorra, "5g-transformer: Slicing and orchestrating transport networks for industry verticals," IEEE Communications Magazine, vol. 56, pp. 78–84, 2018.

[4] N. Panwar, S. Sharma, and A. Singh, "A survey on 5g: The next generation of mobile communication," Physical Communication, vol. 18, 03 2016.

[5] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5g wireless networks: A comprehensive survey," IEEE Communications Surveys & Tutorials, vol. 18, no. 3, pp. 1617–1655, 2016.

[6] J. Comellas, L. Vicario, and G. Junyent, "Proactive defragmentation in elastic optical networks under dynamic load conditions," Photonic Network Communications, vol. 36, no. 1, pp. 26–34, 2018.

[7] B. C. Chatterjee, S. Ba, and E. Oki, "Fragmentation problems and management approaches in elastic optical networks: A survey," IEEE Communications Surveys Tutorials, vol. 20, no. 1, pp. 183–210, Firstquarter 2018.

[8] R. Sadykov, F. Vanderbeck, A. Pessoa, I. Tahiri, and E. Uchoa, "Primal heuristics for branch and price: The assets of diving methods," INFORMS Journal on Computing, vol. 31, no. 2, pp. 251 – 267, 2019.

[9] W. Carlson, "OSRP protocol," 2000. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.400.1624&rep=rep1&type=pdf

[10] Y. Lee, J. L. Roux, D. King, and E. Oki, "Path Computation Element Communication Protocol (PCEP) Requirements and Protocol Extensions in Support of Global Concurrent Optimization," IETF, RFC 5557, July 2009.

[11] D. King and A. Farrel, "A PCE-Based Architecture for Application-Based Network Operations," IETF, RFC 7491, March 2015.

[12] R. Wang and B. Mukherjee, "Provisioning in elastic optical networks with non-disruptive defragmentation," in Conference on Optical Network Design and Modeling - ONDM, December 2013, pp. 1–6.

[13] N. Jose and K. Somani, "Connection rerouting/network reconfiguration," in Proceedings of IEEE/VDE Workshop on Design of Reliable Communication Networks - DRCN, Banff, Alberta, Canada, October 2003, pp. 23 – 30.

[14] F. Solano and M. Pióro, "Lightpath reconfiguration in WDM networks," IEEE/OSA Journal of Optical Communications and Networking, vol. 2, no. 12, pp. 1010 – 1021, December 2010.

[15] G. Adelson-Velsky and E. Levner, "Mathematics of operations research," AT&T Bell Laboratories Technical Journal, vol. 27, pp. 504 – 517, August 2002.

[16] P. Chakrabarti and S. Ghose, "A general best first search algorithm in AND/OR graphs," Journal of Algorithms, vol. 13, pp. 177 – 187, June. 1992.

[17] B. Ramamurthy and A. Ramakrishnan, "Virtual topology reconfiguration of wavelength-routed optical wdm networks," in IEEE Global Telecommunications Conference - GLOBECOM, vol. 2, 2000, pp. 1269–1275.

[18] D. Banerjee and B. Mukherjee, "Wavelength routed optical networks: Linear formulation, resource budgeting tradeoffs, and reconfiguration study," in INFOCOM, Kobe, Japan, 1997.

[19] R. Ramaswami and K. Sivarajan, "Design of logical topologies for wavelength-routed optical networks," IEEE Journal of Selected Areas in Communications, vol. 14, no. 5, pp. 840–851, June 1996.

[20] S. Raghavan and D. Stanojević, "Branch and price for WDM optical networks with no bifurcation of flow," INFORMS J. on Computing, vol. 23, no. 1, pp. 56–74, Jan. 2011.

[21] L. Maggi, P.-L. Poirion, and J. Leguay, "Reroute backward to better break deadlocks," in IEEE International Conference on Cloud Networking (CloudNet), 2017, pp. 1–6.

[22] M. Tajiki, B. Akbari, and N. Mokari, "Optimal Qos-aware network reconfiguration in software defined cloud data centers," Computer Networks, vol. 120, pp. 71 – 86, 2017.

[23] D. Wang and G. Li, "Efficient distributed bandwidth management for MPLS fast reroute," IEEE/ACM Transactions on Networking, vol. 16, no. 2, pp. 71 – 86, April 2008.

[24] B. Józsa and M. Makai, "On the solution of reroute sequence planning problem in MPLS networks," Computer Networks, vol. 42(2), pp. 199–210, 2003.

[25] E. Olinick and T. Rahman, "Incremental demand rerouting: Optimization models and algorithms," Telecommunication Systems, vol. 46, no. 1, pp. 61–80, 2011.

[26] B. Józsa, D. Orincsay, and L. Tamási, Multi-hour Design of Dynamically Reconfigurable MPLS Networks, ser. Lecture Notes in Computer Science (LNCS). Berlin: Springer, 2004, vol. 3042, pp. 502–513.

[27] C. Barnhart, E. Johnson, G. Nemhauser, M. Savelsbergh, and P. Vance, "Branch-and-price: Column generation for solving huge integer programs," Operations Research, vol. 46, no. 3, pp. 316–329, 1998.

[28] M. Lübbecke and J. Desrosiers, "Selected topics in column generation," Operations Research, vol. 53, pp. 1007–1023, 2005.

[29] V. Chvatal, Linear Programming. Freeman, 1983.

[30] I. Gurobi Optimization, Gurobi Optimizer Reference Manual, 2017. [Online]. Available: http://www.gurobi.com

[31] J. Akinpelu, "The overload performance of engineered networks with nonhierarchical and hierarchical routing," AT&T Bell Laboratories Technical Journal, vol. 63, pp. 1261 – 1281, Sept. 1984.

[32] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," IEEE Access, vol. 3, pp. 2542–2553, 2015.

[33] D. Coudert, F. Huc, D. Mazauric, N. Nisse, and J.-S. Sereni, "Reconfiguration of the routing in WDM networks with two classes of services," in Conference on Optical Network Design and Modeling - ONDM, 2009, pp. 1–6.

[34] N. Cohen, D. Coudert, D. Mazauric, N. Nepomuceno, and N. Nisse, "Tradeoffs in process strategy games with application in the WDM reconfiguration problem," Theoretical Computer Science, vol. 412, no. 35, pp. 4675–4687, 2011.

[35] N. Jose and A. K. Somani, "Connection rerouting/network reconfiguration," in Proceedings of IEEE/VDE Workshop on Design of Reliable Communication Networks - DRCN, 2003, pp. 23–30.

[36] F. Solano, "Analyzing two conflicting objectives of the WDM lightpath reconfiguration problem," in IEEE Global Telecommunications Conference - GLOBECOM, Nov. 2009, pp. 1–7.

[37] A. Kadohata, A. Hirano, F. Inuzuka, A. Watanabe, and O. Ishida, "Wavelength path reconfiguration design in transparent optical WDM networks," IEEE/OSA Journal of Optical Communications and Networking, vol. 5, no. 7, pp. 751 – 761, July 2013.

[38] D. Coudert, D. Mazauric, and N. Nisse, "Experimental evaluation of a branch-and-bound algorithm for computing pathwidth and directed pathwidth," ACM Journal of Experimental Algorithmics, vol. 21, no. 1.3, pp. 1–23, 2016.

[39] S. Beker, D. Kofman, and N. Puech, "Off-line MPLS layout design and reconfiguration: Reducing complexity under dynamic traffic conditions," in International Network Optimization Conference (INOC), October 2003, pp. 61–66.

[40] D. Coudert, D. Mazauric, and N. Nisse, "On rerouting connection requests in networks with shared bandwidth," Electronic Notes in Discrete Mathematics, vol. 32, pp. 109–116, 2009.

[41] S. Brandt, K.-T. Förster, and R. Wattenhofer, "On consistent migration of flows in SDNs," in IEEE Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM, 2016, pp. 1–9.

[42] D. Coudert and J.-S. Sereni, "Characterization of graphs and digraphs with small process number," Discrete Applied Mathematics, vol. 159, no. 11, pp. 1094–1109, Jul. 2011.

[43] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, Introduction to Algorithms, 2nd ed. Cambridge: The MIT Press, 2001.

[44] R. Ahuja, T. Magnanti, and J. Orlin, Network Flows: Theory, Algorithms and Applications. Prentice Hall, 1993.

[45] J. Lai, Q. Fu, and T. Moors, "Using SDN and NFV to enhance request rerouting in ISP-CDN collaborations," Computer Networks, vol. 113(1), pp. 176–187, Fev. 2017.

[46] H. Duong, B. Jaumard, D. Coudert, and R. Armolavicius, "Efficient Make Before Break Capacity Defragmentation," in IEEE International Conference on High Performance Switching and Routing. Bucharest, Romania: IEEE, Jun. 2018, p. 6. [Online]. Available: https://hal.inria.fr/hal-01930552

[47] B. Jaumard, H. Quang Duong, R. Armolavicius, T. Morris, and P. Djukic, "Efficient real-time scalable make-before-break network re-routing," Journal of Optical Communications and Networking, vol. 11, p. 52, 03 2019.

[48] F. Vanderbeck, "A nested decomposition approach to a three-stage, two-dimensional cutting-stock problem," Management Science, vol. 47, no. 6, pp. 864–879, 2001.

[49] H. Duong and B. Jaumard, "A nested decomposition model for reliable nfv 5g network slicing," in International Network Optimization Conference, Avignon, France, June 2019. [Online]. Available: https://openproceedings.org/2019/conf/inoc/INOC_2019_paper_38.pdf

[50] A. Pessoa, R. Sadykov, E. Uchoa, and F. Vanderbeck, "Automation and combination of linear-programming based stabilization techniques in column generation," INFORMS Journal on Computing, vol. 30, no. 2, pp. 339–360, 2018.

[51] V. López, L. Velasco et al., "Elastic optical networks," Architectures, Technologies, and Control, Switzerland: Springer Int. Publishing, 2016.

[52] S. Talebi, F. Alam, I. Katib, M. Khamis, R. Salama, and G. N. Rouskas, "Spectrum management techniques for elastic optical networks: A survey," Optical Switching and Networking, vol. 13, pp. 34 – 48, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1573427714000253

[53] A. A. Hussien and A. H. Ali, "Comprehensive investigation of coherent optical ofdm-rof employing 16qam external modulation for long-haul optical communication system," International Journal of Electrical and Computer Engineering (IJECE), vol. 10, no. 3, pp. 2607–2616, 2020.

[54] A. Devarajan, K. Sandesha, R. Gowrishankar, B. S. Kishore, G. Prasanna, R. Johnson, and P. Voruganti, "Colorless, directionless and contentionless multi-degree roadm architecture for mesh optical networks," in 2010 Second International Conference on COMmunication Systems and NETworks (COMSNETS 2010). IEEE, 2010, pp. 1–10.

[55] G. Li, K. Yan, L. Huang, B. Xia, F. Kong, and Y. Li, "How much is cd roadm contention blocking?" in Optical Fiber Communication Conference. Optical Society of America, 2018, p. W4A.5. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=OFC-2018-W4A.5

[56] D. Moniz, A. Eira, A. de Sousa, and J. Pires, "On the comparative efficiency of non-disruptive defragmentation techniques in flexible-grid optical networks," Optical Switching and Networking, vol. 25, pp. 149 – 159, 2017.

[57] J. M. Simmons, Optical Network Design and Planning, 2nd ed. Springer, 2014.

[58] B. C. Collings, "Advanced roadm technologies and architectures," in 2015 Optical Fiber Communications Conference and Exhibition (OFC), 2015, pp. 1–3.

[59] G. Li, K. Yan, L. Huang, B. Xia, F. Kong, and Y. Li, "How much is cd roadm contention blocking?" in 2018 Optical Fiber Communications Conference and Exposition (OFC), March 2018, pp. 1–3.

[60] P. Pavon-Marino and M. Bueno-Delgado, "Dimensioning the add/drop contention factor of directionless ROADMs," Journal of Lightwave Technology, vol. 29, no. 21, pp. 3265–3274, November 2011.

[61] A. N. Patel, P. N. Ji, J. P. Jue, and Ting Wang, "Defragmentation of transparent flexible optical wdm (fwdm) networks," in 2011 Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference, 2011, pp. 1–3.

[62] S. Fernández-Martínez, B. Barán, and D. P. Pinto-Roa, "Spectrum defragmentation algorithms in elastic optical networks," Optical Switching and Networking, vol. 34, pp. 10–22, 2019.

[63] Boost c++ library. Version 1.75. [Online]. Available: http://www.boost.org

[64] M. Batayneh, D. Schupke, M. Hoffmann, A. Kirstaedter, and B. Mukherjee, "On routing and transmission-range determination of multi-bit-rate signals over mixed-line-rate WDM optical networks for carrier ethernet," IEEE/ACM Transactions on Networking, vol. 19, pp. 1304–1316, October 2011.

[65] I. Gurobi Optimization, Gurobi Optimizer Reference Manual, 2019. [Online]. Available: http://www.gurobi.com

[66] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. Ramos-Muñoz, J. Lorca, and J. Folgueira, "Network slicing for 5G with SDN/NFV: concepts, architectures and challenges," IEEE Communications Magazine, vol. 55, pp. 80–87, 2017.

[67] M. Bonfim, K. Dias, and S. Fernandes, "Integrated nfv/sdn architectures: A systematic literature review," Journal ACM Computing Surveys (CSUR), vol. 51, no. 6, pp. xxx – xxx, 2019.

[68] B. Yi, X. Wang, K. Li, S. Das, and M. Huan, "A comprehensive survey of network function virtualization," Computer Networks, vol. 133, pp. 212 – 262, 2018.

[69] L. Tang, G. Zhao, C. Wang, P. Zhao, and Q. Chen, "Queue-aware reliable embedding algorithm for 5G network slicing," Computer Networks, vol. 146, pp. 138 – 150, December 2018.

[70] X. Li, M. Samaka, H. Chan, D. Bhamare, L. Gupta, C. Guo, and R. Jain, "Network slicing for 5G: Challenges and opportunities," IEEE Internet Computing, vol. 21, no. 5, pp. 20–27, 2017.

[71] R. Lin, S. Luo, J. Zhou, S. Wang, B. Chen, X. Zhang, A. Cai, W.-D. Zhong, and M. Zukerman, "Column generation algorithms for virtual network embedding in flexi-grid optical networks," Optics Express, vol. 26, no. 8, pp. 10 898–10 913, Apr 2018.

[72] A. Destounis, G. Paschos, S. Paris, J. Leguay, L. Gkatzikis, S. Vassilaras, M. Leconte, and P. Medagliani, "Slice-based column generation for network slicing," in Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM. Honolulu, HI, USA: IEEE, April 2018, pp. 1–2.

[73] G. Carella, M. Pauls, A. Medhat, L. Grebe, and T. Magedanz, "A network function virtualization framework for network slicing of 5G networks," in Mobilkommunikation–Technologien und Anwendungen. Osnabrück, Deutschland: ITG-Fachtagung, 2017, pp. 1–7.

[74] S. Song, "A nested column generation algorithm to the meta slab allocation problem in the steel making industry," Journal International Journal of Production Research, vol. 47, no. 13, pp. 3625–3638, 2009.

[75] A. Dohn and A. Mason, "Branch-and-price for staff rostering: An efficient implementation using generic programming and nested column generation," European Journal of Operational Research, vol. 230, pp. 157–169, 2013.

[76] S. Karabuk, "A nested decomposition approach for solving the paratransit vehicle scheduling problem," Transportation Research Part B, vol. 43, pp. 448–465, 2009.

[77] M. E. Cóccola, C. A. Méndez, and R. G. Dondo, "Optimizing the inventorying and distribution of chemical fluids: An innovative nested column generation approach," Computers & Chemical Engineering, vol. 119, pp. 55 – 69,

2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0098135418302849

[78] F. Hennig, B. Nygreen, and M. Lübbecke, "Nested column generation applied to the crude oil tanker routing and scheduling problem with split pickup and split delivery," Naval Research Logistics, vol. 59, pp. 298–310, April - June 2012.

[79] L. Lasdon, Optimization Theory for Large Systems. New York: MacMillan, 1970.

[80] J. Gauthier, J. Desrosiers, and M. Lübbecke, "Vector space decomposition for solving large-scale linear programs," Operations Research, vol. 66, no. 5, pp. 1376–1389, 2018.

[81] G. L. Nemhauser and L. A. Wolsey, Integer and Combinatorial Optimization. New York: Wiley, 1988.

[82] N. Huin, B. Jaumard, and F. Giroire, "Optimal network service chain provisioning," IEEE/ACM Transactions on Networking, vol. 26, no. 3, pp. 1320–1333, June 2018.

[83] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," in Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium, April 2007, pp. 276–286.