LP approaches to survivable networks with single path routing

Diplomarbeit bei
Prof. Dr. M. Grötschel

vorgelegt von
Ulrich Menne

Fachbereich Mathematik der TU Berlin,
Studiengang Wirtschaftsmathematik

Berlin, den 17. Dezember 2003

Contents

Chapter 1

Introduction

Customer growth, increasing demands for old and new services, deregulation and the innovations in switching and transmission technologies are all placing pressure on telecommunication companies to upgrade and expand their networks. With over 50 percent of a telephone company's total investment in communication facilities and with deregulation making the market more competitive, these expansion decisions can have huge economic and strategic ramifications.
The conflictive aims towards network design are the creation of a low-cost network capable of handling the communication demands and maximum customer satisfaction in terms of quality of service. A crucial aspect of customer satisfaction is *survivability*, which means that the offered service has to survive certain failure situations. Common types of network failures are cable cuts caused by dredgers or other external impacts as well as defects of hardware or overloaded network.

The design of communication networks bases on an estimation of communication demands between locations. This communication traffic has to be routed over the connection *links* between these locations. When traffic is routed over the network it consumes capacity provided by some installed hardware. The problem can be modeled as an optimization problem where, on the one hand, the link capacity must be assigned at the lowest cost and, on the other hand, the communication flow must be routed over the network requiring the accomplishment of certain conditions. There exist different approaches towards survivability. The main strategies are *protection* and *restoration*. Protection ensures survivability by conditions on the normal operating state routing while no or little predetermined reconfiguration is required in case of a failure. Restoration techniques react in case of a failure by rerouting the affected traffic (ensuring firstly the existence of spare capacity). Another concept is the approach towards *resilience*, which means that there exist certain hardware technologies, installed at the locations and links of a network, which can never fail. In this thesis we focus on the latter approach. Also the exclusion of routing demands over unacceptably long paths in the network is taken into account (to decrease the probability of a failure of a path component and transmission delays) as well as certain restrictions on the network

locations.

This thesis is motivated by the network planning problem discussed in the context of the ROCOCO project ("Recherche Opérationnelle et COntraintes pour la COnception de réseaux"). The participants of this project (ILOG, France Télécom CNET, LRI and INRIA [2]) aimed at developing algorithms integrating the effectiveness of combinatorial optimization and the flexibility of Constraint Programming to optimize the dimensioning and the redimensioning of networks of telecommunications companies in particular. *Robustness* of optimization techniques was a main aspect in their research endeavors. *Robust* means that the applied algorithm must not only provide "good" solutions to problem instances of different size and numerical characteristics, but also that the algorithm must continue to work well when constraints are added or removed. Hence a network design problem is considered including a number of different additional constraints. These constraints are added arbitrarily to the base problem. No proper mathematical model integrating all constraints simultaneously has been published until now and only an informal problem description was available to us.

The fact that no publicly accessible model exists represents the initial point of this thesis. We develop a comprehensive approach to cope with the described problem including an extensive model able to handle all conditions considered within the ROCOCO project. Chapter 2 gives a short overview on notations. In the first part of Chapter 3 we present the given description, point out the relation to reality and explain the effects of certain constraints. In the second section of this chapter we introduce in detail the developed model including all constraints and restrictions on the routing and capacity installation. It is inclusively capable of modeling any possible combination of the additional constraints. In Chapter 4 we examine the associated polyhedron and elaborate valid inequalities. We show that some of them are even facet-defining for a relaxation of the problem. We apply an LP-based approach to solve the given network design problem. The developed inequalities are used in a branch-and-cut algorithm presented in Chapter 5. Also preprocessing and heuristic endeavors are introduced and described in detail in the last section of this chapter. We evaluate different strategies and measure the quality of our approach on an extensive benchmark suite built on the basis of real network design data provided by France Tèlècom R&D [13]. This suite has been developed in the context of the prementioned ROCOCO project. The inhomogeneous structure of the benchmark reflects their ambitions towards robustness. The achieved computational results are presented in Chapter 6, together with a study of the given data. Finally, in Chapter 7 some conclusions are drawn.

Chapter 2

Graphs and polyhedra

In this chapter, we present an outline of the elementary mathematical tools we utilized to approach our problem. We describe graph theory, since we present the given problem in this context. We further modeled the problem as a linear program and surveyed the respective polyhedron, hence we introduce some facts from the polyhedral theory.
We assume some basic knowledge of linear optimization. For these fundamentals the reader is referred to [8], and especially regarding integer programming [14] and [11].

2.1 Basic notation

We will denote by $\mathbb{R}$ ($\mathbb{Q}$, $\mathbb{Z}$) the real (rational, integer) numbers. The sets $\mathbb{R}_+$ ($\mathbb{Q}_+$, $\mathbb{Z}_+$) stand for the non-negative real (rational, integer) numbers. For ease of notation, we use $\mathbb{K}$ (or $\mathbb{K}_+$) if one of these three sets can be applied. We denote the set of positive integer numbers without zero by $\mathbb{N} = \mathbb{Z}_+ \backslash \{0\}$. For some $n \in \mathbb{N}$, we define by $\mathbb{K}^n$ the set of vectors with n components from $\mathbb{K}$. The transposition of a vector x is x^T.

2.2 Graph Theory

Formally, an **(undirected) graph** is a triple $G = (V, E, \Psi_1)$ consisting of a nonempty set V, called the **nodes** (or vertices), a set E, called the **edges** (or links), and a relation of **incidence** $\Psi_1 : E \to V \times V$ that associates with each edge two nodes, called its **ends**. Usually we just write $G = (V, E)$ and assume that the incidence relation is given implicitly in E. For each edge $e \in E$ there exist nodes $u, v \in V$ such that $\Psi_1(e) = \{u, v\} = \{v, u\}$. Two nodes that are ends of an edge are **adjacent** to one another (**neighbors**). The **degree** $|\delta(v)|$ of a node v is the number of incident edges to v. An edge with identical ends is called a **loop**. If two edges join the same pair of ends, they are called **parallel**. A graph is **simple** if it has neither loops nor parallel edges. For a subset $W \subseteq V$ of nodes, $E(W) \subseteq E$ denotes the subset of edges with both ends in W.

A **digraph** (directed graph) is a triple $G = (V, A, \Psi_1)$ consisting of a nonempty set V, called the **nodes** (or vertices), a set A, called the **arcs**, and a relation of **incidence** $\Psi_1 : A \to V \times V$ that associates with each arc an ordered pair of nodes, called its **ends**. Usually, we just write $D = (V, A)$ and assume that the incidence relation is given implicitly in A. For each arc $a = (u, v)$ we call u the **source** and v the **target** of a. Parallel arcs and loops are defined as for undirected graphs. Two arcs (u, v) and (v, u) are called **associated**. For an arc (u, v), the arc (v, u) is called its **associated backward arc**. A digraph where each arc has its associated backward arc is called **bidirectional**. We call the graph $G = (V, E)$ the **underlying graph** of the digraph $D = (V, A)$ if there is a bijection between the arcs of D and the edges of G, such that for each arc $a = (u, v) \in A$ there is an edge $e = \{u, v\} \in E$, and for each edge $e = \{u, v\} \in E$ the arc $a = (u, v)$ and the arc $a' = (v, u)$ are in A. The **overlaying digraph** $D(G)$ of a graph G is the digraph obtained from G by replacing each edge by two associated arcs with the same ends.

In the following, let $G = (V, E)$ denote a graph and $D = (V, A)$ a digraph.

A **path** $\mathfrak{p}$ in G (or a directed path in D) from v_0 to v_l is a sequence $\mathfrak{p} = (v_0, e_1, v_1, ..., e_l, v_l)$ of nodes $v_0, ..., v_l \in V$ and edges (arcs) $e_1, ..., e_l \in V(\in A)$ of G (D), such that the nodes v_{i-1} and v_i are the ends of edge e_i (are source and target of e_i) for each $1 \leq i \leq l$. Node v_0 is called the **source** and v_l the **target** of $\mathfrak{p}$, while both are denoted as the **endnodes** of $\mathfrak{p}$. The nodes $v_1, ..., v_{l-1}$ are called the **inner nodes** of $\mathfrak{p}$. The **length** of a path is the number of edges (arcs). We use the notation $e \in \mathfrak{p}$ ($a \in \mathfrak{p}$) or $v \in \mathfrak{p}$, if $e \in E$ ($a \in A$) is an edge (arc) of $\mathfrak{p}$ or $v \in V$ is a node of $\mathfrak{p}$. We denote by $V(\mathfrak{p})$ and $E(\mathfrak{p})$ ($A(\mathfrak{p})$) the set of inner nodes and edges (arcs) of $\mathfrak{p}$. That is, for a path $\mathfrak{p} = (v_0, e_1, v_1, ..., e_l, v_l)$ in G (D) we have $V(\mathfrak{p}) = \{v_1, v_2, ..., v_{l-1}\}$ and $E(\mathfrak{p}) = \{v_1, v_2, ..., v_{l-1}\}$ ($= A(\mathfrak{p})$). We will use the term **simple path** to denote paths without node repetition. A **(simple) cycle** is a (simple) path where the endnodes are identical. Two paths $\mathfrak{p}_1$ and $\mathfrak{p}_2$ are **node-disjoint** if $V(\mathfrak{p}_1) \cap V(\mathfrak{p}_2) = \emptyset$. Analogously, $\mathfrak{p}_1$ and $\mathfrak{p}_2$ are **edge-disjoint** (**arc-disjoint**) if $E(\mathfrak{p}_1) \cap E(\mathfrak{p}_2) = \emptyset$ ($= A(\mathfrak{p}_1) \cap A(\mathfrak{p}_2)$).

A graph $\tilde{G} = (\tilde{V}, \tilde{E})$ is a **subgraph** of $G = (V, E)$ if $\tilde{V} \subseteq V$ and $\tilde{E} \subseteq E$. A graph $G = (V, E)$ is said to be **connected** if there is a path between any two nodes. A graph G is **k-node(edge)-connected** if there exist k node (edge)-disjoint paths between any two nodes. A **tree** is a connected graph with no cycles. A **spanning tree** is a subgraph of G which has the same set of nodes of G and is a tree.

If $G = (V, E)$ is a graph and $X \subseteq V$, then the set of edges $\delta(X) := \big\{\{u, v\} \in E \mid u \in X, v \notin X\big\}$ is a **cut**. For a digraph $D = (V, A)$ and a subset $X \subseteq V$ of nodes, let $\delta(X)^+ := \{(u, v) \in A \mid u \in X, v \notin X\}$, $\delta(X)^- := \delta(V \backslash X)^+$ and $\delta(X) := \delta^+(X) \cup \delta)^-(X)$. The arcset $\delta(X)^+$ is called a **directed cut**.

2.3 Polyhedral Theory

A set $X \subseteq \mathbb{K}^n$ is **bounded**, if $M \in \mathbb{K}_+$ exists with $\| x \| \leq M$ for all $x \in X$ and some norm $\| \cdot \| \colon \mathbb{K}^n \to \mathbb{K}_+$. For $X \subseteq \mathbb{K}^n$ we define:

$$\begin{aligned} \text{aff}(X) &:= \{x \in \mathbb{K}^n \mid \exists \lambda_1, ..., \lambda_t \in \mathbb{K} \text{ and } \exists x_1, ..., x_t \in X, t \in \mathbb{N} \text{ s.t.} \\ &\qquad \textstyle\sum_{i=1}^t \lambda_i = 1 \text{ and } x = \sum_{i=1}^t \lambda_i x_i\}, \\ \text{conv}(X) &:= \{x \in \mathbb{K}^n \mid \exists \lambda_1, ..., \lambda_t \in \mathbb{K}_+ \text{ and } \exists x_1, ..., x_t \in X, t \in \mathbb{N} \text{ s.t.} \\ &\qquad \textstyle\sum_{i=1}^t \lambda_i = 1 \text{ and } x = \sum_{i=1}^t \lambda_i x_i\}, \end{aligned}$$

to be the **affine** and **convex hull** of X, respectively. The **dimension** $\dim(W)$ of W is $\dim(\text{aff}(X))$, that is the maximum number of linear independent vectors in $\text{aff}(X)$. If $\dim(W) = n$, we call W **full-dimensional**.

Given $a \in \mathbb{K}^n \backslash \{0\}$ and $\alpha \in \mathbb{K}$, the set $\{x \in \mathbb{K}^n \mid a^T x \leq \alpha\}$ is a **halfspace** and $\{x \in \mathbb{K}^n \mid a^T x = \alpha\}$ is a hyperplane. The finite intersection of halfspaces given by $\{x \in \mathbb{K}^n \mid Ax \leq b\}$ with $A \in \mathbb{K}^{m \times n}$ and $b \in \mathbb{K}^m$ is a **polyhedron**, where $\mathbb{K}^{m \times n}$ is the space of matrices with m rows and n columns. A bounded polyhedron is called a **polytope**.

For the polyhedron $\mathcal{P} = \{x \in \mathbb{K}^n \mid Ax \leq b\}$ and $M = \{1, ..., m\}$, the set $M^= = \{i \in M \mid a_i x = b_i \ \forall x \in \mathcal{P}\}$ is called the **equality set** of $\mathcal{P}$ and the set $M^\leq = M \backslash M^=$ the **inequality set**. Let $(A^=, b^=)$ and $(A^\leq, b^\leq)$ the corresponding rows of (A, b).

The inequality $a^T x \leq \alpha$ for $a \in \mathbb{K}^n$, $\alpha \in \mathbb{K}$ is **valid** for a polyhedron $\mathcal{P}$, if $\mathcal{P} \subseteq \{x \in \mathbb{K}^n \mid a^T x \leq \alpha\}$, and is **tight** for $\mathcal{P}$ if it is valid and $\mathcal{F}_{a,\alpha} = \mathcal{P} \cap \{x \in \mathbb{K}^n \mid a^T x = \alpha\} \neq \emptyset$. We say, $\mathcal{F}_{a,\alpha}$ is a **face** of $\mathcal{P}$ **induced** by $a^T x \leq \alpha$. A face $\mathcal{F} \neq \mathcal{P}$ of a polyhedron $\mathcal{P}$ is a **facet** of $\mathcal{P}$ if it is maximal with respect to inclusion. If $a^T x \leq \alpha$ is valid for $\mathcal{P}$ and $\mathcal{F} = \mathcal{P} \cap \{x \in \mathbb{K}^n \mid a^T x = \alpha\}$ is a facet of $\mathcal{P}$, we say that $a^T x \leq \alpha$ is **facet-defining**. An equivalent characterization of a facet is that $, \dim(\mathcal{F}) =, \dim(\mathcal{P}) - 1$. If a polyhedron $\mathcal{P}$ is full-dimensional, and $a^T x \leq \alpha$ and $b^T x \leq \beta$ are facet-defining with $\mathcal{F}_{a,\alpha} = \mathcal{F}_{b,\beta}$, then there exist $\lambda \in \mathbb{R}_+$ with $\lambda a = b$ and $\lambda \alpha = \beta$.
If the polyhedron $\mathcal{P}$ is not full-dimensional and $a^T x \leq \alpha$ and $b^T x \leq \beta$ represent the same facet, then one facet can be obtained from the other by multiplication by a positive scalar and then adding multiples of equations of the linear equation system $(A^=, b^=)$ of $\mathcal{P}$.

Chapter 3

The network design problem

In this chapter, we present the given problem description and point out possible real world relations. Afterwards, a mathematical model is constructed which models the specified network planning problem.

The initial objective of the ROCOCO project was to design an algorithm providing best solutions and/or lower bounds on average for all instances of the applied benchmark suite and all combinations of additional constraints. For each instance and combination a CPU limit of 10 minutes was given. A wide range of techniques have been tested, such as Constraint Programming, Local Search, Linear Programming, etc. The best published upper bounds so far have been obtained by a hybrid algorithm of Constraint Programming and Local Search, presented in [6]. A newer approach comes from Alain Chabrier who applied a Heuristic Branch-and-Price-and-Cut procedure [7]. But until now, only outlines of solution approaches exist in literature and no publicly accessible model including all constraints has been published so far.

3.1 Problem description

In this section, we introduce the information available to us, including an attempt at explanation towards reality and importance.

The network design and routing problem studied in the ROCOCO project can be described as follows: Given is a set of nodes and a set of arcs corresponding to all potential connections between the nodes. The graph has no parallel arcs and is bidirectional. There also exists a set of directed demands representing the communication demand between two locations (nodes), which have to be routed over the network along a single path from their origin to their destination node. When these demands are routed over some arc they consume capacity in the corresponding direction. The problem consists of selecting from a discrete set of possible capacities which one to install on each arc and how often. The capacities must be chosen in such a way, that all demands can be routed over the network simultaneously without exceeding the capacities and the

capacity installation cost is minimal. Also some additional constraints concerning the routing or the capacity choices are present and arbitrarily considered.
Each arc has an indexed set of directed base capacities, out of which at most one can be chosen. Associated arcs have the same number of base capacities and the corresponding choices are linked. The capacity can be increased by an integer multiplier coming out a certain range assigned by the base capacity. Thus the overall capacity of the arc is the product of the base capacity and the multiplier. The choices of an arc determine the choices of the associated backward arc. The multiplier of associated arcs has to be the same, as well as the index of the base capacity. This does not necessarily result in the same amount of capacity for both arcs, since the sets of base capacities may vary. The overall cost for both arcs results from the product of the cost of the base capacity and the chosen multiplier.

Example 3.1 For further explanation, consider the example shown in Figure 3.1. For arc (a, b) from node a to node b the base capacity $Capa_1(a, b)$ is given with the associated multiplier range from 0 to 3. This capacity is chosen, such that the choice for the arc (b, a) is imposed at the same time: (b, a) is equipped with $Capa_1(b, a)$. The multiplier for both is the same (in our case 3, the available maximum) and the overall costs for both were calculated by the cost of the base capacity $Capa_1(a, b)$ times 3. For the arc (d, a), the range of the multiplier associated to the base capacity $Capa_2(d, a)$ is from 1 to 2. Thus, if this capacity is chosen on the arc, it must be at least equipped once with this capacity.

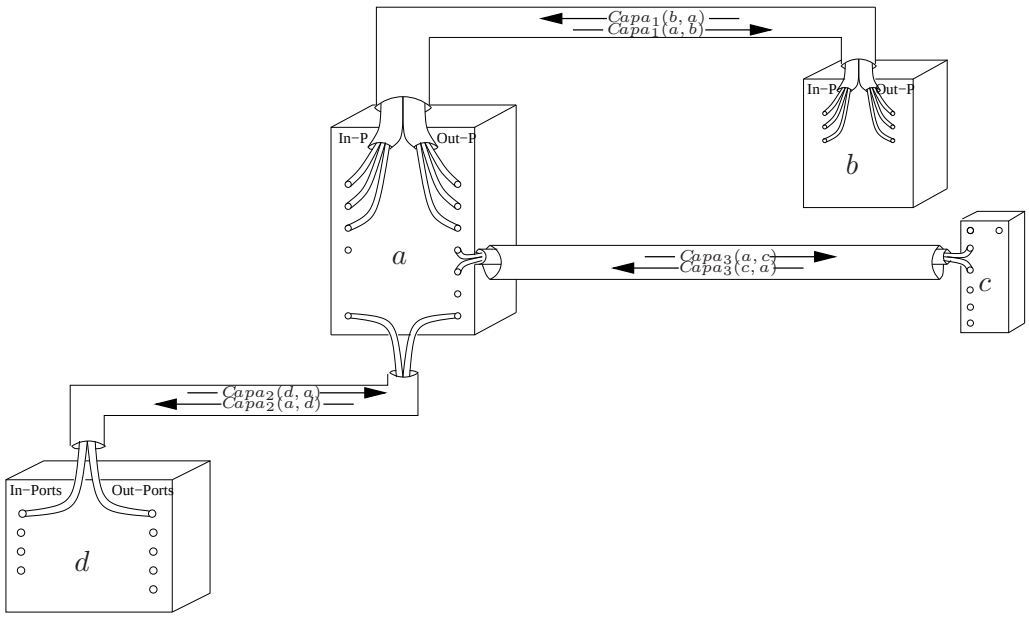


Figure 3.1: Example of network with capacity installation.

Part of the benchmark suite is also the consideration of six constraints which were added to the problem in arbitrary combinations.

sec Each node, capacity and demand has an indicator which determines its security level. It can be risky or secure. When this constraint is taken into account, a secure demand must only be routed over secure nodes and arcs which are equipped with secure capacity. For the remaining demands, this constraint is of no matter. Also the secure status of the endnodes of any demand is not relevant. This constraint can be interpreted regarding the anti-interception as well as the survivability aspect. For communication demands transporting sensitive data, there could exist a set of specially shielded nodes and capacities (like ssh- or https-connections). Alternatively and under the condition of the existence of secure, indestructible nodes and links (for which we have a guarantee that they never collapse), this constraint ensures the satisfiability of certain communication demands at any rate.

nomult This constraint forbids the multiplication of the base capacities of arcs. For each arc we have to consider two cases:

> If there exists a base capacity with a lower bound of the associated multiplier range greater than 0, this capacity has to be chosen with the minimal multiplier (if we want to install any capacity on this arc). The arc can not be equipped with any other base capacity.
>
> Otherwise the choice of capacity is free but a multiplier less than or equal to 1 is imposed.

This leads to the application of large capacities on the arcs instead of multiple smaller ones.

Example 3.2 We recall the example in Figure 3.1. It follows that for the arcs (a, d) and (d, a) the capacity and multiplier choice is imposed. Concerning the base capacity $Capa_2(a, d)$, the lower bound of the associated multiplier is greater than 0. Therefore the arcs (a, d) and (d, a) must be equipped with the base capacity $Capa_2(a, d)$ and $Capa_2(d, a)$ respectively, and the corresponding multiplier is 1.
For the arcs (a, b) and (b, a), the possible multiplier regarding the capacities $Capa_1(a, b)$ and $Capa_1(b, a)$ respectively, is 0 or 1 (and this only if there is no other base capacity available with a lower bound of the associated multiplier greater than 0).

Remark 3.3 The given problem description does not specify how to deal with two or more base capacities having a lower bound of the associated multiplier range greater than 0. We assume that this cannot happen.

symdem This constraint states that two demands where the origin of one is the destination of the other and vice versa, have to be routed symmetrically. Hence for a demand from node a to node b, if there exists a demand from b to a, then the paths used to route these demands must be symmetric, i.e. that the two paths were indicated by the same node set and differ only in an opposite order of the nodes. From this it follows that demands with the same origin and destination have to use the same path, if there exists at least one reverse demand for them. This symmetric routing is required by some routing protocols, e.g. the Network Time Protocol (NTP) [1] which is used to synchronize the time of devices in a network.

bmax Each demand has a parameter limiting the length of its routing path.
This limit is normally referred to as *hoplimit*, as it takes into account only the amount of arcs used by the path, and not its actual physical length. It avoids unacceptably long paths and decreases the probability of a failure of a path component or transmission delays.

pmax For each node an upper bound on incoming and outgoing ports is imposed. The capacities installed on the incoming arcs consume the inports amounting to their associated multipliers and analogously for the outgoing arcs and the outports. If the particular directed capacity is zero, no ports are consumed.
This constraint ensures that the ports are not exhausted. It makes sense if the node configuration is predetermined, as well as for survivability. If for a node the number of incident arcs on which capacity can be installed is restricted, only these arcs are affected in case of a failure of this node.

Example 3.4 Recall the example of Figure 3.1: for the arc (a, c) this means that the chosen capacity and associated multiplier require 2 inports at node c and 2 outports at node a. The base capacity installed on the associated backward arc (c, a) does not require any ports, because it does not provide positive capacity. Since the node c has only one outport, it follows that at most one outgoing arc can be equipped with a capacity, and thus used by some routing.

tmax This constraint associates an upper traffic bound to each node. The sum of all demands routed from, to or through a node must not exceed this fixed capacity. Like the **pmax** constraint, this one has two aspects. The configuration on the node may be predetermined or the network designer wants to route only a certain amount of data through the node to limit the probably vulnerable data.

Remark 3.5 In the ROCOCO project, all possible combinations of these additional constraints were considered. But the restriction to some, more plausible/probable variants may be worth considering.
The bundle of **tmax** and **pmax** constraints would be convenient in case of a predetermination of the technology installed at the nodes. If there exists a restriction on

the nodes it seems plausible that the number of ports is limited as well as the traffic. The **nomult** can be related to these constraints as well, because it reduces port consumption of the links regarding their endnodes and thus helps to fulfill the **pmax** constraint.
Concerning the survivability aspect, the combination of the **bmax** and the **sec** constraint seems reasonable. The **bmax** constraint decreases the probability of a failure of a path component and the **sec** constraint ensures the satisfiability of certain communication demands at any rate. Also the restrictions on the nodes (**tmax**, **pmax**) can be imposed in this context.

3.2 Model

In this section we introduce some notation and our mathematical model for the described problem. The constraints are classified in two subproblems.

Base model All demands have to be routed simultaneously over a single cycle-free path from their origin to their destination (monorouting). In the target network, sufficient capacity must be installed on the edges such that these can accommodate a feasible routing of the communication demands.

Additional constraints This part of the model is dedicated to the additional constraints. We derive inequalities for the **sec**, **pmax**, **bmax** and **tmax** constraints. At first we also deduce inequalities for the **symdem** constraint, but we finally integrate it into the data transformation, as well as the **nomult** requirements.

Remark 3.6 Due to the dependency of the capacity choice for a forward and the associated backward arc, we deviate from the arc view. We consider only undirected edges which can be equipped with linkdesigns providing directed capacities. It is only necessary to make one choice per edge, while the various capacities per direction and the multiplier are mapped to some linkdesigns. The result provides the amount of directed edge capacity.

3.2.1 Notation and parameters

Notation 3.7 (Telecommunication network) We denote the considered telecommunication network by a graph $G = (V, E)$, with V the set of nodes representing locations. The edges $e \in E$ represent the possible connections between the locations.

Notation 3.8 (Nodes) We denote the traffic limit of a node by the parameter *node capacity* which is consumed by the demands routed over. Each node $v \in V$ has the parameters:

$P_v^- \in \mathbb{Z}_+$ The amount of incoming ports.
$P_v^+ \in \mathbb{Z}_+$ The amount of outgoing ports.
$C_v \in \mathbb{Z}_+$ The provided capacity.
$S_v \in \{0,1\}$ The security status of a node is 1 if it is secure, 0 otherwise.

Notation 3.9 (Linkdesigns) The (finite) set of all linkdesigns which can be installed on an edge $\{u,v\} \in E$ is denoted by $\mathcal{L}_{\{u,v\}}$. Each edge can be equipped with at most one linkdesign. In the elements of the linkdesign set we merge the choice of the capacity and the associated multiplier. Therefore, for each edge and each base capacity of the associated arc arise as many linkdesigns as integers can be found in the associated multiplier range joint with the $\{0\}$-set. In that way we get two directed capacities per edge which are implied by the base capacities of the associated arcs and the multiplier. The transformation is analogously the same with the costs and the ports required. Other available base capacities for this arc should be dealt with analogously and the created linkdesigns should be added in the associated set $\mathcal{L}_{\{u,v\}}$.

At this point we already include the **nomult** constraint. If we take it into account, the set of linkdesigns is restricted to the linkdesigns with a port consumption of 1 or (when such one exists) the linkdesign with the corresponding multiplier lower bound greater than 0.

For a linkdesign $l \in \mathcal{L}_{\{u,v\}}$ corresponding to the edge $\{u,v\} \in E$, the following parameters are given:

$C_{uv}^l \in \mathbb{Z}_+$ The directed integer routing capacity from node u to node v.
$C_{vu}^l \in \mathbb{Z}_+$ The directed integer routing capacity from node v to node u.
$P_{uv}^l \in \mathbb{Z}_+$ The amount of ports required for the direction u to v i.e. the number of inports consumed at v and the number of outports consumed at u.
$P_{vu}^l \in \mathbb{Z}_+$ The amount of ports required for the direction v to u. i.e. the number of inports consumed at u and the number of outports consumed at v.
$S_{\{u,v\}}^l \in \{0,1\}$ The security status of a linkdesign is 1 if it is secure, 0 otherwise.
$W_{\{u,v\}}^l \in \mathbb{Z}_+$ The installation costs for linkdesign l on the edge $\{u,v\}$.

Remark 3.10 We have a parameter for the number of ports required for each direction. Actually the consumption of ports is equal for both directions, but the special case that the capacity for one direction is 0 while the other is greater than 0 makes it necessary to split the parameter.

Example 3.11 Recalling the example shown in Figure 3.1, consider the pair of arcs between the nodes a and b. For $Capa_1(a,b)$ a multiplier range from 0 to 3 is given. We assume that the base capacity value of $Capa_1(a,b)$ is α and the one of $Capa_1(b,a)$ is β, it is secure and the base cost is ϖ. Our transformation creates for each possible multiplier a different linkdesign, so that we get 4 linkdesigns for the edge $e = \{a,b\}$ out of the base

capacity $Capa_1(a,b)$. The edge capacity regarding a certain direction is the product of the base capacity of the associated arc with the concerned multiplier. The cost for the linkdesign is the multiplier times the cost for $Capa_1(a,b)$. The concerned multiplier also imposes the required ports (but the associated capacities must be regarded to ensure that it is not 0). In Table 3.1 we present the linkdesigns created from $Capa_1(a,b)$ and $Capa_1(b,a)$.

LD	C^l_{ab}	C^l_{ba}	P^l_{ab}	P^l_{ba}	$S^l_{\{a,b\}}$	$W^l_{\{a,b\}}$
l_0	0	0	0	0	1	0
l_1	α	β	1	1	1	ϖ
l_2	2α	2β	2	2	1	2ϖ
l_3	3α	3β	3	3	1	3ϖ

Table 3.1: Generated linkdesigns

Due to the **nomult** constraint, the set of linkdesigns $\mathcal{L}_{\{a,d\}}$ consists of only one element: The generated linkdesign corresponding to the base capacities $Capa_1(a,d)$ and $Capa_1(d,a)$ respectively, and the imposed multiplier 1.

Notation 3.12 (Commodities) In our model, all commodities consist of exactly one demand. In order to handle the secure routing condition we introduce a secure value for each commodity. This secure value is equal to the demand value when it has to be routed securely and otherwise 0. We denote by $\mathcal{K}$ the set of commodities. For a commodity $k \in \mathcal{K}$ the following parameters are specified:

$o_k \in V$ The node where the demand comes from (origin).
$d_k \in V$ The node to which the demand has to be routed to (destination).
$U_k \in \mathbb{Z}_+$ The amount of data that has to be routed from o_k to d_k.
$S_k \in \{0,1\}$ The security status of a commodity is 1 if it has to be routed securely, 0 otherwise.
$U^S_k \in \mathbb{Z}_+$ The amount of data that has to be routed from o_k to d_k securely, $U^S_k = U_k \cdot S_k$.
$M_k \in \mathbb{Z}_+$ The maximal number of hops, i.e. the allowed number of edges of an o_k-d_k-path.

Remark 3.13 In literature on multicommodity network flow problems, the demands are often aggregated with respect to their source (or destination) nodes. One commodity for each node with positive supply is defined [5]. The aggregated formulation has a smaller amount of commodities compared to our approach, which often leads to a significantly reduced size of the problem.
However, in our problem no such aggregation was possible since in the aggregated version, no proper control of the fulfillment of the varying demand properties is possible. Since the requirements on the demands are quite different (concerning the **sec**, **bmax** and **symdem** constraints) bundling demands would be impossible.

3.2.2 Variables

We formulate the described problem with the following variables, where one variable corresponds to the routing of commodities in the network and the other one to the installation of linkdesigns on an edge of the network.

Commodity edge flow variables

The routing of the commodities in our network is modeled via directed binary commodity edge flow variables. For each edge $\{u, v\} \in E$ and each commodity $k \in \mathcal{K}$, the binary variable f_{uv}^k decides whether k is routed over $\{u, v\}$ in the direction u to v or not. Analogous for the variable f_{vu}^k concerning the direction v to u. This yields for all $k \in \mathcal{K}$ and $\{u, v\} \in E$

$$f_{uv}^k = \begin{cases} 1 & \text{if } k \text{ is routed over } e \text{ in the direction } u \text{ to } v \\ 0 & \text{otherwise} \end{cases}$$

$$f_{vu}^k = \begin{cases} 1 & \text{if } k \text{ is routed over } e \text{ in the direction } v \text{ to } u \\ 0 & \text{otherwise} \end{cases}$$

Linkdesign installation variables

For each edge $e \in E$ and linkdesign $l \in \mathcal{L}_e$ we introduce the binary variable x_e^l indicating if l is installed on e. Therefore for all $e \in E$ and $l \in \mathcal{L}_e$

$$x_e^l = \begin{cases} 1 & \text{if } l \text{ is installed on } e \\ 0 & \text{otherwise} \end{cases}$$

Remark 3.14 Another alternative to modeling the choice of technology for the edges would be the application of an integer variable for all available associated base capacities: greater 0 iff the capacity is chosen and the value denoting the chosen multiplier. This would mean a significant decrease of linkdesigns but the range for these variables would be the union of $\{0\}$ and the multiplier range. The case of a base capacity with a lower bound of the corresponding multiplier range greater than 1 would be a problem. The 1 would be prohibited while 0 and the integral lower bound would be feasible. This gap would provoke problems in our further solution approaches.
The maximum multipliers in our data set are furthermore quite small (at most 5) and so the disprofit of the greater amount of $\{0, 1\}$-variables is not that serious. On the other hand, the binary range of the variables has a lot of advantages, as we will see later.

3.2.3 Inequalities

There exist various base constraints which have to be complied by a feasible routing and linkdesign installation. We first model these conditions in the following subsection and later introduce the inequalities that correspond to the additional constraints.

Inequalities for the base model

In this subsection we introduce the inequalities concerning the base model. We require all demands to be routed over a single cycle-free path from their origin to their destination. For each edge we can choose at most one linkdesign in a way such that the provided capacity can accommodate a feasible routing of all commodities simultaneously.

Linkdesign installation constraint. In this inequality we model the constraint that only one capacity can be chosen on each arc as stated in the given problem description. Therefore each edge $e \in E$ can only be equipped with at most one linkdesign $l \in \mathcal{L}_e$.

$$\sum_{l \in \mathcal{L}_e} x_e^l \leq 1 \quad \forall e \in E \tag{3.1}$$

Flow balance constraint. To ensure that the demand between two nodes is satisfied using a single path and to state that the flow balance is fulfilled we introduce this constraint.

$$\sum_{\{u,v\} \in \delta(v)} (f_{uv}^k - f_{vu}^k) = \begin{cases} -1 & \text{if } v = o_k \\ 1 & \text{if } v = d_k \\ 0 & \text{otherwise} \end{cases} \quad \forall v \in V, k \in \mathcal{K} \tag{3.2}$$

This constraint provides for each commodity $k \in \mathcal{K}$ a single o_k-d_k-path since the origin node o_k has a net outflow of 1, the destination node d_k a net inflow of 1 and exploiting the fact that we use binary commodity edge flow variables. The flow has no chance to get lost, because all other nodes have a net flow of 0. But in this path can occur cycles, that is why we introduce the following constraint.

Subtour elimination constraint. To avoid cycles in our routing, we restrict for each commodity $k \in \mathcal{K}$ and each subset W of V the number of edges in $E(W)$ which are used by this commodity.

$$\sum_{\{u,v\} \in E(W)} (f_{uv}^k + f_{vu}^k) \leq |W| - 1 \quad \forall W \subseteq V, k \in \mathcal{K} \tag{3.3}$$

These constraints prevent the occurrence of cycles, because a commodity $k \in \mathcal{K}$ cannot be routed twice through a node $v \in V$.

Edge capacity constraints. In the ROCOCO problem the arc capacities must not be exceeded. Consequentially, the directed capacity of an edge provided by the installed linkdesign must not be overstepped.

$$\sum_{k\in\mathcal{K}} U_k f_{uv}^k \leq \sum_{l\in\mathcal{L}_{\{u,v\}}} C_{uv}^l x_{\{u,v\}}^l \quad \forall\{u,v\}\in E \tag{3.4}$$

$$\sum_{k\in\mathcal{K}} U_k f_{vu}^k \leq \sum_{l\in\mathcal{L}_{\{u,v\}}} C_{vu}^l x_{\{u,v\}}^l \quad \forall\{u,v\}\in E \tag{3.5}$$

Inequalities for the additional constraints

In this subsection we incorporate the additional constraints in the mathematical model. The **nomult** constraints have already been included. We express the requirements of the **pmax**, **tmax** and **bmax** constraints by introducing new inequalities for the model. The **sec** constraint is split into two parts: one concerning the edges and the other considering the secure node throughput. The **symdem** constraint is integrated by a commodity merge process and the adaptation of some of the preestablished model constraints.

The next two inequalities include the **pmax** constraint in our model.

Inport constraints. Each node provides a certain number of inports which are consumed by the linkdesigns installed on the incident edges and must not be exhausted.

$$\sum_{\{u,v\}\in\delta(v)} \sum_{l\in\mathcal{L}_{\{u,v\}}} P_{uv}^l x_{\{u,v\}}^l \leq P_v^- \quad \forall v\in V \tag{3.6}$$

Outport constraints. Each node provides a certain number of outports which are consumed by the linkdesigns installed on the incident edges and must not be exhausted.

$$\sum_{\{u,v\}\in\delta(v)} \sum_{l\in\mathcal{L}_{\{u,v\}}} P_{vu}^l x_{\{u,v\}}^l \leq P_v^+ \quad \forall v\in V \tag{3.7}$$

The following type of inequalities integrates the **tmax** constraint in our model.

Node capacity constraints. The amount of data that is conducted through $v \in V$ is the sum of the values of the commodities that flow into v plus the commodities that have their origin in v. This throughput must not exceed the node capacity.

$$\sum_{k\in\mathcal{K}} \sum_{\{u,v\}\in\delta(v)} U_k f_{uv}^k \leq C_v - \sum_{k\in\mathcal{K}:v=o_k} U_k \quad \forall v\in V \tag{3.8}$$

Remark 3.15 This constraint differs from the usually considered switching capacity which gives a limit on the capacity of its incident edges and which is therefore normally modeled via linkdesign installation variables (e.g. [12]). The considered **tmax** takes into account the flow that is actually routed over the node and not only the flow that could be routed potentially like the switching capacity.

For an easier integration of the **sec** constraint we split it in two parts. One type of constraint focuses on the edges and the other type on the nodes.

Edge secure capacity constraints. If a secure demand $k \in \mathcal{K}$ with $S_k = 1$ is routed over an edge $e \in E$ it requires this edge to be equipped with a secure linkdesign $l \in \mathcal{L}_e$ with $S_e^l = 1$. We achieve this by introducing a secure capacity which is only provided by secure linkdesigns and consumed by secure commodities. Over an edge $e \in E$ regarding a certain direction, only as much secure data can be routed as directed secure capacity is provided. The amount of secure capacity is determined by the installed linkdesigns and their secure status.

$$\sum_{k \in \mathcal{K}} U_k^S f_{uv}^k \leq \sum_{l \in \mathcal{L}_{\{u,v\}}} S_{\{u,v\}}^l C_{uv}^l x_{\{u,v\}}^l \quad \forall \{u,v\} \in E \tag{3.9}$$

$$\sum_{k \in \mathcal{K}} U_k^S f_{vu}^k \leq \sum_{l \in \mathcal{L}_{\{u,v\}}} S_{\{u,v\}}^l C_{vu}^l x_{\{u,v\}}^l \quad \forall \{u,v\} \in E \tag{3.10}$$

Node secure capacity constraints. If a node $v \in V$ is risky, a secure commodity $k \in \mathcal{K}$ must not be routed over it. We ensure this constraint by prohibiting that a secure commodity is routed over edges incident to v (except the case that $v = o_k$ or $v = d_k$).

$$f_{uv}^k \leq 1 - S_k + \begin{cases} 1 & \text{if } \begin{array}{l} u \in \{o_k, d_k\}, \\ v \in \{o_k, d_k\} \end{array} \\ S_v & \text{if } \begin{array}{l} u \in \{o_k, d_k\}, \\ v \notin \{o_k, d_k\} \end{array} \\ S_u & \text{if } \begin{array}{l} u \notin \{o_k, d_k\}, \\ v \in \{o_k, d_k\} \end{array} \\ S_u S_v & \text{otherwise} \end{cases} \quad \forall \{u,v\} \in E, k \in \mathcal{K} \tag{3.11}$$

$$f_{vu}^k \leq 1 - S_k + \begin{cases} 1 & \text{if } \begin{array}{l} u \in \{o_k, d_k\}, \\ v \in \{o_k, d_k\} \end{array} \\ S_v & \text{if } \begin{array}{l} u \in \{o_k, d_k\}, \\ v \notin \{o_k, d_k\} \end{array} \\ S_u & \text{if } \begin{array}{l} u \notin \{o_k, d_k\}, \\ v \in \{o_k, d_k\} \end{array} \\ S_u S_v & \text{otherwise} \end{cases} \quad \forall \{u, v\} \in E, k \in \mathcal{K} \tag{3.12}$$

The constraint is modeled in these inequalities. Let $k \in K$ and $e \in E$. In case of a risky commodity ($S_k = 0$) they hold in any case, as well as when the origin and the destination of the commodity are equal to the endnodes of e. When the commodity k is secure the potential choice of edge e for routing depends on the security status of the endnodes. If one endnode is the origin or destination of k, e can only be used to route k if the other node is secure. In the case of a secure commodity where any of the endnodes of e is neither origin nor destination of it, the edge is prohibited to use for routing k if one of the node is not secure.

This constraint has actually not to be included in the formulation, because practically (3.11) and (3.12) are used in terms of variable reduction.

Path length limitation constraint. The **bmax** constraint is represented by this class of inequalities which claim that the number of edges over which a commodity $k \in \mathcal{K}$ is routed must not exceed the predetermined limit.

$$\sum_{\{u,v\} \in E} (f_{uv}^k + f_{vu}^k) \leq M_k \quad \forall k \in \mathcal{K} \tag{3.13}$$

In the next paragraph we describe the mechanism used to integrate the symmetrical routing constraint **symdem** in the model.

Symmetry constraints. First we present some notation to simplify the discussion: For $k, k' \in \mathcal{K}$:

- k, k' are **parallel** if $o_k = o_{k'}$ and $d_k = d_{k'}$.
- k, k' are **coupled** if $o_k = d_{k'}$ and $d_k = o_{k'}$.
- k, k' are **connected** if k, k' are parallel or coupled.
- k is **isolated** if no commodity is coupled with k.

We define commodity subsets, consisting of all commodities with the same origin and destination nodes:

$$\mathcal{K}_{pq} := \{k \in \mathcal{K} \mid o_k = p, d_k = q\} \quad \forall p, q \in V.$$

The **symdem** constraint demands that two coupled commodities have to be routed symmetrically. This means that if one commodity uses an edge $e \in E$ in a certain direction, the other commodity must use the same edge but in the opposite direction. This constraint also implies that parallel commodities have to use the same edges in equal direction if at least one coupled commodity exists. For simplification we assume from now on that there exist only parallel commodities if, at the same time, also at least one coupled commodity exists for them. Therefore we can formalize the requirements as follows:

$$f^k_{uv} = f^{k'}_{vu} \quad \forall\{u,v\} \in E, k, k' \in \mathcal{K} : k, k' \text{coupled} \tag{3.14}$$
$$f^k_{uv} = f^{k'}_{uv} \quad \forall\{u,v\} \in E, k, k' \in \mathcal{K} : k, k' \text{parallel} \tag{3.15}$$
$$f^k_{vu} = f^{k'}_{vu} \quad \forall\{u,v\} \in E, k, k' \in \mathcal{K} : k, k' \text{parallel} \tag{3.16}$$

We use the variable-reducing character of this constraint by modifying the data and expanding a part of our model inequalities. Merging connected commodities into a new commodity scales down our problem size.

In the following, we present the merging process concerning commodities respective to two nodes $p, q \in V$. We merge all commodities from $\mathcal{K}_{pq} \cup \mathcal{K}_{qp}$ in a new commodity k_{pq}. This commodity is uniquely identified by the node pair since it is finally the only commodity with a relation to both nodes. All properties of the commodities are assigned to the aggregation k_{pq}, whereby we always pass on the strongest property. Afterwards we present how the respective variables of the commodities $k \in \mathcal{K}_{pq} \cup \mathcal{K}_{qp}$ can be merged into new variables for k_{pq}. Finally we dwell on the changes of the predeveloped model constraints becoming necessary as a result of the merging process.

Passing properties In the following we explain how the properties of the connected commodities are passed on to the newly generated commodity. Let $p, q \in V$.

Origin/Destination The new commodity k_{pq} represents the communication demand between the nodes p and q. It is still directed with the origin p and the destination q, such that

$$o_{k_{pq}} = p,$$
$$d_{k_{pq}} = q.$$

Value The value of k_{pq} indicates the amount of data which has to be routed from p to q, hence it is the sum of the values of all commodities $k \in \mathcal{K}_{pq}$.

Concerning the communication demand from q to p we introduce for each commodity $k \in \mathcal{K}$ a new parameter **coupled value** $\overline{U}_k$. This parameter indicates the amount of data that has to be routed from the destination node d_k to the origin node o_k.

Considering the commodity k_{pq}, this means that the coupled value is the sum of the values of all commodities $k \in \mathcal{K}_{qp}$. Now it follows that

$$U_{k_{pq}} = \sum_{k \in \mathcal{K}_{pq}} U_k,$$
$$\overline{U}_{k_{pq}} = \sum_{k \in \mathcal{K}_{qp}} U_k.$$

Secure status All secure commodities have to be routed securely. So if at least one secure commodity $k \in \mathcal{K}_{pq} \cup \mathcal{K}_{qp}$ with $S_k = 1$ exists, all commodities have to be routed securely, and as a consequence also k_{pq}. Because of this condition, the secure value of k_{pq} is derived as followed:

$$S_{k_{pq}} = \max_{k \in \mathcal{K}_{pq} \cup \mathcal{K}_{qp}} S_k$$

Secure value The secure value of k_{pq} takes the amount of the value of k_{pq} if k_{pq} has to be routed securely, otherwise 0.

To cope with the reverse direction we introduce for each commodity $k \in \mathcal{K}$ the new parameter **coupled secure value** $\overline{U}_k^S$, equal to the coupled value $\overline{U}_k$ if k has to be routed securely, otherwise 0.

Therefore it follows that

$$U_{k_{pq}}^S = U_{k_{pq}} \cdot S_{k_{pq}},$$
$$\overline{U}_{k_{pq}}^S = \overline{U}_{k_{pq}} \cdot S_{k_{pq}}.$$

Hoplimit Since all commodities $k \in \mathcal{K}_{pq} \cup \mathcal{K}_{qp}$ have to use the same edges and since the path length limitation constraint has to be fulfilled for all k at the same time, the used path must contain at most the same amount of edges as the minimal hoplimit of all commodities:

$$M_{k_{pq}} = \min_{k \in \mathcal{K}_{pq} \cup \mathcal{K}_{qp}} M_k$$

Example 3.16 The merging procedure concering three connected commodities into one is presented in Table 3.2.

Merging variables The commodity k_{pq} has to be routed from $o_{k_{pq}}$ to $d_{k_{pq}}$ over the same path as all commodities $k \in \mathcal{K}_{pq}$. At the same time it has to use the same edges as all commodities $k \in \mathcal{K}_{qp}$, only in the opposite direction. This relation is formalized as follows:

$$f_{uv}^{k_{pq}} = f_{uv}^k \quad \forall \{u,v\} \in E, k \in \mathcal{K}_{pq} \tag{3.17}$$
$$f_{vu}^{k_{pq}} = f_{vu}^k \quad \forall \{u,v\} \in E, k \in \mathcal{K}_{pq} \tag{3.18}$$
$$f_{uv}^{k_{pq}} = f_{vu}^k \quad \forall \{u,v\} \in E, k \in \mathcal{K}_{qp} \tag{3.19}$$

commodity	o_k	d_k	U_k	$\overline{U}_k$	S_k	U_k^S	$\overline{U}_k^S$	M_k
k_1	p	q	12	0	0	0	0	6
k_2	p	q	5	0	0	0	0	2
k_3	q	p	13	0	1	13	0	4
k_{pq}	p	q	17	13	1	17	13	2

Table 3.2: Merging of connected commodities. *The three connected commodities k_1, k_2 and k_3 are merged in the new commodity k_{pq}*

Effects of the transformation on the model inequalities: After this transformation, we consider which of our model constraints are affected by the changes. Starting with the original commodities $k \in \mathcal{K}_{old}$ and assuming that the symmetric routing constraints (3.14)–(3.16) are fulfilled, we argue why it is equivalent considering only the new merged commodities $k \in \mathcal{K}_{new}$ according to the fulfillment of the model constraints including minor modifications.

Let $p, q \in V$. If $k_{pq} \in \mathcal{K}_{new}$ fulfills the flow balance constraints (3.2), these are fulfilled by all commodities $k \in \mathcal{K}_{pq} \cup \mathcal{K}_{qp} \subseteq \mathcal{K}_{old}$, subject to being routed symmetrically. This also holds for the subtour elimination constraints (3.3) and by construction of $M_{k_{pq}}$ and $S_{k_{pq}}$ also for the path length restriction (3.13) and node secure capacity constraints (3.11), (3.12). We only have to concentrate on the node and edge capacity constraints (3.4), (3.5) and (3.8)–(3.10). Now also the couple-(secure)-value has to be integrated.

Adaptation of the edge capacity constraints: For the expansion of the edge capacity constraints we consider the original commodity set $\mathcal{K}_{old}$. The commodities $k \in \mathcal{K}_{old}$ accomplish the symmetric routing constraints (3.14)–(3.16). We show how the edge capacity constraints have to be expanded and modified such that we only have to consider the new commodities $k \in \mathcal{K}_{new}$ generated by the merging process.

Let $v_i \in V,\ i = 1, ..., |V|$ and $\{u, v\} \in E$.

$$\begin{aligned}
\sum_{k \in \mathcal{K}_{old}} U_k f_{uv}^k &= \sum_{i=1}^{|V|} \sum_{j>i}^{|V|} \Big(\sum_{k \in \mathcal{K}_{v_i v_j}} U_k f_{uv}^k + \sum_{k \in \mathcal{K}_{v_j v_i}} U_k f_{uv}^k \Big) \\
&= \sum_{i=1}^{|V|} \sum_{j>i}^{|V|} \big(U_{k_{v_i v_j}} f_{uv}^k + \overline{U}_{k_{v_i v_j}} f_{vu}^k \big) \\
&= \sum_{k \in \mathcal{K}_{new}} U_k f_{uv}^k + \sum_{k \in \mathcal{K}_{new}} \overline{U}_k f_{vu}^k
\end{aligned}$$

Applying this analogously for the other direction, edges and also the secure edge capacity constraints, the modified capacity constraints for edges can be stated as fol-

lows:

$$\sum_{k\in\mathcal{K}} U_k f^k_{uv} + \sum_{k\in\mathcal{K}} \overline{U}_k f^k_{vu} \leq \sum_{l\in\mathcal{L}_{\{u,v\}}} C^l_{uv} x^l_{\{u,v\}} \qquad \forall\{u,v\}\in E \qquad (3.20)$$

$$\sum_{k\in\mathcal{K}} U_k f^k_{vu} + \sum_{k\in\mathcal{K}} \overline{U}_k f^k_{uv} \leq \sum_{l\in\mathcal{L}_{\{u,v\}}} C^l_{vu} x^l_{\{u,v\}} \qquad \forall\{u,v\}\in E \qquad (3.21)$$

$$\sum_{k\in\mathcal{K}} U^S_k f^k_{uv} + \sum_{k\in\mathcal{K}} \overline{U}^S_k f^k_{vu} \leq \sum_{l\in\mathcal{L}_{\{u,v\}}} S^l_{\{u,v\}} C^l_{uv} x^l_{\{u,v\}} \quad \forall\{u,v\}\in E \qquad (3.22)$$

$$\sum_{k\in\mathcal{K}} U^S_k f^k_{vu} + \sum_{k\in\mathcal{K}} \overline{U}^S_k f^k_{uv} \leq \sum_{l\in\mathcal{L}_{\{u,v\}}} S^l_{\{u,v\}} C^l_{vu} x^l_{\{u,v\}} \quad \forall\{u,v\}\in E \qquad (3.23)$$

Adaptation of the node capacity constraints: Concerning the capacity of a node $v \in V$ we now also have to consider the coupled value of the commodities that pass through v. We consider once more the original commodities $k \in \mathcal{K}_{old}$ fulfilling the symmetric routing constraints (3.14)–(3.16). We demonstrate that by use of the coupled value these commodities can be replaced by the new commodities $k \in \mathcal{K}_{new}$ generated by the merging process.

Let $v, v_i \in V,\ i = 1, ..., |V|$ with $v = v_{i^*}$, it follows that

$$\begin{aligned}
&\sum_{k\in\mathcal{K}_{old}} \sum_{\{u,v\}\in\delta(v)} U_k f^k_{uv} + \sum_{\substack{k\in\mathcal{K}_{old}\\ v=o_k}} U_k \\
&\quad= \sum_{i=1}^{|V|}\sum_{j>i}^{|V|} \sum_{\{u,v\}\in\delta(v)} \Big(\sum_{k\in\mathcal{K}_{v_iv_j}} U_k f^k_{uv} + \sum_{k\in\mathcal{K}_{v_jv_i}} U_k f^k_{uv}\Big) \\
&\qquad + \sum_{i>i^*}^{|V|} \sum_{\substack{k\in\mathcal{K}_{vv_i}\\ o_k=v}} U_k + \sum_{i<i^*}^{|V|} \sum_{\substack{k\in\mathcal{K}_{v_iv}\\ o_k=v}} U_k \\
&\quad= \sum_{i=1}^{|V|}\sum_{j>i}^{|V|} \sum_{\{u,v\}\in\delta(v)} U_{k_{v_iv_j}} f^k_{uv} + \sum_{i=1}^{|V|}\sum_{j>i}^{|V|} \sum_{\{u,v\}\in\delta(v)} \overline{U}_{k_{v_iv_j}} f^k_{vu} \\
&\qquad + \sum_{i>i^*}^{|V|} U_{k_{vv_i}} + \sum_{i<i^*}^{|V|} \overline{U}_{k_{v_iv}} \\
&\quad= \sum_{k\in\mathcal{K}_{new}} \sum_{\{u,v\}\in\delta(v)} U_k f^k_{uv} + \sum_{k\in\mathcal{K}_{new}} \sum_{\{u,v\}\in\delta(v)} \overline{U}_k f^k_{vu} \\
&\qquad + \sum_{\substack{k\in\mathcal{K}_{new}\\ v=o_k}} U_k + \sum_{\substack{k\in\mathcal{K}_{new}\\ v=d_k}} \overline{U}_k.
\end{aligned}$$

Therefore the modified node capacity constraints can be written:

$$\sum_{k\in\mathcal{K}}\sum_{\{u,v\}\in\delta(v)} U_k f^k_{uv} + \sum_{k\in\mathcal{K}:v=o_k} U_k + \sum_{k\in\mathcal{K}}\sum_{\{u,v\}\in\delta(v)} \overline{U}_k f^k_{vu} + \sum_{k\in\mathcal{K}:v=d_k} \overline{U}_k \le C_v \quad \forall v\in V \tag{3.24}$$

Taking into account the flow balance constraints (3.2) we can prove the following claim:

Claim 3.17 *Let $v \in V$, it follows that*

$$\sum_{k\in\mathcal{K}}\sum_{\{u,v\}\in\delta(v)} \overline{U}_k f^k_{vu} + \sum_{k\in\mathcal{K}:v=d_k} \overline{U}_k = \sum_{k\in\mathcal{K}}\sum_{\{u,v\}\in\delta(v)} \overline{U}_k f^k_{uv} + \sum_{k\in\mathcal{K}:v=o_k} \overline{U}_k \tag{3.25}$$

Proof.

$$\begin{aligned}
&\sum_{k\in\mathcal{K}}\sum_{\{u,v\}\in\delta(v)} \overline{U}_k f^k_{vu} + \sum_{k\in\mathcal{K}:v=d_k} \overline{U}_k \\
&\quad= \sum_{\substack{k\in\mathcal{K}\\ v\neq o_k, v\neq d_k}} \overline{U}_k \sum_{\{u,v\}\in\delta(v)} f^k_{vu} + \sum_{\substack{k\in\mathcal{K}\\ v=o_k}} \overline{U}_k \sum_{\{u,v\}\in\delta(v)} f^k_{vu} \\
&\qquad+ \sum_{\substack{k\in\mathcal{K}\\ v=d_k}} \overline{U}_k \sum_{\{u,v\}\in\delta(v)} f^k_{vu} + \sum_{\substack{k\in\mathcal{K}\\ v=d_k}} \overline{U}_k \sum_{\{u,v\}\in\delta(v)} (f^k_{uv} - f^k_{vu}) \\
&\quad= \sum_{\substack{k\in\mathcal{K}\\ v\neq o_k, v\neq d_k}} \overline{U}_k \Big(\sum_{\{u,v\}\in\delta(v)} f^k_{vu} + \sum_{\{u,v\}\in\delta(v)} (f^k_{uv} - f^k_{vu}) \Big) \\
&\qquad+ \sum_{\substack{k\in\mathcal{K}\\ v=d_k}} \overline{U}_k \Big(\sum_{\{u,v\}\in\delta(v)} f^k_{vu} + \sum_{\{u,v\}\in\delta(v)} (f^k_{uv} - f^k_{vu}) \Big) \\
&\qquad+ \sum_{\substack{k\in\mathcal{K}\\ v=o_k}} \overline{U}_k \sum_{\{u,v\}\in\delta(v)} (f^k_{vu} - f^k_{uv} + f^k_{uv}) \\
&\quad= \sum_{\substack{k\in\mathcal{K}\\ v\neq o_k v,\neq d_k}} \overline{U}_k \sum_{\{u,v\}\in\delta(v)} f^k_{uv} + \sum_{\substack{k\in\mathcal{K}\\ v=d_k}} \overline{U}_k \sum_{\{u,v\}\in\delta(v)} f^k_{uv} \\
&\qquad+ \sum_{\substack{k\in\mathcal{K}\\ v=o_k}} \overline{U}_k \sum_{\{u,v\}\in\delta(v)} f^k_{uv} + \sum_{\substack{k\in\mathcal{K}\\ v=o_k}} \overline{U}_k \sum_{\{u,v\}\in\delta(v)} (f^k_{vu} - f^k_{uv}) \\
&\quad= \sum_{k\in\mathcal{K}}\sum_{\{u,v\}\in\delta(v)} \overline{U}_k f^k_{uv} + \sum_{k\in\mathcal{K}:v=o_k} \overline{U}_k
\end{aligned}$$

□

After this short observation, we can finally formulate the new modified node capacity constraints. Combining (3.24) and (3.25) it follows:

$$\sum_{k\in\mathcal{K}}\sum_{\{u,v\}\in\delta(v)} (U_k + \overline{U}_k) f^k_{uv} + \sum_{k\in\mathcal{K}:v=o_k} (U_k + \overline{U}_k) \le C_v \quad \forall v\in V. \tag{3.26}$$

Remark 3.18 Note that, when the **symdem**-constraint is not considered, the couple-(secure)-values are all 0, so that we have the same constraints as in (3.4),(3.5) and (3.8)–(3.10).

Remark 3.19 At first sight the **symdem**-constraints (3.14)-(3.16) seem to promise the possibility to halve the amount of directed edge flow variables by simply considering the undirected routing case. This is also proposed in the existing literature concerning the given problem [2]. This would only be reasonable in the case of symmetrical homogeneous data, or more precisely, if for all nodes $p, q \in V$, all edges $\{u, v\} \in E$ and linkdesigns $l \in \mathcal{L}_{\{u,v\}}$

- $\sum_{k \in \mathcal{K}_{pq}} U_k = \sum_{k \in \mathcal{K}_{qp}} U_k$
- $M_{k_1} = M_{k_2} \quad \forall k_1, k_2 \in \mathcal{K}_{pq} \cup \mathcal{K}_{qp}$
- $S_{k_1} = S_{k_2} \quad \forall k_1, k_2 \in \mathcal{K}_{pq} \cup \mathcal{K}_{qp}$
- $C^l_{uv} = C^l_{vu}$

For our model it is impossible to apply this simplification to the undirected case. We illustrate this by a short example:

Example 3.20 Let $p_1, p_2, q_1, q_2 \in V$, $e = \{u, v\} \in E$, $\mathcal{L}_e = \{l\}$. The commodities $k_1 \in \mathcal{K}_{p_1 q_1}$ and $k_2 \in \mathcal{K}_{p_2 q_2}$ are isolated with values $U_{k_1} = U_{k_2} = \alpha$. The linkdesign l provides capacity α for both directions ($C^l_{uv} = C^l_{vu} = \alpha$). We assume that e is equipped with l.
In our model the directions in which the commodities are routed over e ascertain the feasibility of this routing regarding the respective edge capacity constraint. If k_1 and k_2 are routed in the same direction over e, it is not feasible, whereas if they are routed in oppositional directions, it would be feasible. We can not cope with this difficulty in an undirected model with edge flow variables, even though it would have meant an easier reduction in terms of the problem size.

3.2.4 Objective function

Since the overall cost for two associated arcs has been bundled in the linkdesign parameter W^l_e, the objective of the minimization of the overall costs can be formalized as follows:

$$\min \sum_{e \in E} \sum_{l \in \mathcal{L}_e} W^l_e x^l_e$$

3.3 The model

$$\min \sum_{e\in E}\sum_{l\in\mathcal{L}_e} W_e^l x_e^l$$

$$\begin{array}{lcll}
\sum_{l\in\mathcal{L}_e} x_e^l & \leq & 1 & \forall e\in E \\
\sum_{\{u,v\}\in\delta(v)}(f_{uv}^k - f_{vu}^k) & = & \left\{\begin{array}{ll} -1 & \text{if } v=o_k \\ 1 & \text{if } v=d_k \\ 0 & \text{otherwise}\end{array}\right. & \forall v\in V, k\in\mathcal{K} \\
\sum_{\{u,v\}\in E(S)}(f_{uv}^k + f_{vu}^k) & \leq & |S|-1 & \forall S\subseteq V, k\in\mathcal{K} \\
\sum_{k\in\mathcal{K}} U_k f_{uv}^k + \sum_{k\in\mathcal{K}} \overline{U}_k f_{vu}^k & \leq & \sum_{l\in\mathcal{L}_{\{u,v\}}} C_{vu}^l x_{\{u,v\}}^l & \forall\{u,v\}\in E \\
\sum_{k\in\mathcal{K}} U_k f_{vu}^k + \sum_{k\in\mathcal{K}} \overline{U}_k f_{uv}^k & \leq & \sum_{l\in\mathcal{L}_{\{u,v\}}} C_{uv}^l x_{\{u,v\}}^l & \forall\{u,v\}\in E \\
\sum_{k\in\mathcal{K}} U_k^S f_{uv}^k + \sum_{k\in\mathcal{K}} \overline{U}_k^S f_{vu}^k & \leq & \sum_{l\in\mathcal{L}_{\{u,v\}}} S_{\{u,v\}}^l C_{vu}^l x_{\{u,v\}}^l & \forall\{u,v\}\in E \\
\sum_{k\in\mathcal{K}} U_k^S f_{vu}^k + \sum_{k\in\mathcal{K}} \overline{U}_k^S f_{uv}^k & \leq & \sum_{l\in\mathcal{L}_{\{u,v\}}} S_{\{u,v\}}^l C_{uv}^l x_{\{u,v\}}^l & \forall\{u,v\}\in E \\
\sum_{k\in\mathcal{K}}\sum_{\{u,v\}\in\delta(v)}(U_k+\overline{U}_k) f_{uv}^k & \leq & C_v - \sum_{\substack{k\in\mathcal{K}\\ v=o_k}} (U_k+\overline{U}_k) & \forall v\in V \\
\sum_{\{u,v\}\in\delta(v)}\ \sum_{l\in\mathcal{L}_{\{u,v\}}} P_{uv}^l x_{\{u,v\}}^l & \leq & P_v^- & \forall v\in V \\
\sum_{\{u,v\}\in\delta(v)}\ \sum_{l\in\mathcal{L}_{\{u,v\}}} P_{vu}^l x_{\{u,v\}}^l & \leq & P_v^+ & \forall v\in V \\
\sum_{e=\{u,v\}\in E}(f_{uv}^k + f_{vu}^k) & \leq & M_k & \forall k\in\mathcal{K}
\end{array}$$

$$\begin{array}{ll}
f_{uv}^k \in \{0,1\} & \forall\{u,v\}\in E, k\in\mathcal{K} \\
f_{vu}^k \in \{0,1\} & \forall\{u,v\}\in E, k\in\mathcal{K} \\
x_e^l \in \{0,1\} & \forall e\in E, l\in\mathcal{L}_e
\end{array}$$

We define the polyhedron associated with this integer linear program and its LP-relaxation as

$$\mathcal{P} := conv\left\{(x,f)\in\{0,1\}^{\sum_{e\in E}|\mathcal{L}_e|}\times\{0,1\}^{2\times|E|\times|\mathcal{K}|}\,\middle|\,(x,f)\text{ satisfies }\begin{array}{c}(3.1)-(3.3),(3.6),(3.7),\\(3.13),(3.20)-(3.26)\end{array}\right\}$$

$$\mathcal{LP} := conv\left\{(x,f)\in[0,1]^{\sum_{e\in E}|\mathcal{L}_e|}\times[0,1]^{2\times|E|\times|\mathcal{K}|}\,\middle|\,(x,f)\text{ satisfies }\begin{array}{c}(3.1)-(3.3),(3.6),(3.7),\\(3.13),(3.20)-(3.26)\end{array}\right\}$$

Chapter 4

Classes of valid inequalities

In this chapter we present different types of valid inequalities. We give some examples to show that in special cases they really cut off some fractional solution points and which conditions have to be fulfilled such that some are even facet defining.

4.1 GUB cover inequalities

Anyone of our model inequalities can be represented as a **knapsack inequality**, which is an inequality of the form

$$a^T x = \sum_{j \in J} a_j x_j \leq b \text{ with } x \in \{0,1\}^{|J|} \tag{4.1}$$

In [9] it was shown that these inequalities can be used to derive valid inequalities. These so called **cover inequalities** are subsets of the variables in the inequality such that if all variables were set to 1, the inequality would be violated, but if any one variable were excluded, the inequality would be satisfied. Inequalities of the form of the sum of the variables in the cover must be less than or equal to the size of the cover less one describe the relationship of the variables in the cover. These inequalities have been extended to knapsacks with generalized upper bounds [18]. These so callled **GUB** constraint for a set of binary variables is the sum of variables less than or equal to one. If the variables in a GUB constraint are also members of a knapsack constraint, then the minimal cover can be selected with the additional consideration that at most one of the members of the GUB constraint can be one in a solution. This additional restriction makes the GUB cover inequalities stronger (that is, more restrictive) than ordinary cover inequalities.

In our model arise three types of GUBs, one are the linkdesign installation constraints (3.1) which consider the linkdesign installation variables:

$$\sum_{l \in \mathcal{L}_e} x_e^l \leq 1 \quad \forall e \in E \tag{4.2}$$

The other two classes of GUBs are deduced by the subtour elimination constraints (3.3) and the third also by the flow balance equalities (3.2). Both consider sets of commodity edge flow variables and we apply them alternately:

$$f^k_{uv} + f^k_{vu} \leq 1 \quad \forall k \in \mathcal{K}, \{u,v\} \in E \tag{4.3}$$

$$\sum_{\{u,v\} \in \delta(v)} f^k_{uv} \leq 1 \quad \forall k \in \mathcal{K}, v \in V \tag{4.4}$$

First we generally describe how the so called **GUB cover inequalities** are deduced.

4.1.1 Valid inequalities from knapsack and GUB

Consider the 0-1 knapsack set $H = \{x \in \{0,1\}^{|J|} \mid \sum_{j \in J} a_j x_j \leq b\}$. Assuming that all $a_j \neq 0$ (erasing previously all x_j with $a_j = 0$), we can split J into two disjoint subsets J_P and J_N, where P stands for *positive* and N for *negative*. J_P consists of the indices with the corresponding coefficient being positive, and for J_N accordingly negative. We can now rewrite $H = \{x \in \{0,1\}^{|J|} \mid \sum_{j \in J_P} \overline{a}_j x_j - \sum_{j \in J_N} \overline{a}_j x_j \leq b\}$ with all coefficients $\overline{a}_j > 0$ ($\overline{a}_j = |a_j| \; \forall j \in J = J_P \cup J_N$). Using the presence of some GUBs, these subsets J_P and J_N can be further split into subsets $\mathbb{G}_i$ tying together all variables affected by one GUB. Note that we only apply GUBs that bind together exclusively the variables of the respective uppersets and that these GUBs are pairwise disjoint with each other. When some variable x_j does not appear in a GUB we apply the trivial GUB $x_j \leq 1$ since we only consider binary variables. We denote by I_P and I_N the index sets for the used GUBs, such that:

$$J_P = \bigcup_{i \in I_P} \mathbb{G}_i \,, \quad J_N = \bigcup_{i \in I_N} \mathbb{G}_i \quad \text{and} \quad \mathbb{G}_i \cap \mathbb{G}_j = \emptyset \quad \forall i,j \in I = I_P \cup I_N$$

Now we consider the knapsack polytope with explicit GUB constraints

$$X_{GUB} := \left\{ x \in \{0,1\}^{|J|} \;\middle|\; \sum_{j \in J_P} \overline{a}_j x_j - \sum_{j \in J_N} \overline{a}_j x_j \leq b, \; \sum_{j \in \mathbb{G}_i} x_j \leq 1 \;\; \forall i \in I = I_P \cup I_N \right\}$$

We say that $C := C_P \cup C_N$ is a **GUB cover** if

1. $C_P \subseteq J_P$ and $C_N \subseteq J_N$
2. $|C_P \cap \mathbb{G}_i| \leq 1 \quad \forall i \in I_P$ and $|C_N \cap \mathbb{G}_i| \leq 1 \quad \forall i \in I_N$
3. $\sum_{j \in C_P} \overline{a}_j - \sum_{j \in C_N} \overline{a}_j > b$

With the GUB cover C, we associate the sets:

$$I_P^+ := \{i \in I_P \mid C_P \cap \mathbb{G}_i \neq \emptyset\} \tag{4.5}$$
$$I_N^+ := \{i \in I_N \mid C_N \cap \mathbb{G}_i \neq \emptyset\} \tag{4.6}$$
$$\mathbb{G}_i^+ := \{j \in \mathbb{G}_i : \overline{a}_j \geq \overline{a}_l \quad \text{for } l \in C_P \cap \mathbb{G}_i\} \quad \forall i \in I_P^+ \tag{4.7}$$
$$\mathbb{G}_i^+ := \{j \in \mathbb{G}_i : \overline{a}_j \leq \overline{a}_l \quad \text{for } l \in C_N \cap \mathbb{G}_i\} \quad \forall i \in I_N^+ \tag{4.8}$$

With these definitions, the following was derived in [18]:

Proposition 4.1 *The inequality*

$$\sum_{i \in I_P^+} \sum_{j \in \mathbb{G}_i^+} x_j \leq |C_P| - 1 + \sum_{i \in I_N^+} \sum_{j \notin \mathbb{G}_i^+} x_j + \sum_{i \in I_N \setminus I_N^+} \sum_{j \in \mathbb{G}_i} x_j \tag{4.9}$$

is valid for X_{GUB}.

For the special case that there exist only positive coefficients in the knapsack inequality, stronger inequalities were introduced. For the GUB cover C, let

$$E(C) := \{k \in J_P \mid \overline{a}_k \geq \overline{a}_j \quad \forall j \in C\} \setminus \bigcup_{i \in I_P^+} \mathbb{G}_i^+. \tag{4.10}$$

Proposition 4.2 *The inequality*

$$\sum_{i \in I_P^+} \sum_{j \in \mathbb{G}_i^+} x_j + \sum_{j \in E(C)} x_j \leq |C_P| - 1 \tag{4.11}$$

is valid for X_{GUB} *if* $J_N = \emptyset$.

We use this class of inequalities in the following variants:

4.1.2 Edge (secure) capacity constraints

Edge capacity constraints: For an edge $\{u, v\} \in E$ we first concern the direction u to v. In this case the underlying knapsacks $a^T x \leq b$ are the inequalities (3.20),

$$\sum_{k \in \mathcal{K}} U_k f_{uv}^k + \sum_{k \in \mathcal{K}} \overline{U}_k f_{vu}^k - \sum_{l \in \mathcal{L}_{\{u,v\}}} C_{uv}^l x_{\{u,v\}}^l \leq 0 \quad \forall \{u, v\} \in E \tag{4.12}$$

We apply the linkdesign GUB (4.2) and commodity edge flow variable GUB (4.3), such that for the above constraints we have one negative and $|\mathcal{K}|$ positive GUBs. The $\overline{a}_j$ correspond to the capacity of the installable linkdesigns and the value (coupled value) of a commodity routed over $\{u, v\}$ in direction u to v (v to u), respectively.

A GUB cover corresponds to the election of two commodity subsets $K_{uv} \subseteq \mathcal{K}$ and $K_{vu} \subseteq \mathcal{K}$ (with $K_{uv} \cap K_{vu} = \emptyset$) and the choice of a particular linkdesign $\hat{l}$ for the edge $\{u, v\}$. The commodities from K_{uv} (K_{vu}) are routed over $\{u, v\}$ in direction u to v (v to u). Let

$$\hat{K}_{uv} = K_{uv} \cup \{k \in K_{vu} \mid U_k \geq \overline{U}_k\} \tag{4.13}$$

$$\hat{K}_{vu} = K_{vu} \cup \{k \in K_{uv} \mid \overline{U}_k \geq U_k\}. \tag{4.14}$$

The resulting inequality,

$$\sum_{k \in \hat{K}_{uv}} f^k_{uv} + \sum_{k \in \hat{K}_{vu}} f^k_{vu} - \sum_{\substack{l \in \mathcal{L}_{\{u,v\}}: \\ C^l_{uv} > C^{\hat{l}_{\{u,v\}}}_{uv}}} x^l_{\{u,v\}} \leq |K_{uv}| + |K_{vu}| - 1 \tag{4.15}$$

states that, choosing the cover commodities to be routed over the edge $\{u, v\}$, a linkdesign with higher directed capacity than the cover capacity has to be installed.

An analogous inequality follows for the direction v to u.

Example 4.3 Consider an instance with three nodes $V = \{v_1, v_2, v_3\}$, three edges $E = \big\{\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}\big\}$ and two commodities $\mathcal{K} = \{k_1, k_2\}$. The commodities have the following demand values: $U_{k_1} = U_{k_2} = 3$ and $\overline{U}_{k_1} = \overline{U}_{k_2} = 0$ respectively. For the edge $\{v_2, v_3\}$ the linkdesign set consists of two linkdesigns $\mathcal{L}_{\{v_2,v_3\}} = \{l_1, l_2\}$ providing capacity for the direction v_2 to v_3 in amount of $C^{v_2v_3}_{l_1} = 5$ and $C^{v_2v_3}_{l_1} = 10$ respectively.
After solving the LP-relaxation, the optimal solution results in $f^{k_1}_{v_2v_3} = f^{k_2}_{v_2v_3} = 0.8$ and $x^{l_1}_{\{v_2,v_3\}} = 1$, fulfilling the corresponding edge capacity inequality. Regarding the cover for the edge $\{v_2, v_3\}$, direction v_2 to v_3, represented by the commodities k_1 and k_2 and the linkdesign l_1, we obtain the inequality

$$f^{k_1}_{v_2v_3} + f^{k_2}_{v_2v_3} - x^{l_2}_{\{v_2,v_3\}} \leq 1. \tag{4.16}$$

This GUB cover inequality cuts off the current non-integral solution.

Edge secure capacity constraints: In analogy to the edge capacity constraints above, we derive GUB cover inequalities from the secure constraints (3.22) and (3.23).

4.1.3 Port constraints

Inport constraints: In this case the underlying knapsacks $a^T x \leq b$ are the inequalities (3.6),

$$\sum_{\{u,v\} \in \delta(v)} \sum_{l \in \mathcal{L}_{\{u,v\}}} P^l_{uv} x^l_{\{u,v\}} \leq P^-_v \quad \forall v \in V \tag{4.17}$$

Applying the linkdesign installation GUB (4.2) we obtain that we have $|\delta(v)|$ positive GUBs and no negative GUB for these constraints. The $\overline{a}_j$ correspond to the inport consumption of the installable linkdesigns.

A GUB cover corresponds to the choice of a particular linkdesigns $\hat{l}_{\{u,v\}}$ on each edge $\{u,v\}$ of a subset $E' \subseteq \delta(v)$. We define the maximal inport consumption of all edges and linkdesigns in the cover:

$$P_{max}(C) := \max_{\{u,v\}\in E'} P_{uv}^{\hat{l}_{\{u,v\}}} \tag{4.18}$$

The resulting inequality,

$$\sum_{\{u,v\}\in E'} \sum_{\substack{l\in\mathcal{L}_{\{u,v\}}:\\ P_{uv}^{l}\geq P_{uv}^{\hat{l}_{\{u,v\}}}}} x_{\{u,v\}}^{l} + \sum_{\{u,v\}\in\delta(v)\setminus E'} \sum_{\substack{l\in\mathcal{L}_{\{u,v\}}:\\ P_{uv}^{l}\geq P_{max}(C)}} x_{\{u,v\}}^{l} \leq |E'| - 1 \tag{4.19}$$

states that at least at one edge in E', a linkdesign with a lower inport consumption than in the cover has to be installed.

Example 4.4 Consider the instance from Example 4.3. For v_1 the number of inports is $P_{v_1}^{-} = 5$. The set of linkdesigns for the edges $\{v_1, v_2\}$ and $\{v_1, v_3\}$ are the following:

$$\mathcal{L}_{\{v_1,v_2\}} = \{l_1, l_2\} \text{ with } P_{v_2v_1}^{l_1} = 3 \text{ and } P_{v_2v_1}^{l_2} = 4$$
$$\mathcal{L}_{\{v_1,v_3\}} = \{l'\} \text{ with } P_{v_3v_1}^{l'} = 3.$$

Solving the LP-relaxation results in an optimal solution with $x_{\{v_1,v_2\}}^{l_2} = x_{\{v_1,v_3\}}^{l'} = 0.8$ fulfilling the corresponding inport inequality. The cover concerning v_1 consists of the edges $\{v_1, v_2\}$ and $\{v_1, v_3\}$ and the linkdesigns l_1, l' respectively. The implied inport GUB cover inequality

$$x_{\{v_1,v_2\}}^{l_1} + x_{\{v_1,v_2\}}^{l_2} + x_{\{v_1,v_3\}}^{l'} \leq 1 \tag{4.20}$$

cuts off the current fractional solution.

Outport constraints: Analogous inequalities follow from the outport inequalities (3.7).

4.1.4 Node capacity constraints

Node capacity constraints: In this case the underlying knapsacks $a^T x \leq b$ are the inequalities (3.8),

$$\sum_{k\in\mathcal{K}} \sum_{\substack{e\in\delta(v)\\ e=\{u,v\}}} (U_k + \overline{U}_k) f_{uv}^{k} \leq C_v - \sum_{k\in\mathcal{K}:v=o_k} (U_k + \overline{U}_k) \quad \forall v \in V \tag{4.21}$$

Utilizing the commodity edge flow variable GUB (4.4) it results that for the above constraints we have no negative and $|\mathcal{K}|$ positive GUBs. The $\overline{a}_j$ correspond to the overall demand values $U_k + \overline{U}_k$.

A GUB cover corresponds to the election of a commodity subset $K \subseteq \mathcal{K}$ and the choice of a particular edge $\hat{e}_k$ of $\delta(v)$ for each commodity $k \in K$, over which it is routed into the node v. We define the maximal overall demand value for the commodities in the cover:

$$U_{max}(C) := \max_{k \in K}(U_k + \overline{U}_k) \tag{4.22}$$

The resulting inequality,

$$\sum_{k \in K} \sum_{\{u,v\} \in \delta(v)} f_{uv}^k + \sum_{\substack{k \in \mathcal{K} \setminus K: \\ U_k + \overline{U}_k \geq U_{max}(C)}} \sum_{\{u,v\} \in \delta(v)} f_{uv}^k \leq |K| - 1 \tag{4.23}$$

states that number of commodities of the cover (or commodities with even higher overall demand values) which are routed over the node v must be less than the number of commodities in the cover.

Example 4.5 Consider the instance from Example 4.3 with the there presented commodities k_1 and k_2. The node v_1 has a capacity of $C_{v_1} = 5$.
The optimal solution of the LP-relaxation contains the values $f_{v_1v_3}^{k_1} = f_{v_1v_3}^{k_2} = 0.8$, fulfilling the node capacity constraint for v_1. The GUB cover consists of the commodity k_1 with the edge $\{v_1, v_2\}$ and k_2 with $\{v_1, v_3\}$ respectively. We obtain the GUB cover inequality

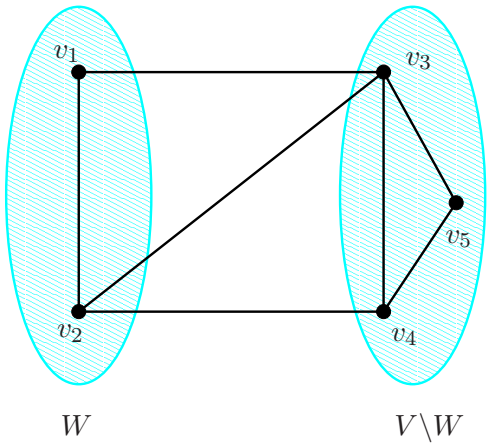
$$\sum_{\{u,v_1\} \in \delta(v_1)} f_{uv_1}^{k_1} + \sum_{\{u,v_1\} \in \delta(v_1)} f_{uv_1}^{k_2} \leq 1 \tag{4.24}$$

cutting off the current fractional solution.

These GUB Cover inequalities are all valid for our feasibility space since the knapsack and GUB inequalities are all part of our model.

4.2 Cutset inequalities

In this section we present the types of inequalities which arise from the consideration of cuts.



	o_k	d_k	U_k	$\overline{U}_k$	U_k^S	$\overline{U}_k^S$
k_1	v_1	v_3	100	0	0	0
k_2	v_3	v_2	200	0	0	0
k_3	v_2	v_4	20	00	20	00
k_3	v_5	v_2	30	0	30	0

Figure 4.1: Cut traffic

Consider the node subset $W \subseteq V$. Commodities crossing the cut $\delta(W)$ impose certain requirements on the capacity provided by some linkdesigns installed on the edges of $\delta(W)$.
Using Figure 4.1, we present these conditions. The amount of traffic that has to be routed from W to $V\backslash W$ is 120. Thus, a overall capacity of at least 120 must be provided by the linkdesigns installed on the edges $\{v_1, v_3\}$, $\{v_2, v_3\}$, and $\{v_2, v_4\}$ with respect to the direction out of W.
Taking into account the single path condition, we state that the above commodities k_1,k_2,k_3, and k_4 must cross the cut using at least one edge $e \in \delta(W)$. Considering the commodities k_1 and k_2 with their relatively high data values, it follows that at least one edge $\{v_1, v_3\}$, $\{v_2, v_3\}$, or $\{v_2, v_4\}$ must be equipped with a linkdesign providing at least a capacity of 100 w.r.t. the direction out of W, to route commodity k_1. Another edge $e \in \delta(W)$ must be equipped with a linkdesign providing at least a capacity of 200 w.r.t. the direction into the node subset W, such that commodity k_2 can be routed over the cut.

In the following we specify these requirements. At first we define useful denominations:

Definition 4.6 (Sourced commodities) Let $W \subseteq V$. A commodity is called W *sourced commodity* for the node subset W, if its origin is in W and its destination in $V \setminus W$. Define

$$\mathcal{K}_{W^+} := \{k \in \mathcal{K} \mid o_k \in W, d_k \in V \setminus W\}$$

the set of W *sourced commodities*.

Definition 4.7 (Targeted commodities) Let $W \subseteq V$. A commodity is called W *targeted commodity* with respect to the node subset W, if its destination lays in W

and its origin in $V \setminus W$. Define

$$\mathcal{K}_{W^-} := \{k \in \mathcal{K} \mid d_k \in W, o_k \in V \setminus W\}$$

the set of *W targeted commodities*.

4.2.1 Cut traffic inequalities

This class of inequalities states that the directed (secure) capacity across a cut must be at least as large as the (secure) demand data across the cut. First we define this cut-crossing traffic in the following:

Definition 4.8 ((Secure) outbound traffic) Let $W \subseteq V$. We define

$$T^+(W) := \sum_{k \in \mathcal{K}_{W^+}} U_k + \sum_{k \in \mathcal{K}_{W^-}} \overline{U}_k$$

respectively

$$T^{S+}(W) := \sum_{k \in \mathcal{K}_{W^+}} U_k^S + \sum_{k \in \mathcal{K}_{W^-}} \overline{U}_k^S$$

to be the amount of *(secure) outbound traffic* of W.

Let $W \subseteq V$. Considering the normal data we obtain the following inequality

$$\sum_{\substack{\{u,v\} \in E: \\ u \in W, v \in V \setminus W}} \sum_{l \in \mathcal{L}_{\{u,v\}}} C_{uv}^l x_{\{u,v\}}^l \geq T^+(W) \tag{4.25}$$

Likewise, we can generate inequalities concerning the outbound traffic which has to be routed securely. We obtain the inequality

$$\sum_{\substack{\{u,v\} \in E: \\ u \in W, v \in V \setminus W}} \sum_{l \in \mathcal{L}_{\{u,v\}}} S_{\{u,v\}}^l C_{uv}^l x_{\{u,v\}}^l \geq T^{S+}(W) \tag{4.26}$$

Validity of (4.25) and (4.26) for the considered polyhedron $\mathcal{P}$ is obvious. We apply the above inequalities (4.25) and (4.26) in terms of base knapsack inequalities and employ the same procedures as in Section 4.1. As GUB we use the linkdesign installation constraints (3.1).

4.2.2 Particular commodity inequalities

Investigating the cut we can also deal with the single path routing constraint. All commodities crossing the cut have to use at least one edge entirely, since the flow must not be splitted. Considering *remarkable* commodities (in terms of (secure) amount of demand value) this fact imposes the requirement that at least one edge must be equipped with a linkdesign providing enough capacity for a remarkable commodity. To simplify the argumentation, we first define the following:

Definition 4.9 (Maximum (secure) outbound value) Concerning the node subset W we define the largest (secure) single communication demand from W to $V \setminus W$ Let $W \subseteq V$:

$$U_{max}^{W^+} := \max\{\max_{k \in K_{W^+}} U_k, \max_{k \in K_{W^-}} \overline{U}_k\}$$

respectively

$$U_{max}^{W_S^+} := \max\{\max_{k \in K_{W^+}} U_k^S, \max_{k \in K_{W^-}} \overline{U}_k^S\}$$

the *maximum (secure) outbound value* of W.

Definition 4.10 (Maximum (secure) inbound value) Concerning the node subset W we denote the biggest (secure) single communication demand from $V \setminus W$ to W. Let $W \subseteq V$. Define

$$U_{max}^{W^-} := \max\{\max_{k \in K_{W^-}} U_k, \max_{k \in K_{W^+}} \overline{U}_k\}$$

respectively

$$U_{max}^{W_S^-} := \max\{\max_{k \in K_{W^-}} U_k^S, \max_{k \in K_{W^+}} \overline{U}_k^S\}$$

the *maximum (secure) inbound value* of W.

Definition 4.11 ((Secure) outbound sufficient linkdesign) A linkdesign is called *(secure) outbound sufficient* with respect to a node subset W, if it provides as least as much (secure) capacity as the maximum (secure) outbound value of W. Let $W \subseteq V$, $\{u, v\} \in \delta(W)$ with $u \in W, v \in V \setminus W$, and let

$$\mathcal{L}_{\{u,v\}}^{W^+} := \{l \in \mathcal{L}_{\{u,v\}} \mid C_{uv}^l \geq U_{max}^{W^+}\}$$

respectively

$$\mathcal{L}_{\{u,v\}}^{W_S^+} := \{l \in \mathcal{L}_{\{u,v\}} \mid S_{\{u,v\}}^l C_{uv}^l \geq U_{max}^{W^+}\}$$

be the set of *(secure) outbound sufficient linkdesigns* for the node subset W.

Definition 4.12 ((Secure) inbound sufficient linkdesign) A linkdesign is called *(secure) inbound sufficient* with respect to a node subset W, if it provides as least as much (secure) capacity as the maximum (secure) inbound value of W. Let $W \subseteq V$, $\{u, v\} \in \delta(W)$ with $u \in W, v \in V \setminus W$, and let

$$\mathcal{L}_{\{u,v\}}^{W^-} := \{l \in \mathcal{L}_{\{u,v\}} \mid C_{uv}^l \geq U_{max}^{W^+}\}$$

respectively

$$\mathcal{L}_{\{u,v\}}^{W_S^-} := \{l \in \mathcal{L}_{\{u,v\}} \mid S_{\{u,v\}}^l C_{uv}^l \geq U_{max}^{W^+}\}$$

be the set of *(secure) inbound sufficient linkdesigns* for the node subset W.

We derive the inequalities

$$\sum_{\{u,v\}\in\delta(W)} \sum_{l\in\mathcal{L}^{W^+}_{\{u,v\}}} x^l_{\{u,v\}} \geq 1 \tag{4.27}$$

$$\sum_{\{u,v\}\in\delta(W)} \sum_{l\in\mathcal{L}^{W^+_S}_{\{u,v\}}} x^l_{\{u,v\}} \geq 1 \tag{4.28}$$

$$\sum_{\{u,v\}\in\delta(W)} \sum_{l\in\mathcal{L}^{W^-}_{\{u,v\}}} x^l_{\{u,v\}} \geq 1 \tag{4.29}$$

$$\sum_{\{u,v\}\in\delta(W)} \sum_{l\in\mathcal{L}^{W^-_S}_{\{u,v\}}} x^l_{\{u,v\}} \geq 1 \tag{4.30}$$

Proposition 4.13 *The above inequalities are valid for $\mathcal{P}$.*

Proof. Assume that $\hat{p} = (\hat{x}, \hat{f})$ does not satisfy the inequality (4.27), i.e. all edges in $\delta(W)$ provide less capacity than $U^{W^+}_{max}$. On the other hand the commodity corresponding to $U^{W^+}_{max}$ has to cross the cut unsplitted, i.e. has to use one single edge. Hence the edge capacity constraint is violated concerning this particular edge and therefore $\hat{p}$ is not feasible for $\mathcal{P}$. Likewise for the other inequalities. □

The inequalities (4.27)–(4.30) can be combined. Consider the case that we have four different commodities, each remarkable in its own way. Due to the availability of linkdesigns, these four commodities may be forced to be routed over four different edges of the cut. Let $W \subseteq V$ and four commodities $k_1, k_2, k_3, k_4 \in \mathcal{K}$, where k_1 is the corresponding commodity of $U^{W^+}_{max}$, k_2 of $U^{W^+_S}_{max}$, k_3 of $U^{W^-}_{max}$, and k_4 of $U^{W^-_S}_{max}$. When at the same time $\mathcal{L}^{W^+}_e \cap \mathcal{L}^{W^+_S}_e \cap \mathcal{L}^{W^-}_e \cap \mathcal{L}^{W^-_S}_e = \emptyset \quad \forall e \in \delta(W)$ then we obtain the inequality

$$\sum_{e\in\delta(W)} \sum_{l\in\mathcal{L}^{W^+}_e \cup \mathcal{L}^{W^+_S}_e \cup \mathcal{L}^{W^-}_e \cup \mathcal{L}^{W^-_S}_e} x^l_e \geq 4 \tag{4.31}$$

Example 4.14 We consider the node subset $W \subseteq V$ from Figure 4.1 and suppose that $\mathcal{L}_{\{v_1v_3\}} = \mathcal{L}_{\{v_2v_3\}} = \mathcal{L}_{\{v_2v_4\}} = \{l_1, l_2, l_3, l_4\}$ with the properties presented in Table 4.1.

In this case all commodities k_1, k_2, k_3, and k_4 have to use different edges to cross the cut. Thus each edge $e \in \delta(W)$ must be equipped with an other linkdesign $l \in \{l_1, l_2, l_3, l_4\}$.

	C^l_{uv}	C^l_{vu}	$S^l_{\{u,v\}}$
l_1	200	0	0
l_2	0	200	0
l_3	50	0	1
l_4	0	50	1

.

Table 4.1: Available linkdesigns for $e \in \delta(W)$

4.3 Edge capacity inequalities

We now present a class of inequalities that are valid for our considered polyhedron. First, we introduce some notation and definitions to simplify the discussion and prove validity. Then we consider a relaxation of our problem and identify the members of the class which define facets of the associated polyhedron.

4.3.1 Setup

Definition 4.15 (Arc-inclusive paths) Let $a \in A$ an arc of the overlaying digraph $D(G)$ of G (as defined in Section 2.2) and $s, t \in V$. An s-t-path is called *a-inclusive* if it contains a. Let

$$\mathfrak{P}^{a-incl}_{st} := \{\mathfrak{p} \text{ path from } s \text{ to } t \mid a \in \mathfrak{p}\}$$

be the set of *a-inclusive* s-t-paths.

Definition 4.16 (Arc-exclusive paths) Let $a \in A$ be an arc of the overlaying digraph $D(G)$ of G and $s, t \in V$. An s-t-path is called *a-exclusive* if it does not contain a. We define

$$\mathfrak{P}^{a-excl}_{st} := \{\mathfrak{p} \text{ path from } s \text{ to } t \mid a \notin \mathfrak{p}\}$$

as the set of *a-exclusive* s-t-paths.

Definition 4.17 (Edge-exclusive cycle) Let $e \in E$. A cycle C is called *e-exclusive* if it is simple and does not contain e. Let

$$\mathfrak{C}^{e-excl} := \{C \text{ simple cycle in } G \mid e \notin C\}$$

be the set of *e-exclusive* cycles.

Definition 4.18 (Oversized linkdesign) Let $\{u, v\} \in E$. A linkdesign is called *oversized* if it provides enough capacity such that it cannot be exceeded by any routing of the commodities $k \in \mathcal{K}$, even when the routing contains cycles. Let

$$\mathcal{L}^{max}_{\{u,v\}} := \{l \in \mathcal{L}_{\{u,v\}} \mid \sum_{k \in \mathcal{K}} (U_k + \overline{U}_k) \leq C^l_{uv}, \sum_{k \in \mathcal{K}} (U_k + \overline{U}_k) \leq C^l_{vu}\}$$

be the set of *oversized* linkdesigns for $\{u,v\} \in E$.

Definition 4.19 (Sufficient linkdesign) Let $\{u,v\} \in E$, $K_{uv}, K_{vu} \subseteq \mathcal{K}$. A linkdesign $l \in \mathcal{L}_{\{u,v\}}$ is called *sufficient* with respect to the commodity subsets K_{uv} and K_{vu}, if it provides enough capacity that all commodities $k \in K_{uv}$ can be routed from u to v and all commodities $k \in K_{vu}$ from v to u simultaneously. Accordingly we define

$$\mathcal{L}^{+}_{K_{uv}K_{vu}} := \{l \in \mathcal{L}_{\{u,v\}} \mid \sum_{k\in K_{uv}} U_k + \sum_{k\in K_{vu}} \overline{U}_k \leq C^l_{uv}, \sum_{k\in K_{uv}} \overline{U}_k + \sum_{k\in K_{vu}} U_k \leq C^l_{vu}\}$$

to be the set of *sufficient* linkdesigns for the commodity subsets K_{uv} and K_{vu}.

Definition 4.20 (Insufficient linkdesign) Let $\{u,v\} \in E$, $K_{uv}, K_{vu} \subseteq \mathcal{K}$. A linkdesign $l \in \mathcal{L}_{\{u,v\}}$ is called *insufficient* with respect to the commodity subsets K_{uv} and K_{vu}, if it is not sufficient. Accordingly we define

$$\mathcal{L}^{-}_{K_{uv}K_{vu}} := \mathcal{L}_{\{u,v\}} \backslash \mathcal{L}^{+}_{K_{uv}K_{vu}}$$

to be the set of *insufficient* linkdesigns for the commodity subsets K_{uv} and K_{vu}.

Definition 4.21 (Large linkdesign) Let $\{u,v\} \in E$, $K_{uv}, K_{vu} \subseteq \mathcal{K}$. A linkdesign $l \in \mathcal{L}_{\{u,v\}}$ is called *large* with respect to the commodity subsets K_{uv} and K_{vu}, if it is sufficient and furthermore provides enough capacity such that all commodities $k \in \mathcal{K}\backslash(K_{uv} \cup K_{vu})$ can be routed over $\{u,v\}$ in arbitrary directions simultaneously. Accordingly we define

$$\mathcal{L}^{large}_{K_{uv}K_{vu}} := \{l \in \mathcal{L}_{\{u,v\}} \mid \sum_{k\in K_{uv}} U_k + \sum_{k\in K_{vu}} \overline{U}_k + \sum_{k\in\mathcal{K}\backslash(K_{uv}\cup K_{vu})} \max\{U_k, \overline{U}_k\} \leq C^l_{uv}, \sum_{k\in K_{uv}} \overline{U}_k + \sum_{k\in K_{vu}} U_k + \sum_{k\in\mathcal{K}\backslash(K_{uv}\cup K_{vu})} \max\{U_k, \overline{U}_k\} \leq C^l_{vu}\}$$

the set of *large* linkdesigns for the commodity subsets K_{uv} and K_{vu}.

Obviously, $\mathcal{L}^{max}_{\{u,v\}} \subseteq \mathcal{L}^{large}_{K_{uv}K_{vu}} \subseteq \mathcal{L}^{+}_{K_{uv}K_{vu}} \subseteq \mathcal{L}_{\{u,v\}}$ holds for each $\{u,v\} \in E$, $K_{uv}, K_{vu} \subseteq \mathcal{K}$.

Definition 4.22 (Close linkdesign) Let $\{u,v\} \in E$, $K_{uv}, K_{vu} \subseteq \mathcal{K}$. A linkdesign $l \in \mathcal{L}_{\{u,v\}}$ is called *uv-close* with respect to the commodity subsets K_{uv} and K_{vu}, if it is insufficient, but provides enough capacity such that as soon as any commodity $k^* \in K_{uv}$ is not routed from u to v, all other commodities $k \in K_{uv}\backslash\{k^*\}$ can be routed from u to v, all commodities $k \in K_{vu}$ from v to u, and in addition, all commodities

$k \in \mathcal{K}\backslash(K_{uv} \cup K_{vu})$ can be route over $\{u, v\}$ in arbitrary directions simultaneously. Accordingly we define

$$\begin{aligned}\mathcal{L}^{uv-close}_{K_{uv}K_{vu}} := \{l \in \mathcal{L}^{-}_{K_{uv}K_{vu}} \mid & \sum_{k\in K_{uv}\backslash\{k^*\}} U_k + \sum_{k\in K_{vu}\cup\{k^*\}} \overline{U}_k \\ & + \sum_{k\in\mathcal{K}\backslash(K_{uv}\cup K_{vu})} \max\{U_k, \overline{U}_k\} \leq C^l_{uv}, \\ & \sum_{k\in K_{uv}\backslash\{k^*\}} \overline{U}_k + \sum_{k\in K_{vu}\cup\{k^*\}} U_k \\ & + \sum_{k\in\mathcal{K}\backslash(K_{uv}\cup K_{vu})} \max\{U_k, \overline{U}_k\} \leq C^l_{vu} \quad \forall k^* \in K_{uv}\}\end{aligned}$$

to be the set of *uv-close* linkdesigns for the commodity subsets K_{uv} and K_{vu}.

Analogously we define $\mathcal{L}^{vu-close}_{K_{uv}K_{vu}}$, the set of *vu-close* linkdesigns for the commodity subsets K_{uv} and K_{vu}. These linkdesigns are insufficient, but provide enough capacity such that as soon as any commodity $k^* \in K_{vu}$ is not routed from v to u all other commodities can be routed accordingly.

Example 4.23 It seems un-apparent that these uv-close linkdesigns can exist. Therefore we present an example. Let $G = (V, E)$ with $\{u, v\} \in E$ and $\mathcal{K} = \{k_1, k_2, k_3, k_4\}$ with the properties presented in Table 4.2. We choose $K_{uv} = \{k_1, k_2\}$ and $K_{vu} = \{k_3\}$. It results that the linkdesign l^* with the capacity values $C^{l^*}_{uv} = C^{l^*}_{vu} = 80$ is uv-close and vu-close for the commodity subsets K_{uv} and K_{vu}.

	U_k	$\overline{U}_k$
k_1	30	0
k_2	30	0
k_3	10	30
k_4	10	10

Table 4.2: Existing commodities

4.3.2 Inequality

We now introduce a class of inequalities that are valid for our considered polyhedron. The inequalities were motivated by the fact that the routing of a set of commodities over an edge $e \in E$ forbids the installation of certain linkdesigns on e.

Proposition 4.24 *Let $\{u, v\} \in E$, $K_{uv}, K_{vu} \subseteq \mathcal{K}$ with $K_{uv} \cap K_{vu} = \emptyset$,then*

$$\sum_{k\in K_{uv}} f^k_{uv} + \sum_{k\in K_{vu}} f^k_{vu} + \sum_{l\in\mathcal{L}^{-}_{K_{uv}K_{vu}}} x^l_{\{u,v\}} \leq |K_{uv}| + |K_{vu}| \tag{4.32}$$

is a valid inequality for $\mathcal{P}$, stating that when we want to use an edge for the routing of some commodities, it must be equipped with an adequate linkdesign.

Proof. Assume that an integer point $\hat{p} = (\hat{x}, \hat{f}) \in \mathcal{P}$ does not satisfy (4.32) for an edge $\{u, v\} \in E$ and the commodity subsets K_{uv} and $K_{vu} \in \mathcal{K}$ where $K_{uv} \cap K_{vu} = \emptyset$. The only way to violate the inequality is to route all commodities $k \in K_{uv}$ over $\{u, v\}$ in direction u to v, all commodities $k \in K_{vu}$ over $\{u, v\}$ in direction u to v, and equip at the same time $\{u, v\}$ with a linkdesign $\hat{l} \in \mathcal{L}^{-}_{K_{uv}K_{vu}}$, which is insufficient with respect to K_{uv} and K_{vu}. This implies

$$\begin{aligned} \hat{f}^k_{uv} &= 1 \quad \forall k \in K_{uv} \\ \hat{f}^k_{vu} &= 1 \quad \forall k \in K_{vu} \\ \hat{x}^{\hat{l}}_{\{u,v\}} &= 1. \end{aligned}$$

In this case, $\hat{p}$ does not satisfy one of the edge capacity constraints (3.20) or (3.21) concerning $\{u, v\}$:

$$\sum_{k\in\mathcal{K}} U_k \hat{f}^k_{uv} + \sum_{k\in\mathcal{K}} \overline{U}_k \hat{f}^k_{vu} \geq \sum_{k\in K_{uv}} U_k + \sum_{k\in K_{vu}} \overline{U}_k > C^{\hat{l}}_{uv} = \sum_{l\in\mathcal{L}_{\{u,v\}}} C^l_{uv} \hat{x}^l_{\{u,v\}}$$

or

$$\sum_{k\in\mathcal{K}} U_k \hat{f}^k_{vu} + \sum_{k\in\mathcal{K}} \overline{U}_k \hat{f}^k_{uv} \geq \sum_{k\in K_{vu}} U_k + \sum_{k\in K_{uv}} \overline{U}_k > C^{\hat{l}}_{vu} = \sum_{l\in\mathcal{L}_{\{u,v\}}} C^l_{vu} \hat{x}^l_{\{u,v\}}$$

so that $\hat{p} \notin \mathcal{P}$, which is a contradiction. □

4.3.3 Relaxation

Due to the fact that our problem with its high number of different constraints is quite complicated, the determination of facets resulted to be a bold venture. Therefore, we concentrate on a relaxation of our problem. We restrict ourselves to the problem of finding a feasible routing for our commodity set $\mathcal{K}$ and a feasible linkdesign installation. We do not consider any port, hoplimit, secure or node capacity constraints and do allow cycles. We define the polyhedra associated with this integer linear program by

$$\mathcal{P}_{base} := conv\left\{ (x, f) \in \{0,1\}^{\sum_{e\in E} |\mathcal{L}_e|} \times \{0,1\}^{2\times|E|\times|\mathcal{K}|} \;\middle|\; (x, f) \text{ satisfies } \begin{matrix} (3.1), (3.2), \\ (3.20), (3.21) \end{matrix} \right\} \tag{4.33}$$

Among the inequalities (4.32), some are facet-defining for this polyhedron. The following proposition identifies some of these:

Proposition 4.25 *Let $\{u, v\} \in E$, $K_{uv}, K_{vu} \subseteq \mathcal{K}$. If*

1. *$G = (V, E)$ is 3-connected (for every two nodes $i, j \in V$, there exist three node disjoint paths from i to j)*
2. *$\mathcal{L}_e^{max} \neq \emptyset \quad \forall e \in E$*
3. *$\mathcal{L}_{\{u,v\}} \subseteq (\mathcal{L}^{uv-close}_{K_{uv}K_{vu}} \cup \mathcal{L}^{vu-close}_{K_{uv}K_{vu}} \cup \mathcal{L}^{large}_{K_{uv}K_{vu}})$*
4. *$\mathcal{L}^{uv-close}_{K_{uv}K_{vu}} \cap \mathcal{L}^{vu-close}_{K_{uv}K_{vu}} \neq \emptyset$,*

then (4.32) defines a facet of $\mathcal{P}_{base}$.

Proof. Validity of (4.32) for $\mathcal{P}_{base}$ follows from Proposition 4.24 (since each point in $\mathcal{P}_{base}$ has to fulfill the edge capacity constraints). Let

$$\mathcal{F} := \left\{ (x, f) \in \mathcal{P}_{base} \;\middle|\; \sum_{k \in K_{uv}} f^k_{uv} + \sum_{k \in K_{vu}} f^k_{vu} + \sum_{l \in \mathcal{L}^-_{K_{uv}K_{vu}}} x^l_{\{u,v\}} = |K_{uv}| + |K_{vu}| \right\} \quad (4.34)$$

be the face of $\mathcal{P}_{base}$ induced by (4.32). We prove that $\mathcal{F}$ is a facet of $\mathcal{P}_{base}$ in three steps: First, $\mathcal{F} \neq \emptyset$ is shown by representing points in $\mathcal{F}$. Second, we show that any face containing $\mathcal{F}$ is either $\mathcal{P}_{base}$ or $\mathcal{F}$ itself. Third, we show that $\mathcal{P}_{base} \neq \mathcal{F}$ by representing a point in $\mathcal{P}_{base} \backslash \mathcal{F}$.

Claim 4.26 *The face $\mathcal{F}$ is not empty*

For each two nodes $v_1, v_2 \in V$ there exits a v_1-v_2-path in the overlaying directed graph D of G (as defined in Section 2.2). We construct a point $\tilde{p}$ by considering for each commodity $k \in \mathcal{K} \backslash (K_{uv} \cup K_{vu})$ a o_k-d_k-path and setting the respective commodity edge flow variables to 1. For each commodity $k \in K_{uv}$ we consider an (u, v)-inclusive o_k-d_k-path and for all $k \in K_{vu}$ an (v, u)-inclusive o_k-d_k-path, and fixate the corresponding commodity edge flow variables to 1. Finally we equip all edges $e \in E$ with an oversized linkdesign $l \in \mathcal{L}_e^{max}$, so that we fulfill all edge capacity constraints and derive that $\tilde{p} \in \mathcal{F}$.

Claim 4.27 *Any inequality valid for $\mathcal{P}_{base}$ representing $\mathcal{F}$ is a positive multiple of (4.32) plus a linear combination of (3.2)*

At first we group the points in $\mathcal{F}$ and access the similarities within these groups later. Then we proof the claim by representing points in $\mathcal{F}$ and combining the flow balance constraints.

There exist three different types of points in $\mathcal{F}$:

- All commodities $k \in K_{uv}$ are routed over the edge $\{u, v\}$ in the direction from u to v and all commodities $k \in K_{vu}$ from v to u. In this case $\{u, v\}$ is equipped with a linkdesign $l \in \mathcal{L}^{large}_{K_{uv}K_{vu}} = \mathcal{L}_{\{u,v\}} \backslash \mathcal{L}^-_{K_{uv}K_{vu}}$.

- Exactly one commodity $k^* \in K_{uv}$ is not routed from u to v, while the rest of the commodities $k \in K_{uv} \backslash \{k^*\}$ are routed over the edge $\{u, v\}$ in the direction from u to v and all commodities $k \in K_{vu}$ from v to u. In this case $\{u, v\}$ is equipped with a uv-close linkdesign $l \in \mathcal{L}^{uv-close}_{K_{uv}K_{vu}} \subseteq \mathcal{L}^{-}_{K_{uv}K_{vu}}$.
- Exactly one commodity $k^* \in K_{vu}$ is not routed from v to u, while the rest of the commodities $k \in K_{vu} \backslash \{k^*\}$ are routed over the edge $\{u, v\}$ in the direction from v to u and all commodities $k \in K_{uv}$ from u to v. In this case $\{u, v\}$ is equipped with a vu-close linkdesign $l \in \mathcal{L}^{vu-close}_{K_{uv}K_{vu}} \subseteq \mathcal{L}^{-}_{K_{uv}K_{vu}}$.

We notice that the routing of any commodity $k \in \mathcal{K} \backslash (K_{uv} \cup K_{vu})$ does not influence the capacity on the edge $\{u, v\}$. In all three types of points of $\mathcal{F}$ any routing of the commodities of $\mathcal{K} \backslash (K_{uv} \cup K_{vu})$ neither consumes nor releases a notable amount of capacity on $\{u, v\}$ such that the routing of the commodities $k \in K_{uv} \cup K_{vu}$ is affected. Thus the routing of a commodity $k \in \mathcal{K} \backslash (K_{uv} \cup K_{vu})$ is totally independent of the capacity installed on the edge $\{u, v\}$ as well as from the routing of any other commodity over $\{u, v\}$.

Suppose there exists an equation of the form

$$\alpha x + \beta f = \pi \tag{4.35}$$

satisfied by all points of $\mathcal{F}$, where α and β are vectors of an appropriate dimension and π is a real number. We show that (4.35) is a positive multiple of (4.32) plus a linear combination of the equal constraints of our polyhedral (i.e. the flow balance constraints (3.2)), partially using a similar proof as presented in [15].

From the point $\tilde{p} \in \mathcal{F}$ constructed above we can derive new points. Let $e^* \in E \backslash \{u, v\}$, we consider the set of commodities $K_{over\_e^*}$ which are routed over e^* in the routing employed in $\tilde{p}$. Since G is 3-connected there exist for each two nodes $o, d \in V$ two edge-disjoint o-d-paths in D, so that we can reroute all commodities $k \in K_{over\_e^*}$ in the way that they do not use e^* anymore. Since we use the same linkdesign installation as in $\tilde{p}$ there exists enough capacity and we get a new point of $\mathcal{F}$ with a routing where none of the commodities $k \in \mathcal{K}$ is using e^*. We can now generate new points of $\mathcal{F}$ by equipping e^* with none or an arbitrary linkdesign $l \in \mathcal{L}_{e^*}$. Since the latter points all satisfy (4.35) with equality it follows that $\alpha^l_{e^*} = 0 \quad \forall l \in \mathcal{L}_{e^*}$ and since the edge was chosen arbitrarily we can conclude that

$$\alpha^l_e = 0 \quad \forall e \in E \backslash \{u, v\}, l \in \mathcal{L}_e.$$

Now we consider the remaining coefficients corresponding to the linkdesign installation variables of the edge $\{u, v\}$. As mentioned above, the points of $\mathcal{F}$ can be categorized in three types. All points of a certain type have in common that the edge $\{u, v\}$ is equipped with a linkdesign of the corresponding linkdesign set ($\mathcal{L}^{large}_{K_{uv}K_{vu}}$, $\mathcal{L}^{uv-close}_{K_{uv}K_{vu}}$ or $\mathcal{L}^{vu-close}_{K_{uv}K_{vu}}$). Taking a point from $\mathcal{F}$ with a certain type, we can construct another point

of the same type by equipping $\{u, v\}$ with another linkdesign from the corresponding linkdesign set. Since these two points fulfill (4.35) with equality and exploiting the fact that $\mathcal{L}^{uv-close}_{K_{uv}K_{vu}} \cap \mathcal{L}^{vu-close}_{K_{uv}K_{vu}} \neq \emptyset$, it follows that

$$\begin{aligned} \alpha^{l_1}_{\{u,v\}} &= \alpha^{l_2}_{\{u,v\}} \quad \forall l_1, l_2 \in \mathcal{L}^{large}_{K_{uv}K_{vu}} \\ \alpha^{l_1}_{\{u,v\}} &= \alpha^{l_2}_{\{u,v\}} \quad \forall l_1, l_2 \in \mathcal{L}^{uv-close}_{K_{uv}K_{vu}} \cup \mathcal{L}^{vu-close}_{K_{uv}K_{vu}} \end{aligned}$$

We now deal with the coefficients corresponding to the commodity edge flow variables of an arbitrary commodity $\check{k} \in \mathcal{K}\backslash(K_{uv} \cup K_{vu})$.

We start once more from the point $\tilde{p} \in \mathcal{F}$ from the beginning and an edge $\{i, j\} \in E$. If the commodity $\check{k}$ is routed over $\{i, j\}$ we modify $\tilde{p}$ by rerouting $\check{k}$ but employing the same routing for the other commodities as well as the same link design installation. Since G is 3-connected we can find a $o_{\check{k}}$-$d_{\check{k}}$-path which is not using $\{i, j\}$, such that we obtain the new point p'. A new point p'' of $\mathcal{F}$ is generated employing exactly the same routing strategy for all commodities $k \in \mathcal{K}$ as in p', except that commodity $\check{k}$ is additionally routed over the edge $\{i, j\}$ in the directions i to j and j to i. Since both points p' and p'' satisfy the equality, $\alpha^l_e = 0 \quad \forall e \in E\backslash\{u, v\}$ and routing of commodities $k \in \mathcal{K}\backslash(K_{uv} \cup K_{vu})$ does not notably affect the capacity of $\{u, v\}$ it follows that $\beta^{\check{k}}_{ij} + \beta^{\check{k}}_{ji} = 0 \quad \forall\{i, j\} \in E$.

Next we show that for all cycles C in the overlaying digraph D of G it holds that $\sum_{(i,j)\in C} \beta^{\check{k}}_{ij} = 0$. Since any cycle in the graph can be decomposed into a collection of simple cycles (i.e. cycles that visit each node at most once) it follows that we only have to prove this claim for simple cycles. If C is a 2-cycle the claim is already shown. Let $\mathfrak{p}$ be a simple path from $o_{\check{k}}$ to $d_{\check{k}}$ in the digraph D. If the number of nodes on the path $\mathfrak{p}$ that are also on the cycle C is less than or equal to one, then we use similar arguments as before to show that $\sum_{(i,j)\in C} \beta^{\check{k}}_{ij} = 0$. We consider a point in $\mathcal{F}$ using path $\mathfrak{p}$ for the routing of commodity $\check{k}$ and simply create a new point by conducting $\check{k}$ additionally over the cycle C.
If the number of nodes on the path $\mathfrak{p}$ that are also on the cycle C is greater than or equal to 2, then define v_1 as the first, and v_2 to be the last node on the path that is also on the cycle. As a result, path $\mathfrak{p}$ can be decomposed into three parts $\mathfrak{p}_1, \mathfrak{p}_2$ and $\mathfrak{p}_3$, where $\mathfrak{p}_1$ is a path from $o_{\check{k}}$ to v_1, $\mathfrak{p}_2$ is a path from v_1 to v_2, and $\mathfrak{p}_3$ is a path from v_2 to $d_{\check{k}}$. Similarly, the cycle C can be decomposed into a path C_1 from v_1 to v_2 and a path C_2 from v_2 to v_1. Given these definitions, we can construct two new paths from $o_{\check{k}}$ to $d_{\check{k}}$ in the graph. The first path can be represented as $\mathfrak{p}_1, C_1, \mathfrak{p}_3$ and the second path as $\mathfrak{p}_1, C_2^r, \mathfrak{p}_3$, where C_2^r is the reversed path of C_2. Consider a point in $\mathcal{F}$ using the first path for the routing of commodity $\check{k}$. We can construct now another point in $\mathcal{F}$ by employing the same routing strategy for all commodities $k \in \mathcal{K}\backslash\{\check{k}\}$, but using the second path for commodity $\check{k}$. Since both points satisfy the equality it follows that $\sum_{(i,j)\in C_1} \beta^{\check{k}}_{ij} - \sum_{(i,j)\in C_2^r} \beta^{\check{k}}_{ij} = 0$. Exploiting the fact that $\beta^{\check{k}}_{ij} = -\beta^{\check{k}}_{ji} \quad \forall\{i, j\} \in E$, it follows that $\sum_{(i,j)\in C} \beta^{\check{k}}_{ij} = \sum_{(i,j)\in C_1} \beta^{\check{k}}_{ij} + \sum_{(i,j)\in C_2} \beta^{\check{k}}_{ij} = 0$,

which proves our intermediate claim. Since the commodity $\check{k}$ was chosen arbitrarily from $\mathcal{K}\backslash(K_{uv} \cup K_{vu})$, we can conclude that

$$\sum_{(i,j)\in C} \beta_{ij}^k = 0 \quad \forall k \in \mathcal{K}\backslash(K_{uv} \cup K_{vu}), \{i,j\} \in E, C \text{ cycle}$$

Next, for a commodity $\check{k} \in \mathcal{K}\backslash(K_{uv} \cup K_{vu})$, for all $q \in V$ and a path $\mathfrak{p}$ from $o_{\check{k}}$ to q in the graph, let $\mu_q^{\check{k}} = \sum_{(i,j)\in\mathfrak{p}} \beta_{ij}^{\check{k}}$. We claim that the value of $\mu_q^{\check{k}}$ is independent of the selected path $\mathfrak{p}$. To verify this claim, let $\mathfrak{p}_1, \mathfrak{p}_2$ be two paths from $o_{\check{k}}$ to q in the graph, and let $\mathfrak{p}_1^r, \mathfrak{p}_2^r$ be the reversed paths. Then $\mathfrak{p}_1 \cup \mathfrak{p}_2^r$ forms a cycle, hence $\sum_{(i,j)\in\mathfrak{p}_1\cup\mathfrak{p}_2^r} \beta_{ij}^{\check{k}} = 0$. Using $\beta_{ij}^{\check{k}} = -\beta_{ji}^{\check{k}}$ it then follows that $\sum_{(i,j)\in\mathfrak{p}_1} \beta_{ij}^{\check{k}} = \sum_{(i,j)\in\mathfrak{p}_2} \beta_{ij}^{\check{k}}$, thus indeed, the value of $\mu_q^{\check{k}}$ is independent of the selected path from $o_{\check{k}}$ to q.
If we multiply the flow balance equalities corresponding to the commodities of $\mathcal{K}\backslash(K_{uv} \cup K_{vu})$ by these multipliers and add them all up, we obtain the following expression:

$$\sum_{k\in\mathcal{K}\backslash(K_{uv}\cup K_{vu})} \sum_{i\in V} \mu_i^k \Big(\sum_{\{i,j\}\in\delta(i)} (f_{ji}^k - f_{ij}^k) \Big) =$$
$$\sum_{k\in\mathcal{K}\backslash(K_{uv}\cup K_{vu})} \sum_{\{i,j\}\in E} ((\mu_i^k - \mu_j^k) f_{ji}^k + (\mu_j^k - \mu_i^k) f_{ij}^k) =$$
$$\sum_{k\in\mathcal{K}\backslash(K_{uv}\cup K_{vu})} \sum_{\{i,j\}\in E} (\beta_{ij}^k f_{ij}^k + \beta_{ji}^k f_{ji}^k)$$

This implies that the coefficients β_{ij}^k corresponding to commodities $k \in \mathcal{K}\backslash(K_{uv} \cup K_{vu})$ are derived from a linear combination of the model equalities (3.2).

Now we address ourselves to the remaining coefficients of the commodity edge flow variables of the commodities $k \in K_{uv} \cup K_{vu}$. Starting from the point $\tilde{p} \in \mathcal{F}$ from the beginning and an edge $\{v_1, v_2\} \in E\backslash\{u,v\}$. Concerning a commodity $\check{k} \in K_{uv}$ and exploiting the fact that the graph is 3-connected, we can find an (u,v)-inclusive $o_{\check{k}}$-$d_{\check{k}}$-path $\mathfrak{p}$ which does not use the edge $\{i,j\}$. We modify $\tilde{p}$ by rerouting $\check{k}$ using the prementioned path $\mathfrak{p}$. We employ the same routing for the other commodities as well as the same link design installation. A new point of $\mathcal{F}$ is generated employing exactly the same routing strategy for all commodities $k \in \mathcal{K}$, except that commodity $\check{k}$ is additionally routed over the edge $\{i,j\}$ in the directions i to j and j to i. Since both points satisfy the equality and $\alpha_e^l = 0 \quad \forall e \in E\backslash\{u,v\}$ it follows that $\beta_{ij}^{\check{k}} + \beta_{ji}^{\check{k}} = 0 \quad \forall \{i,j\} \in E\backslash\{u,v\}, l \in \mathcal{L}_{\{u,v\}}$. Since $\check{k}$ was chosen arbitrarily, and applying an analogue argumentation for the commodities $k \in K_{vu}$, we can conclude :

$$\beta_{ij}^k + \beta_{ji}^k = 0 \quad \forall k \in K_{uv} \cup K_{vu}, \{i,j\} \in E\backslash\{u,v\}$$

Using a similar argumentation as before and exploiting the fact that G is 3-connected, we can show that

$$\sum_{(i,j)\in C} \beta_{ij}^k = 0 \quad \forall k \in K_{uv} \cup K_{vu}, \{i,j\} \in E, C \in \mathfrak{C}^{\{u,v\}-excl}.$$

Now we consider a spanning tree $T = (V, E')$ of G using edges in $E \backslash \{u, v\}$. We choose a node $v_r \in V$ and direct the edges of T suitably such that we obtain the directed tree $T' = (V, A')$ with the root node v_r. For an arc $(v_r, v_c) \in \delta^+(v_r) \cap A'$ and a commodity $k \in K_{uv} \cup K_{vu}$ we consider the respective variable $f^k_{v_r v_c}$. Since this variable appears in the flow balance equality of v_c and k, we can subtract a multiple of this equality from (4.35) and assume for the respective coefficient $\beta^k_{v_r v_c} = 0$. We can do this analogously traversing the tree from the root v_r to its leaves, subtracting successively for every arc $a \in A'$ multiples of the flow balance equalities respective to its target and the commodity k.
For $k \in K_{uv} \cup K_{vu}$ appears each variable f^k_{ij} with $(i, j) \in A'$ in two flow balance equalities, one of the node i and the other of j. On our way from v_r to the leaves of T' we subtract at first a multiple of the flow balance equality of i. In the next step we subtract the flow balance equality of j multiplied with (if necessary a negation of) the current coefficient β^k_{ij} of the variable f^k_{ij}. Since we never pass the nodes i or j again on the path to the leaves, the variable f^k_{ij} will not be considered anymore. So we derive $\beta^k_{ij} = 0 \quad \forall (i, j) \in A'$. Since $\beta^k_{ij} = -\beta^k_{ji}$ for any $\{i, j\} \in E \backslash \{u, v\}$, this implies that $\beta^k_{ij} = 0 \quad \forall (i, j) \in E'$ and $k \in K_{uv} \cup K_{vu}$.
For $\{v_1, v_2\} \in (E \backslash \{u, v\}) \backslash E'$ we can find the unique cycle C formed by $\{v_1, v_2\}$ and the edges in E'. Since C does not contain $\{u, v\}$ it follows that $\sum_{(i,j) \in C} \beta^k_{ij} = 0$ and therefore $\beta^k_{v_1, v_2} = 0$. We can finally conclude that

$$\beta^k_{ij} = \beta^k_{ji} = 0 \quad \forall k \in K_{uv} \cup K_{vu}, \{i, j\} \in E \backslash \{u, v\}.$$

Once more making use of the point $\tilde{p} \in \mathcal{F}$ presented in the beginning of the proof we consider the commodity $k^* \in K_{uv}$. Exploiting the fact that the graph is 3-connected, we can find a (u, v)-inclusive o_{k^*}-d_{k^*}-path $\mathfrak{p}$ and a (u, v)-inclusive cycle C, such that both do not use the same edges in the same direction ($\mathfrak{p} \cap C = \emptyset$). We can now generate two new points in $\mathcal{F}$ from $\tilde{p}$ by employing the same linkdesign installation and commodity routing, except that k^* is routed in both cases over the path $\mathfrak{p}$ and in one of the new points additionally on the cycle C. Since these two new points fulfill (4.35) with equality and $\beta^{k^*}_{ij} = 0 \quad \forall (i, j) \notin \{(u, v), (v, u)\}$ it follows that $\beta^{k^*}_{vu} = 0$. Since k^* was chosen arbitrarily, and applying the argumentation analogously to the commodities in K_{vu} we conclude that

$$\begin{aligned} \beta^k_{vu} &= 0 \quad \forall k \in K_{uv} \\ \beta^k_{uv} &= 0 \quad \forall k \in K_{vu}. \end{aligned}$$

We now address ourselves to the other types of points in $\mathcal{F}$ presented in the beginning of the proof. We construct a point $\tilde{p}_{uv}$ of the second group by installing on the edge $\{u, v\}$ a $uv-$close linkdesign $l \in \mathcal{L}^{uv-close}_{K_{uv} K_{vu}} \cap \mathcal{L}^{vu-close}_{K_{uv} K_{vu}}$ and equipping the other

edges with a maximal linkdesign. One commodity $k^* \in K_{uv}$ will be routed over an (u,v)-exclusive o_{k^*}-d_{k^*}-path, while the commodities $k \in K_{uv} \backslash \{k^*\}$ are routed over (u,v)-inclusive o_k-d_k-paths and all commodities $k \in K_{vu}$ over (v,u)-inclusive o_k-d_k-paths. The other commodities $k \in \mathcal{K} \backslash (K_{uv} \cup K_{vu})$ can be routed arbitrarily since there exists enough capacity in respect of every edge and direction.
We can now construct a new point in $\mathcal{F}$ by employing the same linkdesign installation and routing of commodities, except that we replace k^* by another commodity $k' \in K_{uv}$. Commodity k' is now routed over an (u,v)-inclusive o_{k^*}-d_{k^*}-path and k' over an (u,v)-exclusive $o_{k'}$-$d_{k'}$-path. Since these two points fulfill (4.35) with equality we can state $\beta_{uv}^{k^*} = \beta_{uv}^{k'}$.
A point in the third group can be generated from $\tilde{p}_{uv}$ by routing k^* over an (u,v)-inclusive o_{k^*}-d_{k^*}-path and choosing one commodity $k'' \in K_{vu}$ not to be routed over $\{u,v\}$ in direction v to u. Since we employ in both points the same linkdesign installation and both fulfill (4.35) with equality it follows that $\beta_{uv}^{k^*} = \beta_{uv}^{k''}$.
We can now conclude that

$$\beta_{uv}^{k_1} = \beta_{vu}^{k_2} \quad \forall k_1 \in K_{uv}, k_2 \in K_{vu}.$$

Since $\tilde{p}_{uv}$ and $\tilde{p}$ are both points in $\mathcal{F}$ they both fulfill (4.35) with equality. The difference between them is the fact that k^* is or is not routed over $\{u,v\}$ in direction u to v and $\{u,v\}$ is equipped with different linkdesigns $l' \in \mathcal{L}_{K_{uv}K_{vu}}^{uv-close}$ or $l'' \in \mathcal{L}_{K_{uv}K_{vu}}^{large}$. Since all other $\beta_{ij}^{k^*}$ are 0 we obtain that $\alpha_{\{u,v\}}^{l'} = \beta_{uv}^{k^*} + \alpha_{\{u,v\}}^{l''}$. Considering other commodities in K_{uv} analogously we can conclude the following:

$$\alpha_{\{u,v\}}^{l_1} = \beta_{uv}^{k} + \alpha_{\{u,v\}}^{l_2} \quad \forall k \in K_{uv}, l_1 \in \mathcal{L}_{K_{uv}K_{vu}}^{uv-close}, l_2 \in \mathcal{L}_{K_{uv}K_{vu}}^{large}$$

Defining the following auxiliary variables

$$\begin{aligned} \beta &:= \beta_{uv}^{k} && \forall k \in K_{uv} \\ \alpha &:= \alpha_{\{u,v\}}^{l} && \forall l \in \mathcal{L}_{K_{uv}K_{vu}}^{uv-close} \\ \alpha^{max} &:= \alpha_{\{u,v\}}^{l} && \forall l \in \mathcal{L}_{K_{uv}K_{vu}}^{large}, \end{aligned}$$

we can state the intermediate result as follows:

$$\beta\Bigg(\sum_{k \in K_{uv}} f_{uv}^{k} + \sum_{k \in K_{vu}} f_{vu}^{k}\Bigg) + \alpha\Bigg(\sum_{l \in \mathcal{L}_{K_{uv}K_{vu}}^{-}} x_{\{u,v\}}^{l}\Bigg) + \alpha^{max}\Bigg(\sum_{l \in \mathcal{L}_{K_{uv}K_{vu}}^{+}} x_{\{u,v\}}^{l}\Bigg) \leq \pi, \tag{4.36}$$

$$\beta = \alpha - \alpha^{max}. \tag{4.37}$$

We multiply the initial inequality (4.32) with β and add the linkdesign installation inequality (3.1) multiplied with α^{max}. We obtain:

$$
\begin{array}{lcll}
\beta\big(\sum_{k\in K_{uv}} f^k_{uv} + \sum_{k\in K_{vu}} f^k_{vu}\big) & & & \\
+(\alpha-\alpha^{max})\sum_{l\in\mathcal{L}^-_{K_{uv}K_{vu}}} x^l_{\{u,v\}} & \leq & \beta\big(|K_{uv}|+|K_{vu}|\big) & (I) \\
\alpha^{max}\sum_{l\in\mathcal{L}^+_{\{u,v\}}} x^l_{\{u,v\}} & & & \\
+\alpha^{max}\sum_{l\in\mathcal{L}^-_{\{u,v\}}} x^l_{\{u,v\}} & \leq & \alpha^{max} & (II) \\
\\
\beta\big(\sum_{k\in K_{uv}} f^k_{uv} + \sum_{k\in K_{vu}} f^k_{vu}\big) & & & \\
+\alpha^{max}\sum_{l\in\mathcal{L}^+_{K_{uv}K_{vu}}} x^l_{\{u,v\}} & & & \\
+\alpha\sum_{l\in\mathcal{L}^-_{K_{uv}K_{vu}}} x^l_{\{u,v\}} & \leq & \beta\big(|K_{uv}|+|K_{vu}|\big)+\alpha^{max}. & (I+II)
\end{array}
\tag{4.38}
$$

If $\alpha^{max} > 0$, the inequality (4.38) (and thus (4.36)) would be weaker than the initial inequality (4.32). Therefore α^{max} has to be 0 and

$$\alpha^l_{\{u,v\}} = 0 \quad \forall l \in \mathcal{L}^{large}_{K_{uv}K_{vu}}.$$

The above paragraph covers all relevant coefficients such that the Claim 4.27 is proven.

Claim 4.28 $\mathcal{F} \neq \mathcal{P}_{base}$

We can generate a point in $\mathcal{P}_{base}\backslash\mathcal{F}$ easily by equipping all edges $e \in E$ with a maximal linkdesign $l \in \mathcal{L}^{large}_{K_{uv}K_{vu}}$, routing all commodities $k \in K_{uv}$ over (u,v)-exclusive o_k-d_k-paths, all commodities $k \in K_{vu}$ over (v,u)-exclusive o_k-d_k-paths and all $k \in \mathcal{K}\backslash(K_{uv} \cup K_{vu})$ arbitrarily.

This completes the proof that (4.32) is in fact facet-defining. □

Remark 4.29 The assumption of 3-connectedness seems to be a very restrictive condition since telecommunication networks are typically sparse. This is not the case in the given data. All the networks of all considered problem instances are extremely dense and the great majority even consist of complete graphs.

Remark 4.30 The condition $\mathcal{L}_{\{u,v\}} \subseteq (\mathcal{L}^{uv-close}_{K_{uv}K_{vu}} \cup \mathcal{L}^{vu-close}_{K_{uv}K_{vu}} \cup \mathcal{L}^{large}_{K_{uv}K_{vu}})$ is indeed very restrictive; we apply a lifting approach to include linkdesigns which are not in this union.

Remark 4.31 The assumption $\mathcal{L}^{uv-close}_{K_{uv}K_{vu}} \cap \mathcal{L}^{vu-close}_{K_{uv}K_{vu}} \neq \emptyset$ also results to be very restrictive, since it requires a certain composition for the commodity subsets K_{uv} and K_{vu}. Defining $\Delta := \sum_{k \in \mathcal{K} \setminus (K_{uv} \cup K_{vu})} \max\{U_k, \overline{U}_k\}$, the subsets must fulfill the following: Either

$$\Delta < U_{k_1} - \overline{U}_{k_1} \text{ and } \Delta < \overline{U}_{k_2} - U_{k_2} \quad \forall k_1 \in K_{uv},\ k_2 \in K_{vu}$$

or

$$\Delta < \overline{U}_{k_1} - U_{k_1} \text{ and } \Delta < U_{k_2} - \overline{U}_{k_2} \quad \forall k_1 \in K_{uv},\ k_2 \in K_{vu}.$$

4.3.4 Lifting

In the latter section we have shown that the inequalities of the type (4.32) are facet-defining for a relaxation $\mathcal{P}_{Base}$ of our problem, in the case that some conditions are fulfilled. Since these requirements are quite restrictive, we lift the inequality (4.32). The next proposition from [11] is given here for readers unfamiliar with the lifting procedure.

Proposition 4.32 *Suppose* $S = \{x \in \{0,1\}^n \mid Ax \leq b\}$, $S^\delta = \{x \in S \mid x_1 = \delta \text{ for } \delta \in \{0,1\}\}$, *and* $\sum_{j=2}^n \pi_j x_j \leq \pi_0$ *is valid for* S^0. *If* $S^1 = \emptyset$, *then* $x_1 \leq 0$ *is valid for* S. *If* $S^1 \neq \emptyset$, *then*

$$\alpha_1 x_1 + \sum_{j=2}^n \pi_j x_j \leq \pi_0 \tag{4.39}$$

is valid for any $\alpha_1 \leq \pi_0 - \xi$ *where*

$$\xi = \max \sum_{j=2}^n \pi_j x_j \quad x \in S^1 \tag{4.40}$$

Moreover, if $\alpha_1 = \pi_0 - \xi$ *and* $\sum_{j=2}^n \pi_j x_j \leq \pi_0$ *is facet-defining for* $conv(S^0)$, *then* (4.39) *is facet-defining for* $conv(S)$.

Proposition 4.32 represents a lifting with respect to the variable x_1. We call α_1 the lifting coefficient for x_1.

In the following we show how the inequality (4.32) concerning an edge and the two commodity subsets, has to be lifted, such that it is facet-defining even when the corresponding set of linkdesigns contains also linkdesigns which are neither large nor

uv- or vu-close. Let $\{u,v\} \in E$, $K_{uv}, K_{vu} \subseteq \mathcal{K}$, we define:

$$\mathcal{L}^{res}_{K_{uv}K_{vu}} := \left\{ l \in \mathcal{L}_{\{u,v\}} \;\middle|\; l \notin \mathcal{L}^{uv-close}_{K_{uv}K_{vu}} \cup \mathcal{L}^{vu-close}_{K_{uv}K_{vu}} \cup \mathcal{L}^{large}_{K_{uv}K_{vu}} \right\} \tag{4.41}$$

$$S^{res}_{base} := \left\{ (x,f) \in \{0,1\}^{\sum_{e\in E}|\mathcal{L}_e|} \times \{0,1\}^{2\times|E|\times|\mathcal{K}|} \;\middle|\; (x,f) \text{ satisfies } \begin{matrix}(3.1),(3.2),\\(3.20),(3.21),\end{matrix} \sum_{l\in\mathcal{L}^{res}_{K_{uv}K_{vu}}} x_{\{u,v\}^l} \leq 0 \right\}. \tag{4.42}$$

With assistance of the fixing of the variables corresponding to the linkdesigns $l \in \mathcal{L}^{res}_{K_{uv}K_{vu}}$, Proposition 4.25 implies that (4.32) is facet-defining for $conv(S^{res}_{base})$, if

1. $G = (V,E)$ is 3-connected
2. $\mathcal{L}^{max}_e \neq \emptyset \quad \forall e \in E$
3. $\mathcal{L}^{uv-close}_{K_{uv}K_{vu}} \cap \mathcal{L}^{vu-close}_{K_{uv}K_{vu}} \neq \emptyset$.

We consider lifting the variable $x^{\check{l}}_{\{u,v\}}$ corresponding to a linkdesign $\check{l} \in \mathcal{L}^{res}_{K_{uv}K_{vu}}$ and define:

$$S^{res}_{base}(\check{l}) := \left\{ (x,f) \in \{0,1\}^{\sum_{e\in E}|\mathcal{L}_e|} \times \{0,1\}^{2\times|E|\times|\mathcal{K}|} \;\middle|\; (x,f) \text{ satisfies } \begin{matrix}(3.1),(3.2),\\(3.20),(3.21),\end{matrix} \sum_{l\in\mathcal{L}^{res}_{K_{uv}K_{vu}}\setminus\{\check{l}\}} x_{\{u,v\}^l} \leq 0 \right\}. \tag{4.43}$$

The lifting coefficient is $\alpha_{\check{l}} = |K_{uv}| + |K_{vu}| - \xi_{\check{l}}$, where $\xi_{\check{l}}$ can be obtained as follows:

$$\begin{aligned} \xi_{\check{l}} &= \max_{(x,f)\in S^{res}_{base}(\check{l})} \Big\{ \sum_{k\in K_{uv}} f^k_{uv} + \sum_{k\in K_{vu}} f^k_{vu} + \sum_{l\in\mathcal{L}^-_{K_{uv}K_{vu}}} x^l_{\{u,v\}} \;\Big|\; x^{\check{l}}_{\{u,v\}} = 1 \Big\} \\ &= \max_{(x,f)\in S^{res}_{base}(\check{l})} \Big\{ \sum_{k\in K_{uv}} f^k_{uv} + \sum_{k\in K_{vu}} f^k_{vu} \;\Big|\; x^{\check{l}}_{\{u,v\}} = 1 \Big\} \end{aligned} \tag{4.44}$$

The value $\xi_{\check{l}}$ is the amount of commodities $k \in K_{uv}$ and K_{vu} which can be routed simultaneously over the edge $\{u,v\}$ in direction u to v, v to u respectively, without exceeding the directed capacities provided by the linkdesign $\check{l}$. More precisely:

$$\begin{aligned} \xi_{\check{l}} = \max \Big\{ |\check{K}_{uv}| + |\check{K}_{vu}| \;\Big|\; & \check{K}_{uv} \subseteq K_{uv},\ \check{K}_{vu} \subseteq K_{vu}, \\ & \sum_{k\in\check{K}_{uv}} U_k + \sum_{k\in\check{K}_{vu}} \overline{U}_k \leq C^{\check{l}}_{uv}, \\ & \sum_{k\in\check{K}_{uv}} \overline{U}_k + \sum_{k\in\check{K}_{vu}} U_k \leq C^{\check{l}}_{vu} \Big\}. \end{aligned} \tag{4.45}$$

Since all linkdesign installation variables $\mathcal{L}_{\{u,v\}}$ are contained in one GUB (represented by the corresponding linkdesign constraints (3.1)), we can calculate all lifting coefficients independent of the order. Hence (4.32) can be lifted by solving (4.45) for each $l \in \mathcal{L}^{res}_{K_{uv}K_{vu}}$, such that

$$\sum_{k\in K_{uv}} f^k_{uv} + \sum_{k\in K_{vu}} f^k_{vu} + \sum_{l\in\mathcal{L}^{res}_{K_{uv}K_{vu}}} \alpha_l x^l_{\{u,v\}} + \sum_{l\in\mathcal{L}^{uv-close}_{K_{uv}K_{vu}}\cup\mathcal{L}^{vu-close}_{K_{uv}K_{vu}}} x^l_{\{u,v\}} \leq |K_{uv}| + |K_{vu}| \tag{4.46}$$

is facet-defining for $\mathcal{P}_{Base}$, if

1. $G = (V, E)$ is 3-connected
2. $\mathcal{L}^{max}_e \neq \emptyset \quad \forall e \in E$
3. $\mathcal{L}^{uv-close}_{K_{uv}K_{vu}} \cap \mathcal{L}^{vu-close}_{K_{uv}K_{vu}} \neq \emptyset$.

Chapter 5

Algorithmic approach and implementational aspects

This chapter describes the algorithmic procedure used to solve the problems of the given benchmark suite. In the first part of this chapter we give an outline of the employed branch-and-cut approach. Later, we present separation approaches for inequalities, various preprocessing mechanisms and introduce a heuristic for finding feasible integral solutions.

5.1 Branch-and-Cut

5.1.1 Overview

We may find a solution by examining all possible combinations exhaustively. However, the number of combinations becomes explosively large. Thus a simple enumerative method is not applicable to our problem. In order to cope with the vast solution space, we have employed a branch-and-cut technique for effective exploration. This technique is a generalization of branch-and-bound approach with application of cutting planes.

At first we give a short survey of the generic branch-and-bound algorithm: Let N be a variable index set and $Z \subseteq N$. Let $c \in \mathbb{R}^N$ be an objective vector, and $P \subseteq \mathbb{R}^N$ be a polyhedron, e.g. a relaxation of some MIP. Let $\underline{x}, \overline{x} \in \mathbb{R}^N$. Consider the optimization problem

$$\begin{aligned} \min\{c^T x \mid \quad & x \in P, \\ & \underline{x} \leq x \leq \overline{x}, \\ & x_i \text{ integer } \forall i \in Z\}. \end{aligned} \tag{5.1}$$

The branch-and-bound algorithm solves this problem using a modified divide-and-conquer approach. The following algorithm summarizes the whole procedure:

Algorithm 5.1 (Branch-and-bound)
Input : Bounded problem (5.1): $\min\{c^T x \mid x \in P, \underline{x} \leq x \leq \overline{x}, x \in \mathbb{R}^N, x_i \in \mathbb{Z}\ \forall i \in Z\}$
Output : Optimal solution for (5.1)

(1) $\mathcal{T} \leftarrow \{(\underline{x}', \overline{x}')\}, \mathcal{A} \leftarrow \{(\underline{x}', \overline{x}')\},\ \{(\underline{x}', \overline{x}')\}, z^* \leftarrow +\infty$
(2) **while** $\mathcal{A} \neq \emptyset$ **do**
(2.1) Select a $(\underline{x}', \overline{x}' \in \mathcal{A}$
(2.2) $\mathcal{A} \leftarrow \mathcal{A} \backslash \{(\underline{x}', \overline{x}')\}$
(2.3) Solve relaxation $\min\{c^T x \mid x \in P, \underline{x}' \leq x \leq \overline{x}', x \in \mathbb{R}^N\}$.
(2.4) **if** there exists an optimal solution x' with $c^T x < z^*$ **then**
(2.4.1) **if** there exists an $i \in Z$ such that x'_i is fractional **then**
(2.4.1.1) Branch on i: Create two subproblems $(\underline{x}^{\downarrow}, \overline{x}^{\downarrow})$ and $(\underline{x}^{\uparrow}, \overline{x}^{\uparrow})$ that equal $(\underline{x}, \overline{x})$ with the exception that $\overline{x}_i^{\downarrow} = \lfloor x'_i \rfloor$ and $\underline{x}_i^{\uparrow} = \lceil x'_i \rceil$.
(2.4.1.2) $\mathcal{T} \leftarrow \mathcal{T} \cup \{(\underline{x}^{\downarrow}, \overline{x}^{\downarrow}), (\underline{x}^{\uparrow}, \overline{x}^{\uparrow})\}$
(2.4.1.3) $\mathcal{A} \leftarrow \mathcal{A} \cup \{(\underline{x}^{\downarrow}, \overline{x}^{\downarrow}), (\underline{x}^{\uparrow}, \overline{x}^{\uparrow})\}$
(2.4.2) **else**
(2.4.2.1) x' is valid solution for the original problem (5.1).
(2.4.2.2) **if** $c^T x' < z^*$ **then**
(2.4.2.2.1) $z^* \leftarrow c^T x', x^* \leftarrow x'$
(3) **if** $z^* = \infty$ **then**
(3.1) **return** 'problem (5.1) has no solution'
(4) **else**
(4.1) **return** 'optimal solution x^* with objective value z^*'

Algorithm 5.1 solves the relaxation of (5.1) obtained by dropping the integrality constraints. If the solution contains variables that are fractional, albeit they are required to be integral in the original problem, it creates two subproblems such that the current point is feasible for neither of the two, but any solution of the original problem is applicable to one of them.
Since this technique is widely known, we do not go into further details. Rather some facts which become important later in this work are discussed now:

- The problems $(\underline{x}', \overline{x}') \in \mathcal{T}$ have a tree-like structure, because each problem, except for the initial one, has a unique parent problem from which it was created. It is common to refer to elements of $\mathcal{T}$ as node in a tree, and to call the initial problem the root node.

- The optimal solution value of a node's relaxation is a lower bound for the according MIP. The offspring problems of a node contain all integral (w.r.t. Z) points of that node's MIP. Hence, the greater of the two lower bounds can be used as lower bound for the node itself. This way, bounds get propagated up to the root node, whose lower bound is valid for the original problem and all nodes.

- If the optimal solution to a node's problem relaxation is integer w.r.t. Z, (2.4.2) gets executed and no child problems are generated, even if $(\underline{x}' \neq \overline{x}')$. This is correct because no child problem can have a fractional solution of better objective, let alone integral ones. This reduces the size of $\mathcal{T}$.

- The **if**-clause (2.4) could be omitted without breaking algorithm correctness. It exploits the fact that $c^T x'$ is a lower bound for all integral (w.r.t. Z) solutions of the node's local problem. Hence if $c^T x' < z^*$, the node can only contain an optimal solution to (5.1) if the already known x^* is optimal as well. This method of removing (pruning) sub-trees is called bounding.

Branch-and-cut is a generalization of branch-and-bound where, after solving the LP-relaxation and having not been successful in pruning the node on basis of the LP solution, we try to find a violated inequality of a set of valid inequalities. The problem of finding such an inequality is called the separation problem. Sometimes, the separation problem is restricted to a certain class of inequalities, in which case we are looking for a violated inequality of this class. If we are able to find such a inequality, we can strengthen the relaxation and the LP is reoptimized. In that way we cut off fractional solutions such that local and global lower bounds rise faster and integral solutions found more quickly.

Figure 5.1 gives a survey on the used algorithm, where Z^* denotes the best known upper bound and Z_{LP} the lower bound, obtained from the LP-relaxation.

There are two steps in the branch-and-bound part that leave some choices. In step (2.1) of Algorithm 5.1, we have to select the next (sub)-problem from the list of unsolved problems to work on next, denoted as node selection. In step (2.4.1.1) we must decide how to split the problem into subproblems, referred to as variable selection or branching. In the next two sections we describe the applied branch-and-bound strategies.

5.1.2 Node selection

In the first usage of the node selection, we look for the subproblem with the worst lower bound. We select this problem with the intent of improving the global lower bound. After a certain number of branch-and-bound iterations, we change the strategy and *dive* in the branch-and-bound tree, i.e. the "up"-node of the previous iteration is chosen until no child nodes are generated for the chosen nodes. This phase is equivalent to subsequently rounding up variables until the local problem is either feasible, or a solution is found. This changeover is applied to find integer solution and hence improve the upper bound.

5.1.3 Variable selection

In our implementation, we used three different strategies of variable selection, a generic one and two newly developed ones, which utilize information about the relation of certain variables.

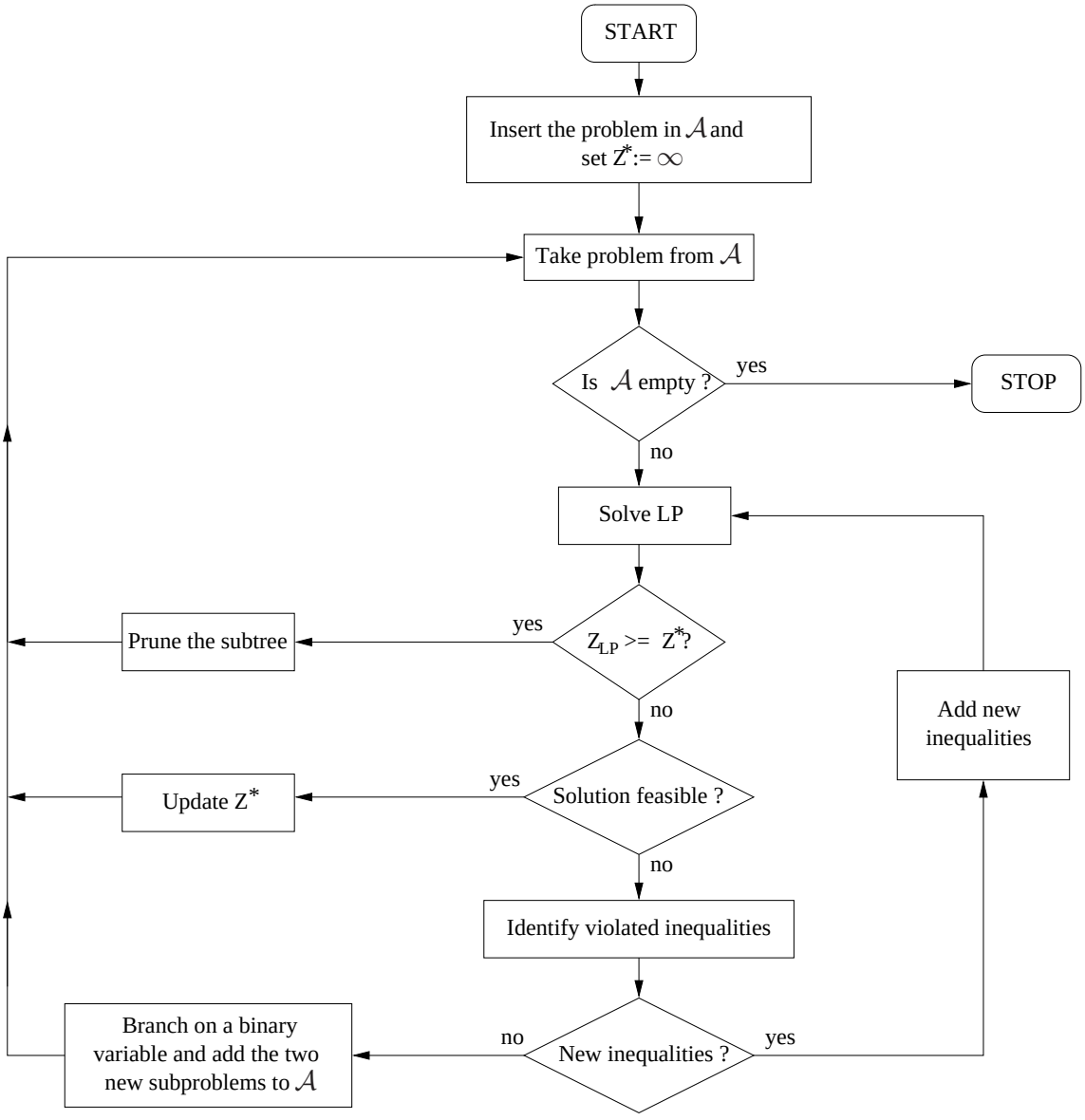


Figure 5.1: Flowchart of the applied branch-and-cut algorithm

The generic branching rule is the fixation of the most fractional variable to 1 or 0 for the following two branches. The other two branching rules are branching over the commodity edge flow variables primarily, considering the paths manifested in the current solution. If no fractional commodity edge flow variables exist, we branch over the linkdesign installation variables using the above generic branching rule. Given a fractional solution $(\hat{x}, \hat{f})$, these rules can be applied as follows:

most fractional Branch on a variable $\hat{x}_e^l$ or $\hat{f}_{uv}^k$ that is closest to 0.5.

most splitted path We compute the node-commodity-pair where the flow is most splitted. This means that we look for a node $v \in V$ and a commodity $k \in \mathcal{K}$ where the maximal number of edges are used to route the flow of k out of v:

$$\max_{v \in V,\, k \in \mathcal{K}} \left| \left\{ \{u, v\} \in \delta(v) \;\middle|\; \hat{f}_{vu}^k > 0 \right\} \right|$$

From the set of outgoing variables of v and k we pick the greatest and fix it to 0 respectively 1. If none of the paths is splitted, which means that there exist no fractional commodity edge flow variables, we branch over the linkdesign installation variables using the above most fractional branching rule.

least splitted path The only difference to the above most splitted path branching rule is the choice of the node-commodity-pair. We look for a node $v \in V$ and a commodity $k \in \mathcal{K}$ where the splitted flow of k is routed out of v using the least number of edges. We only consider commodity edge flow variables with values less than 1 because we still want to detect splitted flows.

$$\min_{v \in V,\, k \in \mathcal{K}} \left| \left\{ \{u, v\} \in \delta(v) \;\middle|\; 0 < \hat{f}_{vu}^k < 1 \right\} \right|$$

5.2 Separation

The separation problem can be stated as follows. Given a solution to the LP-relaxation of a problem, decide whether the solution satisfies a given set of inequalities, and if not, find one or more inequalities violated by the solution.
We separated the inequalities presented in Chapter 3. Note that if one of these inequalities is violated, but is not identified, this slows down the solution process (the bound could have been tightened), but the algorithm still produces an optimal solution.
Furthermore we did not include all of the model inequalities, in order to decrease the size of the formulation (and thus the solving time of the LP-relaxation). These inequalities are also separated.

5.2.1 Separation of the GUB cover inequalities

For the separation of the GUB cover inequalities presented in section 4.1 we apply the separators proposed in [16].

5.2.2 Separation of the hoplimit constraints

Since the size of the problem formulation resulted in a major difficulty for large problem instances, we tried leaving out the hoplimit constraints (3.13) and separated the respective inequalities. Two different techniques were applied:

Trivial separation We can find violated length restriction inequalities easily by simply summing up the current values of all commodity edge flow variables of a commodity $k \in \mathcal{K}$. If the sum is greater than the hoplimit M_k we just add the associated inequality (3.13) to the formulation.

Separation using shortest paths We noticed that the support (the amount of nonzero coefficients) of added inequalities influences the solving time of the resulting LP-relaxation. So we identify for each commodity $k \in \mathcal{K}$ the path $\mathfrak{p}$ over which it is mostly routed, check the length of $\mathfrak{p}$ and (in case of violation) add the inequality $\sum_{(u,v)\in\mathfrak{p}} f^k_{uv} \leq M_k$. These inequalities are weaker than the model path length constraints (3.13) used in the above paragraph, but at the same time the support is smaller and they reliably exclude solutions with violated paths when the flow for a commodity is determined to a large extent.

5.2.3 Separation of the subtour elimination constraints

Since the amount of subtour elimination constraints (3.3) is exponential in the number of nodes, we are separating these constraints. In this subsection we present two approaches in order to handle this. The first separation procedure, according to an idea presented in [17] in the context of the traveling salesman problem, utilizes the consideration of a linear program. For the second approach we introduce the weaker subtour elimination constraint, which can be separated applying simple combinatorial techniques.

Let $(\tilde{x}, \tilde{f})$ be the optimal solution of the LP-relaxation without the subtour elimination constraints. For each $k \in \mathcal{K}$ we look for the most violated subtour elimination inequality (3.3), which is equivalent to the maximization problem

$$\zeta_k = \max_{W\subseteq V} \Big\{ \sum_{\{u,v\}\in E(W)} (\tilde{f}^k_{uv} + \tilde{f}^k_{vu}) - |W| \Big\}. \tag{5.2}$$

If $\zeta_k \leq -1$ for every commodity $k \in \mathcal{K}$, then all subtour elimination inequalities (3.3) are fulfilled and $(\tilde{x}, \tilde{f}) \in \mathcal{LP}$, the LP-relaxation of our problem defined in Section 3.3. We first convert the maximization problem (5.2) into a linear integer program, and then we observe that the LP-relaxation is integral. We represent the unknown set $W \subseteq V$ by defining variables for all $v \in V$ as follows:

$$z_v = \begin{cases} 1 & \text{if } v \in W \\ 0 & \text{otherwise.} \end{cases} \tag{5.3}$$

With these variables, (5.2) is equivalent to

$$\zeta_k = \max\Big\{ \sum_{\{u,v\}\in E} \big((\tilde{f}^k_{uv} + \tilde{f}^k_{vu}) z_u z_v\big) - \sum_{v\in V} z_v \;\mid\; z_v \in \{0,1\}^{|V|}\Big\}. \tag{5.4}$$

We can eliminate the quadratic terms by introducing additional variables :

$$w_{\{u,v\}} = \begin{cases} 1 & \text{if } z_u = z_v = 1 \\ 0 & \text{otherwise} \end{cases} \qquad \forall\{u,v\} \in E \tag{5.5}$$

Hence we can reformulate (5.4) as the following integer program:

Formulation 5.2 (Subtour-IP)

$$\max \sum_{\{u,v\}\in E} \big((\tilde{f}^k_{uv} + \tilde{f}^k_{vu}) w_{\{u,v\}}\big) - \sum_{v\in V} z_v$$

$$w_{\{u,v\}} \le z_u \qquad \forall\{u,v\} \in E \tag{5.6}$$

$$w_{\{u,v\}} \le z_v \qquad \forall\{u,v\} \in E \tag{5.7}$$

$$w_{\{u,v\}} \ge z_v + z_u - 1 \quad \forall\{u,v\} \in E \tag{5.8}$$

$$w_{\{u,v\}} \in \{0,1\} \quad \forall\{u,v\} \in E$$

$$z_v \in \{0,1\} \quad \forall v \in V$$

Proposition 5.3 *If $\tilde{f}^k_{uv}$, $\tilde{f}^k_{vu} \ge 0$ for all $\{u,v\} \in E$, then the optimal solution of the LP-relaxation of the Subtour-IP is integral.*

Proof. From (5.6) and (5.7) it follows that $w_{\{u,v\}} \le \min\{z_u, z_v\}\ \forall\{u,v\} \in E$. The Subtour-IP is a maximization problem and $\tilde{f}^k_{uv}$ and $\tilde{f}^k_{vu} \ge 0$ for all $\{u,v\} \in E$. Thus there exists an optimal solution for the Subtour-IP with $w_{\{u,v\}}$ as large as possible such that (with the assistance of (5.6) and (5.7)) $w_{\{u,v\}} = \min\{z_u, z_v\}$. Since $z_v \le 1$, for all $v \in V$, it holds that $\min\{z_u, z_v\} \ge z_v + z_u - 1$ such that the inequalities (5.8) are fulfilled and can be omitted.
In each constraint of (5.6) and (5.7) there exist exactly two nonzero coefficients: one with $+1$ and one with -1. From this it follows that the constraint matrix without the conditions (5.8) is totally unimodular [14]. Hence solving the LP-relaxation of the Subtour-IP without the constraints (5.8) will result in an integer optimal solution. □

From Proposition 5.3 it follows that it is possible to separate the subtour elimination constraints (3.3) by solving a linear program. Actually, these inequalities can be separated in polynomial time. In spite of that it is too time consuming to set up a linear program and solve it in every separation step. On the other hand it is possible

to separate weaker subtour elimination inequalities with simple combinatorial methods. The subtour elimination inequalities (3.3) exclude cycles on a node subset. The inequalities

$$\sum_{a \in C} f_a^k \leq |C| - 1 \quad \forall k \in \mathcal{K}, C \subseteq A : C \text{ is a cycle} \tag{5.9}$$

exclude cycles from the solutions as well, but they are weaker than (3.3).

For any commodity $k \in \mathcal{K}$, consider the overlaying directed graph $D_k = (V, A)$ of G (as defined in Section 2.2) with the arc weights $x_{(u,v)} = f_{uv}^k$ and $x_{(v,u)} = f_{vu}^k \quad \forall e = \{u, v\} \in E$. In order to prohibit solutions with cycles we introduce the weaker subtour elimination inequality:

$$\sum_{a \in C} x_a \leq |C| - 1 \quad \forall C \subseteq A : C \text{ is a cycle} \tag{5.10}$$

When the above inequality is fulfilled, then there exist no cycles in the routing for the commodity $k \in \mathcal{K}$.

Separation of the weaker subtour elimination inequality:

Finding the most violated inequality is equivalent to the maximation problem

$$\zeta = \max_{C \text{ cycle}} \Big\{ \sum_{a \in C} x_a - |C| \Big\}. \tag{5.11}$$

All inequalities are fullfilled if $\zeta \leq -1$. Defining a new arc weight function y with $y_a = 1 - x_a \quad \forall a \in A$, we can reformulate:

$$\max_{C \text{ cycle}} \Big\{ \sum_{a \in C} x_a - |C| \Big\} = \max_{C \text{ cycle}} \Big\{ \sum_{a \in C} (1 - y_a) - |C| \Big\} = \max_{C \text{ cycle}} \sum_{a \in C} (-y_a).$$

Therefore (5.11) is equivalent to

$$\min_{C \text{ cycle}} \sum_{a \in C} y_a = -\zeta. \tag{5.12}$$

If $\min_{C \text{ cycle}} \sum_{a \in C} y_a \geq 1$ ($\zeta \leq -1$) is fullfilled, then no violated inequality exists. This can be verified by searching for a shortest cycle in D_k with arc weights $y_a = 1 - x_a$. When we detect such a cycle C for a commodity $k \in \mathcal{K}$, we add the inequality

$$\sum_{\{u,v\} \in E(C)} (f_{uv}^k + f_{vu}^k) \leq |C| - 1 \tag{5.13}$$

to the formulation.

Remark 5.4 Since $x_a \in [0, 1] \quad \forall a \in A$, none of the y_a is negative, and we can apply a modified Floyd-Warshall algorithm in order to find a shortest cycle. This All-Pairs-Shortest-Path algorithm can be extended to find the shortest v-v-path for all nodes $v \in V$. We simply iterate in all three nested loops over all nodes.

5.3 Preprocessing

We apply a series of preprocessing steps in order to reduce our problem size. We take advantage of natural observations of the problem as well as particular characteristics of the structure of the used data. Our aim is, concerning particular problem instances, to keep out irrelevant constraints and to prevent the consideration of unnecessary variables. We state the main ideas of these reductions and specify the relevancy concerning the given data. To comprehend the assumptions the reader is referred to Chapter 6 and especially to Table 6.1 which contains the main information about the problem instances of the whole benchmark suite.

5.3.1 Constraints reduction

Particularities of the given problem instances might result in excluding some constraints.

Nodes with many ports If the port limits for a node $v \in V$ are very large we do not have to include the corresponding **pmax** constraints (3.6) and (3.7) into the formulation. These constraints can be totally ignored for a certain node $v \in V$ if the amount of its ports is very big, or more precisely if

$$\min\{P_v^-, P_v^+\} \geq |\delta(v)| \sum_{\{u,v\}\in\delta(v)} \max_{l\in\mathcal{L}_{\{u,v\}}} \{P_{uv}^l, P_{vu}^l\}$$

This case, in which the port amount of a node could never be exceeded, appears for example in the problem instance A04 or A05.

Only secure commodities In problem instances where all commodities are secure, all edge capacity constraints (3.20) and (3.21) are dominated by the corresponding stronger secure edge capacity constraints (3.22) and (3.23). This is due to the equality of the secure and insecure value of each commodity $k \in \mathcal{K}$ ($U_k = U_k^S$ and $\overline{U}_k = \overline{U}_k^S$) which yields

$$\begin{aligned}\sum_{k\in\mathcal{K}} U_k f_{uv}^k + \sum_{k\in\mathcal{K}} \overline{U}_k f_{vu}^k &= \sum_{k\in\mathcal{K}} U_k^S f_{uv}^k + \sum_{k\in\mathcal{K}} \overline{U}_k^S f_{vu}^k \\ &\leq \sum_{l\in\mathcal{L}_{\{u,v\}}} S_{\{u,v\}}^l C_{uv}^l x_{\{u,v\}}^l \leq \sum_{l\in\mathcal{L}_{\{u,v\}}} C_{uv}^l x_{\{u,v\}}^l \quad \forall\{u,v\}\in E \end{aligned} \tag{5.14}$$

respectively

$$\begin{aligned}\sum_{k\in\mathcal{K}} U_k f_{vu}^k + \sum_{k\in\mathcal{K}} \overline{U}_k f_{uv}^k &= \sum_{k\in\mathcal{K}} U_k^S f_{vu}^k + \sum_{k\in\mathcal{K}} \overline{U}_k^S f_{uv}^k \\ &\leq \sum_{l\in\mathcal{L}_{\{u,v\}}} S_{\{u,v\}}^l C_{vu}^l x_{\{u,v\}}^l \leq \sum_{l\in\mathcal{L}_{\{u,v\}}} C_{vu}^l x_{\{u,v\}}^l \quad \forall\{u,v\}\in E \end{aligned} \tag{5.15}$$

The special case that all commodities have to be routed securely appears in the instances C10 and C12.

No secure commodities If, on the other side, none of the commodities have to be routed securely we do not consider the secure constraints, as well as in the case where only secure linkdesigns exist.

High hoplimits If the hoplimit for a commodity $k \in \mathcal{K}$ is very large ($M_k \geq |V| - 1$), we do not consider the corresponding constraint (3.13).

High node capacity When the traffic limit of a node $v \in V$ is very large ($C_v \geq \sum_{k \in \mathcal{K}}(U_k + \overline{U}_k)$), we do not consider the respective node capacity constraint (3.8).

Nomult As a result of the generation of linkdesigns, we do not need any restrictive conditions concerning an edge for the **nomult** case.

5.3.2 Reduction of the commodity edge flow variables

Some commodity edge flow variables can be erased or fixed in the preprocessing phase if the data holds certain characteristics.

No cycles Since we do not want to allow cycles in our routing it is always prohibited to route a commodity into its origin or out of its destination node. Hence for each commodity $k \in \mathcal{K}$ and appropriate node $v \in V$ the variables $f^k_{uo_k}$ and $f^k_{d_k v}$ do not have to be generated.

Risky node exclusion Concerning the security constraint **sec**, various commodity edge flow variables can be fixed or erased. Since a secure commodity must not be routed over a risky node, certain variables can be set to 0 as described in (3.11) and (3.12). We can not profit from this reduction in the larger instance since all instances of the series B and C (except for C16) contain only secure nodes.

Small hoplimits We can also apply variable reduction exploiting the existence of hoplimits. Actually, the hoplimit constraint should be perfectly suited for preprocessing as a number of the commodity edge flow variables might be fixed to 0. Regarding a commodity $k \in \mathcal{K}$ with the hoplimit M_k we calculate all o_k-d_k-paths with a length lower or equal to M_k (with respect to the overlaying directed graph). Now all variables whose corresponding arc does not appear in any of these paths can be fixed to 0.
Unfortunately, the graphs of the ROCOCO benchmark suite are very dense (there are indeed only three instances (C10, C12 and C16) which are not complete), such that in general two nodes are connected by an edge. Hence, as soon as the hoplimit of a commodity k is greater 2 nearly each edge and direction (or respectively arc) lies on a potential o_k-d_k-path. In the case when $M_k = 2$, all

variables f^k_{vu} and f^k_{vu} with $u, v \notin \{o_k, d_k\}$ can be erased or fixed to 0. This very strict hop limit appears, for example, in the instances A06 and C20.

Symmetrical routing As described in Chapter 3, commodities that have to be routed symmetrically can be merged together. In general the number of commodities and thus the number of commodity edge flow variables is halved. In the problem instance C20, for example, the 404 commodities could be reduced to 190.

Low node capacity The observation of the node capacities may result in the elimination of some variables. If for a commodity $k \in \mathcal{K}$ the amount of data $\max\{U_k, \overline{U}_k\}$ is greater than the node capacity C_v of some node $v \in V$, the commodity edge flow variables corresponding to k and the edges $e \in \delta(v)$ in both directions can be eliminated. This preprocessing approach has no effect on the given data since the node capacity is at least as large as the commodity with the highest amount of data.

5.3.3 Reduction of the linkdesign installation variables

When the data fulfills certain conditions, it is possible that the set $\mathcal{L}_e$ of linkdesigns of an edge $e \in E$ and hence the number of linkdesign installation variables can be reduced significantly.

High port consumption Concerning the port constraints **pmax**, all linkdesigns with an extraordinary high port consumption can be excluded from the set of linkdesigns. A linkdesign is dispensable if its minimal port consumption is greater than the maximum of in and out ports of all nodes. Let $\{u, v\}, l \in \mathcal{L}_{\{u,v\}}$; we can exclude l if

$$\min\{P^l_{uv}, P^l_{vu}\} > \max_{v \in V}\{P^-_v, P^+_v\}.$$

This simplification has been applied successfully for example in the problem instance C11

No multiplier Restrictions of the base capacity multiplier, denoted by **nomult**, reduce the set of linkdesigns (as described in Chapter 3 concerning the data transformation and generation of the linkdesigns) so that the corresponding linkdesign installation variables are not even created.

Only secure commodities Using the secure parameters, it is also possible to reduce the number of linkdesigns. When all commodities have to be routed securely, there is no use for the application of risky linkdesigns. Concerning for example the instance C12 this yields to halve the amount of linkdesigns corresponding to an edge $e \in E$.

5.4 Heuristics

We applied a heuristic to find feasible integral solutions which, at the same time, provide upper bounds in the branch-and-bound tree in order to enable fathoming of branches.
The main idea of our heuristic approach is to determine a feasible routing which imposes certain capacity requirements for each edge. Based on these values, we solve an IP on the linkdesign installation variables and achieve a feasible linkdesign configuration. This IP can be solved with a MIP-solver in reasonable time, since it contains a relative small number of variables and constraints.

Figure 5.2 gives a survey on the used algorithm. For each commodity $k \in \mathcal{K}$ we determine a o_k-d_k-path $\mathfrak{p}_k$ in $D = (V, A)$, the overlaying digraph of G (as defined in Section 2.2), fulfilling certain restrictions. Each arc in A has a weight appointed by the current commodity and $len(\mathfrak{p}_k)$ denotes the length of the path respective to these weights, while $|\mathfrak{p}_k|$ denotes the number of arcs in $\mathfrak{p}_k$. The set of paths (one for each commodity) represents a routing, for which we calculate a feasible linkdesign installation.

5.4.1 Determination of a feasible routing

In order to compute a feasible routing we primarily have to consider the flow balance constraints (3.2), the subtour elimination constraints (3.3), the path length constraints (3.13), the node capacity constraints (3.26), and the node secure capacity constraints (3.11), (3.12). The charge concerning the fulfillment of the edge (secure) capacity constraints (3.20)–(3.23) will be passed over to the problem of finding a feasible linkdesign installation for this routing.
Based on the (fractional) values of the commodity edge variable $\hat{f}$ we generate a feasible integer routing fulfilling the above constraints described as follows:

Determine a single path For each commodity $k \in \mathcal{K}$ we determine an o_k-d_k-path $\mathfrak{p}_k$ in the overlaying digraph $D = (V, A)$. This path can be found by searching for a shortest o_k-d_k-path with respect to preassigned nonnegative arc weights. These weights may be influenced by the current fractional routing denoted by $\hat{f}$. Since $\mathfrak{p}_k$ is simple, the routing is feasible with respect to the flow balance and the subtour elimination constraints.

Attending hoplimits Calculating the individual/specific paths concerning a commodity $k \in \mathcal{K}$, we can easily verify whether the amount of the used arcs is greater than the limitation M_k. If this happens we can try to find a shorter $o_k - d_k$-path applying another weight function or finally terminate this heuristic iteration unsuccessfully. If we continue we know that the routing is feasible concerning the path length constraints.

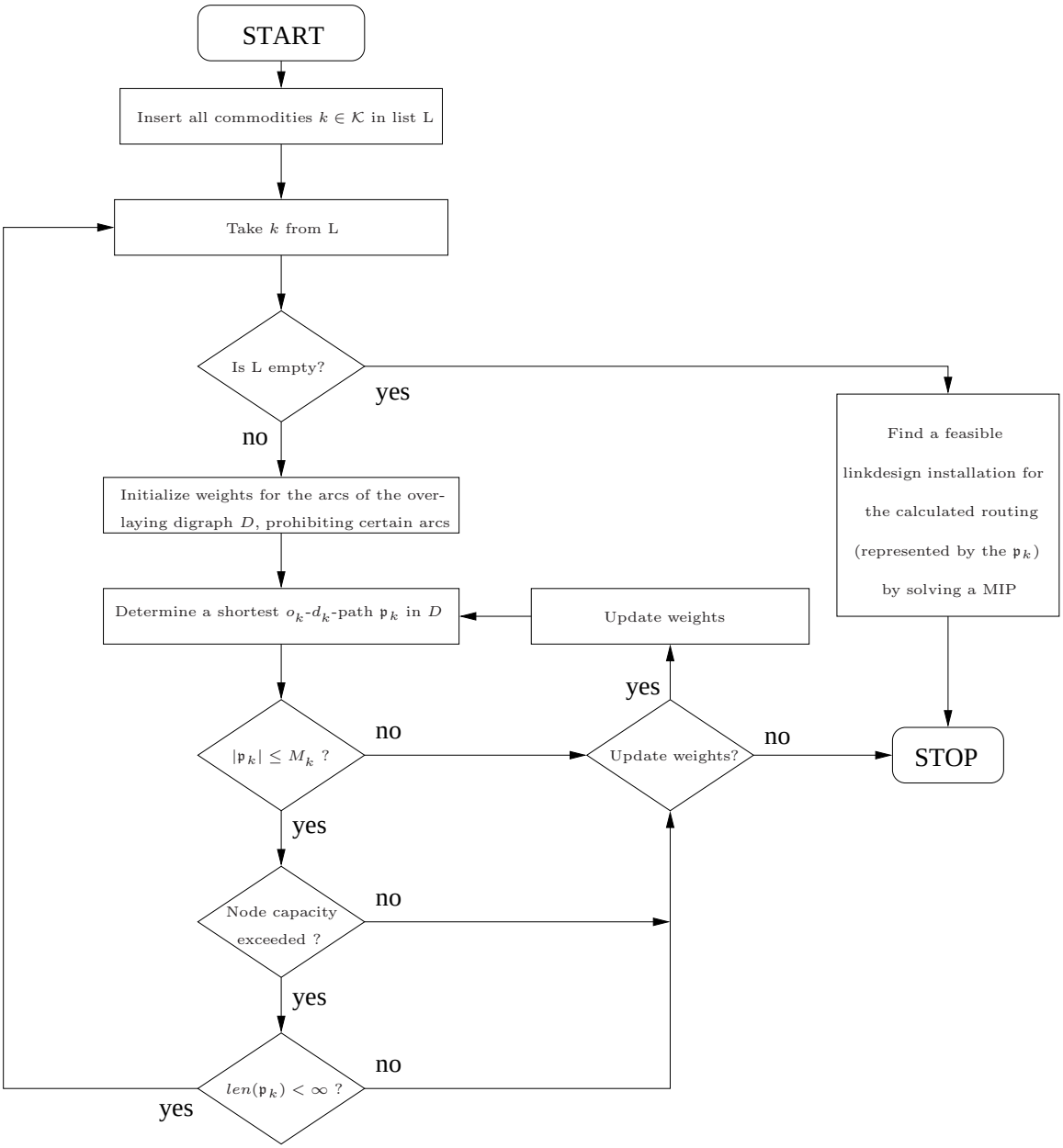


Figure 5.2: Flowchart of the applied heuristic algorithm

Attending node capacity Whenever we have calculated a feasible path for a commodity $k \in \mathcal{K}$, we add, for each node $v \in V$ on the path, the value of k to the sum of the commodities which we already dealt with and which were also routed over v. If the sum exceeds the node capacity C_v, we try to find another path for k prohibiting the arcs incident to v (by setting the corresponding arc weights to the maximal number). If this approach is not successful we can apply another weight function or finally terminate the heuristic unsuccessfully. When we continue we know that the node capacity constraints are fullfilled by the found routing.

Attending secure node capacity We now assert that the routing does not violate the node secure capacity constraints. For a secure commodity $k \in \mathcal{K}$, we prohibit nodes that are not secure (and neither origin nor destination of k) by setting the weight of the incident arcs to the maximal number). So when we determine a shortest $o_k - d_k$-path with a length less than the maximal number, we know that k is not routed over risky nodes, such that the routing is feasible concerning the node secure capacity constraints.

After calculation of the feasible routing, we have for each commodity $k \in \mathcal{K}$ a set of arcs $A_k \subseteq A$ representing a cycle-free o_k-d_k-path. This corresponds to the fixed variable assignment $\tilde{f}$ with

$$\tilde{f}^k_{uv} = \begin{cases} 1 & \text{if } (u,v) \in A_k \\ 0 & \text{otherwise} \end{cases} \quad \forall (u,v) \in A.$$

5.4.2 Determination of a feasible linkdesign installation

The mechanism described above provides us with a feasible routing represented by the commodity edge variable assignment $\tilde{f}$. In order to get a feasible linkdesign installation for this routing, we have to consider the (secure) capacity constraints (3.20)–(3.23), the linkdesign installation constraints (3.1) as well as the port constraints (3.6) and (3.7). The optimal solution $\tilde{x}$ of the following IP provides us with an optimal linkdesign installation concerning the given routing.

Formulation 5.5 (Linkdesign installation IP)

$$\min \sum_{e \in E} \sum_{l \in \mathcal{L}_e} W^l_e x^l_e$$

$$\sum_{k\in\mathcal{K}} U_k \tilde{f}^k_{uv} + \sum_{k\in\mathcal{K}} \overline{U}_k \tilde{f}^k_{vu} \leq \sum_{l\in\mathcal{L}_{\{u,v\}}} C^l_{uv} x^l_{\{u,v\}} \qquad \forall\{u,v\}\in E \tag{5.16}$$

$$\sum_{k\in\mathcal{K}} U_k \tilde{f}^k_{uv} + \sum_{k\in\mathcal{K}} \overline{U}_k \tilde{f}^k_{vu} \leq \sum_{l\in\mathcal{L}_{\{u,v\}}} C^l_{vu} x^l_{\{u,v\}} \qquad \forall\{u,v\}\in E \tag{5.17}$$

$$\sum_{k\in\mathcal{K}} U^S_k \tilde{f}^k_{uv} + \sum_{k\in\mathcal{K}} \overline{U}^S_k \tilde{f}^k_{vu} \leq \sum_{l\in\mathcal{L}_{\{u,v\}}} S^l_{\{u,v\}} C^l_{uv} x^l_{\{u,v\}} \qquad \forall\{u,v\}\in E \tag{5.18}$$

$$\sum_{k\in\mathcal{K}} U^S_k \tilde{f}^k_{uv} + \sum_{k\in\mathcal{K}} \overline{U}^S_k \tilde{f}^k_{vu} \leq \sum_{l\in\mathcal{L}_{\{u,v\}}} S^l_{\{u,v\}} C^l_{vu} x^l_{\{u,v\}} \qquad \forall\{u,v\}\in E \tag{5.19}$$

$$\sum_{\{u,v\}\in\delta(v)} \sum_{l\in\mathcal{L}_{\{u,v\}}} P^l_{uv} x^l_{\{u,v\}} \leq P^-_v \quad \forall v\in V \tag{5.20}$$

$$\sum_{\{u,v\}\in\delta(v)} \sum_{l\in\mathcal{L}_{\{u,v\}}} P^l_{vu} x^l_{\{u,v\}} \leq P^+_v \quad \forall v\in V \tag{5.21}$$

$$\sum_{l\in\mathcal{L}_e} x^l_e \leq 1 \quad \forall e\in E \tag{5.22}$$

$$x^l_e \in \{0,1\} \quad \forall e\in E, l\in\mathcal{L}_e$$

Combining both partial solutions $\tilde{f}$ and $\tilde{x}$, the heuristic yields a feasible integral solution $\tilde{p} = (\tilde{x}, \tilde{f})$ to our problem.

5.4.3 Additional edge exclusion

Since our approach (due to the utilization of shortest path calculation) results in a dense network, we run into some major problems when we are confronted with strict port limits. The number of ports of a node imposes an upper bound of the number of edges which can be equipped by some linkdesigns and therefore can be used by some routing. Small port limits enforce sparse networks while the resulting routings of our heuristic approach require dense networks. We coped with this problem by adapting the weights of arcs and commodity after every iteration. (Arcs of) edges which are already used by some commodities get very cheap as long as there is still enough capacity left. *Virgin* edges (edges were nothing is routed until now) are quite expensive. Every time we use an edge the first time we remove a port of its incident nodes. If the ports of a node are exceeded we forbid the incident arcs (by setting their weight to the maximal limit).

5.4.4 Heuristic parameters

We have two options to change the attitudes of our heuristic while determinating a feasible routing. Different commodity sequences are imaginable as well as various arc weight functions.

Order of commodities

The importance of the order of the commodities follows from the idea that the first considered commodities can already establish the outline of the required linkdesign installation. It seems reasonable that commodities with high needs (in terms of large (secure) demand value or strict hoplimits) should first have the possibility to fulfill their requirements.

demand value We order the commodities in a way such that the routing of the commodities with a high demand value ($U_k + \overline{U}_k$) is fixed first.

secure data amount The commodities are ordered such that at first the routing of commodities with a big secure demand value ($U_k^S + \overline{U}_k^S$) is determined.

hoplimit The order of the commodities is associated with the hoplimits, so that first the commodities with a very small hoplimit are concerned.

Weights of arcs

To each commodity $k \in \mathcal{K}$ and each arc $(u, v) \in A$ we can apply various arc weights $w^k : A \to \mathbb{R}_+$. We can distinguish between two types of arc weight functions; *static* functions which always provide the same weights and *flexible* functions influenced by the current fractional solution. Since the static weight functions always provide the same routing, we only apply them once in the root node of the branch-and-bound tree. The heuristic implementation using the flexible weight functions is employed several times. In any case we disable the arcs corresponding to the commodity edge flow variables eliminated or fixed to zero before, by setting their weights to the maximal number.

trivial distance We search a shortest path with respect to the number of arcs by setting

$$w^k_{(u,v)} = 1 \quad \forall (u, v) \in A.$$

trivial costs The average installation costs of linkdesigns on an edge is integrated in the weight generation of the respective arcs:

$$w^k_{(u,v)} = \Big(\sum_{l \in \mathcal{L}_{\{u,v\}}} W^l_{\{u,v\}} \Big) \,/\, |\mathcal{L}_{\{u,v\}}| \quad \forall (u, v) \in A.$$

current flow We use the current solution values of the commodity edge flow variables by setting

$$w^k_{(u,v)} = (1 - \hat{f}^k_{uv}) \quad \forall (u,v) \in A.$$

current pricy flow The current flow and the average costs of an edge are integrated in the weight value of the respective arc:

$$w^k_{(u,v)} = \big((\sum_{l \in \mathcal{L}_{\{u,v\}}} W^l_{\{u,v\}}) \,/\, |\mathcal{L}_{\{u,v\}}| \big) \cdot (1 - \hat{f}^k_{uv}) \quad \forall (u,v) \in A$$

Remark 5.6 Since the weights of the arcs will always be nonnegative, the shortest paths in (5.4.1) can be determined efficiently using the DIJKSTRA algorithm.
We could have applied a hop-limited DIJKSTRA, so that we would not have had to attend the corresponding constraints afterwards.

Chapter 6

Data and computational results

In this chapter we describe the problem instances contained in the ROCOCO benchmark suite and expose some structural characteristics of certain instances. Afterwards, we present the impact of preprocessing and data transformation and the results applying certain algorithm options obtained by extensive tests using the given data. The chapter concludes with a presentation of our computational results for every problem instance and combination and a comparison with the the corresponding best published results.
Concerning the smaller problem instances, our implementation provided exactly the same optimal solution values as the ones published, which indicates that our approach models the given problem description properly. For some instances and combinations we were even able to obtain better upper bounds than the ones found until now. But in general the gap between the obtained upper and lower bounds resulted to be quite large due to the restrictive time limit.

6.1 Data set

The applied benchmark suite consists of three series, each including 7 instances whereupon 6 parameters are included in any combination. This yields 1344 problem instances. Series A contains the smallest instances, from 4 to 10 nodes, while the series B and C include larger graphs up to 25 nodes. What seems remarkable is the fact that the graphs of all problem instances are extremely dense; nearly all (except C10, C12 and C16) are even complete. There exist instances with multiple commodities having the same origin and destination (in Section 3.2 referred to as parallel), e.g. in the instance C16. There are as well instances with commodities which are isolated, as defined in Section 3.2. In every instance there appears at least one commodity which has to be routed securely and there also always exist both, secure and risky linkdesigns. Table 6.1 shows also the occurrence of unidirected linkdesigns in instance C11 (i.e. linkdesigns providing a capacity greater 0 for one direction and no capacity for the other). It can also be noticed that the number of ports of the nodes in the instances of

the series B and C are quite small, and that therefore the port constraint proves to be quite restrictive. In C10, for instance, all nodes have two inports and two outports so that at each node, at most two incident edges can be equipped with a linkdesign. This imposes that the resulting capacity network is either a chain or a single loop. Except for C20, in every instance all commodities have the same hoplimit.

6.2 Computational results

In this section we present a listing of our results on the ROCOCO benchmark suite. We present the influence of the preprocessing and data generation endeavors in order to minimize the size of the problem. Additionally, we used the data sets for further analysis of certain settings for algorithm and heuristic. Our implementation was run on all 27 instances of the benchmark suite, and each combination of the additional constraints was considered. For upper bounds, we made a comparison with the best solutions published until now.

The possible 64 combinations are identified by a six-bit vector, whereby the first digit indicates if the security constraint **sec** is considered. The second bit corresponds to the consideration of the **nomult** constraint, while the third states if the symmetrical routing constraint **symdem** was taken into account. The forth and fifth bit indicate the consideration of the hoplimit constraints **bmax** and the port constraints **pmax** respectively. The last digit corresponds to the consideration of the node capacity constraints **tmax**. For example, "011000" indicates that only the second constraint **nomult** and the third constraint **symdem** were taken into account. The order is presented in Table 6.2.

We implemented our algorithm on top of the network dimensioning tool DISCNET [4]. The data structures and algorithms are implemented in C++, using the C++ library and partially LEDA [3]. We used CPLEX 8.0 [10] to solve the LPs in the branch-and-bound tree and the MIPs in the heuristic. All results were obtained within a 10-minute time limit on an Intel(R) Xeon(TM) CPU 3.06GHz running the Linux operating system.

6.2.1 Downsizing results

In this subsection we present the results towards downsizing the problems. We consider the consequences of our endeavors concerning the reduction of variables as well as the constraint reduction. Considering the combination where all additional constraints are taken into account (i.e. the combination 111111), we start with the initial problem size corresponding to a *naive* formulation of the problem. Then we document the impact of our more sophisticated model and the consequential data transformation on the number of variables/constraints and finally the influence of preprocessing.

No.	Network					Commodities							Linkdesigns					LP numbers	
	$\|V\|$	$\|V^S\|$	C_V	P_V	$\|E\|$	$\|\mathcal{K}\|$	$\|\mathcal{K}^S\|$	$\|\mathcal{K}_p\|$	$\|\mathcal{K}_c\|$	$\|\mathcal{K}_i\|$	$U_{\mathcal{K}}$	$M_{\mathcal{K}}$	$\|\mathcal{L}\|$	$\|\mathcal{L}^S\|$	$\|\mathcal{L}_u\|$	$C_{\mathcal{L}}$	$P_{\mathcal{L}}$	$\|Vars\|$	$\|Constr\|$
A04	4	2	256	2-256	6	12	2	0	6	0	4-65	2	5	4	0	64-256	1 - 3	180	66-180
A05	5	3	256	2-256	10	20	2	0	10	0	3-65	2	5	4	0	64-256	1 - 3	460	130-395
A06	6	3	512	2-256	15	26	2	0	13	0	4-65	2	6	5	0	64-512	1 - 3	885	201-680
A07	7	4	512	2-256	21	36	2	0	18	0	4-65	3	6	5	0	64-512	1 - 3	1659	315-1191
A08	8	5	512	2-256	28	40	4	0	20	0	2-66	3	6	5	0	64-512	1 - 3	2436	404-1672
A09	9	6	512	2-256	36	50	4	0	25	0	2-66	3	6	5	0	64-512	1 - 3	3852	558-2543
A10	10	7	512	2-256	45	62	6	0	31	0	2-66	3	6	5	0	64-512	1 - 3	5895	755-3772
B10	10	10	4096	2-4	45	87	34	0	45	3	14-198	4	25	20	0	128-9600	1 - 5	9000	1005-5037
B11	11	11	4096	4	55	108	39	0	55	2	10-197	4	25	20	0	128-9600	1 - 5	13310	1353-7489
B12	12	12	4096	6	66	108	59	0	63	18	64-249	4	25	20	0	128-9600	1 - 5	15972	1494-7776
B15	15	15	4096	6	105	184	98	0	104	24	20-128	4	25	20	0	128-9600	1 - 5	41370	3075-20419
B16	16	16	4096	6	120	211	87	0	116	21	10-100	4	25	20	0	128-9600	1 - 5	53760	3736-27155
B20	20	20	8192	6	190	366	217	0	190	14	5-128	4	25	20	0	128-9600	1 - 5	144020	7890-75766
B25	25	25	8192	6	300	462	303	0	284	106	16-64	4	25	20	0	128-9600	1 - 5	285000	12450-120687
C10	10	10	85000	2	41	66	66	0	33	0	148-11660	5	6	3	0	2048-102000	1 - 3	5699	783-3708
C11	11	11	4096	4	55	108	39	0	55	2	10-197	4	20	15	10	1024-10240	1 - 5	13035	1353-7489
C12	12	12	272000	6	62	132	132	0	66	0	1732-19427	4	10	5	0	2048-170000	1 - 5	17050	1770-10308
C15	15	15	4096	6	105	184	98	0	104	24	20-128	4	10	5	0	1024-10240	1 - 5	39795	3075-19921
C16	16	10	136000	6	115	186	66	0	93	0	148-11660	5	6	3	0	2048-102000	1 - 3	43585	3321-25290
C20	20	20	8192	6	190	404	38	76	190	14	3-128	2-6	10	5	0	1024-10240	1 - 5	155610	8650-105444
C25	25	25	8192	6	300	462	303	0	284	106	16-64	4	10	5	0	1024-10240	1 - 5	280500	12450-120687

Table 6.1: The ROCOCO data

Network :

$\|V\|$	the number of nodes
$\|V^S\|$	the number of secure nodes
C_V	traffic/capacity range of the nodes
P_V	port range of the nodes
$\|E\|$	the number of edges

Commodities :

$\|\mathcal{K}\|$	the number of commodities
$\|\mathcal{K}^S\|$	the number of secure commodities
$\|\mathcal{K}_p\|$	the number parallel commodities
$\|\mathcal{K}_c\|$	the number of connected commodity-sets
$\|\mathcal{K}_i\|$	the number of isolated commodities
$U_{\mathcal{K}}$	the data range of the commodities
$M_{\mathcal{K}}$	the hoplimit range of the commodities

Linkdesigns :

$\|\mathcal{L}\|$	the number of linkdesigns
$\|\mathcal{L}^S\|$	the number of secure linkdesigns
$\|\mathcal{L}_u\|$	the number of unidirected linkdesigns
$C_{\mathcal{L}}$	the capacity range of the linkdesigns
$P_{\mathcal{L}}$	the port consumption range of the linkdesigns

Initial LP: (corresponding to a naive formulation)

$\|Vars\|$	the number of variables in the initial LP
$\|Constr\|$	the number of constraints in the initial LP
	for all combinations

sec	**nomult**	**symdem**	**bmax**	**pmax**	**tmax**
$\{0,1\}$	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$

Table 6.2: Combination of additional constraints

Downsizing was a really important ambition for us. At first the LP-formulation of the problem was so large, that for the big problem instances (B15−B25 and C15−C25) no solutions could be obtained since the solving time of the LP-relaxation overstepped the time limit. Not until we applied preprocessing and elaborate data transformation did we achieve a breakthrough.

Reduction of variables

We present the average reduction of variables per series, which arise due to the data transformation and certain preprocessing procedures as introduced in Section 5.3.2 and Section 5.3.3. The first two reduction steps correspond to the transformation of the problem from a naive formulation to our more sophisticated one. In Table 6.1 it can be seen that the merging process for the *symmetrical routing* presented in the end of Section 3.2.3 resulted for all instances in the greatest reduction. Also the restriction of the linkdesign sets as a result of the **nomult** constraint had major reduction effects. The greatest impact of the corresponding linkdesign elimination was achieved on the instances of series B where for all edges the set of linkdesigns were reduced from 25 to 5 linkdesigns. The average effect of the latter preprocessing mechanisms on the instances of the series B and C was (except for the *nocycle* procedure) quite poor. In the instances of these two series there exist only commodities with a hoplimit greater than 2 (except for C20) and only secure nodes (except for C16), such that the corresponding preprocessing procedures (*small hoplimits* respectively *risky node exclusion*) hardly influence the number of variables. The *only secure commodities* mechanism effects only the number of variables for the instances C10 and C12 since they are the only instances in which all commodities have to be routed securely.
For the instances of series A we were most successful. After the transformation and preprocessing, we were only confronted with 34% of the initial variables. But also the influence on the number of variables for the instances of series B and C was worthwhile since we were only confronted with 46%, or 43% respectively of the initial variables.
Note that the order of the applied variable reducing strategies influences the respective impact. The mechanisms may effect/delete the same variables.

Reduction of constraints

As in the latter subsection the first two reduction steps correspond to the transformation of the problem from a naive formulation to our more sophisticated one. In Table

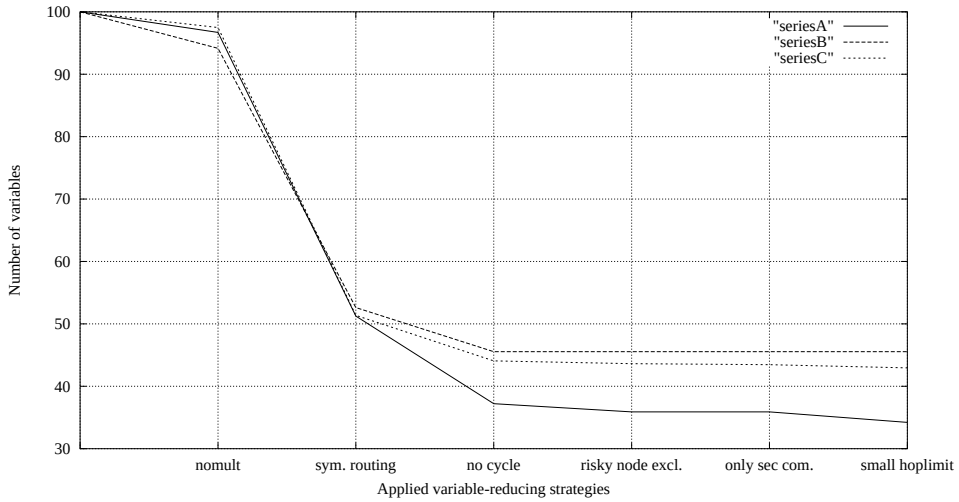


Figure 6.1: Reduction of variables. *Influence on the average number of variables per series concerning the combination 111111.*

6.2 we can see the effects of the exclusion of constraints thru separation (see Section 5.2.2), as well as the further preprocessing on the number of constraints.
Like above the *symmetrical routing* resulted in the major reduction. As a consequence of the merging mechanism the corresponding inequalities (3.14)–(3.16) became irrelevant, and at the same time the number of hoplimit (3.13) and flow balance inequalities (3.2) were reduced, since their amount depends on the number of commodities. Compared to this major effect the impact of the remaining presented preprocessing mechanisms (presented in Section 5.3.1) seems marginal. Only in the instances of series A there exist nodes with an extremely high number of ports and C10 and C12 are the only instances with only secure commodities. The results of the separation of the hoplimit constraints, thus their omission, were relatively slight, even though there were always excluded as many inequalities as merged commodities were present.
Concluding, the number of constraints could be significantly reduced, such that after the application of our mechanisms in general only 10% of the initial constraints for the instances of the series B and C, or 23% for the series A respectively, were present.

6.2.2 Branching rule comparison

We tested the branching rules introduced in Section 5.1.3 using four problem instances of series A and six of the series B and C. Our implementation ran on all possible combinations $000000 - 111111$ for each of these 16 instances.
Table 6.3 gives an overview of the results for the different branching rules. We summarized the results of all combinations for each problem instance consisting of the

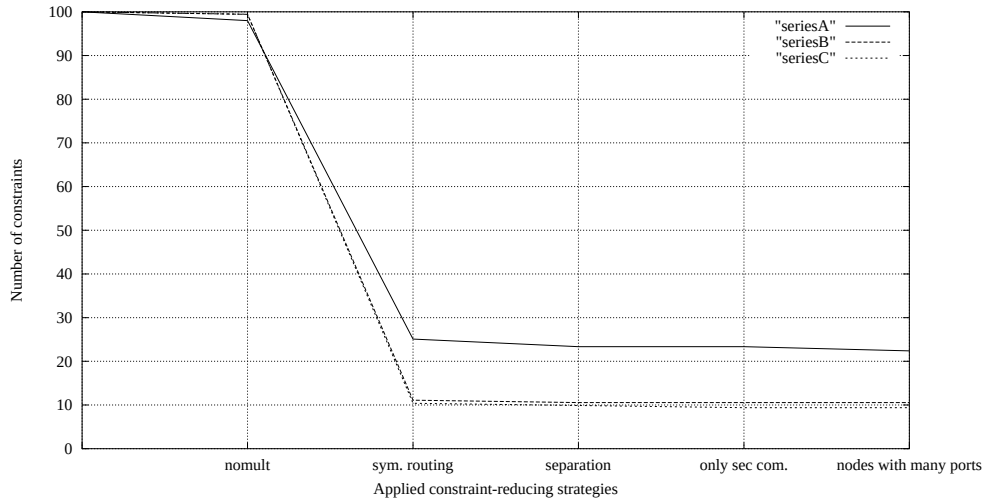


Figure 6.2: Reduction of constraints. *Influence on the average number of constraints per series concerning the combination 111111.*

average gap ⌀GAP (the value 100(upper bound − lower bound)/(lower bound)) and the number of combinations #nU where our implementation did not find any feasible integer solution within the time limit of 10 minutes. We applied the heuristic presented in Section 5.4 (with the additional edge exclusion, the highest secure amount as sort criteria and the pricy flow as arc weights) and the trivial separation to include the hoplimit constraints (see Section 5.2.2). Testing the generic **most fractional** branching rule and the two edge variable brancher developed, **most splitted** and **least splitted**, the following observations seem appropriate:

- In the small and medium problem instances (except for A10) our edge variable brancher resulted in better solutions than the generic most fractional branching rule.

- No great difference between edge variable brancher concerning most/least splitted flow can be noticed. In general the least splitted flow strategy tends to be slightly more efficient.

- For the large instances B16, B20, C16 and C20, we could obtain better results applying the most fractional branching rule than by using the other variable selection strategies. It seems that the overhead caused by the more sophisticated edge variable brancher consumes too much time, such that it is impossible to regain the lost time taking advantage of the benefit of these techniques.

	most fractional		most splitted		least splitted	
	⌀GAP	#nU	⌀GAP	#nU	⌀GAP	# nU
A07 000000 - 111111	7.15	0	3.07	0	2.88	0
A08 000000 - 111111	19.39	0	12.41	0	11.8	0
A09 000000 - 111111	37.59	0	25.16	0	23.91	1
A10 000000 - 111111	43.92	0	32.2	7	31.0	5
B10 000000 - 111111	81.19	21	57.48	20	61.26	19
B11 000000 - 111111	102.08	18	71.14	16	70.86	16
B12 000000 - 111111	91.78	18	80.06	14	74.23	15
B15 000000 - 111111	230.9	1	239.45	0	247.3	0
B16 000000 - 111111	308.63	3	349.06	1	351.74	1
B20 000000 - 111111	287.74	9	314.92	8	315.17	8
C10 000000 - 111111	50.59	26	34.07	21	33.73	20
C11 000000 - 111111	153.2	12	109.2	12	105.95	11
C12 000000 - 111111	47.57	26	40.17	24	39.42	24
C15 000000 - 111111	430.89	0	506.55	1	507.82	1
C16 000000 - 111111	151.55	1	166.99	5	192.69	4
C20 000000 - 111111	1067.97	5	1474.17	5	1482.06	5

Table 6.3: Branching rule comparison. *Results of running three branching rules from Section 5.1.3 on* $16 \cdot 64$ *data sets.*

6.2.3 Hoplimit inclusion

We tested three different strategies taking into account the hoplimit constraints: The two separation approaches presented in Section 5.2.2 were run against the inclusion of the corresponding inequalities into the LP-formulation. For each series we have chosen three problem instances and ran our implementation on all combinations taking into account the hoplimit constraints. We applied the generic most fractional variable selection strategy and the same heuristic settings as in the branching rule test runs.

As it can be seen in Table 6.4, the inclusion of inequalities into the LP-formulation performed best for the most tested instance. Only for the instance C10 more solutions could be found applying the hoplimit inequality separators. In this instance all commodities have the same, relatively large hoplimit of 5. For comparison: In the similar instances A10 and B10 the commodities have hoplimits of 3 or 4, respectively. So we propose an application of the separation approaches when large hoplimits are present. For problem instances with small hoplimits (like the ones in the given benchmark), the inclusion of the hoplimit constraints resulted more appropriate. It appears that the advantage of a faster solving of the LP-relaxation is exhausted by the separation routines.

6.2.4 Heuristic modes

In Section 5.4.4 we presented two different possibilities to vary the attitudes of the applied heuristic. Various settings were tested together with the application of the additional edge exclusion. Our implementation was applied on the secure combinations $100000 - 111111$ for 3 problem instances per problem series.

We only implemented and tested the first two sort criteria (**highest amount** vs. **highest secure amount**) for ordering the set of commodities. The third criterion would only have made sense for exploiting the existence of different hoplimits. Since this occurs only in one instance we concentrated on the other two sort criteria.
Two ways of generating edge weights were compared. The **current flow** which uses the current solution values of the commodity edge flow variables, and the **current pricy flow** which is additionally influenced by the standardized costs of the edge. We also analyzed the influence of the **edge exclusion**, which is applied in the case of low port limits for the nodes.
Table 6.5 gives an overview of the results for the different heuristic settings. As in Table 6.3 the results consists of the average gap ⌀GAP and #nU. We tested all secure combinations applying trivial separation to include the hoplimit constraints and the generic most fractional variable selection strategy.

The following observations seem appropriate:

- The edge exclusion was indispensable since it leads to more feasible integral solutions and, in general, to smaller gaps

	Inclusion into formulation		Separation trivial		Separation using shortest path	
	⌀GAP	#nU	⌀GAP	#nU	⌀GAP	# nU
A07 000100 - 111111	2.99	0	3.19	0	2.96	0
A09 000100 - 111111	22.71	1	24.87	2	28.13	2
A10 000100 - 111111	27.72	4	37.26	4	34.82	7
B10 000100 - 111111	59.43	10	66.92	12	63.15	11
B15 000100 - 111111	202.17	0	269.82	1	268.75	1
B20 000100 - 111111	296.27	4	296.67	4	298.0	5
C10 000100 - 111111	33.86	12	33.4	10	35.09	10
C12 000100 - 111111	31.72	14	34.4	14	34.38	15
C16 000100 - 111111	165.38	4	210.13	5	213.55	4

Table 6.4: Hoplimit integration comparison. *Results of running three different hoplimit constraint integration approaches from Section 5.2.2 on all combinations taking into account the hoplimit constraints, each time for three instances per series.*

- The differences between the obtained gaps for the two sort criteria and also the two arc weight functions seemed marginal. Hence we uprated the number of integer solutions found. Under this criterion, the combination of the current flow as arc weight and the highest amount as sort criteria resulted to be the most advisable together with the application of the additional edge exclusion.

6.2.5 Results

Our implementation was run on all 27 instances of the ROCOCO benchmark suite. All 64 combinations were tested employing the 10-minutes timelimit. Using the results of the previous test series, we applied the most auspicious heuristic settings (with the additional edge exclusion, the highest amount as sort criteria and the current flow as arc weights) and included the hoplimit inequalities into the LP-formulation. For the small and medium instances we employed the developed edge variable brancher (least splitted) and for the large instances the generic most fractional branching strategy. The following tables Table 6.6 - Table 6.12 show the achieved results. For every problem instance and constraint combination, we present the obtained lower bound **LB** and the best feasible integer solutions **UB** found. We compare these values with the best so far published upper bounds **Ilog-UB** listed aside.
Considering the results, the following conclusions can be made:

- The given external upper bounds exactly meet our optimal solution in the instances A04−A06 (and in some combinations for the instance A07). In 19 combinations, applied together with the problem instance C16, we achieved better upper bounds (even though the gap is still high). Also in a combination with the instance B15 and in one with B16 we could identify better solutions. In all other cases the external upper bounds lay in the range imposed by our obtained lower and upper bounds.

- The problem instances B15, B16, B20, C15, C16, C20 and C20 are very large. Since the resulting LP is out of scale, the solution time for the LP-relaxation critically approximates the 10-minute time limit. Therefore the remaining time frame is too tight, so that only poor benefit can be drawn from our extensive endeavors. This fact is revealed in the relatively large gaps between the obtained lower and upper bounds.

- For the instances B25 and C25 we obtained no upper bounds since the solving-time of the LP-relaxation overstepped the time limit, such that the heuristic could not be applied.

- In many combinations for the instances B10 and C10 no integer solution could be found. Since in both instances appear nodes with only 2 inports and 2 outports, our heuristic failed in the combinations considering the **pmax** constraint.

	highest amount								highest secure amount							
	edge exclusion				no edge exclusion				edge exclusion				no edge exclusion			
	flow		pricy flow		flow		pricy flow		flow		pricy flow		flow		pricy flow	
	⌀GAP	#nU	⌀GAP	#nU	⌀GAP	#nU	⌀GAP	#nU	⌀GAP	#nU	⌀GAP	#nU	⌀GAP	#nU	⌀GAP	#nU
A07 100000 - 111111	8.26	0	7.85	0	8.36	0	7.87	0	7.57	0	7.12	0	7.94	0	7.77	0
A08 100000 - 111111	15.81	0	15.59	0	18.56	0	17.36	0	15.88	0	15.56	0	18.40	0	17.58	0
A10 100000 - 111111	44.99	0	41.24	2	47.54	8	64.02	6	43.88	1	45.81	1	63.37	5	82.27	6
B10 100000 - 111111	70.94	16	69.55	16	68.20	16	72.73	16	68.50	16	69.67	16	73.04	16	71.85	16
B12 100000 - 111111	86.17	9	91.03	10	87.99	11	91.03	11	88.56	9	88.93	10	87.84	11	89.83	11
B15 100000 - 111111	190.25	1	186.25	1	170.73	12	172.37	12	191.17	1	194.51	2	167.10	12	171.99	13
C11 100000 - 111111	183.30	12	188.86	12	182.01	12	186.37	12	177.69	12	185.65	12	181.35	12	186.53	12
C15 100000 - 111111	409.68	0	405.00	0	432.26	13	428.23	13	405.85	0	405.83	0	435.91	13	428.03	13
C16 100000 - 111111	182.07	2	186.00	2	126.02	13	124.19	13	178.93	2	169.92	2	133.69	12	127.95	12

Table 6.5: Heuristic approach comparison. *Results of running 4 heuristic parameter settings with/without an advanced edge exclusion presented in Section 5.4.4 on all secure combinations for* 9 *problem instances.*

- In the instance C12 we had problems finding integer solutions for the combinations taking into account the **nomult** constraint. In this instance the mentioned constraint cause a reduction of the linkdesign sets, such that for all edges only two linkdesigns are available. One of these linkdesigns provides only an extremely small amount of capacity (it exists only one commodity out of 132 with a data value less its capacity). The other linkdesign provides much more capacity, but at the same time, it is also much more expensive. The heuristic had problems dealing with this special characteristic. Hence in many combinations no integer solutions could be found for C12.

Instance:	A04			A05			A06		
Comb:	LB	UB	Ilog-UB	LB	UB	Ilog-UB	LB	UB	Ilog-UB
000000	22267.0	22267	22267	30744.0	30744	30744	37716.0	37716	37716
000001	25300.0	25300	25300	32758.0	32758	32758	37716.0	37716	37716
000010	22267.0	22267	22267	31386.0	31386	31386	37970.0	37970	37970
000011	25300.0	25300	25300	32758.0	32758	32758	37970.0	37970	37970
000100	23286.0	23286	23286	31353.0	31353	31353	40789.0	40789	40789
000101	25300.0	25300	25300	34105.0	34105	34105	40804.0	40804	40804
000110	25300.0	25300	25300	31386.0	31386	31386	40789.0	40789	40789
000111	25300.0	25300	25300	39136.0	39136	39136	41349.0	41349	41349
001000	22267.0	22267	22267	30744.0	30744	30744	37716.0	37716	37716
001001	25300.0	25300	25300	32758.0	32758	32758	37716.0	37716	37716
001010	22267.0	22267	22267	31386.0	31386	31386	37970.0	37970	37970
001011	25300.0	25300	25300	32758.0	32758	32758	37970.0	37970	37970
001100	23286.0	23286	23286	31353.0	31353	31353	40789.0	40789	40789
001101	25300.0	25300	25300	34105.0	34105	34105	40804.0	40804	40804
001110	25300.0	25300	25300	31386.0	31386	31386	40789.0	40789	40789
001111	25300.0	25300	25300	39136.0	39136	39136	41349.0	41349	41349
010000	22267.0	22267	22267	30744.0	30744	30744	37716.0	37716	37716
010001	25300.0	25300	25300	32758.0	32758	32758	37716.0	37716	37716
010010	22267.0	22267	22267	31386.0	31386	31386	37970.0	37970	37970
010011	25300.0	25300	25300	32758.0	32758	32758	37970.0	37970	37970
010100	23286.0	23286	23286	31353.0	31353	31353	40789.0	40789	40789
010101	25300.0	25300	25300	34105.0	34105	34105	40804.0	40804	40804
010110	25300.0	25300	25300	31386.0	31386	31386	40789.0	40789	40789
010111	25300.0	25300	25300	39136.0	39136	39136	41349.0	41349	41349
011000	22267.0	22267	22267	30744.0	30744	30744	37716.0	37716	37716
011001	25300.0	25300	25300	32758.0	32758	32758	37716.0	37716	37716
011010	22267.0	22267	22267	31386.0	31386	31386	37970.0	37970	37970
011011	25300.0	25300	25300	32758.0	32758	32758	37970.0	37970	37970
011100	23286.0	23286	23286	31353.0	31353	31353	40789.0	40789	40789
011101	25300.0	25300	25300	34105.0	34105	34105	40804.0	40804	40804
011110	25300.0	25300	25300	31386.0	31386	31386	40789.0	40789	40789
011111	25300.0	25300	25300	39136.0	39136	39136	41349.0	41349	41349
100000	25114.0	25114	25114	33295.0	33295	33295	38963.0	38963	38963
100001	26293.0	26293	26293	33751.0	33751	33751	38963.0	38963	38963
100010	26133.0	26133	26133	33372.0	33372	33372	39343.0	39343	39343
100011	28988.0	28988	28988	43684.0	43684	43684	39343.0	39343	39343
100100	26293.0	26293	26293	33372.0	33372	33372	41782.0	41782	41782
100101	26293.0	26293	26293	35098.0	35098	35098	41797.0	41797	41797
100110	27087.0	27087	27087	33372.0	33372	33372	41782.0	41782	41782
100111	28988.0	28988	28988	46300.0	46300	46300	43184.0	43184	43184
101000	25114.0	25114	25114	33295.0	33295	33295	38963.0	38963	38963
101001	26293.0	26293	26293	33751.0	33751	33751	38963.0	38963	38963
101010	26133.0	26133	26133	33372.0	33372	33372	40258.0	40258	40258
101011	28988.0	28988	28988	43684.0	43684	43684	40258.0	40258	40258
101100	26293.0	26293	26293	33372.0	33372	33372	41782.0	41782	41782
101101	26293.0	26293	26293	35098.0	35098	35098	41797.0	41797	41797
101110	27087.0	27087	27087	33372.0	33372	33372	41782.0	41782	41782
101111	28988.0	28988	28988	46300.0	46300	46300	43184.0	43184	43184
110000	35047.0	35047	35047	43114.0	43114	43114	47615.0	47615	47615
110001	36226.0	36226	36226	43684.0	43684	43684	47615.0	47615	47615
110010	35047.0	35047	35047	43114.0	43114	43114	48193.0	48193	48193
110011	36226.0	36226	36226	43684.0	43684	43684	48193.0	48193	48193
110100	36226.0	36226	36226	43852.0	43852	43852	50980.0	50980	50980
110101	36226.0	36226	36226	45031.0	45031	45031	50980.0	50980	50980
110110	36226.0	36226	36226	43852.0	43852	43852	51715.0	51715	51715
110111	36226.0	36226	36226	50062.0	50062	50062	52563.0	52563	52563
111000	35047.0	35047	35047	43114.0	43114	43114	47615.0	47615	47615
111001	36226.0	36226	36226	43684.0	43684	43684	47615.0	47615	47615
111010	35047.0	35047	35047	43114.0	43114	43114	48193.0	48193	48193
111011	36226.0	36226	36226	43684.0	43684	43684	48193.0	48193	48193
111100	36226.0	36226	36226	43852.0	43852	43852	50980.0	50980	50980
111101	36226.0	36226	36226	45031.0	45031	45031	50980.0	50980	50980
111110	36226.0	36226	36226	43852.0	43852	43852	51715.0	51715	51715
111111	36226.0	36226	36226	50062.0	50062	50062	52563.0	52563	52563
TOTAL	1782558.0	1782558	1782558	2351778.0	2351778	2351778	2708264.0	2708264	2708264

Table 6.6: Results for instances A04, A05 and A06. *The implementation was run on all 64 combinations for every instance. In every instance the optimal solution could be found. These values are identical to the external given upper bounds.*

Instance:	A07			A08			A09		
Comb:	LB	UB	Ilog-UB	LB	UB	Ilog-UB	LB	UB	Ilog-UB
000000	46429.548	49093	47728	53175.965	62713	56576	65150.363	76956	70885
000001	46418.496	48500	47728	53652.759	62482	58195	65216.398	81746	71377
000010	47070.224	48643	48643	53635.708	64218	58025	66471.1	96757	73839
000011	47092.263	48643	48643	53961.491	61608	60804	66518.871	97778	75347
000100	46541.233	50450	48643	53371.244	60455	57185	65127.555	85318	71635
000101	46533.076	49795	48643	53480.096	60863	58195	65084.734	75747	71635
000110	47149.181	48643	48643	53739.592	63933	58372	66364.732	105964	75083
000111	47137.797	49876	48643	53882.898	63562	61608	66433.873	107199	76744
001000	47044.652	47728	47728	53704.389	56576	56576	66112.323	75133	70885
001001	47116.025	47728	47728	54145.794	60907	58195	66333.272	73953	71635
001010	47850.902	48818	48818	53973.551	60991	58025	67524.218	81917	73839
001011	47836.44	49376	48818	54248.265	61608	61172	67758.388	82786	75347
001100	47084.411	49430	48818	53866.751	58430	57185	66003.021	78545	71635
001101	47056.703	48818	48818	54066.047	61240	58195	66212.364	79336	71635
001110	47877.653	48818	48818	54262.525	59365	58372	67531.649	85369	75083
001111	47821.011	48818	48818	54412.656	62086	61608	67621.808	86223	76943
010000	46597.772	47728	47728	53741.181	60391	56576	65550.337	77679	70885
010001	46588.212	48500	47728	54142.405	61614	58195	65457.155	78043	71377
010010	47245.143	50051	48643	54067.474	58372	58025	66815.065	77713	73839
010011	47307.23	48643	48643	54411.455	67480	60804	66966.342	98507	75347
010100	46753.005	48643	48643	53556.313	58372	57185	65212.596	88651	71635
010101	46533.853	49051	48643	53694.875	63916	58195	65218.981	86253	71635
010110	47505.646	48818	48643	53834.679	60433	58372	66600.525	120266	75083
010111	47523.767	49898	48643	54164.867	68847	61608	66806.631	114553	76744
011000	47440.083	47728	47728	54587.931	58930	56576	66474.797	75425	70885
011001	47728.0	47728	47728	55115.224	60254	58195	66896.139	75046	71635
011010	48683.759	48818	48818	55386.571	59916	58025	68053.934	87663	73839
011011	48262.383	48818	48818	55569.692	63783	61172	68440.204	87956	75347
011100	47721.319	48818	48818	54979.189	57185	57185	66425.431	78476	71635
011101	47737.487	48818	48818	55202.016	60622	58195	66727.598	76442	71635
011110	48551.967	48818	48818	55222.397	58372	58372	68033.191	86921	75083
011111	48818.0	48818	48818	55954.423	62921	61608	68315.914	82224	76943
100000	47832.533	49811	49636	56277.72	58562	58562	67830.38	77672	72871
100001	47818.451	50392	49636	56816.659	62786	61106	68359.203	84725	73363
100010	49663.613	51268	50853	60949.655	68630	65504	70192.213	94073	79549
100011	49458.478	52399	50853	61012.843	72544	69678	70381.4	102149	82197
100100	47868.643	51417	49636	56146.023	62991	59171	67919.422	85270	73621
100101	47852.739	51589	49636	56580.439	62608	61106	68341.467	84481	73621
100110	49781.69	51146	50853	60541.807	73309	65504	70157.744	93742	79838
100111	49818.874	51146	50853	60399.954	73290	70694	70345.077	99693	82208
101000	48464.167	49811	49811	56556.23	58562	58562	68631.128	76509	72871
101001	48390.278	50450	49811	57145.426	63286	61106	69394.0	78121	73363
101010	50284.08	51268	50853	61837.503	68630	65504	71683.936	84238	79549
101011	50276.277	51268	50853	61675.826	72392	69678	72212.998	88623	82197
101100	48519.799	49811	49811	56659.907	60425	59171	68754.593	79208	73621
101101	48429.288	50821	49811	57043.515	62608	61106	69374.211	77798	73621
101110	50525.882	51146	50853	61419.02	71729	65504	71781.321	90438	79838
101111	50662.131	51146	50853	61806.09	73905	70694	72147.872	93137	82208
110000	57095.837	59308	58101	65940.393	69349	68630	77464.489	92331	81931
110001	57138.809	59592	58101	68378.911	76240	72686	80461.413	86514	85623
110010	57970.2	58413	58413	66261.017	68630	68630	77756.91	86734	82351
110011	58380.069	59106	58413	69581.133	77495	74942	80357.568	110518	91065
110100	57187.121	58588	58101	66213.139	73401	70245	77432.98	93597	82890
110101	57090.535	60497	58101	68162.978	78884	74476	79932.857	98992	85623
110110	58588.0	58588	58588	66297.039	72424	70245	77819.968	96410	83108
110111	58429.554	59467	59106	69237.526	78559	75303	81647.747	112118	92648
111000	58036.375	58311	58101	66865.026	68630	68630	78187.663	82901	81931
111001	58043.607	58899	58101	69329.842	74150	72686	81575.848	88167	85623
111010	58588.0	58588	58588	67068.82	68630	68630	78696.698	84342	82351
111011	59012.0	59012	59012	70865.48	74972	74942	83860.812	97815	91065
111100	58013.834	58101	58101	67099.492	70747	70245	78380.696	85126	82890
111101	58059.166	58101	58101	69439.228	76392	74476	81592.06	90522	86043
111110	58588.0	58588	58588	67197.01	70605	70245	78757.373	85371	83108
111111	59106.0	59106	59106	71105.311	78047	75303	84160.308	103287	93062
TOTAL	3230031.271	3320027	3290590	3767121.385	4205857	4049540	4525051.864	5647167	4952942

Table 6.7: Results for instances A07, A08 and A09. *The implementation was run on all 64 combinations for every instance. An optimal solution could be found for some combinations concerning the instance A07. In the rest the external given upper bounds are situated in our lower bound upper bound range.*

Instance:	A10			B10			B11		
Comb:	LB	UB	Ilog-UB	LB	UB	Ilog-UB	LB	UB	Ilog-UB
000000	75092.999	104341	82306	14919.499	20570	19390	24732.746	41369	32246
000001	75095.785	109118	83112	14873.945	21265	21012	24509.971	42362	39522
000010	75477.077	108678	82511	15011.515	22473	19390	24689.588	36331	32943
000011	75371.488	108824	86324	14992.953	27549	21090	24584.978	44438	39522
000100	75079.363	100340	83136	14794.282	22914	19390	24043.417	47205	32246
000101	75000.507	111523	84433	14715.403	26906	22163	22767.652	48952	39522
000110	75409.931	104397	83695	14964.038	—	19390	23550.701	39964	33169
000111	75327.88	143311	86324	14383.967	—	22163	24360.083	50974	39522
001000	75374.001	98844	82306	15359.208	23844	19413	25417.289	38473	32312
001001	75382.241	105641	83112	15358.72	23521	21012	25480.462	43030	39682
001010	75698.364	106869	82511	15486.751	21850	19707	25458.984	40847	32943
001011	75716.752	104741	86342	15471.158	24686	21265	25436.231	44026	39682
001100	75375.286	97800	83136	15343.621	23736	19413	25412.736	46826	32312
001101	75216.06	99949	84433	15338.225	27716	22306	25387.551	54222	39948
001110	75703.552	108576	83695	15466.342	—	19787	25492.434	—	33472
001111	75660.072	111791	86384	15430.624	28228	22345	25479.081	52968	39948
010000	75430.939	103586	82306	14680.924	21614	19390	25180.483	38133	32246
010001	75198.703	104998	83112	14830.036	23438	21090	25082.069	44806	39522
010010	75775.952	111766	82511	14813.237	22101	19390	25162.602	35062	32943
010011	75714.233	108657	86324	14641.769	24209	21090	25083.586	43925	39522
010100	75321.055	109023	83695	14243.003	23178	19390	24705.849	40859	32246
010101	75241.669	109876	84757	14026.965	55523	22163	23170.568	51392	39522
010110	75715.213	118644	83695	14213.514	21465	19390	24132.872	33894	33169
010111	75656.973	115252	86324	15030.723	—	22163	23066.577	—	39522
011000	76291.939	91436	82306	15194.494	20400	19449	25631.551	38120	32312
011001	76279.746	99343	83112	15127.377	22932	21265	25624.628	40545	39948
011010	76740.61	110459	82511	15250.135	20570	19707	25631.985	36078	32943
011011	76816.051	100743	86342	15072.923	22519	21265	25541.145	42636	39948
011100	76076.563	99228	83695	15263.683	24701	19449	25606.287	38918	32855
011101	76111.249	98838	84757	15307.041	23781	22580	25505.738	56546	39948
011110	76668.214	103596	83695	15287.353	22843	19787	25501.532	—	33472
011111	76695.987	100124	86384	15204.949	24501	22580	25049.222	—	39948
100000	78311.77	104821	85435	19392.013	31193	22938	32599.196	53182	41461
100001	78187.344	106554	86582	19277.975	33259	23356	32522.741	57917	42444
100010	79731.629	112599	92428	19549.898	—	22938	32637.312	—	41461
100011	79527.273	111431	96305	19545.958	—	23411	32540.445	—	42444
100100	78260.864	107693	87027	19263.262	31478	23330	32573.515	55278	41627
100101	78218.15	110314	87905	19279.075	31891	23356	32105.132	59847	42444
100110	79782.111	105720	93016	19533.62	—	23330	32508.909	—	41627
100111	79446.195	264894	97396	19537.043	—	23411	32530.949	—	42444
101000	78604.598	101867	85435	20006.302	28163	23385	33991.69	51719	41461
101001	78444.024	112443	87436	20020.522	29470	24113	34075.557	51699	43012
101010	80008.428	111415	92554	20181.676	—	23658	34060.345	—	41461
101011	79872.463	118243	96305	20224.661	—	24223	34120.289	—	43012
101100	78515.695	108291	87027	20020.783	28111	23813	33994.969	52699	42247
101101	78502.763	101896	87905	19968.225	29934	24372	33845.724	50662	43580
101110	80005.751	102606	93016	20227.271	—	23813	33658.191	—	43317
101111	79782.374	131923	97396	20242.006	—	24412	34144.502	—	43580
110000	90978.401	115175	100141	19638.346	30997	22938	33012.411	49187	41461
110001	91151.892	135085	105144	19607.944	27351	23356	32876.626	59241	42444
110010	91068.875	116609	100141	19689.414	—	22938	33005.595	52338	41461
110011	91074.009	150149	109073	19658.136	—	23902	32928.27	—	42444
110100	91072.596	124144	101289	19617.554	39510	23330	32893.258	73366	41627
110101	91321.228	132835	105659	19623.004	26696	23356	32611.364	87950	42444
110110	90996.641	148596	101289	19646.317	—	23330	32900.51	—	41627
110111	90679.513	165083	109754	19553.963	—	24068	32841.095	—	42444
111000	91880.662	113153	100141	20314.36	28277	23658	34218.916	49771	41461
111001	92335.252	121701	105144	20350.128	29485	24113	34098.757	49796	43012
111010	91893.055	107851	100141	20396.878	—	23658	34149.769	49249	41461
111011	92261.176	119701	109754	20387.78	27662	24223	34267.307	47650	43012
111100	91898.819	122699	101289	20215.142	34354	23813	34245.832	53734	42247
111101	92215.761	127116	105659	19866.979	36800	24372	33861.359	51745	43580
111110	91790.349	131172	101289	20333.134	—	23813	34064.428	—	43317
111111	92150.723	161077	109754	20332.922	—	24472	34199.273	51978	43580
TOTAL	5151686.838	7415168	5808115	1115600.668	—	1416583	1864564.834	—	2514301

Table 6.8: Results for instances A10, B10 and B11. *The implementation was run on all 64 combinations for every instance. In all instances the external given upper bounds are situated in our lower bound upper bound range.*

Instance:	B12			B15			B16		
Comb:	LB	UB	Ilog-UB	LB	UB	Ilog-UB	LB	UB	Ilog-UB
000000	19995.476	29168	24681	12313.854	33248	26126	11206.359	65402	28335
000001	19799.097	39219	36644	12296.277	39617	33966	11095.792	65427	30193
000010	19847.58	28493	24878	12313.486	33807	26337	11194.959	65898	28335
000011	19877.379	—	36708	12171.657	52466	33966	11055.97	71155	31962
000100	19688.324	30955	24970	12310.746	55484	26939	11207.665	60574	28496
000101	20092.506	39506	36936	12296.277	57343	35775	11129.273	66357	35682
000110	19801.004	37504	24970	12317.144	64885	27800	11194.959	70344	30193
000111	20276.998	40753	36936	12152.607	67451	35775	11051.98	68607	35682
001000	20424.501	27958	25036	12874.965	28542	26126	11579.317	37261	28335
001001	21149.994	39959	37094	12806.18	39172	33966	11565.579	46951	31962
001010	20431.125	26362	25036	12873.46	38864	26978	11571.995	47303	28335
001011	21128.72	38241	37094	12703.729	37151	33966	11551.524	59362	31962
001100	20402.485	33134	25036	13372.93	55070	28866	11575.226	56416	29308
001101	21140.131	41023	37212	12710.553	51162	36003	11571.995	58458	35682
001110	20440.497	28415	25036	12851.572	56106	29259	11573.624	42065	30193
001111	21156.741	41129	37212	12720.574	56112	36003	11460.492	67648	35682
010000	20091.456	28665	24731	12407.929	51432	26126	11564.828	70250	28586
010001	20427.207	38859	36644	12324.267	51944	33966	11129.273	73328	31872
010010	20250.174	27961	24878	12510.544	28341	27518	11564.828	63202	28656
010011	20200.458	39106	36708	12205.793	54176	33966	11181.508	65898	31962
010100	20159.993	53168	24970	12386.631	64687	27800	11514.623	70250	29483
010101	20216.426	39047	37212	12324.267	60602	36522	11099.234	76718	36074
010110	20128.987	—	24970	12407.929	62185	27800	11564.828	71172	31428
010111	20054.891	38201	37212	12324.267	67290	36664	11129.273	70828	36074
011000	20620.755	28810	25036	15171.659	28644	26126	12574.999	30891	28586
011001	21125.664	39471	37094	14844.029	35850	33966	12497.765	31701	31962
011010	20620.784	26527	25036	15189.089	27158	28809	12536.092	30169	28656
011011	21097.479	40882	37094	13324.785	35593	33966	12472.294	32832	31962
011100	20558.417	36023	25036	13118.238	39071	28866	12233.743	49494	31212
011101	20951.661	39711	37212	13103.235	66757	36701	11935.82	65650	36074
011110	20583.003	—	25036	14818.459	55859	32458	11935.82	71179	31428
011111	21197.704	37892	37212	14354.903	54236	36701	12233.743	71179	36074
100000	26778.641	46321	34697	18618.146	52893	32797	16601.026	69422	34401
100001	26960.427	49320	37438	18587.063	51342	35823	16326.25	67473	35528
100010	27096.288	—	34729	18622.79	64303	33067	16588.115	69316	34401
100011	26746.013	—	37815	18586.142	65263	36110	16326.25	68150	35528
100100	26746.857	46944	34697	18691.753	46039	32797	16597.637	69422	35889
100101	26909.985	48124	38017	18429.519	58051	36522	16468.466	69290	36498
100110	26826.713	—	35202	18618.146	65964	33067	16579.91	68957	36544
100111	26085.096	54223	38168	18581.498	61511	36664	16509.74	—	36544
101000	29089.965	40306	35202	19606.127	45747	34387	17671.42	47927	35207
101001	29264.052	42711	38017	19600.994	43857	35823	16090.77	45504	35528
101010	29053.473	—	35202	19671.425	57350	34662	16154.413	58470	35207
101011	29309.091	47929	38147	19599.59	54721	36247	16154.413	72915	35528
101100	29067.414	41954	35202	19603.244	48095	34387	16090.77	55250	35972
101101	29268.588	41359	38017	19543.275	46714	37403	17071.002	50691	36498
101110	29007.843	49905	35202	19606.127	61398	34927	16230.814	70264	36544
101111	29106.371	42871	38168	19590.656	60299	37403	16154.413	71580	36544
110000	27011.974	67449	34729	18714.368	69370	33023	16739.927	79436	34401
110001	26869.815	73124	37815	18752.797	75042	35823	16518.44	75790	35528
110010	26881.4	—	34729	18787.694	58272	33067	16739.927	66040	34401
110011	26591.289	45477	37815	18737.293	65623	36110	16549.812	68439	35528
110100	26689.91	69136	34868	18786.112	69202	33023	16661.023	79436	35889
110101	26550.489	72144	38911	18733.822	71098	36522	16488.629	78772	36544
110110	26465.435	—	35202	18758.3	65964	33067	16694.255	68957	36544
110111	26614.56	—	38911	18775.073	67785	36664	16549.812	68249	36544
111000	29203.323	68660	35202	19943.633	38168	34642	17834.97	40366	35510
111001	29381.657	41855	38147	20815.554	62244	35823	18025.851	37408	35528
111010	29223.01	—	35202	21238.38	38096	34662	17834.97	59387	35510
111011	29355.476	—	38147	20672.169	37742	36247	17245.072	39236	35528
111100	29217.623	64716	35202	19872.551	64149	34927	17890.953	64221	36544
111101	29439.263	43302	39720	20891.536	69553	38007	17536.851	61643	36544
111110	29187.214	—	35202	21179.091	58428	34927	17834.97	67671	36544
111111	29315.671	—	40441	20954.028	62040	38007	17713.758	71000	36544
TOTAL	1549252.543	—	2165723	1035070.931	3406628	2132433	909230.009	—	2156418

Table 6.9: Results for instances B12, B15 and B16. *The implementation was run on all 64 combinations for every instance.In the instance B15 we achieved a better upper bound concerning the combination 011010. In the instance B16 we could identify a better upper bound for the combination 011001. In the other instances and combinations, the external given upper bounds are situated in our lower bound upper bound range.*

Instance:	B20			B25			C10		
Comb:	LB	UB	Ilog-UB	LB	UB	Ilog-UB	LB	UB	Ilog-UB
000000	21245.583	106600	47180	24865.103	—	74051	10398.147	11675	11472
000001	21245.583	106600	47180	24865.103	—	74051	10982.051	14485	13388
000010	21245.583	120541	47465	24865.103	—	74051	12376.412	16334	16194
000011	21245.583	118326	47639	24865.103	—	74051	13010.948	—	18448
000100	21245.583	106600	49770	24865.103	—	80281	10293.747	12270	11472
000101	21245.583	106600	51625	24865.103	—	85721	10874.154	15263	13388
000110	21245.583	—	49770	24865.103	—	90916	12017.998	—	16194
000111	21245.583	—	51625	24865.103	—	90916	12335.761	—	18925
001000	21855.158	103948	47180	25082.308	—	74051	10428.978	11912	11465
001001	21855.158	103948	47180	25082.308	—	74051	11200.251	14098	13388
001010	21855.158	115086	47639	25082.308	—	74051	12909.566	16587	16194
001011	21855.158	113348	47639	25082.308	—	74051	13825.834	—	18925
001100	21855.158	103441	50317	25082.308	—	87244	10447.256	11645	11483
001101	21855.158	103441	53374	25082.308	—	87244	11263.987	15059	13388
001110	21855.158	110078	50514	25082.308	—	92278	12940.713	20449	16194
001111	21855.158	110078	53602	25082.308	—	92278	13846.861	—	21560
010000	21245.583	111184	47180	24865.103	—	74051	11465.957	17373	14141
010001	21245.583	111184	47180	24865.103	—	74051	12096.498	19228	16043
010010	21245.583	117225	47465	24865.103	—	74051	13210.033	—	16194
010011	21245.583	116695	47639	24865.103	—	74051	13970.733	—	18448
010100	21245.583	111184	49770	24865.103	—	80281	11501.616	14973	14141
010101	21245.583	111184	51625	24865.103	—	85721	12013.898	17953	16043
010110	21245.583	—	49770	24865.103	—	92124	13341.438	—	16194
010111	21245.583	—	51625	24865.103	—	92124	14031.767	—	18925
011000	22057.543	120466	47180	25148.987	—	74051	11706.79	15139	14141
011001	21855.158	120466	47180	25148.987	—	74051	12510.421	17212	16085
011010	22081.431	115086	47639	25107.781	—	74051	14193.801	17314	16194
011011	21855.158	113348	47639	25107.781	—	74051	14466.228	—	18925
011100	21937.53	119959	50376	25107.781	—	87244	11722.196	15264	14141
011101	21855.158	120466	53374	25148.987	—	87244	12527.024	17491	16164
011110	22057.543	110078	51125	25107.781	—	93840	14325.729	19339	16194
011111	21855.158	110078	53602	25107.781	—	93840	14602.132	—	21560
100000	35631.138	109583	67370	41017.034	—	90297	12227.56	16647	15157
100001	35631.138	109132	67370	41017.034	—	90297	12185.061	17926	16664
100010	35631.138	—	68233	41017.034	—	92077	12666.657	17894	16194
100011	35631.138	—	68233	41017.034	—	92077	13283.854	—	18448
100100	35631.138	109132	67574	41017.034	—	98439	12084.579	15494	15157
100101	35631.138	109132	67574	41017.034	—	103893	11651.743	20216	16664
100110	35631.138	116721	71881	41017.034	—	98439	12561.78	—	16194
100111	35631.138	116721	72884	41017.034	—	105268	12950.89	—	18925
101000	36993.179	103056	67574	41933.815	—	90297	12549.585	16396	15157
101001	36993.179	103056	67574	41933.815	—	90297	12613.21	18304	16664
101010	36993.179	113768	69119	41933.815	—	92077	13452.852	16628	16194
101011	36993.179	113768	69119	41933.815	—	92077	14037.853	20114	18925
101100	36993.179	103056	67574	41933.815	—	98439	12590.487	16322	15157
101101	36993.179	103056	67574	41933.815	—	103893	12677.734	18171	17015
101110	36993.179	110436	73982	41933.815	—	98439	13432.278	16556	16194
101111	36993.179	110436	74372	41933.815	—	106396	13868.179	—	21560
110000	35631.138	120988	67370	41017.034	—	90297	12276.721	15919	15157
110001	35631.138	120988	67370	41017.034	—	90297	12721.835	18952	16664
110010	35631.138	—	68233	41017.034	—	92077	13483.561	16194	16194
110011	35631.138	—	68233	41017.034	—	92077	14411.624	—	18448
110100	35631.138	120988	67574	41017.034	—	98439	12275.212	17507	15157
110101	35631.138	120988	67574	41017.034	—	103893	12779.126	18420	16664
110110	35631.138	116721	71881	41017.034	—	98439	13652.857	—	16194
110111	35631.138	116721	72884	41017.034	—	105268	13879.94	—	18925
111000	36993.179	119101	67574	41962.425	—	90297	12718.958	15919	15157
111001	36993.179	119101	67574	41962.425	—	90297	13479.674	18956	16664
111010	36993.179	113768	69119	41962.425	—	92077	14500.453	16194	16194
111011	36993.179	113768	69119	41962.425	—	92077	15634.18	—	18925
111100	36993.179	119101	67574	41962.425	—	98439	12779.648	15499	15157
111101	36993.179	119101	67574	41962.425	—	103893	13418.02	17531	17015
111110	36993.179	110436	73982	41962.425	—	98439	14682.532	16536	16194
111111	36993.179	110436	76937	41962.425	—	106396	15779.27	—	21560
TOTAL	1852314.343	—	3798647	2126928.442	—	5689516	820146.838	—	1046149

Table 6.10: Results for instances B20, B25 and C10. *The implementation was run on all 64 combinations for every instance. In all instances the external given upper bounds are situated in our lower bound upper bound range. In the instance B25 we obtained no upper bounds since the solving-time of the LP-relaxation overstepped our time limit.*

Instance:	C11			C12			C15		
Comb:	LB	UB	Ilog-UB	LB	UB	Ilog-UB	LB	UB	Ilog-UB
000000	15766.287	23913	20889	30711.879	43582	35334	17599.665	56850	39347
000001	15649.138	28634	25641	30729.697	42964	35373	17428.125	111678	50491
000010	15721.777	24848	21428	30740.302	52806	35480	17599.713	114951	39347
000011	15609.039	31026	25641	30730.012	—	35480	17451.851	139597	50491
000100	14600.493	24395	20889	30701.603	41210	35334	17597.224	60059	42086
000101	14127.848	27850	25641	30526.886	44970	35373	17533.874	182879	54042
000110	15136.679	25435	21685	30706.013	—	35480	17589.153	158968	42378
000111	14548.701	34504	25641	30679.124	—	35480	17430.969	152975	54654
001000	16035.232	22655	20889	31110.932	41250	35334	18361.316	61884	39347
001001	16402.051	29565	25991	31111.363	40778	35835	18212.493	99194	51108
001010	16271.685	21727	21428	31125.565	49280	35880	18401.786	57528	39347
001011	16157.905	32383	26085	31129.743	49267	35880	18361.029	128510	51108
001100	16261.012	28343	20889	31108.933	41280	35334	18361.316	113151	42631
001101	16280.557	30011	26341	31101.255	42201	35880	18340.855	137915	54042
001110	16303.241	25338	21685	31111.35	45008	35880	18380.286	144495	43328
001111	16256.317	32831	26341	31100.704	—	35880	18251.522	142216	55796
010000	15852.657	24059	20889	30947.689	—	35480	17812.716	47436	39347
010001	15643.608	29227	25641	30909.855	—	35480	17595.214	73415	50491
010010	15651.183	23002	21428	30963.198	42516	35480	17783.952	48450	39347
010011	15671.104	30278	25641	30892.693	41769	35480	17579.968	72975	50491
010100	15179.538	23512	20889	30676.854	—	35480	17639.586	273805	42086
010101	14463.395	29432	25641	30469.436	—	35480	17619.336	71647	54042
010110	14930.167	25630	21685	30717.193	39046	35480	17639.586	124847	42378
010111	14363.316	32445	25641	30396.015	40454	35480	17671.723	143608	54654
011000	16104.933	23136	20889	31319.902	—	35835	21401.021	46493	39347
011001	16328.097	30665	25991	31298.706	—	35835	20635.778	68654	51108
011010	16121.225	22425	21428	31283.627	40985	35909	21293.72	47692	39347
011011	16303.618	29376	26085	31307.243	39302	36742	20491.941	71331	51108
011100	16069.693	25336	20889	31314.226	—	35909	20987.133	60603	42631
011101	16403.271	31925	26528	31290.481	—	36238	20251.616	154987	54042
011110	15994.096	28468	21685	31299.84	40649	35909	20367.978	98284	43679
011111	16377.795	31703	26528	31293.523	42292	36857	20373.53	106285	55796
100000	21524.023	41945	29468	30908.82	43756	35512	30217.923	97310	54638
100001	21065.776	42693	29530	30914.458	44465	35880	29991.403	169694	60531
100010	21509.88	—	29577	30915.447	—	35880	30262.853	144023	56688
100011	21346.636	—	29577	30912.288	—	35880	30014.558	143231	60531
100100	21683.772	62490	29530	30745.884	43085	35593	30196.512	174186	54638
100101	21372.017	53039	29530	30719.075	47248	35880	29767.427	189087	60531
100110	21583.586	—	29577	30740.461	—	35880	30261.988	154356	56688
100111	21060.62	—	29577	30719.476	—	35880	30030.079	147944	60531
101000	22997.951	42772	29468	31260.105	42313	35880	32086.238	115732	55022
101001	22735.643	42662	29577	31265.268	40874	35880	31751.85	158515	60531
101010	23146.494	—	29577	31252.294	52948	35880	31765.444	121698	57090
101011	23141.455	—	29577	31249.265	49448	35880	32094.208	126964	60531
101100	23016.172	43524	29577	31239.125	42191	35880	31751.882	178806	55022
101101	23233.525	43932	29577	31235.081	44117	35880	31839.74	157858	60531
101110	23164.923	—	29577	31225.315	42380	35880	32186.267	146161	57090
101111	23105.816	36884	29577	31251.797	48464	35880	32164.584	140618	60531
110000	21711.565	101300	29577	30954.871	—	35909	30309.81	93993	54638
110001	21131.305	44665	29577	30913.109	—	35933	30309.81	94705	60531
110010	21385.067	39383	29577	30947.249	39059	35909	30309.81	88716	56688
110011	20991.058	49459	29577	30892.06	42653	36604	30309.81	94705	60531
110100	21477.922	45067	29577	30663.005	—	35909	30248.063	308164	54638
110101	20751.852	114975	29577	30423.093	—	36238	30210.972	308461	60531
110110	21553.612	—	29577	30671.944	41202	35909	30244.933	154356	56688
110111	21125.161	47113	29577	30408.018	41264	36857	30248.063	154356	60531
111000	23087.41	41610	29577	31319.936	—	35909	34029.275	81210	55022
111001	22733.92	99347	29577	31347.77	—	35933	33414.061	96785	60531
111010	23063.378	37683	29577	31318.681	40147	35909	34029.275	86651	57480
111011	22947.672	—	29577	31333.254	40350	36857	33539.322	82046	60531
111100	23182.129	92149	29577	31318.329	—	35909	34002.6	271354	55022
111101	23156.913	38324	29577	31274.67	—	36238	33414.061	291057	60531
111110	23023.702	—	29577	31294.798	42175	35909	33920.081	162267	57480
111111	23030.076	—	29577	31287.386	40082	36857	33367.292	142846	60531
TOTAL	1212626.689	—	1700687	1983728.174	—	2294525	1602336.169	8251217	3362465

Table 6.11: Results for instances C11, C12 and C15. *The implementation was run on all 64 combinations for every instance. In all instances the external given upper bounds are situated in our lower bound upper bound range.*

Instance:	C16			C20			C25		
Comb:	LB	UB	Ilog-UB	LB	UB	Ilog-UB	LB	UB	Ilog-UB
000000	12195.029	28313	26289	30264.927	650420	68831	23247.521	—	70994
000001	12170.665	25744	26289	30264.927	669547	69564	23247.521	—	70994
000010	12176.177	40550	26482	30264.927	238627	68831	23247.521	—	72271
000011	12174.139	40550	26754	30264.927	234785	70526	23247.521	—	72271
000100	12175.682	28117	26289	30264.927	660834	71705	23247.521	—	81267
000101	12170.733	25744	26289	30264.927	660834	76999	23247.521	—	85278
000110	12174.776	49162	26482	30264.927	241290	73658	23247.521	—	87710
000111	12174.776	47913	26754	30264.927	243122	78376	23247.521	—	88232
001000	13182.539	25449	26289	31332.591	118870	68831	23451.233	—	73161
001001	13015.877	25918	26289	31315.902	234947	70526	23451.233	—	73508
001010	13174.672	28874	26545	31603.236	238833	68831	23451.233	—	73508
001011	13003.574	30552	26769	31603.236	241513	70526	23451.233	—	73508
001100	13187.133	25449	26289	31499.549	148581	71968	23451.233	—	86598
001101	13015.877	25918	26289	31315.902	299554	76999	23451.233	—	95008
001110	13173.811	26721	26566	31603.236	243609	73658	23451.233	—	96768
001111	12994.547	30988	26769	31603.236	238833	78851	23451.233	—	96768
010000	12231.569	25622	27054	30465.041	700156	68831	23268.697	—	73161
010001	12175.293	35317	27191	30465.041	700156	69564	23268.697	—	75486
010010	12231.569	25016	27243	30465.041	234635	68831	23247.521	—	73990
010011	12171.137	27019	27267	30465.041	236990	70593	23247.521	—	75486
010100	12171.137	62524	27098	30465.041	700156	71722	23247.521	—	83268
010101	12171.137	34746	27265	30465.041	700156	77338	23268.697	—	88016
010110	12171.137	31010	27258	30441.054	244260	73900	23247.521	—	87710
010111	12171.137	31038	27267	30465.041	248278	78851	23247.521	—	88232
011000	13223.411	28028	27081	31587.979	587420	68831	23552.609	—	73161
011001	13114.439	27532	27191	31587.979	89320	70593	23509.59	—	75605
011010	13240.523	24775	27243	31587.979	95617	68831	23473.686	—	75332
011011	13071.18	28852	27267	31587.979	129679	70593	23473.686	—	75605
011100	13197.372	—	27098	31587.979	307957	72764	23509.59	—	88688
011101	13092.1	28485	27265	31587.979	253468	78851	23509.59	—	95008
011110	13213.766	30164	27258	31587.979	243609	74253	23473.686	—	96768
011111	13059.283	30894	27267	31587.979	115818	78851	23473.686	—	96768
100000	12588.934	30047	26445	30986.232	665844	95842	43438.097	—	100569
100001	12527.841	26483	26670	30986.232	665844	96948	43438.097	—	100569
100010	12601.262	56445	26872	30977.462	—	95842	43438.097	—	110945
100011	12527.841	27678	27010	30977.462	—	96948	43438.097	—	112726
100100	12583.611	28961	26670	30986.232	665844	97979	43438.097	—	108517
100101	12517.96	27522	26670	30986.232	665844	99098	43438.097	—	113323
100110	12586.717	54024	26888	30986.232	249775	97979	43438.097	—	115888
100111	12549.338	60518	27010	30986.232	251448	100055	43438.097	—	115888
101000	14508.66	24710	26445	34605.267	179541	95842	44662.061	—	100569
101001	14337.572	27024	26670	34605.267	233636	98173	44662.061	—	100569
101010	14509.414	26702	26872	34784.757	238622	95842	44662.061	—	111147
101011	14232.443	26353	27010	34784.757	239346	99414	44662.061	—	114915
101100	14486.785	29333	26670	34784.757	189567	97979	44662.061	—	113323
101101	14337.572	27960	26670	34621.956	204223	99098	44662.061	—	113323
101110	14499.008	44848	26888	34890.751	240478	97979	44662.061	—	115888
101111	14204.111	27914	27010	34890.007	240478	100976	44662.061	—	115888
110000	12580.537	59029	27135	30986.691	784186	95842	43438.097	—	100569
110001	12549.182	30002	27254	31175.761	669755	96948	43438.097	—	100569
110010	12672.413	22148	27243	30986.691	—	95842	43438.097	—	111147
110011	12549.182	34245	27267	31175.761	—	96948	43438.097	—	112726
110100	12580.537	59029	27267	31175.761	669755	100323	43497.628	—	108517
110101	12549.182	39707	27267	31175.761	669755	100323	43497.628	—	113323
110110	12580.537	33098	27267	31175.761	251221	101313	43497.628	—	115888
110111	12549.182	35247	27267	31175.761	—	101313	43497.628	—	115888
111000	14654.615	22729	27243	34873.187	583326	95842	44704.017	—	100569
111001	14508.116	27557	27254	34873.187	420250	98173	44704.017	—	100569
111010	14745.154	25082	27243	34873.187	202346	95842	44704.017	—	111147
111011	14542.634	26971	27267	34873.187	239346	101313	44704.017	—	114915
111100	14747.503	25416	27267	34873.187	583326	100323	44704.017	—	113323
111101	14534.614	32509	27267	34873.187	415263	100323	44704.017	—	113323
111110	14637.785	22969	27267	34873.187	205734	101313	44704.017	—	115888
111111	14488.57	31900	27267	34873.187	212660	101313	44704.017	—	115888
TOTAL	837935.009	—	1723258	2044249.756	—	5471165	2157786.151	—	6148424

Table 6.12: Results for instances C16, C20 and C25. *The implementation was run on all 64 combinations for every instance. In 19 combinations of the instance C16 we are able to identify better upper bounds. In all the other instances the external given upper bounds are situated in our lower bound upper bound range. In the instance C25 we obtained no upper bounds since the solving-time of the LP-relaxation overstepped our time limit.*

Chapter 7

Conclusion

In this thesis, a sophisticated model for the problem considered in the ROCOCO project has been developed. It is the first model published which includes all additional constraints such that any combination can be handled by parameterization. We have examined the associated polyhedra and have introduced a number of valid inequalities. We have been able to show that some of these are even facet-defining for a relaxation of the problem. These inequalities have been used as cutting planes in the applied branch-and-cut algorithm. The algorithm has been briefly explained and some strategies for obtaining better solutions have been introduced together with separation approaches. A detailed description to the applied preprocessing steps has been given which resulted to be very effective. We have presented a newly developed and implemented heuristic which reliably identifies integer solutions for the problem. This heuristic splits the network design problem into two subproblems (Determination of a feasible routing / Calculation of a appropriate linkdesign installation) and solves them successively. We have evaluated different algorithm strategies and have measured the effects of the applied preprocessing and elaborate data transformation using the given benchmark suite.

We have found optimal solutions for the small instances with exactly the same values as the best published results. This indicates that the developed model properly represents the initial problem description. For the larger problem instances the gap between the upper and lower bound turned out to be quite large, but for some instances we have been even able to identify better integer results than the best published ones.

Altogether, we have proposed a framework that successfully provides good solutions for the tested small and medium problem instances. The main problem has turned out to be the fact that the networks of the given problem instances are extremely dense. For the largest instances, the number of variables is much larger than what can be handled by any state of the art solver within the assumed 10-minutes time limit. Hence more preprocessing or ultimately a path based model with column generation approach may be worth considering. Further improvement could be achieved by increasing the benefit of the given grouping of variables (e.g. GUB-branching) and also more tests for the enhancement of the branch-and-cut algorithm seem appropriate.

List of Tables

List of Figures

Bibliography

[1] Network Time Protocol (NTP) project. Information available at `http://www.ntp.org`.

[2] E. Danna C. LePape. A. Chabrier. Solving a network design problem. Technical Report 02-005, ILOG, 2002.

[3] Algorithmic Solutions Software GmbH, Schützenstr. 3-5, D- 66123 Saarbrücken, Germany. LEDA—*Library of Efficient Data types and Algorithms*, 1998–2003. Information available at `http://www.algorithmic-solutions.com`.

[4] atesio GmbH, Rubensstr. 126, D-12157 Berlin, Germany. DISCNET, 2000–2003. Information available at `http://www.atesio.de`.

[5] D. Bienstock and O. Gunluk. Capacitated network design - polyhedral structure and computation.

[6] J.C. Régin C. Le Pape, L. Perron and Paul Shaw. Robust and parallel solving of a network design problem, 2002.

[7] Alain Chabrier. Heuristic branch-and-price-and-cut to solve a network design problem, 2003.

[8] V. Chvátal. *Linear programming*. A series of books in the mathematical sciences. New York - San Francisco: W. H. Freeman and Company, 1983.

[9] H. Crowder, E. Johnson, and M. Padberg. Solving large-scale 0-1 linear programming problems, 1983.

[10] ILOG CPLEX Division, 889 Alder Avenue, Suite 200, Incline Village, NV 89451, USA. *ILOG CPLEX 8.0 Reference Manual*, 2002. Information available at `http://www.cplex.com`.

[11] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., 1988.

[12] S. Orlowski. Local and global restoration of node and link failures in telecommunication networks. Master's thesis, TU Berlin, 2003.

[13] Claude Le Pape Laurent Perron Raphal Bernhard, Jaques Chambon and Jean Charles Régin. Résolution d'un problème de conception de réseau avec parallel solver, 2002.

[14] A. Schrijver. *Theory of linear and integer programming.* Wiley-Interscience Series in Discrete Mathematics. A Wiley-Interscience Publication. Chichester: John Wiley & Sons Ltd., 1986.

[15] C. P. M. van Hoesel, A. M. C. A. Koster, R. L. M. J. van de Leensel, and M. W. P. Savelsbergh. Polyhedral results for the edge capacity polytope. *Mathematical Programming, series A*, 92(2):335–358, 2002.

[16] R. Wessäly. *Dimensioning survivable capacitated networks.* PhD thesis, TU Berlin, 2000.

[17] L. A. Wolsey. *Integer Programming.* Wiley-Interscience Series in Discrete Mathematics and Optimization. New York: WileyInterscience, 1998.

[18] L.A. Wolsey. Valid inequalities for 0–1 knapsack and MIPs with generalized upper bound constraints. *Discrete Applied Mathematics*, 29:251–261, 1990.

German Summary, Zusammenfassung

In der vorliegenden Diplomarbeit untersuchen wir die Optimierung von ausfallsicheren Telekommunikationsnetzen. Motiviert wurde diese Arbeit durch eine Problemstellung, welche im Kontext des ROCOCO-Projekts bearbeitet wurde und zum Ziel hatte *robuste* Algorithmen zu entwickeln. Als robust werden Algorithmen bezeichnet, welche nicht nur gute Ergebnisse für Probleminstanzen verschiedener Grösse und Charakteristik liefern, sondern auch weiterhin gut arbeiten wenn Bedingungen hinzugefügt oder entfernt werden.

Ausgehend von einer textuellen Problembeschreibung entwickeln wir ein umfassendes Modell. Es handelt sich um das nach unserer Kenntnis erste Modell, durch das sich jede beliebige Kombination der betrachteten Bedingungen abbilden lässt. Das durch das Model induzierte Polyeder wird untersucht und gültige Ungleichungen werden vorgestellt. Wir zeigen, dass einige dieser Ungleichungen facettendefinierend für eine Relaxierung unseres Problems sind. Die ermittelten Ungleichungen werden als Schnittebenen in einem Branch-and-Cut-Algorithmus verwendet, welcher als zentraler Algorithmus eingesetzt wird. Auf diesen Algorithmus wird genauer eingegegangen, und es werden Strategien eingeführt, durch die die Lösbarkeit verbessert werden kann. Weiterhin werden umfangreiche Preprocessing-Prozeduren und eine neu entwickelte und implementierte Heuristik vorgestellt.

Wir haben unsere Implementation an der für das ROCOCO-Project entwickelten Benchmarksuite getestet, was ausführlich beschrieben wird. Desweiteren werden für alle Datensätze und Kombinationen von Bedingung die erreichten Ergebnisse mit den besten verfügbaren Lösungen verglichen. Es wird gezeigt, dass wir in den kleineren Probleminstanzen exakt dieselben Optimallösungen erreichen und für einige Instanzen sogar ganzzahlige Lösungen erhalten, die besser als die bis jetzt veröffentlichten Lösungen sind.

Statutory Declaration, Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt die selbständige und eigenhändige Anfertigung dieser Diplomarbeit.

(Ulrich Menne)