Chapter 3

Branch-and-Cut Techniques for Solving Realistic Two-Layer Network Design Problems


Abstract  We study a planning problem arising in SDH/WDM multilayer telecommunication network design. The goal is to find a minimum cost installation of link and node hardware of both network layers such that traffic demands can be realized via grooming and a survivable routing. We present a mixed-integer programming formulation for a predefined set of admissible logical links that takes many practical side constraints into account, including node hardware, several bit rates, and survivability against single physical node or link failures. This model is solved using a branch-and-cut approach with problem-specific preprocessing, MIP-based heuristics, and cutting planes based on either of the two layers. On several realistic two-layer planning scenarios, we show that these ingredients can be very useful to reduce the optimality gaps in the multilayer context.

Key words: telecommunication networks, multilayer network design, preprocessing, integer linear programming, cutting planes, MIP-based heuristics

Sebastian Orlowski
atesio GmbH, Sophie-Taeuber-Arp-Weg 27, D-12205 Berlin, Germany, e-mail: orlowski@atesio.de

Christian Raack
Zuse Institute Berlin (ZIB), Takustr. 7, D-14195 Berlin, Germany, e-mail: raack@zib.de

Arie M. C. A. Koster
Lehrstuhl II für Mathematik, RWTH Aachen University, Wülknstr. zwischen 5 und 7, D-52062 Aachen, Germany, e-mail: koster@math2.rwth-aachen.de

Georg Baier
Siemens AG, Munich, Germany, e-mail: georg.baier@siemens.com

Thomas Engel
Nokia Siemens Networks GmbH & Co. KG, Munich, Germany, e-mail: thomas.1.engel@nsn.com

Pietro Belotti
Dept. of Industrial & Systems Engineering, Lehigh University, Bethlehem, PA, USA, e-mail: belotti@lehigh.edu
3.1 Introduction

Planning a telecommunication network is a nontrivial task. For a single network layer such as MPLS, SDH, or WDM, many mathematical models and algorithmic approaches have been proposed during the last 15 years. Links in an SDH network, for instance, may be equipped with a bandwidth of, say, 2.5 Gbit/s or 10 Gbit/s. This bandwidth is used to route several communication demands of lower granularity, like 155 Mbit/s. Dimensioning the links and routing the communication demands in the resulting network is a classical network design problem.

A practical telecommunication network, however, consists of several network layers which are embedded in each other. The bandwidth of an SDH link, for instance, can actually be realized by a capacitated lightpath in an underlying optical fiber network. The SDH and the WDM layer are highly interdependent: first, only a limited number of SDH links can traverse a given optical fiber, and second, the failure of a single optical fiber can disrupt many SDH links, and even more demand connections. In order to get a survivable network in practice, it is indispensable to plan both layers together.

More generally, the two-layer network design problem considered in this paper can be summarized as follows. Given is a set of network nodes together with potential connections between them. This network is called the physical layer and corresponds to the optical fiber network. On every fiber, a limited number of lightpath channels can be transmitted simultaneously, each of them corresponding to a capacitated path in the physical network. The nodes together with the lightpath connections form a so-called logical network on top of the physical one, as illustrated in Figure 3.1. In principle, any path in the physical network can be used for a lightpath, which leads to many parallel logical links. Even if the set of admissible lightpaths is often restricted to several short paths between each node-pair in practice, the resulting logical network is still much denser than a simple complete graph, which makes the network design problem hard to solve.

Fig. 3.1 Upper layer logical links (solid) correspond to paths (dashed) in the lower physical layer

Similar settings occur in many other technologies. An MPLS path, for example, can consist of links which are MPLS paths themselves. In an ATM/SDH setting, capacitated ATM links may be realized by an SDH radio link. Even if that radio link seems “less physical” than an optical fiber, it is called a physical link as well to indicate that it serves as part of a logical link. Similarly, the term “logical link” illustrates the fact that it looks like a direct link to its end nodes: first, it can be equipped with a discrete capacity, and second, traffic with lower granularity is sent
into the link at one end, extracted at the other end, and cannot be accessed inside. The actual physical representation of a logical link usually does not matter to its end nodes.

As this work was motivated by a project with Nokia-Siemens Networks (NSN) on SDH/WDM network planning, we will use that setting to explain our model and algorithm, which are actually in use at NSN in the strategic planning process now. But the concepts are more general and can, at least with slight modifications, be applied to many technological settings.

In our SDH/WDM scenario, a lightpath can be equipped with different bandwidths, and lower-rate traffic demands have to be routed via the lightpaths without exceeding their capacities. A demand may be 1+1-protected, i.e., twice the demand value must be routed such that in the case of any single physical link or node failure, at least the demand value survives. To terminate a lightpath, a sufficiently large electrical cross-connect (EXC) must be installed at both end nodes. The EXC converts the wavelength signal into an electrical SDH signal and extracts lower-rate traffic from it. The latter is either terminated at that node or recombined with other traffic to form new wavelength signals which are sent out on other lightpaths. This process is called grooming. The optimization goal is to minimize total installation cost.

Like in any other publication where an integrated two-layer model is actually used for computations, we do not explicitly assign wavelengths to the lightpaths because finding a suitable wavelength assignment is an extremely hard problem on its own. Instead, we make sure that the maximum number of lightpaths on each fiber is not exceeded, and propose to solve the wavelength assignment and converter installation problem in a subsequent step, as successfully done in [19]. It has been shown in [20] that such an approach causes at most a marginal increase in the overall installation cost in practical instances.

Already, the optimal design of a single layer network is a challenging task that has been considered by many research groups; see for instance [4, 14, 28] and references therein. A branch-and-cut algorithm enhanced by user-defined, problem-specific cutting planes has been proved to be a very successful solution approach in this context. The combined optimization of two layers significantly increases the complexity of the planning task. In most previous publications, mixed-integer programming techniques have been used for designing a logical layer with respect to a fixed physical layer [5, 11, 12] or for solving an integrated two-layer planning problem with some simplifying assumptions, like no node hardware or wavelength granularity demands [15, 21]. Knippel and Lardeux [17] and Fortz and Poss [13] have modeled the two-layer network design problem using metric inequalities for both network layers. Recently, Belotti et al. [6] have used a Lagrangean approach for a two-layer network design problem with simultaneous mean demand values and non simultaneous peak demand values. Raghavan and Stanojevic [29] consider the case where all logical links are eligible and develop a branch-and-price algorithm with respect to a fixed physical layer for the case of unprotected demands and one facility on the logical links. Orlowski et al. [25] present several heuristics for a two-layer network design problem, which solve a restricted version of the original problem as a sub-MIP within a branch-and-cut framework. In a recent paper [18],
we have presented a mathematical model for the described planning problem with a predefined set of logical links. It includes node hardware, several bit rates on the logical links, and survivability against physical node and link failures. To our knowledge, this was the first time that so many practically relevant side constraints, and in particular, multilayer survivability, were taken into account in an integrated two-layer planning model.

We have solved this model using a branch-and-cut approach with problem-specific preprocessing, user-defined cutting planes, and heuristics. This chapter combines results from [25] and [18]. In addition to the preprocessing and cutting planes presented in [18], we have also adapted the MIP-based primal heuristics from [25] to the full planning problem and call them at various places during the branch-and-cut tree. The algorithm is tested on several network instances provided by Nokia Siemens Networks. The paper is structured as follows. In Section 3.2, we present and discuss our mixed-integer programming model. Section 3.3 describes our MIP-based primal heuristics used within the branch-and-cut algorithm. In Section 3.4, we describe the cutting planes used and state some known results about their strength. Computational results are provided in Section 3.5. We conclude with Section 3.6.

We assume that the reader has basic knowledge of mixed-integer programming and branch-and-cut techniques; good introductions to this topic are [24] and [31].

### 3.2 Mathematical Model

#### 3.2.1 Mixed-Integer Programming Model

We will now introduce the mixed-integer programming (MIP) model on which our cutting planes are based. Afterwards, we will describe some basic preprocessing steps that we have applied to strengthen the formulation.

**Parameters**

The physical network is represented by an undirected graph \((V,E)\). The logical network is modeled by an undirected graph \((V,L)\) with the same set of nodes and a fixed set \(L\) of admissible logical links. Each logical link represents an undirected path in the physical network. In consequence, any two nodes \(i, j \in V\) may be connected by many parallel logical links corresponding to different physical paths, collected in the set \(L_{ij} = L_{ji}\). Looped logical links are forbidden, i.e., \(L_{ii} = \emptyset\) for all \(i \in V\). Let \(\delta_L(i) = \cup_{j \in Y} L_{ij}\) be the set of all logical links starting or ending at \(i\). Eventually, \(L_e \subseteq L\) denotes the set of logical links containing edge \(e \in E\), and likewise, \(L_i \subseteq L\) refers to the set of logical links containing node \(i \in V\) as an inner node.
We consider different types of capacities for logical links, physical links, and nodes. Each logical link \( \ell \in L \) has a set \( M_{\ell} \) of available capacity modules, each of them with a cost of \( \kappa_{\ell}^{m} \in \mathbb{R}_{+} \) and a base capacity (bit rate) of \( C_{\ell}^{m} \in \mathbb{Z}_{+} \) that can be installed on \( \ell \) in integer multiples. Similarly, every node \( i \in V \) has a set \( M_{i} \) of node modules (representing different EXC types), at most one of which may be installed at \( i \). Module \( m \in M_{i} \) provides a switching capacity of \( C_{m}^{i} \in \mathbb{Z}_{+} \) (e.g., in bits per second) at a cost of \( \kappa_{m}^{i} \in \mathbb{R}_{+} \). On a physical link \( e \in E \), a fiber may be installed at a cost of \( \kappa_{e} \in \mathbb{R}_{+} \). Each fiber supports up to \( B \in \mathbb{Z}_{+} \) lightpaths.

For the routing part, a set \( H \) of undirected point-to-point communication demands is given, which may be protected or unprotected. Protected demands are expected to survive any single physical node or link failure, whereas unprotected demands are allowed to fail. Each demand \( h \in H \) has a source node, a target node, and a demand value \( d_{h} \) to be routed between these two nodes. Without loss of generality, we may assume the demands to be directed in an arbitrary way. For 1+1-protected demands, \( d_{h} \) refers to the original demand value that would have to be routed if the demand were unprotected. Adding constraints that limit the amount of flow for a protected commodity through a node or physical link to \( \frac{1}{2}d_{h} \) guarantees that at least the original demand survives any single physical link or node failure. This survivability model, called diversification [3], is a slight relaxation of 1+1-protection, but its solutions can often be transformed into 1+1-solutions.

From the demands, two sets \( K^{P} \) and \( K^{U} \) of protected and unprotected commodities are constructed, where \( K = K^{P} \cup K^{U} \) denotes the set of all commodities. With every commodity \( k \in K \) and every node \( i \in V \), a net demand value \( d_{i}^{k} \in \mathbb{Z} \) is associated such that \( \sum_{i \in V} d_{i}^{k} = 0 \). Every protected commodity \( k \in K^{P} \) consists of a single 1+1-protected point-to-point demand, i.e., \( d_{i}^{k} \neq 0 \) only for the source and target node of the demand. In contrast, unprotected commodities \( k \in K^{U} \) are derived by aggregating unprotected point-to-point demands at a common source node. Summarizing, every commodity \( k \in K \) has a unique source node \( s^{k} \in V \). Unprotected commodities may have several target nodes, whereas protected commodities have a unique target \( t^{k} \in V \). The (undirected) emanating demand of a node \( i \in V \), i.e., the total demand value starting or ending at node \( i \), is given by \( d_{i} = \sum_{k \in K} |d_{i}^{k}| \). The demand value \( d_{i}^{k} \) of a commodity is defined as the demand for \( k \) emanating from its source node, i.e., \( d_{i}^{k} = d_{s}^{k} > 0 \). Notice that for protected commodities, this value is twice the requested bandwidth to ensure survivability.

**Variables**

The model comprises four classes of variables representing the flow and different capacity types. First, for a logical link \( \ell \in L \) and a 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 \( \ell \). For a physical link \( e \in E \), the binary physical link capacity variable \( z_{e} \in \{0, 1\} \) indicates whether \( e \) is equipped with a fiber or not. Similarly, for a node \( i \in V \) and a node module \( m \in M_{i} \), the binary variable \( x_{m}^{i} \in \{0, 1\} \) denotes whether module \( m \) is installed at node \( i \) or not. Eventually, the routing of the commodities is modeled.
by flow variables. In order to model diversification of protected commodities, we need fractional flow variables $f^k_{ij}, f^k_{ji} \in \mathbb{R}_+$ representing the flow for commodity $k \in K$ on logical link $\ell \in \mathcal{L}_{ij}$ directed from $i$ to $j$ and from $j$ to $i$, respectively. For notational convenience, $f^k_{\ell} = f^k_{ij} + f^k_{ji}$ denotes the total flow for $k \in K$ on $\ell \in \mathcal{L}_{ij}$ in both directions.

In our model, a flow variable $f^k_{ij}$ for commodity $k$ and logical link $\ell \in \mathcal{L}_{ij}$ is omitted if any of the following conditions is satisfied: (i) $j = s_k$, (ii) $k \in \mathcal{K}_p$ and $i = t_k$, and (iii) $k \in \mathcal{K}_p$ and $\ell$ contains the source or target node of $k$ as an inner node. The first two types of variables represent flow into the unique source node or out of the unique target node of a protected commodity. They are not generated in order to reduce cycle flows in the edge-flow formulation. For aggregated unprotected commodities, we have to allow flow from one target node to another, and thus flow out of target nodes. The third type of variable would allow flow to be routed through an end node $u$ of a protected commodity without terminating at that node, and then back to $u$ on another logical link. As such routings are not desired in practice, we exclude flow variables whose logical link contains an end node of the corresponding commodity as an inner node. Again, in the unprotected case, such variables have to be admitted because commodities may consist of several aggregated demands.

Objective and Constraints

The objective and constraints of our MIP model read as follows:

\[
\begin{align*}
\text{min} & \quad \sum_{i \in V} \sum_{m \in M_i} \kappa^m_i x^m_i + \sum_{\ell \in L} \sum_{m \in M_\ell} \kappa^{m}_{\ell} y^m_{\ell} + \sum_{e \in E} \kappa^e z_e \\
\text{s.t.} & \quad \sum_{j \in V} \left( f^k_{ij} - f^k_{ji} \right) = d^k_i \quad \forall i \in V, \forall k \in K \\
& \quad \sum_{m \in M_\ell} C^m_{\ell} y^m_{\ell} - \sum_{k \in K} f^k_{\ell} \geq 0 \quad \forall \ell \in L \\
& \quad \sum_{\ell \in L_i} f^k_{\ell} + \sum_{\ell \in \partial_i(\ell)} \frac{1}{2} f^k_{\ell} \leq \frac{1}{2} d^k_i \quad \forall i \in V, \forall k \in \mathcal{K}_p \\
& \quad f^k_{i,j,k,p} \leq \frac{1}{2} d^k_i \quad \forall k \in \mathcal{K}_p, \quad \ell = e = \{s^k, t^k\} \\
& \quad \sum_{m \in M_i} x^m_i \leq 1 \quad \forall i \in V \\
& \quad 2 \sum_{m \in M_h} C^m_{\ell} x^m_{\ell} - \sum_{\ell \in \partial_i(\ell) m \in M_\ell} C^m_{\ell} y^m_{\ell} \geq d_i \quad \forall i \in V \\
& \quad B z_e - \sum_{\ell \in \mathcal{L}_e, m \in M_\ell} y^m_{\ell} \geq 0 \quad \forall e \in E \\
& \quad f^k_{i,j}, f^k_{j,i} \in \mathbb{R}_+, \quad y^m_{\ell} \in \mathbb{Z}_+, x^m_i, z_e \in \{0, 1\} 
\end{align*}
\]
The objective (3.1a) aims at minimizing the total installation cost. The flow conservation (3.1b) and capacity constraints (3.1c) describe a multi-commodity flow and modular capacity assignment problem on the logical layer. For protected commodities, the flow diversification constraints (3.1d) restrict the flow through an intermediate node to half the demand value. In this way, the original demand is guaranteed to survive single node failures as well as single physical link failures, except for the direct physical link between source \(s_k\) and target \(t_k\). This exception is covered by the variable bound (3.1e). In fact, to reduce cycle flows in the LP, we set an upper bound of \(d^k\) and \(\frac{1}{2}d^k\) on all flow variables for unprotected and protected commodities, respectively. The generalized upper bound constraints (3.1f) guarantee that at most one node module is installed at each node. The node switching capacity constraints (3.1g) ensure that the switching capacity of the network element installed at a node is sufficient for all traffic that can potentially be switched at that node. Since all traffic is counted twice, it is compared to twice the installed node capacity. Finally, the physical link capacity constraints (3.1h) make sure that the maximum number of modules on a physical link is not exceeded, and set the physical link capacity variables to 1 whenever a physical link is used.

Discussion of the Model

Several design choices in our model deserve a brief discussion. First, we assume a fractional multi-commodity flow on the logical layer although SDH requires an integer routing in practice. This is motivated by our observations that in good solutions, the routing is often nearly integer even if this is not required, and that relaxing the integrality conditions on the flow variables significantly reduces the computation times. If an integral routing is indispensable, it can be obtained in a postprocessing step, which usually does not deteriorate the cost of the solutions very much if properly done. Notice that the lower bound computed for the model with fractional flow can also be used to assess the quality of the postprocessed integral solutions.

Second, we assume a predefined set of logical links for computational reasons. Considering all possible physical paths as logical links in combination with the practical side constraints and the survivability requirements would ask for a branch-and-cut-and-price approach with a nontrivial pricing problem already in the root node. Such an approach can only be successful if the problem with a limited set of logical links can be solved efficiently. For a branch-and-price approach that deals with all possible logical links on a fixed physical layer using a simplified model without survivability, the reader is referred to [29].

3.2.2 Preprocessing

To strengthen the formulation, we now describe the preprocessing steps applied to the model presented in Section 3.2. In addition to these steps, we have used probing
techniques to set further bounds on variables [30]. Probing is nowadays part of any modern MIP solver like SCIP [1] or CPLEX [16].

• If a physical link \( e \in E \) has zero fiber cost, the variable \( z_e \) can be fixed to 1.

• Obviously, at least the emanating demand must be switched at every node. Consequently, an EXC must be installed at every demand end node \( i \in V \), so the node GUB inequality (3.1f) can be changed to an equality at such nodes: \( \sum_{m \in M_i} x_{im} = 1 \). This strengthens the LP relaxation significantly.

• For the same reason, node modules whose switching capacity is smaller than the emanating demand at a node cannot be installed at that node. Consequently, if \( C_{m_i}^i < d_i \) for some node \( i \in V \) and a node module \( m \in M_i \), the corresponding variable \( x_{im} \) can be removed from the MIP formulation. This often leads to more integral \( x_{im} \) variables in the LP relaxation and to better LP values, especially when combined with the previous rule.

• Two bounds on logical link module variables can be derived from the fiber capacity bounds and the total demand in the network. Both bounds are usually not tight in the LP relaxation, but may help the MIP solver in deriving further relations between the variables to strengthen other bounds.

First, as no more than \( B \) channels can be routed through a given physical link \( e \in E \), every logical link module variable can be bounded by \( B \). Second, 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 (except for undesired cycle flow in the edge-flow formulation). Consequently, the number of modules of type \( m \in M_\ell \) that possibly need to be installed on \( \ell \) is bounded by

\[
\chi_{\ell m}^{i} \leq \left[ \frac{1}{C_{\ell m}} \left( \sum_{h \in H_{m}} d_{h} + \sum_{h \in H_{p}} \frac{1}{2} d_{h} \right) \right].
\]

• 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 or more capacity at a lower price. More precisely, consider a link \( \ell \in L \) and two of its capacity modules \( m_1, m_2 \in M_\ell \) such that \( C_{\ell m_1} \leq C_{\ell m_2} \). If the relation

\[
r = \frac{C_{\ell m_2}}{C_{\ell m_1}} \leq \frac{m_2}{m_1}
\]

holds, then at most \( r \) modules of type \( m_1 \) will be installed in any optimal solution because \( r \) modules of type \( m_1 \) incur the same cost as one unit of type \( m_2 \), but the latter provides the same or more capacity. Furthermore, even if equality holds in the above relation, one large module is preferable to several smaller ones because every module uses one physical channel, independently of its bit rate. Consequently, the variable bound \( y_{\ell m_1} \leq \lfloor r - 1 \rfloor \) can be added to the formulation. It cuts off some non-optimal solutions and maybe some optimal ones
(if equality holds), but always leaves at least one optimal solution if one exists. Notice that the value \([r - 1]\) is exactly \(r - 1\) if \(r\) is integer, and \([r]\) otherwise.

### 3.3 MIP-Based Heuristics Within Branch-and-Cut

We solve the mixed-integer programming formulation using the branch-and-cut framework SCIP 0.90 [2]. In addition, we have implemented several heuristics to construct feasible network configurations based on integer or fractional solutions. At every node of the search tree, SCIP generates cutting planes and calls both our heuristics and some of its own to identify feasible integer solutions. If a new best solution is identified, it is added to SCIP’s solution pool such that it can be used by other heuristics which take feasible solutions as a basis for their work. We will now describe our heuristics and their use within the branch-and-cut framework.

Our MIP-based heuristics address two major subtasks. GROOMCAPMIP and GROOMCAPHEUR solve the grooming and capacity installation subproblem for a given routing exactly and heuristically, respectively, whereas REROUTINGMIP computes a routing within certain link capacities, trying to reduce the required capacity at the same time. By construction, the MIP-based heuristics can easily be adapted to include additional planning requirements, such as node hardware or survivability constraints.

#### 3.3.1 Computing Capacities over a Given Flow

**GROOMCAPMIP**

The GROOMCAPMIP procedure addresses the grooming and capacity assignment subproblem for a given routing by solving a MIP. For a logical link \(\ell \in L_{ij}\), let \(f^*_{\ell} = \sum_{k \in K} \sum_{i \in L} (f_{\ell,ij}^k + f_{\ell,ji}^k)\) be the total flow on \(\ell\) in an integer or LP solution (after removing possible cycle flows). We construct a sub-MIP of the original formulation (3.1a)–(3.1i) that contains logical and physical capacity variables but no routing information:

\[
\begin{align*}
\text{min} \quad & \{ (3.1a) \text{ subject to } (3.1f) - (3.1h), \sum_{m \in M_{\ell}} C^m_{\ell} y^m_{\ell} \geq \lceil f^*_{\ell} \rceil \ \forall \ell \in L, \ z_{\ell}, y^m_{\ell} \in \mathbb{Z}_+ \}.
\end{align*}
\]

Using SCIP’s branch-and-cut algorithm, this sub-MIP is solved as an improvement heuristic every time a new best solution is identified, trying to reduce link capacity cost based on the given routing. As the focus of the sub-MIP is on feasible solutions and not on the lower bound, we disable cut generation and expensive heuristics in the subproblem and impose a node limit of 20,000 and a stall node limit of 10,000, i.e., the sub-MIP is stopped if either a total of 20,000 branch-and-cut nodes has
been computed or if the primal bound could not be improved during the last 10,000 nodes.

**GROOMCapHEUR**

In contrast to the GROOMCapMIP algorithm, which solves the grooming and capacity assignment problem exactly, the fast and simple GROOMCapHEUR procedure addresses this problem heuristically by decomposition. Again, let $f^\ell$ be the total flow on logical link $\ell \in L$ in an integer or LP solution after removing cycle flows. Installing capacities on $\ell$ at minimum cost with a lower bound of $f^\ell$ can be formulated as an integer knapsack problem:

$$\min \left\{ \sum_{m \in M_\ell} \kappa^m \gamma^m \bigg| \sum_{m \in M_\ell} C^m \gamma^m \geq \lceil f^\ell \rceil, \gamma^m \in \mathbb{Z}^+ \right\}.$$ 

For $|M_\ell| = 1$ this knapsack problem is trivial to solve. Otherwise, it is solved heuristically for each logical link $\ell \in L$ using a greedy algorithm, taking the maximum capacity of each physical link into account. In a second step, node capacities are installed as much as needed for the given link capacities (if possible). As this heuristic runs very fast, we call it at every branch-and-cut node to construct feasible solutions from the current LP solution.

### 3.3.2 Rerouting Flow to Reduce Capacities

**ReroutingMIP**

The ReroutingMIP 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 $U^\ell$ on the capacity of each logical link $\ell \in L$, ReroutingMIP solves the following problem using SCIP’s branch-and-cut capabilities:

$$\min \left\{ (3.1a) \text{ subject to } (3.1b)-(3.1i), \sum_{m \in M_\ell} C^m \gamma^m \leq U^\ell \quad \forall \ell \in L \right\}.$$ 

With small $U^\ell$, this problem is much easier to solve than the original problem. By setting $U^\ell$ to the total capacity of link $\ell \in L$ in an integer solution, ReroutingMIP can be used as an improvement algorithm that tries to reduce capacities by rerouting flow. This generalizes the rerouting step in the iterative heuristics proposed in [15, 21], making it independent of the ordering of the demands.

We employ ReroutingMIP not as an improvement heuristic but as a construction algorithm. Given some value $\kappa \geq 1$ and an LP solution with total logical link
capacities $y^*_\ell = \sum_{m \in M} C^0_{\ell m} y^*_m$, we solve the above sub-MIP with $U^*_\ell := C^0 \left\lceil \frac{\kappa y^*_\ell}{\kappa y} \right\rceil$ where $C^0$ is the smallest module capacity installable on $\ell$. If the installable capacities form a divisibility chain (which is often the case in practical applications), $U^*_\ell$ is the smallest installable integer capacity greater than or equal to $\kappa y^*_\ell$. Obviously, a higher value of $\kappa$ 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 allowed us to quickly determine good solutions in the sub-MIP.

As the REROUTINGMIP algorithm consumes much more time than the other heuristics, we restrict its application to the LP solution at the end of the branch-and-cut root node. In the sub-MIP (as well as in the original problem), good solutions are often found within the first few branch-and-bound nodes, whereas much time is spent afterwards on proving optimality of the solution. Hence, we disable cut generation and expensive heuristics in the subproblem, and we impose a node limit of 10,000 nodes and a stall node limit of 5,000 nodes. To increase the chance of finding good solutions, we also apply the GROOMCAPHEUR and GROOMCAPMIP algorithms within the sub-MIP, which tends to improve the overall solution quality.

3.4 Cutting Planes

Backed by theoretical results of polyhedral combinatorics, cutting plane procedures have proved to be a feasible approach to improve the performance of mixed-integer programming solvers for many single-layer network design problems. In this section we show how an appropriate selection of these inequalities can be adapted to our problem setting. Their separation within a branch-and-cut algorithm, i.e., the problem to find a violated inequality that cuts off a fractional LP solution or to determine that no such inequality exists, is only briefly summarized here; details can be found in [18].

3.4.1 Cutting Planes on the Logical Layer

On the logical layer, we consider cutset inequalities and flow-cutset inequalities. These cutting planes have, for instance, been studied in [4, 8, 10, 22, 28] for a variety of network settings (e.g., directed, undirected, and bidirected link models, single or multiple capacity modules) and have been successfully used within branch-and-cut algorithms for capacitated single-layer network design problems [7, 14, 28].

To be precise, the inequalities on the logical layer are valid for the polyhedron $P$ defined by the multi-commodity flow constraints (3.1b) and the capacity constraints (3.1c). That is,

$$P = \text{conv} \left\{ (f, y) \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{n_2} \mid (f, y) \text{ satisfies } (3.1b), (3.1c) \right\},$$
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, the inequalities are also valid for that model.

We introduce the following notation. For any subset \( \emptyset \neq S \subset V \) of nodes, 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 \( S \). Furthermore, define \( d_S^k = \sum_{\ell \in S} d_{\ell}^k \geq 0 \) to be the total demand value to be routed over the cut \( L_S \) for commodity \( k \in K \). By reversing the direction of demands and exchanging the corresponding flow variables, we may w.l.o.g. assume that \( d_S^k \geq 0 \) for all \( k \in K \) (i.e., the commodity is directed from \( S \) to \( V \setminus S \), or the end nodes of \( k \) are either all in \( S \) or all in \( V \setminus S \)). This reduction is done implicitly in our code. More generally, let \( d_S^Q = \sum_{k \in Q} d_S^k \) denote the total demand value to be routed over the cut \( L_S \) for all commodities \( k \in Q \).

**Mixed-Integer Rounding (MIR)**

In order to derive strong valid inequalities on the logical layer we aggregate model inequalities and apply a strengthening of the resulting base inequalities; this is known as mixed-integer rounding (MIR). It exploits the integrality of the capacity variables. Further details on mixed-integer rounding can be found in [23], for instance.

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

\[
F_{d,c} : \mathbb{R} \to \mathbb{R} : a \mapsto \left\lfloor \frac{a}{c} \right\rfloor r_{a,c} = a - c\left(\left\lfloor \frac{a}{c} \right\rfloor - 1\right) > 0
\]

be the remainder of the division of \( a \) by \( c \) if \( \frac{d}{c} \notin \mathbb{Z} \), and \( c \) otherwise. Now assume that \( d/c \notin \mathbb{Z} \) and consider the subadditive MIR functions

\[
F_{d,c} : \mathbb{R} \to \mathbb{R} : a \mapsto \left\lfloor \frac{a}{c} \right\rfloor r_{a,c} = (a - r_{a,c})^+
\]

and \( \bar{F}_{d,c}(a) = \lim_{t \to 0^+} \frac{F_{d,c}(at)}{t} = a^+ \). Given any valid inequality for our problem, applying \( F_{d,c} \) and \( \bar{F}_{d,c} \) to the integer and continuous variables, respectively, yields another valid inequality [24]. Moreover, the resulting coefficients are integral (if \( a, c, \) and \( d \) are integral) and \( |F_{d,c}(a)|, |\bar{F}_{d,c}(a)| \leq |a| \), as shown in [28]. Both features are desirable from a numerical point of view. For more details and explanations see [28].

**Cutset Inequalities**

Let \( L_S \) be a cut 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:

\[
\sum_{\ell \in L_S} \sum_{m \in M_\ell} C_{\ell,m} \gamma_{\ell,m}^k \geq d_S^k. \tag{3.2}
\]
Since all coefficients are nonnegative in inequality (3.2) and $y_{m} \in Z_{+}$, we can round down all coefficients to the value of the right-hand side (if larger). For notational convenience we assume from now on $C_{m}^{\ell} \leq d_{S}^{K}$ for all $\ell \in L$ and $m \in M$. Mixed-integer rounding exploits the integrality of the capacity variables. Setting $c > 0$ to any of the available capacities on the cut and applying the MIR function $F_{c} = F_{d_{S}^{K}}$ to the coefficients and the right-hand side of inequality (3.2) results in the cutset inequality

$$\sum_{\ell \in L} \sum_{m \in M} F_{c}(C_{m}^{\ell})y_{m} \geq F_{c}(d_{S}^{K}).$$

(3.3)

A crucial necessary condition for inequality (3.3) to define a facet for $P$ is that the two subgraphs defined by the network cut be connected, which is trivially fulfilled if $L$ contains logical links between all node pairs.

Given a fractional LP solution, we look for violated MIR inequalities by setting weights on the logical links based on the primal and dual LP solutions, shrinking the logical graph with respect to these weights until only a small number of nodes (say, four or five) remain. In this shrunken graph, we enumerate all cuts, construct the corresponding MIR inequality, and test it for violation. For details, the interested reader is referred to [18].

Flow-Cutset Inequalities

Cutset inequalities can be generalized to flow-cutset inequalities, which have non-zero coefficients also for flow variables. Like cutset inequalities, flow-cutset inequalities are derived by aggregating capacity and flow conservation constraints on a logical cut $L$ 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 [4, 8, 10, 28]. Necessary and sufficient conditions for flow-cutset inequalities to define a facet of $P$ can be found in [28].

Consider fixed nonempty subsets $S \subset V$ of nodes and $Q \subseteq K$ of commodities. Assume that logical link $\ell \in L$ has end nodes $i \in S$ and $j \in V \setminus S$. We will denote by $f_{\ell,i} = f_{\ell,i}^{+}$ inflow into $S$ on $\ell$ while $f_{\ell,j} = f_{\ell,i}^{-}$ refers to outflow from $S$ on $\ell$. We now construct a base inequality to which a suitable mixed-integer rounding function will be applied. First, we obtain a valid inequality from the sum of the flow conservation constraints (3.1b) for all $i \in S$ and all commodities $k \in Q$:

$$\sum_{\ell \in L} \sum_{k \in Q} (f_{\ell,+}^{k} - f_{\ell,-}^{k}) \geq d_{S}^{Q}$$

Given a subset $L_{1} \subseteq L$ of cut links and its complement $L_{1} = L_{S} \setminus L_{1}$ 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 $L_{1}$:
\begin{equation}
\sum_{\ell \in L_1} \sum_{m \in M_1} C_{m} y_{m}^{\ell} + \sum_{\ell \in L_1} \sum_{k \in Q} f_{\ell,k}^{k} \geq d_{Q}^{S}.
\tag{3.4}
\end{equation}

Again, we may assume $C_{m} \leq d_{K}^{S}$ for all $\ell \in L_1$ and $m \in M_1$. Let $c > 0$ be the capacity of a module available on the cut and define $F_{c} = F_{d_{Q}^{S}}$ and $\bar{F}_{c} = \bar{F}_{d_{Q}^{S}}$. Applying these functions to the base inequality (3.4) results in the flow-cutset inequality

\begin{equation}
\sum_{\ell \in L_1} \sum_{m \in M_1} F_{c}(C_{m}) y_{m}^{\ell} + \sum_{\ell \in L_1} \sum_{k \in Q} f_{\ell,k}^{k} \geq F_{c}(d_{Q}^{S}).
\tag{3.5}
\end{equation}

Notice that $\bar{F}_{c}(1) = 1$, so the coefficients of the flow variables remain unchanged. This inequality can be generalized to a flow-cutset inequality also containing inflow variables [28]. By choosing $L_1 = L_{S}$ and $Q = K$, inequality (3.5) reduces to inequality (3.3).

For separating a flow-cutset inequality, a suitable set $S$ of nodes, a subset $Q$ of commodities, a capacity $c$, and a partition $(L_1, L_1)$ of the cut links $L_{S}$ have to be chosen. We apply two different separation heuristics. The first heuristic considers commodity subsets $Q$ with a single commodity $k \in K$ and node sets $S$ 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 can be obtained in linear time; see [4, 18]. 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 $c$ using a Min-Cut Algorithm; see [4].

### 3.4.2 Cutting Planes on the Physical Layer

If the fixed-charge cost values $\kappa_{e}$ are zero, then the corresponding variables $z_{e}$ can be assumed equal to 1 in any optimal solution. If, on the other hand, this cost is positive, the variables will take on fractional values in linear programming (LP) 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. Note that information of 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, e.g., [9, 27]. Let $S \subset V$ be a set of nodes and $\delta(S)$ be the corresponding cut in the physical network. If some demand has to cross the cut, then the inequality

\begin{equation}
\sum_{e \in \delta(S)} z_{e} \geq 1
\tag{3.6}
\end{equation}
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.

If the demand graph (defined by the network nodes and edges corresponding to traffic demands) has \( p \) connected components (usually \( p = 1 \), then

\[
\sum_{e \in E} z_e \geq |V| - p \quad (3.7)
\]

is valid, because the installed physical links can consist of at most \( p \) connected components as well, each one being at least a tree. If protected demands exist and the demand graph is connected, inequality (3.7) can be strengthened by setting the right-hand side to \(|V|\). If protected demands exist for all demand end nodes, this inequality is dominated by the inequalities (3.6) for all demand end nodes as single node subsets.

As the number of inequalities (3.6) and (3.7) is very small, we do not separate them but just add them all in the beginning of the branch-and-bound process.

### 3.5 Computational Results

#### 3.5.1 Test Instances and Settings

For our computational experiments we used the network instances summarized in Table 3.1. In addition to the number of nodes and physical and logical links, the number \( |H| \) of communication demands is given, from which the commodities were constructed (\(|K| = |V| - 1\) if all demands are unprotected and \(|K| = |H|\) if all demands are protected). Further, we report the number \( |M_i| \) of node modules installable at each node and the size of the installable logical link modules. Finally, Table 3.1 indicates whether the instance has physical link cost or not. The first three instances are realistic scenarios provided by Nokia Siemens Networks, whereas the small ring network Ring7 has been constructed out of the larger instance Ring15 in order to study the effect of the cutting planes on the number of branch-and-cut nodes, needed to prove optimality.

| instance     | \( |V| \) | \( |E| \) | \( |L| \) | \( |H| \) | \( |M_i| \) | \( C_1 \), \( C_2 \), \( C_3 \) | physical cost? |
|--------------|--------|--------|--------|--------|--------|----------------|----------------|
| Germany17    | 17     | 26     | 674    | 121    | 16     | 1, 4, 16       | no             |
| Germany17-fc | 17     | 26     | 564    | 121    | 16     | 1, 4, 16       | yes            |
| Ring15       | 15     | 16     | 184    | 78     | 5      | 16, 64, 256    | no             |
| Ring7        | 7      | 8      | 32     | 10     | 5      | 16, 64, 256    | no             |
Germany17 and Germany17-fc are based on a physical 17-node German network available at SNDLib [26]. In both networks, the set of admissible logical links consists of three to five short paths in the physical network between each pair of nodes. Ring15 consists of a physical ring with a chord representing a regional subnetwork connected to a larger national network. The set of logical links consists basically of the two possible logical links for each node pair, one in each physical direction of the ring. Ring7 has been constructed from Ring15 by successively removing nodes with the smallest emanating demand value. Because in our ring instances every node is a demand end node and the demand graph is connected, nearly all physical links have to be used in any feasible solution. We thus do not consider ring variants with physical link cost because doing so would only add a constant to the objective function. In all networks, up to three capacity modules corresponding to 2.5, 10, and 40 Gbit/s can be installed on each logical link depending on its physical path length.

All computations were done on a Linux-operated machine with a 2 × 3 GHz Intel P4 processor and 2 GB of memory. In a first series of test runs, we assumed unprotected demands with physical fibers supporting $B = 40$ wavelengths. In a second series, we made all demands 1+1-protected, assuming $B = 80$ wavelengths in order to allow for feasible solutions with the doubled demand values.

The focus of our computational results is on the effect of the cutting planes, which is discussed in Section 3.5.2 for unprotected networks and in Section 3.5.3 for networks with 1+1 protection. In the corresponding tests, we have always used the preprocessing steps described in Section 3.2.2 and the primal heuristics from Section 3.3 unless otherwise stated. The effect of the heuristics and our preprocessing is discussed in Section 3.5.4.

3.5.2 Unprotected Demands

As cutting planes are primarily thought to increase the lower bound of the LP relaxation, we first consider the effect of the different types of cutting planes on the lower bound at the branch-and-bound root node. We separated the classes cutset inequalities, flow-cutset inequalities, and fixed-charge inequalities on their own as well as all together. Figure 3.2 shows the improvement over time of the lower bound in the root node of the search tree for all test instances. The solid red line at the top marks the value of the best-known solution, which cannot be exceeded by the dual bound curves. The line “no cutting planes” refers to the dual bound with SCIP’s built-in general-purpose cuts only.

It can be seen that in the two Germany17 instances and on the small ring network, our cutting planes reduce the gap between the lower bound and the best-known solution at the root node by 50%–75%. In all three problem instances, flow-cutset inequalities performed better than cutset inequalities, which is in contrast to the results presented by Raack et al. [28] for a single-layer problem. There might be several reasons for this effect. A good candidate is the structural difference between single-layer networks and the logical layer in multilayer problems: the logical layer
graph \((V, L)\) contains edges between almost all node pairs, whereas only a few links cross a cut in single-layer graphs. Further, we have implemented our cutting planes as callbacks in SCIP, whereas in [28], CPLEX was used as a branch-and-cut framework, which means that different general-purpose cutting planes have been used.

For the problem Germany17-fc with physical cost, most of the optimality gap comes from the \(z_c\) variables whose values are highly fractional and close to 0 in the solution of the LP relaxation. A major part of this gap is closed by the fixed-charge inequalities that operate on the physical layer. Of course, the contribution of these inequalities changes with the ratio of the costs of the physical fiber links to the logical wavelength links and the node hardware.

In contrast to these three instances, the problem-specific cutting planes have only a marginal effect on the dual bound for Ring15 compared to that of SCIP’s built-in general-purpose cuts. This is probably due to the fact that in SCIP’s default settings, the dual bound at the end of the root node is within 0.4 % of the optimal solution value, so there is not much room for improvement at all. We also observed that on this instance, our cuts seem to interfere with the \(c\)-mir and Gomory cuts separated by SCIP, which are based on a mixed-integer rounding procedure similar to the one described in Section 3.4. With these cuts disabled in SCIP, our inequalities could reduce the relative distance between the root dual bound and the best-known solution from 3.8 % to 0.4 %, thus achieving the same dual bound as that of SCIP’s...
cutting planes. The number of violated cutting planes found in this setting is reported in Table 3.2 for all instances.

**Table 3.2** Number of violated cutset (3.3), flow-cutset (3.5), and fixed-charge inequalities (3.6) found in root of branch-and-bound tree without separation of SCIP built-in cuts

<table>
<thead>
<tr>
<th>instance</th>
<th>cutset</th>
<th>flow-cutset</th>
<th>fixed-charge</th>
<th>cutset</th>
<th>flow-cutset</th>
<th>fixed-charge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Germany17</td>
<td>37</td>
<td>1521</td>
<td>-</td>
<td>4</td>
<td>940</td>
<td>-</td>
</tr>
<tr>
<td>Germany17-fc</td>
<td>34</td>
<td>1046</td>
<td>35</td>
<td>7</td>
<td>844</td>
<td>20</td>
</tr>
<tr>
<td>Ring15</td>
<td>66</td>
<td>652</td>
<td>-</td>
<td>26</td>
<td>489</td>
<td>-</td>
</tr>
<tr>
<td>Ring7</td>
<td>41</td>
<td>98</td>
<td>-</td>
<td>15</td>
<td>24</td>
<td>-</td>
</tr>
</tbody>
</table>

![Graph](image)

(a) Germany17 (b) Germany17-fc (c) Ring15 (d) Ring7

**Fig. 3.3** Unprotected demands: dual bound during test runs of three hours

In a second study, we have investigated the lasting effect of the cutting planes on the dual bound in longer computations. Figure 3.3 shows the development of the dual bound with and without all cutting planes from Section 3.4 during a computation with a time limit of three hours for all four test instances, compared to the best-known solution. Similarly to most of SCIP’s own cutting planes, we separated our inequalities only at the root node of the branch-and-cut tree.
By applying all separators we could solve the problem Ring7 to optimality within ten minutes, whereas without our cutting planes the computation was aborted after nearly one hour with a nonzero optimality gap due to the memory limit of 2 GB. The size of the search tree was 1.2 million unexplored nodes at this point (and four million explored nodes). Figure 3.3 shows that the dual bounds obtained with our cutting planes are very close to their maximum possible values. In fact, as the upper bound improved in both cases, the relative gap between the dual bound and the best solution found in that specific run (as opposed to the best solution known) could be improved from 4% to 0.36% and from 12.4% to 3.1%, respectively. For Ring15 the improvement of the dual bound by the cutting planes was much smaller than that for the other instances, probably for the reasons discussed above.

### 3.5.3 Protected Demands

![Graphs showing dual bounds and time for Germany17, Germany17-fc, Ring15, and Ring7](image)

**Fig. 3.4** Protected demands: lower bound in test runs of three hours

In the case of protected demands, we first of all would like to point out that the problem size drastically increases compared to the unprotected case. Instead of $|V| - 1$ commodities, $|H|$ commodities have to be routed, increasing the number of
variables and constraints considerably. Consequently, solving the initial LP relaxation, as well as reoptimizing the LP after adding a cutting plane or a branching constraint, takes more time with protection than without.

With 1+1 protected demands, the cutting planes have only a marginal effect on the dual bound. Figure 3.4 shows the increase of the dual bound in a three-hour test run with and without cutting planes (again, the solid red line at the top indicates the best-known solution value). It can be seen that the dual bound always increases, but only by a very limited amount. More detailed investigations revealed that the small progress is mainly due to the strength of the general-purpose c-mir and Gomory cuts generated by SCIP. Experiments where these cuts were turned off showed that our inequalities still contribute significantly to closing the optimality gap at the root node. Table 3.2 shows the number of violated inequalities found at the root node in this setting. Only slightly lower numbers of violated inequalities are found with c-mir and Gomory cuts turned on, but their impact on the dual bound is limited in such a case; cf. Figure 3.4.

### 3.5.4 Preprocessing and Heuristics

![Figure 3.5: Optimality gaps after three hours without cuts and heuristics; with heuristics only; and with both cuts and heuristics.](image)

(a) Unprotected instances  
(b) Protected instances

We also tested the combined effect of our primal heuristics and cutting planes on the optimality gap after three hours. Figure 3.5 shows these gaps for each of the networks in three settings: without cuts and heuristics; with heuristics; and with both heuristics and cuts. The protected Ring7 network has no bars because it was solved to optimality in all cases; we will discuss this network below.

In five out of the seven other instances, adding our cutting planes reduced the optimality gaps. There were two exceptions: On the protected Germany17-fc network, separating the cuts at the root nodes took so much time that a significantly smaller
number of branch-and-cut nodes could be solved within the time limit, leading to a much worse upper bound and a slightly worse lower bound. On the unprotected Ring15 network, the final dual bound was better with cuts than without, but the primal bound was a bit worse, leading to a slightly larger gap. From a practical point of view, however, the difference is marginal.

The effect of our primal heuristics is similar. On five out of the eight instances, the heuristics helped to reduce the optimality gap or the time needed to solve the problem to optimality, and in one instance (unprotected Ring7) there was no difference. In fact, our heuristics found the best solution after three hours in nine out of the 16 cases where they were called; in four of them, the best solution was found by REROUTINGMIP at the root node. Also, on the two instances Ring15 (unprotected) and Germany17-fc (protected), the heuristics found improving solutions early in the branch-and-cut tree, but the resulting traversal of the search tree led to a worse primal bound after the fixed time limit of three hours than without the heuristics. Unfortunately, such effects are rather unpredictable.

![Fig. 3.6 Number of unexplored branch-and-cut nodes on the protected Ring7 network](image)

For the protected Ring7 network, Figure 3.6 shows that the maximum number of unexplored nodes in the search tree was roughly reduced by 2/3 by our cutting planes, even though they were added only in the root node. Also, the heuristics helped in getting a smaller search tree by finding good solutions early in the search tree. The GROOMCAPHEUR heuristic found an optimal solution after 353 nodes, compared to 6,825 nodes without the heuristics. This caused large parts of the search tree to be cut off. A separate run where cutting planes, heuristics, and preprocessing were switched off is shown in the fourth curve at the top of Figure 3.6. It can be seen that the preprocessing also significantly helped to reduce the size of the search tree. With all our plug-ins disabled, an optimal solution was found only after 9,626 nodes, and the size of the search tree grew to more than 780,000 unexplored nodes because of the weak lower bound.
3.6 Conclusions

In this work, we have presented a mixed-integer programming model for a two-layer SDH/WDM network design scenario. The model includes many practically relevant side constraints such as many parallel logical links, various bit rates, node capacities, and survivability with respect to physical node and link failures. To accelerate the solution process for this planning task, we have applied problem-specific preprocessing, a variety of network design-specific cutting planes, and MIP-based primal heuristics within the branch-and-cut framework SCIP. These ingredients have been tested on several realistic planning scenarios provided by Nokia Siemens Networks.

With unprotected demands, our cutting planes significantly raised the lower bounds to close to the optimal solution value. With 1+1 protection against physical failures, they also helped to improve the dual bounds, but less than in the unprotected case. The preprocessing steps, although relatively simple, turned out to be crucial for reducing the size of the branch-and-cut tree. Although the effect of the MIP-based heuristics was not so clear, they found the optimal solution early in the search tree in several instances, sometimes even at the root node. The fact that these heuristics can easily be generalized to other network design problems and side constraints makes the sub-MIP approach very flexible.

Although the presented methods could significantly reduce the computation times for the considered realistic networks, they can still be improved. First, the presented methods do not scale well with the network size because the edge-flow formulation gets too large. Second, fast combinatorial routing heuristics have to be developed in addition to the MIP-based heuristics in order to find good survivable routings that can be used in primal solutions. Third, cutting planes are needed that better take the inter-layer dependencies into account.

Acknowledgements: This work was supported by EU COST action 293 – Graphs and Algorithms in Communication Networks (GRAAL).

The contribution of Sebastian Orlowski was partially supported by the DFG Research Center MATHEON “Mathematics for Key Technologies”.

The contribution of Arie Koster was partially supported by the Zuse Institute Berlin (ZIB) and the Centre for Discrete Mathematics and its Applications (DIMAP), University of Warwick, EPSRC award EP/D063191/1.

References


