Optimal Design of Survivable Multi-layer Telecommunication Networks

vorgelegt von
Diplom-Wirtschaftsmathematiker
Sebastian Orlowski
aus Hamburg

Von der Fakultät II – Mathematik und Naturwissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
Dr. rer. nat.

genehmigte Dissertation

Promotionsausschuss:
Vorsitzender: Prof. Dr. Fredi Tröltzsch
Berichter: Prof. Dr. Martin Grötschel
Prof. Dr. Michał Pióro

Tag der wissenschaftlichen Aussprache: 4. Mai 2009

Berlin 2009
D 83

Zusammenfassung

Telekommunikationsnetze bestehen aus verschiedenen technologischen Schichten, sogenannten *Layern*, die eng miteinander verknüpft sind. Ein Beispiel für ein solches Netz ist ein IP-Kernnetz, bei dem Verbindungen zwischen Internet-Routern durch Lichtwege in einem zugrundeliegenden optischen Glasfasernetz realisiert werden. Um sicherzustellen, dass das Netz in jeder Situation alle anfallenden Kommunikationsbedarfe routen kann, müssen die Abhängigkeiten zwischen den verschiedenen Schichten schon bei der Planung des Netzes berücksichtigt werden. Dies ist vor allem dann von Bedeutung, wenn Verbindungen in einer Schicht gegen Kabel- oder Geräteausfälle in einer anderen Schicht geschützt werden müssen. Der traditionelle sequentielle Ansatz, bei dem eine Schicht nach der anderen optimiert wird, kann solche Abhängigkeiten nicht ausreichend berücksichtigen. Dies ist nur mit einem integrierten Planungsansatz möglich, bei dem mehrere Schichten gemeinsam geplant werden.

Diese Arbeit behandelt mathematische Modelle und Lösungsverfahren für die integrierte Optimierung zweier zusammenhängender Netzschichten mit Ausfallsicherheitsanforderungen. Wir stellen ein Multi-layer-Netzplanungsproblem vor, das in mehreren Technologien auftritt, und modellieren es in Form von gemischt-ganzzahligen Optimierungsmodellen (MIPs). Diese Modelle decken viele praktisch relevante Nebenbedingungen aus verschiedenen Technologien ab. Im Gegensatz zu vorherigen Modellen aus der Literatur können sie zur Lösung großer ausfallsicherer Zwei-layer-Netze verwendet werden. Im Zusammenhang damit diskutieren wir verschiedene Modellierungsoptionen für wesentliche Teile eines Multi-layer-Netzes.

Wir lösen unsere Modelle mit einem Branch-and-cut-and-price-Verfahren in Verbindung mit verschiedenen problemspezifischen Techniken. Dies umfasst Presolving-Techniken zur Reduktion der Problemgröße, kombinatorische und MIP-basierte Primalheuristiken zur Berechnung von Netzkonfigurationen, spezielle Schnittebenen für Ausfallsicherheit über mehrere Schichten hinweg zur Verbesserung der unteren Schranke an die optimalen Netzkosten, sowie Spaltengenerierung zur dynamischen Erzeugung von Flussvariablen während des Algorithmus. Darüber hinaus entwickeln wir Verfahren, um Rechnungen mit einem Benders-Dekompositionsansatz zu beschleunigen.

Die entwickelten Techniken werden verwendet, um große ausfallsichere Zwei-layer-Netze mit Methoden der linearen und ganzzahligen Optimierung zu planen. Wir evaluieren unsere Techniken auf realistischen Testinstanzen mit bis zu 67 Netzknoten und Ausfallsicherheitsanforderungen und berechnen mit ihrer Hilfe gute Netzkonfigurationen mit Qualitätsgarantien. Die meisten unserer Testinstanzen mit bis zu 17 Knoten können wir nahezu optimal lösen. Darüber hinaus können wir selbst für große ausfallsichere Netze Lösungen und aussagekräftige untere Schranken für die optimalen Netzkosten berechnen, was bisher nicht möglich war.

Schlüsselworte: Telekommunikationsnetze, Planung von Multi-layer-Netzen, Ausfallsicherheit, gemischt-ganzzahlige Optimierung, Branch-and-cut-and-price
Mathematics Subject Classification (2000): 90C34, 90C11, 68M10

Abstract

Telecommunication transport networks consist of a stack of technologically different subnetworks, so-called *layers*, which are strongly interdependent. For example, one layer may correspond to an Internet (IP) backbone network whose links are realized by lightpath connections in an underlying optical fiber layer. To ensure that the network can fulfill its task of routing all communication requests, the inter-layer dependencies have to be taken into account already in the planning phase of the network. This is particularly important with survivability constraints, where connections in one layer have to be protected against cable cuts or equipment failures in another layer. The traditional sequential planning approach where one layer is optimized after the other cannot properly take care of the inter-layer dependencies; this can only be achieved with an integrated planning of several network layers at the same time.

This thesis provides mathematical models and algorithmic techniques for the integrated optimization of two network layers with survivability constraints. We describe a multi-layer network design problem which occurs in various technologies, and model it mathematically using mixed-integer programming (MIP) formulations. The presented models cover many important practical side constraints from different technological contexts. In contrast to previous models from the literature, they can be used to design large two-layer networks with survivability requirements. We discuss modeling alternatives for various aspects of a multi-layer network and compare different routing formulations under multi-layer survivability constraints.

We solve our models using a branch-and-cut-and-price approach with various problem-specific enhancements. This includes a presolving technique based on linear programming to reduce the problem size, combinatorial and sub-MIP-based primal heuristics to compute feasible network configurations, cutting planes which take the multi-layer survivability constraints into account to improve the lower bound on the optimal network cost, and column generation to generate flow variables dynamically during the algorithm. We develop techniques to speed up computations in a Benders decomposition approach and compare this approach to the standard formulation with a single MIP.

We use the developed techniques to design large survivable two-layer networks by means of linear and integer programming methods. On realistic test instances with up to 67 network nodes and survivability constraints, we investigate the algorithmic impact of our techniques and show how to use them to compute good network configurations with quality guarantees. Most of the smaller test instances with up to 17 nodes can be solved to near-optimality. Moreover, we can compute feasible solutions and dual bounds even for large networks with survivability constraints, which has not been possible before.

Keywords: telecommunication networks, survivable multi-layer network design, mixed-integer programming, branch-and-cut-and-price

Mathematics Subject Classification (2000): 90C34, 90C11, 68M10

Acknowledgments

The research presented in this thesis has been conducted within the MATHEON project *Integrated Planning of Multi-layer Networks* and various projects with industrial partners. Several colleagues from ZIB and other institutions and companies have been involved in these projects, and many people have contributed to this thesis by supporting me in different ways.

First of all, I would like to thank my colleagues Roland Wessäly, Arie Koster, and Andreas Bley for collectively guiding my research and heading the projects in which I was involved. Their different views on the considered planning problems very much contributed to my understanding of multi-layer networks and different working techniques. I would also like to thank Martin Grötschel, who gave me the necessary freedom to work on this topic both at ZIB and at other places all over Europe during several research visits and conferences. Adrian Zymolka and Christian Raack, who shared the office with me during the last years, made me enjoy my work at ZIB, as well as many other colleagues.

During the first phase of my thesis, conducted together with Roland Wessäly, I learned a lot about the technical aspects of multi-layer networks from our discussion with partners from T-Systems Nova and E-Plus. Especially Fritz-Joachim Westphal, Ralf Hülsermann, Monika Jäger, and Peter Jonas guided me through the jungle of abbreviations and different technological concepts appearing in multi-layer networks. They helped me to get at least a basic understanding of of the various technologies involved, and also gave us feedback on our mathematical models.

Following this understanding and modeling phase, it was time to develop suitable algorithms to actually design some multi-layer networks. Luckily, I ran across Pietro Belotti at the Optimization 2004 conference in Lisbon. Realizing that we were working on very similar problems, we decided to join forces, and started developing a branch-and-cut-and-price algorithm for two-layer network planning during nice and fruitful mutual research visits in Milan, Berlin, and Paris. Many ideas and the code emerging from this cooperation later served as a basis for the framework used to develop the algorithmic methodology presented in this thesis.

In early 2006, our working group at ZIB launched a project together with Siemens on the integrated planning of two-layer transport networks. This project was a great chance for me: first, it gave me a specific practical planning problem to solve, and second, this planning problem perfectly extended the one I was investigating with Pietro. In a first phase, the goal was to improve the dual bounds of a standard mixed-integer programming solver in order to reduce the enormous running times even for small networks. During this year, Arie Koster, Christian Raack, and myself developed some of the presolving techniques and cutting planes presented in this thesis. As this approach was quite successful, our cooperation partners (who were now at Nokia-Siemens Networks) agreed on a follow-up project a few months later. As Arie had left to Warwick in the meantime and Christian was absorbed by other activities, I found myself being the principal worker in this project with some guidance from Andreas

Bley. Unfortunately (or fortunately, from the point of view of this thesis), our partners got more demanding: instead of the 17-node networks investigated before, they now wanted to solve two-layer networks with more than 50 nodes and survivability constraints, a network size which is already hard to deal with in single-layer networks! Although we were not sure whether this would be a good idea, we eventually agreed on this challenge. Luckily, the idea turned out to be a good one. First, we found that linear and mixed-integer programming techniques can to some extent even be applied to such large multi-layer networks, and second, this project allowed me to further understand the mathematical structures behind multi-layer networks and to push forward our algorithmic methodology. Many of the results presented in this thesis are based on work performed within and between these two projects. I would thus take the opportunity to thank all my co-workers at ZIB as well as our cooperation partners Thomas Engel, Georg Baier, and Achim Autenrieth for the really nice and fruitful working atmosphere! Special thanks goes to Thomas Engel for his permission to use the test instances from these projects in my thesis.

During my implementation work, I received much support from the SCIP developers at ZIB. I am deeply thankful to the whole SCIP team, most notably Tobias Achterberg and Kati Wolter, for their seemingly infinite patience in assisting me to trace bugs (both in SCIP and in my own code) and in bearing me coming into their offices all the time to ask yet another question about the behavior of SCIP or CPLEX in some strange situation. Especially Tobias has helped me a lot by discussing details of linear programming and the simplex algorithm with me in a depth which I had never thought to need again after my university courses, and by giving me hints on useful parameter settings.

Despite my focus on multi-layer networks, I have always been continuing to work on survivable single-layer networks as well. Even though the results from this work are not, strictly speaking, part of this thesis, they are related to its topic and have contributed to my better understanding of some aspects of multi-layer networks as well. In this context, I would like to thank Michał Pióro and Artur Tomaszewski for our nice common work on the Survivable Network Design Library SNDLIB and the associated paper. These activities and another joint paper on survivability concepts in single-layer networks gave me the opportunity to visit them in Warsaw twice, which I very much enjoyed.

During the final phase of this thesis, many people have helped to improve it by pointing out errors or giving useful comments. I am particularly obliged to Christian, Andreas, and Roland, who invested quite some time to read large parts of this thesis, especially those focusing on mathematical and algorithmic aspects. Thanks also go to Thomas Engel, who kindly commented on my technical description of multi-layer networks, and to a number of friends who have read, or offered to read, the non-mathematical parts (and sometimes even more) to tell me at which places I had failed to explain things properly. This includes Mathias Schulz, Ines Spenke, Julius Kusserow, Filip Idzikowski, Alexander Kröller, and my beloved, Karola. The latter deserves a special thanks for taking me out to all kinds of activities even during the final phase of my thesis, and for just being there whenever I needed her.

Contents

Chapter 1

Introduction

Telecommunication transport networks consist of a stack of subnetworks with different technologies, so-called *layers*. Every link in an upper layer is realized by one or more paths in the next lower layer. For example, an IP link between two Internet routers may be realized by one or more lightpaths in an underlying optical fiber network, as illustrated in Figure 1.1.

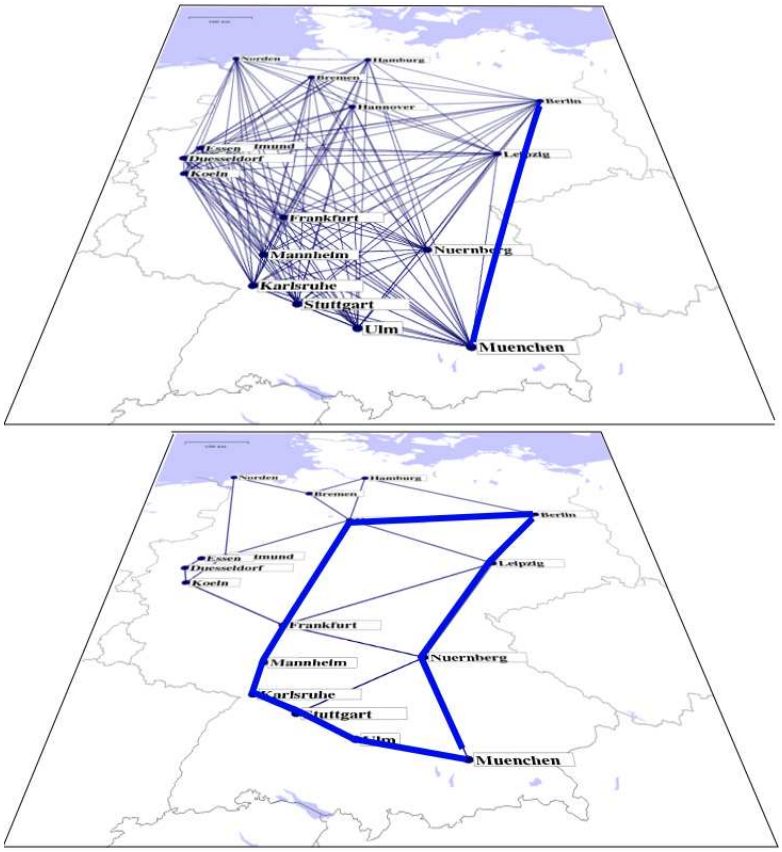


Figure 1.1: A two-layer network. The upper layer might correspond to an Internet backbone IP network and the lower layer to an optical fiber network. The IP connection between two Internet routers in Berlin and Munich is realized by the two lightpaths in the underlying fiber network between the same nodes. All traffic sent on this seemingly direct link actually travels via one of the two paths.

To route communication demands through the network, sufficient capacity must be installed on the links of all layers, and the nodes must be equipped with suitable hardware. Classical network design methods focus on one of these layers only. Different layers of the stack have been planned sequentially by a series of single-layer network design problems, where the planning output of one layer serves as planning input for the next layer.

The drawback of the sequential planning approach is that it cannot properly take the strong interdependencies between the various layers into account. As a result, it may lead to expensive network configurations, or even fail to construct a network which satisfies all side constraints. This is particularly important with survivability constraints, where connections of one layer have to be protected against cable cuts or equipment failures in another layer. Survivability requirements are of major practical importance, but make the design of a multi-layer network much more difficult. An integrated planning process for several network layers at the same time can better take such dependencies into account, and thus reduce network cost and the impact of equipment failures.

This thesis provides mathematical models and algorithmic techniques for the integrated planning of two network layers with survivability constraints. We describe the practical planning problem and model it mathematically using mixed-integer programming (MIP) formulations. The presented models cover many practical side constraints and can be used to design even large networks with survivability requirements. Because of the enormous computation times even for small networks, however, it is not possible to just apply an out-of-the-box MIP solver. We thus employ a branch-and-cut-and-price approach enhanced by a variety of problem-specific techniques. In particular, this includes presolving techniques to reduce the problem size, primal heuristics to compute feasible network configurations, cutting planes to compute lower bounds on the minimum possible network cost, and column generation to generate variables dynamically for large networks. On realistic test networks with up to 67 nodes, we investigate the algorithmic impact of our techniques and show how they can be used to compute provably good network configurations.

The work presented in this thesis is based on the MATHEON project *Integrated planning of multi-layer networks* [112]. This work has been performed in cooperation with various industrial partners, in particular E-Plus, T-Systems Nova, and Nokia-Siemens Networks. It turned out that although these network operators and equipment vendors deal with different technologies, the mathematical structures behind their planning problems are similar. Consequently, the models and solution methods presented in this thesis focus on the common kernel of these planning problems, but have been developed with their applicability in different technological contexts in mind.

In the following section, we briefly explain the basic idea of a multi-layer network and describe the decisions involved in the design of such a network. Section 1.2 summarizes the most important contributions of this thesis. Eventually, Section 1.3 gives an overview on the following chapters.

1.1 Introduction to multi-layer network design

In this section, we will briefly explain the concept of network design in the context of single-layer networks, and extend it to multi-layer networks afterwards.

1.1.1 Network design

Modern telecommunication transport networks accommodate various kinds of traffic. This includes, but is not limited to, IP traffic from the Internet, traffic from dedicated virtual private networks of large companies, video traffic, and voice traffic from fixed-line or mobile telephone calls. To route all this data through the network, the network nodes have to be equipped with routing and switching hardware to receive traffic on an incoming link and send it out on some outgoing link. Furthermore, the links between these nodes must provide sufficient capacity (reserved bandwidth) to accommodate all requested connections. Usually, this capacity can only be installed in discrete steps.

When building such a network, or when adapting an existing network to changing demands, the network operator faces a *network design problem*. Network design problems occur in many different flavors depending on the specific technologies and side constraints. Stated in a general form, such a problem consists of choosing a set of nodes with links between them (a so-called *network topology*), installing capacities on the links and hardware at the nodes, and routing the traffic demands of all customers within these capacities. During this design process, a variety of hardware compatibility constraints and routing constraints has to be taken into account. In order to protect the traffic against hardware failures or cable cuts, spare capacity may have to be installed in the network in such a way that affected traffic can be rerouted around the failing node or link. This is illustrated in Figure 1.2. Many optimization goals are possible, such as minimizing total installation cost or accommodating as much traffic as possible in a given network configuration.

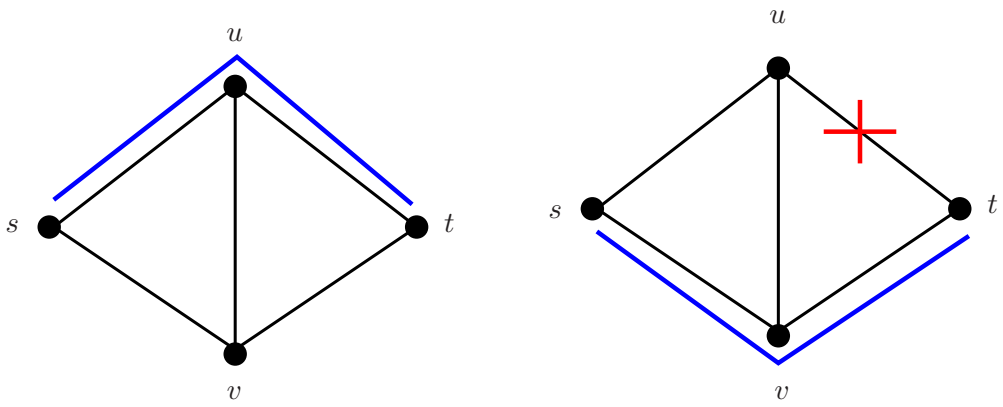


Figure 1.2: Survivability by 1:1 protection: normally, the flow from s to t is routed on the path s–u–t. If this path fails, its flow is switched to the disjoint backup path s–v–t, which must have been equipped with sufficient capacity during the network design phase. Many other possible rerouting schemes have been proposed in the literature.

Mathematical models and algorithms to solve such network design problems have been extensively studied during the last twenty years; see Grover [62], Pióro and Medhi [109], and references therein. Much progress has been made in this area, and many large network design problems can nowadays be solved to a degree which is sufficient for practical applications. Computing an *optimal* network configuration and proving its optimality, on the other hand, can still be a challenge.

1.1.2 Multi-layer network design

Introduction to multi-layer networks

Unfortunately, real telecommunication networks are more complicated than just described. They consist of a complex hierarchy of *layers*. For instance, an IP link in a nationwide backbone network is usually not realized as a direct connection in the form of an Ethernet cable or a radio link. Instead, it is realized by one or more paths in an underlying transport layer, such as lightpath connections in an optical fiber network. When focusing on two adjacent network layers from the hierarchy, a link realized by a path in some underlying transport layer is called a *logical link* (or *virtual link*), while the links of the realizing path in the transport layer are called *physical links*.

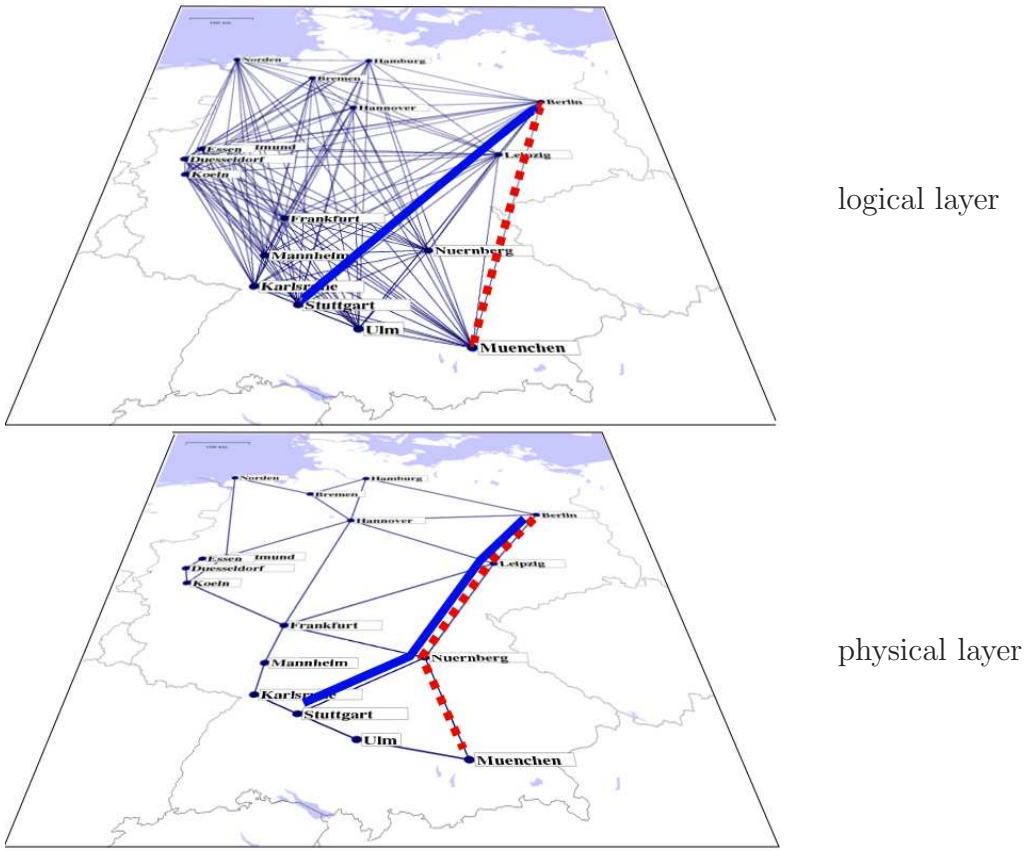


Figure 1.3: A two-layer German backbone network. The red (dashed) logical link Berlin–Munich is realized by a physical path between these nodes via Leipzig and Nuremberg. The blue (solid) logical link Berlin–Stuttgart is realized by a path from Berlin via Leipzig and Nuremberg to Stuttgart. A communication demand from Munich to Stuttgart may be routed on a path in the logical layer from Munich via Berlin to Stuttgart. It cannot leave the physical paths at intermediate nodes.

Such a two-layer network is illustrated in Figure 1.3. It shows an upper logical layer based on an underlying physical layer. Both layers are based on the same set of nodes, corresponding to important network nodes in a German backbone network. In an optical networking context, the lower physical layer represents a fiber network where each link in the upper logical layer

is realized by a lightpath connection between the corresponding end-nodes in the fiber layer. In Figure 1.3, the blue (solid) logical link between Berlin and Stuttgart is realized by a path between these cities via Leipzig and Nuremberg. The red (dashed) logical link between Berlin and Munich is implemented by the path Berlin–Leipzig–Nuremberg–Munich. A demand from Munich to Stuttgart may be routed via Berlin on a path consisting of these two logical links. It is *not* possible for the demand to quit the physical paths and to take the shortcut Munich–Nuremberg–Stuttgart, for instance; all data entering a logical link at its source node must travel the whole link until its target node. Usually, the requested bandwidth of a communication demand is much lower than the capacity provided by a logical link, such that a logical link can accommodate the routing paths of many demands simultaneously.

The figure also illustrates the potential impact of physical failures on the logical layer. Both logical links Berlin–Munich and Berlin–Stuttgart traverse the node Leipzig. Hence, an equipment failure at Leipzig causes both logical links to break down simultaneously, as well as all connections between end users which are routed through any of these links, such as the connection Munich–Stuttgart. When looking at the logical layer alone, these links seem to be disjoint and independent of node Leipzig. Similar considerations hold if a cable cut occurs between Berlin and Leipzig. This example shows that routing decisions in the upper layer must take the realization of logical links in the underlying physical layer into account to achieve survivability against equipment failures.

Key concepts of a multi-layer network

The concept of a multi-layer network is not limited to the IP-over-optical network setting described above. Many transport network technologies and protocols, like Multi-Protocol Label Switching (MPLS), Synchronous Digital Hierarchy (SDH), Asynchronous Transport Module (ATM), or Wavelength Division Multiplexing (WDM), have inherent multi-layer properties following from their technical specification; this is explained in more detail in Section 2.1.2. In particular, the "physical" links of a transport layer may in turn be realized by paths in another underlying network layer. A real network may consist of a complex stack of seven or more layers if all sublayers within a technology are taken into account.

A major advantage of the layered structure of telecommunication networks is that it simplifies hardware and network management at the network nodes. For example, a cross-connect installed at a node of an optical fiber network only forwards aggregated lightpath signals of several Gbit/s from an input port to an output port by means of mirrors or in other ways. It need not convert the optical signal into an electrical one and inspect individual data packets embedded in this signal. This only happens at IP routers at the end-nodes of each lightpath. An IP router, on the other hand, only needs to know that there exists a link with a certain capacity to another router. It need not care how this link is routed in the underlying transport network and what transmission technology is used to realize it.

Three key concepts are common to all multi-layer telecommunication networks, independent of the specific technology:

1. Demands at the logical layer may be routed on paths consisting of logical links; each of these links corresponds to a one or more paths in some underlying physical layer.

2. Capacities are defined not only on physical links but also on logical links, i. e., on paths in the underlying physical layer. In most cases, capacities can only be installed in discrete steps which are much larger than the requested bandwidth of an individual demand.

3. At each end-node of a logical link, traffic can be extracted from incoming aggregated signals and recombined into new aggregated signals which are sent out on outgoing logical links. This process, called *grooming*, usually incurs a cost (e. g., for hardware) and is limited by a node capacity.

These key concepts also occur in other planning contexts outside telecommunication where transported goods (or passengers or data) are recombined at network nodes to be routed together for some distance, and integral capacities are installed not only on links, but also on subpaths. We would like to illustrate this by two examples:

- In a bus line planning problem in public transport networks [31], the physical links correspond to connections between bus stations and the logical links to bus lines or parts thereof. At each station, passengers are extracted from the buses and recombined to new bus loads, continuing their way on another bus line. Buses, bus lines, and stations have a limited capacity.
- In railroad blocking [6, 16], goods have to be transported between cities via a railroad network. The physical links correspond to railroad lines and the logical links to segments on which parts of different trains are routed together for some distance. At various places in the network, trains can be split and recombined according to their destination. These places have a limited capacity and require costly infrastructure.

While the mathematical models presented in the above mentioned literature are similar to the ones presented in this thesis, the side constraints on equipment and routing vary widely among the different applications. Even though the central concepts are more general, we therefore restrict ourselves to telecommunication networks in this thesis.

The two-layer network design problem

We develop mathematical models and algorithmic techniques to design two adjacent network layers in an integrated step. The *two-layer network design problem* as considered in this thesis can be summarized as follows:

Given are

- a set of potential network nodes and links on both layers, and information on how the admissible logical links may be realized in the physical layer,
- installable node hardware and link capacities in both layers, and
- traffic demands at the logical layer with corresponding routing constraints, such as survivability requirements or path length restrictions.

A **feasible network configuration** consists of

- a network topology on both layers chosen from the available nodes and links,
- a realization of the logical links in the physical layer,
- a node hardware and link capacity installation at both layers, and
- a routing of the demands within the installed logical link capacities.

The **planning goal** is to minimize the total installation cost of node hardware, logical and physical link capacities.

Optimally designing several layers in an integrated step has not been possible until recently due to a lack of suitable mathematical models, algorithms, and computation power. A common approach in practice has thus been to decompose the multi-layer planning problem into a series of single-layer planning problems as described in the previous section: first, a topology, capacities, and a routing are planned in the topmost layer; the capacities of this layer then have to be routed as demands in the next lower layer, capacities have to be determined for this routing, and so on. In particular, the demand graph, which contains a link between every pair of end-nodes of a communication demand, can be seen as an additional layer on top of the layer stack. When designing transport networks, these demands are usually capacities from an upper routing layer, such as IP. This top-down approach has been supported by the fact that in large commercial networks, different layers are often under the auspices of different network planners or even different departments of the network operator.

Unfortunately, the top-down approach cannot properly take the inter-layer dependencies into account and may easily lead to expensive or infeasible configurations. In a two-layer network, the structure of the physical layer imposes survivability and capacity restrictions which must be accounted for in the design of the logical layer. On the other hand, capacity decisions at the logical layer have direct consequences on the topology of the underlying physical layer. In other words, both layers influence each other, but the top-down approach only takes one of these directions into account. To take care of both directions, the logical and physical network layer must be planned together in an integrated planning process.

Depending on the input data, even planning a single network layer can be a challenge. Planning two adjacent network layers together is a very hard problem for which suitable algorithms are just being developed. Planning more than two network layers at the same time is still beyond reach for realistic network models. We thus focus on two adjacent network layers in this thesis, and provide mathematical models and algorithmic techniques to solve integrated two-layer planning problems.

1.2 Contributions of this thesis

Integrated multi-layer network design problems have already been dealt with in many publications during the last decade; a detailed literature review will be given in Section 3.5. Various mathematical models and algorithmic approaches have been proposed for different kinds of multi-layer technologies, and some of them have successfully been used to solve practical planning problems. So why is there a need for yet another publication on this topic?

The reasons are manifold. First, many mathematical models presented in the literature are tailored to specific technologies, most often IP-over-WDM, or to special network structures, such as ring or star networks. Second, already modeling a multi-layer network is a non-trivial task. Many models from the literature have been presented merely as a basis for discussion or to define the planning problem, but are not suitable to design multi-layer networks of practically relevant size. Third, important practical side constraints and sources of network cost, such as node hardware, have often been omitted. Survivability requirements, which are practically important but make the planning problem significantly harder to solve, are covered in very few mathematical models only. Fourth, in contrast to the single-layer case, little research has been done yet on the underlying mathematical structures and on exact algorithms to solve realistic multi-layer network design problems. Most authors either compute feasible network configurations heuristically without any information on their quality, or they try to

solve their mathematical models using a black-box mixed-integer programming solver. The published results show that this approach does not work for networks of realistic size. Multi-layer specific cutting planes or similar branch-and-cut techniques have barely been developed yet.

The main contributions of this thesis can be summarized as follows:

- We provide integrated mathematical mixed-integer programming models for two-layer transport network planning with and without survivability requirements. Although these models have been designed to be mostly technology-independent, they cover many important practical side constraints and sources of cost in various technologies, in particular node hardware and survivability of the logical routing against single physical node and link failures.

- For the first time in the literature, we provide a detailed discussion of different modeling alternatives for various parts of a multi-layer network, such as logical links, nodes, routing, and survivability. Which of these modeling alternatives is appropriate or preferable depends on the specific planning context. We describe the consequences of multiple logical link failures on the choice of routing formulation, and explain our choice for the models used to solve the considered multi-layer problems. In addition, we provide a detailed literature review on previous integrated multi-layer planning approaches.

- We present various problem-specific techniques to be used within a branch-and-cut-and-price framework. This includes

  - a presolving technique based on linear programming (LP) which removes unnecessary flow variables from a survivable multi-layer routing formulation,
  - several LP- and MIP-based heuristics which allow us to compute feasible solutions within the branch-and-cut tree,
  - cutting planes that take inter-layer survivability constraints into account,
  - different formulations for the routing subproblem and a dynamic LP construction technique to speed up a Benders decomposition approach with metric inequalities, and
  - column generation techniques to create routing variables dynamically.

  These techniques significantly reduce the computation times needed to solve the considered two-layer network design problems. Furthermore, they allow us to apply linear and mixed-integer programming techniques to large network instances with survivability constraints. With an out-of-the-box MIP solver, even solving the LP relaxation is out of question for large instances.

- We test our solution methods on six structurally different networks with up to 67 nodes and realistic cost structures and demands. Each of these networks is considered in two versions, with and without survivability requirements. By integrating our problem-specific techniques into the branch-and-cut-and-price framework SCIP [3], we can construct feasible network configurations for the large network instances and compute useful lower bounds on the optimal network cost. These bounds can be used to estimate the quality of the computed solutions compared to a cost-minimal network configuration.

Most of the smaller network instances with up to 17 nodes can be solved within 1% of the optimum value.

In particular, we can solve multi-layer networks of practically relevant size well enough to perform useful parameter studies now. This has not been possible before. As an example, we investigate the impact of different multi-layer routing restrictions on the network cost on the large test instances.

The methods developed in this thesis have been applied in two projects conducted with Nokia-Siemens Networks. They have also been used within the EIBONE project, where several network operators and equipment vendors were involved, to investigate the influence of various planning parameters on the total network cost in different network architectures [30].

1.3 Structure of this document

This thesis is organized as follows. In Chapter 2, we give a more detailed introduction to multi-layer telecommunication networks, and describe the practical planning problem addressed in this work.

In Chapter 3, we introduce our mixed-integer programming models. We discuss various modeling alternatives for different parts of a multi-layer network, compare different routing formulations in the presence of multiple link failures, and motivate the choices made in our models. Furthermore, we give a detailed review of the literature on integrated multi-layer planning approaches.

Chapter 4 describes the algorithmic methodology developed to solve the models presented in the previous chapter. After an overview on the overall solution approach, we present various problem-specific techniques for a branch-and-cut-and-price framework together with the necessary mathematical background. The presented techniques cover several parts of a modern branch-and-cut-and-price framework: presolving, primal heuristics, cutting planes, an exact decomposition approach, and column generation.

In Chapter 5, we evaluate the developed solution techniques on realistic test networks with and without survivability constraints. We provide extensive studies for each of the presented techniques and investigate their interplay where appropriate. We use them to investigate the impact of different routing restrictions on the total network cost on our large test instances.

In Chapter 6, we summarize our findings and indicate further research directions.

Chapter 2

Multi-layer network planning

The mathematical models and solution methods discussed in this thesis can be applied to many types of telecommunication networks. Although the hardware details and the exact side constraints depend on the specific technologies involved, the concept of a layered network appears at the heart of many telecommunication network planning problems.

This chapter explains the basic principles of a multi-layer network and their application to specific technologies, defines the planning task, and reviews the relevant literature.

2.1 Principles of a multi-layer network

In this section, we will describe the basic concepts of a multi-layer network:

- physical and logical links,
- demands and grooming,
- node hardware,
- sources of cost in a multi-layer network, and
- multi-layer routing and survivability.

As these concepts have applications in different planning contexts, we will introduce them in a technology-independent way and illustrate them using examples from optical networking. Section 2.1.2 transfers these concepts to other technological settings. We will not go into the technical details more than necessary and refer to Perlman [108], Sexton and Reid [125], and Wu [137] for more information on the related technologies.

2.1.1 Basic concepts

Physical and logical links

A two-layer network consists of two subnetworks: a *physical* and a *logical* network layer. The logical layer network nodes form a subset of those in the physical layer; often, the nodes in both layers can be identified with each other. Every link in the logical layer is defined by a path in the physical one, as illustrated in Figure 2.1. In particular, there may be exponentially many parallel logical links between each pair of nodes. The logical links must be equipped

with capacities to route demands, most often in discrete steps. Also on the physical links, enough capacity has to be installed to support the logical link capacities.

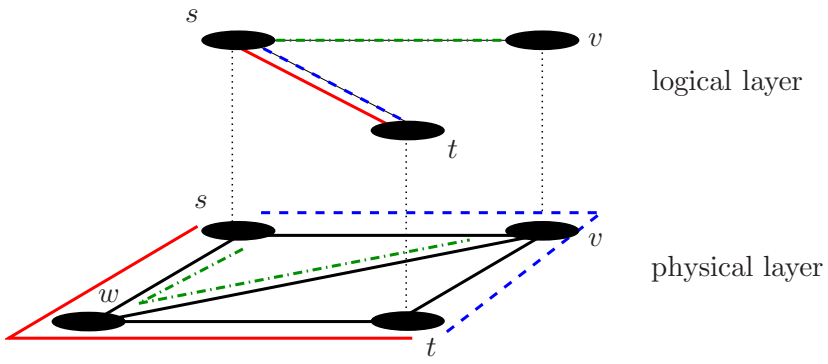


Figure 2.1: A simple two-layer network consisting of a lower physical layer and an upper logical layer with a subset of the nodes. Every logical link represents a path in the physical network, as indicated by the line styles and colors. This leads to the two parallel logical links from s to t, which correspond to different physical paths. Logically link-disjoint paths are not necessarily physically link-disjoint; for instance, the one-hop logical path s–v (green, dash-dotted) and s–t (red, solid) have the physical link s–w in common.

In an optical WDM (Wavelength Division Multiplexing) network, for example, the physical links may correspond to optical fiber links. The logical links represent lightpaths, i. e., paths in the fiber network with a specified bandwidth (e. g., 2.5, 10, or 40 Gbit/s) and wavelength. Different lightpaths sharing a common fiber must have different wavelengths, and every fiber supports up to 40 or 80 wavelengths.

Beware that a logical link can be seen in different ways, depending on the considered technology, side constraints, and mathematical model in mind. In the above definition, a logical link corresponds to *exactly one* physical path, with the consequence of parallel logical links. This is the point of view adopted in most parts of this thesis. Under certain assumptions, however, it is not necessary to distinguish between parallel logical links. In such a case, the equivalence class of all parallel logical links between a given pair of nodes can be represented by a single logical link whose capacity has to be realized by a routing via on or more paths in the underlying physical layer. This is the point of view adopted in many publications from the beginning of this decade, and also in the introduction of this thesis for easier explanation. Switching between the two notions of a logical link is no problem if appropriate assumptions are satisfied; these are discussed in more detail in Section 3.3.1. But one should keep in mind that both notions exist.

Demands

Communication demands defined between the logical network nodes have to be routed in the logical network within the logical link capacities. These demands correspond to connection requests with a specified bandwidth; in transport networks, they are often derived from capacities of an even higher routing layer, such as an IP network. Usually, the demands are given in much smaller units than the available logical link capacities. An end-to-end routing path between the demand end-nodes may consist of path consisting of a series of logical links,

each of which is realized by a path in the physical network. As a consequence, an end-to-end routing path in the physical layer may traverse a given node or physical link several times. This can be seen from Figure 2.2: a demand routed from Munich to Stuttgart via Berlin on the two available logical links traverses the node Leipzig, as well as the physical link Berlin–Leipzig, twice.

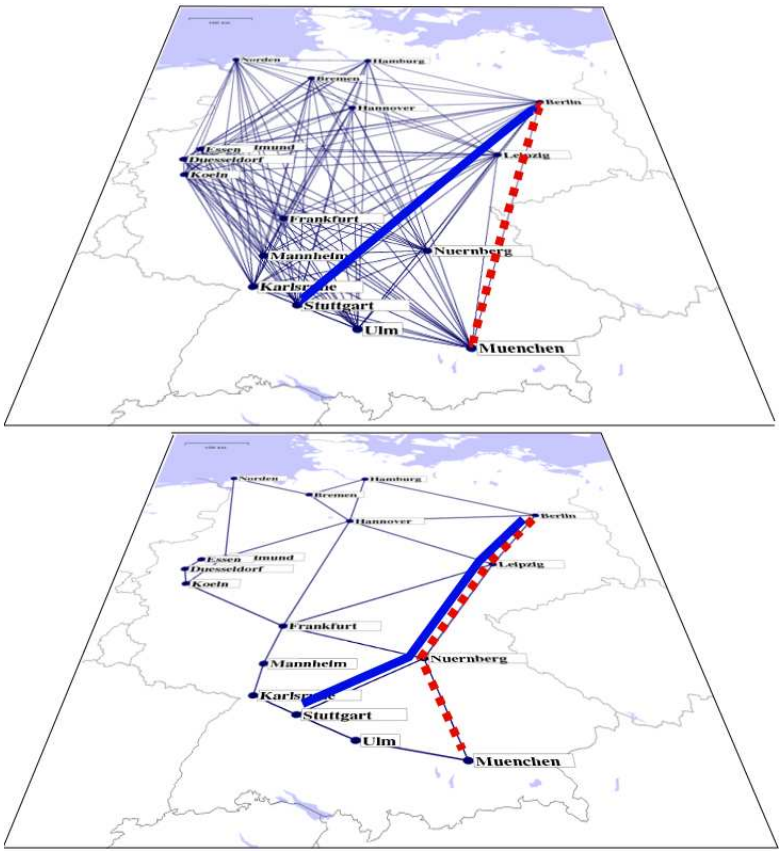


Figure 2.2: A demand from Munich to Stuttgart may be routed via Berlin on the two logical links. It cannot leave the corresponding physical paths at intermediate nodes. Hence, it traverses the subpath Nuremberg–Leipzig–Berlin twice.

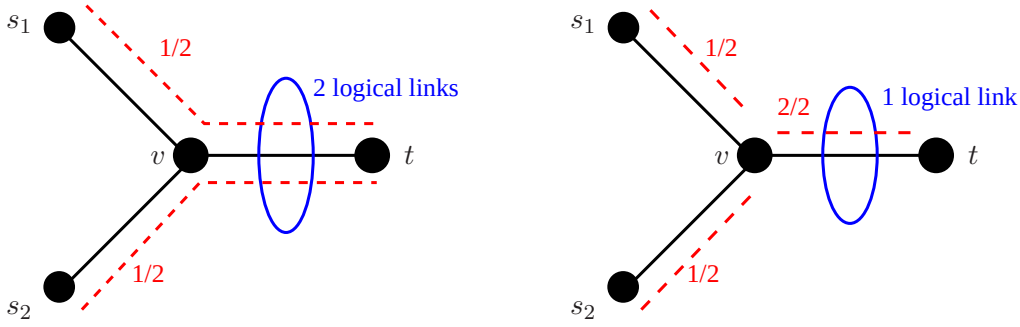
In an optical network, the demands are usually aggregated connection requests from different sources (telephone, Internet, and video traffic). The requested bandwidths are in the order of several hundreds of Mbit/s, which is much smaller than the available logical link bandwidths of several Gbit/s.

Grooming

Intuitively, a logical link with a fixed bandwidth can be imagined as a pipe whose diameter represents the bandwidth. Traffic inserted into the pipe must travel until the end of the pipe and cannot leave it in-between. *Grooming* happens at each end-node of such a pipe. This term refers to the process of extracting traffic with a low data rate from a pipe with high bandwidth, recombining the traffic and sending it out one some outgoing pipes.

A lightpath, for example, provides a pipe with a fixed bandwidth between its end-nodes. To these, it seems like a direct link, hence the name *logical link* (or *virtual link*): traffic with lower granularity, like Internet traffic from an upper IP layer, is inserted into the lightpath at one end, taken out at the other end, and cannot be accessed inside.

Grooming involves two trade-offs. One of them also occurs in single-layer networks, whereas the other one is specific to multi-layer networks. First, in most technologies, the cost of a logical link depends mainly on its capacity; its geographical length plays only a minor role for the cost. Hence, a routing path consisting of two logical links incurs nearly twice the cost as a direct logical link. If any arbitrary continuous capacity can be installed on the logical links (as in MPLS networks) and demands are not protected against failures, the cheapest solution from a logical layer perspective is thus to route every demand on a direct logical link from the source node to the target node if this is technically possible. Grooming may be necessary, however, to route a demand through a sequence of logical links if there is no direct logical link between the end-nodes of the demand. This happens, for example, if the nodes are geographically far away from each other and the available equipment does not support such long logical links. Furthermore, most technologies allow only integer multiples of some base capacities to be installed on the logical links. By routing several demands through the same logical link, grooming allows to make use of available spare capacities. These issues are also important in single-layer networks with discrete link capacities.



(a) Solution without grooming (b) Solution with grooming

Figure 2.3: A two-layer network showing the trade-off between logical cost and physical capacities. The solid lines represent physical links, whereas the dashed lines correspond to logical links on which capacities can be installed in multiples of 2. There are two demands s_1-t and s_2-t, both of value 1. The numbers represent flow and capacity on the logical links. The transparent solution on the left uses two direct end-to-end logical links and no grooming. It requires only two logical links but consumes two capacity units on the physical link v-t. The opaque solution on the right, where both demands are groomed into a single logical link at the intermediate node v, needs three logical links but consumes only one capacity unit on the physical link v-t.

In multi-layer networks, grooming involves an additional trade-off between logical link cost and physical link capacity, as illustrated in Figure 2.3. One extreme solution is a fully *transparent* network, where no grooming is done at all. Every demand is routed on exactly one logical link from its source to destination, as illustrated in Figure 2.3(a); this corresponds to a hop limit of 1 for routing demands in the logical layer. This solution tends to be relatively cheap from a logical link cost perspective, but it may exceed the available physical link capacity or incur high physical link cost. The other extreme solution is a fully *opaque*

network where a logical link can only be realized by paths consisting of exactly one physical link. This corresponds to a hop limit of 1 for routing the logical link capacity in the physical layer. In this setting, traffic is groomed at every intermediate node of a routing path, as illustrated in Figure 2.3(b). This is the cheapest solution from a physical layer perspective, but it usually leads to high logical link cost (and high node cost, see Section 2.1.3). This trade-off is specific to multi-layer networks.

How to construct an overall cost-minimal solution depends on the capacity and cost structures. The impact of a planning decision at the logical layer on the overall network cost is larger than in single-layer networks, and even harder to grasp without mathematical optimization techniques.

2.1.2 Examples from different technologies

The ideas described above occur not only in optical networks but also in other technologies. Although the technological details vary, the key concept of logical and physical links is always the same:

- The SDH (Synchronous Digital Hierarchy), which is a common technology employed for transport networks, defines a complex hierarchy of standardized connection bandwidths called VC-N (Virtual Containers), which can be embedded into each other. The embedding VC-N connections are in turn realized via STM-N (Synchronous Transport Module) connections, also with a fixed bandwidth. For example, the physical links may correspond to STM-16 connections with a capacity of 2.5 Gbit/s. A logical link may represent a connection with a capacity of VC-4 (corresponding to 140 Mbit/s), which is routed via a series of physical links. The demand requests usually have smaller bandwidth, like VC-12 (corresponding to 2 Mbit/s). SDH is inherently a multi-layer technology; the SDH standard defines even many more layers than this example. The same applies to its high-capacity extension OTN (Optical Transport Network).

- The ATM (Asynchronous Transport Module) protocol defines so-called *virtual paths*. These connections, which can be permanent or temporary, are virtual connections which are realized via some underlying transport layer, such as SDH. A traffic demand is routed through a *virtual channel* which corresponds to a series of virtual paths. It can thus be seen as a path in the logical network. ATM is mainly used as a transport and monitoring technology in mobile phone networks and to support ADSL (Asymmetric Digital Subscriber Line) connections. In both cases, the traffic demands are small, in the order of Kbit/s to a few Mbit/s.

- In the Internet, the MPLS (Multi-Protocol Label Switching) protocol provides labeled, capacitated routing paths. The links of these routing paths may themselves be other MPLS paths, even in an iterated manner [13, 121]. Hence, MPLS is inherently a multi-layer protocol, similarly to SDH.

Notice that practical multi-layer networks often have more than two layers. An MPLS path, for example, can be implemented by a series of other MPLS paths, which in turn are routed through SDH microwave radio links or through optical fiber links using WDM technology. In other words, if more than two layers are involved, the "physical" links of one layer can be logical links themselves, realized in the next lower layer. The term *physical link*

does not imply any physical structure like a fiber, but only means that the link is part of the lowest layer currently under consideration.

Throughout this thesis, we assume *connection-oriented* routing in all layers, that is, a connection has to be configured before any data can be sent. All routing protocols described above satisfy this requirement. Once such a path has been configured, the nodes on that path can forward traffic based on path information rather than on information depending on individual data packets. This allows, for instance, MPLS paths to be realized via optical lightpath connections, because cross-connects at the optical layer nodes can forward the optical signal without being able to deal with individual data packets. This is not possible with *connection-less* protocols, such as the OSPF protocol for IP networks, where each incoming data packet is individually investigated at a node in order to determine its outgoing link, and where subsequent packets are routed independently of each other. Details on the notion of connection- and packet-oriented protocols can be found in Perlman [108] and Colle [37].

2.1.3 Node hardware

The end-nodes and intermediate nodes of a logical link require appropriate hardware whose details depend on the specific technology. Nevertheless, it is possible to extract some unifying concepts which are similar everywhere. As this thesis focuses on the common kernel of multi-layer planning problems in different technologies, we will not go into the technical details more than necessary, but only explain the basic concepts and illustrate them by a few examples. The node model described in the sequel is based on work by Kröller and Wessäly [82] and by Orlowski and Wessäly [102] in cooperation with T-Systems Nova and E-Plus, as well as on results from the European NOBEL project [2]. Hülsermann et al. [72] provide a more detailed and more technical description of this node model as applied to IP/MPLS, Ethernet, SDH/OTN, and WDM networks. For MPLS, ATM, and SDH/OTN networks, the interested reader may find more information in the book by Perlman [108]. A comprehensive description of optical networks can be found in the book by Ramaswami and Sivarajan [120]; Zymolka [142] summarizes important issues which are relevant for optical network design.

From a conceptual perspective, only few functionalities are relevant for dimensioning a multi-layer network:

- The source node of a logical link needs a *multiplexing* device, which combines several low-rate signals (like MPLS signals of a few Mbit/s each) into one high-rate signal (such as a lightpath with 2.5 Gbit/s).
- At the target node of a logical link, a *demultiplexing* device extracts the low-rate signals from the high-rate signal.
- At intermediate nodes of a physical path realizing a logical link, some device has to forward the signal from one physical link to the next one.

In electrical networks with MPLS, ATM, or SDH/OTN technology, all these functionalities are usually covered by a single device, whereas in optical networks, different devices may be responsible for different functionalities. We will explain the involved hardware at the example of Figure 2.4. The figure illustrates typical node hardware in a two-layer SDH-over-WDM setting, where physical links correspond to optical fibers, and logical links to lightpaths. The situation is similar if SDH is replaced by MPLS, ATM, or OTN. Notice that the picture shows only one out of many possible configurations. Electrical and optical networking hardware

comes in many different flavors, each of which has its own advantages and drawbacks with respect to its capabilities, cost, compatibility with other devices, etc. Details can be found in the above cited publications.

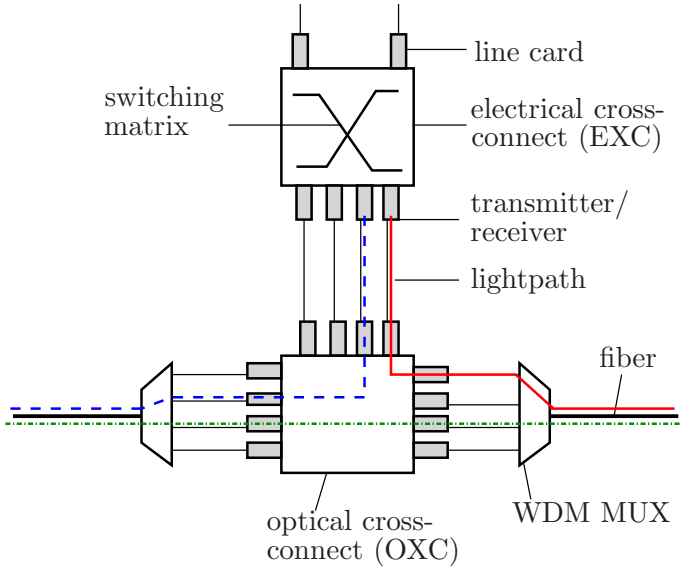


Figure 2.4: Typical node hardware in a two-layer SDH-over-WDM network. The upper part of the figure shows an electrical cross-connect with attached line cards. These terminate logical links, which are switched to the correct fiber by the optical cross-connect in the lower part. A WDM MUX combines several wavelength signals on a single fiber.

The box at the top of Figure 2.4 symbolizes an electrical cross-connect (EXC) from SDH technology. In MPLS and ATM networks, the corresponding device is called *router* or *switch*, but its design and functionality are similar. It provides slots for *line cards*, illustrated by the small grey boxes at its top and bottom. These line cards provide *ports* to terminate logical links. For instance, the ports at the bottom may consist of a *transmitter* (laser) to send out a lightpath signal and a *receiver* to receive such a signal. The line cards provide the translation between electrical and optical signals; in the figure, they terminate a blue (dashed) and a red (solid) lightpath. The ports at the top of the EXC process electrical signals only and may lead to an IP router, for instance.

The purpose of the cross-connect is grooming and switching. It receives high-rate signals with, say 2.5 Gbit/s, on its line cards, extracts low-rate signals such as VC-4 containers (140 Mbit/s) from them, reorganizes them into new 2.5 Gbit/s signals, and sends these out on another line card according to their destination. The cross in the middle of the cross-connect symbolizes the *switching matrix* of the cross-connect, which processes the low-rate traffic. It has a limited *switching capacity*, which denotes the maximum amount of data which can be switched per time unit. It also has a *switching granularity*, which is the lowest data rate it can process. For example, a cross connect with switching granularity VC-12 can extract VC-12 signals with a bandwidth of 2 Mbit/s out of the optical high-rate signal; otherwise, it can only extract VC-4 signals with a bandwidth of 140 Mbit/s. A smaller switching granularity allows for more grooming, but also raises the complexity and the cost of the cross-connect.

The lower part of Figure 2.4 shows some of the WDM devices involved. The box in

the middle represents an *optical cross-connect* (OXC) with attached line cards. It is usually placed only a few meters away from the electrical SDH cross-connect, and connected to it via several short fibers. Each of those carries at most one signal with a wavelength granularity of, say, 2.5 Gbit/s. To the left and to the right of the OXC, a *multiplexer/demultiplexer* (WDM MUX) receives up to 40 or 80 signals with different wavelengths, and sends them all together on a single fiber which leads to the next optical node, or vice versa. There also exist devices which combine the functionalities of an OXC and WDM MUX devices in a single box; they are called *optical add-drop multiplexers (OADM)*. The fiber emanating from the WDM MUX may be several hundreds of kilometers long and lead, for instance, to the next big city. If this fiber link is very long, *amplifiers* may be needed at certain distances to augment the signal strength. Unfortunately, the amplifier also amplifies noise in the signal. With increasing distance, the signal quality degrades more and more because of dispersion, jitter, and other nonlinear effects. After a maximum distance which depends, among other criteria, on the bandwidth and on the type of fiber and transmitters used, the signal thus has to be converted into an electrical signal at an EXC or by a regenerator and retransmitted as a new lightpath.

The purpose of the OXC is to switch lightpath signals from an incoming port to an outgoing port. This can be done in a hard-wired or in a configurable way, depending on the type of OXC. Of course, increased flexibility also comes at increased cost. In Figure 2.4, the red signal from the top is directed to the right, and the blue signal to the left. A third signal at the bottom is passed through from the left to the right without being converted into an electrical signal and being demultiplexed at the EXC. This signal belongs to a lightpath which contains the depicted node as an intermediate node.

To avoid that two lightpaths use the same wavelength on any fiber, lightpath signals can be sent to the EXC, converted into an electrical signal, and recreated using a different wavelength. Alternatively, if no EXC is required for grooming at that node, wavelength converters can be connected to the OXC to perform this task at lower cost and without opto-electronic conversion.

Of course, the hardware concept just described does not cover every possible detail of different technologies. Nevertheless, it covers the most important parts with respect to hardware constraints and cost, and is detailed enough for strategic planning purposes.

2.1.4 Cost

In a multi-layer network, cost is incurred at various places: switching and converting devices (cross-connects, wavelength converters, multiplexers/demultiplexers), terminating devices (line cards, ports), and transmission equipment (fibers, radio links, leased lines, and optical amplifiers). The exact cost structures depend on the specific technology. We now show that under reasonable assumptions, the cost of the above devices can be attributed to nodes, logical links, and physical links.

In many technological settings, a large part of the network cost is caused by line cards and ports. The cost of a port can be directly attributed to the logical link it terminates. Under the—often satisfied—assumption that every line card provides a single port, i. e., it terminates a single logical link, its cost can be attributed to that logical link as well. Consequently, the cost of a logical link depends mostly on its capacity, which is determined by the ports at the end-nodes of the link. The cost of a logical link may also depend on its geographical length, because long-range equipment is more expensive than short-range equipment, especially in optical networks. Furthermore, the length of the link may restrict the set of available ca-

pacities because high data rates cannot be realized on very long links without intermediate amplification and regeneration.

A physical link incurs transmission cost, e.g., for a fiber or for the radio link equipment. In many cases, it is composed of a fixed setup cost and a length-dependent cost component. In optical networks, every multiplexer and demultiplexer terminates a single fiber; consequently, its cost can be attributed to that fiber. Also amplifier cost can be attributed to the corresponding fiber because an amplifier acts on all wavelengths on the fiber simultaneously.

Eventually, the network nodes incur cost for cross-connects and wavelength converters. The cost of splicing two optical fibers together or the cost for node-internal cables are negligible compared to the total network cost and can be omitted in a strategic planning process.

For the purpose of this thesis, our goal is to minimize total installation cost of the network. This installation cost might comprise operational cost which is required for the initial setup of the network. For instance, the cost of digging a hole for a new fiber can be attributed to the fiber itself. In contrast, we do not consider operational cost which is incurred while the network is running, such as routing cost or the cost of personnel required for supervising and ensuring smooth operation of the network.

2.1.5 Multi-layer routing and survivability

Routing The routing of demands in the logical layer may be subject to various side constraints depending on the technological context. In optical networks, a typical requirement is that every demand should be routed from its source to its destination on a single path (or on two disjoint paths). In MPLS networks, on the other hand, the protocol allows to distribute the flow on several paths in an arbitrary way. A network operator may impose further restrictions on the routing, such as a maximum number of logical links per routing path in order to reduce transmission delays. In our models, we assume that the flow can be arbitrarily split; certain hop limits can be taken into account.

Survivability In a telecommunication network, hardware components at the network nodes or physical fibers may fail, and do so regularly. For such failure situation, some backup capacity has to be provided in the network in such a way that affected traffic can be rerouted around the failure.

Already in the context of single-layer networks, a plethora of survivability mechanisms has been developed. Some of them reroute affected flow locally around the failure, others compute new end-to-end routing paths. These mechanisms differ in the amount of required backup capacity, the complexity of the action needed when the failure occurs, the signaling requirements, the recovery time, and the way in which flow can be rerouted. A survey on the most important protection and restoration mechanisms can be found in the books by Grover [62] and Pióro and Medhi [109]. Which kind of restoration concept is applicable or preferable depends on the monitoring and signaling capabilities of the employed technologies, as well as on cost and delay considerations, contracts with customers, and other criteria.

In a multi-layer network, the situation is even more complicated because recovery can take place at the logical layer, at the physical layer, or in both layers. In the latter case, the recovery action of both layers might be coordinated by some central unit, or it might be driven by simple techniques like time-outs. In each case, various rerouting mechanisms may be applicable at the different network layers. Furthermore, single failures at the physical layer may lead to multiple failures at the logical layer, which are difficult to deal with both from

a practical and from a theoretical point of view. Even in practice, it is not always evident which of these options can and should be implemented; it is even less clear how to model these mechanisms mathematically. Demeester et al. [44] and Autenrieth [10] discuss several survivability options in different technological contexts from a practical perspective.

A widespread way to ensure survivability in practice, especially in transport networks, is *1+1 protection*. With this mechanism, every unit of demand is duplicated at its source node, and routed on two paths which do not have any node or link in common. The target node then chooses the better of the two signals. In this way, the demand will survive any single node failure or single fiber cut. A similar method is known as *1:1 protection* (or *hot standby protection*). With this concept, capacity on the backup path is reserved for a particular demand, but is only used when the failure occurs. From a mathematical point of view, both concepts are equivalent in the sense that they have the same capacity requirements.

Throughout this thesis, we assume that if physical links and nodes fail, they fail completely, and that every demand is protected against single physical link or node failures using 1+1 protection. That is, we double each demand value and make sure that at least the original demand value survives any physical link or node failure. In this way, every demand will survive fiber disruptions as well as failures of single ports, multiplexers, or cross-connects.

2.2 The planning task

2.2.1 Integrated planning of several layers

From a practical point of view, the two-layer planning problem considered in this thesis can be summarized as follows. Given are a set of potential network nodes at both layers, a set of potential physical and logical links between these nodes, a discrete set of installable capacities on the logical links with cost values, restrictions of the physical links on the logical link capacities, installable hardware at the nodes with cost values, and a set of communication demands with routing requirements. A feasible network configuration consists of

- a subset of the available nodes and links at both layers,
- a hardware installation at the nodes of both layers,
- a capacity installation on the logical links, and
- a routing of the demands within the logical link capacities,

such that all side constraints are satisfied. The goal of the two-layer planning problem is to find a feasible network configuration with minimum total installation cost.

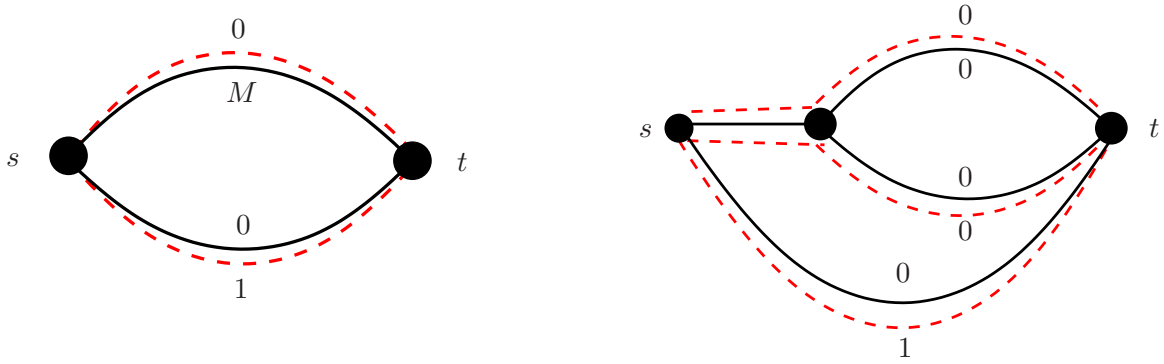
Our mathematical models, which will be presented in the next chapter, do not cover all of the side constraints presented in the previous section. Still, they take the most important restrictions into account, and many more than any other multi-layer network planning model known from the literature. These issues are dealt with in more detail in the model discussion in Section 3.3.

2.2.2 Sequential vs. integrated planning

From a network management perspective, the various network layers form a stack with a client-server relationship. At a network operator's company, different layers are often under the responsibility of different network planners, or even different departments.

The traditional approach to multi-layer network design and routing problems in practice is to plan each layer individually in a top-down manner. More precisely, routing and capacities on the highest layer are computed for the given demands such as to minimize the total cost of that layer. These capacities are then given as input to the planning problem of the next lower layer in the form of point-to-point bandwidth requests (i. e., demands) between the end-nodes of logical links. In particular, the demand graph, where each demand request between two nodes is represented by a link between these nodes, can be seen as an additional layer on top of the layer stack, and the demands of a transport network design problem are often derived from capacities of a higher layer.

On the one hand, this approach is attractive to network planners, because a series of single-layer problems can be tackled much easier computationally than one multi-layer problem. On the other hand, the sequential approach may lead to arbitrarily bad results, or even fail to find feasible network configurations, as shown by the following two propositions.



(a) Sequential planning can lead to arbitrarily bad solutions.

(b) Sequential planning can fail to identify feasible solutions.

Figure 2.5: Worst-case examples for Propositions 2.1 and 2.2.

Proposition 2.1. *Sequential planning of two network layers may lead to arbitrarily bad results even in an extremely basic multi-layer setting.*

Proof. Figure 2.5(a) shows an example where sequential planning of both network layers leads to arbitrarily bad solutions. It consists of two nodes s and t connected by two physical links (solid), and two logical one-hop links (dashed). Installing the physical links incurs a cost of 0 and M. A capacity of 1 can be installed at cost 1 and 0 on the logical links, respectively. Assume that there is one demand of value 1 from s to t, and neither survivability constraints nor node hardware are considered.

For $M > 1$, the unique optimal solution is to route the demand on the lower logical link at cost 1, which incurs a physical cost of 0 and thus a total cost of 1. In contrast, the sequential planning approach would route the demand on the upper logical link at cost 0, which leads to a physical cost of M. By adjusting M, the cost of the sequential solution compared to the optimal solution can exceed any positive number. □

Proposition 2.2. *The sequential approach may fail to identify feasible solutions if survivability requirements are taken into account.*

Proof. Figure 2.5(b) shows an example where the sequential approach fails to identify an existing feasible network configuration with survivability constraints. Again, the solid lines are the physical links, and the dashed lines represent logical links. Suppose that there is a 1+1-protected demand of value 1 from s to t, which must be routed on two physically disjoint paths. The only way to do so is to route one unit on one of the upper logical links, and one unit on the lowest logical link. The sequential approach will fail to find this solution: the cheapest logically node- and link-disjoint way of routing in the logical layer is to use the two upper logical links (which directly connect s and t), but these are not physically disjoint.

Notice that a similar example can easily be constructed without survivability constraints if the number of capacity units on all logical links traversing a given physical link is limited (which is usually the case). □

Although these two examples represent worst-case scenarios, they show that the integrated planning approach should be used whenever possible. Of course, a sequential planning approach has to be employed if the two layers under consideration are under the control of different network operators; such an approach may or may not lead to good network configurations in practice. Still, the integrated planning approach is useful in such a case to estimate the quality of the computed solutions.

Chapter 3

Mixed-integer programming models

Modeling a multi-layer network is a highly non-trivial task. Different ways of formulating the planning problem mathematically may be possible or appropriate, depending on the technological context, the exact side constraints, and the intended solution method. In the literature on multi-layer network design, several mixed-integer programming formulations have been proposed. Whereas some of them have been used in a branch-and-cut algorithm, others turned out to be computationally intractable because of their symmetries or for other reasons.

In this section, we develop a series of models which are designed for computational use within a branch-and-cut algorithm. Section 3.1 introduces the necessary problem parameters. In Section 3.2, we present three models: a straightforward model for integrated two-layer network design in the unprotected case, an improved formulation where parallel logical link flow variables are aggregated into a single variable, and an extension to 1+1 protection. In Section 3.3, we provide a detailed discussion of different modeling alternatives for various parts of the problem, and motivate our modeling decisions. Because of their importance, routing and survivability issues are dealt with separately in Section 3.4. Such a discussion has not been presented in the multi-layer literature yet, neither for the network and hardware part nor for the routing and survivability part. Eventually, we review the relevant multi-layer planning literature in Section 3.5 and relate our approach to previous publications.

3.1 Problem parameters

This section introduces all necessary problem parameters; they are summarized in Table 3.1. A full list of symbols can be found in the Appendix on page 181.

Network The physical network is represented by an undirected graph $G_P = (V, E)$. The logical network is modeled by an undirected graph $G_L = (V, L)$ with the same set of nodes and a fixed set L of admissible logical links, where each $\ell \in L$ is defined by an undirected path in the physical network. Consequently, there may be many parallel logical links corresponding to different physical paths between any two nodes. Let $\delta_L(i) \subseteq L$ be the subset of all logical links incident to node $i \in V$, i. e., starting or ending at i. For ease of notation, define $L_{ij} := \delta_L(i) \cap \delta_L(j)$ as the subset of all logical links connecting nodes i and j, with $L_{ii} := \emptyset$ for all $i \in V$ (that is, we disallow loops in the logical network). $L^e \subseteq L$ refers to the sets of logical links containing physical link $e \in E$. Likewise, $L^i \subseteq L$ denotes the set of logical links containing i as an inner node (not as an end-node).

notation	interpretation
V, E, L, R	set of all nodes, physical links, logical links, and demands
L_{ij}	set of all logical links $\ell \in L$ between nodes i and j
L^e	set of all logical links $\ell \in L$ containing physical link $e \in E$
L^i	set of all logical links $\ell \in L$ containing $i \in V$ as an inner node
$\delta_L(i)$	set of all logical links $\ell \in L$ starting or ending in node $i \in V$
M_ℓ, M_i	set of capacity modules installable at logical link $\ell \in L$ or node $i \in V$
C^m_ℓ, C^m_i	capacity of a logical link module $m \in M_\ell$ or a node module $m \in M_i$
C_i	capacity needed at a demand end-node for the access link from an upper layer
κ_e	cost of a capacity unit on physical link $e \in E$
κ^m_ℓ, κ^m_i	cost of a logical link module $m \in M_\ell$ or a node module $m \in M_i$
B_e	maximum number of logical link modules supported by a physical module
R^u, R^p, R	set of all unprotected, all protected, and all demands
K^u, K^p, K	set of all unprotected, all protected, and all commodities
R_{ij}	set of all demands from node $i \in V$ to node $j \in V$
d^r, d^k	demand value of a demand $r \in R$ or a commodity $k \in K$
d^k_i	net demand for commodity $k \in K$ that has to leave node $i \in V$
d_i	total emanating demand of node $i \in V$
s^k, t^k	source of a commodity $k \in K$, target of a protected commodity $k \in K^p$

Table 3.1: Parameters of the two-layer planning problem

In a real network, various devices may be installed at each location in different network layers. In our model, the nodes represent whole locations (e. g., cities) rather than individual devices; this is discussed in more detail in Section 3.3.3.

Capacities Each logical link $\ell \in L$ has a set M_ℓ of installable capacity modules (different bit-rates, e. g., 2.5, 10, or 40 Gbit/s) that can be installed in arbitrary combination. Each module $m \in M_\ell$ has a capacity of $C^m_\ell \in \mathbb{Z}_+$ and a cost of $\kappa^m_\ell \in \mathbb{R}_+$. Similarly, every node $i \in V$ has a set M_i of installable node modules, at most one of which may be chosen. Every node module $m \in M_i$ has a switching capacity of $C^m_i \in \mathbb{Z}_+$ (e. g., in Gbit/s) and a cost of $\kappa^m_i \in \mathbb{R}_+$. On a physical link $e \in E$, capacities can be installed at a cost of $\kappa_e \in \mathbb{R}_+$ per unit (corresponding to fibers, for example). Each physical capacity unit supports up to $B_e \in \mathbb{Z}_+$ logical capacity modules (e. g., lightpaths).

Demands and commodities Connection requests between the network nodes are modeled by a set $R = R^p \uplus R^u$ of undirected point-to-point demands, where R^p comprises the *1+1-protected* demands and R^u the *unprotected* ones. Protected demands are expected to survive single physical node and link failures. For $i, j \in V$, let $R_{ij} \subseteq R$ be the subset of demands defined between nodes i and j. We assume $R_{ii} = \emptyset$ for all $i \in V$, that is, we exclude demands from a node to itself. For each demand $r \in R_{ij}$, a demand value of d^r has to be routed between i and j, i.e., a bandwidth of d^r has to be reserved between these nodes (in arbitrary direction). For 1+1-protected demands, protection is modeled by doubling the demand value and requiring that at most the original demand value is routed through any single physical

link or node. In this way, it is guaranteed that at least the original demand survives any single physical link or node failure. For protected demands, the notation d^r therefore refers to twice the original demand value that would have to be routed if the demand was unprotected.

From now on, these demands are assumed to be directed in an arbitrary way. For any pair of nodes $i, j \in V$, let R_{ij} be the set of all demands directed from i to j, where $R_{ii} = \emptyset$ for all $i \in V$. As the direction of the demands is arbitrary, we may assume without loss of generality that for all nodes $i, j \in V$, either R_{ij} or R_{ji} is empty.

A set $K = K^p \uplus K^u$ of protected and unprotected commodities is constructed from these demands as follows. Every *protected commodity* $k \in K^p$ consists of a single 1+1-protected point-to-point demand. In contrast, *unprotected commodities* $k \in K^u$ are derived by aggregating all unprotected point-to-point demands that share a common source node. The source node of a commodity is defined in a natural way by the source node of its demands. A protected commodity has exactly one target node $t^k \in V$, whereas an unprotected commodity may have several ones.

Let $R^k \subseteq R$ be the set of demands from which commodity $k \in K$ has been constructed at the common source node $s^k \in V$ of all these demands. The commodity demand value

$$d^k := \sum_{r \in R^k} d^r = \sum_{i \in V} \sum_{r \in R_{s^k,i}} d^r$$

of the commodity is defined as the sum of its demand values. For any node $i \in V$, let

$$d_i^k := \begin{cases} d^k & \text{if } i = s^k, \\ -\sum_{r \in R_{s^k,i}} d^r & \text{if } i \neq s^k \end{cases}$$

be the supply (positive) or demand (negative) for commodity k at node i. By construction, $\sum_{i \in V} d_i^k = 0$ holds for every commodity k. Furthermore, for any node $i \in V$, denote by d_i the total demand value starting or ending in i:

$$d_i := \sum_{j \in V} \sum_{r \in R_{ij} \cup R_{ji}} d^r = \sum_{k \in K} |d_i^k|$$

This value will also be referred to as the (undirected) *emanating demand* of node i.

The switching capacity of a node i must be sufficient to switch all the traffic on logical links terminating at i. With such a tight dimensioning, however, changing the routing within given link capacities might make it necessary to re-dimension some nodes, which is inconvenient from a practical point of view. As a remedy, we assume that the switching capacity of a node i must be at least as large as the total capacity of all logical links terminating at i. Furthermore, we request some additional switching capacity $C_i \geq d_i$ for the emanating demand of node i, which is actually traffic from some upper routing network layer that has to be switched at node i. The value C_i represents the capacity of the access link from the next upper layer.

3.2 Mixed-integer programming models

This section presents a series of mixed-integer programming models for the two-layer network design problem described in the previous chapter. The first model describes the basic problem in the unprotected case, i.e., no backup capacity is provided for the case of node or link failures. The second model is a simplification of the first one, where flow variables on single logical links

are aggregated to flow variables between pairs of nodes. Eventually, the third model provides an extension to the first model where some demands may be protected against physical node or link failures.

3.2.1 Basic model without protection

We first consider the case where $K = K^u$, that is, demands are not protected against node or link failures. This model is called UNPROTECTED-BASE.

Variables

The model comprises four classes of variables representing the traffic flow and different capacity types.

- For each logical link $\ell \in L$ and each module $m \in M_\ell$, the logical link capacity variable $y_\ell^m \in \mathbb{Z}_+$ represents the number of modules of type m installed on ℓ.
- For any physical link $e \in E$, the integer physical link capacity variable $z_e \in \mathbb{Z}_+$ denotes the number of capacity units installed on physical link e.
- For each node $i \in V$ and each node module $m \in M_i$, the binary variable $x_i^m \in \{0, 1\}$ denotes whether module m is installed at node i or not.
- Eventually, the flow variables $f_{\ell,ij}^k, f_{\ell,ji}^k \in \mathbb{R}_+$ represent the flow for commodity $k \in K$ on logical link $\ell \in L_{ij}$ directed from i to j and from j to i, respectively.

Objective

The objective is to minimize the total cost of node, logical, and physical link capacity:

$$\min \sum_{i \in V} \sum_{m \in M_i} \kappa_i^m x_i^m + \sum_{\ell \in L} \sum_{m \in M_\ell} \kappa_\ell^m y_\ell^m + \sum_{e \in E} \kappa_e z_e \tag{3.1a}$$

Constraints

The model consists of a capacitated multi-commodity flow on the logical layer (3.1b)–(3.1c), node module selection and capacity constraints (3.1d)–(3.1e), and physical link capacity constraints (3.1f). First, the usual flow conservation constraints on the logical layer must be satisfied:

$$\sum_{j \in V} \sum_{\ell \in L_{ij}} (f_{\ell,ij}^k - f_{\ell,ji}^k) = d_i^k \qquad \forall\, i \in V,\ \forall\, k \in K \tag{3.1b}$$

The capacity on every logical link must be sufficient to accommodate the flow in both directions:

$$\sum_{m \in M_\ell} C_\ell^m y_\ell^m - \sum_{k \in K} (f_{\ell,ij}^k + f_{\ell,ji}^k) \geq 0 \qquad \forall\, i, j \in V,\ \ell \in L_{ij} \tag{3.1c}$$

At most one module may be installed at each node:

$$\sum_{m \in M_i} x_i^m \leq 1 \qquad \forall\, i \in V \tag{3.1d}$$

The module installed at node i must be able to switch all traffic on logical links starting or ending at i, including the access link capacity C_i:

$$\sum_{m \in M_i} C_i^m x_i^m - \sum_{\ell \in \delta_L(i)} \sum_{m \in M_\ell} C_\ell^m y_\ell^m \geq C_i \qquad \forall\, i \in V \tag{3.1e}$$

Every installed module installed on a logical link $\ell \in L$ provides capacity on ℓ but consumes capacity on its physical links. For instance, if the logical links correspond to wavelength channels, a limited number of them can be routed through any optical fiber. Hence, for every physical link $e \in E$, the constraint

$$B_e z_e - \sum_{\ell \in L^e} \sum_{m \in M_\ell} y_\ell^m \geq 0 \qquad \forall\, e \in E \tag{3.1f}$$

ensures the maximum number B_e of logical link modules supported by a physical link capacity module is not exceeded. Furthermore, it sets the physical link capacity variable z_e to at least 1 as soon as physical link e is used. Eventually, the variable domains and integrality constraints must be satisfied:

$$y_\ell^m, z_e \in \mathbb{Z}_+, \tag{3.1g}$$
$$x_i^m \in \{0, 1\}, \tag{3.1h}$$
$$f_{\ell,ij}^k, f_{\ell,ji}^k \in \mathbb{R}_+ \tag{3.1i}$$

Depending on the input data, some variables in this model can be removed or bounded in order to strengthen the LP relaxation and to reduce the generation of cycles in LP solutions. These model reductions are described in Section 4.2.

3.2.2 Aggregating logical link flow to node-pair flow

The above model has two flow variables for every logical link and every commodity. As a given pair of nodes may be connected by many parallel logical links, the number of flow variables can be very high. As long as no survivability against physical node and link failures is required, however, the physical restrictions only affect the capacity variables but not the flow variables. That is, from a flow point of view, parallel links are equivalent.

In this case, the multi-commodity flow on the huge number of parallel logical links can be reduced to a multi-commodity flow on a simple complete logical graph. The basic idea, used for instance in [14, 52, 86, 111, 127], is to formulate aggregated capacity constraints on the set of all logical links between a given pair of nodes rather than on individual logical links. For the aggregated edge-flow formulation, we introduce new flow variables

$$f_{ij}^k := \sum_{\ell \in L_{ij}} f_{\ell,ij}^k, \qquad f_{ji}^k := \sum_{\ell \in L_{ij}} f_{\ell,ji}^k \tag{3.2}$$

for every pair of nodes $i, j \in V$, $i \neq j$, denoting the total flow between nodes i and j for commodity k in forward and backward direction, respectively. The flow-conservation (3.1b) and capacity constraints (3.1c) are replaced by their equivalents with the new flow variables.

The whole aggregated edge-flow formulation UNPROTECTED-AGGREGATED reads as follows:

$$\min \sum_{i\in V}\sum_{m\in M_i} \kappa_i^m x_i^m + \sum_{\ell\in L}\sum_{m\in M_\ell} \kappa_\ell^m y_\ell^m + \sum_{e\in E} \kappa_e z_e \tag{3.3a}$$

$$\sum_{j\in V}(f_{ij}^k - f_{ji}^k) = d_i^k \qquad \forall\, i \in V,\ \forall\, k \in K \tag{3.3b}$$

$$\sum_{\ell\in L_{ij}}\sum_{m\in M_\ell} C_\ell^m y_\ell^m - \sum_{k\in K}(f_{ij}^k + f_{ji}^k) \geq 0 \qquad \forall\, i,j \in V,\ i \neq j \tag{3.3c}$$

$$\sum_{m\in M_i} x_i^m \leq 1 \qquad \forall\, i \in V \tag{3.3d}$$

$$\sum_{m\in M_i} C_i^m x_i^m - \sum_{\ell\in\delta_L(i)}\sum_{m\in M_\ell} C_\ell^m y_\ell^m \geq C_i \qquad \forall\, i \in V \tag{3.3e}$$

$$B_e z_e - \sum_{\ell\in L^e}\sum_{m\in M_\ell} y_\ell^m \geq 0 \qquad \forall\, e \in E \tag{3.3f}$$

$$f_{ij}^k, f_{ji}^k \in \mathbb{R}_+,\ y_\ell^m, z_e \in \mathbb{Z}_+ \tag{3.3g}$$

Lemma 3.1. *Every solution of* UNPROTECTED-BASE *can be transformed into a solution of* UNPROTECTED-AGGREGATED *with the same cost, and vice versa. In other words, these two models are equivalent with respect to their solution space.*

Proof. Any solution for UNPROTECTED-BASE can be converted into a solution of the aggregated model with the same cost by applying the variable transformation (3.2). Vice versa, a solution to UNPROTECTED-AGGREGATED given in terms of node-pair flow variables f_{ij}^k, f_{ji}^k can be transformed into logical link flow variables $f_{\ell,ij}^k$, $f_{\ell,ji}^k$ by distributing the flow between i and j on the logical links $\ell \in L_{ij}$ within the capacities $y_\ell := \sum_{m\in M_\ell} C_\ell^m y_\ell^m$ in an arbitrary way. This is always possible because the flow can be arbitrarily split among logical links, and Constraints (3.3c) ensure that enough capacity is installed in total on logical links $\ell \in L_{ij}$. The transformation would even be possible with integer flow variables because the logical link capacities are integer as well. □

The equivalence of these two models has been implicitly used in previous publications without being mentioned. In those publications where node-pair capacity variables have been used, the mixed-integer programming formulations always include an additional multi-commodity flow problem to route the logical link capacities in the physical layer. With the exception of Stanojević [127], flow and capacities of individual logical links have not been considered at all in these publications because this information is not necessary with the simplifying assumptions made in those formulations. The advantages and drawbacks of this approach are discussed in Section 3.3.1.

As logical links have to be explicitly configured in practice, a solution intended for use by a network operator should be given in terms of explicit logical link capacities. We thus compute solutions for UNPROTECTED-AGGREGATED and apply the above transformation in a postprocessing step to construct solutions for the original model UNPROTECTED-BASE.

3.2.3 Protection against physical node or link failures

For commodities $k \in K^p$, which are protected against physical node or link failures, the realization of a logical link in the physical layer has to be taken into account in the model. Consequently, the logical link flow variables cannot be aggregated as in the previous section.

In addition to the node-pair flow variables f^k_{ij} for unprotected commodities, we thus introduce logical link flow variables $f^k_{\ell,ij}$ for protected commodities. We exclude those logical link flow variables $f^k_{\ell,ij}$, $f^k_{\ell,ji}$ where the logical link ℓ contains an end-node of the protected commodity k as an inner node, for reasons discussed in Section 3.4. Moreover, we do not generate flow variables into the source or out of the target of a protected (point-to-point) commodity to avoid unwanted cycle flows. That is, the set of admissible flow variables for protected commodities can be described as

$$\left\{ f^k_{\ell,ij} \mid k \in K^p,\, i,j \in V,\, i \neq j,\, \ell \in L_{ij} \setminus \{L^{s^k} \cup L^{t^k}\},\, i \neq t^k,\, j \neq s^k \right\}.$$

The node-pair (3.3c) and logical link capacity constraints (3.1c) are adapted to the new variables, and additional demand (3.4c) and survivability constraints (3.4d), (3.4e) ensure that enough flow is routed both in the normal network state and in any single node or physical link failure state. The model PROTECTED with protected and unprotected commodities reads as follows:

$$\min \sum_{i \in V} \sum_{m \in M_i} \kappa^m_i x^m_i + \sum_{\ell \in L} \sum_{m \in M_\ell} \kappa^m_\ell y^m_\ell + \sum_{e \in E} \kappa_e z_e \tag{3.4a}$$

$$\sum_{j \in V} \sum_{\ell \in L_{ij}} (f^k_{ij} - f^k_{ji}) = d^k_i \qquad \begin{array}{l} \forall\, i \in V, \\ \forall\, k \in K^u \end{array} \tag{3.4b}$$

$$\sum_{j \in V} \sum_{\ell \in L_{ij}} (f^k_{\ell,ij} - f^k_{\ell,ji}) = d^k_i \qquad \begin{array}{l} \forall\, i \in V, \\ \forall\, k \in K^p \end{array} \tag{3.4c}$$

$$\frac{1}{2} \sum_{\ell \in \delta_L(v)} (f^k_{\ell,ij} + f^k_{\ell,ji}) + \sum_{\ell \in L^v} (f^k_{\ell,ij} + f^k_{\ell,ji}) \leq \frac{d^k}{2} \qquad \begin{array}{l} \forall\, k \in K^p, \\ \forall\, v \in V \end{array} \tag{3.4d}$$

$$\sum_{\ell \in L^e} (f^k_{\ell,ij} + f^k_{\ell,ji}) \leq \frac{d^k}{2} \qquad \begin{array}{l} \forall\, k \in K^p, \\ \forall\, e \in E \end{array} \tag{3.4e}$$

$$\sum_{m \in M_\ell} C^m_\ell y^m_\ell - \sum_{k \in K^p} (f^k_{\ell,ij} + f^k_{\ell,ji}) \geq 0 \qquad \begin{array}{l} \forall\, i,j \in V, \\ \forall\, \ell \in L_{ij} \end{array} \tag{3.4f}$$

$$\sum_{\ell \in L_{ij}} \left(\sum_{m \in M_\ell} C^m_\ell y^m_\ell - \sum_{k \in K^p} (f^k_{\ell,ij} + f^k_{\ell,ji}) \right) - \sum_{k \in K^u} (f^k_{ij} + f^k_{ji}) \geq 0 \qquad \begin{array}{l} \forall\, i,j \in V, \\ i \neq j \end{array} \tag{3.4g}$$

$$\sum_{m \in M_i} x^m_i \leq 1 \qquad \forall\, i \in V \tag{3.4h}$$

$$\sum_{m \in M_i} C^m_i x^m_i - \sum_{\ell \in \delta_L(i)} \sum_{m \in M_\ell} C^m_\ell y^m_\ell \geq C_i \qquad \forall\, i \in V \tag{3.4i}$$

$$B_e z_e - \sum_{\ell \in L^e} \sum_{m \in M_\ell} y^m_\ell \geq 0 \qquad \forall\, e \in E \tag{3.4j}$$

$$f^k_{\ell,ij},\, f^k_{\ell,ji},\, f^k_{ij},\, f^k_{ji} \in \mathbb{R}_+, \quad y^m_\ell,\, z_e \in \mathbb{Z}_+, \quad x^m_i \in \{0,1\} \tag{3.4k}$$

In addition to the objective (3.4a) and the constraints (3.4b), (3.4h), (3.4i) and (3.4j), which have been taken unchanged from the previous model, this model contains some new constraints related to survivability. The new demand constraints for protected commodities (3.4c) make sure that a flow of d_i^k is routed in the normal network state.

To model survivability, we use the *diversification* concept proposed by Dahl and Stoer [40]. The node diversification constraints (3.4d) ensure that at most $d^k/2$ (i. e., the original demand value) is routed through any physical node, i. e., at least a flow of $d^k/2$ survives any single physical node failure. The first sum contains all logical links starting or ending in node v, whereas the second sum contains those logical links having v as an inner node. Notice that if node v is an end-node of commodity k, the second sum in the corresponding node diversification constraint (3.4d) is empty. As we have also forbidden flow into the source node or out of the target node of k, this diversification constraint is dominated by the corresponding demand constraint (3.4c) and can be omitted. The edge diversification constraints (3.4e) ensure that the routing also survives single physical link failures. For practical reasons, we forbid some variables in the model; this is discussed in more detail in Section 3.4.

The capacity constraints (3.3c) from the model without protection are replaced by two new classes of inequalities. Constraints (3.4f) state that the capacity of every single logical link should be sufficient to accommodate the *protected flow* on that link. Based on these inequalities, Constraints (3.4g) ensure that between any two nodes $i, j \in V$, the total remaining capacity suffices to accommodate the *unprotected flow*. As in the previous model, the unprotected flow can be arbitrarily distributed within this free capacity in a postprocessing step. In the special case where all commodities are protected, i.e., $K = K^p$, Constraints (3.4g) are only aggregations of Constraints (3.4f) and can be omitted.

3.3 Discussion of multi-layer network models

Modeling a multi-layer network is a non-trivial task. Various models have been proposed in the literature; some of them are generic but leave out important side constraints; others are more detailed but tailored to a specific technological setting; a third group of models covers many aspects but is too complex to be used for computations. While it is clear that not all practical aspects can be included in a mathematical model, it is not always obvious which ones have to be taken into account and which ones can be left out. Judging from the multi-layer network design literature, there is a need for a detailed discussion of the impact of certain modeling decisions on the practical relevance and solvability of a model.

The goal of this section is to discuss various important aspects that have to be considered when modeling a multi-layer network, to present different modeling approaches from the literature with their features and limitations, and to motivate some of our modeling decisions.

3.3.1 Logical link model

The most important aspect when describing a multi-layer network is the way in which logical links are modeled. Two different approaches have been proposed in the literature:

1. In the explicit approach, which is used in this thesis, every logical link has a prespecified representation in the physical layer. This leads to many parallel logical links corresponding to different physical paths. A subset of these has to be chosen and equipped with integer capacities.

A drawback of this approach is that if all possible logical links are admissible, modeling the problem using mixed-integer programming techniques requires an exponential number of integer variables. A branch-and-cut-and-price algorithm on such a model includes an $\mathcal{NP}$-hard column generation problem even in the most basic version with a fixed physical layer, a single capacity type, and without survivability (see Raghavan and Stanojević [119] for basic ideas on this approach). The problem is that every new column also generates a new row, with the result that not all dual variables are known in the column generation subproblem. Several attempts in this direction failed because of wrong algorithmic assumptions (see Section 3.5).

The advantage of the explicit approach, on the other hand, is that this kind of models is very flexible if not all logical links are admissible or if additional side constraints have to be taken into account (see below). Consequently, this is the approach employed in most publications from recent years, including this thesis.

2. In the implicit approach, each logical link represents the whole set of parallel physical paths between its end-nodes. The actual representation of the logical link by physical paths is determined during the algorithm. In this case, parallel logical links can be aggregated into a single link, such that the logical graph is simple.

   The physical representation of a logical link is usually modeled by interpreting logical link capacities as physical demands and routing them in the physical layer either using a multi-commodity flow routing [14, 83] or by means of metric inequalities [52, 76]. This corresponds to the traditional top-down view of a multi-layer network, where the capacities of each layer serve as demands in a lower layer. This kind of model has been used in several publications from the optical networking community around the year 2000, as well as in papers focusing on metric inequalities (see the literature review in Section 3.5).

Provided that all physical paths are admissible as logical links, the same capacities can be installed on all logical links, node hardware does not depend on the flow through a node, and node failures are not considered, both approaches are equivalent in the sense that their solutions can be transformed into each other. Many practical side constraints, like several capacity types, a maximum number of logical links using a physical link, and flow-independent node capacities can be modeled using both approaches. Also physical link failures can be formulated in both models (although the implicit model is restricted to protecting link capacity, whereas the explicit model can protect either link capacity or flow). Under the above assumptions, the main advantage of the implicit approach is that it can—at least for small instances—be used to compute a dual bound on the total network cost with respect to *all* possible logical links.

In this thesis, however, which is motivated by practical planning problems of industrial cooperation partners, node failures are an issue. Furthermore, the set of possible logical links may be restricted by a maximum geographical length, and the length of a logical link affects its set of installable capacity types and their cost. We thus use an explicit model with a predefined set of logical links with their physical representation.

Our model is inspired by an SDH-over-WDM or MPLS-over-WDM scenario, where every lightpath consumes exactly one wavelength on a fiber. In other technological settings, a logical link may consume more than one unit on a physical link. This can be easily modeled

by changing the coefficients of the logical link capacity variables y_ℓ^m in Constraint (3.4j) from 1 to some higher value.

3.3.2 Physical link model

Depending on the specific context, physical links have been modeled in various ways in the literature: fixed preinstalled capacities [11, 14, 41, 43, 48, 49, 70, 111, 118, 119, 129, 141], continuous capacities [85], or one or several capacity types which can be installed in integer multiples (called *modular* capacity model in [105]), or one out of which has to be chosen (*explicit* capacity model) [18–20, 67, 68, 76, 81, 84, 86, 102, 104].

The choice of the physical link model depends on the considered technological setting. First, physical capacities can be fixed if a network operator wants to design (or reconfigure) a logical network on top of an existing physical network, or both layers can be planned together. Second, parallel physical connections can be modeled in two different ways: either as parallel physical links which fail separately, or as a single physical link which fails as a whole. With parallel physical links, the surviving ones can be used for recovery purposes in case of a failure. If, in contrast, parallel connections are aggregated into one physical link, their capacity is aggregated as well, and all of these connections fail together.

Which of these models is appropriate depends on the context: in case of an optical physical network, parallel fibers are usually bundled together and thus also fail together; parallel radio links, on the other hand, usually fail separately because the devices at their end-nodes are independent. In our models, both settings can be described: the first one by using integer z_e variables as in the above models, and the second one by introducing parallel physical links and restricting the z_e variables to $\{0, 1\}$.

Notice that the parameter B_e may vary from one physical link to the other. This may happen in practice if some wavelengths on a fiber are reserved for other purposes and cannot be freely disposed of during the planning process.

3.3.3 Node model

In most papers on single- and multi-layer network design, node hardware has not been mentioned at all, or it has been deliberately ignored because node cost was assumed to be small compared to the link cost. In several technologies, however, network nodes do cause some restrictions. First, cross-connects installed at the nodes have a limited number of slots for port cards and a limited switching capacity, and second, the switching granularity of a cross-connect may be restricted, see Section 2.1.3.

In the few multi-layer publications where node hardware is taken into account, it is modeled by a set of devices with different switching capacity and cost, at most one out of which must be chosen [11, 19, 20, 81, 102]. We adopt the same node model. The reason is that this thesis focuses on the structure of multi-layer networks imposed by embedding logical links in the physical layer. The goal is to describe the most important aspects of multi-layer networks in several technological contexts rather than hardware details of a specific technology. The above node model takes the most important restriction into account, namely switching capacity and cost, and can be used in several application settings. Our node modules represent combinations of logical and physical layer devices at a node. A restricted switching granularity of a node module can be taken into account by restricting the set of installable capacity types on the incident logical links depending on the choice of node module (if an

integer multi-commodity flow routing is used). Changing to a more detailed model like the one of Kröller and Wessäly [82] neither affects the basic structure of the planning problem nor the used algorithms.

In general, the logical nodes may form a subset of the physical nodes; We assume the two node sets to be equal for notational convenience; this does not impose any restriction. First, it is always possible to introduce artificial logical nodes with a zero-cost node module and infinite switching capacity, such that the sets of nodes are equal on both layers. Second, as demands are defined between logical nodes, every source or target node of a demand must have some device both at the logical and the physical layer; in our test instances, all nodes are end-nodes of some demand.

3.3.4 Objective function

Different objective functions have been considered in publications on multi-layer network design:

- minimize total installation cost [18–20, 41, 52, 67, 68, 76, 81, 86, 93, 102, 104, 129],
- maximize throughput with respect to a fixed physical network [43, 118, 119, 141],
- minimize the maximum load on any logical link [48],

or various combinations and variants thereof [49, 84, 85, 111]. Sometimes, measures like the number of needed wavelengths or the average number of routing hops per demand are used as an easy-to-compute substitute for the installation cost [14, 70]. Again, the choice of objective function depends on the planning context: minimizing installation cost is mostly used in strategic network design, maximizing traffic throughput in a completely or partially fixed network is an important task in short-term traffic engineering, and minimizing the maximum load is a typical objective in IP networks to avoid congestion problems. As this thesis deals with the strategic dimensioning of a network, we minimize total installation cost.

3.3.5 Integer vs. fractional flow

In our models, we choose a subset of the available logical links together with a physical representation and install integer capacities on these logical links. Hence, the routing of logical link capacities in the physical layer in these models can be interpreted as an integer multi-commodity flow routing with a limited set of routing paths.

For routing of demands in the logical layer, on the other hand, we assume a fractional multi-commodity flow routing. Flow can be arbitrarily split among several routing paths. From a practical point of view, a single-path routing or an integer multi-commodity flow routing would be more appropriate in most technologies, because also the demands are defined in terms of few granularities (VC-12, VC-4, STM-1) which have to be routed in integer units. We have nevertheless chosen a fractional multi-commodity flow routing for several reasons:

- With a fractional multi-commodity flow, all demands between the same pair of end-nodes can be aggregated into a single demand by expressing the demand values in a common base unit. With an integral routing, communication requests given in terms of different granularities have to be modeled as separate demands, thus increasing the size of the mixed-integer programming formulation.

- The solution times are much smaller with fractional flow variables. Moreover, Benders decomposition can be applied to generate metric inequalities by solving a linear program (see Section 4.5). With integer flow variables, an integer flow problem would have to be solved to generate a single metric inequality.

- We have observed that in good solutions, large parts of the routing are often integral even if this is not explicitly required. In many cases, a fractional routing can be made integer by rerouting some flow and installing very few additional capacities. Other authors have observed the same effect, see Baier et al. [11]. This means that a lower bound on the total network cost computed for a fractional routing is still useful for the problem with an integral flow. In practice, an common way of dealing with this issue is to reduce the installable capacities in the optimization process by, say 5-10%, and to round up fractional flows afterwards. Such downscaling of capacities is often done anyway to avoid fully loaded links.

- Several publications on multi-layer networks (for instance, Höller and Voß [68]) assume an unsplittable routing on a single shortest path with respect to the hop count or km-length because this is the predominant routing mechanism in IP networks. Modern IP protocols, however, like equal-cost multi-path routing (ECMP) or the path-based multi-protocol label switching (MPLS) protocol allow to split the traffic for a demand on several paths. Fortz and Thorup [53, 54] showed that under realistic assumptions, ECMP is very close to a fractional multi-commodity flow.

Instead of enforcing an integral flow in the MIP formulation, we thus admit a fractional multi-commodity flow routing. If needed, the solutions can be post-processed to make the routing integer, possibly by installing additional capacities. For strategic network planning in practice, this approach is a reasonable compromise between the quality of the estimation of the optimal network cost and the computation time.

3.3.6 Wavelength assignment in optical networks

In an SDH over WDM setting, the logical links correspond to lightpaths in the fiber network, and wavelengths must be assigned to the lightpaths such that any two lightpaths sharing a common fiber have different wavelengths.

We have deliberately chosen not to explicitly assign wavelengths to the lightpaths in our model. First, it is technology-specific, and second, finding a conflict-free wavelength assignment is an extremely hard problem on its own. In all previous attempts to integrate both the multi-layer network design problem and the wavelength assignment problem in a single MIP, the model could either only be solved on very small network instances [70, 141], or the authors did not even try to use the model for computations [48, 49, 111].

Instead, our models limit the number of logical links that may traverse a given physical link. In an optical networking setting, we propose to impose this limit on the maximum number of lightpaths on each fiber, and to solve the wavelength assignment and converter installation problem in a postprocessing step on the best computed solutions, as done in [43, 79]. In a computational study on realistic networks [80], this approach caused at most a marginal increase in the overall installation cost on practical instances.

3.4 Modeling multi-layer routing and survivability

A major difficulty in multi-layer network planning with survivability constraints is that single physical link or node failures may induce multiple link failures at the logical layer. Moreover, logical end-to-end routing paths may contain physical loops. These two facts have important consequences both on the survivability model and on the computational complexity of certain subproblems, as discussed in this section.

3.4.1 Routing paths may have physical loops

In multi-layer networks, the physical representation of a logical end-to-end routing path may contain nodes or physical links more than once.

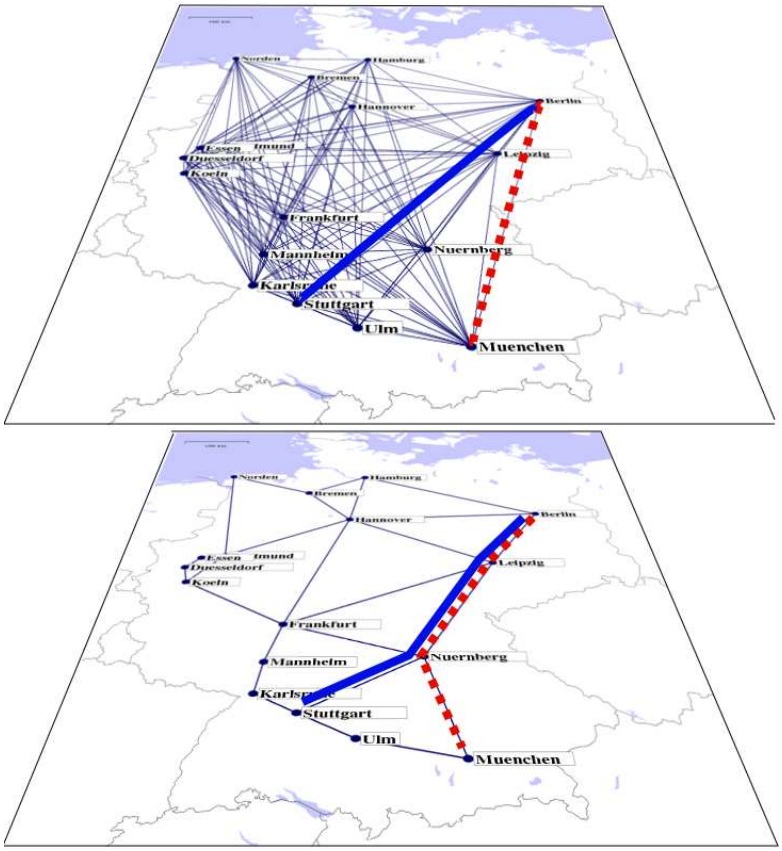


Figure 3.1: Routing paths at the logical layer may contain physical loops, and such a construction may save cost. If the two logical links have free spare capacity, routing a demand from Munich to Stuttgart via Berlin may be cheaper than setting up a new direct logical link between Berlin and Stuttgart, although the demand traverses the segment Nuremberg–Berlin twice.

Allowing end-to-end routing paths which contain a node or physical link more than once may save cost. At first glance, this may seem counterintuitive. Indeed, in a simple setting where all logical links are admissible, node hardware and survivability are not considered, and any fractional capacity can be installed on the logical links, physical loops do not make sense because removing the loop would yield another solution with less capacity requirements. If,

on the other hand, any of the above assumptions is violated, considering paths with physical loops can make sense. For example, if capacities can only be installed in integer multiples, such a path may save cost by exploiting slack capacities on the logical links. For this reason, end-to-end paths with node or physical link repetitions are allowed both in practice and in our models.

This is illustrated in Figure 3.1. Assume that the two logical links from Munich to Berlin and from Berlin to Stuttgart have a large capacity installed which is only partially used. To route a demand from Munich to Stuttgart, it may be wise to use this free slack capacity instead of paying for an additional direct logical link between these nodes.

There is one situation where logical paths with physical loops are not desired: according to our industrial cooperation partner Nokia-Siemens Networks, routing a demand through one of its end-nodes and back is not a particularly popular option in practice. In Figure 3.1, for example, the logical link between Munich and Berlin should not be used to route a demand targeted to Leipzig, because the flow would pass through its destination node Leipzig and then back to that node. It is hard to explain the potential benefits of such a seemingly weird routing to a practitioner who has to implement the solution in the field. As different network layers are often under the responsibility of different departments of a network operator's company, employees at this level tend to have a local view on "their" network layer (or even only parts of it), rather than a global view on two network layers simultaneously.

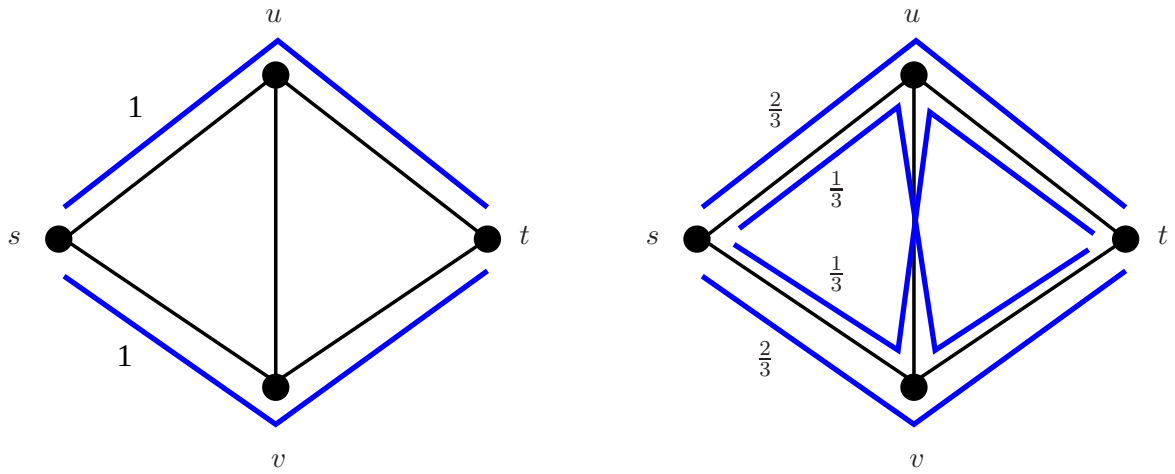
Together with our cooperation partner, we thus decided to forbid flow variables $f^k_{\ell,ij}$ for those logical links $\ell \in L$ whose physical path contains any end-node of a protected commodity $k \in K^p$ as an intermediate node. In contrast to the hard technical side constraints, this is a "weak" condition which is enforced only for protected point-to-point commodities. For unprotected commodities $k \in K^u$, on the other hand, we employ node-pair flow variables f^k_{ij}, which carry no information about the physical routing and thus do not allow to impose such conditions. If this restriction is a must, then also unprotected commodities have to be modeled as point-to-point commodities with flow variables $f^k_{\ell,ij}$. As this would dramatically increase the size of the model, we do not follow this approach.

3.4.2 Modeling 1+1 protection

As described in Section 2.1.5, the operator of a multi-layer network may have several options to reconfigure parts of the routing in case of an equipment failure. Which of these options are applicable or desirable depends on the technological context, long-term contracts, and other criteria. As we are dealing with transport networks, we assume that survivability against single physical link or node failures is ensured by 1+1 protection. This survivability mechanism, which is commonly used in transport networks, also provides survivability against single failures of ports or line cards.

With 1+1-protection, a demand is routed on two physically node- and link-disjoint paths simultaneously, and the target node chooses the better of the two arriving signals. If any single node or link fails, at least one of these paths survives (unless one of the end-nodes fails). This is illustrated in Figure 3.2(a). The assignment of backup paths for a given working path is fixed and does not depend on the failure state. One important consequence is that backup capacity cannot be shared between different demands depending on the failure state, but has to be reserved and preconfigured for each particular demand. Although this approach may require much backup capacity which is normally unused, it is attractive for network operators because nothing has to be reconfigured in case of a single node or link failure, such that all

connections can be continued without disruption.



(a) 1+1 protection: each unit of flow is routed twice on *exactly* two disjoint paths.

(b) Diversification: each unit of flow is routed twice on *at least* two paths in a way such that half of the flow survives.

Figure 3.2: Diversification relaxes the single-path constraint of 1+1 protection. With both concepts, at least half of the flow survives any intermediate single node or link failure. The numbers represent flow on routing paths for a protected demand of value 2 from s to t, one unit of which must survive.

Solving a mixed-integer program which exactly models 1+1 protection is hard. All such models from the literature involve either integer cycle variables or integer working and backup path variables for each demand. Both approaches lead to a huge number of integer variables. Solving such models with a branch-and-price approach is highly non-trivial because the branching and the pricing decisions influence each other. Furthermore, explicitly assigning working and backup flows to routing paths, as done in many formulations from the literature, introduces symmetries in the model because exchanging working and backup paths of a demand leads to an equivalent solution. With any of these options, much effort must be spent to obtain a valid lower bound on the optimal network cost from the MIP model.

Instead, we employ the diversification concept by Dahl and Stoer [40], a relaxation of 1+1 protection which can also be used with an edge-flow formulation. In general, this concept ensures that at most a certain fraction of each demand is routed through any node or edge. In our case, we have doubled the demand values for protected demands $r \in R^p$ in the definition of d^r, and ensure that at least a flow of $d^r/2$ survives any single failure state by restricting the amount of flow through a node or physical edge to $d^r/2$. This is illustrated in Figure 3.2(b). The diversification model is a relaxation of 1+1 protection because it allows to split the flow on more than two paths. It does not explicitly distinguish between working and backup paths (thus avoiding symmetries), but merely assigns a set of routing paths to each demand and failure situation, which can be used either for working or backup flow. The assignment of working and backup flow to the routing paths of a demand can be done in a postprocessing step in all but very few cases. In contrast to 1+1 protection, the failure routing of a demand depends on the specific failure situation with diversification. There exist indeed situations where a diversification routing cannot be decomposed into a state-independent routing for 1+1 protection, e. g., a system of four paths for a demand in which every pair of paths has

a link in common. On practical planning instances, however, we have observed that such situations can almost always be avoided by rerouting some flow within the given capacities of a solution. In Figure 3.2(b), for instance, the links on the upper and lower path must have capacity at least 1 and thus also support the 1+1 protection routing from Figure 3.2(a).

3.4.3 Consequences of multiple failures

In the following, we will discuss two possibilities of modeling diversification in multi-layer networks. Both models have slightly different interpretations in connection with multiple link failures, which appear naturally in multi-layer networks as a consequence of single failures in an underlying physical layer. These differences have important consequences on the definition of a feasible solution and on the complexity of several subproblems. The goal of this section is to highlight the implications of using one model or the other, and to motivate our choice of routing formulation in the model PROTECTED (3.4).

For notational ease, we will concentrate on single link failures first and address node failures only at the end of the section. We will start with a brief review of mathematical models for diversification in single-layer networks and extend them to multi-layer networks in a second step. The focus of this section being on topological issues, we ignore any capacity restrictions for the time being.

Diversification with single link failures

To begin with, consider only a single logical network layer $G_L = (V, L)$ without an underlying physical layer. Let $k \in K^p$ be a demand of value 1 which has to be routed from s^k to t^k and protected against the failure of any single link $\ell \in L$ (i. e., $d^k = 2$). This leads to the demand and diversification constraints (3.5). In accordance with the notation from model (3.4), the values d_i^k, $i \in V$, are 2, -2, or 0. As an alternative, model (3.6) provides a path-flow formulation for this single-commodity routing problem. The set P^k denotes the set of all undirected paths on logical links from s^k to t^k which do not contain any node more than once. The variable f_p^k denotes the flow for commodity k on path $p \in P^k$.

$$\sum_{j \in V} \sum_{\ell \in L_{ij}} (f^k_{\ell,ij} - f^k_{\ell,ji}) = d_i^k \quad \forall\, i \in V \tag{3.5a}$$

$$f^k_{\ell,ij} + f^k_{\ell,ji} \leq 1 \quad \forall\, \ell \in L \tag{3.5b}$$

$$f^k_{\ell,ij}, f^k_{\ell,ji} \in \mathbb{R}_+ \tag{3.5c}$$

$$\sum_{p \in P^k} f_p^k = 2 \tag{3.6a}$$

$$\sum_{p \in P^k:\, \ell \in p} f_p^k \leq 1 \quad \forall\, \ell \in L \tag{3.6b}$$

$$f_p^k \in \mathbb{R}_+ \tag{3.6c}$$

It is easy to see that these two formulations are equivalent in the sense that every solution without cycle flows of the edge-flow model can be transformed into a solution of the path-flow model, and vice versa. This holds as long as only single link failures are considered, i. e., either only one network layer is considered, or every physical link is contained in at most one logical link.

Diversification with multiple link failures

In a more general two-layer setting, failure of a physical link $e \in E$ induces the failure of the whole set L^e of logical links traversing e. In this case, the diversification constraints (3.5b)

of the above link-flow formulation have to be defined for each *physical* link, similarly to the link diversification constraints (3.4e) in our PROTECTED model:

$$\sum_{\ell \in L^e} (f^k_{\ell,ij} + f^k_{\ell,ji}) \leq 1 \qquad \forall\, e \in E \tag{3.7}$$

An important consequence of this formulation is that if a physical link $e \in E$ is traversed by m logical links of the same routing path, the flow on that path is counted m times on the left-hand side of (3.7), and thus restricted to at most $1/m$ (more generally, to $d^k/2m$). Consequently, more than two routing paths may be needed between the end-nodes of a demand to ensure a survivable routing. In Figure 3.1, for instance, the flow on the indicated routing path from Munich to Stuttgart via Berlin is limited to $d^k/4$ because the path traverses the physical link between Berlin and Leipzig twice.

In terms of path-flow variables, a possible definition of the diversification constraints is

$$\sum_{p \in P^k:\, p \cap L^e \neq \emptyset} f^k_p \leq 1 \qquad \forall\, e \in E. \tag{3.8}$$

The sum is taken over all routing paths which traverse at least one logical link containing physical link e. In contrast to the edge-flow constraint (3.7), the path-flow constraint (3.8) counts the flow on each routing path only once, regardless of how often a specific physical link is used. In particular, the edge-flow formulation (3.7) and the path-flow formulation (3.8) of the diversification constraints are not equivalent.

At first glance, the path-flow version (3.8) seems to be more natural than the edge-flow version (3.7) because if a physical link $e \in E$ fails, the flow on a routing path through e fails only once, even if it traverses the link twice. Unfortunately, this formulation also has some drawbacks, as shown below. We will now discuss the implications of the path-flow formulation both from a practical and from a mathematical and computational point of view, and motivate why we have chosen an edge-flow formulation in our models.

A path-flow formulation may save cost

Apart from the more intuitive interpretation, the main advantage of the path-flow version is that it admits more cycle-free routings than the edge-flow version. As cycle flows may only occur in our solutions due to is slack in the capacity constraints and do not affect the network cost, this implies that the path-flow version may allow for cheaper network configurations. To illustrate this, we will consider a simplified version of the survivable multi-layer network design problem with only one demand where node hardware and physical link capacities are ignored. For this routing subproblem, we give an example of a network where the difference between the two formulations can be made arbitrarily large by varying the input parameters. The result can be transferred to the original problem.

Suppose that a single protected demand $k \in K^p$ with value 1 has to be routed from s^k to t^k in the logical layer in such a way that it survives any single link failure. Moreover, suppose that every unit of flow routed on a logical link $\ell \in L$ incurs a cost of $\kappa_\ell \geq 0$. The goal is to find a minimum cost routing on the logical links which is survivable against any physical link failure. This *minimum-cost survivable routing* problem is solved at various places in our algorithm to identify interesting routing paths. With the two diversification versions described above, this problem can be formulated as follows:

$$\min \sum_{\ell \in L} \kappa_\ell (f^k_{\ell,ij} + f^k_{\ell,ji}) \tag{3.9a}$$

$$\sum_{j \in V} \sum_{\ell \in L_{ij}} (f^k_{\ell,ij} - f^k_{\ell,ji}) = d^k_i \quad \forall\, i \in V \tag{3.9b}$$

$$\sum_{\ell \in L^e} (f^k_{\ell,ij} + f^k_{\ell,ji}) \leq 1 \quad \forall\, e \in E \tag{3.9c}$$

$$f^k_{\ell,ij}, f^k_{\ell,ji} \in \mathbb{R}_+ \tag{3.9d}$$

$$\min \sum_{p \in P^k} \sum_{\ell \in p} \kappa_\ell f^k_p \tag{3.10a}$$

$$\sum_{p \in P^k} f^k_p = 2 \tag{3.10b}$$

$$\sum_{p \in P^k \colon p \cap L^e \neq \emptyset} f^k_p \leq 1 \quad \forall\, e \in E \tag{3.10c}$$

$$f^k_p \in \mathbb{R}_+ \tag{3.10d}$$

Proposition 3.2. *The path-flow formulation (3.10) is a strict relaxation of the edge-flow formulation (3.9) in the sense that every solution of the edge-flow version can be transformed into a path-flow solution with at most the same cost, but not necessarily vice versa.*

Proof. First, let $(f^k_{\ell,ij}, f^k_{\ell,ji})_{i,j \in V,\, \ell \in L_{ij}}$ be a solution of (3.9). Without loss of generality, we may assume that this solution does not contain cycle flows at the logical layer; otherwise, removing them yields another solution with less capacity requirements and cost. Without diversification constraints, a standard result of network flow theory states that this solution can be transformed into a path-flow routing $(f^k_p)_{p \in P^k}$ such that

$$f^k_{\ell,ij} + f^k_{\ell,ji} = \sum_{p \in P^k \colon \ell \in p} f^k_p, \tag{3.11}$$

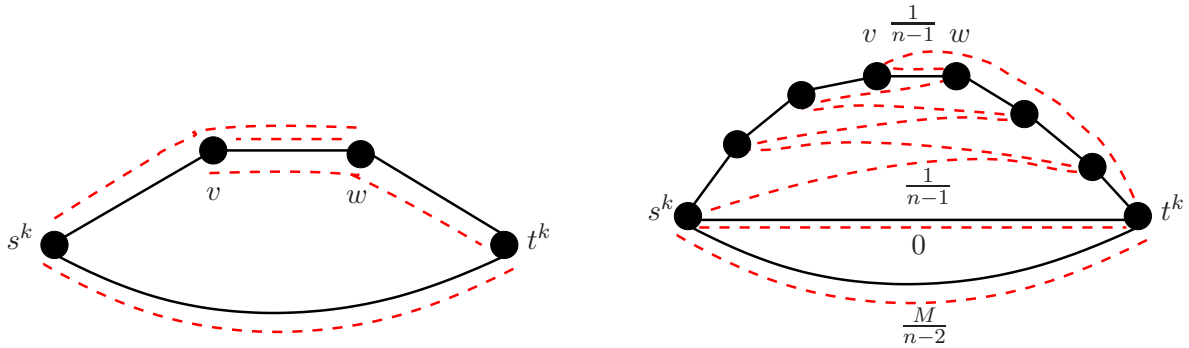
and vice versa, see Ahuja et al. [5]. The diversification constraints do not affect this result but only limit the flow on the logical links; the important point is that the flow conservation constraints are satisfied. The resulting path-flow routing fulfills the corresponding diversification constraints (3.10) for every physical link $e \in E$:

$$1 \geq \sum_{\ell \in L^e} (f^k_{\ell,ij} + f^k_{\ell,ji}) = \sum_{\ell \in L^e} \sum_{p \in P^k \colon \ell \in p} f^k_p = \sum_{p \in P^k} |p \cap L^e|\, f^k_p \geq \sum_{p \in P^k \colon p \cap L^e \neq \emptyset} f^k_p \tag{3.12}$$

The first inequality comes from the edge-flow diversification constraints (3.9c), and the last one from the fact that $p \cap L^e \neq \emptyset$ is equivalent to $|p \cap L^e| \geq 1$. The second equality reflects the fact that in the edge-flow version, every unit of flow on a routing path is counted as often as the path contains the physical link e.

To see that the path-flow version admits solutions which cannot be transformed into a feasible edge-flow routing, consider the graph depicted in Figure 3.3(a). It shows a network with four nodes, four physical links (black, solid), and four logical links (red, dashed). There are only two possible logical routing paths from s^k to t^k. As the upper path uses the physical link between v and w three times, it can carry at most a flow of $1/3$ with the edge-flow formulation. Consequently, it is not possible to route two units of flow in a survivable way according to (3.9). In contrast, the path-flow version (3.10) allows to route one unit of flow on each of the two logical routing paths. This shows that in the sense defined in this proposition, the edge-flow formulation is strictly stronger than the path-flow formulation. □

The following proposition shows that the edge-flow formulation can be arbitrarily bad compared to the path-flow formulation. The idea of the proof is to force parts of the flow to use an expensive path by restricting the flow on cheaper routing paths with the edge-flow diversification constraints. With the path-flow formulation, the flow may use the cheap path. By adjusting the cost, the difference between both versions can be made arbitrarily large.



(a) A network with two logical routing paths from s^k to t^k. With $d^k = 2$, the upper path can carry a flow of 1 with the path-flow formulation but only 1/3 with the edge-flow formulation.

(b) A network with n nodes and three logical routing paths from s^k to t^k. All $n-1$ upper logical links traverse the edge v–w and have cost $1/(n-1)$. The two lower logical links have cost 0 and $M/(n-2)$.

Figure 3.3: Example networks for Propositions 3.2 and 3.3.

Proposition 3.3. *There exist networks where the ratio between the optimal objective values of (3.9) and (3.10) can be made arbitrarily large by adjusting the cost values.*

Proof. Consider the network depicted in Figure 3.3(b). It shows a network of $n \geq 4$ nodes, $n+1$ physical links, and three physical paths from s^k to t^k. For ease of argumentation, assume that n is even; the case where n is odd is very similar. The upper path contains all n nodes. On this physical path, $n-1$ logical links are defined in the way shown in the figure: from s^k to the next-but-last node, from there to the second node, and so on. This scheme goes on until node w, which is the $(n/2+1)$-th node on the line, then to v, which is the $n/2$-th node, and finally to t^k. As a consequence, there is only one possible logical path in the upper part of the network. It traverses the physical link v–w exactly $n-1$ times; consequently, its flow is restricted to $1/(n-1)$ by the diversification constraints (3.9c). As all of these $n-1$ logical links have a cost of $\kappa_\ell = 1/(n-1)$, the upper path incurs a cost of 1 per flow unit.

The lower part of the network contains two direct logical links between s^k and t^k, each of them consisting of only one physical link. These logical links have cost $\kappa_\ell = 0$ and $\kappa_\ell = M/(n-2)$ per flow unit, respectively, where $M \geq n-1$. In particular, the lowest path is more expensive than the topmost path.

With the path-flow formulation (3.10), the cheapest way to route the flow in a survivable way is to send one unit of flow on the upper and the middle logical path, which incurs a total cost of

$$c_{PF} = 1 \cdot 1 + 1 \cdot 0 = 1.$$

With the edge-flow formulation, one unit of flow is sent on the middle path at zero cost as well. As the flow on the cheaper upper path is restricted to $1/(n-1)$, a flow of $(n-2)/(n-1)$ must be routed on the expensive lower path. The total cost of this routing is

$$c_{EF} = \underbrace{1 \cdot 0}_{\text{middle path}} + \underbrace{\frac{1}{n-1} \cdot 1}_{\text{upper path}} + \underbrace{\frac{n-2}{n-1} \cdot \frac{M}{n-2}}_{\text{lower path}} = \frac{M+1}{n-1}.$$

By adjusting M, the ratio c_{EF}/c_{PF} can be made arbitrarily large. □

The reason for the potential cost advantage of the path-flow version is that a value of $d^k/2$ can be routed through any physical link an arbitrary number of times, which is not possible with the edge-flow version. Unfortunately, this cost advantage comes at the expense of such "weird" routings as in Figure 3.3(b). Network operators tend to avoid such logical paths in practice to simplify the management of the network. Although the edge-flow version (3.8) does not strictly forbid such paths, it at least limits the flow on them.

A path-flow formulation makes important subproblems $\mathcal{NP}$-hard

A serious computational drawback of the path-flow formulation is that it makes several important subproblems $\mathcal{NP}$-hard, whereas their edge-flow versions can be solved in polynomial time by a linear program. The reason is that multiple link failures may occur at the logical layer. Consequently, all shortest-path or shortest-cycle problems, which naturally appear as subproblems in column generation, primal heuristics, and other contexts, have to cope not only with weights on single links, but with weights on *subsets* of logical links failing together.

One example of such a subproblem is the minimum-cost survivable routing problem as described in (3.9) and (3.10). In the edge-flow version, it can be solved in polynomial time because the LP formulation (3.9) is polynomial in the size of the network. In the path-flow version (3.10), the number of flow variables can be exponential in the network size. Without capacity constraints, an optimal solution is to route one unit of flow on each of two physically disjoint logical paths with minimum total cost. Hu [69] showed by reduction to the set splitting problem and the node covering problem [55] that finding such a routing is $\mathcal{NP}$-hard.

With the path flow-formulation (3.10), path variables have to be generated dynamically during the algorithm to obtain a feasible dual bound with respect to all variables. To find a potentially improving variable or to determine that no improving variable exists, the so-called *pricing problem* must be solved. Straightforward application of LP duality shows that the pricing problem corresponding to (3.10) consists of finding a path p from s^k to t^k minimizing

$$c(p) := \sum_{\ell \in p} \kappa_\ell + \sum_{e \in E:\, p \cap L^e \neq \emptyset} \rho_e^k,$$

where ρ_e^k are the dual variables of the diversification constraints in the current LP solution. By setting the values κ_ℓ to 0, this problem contains the $\mathcal{NP}$-hard *minimum-color shortest path problem* (MC-Path) as a special case.

The MC-Path problem, also known as *minimum label shortest-path problem*, is a variant of the shortest path problem where a set of colors is assigned to every link. Each color has a cost, and the goal is to find a shortest path with respect to the color cost. In contrast to the traditional shortest path problem, the weight of each color contained in the path is counted only once even if the path contains several links of that color. This problem has been shown to be $\mathcal{NP}$-hard for a general color setting, and various inapproximability results for MC-Path are known [38, 65, 140].

By defining a color for every physical link $e \in E$, assigning it to every logical link $\ell \in L^e$, and setting $\kappa_\ell = 0$, the above pricing problem corresponds exactly to an MC-Path problem in the logical layer. Hence, this pricing problem is $\mathcal{NP}$-hard as well. Details can be found in a recent survey on column generation in network design by Orlowski and Pióro [100].

In view of these results, one might think that it was just a bad idea to choose the diversification concept as survivability model, and that maybe some other survivability mechanism could be handled more easily. Unfortunately, this is not true. The complexity is inherent in

the fact that we have to deal with multiple failures at the logical layer. Orlowski and Pióro [100] showed that the pricing problem for path variables with multiple logical link failures is $\mathcal{NP}$-hard for basically every path-based survivability mechanism that has been considered in the network design literature so far. The only exception is the impractical concept *unrestricted reconfiguration* (also known as *reservation* [128]), where the routings in different network situations are completely independent of each other. The difficulty is thus not caused by the survivability concept but inherent in the presence of multiple logical link failures.

As a consequence of these results, it is hard to obtain a valid dual bound for the overall planning problem with the path-flow formulation, because already the variable generation problem is $\mathcal{NP}$-hard, as well as several other important subproblems.

Conclusions

In view of the results described in the previous sections, we decided to go for an edge-flow formulation in our models for several reasons:

- From a practical point of view, the potential cost advantage of the path-version flow can only be exploited by using routing paths with many physical node or link repetitions. Such paths are not desired in practice.
- From a computational point of view, the edge-flow formulation allows us to compute feasible lower bounds on the network cost using more or less standard linear and integer programming methods. With the path-flow formulation, many important subproblems are $\mathcal{NP}$-hard and it is difficult to obtain a valid dual bound at all, as shown above.

Even though this survivability model may not be perfect, it is close enough to reality for strategic planning purposes. It has also been used in projects with our cooperation partner Nokia-Siemens Networks, see Koster et al. [81].

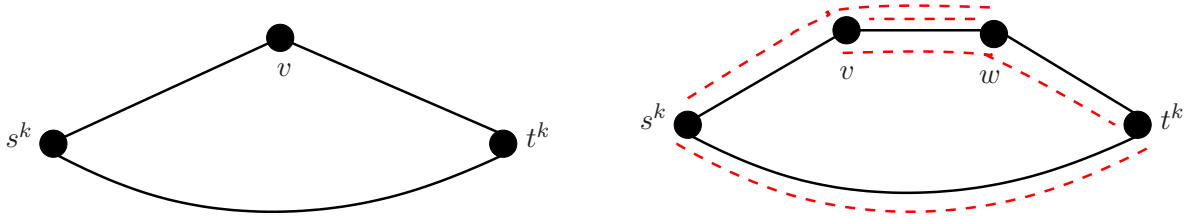
Notice that if the diversification constraints in (3.10) are replaced by

$$\sum_{p \in P^k} |p \cap L^e| \, f_p^k \leq 1 \qquad \forall\, e \in E \tag{3.13}$$

such that the flow of each path is counted as often as the path contains link $e \in E$, the resulting routing formulation is equivalent (up to cycle flows) to the edge-flow model (3.10), and the pricing problem reduces to a shortest-path problem on the logical links. We use the edge-flow formulation nevertheless. On smaller instances, this allows us to solve the problem without column generation. On larger instances, our path-based column generation procedure described in Section 4.6.3 "simulates" the corresponding procedure with the path-flow formulation (3.13), but allows us to generate logical link flow variables which can be used by many new routing paths simultaneously.

Diversification with node failures

In our model with survivability (3.4), we consider not only single physical link failures, but also node failures. All results from this section can be transferred to node failures. There is, however, one additional issue that has to be taken into account with an edge-flow formulation, namely that more link diversification constraints are needed in a multi-layer network than in a single-layer context.



(a) Single-layer: with a demand of value 2 from s^k to t^k, the flow through the links incident to v is restricted to 1 by the node diversification constraint of v. The flow on the direct s^k-t^k-link has to be restricted by the corresponding link diversification constraint.

(b) Multi-layer: if only the node diversification constraints (3.4d) of v and w are added, the logical link v–w may carry a flow of 1/2. This violates the link diversification constraint (3.4e) of the physical link v–w, which restricts the flow to 1/3.

Figure 3.4: Node diversification is not sufficient with a multi-layer routing.

In a single-layer survivable network design problem, the diversification constraint for most links $\ell \in L$ is dominated by the node diversification constraints of the end-nodes of ℓ, because all flow passing through ℓ also passes through its end-nodes. The only exception are direct links between the end-nodes s^k and t^k of the commodity k, because the flow through these nodes is not restricted. This is illustrated in Figure 3.4(a). Hence, if all node diversification constraints for a commodity are included in the model, link diversification constraints are only needed for direct links between the commodity end-nodes.

In a multi-layer network, this is not necessarily true, as Figure 3.4(b) shows. Suppose that the demand value from s^k to t^k is 2, and 1 unit must survive any single physical failure. The flow on the upper path traverses nodes v and w twice but passes three times by the physical link v–w. Hence, even if the flow through nodes v and w is restricted to 1/2 by the node diversification constraints (3.4d) for v and w, it may violate the link diversification constraint (3.4e) for the physical link v–w, which only allows a flow of 1/3.

Even though such situations appear in our computed routings, they are rare. In most obtained LP solutions, the link diversification constraints are satisfied if all node diversification constraints are included in the LP. We thus add only the link diversification constraints for direct s^k-t^k-links from the beginning and generate the other ones dynamically when needed.

3.5 Literature review

The goal of this section is to review previous work on multi-layer planning problems and to relate this thesis to the existing literature.

First mathematical models and solution methods for single-layer network design have emerged during the 1980s. It was only in the middle of the 1990s that practically relevant survivability restrictions could be taken into account. At that time, planning two layers together was out of question for computational reasons. Although practical networks have always been composed of several layers, already planning a single network layer with realistic side constraints was (and still is) a complex task. By the end of the 1990s, algorithms and computation power were advanced to a point such that first restrictions from an underlying physical layer could be considered in the planning process. A fully integrated planning of two network layers, even without survivability constraints, has come into reach only a few

years ago. The development of computationally useful multi-layer models and algorithms with realistic survivability restrictions has just recently started with Koster et al. [81], Kubilinskas [83], and the work presented in this thesis.

According to the focus of this thesis, the following literature review concentrates on offline planning approaches for arbitrary network topologies which provide a serious mathematical model or a solution approach that goes beyond simple shortest-path computations. There is a plethora of literature on shortest-path based approaches, see Arakawa et al. [7], Konda and Chow [78], and references therein.

Other authors have focused on topological subproblems, see Armitage et al. [8], Lee et al. [87], Modiano and Narula-Tam [94, 95], and Todimala and Ramamurthy [131]. Many publications, notably among those which appeared around the millennium change, focus on physical ring networks, see Gerstel et al. [56, 57], Sasaki and Lin [122], Battiti and Brunato [17], Berry and Modiano [23], Cho et al. [32], Cinkler et al. [36] and references therein, Kandula and Sasaki [75], Srinivasan and Somani [126], Dutta and Rouskas [50], Wang et al. [134], Wang and Mukherjee [133], and Bermond and Ceroi [22]. Although rings are a common network substructure in practice, the corresponding planning tasks are, of course, also covered by more general models for arbitrary network structures.

Furthermore, there are a few publications where the demands have the same granularities as the logical link capacities (usually a lightpath granularity of 2.5 Gbit/s), see Melian et al. [93], Höller and Voß [67, 68], and Dawande et al. [43]. Under this assumption, the grooming subproblem vanishes, all demands can be routed on direct logical links from their source to their destination, and the planning task reduces to a single-layer network design problem.

Several papers discussed in this section present mixed-integer programming models which are not used for computations, especially when survivability constraints or wavelength assignment for lightpaths are involved. Although this may seem fruitless at first glance, a mathematical model can help to understand and precisely define the planning task and serve as a basis for further discussion even if it is too complex to be of computational use. This is especially true for multi-layer planning tasks, where already modeling the practical problem is non-trivial.

For computational reasons, many authors heavily restrict the number of demands, logical links, or admissible routing paths in their test instances. In the following literature overview, the *density* of a graph with n nodes is defined as the number of links divided by $n(n-1)$, i. e., by the number of links in a complete graph. A complete graph has density 1. In our test instances, which have between 12 and 67 nodes, the density of the logical layer ranges from 0.31 to 36.98, where each logical link corresponds to exactly one physical path. Between each pair of nodes, we have defined 2–5 parallel logical links in most instances, but this number ranges up to 50 for our smallest network. The density of our demand graphs ranges from 0.74 to 1, and all routing paths are admissible at the logical layer.

The literature review is divided into two parts. In Section 3.5.1, we review papers where the physical layer is assumed to be fixed and installed in advance. Hence, the physical network does not incur cost, but imposes capacity restrictions and sometimes survivability constraints. In Section 3.5.2, we summarize those publications where the physical layer is designed together with the logical layer. Within these sections, the publications are sorted roughly chronologically and by classes of mathematical models. Eventually, we draw some summarizing conclusions in Section 3.5.3 and discuss the added value of this thesis.

3.5.1 Fixed physical layer

With a fixed physical layer, the planning task is to determine a demand routing and a suitable logical network supported by the physical link capacities.

Dahl et al. (1999) [41] Mathematical research on multi-layer network design has been initiated by Dahl, Martin, and Stoer. Given are a fixed physical fiber network with capacities and a set of logical links defined by short physical paths. The authors present a mixed-integer programming (MIP) model to select a subset of these links, to install one base capacity in integer multiples on the logical links within the physical capacities, and to route demands of value 1 or B through the logical network on a single path.

The model is solved using a branch-and-cut algorithm with additional knapsack cover and cutset inequalities, as well as a subclass of the newly introduced hypo-matchable inequalities. For the latter type of inequalities, which are derived from the unsplittable flow requirements and knapsack cover inequalities, the authors provide sufficient facet conditions. Until recently, this paper has been the only one where the polyhedral structure of multi-layer networks has been studied. To compute feasible solutions, the authors present a diving heuristic.

The algorithm is tested on 10 modified real-world networks with 27–62 physical nodes and 44–81 physical links, sparse logical networks (5–56 logical nodes and logical link density 0.06–0.21), and few demands (density 0.002–0.05). With varying cost parameters for the logical links, they can solve all of the small instances (those with up to 4202 variables) to optimality at the root node within a few seconds. On the large instances, the computation was stopped with gaps of 5–9% after several hours.

Banerjee and Mukherjee (2000) [14] Banerjee and Mukherjee present a MIP model for planning a logical lightpath network and a routing of demands in it. One type of capacity is available on the lightpaths, and each of the fixed physical fibers supports a limited number of them. The model comprises two edge-flow formulations, one to route the demands on the logical lightpath layer with a multi-commodity flow routing, and another one to choose a subset of these parallel paths for every logical link, i. e., to route the logical links in the fixed physical fiber layer.

The routing of the demands in the physical layer is restricted to few physical paths. The goal of the planning problem is to divide the physical paths into subpaths such as to minimize the average number of logical hops per demand, i. e., the number of logical links used. Using only direct end-to-end logical links is thus optimal if this solution is feasible. A physical path may have to be split into several logical links either because of limited physical capacities or because of a given limit on the number of lightpaths incident to each node (representing the available transmitters and receivers).

The network design model is solved using a black-box MIP solver and two shortest-path based construction heuristics. Numerical results are given on the 14-node NSFNET network, the 15-node PACBELL network, and a 20-node randomly generated network. The authors assume single-facility capacities on the lightpaths and few (1–7) available wavelengths on each fiber. For a series of 100 randomly generated demand patterns, they report that on average, Cplex terminates after 2 branch-and-cut nodes, and the LP value is within 0.1% of the integer optimal value. The heuristics are compared to the MIP approach on the NSFNET network with a limit of 1–7 wavelengths per fiber and at most 4–8 wavelengths incident to each node. The MIP approach produces better results; computation times are not given.

Dutta and Rouskas (2000) [48] The MIP model presented by Dutta and Rouskas is similar to the one of Banerjee and Mukherjee [14]. In contrast to the latter, Dutta and Rouskas assume a single-path routing of the logical links in the physical layer, which is further restricted by a hop limit. The logical links, if installed, have infinite capacity. The goal is to minimize the maximum load on any logical link, subject to the available physical capacity and the assignment of a wavelength to each logical link. The formulation exhibits lots of symmetries because of the wavelength constraints; no algorithm or computations are presented. Instead, the authors discuss some topological bounds on the objective function and provide an extensive literature survey on heuristic planning approaches. In Dutta's PhD thesis [47], the model from [48] is specialized to ring, star, and tree networks. The author presents various primal and dual heuristics for the corresponding planning problems. The presented MIPs are never used in the computation process.

Dutta and Rouskas (2002) [49] In this survey, Dutta and Rouskas provide a detailed explanation of the grooming problem, an extensive literature review, and some complexity results. The presented MIP is similar to the one presented in Dutta and Rouskas [48], but with varying objective functions and single-facility capacities on the logical links. Again, neither node hardware nor survivability constraints are considered, but wavelength assignment is part of the model. No algorithm or computations are given.

Hu and Leida (2002) [70] The edge-flow formulation provided by Hu and Leida is similar to the ones discussed above. Based on a fixed physical layer and a predefined set of logical links, a subset of these links has to be chosen and equipped with one type of lightpath capacities and a wavelength, subject to physical link capacities and wavelength blocking constraints. Demands must be routed in the logical network on a single path. The authors also extend the model by 1+1-protection against single physical link failures by introducing separate binary edge-flow variables for working and backup paths, as well as additional big-M constraints to take care of the wavelength bound. This approach introduces additional symmetries in the model. The goal is to minimize the number of wavelengths, subject to a single-path routing of demands in the logical network.

The authors solve the problem by decomposition. They relax the integrality constraints of some variables, solve the remaining problem (without wavelength assignment) as a MIP, and round up fractional solution values. In a second step, the wavelength assignment problem is heuristically solved using a greedy algorithm.

On the model without survivability, the authors compare the decomposition approach to solving the original MIP using Cplex. The test networks have 12–144 nodes, 17–162 physical links, a very sparse logical layer (density 0.36 on small networks, 0.03 on the largest ones), and few demands (density 0.78–0.03). The authors do not state whether the networks and demands are real or randomly generated. The full MIP formulation with or without wavelength assignment can be solved to optimality on the smallest network only. On the two large networks, Cplex does not find any feasible solution. The decomposition approach finds feasible solutions within less than 40 minutes on all test instances. On the smallest instance, it yields a 7% worse solution but runs 20 times faster than Cplex. According to the authors, the time required by the greedy wavelength assignment algorithm is negligible. The extended model with survivability constraints is not used in the computations.

Zhu and Mukherjee (2002) [141] Similarly to the previous models, Zhu and Mukherjee study the problem of designing a logical layer based on a fixed physical network. The objective is to maximize the total throughput in the network (weighted for each demand) by installing single-facility capacities on the logical links and routing the demands through the logical network. The restrictions are fixed physical link capacities, a maximum number of lightpaths incident to a node, and wavelength assignment constraints. Optionally, the authors impose a logical hop limit of 1, i. e., every demand has to be routed on a direct end-to-end path.

On a 6-node test network, the model can be solved using CPLEX as a black-box algorithm. A 15-node test network is solved using an iterative shortest-path based heuristic, followed by a greedy (first-fit) wavelength assignment heuristic. The authors study the effect of different values for the physical link and node restrictions. The logical network consists of all possible physical paths; the demands are randomly generated. No computation times are given.

Prathombutr et al. (2005) [111] Also Prathombutr, Stach, and Park use a similar model as the previous publications, with varying objective functions. The goal is to minimize the average traffic delay or the number of transceivers (i. e., the number of lightpaths), or to maximize traffic throughput. Single-facility capacities can be installed on the logical links, and demands have to be routed in the logical network using a fractional multi-commodity flow. Fixed physical link capacities and a limit on the number of lightpaths incident to a node have to be taken into account. The presented MIP formulation is not used in the computations. Instead, the authors compute feasible solutions for a 6-node test network and the 14-node NSFNET network using a multi-objective evolutionary algorithm. All logical links are admissible; the demands are randomly generated.

Sung and Song (2003) [129] In the context of virtual path layout in ATM networks, Sung and Song consider a model with infinite logical link capacities. They assume a single-path routing of demands in the logical layer and a single-path routing of the logical links in the physical layer. Hence, the goal is to compute one physical path for each demand and to divide it into logical links such that a maximum number of logical links on each physical link is not exceeded. Cost is incurred for the setup of a logical link, for every unit of flow on a given logical link, and for combinations of logical and physical links. The authors introduce two MIP models with binary variables only: one with edge-flow formulations on both layers, and another one with path-flow formulations on both layers.

To solve this model, the authors propose a branch-and-price algorithm where branching is performed on the binary logical link setup variables only. The authors claim that when these variables are fixed, solving the remaining problem as a linear program would yield an integral solution. Unfortunately, this claim, and thus the whole algorithm, are based on the wrong assumption that a single-path routing for each logical link within the physical link capacities could be determined in polynomial time using a linear program.

Baier et al. (2006) [11] Baier, Engel, Autenrieth, and Leisching present a mixed-integer programming model for multi-period planning with node-switching capacities and fixed physical link capacities. In contrast to most other multi-layer models, their MIP formulation assumes a multi-facility capacity model on the logical links to support lightpaths with different bandwidth. The goal is to minimize overall installation cost over all time periods. The model is solved using CPLEX as a black-box algorithm.

As the MIP is too complex to be solved for all time periods in one step, the authors investigate an incremental strategy where a network is computed for each time period separately. Under the assumption that all demands increase over time, they extend the network step by step. Capacities and routings from previous time periods have to be preserved in the next period, but existing spare capacities can be reused. This strategy is tested on a German network with 17 nodes, 26 physical links, and a complete demand matrix from the European NOBEL project [2]. The set of admissible logical links is defined by some short physical paths. To allow for a feasible solution for the last period, the physical link capacities have to be restricted first, being increased from one period to the other. This method is compared to a benchmark approach where the network can be freely reconfigured from one period to the other. The latter approach requires much less wavelengths.

Raghavan and Stanojević (2007) [118] Raghavan and Stanojević study a two-layer network design problem with a single-path routing of demands at the logical layer; this paper is based on Stanojević's PhD thesis [127]. The authors assume single-facility logical link capacities, fixed physical link capacities, and a fixed limit on the number of logical link capacity units incident to each node. The objective is to minimize the lost traffic under these restrictions. Alternative objectives are to minimize the average packet delay caused by the physical links, the maximum flow on any logical link, or the number of logical links incident to a node (instead of a fixed limit). The authors propose a series of MIP formulations where the flow variables are either defined by explicit logical links or by node-pairs, similar to our models presented in Section 3.

To solve the problem with all possible logical links, the authors develop a sophisticated branch-and-price algorithm where logical link variables are generated dynamically during the algorithm. The challenge is that every new logical link leads to both new variables and new constraints. By substituting the logical link capacities by the flow on logical links which are not yet part of the LP, the authors show that the pricing procedure consists of a series of shortest path problems on either of the two layers. When adding a new logical link variable, also the corresponding capacity constraints are added. To avoid interference of branching and pricing, various single-path based branching rules in the spirit of Barnhart et al. [16] are used on both layers in addition to standard dichotomy branching. Feasibility of all LPs in the branch-and-cut process is ensured by adding dummy variables. The pricing procedure generates many logical link variables which are equivalent to other variables forbidden by some branching rule. Hence, the authors price all required variables to compute a valid dual bound, but employ another LP formulation with incomplete pricing to compute feasible solutions.

To strengthen the LP relaxation, single-node cutset inequalities and lifted knapsack cover cuts on the logical capacity constraints are generated dynamically. The latter do not improve the dual bound because of the many equivalent variables generated by the pricing procedure.

The algorithm is tested on several test instances: first, on a 6-node network and the 14-node NSFNET network, both with 3–4 admissible wavelengths per fiber and 3–7 wavelengths per node; second, on a series of networks with a complete physical layer on 5–20 nodes, four different randomly generated traffic patterns, and at most two wavelengths per fiber; third, on the same series of networks but with a randomly generated subset of all physical links and 40 wavelengths per fiber. All possible logical links are admissible. The problem turns out to be hard to solve. Within the 1-hour time limit, hardly any instance can be solved to optimality, even on 5 and 7 nodes. On the optimally solved instances, all traffic can be accommodated,

i. e., the optimal objective value is 0. On the NSFNET network, only the trivial lower bound of 0 can be computed in most settings.

Raghavan and Stanojević (2007) [119] In this companion paper to [118], Raghavan and Stanojević study the same problem but with a multi-commodity flow routing on the logical layer. As a consequence, the single-path branching rules described in the latter paper are not applicable. In the described branch-and-price algorithm, the dual bounds of branching constraints may lead to negative cycles in the shortest-path problems solved during the pricing procedure. As the authors do not address this issue, it is not clear whether the pricing problems after the root node are solved exactly, and if the computed dual bounds are valid for the considered planning problem or not. According to personal communication with Raghavan [117], negative link weights (let alone negative cycles) did not occur in the computations, but the authors do not known any proof that they cannot occur.

3.5.2 Integrated planning of both layers

In the publications discussed in this section, the topology and the capacities of the physical layer are determined together with the logical layer and a routing of demands in the logical network. Compared to a fixed physical layer, this allows more flexibility in the planning process but also increases its complexity because of the additional interdependencies between decisions in the physical and logical layer.

Kubilinskas et al. (2005) [85] Kubilinskas, Pióro, and Nilsson present three mathematical models for two-layer network design with a global cost budget. In case of a physical link failure, the network is reconfigured in the upper layer, in the lower layer, or in both layers simultaneously. Both capacities and flow are assumed to be continuous. In case of a failure, all routing paths can freely be reconfigured even if they are not affected by the failure. The demand values for the logical layer are variables which can be changed within given bounds to model IP traffic with variable bandwidth (called *elastic demands*).

For reconfiguration in the lower layer, the objective is to maximize a revenue function consisting of a weighted sum of the logarithms of the flow for each demand, which can be solved as a linear program. If reconfiguration is performed in the upper layer or in both layers, the objective is to lexicographically maximize a vector of such revenue values. These objective functions are derived from a max-min-fairness argument. The authors address this convex problem by solving a series of linear programs derived from a piecewise linear approximation of the logarithms occurring in the objective function.

The algorithm is tested on two networks: the Polska network from Sndlib [105] with 12 nodes, 18 physical and 22 logical links, and 66 demands, and another network with 41 physical and 21 logical nodes, 72 physical links (21 of which can fail) and 37 logical links, and 209 demands. They compare the three different reconfiguration variants with 2 to 3 admissible physical paths for each logical link (corresponding to 2–3 parallel logical links per node-pair in our model) and 6–14 admissible logical paths for each demand.

Kubilinskas and Pióro (2005) [84] In another publication, Kubilinskas and Pióro extend this formulation by a single-facility capacity model on the logical links and a single-path routing of demands in the logical layer. To protect the logical link capacities against physical link failures, they assume an integer multi-commodity flow with diversification to compute a

representation of logical links in the physical layer. Similar to [85], the objective is to maximize a weighted sum of the logarithms of the demand flows under a global budget constraint. As an alternative, they also present a formulation with explicit working and backup paths to model 1+1 protection.

To solve these problems, the authors propose an iterative search procedure. It repeatedly reroutes the demands using a shortest path algorithm with changing link weights, and solves a sub-MIP to install as much logical and physical capacity as needed for the computed routing. On the networks from [85] and two other networks with 30 and 60 nodes, the authors show that the iterative heuristic performs well compared to a branch-and-cut algorithm.

Kubilinskas (2008) [83] In his PhD thesis, Kubilinskas extends the results from [84, 85] and investigates various issues arising in multi-layer networks with IP/MPLS at the logical layer. This includes several models for designing survivable two-layer networks, some of which are similar to the ones presented in this thesis. In particular, [83] is one of the very few publications on integrated two-layer network design to cover realistic survivability concepts in models which are suitable for computational purposes.

First, Kubilinskas presents several mixed-integer programming formulations for two-layer networks to model different survivability mechanisms under a single physical link failure scenario. The considered mechanisms are unrestricted reconfiguration, path restoration without stub release (see [101, 109]), diversification, and 1+1 protection. They are applied either to the upper layer only or to the lower layer only, assuming that the capacities and routing of the respective other layer are fixed. For the path restoration concept, also a working path routing in the normal network state is assumed to be given.

The author also introduces a MIP formulation for the integrated design of a two-layer IP-over-DWDM network. Path-flow formulations are used both to route elastic demands on a single path in the logical network and to realize logical link capacities in the physical layer. A fractional multi-commodity flow and a single-facility link capacity model are assumed on both layers. To protect logical link capacities against physical link failures, the routing of the logical link capacities in the physical layer has to obey link diversification constraints. The goal is to maximize a revenue function derived from a fairness argument as in [85], subject to a global cost budget imposed on the physical link cost. Node hardware is not considered, and logical links do not incur any cost. As an alternative, Kubilinskas introduces another integrated planning formulation with 1+1 protection at the logical layer. Each demand is routed on a single pair of physically link-disjoint paths. Three different objectives are considered: first, the max-revenue objective from the previous model; second, maximizing throughput (limited only by the global cost budget); third, minimizing total physical link cost.

These four models are solved approximately using an iterative algorithm. It routes demands in the logical layer with respect to some logical link cost values, computes sufficient logical link capacities, routes these in the lower layer, updates the logical link cost according to the resulting link loads, and reiterates until some stopping criterion is met. With slight modifications, this iterative solution method can be applied to all four models. The subproblems of optimizing one of the two layers with a fixed other layer are solved as a sub-MIP using the single-layer models discussed above.

The iterative algorithm is compared to CPLEX as a black-box MIP solver with a time limit of three hours. The tests are done on the POLSKA network and three larger randomly generated networks from [85]. These networks have 12–60 physical nodes, 12–50 logical nodes,

and a complete demand matrix. To reduce the complexity of the problem, the set of admissible routing paths for each demand is limited to 3–6 on the large networks and 4–14 on the small networks. The networks have 18–142 physical and 22–125 logical links, respectively. The logical layer does not connect all pairs of nodes, but each existing logical link has 2–5 possible realizations in the physical layer (3 on the larger networks). The upper and lower layer module capacities are set to 128 and 64 per unit, which corresponds to $B_e = 2$ in our models.

In all considered settings, the value of the linear relaxation is extremely close to the best solution value; the relative gap is always below 0.1%. With the first three objectives, the iterative procedure produces near-optimal solutions (with respect to the predefined set of routing paths), and is faster than the MIP approach in most cases. If physical link cost is minimized with 1+1 protection at the logical layer, the gaps between the linear and integer solutions are also negligible, but the iterative solution performs slightly worse than with the other objectives (up to 8% relative gap compared to the MIP solution). On the three instances with up to 41 nodes, the MIP is solved in at most 91 seconds. On the instance with 60 physical and 50 logical nodes, the iterative approach computes a solution which is only 0.01% above the LP value, and both the LP and the iterative algorithm take around one minute of computation time. The MIP does not produce feasible solutions within 3 hours.

Some of the work in Kubilinskas [83] is very close to parts of this thesis. Even though these models have been developed independently, the cost minimizing formulation in [83] is similar to our model PROTECTED presented in Section 3.2.3. From an application point of view, the main differences are that we consider logical link cost, several logical link facilities, and node hardware with associated cost, and that our routing is protected not only against physical link failures but also against node failures. From a modeling point of view, we use an edge-flow formulation to model a fractional multi-commodity flow routing at the logical layer, whereas Kubilinskas uses a path-flow formulation where each demand is routed on a single pair of paths. Notice that already in the LP relaxation of his path-flow model, the column generation problem is $\mathcal{NP}$-hard because multiple logical link failures have to be taken into account, see Orlowski and Pióro [100]. To solve the model, Kubilinskas thus restricts the set of admissible routing paths to few predefined paths per demand. On the computational side, the size of the networks and the number of parallel logical links of the instances considered in Kubilinskas [83] are similar to our test instances investigated in Section 5.1. Apart from the realistic background of our network instances, the major differences are that most of our logical networks are complete, and that we admit all possible routings at the logical layer. Consequently, we cannot solve our larger instances to near-optimality within a few minutes as in [83], and have to employ sophisticated branch-and-cut-and-price techniques, see Section 4.

Lardeux et al. (2003) [86] Lardeux, Knippel, and Geffard propose an integer programming formulation based on metric inequalities. These inequalities are used to formulate both the capacity requirements for routing of demands in the logical layer and the requirements for routing the logical capacities in the physical layer. One out of several discrete capacities has to be chosen on both layers, and the goal is to minimize total capacity cost. Neither node hardware nor survivability constraints are considered.

The authors investigate several methods to solve the problem. The first one solves the problem as a MIP, where cutset inequalities and more general metric inequalities are generated dynamically. Another method approximates the MIP by a multiple choice multi-dimensional knapsack problem. The latter is solved heuristically by a taboo search procedure and a greedy

algorithm which makes infeasible solutions feasible by increasing capacities step by step.

The computation times of the algorithms are compared on two networks with 6 and 8 nodes, 9–14 physical links, and a complete logical layer and demand matrix. The heuristic approach yields solutions which are within 0–14% of the optimal value, and it takes between 30% and 120% of the time needed for the MIP approach.

Knippel and Lardeux (2007) [76] In a later paper, Knippel and Lardeux generalize the model from [86] to an arbitrary number of layers. They present an integer programming (IP) formulation with metric inequalities where the discrete capacities of one layer define demands for the next lower layer.

For two layers, the model is solved using a Benders decomposition approach where cutset and metric inequalities are generated on both layers. In addition, redundant metric inequalities combine the demands with the physical layer. Between two separation rounds, the IP is always resolved to optimality.

The algorithm is tested on the 6- and 8-node instances from [86] and on one instance with 10 nodes, 18 physical links, and a complete logical layer and demand matrix. For varying cost ratios between logical and physical capacities, the authors compare the cost of iterative top-down planning to the integrated planning approach. Only the small instances with 6 and 8 nodes can be solved to optimality. The authors find that sequential planning is faster by orders of magnitude but can lead to 0–40% more expensive results.

Fortz and Poss (2008) [52] The work of Fortz and Poss is based on the model of Lardeux et al. [86]. The authors compare two different solution approaches. The first one is classical Benders decomposition as employed in [86], where the MIP is always solved to optimality before a new metric inequality is generated. The other algorithm generates cutting planes also for fractional LP solutions during the branch-and-cut procedure; in addition, cutset inequalities are generated, and cutting planes are strengthened using a simple form of mixed-integer rounding. Neither node hardware nor survivability constraints are considered.

Both solution methods are compared on random instances with 8 and 9 nodes, 14 and 16 physical links, and a complete logical layer and demand matrix. The more sophisticated algorithm performs much better than the standard textbook method. Whereas the latter cannot solve all instances to optimality within 1 and 10 hours, respectively, the improved version solves all instances within a few seconds on the 8-node network and within 30 minutes on the 9-node network. On a larger network with 10 nodes, the authors show that the rounding procedure applied to the generated inequalities reduces the computation times.

Orlowski and Wessäly (2004) [102] In this technical report, Orlowski and Wessäly give an introduction to the basic concepts of multi-layer networks from various modeling perspectives and discuss where these concepts occur in different technological contexts. They present a technology-independent mixed-integer programming formulation with detailed node hardware, multi-facility link capacities on all layers, several demand granularities, and an integer multi-commodity flow routing of demands which is survivable against physical link and node failures. In contrast to all previous publications, the model is designed to be applicable in various technologies, and logical links are interpreted as subpaths of end-to-end routing paths. An algorithmic approach is sketched based on Benders decomposition and column generation to deal with all possible logical links, but no computational results are given. In

retrospect, the presented model was not particularly useful for computational purposes, but the report stimulated a lot of discussions in the multi-layer network design community.

Belotti and Malucelli (2005) [18] Similarly to Orlowski and Wessäly [102], Belotti and Malucelli propose a MIP formulation where the variables correspond to paths and subpaths. Assuming single-facility capacities on the logical links and a multi-commodity flow routing, the goal is to minimize the total physical and logical capacity cost. The physical capacities are determined by the logical flow rather than by the logical capacities; the practical motivation for this modeling decision is not clear.

The authors sketch the basic idea of a pricing procedure for logical link capacity variables and end-to-end paths simultaneously. Every new logical link capacity variable may also lead to a new row. The authors do not address the issue of how to determine the dual variables of missing logical link capacity constraints, which cannot be simply assumed to be zero. It is thus not clear whether the obtained dual bounds are valid with respect to all logical links or not. According to private communication with Pietro Belotti, the LP relaxation is solved heuristically with a pricing procedure which defines overlapping parts of used logical links as new logical links. Afterwards, the authors round up the fractional LP solution and apply a rerouting heuristic to improve the rounded solution.

Computational results are given for seven networks with 29–358 nodes, 39–563 physical links, and 64–495 demands (corresponding to demand densities between 0.3% and 12.5%). The computation times for the root relaxation and the subsequent heuristic range from a few minutes to 20 hours, where most of the time is needed to solve the intermediate LPs. The gap between the heuristic LP bound and the cost of the computed integer solution ranges from 2% to 245%, most of the gaps ranging between 30% and 70%.

Belotti et al. (2007) [19] In the conference paper [19] and the extended version [20], Belotti, Capone, Carello, Malucelli, Senaldi, and Totaro extend the model from [18] by statistical multiplexing to take varying demand values into account. Every demand has a mean value and a peak value. The logical link capacities are designed for simultaneous mean values and non-simultaneous peak values; they have to accommodate the sum of all mean demands plus any peak demand. The physical network must be sufficient for all peak demands simultaneously because it cannot be changed as easily as the logical network. A single-facility link capacity model with the same base capacity λ is assumed in both layers. Both logical and physical link capacities are determined by the flow on the logical layer. To groom traffic demands at a node, this node must be equipped with a cross-connect which incurs cost. The goal is to minimize the sum of node cost, physical and logical link cost, where the cost of a capacity unit λ on a logical link is defined as the total cost of installing λ on each of its physical links. In the extended version [20], the model comprises multi-facility capacities on the logical and physical links, and different kinds of cross-connects at the nodes.

The authors solve their model by applying Lagrangean relaxation to all capacity constraints (in particular, all inter-layer constraints). In this way, the integral problem decomposes into a series of shortest-path problems and a linear program. The linear program is solved using column generation for logical link variables to deal with all possible logical links. As the dual variables of missing logical link capacity constraints are computed heuristically, the obtained lower bound is heuristic as well. Feasible solutions are computed by two shortest-path procedures, followed by two variants of a local search improvement algorithm which tries

to reduce capacities by rerouting flow.

The proposed algorithms are tested on various real-world instances provided by Alcatel. On two networks with $(|V|, |E|, |R|) = (25, 63, 275)$ and $(39, 86, 300)$, no node cost, and low, medium, and high peak values, the authors compare the computed solutions to the heuristic lower bound computed by Lagrangean relaxation. All gaps are in the order of 40–50%, with an average computing time of more than 9 hours. According to the authors, most time is spent in the local search algorithm, which improved the initial solution by about 11% on average. In a second test, they evaluate the effect of different ratios of node to link cost on three other network instances with 35–45 nodes, 63–106 physical links, and 340–575 demands. As expected, it turns out that the higher the node cost, the fewer nodes are equipped with a cross-connect, i. e., less grooming is performed in the network. Even with low node cost, not all nodes are equipped with a cross-connect. Although this is not clearly stated in the model, an assumption seems to be that a cross-connect is only needed for intermediate grooming, but not at the source and target node of a demand.

Orlowski et al. (2007) [104] Orlowski, Koster, Raack, and Wessäly introduce some of the heuristics presented in Section 4.3 of this thesis. The model employed in [104] is a relaxed version of the model UNPROTECTED-BASE (3.1) presented in Section 3.2. It comprises multi-facility logical link capacities and single-facility physical link capacities. The goal is to route demands in the logical layer and to install link capacities in both layers at minimum cost.

To support the branch-and-cut algorithm in finding good solutions, the authors propose the INSTALLCAPHEUR, INSTALLCAPMIP, and REROUTEMIP heuristics, which are explained in Section 4.3. These heuristics are called at various places in the search tree.

They are tested on six networks with realistic background from SNDLIB [105]. Two of these instances are also used in this thesis. The size $(|V|, |E|, |R|)$ of the networks ranges from $(12, 18, 66)$ to $(40, 89, 780)$. Between 1 and 50 logical links are admissible between each pair of nodes, which leads to several thousand logical links in the network. It turns out that although the heuristics have little impact on the final gap after three hours, they help to find good solutions earlier in the search tree.

Koster et al. (2008) [81] The model introduced by Koster, Orlowski, Raack, Baier, and Engel is the basis for the models presented in Section 3.2 of this thesis. It is a variant of the planning model PROTECTED (3.4). In contrast to this thesis, the physical link capacity variables z_e are assumed to be binary (i. e., a potential fiber can either be installed or not), and logical link flow variables $f^k_{\ell,ij}$ are used also for unprotected commodities.

The authors extend various cutting planes from single-layer networks to the multi-layer case, and introduce some problem-specific presolving steps. These results are part of the work described in Sections 4.2.1 and 4.4 of this thesis. The cutting planes are tested on two realistic networks provided by Nokia-Siemens Networks: a 17-node German network with 26 physical links, both with and without physical link cost, and a physical ring network with 15 nodes. Depending on the network, the logical layer consists of the complete graph with 2–5 parallel logical links between each pair of nodes. Both networks are also used in this thesis.

On a subgraph of the ring network with seven nodes, the problem-specific presolving can save a huge number of branch-and-cut nodes, and significantly reduces the computation time. On the larger instances, the cutting planes significantly help closing the integrality gap without protection, but contribute only little to the lower bound in the protected case. A sug-

gested reason is that the considered cutting planes do not exploit the multi-layer survivability requirements. This subject is further investigated in Sections 4.4.3 of this thesis.

3.5.3 How this thesis adds to the literature

Summarizing the two previous sections, a variety of mathematical models has been developed during the last decade. Whereas some of them have been used computationally, other models have been introduced solely to define the planning problem and to serve as a basis for discussion. Many formulations are based on a fixed physical layer or on other restrictions (like physical ring networks), either for computational reasons or motivated by practical applications. The practical motivation for certain modeling decisions is not always evident.

The computational results presented in most papers make clear that multi-layer planning problems are extremely hard to solve, and that merely applying a black-box MIP solver is not sufficient to solve networks of realistic size. Quite often, the set of logical links, demands, and admissible routings is heavily restricted for computational reasons. Notice that we do the same at the physical layer by predefining a set of admissible logical links; on the contrary, we do not restrict the set of admissible routing paths at the logical layer.

Models including wavelength assignment constraints could be solved at most on toy instances because of the symmetries induced by the wavelength continuity constraints. Many authors test their algorithms on randomly generated data, probably by lack of realistic data from industry partners. Unfortunately, little such data for multi-layer network planning is publicly available. Without further information, generating realistic instances—in particular demand patterns—is a non-trivial task.

The additional value of this thesis compared to the existing literature can be summarized as follows:

- The models presented in Section 3.2 comprise more practical side constraints than any previous publication on multi-layer network planning (setting aside our own paper [81]). The models cover the most important restrictions and sources of cost in various technologies. At the same time, they can be used to compute good feasible solutions and useful dual bounds for large multi-layer networks with realistic survivability constraints.

- In Section 3.3, we have discussed different modeling alternatives for various parts of a multi-layer network in detail, such as node, physical and logical links, node hardware, and survivability in a multi-layer network. Based on this discussion, which fills a gap in the literature, we have motivated our modeling choices.

- We enhance our branch-and-cut algorithm by a variety of integer programming techniques, such as problem-specific primal heuristics, cutting planes, presolving, and column generation. These techniques enable us to compute network configurations and dual bounds for large-scale networks with more than 50 nodes, a dense logical layer, and a complete demand matrix. With an out-of-the-box MIP solver, this is not yet possible.

- We test our computations on a variety of structurally different networks with realistic cost structures and demands. Some of them have been provided by our industry partner Nokia-Siemens Networks, others are taken from SNDlib. Based on these networks, we investigate the impact of different restrictions of the solution space.

Chapter 4

Solution approach

This chapter describes the algorithmic methods used to solve the mixed-integer programming models described in Section 3.2. The algorithm is based on a branch-and-cut-and-price procedure enhanced by a variety of problem-specific plugins. This includes problem reductions, cutting planes, and primal heuristics called from within the branch-and-cut tree, as well as a decomposition method and column generation to deal with large network instances.

To start with, Section 4.1 gives an overview on the whole algorithmic approach and summarizes the contributions of each of the following sections. Sections 4.2 and 4.3 present our variable reduction techniques and problem-specific primal heuristics. Our network design specific cutting planes are explained in Section 4.4. As an alternative to solving the whole model in one big mixed-integer program, Section 4.5 describes a decomposition approach where the routing requirements are incorporated via so-called metric inequalities. Eventually, Section 4.6 deals with a column generation procedure to solve large network instances.

This chapter focuses on the algorithmic description of the presented approaches and their mathematical background. Their computational impact is studied in Chapter 5. Throughout this chapter, we assume that the reader is familiar with basic linear and integer programming concepts; especially Sections 4.4–4.6 describe advanced LP/IP techniques. More information on the underlying concepts can be found in [97, 124], for instance. An introduction to the concept of branch-and-cut can be found in [107].

4.1 Overview on the algorithm

We solve the mixed-integer programming models presented in Chapter 3 using the branch-and-cut framework SCIP [3] with CPLEX [39] as the underlying LP solver. We use the default branching rules of SCIP. At every node of the branch-and-cut tree, several built-in and user-defined plugins are called:

- primal heuristics to compute feasible integer solutions,
- separators to cut off infeasible LP solutions by new inequalities,
- incumbent callbacks to test integer solutions for feasibility with respect to constraints which are not modeled in the LP.

We have developed various problem-specific plugins of different types. The employed techniques can be summarized as follows.

Problem reductions To eliminate unnecessary variables and to strengthen variable bounds, we first apply a number of presolving routines specific to multi-layer network design. They are explained in Section 4.2. In addition to basic presolving steps for the integral node and link capacity variables, we introduce an LP-based presolving procedure to remove many unnecessary flow variables. The latter procedure is considered in two different variants. Section 4.2 also describes several global problem restrictions which we use to compute feasible solutions for large network instances. Their computational effect is studied in Section 5.3.

Primal heuristics To identify good solutions early in the branch-and-cut tree, we have developed a variety of problem-specific heuristics which are called in various places of the search tree. They are described in detail in Section 4.3. Most of the heuristics use LP information of the branch-and-cut node from where they are called. Whereas some of them are based on combinatorial algorithms, others solve sub-LPs or sub-MIPs to find feasible solutions. In addition, we use the built-in heuristics of SCIP with their default settings.

Cutting planes To strengthen the LP relaxation, we add various types of cutting planes to the mixed-integer programming formulation, as explained in Section 4.4. They are based on strong inequalities known from single-layer network planning. We have adapted these inequalities to the multi-layer context and extended several of them such as to take multi-layer survivability restrictions into account.

Benders decomposition and metric inequalities Instead of writing all variables and constraints into one big MIP, a network design problem can be decomposed into a hardware and capacity part and a routing part. The master MIP, to which the branch-and-cut procedure is applied, contains only hardware and capacity information but no flow variables or constraints.

Every time an integer capacity installation is found in the master MIP, we test whether a feasible routing exists within these capacities. This is done by solving an auxiliary routing LP that contains the flow variables and the corresponding routing constraints. We consider two formulations for this routing LP, either with or without a capacity bottleneck variable. If a feasible routing is found, the integer capacities and the identified routing together form a feasible solution of the whole problem. Otherwise, a so-called *metric inequality* can be generated which, when added to the master MIP, cuts off the infeasible integer capacities. To speed up the solution process, the same technique can be used to separate not only integer solutions but also infeasible fractional solutions from the master LP relaxation.

The branch-and-cut process on the hardware MIP combined with the feasibility tests from the routing LP forms an exact algorithm and will eventually yield an optimal solution to the whole planning problem. As usual, the solution process can be stopped when a certain optimality gap or time limit is reached.

In Section 4.5, we describe the decomposition approach in more detail, discuss different formulations of the routing LP, and show how to generate violated metric inequalities from these formulations. For large network instances, we describe a new technique to reduce the solving times of the routing LP by adding commodities dynamically and strengthening the resulting metric inequalities.

Column generation For large network instances, the routing formulation is too big to be solved in one step. In such cases, we use a column generation procedure, i.e., we start with a small subset of variables and add further variables only when needed. In Section 4.6, we investigate an enumerative and a path-based algorithm to identify missing variables.

Combining all ingredients Combining all these plugins is not trivial. The success of the overall algorithm largely depends on the interplay between its ingredients, and even ensuring correctness of the algorithm requires a lot of care. The most important parts of this interplay will be mentioned in the corresponding sections. Some examples are:

- Variable bounds from presolving have to be taken into account in the column generation procedure and in the construction of metric inequalities.
- In the decomposition approach, the choice of formulation for the routing LP has implications on the computation of metric inequalities and on the column generation procedure. Primal heuristics have to exchange information with the routing LP in order to generate variables or get information about a routing. Cutting planes involving flow variables, like flow-cutset inequalities, cannot be used with the decomposition approach.
- Numerical troubles occur regularly and have to be dealt with.

4.2 Problem reductions

This section presents a number of problem-specific problem reduction techniques to strengthen variable bounds or remove unnecessary variables. First, we describe some basic presolving steps in Section 4.2.1. Already these techniques significantly improve the LP relaxation and speed up the solution process. In Section 4.2.2, we present a new LP-based presolving technique which exploits the topological requirements imposed by the diversification constraints. On those test instances with protected demands only, this technique allows us to remove 9–45% of all flow variables without changing the solution space. The results of these two sections are joint work with Arie Koster and Christian Raack; the basic presolving steps described in Section 4.2.1 have already been published in Koster et al. [81]. Eventually, Section 4.2.3 describes a variety of problem-specific restrictions of the solution space. Their goal is to easier compute feasible solutions for the global problem. These restrictions can either be applied as a sub-MIP heuristic (for smaller instances) or as a global restriction (for large instances). For the latter setting, their computational impact on the dual bound is studied in Section 5.3.

4.2.1 Basic presolving

To strengthen the formulation, we apply some simple improvements to our models.

- Parallel demands with the same end-nodes and the same survivability requirements can be aggregated into one demand.
- For unprotected commodities $k \in K^u$, the LP formulations presented in Section 3.2 allow to route more than the commodity demand value by using cycle flows, but there is no reason to do so. Bounding all flow variables by the demand value does not change the optimal objective value but usually leads to less cycling in LP solutions: $f^k_{\ell,ij}, f^k_{\ell,ji} \leq d^k$.

For protected commodities $k \in K^p$, the survivability requirements forbid to route more than half the demand value through any logical link: $f^k_{\ell,ij}, f^k_{\ell,ji} \leq \frac{1}{2}d^k$. Notice that these bounds are dominated by the diversification constraints (3.4d) and (3.4e). Nevertheless, setting them explicitly may help the MIP solver during its presolving step because the implicit bounds via the diversification constraints and the network structure are hard to detect for a general-purpose MIP solver.

- At every demand end-node, at least the emanating demand must be switched, which means that some node module must be installed. Consequently, the node GUB inequality (3.4h) can be changed into an equality at such nodes: $\sum_{m \in M_i} x^m_i = 1$. This does not affect the optimal MIP value but strengthens the LP relaxation significantly.

- For the same reason, node modules whose switching capacity is smaller than the emanating demand of a node cannot be installed at that node. Consequently, if $C^m_i < d_i$ for some node $i \in V$ and a node module $m \in M_i$, the corresponding variable x^m_i can be removed from the MIP formulation. This often leads to less fractional x^m_i values in the LP solutions and to better LP values, especially when combined with the previous rule.

- The total demand in the network leads to another bound on logical link module variables. It is usually not tight and does not affect the LP solution, but may help the MIP solver in deriving further relations between the variables to strengthen other bounds.

  The amount of flow that can be routed through any logical link $\ell \in L$ is bounded by the total unprotected demand plus half the protected demand in the network (setting aside undesired cycle flow). Consequently, the number of modules of type $m \in M_\ell$ that possibly needs to be installed on ℓ is bounded by

$$y^m_\ell \leq \left\lceil \frac{1}{C^m_\ell} \left(\sum_{r \in R^u} d^r + \sum_{r \in R^p} \frac{d^r}{2} \right) \right\rceil .$$

- Sometimes it is evident that in any optimal solution, a small link module will not be installed more than a given number of times on a link because a larger module provides the same capacity at a lower price. Consider a link $\ell \in L$ and two of its capacity modules $m_1, m_2 \in M_\ell$ such that $C^{m_1}_\ell \leq C^{m_2}_\ell$. If the relation

$$r := \frac{\kappa^{m_2}_\ell}{\kappa^{m_1}_\ell} \leq \frac{C^{m_2}_\ell}{C^{m_1}_\ell}$$

  holds then r modules of type m_1 incur the same cost as one unit of type m_2, but the latter one provides at least the same capacity. Hence, at most r modules of type m_1 will be installed in any optimal solution. Furthermore, even if equality holds in the above relation, one large module is preferable to several smaller ones because every logical link module consumes one physical capacity unit in (3.4j), independently of its capacity. Consequently, the variable bound

$$y^{m_1}_\ell \leq \lceil r \rceil - 1$$

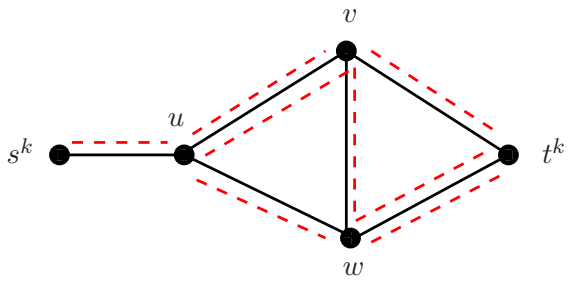
  can be added to the formulation. It cuts off some non-optimal solutions and maybe some optimal ones (if equality holds), but always keeps at least one optimal solution if one exists.

4.2.2 Probing continuous flow variables

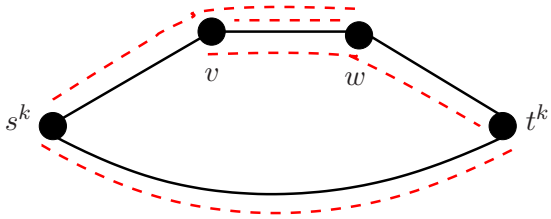
Because of the way in which the diversification constraints are formulated, many flow variables for 1+1-protected commodities can be removed or bounded for topological reasons. The presolving approach described here can be seen as a kind of probing [123], which is a technique employed in state-of-the-art MIP solvers to test whether a variable can have a positive value in any feasible or optimal solution.

Basic idea

The basic idea is to exploit situations like the ones depicted in Figure 4.2.2. Consider a protected commodity $k \in K^p$ of value d^k that has to be routed from s^k to t^k. In Figure 4.1(a), the nodes v and w form a physical 2-vertex cut, that is, the network is disconnected if they are removed. As k is protected, exactly $d^k/2$ must be routed through w from the left to the right. Consequently, there cannot be any flow through w from the right to the left without violating either a flow conservation constraint (3.4c) or the node diversification constraint (3.4d) of w. Consequently, all flow variables $f^k_{\ell,ij}$ for k which contain w as an inner node or end-node and which are directed from the right to the left (like f^k_{wv}) can be removed from the flow formulation because they will never have a positive value in any feasible LP or MIP solution. Similarly, logical links which traverse both v and w cannot carry flow for this commodity in any direction, like the logical link $u \to v \to w \to t^k$.



(a) There cannot be any flow through w from the right to the left because $d^k/2$ must be routed through w from the left to the right.



(b) The flow through the logical link v-w is bounded by $d^k/6$ by a combination of flow conservation and link diversification constraints.

Figure 4.1: Some flow variables for protected commodities may be removed or bounded.

In some cases, it is not possible to fix a flow variable to zero but to derive an upper variable bound smaller than $d^k/2$. In Figure 4.1(b), for instance, the flow on the logical link v-w is bounded by $d^k/6$ by the flow conservation constraints together with the diversification constraint for the physical link v-w.

There are many such situations. Unfortunately, they are hard to detect for the general-purpose presolving routines of SCIP because they are derived from a combination of many constraints of different types. Of course, the variable fixings and bounds described above follow automatically from the constraints in the LP relaxation when it is solved. But by computing and applying them in advance, we can significantly reduce the size of the linear relaxation, and enable SCIP to use these variables bounds in its own presolving.

Probing LP

We have developed an LP-based method to cover the cases described above and others as follows. First, all basic presolving techniques from Section 4.2.1 are applied. Now, for every 1+1-protected point-to-point commodity $k \in K^p$, we construct a linear program consisting of the demand constraints (3.4c), the diversification constraints (3.4d) and (3.4e) for this commodity, and the variable bounds. As this LP only deals with topological constraints where capacities are not an issue, all flow values can be scaled down by d^k, i. e., we route 2 units, 1 of which has to survive any physical node or link failure. To reduce cycle flows in the LP, we also set an upper bound of 1 on all flow variables.

For every commodity $k \in K^p$, every logical link $\ell_0 \in L$, and each of the two directions of flow on ℓ_0, we maximize the flow variable $f^k_{\ell_0,ij}$ subject to the following constraints:

$$\max \; f^k_{\ell_0,ij} \tag{4.1a}$$

$$\sum_{j\in V}\sum_{\ell\in L_{ij}} (f^k_{\ell,ij} - f^k_{\ell,ji}) = \begin{cases} 2 & \text{if } i = s^k \\ -2 & \text{if } i = t^k \\ 0 & \text{otherwise} \end{cases} \qquad \forall\, i \in V \tag{4.1b}$$

$$\frac{1}{2}\sum_{\ell\in\delta_L(i)} (f^k_{\ell,ij} + f^k_{\ell,ji}) + \sum_{\ell\in L^i} (f^k_{\ell,ij} + f^k_{\ell,ji}) \le 1 \qquad \forall\, i \in V \tag{4.1c}$$

$$\sum_{\ell\in L^e} (f^k_{\ell,ij} + f^k_{\ell,ji}) \le 1 \qquad \forall\, e \in E \tag{4.1d}$$

$$0 \le f^k_{\ell,ij}, f^k_{\ell,ji} \le 1 \qquad \forall\, i,j \in V,\ \ell \in L_{ij} \tag{4.1e}$$

The optimal value c of this linear program denotes the maximum flow on logical link ℓ_0 for commodity k which is possible under the above demand and survivability constraints. Consequently, the variable $f^k_{\ell_0,ij}$ in the original formulation PROTECTED can be bounded from above by cd^k. In particular, the variable can be removed if $c = 0$.

Every solution of the probing LP defines a feasible routing. To avoid solving unnecessarily many linear programs, the probing LP to maximize $f^k_{\ell_0,ij}$ can be skipped if that variable has been at its upper bound 1 in any previous solution of the probing LP, because the upper bound can obviously be achieved.

Testing for zero bounds only

On large network instances with 50 or more nodes, solving all these LPs can take several hours. We thus developed a variant of the above algorithm: by imposing a small upper bound, say 0.01, on the maximized variable, the LP solving process is stopped as soon as the objective value is positive. This hope is that this technique accelerates the probing procedure. The drawback is that no upper bounds on the variables are computed. Tests on smaller instances, however, indicate that the impact of the missing bounds on the overall solution process is marginal, see Section 5.2.1. Removing variables with an upper bound of 0 is much more important than setting nonzero bounds. Similarly to the full version, an LP can be skipped if the variable in the objective function has already been positive in some previous routing.

Implementation

In our implementation, we have constructed only one single-commodity LP where the flow variables $f_{\ell,ij}$ are indexed by their logical link and the direction of the flow but not by their commodity. In this LP, we update the right-hand side of the demand constraints and the objective depending on the considered commodity and the tested flow variable. The diversification constraints are the same for all commodities because we normalized all demand values to 2. Of course, bounds derived either from previous LP solves or from other presolving steps (e.g., for hop limits) can be used to strengthen the LP formulation (4.1a).

When switching from one commodity to the next, only the right-hand side of the demand constraints (4.1b) changes. Consequently, the current basis is still valid for the dual problem and the next LP should be solved using the dual simplex algorithm. In contrast, when changing from one flow variable to another one for the same commodity, only two coefficients in the objective function change. In this case, the primal simplex algorithm should be employed.

The LP probing as well as all other problem reductions are implemented as variable filters. Every candidate variable passes a chain of filters that can set upper and lower bounds on the variable. If the upper bound at the end of the chain is 0, the variable is discarded, otherwise it is created with the final bounds. This very flexible approach allows to apply problem reductions either to the global MIP or within primal heuristics. In the first case, the solution space is changed, in the second case it is not.

4.2.3 Restricting the solution space

For large network instances where only a limited number of branch-and-cut nodes can be solved in reasonable time, it may be useful to restrict the solution space in order to compute feasible solutions at all. Such a restriction can be applied after the root node, for instance; in this case, the dual bound of the root node is globally valid. Another possibility, which is used in most publications on multi-layer planning, is to change the problem globally. The consequence is that the dual bound is defined with respect to the restricted solution space.

The interesting question, of course, is how to narrow down the solution space. The restriction should make the problem easier to solve without affecting the solution quality too much. In this section, we discuss several possible restrictions of the solution space:

- physical or logical hop limits, and
- different variants of a core network heuristic.

Hop limits are a common way to deal with a huge number of routing paths; sometimes they are also motivated by technological reasons. The core network approach defines a hierarchy on the set of nodes and restricts the routing according to this hierarchy. Partitioning a network into a core network and an access network is a common approach employed by network operators to simplify the management of their network. Although the partitioning problem as such has been intensively studied in the literature [28, 29, 77], using such an approach has not yet been employed in the literature to solve multi-layer network design problems. We discuss various ways of defining a core subnetwork and restricting the routing accordingly.

It can easily be seen that there are no quality guarantees for any of these restrictions; nevertheless, some of them have allowed us to compute the best solutions known so far

for our large network design instances. Their effect on the dual bound is investigated in a computational study in Section 5.3.

Physical and logical hop limits

In the network design literature, hop limits are a common way of restricting the set of admissible routing paths [45, 46, 66, 67, 73, 139]. Such a restriction may have technological reasons because long routing paths tend to cause large transmission delays and extra cost, e. g., for signal repeaters. Another possible reason to apply hop limits is to make the problem easier to solve and to avoid implementing a column generation procedure. In any case, it is interesting to know how much hop limits affect the total network cost. This knowledge allows network planners to choose suitable hop limits that do not restrict the solution space too much, and it allows network operators to balance, for example, the trade-off between increased network cost and reduced delay. For single-layer networks, Orlowski and Wessäly [103] have studied the effect of hop limits on the network cost under realistic assumptions; for multi-layer networks, the literature does not provide any such study yet.

We investigate both logical and physical hop limits. By imposing a physical hop limit of k, we admit only logical links consisting of at most k physical links. The case $k = 1$ models an *opaque* network where all flow is groomed at every intermediate node of a routing path.

At the logical layer, we consider a hop limit of 1 or 2, which can easily be implemented in an edge-flow formulation by restricting the set of flow variables. With a hop limit of 1, every demand must be routed on a direct logical link between its end-nodes. In optical network terminology, this corresponds to a *transparent* network. With a hop limit of 2, one intermediate node can be used.

Core network heuristics

Core and access networks To simplify the management of their network, many network operators impose a hierarchy on the network nodes that reflects their relative importance. For example, a core network can be defined between cities which are an important source of traffic or which are located at geographically strategic points. The core nodes are connected via "highway" links with large bandwidth. Small demands between distant and peripheral nodes must first be routed to the core network on a short path in a regional access network, travel most of the distance via core nodes, and leave the core network to the destination on a short path again. Intuitively, the core nodes are points of aggregation points in the network where traffic is collected, reorganized, and forwarded to another node. In other words, traffic is groomed at these nodes.

Predefining a core network can also be used to speed up the computation process. From a practical point of view, the resulting solutions tend to look "nice" in the sense that most logical links are defined between geographically close nodes. Even though forcing such a topology may increase the network cost, it corresponds to the intuition that "short links are better". The goal is thus to predefine a suitable core network with corresponding routing restrictions in such a way that the solution space contains "nice" solutions which are not too expensive. We introduce different strategies to do so. Their computational impact on the network cost is studied in Section 5.3.

Restricting the routing Given a subset of core nodes, we define several routing restrictions which can be combined. Some of them are demand-dependent, others are not.

1. A demand defined between two core nodes must only be groomed at other core nodes.
2. Traffic starting from a non-core node must be routed on at most k physical links either to its destination node if this is possible, or to the core network (and similarly for the other end of the routing path).
3. Optionally, traffic must be routed on a direct link either to the destination node or to the core network, i.e., we disallow grooming on access paths to the core network.
4. Optionally, traffic must always be groomed at core nodes, i.e., we forbid transit traffic on logical links passing through a core node without being terminated there.

Under the assumption of point-to-point commodities (which is satisfied for protected commodities), all these restrictions can be implemented by variable filters that forbid certain flow variables in a commodity-dependent way. Of course, these filters must also be taken into account in the column generation procedure described in Section 4.6.

To realize the second restriction algorithmically for a given point-to-point commodity $k \in K^p$, we compute all those paths that start in a demand end-node and reach either the opposite demand end-node or the core network within k physical hops. This is done using a breadth-first search procedure. Afterwards, we generate only those flow variables $f^k_{\ell,ij}$ that correspond to logical links which are part of any the computed candidate access paths.

Defining a core network Partitioning a network into a core network and an access network such that certain routing and capacity constraints are satisfied is itself a hard problem [28, 29, 77]. From a practical point of view, such a partitioning only makes sense for large and dense networks. On the US 67-node network us67 (see Section 5.1 for details), the logical network is sparse because of the large geographical distances. Consequently, the problem quickly gets infeasible if grooming is seriously restricted, and we cannot apply our core heuristics. For the German 50-node instance g50, we pre-define a core network according to several criteria:

- The first idea is to define the 17-node subnetwork g17 (see Section 5.1) as the core network. These nodes are either "geographically central" or have large emanating demand. It is likely that this subnetwork has been defined in the European NOBEL project [2] exactly by these criteria. In fact, this definition turned out to be the best choice with respect to the quality of the computed solutions.
- To generalize this idea, we assign a score to each node according to different criteria, and define the core network to consist of the roughly $p\%$ of nodes with highest score, for some suitable value of p.

  With the *region demand* strategy, the score of a node $v \in V$ is computed as its emanating demand plus the emanating demand of its physically adjacent nodes. The idea of this strategy is to favor core nodes which either have a large emanating demand themselves or which are surrounded by several nodes with large emanating demand.
- With the *connectivity* strategy, the score of a node v is defined by the importance of v for the connectivity of the network: define $\mathrm{dist}(i,j)$ as the shortest hop distance between

i and j in the physical network, and let

$$\sum_{i,j \in V} \text{dist}(i,j) \cdot \sum_{r \in R_{ij} \cup R_{ji}} d^r$$

be the total hop distance in the network, weighted with the demand values. Compute this value with and without node v in the network; the score of v is the difference between these two values. The idea of this score is that in a real-world network, removing geographically central nodes often leads to much longer routing paths for many demands, whereas peripheral nodes barely affect the shortest-path lengths.

These strategies are studied computationally in Section 5.3.2.

4.3 Primal heuristics

To compute feasible solutions, we have developed a variety of primal heuristics, which are called at various places during the branch-and-cut algorithm. Some of them are combinatorial, others solve a restricted version of the original problem as a mixed-integer program. The restrictions are either based on topological properties of the network in combination with the demand structure or on previously computed feasible solutions.

Section 4.3.1 describes two heuristics aiming at the capacity installation subproblem for a given routing. In Section 4.3.2, we explain a combinatorial heuristic to compute an initial routing under multi-layer survivability constraints. The following three sections present heuristics that solve a restricted version of the original problem as a sub-MIP or sub-LP. Section 4.3.3 presents a crossover heuristic which restricts the potential topology of the problem based on already known feasible solutions. Section 4.3.4 describes a MIP-based heuristic that tries to reduce capacities by rerouting flow based on an LP solution. The latter heuristic and the ones discussed in Section 4.3.1 have been previously published in Orlowski et al. [104]. The heuristics discussed in 4.3.5 reroute flow based on a given feasible network configuration, either combinatorially or by solving a sub-MIP or a series of sub-LPs. Eventually, we conclude with a note on the literature and implementational issues of sub-MIP heuristics in Section 4.3.6.

For ease of exposition, we explain all heuristics using the notation from the PROTECTED model (3.4) with logical link flow variables $f^k_{\ell,ij}$. Extending the presented ideas to node-pair flow variables f^k_{ij} for unprotected commodities is straightforward.

4.3.1 Computing capacity and hardware for a given routing

The INSTALLCAPHEUR and INSTALLCAPMIP procedures described below address the *capacity assignment subproblem*. It consists of installing capacity and node hardware such that a given routing can be accommodated. For every logical link $\ell \in L_{ij}$, let

$$f^*_\ell := \sum_{k \in K} \sum_{\ell \in L} (f^k_{\ell,ij} + f^k_{\ell,ji})$$

be the total flow on ℓ in an integer or LP solution (after removing possible cycle flows). Then the capacity assignment subproblem is defined by the following sub-MIP of the original

formulation (3.4) (on page 41), which contains logical and physical capacity variables but no routing information:

$$\min \left\{ \text{(3.4a) subject to (3.4h)–(3.4j)}, \sum_{m \in M_\ell} C_\ell^m y_\ell^m \geq \lceil f_\ell^* \rceil \quad \forall \ell \in L, \quad z_e, y_\ell^m \in \mathbb{Z}_+ \right\}. \tag{4.2}$$

More generally, f_ℓ^* need not necessarily be defined by a routing; indeed, it can be any lower bound on the total capacity of logical link $\ell \in L$.

InstallCapMip

The InstallCapMip procedure addresses subproblem (4.2) by solving it as a sub-MIP using Scip's branch-and-cut capabilities. It can be used either as a construction heuristic based on LP solutions or as an improvement heuristic. With a limit on the number of branch-and-cut nodes, we employ it as an improvement heuristic every time a new best solution is identified, trying to reduce capacity and hardware cost based on the given routing. The InstallCapMip heuristic often improves solutions computed by Scip's built-in heuristics, where many logical links needlessly carry much more capacity than flow.

InstallCapHeur

In contrast to InstallCapMip, the fast and simple InstallCapHeur procedure addresses problem (4.2) heuristically by decomposition, as outlined in Algorithm 1.

Algorithm 1 InstallCapHeur

1: For all $\ell \in L$, install minimum cost capacities on ℓ by solving the following knapsack problem:

$$\min \left\{ \sum_{m \in M_\ell} \kappa_\ell^m y_\ell^m \text{ subject to } \sum_{m \in M_\ell} C_\ell^m y_\ell^m \geq \lceil f_\ell^* \rceil, \quad y_\ell^m \in \mathbb{Z}_+ \right\}. \tag{4.3}$$

2: Install enough capacity on every physical link $e \in E$ by setting

$$z_e := \left\lceil \frac{1}{B_e} \sum_{\ell \in L^e} \sum_{m \in M_\ell} y_\ell^{m*} \right\rceil.$$

3: For each node $i \in V$, select the cheapest node module $m \in M_i$ which is sufficient for the available link capacities according to (3.4i).

If $|M_\ell| = 1$, the knapsack problem (4.3) is trivial to solve. Otherwise, it is solved heuristically for each logical link $\ell \in L$ using a greedy algorithm. The greedy algorithm first tries to cover a capacity of $f_\ell{}^*$ with the largest module only, then substitutes one of these modules with a suitable number of the second-largest type if this configuration is cheaper, and so on. This approach prefers few large modules over many small ones, which is also favorable for the goal of minimizing the required physical link capacities. Under the usually satisfied assumption that the cost per capacity unit goes down as the capacity increases, the greedy algorithm solves the knapsack subproblem (4.3) optimally in linear time with respect to $|M_\ell|$.

The InstallCapHeur heuristic can be used either as a construction heuristic based on LP solutions, or as an improvement heuristic to reduce capacity and hardware cost of a given solution. As it runs very fast, we call it as a construction heuristic at every branch-and-cut node to construct feasible solutions from the current LP solution. To obtain an initial feasible solution from the beginning, we also call the heuristic before starting the cutting plane process at the root node, based on a combinatorially constructed routing. On our large network instances, InstallCapHeur is the only heuristic finding feasible solutions at all, as described in the computational results in Section 5.2.2.

4.3.2 Computing a feasible routing

At various places in our algorithm, a full or partial routing has to be computed, either with or without capacity constraints. For example, capacity constraints occur in rerouting heuristics when flow is rerouted within given capacities of a solution; on the other hand, capacities do not matter if the goal is to compute a feasible set of routing paths to initialize the routing LP in a column generation approach. For capacity values c_ℓ and cost values κ_ℓ on the logical links, the routing subproblem considered in this section consists of finding a feasible routing with minimum logical link cost for each commodity. Feasibility is defined with respect to the routing constraints from the MIP model Protected (3.4). More precisely, the flow has to satisfy flow-conservation constraints, diversification constraints if the commodity is protected, and the flow on logical link $\ell \in L$ must not exceed c_ℓ. We will now describe our algorithms used to compute such a routing. First, consider the unprotected case.

Without capacity constraints, the problem decomposes into a separate problem for each commodity. For unprotected commodities, only the flow-conservation constraints have to be satisfied, and the physical layer does not matter. Hence, the routing problem can be solved by computing a shortest-path tree at the logical layer with respect to κ_ℓ for each commodity $k \in K^u$, rooted in the source node s^k of the commodity. With capacity constraints, the problem does not decompose and has to solved as a multi-commodity min-cost flow problem. So far, no combinatorial algorithm is known for that problem. As an alternative to using a linear programming formulation, we use a simple approach which works quite well with our dense logical graphs: we search for shortest paths for commodity $k \in K^u$ only on logical links with $c_\ell \geq d^k$.

For protected commodities, the routing has to obey additional survivability restrictions imposed by the physical layer. In this case, the routing problem is to compute a survivable minimum-cost routing. Without capacity constraints, this problem can be modeled and solved either as a single-commodity linear program for each commodity, as in model (3.9) on page 52, or as a multi-commodity flow LP with the routing constraints from model Protected. If the commodities have to share common link capacities, a multi-commodity flow formulation with additional capacity constraints has to be used. Unfortunately, the LP-based approach can be time-consuming. In some contexts, the routing subproblem has to be solved so often that a faster algorithm is needed.

We thus implemented a simple iterative procedure to search for two node- and physical link-disjoint routing paths at the logical layer. As outlined in Algorithm 2, it consists of finding a shortest path, blocking all logical links which fail together with that path, and finding another shortest path. This is done for every protected commodity. With capacity constraints, we hide all logical links from the graph whose capacity is smaller than $d^k/2$, which is the maximum flow which can be routed through any node or physical link, and thus

through any logical link as well.

Algorithm 2 FindFeasibleRouting

Input: commodities K, logical link cost κ_ℓ, logical link capacities c_ℓ.
Output: a routing for all commodities satisfying the demand and diversification constraints of model (3.4), and capacity constraints with respect to c_ℓ.
for all $k \in K^u$ **do**
    Hide all logical links $\ell \in L$ with $c_\ell < d^k/2$ from the graph.
    Compute a shortest-path routing in the logical layer with respect to the κ_ℓ.
end for
for all $k \in K^p$ **do**
    Hide all logical links $\ell \in L$ with $c_\ell < d^k/2$ from the graph.
    Find a shortest logical path with respect to the given logical link cost.
    Hide all logical links from the graph failing together with any link from the first path.
    Find another shortest path in the remaining logical network.
    If this approach fails, call Algorithm 3.
end for

In many cases, this iterative approach succeeds in finding two failure-disjoint paths. If the algorithm fails for some protected commodity $k \in K^p$, we use the more sophisticated but also computationally more expensive algorithm CoverPhysicalCycles, which is sketched in Algorithm 3. It computes two disjoint paths from s^k to t^k in the physical layer and covers them with cheap logical links.

Algorithm 3 CoverPhysicalCycles

Input: a commodity $k \in K^p$, logical link cost κ_ℓ, logical link capacities c_ℓ.
Output: a routing for commodity k satisfying the routing constraints of model (3.4).
Find a shortest cycle through s^k and t^k in the physical graph with respect to some physical link cost. This yields two physical paths p_1, p_2 from s^k to t^k.
Cover p_1, p_2 with few logical links using an auxiliary graph H (see Figure 4.2):
for all $p \in \{p_1, p_2\}$ **do**
    Define a node in H for every node of p.
    for all $\ell \in L$ with $c_\ell \geq d^k/2$ **do**
        If ℓ is part of p, define an undirected edge in H with cost κ_ℓ.
    end for
    Search for a shortest s^k-t^k-path in H.
end for

First, we search for a shortest cycle in the physical network through the end-nodes of the demand using Suurballe's algorithm [130], which corresponds to a minimum-cost flow problem between s^k and t^k with value 2 and node and link capacities 1. The physical link cost can be set in various ways depending on the context, for instance to 1 or to the physical link installation cost κ_e from the original planning model Protected. The computed cycle provides us with two disjoint physical s^k-t^k-paths p_1, p_2 (if no cycle exists then the overall planning problem is infeasible). We then cover these two paths with cheap logical links using a shortest-path algorithm in an auxiliary graph H. This graph, which is depicted in Figure 4.2, basically corresponds to the logical network restricted to one of the paths p_i; its construction

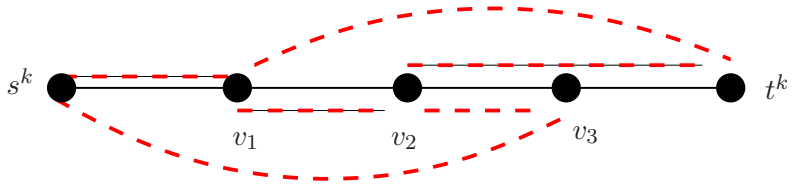


Figure 4.2: A physical s^k-t^k-path p (black, solid) and the auxiliary undirected graph H consisting of the nodes of p and six logical links (red, dashed) whose physical representation is part of p. There are four logical s^k-t^k-paths without logical link repetitions: first, $s^k \to v_3 \to v_2 \to t^k$; second, $s^k \to v_3 \to v_2 \to v_1 \to t^k$; third, $s^k \to v_1 \to t^k$; fourth, $s^k \to v_1 \to v_2 \to t^k$. If all cost values κ_ℓ are 1, a shortest-path algorithm finds the third path, which uses the least number of logical links.

is described in Algorithm 3.

We use this heuristic in various parts of the algorithm. First, in connection with column generation, it is used to compute an initial set of flow variables in the routing LP, see Section 4.6.5. Second, it is combined with the hardware construction heuristic described in Section 4.3.1 to compute an initial feasible solution. Third, with metric inequalities and dynamic addition of commodities to the routing LP (see Section 4.5.4), we use it to compute a routing for commodities that are not yet part of the LP.

4.3.3 Topology-based crossover heuristic

The TopologyCrossover heuristic takes the best k feasible solutions known so far, restricts the problem to those logical links which are used in any of these solutions, and solves the resulting problem as a sub-MIP. The idea is that logical links which are not used in several good solutions are probably not useful to obtain a cost-efficient network configuration.

This heuristic can be seen as a variant of the general-purpose crossover heuristic implemented in Scip [3], which fixes variables to c that have the same value c in several solutions. In our setting, the fixing is applied to those logical link capacity variables which have value 0 in all k considered solutions.

As with all crossover-type heuristics, there is a danger of inbreeding if offspring from a crossover heuristic serves as input to the same heuristic in future iterations. A solution generated by TopologyCrossover does not add any new information when used as input for the same heuristic, because it uses only logical links from any of its parent solutions. To overcome this problem, this heuristic should be combined with other heuristics that generate solutions based on other criteria.

4.3.4 Rerouting flow within given capacities using a MIP

The RerouteMip heuristic determines a routing together with a minimum-cost capacity installation, subject to an upper capacity bound on the logical links. More precisely, given an upper bound U^*_ℓ on the capacity of each logical link $\ell \in L$, RerouteMip solves the following restriction of the original problem as a sub-MIP:

$$\min \left\{ \text{(3.4a) subject to (3.4b)–(3.4k)}, \quad \sum_{m \in M_\ell} C^m_\ell y^m_\ell \le U^*_\ell \quad \forall \ell \in L \right\}. \tag{4.4}$$

For sufficiently large U_ℓ^*, this corresponds to the original problem. With small U_ℓ^*, however, this subproblem is much easier to solve. We employ RerouteMip as a construction algorithm based on LP solutions. Given some value $\kappa \geq 1$ and an LP solution with total logical link capacities $y_\ell^* := \sum_{m \in M_\ell} C_\ell^m y_\ell^{m*}$, we solve the sub-MIP (4.4) with

$$U_\ell^* := C^0 \left\lceil \frac{\kappa}{C^0} y_\ell{}^* \right\rceil,$$

where C^0 is the smallest module capacity installable on ℓ. In practical applications, the installable capacities often form a divisibility chain, that is, if the capacity modules $m_1, m_2, \ldots$ of a logical link $\ell \in L$ are sorted by increasing capacity, these capacities divide each other: $C_\ell^{m_1} | C_\ell^{m_2} | \ldots$. Under this assumption, which is fulfilled in all of our test instances, U_ℓ^* is the smallest installable integer capacity greater than or equal to κy_ℓ^*. Otherwise, this value can be obtained by solving a knapsack problem on the capacities of logical link $\ell \in L$. Obviously, a higher value of κ augments the solution space of the subproblem, allowing for better solutions but also making it harder to solve. Experimenting with different values, we found that $\kappa = 2$ often allows to quickly determine good solutions in the sub-MIP.

As the RerouteMip algorithm is rather time-consuming, we restrict its application to the LP solution at the end of the branch-and-cut root node (with a limit on the number of nodes solved in the sub-MIP). We apply the both the InstallCapHeur and InstallCapMip heuristics within the sub-MIP. Furthermore, we employ RerouteMip as an improvement heuristic, as described in the following subsection.

4.3.5 Solution-based capacity reduction heuristics

To reduce capacities in a given solution, we have developed a variety of rerouting heuristics, jointly with Urs Kramer and Christian Raack.

First, we employ the RerouteMip heuristic described in the previous section as an improvement heuristic. Given an integer solution, we set U_ℓ^* to the installed total capacity of logical link $\ell \in L$. That is, we solve a sub-MIP where all demands can be freely rerouted within the current capacities such as to minimize total installation cost.

Second, as solving the sub-MIP is rather time-consuming, we have also developed a faster alternative based on an iterative rerouting procedure. It reduces the capacity of a link and tries to reroute only the flow affected by this reduction. If this succeeds, a better solution has been found. After all links have been considered, it recomputes a cost-effective capacity and hardware installation using the InstallCapHeur heuristic from Section 4.3.1, and re-iterates. The whole procedure is described in Algorithm 4.

Several steps in Algorithm 4 have to be defined in more detail:

1. In step 5, the logical links can be sorted in different ways. We order them by increasing capacity $\sum_{m \in M_\ell} C_\ell^m y_\ell^m$ in the solution s. In case of ties, we order them by increasing flow as a second sorting criterion. The reason is that in many solutions, there are links with only one small capacity module and only one or two demands routed through them. Intuitively, such links have a higher chance to be eliminated from the solution than links with high capacity and much flow. Of course, other sorting criteria are possible.

2. There are several possibilities to reduce the capacity of a link $\ell \in L$ in step 6. We tried two different approaches: removing one unit of the smallest installed module $m \in M_\ell$,

Algorithm 4 IterativeCapReduction

```
1: Input: A solution s of model Protected (3.4).
2: Output: Solution s with lower or equal cost (3.4a).
3: repeat
4:     improved ← false
5:     for all ℓ ∈ L do
6:         Reduce the capacity of ℓ in solution s.
7:         if all demands can be rerouted within the reduced capacities then
8:             Replace the routing of s by the new one.
9:         else
10:            Undo the capacity reduction.
11:        end if
12:    end for
13:    if capacity could be reduced on any link then
14:        improved ← true
15:        Recompute capacity and hardware using the InstallCapHeur heuristic.
16:        if InstallCapHeur could reduce the cost then
17:            Replace the capacity and hardware configuration.
18:        end if
19:    end if
20: until not improved
```

or setting the capacity of ℓ to zero. The latter version turned out to be more effective in our tests.

3. In step 7, the demands can be rerouted in various ways, with different degrees of freedom:

   (a) The potentially most cost-effective, but also most time-consuming way is to solve a routing LP for all demands which contains demand, diversification, and capacity constraints (with the fixed reduced capacity). This is exactly the routing LP (4.30) used to separate metric inequalities, see Section 4.5.2 on page 103.

   (b) In a slightly more restricted version, the routing LP is restricted to those demands whose routing is affected by the capacity reduction.

   (c) A further variant is to sort the demands in some way (e. g., by demand value, or by the amount of flow for each demand routed through the reduced logical link ℓ), and to decompose the multi-commodity flow LP into a series of single-commodity flow LPs. This variant may or may not be faster depending on the input data; of course, the success depends on the ordering of the demands.

   (d) Eventually, the demands can be rerouted iteratively within the reduced capacities using a combinatorial algorithm. For unprotected demands, this is simply a minimum-cost flow problem. For protected demands, we apply the routing heuristic described in Section 4.3.2. This is similar to the rerouting approach presented in Höller and Voß [68].

If the capacity on some link could be reduced, we nevertheless recompute a cost-effective capacity and hardware installation using the InstallCapHeur heuristic. This may even

further reduce the cost. The reason is that although we reduce the capacity of only one link in step 6, rerouting demands may decrease the flow on several other links as well. Furthermore, it may pay off to use a different combination of capacity modules on the logical links than currently installed. The algorithm is iterated until no more capacity can be reduced, which is usually the case after at most two or three iterations.

4.3.6 A note on sub-MIP heuristics

The advantage of the combinatorial heuristics presented in this section is that they are fast. Their main drawback is that their success may heavily depend on the ordering of demands, logical links, etc. Moreover, adapting them to slightly changing problem constraints, like bounds on certain variables, can be cumbersome. In contrast, sub-MIP heuristics can easily be adapted to other planning requirements because they are based on the same model as the overall branch-and-cut algorithm.

Solving sub-MIPs within branch-and-cut heuristics has become possible only in recent since general-purpose MIP solvers have become powerful enough. In the context of general mixed-integer programming, a variety of MIP-based heuristics has been developed during the last few years; examples are Rins, Rens, Crossover, or the feasibility pump [3, 4]. In the context of single- or multi-layer network design, this approach has been used only in very few publications yet, see Kubilinskas [83], Kubilinskas and Pióro [84], Orlowski et al. [104].

MIP-based heuristics within a branch-and-cut algorithm can be seen as *large-scale neighborhood search (LNS)* [42], which is a special case of *local search* [58, 132]. Local search heuristics define a neighborhood of a given solution and explore it for better alternatives. LNS heuristics define this neighborhood by the set of all feasible solutions of a sub-MIP. Of course, defining a good neighborhood is crucial for the success of such an approach. It should be large enough to contain good solutions, but small enough such to be explored in reasonable time. Our advantage over Scip's built-in general-purpose heuristics is that we can use problem-specific information, like topological properties of the network, to define the neighborhood of a solution.

All sub-MIP heuristics considered in this section focus on finding good feasible solutions, rather than on proving a lower bound. Hence, we disable expensive cut generation (like flow-cutset inequalities) and time-consuming heuristics (such as other sub-MIP heuristics) in the subproblem, and we impose node limits and stall node limits to the subproblems. That is, the sub-MIP is stopped if either a total of N branch-and-cut nodes has been computed or if the primal bound could not be improved during the last S nodes. Depending on the specific sub-MIP, typical values of N in our setting are between 1000 and 20000, with $S \approx N/2$. Furthermore, we observed that in many cases, the sub-MIPs found an improving solution within the node limit if and only if an improving solution was found within the first 50 branch-and-cut nodes. We thus stop the sub-MIP process after 50 nodes if no solution has been found until then. To increase the chance of finding good solutions, we also apply the combinatorial heuristics from Sections 4.3.1 and 4.3.5 within the sub-MIPs, which tends to improve the overall solution quality.

4.4 Cutting planes

To strengthen the LP relaxations solved during the branch-and-cut process, cutting planes can be added to the LP. These inequalities are satisfied for all feasible integer points but

violated by the current fractional LP solution. Given a point $\hat{p}$, the *separation problem* is to find an inequality (of a certain class) that cuts off $\hat{p}$ from the polyhedron of all feasible integer points, or to determine that no such inequality exists.

For single-layer network design, a variety of cutting planes has been developed during the last 15 years. Many of them are known to define facets of suitable relaxations of the studied network design polyhedra under appropriate conditions, and have also been successfully used in computations. Most of these cutting planes have been derived *mixed-integer rounding* (MIR). The idea is the following. First, model inequalities are aggregated to obtain a valid *base inequality*. Afterwards, a strengthening procedure is applied to this base inequality, which exploits the integrality of some of the involved variables. More precisely, applying a suitable *MIR function* to the coefficients and the right-hand side of the base inequality yields a (usually stronger) *MIR inequality* which serves as a cutting plane to cut off infeasible fractional points.

This idea, which goes back to Gomory [59], has been applied in various ways to network design problems to exploit the integrality of capacity variables. Many types of cutting planes have been derived in this way, for instance cutset, partition, arc-residual capacity, [15, 89, 89, 91, 115], flow-cutset [9, 34], and flow-cover inequalities [9, 88]. They differ in the choice of aggregated base inequalities and in the definition of the mixed-integer rounding function applied to the coefficients and right-hand side of the base inequality. We also use this concept by defining appropriate aggregations and MIR functions.

In this thesis, we have adapted and extended several strong cutting planes known from single-layer network planning to the multi-layer context. This section describes these inequalities and how to separate them. Section 4.4.1 deals with cutset and flow-cutset inequalities at the logical layer, whereas Section 4.4.2 covers connectivity cuts at the physical layer. These results have been published before in Koster et al. [81]. In Section 4.4.3, we extend cutset and flow-cutset inequalities to take multi-layer survivability restrictions into account. Major parts of Sections 4.4.1–4.4.3 are joint work with Christian Raack and Arie Koster. Section 4.4.4 generalizes the single-layer Q-subset inequalities of Magnanti and Wang [90] and Bienstock and Muratore [26] to two layers. These inequalities also incorporate the protection requirements. These results are joint work with Josefine Müller and Christian Raack, published in the diploma thesis [96]. Eventually, we discuss in Section 4.4.5 how to postprocess identified cutting planes to improve their numerical behavior.

For ease of exposition, we assume throughout this section that $K = K^p$, such that the model contains only logical link flow variables $f^k_{\ell,ij}$ but no aggregated node-pair flow variables f^k_{ij}. Everything can be done similarly with unprotected commodities and aggregated flow variables (as in our implementation).

4.4.1 Logical layer cutset and flow-cutset inequalities

On the logical layer, we consider *cutset inequalities* and *flow-cutset inequalities*. These cutting planes have, for instance, been studied in [9, 25, 34, 89, 116] for a variety of network settings (directed, undirected, and bidirected link models, single or multiple capacity modules, etc.) and have been successfully used within branch-and-cut algorithms for capacitated single-layer network design problems [25, 27, 64, 115].

The inequalities on the logical layer are valid for the polyhedron P defined by the multi-commodity flow demand constraints (3.4c) and capacity constraints (3.4f). That is,

$$P := \operatorname{conv}\left\{(f,y) \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{n_2}_+ \mid (f,y) \text{ satisfies } (3.4c), (3.4f)\right\},$$

where $n_1 := 2|K||L|$ and $n_2 := \sum_{\ell \in L} |M_\ell|$. As P is a relaxation of the model discussed in Section 3.2.1, the cutset and flow-cutset inequalities are also valid for that model. Adding survivability or other constraints to the model may affect the strength of the cutting planes but not their validity.

We introduce the following notation. For any subset $\emptyset \neq S \subset V$ of the nodes V, let

$$L_S := \{\ell \in L \mid \ell \in L_{ij},\ i \in S,\ j \in V \setminus S\}$$

be the set of logical links having exactly one end-node in S, as shown in Figure 4.3.

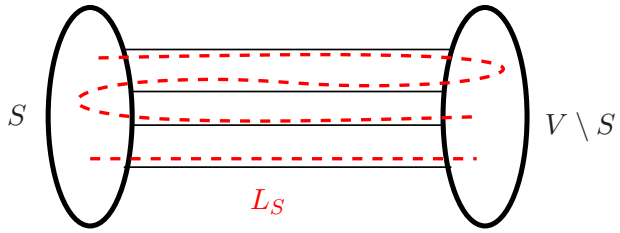


Figure 4.3: A cut in the network defined by a subset of nodes $S \subset V$, with the set of traversing physical links (black, solid) and the set L_S consisting of two logical cut links (red, dashed). Logical cut links $\ell \in L_S$ may physically traverse the cut any odd number of times.

Define $d_S^k := \sum_{i \in S} d_i^k \geq 0$ to be the total demand value to be routed over the cutset L_S for commodity $k \in K$. By reversing the direction of demands and exchanging the corresponding flow variables, we may assume without loss of generality that $d_S^k \geq 0$ for all $k \in K$ (i. e., either the commodity is directed from S to $V \setminus S$, or the end-nodes of k are all in S or all in $V \setminus S$). More generally, let $d_S^Q := \sum_{k \in Q} d_S^k \geq 0$ denote the total demand value to be routed over the cut L_S for all commodities $k \in Q$.

Mixed-integer rounding (MIR)

In this section, we summarize important properties of mixed-integer rounding which are needed to construct our inequalities. Further details can be found in [92, 113], for instance. In the following sections, we will only describe how to construct the base inequality, and briefly define the MIR function without going into the details.

Let $a, c, d \in \mathbb{R}$ with $c > 0$ and $\frac{d}{c} \notin \mathbb{Z}$ and define $a^+ := \max(0, a)$. Furthermore, let

$$r(a, c) := a - c(\lceil \tfrac{a}{c} \rceil - 1) > 0$$

be the remainder of the division of a by c if $\frac{a}{c} \notin \mathbb{Z}$, and c otherwise. A function $f : \mathbb{R} \to \mathbb{R}$ is called *subadditive* if $f(a) + f(b) \geq f(a + b)$ for all $a, b \in \mathbb{R}$. The MIR function

$$F_{d,c} :\ \mathbb{R} \to \mathbb{R} :\ a \mapsto \lceil \tfrac{a}{c} \rceil\, r(d, c) - (r(d, c) - r(a, c))^+$$

is subadditive and nondecreasing with $F_{d,c}(0) = 0$. If $\frac{d}{c} \notin \mathbb{Z}$ then

$$\bar{F}_{d,c}(a) := \lim_{t \searrow 0} \frac{F_{d,c}(at)}{t} = a^+$$

for all $a \in \mathbb{R}$; otherwise $\bar{F}_{d,c}(a) = a$ for all $a \in \mathbb{R}$, see [115]. Because of these properties, if the inequality

$$\sum_i a_i x_i + \sum_j b_j f_j \geq z$$

is valid for some polyhedron Q, where x and f are the vectors of integral and continuous variables, then the inequality

$$\sum_i F_{d,c}(a_i) x_i + \sum_j \bar{F}_{d,c}(b_j) f_j \geq F_{d,c}(z)$$

is also valid for Q, see Nemhauser and Wolsey [97]. In particular, if a valid inequality for the logical layer polyhedron P contains continuous flow variables and integer capacity variables then applying $F_{d,c}$ to its capacity coefficients and $\bar{F}_{d,c}$ to its flow coefficients yields another valid inequality for P. More details and explanations can be found in Raack et al. [115], where it is also shown that the MIR function $F_{d,c}$ is integral if a, c, and d are integral, and that $|F_{d,c}(a)| \leq |a|$ for all $a \in \mathbb{R}$. Both properties are desirable from a numerical point of view.

Cutset inequalities

Inequalities Let L_S be a cutset in the logical network as defined above. Obviously, the total capacity on the cut links L_S must be sufficient to accommodate the total demand over the cut:

$$\sum_{\ell \in L_S} \sum_{m \in M_\ell} C_\ell^m y_\ell^m \geq d_S^K. \tag{4.5}$$

Since all coefficients are nonnegative in (4.5) and $y_\ell^m \in \mathbb{Z}_+$, we can round down all coefficients to the value of the right-hand side (if larger). For notational convenience we assume $C_\ell^m \leq d_S^K$ for all $\ell \in L_S$ and $m \in M_\ell$ from now on. For any $c > 0$, applying the MIR-function $F_c := F_{d_S^K,c}$ to the coefficients and the right-hand side of (4.5) results in the *cutset inequality*

$$\sum_{\ell \in L_S} \sum_{m \in M_\ell} F_c(C_\ell^m) y_\ell^m \geq F_c(d_S^K). \tag{4.6}$$

In principle, c can be set to any positive value. For the case that c is one of the installable capacity values on a cutset link $\ell \in L_S$ (i. e., a coefficient of the base inequality (4.5)), Atamtürk [9] has shown that the resulting MIR inequalities can define a facet of the directed analogon of P. Moreover, for the single-facility case, where only one module type m_0 can be installed on all logical links, he has shown that the cutset inequalities constructed in this way together with the trivial facets completely describe the cutset polyhedron. Raack [113] proved that this result does not hold with undirected capacities. For values of c which are not coefficients of the base inequality, no general facet conditions are known.

Another crucial necessary condition for (4.6) to define a facet for P is that the two subgraphs defined by the network cut are connected [25]. This is trivially fulfilled if L contains logical links between all node pairs, which is usually the case in a multi-layer network.

Separation We now describe how we find violated cutset inequalities. A cutset inequality (4.6) is completely determined by its base inequality (4.5), which in turn depends only on the choice of the cutset in the logical network. Our separation procedure works as follows:

1. Choose a subset S of nodes and compute the corresponding logical cutset L_S.
2. Compute the base inequality (4.5) corresponding to L_S.
3. For all different capacity coefficients c occurring in the base inequality, compute the cutset inequality (4.6) using the MIR function $F_{d_S^K,c}$, and check it for violation.

In this way, we reduce the problem to finding a suitable cut in the logical network. In general, it is $\mathcal{NP}$-hard to find a cut for which the cutset inequality is maximally violated, see Bienstock et al. [27]. We thus apply a heuristic shrinking procedure to the logical network, similar to what has been done in [27, 64, 115] for single-layer problems:

1. For every logical link $\ell \in L$, let s_ℓ and π_ℓ be the slack and the dual value of the capacity constraint (3.4f) for link ℓ with respect to the current LP solution.
2. Construct an undirected graph from the logical graph (V, L) such that all parallel links are aggregated into one single link. For every such aggregated link ij, let $s_{ij} := \sum_{\ell \in L_{ij}} s_\ell$, and analogously $\pi_{ij} := \sum_{\ell \in L_{ij}} \pi_\ell$ and $w_{ij} := s_{ij} - \pi_{ij}$.
3. Iteratively shrink an aggregated link ij with maximum weight w_{ij}. If this produces parallel links, aggregate them and add up their weights. Every such shrinking step reduces the number of nodes by 1. Repeat this procedure until a pre-specified number $2 \leq k \leq |V| - 1$ of nodes has been reached.
4. Enumerate all cuts in this shrunken graph. For each of these cuts, compute the corresponding cutset inequalities and test them for violation, as described above.

The definition of w_ℓ is based on the heuristic argument that a cutset inequality is most likely to be violated if the slack of the base inequality is small. We thus want to keep links in the shrunken graph that have a small slack in the capacity constraints, i. e., we have to shrink links with a large slack s_ℓ. Notice that if a logical link has slack, the dual value of the capacity constraint is 0. Usually, many capacity constraints are tight in LP solutions, i. e., many slacks s_ℓ are 0. In those cases, we use the (often nonzero) dual values as a second sorting criterion, as an indicator of the expected increase of the LP value if capacity on a logical link is augmented. We shrink the graph until it has k nodes, where k is between 2 and 6; experiments have shown that larger values of k only lead to lots of violated cuts without further increasing the dual bound. In addition to the described shrinking procedure we check all single-node cutset inequalities for violation, that is, those with $|S| = 1$.

After calling all of its own and all user-defined separators, SCIP selects the best inequalities based on criteria such as the Euclidean distance of the corresponding hyperplane to the current fractional point, the degree of orthogonality to the objective function, and the degree of orthogonality among the violated inequalities, see Achterberg [3]. We left the corresponding parameters at their default values.

Flow-cutset inequalities

Inequalities Cutset inequalities can be generalized to flow-cutset inequalities, which have nonzero coefficients also for flow variables. Like cutset inequalities, flow-cutset inequalities are derived by aggregating capacity and flow-conservation constraints on a logical cut L_S and applying a mixed-integer rounding function to the coefficients of the resulting inequality.

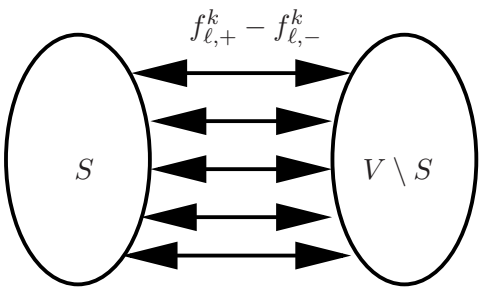
However, the way of aggregating the inequalities is more general. Various special cases of flow-cutset inequalities have been discussed in [9, 25, 34, 115, 116]. Necessary and sufficient conditions for flow-cutset inequalities to define a facet of P can be found in Raack et al. [116].

Consider fixed nonempty subsets $S \subset V$ of nodes and $Q \subseteq K$ of commodities. Assume that logical link $\ell \in L_S$ has end-nodes $i \in S$ and $j \in V \setminus S$. We will denote by $f^k_{\ell,-} := f^k_{\ell,ji}$ inflow into S on ℓ; similarly, $f^k_{\ell,+} := f^k_{\ell,ij}$ refers to outflow from S on ℓ.

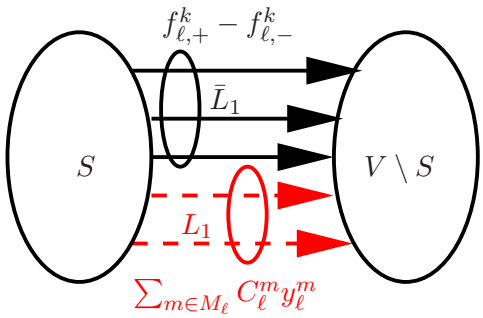
We construct a base inequality as follows. First, we obtain a valid inequality from the sum of the flow conservation constraints (3.4c) for all $i \in S$ and all commodities $k \in Q$:

$$\sum_{\ell \in L_S} \sum_{k \in Q} (f^k_{\ell,+} - f^k_{\ell,-}) \geq d^Q_S$$

This inequality, which is illustrated in Figure 4.4(a), states that the net flow out of S must be at least the demand from S to $V \setminus S$.



(a) Inflow minus outflow on all links must be at least d^Q_S.



(b) Capacity on L_1 plus outflow on $\bar{L}_1$ must be at least d^Q_S.

Figure 4.4: Construction of the flow-cutset base inequality.

Given a subset $L_1 \subseteq L_S$ of cut links and its complement $\bar{L}_1 := L_S \setminus L_1$ with respect to the cut, we can relax the above inequality by omitting the inflow variables and by replacing the flow by the capacity on all links in L_1:

$$\sum_{\ell \in L_1} \sum_{m \in M_\ell} C^m_\ell y^m_\ell + \sum_{\ell \in \bar{L}_1} \sum_{k \in Q} f^k_{\ell,+} \geq d^Q_S. \tag{4.7}$$

This is illustrated in Figure 4.4(b). Again we may assume $C^m_\ell \leq d^K_S$ for all $\ell \in L_1$ and $m \in M_\ell$.

For any module capacity $c > 0$ available on a logical cut link, define $F_c := F_{d^Q_S,c}$ and $\bar{F}_c := \bar{F}_{d^Q_S,c}$. Applying F_c to the capacity coefficients and $\bar{F}_c$ to the flow coefficients of the base inequality (4.7) leads to the valid *flow-cutset inequality*

$$\sum_{\ell \in L_1} \sum_{m \in M_\ell} F_c(C^m_\ell) y^m_\ell + \sum_{\ell \in \bar{L}_1} \sum_{k \in Q} f^k_{\ell,+} \geq F_c(d^Q_S). \tag{4.8}$$

Since $\bar{F}_c(1) = 1$, the coefficients of the flow variables remain unchanged. This inequality can be generalized to a flow-cutset inequality also containing inflow-variables, see Raack et al.

[115]. Studies of related polyhedra can be found in [9, 25, 34]. By choosing $L_1 = L_S$ and $Q = K$, inequality (4.8) reduces to the cutset inequality (4.6).

Separation For a more general class of flow-cutset inequalities which also contain inflow variables into the node set S, Atamtürk [9] showed by reduction to the max-cut problem that it is $\mathcal{NP}$-hard in general to find a most violated flow-cutset inequality which cuts off a given point. We follow a heuristic separation approach which is based on ideas from [9].

For separating a flow-cutset inequality, a set $S \subset V$ of nodes, a subset $Q \subseteq K$ of commodities, a capacity c, and a partition $(L_1, \bar{L}_1)$ of the cut links L_S have to be chosen. We apply two different separation heuristics. Both restrict the separation procedure to special subclasses of flow-cutset inequalities. Nevertheless, a large number of violated inequalities is found already with these restrictions.

The first heuristic considers singleton commodity subsets $Q = \{k\}$ and node subsets S consisting of one or two end-nodes of k. After fixing S and k and choosing an available capacity $c > 0$ on the cut, a partition of the cut links that maximizes the violation for flow-cutset inequalities is obtained by setting

$$L_1 := \left\{ \ell \in L_S \mid \sum_{m \in M_\ell} F_c(C_\ell^m) \bar{y}_\ell^m \leq \sum_{k \in Q} \bar{f}_{\ell,+}^k \right\}, \tag{4.9}$$

where $(\bar{f}, \bar{y})$ are the flow and capacity values on the logical graph in the current LP solution, see Atamtürk [9]. Obviously, the calculation of L_1 is done in linear time.

The second, more time-consuming heuristic finds a most violated flow-cutset inequality for a fixed single commodity $k \in K$ and a fixed capacity c, see Atamtürk [9]. First consider a protected point-to-point commodity $k \in K^p$. As we search for a subset S of nodes such that the end-nodes of the commodity are on different shores of the corresponding cut, the value d_S^Q equals the demand value d^k of the commodity. The crucial observation is that once k and c are fixed, both the right-hand side of the flow-cutset inequality (4.8) and the two values compared in (4.9) are known, and thus the partition of the potential cut links into L_1 and $\bar{L}_1$. The only remaining question is which links are part of the cut. For a point-to-point commodity k, this question can be answered in polynomial time by searching for a minimum-weight cut in the logical network between the end-nodes of the commodity with respect to the link weights

$$w_\ell := \min \{ \bar{f}_{\ell,+}^k, \sum_{m \in M_\ell} F_c(C_\ell^m) \bar{y}_\ell^m \}.$$

In the more general case of an aggregated unprotected commodity $k \in K^u$, we define an artificial target node linked to all targets of the commodity with a large weight, and search for a minimum-weight cut between the source and the artificial target. If no violated inequality is found in this way, we also try other partitions of the source and target nodes of k on the two sides of the cut, for instance, with the source and one target node on one side of the cut and all other target nodes on the other side, or with two targets on one side and all other end-nodes on the other side. The basic procedure always remains the same, except that the demand d_S^k over the cut has to be adapted according to the partition of nodes on the two sides.

4.4.2 Physical layer connectivity inequalities

Inequalities Because of the large values B_e, the physical link variables z_e usually have small fractional values in linear programming relaxations. By the demand routing requirements, we know that certain pairs of nodes have to be connected not only on the logical layer but also on the physical layer. Consequently, the variables z_e have to satisfy certain connectivity constraints. Notice that information from the physical layer is combined with the demands here, skipping the intermediate logical layer.

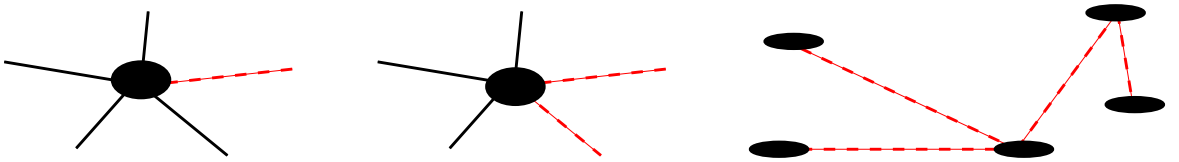
Connectivity problems have been studied on several occasions, in particular in the context of the Steiner Tree problem and fixed-charge network design, see [33, 61, 106]. Let $S \subset V$ be a subset of nodes and $\delta_E(S)$ the corresponding cut in the physical network. If some demand has to cross the cut then the inequality

$$\sum_{e \in \delta_E(S)} z_e \geq 1 \tag{4.10}$$

ensures that at least one physical link is installed on the cut. If a protected demand has to cross the cut, the right-hand side can even be set to 2 because the demand must be routed on at least two physically disjoint paths:

$$\sum_{e \in \delta_E(S)} z_e \geq 2. \tag{4.11}$$

Figures 4.5(a) and 4.5(b) illustrate these two inequality types for the special case $|S| = 1$.



(a) Every demand end-node must have physical degree at least 1.

(b) Every end-node of a protected demand must have physical degree at least 2.

(c) With a connected demand graph, the physical network must be at least a tree.

Figure 4.5: Degree and tree inequalities at the physical layer.

Furthermore, let $W := \{i \in V \mid d_i > 0\}$ be the set of demand end-nodes. Consider the demand graph, which is defined by the network nodes W with edges corresponding to traffic demands. If this demand graph is connected (which is the case in our instances) then the inequality

$$\sum_{e \in E} z_e \geq |W| - 1 \tag{4.12}$$

is valid. It says that at least a tree must be installed on the physical layer, because a path must exist from every demand end-node to every other demand end-node. This is illustrated in Figure 4.5(c). If protected demands exist and the demand graph is connected, inequality (4.13) can be strengthened by setting the right hand side to $|W|$. It can easily be seen, however, that if protected demands exist between all pairs of nodes, this inequality is dominated by the set of all single-node inequalities (4.11) with $|S| = 1$.

More generally, let p be the number of connectivity components of the demand graph with at least two nodes. Then the smallest number of physical links is used by constructing a tree in each demand component and not using any links between the components. This translates to the generalized tree inequality

$$\sum_{e \in E} z_e \geq |W| - p. \tag{4.13}$$

Notice that more links may be necessary if any component of the demand graph is physically disconnected, and that there may be other solutions that use the same minimum number of physical links. For this reason, we cannot construct a tree inequality (4.12) for every component of the demand graph.

Separation The tree inequality (4.13) can simply be added to the initial MIP formulation. The number of components of the demand graph is determined by a depth-first search.

The physical cutset inequalities (4.10) and (4.11) are identified in two ways. First, we test all cuts defined by single nodes $i \in V$ in each iteration, as these cuts turned out to be quite important. Second, we determine violated inequalities using a min-cut algorithm. The weight of a physical link e is set to its capacity value $\bar{z}_e$ in the current LP solution, which is exactly its contribution to the left-hand side of the inequality if the link is part of the cut. Then we search for a minimum cut with respect to these weights between every pair of nodes, and test the corresponding cutset inequalities for violation. Alternatively, one could determine an all-pairs minimum cut using a Gomory-Hu tree [60], which would lead to only one most violated inequality.

Proposition 4.1. *If $K = K^p$ or $K = K^u$ then the min-cut separation procedure is exact, i.e., it finds a violated physical layer cutset inequality (4.10) or (4.11) whenever one exists.*

Proof. If either all demands are unprotected or all demands are protected, the right-hand side of the inequality is fixed to 1 or 2 and does not depend on the cut. By construction, the weight of a minimum cut is exactly the minimum value of the left-hand side of (4.10) and (4.11). That is, the described procedure finds a cutting plane of this type with maximum violation. □

If, on the other hand, there are both protected and unprotected commodities, the right-hand side of the inequality depends on the cut (more specifically, on whether a protected demand has to be routed over the cut), and the procedure may fail to identify existing violated inequalities.

4.4.3 Two-layer link failure cutset and flow-cutset inequalities

During our computational tests, it turned out that the cutset and flow-cutset inequalities were extremely helpful with unprotected demands, but had little effect on the lower bound when protected demands were involved. A possible reason for this is the fact that the considered cutting planes have been originally designed for a multi-commodity flow in a single network layer and do not capture the additional survivability restrictions from the physical layer.

We thus have extended cutset and flow-cutset inequalities to address this issue. For each link failure state, we construct failure-specific cutset and flow-cutset inequalities to install enough capacity on the surviving cut links. To our knowledge, these are the first cutting

planes that take inter-layer survivability into account. They are based on an idea of Wang et al. [12, 90] and Bienstock and Muratore [26], who have studied the polyhedral properties of link failure cutset inequalities in the context of link restoration in single-layer networks. We have extended their cutting planes to the multi-layer context, to the case that both protected and unprotected demands may be present in the problem formulation, and to flow-cutset inequalities. The inequalities developed in this section are in turn used as model inequalities in Section 4.4.4 to extend the Q-subset inequalities of Magnanti and Wang [90] to two layers.

Apart from the results in this thesis, the polyhedral structure of multi-layer network design problems is still little known. Only recently, Raack and Koster [114] have investigated another kind of two-layer cutting planes. Whereas the cutting planes in this thesis aim at the survivability requirements, those studied in [114] focus on the connection of logical and physical layer capacities given by the physical link capacity inequality (3.4j). They exploit the fact that a logical link $\ell \in L_S$ always uses an odd number of physical cut links $e \in \delta_E(S)$. By combining the logical and physical layer capacity constraints and applying mixed-integer rounding, the authors derive valid inequalities and investigate under which conditions they define facets of an appropriately defined polyhedron in the space of logical and physical capacity variables.

Link failure cutset inequalities

For a given subset $S \subset V$ of nodes, let

$$d_S^{K,u} := \sum_{i \in S} \sum_{k \in K^u} d_i^k \quad \text{and} \quad d_S^{K,p} := \sum_{i \in S} \sum_{k \in K^p} d_i^k$$

be the total unprotected and protected demand to be routed over the cut links L_S. By definition of these values, the cutset base inequality (4.5) reads

$$\sum_{\ell \in L_S} \sum_{m \in M_\ell} C_\ell^m y_\ell^m \geq d_S^K = d_S^{K,u} + d_S^{K,p}. \tag{4.14}$$

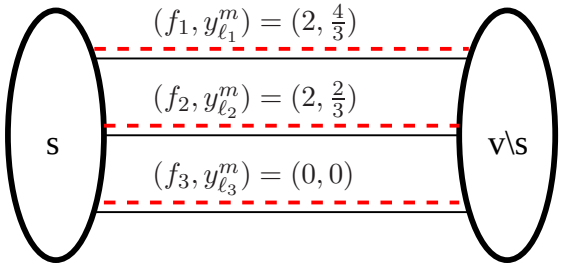
Balakrishnan et al. [12], Magnanti and Wang [90] observed that in any link failure situation, enough capacity must survive on the cut to reroute the affected protected demand. In our multi-layer context, this means that in any physical link failure situation, enough capacity must be installed on the surviving logical links to route a value of $d_S^{K,p}/2$ over the cut. This translates to the following inequality for every physical link $e \in E$ contained in some logical cut link:

$$\sum_{\ell \in L_S \setminus L^e} \sum_{m \in M_\ell} C_\ell^m y_\ell^m \geq \frac{d_S^{K,p}}{2}, \tag{4.15}$$

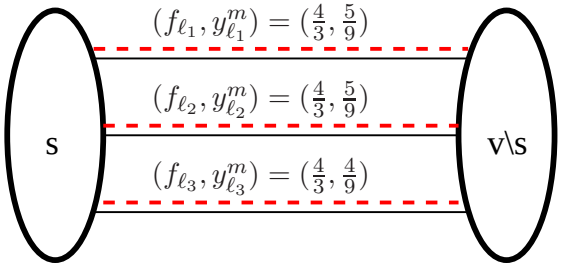
where L^e is the set of logical links containing physical link e. By applying the MIR function $F_c := F_{d_S^{K,p}/2,c}$ to this base inequality, we obtain the following *link failure cutset inequality*:

$$\sum_{\ell \in L_S \setminus L^e} \sum_{m \in M_\ell} F_c(C_\ell^m) y_\ell^m \geq F_c\Big(\frac{d_S^{K,p}}{2}\Big) \qquad \forall e \in E. \tag{4.16}$$

Again, the coefficients can be reduced to the right-hand side of the inequality before or after applying MIR (the result is the same, see Raack [113]).



(a) This fractional solution satisfies the cutset inequality (4.6) for the whole cut $y^m_{\ell_1} + y^m_{\ell_2} + y^m_{\ell_3} \geq 2$, but violates the link failure cutset inequality (4.16) $y^m_{\ell_2} + y^m_{\ell_2} \geq 1$ upon failure of the first link.



(b) This fractional solution satisfies the link failure cutset inequalities (4.16) $y^m_{\ell_i} + y^m_{\ell_j} \geq 1$ but violates the cutset inequality (4.6) for the whole cut $y^m_{\ell_1} + y^m_{\ell_2} + y^m_{\ell_3} \geq 2$.

Figure 4.6: Illustration of example 4.2: a cut consisting of three physical and three logical links. The protected demand out of S is 4, two flow units must survive, and logical link capacities can be installed in multiples of 3.

The following example shows that for a given cut L_S, the link failure cutset inequalities (4.16) may cut off fractional solutions satisfying the original cutset inequalities (4.6), but do not necessarily dominate the original inequalities if the cut has at least three physical links.

Example 4.2. Assume that a subset S of nodes is given such that $L_S = \{\ell_1, \ell_2, \ell_3\}$, and every logical link ℓ_i consists of exactly one physical link in the cut (i.e., the planning problem on the cut is essentially a single-layer planning problem). On the cut links, one module m of capacity $C^m_\ell = 3$ can be installed in integer multiples. Assume that $K^p = K$, and there is only one protected demand with value 4 over the cut, 2 of which must survive any link failure. The cutset MIR inequality (4.6) reads

$$y^m_{\ell_1} + y^m_{\ell_2} + y^m_{\ell_3} \geq 2,$$

whereas the link failure cutset inequalities (4.16) have the form

$$y^m_{\ell_i} + y^m_{\ell_j} \geq 1,$$

for $i, j \in \{1, 2, 3\}$, $i \neq j$. For ease of notation, assume that there is no flow into S; all flow variables in the sequel refer to flow on L_S out of S. The logical link flow vector $f = (2, 2, 0)$ defines a feasible flow on the cutset satisfying the demand and diversification constraints. The fractional capacity vector $y = \frac{1}{3}(4, 2, 0)$ defines sufficient capacities for this flow (recall that $C^m_\ell = 3$). These vectors are illustrated in Figure 4.6(a). The vector y satisfies the cutset inequality but not the link failure cutset inequality $y^m_{\ell_2} + y^m_{\ell_3} \geq 1$ for the first link. Conversely, for the feasible flow vector $f = \frac{4}{3}(1, 1, 1)$, the fractional capacity vector $y = \frac{1}{9}(5, 5, 4)$, which is illustrated in Figure 4.6(b), satisfies all link failure cutset inequalities but not the cutset inequality for the whole cut.

The example points out that both the standard cutset inequalities and the link failure cutset inequalities may be helpful in a cutting plane algorithm. Hence, for any cut in the network computed by the shrinking procedure described in the previous section, we construct both types of inequalities and test them for violation. The only exception are cutsets L_S with

only two physical links. In this case, the standard cutset inequality for the whole cut is just the sum of the two link failure cutset inequalities and thus dominated by these.

Similar inequalities can be constructed for node failures; under the assumption that end-nodes of cut links do not carry demand, such inequalities have been used by Bienstock and Muratore [26] as model constraints. Based on these, they derive a generalization of the k-cardinality constraint (see Section 4.4.4 and [12, 26]) using an aggregation and mixed-integer rounding procedure. If all nodes are demand end-nodes, as in our network instances, the demand value to be routed over the cut has to be adjusted for each failure situation because some demand may be lost in case of a node failure. As we could already identify lots of violated inequalities with the above link failure inequalities, we have restricted our implementation to link failure inequalities.

Link failure flow-cutset inequalities

A similar approach can also be applied to flow-cutset inequalities (4.8). Let S and Q be subsets of nodes and commodities. As for the link failure cutset inequalities, define

$$d_S^{Q,u} := \sum_{i \in S} \sum_{k \in K^u \cap Q} d_i^k \quad \text{and} \quad d_S^{Q,p} := \sum_{i \in S} \sum_{k \in K^p \cap Q} d_i^k,$$

such that $d_S^Q = d_S^{Q,u} + d_S^{Q,p}$. For any physical link $e \in E$, the valid inequality

$$\sum_{\ell \in L_S \setminus L^e} \sum_{k \in Q} (f_{\ell,+}^k - f_{\ell,-}^k) \geq \frac{d_S^{Q,p}}{2}$$

states that enough flow must be routed out of the node set S even if link e fails. By defining $F_c := F_{d_S^{Q,p}/2,c}$ and following the construction procedure for flow-cutset inequalities described in Section 4.4.1, we derive the *link failure flow-cutset inequality*

$$\sum_{\ell \in L_1 \setminus L^e} \sum_{m \in M_\ell} F_c(C_\ell^m) y_\ell^m + \sum_{\ell \in \bar{L}_1 \setminus L^e} \sum_{k \in Q} f_{\ell,+}^k \geq F_c\Big(\frac{d_S^{Q,p}}{2}\Big) \qquad \forall e \in E. \tag{4.17}$$

These inequalities are valid for P because they have been constructed by aggregating and relaxing valid inequalities and applying a suitable mixed-integer rounding function. As for cutset inequalities, they may cut off fractional points satisfying the standard flow-cutset inequalities, but do not dominate the latter.

4.4.4 Two-layer Q-subset inequalities

In the context of link restoration in a single-layer graph (V, L), Magnanti and Wang [90] have studied the following polyhedron defined in the space of capacity variables on a cut L_S (transferred to our notation):

$$Y_S^{MW} := \operatorname{conv}\Big\{ y \in \mathbb{Z}_+^{|L_S|} \;\Big|\; \sum_{\ell \in L_S \setminus \ell_0} y_\ell \geq b_{\ell_0} \quad \forall\, \ell_0 \in L_S \Big\}, \tag{4.18}$$

where $b_\ell \geq 0$ is the predefined working flow on logical link ℓ. The inequalities defining this polyhedron state that in every failure situation, enough capacity should be installed on the surviving logical links to reroute the affected demand.

The authors introduce the so-called *Q-subset inequalities*. For a given cut L_S of cardinality k, let $Q \subseteq L_S$ be a subset of the cut links of cardinality $q \geq 2$. Denote by $B_Q := \sum_{\ell \in Q} b_\ell$ the total working flow routed over links in Q, and define the mixed-integer rounding remainder $r_Q := r(B_Q, q-1)$. That is, r_Q is the remainder of the division of B_Q by $q-1$ if $B_Q/(q-1)$ is fractional, and $r_Q = q-1$ otherwise. Then the Q-subset inequalities have the following form:

$$r_Q \sum_{\ell \in Q} y_\ell + (r_Q + 1) \sum_{\ell \in L_S \setminus Q} y_\ell \geq r_Q \left\lceil \frac{B_Q}{q-1} \right\rceil . \tag{4.19}$$

In our terminology, these inequalities are constructed by aggregating the model inequalities in (4.18) for all links in Q and applying the mixed-integer rounding function $F_{B_Q,q-1}$ to the coefficients of the resulting base inequality (see Section 4.4.1 for details on mixed-integer rounding). In the special case where Q consists of all cut links ($q = k$), Inequality (4.19) is known as *k-cardinality inequality* and has been introduced by Balakrishnan et al. [12]. Magnanti and Wang derive necessary and sufficient conditions for the Q-subset inequalities to define facets of Y_S^{MW}, and demonstrate that these inequalities together with the non-negativity constraints and the model inequalities completely describe Y_S^{MW}.

Bienstock and Muratore [26] study a similar polyhedron, namely

$$Y_S^{BM} := \text{conv}\left\{ y \in \mathbb{Z}_+^{|L_S|} \;\middle|\; \sum_{\ell \in L_S} y_\ell \geq D, \quad \sum_{\ell \in L_S \setminus \ell_0} y_\ell \geq B \quad \forall\, \ell_0 \in L_S \right\}. \tag{4.20}$$

In contrast to Magnanti and Wang, Bienstock and Muratore consider an additional demand constraint, but assume the same right-hand side B for all failure inequalities. They give necessary and sufficient conditions for the k-cardinality constraint to the facet-defining for Y_S^{BM}, and describe the set of vertices of this polyhedron. They complement the results of Magnanti and Wang [90] by showing that every nontrivial facet of Y_S^{BM} is either one of the model inequalities defining the polyhedron or of the form

$$\sum_{\ell \in Q} y_\ell + \alpha \sum_{\ell \in L_S \setminus Q} y_\ell \geq R$$

for some $\alpha \geq 1$ and some suitable value of R, and that these inequalities can be obtained using sequential lifting. This formula includes the Q-subset inequalities. In contrast to Magnanti and Wang, Bienstock and Muratore do not characterize the facets of Y_S^{BM} by conditions on the input parameters, but by conditions on the number of positive entries in extreme points of this polyhedron. They also extend their model to a special case of node failures, as discussed in the previous subsection, and investigate various generalizations of Y_S^{BM}, which are defined in the space of capacity and flow variables. We will not discuss these in detail here.

We extend the Q-subset inequalities to multi-layer networks. For a subset S of nodes and given values $D, B \in \mathbb{Z}_+$, we study the polyhedron

$$Y_{2S} := \text{conv}\left\{ y \in \mathbb{Z}_+^{|L_S|} \;\middle|\; \sum_{\ell \in L_S} y_\ell \geq D, \quad \sum_{\ell \in L_S \setminus L^e} y_\ell \geq B \quad \forall\, e \in \delta_E(S) \right\}. \tag{4.21}$$

This model is the two-layer generalization of the model of Bienstock and Muratore [26]. The inequalities state that at least a capacity of D must be installed on the logical cut links in the failureless network state, and that upon any failure of a physical cut link, at least a

capacity of B must survive. Our motivation for studying these inequalities is that if $K = K^p$ and a single capacity module with capacity C is available on all cut links, these inequalities correspond precisely to the cutset (4.6) and link failure cutset inequalities (4.16) with

$$D := \lceil d_S^K / C \rceil \qquad \text{and} \qquad B := \left\lceil d_S^{K,p} / 2C \right\rceil . \tag{4.22}$$

As we consider physical link failures now, let $Q \subseteq \delta_E(S)$ be a subset of the physical cut links of cardinality $q := |Q| \geq 2$. Aggregating the failure inequalities of Y_{2S} on Q leads to the valid base inequality

$$\sum_{\ell \in Q} c_\ell y_\ell \geq qB \tag{4.23}$$

where $c_\ell \leq q$ is the number of physical links in the cut that are *not* traversed by ℓ. Define the MIR remainder $r_q := r(qB, q-1)$. It is the analogon of the value r_Q of Magnanti and Wang. Similar to Bienstock and Muratore [26], we assume the same right-hand side B for all failure inequalities (which is the case in our application). Consequently, the value r_q does not depend on the specific subset Q of links but only on its cardinality q. By applying the mixed-integer rounding function $F_{qB,q-1}$ to the coefficients of the base inequality (4.23), we obtain the *two-layer Q-subset inequality*

$$\sum_{\ell \in Q} \hat{c}_\ell y_\ell \geq r_q \left\lceil \frac{qB}{q-1} \right\rceil , \tag{4.24}$$

where

$$\hat{c}_\ell := F_{qB,q-1}(c_\ell) = \begin{cases} \min(c_\ell, r_q) & \text{if } c_\ell \leq q-1 \\ r_q + 1 & \text{if } c_\ell = q \end{cases} \tag{4.25}$$

is the MIR coefficient of c_ℓ.

Let the *cut length* of a logical link $\ell \in L_S$ be the number of physical links $e \in \delta_E(S)$ traversed by ℓ. If all logical links $\ell \in L_S$ have cut length 1, i.e., they contain only one physical link $e \in \delta_E(S)$, then all values c_ℓ are either $q-1$ or q (depending on whether this physical link is in Q or not) and the two-layer Q-subset inequality reduces to the single-layer Q-subset inequality of Magnanti and Wang [90].

Since these two-layer cutset inequalities have been constructed by aggregating model inequalities and applying mixed-integer rounding, the following proposition holds:

Proposition 4.3. *The two-layer Q-subset inequalities (4.24) are valid for the polyhedron Y_{2S} of feasible capacities.*

In the following, we summarize several necessary and sufficient facet conditions for two-layer Q-subset inequalities. For ease of notation, let $z := \frac{qB}{q-1}$.

Proposition 4.4. *If the two-layer Q-subset inequality (4.24) defines a facet of Y_{2S} then $q \geq 3$ and $z \notin \mathbb{Z}$.*

Proof. If $z \in \mathbb{Z}$, then (4.24) is a sum of model inequalities and cannot define a facet. If $q = 1$ then Y_{2S} is empty. If $q = 2$ then z is integer, which we just excluded. □

Proposition 4.5. *Under the following assumptions and those of Proposition 4.4, the two-layer Q-subset inequalities (4.24) define a facet of Y_{2S}:*

1. $D \leq \lfloor z \rfloor$,
2. $B \geq r_q q$,
3. $q < k$.

Proof. For the short logical links with cut length 1, the proof follows the lines of the corresponding proof for single-layer Q-subset inequalities in Magnanti and Wang [90]. For logical links ℓ of cut length larger than 1, an additional, quite technical case differentiation is necessary to cover the various possible values of the coefficient $\hat{c}_\ell$. For each of the three cases $c_\ell < r_q$ or $r_q \leq c_\ell \leq q-1$ or $c_\ell = q$, the idea is to construct a feasible solution which satisfies (4.24) at equality, and to relate $\hat{c}_\ell$ to the coefficients of short logical links which are already known (up to a common multiple) to be either r_q or $r_q + 1$. Details can be found in the diploma thesis [96]. □

The third facet condition of Proposition 4.5 is needed to construct feasible solutions in the proof. There are, however, cases where the two-layer Q-subset inequality defines a facet also for $q = k$; some of these are characterized in more detail in [96].

The second facet condition, $B \geq r_q q$, replaces the sufficient condition $kB/(k-1) < \lfloor qB/(q-1) \rfloor$ of Magnanti and Wang [90] (transferred to our notation) and is needed in the proof to construct a certain feasible point in the polyhedron Y_{2S}. Notice that the condition of Magnanti and Wang is violated for $q = k$. Investigations with PORTA [35] showed that there exist two-layer Q-subset inequalities which define facets of Y_{2S} but violate the condition $B \geq r_q q$.

The first facet condition, $D \leq \lfloor z \rfloor$, says that the demand value must not be too large compared to B. Magnanti and Wang [90] have no such condition by lack of a demand constraint. For the special case that all logical links have cut length 1 (which corresponds to the single-layer case), Müller [96] showed that if this condition is violated, the demand inequality dominates the Q-subset inequalities if either $D = \lceil z \rceil$, or $q = k$ and $D > \lceil z \rceil$. To avoid the latter case, Bienstock and Muratore [26] set the right-hand side of the single-layer k-cardinality constraint to $\max\{D, \lceil z \rceil\}$. If there are logical links of cut length larger than 1, or if $q < k$ and $D > \lceil z \rceil$, examples investigated with PORTA revealed that the two-layer Q-subset inequality may define a facet of Y_{2S} even though the condition is violated.

Unfortunately, the latter facet condition is nearly always violated in our application: if all demands are protected then

$$D \approx 2B \geq \left\lfloor \frac{qB}{q-1} \right\rfloor$$

holds for every $q \geq 3$ and every $B \in \mathbb{Z}_+$, see (4.22). In a computational study with only one facility on each logical link [96], violated two-layer Q-cutset inequalities could only be found for a demand-wise shared protection model [63, 71, 136] where the ratio D/B is smaller than in our 1+1 protection model. With 1+1 protection, no violated two-layer Q-subset inequalities could be identified if the cutset inequalities (4.6) are already present in the LP (even in the basic form, without exploiting link failures). In view of these results, we refrained from implementing and testing the two-layer Q-subset inequalities for the purpose of this thesis.

4.4.5 Improving the numerical stability

Often, the identified cutset and flow-cutset inequalities have large integer coefficients with a greatest common divisor larger than 1. The separated metric inequalities (see Section 4.5),

on the other hand, usually have fractional coefficients between 0 and 1 which can be made integer by scaling the inequality with a small positive integer. To improve the numerical stability of the solution process, we apply a scaling procedure to all separated inequalities:

- First, if there are fractional coefficients, multiply the inequality with $360 = 2^3 \cdot 3^2 \cdot 5$. This often makes all coefficients integer.
- If this does not work, then the coefficients are often all equal and of the form $1/n$ for some prime number $n \geq 7$. We iterate through the coefficients; as long as we find a fractional coefficient c, we divide the inequality by c. If all variables have the same coefficient $c = 1/n$ for some integer n, this step makes the inequality integer.
- If all coefficients are integer after any of these steps, we divide the whole inequality by the greatest common divisor of all coefficients to make them smaller.

Using this procedure, we often end up with "nice" inequalities having small integer coefficients. If intermediate numbers get too large or if the final coefficients are still fractional, we add the separated inequality to the LP as-is.

4.5 Metric inequalities

In this section, we describe a decomposition approach where the routing formulation is moved to a subproblem and used to generate so-called *metric inequalities*. We first explain the basic idea of the decomposition and its consequences in Section 4.5.1. In Sections 4.5.2 and 4.5.3, we describe two variants of the routing LP, some properties of metric inequalities, and how to separate them in case of feasible or infeasible LPs, respectively. Section 4.5.4 presents a new technique to limit the size of the involved linear programs, namely adding commodities to the routing formulation dynamically. Section 4.5.5 explains how to strengthen the resulting metric inequalities in order to incorporate information about all commodities even if only a subset of them is included in the routing formulation. Eventually, we discuss some implementational issues in Section 4.5.6, which are necessary make the approach work in practice.

4.5.1 Basic idea

Polyhedra In our models, we consider a fractional multi-commodity routing without flow cost; the only cost is incurred by capacity variables. From a cost perspective, the most interesting polyhedron is the convex hull of all feasible integer node and link capacity vectors. The term "feasible" is defined in two parts. One part is given by the constraints on node, logical link, and physical link capacities. The other part is defined by the condition that the logical link capacities must be sufficient to satisfy all routing constraints. From this point of view, the flow variables are merely auxiliary variables to define feasibility of a capacity vector.

Stated more formally (in terms of model UNPROTECTED-BASE (3.1)), the original multi-layer network design polyhedron is defined by

$$P := \text{conv } \{a = (f, y, z, x) \mid a \text{ satisfies (3.1b)–(3.1i)}\}. \tag{4.26}$$

By projecting out the flow variables and defining the total capacity of a logical link $\ell \in L$ as

$$y_\ell := \sum_{m \in M_\ell} C_\ell^m y_\ell^m, \tag{4.27}$$

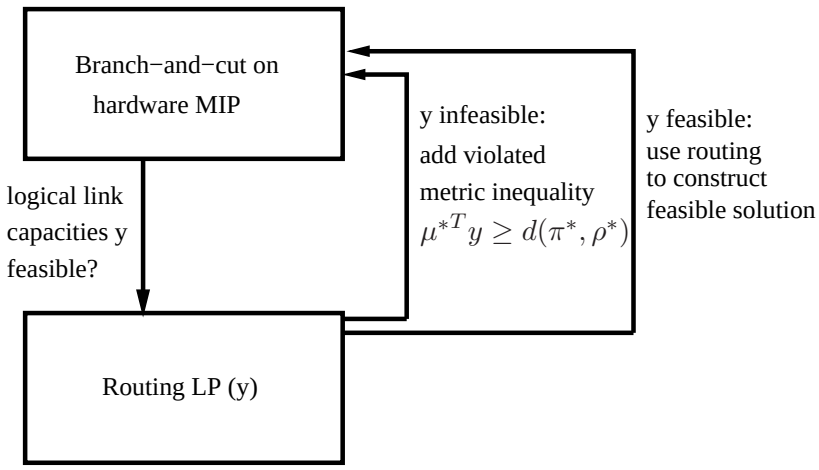


Figure 4.7: Benders decomposition applied to network design.

the polyhedron of feasible fractional capacity vectors is given as

$$Y := \text{conv}\,\{y \in \mathbb{R}_+^{|L|} \mid y \text{ allows for a routing satisfying } (3.1b), (3.1c), (3.1i), (4.27)\}. \quad (4.28)$$

A feasible solution to the overall network design problem is then defined by the following polyhedron:

$$P_{\text{metric}} := \text{conv}\ \{(y, z, x) \mid (y, z, x) \text{ satisfies } (3.1d)\text{–}(3.1h),\ y \in Y\}\,. \quad (4.29)$$

By definition, the polyhedra P and P_{metric} are equivalent: for every point in one polyhedron, there is a point in the other polyhedron with the same cost according to (3.1a).

The interesting point about this decomposition is that the polyhedron Y can be described by so-called *metric inequalities* which are defined in terms of capacity variables y and can be separated in polynomial time. They will be explained in detail in Section 4.5.2.

Benders decomposition The decomposition can be used algorithmically as illustrated in Figure 4.7. The branch-and-cut algorithm is applied to a master problem containing the objective function, the node and link capacity variables, and the node and physical link capacity constraints, i.e., it is based on the polyhedron P_{metric}. The multi-commodity flow formulation on the logical layer is moved to a satellite problem. Whenever a feasible solution (y, z, x) of the master problem is found, the condition $y \in Y$ is tested, i.e., the capacities are tested for feasibility with respect to the routing constraints. If a feasible routing exists within these capacities, a solution to the whole problem has been identified. If no feasible routing exists, i.e., $y \notin Y$, a violated metric inequality can be generated from the satellite problem to cut off the infeasible capacity vector.

In a more general setting, this decomposition approach is known as *Benders decomposition* [21]. This term refers to the decomposition of a problem into a master MIP and one or more satellite problems. With the classical method, the master MIP is solved to optimality. The optimal solution is then tested for feasibility or optimality with respect to the satellite problems; if it turns out to be infeasible or suboptimal, it is cut off by a *feasibility cut* or *optimality cut* generated from the corresponding satellite problem, and the master MIP is solved

again. In practice, the feasibility test should not only be applied to the optimal MIP solution but also to intermediate feasible or partially feasible solutions; for instance, to intermediate integer feasible solutions or to fractional LP solutions. This is discussed in more detail in Section 4.5.6.

The possible advantage of this decomposition in the network design setting is that the largest part of the MIP, namely the routing variables and constraints, are moved to a satellite LP. Consequently, the linear programs occurring during the branch-and-cut process are by orders of magnitude smaller compared to the full MIP with flow variables, and are solved much faster. Of course, the routing LP must still be solved to separate metric inequalities, but if this does not happen too often, the decomposition approach allows to solve many more branch-and-cut nodes than with the full MIP.

Literature The decomposition approach just described can be applied to a variety of network design problems. Since only the total capacity of the logical links is relevant in the definition of Y, other capacity models can easily be incorporated by adjusting the definition of y_ℓ in (4.27). Different routing constraints can be incorporated by defining Y accordingly in (4.28).

The method of applying Benders decomposition to network design problems by means of metric inequalities has been known for a long time and is due to Iri, Onaga, and Kakusho [74, 98]. It has been successfully used in various single-layer network design problems [27, 99, 110, 135], and recently also in the context of a very simple multi-layer setting, see Fortz and Poss [52], Knippel and Lardeux [76]. Nevertheless, it is non-trivial to implement and use metric inequalities correctly and efficiently in the context of a branch-and-cut algorithm with many ingredients, as discussed in Section 4.5.6.

We follow two different approaches to test whether a given logical link capacity vector y allows for a feasible routing. The first one minimizes a bottleneck variable describing the extra capacity needed, whereas the other one tests the routing LP for feasibility only and works with a dual ray in case of infeasibility. These two methods will be described in the following two subsections. For ease of exposition, we will explain most of the techniques only for the model PROTECTED (3.4), assuming protected commodities with logical link flow variables $f^k_{\ell,ij}$. Adapting the presented ideas to node-pair flow variables f^k_{ij} is straightforward and will only briefly be discussed at the end of Section 4.5.2.

4.5.2 Construction using a bottleneck variable

Routing LP

In the bottleneck approach, we turn the feasibility problem

> Do the capacities y allow for a feasible routing?

into the optimization problem

> What is the minimum capacity α that has to be installed on any link such that the capacities $y_\ell + \alpha$ allow for a feasible routing?

To this end, we introduce a slack variable α in the capacity constraints and minimize it. This variable denotes the maximum extra capacity that has to be installed on any logical link in order to allow for a feasible routing. Of course, it would also be possible to minimize a sum

of link-dependent variables α_ℓ. For notational ease, assume first that there are only protected commodities, i. e., $K = K^p$; the case of unprotected commodities is dealt with at the end of this section. Furthermore, let $f^k_\ell := f^k_{\ell,ij} + f^k_{\ell,ji}$ denote the total flow on logical link ℓ in any direction. Given a fractional or integer capacity vector y from the master MIP, the routing LP reads

$$\min \quad \alpha \tag{4.30a}$$

$$\text{s.t.} \quad [\pi^k_i \in \mathbb{R}] \quad \sum_{j\in V}\sum_{\ell\in L_{ij}} (f^k_{\ell,ij} - f^k_{\ell,ji}) = d^k_i \qquad \forall\, i \in V,\ \forall\, k \in K \tag{4.30b}$$

$$[\mu_\ell \geq 0] \quad \alpha - \sum_{k\in K} f^k_\ell \geq -y_\ell \qquad \forall\, \ell \in L \tag{4.30c}$$

$$[\rho^k_i \geq 0] \quad -\sum_{\ell\in L^i} f^k_\ell - \sum_{\ell\in\delta_L(i)} \frac{1}{2} f^k_\ell \geq -\frac{1}{2} d^k \qquad \forall\, i \in V,\ \forall\, k \in K^p \tag{4.30d}$$

$$[\rho^k_e \geq 0] \quad -\sum_{\ell\in L^e} f^k_\ell \geq -\frac{1}{2} d^k \qquad \forall\, e \in E,\ \forall\, k \in K^p \tag{4.30e}$$

$$\alpha, f^k_{\ell,ij}, f^k_{\ell,ji} \geq 0 \tag{4.30f}$$

The symbols in brackets on the left-hand side denote the dual variables of the corresponding constraints. In this section we assume that this LP is feasible and can be solved to optimality either as-is or using column generation. In particular, this means that the set of variables included in the LP is sufficient to route the demands in a survivable way if no capacity constraints are present. The case of infeasible LPs is dealt with in the next section. Alternatively, the LP could be made feasible by introducing further dummy variables in the demand constraints, which are then minimized together with α.

As a reference in this and the next chapter, we now present the dual LP of (4.30). To simplify notation, let

$$\rho^k_\ell := \frac{1}{2}(\rho^k_i + \rho^k_j) + \sum_{v\in V:\, \ell\in L_v} \rho^k_v + \sum_{e\in\ell} \rho^k_e \tag{4.31}$$

be the aggregated diversification dual of logical link $\ell \in L$ for commodity $k \in K^p$. An interpretation of these values will be given in Section (4.6.2) in the context of column generation. By straightforward application of LP duality, the dual linear program to (4.30) reads as follows:

$$\max \quad \sum_{k\in K}\sum_{i\in V} d^k_i \pi^k_i - \sum_{k\in K^p} \frac{d^k}{2}\Big(\sum_{i\in V} \rho^k_i + \sum_{e\in E} \rho^k_e\Big) - \sum_{\ell\in L} \mu_\ell y_\ell \tag{4.32a}$$

$$[\alpha] \quad \sum_{\ell\in L} \mu_\ell \leq 1 \tag{4.32b}$$

$$[f^k_{\ell,ij}] \quad \pi^k_i - \pi^k_j \leq \mu_\ell + \rho^k_\ell \qquad \forall\, i,j \in V,\ i \neq j,\ \forall\, \ell \in L_{ij},\ \forall\, k \in K \tag{4.32c}$$

$$\mu_\ell, \rho^k_\ell \geq 0,\ \pi^k_i \in \mathbb{R} \tag{4.32d}$$

Notice that the dual flow variable constraints (4.32c) are defined for both ordered pairs (i, j) and (j, i), according to the two directions of flow variables on a logical link.

Metric inequalities

For notational convenience, define

$$d(\pi, \rho) := \sum_{k \in K} \sum_{i \in V} d_i^k \pi_i^k - \sum_{k \in K^p} \frac{d^k}{2} \Big(\sum_{i \in V} \rho_i^k + \sum_{e \in E} \rho_e^k \Big) \tag{4.33}$$

as the demand-dependent part of the dual objective function (4.32a). Let α^* be the optimal solution value of (4.30). Since the dual LP has an optimal solution with the same objective value and the dual LP is a maximization problem, every feasible dual solution (π, μ, ρ) has an objective value

$$d(\pi, \rho) - y^T \mu \leq \alpha^*.$$

By construction, the following proposition holds:

Proposition 4.6. *The capacities y are feasible with respect to the routing constraints if and only if $\alpha^* \leq 0$, i.e., if and only if every feasible dual solution (π, μ, ρ) satisfies*

$$y^T \mu \geq d(\pi, \rho). \tag{4.34}$$

Inequalities (4.34) are referred to as *metric inequalities*. The above statement says that under the assumption of a fractional multi-commodity flow, the polyhedron Y of all feasible fractional capacity vectors is completely described by all metric inequalities. Notice that although there are infinitely many metric inequalities (every dual solution defines one), only a finite number is actually needed because Y is a polyhedron and can be described by a finite set of linear inequalities. The described construction can easily be extended to other types of routing constraints like hop limits or other survivability restrictions if the metric inequalities are adapted accordingly, see [99, 135] for instance.

Cutset base inequalities (4.5) are a special case of metric inequalities. Assume that there are only point-to-point commodities (aggregated unprotected commodities can be handled similarly). Given a cut L_S in the logical network, we can construct a feasible dual solution as follows:

- $\mu_\ell := 1/|L_S|$ for all $\ell \in L_S$, and $\mu_\ell := 0$ otherwise;
- $\pi_i^k := 1/|L_S|$ if i and t^k are on different shores of the cut, and 0 otherwise;
- $\rho := 0$.

An interpretation of these values will be given in Section 4.5.5. The resulting metric inequality, after scaling with $|L_S|$, reads as $\sum_{\ell \in L_S} y_\ell \geq d_S^K$, which is precisely the base inequality (4.5) for cutset inequalities. The condition $\rho = 0$ reflects the fact that cutset inequalities in their basic form do not take the survivability constraints into account. General metric inequalities may have a stronger right-hand side even if defined on the same cutset, and they may be defined on more general sets of logical links than on a network cut.

Separation

If $\alpha^* \leq 0$ then all metric inequalities are satisfied by construction. Suppose, in contrast, that the LP has been optimally solved with an objective value of $\alpha^* > 0$, i.e., the capacities are not sufficient. Let (π^*, μ^*, ρ^*) be a corresponding optimal dual solution. From the last section,

we know that this dual solution satisfies $y^T\mu^* < d(\pi^*, \rho^*)$. Hence, if the optimal value of the routing LP is positive, we fetch the dual solution from the LP solver and construct the metric inequality $y^T\mu^* \geq d(\pi^*, \rho^*)$, which asks for more capacity than currently installed. To take the integrality of the capacity variables into account, we additionally apply mixed-integer rounding to the separated cutting planes and scale the resulting inequality as described in Section 4.4.5.

Taking variable bounds into account

In publications on network design with metric inequalities, flow variables are usually assumed to be unbounded from above. In our case, however, the flow variables are bounded by their demand value. From the LP-based probing described in Section 4.2.2, they may be additionally bounded from above by some value smaller than the demand value. These bounds have to be taken into account; otherwise the separated metric inequalities may be invalid and cut off optimal solutions.

Even though we do not have nonzero lower bounds on flow variables, assume for the sake of completeness that a specific variable $f^k_{\ell,ij}$ is bounded by $l^k_{\ell,ij} \leq f^k_{\ell,ij} \leq u^k_{\ell,ij}$. By adding the bounds to the routing LP (4.30) as explicit inequalities with associated dual variables $\underline{\gamma}^k_{\ell,ij} \geq 0$ and $\overline{\gamma}^k_{\ell,ij} \leq 0$, it follows from standard LP duality that the right-hand side of the metric inequality (4.34) must be augmented by the term

$$l^k_{\ell,ij}\underline{\gamma}^k_{\ell,ij} + u^k_{\ell,ij}\overline{\gamma}^k_{\ell,ij} \tag{4.35}$$

for all bounded flow variables $f^k_{\ell,ij}$ in the routing LP.

In any useful implementation, the flow variable bounds are stored as such and not as explicit inequalities. In this case, the dual variables of the bound constraints are precisely given by the reduced cost values of the variables as obtained from the LP solver. If the reduced cost r of a variable $f^k_{\ell,ij}$ is positive then this variable is at its lower bound and $r = \underline{\gamma}^k_{\ell,ij}$; if $r < 0$ then the upper bound is binding and $r = \overline{\gamma}^k_{\ell,ij}$; if $r = 0$ then $f^k_{\ell,ij}$ is either in the basis or 0 and r does not affect the metric inequality. Notice that in the approach just described, the right-hand side of the metric inequality can alternatively be computed as

$$d(\pi^*, \rho^*) := \alpha^* + y^T\mu^* \tag{4.36}$$

This formula, however, is not applicable with the approach without slack variable described in the following Section 4.5.3.

Combining protected and unprotected commodities

If there are unprotected commodities $k \in K^u$ with aggregated node-pair flow variables f^k_{ij}, the routing LP (4.30) contains the additional node-pair capacity constraints

$$\alpha - \sum_{k\in K^u}(f^k_{ij} + f^k_{ji}) - \sum_{\ell\in L_{ij}}\sum_{k\in K^p} f^k_\ell \geq -\sum_{\ell\in L_{ij}} y_\ell \qquad \forall\, i,j \in V,\ i \neq j \tag{4.37}$$

with dual variable $\mu_{ij} \geq 0$. Consequently, the dual objective function (4.32a) changes to

$$\max\ d(\pi,\rho) - \sum_{i,j\in V}\sum_{\ell\in L_{ij}} (\mu_\ell + \mu_{ij})y_\ell$$

By defining an auxiliary variable $y_{ij} := \sum_{\ell \in L_{ij}} y_\ell$ for all pairs of nodes $i, j \in V$, the resulting metric inequality can still be written as $y^T \mu \geq d(\pi, \rho)$, as in (4.34). Also the separation procedure for metric inequalities does not change. Variable bounds for node-pair flow variables f^k_{ij} can be handled in the same way as for logical link flow variables $f^k_{\ell,ij}$.

4.5.3 Construction using infeasible LPs

A major drawback of the above approach is that the LP solver occasionally takes a lot of time to solve the LP to optimality. In particular, the dual simplex algorithm, which always maintains feasible dual solutions, often quickly finds a dual solution with positive objective value, but takes a lot of iterations to actually find the maximum dual objective value α^*. The exact optimal value, however, does not really matter; the only interesting question is whether it is positive or not. It is thus possible to abort the LP solving process as soon as a dual solution with positive objective value c has been found, and use that dual solution to construct a metric inequality. The drawback of this approach is that the violation c of the metric inequality will be very small.

This section describes an alternative approach, namely using infeasibility information to generate metric inequalities. Although this approach is theoretically not new, it has to our knowledge never been used in the network design literature yet.

Metric inequalities from a dual ray

Instead of using a bottleneck variable α as in the previous section, we can remove it from the objective (4.30a) and the capacity constraints (4.30c), and ask whether the remaining LP has a solution or not (i.e., we minimize the zero objective function). If it has a solution then the capacities are feasible. Otherwise, we construct a metric inequality from a dual ray as follows.

First, notice that in our case, the zero vector is always a feasible dual solution. Consequently, infeasibility of the primal LP implies unboundedness of the dual LP. Assume that $c \geq 0$ is a nonnegative vector of appropriate size and that all variables are nonnegative and unbounded from above. Setting aside the variable bounds discussed in the following section, these conditions are satisfied in our case. One of the many implications of Farkas' lemma (see Schrijver [124]) is that if the LP

$$\min_{x \geq 0} \{c^T x \mid Ax \geq b\}$$

is infeasible and the dual LP

$$\max_{y \geq 0} \{b^T y \mid y^T A \leq c^T\}$$

has a feasible solution then there exists a dual ray $r \geq 0$ such that

$$r^T A \leq 0 \quad \text{and} \quad b^T r > 0. \tag{4.38}$$

These properties imply that (a) the dual ray r is itself a feasible dual solution, and (b) for any feasible dual solution y, the vector $y + r$ is also dual feasible and has a larger objective value than y. Hence, this dual ray can be used to prove unboundedness of the dual LP and thus infeasibility of the primal LP.

Applied to our setting, property (a) means that we can interpret r as a dual solution and construct a metric inequality out of it as described in Section 4.5.2, and property (b) implies that this metric inequality will be violated because its dual objective value is positive.

Of course, the same approach can be used if the LP is infeasible for other reasons than insufficient capacities. One such reason might be, for instance, that the set of initial variables in a column generation process does not admit a feasible routing for topological reasons.

In order to retrieve a dual ray from the LP solver in case of primal infeasibility, the dual simplex algorithm must be used, and LP presolving must be switched off. Also notice that with an infeasible LP, there is no guarantee to find a most violated metric inequality, in contrast to the bottleneck approach. That is, metric inequalities can be separated faster without the α variable, but they may be less violated.

Taking variable bounds into account

As in the bottleneck approach, bounds on the flow variables have to be taken into account to ensure validity of the separated metric inequalities. This time, formula (4.36) cannot be applied to compute the right-hand side because there is no optimal solution value α^*. Instead, the value $d(\pi, \rho)$ must be computed directly. Moreover, the dual ray r obtained from the LP solver only contains dual variables for the regular constraints, but not for the bounds. The interpretation of the dual ray r as returned by CPLEX as an LP solver is the following (see also [39]): To prove infeasibility of the system of linear inequalities

$$\{x \geq 0 \mid Ax \geq b,\ l \leq x \leq u\},$$

the LP solver provides us with a dual vector $r \geq 0$ such that the valid inequality

$$r^T Ax \geq r^T b \tag{4.39}$$

cannot be satisfied within the variable bounds. Even a point z maximizing the left-hand side of (4.39), i.e., with

$$z_j := \begin{cases} u_j & \text{if } r^T A_j > 0 \\ l_j & \text{if } r^T A_j < 0 \\ \text{anything between } l_j \text{ and } u_j & \text{if } r^T A_j = 0, \end{cases}$$

will violate inequality (4.39). This proves the invalidity of that constraint, and thus the infeasibility of the primal system. In practice, a dual ray is usually sparse and has nonzero coefficients for a small subset of constraints which, together with the variable bounds, suffices to prove infeasibility.

The vector $y^T := -r^T A$ takes the role of the reduced cost vector in the bottleneck approach. If $y_j < 0$ then the upper bound is binding, and the value $y_j u_j$ must be added to the right-hand side of the metric inequality (thus decreasing its value); if $y_j > 0$ then the lower bound is binding and the right-hand side must be augmented by $y_j l_j$; if $y_j = 0$ then the bounds of this variable are not binding.

4.5.4 Adding commodities dynamically

On large network instances, solving the routing LP to optimality (with a given set of variables) and proving infeasibility of the current capacities can take several hours. Especially at the beginning of the solution process, the LP capacities are relatively small and do not even allow to route a small subset of the commodities. To speed up the infeasibility proof of a capacity vector, we have developed an iterative approach, starting from the empty LP:

1. Add a group of commodities to the routing LP and solve it.
2. If the capacities are infeasible with respect to the current commodities, generate a metric inequality and stop.
3. If there are still commodities to add, go to 1. Otherwise the capacities are feasible, stop.

The advantage of this approach, especially at an early stage, is that the LPs are much smaller and thus solved much faster than if all commodities were added from the beginning. The drawback is that the separated metric inequalities are weaker because they only incorporate information about a subset of commodities. There is thus a trade-off between the size of the LPs and the quality of the separated inequalities. Also notice that as soon as a feasible integer solution is found in the branch-and-cut process, all commodities have to be added to the LP during the feasibility test. It may thus be wise to solve the root node with a subset of commodities and to call primal heuristics only at the end of the root node.

Adding commodities dynamically can be interpreted as a technique which, in the context of general LPs, is known as *partial pricing* [138]. In that context, partial pricing means that the simplex algorithm first considers only a subset of variables which form a structural block in the inequality system. Only when the LP is optimally solved with respect to these variables (or the progress in the LP bounds gets too small), the next block of variables is taken into account. In our multi-layer setting, a block correspond to all flow variables for a group of commodities; to avoid empty constraints without variables, we also add the constraints for a commodity together with the corresponding variables.

There are many possible strategies to group the commodities and to determine the order in which the groups are added to the LP. We have tried two of them: adding the commodities one by one, sorted by decreasing demand value, or adding groups of commodities with the same source or target node, sorted by decreasing emanating demand of the nodes. Furthermore, we have tested different strategies of adding a new group of commodities before the LP is optimally solved, as described in the computational results section 5.2.5.

Some primal heuristics, like the InstallCapHeur heuristic presented in Section 4.3.1, are based on LP routings. Since the master MIP does not comprise any routing variables in the Benders decomposition approach, the routing must be taken from the routing LP. If commodities are added dynamically, we use the LP routing for those commodities that are already included. For the other commodities, we search for a routing within the current capacities using the FindFeasibleRouting heuristic described in Section 4.3.2. The link weights in the shortest-path computation are defined by the dual capacity cost values μ_ℓ from the metric LP in order to favor routing paths on logical links which have slack in the capacity constraints. Notice that if the feasibility version of metric inequalities is used (i.e., without bottleneck variable α), the routing LP may have to be resolved with that bottleneck variable first in order to compute a routing (which may or may not fit into the current capacities).

4.5.5 Strengthening metric inequalities

If only a subset of commodities is included in the routing LP, the generated metric inequalities tend to be weak because they only incorporate information about the included commodities. The right-hand side of the metric inequalities can, however, be strengthened based on information about the missing commodities. The same approach can also be used to improve a dual ray in case of an infeasible routing LP.

We first prove a necessary optimality condition for the dual LP (4.32). Suppose that this LP has an optimal solution (i.e., it is bounded), and that there are only point-to-point commodities.

Proposition 4.7. *For fixed dual vectors μ, ρ of the dual LP (4.32), the dual objective function (4.32a) is maximized by setting π_i^k to the shortest-path distance from i to t^k with respect to the logical link weights $\mu_\ell + \rho_\ell^k$ for every $k \in K$.*

Proof. Let $k \in K$ a point-to-point commodity. Recall that d_i^k is defined as

$$d_i^k := \begin{cases} d^k & \text{if } i = s^k \\ -d^k & \text{if } i = t^k \\ 0 & \text{otherwise} \end{cases}$$

and that the contribution of π^k to the dual objective function (4.32a) is

$$c(\pi^k) := \sum_{i \in V} d_i^k \pi_i^k = d^k(\pi_{s^k}^k - \pi_{t^k}^k).$$

Obviously, for given μ, ρ, maximizing the objective value is equivalent to maximizing $\pi_{s^k}^k - \pi_{t^k}^k$. By summing up the dual constraints (4.32c) for all logical links of any s^k-t^k-path p in the logical network, we obtain the feasibility condition

$$\pi_{s^k}^k - \pi_{t^k}^k \leq \sum_{\ell \in p} (\mu_\ell + \rho_\ell^k). \tag{4.40}$$

It says that $\pi_{s^k}^k - \pi_{t^k}^k$ must be at most the length of any s^k-t^k-path p with respect to $\mu_\ell + \rho_\ell^k$. Setting π_i^k to the shortest-path length with respect to these weights yields a feasible dual solution with $\pi_{t^k}^k = 0$, maximum possible $\pi_{s^k}^k$, and thus maximum objective value. On all shortest paths, equality holds in (4.40). □

Without protection, the values ρ_ℓ^k are zero. A classical interpretation of the above proposition in this case is that if link $\ell \in L$ incurs a cost of μ_ℓ for every unit of flow, then the cheapest way to route a demand is on a shortest path with respect to these weights. Proposition 4.7 generalizes this to the case of protected commodities.

For unprotected commodities consisting of several demands aggregated at a common target node, the same construction works by applying Proposition 4.7 to every single source node of the commodity. Notice that because all demands and links are undirected, this link cost is independent of the direction of the path. Hence, if unprotected demands are aggregated at a common source node, as in our model, the source and target node of demands have to be swapped before applying Proposition 4.7 to every source-target pair. If the LP contains node-pair flow variables f_{ij}^k for unprotected commodities with corresponding capacity constraints (3.4g) and dual variables μ_{ij}, everything can be done analogously by replacing the dual variables μ_ℓ by $\mu_\ell + \mu_{ij}$ for protected commodities and by μ_{ij} for unprotected commodities. Recall that the dual diversification variables ρ_ℓ^k are zero for unprotected commodities.

The following corollary tells us how to strengthen metric inequalities generated from a routing LP with only a subset of commodities.

Corollary 4.8. *Let $y^T\mu \geq d(\pi,\rho)$ be a metric inequality generated from a feasible dual solution (π,μ,ρ) with nonzero entries for a subset of commodities only. Then for all other commodities $k \in K$, the right-hand side of this inequality can be augmented by d^k times the length of a shortest path with respect to the link weights μ.*

Notice that this corollary can also be used to strengthen dual rays in case of an infeasible routing LP. The reason is that the linear programming solver will return *any* dual ray leading to a violated metric inequality, but there is no guarantee that this metric inequality will be a most violated one. In fact, some LP solvers (CPLEX in particular) try to make the dual ray as sparse as possible. That is, if the capacities are not sufficient to route one large commodity, the obtained dual ray will have nonzero coefficients only for that commodity and some bottleneck links because this information is sufficient to prove infeasibility of the LP. If the responsible logical links are a bottleneck also for other commodities, computing π_i^k values for those commodities can strengthen the resulting metric inequality.

By assuming $\rho = 0$ in this shortest-path computation, we do not exploit the fact that protected commodities must be routed on at least two disjoint paths. It might be possible to take this fact into account by setting the values ρ_ℓ^k to some positive value before computing the shortest-path distances, thus augmenting the shortest-path distance. In order to have an effect on the dual objective function, however, the values ρ_ℓ^k must be positive only on shortest paths with respect to the combined link weights $\mu_\ell + \rho_\ell^k$. It is not evident how to choose the diversification weights such as to maximize the dual objective function for protected commodities. Of course, this problem can be solved for every commodity $k \in K^p$ and the given vector μ_ℓ by the single-commodity linear program

$$\max \sum_{i\in V} d_i^k - \frac{1}{2}\Big(\sum_{i\in V}\rho_i^k + \sum_{e\in E}\rho_e^k\Big) \tag{4.41a}$$

$$\pi_i^k - \pi_j^k \leq \mu_\ell + \rho_\ell^k \qquad \forall\, i,j \in V,\ i \neq j,\ \ \forall\, \ell \in L_{ij} \tag{4.41b}$$

$$\rho_i^k, \rho_e^k \geq 0,\ \pi_i^k \in \mathbb{R} \tag{4.41c}$$

(using definition (4.31)). With dynamically added commodities, this approach basically decomposes the dual multi-commodity LP into many single-commodity LPs; it is questionable whether this approach saves much computation time. A combinatorial algorithm would be preferable to compute vectors ρ and π that maximize the dual objective function with respect to a given vector μ. Unfortunately, we do not know of any such algorithm.

4.5.6 Implementational issues

Within the branch-and-cut algorithm, the routing LP has to be solved in various places:

- to reject infeasible integer solutions via the incumbent callback,
- to cut off infeasible fractional solutions via the separation callback, and
- to compute a routing for the hardware and link capacities computed by primal heuristics.

In our implementation, all of these cases are covered by the same LP. To test feasibility of a capacity vector, we simply adapt the values y on the right-hand side of the capacity constraints. After changing the capacity vector y, we solve the LP starting from the last basis (which is still dual feasible) using the dual simplex algorithm. If the capacities have not

changed too much since the last call, this is much faster than starting from scratch. The main advantage of this single-LP approach is that in connection with column generation, newly created variables are also available in other contexts and need not be created again in another routing LP.

To make the approach with metric inequalities work in practice, several techniques have to be applied which are almost never published (with the exception of Fortz and Poss [52] who discuss some basic issues):

- Dual presolving must be switched off to tell the MIP solver that not all constraints are present in the master MIP. Otherwise, the solver would eliminate all capacity variables in presolving because the master MIP does not contain any requirements on the capacity.
- Metric inequalities should not only be separated to cut off an optimal solution of the master MIP, but also intermediate integer solutions, and from time to time also fractional LP solutions. For the latter, however, the metric inequality separator should not be called at every node of the branch-and-cut tree because solving the routing LP can be time-consuming. Instead, it should only be called at certain depths in the branch-and-cut tree, and only if no other separator has found violated cutting planes. In particular, cutset inequalities, which are a special case of metric inequalities, should be separated first. Our experiences in this regard confirm the observations made by [27, 52, 76].
- Often, all coefficients in a separated metric inequality are the same, or some coefficients are small integer multiples of others. For numerical reasons, the inequalities should be "numerically beautified" by scaling techniques as discussed in Section 4.4.5.
- It sometimes occurred in our tests that the LP solver declared both the primal and dual LP to be infeasible, which was obviously wrong and caused by numerical instabilities. In such a case, discarding the old basis and resolving the LP from scratch always led to the correct result.

Storing metric inequalities for later use The purpose of the incumbent callback is to reject infeasible integer solutions if not all constraints are modeled in the master MIP. By design, it only admits the answers *feasible* or *infeasible*, and does not allow adding a violated inequality to the master LP. To use the available infeasibility information nevertheless, we always generate a metric inequality when the LP is infeasible. If the LP has been called via the separation callback, the inequality is directly added to the LP; otherwise, it is stored and added to the LP in the next separation call.

4.6 Column generation

If all commodities have to satisfy protection requirements, the number of variables in our formulation PROTECTED (3.4) is $2 \cdot |K| \cdot |L|$. Although this number is polynomial in the input size, it quickly increases with the number of nodes and logical links in the network. As both $|K|$ and $|L|$ are usually in the order of $|V|^2$ (even with several parallel logical links between each pair of nodes), the number of variables is in the order of $|V|^4$.

On large survivable network instances with 50–70 nodes, solving our routing LP with a standard simplex solver is not possible because the LP is too large to fit into memory. Even after our problem-specific presolving, the routing LP has more than 235,000 constraints and

5.5 million variables for the 50-node German network, and more than 485,000 constraints and 2.2 million variables for the 67-node US network. Even with a larger amount of available memory, the times needed to solve a single LP would be extraordinary.

To solve these huge routing LPs nevertheless, we have implemented two different column generation procedures to generate flow variables $f^k_{\ell,ij}$ dynamically when needed. Whereas the first one explicitly tests all candidate variables for negative reduced cost, the second one is based on shortest paths in the logical network with respect to some dual link cost values.

Section 4.6.1 explains the basic idea of column generation. In Sections 4.6.2 and 4.6.3, we describe our two approaches of generating improving variables. Section 4.6.4 discusses how to cope with infeasible LPs. In Section 4.6.5 we describe the generation of an initial set of variables for the routing LP.

For ease of exposition, we assume that all flow variables have lower bound 0, which is the case in our application. Notice that to ensure this, reduced cost strengthening (see [3, 97]) must be switched off in the MIP solver. To concentrate on the parts of the model which are relevant for column generation, everything will be explained using the bottleneck routing LP presented in Section 4.5.2. Provided that the used branch-and-cut framework supports column generation directly at the nodes of the branch-and-cut tree (such as SCIP), all described techniques can also be applied to the full MIP containing both the hardware and routing formulation. Dual presolving of the MIP solver must be turned off in this case. Our implementation works for both approaches; their computational performance is compared in Section 5.2.4.

4.6.1 Basic idea

For LPs with a huge number of variables, the idea of column generation is to start with a small subset of variables and generate new variables only when needed. For a given optimal LP solution with respect to a restricted set of variables, the *pricing problem* is to identify further columns that can improve the LP value, or to discover that no such columns exist. A column can improve the current optimal solution if it has negative reduced cost. If new improving columns are found, the LP is resolved with the new variables. This process is repeated until no more improving variables are found. We apply this technique to generate logical link flow variables $f^k_{\ell,ij}$ dynamically.

It follows from basic LP theory (see Schrijver [124]) that in an optimal LP solution, the reduced cost of an unbounded variable is precisely the slack of the corresponding dual constraint. In other words, a primal variable has negative reduced cost if and only if the corresponding dual constraint is violated by the current dual solution. In this sense, column generation can be seen as separation in the dual LP. Pricing is also similar to separation in the sense that adding a new variable does not necessarily improve the primal objective value, just the same as a new cutting plane does not necessarily improve the dual bound. It is possible that several new variables have to be added to the LP before any of them can be used, for instance if these variables form a source-destination path for a demand.

Column generation differs, however, from separation in one important aspect: separating inequalities for fractional LP solutions is an optional technique to improve the lower bound, but it is not required for the correctness of the algorithm. Column generation, in contrast, is needed to ensure a correct dual bound. If missing flow variables are not correctly identified, feasible and optimal link capacities might be accidently rejected.

4.6.2 Pricing from a fixed list of variables

In a first approach, we implemented a pricer that explicitly checks all candidate flow variables $f^k_{\ell,ij}$ for dual violation. Since these are at most $2 \cdot |K| \cdot |L|$ variables, this pricing algorithm is polynomial. First assume there are no upper variable bounds. For notational convenience, assume also that $K = K^p$, i. e., all commodities are protected. The case of unprotected commodities is discussed at the end of this section.

Finding improving variables

In every pricing step, the algorithm looks for each candidate flow variable $f^k_{\ell,ij}$ whether the current dual solution (π, μ, ρ) violates the dual constraint of the variable,

$$\pi^k_i - \pi^k_j \leq \mu_\ell + \rho^k_\ell, \tag{4.42}$$

with

$$\rho^k_\ell := \frac{1}{2}(\rho^k_i + \rho^k_j) + \sum_{v \in V \colon \ell \in L_v} \rho^k_v + \sum_{e \in \ell} \rho^k_e.$$

For backward variables $f^k_{\ell,ji}$, the left-hand side of (4.42) changes to $\pi^k_j - \pi^k_i$; all other terms are symmetric in the direction of the logical link.

An intuitive interpretation of the dual variables μ_ℓ and ρ^k_ℓ is the following. The weight μ_ℓ of a logical link $\ell \in L$ can be nonzero only if the capacity constraint of that link is tight with respect to the current LP solution. Similarly, the weight ρ^k_ℓ can be nonzero only if any of the diversification constraints concerning ℓ are tight. The first term in the definition of ρ^k_ℓ deals with the end-nodes of ℓ, the second term with the inner nodes of ℓ, and the third term with the physical links traversed by ℓ. Altogether, a positive link weight $\mu_\ell + \rho^k_\ell$ means that no more flow can be routed on logical link $\ell \in L$ because at least one capacity or diversification constraint concerning ℓ is at its bound.

Variable bounds

In the presence of variable bounds, the dual variables of the bound constraints must be taken into account, similarly to the construction of metric inequalities, see Section 4.5.2. Assume that a flow variable $f^k_{\ell,ij}$ is bounded by $l^k_{\ell,ij} \leq f^k_{\ell,ij} \leq u^k_{\ell,ij}$. By adding the bounds to the routing LP (4.30) as explicit inequalities with associated dual variables $\underline{\gamma}^k_{\ell,ij} \geq 0$ for the lower bound and $\overline{\gamma}^k_{\ell,ij} \leq 0$ for the upper bound, the dual constraint (4.42) changes to

$$\pi^k_i - \pi^k_j \leq \mu_\ell + \rho^k_\ell - \underline{\gamma}^k_{\ell,ij} - \overline{\gamma}^k_{\ell,ij}. \tag{4.43}$$

The pricing algorithm only looks at the dual constraint of variables that are not yet in the LP. Since their bound constraints are not yet in the LP either, we can assume the corresponding dual variables $\underline{\gamma}^k_{\ell,ij}$ and $\overline{\gamma}^k_{\ell,ij}$ to be 0. That is, this pricer is exact also in the presence of variable bounds. As long as there exists a variable with violated dual constraint, the pricer will find it, and the global dual bound is valid.

Unprotected commodities

If unprotected commodities are included in the LP, the primal LP has an additional node-pair capacity constraint (4.37) with associated dual variable $\mu_{ij} \geq 0$, as described for metric inequalities in Section (4.5.2). Consequently, the dual constraint (4.42) of a flow variable $f^k_{\ell,ij}$ for a protected commodity $k \in K^p$ changes to

$$\pi^k_i - \pi^k_j \leq \mu_\ell + \mu_{ij} + \rho^k_\ell. \tag{4.44}$$

Otherwise, everything remains unchanged.

Of course, a similar pricing algorithm approach can be adopted for unprotected commodities $k \in K^u$ with node-pair variables f^k_{ij}. Such a flow variable is only included in the capacity constraint (4.37) for node pair i, j. Hence, the corresponding dual constraint reads as $\pi^k_i - \pi^k_j \leq \mu_{ij}$. Everything else in the algorithm remains unchanged. We did not implement column generation for unprotected commodities because the number of node-pair flow variables f^k_{ij} is much smaller than the number of logical link flow variables $f^k_{\ell,ij}$.

Implementational issues

The pricer iterates through a list of all candidate variables. Just storing these variables is not very memory-intensive; the main part of the memory is consumed by the coefficients in the LP columns. For efficiency reasons, we nevertheless store only those variables that have passed all preprocessors, like for instance hop limits, a core network filter, or the LP probing (see Section 4.2).

In most of the calls to the pricer, only very few dual variables are nonzero. Also notice that the right-hand side of the dual constraint (4.42) is always nonnegative. Consequently, the pricing procedure can significantly be accelerated by iterating only over the nonzero dual variables and by computing the right-hand side only if $\pi^k_i - \pi^k_j > 0$.

The described straightforward approach already allows us to compute valid dual bounds for the large network design instances, which was not possible before. Nevertheless, the pricer usually finds a large number of potentially improving variables. Adding them all quickly increases the size and the solution time of the routing LP. On the other hand, adding a too small number of variables will increase the number of pricing iterations. We have tried several ways of restricting the number of added variables:

- adding only one variable with minimum reduced cost per commodity and iteration,
- adding at most k variables per commodity and iteration,
- considering the commodities in decreasing order of demand value d^k, and stopping the pricing iteration if at least one variable has already been found and a certain percentage of the total demand value has already been covered in the current pricing iteration,
- stopping pricing after a certain number of pricing iterations or below a certain depth in the branch-and-cut tree. Of course, this heuristic approach invalidates the dual bound because not all necessary variables might have been constructed, but it may be useful to solve more branch-and-cut nodes when focusing on feasible solutions.

After many tests, we found that adding several variables per commodity and iteration but limiting them to a small number (≈ 6) is a viable compromise between the number of pricing

iterations and the size of the LP. Furthermore, we stop a pricing iteration after 60% of the total demand value in the network has been covered and at least one variable has been found.

The fact that not all variables are included in the LP from the beginning has consequences for several other parts of the overall algorithm. For instance, only cutting planes without flow variables can be used (in particular, no flow-cutset inequalities). Furthermore, solutions procuced by combinatorial heuristics that are called during the branch-and-cut process may use variables that are not yet included in the LP. We add those variables to the LP before testing the solution for feasibility rather than generating them using a pricing algorithm.

4.6.3 Path-based pricing

With the enumerative pricing algorithm, we observed that flow variables are often generated in a breadth-first search manner starting from the source node of the commodity. These variables can only be used in the LP solution after adding a whole end-to-end routing path. This behavior leads to many generated variables and to stalling effects, i.e., the loop of generating new variables and resolving the LP is repeated over and over again without changing the LP objective value significantly. Sometimes this takes several hours and dominates the whole algorithm. In order to better exploit the network structure, we thus implemented another pricing algorithm that generates end-to-end routing paths in the logical layer.

For notational convenience, assume first that there are no variable bounds on flow variables, and that $K = K^p$. The other cases are dealt with later in this section.

Basic idea

The basic idea of the path-based pricer is to look at the aggregated reduced cost of an end-to-end routing path. Let $k \in K^p$ be a point-to-point commodity and p a routing path in the logical layer from s^k to t^k. For notational convenience, assume that all logical links in the path are oriented from s^k to t^k, such that the flow on the path contains only forward flow variables $f^k_{\ell,ij}$. Aggregating the dual constraints (4.42) for all logical links in p leads to the dual path constraint

$$\pi^k_{s^k} - \pi^k_{t^k} \leq \sum_{\ell \in p} (\mu_\ell + \rho^k_\ell). \tag{4.45}$$

If a routing path p violates this constraint, it must contain at least one flow variable $f^k_{\ell,ij}$ with negative reduced cost, i.e., violating its dual constraint (4.42). It may also contain variables with positive reduced cost, which may or may not be included in the LP yet. We thus test whether any of the above constraints is violated; if this is the case for some path p for commodity k, we add all those flow variables $f^k_{\ell,ij}$ with $\ell \in p$ to the LP which are not yet present. In contrast to the straightforward pricer described in Section 4.6.2 which generates single logical link flow variables, the path-based pricer thus generates whole routing paths which can usually directly be used for routing flow in the following LP iteration.

To find an improving routing path for a given commodity $k \in K^p$, we first compute the left-hand side $c_k := \pi^k_{s^k} - \pi^k_{t^k}$ of (4.45), which depends only on the commodity. If this value is non-positive then no path for this commodity can violate its dual constraint because the right-hand side of (4.45) is always nonnegative. Otherwise, we search for a shortest path from s^k to t^k in the logical graph with respect to the link weights $\mu_\ell + \rho^k_\ell$. If its length is smaller than c_k then we add the corresponding variables to the LP if they do not exist yet; otherwise no path for this commodity violates its dual constraint.

Intuitively, the nonzero dual variables μ_ℓ and ρ_ℓ^k mark links in the network where the capacity constraints or diversification constraints (or both) are tight with respect to the current LP flow. The pricing procedure tends to find paths on logical links with many zero dual variables, indicating that the corresponding constraints still have slack. Again, this pricer works in polynomial time because we solve at most one shortest path problem per demand and pricing iteration.

Correctness

If a dual solution satisfies the dual link constraints (4.42) for all single logical links and commodities, then it obviously also satisfies the dual path constraints (4.45) for all commodities. It is not difficult to see, however, that the reverse is not necessarily true. This raises the question whether the path-based pricing procedure is sufficient to find all necessary variables. Fortunately, this is the case. Intuitively, this result is not surprising because the described pricing procedure simulates the behavior of a classical pricing algorithm for a path-flow formulation. Nevertheless, the result is not obvious and deserves to be discussed.

Proposition 4.9. *The path-based pricing algorithm is exact, i. e., if flow variables with negative reduced cost exist, the algorithm will find one.*

Proof. Let $y = (\pi, \mu, \rho)$ be a feasible dual solution corresponding to the restricted primal LP (4.30) with a subset of variables. Assume that y satisfies all dual path constraints (4.45) but violates some dual link constraint (4.42).

The idea of the proof is to construct another dual solution $\tilde{y} = (\tilde{\pi}, \mu, \rho)$ which satisfies *all* dual constraints and has at least the same objective value as y. To this end, we replace the values π_i^k by new values $\tilde{\pi}_i^k$, which are defined as the shortest-path distance from i to t^k with respect to the logical link weights $\mu_\ell + \rho_\ell^k$. By construction, the new dual vector $\tilde{y}$ satisfies all dual constraints (4.42). Furthermore, as shown in Section 4.5.5, this choice of $\tilde{\pi}$ is optimal for given vectors μ and ρ. In particular, if y is feasible and optimal for the relaxed dual LP then $\tilde{y}$ is feasible and optimal for the complete dual LP. This shows that the restricted primal LP is optimally solved with respect to all variables. Adding further flow variables would only change the dual solution without improving the objective value. Summarizing, this shows that the path-based pricing algorithm is exact, and dual bounds derived from it are valid. □

Pricing graph

If a logical link $\ell_0 \in L_{ij}$ appears in a shortest path, then all other logical links between nodes i and j must be at least as long, i.e.,

$$\mu_{\ell_0} + \rho_{\ell_0}^k \leq \mu_\ell + \rho_\ell^k$$

for all all $\ell \in L_{ij}$. To find shortest paths for a commodity $k \in K^p$, we construct a graph with the same node set V and an undirected edge between every pair of nodes i, j with $L_{ij} \neq \emptyset$, as illustrated in Figure 4.8. With the edge $\{i, j\}$, we associate a weight z_{ij}^k defined by

$$z_{ij}^k := \min_{\ell \in L_{ij}} (\mu_\ell + \rho_\ell^k). \tag{4.46}$$

Notice that the weights z_{ij}^k are symmetric in i and j. As they are nonnegative, we can find a shortest path from s^k to t^k in the pricing graph using Dijkstra's algorithm and substitute

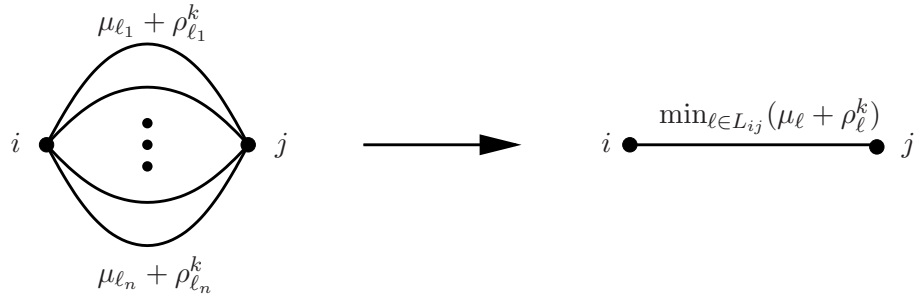


Figure 4.8: Pricing graph: replace parallel logical links by one link with minimum weight.

every edge in this path by a logical link $\ell \in L_{ij}$ for which the minimum in (4.46) was attained. Often, the minimum is zero and attained at several logical links; in this case we prefer those consisting of a small number of physical links to longer ones. Together with the direction from s^k to t^k, the resulting path on logical links translates into a set of logical link flow variables $f^k_{\ell,ij}$, $f^k_{\ell,ji}$ that can be used to model the corresponding path flow. These are added to the LP if not yet existing.

The pricing procedure is a central step of the overall algorithm which may be called several hundreds or thousands of times for every commodity. It is thus vital to use efficient data structures and a fast shortest-path algorithm. We employ a variant of the Dijkstra algorithm that searches a shortest path from both the source and the target node in order to keep the explored neighborhoods small.

Variable bounds

In the presence of variable bounds $l^k_{\ell,ij} \le f^k_{\ell,ij} \le u^k_{\ell,ij}$ on already existing variables, the corresponding dual weights $\underline{\gamma}^k_{\ell,ij} \ge 0$ and $\overline{\gamma}^k_{\ell,ij} \le 0$ have to be taken into account. More precisely, the shortest-path weights z^k_{ij} have to be computed using the dual constraints with bounds (4.43). Notice that these weights are no longer symmetric in i and j. Consequently, every single edge in the pricing graph must be replaced by two anti-parallel edges, each of them corresponding to a set of parallel flow variables with the same direction. According to (4.43), the weight of the directed edge from i to j is computed as

$$z^k_{ij} := \min_{\ell \in L_{ij}} (\mu_\ell + \rho^k_\ell - \underline{\gamma}^k_{\ell,ij} - \overline{\gamma}^k_{\ell,ij}). \tag{4.47}$$

Recall that the values $\underline{\gamma}^k_{\ell,ij}$ and $\overline{\gamma}^k_{\ell,ij}$ are 0 if the flow variable $f^k_{\ell,ij}$ does not exist yet. For already existing variables, the non-positive dual values $\overline{\gamma}^k_{\ell,ij}$ for the upper bound of existing variables can be obtained via the reduced cost vector of the current LP solution, as described in Section 4.5.2. Recall that we assume $l^k_{\ell,ij} = 0$. If a variable is at its lower bound 0, we can construct another feasible dual solution with the same objective value by changing a positive $\underline{\gamma}^k_{\ell,ij}$ to 0, see formula (4.35). In this way, the link weights z^k_{ij} remain nonnegative, and Dijkstra's shortest-path algorithm can still be applied. In our tests, positive values of $\underline{\gamma}^k_{\ell,ij}$ occurred quite often for flow variables that were at their lower bound 0 in the LP solution, so this step is really necessary.

Some preprocessors and heuristic approaches, like hop limits or core network heuristics, forbid certain variables $f^k_{\ell,ij}$. As the pricing weights z^k_{ij} are computed for each commodity separately, such restrictions can easily be taken into account by ignoring the corresponding logical links in the computation of the minimum (4.47).

Unprotected commodities

If node-pair flow variables f^k_{ij} and the corresponding capacity constraints are included in the routing LP, the principle remains the same. The only change is that in all definitions of dual weights in this section, μ_ℓ must be replaced by $\mu_\ell + \mu_{ij}$ for logical links $\ell \in L_{ij}$, where $\mu_{ij} \geq 0$ is the dual weight of the node-pair capacity constraint (4.37).

Generating node-pair flow variables f^k_{ij} for unprotected commodities $k \in K^u$ works in a very similar way. Recall that an unprotected commodity is aggregated at the source node and may have several target nodes. Hence, a routing path p on node pairs $\{i,j\}$ must be found from the source s^k to every target node t^k of every commodity $k \in K^u$. By aggregating the dual constraints $\pi^k_i - \pi^k_j \leq \mu_{ij}$ (see Section 4.6.2) along a path p, the corresponding dual path constraint reads as

$$\pi^k_{s^k} - \pi^k_{t^k} \leq \sum_{\{i,j\} \in p} \mu_{ij}.$$

Hence, the pricing problem consists of finding a shortest path from s^k to t^k in the pricing graph with respect to the link weights $z^k_{ij} := \mu_{ij}$. As the physical layer does not matter at all for the flow variables f^k_{ij}, this is exactly the usual pricing approach for a standard multi-commodity path-flow formulation at the logical layer, see [135], for instance.

Notice that the link weights are independent of the commodity. It is thus possible to compute shortest paths between all pairs of nodes using an all-pairs shortest-paths algorithm. Alternatively, shortest paths from s^k to every target node of k can be computed in a single step by constructing a shortest path tree rooted in s^k.

4.6.4 Pricing from an infeasible LP

Basic idea Column generation is also possible if the LP is infeasible. First, assume that no variable bounds are present. Recall from Section 4.5.3 that if the primal LP

$$\min_{x \geq 0} \{c^T x \mid Ax \geq b\}$$

is infeasible and the dual LP

$$\max_{y \geq 0} \{b^T y \mid y^T A \leq c^T\}$$

has a feasible solution, then there exists a dual ray $r \geq 0$ such that

$$r^T A \leq 0 \quad \text{and} \quad b^T r > 0.$$

If the primal LP contains only part of the variables, i.e., the dual LP contains only part of the constraints, the task of the pricing algorithm is to test whether the dual ray r satisfies the constraints

$$r^T A_j \leq 0 \tag{4.48}$$

for all primal columns j. This can be done in the same way as with the dual solution of a feasible LP, except that the left-hand side must be compared to 0 instead of c_j. In our case, this does not make any difference because the priced variables have objective value 0 anyway.

If all conditions of type (4.48) are satisfied then the dual ray serves as a certificate for the infeasibility of the primal system. A violated dual constraint, on the other hand, will cut off the dual ray. Unless the primal LP is infeasible even with all possible variables, iterating this process will eventually make the dual LP bounded, and thus the primal LP feasible. Once the primal LP is feasible, pricing can be done as described in the previous sections. Adding further variables to the LP will never make it infeasible again because we assumed all lower variable bounds to be zero.

Performance in practice These considerations have important consequences for the algorithm employed for solving the LPs. As long as we cannot be sure that the routing LP is feasible, we have to turn off LP presolving and use the dual simplex algorithm in order to obtain a dual ray in case of infeasibility. Once we know that the LP is feasible, we can use the primal simplex algorithm, which allows to start from the previous basis after new columns have been added.

From a practical point of view, column generation based on feasible LPs can be faster by orders of magnitude. The reason is that dual rays are typically very sparse and have non-zero entries only for one commodity. As a result, new variables are only generated for that commodity. Consequently, testing a capacity vector for violation often requires a huge number of LP solving and pricing iterations if the LPs are infeasible. Intuitively, with this choice of sparse dual rays, the LP solver first tries to make the routing feasible and to satisfy the demand and diversification constraints before it starts considering the capacity constraints.

As a consequence, the choice of whether to separate metric inequalities using a bottleneck variable or using a pure feasibility test heavily affects the performance of the LP solving and pricing loop. With the bottleneck approach and a good set of initial variables, feasibility of the routing LP can usually be obtained after a few pricing iterations. With a pure feasibility test, however, the routing LP is infeasible most of the time, with the above mentioned drawbacks. This trade-off is investigated computationally in Section 5.2.5.

Correctness The correctness proof for the path-based pricer from Section 4.6.3 can easily be extended to infeasible LPs. It suffices to notice that the new dual vector $\tilde{y}$ defines a valid dual ray whose dual objective value is as least as large as that of y. In particular, if y can be used to prove infeasibility of the primal LP and to construct a violated metric inequality then $\tilde{y}$ can be used for this purpose as well.

If the flow variables are bounded, the principle remains the same as for a feasible LP. The dual values of the bound constraints, however, are not part of the dual ray returned by the LP solver and have to be computed via the reduced cost of existing variables, as described in Section 4.5.3.

4.6.5 Choosing an initial set of variables

The choice of initial variables can be crucial for the performance of a column generation algorithm. Such an approach may work very well if the initial set of variables is already close to an optimal one, such that few pricing iterations are sufficient to find the missing variables.

If, in contrast, this set is far away from any optimal routing, a column generation approach is likely to fail in practice.

In our case, the initial set of variables should be small enough to solve the routing LPs quickly, but it should also allow to compute feasible LP routings after a few pricing iterations. These feasible routings can also be used as a basis for LP-based construction heuristics, as explained in Section 4.3. That is, a good initialization of the routing LP can help improving both the primal and dual bound. We have tested several different ways of computing such a subset of initial variables:

1. all flow variables that can be used with a logical hop limit of 1 or 2,
2. all flow variables on logical links consisting of at most k physical hops,
3. all flow variables corresponding to the routing of one or more already known solutions,
4. all flow variables corresponding to two physically disjoint logical paths per demand.

The latter set of flow variables is computed using the routing construction heuristic described in Section 4.3.2 (without link capacities c_ℓ). We observed that using an existing solution to initialize the routing and improving it using column generation was a quite successful strategy, as it allows to compute feasible solutions from the beginning. In this way, it is also possible to incorporate solutions from previous computations or from external sources into a new run of the branch-and-cut-and-price algorithm.

It turned out that some commodity-dependent initialization is needed which computes end-to-end routing paths. Otherwise, lots of variables have to be generated during the first LP iterations, often based on a dual ray, which takes too much time. By default, we use the FindFeasibleRouting heuristic together with all direct logical links between the end-nodes of a commodity.

Chapter 5

Computational results

In this chapter, we investigate the effectiveness of our algorithmic techniques described in the previous chapter. Rather than to give a statistical analysis based on a large number of network instances, our goal is to study the impact of these techniques on few realistic test networks with different characteristics. We investigate the impact of the network and demand structure on the performance of our solution methods, and show how to use them for computational studies.

In Section 5.1, we describe our test framework and the used network instances. We summarize key figures of the networks and describe the background of the network together with the structure of capacities, cost, and demands. Section 5.2 is the central part of this chapter. In this section, we investigate different strategies for presolving, primal heuristics, cutting planes, column generation, and metric inequalities. In Section 5.3, we study the impact of hop limit and core network restrictions on the network cost for large networks with survivability constraints. Eventually, Section 5.4 summarizes our findings from this chapter.

5.1 Test instances and settings

Summary of the test instances Table 5.1 summarizes important characteristics of our test networks. The first column denotes the name of the network. The second to fifth column show the number of nodes, physical and logical links, and demands. The last three columns contain the density of the physical, logical, and demand graph, respectively. In this context, the density of a graph is defined as the number of its links divided by $|V|\,(|V|-1)/2$, that is, the number of links in relation to the number of links in a complete graph. We consider all instances in two versions: with protection against physical failures or without. The instances with protection are postfixed with 'p' in the sequel (like *r15p*), the unprotected versions will be postfixed with 'u' (like *r15u*).

The topologies of those networks where geographical data is available (all but r15) are depicted in Figures 5.1–5.5 on the following pages. In these figures, the area of a node is proportional to its emanating demand. In the logical topologies, every link represents all parallel logical links between the corresponding end-nodes. In particular, the logical layer may be much denser than apparent from the picture.

As can be seen from Table 5.1 and the topology pictures, the network instances have very different structure. As all networks have some realistic background, the density of the physical network decreases with the number of nodes, because only geographically close nodes are

inst	$\|V\|$	$\|E\|$	$\|L\|$	$\|R\|$	density E	density L	density R
p12	12	18	2441	66	0.27	36.98	1.00
r15	15	16	184	78	0.15	1.75	0.74
g17	17	26	564	121	0.19	4.14	0.88
eu28	28	41	756	378	0.10	2.00	1.00
g50	50	89	3156	1225	0.07	2.57	1.00
us67	67	87	690	2211	0.03	0.31	1.00

Table 5.1: Characteristics of the test instances: number of nodes, physical and logical links, and demands, and the density of the physical and logical network and the demand graph, in relation to a complete graph on $|V|$ nodes. The instances are structurally different and have dense logical link and demand graphs.

physically connected. Nevertheless, the average physical node degree (defined as $2\,|E|\,/\,|V|$) is always between 2 and 4, which is a typical range for practical transport networks.

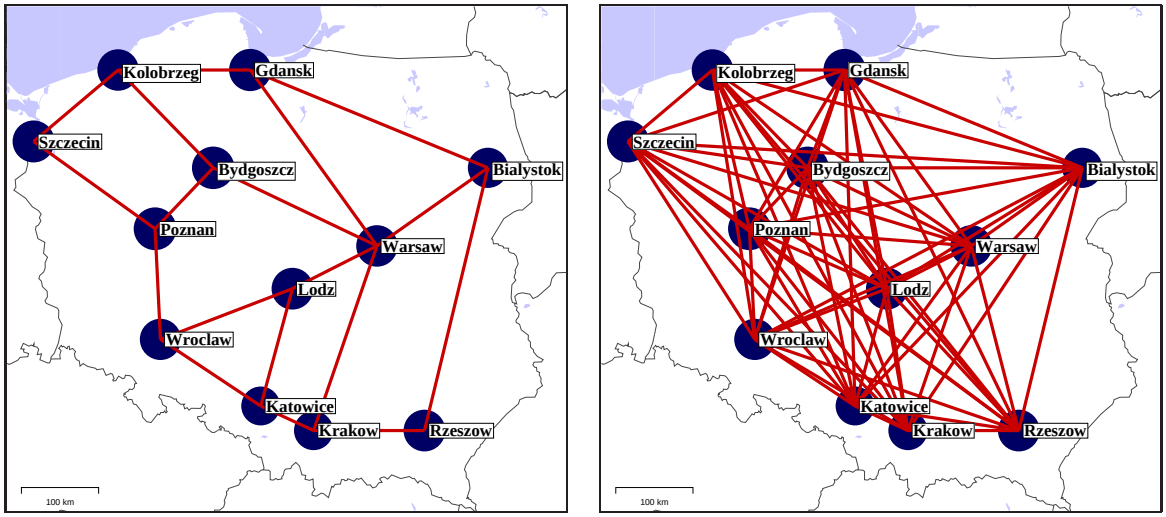
The demand matrix of four of the six networks is complete; in particular, this includes the large networks eu28, g50, and us67. The demand graph of r15 and g17 contains 74% and 88% of all possible demand connections, respectively.

The density of the logical networks varies widely. The p12 logical network has nearly 37 times the number of logical links of a simple complete graph. Also the logical networks of r15, g17, eu28, and g50 networks are complete with parallel logical links. Even though the logical us67 network has parallel logical links and a density of 0.31, it is much sparser than the other logical topologies because of the huge distances between major cities in the United States. The available equipment only allows lightpaths of limited geographical length. An important consequence is that in the us67 network, many demands have to be routed on at least five or six logical hops, and the number of possible routing paths is limited. In the other networks, every demand can be routed from its source to its destination in many different ways using one or two logical hops only.

Background of the instances We will now describe the background of our network instances in more detail, together with the capacity, cost, and demand structures.

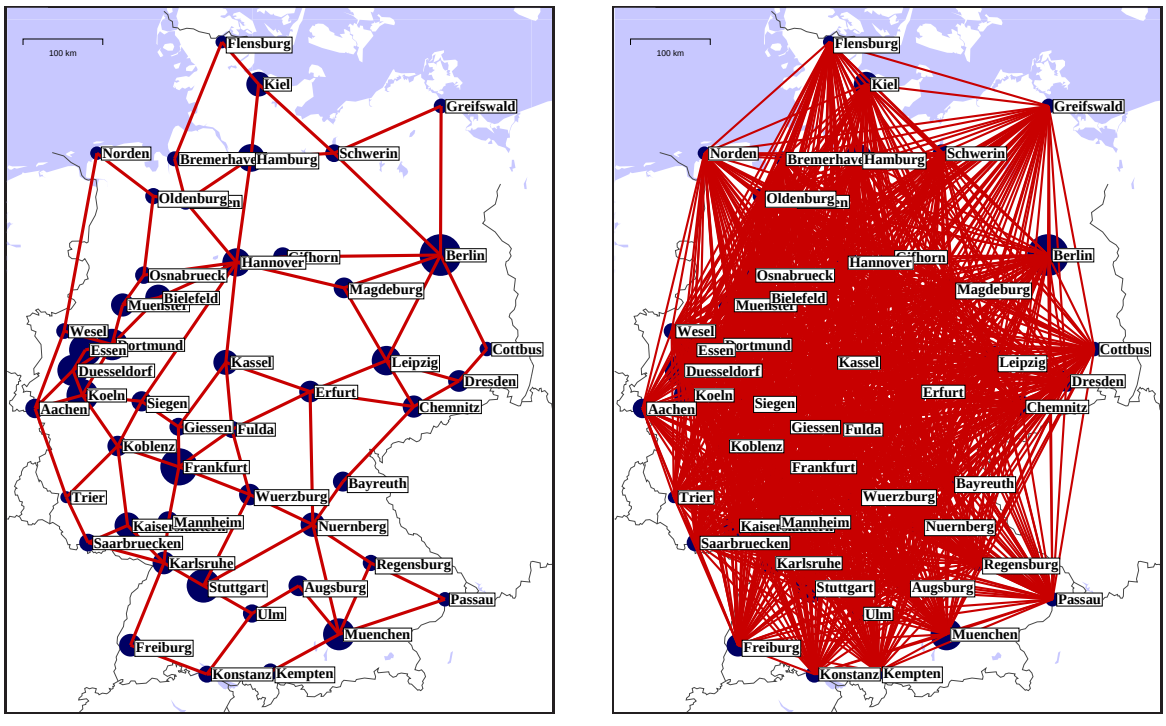
The p12 network, depicted in Figure 5.1, has been derived from the Polska network in Sndlib [105]. It represents the SDH transport network of the Polish telecom from the early 1990s. The physical layer and the demands have been taken from Sndlib; this instance has no node capacities. For each pair of nodes $i, j \in V$, the set L_{ij} of admissible logical links between these nodes is defined by the k shortest physical paths with respect to the geographical length, where k is the minimum of 50 and the number of existing paths between i and j. As shown in Table 5.1, the average number of parallel logical links per node-pair is close to 37; there exist pairs of nodes with 50 parallel logical links between them. STM-1 and STM-4 capacities can be installed on every logical link, and at most 4 capacity units may share a physical link.

The networks r15, g17, g50, and us67 are realistic instances representing optical fiber networks. They have been provided by Nokia-Siemens Networks. The logical links correspond to routes in the optical network on which transparent lightpaths with different bandwidths (2.5, 10, or 40 Gbit/s) can be installed in integer multiples. The set of admissible bandwidths



(a) p12 physical network (b) p12 logical network

Figure 5.1: Topology of the physical and logical p12 network.

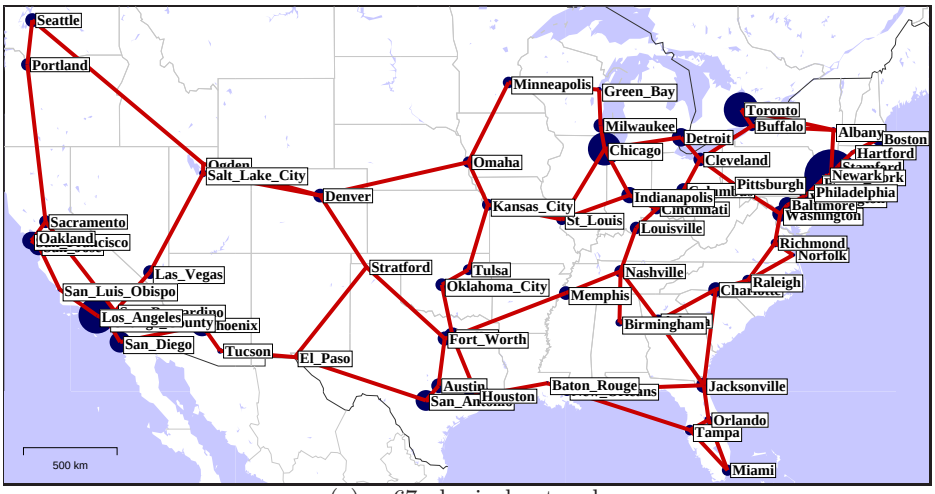


(a) g50 physical network (b) g50 logical network

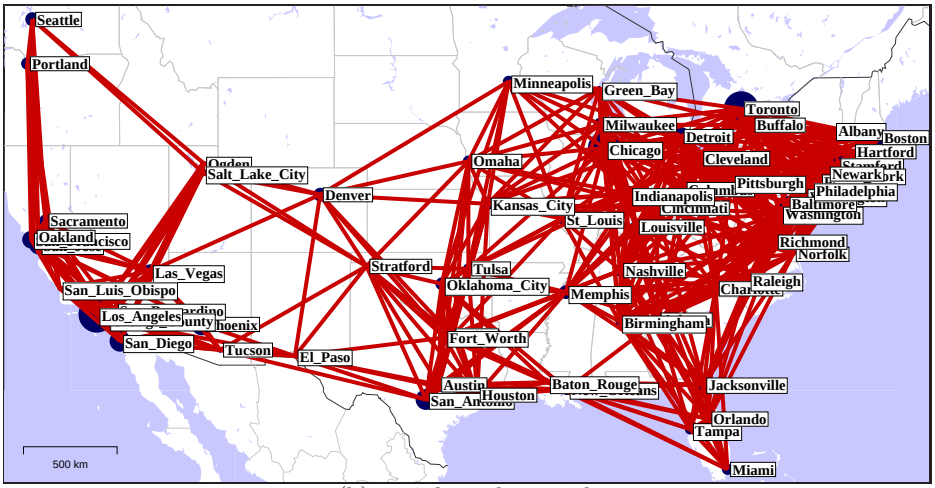
Figure 5.2: Topology of the physical and logical g50 network.

depends on the length of a specific logical link. Logical link cost is incurred by the transponders at the end-nodes of each lightpath. It depends on the installed capacity and on the geographical length of the link, because long-range devices at the end-nodes of a lightpath are more expensive than short-range ones. Every fiber on a physical link supports up to 80 wavelength channels and incurs a length-dependent cost. In the r15 network, physical link cost has been set to zero to model an existing fiber network. At the nodes, one out of several cross-connects with different switching capacity and cost has to be selected.

The r15 instance consists of a physical ring network with a chord. It models a 14-node regional US subnetwork with connections to the remaining network, where the latter is represented by the last node. The logical layer consists of two physical paths between every pair of demand end-nodes within the subnetwork, plus a few logical links to the remaining network.



(a) us67 physical network

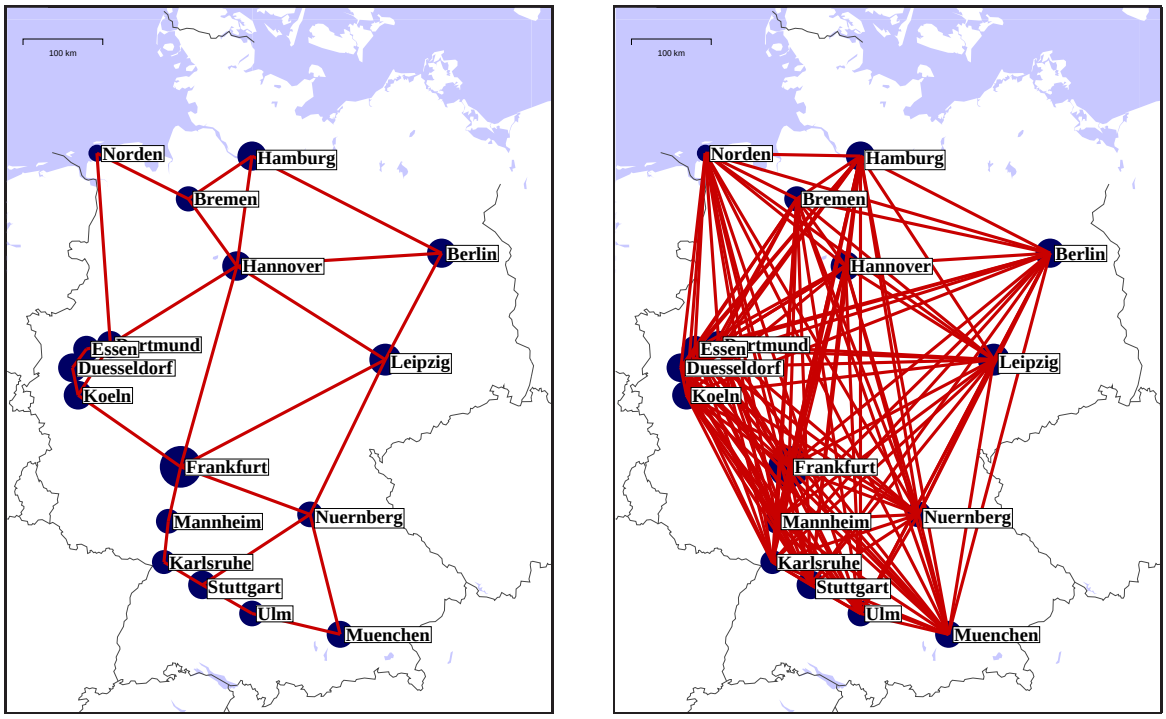


(b) us67 logical network

Figure 5.3: Topology of the physical and logical us67 network.

The g50 and us67 physical networks have been defined as reference data in the German research project EIBONE [1, 30], in which network operators, equipment vendors, universities, and research institutes were involved. These instances represent a German and a US backbone

network, respectively; their physical layers are shown in Figures 5.2(a) and Figures 5.3(a). In the g50 network, the logical layer is complete and has 2–3 parallel logical links between nearly every pair of nodes. Recall that the routing of a demand is not restricted to these direct logical links between its end-nodes; by using a sequence of logical hops, many more routing paths are possible. In the us67 network, logical links are defined in a similar way but only between nodes which are geographically close to each other for technical reasons. Figures 5.2(b) and 5.3(b) show the logical topologies of g50 and us67, where every link in the picture represents 1–3 parallel logical links.



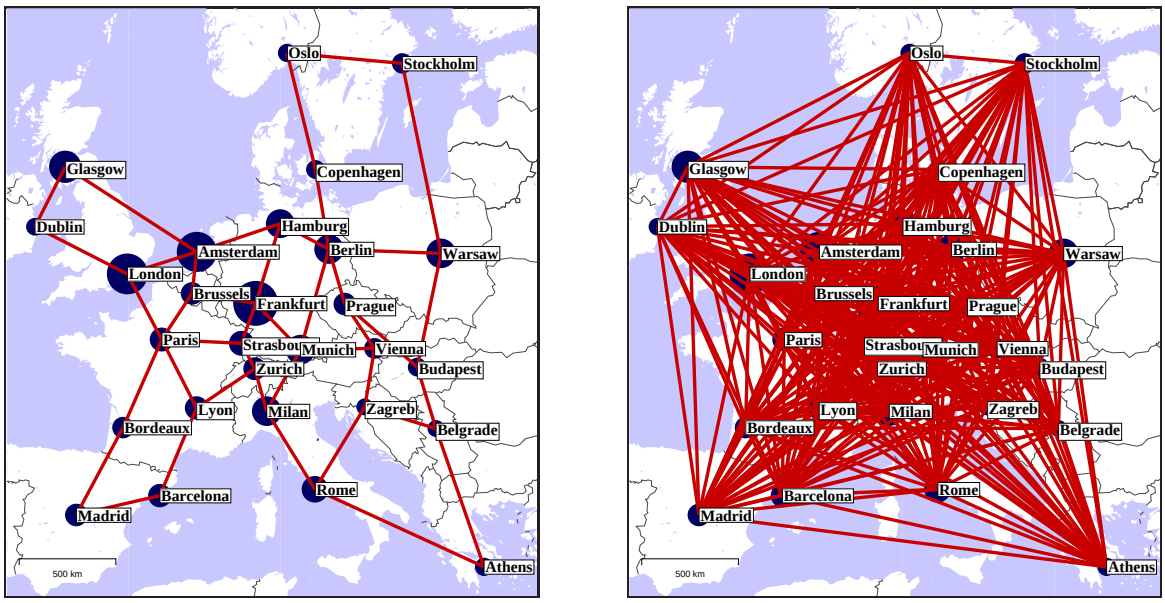
(a) g17 physical network

(b) g17 logical network

Figure 5.4: Topology of the physical and logical g17 network.

The g17 network, which is shown in Figure 5.4, is a subnetwork of g50. It consists of 17 major cities in Germany with connections between them. This network has been defined as reference data in the European NOBEL project [2], is included in SNDLIB, and has been previously used in publications by various authors. The logical layer is defined by 4–5 short physical paths between every pair of nodes.

In all our instances from Nokia-Siemens Networks, demands are based on the Dwivedi-Wagner population model [51]. This traffic model, which has been designed for transport networks, considers aggregated traffic of different classes, namely voice, business data, and IP traffic generated by individual users. These traffic classes show different characteristics; for instance, phone traffic tends to be more local than IP traffic. The demand between two cities is estimated in a different way for each of these classes depending on the number of inhabitants, employees, households, and the distance between the cities. Finally, the demands are scaled such as to meet a specific total demand volume in the network which fits with actual traffic



(a) eu28 physical network

(b) eu28 logical network

Figure 5.5: Topology of the physical and logical eu28 network.

measurements or forecasts, depending on the goal of the study. This approach leads to very different demand structures for different networks, as can be seen from the size of the nodes in the topology figures. In the g50 network, all demands are in a similar order of magnitude, whereas in the us67 network, the four predominant nodes New York, Los Angeles, Toronto, and Chicago generate 38% of the total demand in the network.

The eu28 network is a European network which has served as reference data in the European NOBEL project [2]. The physical topology, which is depicted in Figure 5.5, as well as the demands have been taken from SNDLIB, as well as the capacity and cost structures. As the SNDLIB data does not include node data, we only consider logical and physical link capacities with associated cost in this network. The logical layer consists of two logical links between every pair of nodes, on which wavelength channels of 2.5 Gbit/s can be installed. Every physical fiber incurs a length-dependent cost and supports up to 40 wavelength channels.

Test settings All our tests have been performed on an x86-64 Linux machine with a 3 GHz CPU and 8 GB of memory. We have used SCIP-1.1.0.1 as a branch-and-cut-and-price framework with CPLEX-11.0 as the underlying LP solver. Any time limits imposed in our tests are mentioned in the description of the corresponding results. Unless otherwise stated or required for the correctness of our algorithm, we used the default settings of SCIP.

5.2 Impact of the algorithmic ingredients

In the following, we will investigate the effect of the different algorithmic ingredients introduced in Chapter 4: presolving, primal heuristics, cutting planes, column generation, and metric inequalities.

5.2.1 Presolving

In this section, we evaluate the problem-specific presolving steps described in Section 4.2. The basic presolving described in Section 4.2.1 applies to all network design instances based on the investigated models. Its purpose is to strengthen bounds on integral capacity or node hardware variables, or to remove some of them entirely. In contrast, the LP-based probing from Section 4.2.2 acts on continuous flow variables for protected demands. We will first study the impact of the LP-based probing technique.

Removing flow variables of protected commodities

Table 5.2 demonstrates the impact of the LP-based probing on the number of flow variables. For each protected network instance, the second and third column give the number of flow variables before and after our LP-based probing. The fourth column states the percentage of the flow variables which can be removed from the formulation, i. e., *(before - after)/before*. The last column gives the fraction of flow variables for which a positive bound less than $d^k/2$ is computed in the full probing version.

inst	flowvars before	flowvars after	flowvars removed (%)	flowvars bounded (%)
p12p	56595	44516	21.3	43.7
r15p	10040	5526	45.0	0.0
g17p	76756	47089	38.7	21.0
eu28p	421288	331236	9.6	28.3
g50p	6161426	5567760	21.4	21.0
us67p	2729358	2234001	18.1	10.0

Table 5.2: Our LP-based probing removes or bounds many flow variables. The first two columns show the number of flow variables before and after probing. The last two columns show the fraction of all flow variables which could be removed or upper bounded by some value smaller than $d^k/2$.

The table shows that the LP-based probing technique removes between 9.6% and 45% of all flow variables from the beginning without changing the set of feasible and optimal solutions. The largest reduction can be achieved on the physical ring network r15. With the exception of r15p, the upper bounds of 10% to 43% of all original flow variables can be strengthened to some positive value smaller than $d^k/2$.

Table 5.3 compares the two probing variants from Section 4.2.2 for each protected network instance. Recall that some linear programs can be skipped if the maximized variable has been at its upper bound in a previous solution (with the full probing version which computes variable bounds) or has been nonzero before (with the reduced version which only tests whether a variable can be positive or not). The second to fourth column of Table 5.3 show the number of solved linear programs, the total number of LP iterations, and the time in seconds (rounded to integers) needed by the full probing procedure. The next three columns show the corresponding values for the reduced version. Eventually, the last three columns represent the relative change in percent achieved by using the reduced probing version instead of the full one. If f denotes the value for the full probing variant and r the value for the reduced variant, the relative change is defined as $100 \cdot (r - f)/f$. For example, the last column shows that on p12p, the reduced probing version requires 41% less time than the full version.

	full probing			reduced probing			relative change (%)		
inst	LPs	LPiter	time	LPs	LPiter	time	LPs	LPiter	time
p12p	53800	369998	39	48178	69912	23	−10.4	−81.1	−41.0
r15p	8441	27369	1	8568	15949	1	+1.5	−41.7	−8.0
g17p	70770	475523	20	68573	93427	11	−3.1	−80.4	−45.0
eu28p	384945	6375406	342	373124	382059	84	−3.1	−94.0	−75.4
g50p	5916651	279989344	66239	5693503	4179785	4991	−3.8	−98.5	−92.5
us67p	2262653	36263133	1917	2381108	1598319	635	+5.2	−95.6	−66.9

Table 5.3: The reduced probing version is much faster than the full one with bounds. For each of the two variants, the table shows the number of solved LPs, the number of LP iterations, and the time in seconds needed for the probing procedure. The last three columns show the relative change of these values achieved by using the reduced probing instead of the full one.

The last column of Table 5.3 shows that the reduced probing variant saves a significant amount of time. Setting aside r15p where the probing finishes in less than one second in both versions, the time reduction ranges from 41% to 92%. The LP change and LPiter change columns demonstrate that this result is mainly caused by the reduction of the number of LP iterations by 80–98%.

On the four smaller instances, the time for the probing procedure is less than two minutes. Given the huge number of variables removed from the formulation by the LP-based probing, it is certainly worth spending this time in the beginning of the solution process. Even on the us67p instance, spending 10.5 minutes on removing 18% of all variables is still acceptable, compared to the overall computation times of several hours. On the large g50p network, less than 10% of the variables can be removed, and the probing procedure takes 1.5 hours with the reduced version (and more than 11 hours with the full version), which is certainly too much time to be spent before each test run.

To reduce the time for the test computations in this thesis, we have applied the full probing version only once for the above instances. In all tests described in the following sections, we read the bounds from a file, and the probing time is not included in the indicated computation times unless otherwise stated. Also in practice, this could be a useful approach for computational studies with different parameters on a large network.

To achieve the computation times presented in Table 5.3, proper parameters must be chosen for the underlying LP solver. First, as explained in Section 4.2.2, we use the primal simplex algorithm to start from the previous basis nearly every time a new LP is solved. Solving the linear programs from scratch multiplies the solution time by a factor of 2–4, depending on the instance. Second, an appropriate pricing algorithm must be chosen in the LP solver. The results in the table have been obtained using quickstart steepest-edge pricing [39]. The default steepest-edge pricing procedure computes many row norms before starting to solve a linear program in order to save simplex iterations later on. If only few iterations are needed anyway, this initialization time is spent in vain and may dominate the whole computation time. Quickstart steepest-edge pricing, in contrast, assumes unit row norms in the beginning, and updates them only when they are computed anyway. Switching to the latter pricing strategy brought the probing time with the reduced probing version down from nearly 12 hours to 1.5 hours on g50p, and from one hour to 10.5 minutes on us67p.

Size of the MIP formulation after presolving

Table 5.4 summarizes the size of the MIP formulations after our problem-specific presolving. The built-in presolving of SCIP usually removes only a small portion of the variables and constraints afterwards. For each network instance, the table shows the total number of variables in the formulation, the share of flow variables, and the number of constraints (including the dynamically generated link diversification constraints). Eventually, the last column gives the ratio of the number of variables to the number of constraints to provide an idea of the shape of the MIP formulation.

inst	# variables	% flow variables	# constraints	vars/cons
p12u	8674	15.23	2681	3.24
r15u	2841	73.00	503	5.65
g17u	5289	72.32	1015	5.21
eu28u	21210	92.68	1931	10.98
g50u	129465	90.84	7020	18.44
us67u	77975	95.86	5908	13.20
p12p	51757	86.01	7540	6.86
r15p	6276	88.05	3846	1.63
g17p	48537	97.02	8206	5.91
eu28p	332789	99.53	37463	8.88
g50p	5579607	99.79	235576	23.68
us67p	2237189	99.86	485810	4.61

Table 5.4: MIP sizes after our presolving. The table shows the total number of variables after our presolving, the share of flow variables, the number of constraints, and the ratio between the numbers of variables and constraints. Survivability constraints greatly increase the problem size.

On all network instances with protection, the share of node and link capacity variables is small, and only few of them are removed by our basic presolving. The share of flow variables is always above 86% on these instances, and most often above 97%. Also on the large unprotected instances, more than 92% of all variables are flow variables. It may be interesting to note that although g50p has less nodes than us67p, it has about twice as many flow variables. This is caused by the much higher number of logical links and commodities. On the other hand, g50p has only roughly half of the constraints of us67p because the number of nodes and physical links, and thus the number of diversification constraints, is smaller. As a result, the ratio between the variables and constraints of g50p exceeds the ratio of us67p by a factor of more than five.

The table also highlights the fact that taking survivability conditions into account has a huge impact on the LP size especially on large networks. The main reason is that demands cannot be aggregated at a common source node but have to be modeled as point-to-point demands in the MIP formulation, and that logical link flow variables $f^k_{\ell,ij}$ have to be employed instead of node-pair flow variables f^k_{ij}. In addition, the LP has to be augmented by a huge number of diversification constraints. As a result, introducing protection requirements increases the number of variables and constraints by factors of 23 and 33 on g50, and even by factors of 28 and 82 on us67.

Lasting impact of our presolving techniques

Bounds and closed gap at the root node Our problem-specific presolving can have a remarkable impact on the dual bounds and the size of the branch-and-cut tree. Table 5.5 summarizes its effect on the initial dual bound, the dual bound at the end of the root node, and the gap closed at the root node.

	± initial lb (%)		± root lb (%)		root closed gap (%)	
inst	basic pp	full probing	basic pp	full probing	basic pp	full probing
p12u	±0.00	–	−0.19	–	−2.10	–
r15u	+4.32	–	−0.01	–	−2.42	–
g17u	+5.24	–	−0.02	–	−0.43	–
eu28u	±0.00	–	+0.26	–	2.21	–
g50u	+9.69	–	+3.27	–	10.55	–
us67u	±0.00	–	−0.01	–	−0.04	–
p12p	±0.00	±0.00	±0.00	±0.00	0.00	0.00
r15p	+9.14	+9.09	+6.36	+7.11	79.64	89.07
g17p	+3.83	+3.83	+2.47	+2.76	22.42	25.01
eu28p	±0.00	±0.00	±0.00	+0.08	0.00	1.08
g50p	+3.80	+3.91	+2.58	+2.57	2.68	2.67
us67p	+1.34	+2.27	+1.27	+1.27	3.77	3.77

Table 5.5: Our presolving can significantly reduce the optimality gap, especially with protection. The first four columns show the relative improvement of the initial LP bound and the root dual bound by our basic presolving and full probing, compared to the corresponding values without our presolving. The last two columns give the optimality gap closed at the root node by our presolving.

For each instance, the second column shows the relative improvement of the initial LP bound ilb_{pre} with basic presolving compared to the initial LP bound ilb_{SCIP} with the SCIP default settings. The relative improvement is defined as $100 \cdot (ilb_{\text{pre}} - ilb_{\text{SCIP}})/ilb_{\text{SCIP}}$. For protected instances, the third column gives the improvement of the initial lower bound with full probing compared to ilb_{SCIP}. The fourth and fifth column indicate the relative improvement of the root dual bound rlb_{pre} with presolving compared to the root dual bound rlb_{SCIP} obtained without it. It is computed as $100 \cdot (rlb_{\text{pre}} - rlb_{\text{SCIP}})/rlb_{\text{SCIP}}$. Both SCIP's built-in cutting planes and our own cutting planes were enabled in all tests. Eventually, the last two columns indicate the percentage of the optimality gap closed at the root node with respect to the best known solution ub. The closed gap is defined as

$$clg = 100 \cdot \big(1 - \frac{ub - rlb_{\text{pre}}}{ub - rlb_{\text{SCIP}}}\big).$$

The closed gap measures the impact of presolving on the lower bound in a way that is independent of a scaling factor applied to the bounds and of constant terms added to the objective function. The time needed for the basic presolving steps is negligible; the time spent in the LP-based probing procedure is given in Table 5.3.

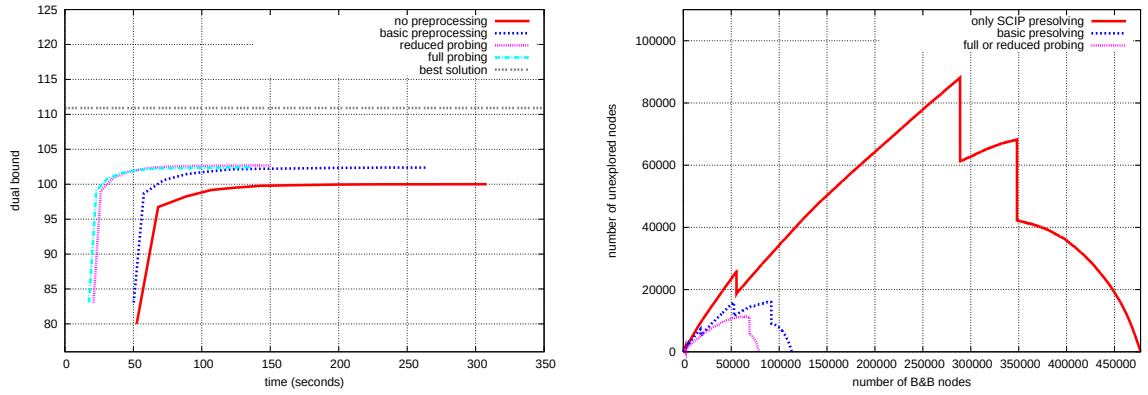
On the unprotected instances, the effect of our presolving is ambiguous. On the one hand, column *initial/basic* of Table 5.5 shows that already the simple presolving steps can increase

the dual bound by as much as 9.69% (on g50u) compared to SCIP's initial LP bound. On the other hand, the impact on the root dual bound is smaller, and this bound even decreases on several instances. Apparently, the employed cutting planes can to some extent do the work which is otherwise done by our presolving. Nevertheless, increasing the root dual bound by 3.27% on g50u closes more than 10% of the optimality gap.

On the protected instances, our presolving has a clearly positive effect. The root dual bound never decreases with basic presolving, and sometimes increases significantly. Apparently, the cutting planes perform better without survivability requirements; this is consistent with our cutting plane results presented in Section 5.2.3. Whereas nothing happens on p12p, already our basic presolving steps close 79% of the optimality gap on r15p; with the LP-based probing, even 89% of the gap are closed.

Notice that the LP-based probing procedure cannot directly influence the initial LP bound, because its decisions are based on LP conditions and do not exploit any integrality constraints. Nevertheless, the initial LP bounds obtained by basic presolving and full probing differ on some instances. The reason is that our probing may have an indirect impact on the initial LP bound by influencing either SCIP's own presolving (on r15p) or our column generation procedure (on g50p and us67p).

Figure 5.6(a) highlights the benefit of our presolving for the dual bound using the g17p network. It shows the progress of the dual bound at the root node without problem-specific presolving, with basic presolving, and with additional full or reduced LP-based probing. The value 100 corresponds to the SCIP's default root dual bound (rlb_{SCIP}). The line at the top indicates the value of the best known solution. The figure illustrates that our presolving closes 25% of the gap between the default root dual bound and the best known solution. Most of the reduction is achieved by the basic presolving steps, and the difference between the two probing variants is marginal.



(a) Root gap reduced by 25% (g17p)

(b) B&C nodes reduction of 84% (r09p)

Figure 5.6: Our presolving has a lasting impact on the dual bound and the branch-and-cut tree.

Impact on the branch-and-cut tree To investigate the lasting effect of presolving on the number of branch-and-cut nodes needed until proven optimality, we constructed a smaller instance r09p with 9 nodes, 10 physical and 21 logical links, and 58 demands, which can be solved to optimality even without problem-specific presolving. The instance has been derived

from r15p by iteratively removing a node with smallest emanating demand and connecting the neighbors of the removed node.

Figure 5.6(b) illustrates the enormous impact of the problem-specific presolving on the size of the branch-and-cut tree. It shows the number of unexplored branch-and-cut nodes over the number of explored nodes in the SCIP default settings, with basic presolving, and with the LP-based probing (both probing variants are equivalent on this instance). Already basic presolving reduces the number of branch-and-cut nodes solved until proven optimality by 76%. The size of the search tree and the computation time are decreased by the same order of magnitude (from half an hour to less than 10 minutes). With additional LP-based probing, the number of solved branch-and-cut nodes is even reduced by 84%!

It may be interesting to note that these results are not only caused by the fact that our primal heuristics found good solutions earlier in the search tree with problem-specific presolving than without. Another test on r09p where the optimal solution value was set as an upper cutoff bound from the beginning led to similar results.

5.2.2 Primal heuristics

Test setting To evaluate our primal heuristics, we compared two test runs on each instance. The first was done with the SCIP default settings (including our presolving and cutting planes), whereas in the second one, all of our heuristics were enabled in addition:

- the routing heuristic FINDFEASIBLEROUTING to compute routings at the beginning of the root node,
- the fast combinatorial hardware installation heuristic INSTALLCAPHEUR as a construction heuristic at the beginning of the root node (based on the result of FINDFEASIBLEROUTING), during the cutting plane loop and at every node, and within various sub-MIPs (based on LP solutions),
- its sub-MIP variant INSTALLCAPMIP as an improvement heuristic every time a new best solution was found,
- the sub-MIP heuristic REROUTEMIP as a construction heuristic based on the LP flow at the end of the root node, and as an improvement heuristic based on previously computed solutions at every 3rd depth of the branch-and-cut tree,
- the TOPOLOGYCROSSOVER improvement heuristic at every 20th depth of the search tree, and
- the ITERATIVECAPREDUCTION improvement heuristic in the combinatorial version, where demands are iteratively rerouted using a min-cost-flow algorithm (for unprotected demands) or the FINDFEASIBLEROUTING heuristic (for protected demands).

On the smaller test instances (eu28u and those with up to 17 nodes), we imposed a time limit of three hours; on the large test instances eu28p, g50u, g50p, us67u, and us67p, the time limit was set to six hours.

We disabled the INSTALLCAPMIP heuristic on some instances where already the combinatorial INSTALLCAPHEUR heuristic performed well due to the cost structure, as described in Section 4.3.1. On the large test instances, we disabled our REROUTEMIP heuristic as a construction heuristic at the root node because it takes too much time on these instances. In

addition to our own heuristics, all built-in heuristics of SCIP were applied with their default settings, which led to 21 different heuristics being applied in total (15 generic ones built into SCIP and 6 problem-specific ones).

Presentation of the results Table 5.6 summarizes the results. For each instance, the second column displays the relative improvement of the dual bound lb_{heur} with our heuristics compared to the dual bound lb_{SCIP} with the SCIP default settings. It is defined as $100 \cdot (lb_{\text{heur}} - lb_{\text{SCIP}})/lb_{\text{SCIP}}$. Hence, positive values represent an improvement. The third column shows the relative increase of the primal bound ub, defined in an analogous way as $100 \cdot (ub_{\text{heur}} - ub_{\text{SCIP}})/ub_{\text{SCIP}}$. In this case, an improvement is indicated by a negative value. In all rows marked with $-\infty$, a solution was found with our heuristics but not by SCIP alone.

The fourth column shows the relative portion of the optimality gap closed by our heuristics. If lb denotes the best known dual bound and ub the upper bound computed with a given setting, the optimality gap is defined as $gap = 100 \cdot (ub - lb)/ub$. This gap definition, which is the same as used by CPLEX, gives an upper bound on how much the primal solution can be improved relative to its current value. The closed gap is defined as

$$clg = 100 \cdot \left(1 - \frac{ub_{\text{heur}} - lb}{ub_{\text{SCIP}} - lb}\right) = 100 \cdot \left(\frac{ub_{\text{SCIP}} - ub_{\text{heur}}}{ub_{\text{SCIP}} - lb}\right)$$

It describes by how much the optimality gap to the best known dual bound could be reduced by the change in the primal bound. Again, a negative value indicates an improvement, and a value of ∞ indicates that a solution was found with our problem-specific heuristics but not with the SCIP default settings.

The closed gap with respect to a fixed lower bound is a common performance measure used in the literature to judge the quality of primal heuristics. As such heuristics are mainly intended to reduce the primal bound, this measure focuses on the influence of that bound. Nevertheless, applying a heuristic can also affect the lower bounds by earlier cutoffs in the search tree or by reduced cost strengthening [3]. To measure the combined impact of our heuristics on the upper bound and lower bound, the fifth column shows the relative change of the optimality gap not with respect to the best known dual bound, but to the dual bounds from the respective test run. It is defined as $100 \cdot (gap_{\text{heur}} - gap_{\text{SCIP}})/gap_{\text{SCIP}}$. This value is precisely the negative value of the closed gap if the dual bound is the same in both runs with and without heuristics (the difference for r15p comes from the fact that the change of the dual bound is below the precision of the table). Negative values represent an improvement.

Eventually, the last column shows the name of the heuristic which found the best solution in the run with all heuristics. CROSSOVER is the crossover heuristic of SCIP. It fixes all variables which have the same value in several solutions, and solves the remaining problem as a sub-MIP. RENS is a built-in heuristic of SCIP which finds an optimal rounding of the LP solution by solving a sub-MIP. A detailed description of both heuristics is given by Berthold [24] and Achterberg [3]. The entry LPRELAXATION means that the best solution was found via an integral LP solution at some branch-and-cut node. All other heuristics from the table are explained in Section 4.3. The postfix 'i' or 'c' means that the REROUTEMIP heuristic was called as an improvement heuristic or as a construction heuristic, respectively.

Discussion of the results The first important observation is that for the five largest instances, our problem-specific heuristics identify a feasible solution, whereas no solution is found with the SCIP default settings.

inst	± dual (%)	± primal (%)	clg (%)	± gap (%)	best sol found by
p12u	+0.03	±0.00	±0.00	−3.37	Crossover
p12p	+0.01	−0.03	3.85	−4.50	RerouteMip (c)
r15u	−0.01	−0.07	25.86	−23.89	Crossover
r15p	±0.00	−0.17	17.13	−16.94	RerouteMip (i)
g17u	±0.00	±0.00	±0.00	±0.00	LPRelaxation
g17p	+1.09	+0.22	−9.12	−20.52	RerouteMip (i)
eu28u	+0.34	+4.47	−94.30	+55.82	RerouteMip (c)
eu28p	−0.36	−∞	∞	−∞	Rens
g50u	+0.02	−∞	∞	−∞	RerouteMip (i)
g50p	−0.40	−∞	∞	−∞	InstallCapHeur
us67u	−0.05	−∞	∞	−∞	RerouteMip (i)
us67p	−0.03	−∞	∞	−∞	InstallCapHeur

Table 5.6: Our heuristics find feasible solutions for the large instances where Scip alone fails. They also improve the gap on most smaller instances. The table shows the relative bound and gap changes in a branch-and-cut run with all our heuristics, compared to a run without them. The time limit was set to three hours (p12, r15, g17, eu28u) or six hours (eu28p, g50p, us67p).

Second, the *clg* column shows that on some of the other instances, large parts of the optimality gap are closed although the primal bounds differ only marginally. Unfortunately, the largest relative changes occur on g17p and eu28u where the quality of the primal bound degrades. Nevertheless, the ±*gap* column shows that the actual gap, which is computed using the dual bound from the respective run, is reduced on most instances. The reason is that the dual bound sometimes also increases. Except for eu28u, where the quality of the primal bound degrades with our heuristics, and g17u, which is solved to optimality in both settings, our heuristics lower the gap in all other 10 instances, sometimes by more than 20%.

Third, the best solution is found by one of our problem-specific heuristics in 8 out of the 12 runs. The RerouteMip heuristic is particularly successful. Applied as a construction heuristic based on the root LP solution, it finds the best network configuration within the time limit in two out of the 12 cases where it was called. Applied as an improvement heuristic which tries to reduce capacities by rerouting flow, it finds the best solution in another four cases. That is, despite the large number of other generic and problem-specific heuristics, this heuristic is responsible for the best solution in half of the instances.

Table 5.7 shows some success statistics for each heuristic, aggregated over all test instances. The FindFeasibleRouting heuristic has been omitted from the table because it only computes routings rather than complete solutions. As explained above, we use it together with InstallCapHeur. For each heuristic, the table shows the number of calls in all test runs, the number of identified solutions, and the number of instances where this heuristic found the best solution within the time limit. The last two columns give the total time in seconds spent in each heuristic (out of 51 hours total running time for all network instances) and the average time per heuristic call, also in seconds.

The InstallCapHeur finds feasible solutions for every network instance. On the large protected network instances g50p and us67p, the solution produced by this heuristic is the only one found within the time limit. The InstallCapMip heuristic, which has only been

heuristic	called	found	best sol	total time	avg. time/call
InstallCapHeur	38070	26	2	191	0.01
InstallCapMip	27	1	0	184	6.82
RerouteMip (c)	7	7	2	5951	850.11
RerouteMip (i)	28	13	4	12122	432.94
IterativeCapReduction	54	0	0	2881	53.36
TopologyCrossover	23	3	0	7045	306.29

Table 5.7: Our InstallCapHeur and RerouteMip heuristics are particularly successful. For each heuristic, the table shows the number of calls and of identified solutions, the number of times the heuristic found the best solution within the time limit, and the total and average time in seconds, aggregated over all instances.

called on p12, r15, and g17, identifies one solution which is later improved by an integral LP solution. In contrast to the sub-MIP heuristic RerouteMip, its combinatorial equivalent IterativeCapReduction does not identify any feasible network configurations. TopologyCrossover finds feasible solutions on p12p, g50u, and us67u, but other heuristics compute better network configurations later on.

Even bad solutions can indirectly improve the primal bound. On eu28p, the best solution is computed by Rens, but only if our FindFeasibleRouting heuristic is called at the start of the root node. Although the initial solution produced by FindFeasibleRouting together with InstallCapHeur is rather bad, Scip uses it to strengthen variable bounds using reduced cost strengthening (see [3] for details). The cutting plane algorithm then continues differently than without the initial solution, which makes Rens succeed after the root node.

Good solutions are often found by combinations of different heuristics, both generic ones built into Scip and problem-specific ones written by ourselves. It frequently happens that a solution computed by some construction heuristic can be improved several times by improvement heuristics. Moreover, we observed that applying our fast combinatorial heuristics within the sub-MIP heuristics tends to improve the overall solution quality.

5.2.3 Cutting planes

In this section, we investigate the effect of the different types of cutting planes on the lower bound at the branch-and-cut root node. First, we add each type of our cutting planes separately to the LP relaxation. In a second test, we add all cuts together. To investigate why our cutset and flow-cutset inequalities have only little impact with protection, we then study their extended two-layer versions which take the survivability requirements into account. Our problem-specific presolving was enabled in all tests; the term "Scip default settings" always refers to Scip with our presolving. In computations beyond the root node, also our heuristics were enabled in all tests.

Notice that not all violated cuts are necessarily added to the LP relaxation. Forcing all user-defined cuts into the LP leads to a huge formulation with extremely long solving times. For this reason, Scip selects a subset of the violated cutting planes to be added to the LP according to various criteria, such as the violation of a cut and its degree of orthogonality to

the objective function and to other identified cutting planes. In an extensive series of tests, we found that we obtained by far the best results by forcing all violated cutset and physical layer inequalities into the LP, but letting SCIP select a subset of the violated flow-cutset inequalities. These settings have been used in all our tests unless otherwise stated.

Contrary to what one might expect, the lower bound at the end of the root node may be lower with all user-defined cuts than without because different LP solutions are obtained after the first cutting plane iteration. Furthermore, state-of-the-art MIP solvers like SCIP or CPLEX stop the separation process when the progress in the dual bound gets too small. A test where we searched for violated cutting planes until no more of them were found led to huge computation times. We thus use the default settings of SCIP for stopping the cutting plane process at the root node.

Unprotected demands

Adding cutting planes separately In a first test, we have added each type of our cutting planes separately to the LP relaxation in addition to the internal cuts of SCIP on the test instances without protection.

	cutset			flow-cutset			fixed-charge		
inst	± lb (%)	clg (%)	cuts	± lb (%)	clg (%)	cuts	± lb (%)	clg (%)	cuts
p12u	+1.27	11.40	64	−0.12	−1.08	189	+0.76	6.87	5
r15u	+0.05	16.70	89	+0.00	0.75	224	–	–	–
g17u	−0.16	−0.57	91	+0.44	1.54	221	+22.51	78.64	53
eu28u	+0.11	0.91	76	+0.05	0.45	390	+0.32	2.69	3
g50u	+2.54	1.48	446	+52.15	30.37	22140	+71.37	41.56	248
us67u	+0.08	0.25	443	+1.59	5.13	10309	+7.74	24.92	189

Table 5.8: Unprotected instances: the physical layer fixed-charge cuts are the most important ones. The table shows the improvement of the lower bound and the gap closed at the root node, as well as the number of violated cutting planes with only one type of cuts in addition to SCIP's default cuts.

The results are summarized in Table 5.8. It presents three values for each instance and each type of cutting planes out of cutset inequalities (4.6) (*cutset*), flow-cutset inequalities (4.8) (*flow-cutset*), and physical layer degree and tree inequalities (4.10) and (4.12) (*fixed-charge*). The first column of each block shows the relative improvement of the dual bound lb_{cuts} at the end of the root node caused by the corresponding type of cutting planes, compared to the dual bound lb_{SCIP} with SCIP's own cuts only. The relative improvement of the dual bound is defined as $100 \cdot (lb_{\text{cuts}} - lb_{\text{SCIP}})/lb_{\text{SCIP}}$. On p12u, for example, the lower bound obtained with cutset inequalities is 1.27% higher than without them. The second column of each block (*clg*) shows the closed gap in percent. If ub denotes the best known upper bound, the optimality gap is defined as $gap = 100 \cdot (ub - lb)/ub$. The closed gap is defined as

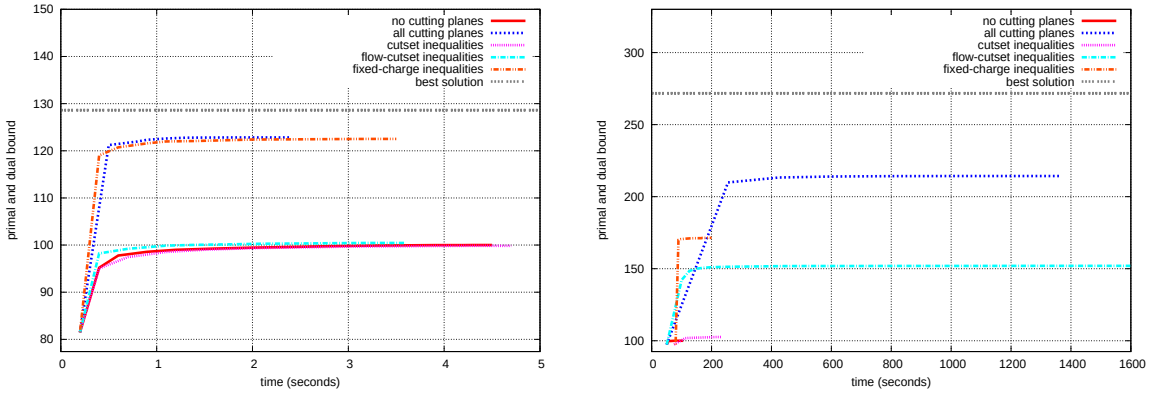
$$clg = 100 \cdot \left(1 - \frac{ub - lb_{\text{cuts}}}{ub - lb_{\text{SCIP}}}\right) = 100 \cdot \frac{gap_{\text{SCIP}} - gap_{\text{cuts}}}{gap_{\text{SCIP}}}.$$

It describes by how much the optimality gap could be reduced by the corresponding type of inequalities. In the third column of each block, the table also shows the number of identified

cutting planes of each class. On r15u, no physical layer inequalities were separated because the physical links do no incur cost and are equipped with as much capacity as needed.

The table shows that our cutting planes increase the lower bound at the end of the root node in most cases. The impact of the cutting planes varies widely among the type of inequalities and the network instances.

The by far most important cutting planes are the physical layer degree and tree inequalities, especially on g17u and g50u. In good solutions computed for these two networks, the physical layer contributes 30-40% to the total network cost. In the initial LP relaxation, many of the physical link capacity z_e are close to zero, implying that also the physical link cost is only counted to some small fractional part. After adding some cutting planes at the physical layer and re-solving the LP, many of these variables are integral, or at least closer to 1.



(a) g17u (b) g50u

Figure 5.7: Unprotected demands: dual bound over time at the root node. Fixed-charge cuts close large parts of the gap on both instances. On g50u, also the flow-cutset inequalities are important.

Figure 5.7 illustrates the importance of the physical layer inequalities for the g17u and g50u network. It shows the progress of the lower bound over time with the SCIP default settings and with different types of cutting planes. Similarly to Table 5.8, the values are scaled such that 100 corresponds to the final dual bound with the SCIP default settings; its progress over time is indicated by the line *no cutting planes*. The line at the top represents the best known solution value which cannot be exceeded by the dual bound curves. For the g17u network, this value is known to be optimal. The figure shows that the fixed-charge inequalities close nearly 80% of the optimality gap on g17u and more than 40% on g50u within short computation time.

In contrast to the cutting planes at the physical layer, the effect of the logical layer cutset inequalities is marginal in most cases. On the g17u network, they even slightly decrease the lower bound. On the small instances p12u and r15u, the small progress in the dual bound is at least partly caused by the fact the the upper and lower bounds are already very close without our cuts; the table shows that the cutset inequalities can close more than 11% and 16% of the optimality gap on these networks, respectively. The small effect of cutset inequalities observed on our instances is in contrast to the results of Raack et al. [115] for single-layer (logical) networks, where cutset inequalities greatly improved the dual bound on a large set of network instances from SNDLIB. One possible reason for the difference might be the much denser logical network structure in multi-layer networks. Another reason might be that in

[115], CPLEX was used as the branch-and-cut framework, whereas our computations are based on SCIP. In particular, this leads to different built-in cutting planes and separation procedures. Furthermore, on the single-layer instances in [115], the integrality gap was precisely caused by the capacity and flow variables targeted by the cutset and flow-cutset inequalities. This is in contrast to this thesis, where also physical layer capacity variables significantly contribute to the gap.

Also the flow-cutset inequalities affect the lower bound only little in most cases, although many violated inequalities of this class are identified. This is in accordance with the results of Raack et al. [115]. There is, however, one exception: the g50u network. On this network, the flow-cutset inequalities increase the lower bound by a factor of 1.52 and close more than 30% of the optimality gap. The table shows that even with our rather simple separation algorithms, more than 22000 cutting planes of this type are identified. Even though SCIP already selects only a subset of these to be added to the LP, Figure 5.7(b) illustrates that flow-cutset inequalities should be applied with care. The dual bound increases quickly in the first iteration where this type of cutting planes is added, and then remains nearly constant for 24 hours (the figure has been cut off at 1600 seconds to show the effect of the other cutting planes as well). Although the time spent at the root node with all our cutting planes is much smaller than with flow-cutset inequalities only, it is still more than 20 minutes, compared to 2 minutes with the SCIP default settings. Summarizing, flow-cutset inequalities should only be applied at the root node of the branch-and-cut tree, and it may be wise to limit their separation to a few cutting plane iterations. Table 5.8 suggests that such a setting is also useful on the other network instances, where many violated flow-cutset inequalities are identified without much impact on the lower bound.

Adding all cutting planes together In a second test, we searched for violated cutset, flow-cutset, and fixed-charge inequalities together in addition to SCIP's default cutting planes.

		gap (%)			# violated cuts		
inst	± lb (%)	SCIP cuts	all cuts	clg (%)	cutset	flow-cutset	fixed-charge
p12u	+2.27	10.02	7.97	20.42	71	174	4
r15u	+0.05	0.28	0.24	16.71	88	230	–
g17u	+22.82	22.25	4.51	79.75	91	210	54
eu28u	+0.58	10.68	10.17	4.86	78	391	3
g50u	+114.31	63.20	21.13	66.56	444	5730	151
us67u	+8.98	23.71	16.85	28.90	443	6327	184

Table 5.9: Unprotected instances: large parts of the integrality gap can be closed with all cuts together. The table shows the improvement of the lower bound, the optimality gap, and the closed gap at the root node with all our cuts compared to the SCIP default settings, and the number of violated cutting planes of each class.

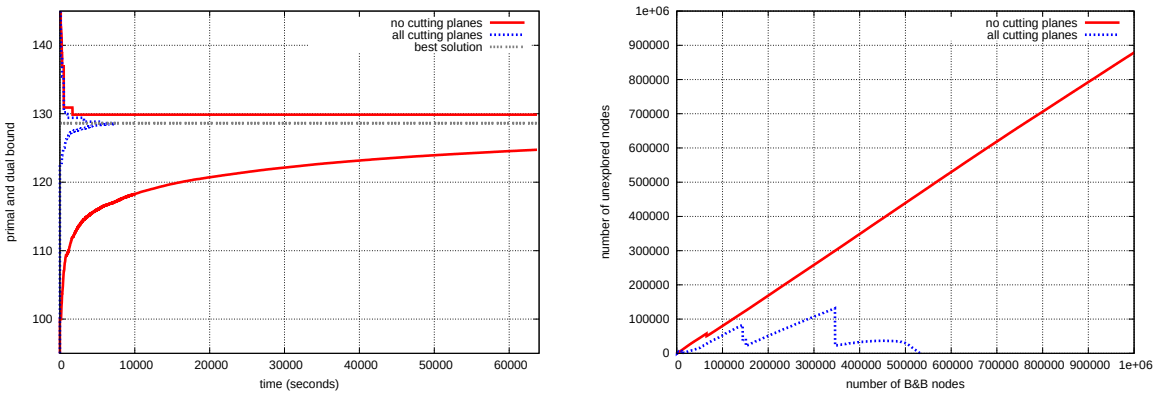
Table 5.9 shows the improvement of the lower bound with all types of cutting planes compared to the SCIP default settings. As the reference value is the same as in Table 5.8, the improvements of the dual bounds from both tables can be compared. The next three columns of Table 5.9 show the optimality gap with the SCIP defaults and with our cutting planes in addition, and the closed gap, defined in the same way as in the previous test. Eventually, the

table shows the number of identified violated inequalities of each class.

With all types of cutting planes in addition to SCIP's cuts, the dual bound can be improved in all cases, and large parts of the remaining optimality gaps can be closed. On r15u, our cutting planes cannot improve the dual bound very much because it is very close to the optimal solution already without our cutting planes. Even the small increase of the lower bound by only 0.05% closes more than 16% of the remaining optimality gap.

On the g50u instance, our problem-specific inequalities close roughly two third of the optimality gap. This is also reflected in Figure 5.7(b). The figure shows that both fixed-charge cuts and flow-cut inequalities play an important role. On the g17u instance, nearly 80% of the gap can be closed, mostly by the physical layer fixed-charge inequalities (see Figure 5.7(a)). Also on the other networks, between 4.8% and 20% of the gap at the root node are closed.

Lasting effect of our cutting planes Eventually, we have investigated the lasting effect of our cutting planes on the dual bound in a longer computation on the g17u network. It turns out that the problem-specific cutting planes can have a huge impact on the size of the branch-and-cut tree.



(a) Primal and dual bound over time.

(b) Size of the branch-and-cut tree: number of unexplored vs. number of explored nodes.

Figure 5.8: Our cuts have a huge impact on the dual bound and the branch-and-cut tree on g17u. The figures show a longer test run with and without our cutting planes in addition to SCIP's cuts.

Figure 5.8(a) illustrates the development of the dual bound in a longer computation with and without all our cutting planes compared to the optimal solution (represented by the straight line in the middle). Again, 100 corresponds to the root dual bound value with the SCIP built-in separators only. In the run with all cutting planes, we searched for violated cutset and fixed-charge inequalities within the whole branch-and-cut tree, but limited the separation of flow-cut inequalities to the root node. Both our presolving routines and primal heuristics were applied in both tests in addition to SCIP's built-in routines.

By applying all separators, we could solve the problem g17u to optimality within slightly more than two hours. Without our cutting planes, the computation was aborted after more than 17 hours because the huge search tree exceeded the available memory of 8 GB. The optimality gap was 3.9% at that point. The growth of the search tree is illustrated by Figure 5.8(b), which shows the number of unexplored branch-and-cut nodes over the number

of explored ones with and without cutting planes. The figure has been cut off at 1 million nodes; the curve without cutting planes continues as a nearly straight line until 7 million explored and 4.6 million unexplored nodes. Summarizing, our cutting planes greatly reduce the size of the search tree on this instance and enable us to solve it to optimality.

Protected demands

In this section, we report on the results of the cutting plane tests for the network instances with protected demands. As in the unprotected case, we investigated the impact of adding either only one type of problem-specific cuts or all types of cuts simultaneously. In addition, all built-in separators of SCIP and our problem-specific presolving were used in all tests.

In a first approach, we used the same inequalities as in the unprotected case, which operate at one layer only. As we had to generate flow variables dynamically to solve the large protected instances g50p and us67p, flow-cutset inequalities could not be applied to these network instances. Tables 5.10 and 5.11 show the results of our cutting planes being applied separately or simultaneously. The columns have the same meaning as in Tables 5.8 and 5.9, respectively.

	cutset			flow-cutset			fixed-charge		
inst	± lb (%)	clg (%)	cuts	± lb (%)	clg (%)	cuts	± lb (%)	clg (%)	cuts
p12p	±0.00	0.00	60	±0.00	0.00	247	+0.08	2.54	12
r15p	−0.01	−0.92	97	±0.00	−0.56	555	–	–	–
g17p	+0.04	0.23	86	−0.02	−0.14	672	+7.56	46.37	214
eu28p	+0.04	0.54	69	+0.04	0.52	81	−0.01	−0.14	28
g50p	+1.22	0.74	439	–	–	–	+36.41	22.09	2331
us67p	+0.14	0.42	455	–	–	–	+1.59	4.67	1558

Table 5.10: Protected instances: our cuts improve the dual bound on most instances, but less than in the unprotected case. The table shows the improvement of the root lower bound, the gap closed at the root node, and the number of violated cutting planes with only one type of our cutting planes in addition to SCIP's built-in ones.

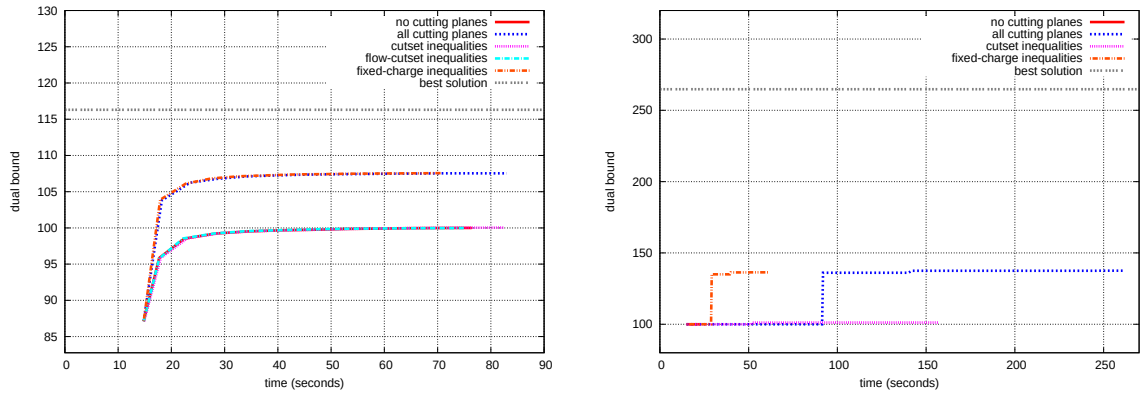
The first observation from Table 5.10 is that the physical layer degree and tree inequalities (4.11) and (4.12) (column *fixed-charge*) still show a positive effect on the dual bound on all network instances except for eu28p, but the improvement is much smaller than in the unprotected case. Second, the cutset inequalities (4.6) (*cutset*), and flow-cutset inequalities (4.8) (*flow-cutset*) have basically no impact although many violated cuts are found.

By comparing the lower bound changes in Tables 5.10 and 5.11, one can see that even if all cuts are applied together, the main progress in the lower bound is caused by the physical layer inequalities. Major parts of the gap can be closed for g17p and g50p only. Figure 5.9 shows the dual bound over time for these two networks with different cutting planes, together with the best known solution (100 corresponds to the root dual bound without our cutting planes). The fixed-charge inequalities close nearly half of the optimality gap for g17p, whereas the cutset and flow-cutset inequalities have only marginal effect. On g50p, 22% of the gap are closed, mainly by the fixed-charge inequalities.

A possible reason for the weakness of the cutset and flow-cutset inequalities in the pro-

		gap (%)			# violated cuts		
inst	± lb (%)	SCIP cuts	all cuts	clg(%)	cutset	flow-cutset	fixed-charge
p12p	+0.08	2.90	2.82	2.54	60	264	12
r15p	±0.00	0.82	0.82	0.34	97	586	–
g17p	+7.53	14.52	8.08	46.19	86	661	235
eu28p	+0.06	7.03	6.97	0.76	69	78	28
g50p	+37.58	62.23	48.04	22.80	439	–	2432
us67p	+1.60	25.43	24.24	4.68	455	–	1557

Table 5.11: Protected instances: with all our cutting planes, the dual bounds are improved, but only slightly. Only on g17p and g50p, they have a significant impact. The table shows the improvement of the lower bound and the gap closed at the root node with all our cuts compared to the run without them, and the number of violated inequalities.



(a) g17p (b) g50p

Figure 5.9: Fixed-charge cuts are the only effective cuts on the protected instances. The figures show the lower bound over time at the root node with and without our cutting planes.

tected case is that these inequalities have originally been designed for a multi-commodity flow in a single network layer (the logical layer in our case) and do not capture the additional survivability restrictions from the physical layer. This conjecture is investigated below.

Inter-layer cuts

We now investigate whether taking the inter-layer protection requirements into account improves the performance of the cutset and flow-cutset inequalities. Table 5.12 summarizes the results of a test at the root node with different types of link failure cutset and flow-cutset inequalities, and two strategies to separate the latter.

The second to forth column repeat the closed gap and the number of separated cutset (*cut*) and flow-cutset (*fcut*) inequalities with our single-layer cuts from Table 5.11 for comparison purposes. As in the previous tables, the closed gap is measured with respect to the dual bound without any problem-specific cutting planes and the best known upper bound.

The next two blocks of columns show the closed gap and the number of violated cuts for two

strategies of separating link failures cutset and flow-cutset inequalities. Given a network cut defined by a subset $S \subset V$ of nodes, we generate not only the single-layer cutset and flow cutset inequalities for this cut, but also link failure cutset (4.16) and flow-cutset inequalities (4.17). With the first strategy, described in the second block of columns in Table **??**, we generate link failure inequalities for all physical links $e \in E$ which cause any logical cut link $\ell \in L_S$ to fail. This may include physical links which are not part of the physical cut. With the second strategy, given in the last block of columns, we generate link failure inequalities only for physical cut links $e \in \delta_E(S)$. In addition to the logical layer cuts, we apply fixed-charge inequalities and SCIP's default cuts in all tests. Again, flow-cutset inequalities cannot be used on g50p and us67p because of column generation.

	single-layer cuts			failure cuts ($e \in E$)			failure cuts ($e \in \delta_E(S)$)		
inst	clg (%)	cut	fcut	clg (%)	cut	fcut	clg (%)	cut	fcut
p12p	2.54	60	264	2.54	1136	2684	2.54	284	1079
r15p	0.34	97	586	−82.73	1492	7239	0.40	299	1045
g17p	46.19	86	661	45.10	2262	16282	47.14	430	2546
eu28p	0.76	69	78	0.47	2757	3511	0.12	315	776
g50p	22.80	439	–	23.21	37126	–	23.22	2575	–
us67p	4.68	455	–	4.71	9644	–	4.70	1733	–

Table 5.12: Although many violated cutting planes are found, the additional value of the inter-layer cuts compared to the single-layer cutting planes is small. The table shows the relative lower bound change at the root node compared to the SCIP default settings and the number of violated cutset and flow-cutset inequalities for different cutting planes and separation strategies.

Link failure cutset and flow-cutset inequalities should be applied carefully. A comparison of the two last blocks of columns in Table 5.12 shows that generating link failure inequalities only for physical cut links $e \in \delta_E(S)$ is preferable to considering all physical cut links. The differences in the closed gap are negligible, but much less cutting planes are generated with the more conservative strategy. It must be noted that the seemingly dramatic deterioration of the optimality gap by 82.73% on r15p is caused by a decrease in the dual bound of 0.68% only; the primal and dual bound are very close on this instance.

Surprisingly, taking the physical layer into account in the cutting planes has little impact on the dual bound. A comparison of the first and the last block shows that even though the closed gap is slightly larger with the extended two-layer inequalities than with their single-layer counterpart on four out of the six instances, the difference is negligible. The number of constraints in the linear relaxation drastically increases without improving the dual bound significantly compared to the single-layer version.

A possible reason for their small impact might be that the SCIP built-in cuts and our logical layer cuts already cover most of the survivability aspects. The theoretical results for two-layer Q-subset inequalities (see Section 4.4.4) indicate that especially on dense physical cutsets, the logical layer cutset inequalities are already quite strong.

Summary of the cutting plane results

Our results of the cutting plane investigations can be summarized as follows:

- For our network instances without protection, the physical layer degree and tree inequalities are the most important cuts. With the exception of g50u, the cutset and flow-cutset inequalities have only marginal impact on the lower bound when applied alone. Applying all cuts together closes large parts of the optimality gaps. A test on g17u showed that the cuts can significantly reduce the size of the branch-and-cut tree.

- With protection, all of our cuts have less effect than without protection. Nevertheless, the physical layer cuts reduce the optimality gap by 44% on g17p and by 22% on g50p.

- The link failure cutset and flow-cutset inequalities, which take the inter-layer protection requirements into account, improve the dual bound in most cases compared to the single-layer version of the cuts. Nevertheless, the progress is small. We found that the cuts must be carefully selected to avoid generating too many cuts. Generating link failure inequalities for physical cut links only is more effective than generating inequalities for all physical links and letting SCIP select a subset of the cuts.

5.2.4 Column generation

This section compares the enumeration-based and the path-based pricing algorithm on the two large protected network instances us67p and g50p. We solved these two instances with a time limit of six hours using a branch-and-cut-and-price approach on the full MIP formulation with all hardware and routing constraints. We disabled all of our primal heuristics except for INSTALLCAPHEUR. It is the only heuristic which finds solutions on these large instances at all. In contrast to combinatorial heuristics, INSTALLCAPHEUR uses only variables from LP solutions and cannot distort the pricer test by generating new variables.

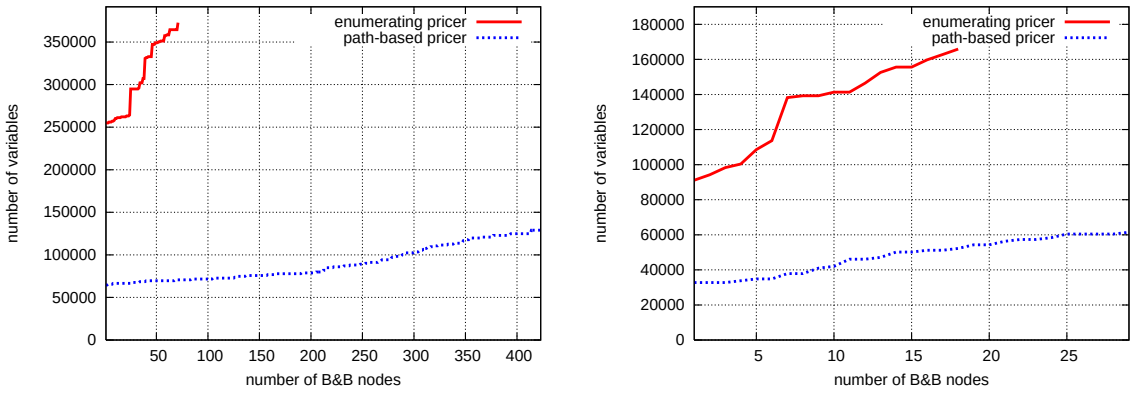
The test revealed that the path-based pricer is clearly preferable. Table 5.13 summarizes some key results. The first column contains the network instance and the pricer ('e' for enumeration-based and 'p' for path-based). The second column shows the time in seconds spent at the root node. The next three columns indicate the dual bound at the end of the root node and the final dual and primal bound after six hours, all relative to the initial LP bound. The following two columns give the number of variables after the root node and in the end relative to the number of variables in the initial LP. Eventually, the last three columns show the number of pricing iterations, the average number of variables generated in each iteration (rounded to integers), and the total time in seconds spent in the pricing procedure.

Already at the root node, the path-based pricer generates only roughly one third of the number of variables generated by the enumeration-based one. Although it is called more often, it consumes less time in total and solves the root node in significantly shorter time. The quality of the computed bounds is nearly the same for both pricers.

Figure 5.10 shows the number of variables over the number of branch-and-cut nodes for us67p and g50p. It illustrates that the path-based pricer generates much less variables. As a result, more branch-and-cut nodes can be solved during the same time (29 instead of 18 nodes for g50c, and 423 instead of 71 nodes for us67c). Notice that both pricers start from the same set of initial variables; the beginning of the curves in the figure indicates the number of variables at the end of the root node. This illustrates the result that the enumeration-based pricer generates many more variables than the path-based one already at the root node. As a consequence of these observations, we have used the path-based pricer in all subsequent tests.

inst/ pricer	time root	bounds/initial lb root lb	final lb	final ub	vars/initial vars root	final	pricing it	vars/it	time
g50p/e	2130	1.38	1.39	4.23	4.33	7.83	286	495	520
g50p/p	730	1.38	1.39	4.22	1.56	2.91	1927	58	444
us67p/e	1485	1.02	1.02	1.34	5.65	8.24	575	557	628
us67p/p	431	1.02	1.02	1.34	1.44	2.86	1282	64	575

Table 5.13: The path-based pricer (p) is clearly better than the enumeration-based pricer (e). The table shows the time in seconds spent at the root node, the root lower bound and the lower and upper bound after 6h in relation to the initial LP bound, and the number of variables after the root node and after 6h in relation to the number of initial variables. The last columns give the number of pricing iterations, the number of variables generated per iteration, and the pricing time in seconds.



(a) us67p (b) g50p

Figure 5.10: Number of variables over the number of branch-and-cut nodes. As the path-based pricer generates much less variables, more nodes can be solved within the time limit of 6h.

5.2.5 Metric inequalities

In this section, we first investigate the technique of adding commodities to the routing LP dynamically. Afterwards, we compare the performance of Benders decomposition with the bottleneck or feasibility LP formulation to the integrated approach where the master MIP contains both hardware and routing information.

Adding commodities dynamically

We now report on our tests of different strategies to add groups of commodities to the routing LP sequentially for the large protected networks. The motivation is that if all commodities are included in the routing LP from the beginning, testing a capacity vector for feasibility with respect to the routing constraints sometimes takes several hours on these instances, as explained in Section 4.5.4.

In a first test, we add the commodities either one by one (*commodity-wise* strategy), or we sort the network nodes decreasingly by demand and add all commodities starting or

ending in the largest node, then those incident to the second-largest node, and so on (*node-wise* strategy). Despite our strengthening procedure for metric inequalities with a subset of commodities described in Section 4.5.4, this simple approach may lead to rather weak dual bounds because the generated metric inequalities only contain information about few commodities. We thus also tried another strategy where we add a new group of commodities to the LP after k LP-solving and pricing iterations, regardless of whether the LP has been optimally solved with respect to the current set of variables or not.

We compared these strategies on the two large protected network instances g50p and us67p in a three-hour test run using the bottleneck routing formulation from Section 4.5.2 and the path-based pricer from Section 4.6.3. We applied all default separators built into SCIP, cutset inequalities, and physical-layer fixed-charge cuts. Similar to the more time-consuming separators of SCIP, we applied the metric inequality separator for fractional solutions only at the root node. As the focus of this test is on the dual bound, we disabled all primal heuristics.

inst	dynamic strategy	B&C nodes	± lb (%) root	± lb (%) final	commodities (%) root	commodities (%) final	# MI
g50p	all commodities	1	+ 0.02	+ 0.02	100.00	100.00	16
	single/10	1	+ 0.01	+ 0.01	13.31	13.31	10
	single/15	2242	+ 0.01	+ 0.18	7.91	13.88	16
	single/20	3611	+ 0.01	+ 0.40	7.91	12.24	17
	single/25	2761	± 0.00	+ 0.37	5.47	7.92	16
	single/∞	1	± 0.00	± 0.00	3.76	3.76	18
	nodes/10	1	+ 0.03	+ 0.03	51.43	51.43	14
	nodes/15	1	+ 0.03	+ 0.03	48.57	48.57	19
	nodes/20	1	+ 0.04	+ 0.04	45.63	45.63	21
	nodes/25	1	+ 0.03	+ 0.03	29.71	29.71	24
	nodes/∞	1	+ 0.01	+ 0.01	7.92	7.92	24
us67p	all commodities	1	+56.45	+56.45	100.00	100.00	30
	single/10	1	+33.17	+33.17	4.34	4.34	728
	single/15	1	+29.63	+29.63	3.66	3.66	861
	single/20	1	+27.41	+27.41	3.35	3.35	1133
	single/25	1	+25.22	+25.22	3.03	3.03	1191
	single/∞	80887	+ 0.64	+ 2.96	0.23	0.23	15
	nodes/10	1	+54.84	+54.84	51.11	51.11	91
	nodes/15	1	+64.62	+64.62	42.33	42.33	94
	nodes/20	1	+63.39	+63.39	32.84	32.84	131
	nodes/25	1	+63.57	+63.57	30.35	30.35	175
	nodes/∞	19410	+13.85	+15.56	2.99	2.99	249

Table 5.14: Different strategies to add commodities to the routing LP dynamically. The table shows the number of branch-and-cut nodes solved, the change of the root and final dual bound, the number of commodities in the LP after the root node and after 6h, and the number of generated metric inequalities. The choice of strategy does not really matter on g50p but it has a huge impact on us67p.

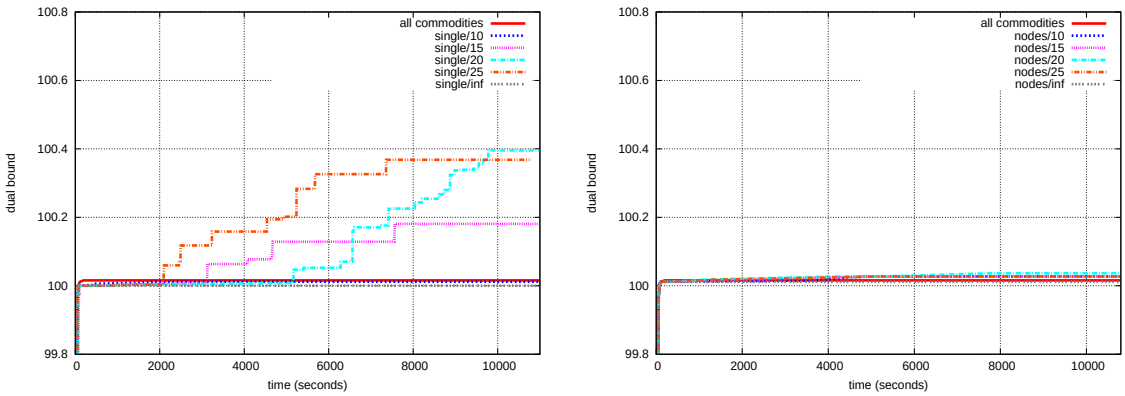
Table 5.14 summarizes the results. The first two columns indicate the name of the instance and the way commodities are added to the routing LP: all at once (*all commodities*), node-wise

(*nodes*), or commodity-wise (*single*). The number behind the strategy indicates the number of pricing iterations after which a new group of commodities is added. A value of infinity means that a new commodity is only added when the routing LP is optimally solved with respect to all already included commodities. The number of solved branch-and-cut nodes is given in the third column. The next two columns give the relative improvement of the dual bound at the root node and after three hours, compared to the dual bound just before the first metric inequality is generated (which is the same bound in all tests for one instance). The following two columns show the number of commodities in the routing LP at the end of the root node and after three hours as a percentage of the total number of commodities (1225 for g50p, 2211 for us67p). Eventually, the last column describes the number of generated metric inequalities.

The table shows that the results on the two instances are completely different. This reflects the different structure of these networks, as discussed below.

g50p On the g50p network, metric inequalities have basically no effect on the dual bound at the root node with any strategy. On this network, the cutset inequalities at the logical layer and the fixed-charge inequalities at the physical layer do most of the work; whether metric inequalities are generated in addition and in which way these are added does not make much difference for the root dual bound.

After the root node, however, the choice of strategy matters. Because of the smaller number of commodities in the LP, many more branch-and-cut nodes can be solved. Consequently, the dual bounds start to differ after the root node because of different branching decisions. Still, the differences are small. The number of generated metric inequalities is very similar for all strategies. The number of branch-and-cut nodes does not increase monotoneously when adding less commodities because the whole computation time is sometimes dominated by several hours needed to separate a single metric inequality. It is hard to determine in advance under which conditions this happens.



(a) Commodity-wise strategies.

(b) Node-wise strategies.

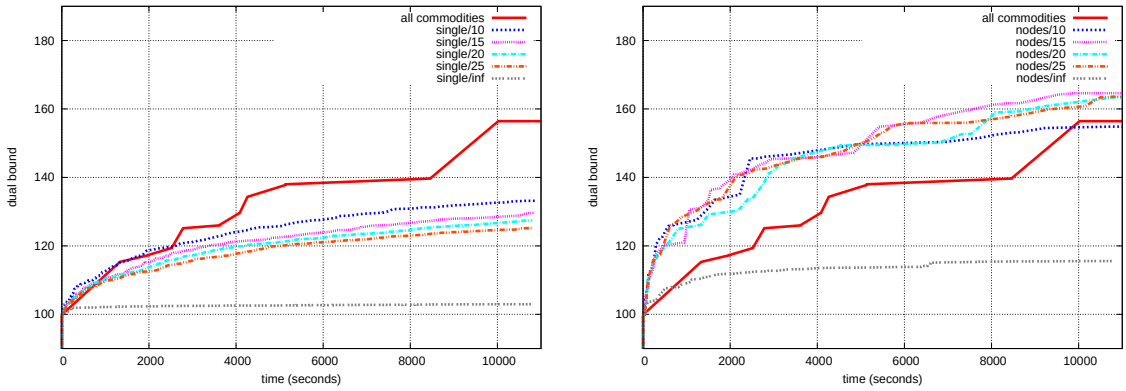
Figure 5.11: On g50p, metric inequalities do not have much effect with any considered strategy. The figures show the dual bound over time with different commodity-wise or node-wise strategies to add commodities dynamically to the routing LP.

Figure 5.11 illustrates the progress of the dual bound over time for the g50p instance with

different commodity- and node-wise strategies. Similarly to Table 5.14, all values are given in relation to the dual bound just before the first metric inequality is added. In both pictures, the solid red line represents the case where all commodities are included in the LP from the beginning. The increase of the dual bound in the commodity-wise cases are all caused by branching decisions after the root node; the dual bound scale shows that the strategies differ only marginally between the various strategies.

The observed effects can to some extent be explained by the structure of g50p. In this network, all demand values are roughly in the same order of magnitude. Consequently, only a small total amount of traffic is routed in the LP if only few commodities are added. With the extreme strategy *single/*∞, for example, the routing LP contains only five out of the 1225 commodities in the end, and the dual bound does not change by adding metric inequalities. Similarly, with the extreme node-wise strategy *nodes/*∞, the routing LP only contains the commodities starting or ending in Berlin, which is the node with largest emanating demand. Furthermore, both the physical and logical network are rather dense. The only effect of metric inequalities in this case is that some small amount of flow is shifted from one link to the other without significantly increasing the LP capacities. Furthermore, it sometimes takes several hours to separate a single metric inequality. The time is indeed spent in the LP solver, not in the pricing process. We observed that if new commodities are added from time to time, such stalling effects tend to occur less often.

us67p On the us67p instance, metric inequalities significantly contribute to the dual bound. As a result, the dual bound decreases notably if only information about few commodities is incorporated into the metric inequalities.



(a) Commodity-wise strategies.

(b) Node-wise strategies.

Figure 5.12: On the us67p, the choice of dynamic strategy matters. The figures show the dual bound over time with different commodity-wise or node-wise strategies to add commodities dynamically to the routing LP.

First, consider the commodity-wise strategies. Table 5.14 and Figure 5.12(a) show that all commodity-wise strategies lead to a weaker dual bound compared to the default setting with all commodities. With the extreme strategy *single/*∞, only five out of the possible 2211 commodities are included in the LP. Consequently, more than 80000 branch-and-cut nodes can be solved, but the generated metric inequalities are very weak. The final dual bound is

only 2.96% above the reference dual bound just before the first metric inequality is added. This strategy is indicated by the lowest curve in the left of Figure 5.12(a). Also with the other commodity-wise strategies, only a small number of commodities is added to the LP, and the dual bound is weak.

The node-wise strategies, in contrast, can improve the dual bound compared to the default setting. Figure 5.12(b) illustrates, however, that the extreme strategy *nodes/*∞ performs badly, and that a more conservative strategy is required to obtain a good dual bound. As can be seen from the table, all considered *nodes/k* strategies with finite k solve the root node only, similar to the default setting. Nevertheless, a better dual bound is obtained with only one third to one half of the possible commodities and with relatively few metric inequalities.

Such a result is possible because of the special structure of the us67p network. First, both the physical and logical network are sparse because of the large geographical distances between the nodes. Second, the demand values vary widely. Very few nodes dominate the whole traffic pattern: New York (16.7% of the total demand in the network), Los Angeles (7.8%), Toronto (7.1%), and Chicago (6.5%) are responsible for 38% of the total demand volume (see the network topology in Figure 5.3(a)). On the other hand, a large number of demands has negligible values. By considering only few demands with large demand values, the size of the LPs can significantly be reduced without losing too much information.

One might wonder why the root node is finished before all commodities are included in the LP, i.e., when either more violated metric inequalities should exist or feasibility of the last fractional LP capacities should have been tested with respect to all commodities. The reason is that Scip starts the branching process already when the progress in the dual bound caused by cutting planes gets too small. In a test where we disabled the corresponding Scip parameters, further metric inequalities were separated until the time limit without improving the dual bound. Hence, the default behavior of Scip makes sense.

Summarizing, we can conclude that adding commodities dynamically can be a useful approach if applied in a conservative way. This is especially true if few demands dominate the whole traffic pattern and if metric inequalities are important for the dual bound. In any case, the corresponding parameters have to be chosen specifically for each network instance.

Bottleneck LP, feasibility LP, or integrated MIP?

With metric inequalities, the choice the routing LP formulation, that is, whether a bottleneck variable is used or not, has consequences both on the behavior of the pricing algorithm and on the generation of metric inequalities, as explained in Sections 4.5.3 and 4.6.4. The advantage of the bottleneck formulation is that it fits better with column generation because the LP can be re-solved with the primal simplex algorithm after adding new variables, starting from the previous basis. In contrast, the feasibility formulation has to be solved with the dual simplex algorithm to obtain a dual ray in case of infeasibility. The drawback of the bottleneck formulation is that it may take a lot of time to minimize the bottleneck variable although it is already clear that its optimal value is positive.

If all variables are included from the beginning or if the branch-and-cut framework supports column generation in the master MIP (Scip does, Cplex does not), another important question is whether it is preferable to use Benders decomposition or to include the whole formulation in one large MIP.

We compared the three approaches on all test instances. The time limit was set to six hours for the large networks eu28, g50, and us67, and to three hours for the smaller networks.

On the two large protected instances g50p and us67p, the path-based pricer was used; all other instances were solved without column generation. We searched for metric inequalities every time an integer solution of the master MIP was found, and also to cut off fractional master MIP solutions at every node of depth 0, 5, 10, 15,... of the branch-and-cut tree. In preliminary tests, the fast *simple rounding* heuristic of SCIP (see Achterberg [3]) was called thousands of times and found lots of integral solutions of the master MIP. These all had to be tested for feasibility with respect to the routing constraints, but were never feasible. We thus disabled this heuristic to avoid that these useless feasibility tests dominate the whole computation time.

Table 5.15 on page 150 summarizes the results of our comparison. The first column shows the name of the instance and the strategy: 'n' for no metric inequalities (hardware and routing in the master MIP formulation), 'b' for the bottleneck approach described in Section 4.5.2, and 'f' for the feasibility approach from Section 4.5.3. The second to fifth column show the number of solved branch-and-cut nodes, the optimality gap in the end (defined as $(ub-lb)/ub$), the best solution, and the lower bound in the end. All upper and lower bounds are scaled such that 1 corresponds to the initial LP bound of the full formulation. The sixth and seventh column contain the lower bound at the end of the root node and the time needed to solve the root node (including the time for several sub-MIP-based heuristics called at the end of the root node). The next column shows the number of violated metric inequalities found. Eventually, the last two columns display the number of pricing iterations and the number of priced variables for g50p and us67p.

Impact on the branch-and-cut tree We first discuss the impact of the formulation on the branch-and-cut tree. The impact on column generation for the large instances g50p and us67p will be discussed later on.

Applying branch-and-cut to the complete formulation is clearly preferable to any metric inequality formulation. The gap and bound columns in Table 5.15 show that both the primal and the dual bounds are always as least as good with the integrated approach than with any decomposition approach, and that they are much better in most cases. This is also true for the dual bound at the end of the root node. In some cases, the final dual bound with metric inequalities is even smaller than the initial LP bound with the integrated formulation; extreme examples are eu28p and us67p.

With the integrated formulation, more branch-and-cut nodes can be solved than with any decomposition approach on 8 out of the 12 instances. The difference can be huge; on g17p, 11869 nodes are solved with the full formulation, compared to 1 and 377 nodes with the bottleneck and feasibility approach. A comparison of the dual bound columns shows that on most instances, the branch-and-cut part after the root node has little effect on the dual bound, even if hundreds of thousands of branch-and-cut nodes are solved. With metric inequalities, branching has even less effect than with the integrated MIP.

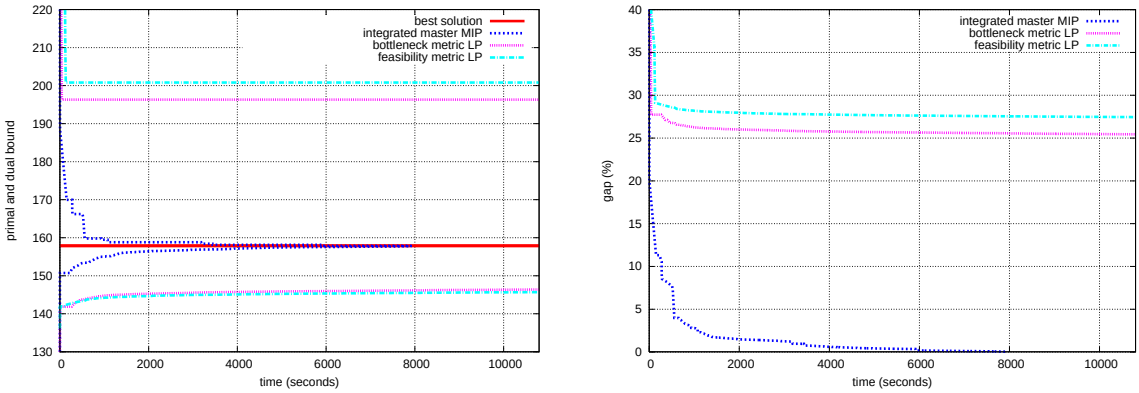
A comparison of the bottleneck and feasibility LP formulations does not exhibit any clear favorite. The dual bounds are always close to each other in both approaches, compared to the primal bound; also the primal bounds of both formulations differ only marginally. The better bounds are achieved with either approach in roughly half of the cases. Also the number of branch-and-cut nodes, the number of generated metric inequalities, and the time spent at the root node vary in both directions.

As an example, Figure 5.13(a) shows the primal and dual bounds over time on g17u

	B&C	final gap and bounds			root			pricing	
inst	nodes	gap (%)	ub	lb	lb	time	# MI	iter	vars
p12u/n	2318082	0.97	1.13	1.12	1.03	28	-	-	-
p12u/b	914629	3.71	1.15	1.10	1.02	24	16795	-	-
p12u/f	1221409	2.39	1.13	1.10	1.02	49	45822	-	-
p12p/n	45208	0.94	1.03	1.02	1.00	184	-	-	-
p12p/b	121	39.49	1.34	0.81	0.80	7594	46	-	-
p12p/f	428	17.97	1.09	0.89	0.89	5341	175	-	-
r15u/n	875829	0.20	1.04	1.04	1.04	62	-	-	-
r15u/b	637719	11.38	1.14	1.01	1.01	8	231981	-	-
r15u/f	529611	14.89	1.19	1.01	1.01	6	192134	-	-
r15p/n	170871	0.84	1.05	1.04	1.04	261	-	-	-
r15p/b	195157	15.87	1.19	1.00	0.99	23	70211	-	-
r15p/f	42955	13.40	1.16	1.00	1.00	466	45023	-	-
g17u/n	532071	0.00	1.58	1.58	1.51	228	-	-	-
g17u/b	250381	25.44	1.96	1.46	1.42	264	38971	-	-
g17u/f	211680	27.45	2.01	1.46	1.42	118	45268	-	-
g17p/n	11869	7.38	1.41	1.31	1.24	3942	-	-	-
g17p/b	1	41.12	1.88	1.11	1.11	10920	253	-	-
g17p/f	377	42.80	1.92	1.10	1.07	5208	369	-	-
eu28u/n	75190	9.65	1.21	1.09	1.03	2929	-	-	-
eu28u/b	14137	43.41	1.85	1.05	1.04	3217	5023	-	-
eu28u/f	27303	42.02	1.72	1.00	0.99	653	6279	-	-
eu28p/n	1	7.49	1.09	1.01	1.01	21600	-	-	-
eu28p/b	1	48.01	1.31	0.68	0.68	21600	15	-	-
eu28p/f	6	44.34	1.31	0.73	0.73	18840	52	-	-
g50u/n	14	27.70	3.05	2.20	2.20	14340	-	-	-
g50u/b	100	62.63	4.74	1.77	1.77	7130	100	-	-
g50u/f	218	61.90	4.64	1.77	1.77	5749	272	-	-
g50p/n	27	56.60	3.19	1.39	1.38	1182	-	316	28655
g50p/b	11	56.71	3.15	1.36	1.36	17040	27	87	15078
g50p/f	1	56.81	3.15	1.36	1.36	23160	17	1066	36460
us67u/n	2142	14.03	1.30	1.11	1.10	12540	-	-	-
us67u/b	1	66.46	3.16	1.06	1.06	21600	1221	-	-
us67u/f	1	69.00	3.16	0.98	0.98	21600	184	-	-
us67p/n	290	24.26	1.34	1.02	1.02	485	-	896	49624
us67p/b	1	55.45	1.53	0.68	0.68	21600	39	93	49741
us67p/f	1	57.14	1.51	0.65	0.65	21600	44	779	142873

Table 5.15: Solving the full MIP formulation ('n') is clearly better than Benders decomposition with the bottleneck ('b') of feasibility ('f') routing LP formulation. The time limit was set to 3h for p12, r15, and g17, and to 6h for the larger networks. The table shows the number of solved nodes, the gap and the bounds after the time limit, the dual bound and the time at the end of the root node, the number of separated metric inequalities, and the number of pricing iterations and dynamically generated variables.

together with the optimal solution value (solid red line). The bounds are scaled such that 100 corresponds to the initial lower bound in the full formulation. The figure shows that the optimal solution is found in slightly more than two hours with the full master MIP approach, whereas the two Benders decomposition curves do not differ significantly and start to stall after half an hour with an optimality gap of 25% and 27%, respectively. Figure 5.13(b) shows the corresponding development of the optimality gap over time for the three strategies.



(a) Primal and dual bound over time

(b) Optimality gap over time.

Figure 5.13: The full MIP formulation on g17u is solved to optimality within two hours, whereas a Benders decomposition approach with the bottleneck or feasibility formulation stalls at 25% and 27% gap.

The possible reasons for the weak bounds obtained by decomposition are manifold. First, the variables are only weakly coupled: flow variables $f^k_{\ell,ij}$ are used to generate metric inequalities; these are defined in the space of logical link capacity variables y^m_ℓ, which in turn influence the node capacity variables x^m_i and the physical link capacity variables z_e. In an integrated MIP, flow and logical link capacity variables are directly coupled by the capacity constraints. Second, SCIP can do much better presolving if all information is available in the MIP from the beginning. Third, the integrated approach allows flow-cutset inequalities to be used, and also SCIP can generate cutting planes with flow and capacity variables.

Our results are in line with the ones of Bienstock et al. [27] in the context of single-layer networks, who also observed much longer running times and weaker bounds with a decomposed formulation. They had, however, compared a full MIP with a link-flow formulation to a Benders decomposition approach with a path-flow formulation in the routing LP. It was thus not clear which of these changes caused the differences in running time. Our tests indicate that it is mainly the decomposition which makes the difference.

Combination with column generation On the g50p network, the primal and dual bounds are very similar with all three approaches. Our previous tests have shown that metric inequalities do not have much effect on this instance and the dual bound is mostly driven by the cutset inequalities. The built-in heuristics of SCIP do not find any solutions; the primal bound is thus determined by the result of our routing heuristic FINDFEASIBLEROUTING from Section 4.3.2 and the hardware installation heuristic INSTALLCAPHEUR from Section 4.3.1, which together compute roughly comparable feasible solutions in all three cases within the first cutting plane iterations. The computation time with the decomposition approaches is

largely dominated by the time needed to solve the routing LP to separate fractional LP solutions. In some cases, solving this routing formulation to optimality with column generation takes several hours. This time is mostly spent in the LP solver; the pricer itself is fast, and few (typically less than 10) LP solving and pricing iterations are needed to solve a given routing LP to optimality with respect to all variables. In another test where the metric inequality separator was only called as an incumbent callback for integer solutions but not to separate fractional capacities, 15278 and 16183 branch-and-cut nodes were solved with the bottleneck and feasibility approach, respectively; nevertheless, the primal and dual bounds were very similar to those in the table.

The impact of the choice of formulation on column generation strongly depends on the parameters. The last three columns of Table 5.15 show that with the feasibility LP, more than twice as many variables are generated as in the bottleneck formulation but less violated metric inequalities are found. The number of pricing iterations is even multiplied by a factor of more than 12. Hence, more variables and many more pricing iterations are necessary to separate a single metric inequality in the feasibility approach. This result, however, does not seem to be stable with respect to changing parameters. In the above described test where metric inequalities were generated only to separate infeasible integer solutions, the number of pricing iterations was much larger with the feasibility version than with the bottleneck version (23446 compared to 129), but only about half the number of variables were generated, contrary to the results described in the table. As the computation time is mostly dominated by the separation of metric inequalities, changes to the corresponding parameters may have a huge impact on the performance. We therefore cannot give any general result here.

On the us67p network, the gap and bound columns show that the integrated MIP version performs much better than both decomposition approaches. Among the two routing LP formulations, there is no clear favorite. The bounds are similar, and the computation does not exceed the root node in both cases. As on the g50p network, the feasibility formulation needs more pricing iterations and generates about thrice as many variables as the bottleneck formulation. In this case, slightly more metric inequalities are generated by the feasibility approach. Nevertheless, the time needed to solve the routing LPs dominates the overall computation time and strongly depends much on the parameters for the metric inequality separator.

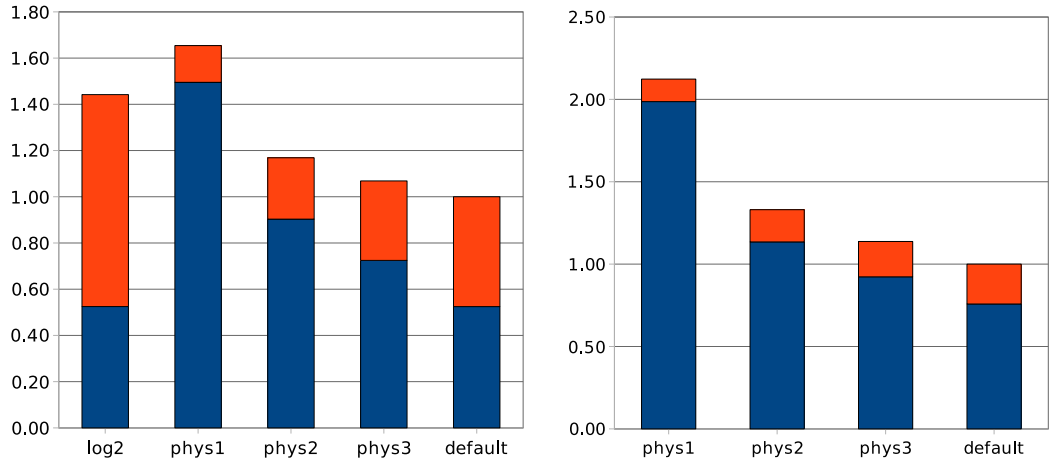
5.3 Parameter studies with problem restrictions

With the algorithmic ingredients investigated in the previous sections, we are now able to perform parameter studies on large survivable two-layer networks. As an example, we now present two studies. First, we impose hop limits on the routing in the large networks g50p and us67p to investigate the impact of grooming on the network cost. Second, we apply different variants of the core network heuristic described in Section 4.2.3 to the g50p network to see which kind of grooming is the most important one. All restrictions investigated in this section are implemented by disallowing certain flow variables, as described in Section 5.1.

Even though we cannot solve these problems to optimality, the obtained bounds can give an intuition of how strong a restriction is. We observed that the relative contribution of logical link cost, physical link cost, and node cost in the fractional LP solution after a few cutting plane iterations are very similar to the relative contributions in good feasible solutions. This suggests that the change in the LP bound caused by a given restriction of the solution space

is a good indicator of its impact on the optimal network cost.

5.3.1 Physical and logical hop limits



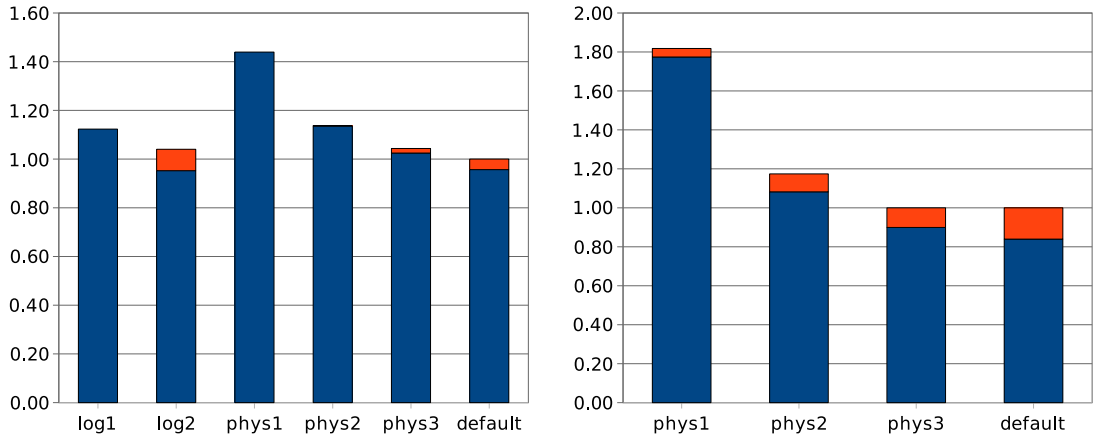
(a) g50p (b) us67p

Figure 5.14: Allowing only very short logical links is a severe restriction; a physical hop limit of 3 leads to acceptable solutions. The figure shows the primal and dual bounds of g50p and us67p obtained within 12h under different logical and physical hop limit restrictions. Value 1 corresponds to the best known upper bound (column *default*).

For the two large networks g50p and us67p with protected demands, Figure 5.14 shows the primal and dual bounds obtained in a 12-hour test run with different physical and logical hop limits. Each column shows the primal and dual bound for a specific setting. The results for the g50p network are shown in Figure 5.14(a). The columns represent a logical hop limit of 2 (that is, traffic can be groomed at most once on the way from its source to its destination), physical hop limits 1, 2, and 3 (only the subset of logical links up to that length are admissible), and the best known solutions and dual bounds (not only from this run) without hop limits. With a logical hop limit of 1, where demand can only be routed on direct logical links from their source to their destination without intermediate grooming, this instance has no solution with the given set of logical links. Figure 5.14(b) shows the corresponding columns for us67p. This instance is infeasible even with a logical hop limit of 2 because of the sparse logical graph resulting from the large geographical distances.

The figure shows that forcing demands to be groomed is expensive. With a physical hop limit of 1 (corresponding to a fully opaque network), logical links must be terminated and re-created at each intermediate node of a routing path. If a large part of the total network cost is caused by the interfaces at the end-nodes of logical links, as in this case, such a setting leads to high network cost. The obtained lower bound with hop limit 1 on g50p is about 50% higher than the best known solution without hop limits. On the us67p network, the cost of an opaque network is even more than twice as high than in the default settings. For larger hop limits, no definitive answer about the cost can be given, but both upper and lower bounds decrease significantly as the hop limit is more and more relaxed.

Logical hop limits seem to restrict the problem less severely. Although the solution obtained for g50p with a hop limit of 2 is rather expensive, the dual bound is very close to the one without any restrictions. Indeed, in most of our solutions for this network, the vast majority of the demands is routed on a direct logical link, some demands are routed over two links, and only very few demands use more than two links.



(a) eu28u (b) us67u

Figure 5.15: Whereas logical hop limits only slightly increase the network cost, small physical hop limits lead to high network cost. The figure shows the primal and dual bounds after 3h with different logical and physical hop limits for eu28u and us67u. Value 1 corresponds to the best known solution.

Experiments on the unprotected versions g50u and us67u and on smaller networks have confirmed these observations. Figure 5.15 shows the corresponding results for eu28u and us67u, where the optimality gaps are much smaller. First, Figure 5.15(a) shows that logical hop limits of 1 and 2 only slightly increase the network cost. Second, the cost of the network as well as the dual bounds quickly go down as the physical hop limit increases. Figure 5.15(b) illustrates that the best known solution for us67u has been found using only logical links consisting of at most three physical links. A hop limit which is not too restrictive can thus be useful to obtain good solutions in shorter time. On the unprotected networks, several of the restricted problems could be solved to optimality or near-optimality within the imposed time limit of three hours. The eu28u network in the fully transparent version, i. e., with only direct logical links, could even be solved to optimality within one minute.

5.3.2 Core network heuristics

On the g50p network, we now investigate which kind of grooming is the most important one. To this end, we define a core network and impose different restrictions on the routing, as described in Section 4.2.3. Recall that the core heuristics cannot be readily applied to the us67p network because it is too sparse.

We consider the following parameters:

- *only direct* or *all*: With *only direct*, traffic from non-core nodes must be routed on direct logical links to the core network or to the destination node. Otherwise, it can be groomed at intermediate non-core nodes on the access path.

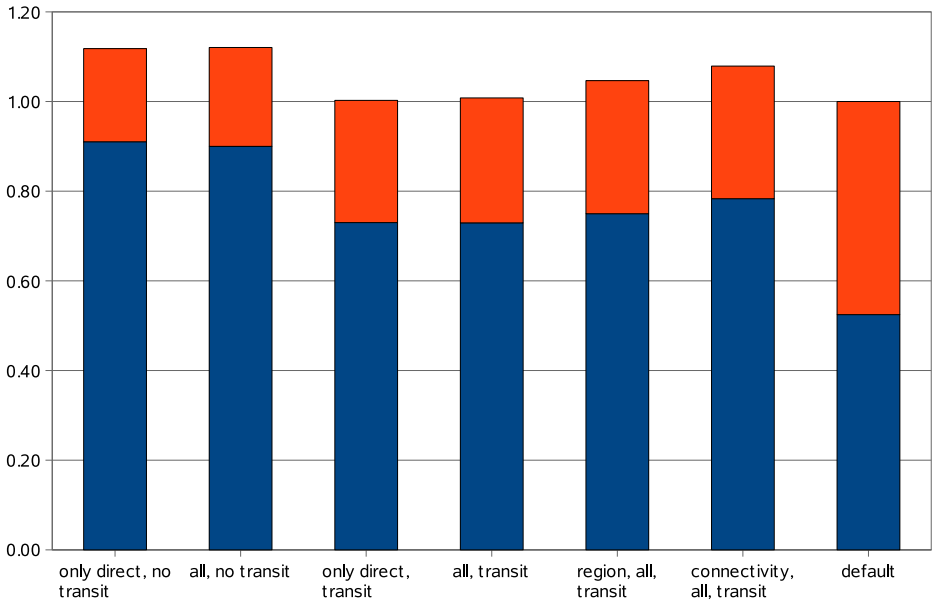


Figure 5.16: Disallowing transit traffic through core nodes is a stronger restriction than disallowing grooming on access paths to the core network. The figure shows the primal and dual bounds obtained within 12h with different core network restrictions on g50p.

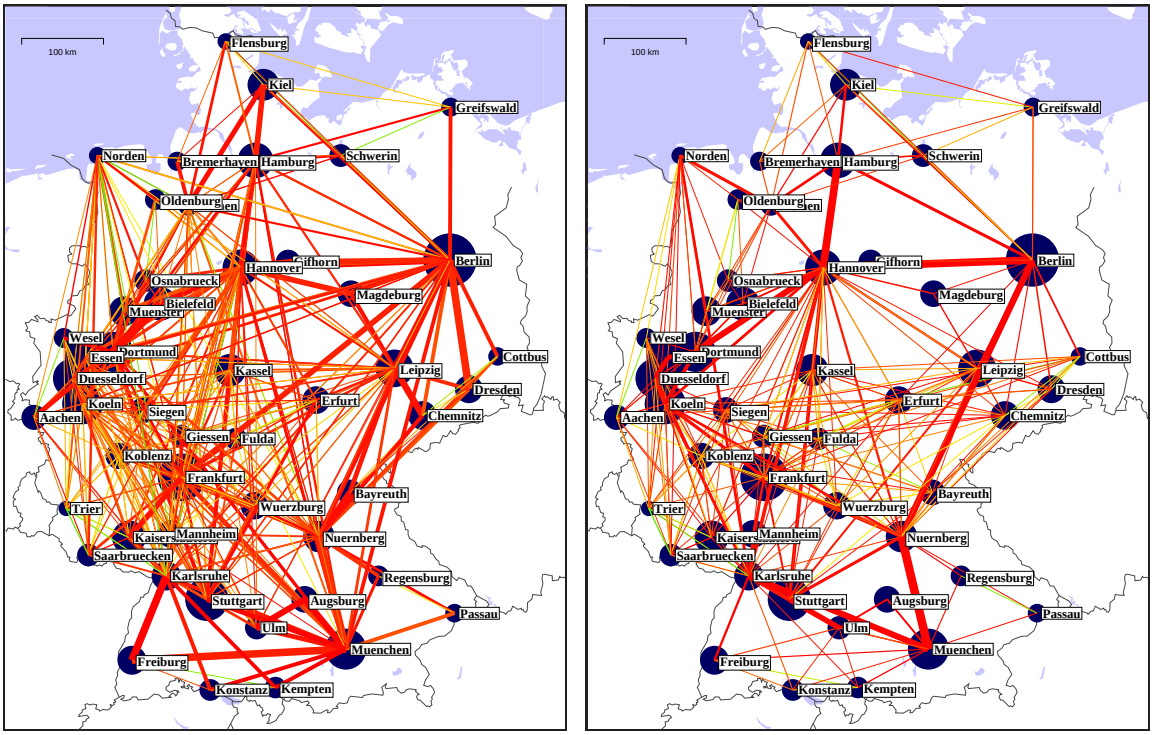
- *transit* or *no transit*: In the *transit* setting, traffic may be routed on logical links passing through core nodes. Otherwise, it must be groomed at every intermediate core node.

These parameters are combined in all four possible combinations.

Figure 5.16 shows the lower and upper bounds obtained within 12 hours for the four settings, where the nodes of the g17 network have been used as the core subnetwork of g50. The fifth and sixth column represent the bounds with the *all/transit* setting and a core network with 17 nodes computed via the region demand and the node connectivity strategy explained in Section 4.2.3. As a reference, the last column shows the best known lower and upper bound in the default setting without any core network restrictions. All values are scaled such that a value of 1 corresponds to the network cost of the best known solution.

First, the figure shows that all computed primal bounds are within 13% of the best known solution value. As the difference to the global dual bound (from the *default* column) is large, we do not know how good these solutions really are; from investigating the obtained network configurations, we expect that they can still be improved. In fact, the best known solution which we know for the g50p instance has been computed by the hardware installation heuristic InstallCapHeur (see Section 4.3.1) in the core network setting *all/transit* with a pricing initialization from previously computed good solutions.

A comparison of the first two columns with the third and fourth column suggests that forcing traffic to be groomed at every intermediate core node (*no-transit*) is a relatively strong restriction which raises the network cost significantly. This is consistent with the observation from the previous section that imposing intermediate grooming causes lots of additional interface cost.



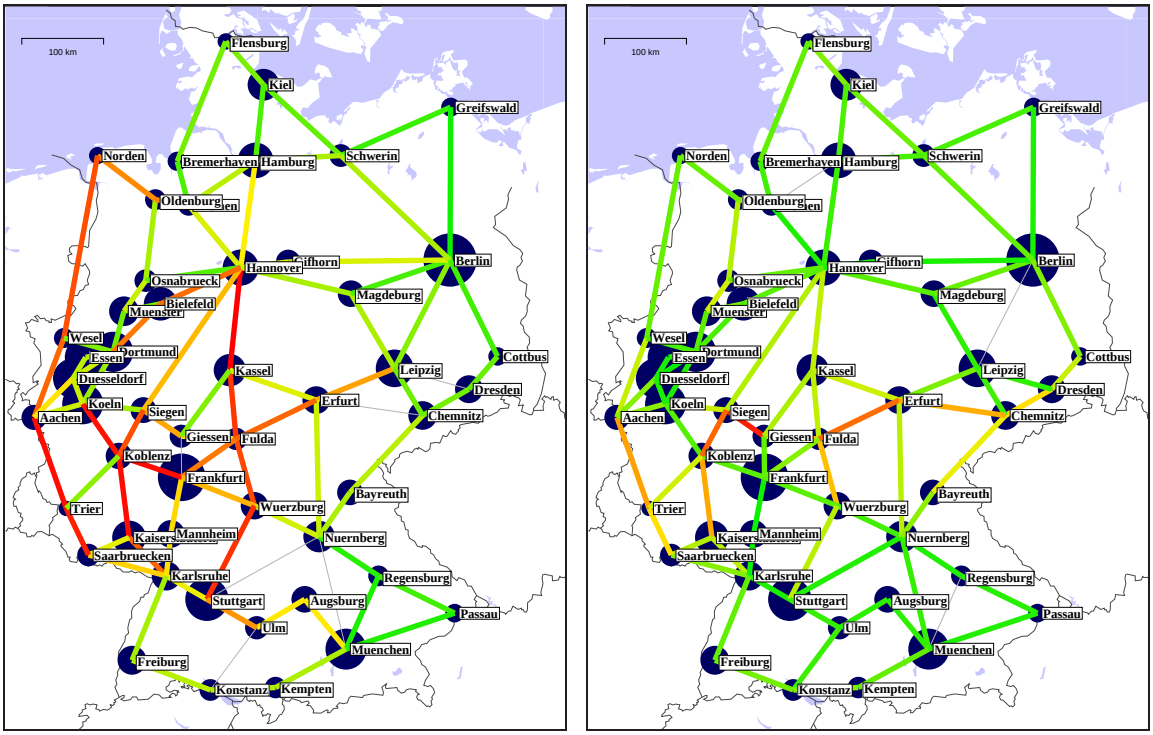
(a) With transit through core nodes.

(b) Without transit through core nodes.

Figure 5.17: Forcing grooming at core nodes leads to a much sparser logical network structure. The figures show the logical layer of the best solutions computed with the g17 core network, where transit through core nodes is either allowed (left) or not (right). Grooming on access paths is allowed in both cases. The width of a link represents its capacity; the color represents its load (green means low use, red means high use).

To show the impact of forcing grooming at core nodes on the network structure, Figure 5.17 illustrates the logical layer of the best computed solutions with the g17 core network and the settings *all/transit* (left) and *all/no transit* (right). The width of a link corresponds to its capacity. Its color represents the load, i. e., the ratio between the flow and the capacity on that link. Red links are highly loaded, whereas green links carry only little flow compared to the capacity. The size of the nodes represents the emanating demand value. The figure illustrates that disallowing transit logical links through core nodes heavily restricts the solution space. The network in Figure 5.17(b), where transit through core nodes is forbidden, is much sparser than the one in Figure 5.17(a) (10.7% of all admissible logical links are used compared to 15.3%), and the logical links tend to connect only geographically close nodes. The right picture shows a typical core network structure where important core nodes are connected by high-capacity links, whereas most other links only have low bandwidth.

Figure 5.18 shows the physical layers of these solutions. In both solutions, at most one capacity module is installed on each physical link; some physical links could be saved completely. The colors, however, illustrate that disallowing transit traffic through core nodes saves physical bandwidth: in Figure 5.18(a), where transit is allowed, several physical links are highly loaded. In Figure 5.18(b), where transit is forbidden, the logical links tend to be



(a) With transit through core nodes.

(b) Without transit through core nodes.

Figure 5.18: With transit through core nodes, the physical links are much more loaded on average. The figure shows the physical layer of the best solutions computed with the g17 core network, where transit through core nodes is allowed (left) or not (right). Unused links are shown as thin gray lines.

shorter; consequently, the physical links are much less loaded. Indeed, the average number of logical links traversing a given physical link is 34.1 if transit is allowed, compared to only 19.3 if it is forbidden.

In contrast to disallowing transit trough core nodes, grooming at intermediate non-core nodes on access paths has nearly no effect on the network cost in our test. We observed that even if grooming on access paths is allowed, most demands are routed directly from the source or destination node to the core network. The structure of the computed solutions is very similar in both cases, regardless of whether grooming on access paths is allowed or not.

The region demand and connectivity strategies yield solutions which are 5–10% above the best known solution. Also the dual bounds are slightly higher than with the g17 core network. Experiments with different core network sizes did not yield better solutions.

5.4 Summary of the computational results

In this chapter, we have investigated the impact of the algorithmic techniques from Chapter 4 on six test networks with realistic background and different structure. We have considered both the protected and the unprotected version of each network. Our main results can be summarized as follows:

Problem-specific presolving Even though SCIP already does a lot of generic presolving, additional problem-specific presolving techniques can make the problem significantly easier to solve. Already the basic presolving steps close up to 89% of the optimality gap on our instances; they are particularly successful in connection with survivability constraints. In addition, the LP-based probing for protected demands removes up to 45% of all flow variables without changing the set of feasible and optimal solutions. The reduced probing version, where no variable bounds are computed, needs up to 92% less computation time than the full probing version but leads to almost the same LP bounds.

Primal heuristics Our problem-specific primal heuristics can help a lot to reduce the optimality gaps or to find feasible solutions at all. In particular, we can compute feasible network configurations for the five largest instances based on LP information; all of SCIP's generic heuristics fail on these instances. Our most successful heuristics are the fast combinatorial hardware installation heuristic INSTALLCAPHEUR, which constructs feasible solutions for all considered network instances, and the sub-MIP heuristic REROUTEMIP, which identifies the best solution within the time limit on half of the tested instances. Good solutions are often found by a combination of construction and improvement heuristics.

Cutting planes We have tested a variety of problem-specific cutting planes acting either on one of the layers only or on both layers simultaneously. On our test instances, the node degree and tree inequalities at the physical layer turned out to be the most important cuts; they close large parts of the optimality gaps. In contrast, the benefit of the logical layer cutset and flow-cutset inequalities is small compared to the effect of our presolving, especially if survivability constraints are imposed. The extended link failure cutset and flow-cutset inequalities, which take the protection requirements from the physical layer into account, have only slightly more impact on the dual bound than their single-layer counterparts. Only a carefully selected subset of the violated cutting planes should be added to the LP relaxation.

Column generation On two large network instances g50p and us67p with protected demands, we have compared the enumeration-based and the path-based pricing algorithm from Section 4.6. The tests revealed that the path-based pricer is clearly preferable, as it achieves almost the same dual bounds with much less variables and in less time. With column generation, we can now apply LP- and MIP-based techniques to compute feasible solutions and meaningful lower bounds for these large instances, which was not possible before.

Metric inequalities For the large network instances, solving the routing LP in a Benders decomposition approach may take several hours. We have tested various strategies to add commodities to the LP dynamically on g50p and us67p. It turned out that such a dynamic strategy can significantly improve the dual bound at the root node if the strategy is chosen according on the structure of the specific network instance. A second test on all network instances revealed that modeling and solving everything in one large MIP is clearly superior to using Benders decomposition with any of the two considered routing LP formulations.

Optimality gaps The network sizes and survivability constraints considered in this thesis make the planning problem much more difficult to solve than any other multi-layer problem considered in the literature. With all the algorithmic techniques mentioned above, we can

solve most of our small network instances with up to 17 nodes to near-optimality. Furthermore, we can now apply LP and MIP techniques to compute feasible network configurations and useful dual bounds for large instances with up to 67 nodes, which was not possible before.

We can construct feasible network configurations for the large network us67 both with and without protection, and show that they are at most 25% and 14.1% away from an optimal solution, respectively (see Table 5.15, row 'n'). Such a large network with protection requirements has not even been considered in the multi-layer literature before. The large and dense network g50 is still a challenge; the gaps are 57% and 28% with and without protection. Nevertheless, we can now compute solutions and dual bounds for this instance, which would not have been possible without the techniques developed in this thesis. The medium instances eu28u and eu28p, as well as g17p, can be solved within 10% of optimality. Except for the latter instance, the optimality gap within three hours on all of our small instances with at most 17 nodes is below 1%.

Parameter study Our tests revealed that on the large network instances, some techniques and parameters have to be applied in an instance-dependent way to obtain good results. We have used the developed solution methodology to investigate the effect of various grooming restrictions on the total network cost. We found that the cost structure of LP solutions is similar to the cost structure in feasible network configurations. A study on the large protected networks showed that hop limits can be a severe restriction, especially when applied to the routing of logical links in the physical layer. This result has been confirmed by experiments on smaller and unprotected instances. With the core network heuristics, we found that forcing demands to be groomed at certain nodes significantly augments the network cost. On the other hand, the us67 network illustrates that some grooming may be necessary to obtain feasible solutions if not all demand end-nodes are connected by a direct logical link.

Chapter 6

Conclusions

6.1 Summary

In this thesis, we have presented mathematical models and algorithmic techniques for the integrated planning of two telecommunication network layers with survivability constraints. They deal with a multi-layer network design problem arising in different technological contexts, such as in MPLS-over-WDM, SDH-over-WDM, or ATM-over-SDH networks.

We have introduced different mixed-integer programming formulations to model a two-layer network with or without survivability constraints. The presented models cover more practical side constraints than any previous model from the literature. In particular, they comprise node hardware and protection of a logical layer routing against physical node or link failures. Contrary to many other models from the literature, our formulations are designed to be computationally tractable by a branch-and-cut-and-price framework even on large networks with survivability requirements.

Multi-layer networks can be modeled in many different ways depending on the specific planning context. In addition to a detailed literature review on previous approaches for integrated multi-layer network design, we have discussed different modeling alternatives for various parts of a network (nodes, logical links, physical links, survivability, ...) and compared different formulations for a logical layer routing under multi-layer survivability constraints.

On the algorithmic side, we have introduced a variety of problem-specific techniques to be used within a branch-and-cut-and-price framework. We have evaluated their performance on six structurally different networks with up to 67 nodes and realistic capacity and cost structures, both with and without inter-layer protection requirements. These networks have been solved in various settings using the branch-and-cut-and-price framework SCIP with and without our techniques. Our main findings can be summarized as follows:

- Our LP-based probing procedure removes 9–45% of all flow variables on our test networks with protection without losing optimality information. A modified version of the probing technique, which only tests whether a flow variable can be positive or not, saves between 41% and 92% of the presolving computation time on almost all of our instances.

- We have introduced a variety of primal heuristics, most of which are based on LP information from the branch-and-cut tree. Some of them solve a restricted version of the original problem as a sub-MIP. On almost all of our smaller network instances, our heuristics reduce the optimality gap. On the large network instances, our heuristics are

the only ones to compute network configurations, whereas SCIP alone fails to identify any feasible solutions. The INSTALLCAPHEUR and REROUTEMIP heuristics are particularly successful.

- To raise the lower bound on the total network cost, we have extended various kinds of cutting planes known from single-layer planning to the multi-layer context. This includes cutset and flow-cutset inequalities at the logical layer and different connectivity cuts at the physical layer. To investigate why the logical layer cuts have only little impact, we have extended them such as to take the inter-layer protection requirements into account. Although these extended cuts have more impact on the dual bound than their basic versions, the progress is small. We found that only a carefully selected subset of all identified violated cutting planes should be added to the linear programming relaxation.

- To deal with large networks, we have investigated two ways of generating flow variables dynamically. The path-based pricing algorithm turned out to be clearly better than the enumeration-based one. The column generation technique allows us to apply many linear and integer programming techniques also to large multi-layer networks with more than 50 nodes and protection requirements. In particular, we can compute network configurations and meaningful dual bounds for these large network instances, which was not possible before.

- Besides a standard MIP, we have also considered a Benders decomposition approach where the routing variables and constraints are moved to a separate linear program. This routing LP is considered in two different formulations. To reduce the LP solving times for large network instances, we have developed various strategies to add commodities to the routing LP dynamically. Such a dynamic strategy can significantly improve the dual bound at the root node if it is chosen according to the structure of the specific network instance. Another computational study revealed that running a branch-and-cut-and-price algorithm on the complete formulation leads to much better results than the decomposition approach. The latter should thus only be used if the branch-and-cut framework does not support column generation in the master MIP.

Using all these algorithmic techniques, we can solve most of our small network instances with up to 17 nodes up to near-optimality with a gap below 1%. Furthermore, our methods make it possible to apply LP and MIP techniques to compute feasible network configurations and useful dual bounds for large instances with 50 and 67 nodes, which was not possible before. For the large and dense network g50, the gaps are 57% and 28% with and without protection. For the 67-node network us67, we can compute solutions which are at most 25% and 14.1% away from the optimal value with and without protection, respectively. Even though these optimality gaps seem rather big, it is a great achievement that we can compute network configurations and non-trivial dual bounds at all. Such large multi-layer networks with realistic survivability constraints and all possible routings at the logical layer have not even been considered in previous publications. With an out-of-the-box MIP solver, even solving the LP relaxation for such large networks is out of question because of the huge amount of memory and time needed. On the European eu28 network with 28 nodes, we achieve optimality gaps below 10% both with and without protection requirements.

Our results make it possible to perform useful parameter studies on large networks. As an example, we have investigated the impact of different routing restrictions and core network

heuristics on the total network cost. The methods and the code developed in this thesis have been used in projects with Nokia-Siemens Networks as well as in the EIBONE project to investigate the influence of various planning parameters on the total network cost in different network architectures [30].

6.2 Further research directions

The area of multi-layer network design problems is not closed with this thesis. Even though our achievements allow us to solve small to medium two-layer network design instances to near-optimality and to apply branch-and-cut techniques to large instances at all, the latter are still challenging, especially if survivability constraints are imposed.

Resilience of a network against equipment failures is an important issue in practice. Many technological concepts have been proposed to ensure survivability either in one of the layers or in both layers simultaneously. The literature on multi-layer network design, however, provides only very few mathematical models to deal with survivability issues. It is not easy to formulate a realistic survivability model which is computationally tractable without severely restricting the set of admissible routings. The models and solution methods presented in this thesis are a first approach to take survivability into account already during the design phase of a multi-layer network. Nevertheless, further research is needed to find models that scale better with the network size.

To address large survivable network instances, instance-specific algorithm parameters and manual interaction are still necessary. This is in contrast to the small and medium instances, which could solved with one set of standard parameters for all instances using the methodology developed in this thesis. On the large instances with 50 nodes or more, the LP solving times are sometimes so long that the effect of a single wrong parameter value can dominate the whole computation time. In addition, the impact of several parameters, like those used to control the separation of metric inequalities, strongly depends on the network structure. The best solution for g50p has been obtained via a manually defined core network and a set of flow variables that has been initialized with previously computed good solutions. Although such an approach is suitable to investigate the structure of a specific network during a series of computational studies, a good set of standard parameters would be preferable.

In order to improve the performance of the branch-and-cut approach on large networks, more mathematical research is needed. On our large test instances, we currently do not know whether the remaining optimality gap is due to the primal bound, the dual bound, or both. From our experience with smaller network instances and by investigating the best known solutions, we expect that the primal bounds can still be improved. It would certainly be useful to combine several of our heuristics in a feedback loop. In addition, fast exact or approximative algorithms are needed to solve important two-layer routing subproblems in connection with survivability, such as the ones described in Section 3.4. Such algorithms could replace linear programs in various places of our solution approach.

On the dual side, very little is known yet about the facial structure of the multi-layer network design problem. It is not clear which of the many polyhedral results from single-layer network planning can be transferred to the multi-layer case, and which ones have to be extended. From first studies in this direction [96, 114], we can infer that although it is not easy to find algorithmically useful two-layer cutting planes, the underlying polyhedral structure offers many possibilities for future research.

Bibliography

[1] Eibone – Efficient Integrated Backbone. http://www.bmbf.de/de/6103.php, 2005–2008. Germany government BMBF project.

[2] NOBEL – Next generation Optical networks for Broadband European Leadership. http://www.ist-nobel.org, 2004–2007. EU project.

[3] T. Achterberg. *Constraint Integer Programming*. PhD thesis, Technische Universität Berlin, 2007. http://opus.kobv.de/tuberlin/volltexte/2007/1611/.

[4] T. Achterberg and T. Berthold. Improving the feasibility pump. *Discrete Optimization*, 4(1):77–86, 2007.

[5] R. Ahuja, T.A. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[6] R. Ahuja, K. Jha, and J. Liu. Solving real-life railroad blocking problems. Working Paper, University of Florida, Gainesville, FL, USA, September 2004.

[7] S. Arakawa, M. Murata, and H. Miyahara. Functional partitioning for multi-layer survivability in IP over WDM networks. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Series A (UK)*, pages 2224–2233, October 2000.

[8] J. Armitage, O. Crochat, and J.-Y. Le Boudec. Design of a survivable WDM photonic network. In *Proceedings of the 16th IEEE Infocom 1997, Kobe, Japan*, pages 244–252, April 1997.

[9] A. Atamtürk. On capacitated network design cut-set polyhedra. *Mathematical Programming*, 92:425–437, 2002.

[10] A. Autenrieth. Differentiated multilayer resilience in IP over optical networks. In *Scuola Superiore G. Reiss Romoli, L'Aquila, Italy*, TU München, July 2002.

[11] G. Baier, T. Engel, A. Autenrieth, and P. Leisching. Mehrperiodenplanung optischer Transportnetze. In *7. ITG-Fachtagung Photonische Netze, Leipzig, Germany*, volume 193, pages 153–160. VDE-Verlag, April 2006.

[12] A. Balakrishnan, T.A. Magnanti, J. Sokol, and Y. Wang. Modeling and solving the single facility line restoration problem. Technical Report OR 326-98, MIT Press, 1998.

[13] A. Banerjee, J. Drake, J. Lang, B. Turner, K. Kompella, and Y. Rekhter. Generalized Multiprotocol Label Switching: An overview of routing and management enhancements. *IEEE Communications Magazine*, 39(1):144–150, January 2001.

[14] D. Banerjee and B. Mukherjee. Wavelength-routed optical networks: Linear formulatation, resource budgeting tradeoffs, and a reconfiguration study. *IEEE Transactions on Networking*, 8(5):598–607, October 2000.

[15] F. Barahona. Network design using cut inequalities. *SIAM Journal on Optimization*, 6:823–837, 1996.

[16] C. Barnhart, H. Jin, and P. Vance. Railroad blocking: A network design application. *Operations Research*, 48(4):603–614, July/August 2000.

[17] R. Battiti and M. Brunato. Reactive search for traffic grooming in WDM networks. *Lecture Notes in Computer Science*, 2170:56–66, 2001.

[18] P. Belotti and F. Malucelli. Multilayer network design: a row-column generation algorithm. In *Proceedings of the 2nd International Network Optimization Conference (INOC 2005), Lisbon, Portugal*, volume 3, pages 422–427, 2005.

[19] P. Belotti, A. Capone, G. Carello, F. Malucelli, F. Senaldi, and A. Totaro. Design of multi-layer networks with traffic grooming and statistical multiplexing. In *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, 2007.

[20] P. Belotti, A. Capone, G. Carello, and F. Malucelli. Multi-layer MPLS network design: the impact of statistical multiplexing. *Computer Networks*, 52(6):1125–1342, 2008.

[21] J. Benders. Partitioning procedures for solving mixed-variable programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

[22] J.-C. Bermond and S. Ceroi. Minimizing SONET ADMs in unidirectional WDM rings with grooming ratio 3. *Networks*, 41(2):83–86, 2003.

[23] R. Berry and E. Modiano. Reducing electronic multiplexing costs in SONET/WDM rings with dynamically changing traffic. *IEEE Journal on Selected Areas in Communications*, 18(10):1961–1971, 2000.

[24] T. Berthold. Primal heuristics for mixed integer programs. Diploma thesis, Technische Universität Berlin, September 2006.

[25] D. Bienstock and O. Günlük. Capacitated network design – polyhedral structure and computation. *INFORMS Journal on Computing*, 8(3):243–259, 1996.

[26] D. Bienstock and G. Muratore. Strong inequalities for capacitated survivable network design problems. *Mathematical Programming*, 89:127–147, 2001.

[27] D. Bienstock, S. Chopra, O. Günlük, and C.Y. Tsai. Mininum cost capacity installation for multicommodity flows. *Mathematical Programming*, 81:177–199, 1998.

[28] A. Bley and M. Pattloch. Modellierung und Optimierung der X-WiN Plattform. *DFN-Mitteilungen*, 67:4–7, 2005.

[29] A. Bley and A. Zymolka. Planung kostenoptimaler Informations- und Kommunikations-Infrastrukturen. In *Proceedings of 8th Kasseler Symposium Energie-Systemtechnik: Energie und Kommunikation*, pages 61–77, Kassel, November 2003.

[30] A. Bley, U. Menne, R. Klähne, C. Raack, and R. Wessäly. Multi-layer network design – A model-based optimization approach. In *Proceedings of the 5th Polish-German Teletraffic Symposium 2008*, pages 107–116, Berlin, Germany, 2008.

[31] R. Borndörfer, M. Grötschel, and M. Pfetsch. A column-generation approach to line planning in public transport. *Transportation Science*, 41(1):123–132, 2007.

[32] W. Cho, J. Wang, and B. Mukherjee. Improved approaches for cost-effective traffic grooming in WDM ring networks: Uniform-traffic case. *Photonic Network Communications*, 3(3):245–254, 2001.

[33] S. Chopra. On the spanning tree polyhedron. *Operations Research Letters*, 8:25–29, 1989.

[34] S. Chopra, I. Gilboa, and S. T. Sastry. Source sink flows with capacity installation in batches. *Discrete Applied Mathematics*, 86:165–192, 1998.

[35] T. Christof and A. Löbel. PORTA - POlyhedron Representation Transformation Algorithm. http://www.zib.de/Optimization/Software/Porta/.

[36] T. Cinkler, D. Marx, C. Larsen, and D. Fogaras. Heuristic algorithms for joint configuration of the optical and electrical layer in multi-hop wavelength routing networks. In *Proceedings of the 19th IEEE Infocom 2000, Tel-Aviv, Israel*, pages 1000–1009, 2000.

[37] D. Colle. *Design and Evolution of Data-centric Optical Networks*. PhD thesis, University of Ghent, Belgium, 2002.

[38] D. Coudert, P. Datta, S. Perennes, H. Rivano, and M.-E. Voge. Shared risk resource group: Complexity and approximability issues. *Parallel Processing Letters*, 17(2):169–184, June 2007.

[39] CPLEX 11.0. CPLEX 11.0 reference manual, 2007. http://www.cplex.com.

[40] G. Dahl and M. Stoer. A cutting plane algorithm for multicommodity survivable network design problems. *INFORMS Journal on Computing*, 10(1):1–11, 1998.

[41] G. Dahl, A. Martin, and M. Stoer. Routing through virtual paths in layered telecommunication networks. *Operations Research*, 47(5):693–702, 1999.

[42] E. Danna, E. Rothberg, and C. Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102(1):71–90, 2005.

[43] M. Dawande, R. Gupta, S. Naranpanawe, and C. Sriskandarajah. A traffic-grooming algorithm for wavelength-routed optical networks. *INFORMS Journal on Computing*, 19(4):565–574, 2006.

[44] P. Demeester, M. Gryseels, K. Van Doorselaere, A. Autenrieth, C. Brianza, G. Signorelli, R. Clemente, M. Ravera, A. Jajszczyk, D. Janukowicz, G. Kalbe, Y. Harada, S. Ohta, and A.G. Rhissa. Resilience in a multi-layer network. In *Proceedings of the 1st International Workshop on the Design of Reliable Communication Networks (DRCN 1998), Brugge, Belgium*, May 1998.

[45] J. Doucette, W.D. Grover, and R. Martens. Modularity and economy-of-scale effects in the optimal design of mesh-restorable networks. In *Proceedings of the 1999 Canadian Conference on Electrical and Computer Engineering, Alberta, Canada*, pages 226–231, May 1999.

[46] R. Doverspike and B. Wilson. Comparison of capacity efficiency of DCSS network restoration routing techniques. *Journal of Network and Systems Management*, 2(2): 95–123, 1994.

[47] R. Dutta. *Virtual Topology Design for Traffic Grooming in WDM Networks*. PhD thesis, North Carolina State University, 2001.

[48] R. Dutta and G. Rouskas. A survey of virtual topology design algorithms for wavelength routed optical networks. *Optical Networks Magazine*, 1(1):77–89, January 2000.

[49] R. Dutta and G. Rouskas. Traffic grooming in WDM networks: Past and future. *IEEE Network*, 16(6):46–56, 2002.

[50] R. Dutta and G. Rouskas. On optimal traffic grooming in WDM rings. *IEEE Journal on Selected Areas in Communications*, 20(1):110–121, 2002.

[51] A. Dwivedi and R. Wagner. Traffic model for USA long distance optimal network. In *Proceedings of the Optical Fiber Communication Conference*, pages 156–158, March 2000.

[52] B. Fortz and M. Poss. An improved Benders decomposition applied to a multi-layer network design problem. Optimization Online preprint 1919, February 2008.

[53] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings of the 19th IEEE Infocom 2000, Tel-Aviv, Israel*, pages 519–528, 2000.

[54] B. Fortz and M. Thorup. Increasing Internet capacity using local search. *Computational Optimization and Applications*, 29:13–48, 2004.

[55] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman and Company, New York, 1979.

[56] O. Gerstel, R. Ramaswami, and G. Sasaki. Cost effective traffic grooming in WDM rings. In *Proceedings of the 17th IEEE Infocom 1998, San Francisco, USA*, pages 69–77, 1998.

[57] O. Gerstel, P. Lin, and G. Sasaki. Combined WDM and SONET network design. In *Proceedings of the 18th IEEE Infocom 1999, New York, USA*, volume 2, pages 734–743, 1999.

[58] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Boston, Dordrecht, London, 1997.

[59] R. E. Gomory. An algorithm for the mixed-integer problem. Technical Report RM-2597, Rand Corporation, 1960.

[60] R.E. Gomory and T.C. Hu. Multi-terminal network flows. *Journal of the Society for Industial Applied Mathematics*, 9:551–570, 1961.

[61] M. Grötschel, C. Monma, and M. Stoer. Facets for polyhedra arising in the design of communication networks with low-connectivity constraints. *SIAM Journal on Optimization*, 2(3):474–504, August 1992.

[62] W.D. Grover. *Mesh-Based Survivable Networks: Options and Strategies for Optical, MPLS, SONET, and ATM Networking*. Prentice-Hall, Inc., 2003.

[63] C. Gruber, R. Wessäly, S. Orlowski, A. Zymolka, and A.M.C.A. Koster. A computational study for demand-wise shared protection. In *Proceedings of the 5th International Workshop on the Design of Reliable Communication Networks (DRCN 2005), Island of Ischia, Italy*, pages 421–428, October 2005.

[64] O. Günlük. A branch-and-cut algorithm for capacitated network design problems. *Mathematical Programming*, 86:17–39, 1999.

[65] R. Hassin, J. Monnot, and D. Segev. Approximation algorithms and hardness results for labeled connectivity problems. *Journal of Combinatorial Optimization*, 14(4):437–453, November 2007.

[66] M. Herzberg and S. Bye. An optimal spare-capacity assignment model for survivable networks with hop limits. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM'94), San Francisco, CA, USA*, pages 1601–1606, September 1994.

[67] H. Höller and S. Voß. An integer programming model for integrated SDH/WDM network planning. In *Proceedings of the 2nd International Network Optimization Conference (INOC 2005), Lisbon, Portugal*, pages 726–731, 2005.

[68] H. Höller and S. Voß. A heuristic approach for combined equipment-planning and routing in multi-layer SDH/WDM networks. *European Journal of Operational Research*, 171(3):787–796, June 2006.

[69] J. Hu. Diverse routing in optical mesh networks. *IEEE Transactions on Communications*, 51(3):489–494, 2003.

[70] J. Hu and B. Leida. Traffic grooming, routing, and wavelength assignment in optical WDM mesh networks. In *Proceedings of 6th INFORMS Telecommunications Conference, Boca Raton, Florida, USA*, April 2002.

[71] R. Hülsermann, M. Jäger, A.M.C.A. Koster, S. Orlowski, R. Wessäly, and A. Zymolka. Availability and cost based evaluation of demand-wise shared protection. In *7. ITG-Fachtagung Photonische Netze, Leipzig, Germany*, pages 161–168, Leipzig, Germany, 2006. VDE Verlag GmbH.

[72] R. Hülsermann, M. Gunkel, C. Meusburger, and D. Schupke. Cost modeling and evaluation of capital expenditures in optical multilayer networks. *Journal of Optical Networking*, 7(9):814–833, September 2008.

[73] R. Iraschko, M. MacGregor, and W.D. Grover. Optimal capacity placement for path restoration in STM, or ATM mesh survivable networks. *IEEE Transactions on Networking*, 6(3):325–336, June 1998.

[74] M. Iri. On an extension of the maximum-flow minimum-cut theorem to multicommodity flows. *Journal of the Operations Research Society of Japan*, 13(3):129–135, 1971.

[75] R. Kandula and G. Sasaki. Gromming of dynamic tributary traffic in WDM rings with rearrangements. In *Proceedings of 39th Annual Allerton Conference on Communication, Control, and Computing 2001, Illinois, USA*, October 2001.

[76] A. Knippel and B. Lardeux. The multi-layered network design problem. *European Journal of Operational Research*, 183(1):87–99, November 2007.

[77] T. Koch and R. Wessäly. Hierachical infrastructure planning in telecommunication networks. In *Proceedings of INFRATRAIN Autumn School: Network Economics – Modeling and Regulating Transport and Energy Sectors in Europe*, 2004.

[78] V. Konda and T. Chow. Algorithm for traffic grooming in optical networks to minimize the number of transceivers. In *Proceedings of the 2001 IEEE Workshop on High Performance Switching and Routing, Dallas, Texas, USA*, pages 218–221, May 2001.

[79] A.M.C.A. Koster and A. Zymolka. Minimum converter wavelength assignment in all-optical networks. In *Proceedings of the 8th Conference on Optical Networks Design and Modeling (ONDM 2004), Ghent, Belgium*, pages 517–535, 2004.

[80] A.M.C.A. Koster and A. Zymolka. Tight LP-based lower bounds for wavelength conversion in optical networks. *Statistica Neerlandica*, 61(1):115–136, 2007.

[81] A.M.C.A. Koster, S. Orlowski, C. Raack, G. Baier, and T. Engel. *Single-layer Cuts for Multi-layer Network Design Problems*, volume 44, chapter 1, pages 1–23. Springer-Verlag, College Park, MD, U.S.A., February 2008. Selected proceedings of the 9th INFORMS Telecommunications Conference.

[82] A. Kröller and R. Wessäly. Integrated optimization of hardware configuration and capacity dimensioning in SDH and opaque WDM networks. In *Proceedings of the 1st International Network Optimization Conference (INOC 2003), Paris, France*, pages 349–354, 2003.

[83] E. Kubilinskas. *Design of Multi-layer Telecommunication Networks*. PhD thesis, Lund University, 2008.

[84] E. Kubilinskas and M. Pióro. An IP/MPLS over WDM network design problem. In *Proceedings of the 2nd International Network Optimization Conference (INOC 2005), Lisbon, Portugal*, pages 718–725, 2005.

[85] E. Kubilinskas, M. Pióro, and P. Nilsson. Design models for robust multi-layer Next Generation Internet core networks carrying elastic traffic. *Journal of Network and Systems Management*, 13(1):57–76, March 2005.

[86] B. Lardeux, A. Knippel, and J. Geffard. Efficient algorithms for solving the 2-layered network design problem. In *Proceedings of the 1st International Network Optimization Conference (INOC 2003), Paris, France*, pages 367–372, 2003.

[87] H. Lee, H. Choi, S. Subramaniam, and H. Choi. Survivable embedding of logical topology in WDM ring networks. In *Proceedings of the Joint Conference on Information Sciences 2002, Research Triangle Park, North Carolina, USA*, pages 1393–1397, March 2002.

[88] Q. Louveaux and L.A. Wolsey. Lifting, superadditivity, mixed integer rounding and single node flow sets revisited. *4OR*, 1(3):173–207, 2003.

[89] T.A. Magnanti and P. Mirchandani. Shortest paths, single origin-destination network design and associated polyhedra. *Networks*, 33:103–121, 1993.

[90] T.A. Magnanti and Y. Wang. Polyhedral properties of the network restoration problem – with the convex hull of a special case. Technical report, MIT Press, November 1997.

[91] T.A. Magnanti, P. Mirchandani, and R. Vachani. Modeling and solving the two-facility capacitated network loading problem. *Operations Research*, 43(1):142–157, 1995.

[92] H. Marchand and L.A. Wolsey. Aggregation and mixed integer rounding to solve MIPs. *Operations Research*, 49(3):363–371, 2001.

[93] B. Melian, B. Laguna, and J. Moreno-Perez. Minimizing the cost of placing and sizing wavelength division multiplexing and optical cross-connect equipment in a telecommunications network. *Networks*, 45(2):199–209, 2005.

[94] E. Modiano and A. Narula-Tam. Survivable routing of logical topologies in WDM networks. In *Proceedings of the 20th IEEE Infocom 2001, Anchorage, USA*, pages 348–357, 2001.

[95] E. Modiano and A. Narula-Tam. Survivable lightpath routing: A new approach to the design of WDM-based networks. *IEEE Journal on Selected Areas in Communications*, 20(4):800–809, May 2002.

[96] J. Müller. Facets of the cutset polyhedron of 2-layer survivable network design problems. Diploma thesis, Technische Universität Berlin, December 2008.

[97] G. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1988.

[98] K. Onaga and O. Kakusho. On feasibility conditions of multicommodity flows in networks. *Transactions on circuit theory*, 18(4):425–429, 1971.

[99] S. Orlowski. Local and global restoration of node and link failures in telecommunication networks. Diploma thesis, Technische Universität Berlin, February 2003.

[100] S. Orlowski and M. Pióro. On the complexity of column generation in survivable network design with path-based survivability concepts. ZIB report ZR-08-51, Zuse Institute Berlin, December 2008.

[101] S. Orlowski and R. Wessäly. Comparing restoration concepts using optimal network configurations with integrated hardware and routing decisions. In *Proceedings of the 4th International Workshop on the Design of Reliable Communication Networks (DRCN 2003), Banff, Alberta, Canada*, pages 15–22, October 2003.

[102] S. Orlowski and R. Wessäly. An integer programming model for multi-layer network design. ZIB report ZR-04-49, Zuse Institute Berlin, December 2004.

[103] S. Orlowski and R. Wessäly. The effect of hop limits on optimal cost in survivable network design. In S. Raghavan and G. Anandalingam, editors, *Telecommunications Planning: Innovations in Pricing, Network Design and Management*, volume 33 of *Operations Research/Computer Science Interfaces*, chapter 8, pages 151–166. Springer-Verlag, January 2006. ISBN 0-387-29222-5.

[104] S. Orlowski, A.M.C.A. Koster, C. Raack, and R. Wessäly. Two-layer network design by branch-and-cut featuring MIP-based heuristics. In *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, 2007.

[105] 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*, 2007. http://sndlib.zib.de, extended version accepted in *Networks*, 2009.

[106] F. Ortega and L.A. Wolsey. A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem. *Networks*, 41:143–158, 2003.

[107] M. Padberg and G. Rinaldi. A branch and cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33:60–100, 1991.

[108] R. Perlman. *Interconnections: Bridges, Routers, Switches and Internetworking Protocols*. Addison-Wesley, 2nd edition, 1999.

[109] M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers, 2004.

[110] F. Poppe and P. Demeester. Economic allocation of spare capacity in mesh-restorable networks: Models and algorithms. In *Proceedings of the 6th International Conference on Telecommunication Systems (ICTSM 1998), Modeling and Analysis, Nashville, USA*, pages 77–86, March 1998.

[111] P. Prathombutr, J. Stach, and E.K. Park. An algorithm for traffic grooming in WDM optical mesh networks with multiple objectives. *Telecommunication Systems*, 28(3,4): 369–386, 2005.

[112] Matheon project B3. Integrated planning of multi-layer networks. ZIB, 2002–2010. http://www.zib.de/Optimization/Projects/Telecom/Matheon-B3.

[113] C. Raack. Employing mixed-integer rounding in telecommunication network design. Diploma thesis, Technische Universität Berlin, December 2005.

[114] C. Raack and A.M.C.A. Koster. An integer set-packing problem arising in two-layer network design. In *Proceedings of the 4th International Network Optimization Conference (INOC 2009), Pisa, Italy*, April 2009.

[115] C. Raack, A.M.C.A. Koster, S. Orlowski, and R. Wessäly. Capacitated network design using general flow-cutset inequalities. ZIB Report ZR-07-14, Zuse Institute Berlin, June 2007. Submitted to Networks.

[116] C. Raack, A.M.C.A. Koster, and R. Wessäly. On the strength of cut-based inequalities for capacitated network design polyhedra. ZIB Report ZR-07-08, Zuse Institute Berlin, June 2007.

[117] S. Raghavan. Personal communication between S. Orlowski and S. Raghavan at the INOC 2009 conference, April 2009.

[118] S. Raghavan and D. Stanojević. Branch-and-price for WDM optical networks with no bifurcation of flow. Working paper, Robert H. Smith School of Business, University of Maryland, April 2007.

[119] S. Raghavan and D. Stanojević. WDM optical design using branch-and-price. Working paper, Robert H. Smith School of Business, University of Maryland, April 2007.

[120] R. Ramaswami and K.N. Sivarajan. *Optical Networks: A Practical Perspective*. Morgan Kaufmann Publishers, 1998.

[121] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. IETF Internet RFC 3031, January 2001.

[122] G. Sasaki and T. Lin. A minimal cost WDM network for incremental traffic. In *Proceedings of the SPIE All-Optical Networking Conference 1999*, 1999.

[123] M. Savelsbergh. Preprocessing and probing for mixed integer programming problems. *ORSA Journal on Computing*, 6:445–454, 1994.

[124] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1986.

[125] M. Sexton and A. Reid. *Broadband Networking – ATM, SDH, and SONET*. Artech House, 1997.

[126] R. Srinivasan and A. Somani. Request-specific routing in WDM grooming networks. In *Proceedings of the IEEE International Conference on Communications 2002 (ICC'02), New York, USA*, volume 5, pages 2876–2880, 2002.

[127] D. Stanojević. *Optimization of Contemporary Telecommunications Networks: Generalized Spanning Trees and WDM Optical Networks*. PhD thesis, University of Maryland, 2005.

[128] M. Stoer and G. Dahl. A polyhedral approach to multicommodity survivable network design. *Numerische Mathematik*, 68(1):149–167, 1994.

[129] C.S. Sung and S.H. Song. Branch-and-price algorithm for a combined problem of virtual path establishment and traffic packet routing in a layered communication network. *Journal of the Operational Research Society*, 54:72–82, 2003.

[130] J. Suurballe. Disjoint paths in a network. *Networks*, 4:125–145, 1974.

[131] A. Todimala and B. Ramamurthy. Survivable virtual topology routing under shared risk link groups in WDM networks. In *First International Conference on Broadband Networks (BROADNETS'04), San Jose, CA, USA*, pages 130–139, October 2004.

[132] J.P. Walser. *Integer Optimization by Local Search*, volume 1637 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[133] J. Wang and B. Mukherjee. Interconnected WDM ring networks: Strategies for interconnection and grooming. *Optical Networks Magazine*, 3(5):10–20, September/October 2002.

[134] J. Wang, W. Cho, V. Vemuri, and B. Mukherjee. Improved approaches for cost-effective traffic grooming in WDM ring networks: ILP formulations and single-hop and multihop connections. *IEEE/OSA Journal of Lightwave Technology*, 19(11):1645–1653, November 2001.

[135] R. Wessäly. *Dimensioning Survivable Capacitated NETworks*. PhD thesis, Technische Universität Berlin, April 2000.

[136] R. Wessäly, S. Orlowski, A. Zymolka, A.M.C.A. Koster, and C. Gruber. Demand-wise shared protection revisited: A new model for survivable network design. In *Proceedings of the 2nd International Network Optimization Conference (INOC 2005), Lisbon, Portugal*, pages 100–105, 2005.

[137] T. Wu. Emerging technologies for fiber network survivability. *IEEE Communications Magazine*, pages 60–74, February 1995.

[138] R. Wunderling. Paralleler und objektorientierter Simplex. ZIB Technical Report TR-96-09, Zuse Institute Berlin, 1996. http://www.zib.de/PaperWeb/abstracts/TR-96-09.

[139] Y. Xiong. Optimal design of restorable ATM networks. In *Proceedings of the 4th IEEE ATM Workshop 1994, Fairfax, Virginia, USA*, May 1994.

[140] S. Yuan, S. Varma, and J.P. Jue. Minimum-color path problems for reliability in mesh networks. In *Proceedings of the 24th IEEE Infocom 2005, Miami, USA*, volume 4, pages 2658–2669, March 2005.

[141] K. Zhu and B. Mukherjee. Traffic grooming in an optical WDM mesh network. *IEEE Journal on Selected Areas in Communications*, 20(1):122–133, 2002.

[142] A. Zymolka. *Design of Survivable Optical Networks by Mathematical Optimization*. PhD thesis, Technische Universität Berlin, December 2007.

List of Tables

List of Figures

List of acronyms

ADSL	Asymmetric Digital Subscriber Line
ATM	Asynchronous Digital Hierarchy
CPLEX	a commercial branch-and-cut framework
DWDM	Dense Wavelength Division Multiplexing
ECMP	Equal-Cost Multi-Path (IP routing protocol)
EXC	Electronic cross-connect
EIBONE	Efficient Integrated Backbone (German government project)
IP	Integer Program, Integer Programming, or Internet Protocol
LNS	Large-Scale Neighborhood Search
LP	Linear Program or Linear Programming
MIP	Mixed-Integer Program or Mixed-Integer Programming
MIR	Mixed-Integer Rounding
MPLS	Multi-Protocol Label Switching
MUX	Multiplexer/Demultiplexer
OADM	Optical Add-Drop Multiplexer
OTN	Optical Transport Network (extension of SDH)
OXC	Optical cross-connect
OSPF	Open Shortest Path First (IP routing protocol)
SCIP	a non-commercial branch-and-cut-and-price framework developed at ZIB
SDH	Synchronous Digital Hierarchy
SNDLIB	Survivable Network Design Library
STM	Synchronous Transport Module (in SDH technology)
VC	Virtual Container (in SDH technology)
WDM	Wavelength Division Multiplexing
ZIB	Zuse Institute Berlin

List of symbols

General sets and notation

$\mathbb{Z}$	the set of integer numbers $\ldots, -2, -1, 0, 1, 2, \ldots$
$\mathbb{Z}_+$	the set of nonnegative integer numbers $0, 1, 2, \ldots$
$\mathbb{R}$	the set of real numbers
$\mathbb{R}_+$	the set of nonnegative real numbers $\{x \in \mathbb{R} \,\vert\, x \geq 0\}$
$\mathbb{R}_-$	the set of nonpositive real numbers $\{x \in \mathbb{R} \,\vert\, x \leq 0\}$
$A \subseteq B$	A is a subset of B ($A = B$ is possible)
$A \subset B$	A is a proper subset of B ($A \neq B$)

Network notation

V	set of all network nodes
E	set of all physical links
L	set of all logical links
R	set of all demands
$\{i, j\}$	unordered pair of nodes or undirected edge between nodes $i \in V$ and $j \in V$
L_{ij}	set of all logical links $\ell \in L$ between nodes i and j
L^e	set of all logical links $\ell \in L$ containing physical link $e \in E$
L^i	set of all logical links $\ell \in L$ containing $i \in V$ as an inner node
$\delta_L(i)$	set of all logical links $\ell \in L$ starting or ending in node $i \in V$
S	subset of nodes $S \subset V$
$\delta_E(S)$	set of physical links $e \in E$ between S and $V \setminus S$
L_S	set of logical links $\ell \in L$ between S and $V \setminus S$
L_1, $\bar{L}_1$	a subset $L_1 \subset L_S$ and its complement $\bar{L}_1 = L_S \setminus L_1$
P	set of admissible paths in the logical layer
P^k	set of admissible logical layer paths for commodity $k \in K$
P_{ij}	set of admissible logical layer paths between nodes i and j
P_ℓ	set of admissible logical layer paths containing logical link $\ell \in L$

Model parameters

M_ℓ	set of capacity modules installable at logical link $\ell \in L$
M_i	set of capacity modules installable at or node $i \in V$
C_ℓ^m	capacity of a logical link module $m \in M_\ell$
C_i^m	switching capacity of a node module $m \in M_i$
C_i	switching capacity needed at a demand end-node $i \in V$ for the access link
B_e	capacity of a physical link capacity module

κ_ℓ^m	cost of a logical link module $m \in M_\ell$
κ_i^m	cost of a node module $m \in M_i$
κ_e	cost of a capacity unit on physical link $e \in E$
R^u	set of all unprotected demands
R^p	set of all protected demands
R	set of all demands, $R = R^u \uplus R^p$
R_{ij}	set of all demands from node $i \in V$ to node $j \in V$
K^u	set of all unprotected commodities
K^p	set of all protected commodities
K	set of commodities, $K = K^u \uplus K^p$
Q	subset of commodities $Q \subseteq K$
d^r	demand value of a demand $r \in R$
d^k	demand value of a commodity $k \in K$
d_i^k	net demand for commodity $k \in K$ that has to leave node $i \in V$
d_i	total emanating demand of node $i \in V$
s^k	source of a commodity $k \in K$
t^k	target of a protected commodity $k \in K^p$

Primal variables and related symbols

$y_\ell^m \in \mathbb{Z}_+$	number of modules of type $m \in M_\ell$ installed on logical link $\ell \in L$
$z_e \in \mathbb{Z}_+$	number of modules of size B_e installed on physical link $e \in E$
$x_i^m \in \{0,1\}$	denotes whether module $m \in M_i$ is installed at node $i \in V$ or not
$f_{\ell,ij}^k \in \mathbb{R}_+$	flow from i to j for commodity $k \in K$ on logical link $\ell \in L_{ij}$
$f_{ij}^k \in \mathbb{R}_+$	flow from i to j for the unprotected commodity $k \in K^u$
$\alpha \in \mathbb{R}_+$	slack variable in the objective function of the metric LP
$l_{\ell,ij}^k \in \mathbb{R}$	lower bound of flow variable $f_{\ell,ij}^k$
$u_{\ell,ij}^k \in \mathbb{R}$	upper bound of flow variable $f_{\ell,ij}^k$

Dual variables

$\pi_i^k \in \mathbb{R}$	dual variable of a demand (flow conservation) constraint
$\mu_\ell \in \mathbb{R}_+$	dual variable of a capacity constraint
$\rho_i^k \in \mathbb{R}_+$	dual variable of a node diversification constraint
$\rho_e^k \in \mathbb{R}_+$	dual variable of a link diversification constraint
$\rho_\ell^k \in \mathbb{R}_+$	aggregated dual diversification variable of $\ell \in L$ and $k \in K^p$
$\underline{\gamma}_{\ell,ij}^k \in \mathbb{R}_+$	dual variable of the lower bound constraint of flow variable $f_{\ell,ij}^k$
$\overline{\gamma}_{\ell,ij}^k \in \mathbb{R}_-$	dual variable of the upper bound constraint of flow variable $f_{\ell,ij}^k$
$y \geq 0$	dual vector of the routing LP of appropriate dimension
$r \geq 0$	dual ray of the routing LP of appropriate dimension

Mixed-integer rounding (MIR)

$r(a,c)$	remainder of division of a by c if $a/c \notin \mathbb{Z}$, and c otherwise
$F_{d,c}$	MIR function for integer variables, see Section 4.4.1
$\bar{F}_{d,c}$	MIR function for fractional variables, see Section 4.4.1

Index

Eidesstattliche Versicherung

Ich versichere an Eides statt, dass ich die von mir vorgelegte Dissertation selbstständig angefertigt habe und alle benutzten Quellen und Hilfsmittel vollständig angegeben habe. Die Zusammenarbeit mit anderen Wissenschaftlern habe ich kenntlich gemacht. Es handelt sich hierbei um Arie Koster und Roland Wessäly, die bereits ihr Promotionsverfahren abgeschlossen haben, sowie Christian Raack und Josefine Müller, die kein Promotionsverfahren beantragt haben.

Erklärung

Teile dieser Arbeit sind bereits veröffentlicht:

- die Heuristiken aus den Abschnitten 4.3.1 und 4.3.4 in: S. Orlowski, A.M.C.A. Koster, C. Raack und R. Wessäly, *Two-layer Network Design by Branch-and-Cut featuring MIP-based Heuristics*, Tagungsband der 3. International Network Optimization Conference (INOC 2007), Spa, Belgien,
- Teile der Modelle aus Abschnitt 3.2, das Basis-Presolving aus Abschnitt 4.2.1 sowie die Schnittebenen aus den Abschnitten 4.4.1 und 4.4.2 in: A.M.C.A. Koster, S. Orlowski, C. Raack, G. Baier und T. Engel, *Single-layer Cuts for Multi-layer Network Design Problems*, Ausgewählter Tagungsband der 9. INFORMS Telecommunications Conference, Band 44, Kap. 1, S. 1–23, Springer-Verlag, 2008.

Eine Anmeldung der Promotionsabsicht habe ich an keiner anderen Fakultät oder Hochschule beantragt.