Network optimization:
Integration of Hardware Configuration and Capacity Dimensioning

Diplomarbeit bei Prof. Dr. M. Grötschel

vorgelegt von Alexander Kröller

am Institut für Mathematik der Technischen Universität Berlin

Berlin, Juni 2003

Contents

Introduction

In this thesis, a model to incorporate hardware configuration decisions into telecommunication network optimization is proposed. The network of a telecommunication company is usually one of the company's major assets. Hence, the ability to build networks of low cost is of great value, and therefore a lot of work has been put into exploring the structure of such networks in the recent years.

In the beginning, optimization of telecommunication networks consisted mostly of finding cost-minimal subgraphs in a given network, with additional connectivity requirements of some type. Such an approach carries the intrinsical assumption that every edge of the network can support whatever data flow is sent over it.

Consequently, advanced models were developed to get rid of this inaccuracy. These consider the fact that every edge of the network is operated with a certain capacity, and communication data must be sent such that these capacities are not exceeded. Several models supporting different capacity structures and flow schemes were developed and applied in planning of real-world networks, see for example [PW92, SD94, BG96].

An important part of a network's quality is robustness against failure of equipment. As network models came into use which reflected the real capacity structures very close, the focus of interest shifted onto survivability and routing questions. Today's models integrate survivability methods of different types, the possibilities and limitations of particular routing protocols, and capacity dimensioning. These models are of great variety. There are simple improvements to generic static routings, like flow diversification, where data is sent on multiple paths to keep some flow if a few of these paths are interrupted. The other end is marked by highly specialized models for particular routing protocols that closely support protocol-specific survivability schemes. Such models can be found e.g. in [AGW97, Wes00], an example for a protocol-based model in [BK02].

In general, the models that were developed over time reflect communication networks very close. They ensure that mathematical solutions produced by implementations of these models can be put into practice, while only minimal changes due to model inaccuracies are required. However, current models fully neglect the actual hardware that is needed in real networks. This hardware has three major impacts on network planning. First, it plays a significant role in a network's cost. Second, not all possible capacity assignments can be realized using available hardware equipment. Third, routing communication demands must respect that some hardware elements might impose constraints on the flow that passes through them. Hence, incorporating hardware configuration into network optimization seems the next step in order to bring models closer to reality. This is where this work is located. To our knowledge, this is the first attempt to develop sophisticated models and according algorithms that does not rely on heuristics.

We propose two models. The first is a novel approach to deal with general network optimization problems. It models placement of generic items called *components* at nodes and links of a network. Components can produce and consume certain *resources* on installation. An installation of components is feasible if resource consumption does not exceed production.

The second model is in fact an application of the first one. We introduce generic hardware and capacity items with specific properties. These items are modelled using according components and resources. It is shown that this setup covers a great amount of hardware configurations.

This work is organized as follows: Chapter 1 introduces terminology and basic questions and problems for telecommunication networks. Because of the large variety of applications, we resort to briefly describing three different planning scenarios, which serve as test bed for our models. In Chapter 2, a short overview on notations and prior results on which this work is based is given. Chapter 3 introduces both models in detail and presents some extensions. The polyhedra associated with the hardware configuration model is further examined in Chapter 4, where valid and facet-defining inequalities are covered. In Chapter 5, some light is shed on the algorithmic side of our models. Heuristics for pre- and postprocessing and a branch-and-cut framework are introduced briefly. Finally, Chapter 6 shows how the model is applied on the planning scenarios from the beginning. We report on computational results using our implementation of the models, thereby proving that they can be successully put into practice.

Chapter 1

Planning Scenarios

In this chapter, we introduce basic terminology that is used when dealing with networks, especially telecommunication networks. First, some planning tasks and design questions that arise naturally are discussed. Then, three different planning scenarios are presented. They later serve as test bed for the applicability of the models and algorithms that are presented in Chapters 3 and 5.

A telecommunication network consists of *locations* (or *nodes*) and *links* between them. Nodes are switching centers, where links get connected to each other using specific hardware devices. Links provide capacity available for data transmission between their end-nodes. There are many possibilities what a link might actually be – a physical cable, a rented path through some other network, a microwave connection, or anything else that fits the definition of end-to-end capacity provision.

Networks are usually organized as hierarchical *layers* (see Figure 1.1), where particular sub-networks can be seen as complete networks by themselves. The lowest hierarchy level consists of *access networks*, where customers get access to the network. Such networks span a relatively small area only. These networks are connected to higher-order layers. The top-most layer contains the *backbone* network. Usually, it covers the entire area of all lower layers. All networks in layers above access network level are called *transport* networks. Inter-layer links are usually called *tributary* links. By ignoring capacities on the links, but rather looking at where links are operated and which nodes are not directly connected, the *topology* of a network is determined. There are some typical topologies appearing in telecommunication networks: First, a *ring* structure, as shown in the intermediate and access layers in Figure 1.1, and a *mesh*, that is used in the figure's backbone network.

A network's ultimate purpose is to transport data. Such data usually enters the network at some node in the access layer (the *source* node), gets passed over links and nodes, possibly over

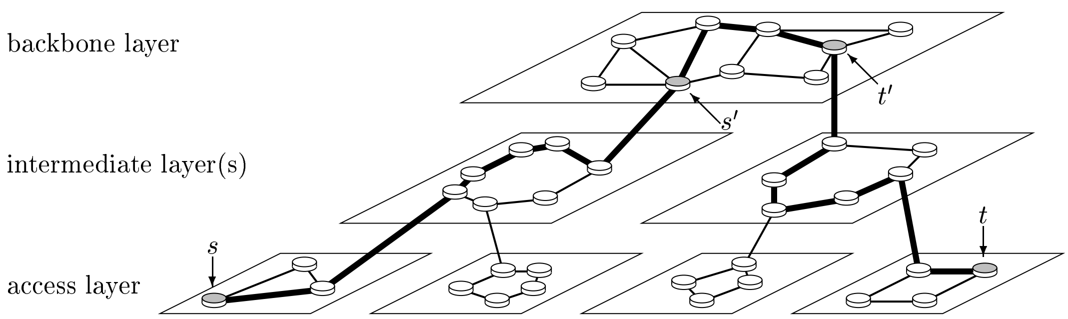


Figure 1.1: Layered network. *Networks are organized in layers containing subnets. The thicker lines form a path between nodes s and t, which actually is a path between s' and t' when focusing the backbone network only.*

multiple layers, and eventually leaves the network at some other node (the *target* node). Figure 1.1 shows a path between two nodes, s and t, through the network.

A telecommunication company operates a network to provide some *services*. A service involves sending particular types of data flow through the network by some means. In today's mobile phone networks there usually are multiple types of services, each with characteristic properties:

- First, there may be voice transmission. Here, a point-to-point data flow with a fixed bandwidth for some time is used. Transmission time, and thus lengths of flow paths, is important because spoken dialogues are very sensible to time delays.
- Second, there may be a messaging service like "SMS" or "MMS", where one client sends a particular piece of data (possibly a large amount, if it happens to be a chunk of multimedia content, like a video) to another client. Again, some point-to-point bandwidth is used, but transmission delays are no serious issue.
- Third, there may be internet access. Here, no data flow between customers appears, but all data is rather exchanged between customers and one or more gateway nodes.
- Fourth, there may be network-internal traffic, used by the network equipment itself to propagate failures, configuration changes and such. Usually, this does not involve large amounts of data. However, since transmitted information is likely crucial for network operation, it is priorized over all other services.

Offering a service requires estimating traffic in form of *demands*. For a pair of nodes, a demand describes the amount of data flow that has to be routed through the network. This amount can reflect many different requirements: It may simply be a bandwidth that is rented to a client as a dedicated virtual line. For telephony services, it can be the lowest bandwidth needed such that, given typical client behaviour and communication needs, an attempt for a call between the areas served by two nodes will only fail due to bandwidth exhaustion with a certain predefined stochastic probability. Demands that belong to the same service are grouped to form *demand sets*. These sets are often referred to as *demand matrices* as well, for a set consisting of demand values for every pair of nodes is often written as a matrix.

A demand set is fulfilled by a *routing*, which assigns bandwidth to each demand such that its traffic can be sent according to the demand's requirements. Obviously it is advisable that capacities and hardware are sufficient to process traffic and routings of all services at once.

Frequently, routings for services or demands are additionally subject to *survivability* requirements of some type. A network is not failure-safe, and there are many reasons why a particular part of the network might suddenly go "out of service", e.g., accidental cable cuts at construction sites, hardware struck by lightnings, or hardware simply breaking down. Since customers do not tend to be very forgiving when a service they paid for is abruptly discontinued, it has to be ensured that at least a reduced amount of traffic survives until full network operation is restored. There are many ways to define "reduced amount of traffic" and "survive", see for example [Wes00, AGW97].

An issue that has to be considered is orientation: A link might be *directed*, offering it's capacity for flow in one direction only, *bidirected*, behaving as two antiparallel directed links, or *undirected*, where capacity is shared between data flows in both directions.

A substantial part of the cost for operating a telecommunication network consists of installation and maintenance of the infrastructure. Hence, careful planning of these networks is of importance.

The process of planning a network consists of many different tasks. Some of them are:

1. Defining services and service classes and their parameters.
2. Estimating demands ("traffic forecasting").
3. Choosing positions of nodes ("location planning").
4. Determining costs of available technologies and hardware based on some planning horizon.

5. Assigning nodes to layers.
6. Choosing a topology.
7. Assigning capacities to the links.
8. Determining routings for demands.
9. Choosing placement and configuration of hardware.
10. Splitting up assigned capacities, providing each service a reserved bandwidth.

These tasks are usually dealt with separately or in groups. It is obvious that solutions to one task might influence the possibilities for another one. If, for example, a planner responsible for task 6 decides not to connect any nodes at all, a subsequent attempt for task 8 will fail. Besides the possibility that earlier decisions render following jobs completely impossible, it may happen that existing good solutions for the overall planning task are not implemented because of limited foresight in early decisions.

On the other hand, planning problems might become simpler by what earlier decisions provide. Figure 1.1 shows how task 8 changes when layer assignment and topology choice is already done: Instead of looking for a path between s and t, it is possible to resort to a path between s' and t' for the backbone network.

Demands change over time, so earlier decisions have to be revised frequently, and the network has to be extended and reconfigured to meet new requirements. Thus, when dealing with above tasks, one has to take the existing network and configuration into account. This may impose various limitations on further decisions, like partially fixed routings or presence of capacities and hardware in places where they are not needed anymore.

This thesis is positioned along solving tasks 6 to 10, which we believe to be a useful choice of planning tasks that should be considered together. To motivate our models and algorithms as presented in later chapters, we now present a set of three "planning scenarios", which are of practical relevance and serve as test bed for the applicability of the presented ideas.

1.1 SDH Network

The first scenario deals with operating a self-owned multi-service SDH network. The synchronous digital hierarchy (SDH) is standardized in a series of recommendations by the International Telecommunication Union, see for example [ITU97] for capacity structure standards, and [ITU95] for survivability concepts.

Capacities

In this scenario, links consist of optical fibers. Over them, data packets called *STM–1 frames* are sent. Here, "STM" stands for *synchronous transport module*. A fiber configuration that can transmit a certain defined number of such frames per second is said to have a capacity of "STM–1". Certain other transmission speeds are defined as well, where a STM–N link carries N times the frames of STM–1. See Figure 1.2 for capacities up to STM–16. (STM–64 exists as well, and STM–256 is currently in the standardization process.)

The payload of the data streams can be split up according to certain rules, leading to *containers* as virtual capacities for application use. For example, a STM–1 always carries some data fragment, which is filled with either one "C–4" container or three smaller fragments. Each of those can either be used for one "C–3" container, or split up to contain seven smaller units. This however involves many different units. Mapping a C–3 container into a STM-1 signal can involve up to seven multiplex steps for different types and sizes of containers:

$$\boxed{\text{C–3}} \to \boxed{\text{VC–3}} \to \boxed{\text{TU–3}} \to \boxed{\text{TUG–3}} \to \boxed{\text{VC–4}} \to \boxed{\text{AU–4}} \to \boxed{\text{AUG–1}} \to \boxed{\text{STM–1}}$$

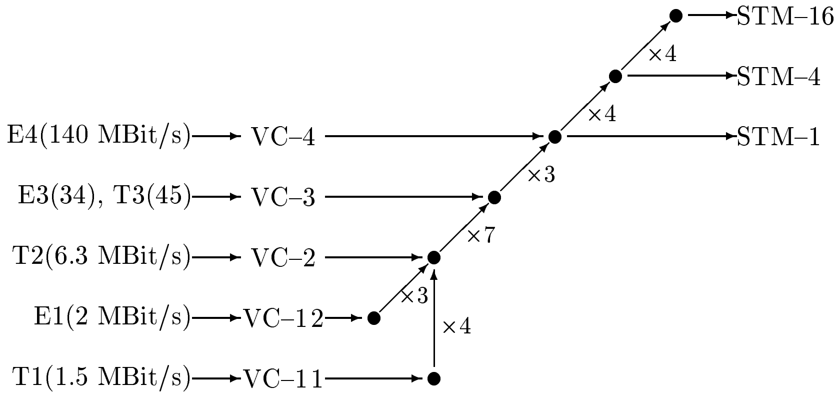


Figure 1.2: SDH hierarchy. *Mapping of PDH data streams into SDH containers VC–N, which are then multiplexed into capacities STM–N. (STM–64 and STM–256 exist as well)*

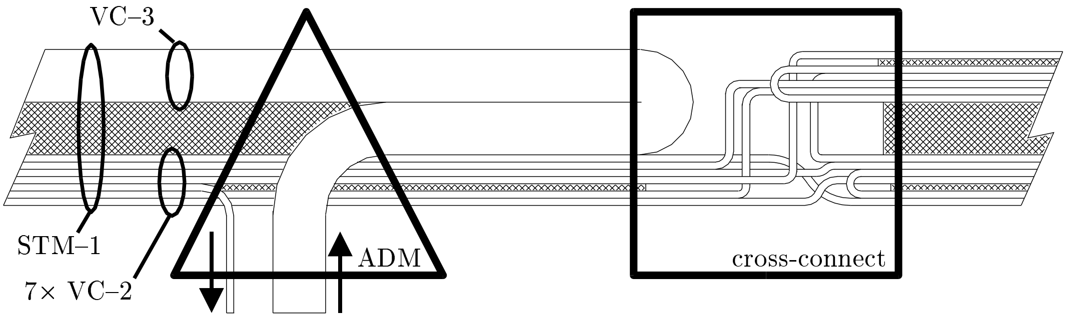


Figure 1.3: Functional SDH overview. *Left to right: STM–1 link containing an used VC–3, an empty VC–3 and seven VC–2 containers is attached to an add-drop-multiplexer ("ADM"), which extracts a tributary VC–2 ("drop") and inserts a tributary VC–3 ("add"). A cross-connect is able for more complex connections between attached links (here: two STM–1 links, but more are possible).*

Each step in this series and each of the intermediate units is precisely defined in the according standards. Because of the complex structure, it is common to simplify the container hierarchy to STM–N capacities and VC–N containers and ignore all other units. Then, each STM–N capacity carries N VC–4 containers, which can be split up into smaller VC–N containers.

Before introduction of the SDH, many voice networks were based on the PDH standard, which similarily defines certain transmission capacities, routing protocols and technologies. Unfortunately, PDH suffers from two major problems: First, accessing a small capacity embedded in larger ones involves breaking up the signal by a series of slow operations. Second, there are two different and incompatible sets of capacities in use worldwide, which leads to many problems at intercontinental handover locations. SDH was designed with these two issues in mind. Additional to addressing both problems, a simple transition from PDH to SDH was made possible by using capacities that may carry PDH signals without too much overhead. Using a container's payload for transmission of data that is no PDH stream is however possible.

Although links are directed, i.e., data is sent in one direction only, fibers usually are installed as antiparallel pairs, and thus can be seen as bidirected capacities, given that network elements and router cards are obtained pair-wise as well.

Hardware

Mandatory for a working node is at least one *multiplexer*, which receives incoming data and passes

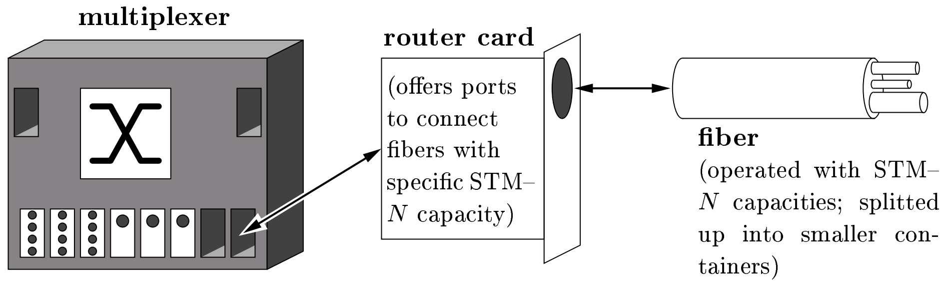


Figure 1.4: SDH equipment. *From left to right: Multiplexer with ten card slots, six of which are already occupied; router card that offers one port; fiber whose capacity is split into three containers.*

it to outgoing links according to some routing rules. Multiplexers usually are either ADMs ("add-drop multiplexers") or DXC's ("digital cross-connects"). The first are designed to be part of a ring structure, inserting small data streams into, or extracting them out of the main flow in the attached ring. The latter are supposed to connect two rings, allowing for more sophisticated data exchange between the rings. There is however no technical reason to organize the network as a collection of rings. The design principles of ADMs and DXCs are shown in Figure 1.3.

Both types of multiplexers share some technical limitations:

- They cannot handle infinite amounts of traffic. There are upper bounds on the total capacity that can be connected to them.
- There is a limit on the smallest possible container that can be extracted from higher-order data: A DXC designed for being a core router in a large backbone network might be able to extract and switch units only of a relatively high bandwidth. It is thus required to multiplex many small containers into larger units at network entry and to route them together to a destination where they get demultiplexed.
- The forwarding table, consisting of rules which container to extract from what input signal for forwarding on which output signal, has a limited size.

Both multiplexer types are of no use by themselves, as they have no direct connection to the fibers. For that matter, they have to be equipped with *router cards*, which contain the hardware required to transmit and receive signals over the fibers. Cards may provide ports for more than one fiber at once. Typically, there is a specific card for each supported SDH capacity: A "Siemens TransXpress SLD16" for example has 10 slots of two different types. There are cards available for STM–1, STM–4 and STM–16 links, each occupying one slot. While the STM–1 card comes with four ports, the other two support just one link each.

Each multiplexer has a certain number of *slots* – possibly of different types – available for card insertion. Usually, these slots are of different types, e.g., there may be two special slots into which router cards get installed that connect to a ring. Additional to slot limitations, there may be compatibility issues and other reasons for limiting the number of cards of a particular type that can be installed into a multiplexer. Figure 1.4 provides an overview over routers, router cards, links and their interconnections.

Routing

Communication demands in a SDH network are based on containers: There may be multiple demands between two nodes, where each requires a different container size for routing. They are typically aggregated from lower-order signals at the outside, and disaggregated when they leave the network at their destination. In a multi-service network, demand sets are parallel: Different services are be offered, each of which comes with different routing requirements, and each containing a demand for every pair of locations. Services might be voice traffic or a virtual private network rented to some other company.

Because a multiplexer has a smallest container size that it can deal with, a routing must respect that containers can be aggregated and sent over a common path, but need specific multiplexers where they join or split.

Although each service can theoretically require another routing protocol, we limit this scenario such that each demand is routed using point-to-point paths, possibly respecting survivability requirements. There are survivability schemes that are directly supported by the SDH hardware, which can be used for the routing. An example is "1+1 protection": The routing consists of two disjoint paths for each demand. At the source, the signal is duplicated and sent over both simultaneously. The multiplexer at the destination automatically picks the incoming signal of better quality, thereby preventing signal loss if equipment on one of the two paths fails.

As the traffic for each demand enters and leaves the network in some predefined containers, demands are determined by this container as well as the number of containers that have to be routed.

Cost Structure

In order to operate a link, one or more optical fibers have to be used. Given a fiber pair of reasonable quality, the available capacities depend on the hardware at the nodes only. The cost of obtaining a fiber pair depends on physical surroundings. The fiber itself is not very expensive, and placing it into some already existing duct results in moderate cost. Renting an existing fiber from another company can be cheaper. On the other hand, digging a new canal to place fibers into is extremely expensive. There are real-world situations where the average equipment for a backbone node is worth less than 3 kilometers of fiber.

1.2 Optical Network

In the second scenario, the network is again completely owned by the telecommunication company. Now, the placement of fiberoptic cables, choice of light wavelengths to form lightpaths, and allocation of associated optical and electrical equipment is discussed.

Capacities

Each link consists of one or more *optical fibers*, over which data is sent optically: *Transmitters* emit light of specific wavelengths that is sent to *receivers*. Data can be sent at different bitrates, resulting in different bandwidths. Here, a stream using a determined bitrate and wavelength is also called *channel*.

Light can however not travel through fibers without loosing brightness and signal sharpness. The distances after which it is still possible to extract the full original information varies, depending on the quality of the fiberoptic, the transmitters and receivers, the initial signal quality, and many other actualities.

Because it is possible to send multiple wavelengths over a single fiber, the total bandwidth that can be operated over a fiber depends not only on bitrates, but on wavelength choices, ensuring that interferences are small enough to still extract all contained signals. This technique is called *Wavelength Division Multiplexing* ("WDM").

Light transmission provides a means for a directed data stream only. It is however possible to use channels of opposite direction using different wavelengths, as is using a pair of fibers, one for each direction. This makes it possible to build directed, undirected and bidirected links.

The total capacity resp. bandwidth of a link is thus roughly

$$\text{number of fibers} \times \text{channels per fiber} \times \text{bitrate per channel}.$$

Notice that there is no need that all fibers of a link provide the same bitrate or number of channels.

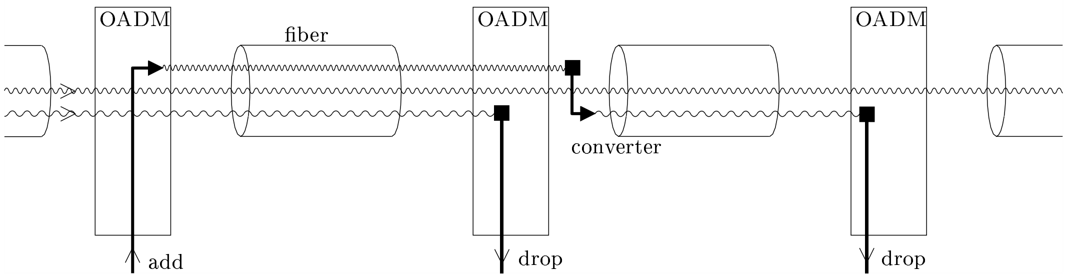


Figure 1.5: Optical network. *Some devices that appear in optical networks: OADMs extract and insert specific channels, converters change a signal's wavelength, and fibers transport light beams. The multiplexers that merge the three wavelengths into a single beam and the according demultiplexers are left out for simplicity.*

Hardware

Basically, hardware can be grouped to two categories: Items that deal with physical light transmission and switching hardware that enables a routing. Figure 1.5 gives an overview over some of the equipment.

Transmitters are lasers that produce a light beam of some wavelength and bitrate that carries a data stream. Receivers decode the signal and pass on the contained data as electrical information. *Regenerators* attempt to improve signal quality. Different types exist whose features range from augmenting amplitude to completely reestablishing a sharp signal shape. Such regenerators are placed at regular distances to prevent signal loss. *Wavelength converters* remodulate the wavelength of a signal to a different one. Usually they completely decode and re-encode the data stream and are thus regenerators as well. *Multiplexers* and *demultiplexers* finally perform the tasks of joining signals of different wavelengths to a single beam for fiber transmission and separating them afterwards. Instead of dealing with all these hardware pieces, ready-made systems are used. They are called *WDM systems* and offer a specific set of usable wavelengths.

Each fiber on each link is operated by some WDM system. The channels can theoretically be operated at different bitrates, for this scenario it is however assumed that a single one is chosen for the entire network. A certain set of fibers as well as WDM systems may be already present.

Similar to DXC and ADM as switching hardware in Scenario 1.1, optical pendants are used here. *Optical cross-connects* ("OXC") can by themselves forward a certain number of incoming channels to outgoing channels, without changing wavelength. This forwarding works fully optical, that is, the optical signal is not converted to digital data in the forwarding process. Multiple types of OXC are available, with different numbers of supported channels. *Optical add-drop-multiplexers* ("OADM") are similar, with the exception that there is a signal that passes the OADM mostly unchanged, and from which only a few signals can be extracted or added to it. Wavelength converters are usually installed directly at the ports of OXCs and OADMs, and thus can be seen as part of the node switching hardware.

Each node may already hold some particular switching devices and converters. Additional hardware may be installed to expand nodes.

From a technical point of view, most of the above hardware types have two important sub-types: Electrical or partly electrical devices that need signal de- and encoding to function properly, and fully optical devices that don't. This difference is however completely irrelevant for this planning scenario.

Routing

The network allows to construct logical channels over multiple links: On each link of a path a specific wavelength is assigned, and each OXC and OADM at the nodes in the path is configured to forward these wavelengths to the correct successor, using a wavelength converter if required.

The resulting series of channels forms effectively a single virtual channel over the path. Such a construction is called *lightpath*.

Demands specify a number of lightpaths that are to be established between the demand's end-nodes. Each such lightpath can then be used for whatever data stream a service produces.

A routing has to properly assign sufficient lightpaths to the demands. Of course physical conditions have to be considered as well, e.g., placement of regenerators at regular intervals, and taking into account that all equipment on the path supports the wavelengths that are used.

Survivability requirements may be present, and have to be fulfilled by the routing. Optical switching devices can have very long reconfiguration delays, up to the time needed to manually exchange lasers. Until OXCs that can dynamically change the routing in very short time are available and widely in use, protection schemes that use a static routing are preferred, like 1+1 protection (see Scenario 1.1).

Cost Structure

As fibers are self-owned in this scenario and have to be physically laid into ground, new fibers are very expensive. Since installation of more than one fiber into a single duct is possible, fiber cost does not increase linear with amount, but economies of scale apply.

All pieces of hardware – OXCs, OADMs, converters, regenerators, and WDM systems – cost a specific amount of money.

The capacity of an existing fiber can be increased by upgrading the equipment at the end nodes, unless the fiber is of poor quality. This leads to the situation where a cost-efficient solution usually consists of the fewest possible new fibers, possibly by completely exchanging node hardware.

A telecommunication company that laid a optical network for single-wavelength transmission at the beginning of the internet boom around the year 2001 now owns a network that can theoretically be operated at capacities far beyond any real traffic using these "old" fibers only. In such a situation, the cost of network planning solutions is completely determined by hardware investments.

1.3 OSPF Network

This scenario deals with an IP network that is rented from a provider but is subject to very peculiar long-term contracts between telecommunication company and the provider of capacity and infrastructure.

Most notably, the network is built in predefined expansion steps over a period of multiple years, where the status of each step is used as basis for network reconfigurations in the subsequent step.

Capacities

The network is built on top of an SDH network as introduced in Scenario 1.1. The capacity of the links is completely used to transmit IP packets between its end nodes. Again, all SDH capacities are available. There are however limits on the amount of changes that are allowed in each expansion step. The contract defines for each phase and each available capacity a maximum of links that may be operated using this capacity. Additionally, a maximum on the number of links that may change their configured capacity from the present state is specified.

Hardware

On each node, SDH switching hardware is required for the low-level SDH transmissions. This hardware is accompanied by an IP router. This router is, similar to SDH multiplexers, equipped with router cards that connect to the cabling.

The fibers connect to the SDH cards of the SDH hardware. Each SDH card is in turn connected to an IP router card of the IP router. Together, the IP router provides the switching based on IP routing protocols, and the SDH router provides the transmission of the packets to their correct destination.

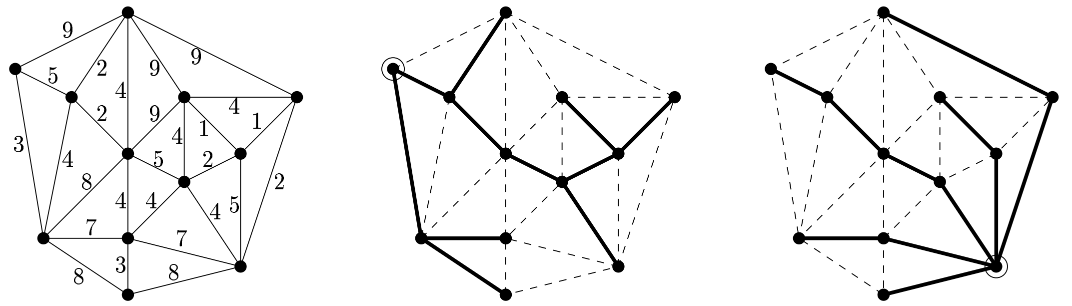


Figure 1.6: OSPF routing. *The leftmost picture shows a network and assign link weights. The other pictures show the two sink-trees that comprise the routing to the two marked nodes. These two exchange data on a common path.*

Again, the contract defines limits on configuration changes for each expansion step. The routers themselves are fully fixed without any configuration choice. The cards on the other hand may be changed, but only according to certain rules: For each type of card, be it an IP or a SDH router card, values are given for the maximum number of cards that may be removed, added, or transferred to different nodes. These limits ensure that updates stay within a cost budget, and can be implemented with the available resources.

Routing
In each expansion step, the predictions for traffic demands are updated. The traffic is routed according to the OSPF protocol. This protocol works by sending all traffic along a shortest path through the network. Shortest paths are determined according to link weights that can be assigned by configuring weight values at the IP routers. Shortest paths should be unique for all source/target pairs to ensure that capacities suffice for the traffic, which is assured by an adept choice of link weights. This routing principle carries the intrinsic fact that all traffic is routed along directed sink trees, see Figure 1.6 for an example. Due to these facts, the planning result are not the routing paths themselves, but rather the link weights that lead to this routing.

Both traffic and capacity is directed by nature in this scenario. Because link weights are the same for both directions of a link, the routing is always symmetrical, i.e., the backwards channel of each routing follows the same path as the forward one. So, an undirected approach can be used here again.

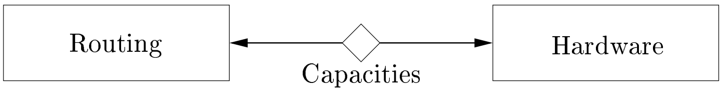
Cost Structure
There is no direct relationship between the configuration in some expansion step and investments of any kind. The underlying contract gives fixed rental cost for the whole network independently of the actual state. The only equipment that has to be bought are the router cards.

In this scenario, the update limitations on capacity types and router cards play the role of cost: They have a major influence on planning decisions and are key to successful contract compliance.

1.4 Conclusions

Notice the similarities in the preceeding scenarios: There are some demands that need to be served by choosing a network topology and link capacities, which need to be large enough to allow for one or more routings. Hardware, on the other hand, limits capacity: It is not generally possible to go for a dense topology, nor does a configuration of hardware necessarily exist when topology and capacity decisions are made.

Consequently, we see “capacity” as binding link between routings and hardware:



An overwhelming amount of work was put into exploring the routing/capacity relationship for various different routing and capacity models. However, little is known about the dependencies between capacity dimensioning and hardware configuration. The only relevant publication we are aware of is [BS01], where placement of ATM switches with a limited switching capacity was considered. This motivates this thesis.

We will concentrate mostly on capacities and hardware, leaving routing issues aside. This does however not mean that they are completely ignored: In Chapter 5, we will show how to integrate both parts into a single framework, such that both routing and hardware decisions are made at once.

Chapter 2

Preliminaries

In this chapter, a few basics are covered briefly. First, some notations are introduced to avoid ambiguities. Then, a very short overview to graph theory is given. The final part comprises the concepts of demands and routings, with some basic results.

In this work, $\mathbb{R}$ denotes the set of real numbers. $\mathbb{Z}$ is the set of integral numbers, positive or negative. $\mathbb{R}_+$ and $\mathbb{Z}_+$ are the sets of nonnegative reals resp. integers. $\overline{\mathbb{Z}}_+$ contains the nonnegative integers and infinity, i.e., $\overline{\mathbb{Z}}_+ := \mathbb{Z}_+ \cup \{+\infty\}$. $\mathbb{B} := \{0,1\}$ is the set containing the two binary digits. For a set M, 2^M is the family of all subsets of M, including $\varnothing$ and M itself. For $n \in \mathbb{Z}_+$, $n \geqslant 1$, $\mathbb{K}^n$ is the space of column vectors with n coefficients, which are elements of $\mathbb{K}$. The row vector of $x \in \mathbb{K}^n$ is denoted by x^T. $\mathbb{K}^{n \times m}$ for $n, m \in \mathbb{Z}_+$, $n \geqslant 1$, $m \geqslant 1$, denotes the set of matrices with n rows and m columns and coefficients in $\mathbb{K}$.

For a finite set N, $\mathbb{K}^N$ equals $\mathbb{K}^{|N|}$ using an implicit index mapping. Coefficients of vectors $x \in \mathbb{K}^N$ are denoted as if N contains indices, i.e., $x = (x_i)_{i \in M}$. For two finite sets M and N, $\mathbb{K}^{N \times M}$ denotes a vector space of dimension $|N||M|$, hence $\mathbb{K}^{N \times M} \simeq \mathbb{K}^{|N \times M|}$, while the set of matrices with $|N|$ rows and $|M|$ columns is denoted by $\mathbb{K}^{|N| \times |M|}$.

2.1 Graphs

An *undirected finite graph* G consists of a pair (V, E) of sets and an incidence function $\psi : E \to V^{(2)}$, where $V^{(2)}$ is the family of all subsets of V with one or two elements, and V and E are finite. For this work, we consider only such graphs and call them simply *graph*.

Elements of V are called nodes (other common names being vertices and points), and elements of E are called edges (also known as links or lines). The existence of ψ is often neglected and $G = (V, E)$ is written. An edge $e \in E$ with $\psi(e) = \{u,v\}$ is simply denoted by $e = uv$. In case two edges $e_1, e_2 \in E$ are different edges ($e_1 \neq e_2$) but $\psi(e_1) = \psi(e_2) = \{u,v\}$ both edges are denoted uv despite being different. In this case, e_1 and e_2 are called *parallel edges*. On the other hand, uv and vu may denote the same edge because $\{u,v\} = \{v,u\}$. An edge e with $|\psi(e)| = 1$ is called loop. If G contains neither loops nor parallel edges, it is called *simple*.

Two nodes $u, v \in V$ are said to be *adjacent*, if they are end-nodes of at least one common edge. For a set $U \subseteq V$ of nodes, the set of all nodes that are adjacent to at least one $u \in U$ is denoted by $\Gamma(U)$. For a singleton $\{u\} \subseteq V$, $\Gamma(\{u\})$ is written $\Gamma(u)$. Nodes in $\Gamma(u)$ are called *neighbors* of u. Notice that U and $\Gamma(U)$ are not necessarily disjoint, and even $u \in \Gamma(u)$ is possible if the loop $uu \in E$.

A node $v \in V$ and edge $e \in E$ are said to be *incident*, if v is end-node of e. For a node $v \in V$, the set of all edges that are incident with v is denoted by $\delta(v)$. For a set of nodes $U \in V$, $\delta(U)$ is the set of all edges that are incident to both a node in U and $V \backslash U$. Notice that $\delta(v) = \delta(\{v\})$ does not hold if the loop $vv \in E$. An edge set $F \subseteq E$ is called *edge cut*, or *cut* in short, if $F = \delta(U)$ for some $U \subseteq V$. In this situation, U and $V \backslash U$ are called *shores* of the cut. Notice that the shores are not necessarily unique for a given cut. For an edge set $F \subseteq E$, the set of nodes incident to some $e \in F$ is denoted by $V(F)$, and for a node set $U \subseteq V$, $E(U)$ denotes the set of edges which are incident to nodes in U only.

A series $p = (v_1, e_1, v_2, e_2, \ldots, e_{n-1}, v_n)$ of nodes $v_1, v_2, \ldots, v_n \in V$ and edges $e_1, e_2, \ldots, e_{n-1} \in E$ is called *path*, if $e_i = v_i v_{i+1}$ for $i \in \{1, \ldots, n-1\}$ and $v_i \neq v_j$ for every $i, j \in \{1, \ldots, n\}$ with $i \neq j$. For $U, W \subseteq V$, p is called *$[U, W]$-path* if $v_1 \in U$ and $v_n \in W$, or $v_1 \in W$ and $v_n \in U$. If U and W are the singletons $\{u\}$ and $\{w\}$, the set brackets are omitted and p is called $[u, w]$-path. If p is a path with $n > 1$ and $v_n v_1 \in E$, $(v_1, e_1, v_2, \ldots, v_n, v_n v_1, v_1)$ is called a *circle*. A set $F \subseteq E$ of edges is called *forest*, if (V, F) contains no circle.

For a set $U \subseteq V$ of nodes and a set $F \subseteq E$ of edges, we define $G - F := (V, E \setminus F)$ and $G - U := (V \setminus U, E(V \setminus U))$.

G is called *connected*, *1-node-connected* and *1-edge-connected*, if there exists a $[u, v]$-path in G for every pair of nodes $u, v \in V$, $u \neq v$. For an integer $k > 1$, G is called *k-node-connected*, if $G - \{v\}$ is $(k-1)$-node-connected for every $v \in V$, and G is called *k-edge-connected*, if $G - \{e\}$ is $(k-1)$-edge-connected for every $e \in E$. G is called *biconnected*, if it is 2-node-connected. For $k \geqslant 1$, G is k-node-connected, if for every $u, v \in V$, $u \neq v$, there exist k-many $[u, v]$-paths in G that have only u and v in common. G is k-edge-connected, if for every $u, v \in V$, $u \neq v$, there exist k-many $[u, v]$-paths that share no edges. A connected forest $F \subseteq E$ is called *tree*, and if additionally $V(F) = V$, F is called *spanning tree*.

2.2 Routing

Although most of this work is independent of particular demand and routing models, there usually is the assumption that some demands exist and a routing of some type must be possible. In some places, we will need to refer to certain common properties of them. It is therefore advisable to introduce simplified problems and models that reveal important aspects which are still valid for larger and more sophisticated models. We introduce a basic model for undirected demands without any survivability conditions on routings based on capacitated network flows.

Let $G = (V, E)$ be a graph, and y an assignment of capacity values for the edges: $y : E \to \mathbb{Z}_+$. A *demand set k* consists of a loop-free graph $(V, \mathfrak{D}_k)$ using the same nodes as G, and an integer $\mathfrak{d}_{k,uv} \in \mathbb{Z}_+$ for each $uv \in \mathfrak{D}_k$. $(V, \mathfrak{D}_k)$ may contain parallel edges. In this case, the notation introduced here is ambiguous. Because it is widely adopted and correct interpretation is simple, we use it anyway.

The edges $uv \in \mathfrak{D}_k$ are called *demands* and the values $\mathfrak{d}_{k,uv}$ are called *demand values*. The latter specify a flow value that has to be established between u and v to satisfy the demand. For two node sets $U, W \subseteq V$, let

$$\mathfrak{D}_k(U, W) := \{uv \in \mathfrak{D}_k \mid (u \in U \wedge v \in W) \text{ or } (v \in U \wedge u \in W)\}$$

denote the set of demands with one end node in U and the other one in W. Notice that U and W are not required to be disjoint or even different. Let

$$\mathfrak{d}_k(U, W) := \sum_{uv \in \mathfrak{D}_k(U, W)} \mathfrak{d}_{k,uv}$$

denote the total value of the demands in $\mathfrak{D}_k(U, W)$. If either U or W is a singleton, the set brackets are sometimes omitted to simplify notation. Hence, for two nodes $u, w \in V$, the demand $\mathfrak{d}_k(\{u\}, \{v\})$ between them is denoted by $\mathfrak{d}_k(u, v)$ in short. For a node $v \in V$, the value $\mathfrak{d}_k(v) := \mathfrak{d}_k(v, V)$ is called *emanating demand* of v.

For each $uv \in \mathfrak{D}_k$, let $\mathcal{P}_{uv}$ be the set of all $[u, v]$-paths in G. A *routing* is a mapping that assigns each demand $uv \in \mathfrak{D}_k$ a set of $[u, v]$-paths and a positive flow value $f_{uv}(P)$ for each path P such that the flow values for each demand sum up to $\mathfrak{d}_{k,uv}$, while the edge capacities suffice for all path flows. One way to model the routing problem is based on linear programming. Consider the following problem:

$$
\begin{aligned}
\min \quad & \alpha \\
\text{s.t.} \quad & \sum_{P \in \mathcal{P}_{uv}} f_{uv}(P) = \mathfrak{d}_{k,uv} && \forall uv \in \mathfrak{D}_k \ , \\
& \sum_{uv \in \mathfrak{D}_k} \sum_{P \in \mathcal{P}_{uv} : e \in P} f_{uv}(P) \leqslant y(e) + \alpha && \forall e \in E \ , \\
& f_{uv}(P) \geqslant 0 && \forall uv \in \mathfrak{D}_k,\ P \in \mathcal{P}_{uv} \ .
\end{aligned}
\tag{2.1}
$$

Obviously, a feasible routing exists if and only if there exists a solution for (2.1) with $\alpha \leqslant 0$, which in turn is equivalent to the optimal objective value being nonpositive. There may be the additional requirement that the routing is integral, i.e., $f_{uv}(P) \in \mathbb{Z}_+$ for all $uv \in \mathfrak{D}_k$, $P \in \mathcal{P}_{uv}$, in which case (2.1) is a relaxation of the routing problem. However, if fractional routing paths are feasible, (2.1) is a complete formulation. Notice that (2.1) contains one variable for each path in G if the demand graph $(V, \mathfrak{D}_k)$ is complete. The dual of (2.1) is

$$
\begin{aligned}
\max \quad & \sum_{uv \in \mathfrak{D}_k} \mathfrak{d}_{k,uv} \pi_{uv} - \sum_{e \in E} y(e) \mu_e \\
\text{s.t.} \quad & \sum_{e \in P} \mu_e \geqslant \pi_{uv} && \forall uv \in \mathfrak{D}_k,\ P \in \mathcal{P}_{uv} \ , \\
& \sum_{e \in E} \mu_e = 1 \ , \\
& \mu_e \geqslant 0 && \forall e \in E \ .
\end{aligned}
\tag{2.2}
$$

Because of LP duality every solution of (2.2) defines a lower bound for α, and hence the condition $\alpha \leqslant 0$ for a feasible routing translates into a *metric inequality* from the objective of (2.2).

Theorem 2.1 *Let $y : E \to \mathbb{Z}_+$ be a capacity mapping. Let α^* be the optimal objective for* (2.1)*.*

(i) If $\alpha^* \leqslant 0$, the given capacities are sufficient to allow for a feasible demand routing. The path variables in the optimal solution for (2.1) *define such a routing.*

(ii) If $\alpha^* > 0$, the given capacities are not sufficient for a routing. However, there does exist an infeasible routing that exceeds the capacity by no more than α^* on any edge; such a routing is described by the path variables.

(iii) If (π, μ) is a feasible solution for (2.2)*, the objective function of* (2.2) *gives the inequality*

$$
\sum_{e \in E} \mu_e y(e) \geqslant \sum_{uv \in \mathfrak{D}_k} \mathfrak{d}_{k,uv} \pi_{uv} \ , \tag{2.3}
$$

which is valid for every capacity mapping y that allows a feasible demand routing. If $\alpha^* > 0$, inequality (2.3) *with optimal solution values for π and μ is still valid for every feasible capacity mapping, but is not fulfilled by y.*

This result is due to Iri, Onaga, and Kakusho [Iri71, OK71]. There are models that extend (2.1) with further constraints, e.g. survivability requirements, while maintaining properties similar to the above theorem. A lot of work has been put into actually solving (2.1) despite its exponential size. Both can be found for example in [Wes00]. However, for this work the facts presented above are sufficient.

Another formulation of the routing problem uses edge-based flows. Let $\mathfrak{D}_{k_1} \dots \mathfrak{D}_{k_n}$ be n demand sets. Let there be an arbitrary orientation for the demands, i.e., mappings s and t that assign each demand a source and a target:

$$
s, t : \bigcup_{i=1}^{n} \mathfrak{D}_{k_i} \to V \quad \text{such that} \quad \forall i \in \{1, \dots, n\}\ ,\ uv \in \mathfrak{D}_{k_i} : \{s(uv), t(uv)\} = \{u, v\} \ .
$$

For each edge $e = uv \in E$ and demand set k_i, two flow variables $f^{k_i}(u,v)$ and $f^{k_i}(v,u)$ are used. They model the flow for k_i on e for the two directions in the following formulation:

$$\sum_{uv\in\delta(v)} f^{k_i}(v,u) - f^{k_i}(u,v) = \sum_{\substack{uw\in\mathfrak{D}_{k_i}:\\ s(uw)=v}} \mathfrak{d}_{k_i,uw} - \sum_{\substack{uw\in\mathfrak{D}_{k_i}:\\ t(uw)=v}} \mathfrak{d}_{k_i,uw} \qquad \begin{array}{l}\forall v \in V,\\ i \in \{1,\dots,n\},\end{array} \tag{2.4}$$

$$\sum_{i=1}^{n} f^{k_i}(u,v) + f^{k_i}(v,u) \leqslant y(e) \qquad \forall e \in E \quad , \tag{2.5}$$

$$f^{k_i}(u,v) \geqslant 0, \quad f^{k_i}(v,u) \geqslant 0 \qquad \begin{array}{l}\forall uv \in E,\\ i \in \{1,\dots,n\}.\end{array} \tag{2.6}$$

This is a relaxation of the routing problem for the demand sets $\mathfrak{D}_{k_1} \dots \mathfrak{D}_{k_n}$ at once. If the demand sets contain single demands only, or all demands of a demand share a common end-node, and fractional routing is feasible, it is exact. Unlike the path-based formulation, this one is of polynomial size.

Notice that sets of demand sets are closed under partitioning and union. Typical applications use a single demand set $\mathfrak{D}_k$ for (2.1) and a partition into smaller demand sets $\mathfrak{D}_k = \mathfrak{D}_{k_1} \cup \dots \cup \mathfrak{D}_{k_n}$ for the flow formulation. In this context, a demand set $\mathfrak{D}_{k_i}$ is sometimes called *commodity* and said to consist of *aggregated demands*, which is due to the fact that the flow formulation assigns an aggregated flow to all demands of one commodity at once. Consequently, the flow formulation is often referred to as *multi-commodity flow formulation.* Two partitioning schemes for a demand set $\mathfrak{D}_k$ are of particular interest: First, each commodity containing a single demand. This is called *disaggregated commodities.* Second, using one commodity $\mathfrak{D}_v$ per node $v \in V$ containing all demands whose assigned source is v: $\mathfrak{D}_v := \{uw \in \mathfrak{D}_k \mid s(uw) = v\}$. This setup is called *node-aggregated commodities.*

Using the simple demand structure as described above, the two formulations seem equivalent, because the flow formulation with disaggregated commodities is exact. There are however simple means to incorporate additional demand constraints into the path formulation that have no known analog for the latter. Hence, throughout this work we assume that the flow formulation can be a relaxation only, but some path-based formulation exists that fulfills properties similar to Theorem 2.1.

Chapter 3

Models

We now present the models we developed for an integrated formulation of hardware and capacities. First, an abstract model for many types of network optimization problems is presented in Section 3.1. It is however too generic to be used as is, so afterwards a specialized version of the model is presented in Section 3.2, which is still flexible enough to suit the needs of the scenarios presented before. The chapter is finished with presenting three extensions to adapt the model to additional surroundings.

3.1 Component-Resource Model

In this section, we introduce the Component-Resource ("CR") model as an abstract model. It is based on two concepts:

Components are things that can be installed somewhere on a network. There are limitations on where a component might be installed, like "only on nodes" or "only on one particular edge". By default, components are available in infinite amounts, and can be installed in batches of one, as is with hardware, where a particular hardware type can be bought in any amount and then placed at certain locations.

The base network gives some special types of components: Each node, each edge and the network itself are components as well, available just once and placed always at themselves. This way, the elements of the network can be used as any other component, and each component installation contains the network as an integral part.

Resources can be provided and consumed by the components. They are aspects of component installations that must not be exhausted by mass consumption, like the initial cost budget available for some network planning problem, which is provided by the network and consumed by installation of components.

Resources have types, defining what places in the network have to be considered together when balancing resources.

As will become clear later, components can be seen as classes of variables and resources as classes of inequalities in some particular integer programs. We however found the new terminology introduced by these concepts to be useful to describe more complex models, and to be interesting by itself. So, we now present the abstract model:

3.1.1 Definition

Let $G = (V, E)$ be a graph as introduced above. We will use the nodes and edges of G, and G itself, as places where components can be installed, so using the set

$$\mathcal{G} = \mathcal{G}(G) := \{G\} \ \dot\cup\ V \ \dot\cup\ E$$

of *graph elements* simplifies the following structures. Second there is a extension set

$$\mathcal{C} \supseteq \mathcal{G}$$

of *components*. Installation of components is bounded by

$$\underline{x} = (\underline{x}_{c,g}) \in \mathbb{Z}_+^{\mathcal{C}\times\mathcal{G}} \text{ and } \overline{x} = (\overline{x}_{c,g}) \in \overline{\mathbb{Z}}_+^{\mathcal{C}\times\mathcal{G}} ,$$

where, for a component $c \in \mathcal{C}$ and a graph element $g \in \mathcal{G}$, c has to be installed at least $\underline{x}_{c,g}$ and no more than $\overline{x}_{c,g}$ times. A graph element, being a component as well, is always installed at itself, and there exactly once. Thus we set

$$\begin{array}{lclcll} \underline{x}_{g,g} & = & \overline{x}_{g,g} & = & 1 & \forall g \in \mathcal{G} \,, \\ \underline{x}_{g,g'} & = & \overline{x}_{g,g'} & = & 0 & \forall g, g' \in \mathcal{G} : \; g \neq g' \,. \end{array}$$

The bounds for other components are just required to fulfill

$$0 \leqslant \underline{x}_{c,g} \leqslant \overline{x}_{c,g} \quad \forall c \in \mathcal{C}, \; g \in \mathcal{G} \,.$$

The above allows to formalize the terms 'installable' and 'installation': A component $c \in \mathcal{C}$ is *installable at graph element* $g \in \mathcal{G}$, if $\overline{x}_{c,g} > 0$. A vector $x = (x_{c,g}) \in \mathbb{Z}^{\mathcal{C}\times\mathcal{G}}$ is called *component installation*, if $\underline{x} \leqslant x \leqslant \overline{x}$.

The second CR concept deals with resources: The index set

$$\mathcal{R}$$

contains these *resources*. Components influence resources as specified by the *resource values*

$$\varrho_{c,g}^r \in \mathbb{R} \quad (c \in \mathcal{C}, g \in \mathcal{G}, r \in \mathcal{R}).$$

We say that component c *provides* resource r on g, if $\varrho_{c,g}^r > 0$, and that it *consumes* it, if $\varrho_{c,g}^r < 0$.

Resources must not be exhausted, i.e., it must be at least as much provided as is consumed. There are many possibilities of how components installed at different graph elements share resources – a total cost budget, for example, exists once, and provision or consumption changes a single excess value. On the other hand, link capacity exists locally: excess of free capacity on one link does not directly influence a deficit on any other link. To model such differences, we introduce *resource types* by partitioning

$$\mathcal{R} = \mathcal{R}^V \,\dot{\cup}\, \mathcal{R}^L \,\dot{\cup}\, \mathcal{R}^G \,,$$

such that each resource $r \in \mathcal{R}$ is either a *node resource* (if $r \in \mathcal{R}^V$), a *local resource* ($\mathcal{R}^L$), or a *global resource* ($\mathcal{R}^G$). Each type specifies sets of graph elements that are considered together in excess calculations. For most applications, using these three types suffices, leading to three *resource inequality* classes, which are called *node resource inequalities* (balancing a node and its incident edges), *local resource inequalities* (a single graph element), and *global resource inequalities* (all graph elements at once), respectively:

$$\sum_{g\in\{v\}\cup\delta(v)} \sum_{c\in\mathcal{C}} \varrho_{c,g}^r x_{c,g} \geqslant 0 \quad \forall v \in V, \; r \in \mathcal{R}^V \,, \tag{3.1}$$

$$\sum_{c\in\mathcal{C}} \varrho_{c,g}^r x_{c,g} \geqslant 0 \quad \forall g \in \mathcal{G}, \; r \in \mathcal{R}^L \,, \tag{3.2}$$

$$\sum_{g\in\mathcal{G}} \sum_{c\in\mathcal{C}} \varrho_{c,g}^r x_{c,g} \geqslant 0 \quad \forall r \in \mathcal{R}^G \,. \tag{3.3}$$

Figure 3.1 shows the graph elements of resource inequalities of different types.

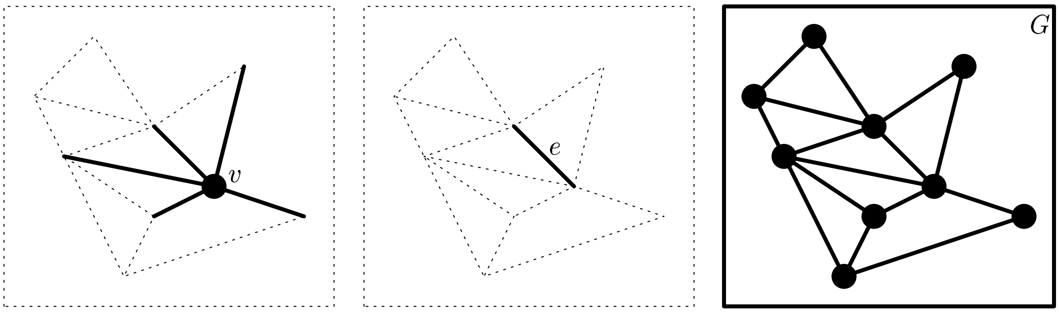


Figure 3.1: Graph elements appearing in resource inequalities. *Left to right: node resource inequality* (3.1) *for node $v \in V$, local resource inequality* (3.2) *for edge $e \in E$, global resource inequality* (3.3) *for the global graph element G (represented by the surrounding box).*

Definition 3.1 (CR integer set X_{CR})
We define X_{CR} to be the set of all feasible component installations, given graph elements, components, resources, and resource values as above, that is

$$\begin{aligned} X_{\mathrm{CR}} &:= X_{\mathrm{CR}}(G, \mathcal{C}, \mathcal{R}^V, \mathcal{R}^L, \mathcal{R}^G, \varrho, \underline{x}, \overline{x}) \\ &:= \left\{ x \in \mathbb{Z}_+^{\mathcal{C} \times \mathcal{G}} \mid \underline{x} \leqslant x \leqslant \overline{x},\ x \text{ satisfies (3.1), (3.2) and (3.3)} \right\} . \end{aligned} \tag{3.4}$$

Notice that this model is basically just a different notation for integer programming sets: Let

$$Y = \{x \in \mathbb{Z}_+^n \mid Ax \geqslant b\}$$

with $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Then Y is equivalent to a CR set using one component per variable and one resource per inequality: Let $M := \{1, \ldots, m\}$ be the set of row indices and $N := \{1, \ldots, n\}$ of column indices of A. Let $G := (\varnothing, \varnothing)$ be the empty graph, thus $\mathcal{G} = \{G\}$. Let $\mathcal{C} := \mathcal{G} \cup N$, $\mathcal{R} := \mathcal{R}^G := M$, $\underline{x}_{G,G} := \overline{x}_{G,G} := 1$, $\underline{x}_{i,G} := 0$ and $\overline{x}_{i,G} := \infty$ for $i \in N$, $\varrho^j_{G,G} := -b_j$ for $j \in M$, and $\varrho^j_{i,G} := A_{j,i}$ for $(j,i) \in M \times N$. Then $X_{\mathrm{CR}}(G, \mathcal{C}, \varnothing, \varnothing, \mathcal{R}^G, \varrho, \underline{x}, \overline{x}) = Y \times \{1\}$.

3.1.2 Notations

There are a few conventions that we use to simplify notation in the remaining part of this work:

- We will often refer to installable components at a given graph element, i.e. all components of a set $\mathcal{S} \subseteq \mathcal{C}$ which can be installed at a graph element $g \in \mathcal{G}$. For that matter, we denote

  $$\mathcal{S}(g) := \{c \in \mathcal{S} \mid \overline{x}_{c,g} > 0\} \quad \forall \mathcal{S} \subseteq \mathcal{C},\ g \in \mathcal{G}.$$

  Notice that this does not define some mapping named $\mathcal{S}$. It is just a notation, where $\mathcal{S}$ is a set of components. For example, let u and v be two different nodes in V, and let $U := \{u,v\}$. The nodes are components as well. According to the notation introduced here, the statement

  $$U(v) = \{u,v\}(v) = \{v\}(v) = \{v\}$$

  holds.

- For $g \in \mathcal{G}$, the components in $\mathcal{C} \backslash \mathcal{C}(g)$ are completely irrelevant at g. Consequently, we will often neglect the existence of $x_{c,g}$ variables in X_{CR} where $\underline{x}_{c,g} = \overline{x}_{c,g} = 0$.

- If resource values for a given component $c \in \mathcal{C}$ and resource $r \in \mathcal{R}$ are equal for all graph elements, i.e. $\varrho^r_{c,g} = \varrho^r_{c,g'}$ for all $g, g' \in \mathcal{G}$, we will simply write ϱ^r_c.

  Notice that $\varrho^r_{c,g}$ can be set to any arbitrary value for $c \in \mathcal{C} \backslash \mathcal{C}(g)$. We will use this fact to use ϱ^r_c for components whose resource values are equal only where they can be installed as well.

- We will need to denote whether a given point fulfills a particular resource inequality. For a resource $r \in \mathcal{R}$ and a graph element $g \in \mathcal{G}$, we denote $x \in R(r,g)$, if x fulfills the resource inequality for r indexed by g, provided such an inequality exists. We aim at using this definition for points of different dimensions and possibly nonmatching pairs of resources and graph elements. For that matter, we introduce $R(r,g)$ as a class step by step.

  First, let $x \in \mathbb{R}^{\mathcal{C}\times\mathcal{G}}$ and $g \in \mathcal{G}$. For a node resource $r_1 \in \mathcal{R}^V$, we define

  $$x \in R(r_1,g) :\iff (\ g \in V \text{ and } \sum_{g' \in \{g\}\cup\delta(g)} \sum_{c \in \mathcal{C}} \varrho^{r_1}_{c,g'} x_{c,g'} \geqslant 0\ ) \text{ or } g \notin V\ .$$

  The according definition for local resources $r_2 \in \mathcal{R}^L$ is

  $$x \in R(r_2,g) :\iff \sum_{c \in \mathcal{C}} \varrho^{r_2}_{c,g} x_{c,g} \geqslant 0\ ,$$

  and for global resources $r_3 \in \mathcal{R}^G$, it is

  $$x \in R(r_3,g) :\iff (\ g = G \text{ and } \sum_{g' \in \mathcal{G}} \sum_{c \in \mathcal{C}} \varrho^{r_3}_{c,g'} x_{c,g'} \geqslant 0\ ) \text{ or } g \neq G\ .$$

  These definitions allow to denote $x \in R(r,g)$ for points of appropriate dimension. Notice that

  $$X_{\mathrm{CR}} = \left\{x \in \mathbb{Z}_+^{\mathcal{C}\times\mathcal{G}} \mid \underline{x} \leqslant x \leqslant \overline{x},\ x \in R(r,g)\ \forall r \in \mathcal{R},\ g \in \mathcal{G}\right\}$$

  holds.

  Additionally, we use this notation for points in other spaces, provided they can be indexed by pairs of components and graph elements. This becomes useful in Chapter 5, where projections of X_{CR} to spaces of other dimensions are used. The according transformations are obvious and not explicitly stated here.

3.1.3 Example

Let $G = (V,E)$ be a graph with edge lengths $(d_e)_{e\in E}$, and $\mathcal{G} = \mathcal{G}(G)$ be its set of graph elements. On the edges, a single type of capacity can be installed. There is one type of router (resp. switch or hub) available. It has s sockets compatible with the cabling. The router is stackable, i.e., it is possible to connect multiple of them without utilizing the sockets. An unlimited number of such routers can be installed at each node. The installed cabling is limited to have a total length of at most D, but pieces of arbitrary length can be produced without loss.

To model this scenario, we use two components: c_1 for the router and c_2 for the cabling. Thus, $\mathcal{C} := \mathcal{G}\dot{\cup}\{c_1, c_2\}$. Their bounds are

$$\begin{array}{lcll} \underline{x}_{c_1,g} & := & 0 & \forall g \in \mathcal{G}, \\ \overline{x}_{c_1,v} & := & \infty & \forall v \in V, \\ \overline{x}_{c_1,g} & := & 0 & \forall g \in \mathcal{G}\setminus V, \\ \\ \underline{x}_{c_2,g} & := & 0 & \forall g \in \mathcal{G}, \\ \overline{x}_{c_2,e} & := & 1 & \forall e \in E, \\ \overline{x}_{c_2,g} & := & 0 & \forall g \in \mathcal{G}\setminus E. \end{array}$$

The set of resources is $\mathcal{R} = \{1,2\}$, with $1 \in \mathcal{R}^V$ and $2 \in \mathcal{R}^G$, where

- resource 1 is used to model free sockets at nodes. It is provided by routers and consumed by installed cablings:

  $$\varrho^1_{c_1} := +s, \quad \varrho^1_{c_2} := -1\ .$$

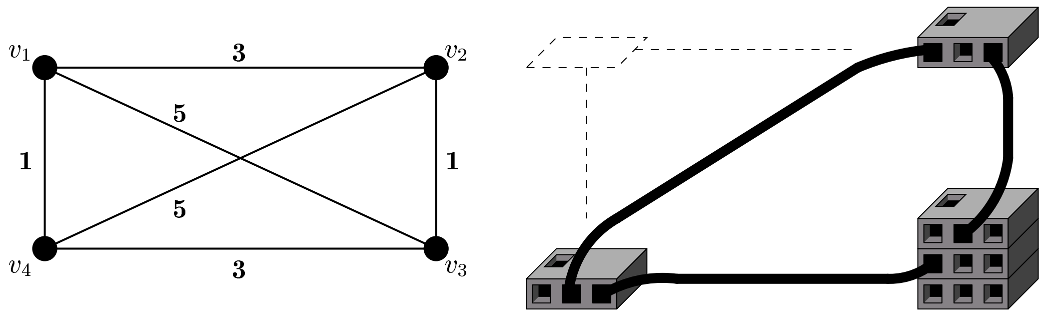


Figure 3.2: Example component installation. *Left: Graph G with edge lengths d_e, Right: Router components c_1 and cabling components c_2 from x^*.*

Here, we use the fact that resource values of a component can be set to any value where the component is not installable: Setting $\varrho^1_{c_1,v} := +s$ for all nodes $v \in V$ and $\varrho^1_{c_1,g} := 0$ for the remaining graph elements $g \in \mathcal{G} \setminus V$ is also correct, but needs complex notations and introduces special cases without any benefit.

- resource 2 models the available cabling. The global graph element G provides it, while installed cabling consumes it:

$$\varrho^2_G := +D, \quad \varrho^2_{c_2,e} := -d_e \; \forall e \in E.$$

Setting the remaining resource values to zero, the induced formulation looks like follows:

$$\sum_{e \in \delta(v)} x_{c_2,e} \leqslant s x_{c_1,v} \quad \forall v \in V \tag{3.5}$$

$$\sum_{e \in E} d_e x_{c_2,e} \leqslant D \tag{3.6}$$

$$x_{c_1,v} \in \mathbb{Z}_+ \quad \forall v \in V$$

$$x_{c_2,e} \in \mathbb{B} \quad \forall e \in E$$

For better readibility, some changes were applied here: The coefficients of $x_{G,G}$ were moved to the right hand side, and variables were left out if they are fixed or have 0-coefficients only.

With parameters $D = 10$, $s = 3$ and using graph G and edge lengths d_e as shown in Figure 3.2, a feasible component installation $x^* \in X_{\text{CR}}$ is the following, which is visualized in Figure 3.2 as well:

$$x^*_{c_1,g} = \begin{cases} 1 & \text{if } g = v_2 \text{ or } g = v_4 \\ 3 & \text{if } g = v_3 \\ 0 & \text{otherwise} \end{cases} \qquad \forall g \in \mathcal{G}\,,$$

$$x^*_{c_2,g} = \begin{cases} 1 & \text{if } g = v_2v_3,\ g = v_2v_4 \text{ or } g = v_3v_4 \\ 0 & \text{otherwise} \end{cases} \qquad \forall g \in \mathcal{G}\,,$$

$$x^*_{g',g} = \begin{cases} 1 & \text{if } g = g' \\ 0 & \text{if } g \neq g' \end{cases} \qquad \forall g, g' \in \mathcal{G}\,.$$

3.2 Hardware-Capacity Model

The CR model as presented in the previous section introduces the concepts of components and resources, which are rather abstract. Now we present a specialization that reflects some aspects of certain network problems in more detail.

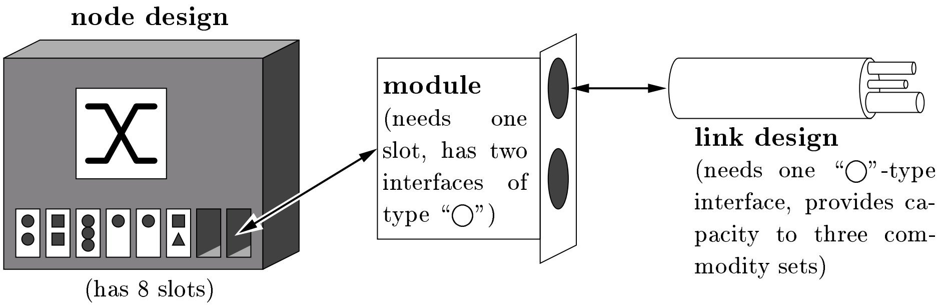


Figure 3.3: Hardware modelled by the HC model. *Left to right: A node design has slots. They are occupied by modules, which offer interfaces. These are consumed by link designs, which offer capacity for consumption by commodity set capacity components.*

First, a generalized set of hardware is presented, together with a capacity model, most of which describes components. Then certain limitations and interactions between these components are introduced as resources.

The issue of applicability is not elaborated here, since Chapter 6 deals specifically with using this model for Scenarios 1.1 to 1.3.

We now introduce node designs, interfaces, modules, link designs and commodity set capacity components for the HC model, refer to Figure 3.3 for a simplified introductory picture.

A *node design* is a predefined base configuration for a node of the network. We allow to pick at most one design per node. The key properties of a node design are the maximum capacity that can be connected to it (the *switching capacity*), the types of supported modules (see below) as well as per-module upper bounds, and the number of free *slots* available for module installation. An example for a typical node design is a single type of router, which motivates the terms 'switching capacity' and 'slot', but aggregated and more complex hardware setups are possible as well. Figure 3.3 shows a node design with eight slots, six of which being already occupied.

Interfaces are connector types. Besides allowing distinction between different types, there is no property modelled for them. Notice that they are not represented by components, but appear as resources below. Interfaces appearing in Figure 3.3 are "□", "△" and "◯".

Modules are added to the node designs, allowing for link connection. It is possible to install multiple modules of same or different type into a node design, if two conditions are met: First, modules may occupy one or more slots, where the number of slots depends only on the module. Second, each node design gives an upper bound on the supported maximum for each module. Modules may carry interfaces in arbitrary amounts and combinations, which are available for link design connection. Figure 3.3 includes a module that uses one slot while providing two interfaces of type "◯".

A *link design* is some determined capacity configuration on a link. It is determined by its payload capacity, and a set of interfaces it connects to in both end nodes of a link. An edge may have a preinstalled link design, which is fixed and can not be removed. This is however not done by real link designs, we rather duplicate all characteristic parameters of link designs for edges. On top of preinstalled capacity, for each edge a set of available link designs of which one may be chosen is given.

A *commodity set* is something that needs reserved capacities on the network's links. The only determining parameter is a commodity set's *routing unit*, specifying the amount of basic capacity units that have to be reserved on a link to provide one unit to the commodity set. Capacity is allocated in integral batches. There may be multiple commodity sets, each with its own routing unit and reserved capacities. Notice that this complies with the description of commodities and commodity sets in Section 2.2. Capacity allocation is done by installing *commodity set capacity components*.

$\mathcal{D}$		set of node design components
$\mathcal{M}$		set of module components
$\mathcal{L}$		set of link design components
$\mathcal{K}$		set of commodity set capacity components
$\mathcal{I}$		index set of interface types
C^d	$(d \in \mathcal{D})$	switching capacity of node design d
C^ℓ	$(\ell \in \mathcal{L})$	routing capacity provided by link design ℓ
C^0_e	$(e \in E)$	preinstalled routing capacity on edge e
$I^{i,m}$	$(m \in \mathcal{M},\ i \in \mathcal{I})$	number of type i interfaces provided by module m
$I^{i,\ell}$	$(\ell \in \mathcal{L},\ i \in \mathcal{I})$	number of type i interfaces consumed by link design ℓ
$I^{i,0}_e$	$(e \in E,\ i \in \mathcal{I})$	number of type i interfaces consumed by preinstalled capacities on e
$M^{d,m}$	$(d \in \mathcal{D},\ m \in \mathcal{M})$	number of modules m supported by node design d
S^m	$(m \in \mathcal{M})$	number of slots occupied by module m
S^d	$(d \in \mathcal{D})$	number of slots provided by node design d
U^k	$(k \in \mathcal{K})$	routing unit of commodity set k
$\hat{Q}$		overall cost budget
$Q_{c,g}$	$(c \in \mathcal{C}, g \in \mathcal{G})$	cost of component c if installed at g

Table 3.1: Problem parameters for HC model. *The first four lines list the component sets, the following entry ($\mathcal{I}$) indexes interface resources, the remainder are parameter values.*

Now, we present components, resources, and resource values step by step, see Table 3.1 for a summary of symbols and parameters used. First, there is the underlying *supply graph*

$$G = (V, E)\ ,$$

where the edges model potential capacity carrying links in the network, i.e., there is an edge $\{u, v\} \in E$ if it is possible to have a direct capacitated link between u and v.

3.2.1 Components

Let

$$\mathcal{C} := \mathcal{G} \,\dot\cup\, \mathcal{D} \,\dot\cup\, \mathcal{M} \,\dot\cup\, \mathcal{L} \,\dot\cup\, \mathcal{K}\ ,$$

where $\mathcal{G}$ is the set of graph elements of G. The remaining component sets and components, and their lower and upper bounds, are:

$\mathcal{D}$: Node design components. Each node design is represented by a *node design component* $d \in \mathcal{D}$, which can be installed at some nodes only, and there maximally once. Thus, the upper bound on a node $v \in V$ is 1, if the node design is available for v, and 0 otherwise:

$$\begin{aligned} \underline{x}_{d,g} &= 0 \quad \forall g \in \mathcal{G}, \\ \overline{x}_{d,g} &= 0 \quad \forall g \in \mathcal{G} \backslash V, \\ \overline{x}_{d,g} &\in \mathbb{B} \quad \forall g \in V. \end{aligned}$$

$\mathcal{M}$: Module components. Modules are modelled as *module components* $m \in \mathcal{M}$. Modules are available at nodes only, without upper bounds:

$$\begin{aligned} \underline{x}_{m,g} &= 0 \quad \forall g \in \mathcal{G}, \\ \overline{x}_{m,g} &= 0 \quad \forall g \in \mathcal{G} \backslash V, \\ \overline{x}_{m,g} &= \infty \quad \forall g \in V. \end{aligned}$$

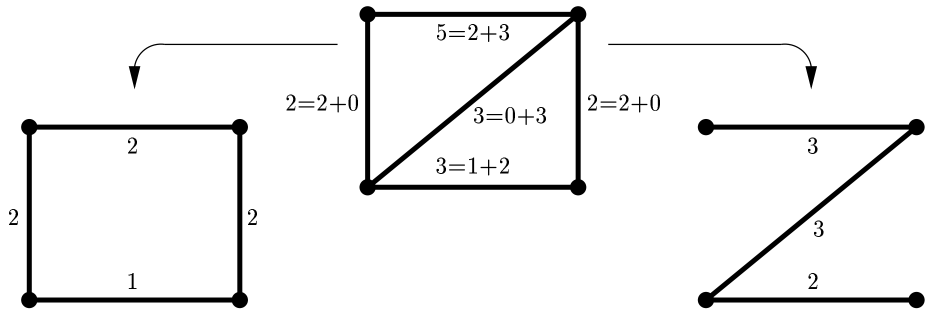


Figure 3.4: Commodity set subgraph. *Installation of commodity set capacity components provides each commodity set with a capacitated subgraph for private use.*

$\mathcal{L}$: **Link design components.** Link designs are modelled as *link design components* $\ell \in \mathcal{L}$. They are installable at some edges only:

$$\begin{array}{rcll} \underline{x}_{\ell,g} & = & 0 & \forall g \in \mathcal{G}, \\ \overline{x}_{\ell,g} & = & 0 & \forall g \in \mathcal{G} \backslash E, \\ \overline{x}_{\ell,g} & \in & \mathbb{B} & \forall g \in E. \end{array}$$

$\mathcal{K}$: **Commodity set capacity components.** For each commodity set, there is a component $k \in \mathcal{K}$. Installation of component k on an edge models reserving some capacity for one particular commodity set on that edge, thereby producing a capacitated subgraph for that commodity set, see Figure 3.4. Variable bounds for k are

$$\begin{array}{rcll} \underline{x}_{k,g} & = & 0 & \forall g \in \mathcal{G}, \\ \overline{x}_{k,g} & = & 0 & \forall g \in \mathcal{G} \backslash E, \\ \overline{x}_{k,g} & = & \infty & \forall g \in E. \end{array}$$

Notice that by using these commodity set capacitaty components, every commodity set can have a different routing model. E.g., if there is a LP-based formulation for the routing model of a commodity set with capacity component $k \in \mathcal{K}$, adding this LP to the component formulation using $x_{k,e}$, $e \in E$, as capacities on the edges, the routing problem for this commodity set is added without interfering with other commodity sets.

3.2.2 Resources

For the resources, we set $\mathcal{R} := \mathcal{R}^V \,\dot{\cup}\, \mathcal{R}^L \,\dot{\cup}\, \mathcal{R}^G$ and

$$\begin{array}{rcl} \mathcal{R}^V & := & \{\text{scap}\} \cup \{\text{ifav}_i \mid i \in \mathcal{I}\} \ , \\ \mathcal{R}^L & := & \{\text{slot}, \text{ndav}, \text{rcap}, \text{ldav}\} \cup \{\text{mod}_m \mid m \in \mathcal{M}\} \ , \\ \mathcal{R}^G & := & \{\text{cost}\} \ , \end{array}$$

where the resources are arbitrary but unique indices in $\mathcal{R}$. We use 4-letter words as symbols for these indices to increase readability. These resources and their purposes, as well as the set $\mathcal{I}$, are presented in detail below. Notice that nearly all of the resources produce some trivial inequalities $0 \leqslant 0$ or $0 \leqslant 1$ in addition to those explicitly listed. These are left out for obvious reasons.

ndav: **Available node designs.** To model the restriction to install at most one node design at a supply node, we use the local resource ndav $\in \mathcal{R}^L$. Each installed node design consumes 1 of this resource, while the nodes themselves provide 1:

$$\varrho_c^{\text{ndav}} := \begin{cases} +1 & \text{if } c \in V \\ -1 & \text{if } c \in \mathcal{D} \\ 0 & \text{otherwise} \end{cases} \quad .$$

The resource inequalities (3.2) for ndav are called *node design GUB inequalities*. They are equivalent to

$$\sum_{d \in \mathcal{D}} x_{d,v} \leqslant 1 \quad \forall v \in V \ . \tag{3.7}$$

slot**: Slots.** For a node design $d \in \mathcal{D}$, the parameter

$$S^d \in \mathbb{Z}_+$$

is the number of slots provided by d, while for a module $m \in \mathcal{M}$ the parameter

$$S^m \in \mathbb{Z}_+$$

is the number of slots consumed by m if it is installed in some node design. We use the local resource slot $\in \mathcal{R}^L$ and consequently set the associated resource values to

$$\varrho_c^{\text{slot}} := \begin{cases} +S^c & \text{if } c \in \mathcal{D} \\ -S^c & \text{if } c \in \mathcal{M} \\ 0 & \text{otherwise} \end{cases} .$$

Thus, the resource inequalities (3.2) for slot are

$$\sum_{m \in \mathcal{M}} S^m x_{m,v} \ \leqslant \ \sum_{d \in \mathcal{D}} S^d x_{d,v} \quad \forall v \in V \tag{3.8}$$

and are called *slot inequalities*.

mod**: Module bounds.** For a module $m \in \mathcal{M}$ and a node design $d \in \mathcal{D}$, the parameter

$$M^{d,m} \in \mathbb{Z}_+$$

specifies how many modules m the node design can support, independent of slot requirements. The local resources $\text{mod}_m \in \mathcal{R}^L \ \forall m \in \mathcal{M}$ are used to model these bounds, with resource values

$$\varrho_c^{\text{mod}_m} := \begin{cases} -1 & \text{if } c = m \\ +M^{c,m} & \text{if } c \in \mathcal{D} \\ 0 & \text{otherwise} \end{cases} \qquad \forall m \in \mathcal{M} \ ,$$

giving the *module inequalities* from local resource inequalities (3.2)

$$x_{m,v} \leqslant \sum_{d \in \mathcal{D}} M^{d,m} x_{d,v} \quad \forall v \in V, \ m \in \mathcal{M} \ . \tag{3.9}$$

ifav,$\mathcal{I}$**: Available interfaces.** Let $\mathcal{I}$ be an index set, where each element models one particular type of interface. Let $i \in \mathcal{I}$. The parameters

$$I^{i,\ell} \in \mathbb{Z}_+$$

specify the number of type-i interfaces a link design $\ell \in \mathcal{L}$ connects to, while

$$I_e^{i,0} \in \mathbb{Z}_+$$

defines the same for the preinstalled capacity on an edge $e \in E$. Interfaces are provided by modules, so the parameter

$$I^{i,m} \in \mathbb{Z}_+$$

controls the number of type-i interfaces provided by a module $m \in \mathcal{M}$. For each interface $i \in \mathcal{I}$, the node resource $\text{ifav}_i \in \mathcal{R}^V$ models interface consumption using resource values

$$\varrho_c^{\text{ifav}_i} := \begin{cases} +I^{i,c} & \text{if } c \in \mathcal{M} \\ -I_c^{i,0} & \text{if } c \in E \\ -I^{i,c} & \text{if } c \in \mathcal{L} \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in \mathcal{I},$$

resulting in node resource inequalities (3.1)

$$\sum_{e \in \delta(v)} \left(I_e^{i,0} + \sum_{\ell \in \mathcal{L}} I^{i,\ell} x_{\ell,e} \right) \quad \leqslant \quad \sum_{m \in \mathcal{M}} I^{i,m} x_{m,v} \quad \forall v \in V,\ i \in \mathcal{I}\,, \tag{3.10}$$

called *interface inequalities.*

scap**: Switching capacity.** Node designs have a switching capacity

$$C^d \in \mathbb{Z}_+ \quad (d \in \mathcal{D})\,,$$

which is consumed by capacities installed on incident edges: On an edge $e \in E$, there is a preinstalled capacity of

$$C_e^0 \in \mathbb{Z}_+\,.$$

Additionally, a link design $\ell \in \mathcal{L}$ consumes additional capacity of

$$C^\ell \in \mathbb{Z}_+\,.$$

This is modelled using the node resource $\text{scap} \in \mathcal{R}^V$ with resource values

$$\varrho_c^{\text{scap}} := \begin{cases} +C^c & \text{if } c \in \mathcal{D} \\ -C_c^0 & \text{if } c \in E \\ -C^c & \text{if } c \in \mathcal{L} \\ 0 & \text{otherwise} \end{cases}\,,$$

leading to the *switching capacity inequalities*

$$\sum_{e \in \delta(v)} \left(C_e^0 + \sum_{\ell \in \mathcal{L}} C^\ell x_{\ell,e} \right) \quad \leqslant \quad \sum_{d \in \mathcal{D}} C^d x_{d,v} \quad \forall v \in V\,. \tag{3.11}$$

ldav**: Available link designs.** There is a constraint specifying that on each edge at most one link design may be chosen, using local resource $\text{ldav} \in \mathcal{R}^L$ and resource values

$$\varrho_c^{\text{ldav}} := \begin{cases} +1 & \text{if } c \in E \\ -1 & \text{if } c \in \mathcal{L} \\ 0 & \text{otherwise} \end{cases}\,,$$

yielding the *link design GUB inequalities* (3.2)

$$\sum_{\ell \in \mathcal{L}} x_{\ell,e} \leqslant 1 \quad \forall e \in E\,. \tag{3.12}$$

rcap**: Routing capacity.** On an edge $e \in E$, there is a preinstalled capacity of

$$C_e^0 \in \mathbb{Z}_+$$

available, which is provided by the edge itself. Additionally, a link design $\ell \in \mathcal{L}$ provides additional capacity of

$$C^\ell \in \mathbb{Z}_+ .$$

Capacity is consumed by the commodity set capacity components $k \in \mathcal{K}$. Installing one such component consumes

$$U^k \in \mathbb{Z}_+,\ U^k > 0$$

base capacity units, and reserves one unit of its commodity set's routing unit for the commodity set. We use the local resource rcap $\in \mathcal{R}^L$ to model this relation by setting

$$\varrho_c^{\text{rcap}} := \begin{cases} +C^c & \text{if } c \in \mathcal{L} \\ +C_c^0 & \text{if } c \in E \\ -U^c & \text{if } c \in \mathcal{K} \\ 0 & \text{otherwise} \end{cases} ,$$

with *routing capacity inequalities*

$$\sum_{k\in\mathcal{K}} U^k x_{k,e} \leqslant C_e^0 + \sum_{\ell\in\mathcal{L}} C^\ell x_{\ell,e} \quad \forall e \in E . \tag{3.13}$$

cost: **Cost.** The parameters $Q_{c,g} \in \mathbb{Z}_+$ specify the cost of installing one component $c \in \mathcal{C}$ at graph element $g \in \mathcal{G}$. Note that the cost of graph elements does not have to be zero — edges, for example, carry the cost of preinstalled capacities.

The parameter $\hat{Q} \in \overline{\mathbb{Z}}_+$ specifies the available budget, that is, total component installation cost must not exceed $\hat{Q}$. This is done using the global resource cost with values

$$\varrho_{c,g}^{\text{cost}} := \begin{cases} \hat{Q} - Q_{c,g} & \text{if } c = g = G \\ -Q_{c,g} & \text{otherwise} \end{cases} ,$$

and its *budget inequality*

$$\sum_{c\in\mathcal{C}} \sum_{g\in\mathcal{G}} Q_{c,g} x_{c,g} \leqslant \hat{Q} . \tag{3.14}$$

Notice that another use for component costs is the objective function of an optimization process: Instead of restricting solutions to those not exhausting some predefined budget, we try to maximize the excess of this resource or, equivalently, minimize the total component installation cost $\sum_{c\in\mathcal{C}} \sum_{g\in\mathcal{G}} Q_{c,g} x_{c,g}$.

Definition 3.2 (HC integer set X_{HC})
Using components and resource as shown above, we define X_{HC} as the set of feasible component installations for the HC model:

$$\begin{aligned} X_{\text{HC}} &:= X_{\text{CR}}(G, \mathcal{C}, \mathcal{R}^V, \mathcal{R}^L, \mathcal{R}^G, \varrho, \underline{x}, \overline{x}) \\ &= \left\{ x \in \mathbb{Z}_+^{\mathcal{C}\times\mathcal{G}} : \underline{x} \leqslant x \leqslant \overline{x},\ x \textit{ satisfies } (3.7) \textit{ to } (3.14) \right\} . \end{aligned} \tag{3.15}$$

As a summary, let us present a description of X_{HC} projected onto the space of nonfixed variables:

$$
\left\{
\begin{array}{rcll}
\sum\limits_{d\in\mathcal{D}(v)} x_{d,v} & \leqslant & 1 & \forall v\in V,\\
\sum\limits_{d\in\mathcal{D}(v)} S^d x_{d,v} - \sum\limits_{m\in\mathcal{M}(v)} S^m x_{m,v} & \geqslant & 0 & \forall v\in V,\\
\sum\limits_{d\in\mathcal{D}(v)} M^{d,m} x_{d,v} \;\; - x_{m,v} & \geqslant & 0 & \forall v\in V,\ m\in\mathcal{M},\\
\sum\limits_{m\in\mathcal{M}(v)} I^{i,m} x_{m,v} - \sum\limits_{e\in\delta(v)} \sum\limits_{\ell\in\mathcal{L}(e)} I^{i,\ell} x_{\ell,e} & \geqslant & \sum\limits_{e\in\delta(v)} I_e^{i,0} & \forall v\in V,\ i\in\mathcal{I},\\
\sum\limits_{d\in\mathcal{D}} C^d x_{d,v} - \sum\limits_{e\in\delta(v)} \sum\limits_{\ell\in\mathcal{L}(e)} C^\ell x_{\ell,e} & \geqslant & \sum\limits_{e\in\delta(v)} C_e^0 & \forall v\in V,\\
\sum\limits_{\ell\in\mathcal{L}(e)} x_{\ell,e} & \leqslant & 1 & \forall e\in E,\\
\sum\limits_{k\in\mathcal{K}} U^k x_{k,e} - \sum\limits_{\ell\in\mathcal{L}(e)} C^\ell x_{\ell,e} & \leqslant & C_e^0 & \forall e\in E,\\
\sum\limits_{c\in\mathcal{C}} \sum\limits_{g\in\mathcal{G}} Q_{c,g} x_{c,g} & \leqslant & \hat{Q} & ,\\
x_{d,v} & \in & \mathbb{B} & \forall v\in V,\ d\in\mathcal{D}(v),\\
x_{\ell,e} & \in & \mathbb{B} & \forall e\in E,\ \ell\in\mathcal{L}(e),\\
x_{m,v} & \in & \mathbb{Z}_+ & \forall v\in V,\ m\in\mathcal{M}(v).
\end{array}
\right.
$$

3.2.3 HC Model Extensions

While the HC model as introduced in the preceeding section is by itself sufficient for a great variety of network planning problems, there are some limitations that hinder direct applicability for the planning scenarios from Chapter 1.

Now, three model extensions are introduced to overcome these limitations. First, support for external traffic at the nodes is added. Second, node design parameters are refined to be adjustable based on the graph element where the design is installed. Third, a means to keep track of changes with respect to a predefined reference installation is presented, and it is shown how to add constraints to such changes.

Tributary Deprivation

A telecommunication network without any connection to the outside world, i.e., to other networks or hosts, is obviously useless. Hence, some or all nodes of a network usually are gateway nodes where data enters and leaves the network. In its basic form, the HC model does not directly support the resource usage resulting from these external connections.

We now introduce a model extension where the nodes simulate the effect of additional links with preinstalled capacities by consuming interfaces and switching capacity. Because these simulated links are typically used to model the data flow to lower- or higher-order layers of a network, which is called tributary traffic, this model extension is called *tributary deprivation*.

For each node $v \in V$, additional parameters

$$I_v^{i,0} \in \mathbb{Z}_+ \quad \forall i \in \mathcal{I}, \text{ and } C_v^0 \in \mathbb{Z}_+$$

are introduced. The former parameter set defines the interfaces that have to be provided, while the latter parameter specifies the additional switching capacity required at v. These parameters are incorporated into the underlying CR model by assigning the resource values for the interface and switching capacity resources:

$$\varrho_{v,v}^{\mathrm{ifav}_i} := -I_v^{i,0}, \quad \varrho_{v,v}^{\mathrm{scap}} := -C_v^0 \quad \forall v \in V,\ i \in \mathcal{I}\ .$$

This changes the interface inequalities (3.10) and the switching capacity inequalities (3.11) by adding a constant to the respective right hand sides. They now read

$$\sum_{e\in\delta(v)} \left(I_e^{i,0} + \sum_{\ell\in\mathcal{L}} I^{i,\ell} x_{\ell,e} \right) \leqslant \sum_{m\in\mathcal{M}} I^{i,m} x_{m,v} - I_v^{i,0} \quad \forall v \in V,\ i \in \mathcal{I}\ , \tag{3.10b}$$

$$\sum_{e\in\delta(v)} \left(C_e^{0} + \sum_{\ell\in\mathcal{L}} C^{\ell} x_{\ell,e} \right) \leqslant \sum_{d\in\mathcal{D}} C^{d} x_{d,v} - C_v^{0} \quad \forall v \in V\ , \tag{3.11b}$$

specifying that of the interfaces and switching capacity provided by installed node designs, a certain fixed amount is not available for consumption on incident edges.

Parameter Localization

A node design $d \in \mathcal{D}$ is described by the problem parameters C^d, S^d, and $M^{d,m}$ ($\forall m \in \mathcal{M}$). None of these parameters depends on the actual node of installation. This does make sense for the envisaged applications: A multiplexer does not change its slot count when moved from one node location to another. However, there actually are situations where the ability to specify different node design parameters per node benefits the model. Foremost, in Chapter 5, a preprocessing method that utilizes such an extension is presented.

For each node $v \in V$, let

$$C_v^d \in \mathbb{Z}_+, \quad S_v^d \in \mathbb{Z}_+, \quad M_v^{d,m} \in \mathbb{Z}_+ \quad \forall m \in \mathcal{M}$$

be localized parameter replacements for C^d, S^d, and $M^{d,m}$, respectively. The according resource values are updated to become

$$\varrho_{c,g}^{\text{slot}} := \begin{cases} +S_g^c & \text{if } c \in \mathcal{D}, g \in V \\ -S^c & \text{if } c \in \mathcal{M} \\ 0 & \text{otherwise} \end{cases} ,$$

$$\varrho_{c,g}^{\text{mod}_m} := \begin{cases} -1 & \text{if } c = m \\ +M_g^{c,m} & \text{if } c \in \mathcal{D}, g \in V \\ 0 & \text{otherwise} \end{cases} \qquad \forall m \in \mathcal{M}\ ,$$

$$\varrho_{c,g}^{\text{scap}} := \begin{cases} +C_g^c & \text{if } c \in \mathcal{D}, g \in V \\ -C_c^0 & \text{if } c \in E \\ -C^c & \text{if } c \in \mathcal{L} \\ 0 & \text{otherwise} \end{cases} .$$

Using these resource values, X_{CR} (and hence X_{HC} as well) implements the localized node design.

This extension does not provide anything new: By using a larger node design set $\mathcal{D}' := \mathcal{D} \times V$, the same can be achieved without changing the model. The only benefit of the proposed way is the intuitive interpretation.

Expansion Tracking

In network expansion problems there may be limitations on how much might be "changed" w.r.t. a reference network state. This may for example be implied because the actual process of installing or removing hardware involves shutting down particular parts of the network, and the resulting loss of operation must be moderate. Now, a possibility to model change budgets for single components is introduced:

Let

$$b \in \mathcal{C}$$

be a component for which change limitations apply. For each graph element $g \in \mathcal{G}$, the parameter

$$\hat{x}_{b,g} \in \mathbb{Z}_+$$

specifies how many components b are considered being currently installed at g.

If installation or removal is limited locally for some $g \in \mathcal{G}$, i.e., $x_{b,g}$ must fulfill $\hat{x}_{b,g} - t^- \leqslant x_{b,g} \leqslant \hat{x}_{b,g} + t^+$ with some parameters $t^-, t^+ \in \mathbb{Z}_+$, according constraints can be incorporated by simply adjusting the variable bounds $\underline{x}_{b,g}$ resp. $\overline{x}_{b,g}$. If limits are however globally defined, more has to be done.

Component installations are checked for three types of differences:

- *Increments*, when additional components are present in the new installation, i.e., $x_{b,g} > \hat{x}_{b,g}$.
- *Decrements*, when components where removed ($x_{b,g} < \hat{x}_{b,g}$), and
- *Changes*, for both increments and decrements: $x_{b,g} \neq \hat{x}_{b,g}$.

First, two artificial components

$$b^+, b^- \in \mathcal{C}$$

are introduced. Their purpose is to count local changes: For a graph element $g \in \mathcal{G}$, we enforce

$$\hat{x}_{b,g} - x_{b^-,g} \quad \leqslant \quad x_{b,g} \quad \leqslant \quad \hat{x}_{b,g} + x_{b^+,g} \tag{3.16}$$

by adding two local resources to the system, namely

$$\mathrm{ldec}_b, \mathrm{linc}_b \in \mathcal{R}^L \quad .$$

The first, ldec_b, ensures that $x_{b^-,g}$ increases when $x_{b,g}$ drops below $\hat{x}_{b,g}$. This is achieved by having a resource deficit at g, which is compensated by both $x_{b^-,g}$ and $x_{b,g}$:

$$\varrho_c^{\mathrm{ldec}_b} := \begin{cases} -\hat{x}_{b,c} & \text{if } c \in \mathcal{G} \\ +1 & \text{if } c = b^- \\ +1 & \text{if } c = b \\ 0 & \text{otherwise} \end{cases} \quad .$$

The local resource inequalities (3.2) induced by this resource then model the left part of (3.16) and are equivalent to

$$x_{b^-,g} \quad \geqslant \quad \hat{x}_{b,g} - x_{b,g} \quad \forall g \in \mathcal{G} \ . \tag{3.17}$$

linc_b is used for increase counting. The graph element itself provides a fixed amount of this resource, the component b^+ further increases it, and installation of b consumes it:

$$\varrho_c^{\mathrm{linc}_b} := \begin{cases} +\hat{x}_{b,c} & \text{if } c \in \mathcal{G} \\ +1 & \text{if } c = b^+ \\ -1 & \text{if } c = b \\ 0 & \text{otherwise} \end{cases} \quad ,$$

and the according resource inequality (3.2) can be written as

$$x_{b^+,g} \quad \geqslant \quad x_{b,g} - \hat{x}_{b,g} \quad \forall g \in \mathcal{G} \ . \tag{3.18}$$

Now that these two components count local changes, three global resources track global changes. Using

$$\mathrm{gdec}_b, \mathrm{ginc}_b, \mathrm{gcha}_b \in \mathcal{R}^G \ ,$$

global budget constraints for decrements, increments and any changes are formulated. The available budgets are determined by the parameters

$$T_b^- \in \mathbb{Z}_+, \quad T_b^+ \in \mathbb{Z}_+, \quad \text{and} \quad T_b^\pm \in \mathbb{Z}_+ \,,$$

for the maximum number of decrements, increments and changes, respectively.

First, resource gdec_b is provided by the global graph element G and consumed by installation of b^- at any graph element:

$$\varrho_c^{\mathrm{gdec}_b} := \begin{cases} +T_b^- & \text{if } c = G \\ -1 & \text{if } c = b^- \\ 0 & \text{otherwise} \end{cases} \,,$$

resulting in global resource inequality (3.3)

$$\sum_{g \in \mathcal{G}} x_{b^-,g} \quad \leqslant \quad T_b^- \,, \tag{3.19}$$

modelling that no more than (T_b^-)-many component decreases may happen globally.

Analogous, resource ginc_b formulates the increment budget, using resource values

$$\varrho_c^{\mathrm{ginc}_b} := \begin{cases} +T_b^+ & \text{if } c = G \\ -1 & \text{if } c = b^+ \\ 0 & \text{otherwise} \end{cases} \,,$$

leading to the global resource inequality

$$\sum_{g \in \mathcal{G}} x_{b^+,g} \quad \leqslant \quad T_b^+ \,. \tag{3.20}$$

Finally, the budget on changes is implemented using the resource gcha_b, which is again provided in fixed amount, $T_b^\pm$, and consumed by both increments and decrements:

$$\varrho_c^{\mathrm{gdec}_b} := \begin{cases} +T_b^\pm & \text{if } c = G \\ -1 & \text{if } c = b^- \\ -1 & \text{if } c = b^+ \\ 0 & \text{otherwise} \end{cases} \,,$$

with global resource inequality

$$\sum_{g \in \mathcal{G}} \left(x_{b^+,g} + x_{b^-,g} \right) \quad \leqslant \quad T_b^\pm \,. \tag{3.21}$$

Altogether, the new inequalities responsible for component change budgets are

$$\left\{ \begin{array}{rcll} x_{b,g} + x_{b^-,g} & \geqslant & \hat{x}_{b,g} & \forall g \in \mathcal{G} \,, \\ x_{b,g} - x_{b^+,g} & \leqslant & \hat{x}_{b,g} & \forall g \in \mathcal{G} \,, \\ \sum_{g \in \mathcal{G}} x_{b^-,g} & \leqslant & T_b^- & , \\ \sum_{g \in \mathcal{G}} x_{b^+,g} & \leqslant & T_b^+ & , \\ \sum_{g \in \mathcal{G}} \left(x_{b^+,g} + x_{b^-,g} \right) & \leqslant & T_b^\pm & . \end{array} \right.$$

Chapter 4

Polyhedral Structure

This chapter provides some insight into the structure of the HC model. We investigate two polyhedra that are associated with X_{HC}. For each, we introduce a class of valid inequalities and present conditions under which they are facet-defining. Additionally, we provide a separation heuristic for one of the classes.

4.1 Node Cut Inequalities

We now introduce a class of inequalities that are valid for any component installation that can be accompanied by a feasible demand routing, and identify members of this class, which define facets of the associated polyhedra. Then, we propose a heuristic separation algorithm for these inequalities.

4.1.1 Setup

Let $x \in X_{\mathrm{HC}}$ be a feasible component installation. There are two means to define link capacity: First, define $y_x : E \to \mathbb{Z}_+$ to be the mapping that gives the routing capacity provided by installed link designs:

$$y_x(e) := C_e^0 + \sum_{\ell \in \mathcal{L}(e)} C^\ell x_{\ell,e} \; . \tag{4.1}$$

Where there is no ambiguity on which point x is meant, y_x is simply denoted by y. Let $k \in \mathcal{K}$ be a commodity set capacity component. It provides a capacity allocation to its commodity set, which leads to another definition of link capacity: The mapping $y_{x,k} : E \to \mathbb{Z}_+$ defined by

$$y_{x,k}(e) := x_{k,e} \tag{4.2}$$

gives the link capacities available for the commodity set of k.

Let

$$X_{\mathrm{HC,R}} := \{x \in X_{\mathrm{HC}} \mid y_{x,k} \text{ allows for a feasible routing } \forall k \in \mathcal{K}\} \tag{4.3}$$

be the set of feasible component installations which represent feasible hardware configurations and simultaneously offer sufficient capacities for a routing.

In this section, we need to deal with actual demand routings. For that matter, we resort to assuming that the commodity sets use a simple routing model:

- There is just one commodity set, and it has a routing unit of one, i.e., $\mathcal{K} = \{k\}$ and $U^k = 1$.
- The commodity set is based on demands as introduced in Section 2.2. The according demand set is $\mathfrak{D}_k$.

- The demands are fulfilled by fractional flow paths, without limitations on the number of paths per demand.
- No additional constraints, e.g. survivability requirements, have to be taken into account.

When discussing routings, a question of importance is whether a node design on some node allows to send the emanating demand of that node over one incident edge. The following definition describes a situation where this question can always be answered:

Definition 4.1 (Sufficient designs)
X_{HC} *is said to have* designs sufficient for emanating demands*, if for each node $v \in V$ and incident edge $e \in \delta(v)$, every node design $d \in \mathcal{D}(v)$ can be accompanied by a link design $\ell \in \mathcal{L}(e)$ such that the emanating demand of v can be routed over e, i.e.:*

For each $v \in V$, $e \in \delta(v)$, and $d \in \mathcal{D}(v)$, there exists a link design $\ell \in \mathcal{L}(e)$ such that a point $x \in X_{\mathrm{HC}}$ exists, which fulfills $x_{d,v} = 1$, $x_{\ell,e} = 1$, and $y_{x,k}(e) \geqslant \mathfrak{d}(v)$.

We assume that on each node and each edge there is at least one node design resp. link design available, i.e., $\mathcal{D}(v) \neq \varnothing$ for each node $v \in V$ and $\mathcal{L}(e) \neq \varnothing$ for all $e \in E$. For each edge $e \in E$, $\hat{\ell}_e \in \mathcal{L}(e)$ denotes a link design of largest capacity, that is,

$$\hat{\ell}_e := \arg\max \left\{ C^\ell \mid \ell \in \mathcal{L}(e) \right\} \; . \tag{4.4}$$

In the same manner, for each $v \in V$, $\hat{d}_v \in \mathcal{D}(v)$ denotes a node design with largest switching capacity available on v:

$$\hat{d}_v := \arg\max \left\{ C^d \mid d \in \mathcal{D}(v) \right\} \; . \tag{4.5}$$

Definition 4.2 (Large node design, large link design)
Let $v \in V$ and $e \in E$. $\hat{d}_v$ is called the large node design *on v and $\hat{\ell}_e$ the* large link design *on e.*

Observe that $\frac{1}{2}\sum_{v\in V} \mathfrak{d}(v)$ is the total flow for all commodities, hence a link whose capacity exceeds this value can support any flow imposed by some routing, as long as simple paths are used. We now formalize the situation where there are very large capacities available, sufficient for any routing, and there is a node design that can support any incident capacity installation on each node:

Definition 4.3 (Trivially bounded)
Let $(\hat{\ell}_e)_{e\in E}$ and $(\hat{d}_v)_{v\in V}$ be defined as above. X_{HC} is said to be trivially bounded*, if the following two conditions hold:*

(i) For each edge $e \in E$, using the large link design $\hat{\ell}_e$ produces a capacity on e that is at least twice the total flow, i.e., $C_e^0 + C^{\hat{\ell}_e} \geqslant \sum_{v\in V} \mathfrak{d}(v)$.

(ii) For each node $v \in V$ and link design choice $L : \delta(v) \to \mathcal{L}$ with $L(e) \in \mathcal{L}(e)$ for every $e \in \delta(v)$ there exists a choice of modules $M : \mathcal{M}(v) \to \mathbb{Z}_+$ such that module bounds, slot inequalities, interface inequalities and switching capacity inequalities are met when installing $\hat{d}_v$ on v and the link designs as specified by L. Furthermore, $\hat{d}_v$ allows any module to be installed one additional time, i.e.,

$$\begin{aligned}
M^{\hat{d}_v,m} &\geqslant M(m) + 1 && \forall m \in \mathcal{M} \; , \\
S^{\hat{d}_v} &\geqslant \sum_{m\in\mathcal{M}} S^m (M(m) + 1) \; , \\
\sum_{m\in\mathcal{M}} I^{i,m} &\geqslant \sum_{e\in\delta(v)} \left(I_e^{i,0} + I^{i,L(e)} \right) && \forall i \in \mathcal{I} \; , \\
C^{\hat{d}_v} &\geqslant \sum_{e\in\delta(v)} C_e^0 + C^{L(e)} \; .
\end{aligned}$$

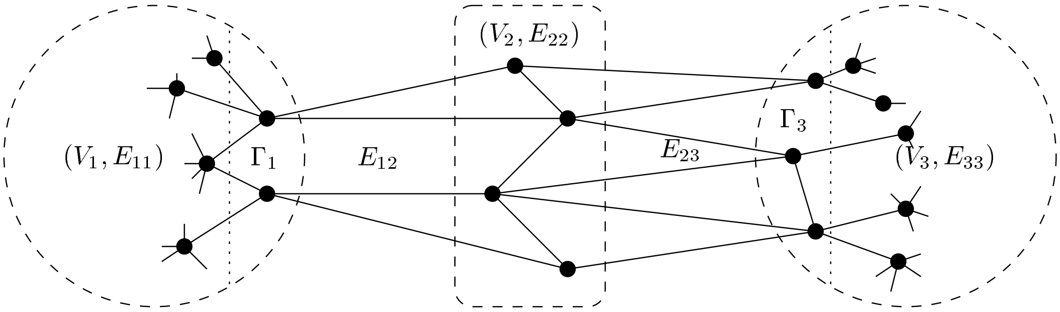


Figure 4.1: Node cut. *The node cut V_2 separates the nodes into three sets, V_1, V_2 and V_3. This induces an edge partition as well, where E_{pq} contains all edges between V_p and V_q. $\Gamma_1 \subseteq V_1$ and $\Gamma_3 \subseteq V_3$ contain the neighbours of V_2 in V_1 and V_3, respectively. The shown node cut is in fact a minimal node cut.*

Notice that for a cost minimization problem, $X_{\mathrm{HC,R}}$ can be adopted to suit this definition by introducing artificial node and link designs of very high cost. It may be surprising that condition (i) requires the capacity resulting from installing $\hat{\ell}$ to be twice the total flow, instead of simply once. In fact, this simplifies multiple argumentations in the proof that certain inequalities are facet-defining. For capacities that are larger than the total flow, actual capacity values do not really matter. The definition uses $\sum_{v \in V} \mathfrak{d}(v)$ as lower bound, because it is a simple term and sufficiently large.

For the remainder of this section, we will use a partition of the graph: Let $V = V_1 \mathbin{\dot\cup} V_2 \mathbin{\dot\cup} V_3$ be a partition of the nodes into nonempty subsets, i.e., $V_p \neq \varnothing$ for $p = 1, 2, 3$. This yields a partition of the edges based on the positions of their end nodes: Define

$$E_{pq} := \{uv \in E \mid u \in V_p,\ v \in V_q\} \quad \forall p, q = 1, 2, 3 :\ p \leqslant q\ . \tag{4.6}$$

Notice that (V_1, E_{11}), (V_2, E_{22}), and (V_3, E_{33}) are the three subgraphs induced by the node partition. We will frequently refer to the neighbors $\Gamma(V_2)$ of the nodes in V_2 in one of the two other partitions. This is denoted by

$$\Gamma_p := \Gamma(V_2) \cap V_p \quad \forall p = 1, 3\ . \tag{4.7}$$

Finally, observe that V_2 is a node cut of G if, and only if, $E_{13} = \varnothing$. This case is shown in Figure 4.1. Throughout this chapter, it is assumed that V_2 is a node cut.

We will refer frequently to the total demand between the partititions, resp. within one node set. For that matter, we define

$$\mathfrak{d}_{pq} := \sum_{k \in \mathcal{K}} U^k \mathfrak{d}(V_p, V_q) \quad \forall p, q = 1, 2, 3 :\ p \leqslant q\ . \tag{4.8}$$

Notice that, following the assumptions for this section, $\mathcal{K} = \{k\}$ and $U^k = 1$, hence

$$\mathfrak{d}_{pq} = \mathfrak{d}(V_p, V_q) \quad \forall p, q = 1, 2, 3 :\ p \leqslant q\ . \tag{4.9}$$

A key property of the node designs that can be installed on V_2 is the capacity they can simultanously support on both sides. This value gives the maximum flow that can be sent from V_1 to V_3 over some $v \in V_2$, if a node design $d \in \mathcal{D}(v)$ is used for v:

Definition 4.4 (Effective side capacity)
Let $v \in V_2$ and $d \in \mathcal{D}(v)$ be an available node design on v. The value

$$\mathrm{esc}(v, d) := \max_{x \in X_{\mathrm{HC,R}} : x_{d,v} = 1} \ \min_{pq \in \{12, 23\}} \ \sum_{e \in \delta(v) \cap E_{pq}} \left(C_e^0 + \sum_{\ell \in \mathcal{L}(e)} C^\ell x_{\ell,e} \right) \tag{4.10}$$

is called effective side capacity *of* d *on* v. *If this value is not defined because* $\{x \in X_{\mathrm{HC,R}} \mid x_{d,v} = 1\} = \varnothing$, *we set* $\mathrm{esc}(v, d) := -\infty$.

The following definition is useful when extending properties like the effective side capacity to node design assignments on all nodes in V_2. We will exploit properties of such an assignment to formulate valid inequalities.

Definition 4.5 (Node cut design band)
Define

$$\mathcal{B} := \{B : V_2 \to \mathcal{D} \mid B(v) \in \mathcal{D}(v)\ \forall v \in V_2\} \tag{4.11}$$

to be the set of all mappings that assign each node in the cut V_2 *an available node design. The mappings* $B \in \mathcal{B}$ *are called* node cut design bands, *or* bands *in short.*

For a band $B \in \mathcal{B}$, those node designs that are available on a node $v \in V_2$ and have a larger switching capacity than the chosen design $B(v)$ will play important roles. Therefore we define according node design sets by

$$\begin{aligned} &H_B : V_2 \to 2^{\mathcal{D}} , \\ &H_B(v) := \left\{ d \in \mathcal{D}(v) \mid C^d > C^{B(v)} \right\} , \end{aligned}$$

which are named as follows:

Definition 4.6 (Heavy node design)
Let $B \in \mathcal{B}$ *and* $v \in V_2$. *A node design* $d \in \mathcal{D}(v)$ *is called* heavy *on* v, *if* $d \in H_B(v)$.

Notice that a node design $d \in \mathcal{D}$ can be heavy on one node and nonheavy on another. The facet-defining inequality we are about to introduce states that if the node designs of some band do not provide a large enough switching capacity to establish a feasible routing, at least one heavy node design must be installed. For that matter, an important operation on bands is updating the node design of one particular node to another design of higher switching capacity. The resulting band is introduced in the following definition:

Definition 4.7 (One-node extension band)
Let $B_1, B_2 \in \mathcal{B}$ *be two bands.* B_2 *is called* one-node extension *of* B_1, *if there exists a node* $v \in V_2$ *such that* $B_2(v)$ *is heavy w.r.t.* B_1, *and* B_1 *equals* B_2 *on the other nodes, i.e.,*

$$B_2(v) \in H_{B_1}(v), \quad B_2(w) = B_1(w) \quad \forall w \in V_2 \setminus \{v\} .$$

To simplify notation, we use B and H_B as indices in the following manner: If x is a vector or mapping indexed by components and graph elements, e.g., $x = (x_{c,g})_{c \in \mathcal{C}, g \in \mathcal{G}}$, we denote by $x(B)$ and $x(H_B)$ the following:

$$\begin{aligned} x(B) &:= \sum_{v \in V_2} x_{B(v),v} , \\ x(H_B) &:= \sum_{v \in V_2} \sum_{d \in H_B(v)} x_{d,v} . \end{aligned}$$

We will use this notation scheme wherever the interpretation seems nonambiguous, as for

$$\begin{aligned} C(B) &:= \sum_{v \in V_2} C^{B(v)} , \\ \mathrm{esc}(B) &:= \sum_{v \in V_2} \mathrm{esc}(v, B(v)) , \end{aligned}$$

which describe the total capacity and effective side capacity of B. Another important property of a band B deals with supported link designs. Those that require heavy node designs in both end nodes when installed on an edge in E_{22} need special treatment:

Definition 4.8 (Chunky link design)
Let $B \in \mathcal{B}$ be a band and $e = uw \in E_{22}$ be an edge. A link design $\ell \in \mathcal{L}(e)$ is called chunky *on e, if it can only be installed if heavy designs are installed on both end-nodes of e, i.e.,*

$$\forall x \in X_{\mathrm{HC}} : x_{\ell,e} = 1 \quad \forall v \in \{u,w\},\ d \in \mathcal{D}(v) : \quad x_{d,v} = 1 \Longrightarrow d \in H_B(v) \ . \tag{4.12}$$

Analoguous to the notation for H_B, we define

$$L_B : E_{22} \to 2^{\mathcal{L}} \ , \quad L_B(e) := \{\ell \in \mathcal{L} \mid \ell \text{ is chunky on } e\} \ . \tag{4.13}$$

We will frequently use mappings that select sets of link designs for the edges in E_{22}. For that matter, three notations are used: Let $L, L' : E_{22} \to 2^{\mathcal{L}}$ be two such mappings. We denote

$$L \subseteq L' \quad :\Longleftrightarrow \quad L(e) \subseteq L'(e) \ \ \forall e \in E_{22} \ . \tag{4.14}$$

Let $E' \subseteq E_{22}$ be a set of edges in E_{22}. We define

$$L{\restriction}E' : E_{22} \to 2^{\mathcal{L}} \ , \quad (L{\restriction}E')(e) := \begin{cases} L(e) & \text{if } e \in E' \\ \varnothing & \text{if } e \notin E' \end{cases} \ . \tag{4.15}$$

Notice that $L{\restriction}E'$ is similar to the restriction of L to E' by the usual definition, it merely defines the undefined values to be $\varnothing$. The support of L is defined in the canonical way as

$$\mathrm{supp}(L) := \{e \in E_{22} \mid L(e) \neq \varnothing\} \ . \tag{4.16}$$

To further simplify notation, we denote by $x(L)$ the sum of coefficients of x as specified by L, analoguous to the definition $x(H_B)$ above, i.e.,

$$x(L) := \sum_{e \in E_{22}} x_{L(e),e} \ .$$

4.1.2 Inequality

We now introduce a class of inequalities that are valid for $X_{\mathrm{HC,R}}$:

Proposition 4.9 *Let $B \in \mathcal{B}$ be a band. Let H_B and L_B be as defined above. Let $T \subseteq E_{22}$ be a (possibly empty) forest in (V_2, E_{22}). Let $L : E_{22} \to 2^{\mathcal{L}}$ with $L \subseteq L_B$. Let $b := 2\mathfrak{d}_{22}+2\mathfrak{d}_{13}+\mathfrak{d}_{12}+\mathfrak{d}_{23}$. If $C(B) < b$, then*

$$x(H_B) - x(L{\restriction}T) \geqslant 1 \tag{4.17}$$

is a valid inequality for $X_{\mathrm{HC,R}}$.

Proof Let $x \in X_{\mathrm{HC,R}}$, thus either $x(L{\restriction}T) = 0$ or $x(L{\restriction}T) \geqslant 1$. These two cases are dealt with separately:

- Assume $x(L{\restriction}T) = 0$. Because $x \in X_{\mathrm{HC,R}}$, a demand routing exists. Because $E_{13} = \varnothing$, demands between V_1 and V_3 are fulfilled by sending flow over edges in E_{12} and over E_{23}. V_1-V_2-demands and V_2-V_3-demands send flow over E_{12} and E_{23}, respectively. This yields that at least a capacity of $\mathfrak{d}_{12} + \mathfrak{d}_{13}$ must be installed on E_{12} and $\mathfrak{d}_{13} + \mathfrak{d}_{23}$ on E_{23}. These capacities must be supported by the node designs installed on V_2. Additionally, demands between nodes in V_2 need a capacity installation that further debits the switching capacity of the node designs at such a demand's source and target. In summary, the node designs installed on V_2 need to provide a switching capacity of at least $\mathfrak{d}_{12} + \mathfrak{d}_{13} + \mathfrak{d}_{13} + \mathfrak{d}_{23} + \mathfrak{d}_{22} + \mathfrak{d}_{22} = b$. Because $C(B) < b$, at least one node design with a larger switching capacity must be present, and hence $x(H_B) \geqslant 1$.

- Assume $x(L\restriction T) \geqslant 1$. Let $T_x \subseteq T$ consist of those edges $e \in T$ on which a link design from $L\restriction T$ is installed, i.e., $\sum_{\ell\in L\restriction T(e)} x_{\ell,e} \geqslant 1$. Because each edge gets at most one link design installed, $x(L\restriction T) = |T_x|$. Because T_x, being a subset of a forest, is itself a forest in (V_2, E_{22}), $|V(T_x)| \geqslant |T_x| + 1$. By construction of L and L_B, none of the nodes in $V(T_x)$ can have a nonheavy node design installed, and thus

$$x(H_B) \quad \geqslant \quad |V(T_x)| \quad \geqslant \quad |T_x| + 1 \quad = \quad x(L\restriction T) + 1\,,$$

which proves the claim. □

Among the inequalities (4.17) are some that are not only valid for $X_{\mathrm{HC,R}}$, but also facet-defining for $\mathrm{conv}X_{\mathrm{HC,R}}$. The following proposition identifies some of these:

Proposition 4.10 *Let $X_{\mathrm{HC,R}}$ be trivially bounded, and have designs sufficient for emanating demands. Let $B \in \mathcal{B}$ be a band with H_B and L_B as above. Let $T \subseteq E_{22}$ be a maximal forest in $(V_2, \mathrm{supp}(L_B))$. Let $b := 2\mathfrak{d}_{22} + 2\mathfrak{d}_{13} + \mathfrak{d}_{12} + \mathfrak{d}_{23}$ and $b^* := 2\mathfrak{d}_{22} + \mathfrak{d}_{13} + \mathfrak{d}_{12} + \mathfrak{d}_{23}$. If*

(i) G, (V_1, E_{11}), and (V_3, E_{33}) are biconnected, and (V_2, E_{22}) is connected,

(ii) the node cut V_2 is minimal w.r.t. set inclusion,

(iii) $C(B) < b$,

(iv) $\mathrm{esc}(B') \geqslant b^*$ for every one-node extension B' of B,

then (4.17) *defines a facet of* $\mathrm{conv}X_{\mathrm{HC,R}}$.

Proof Validity of (4.17) follows from Proposition 4.9. Let $P := \mathrm{conv}X_{\mathrm{HC,R}}$ and

$$\mathcal{F} := \{x \in P \mid x(H_B) - x(L_B\restriction T) = 1\} \tag{4.18}$$

be the face of P that is induced by (4.17). We prove $\mathcal{F}$ is in fact a facet in three steps: First, $\mathcal{F} \neq \varnothing$ is shown by presenting points in $\mathcal{F}$. Second, we show that any face containing $\mathcal{F}$ is either P or $\mathcal{F}$ itself. Third, we show that $P \neq \mathcal{F}$ by presenting a point in $P\backslash\mathcal{F}$.

Simple Facts: Notice that $\hat{\ell}_e \in L_B(e)$ for each $e \in E_{22}$ and $\hat{d}_v \in H_B(v)$ for each $v \in V_2$, because $X_{\mathrm{HC,R}}$ is trivially bounded and hence these design have capacities larger than b, which is more than the total switching capacity $C(B)$ of the band.

T is in fact a spanning tree on (V_2, E_{22}), because (V_2, E_{22}) is connected and there exists a chunky link design on every edge in E_{22}. Furthermore, each node $v \in V_2$ is adjacent to both V_1 and V_3: If there were no edge $e = uv \in E$ with, say, $u \in V_1$, then V_2 would still be a node cut if v was moved to V_3, which would contradict V_2's minimality.

Special Points $\hat{\mathbf{x}}$: We now construct special points $\hat{x} \in \mathcal{F}$, for two reasons: First, this proves $\mathcal{F}$ is nonempty. Second, these points will serve as starting point for exchange operations in subsequent steps. For any three nodes $v_p \in V_p$ $(p = 1, 2, 3)$ that define a path $(v_1, v_1v_2, v_2, v_2v_3, v_3)$, i.e., both $e_{12} := v_1v_2$ and $e_{23} := v_2v_3$ are edges in E, we define a point $\hat{x} = \hat{x}(e_{12}, v_2, e_{23}) \in X_{\mathrm{HC,R}}$ as follows, see the left picture of Figure 4.2 for a visualization:

- On v_2 and all nodes in V_1 and V_3, the large node designs $\hat{d}$ are installed, i.e., $\hat{x}_{\hat{d}_v,v} := 1$ for $v \in V_1 \cup V_3 \cup \{v_2\}$.
- On e_{12}, e_{23}, and all edges in E_{11} and E_{33}, the large link designs $\hat{\ell}$ are installed: $\hat{x}_{\hat{\ell}_e,e} := 1$ for $e \in E_{11} \cup E_{33} \cup \{e_{12}, e_{23}\}$.
- The nodes in $V_2\backslash\{v_2\}$ get the band node design $B(u)$ installed: $\hat{x}_{B(u),u} := 1$ for $u \in V_2\backslash\{v_2\}$.

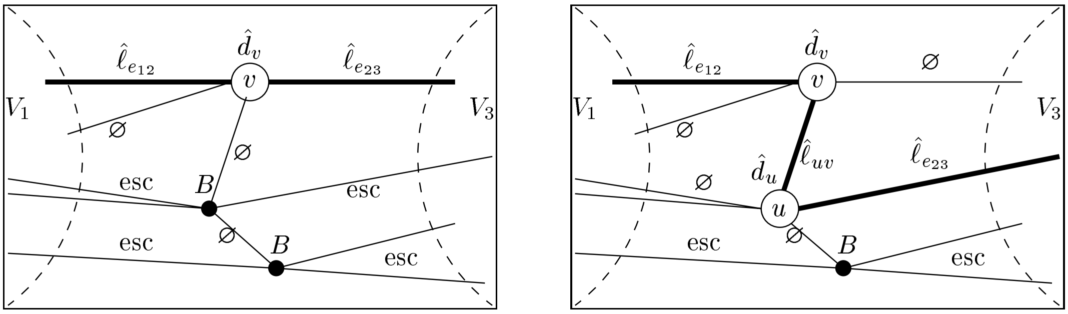


Figure 4.2: Points $\hat{x}$ and $\tilde{x}$. *Left: Points $\hat{x}$ establish a high-capacitated $[V_1, V_3]$-path using one node in V_2. Remaining V_2-nodes use the band node design, and edges in E_{22} get no link design at all. Right: $\tilde{x}$ uses two nodes for the $[V_1, V_3]$-path and install a chunky link design on an edge in the tree $T \subseteq E_{22}$.*

- For each node $u \in V_2 \setminus \{v_2\}$, the links in $\delta(u)$ get a link design installation that attains the maximum in calculation of $\operatorname{esc}(u, B(u))$ according to (4.10) while installing no capacity on $\delta(u) \cap E_{22}$. Such an installation exists because it is possible to omit link designs on $\delta(u) \cap E_{22}$ without changing feasibility or objective value in (4.10).
- On each link, the single commodity set capacity component $k \in \mathcal{K}$ gets installed up to the maximum, that is, $\hat{x}_{k,e} := y_{\hat{x}}(e)$ for all $e \in E$.
- There exists a feasible module installation, because all nodes either have their large node design $\hat{d}_v$ installed, or are band node designs with esc-based link design installation. Thus extending $\hat{x}$ by modules such that $\hat{x} \in X_{\mathrm{HC}}$ is possible.
- All coefficients of $\hat{x}$ that are not listed above are now implicitly fixed to either 0 or 1.

Now that $\hat{x} \in X_{\mathrm{HC}}$, it is obvious that $\hat{x} \in X_{\mathrm{HC,R}}$ as well: All over V_1 and V_3 as well as on a $[V_1, V_3]$-path there is enough capacity to route all demands. The nodes in $V_2 \setminus \{v\}$ have enough capacity on each side to route their emanating demand. Because $\hat{x}$ has one heavy node design and no chunky link design installed, $\hat{x} \in \mathcal{F}$.

For easier reference of these points, we sometimes omit some or all of the parameters e_{12}, v_2, and e_{23} to denote the set of all those points that have the specified parameters and any value for the others, e.g., $\hat{x}(\cdot, v_2, \cdot)$ denotes the set $\{\hat{x}(e', v_2, e'') \mid e' \in \delta(v_2) \cap E_{12},\ e'' \in \delta(v_2) \cap E_{23}\}$.

Containing Face and Claim: Let $\tilde{\alpha}$ be a vector of appropriate dimension and $\tilde{\beta}$ be a real number such that

$$\tilde{\alpha}^{\mathsf{T}} x \geqslant \tilde{\beta} \tag{4.19}$$

is a valid inequality for $X_{\mathrm{HC,R}}$ that fulfills $\mathcal{F}$ at equality, i.e.,

$$\tilde{\alpha}^{\mathsf{T}} x = \tilde{\beta} \quad \forall x \in \mathcal{F} .$$

We claim that (4.19) is a positive multiple of (4.17) plus a linear combination of equalities that are valid for $X_{\mathrm{HC,R}}$. Notice that for every component $c \in \mathcal{C}$ and graph element $g \in \mathcal{G}$ for which $x_{c,g}$ is explicitely or implicitely fixed to some value ξ, we can safely neglect its coefficient in α, because we can obtain any desired value by adding scaled variants of the valid equality $x_{c,g} = \xi$. Consequently, we will ignore such variables in the remaining proof.

For $p \in \{1, 2, 3\}$, let $V_p^= \subseteq V_p$ be the set of those nodes in $v \in V_p$ on which a node design must be installed, i.e., for which

$$\sum_{d \in \mathcal{D}(v)} x_{d,v} = 1 \tag{4.20}$$

is a valid equality for $X_{\mathrm{HC,R}}$. We obtain another equality $\alpha^{\mathsf{T}}x = \beta$ that is still valid for $\mathcal{F}$ by adding linear combinations of (4.20) such that

$$\alpha_{B(v),v} = 0 \quad \forall v \in V_2^{=} \quad \text{and} \tag{4.21}$$
$$\alpha_{\hat{d}_v,v} = 0 \quad \forall v \in V_1^{=} \cup V_3^{=} \tag{4.22}$$

hold.

In the following steps, we inspect all other coefficients of α, showing that $\alpha^{\mathsf{T}}x \geqslant \beta$ is a positive multiple of (4.17), which in turn yields that $\alpha^{\mathsf{T}}x \geqslant \beta$ induces F, giving that (4.19) induces F as well, which finally leads to the proof that (4.17) is facet-defining.

Modules: Let $v_1 \in V_1 \cup V_3$ and $v_2 \in V_2$, and let v be any of the two. Let $m \in \mathcal{M}(v)$ be a module. Let $\hat{x}$ be any of the points in $\hat{x}(\cdot, v_2, \cdot)$. Notice that there is the node design $\hat{d}_v$ installed on v, and, because $X_{\mathrm{HC,R}}$ is trivially bounded, there is an alternate module installation on v for $\hat{x}$ that leaves space to install each other module one additional time. Hence, it is possible to change the module installation on v such that adding one module m does not leave $\mathcal{F}$. The latter, together with the arbitrary choice of v and m gives

$$\alpha_{m,v} = 0 \quad \forall v \in V,\ m \in \mathcal{M}(v)\ . \tag{4.23}$$

Commodity Set Capacity Components on $\mathbf{E \setminus E_{22}}$: Let $e \in E \setminus E_{22}$. Let $\hat{x}$ be one of the points in $\hat{x}(\cdot, \cdot, \cdot)$ with $\hat{x}_{\hat{\ell}_e,e} = 1$. Such a point always exists: If $e \in E_{11} \cup E_{33}$, any of the points will do. If $e \in E_{12} \cup E_{23}$, it is incident to some $v \in V_2$, and there exists a point in $\hat{x}(e, v, \cdot)$ or $\hat{x}(\cdot, v, e)$.

Let $k \in \mathcal{K}$ be the commodity set capacity component. The construction of $\hat{x}$ implies that $\hat{x}_{k,e} \geqslant C^{\hat{\ell}_e}$, which is at least twice the flow over e in any routing because $X_{\mathrm{HC,R}}$ is trivially bounded, and at least 1 because demand exists (otherwise, $b = 0$, which would contradict existence of the chosen band). It is thus possible to reduce $\hat{x}_{k,e}$ by one without leaving $\mathcal{F}$. Hence,

$$\alpha_{k,e} = 0 \quad \forall e \in E \setminus E_{22},\ k \in \mathcal{K}\ . \tag{4.24}$$

Link Designs on $\mathbf{E_{11} \cup E_{33}}$: Let $e = uv \in E_{11}$ and $\ell \in \mathcal{L}(e)$. Let $\hat{x}$ be any of the points in $\hat{x}(\cdot, \cdot, \cdot)$. Because (V_1, E_{11}) is biconnected, there exists a $[u, v]$-path P in (V_1, E_{11}) that does not use e. Starting with $\hat{x}$, all flow over e can be rerouted using P, effectively removing all flow from e. Now we can decrease $\hat{x}_{k,e}$ to zero for $k \in \mathcal{K}$. It is then possible to exchange the installed link design $\hat{\ell}_e$ by ℓ, including necessary module changes in u and v. The resulting point x has ℓ installed on e, but no commodity set capacity components, and is still contained in $\mathcal{F}$. Hence, $x_{\ell,e}$ can be set to zero without leaving $\mathcal{F}$.

Repeating this argumentation with (V_3, E_{33}) for (V_1, E_{11}), we get

$$\alpha_{\ell,e} = 0 \quad \forall e \in E_{11} \cup E_{33},\ \ell \in \mathcal{L}(e)\ . \tag{4.25}$$

Node Designs on $\mathbf{V_1 \cup V_3}$: Let $v \in V_1$ and $d \in \mathcal{D}(v)$ be a node design. We distinguish two cases:

1. $V_1 = \{v\}$: Let $\hat{x}$ be any point in $\hat{x}(\cdot, \cdot, \cdot)$. If $v \notin V_1^{=}$ (hence $V_1^{=} = \varnothing$ and $\mathfrak{d}(v) = 0$), it is possible to remove the commodity set capacity component and all link designs on $\delta(v)$, as well as all modules on v. In the resulting point, the node design d can be installed and removed on v without changing anything else or leaving $\mathcal{F}$, hence $\alpha_{d,v} = 0$.

   On the other hand, if $v \in V_1^{=}$, $\hat{x}$ can be modified such that d is installed on v. It is then possible to upgrade d to $\hat{d}_v$ by adjusting the module installation on v. Because of (4.22) and the modules' coefficients being zero, $\alpha_{d,v} = 0$.

2. $|V_1| > 1$: Let $e = uv \in \delta(v) \cap E_{11}$. Such an edge exists because (V_1, E_{11}) is connected. There exists a link design $\ell \in \mathcal{L}(e)$ that is supported by d and has a large enough capacity to carry the emanating demand of v.

   Let $g_1 \in \Gamma_1 \setminus \{v\}$. Such a node exists because G is biconnected and thus $|\Gamma_1| \geqslant 2$. Let $g_2 \in \Gamma(g_1) \cap V_2$ and $g_3 \in \Gamma(g_2) \cap V_3$. Now $(g_1, g_1g_2, g_2, g_2g_3, g_3)$ is a $[V_1, V_3]$-path that does not use v. Let $\hat{x} = \hat{x}(g_1g_2, g_2, g_2g_3)$. It is now possible to modify $\hat{x}$ such that d is installed on v and ℓ on e, because all flow that passes through v can be rerouted to bypass it (because (V_1, E_{11}) is biconnected and $\Gamma_1 \setminus \{v\}$ is nonempty).

   Similar to the first case, it is now possible to discard d if $v \notin V_1^=$ or upgrade it to $\hat{d}_v$ if $v \in V_1^=$. In both situations, $\alpha_{d,v} = 0$ holds.

Repeating this argument for V_3 leads to

$$\alpha_{d,v} = 0 \quad \forall v \in V_1 \cup V_3,\ d \in \mathcal{D}(v)\ . \tag{4.26}$$

Special Points $\tilde{\mathbf{x}}$: Another set of points is needed for the remaining coefficients of α. Let $(g_1, g_1v, v, vu, u, ug_3, g_3)$ be a $[g_1, g_3]$-path in G with $g_1 \in V_1$, $g_3 \in V_3$, $u \in V_2$, $v \in V_2$ and $vu \in T \subseteq E_{22}$. We define a point $\tilde{x} := \tilde{x}(g_1v, v, vu, u, ug_3)$ as follows, see the right picture of Figure 4.2 for a visualization:

- On u, v, and all nodes in V_1 and V_3, the large node designs $\hat{d}$ are installed, i.e., $\hat{x}_{\hat{d}_w,w} := 1$ for $w \in V_1 \cup V_3 \cup \{u,v\}$.
- On g_1v, vu, ug_3, and all edges in E_{11} and E_{33}, the large link designs $\hat{\ell}$ are installed: $\hat{x}_{\hat{\ell}_e,e} := 1$ for $e \in E_{11} \cup E_{33} \cup \{g_1v, vu, ug_3\}$.
- The nodes in $V_2 \setminus \{v,u\}$ get the band node designs $B(\cdot)$ installed: $\hat{x}_{B(w),w} := 1$ for each $w \in V_2 \setminus \{v,u\}$.
- For each node $w \in V_2 \setminus \{v,u\}$, the links in $\delta(w)$ get a link design installation that attains the maximum in calculation of $\mathrm{esc}(w, B(w))$, by the same means as in construction of $\hat{x}$.
- On each link, the single commodity set capacity component $k \in \mathcal{K}$ gets installed up to the maximum, i.e., $\tilde{x}_{k,e} := y_{\tilde{x}}(e)$ for all $e \in E$.
- There exists a feasible module installation, as for $\hat{x}$, such that $\tilde{x} \in X_{\mathrm{HC}}$.
- All other coefficients of $\tilde{x}$ are now implicitely fixed to either 0 or 1.

Obviously, $\tilde{x} \in X_{\mathrm{HC,R}}$ by a similar argument as for $\hat{x}$. There is again a high-capacitated $[V_1, V_3]$-path for the major traffic and band node designs that only need to transmit the local emanating demands. The link design $\hat{\ell}_{vu}$ is chunky, because its capacity is at least twice the total demand, which is more than b, hence more than the total switching capacity of the band node designs, and thus more than the switching capacity of a single band node design. Because $\tilde{x}$ has two heavy node designs and one chunky link design installed, $\tilde{x} \in \mathcal{F}$.

We again use simplified notations for these points by omitting some or all parameters, analoguous to $\hat{x}$.

Commodity Set Capacity Components on $\mathbf{E_{22}}$: Let $e \in E_{22}$. Let $k \in \mathcal{K}$ be the commodity set capacity component. Notice that $\hat{\ell}_e \in L_B(e)$, therefore a $\tilde{x}$ in $\tilde{x}(\cdot, \cdot, e, \cdot, \cdot)$ exists. Because $\tilde{x}_{k,e} \geqslant C^{\hat{\ell}_e}$ and the latter is more than the flow over e, it is possible to reduce $\tilde{x}_{k,e}$ by one without leaving $\mathcal{F}$. Hence,

$$\alpha_{k,e} = 0 \quad \forall e \in E_{22},\ k \in \mathcal{K}\ . \tag{4.27}$$

Link Designs on $\mathbf{E_{12} \cup E_{23}}$: Let $e = wu \in E_{12}$ and $\ell \in \mathcal{L}(e)$. Because T is a spanning tree in (V_2, E_{22}), there exists an edge $vu \in \delta(u) \cap E_{22}$. Let $\tilde{x}$ be any of the points in $\tilde{x}(\cdot, v, vu, u, \cdot)$. Because w and u have the large node designs $\hat{d}_w$ and $\hat{d}_u$ installed, it is possible to change module installations on w and u such that ℓ can be installed on e. This leads to a point $x \in \mathcal{F}$ with $x_{\ell,e} = 1$, where ℓ can be removed from e without leaving $\mathcal{F}$. A similar argument holds if $e \in E_{23}$, hence

$$\alpha_{\ell,e} = 0 \quad \forall e \in E_{12} \cup E_{23},\ \ell \in \mathcal{L}(e)\ . \tag{4.28}$$

Nonheavy Node Designs on $\mathbf{V_2}$: Let $v \in V_2$ and $d \in \mathcal{D}(v) \setminus H_B(v)$. Let $u \in V_2$, $u \neq v$ be another node in V_2 and $\hat{x}$ be any of the points $\hat{x}(\cdot, u, \cdot)$. Let $e \in \delta(v) \cap (E_{12} \cup E_{23})$.

Notice that the $[V_1, V_3]$-path used in the construction of $\hat{x}$ has enough capacity to carry the complete flow between V_1 and V_3. Hence, it is possible to exchange the band node design on v by d and install a link design on e that suffices to carry the emanating demand of v. Using a suitable module and commodity set capacity component installation, the resulting point x is still contained in $\mathcal{F}$. Hence, $\alpha_{d,v} = \alpha_{B(v),v}$. If $v \in V_2^{=}$, $\alpha_{B(v),v} = 0$ by the above construction. If $v \notin V_2^{=}$, d can be removed together with link designs on incident edges because there is no emanating demand and all other flow was rerouted not to use v. Hence,

$$\alpha_{d,v} = 0 \quad \forall v \in V_2,\ d \in \mathcal{D}(v) \setminus H_B(v)\ . \tag{4.29}$$

Heavy Node Designs on $\mathbf{V_2}$: Let $v \in V_2$ and $d \in H_B(v)$. Let $B' : V_2 \to \mathcal{D}$ be the band defined by

$$B'(u) := \begin{cases} d & \text{if } u = v \\ B(u) & \text{if } u \neq v \end{cases}\ .$$

Let $\hat{x}$ be any of the points $\hat{x}(\cdot, v, \cdot)$. On v, the node design d is now installed, with the link designs attaining the maximum in calculation of $\operatorname{esc}(v, d)$ installed on $\delta(v) \cap (E_{12} \cup E_{23})$ and no link designs on $\delta(v) \cap E_{22}$. A matching module installation on v and $\Gamma(v)$ is installed as well, and the commodity set capacity component is raised to the maximum, resulting in point $x \in X_{\mathrm{HC}}$.

By construction of B', $\operatorname{esc}(B') \geqslant b^*$ holds. Hence, there is a feasible routing that routes all demands between nodes in V_2 through one shore only, regardless which one that is. Thus $x \in X_{\mathrm{HC,R}}$. Because x uses one heavy node design and no chunky link design, $x \in \mathcal{F}$. On v, it is now possible to replace d by $\hat{d}_v$. Besides module installation changes, nothing more has to be changed. Let $A_w := \alpha_{\hat{d}_w,w}$ for each $w \in V_2$. Because d and v were chosen arbitrarily,

$$\alpha_{d,v} = A_v \quad \forall v \in V_2,\ d \in H_B(v)\ . \tag{4.30}$$

Link Designs on $\mathbf{E_{22}}$: Let $e = uv \in E_{22}$ and $\ell \in \mathcal{L}(e)$. Three case have to be considered:

1. $\ell \notin L_B(e)$: ℓ can be installed on e using a nonheavy node design d on at least one of its end-nodes, w.l.o.g. u, because it is not chunky. We start with a point $\hat{x}$ in $\hat{x}(\cdot, v, \cdot)$. It is possible to change $\hat{x}$ to have d installed on u and ℓ on e, e.g., by routing the emanating demand of u over e and v. The new point is still in $\mathcal{F}$, and allows to exchange ℓ on e by some other link design ℓ' on an edge $e' \in \delta(u) \cap (E_{12} \cup E_{23})$. Because $\alpha_{\ell',e'} = 0$ and all module coefficient on u are zero as well, $\alpha_{\ell,e} = 0$.

2. $\ell \in L_B(e), e \notin T$: Because T is a spanning tree in (V_2, E_{22}), there is a set $Z \subseteq T$ such that $Z \cup \{e\}$ defines a circle in (V_2, E_{22}). Let $\hat{x}$ be any of the points $\hat{x}(\cdot, v, \cdot)$. Construct a point x from $\hat{x}$ by installing $\hat{d}_u$ on each $u \in V_2(Z)$ and changing module installation accordingly. It is now possible to add ℓ on e by adjusting modules on u and v, without leaving $\mathcal{F}$. Hence, $\alpha_{\ell,e} = 0$.

3. $\ell \in L_B(e), e \in T$: Let $\hat{x}$ be a point in $\hat{x}(\cdot, v, \cdot)$. Installing ℓ on e is possible if $\hat{d}_u$ is installed on u, thereby replacing the band node design on u, and the module installation on u is updated as well. The resulting point is still in $\mathcal{F}$. Because $\alpha_{m,u} = 0$ for all $m \in \mathcal{M}(u)$ and $\alpha_{B(u),u} = 0$, we get $\alpha_{\ell,e} = -\alpha_{\hat{d}_u,u} = -A_u$.

Altogether, we get

$$\alpha_{\ell,e} = \begin{cases} -A_v & \text{if } \ell \in L_B(e), e \in T \\ 0 & \text{otherwise} \end{cases} \qquad \forall e = uv \in E_{22},\ \ell \in \mathcal{L}(e)\ . \tag{4.31}$$

Notice that u and v are completely interchangable in the preceeding paragraph. Hence, $A_v = A_u$. Because T connects all nodes in V_2 and each edge in $e' \in T$ has at least one chunky link design, namely $\hat{\ell}_{e'}$, $A_v = A_w$ for all $w \in V_2$. Let A denote this value, then

$$\alpha_{\ell,e} = -A \quad \forall e \in T,\ \ell \in L_B(e), \quad \alpha_{d,v} = A \quad \forall v \in V_2,\ d \in H_B(v)\ . \tag{4.32}$$

Conclusion The above paragraphs cover all relevant coefficients: Commodity set capacity components by (4.24) and (4.27), modules by (4.23), link designs by (4.25), (4.28), (4.31), and (4.32), and node design by (4.26), (4.29), (4.30), and (4.32). Hence, $\alpha^\mathsf{T} x \geqslant \beta$ reads

$$Ax(H_B) - Ax(L_B \restriction T) \geqslant \beta\ . \tag{4.33}$$

Let $\hat{x}$ be any of the points $\hat{x}(\cdot,\cdot,\cdot)$. Construct a point x' from $\hat{x}$ by installing the large node design $\hat{d}_v$ on each node $v \in V_2$, adjusting module installations as needed. Because $\hat{d}_v \in H_B(v)$ for each $v \in V_2$ and $|V_2| \geqslant 2$ because G is biconnected, x' is contained in $X_{\mathrm{HC,R}} \setminus \mathcal{F}$. Therefore, $\mathcal{F} \neq \mathrm{conv} X_{\mathrm{HC,R}}$. Because $\alpha^\mathsf{T} x \geqslant \beta$ is a valid inequality for $X_{\mathrm{HC,R}}$, $\alpha^\mathsf{T} x' \geqslant \beta$ and $\alpha^\mathsf{T} \hat{x} = \beta$, which contradicts $A < 0$. Therefore, either $A = 0$ or $A > 0$. In the former case the face induced by $\alpha^\mathsf{T} x \geqslant \beta$ is P, in the latter case $\alpha^\mathsf{T} x \geqslant \beta$ is a positive multiple of (4.17), hence induces $\mathcal{F}$. This completes the proof that (4.17) is in fact facet-defining. □

One precondition of Proposition 4.10 seems unfortunate, namely requiring (V_2, E_{22}) to be connected. Telecommunication networks are typically sparse, and it is unlikely that there are many edges in E_{22}, if V_2 is a minimal node cut.

A situation where the proof breaks is the following: Let $v \in V_2$ be a node of the cut which is not adjacent to any other node in V_2, i.e., $\delta(v) \cap E_{22} = \varnothing$. Assume v is connected to V_1 over just one edge e, hence $\{e\} = \delta(v) \cap E_{12}$. It can be seen that (4.17) may be dominated by another inequality, which uses the variables for link designs on e instead of the heavy node designs on v. If $E_{22} = \varnothing$, this inequality turns out to be a metric inequality on some edges in $E_{12} \cup E_{23}$.

If every node in $v \in V_2$ has a neighbour in V_2, i.e., $\delta(v) \cap E_{22} \neq \varnothing$, the proposition holds. The proof needs to be updated, because it is no longer obvious that the values A_w are equal for all nodes $w \in V_2$. For nodes within the same connected component of (V_2, E_{22}), the given argumentation holds. That these values are equal among different component can be seen using one of the points in $\hat{x}(\cdot, w, \cdot)$ for each $w \in V_2$ and rerouting flow within (V_1, E_{11}) and (V_3, E_{33}).

The proposition also holds if there exists a isolated node $v \in V_2$ in the sense that $\delta(v) \cap E_{22} = \varnothing$ that has at least two incident edges per side, i.e., $|\delta(v) \cap E_{pq}| \geqslant 2$ for $pq \in \{12, 23\}$. The proof breaks in the step showing that coefficients for link designs on E_{12} and E_{23} are zero, because the point $\tilde{x}$ that is used in the step might be nonexisting. By using a rerouting path within (V_1, E_{11}) resp. (V_3, E_{33}) and switching capacities between the edges in $\delta(v) \cap E_{pq}$ for $pq \in \{12, 23\}$, an alternative argumentation can be drawn.

4.1.3 Separation

In this section, we explore separation issues for the inequality class introduced above for fixed node cut V_2. Let $x \geqslant 0$ be a point of appropriate dimension for $X_{\mathrm{HC,R}}$. We attempt to find an inequality (4.17) that is valid for $X_{\mathrm{HC,R}}$ but not fulfilled by x. For now, we assume L_B can be determined for every band, and use L_B only when searching for an inequality.

For a band $B \in \mathcal{B}$, define $T_x(B)$ to be a maximally-weighted forest suitable for Proposition 4.9, that is a forest T in $(V_2, \mathrm{supp}(L_B))$ that maximizes $x(L_B \upharpoonright T)$. Define

$$x_B(e) := \begin{cases} x(L_B \upharpoonright \{e\}) & \text{if } e \in \mathrm{supp}(L_B) \\ 0 & \text{otherwise} \end{cases}$$

Furthermore, define

$$z(B) := x(L_B \upharpoonright T_x(B)) - x(H_B) \ . \tag{4.34}$$

Notice that (4.17) now reads $z(B) \leqslant -1$. The separation problem of finding a maximally violated inequality thus is:

$$\max\{\ z(B) \mid C(B) < b\ , \quad B \in \mathcal{B} \quad \} \ . \tag{4.35}$$

If this problem has an optimal solution B^* with $z(B^*) > -1$, a maximally violated inequality is identified. If it has no solution or its value is $\leqslant -1$, no violated inequality exists.

We are not aware of an algorithm to solve this problem except for enumeration, and hence provide a heuristic instead. For that matter, we first give a few observations on the problem structure of (4.35).

Let $B_1 \in \mathcal{B}$ be a band and $B_2 \in \mathcal{B}$ be a one-node extension of B_1. We are interested in the objective function gain in (4.35) if B_2 is chosen instead of B_1, that is the value

$$\begin{aligned} z(B_1, B_2) &:= z(B_2) - z(B_1) \\ &= \underbrace{x(L_B \upharpoonright T_x(B_2)) - x(L_B \upharpoonright T_x(B_1))}_{=:t_1} + \underbrace{x(H_{B_1}) - x(H_{B_2})}_{=:t_2} \ . \end{aligned}$$

Given that $H_{B_1}(v) \supseteq H_{B_2}(v)$ for every node $v \in V_2$, it is obvious that $t_2 = x(H_{B_1} \setminus H_{B_2}) \geqslant 0$. We are not able to give an exact formula for t_1, but can deduce bounds for it: Because H_{B_1} dominates H_{B_2}, the same holds for L_{B_1} and L_{B_1} as well as the edge weights $x_{B_1}(e)$ and $x_{B_2}(e)$:

$$L_{B_1}(e) \supseteq L_{B_2}(e)\ , \quad x_{B_1}(e) \geqslant x_{B_2}(e) \quad \forall e \in E_{22} \ .$$

Furthermore, because $\mathrm{supp}(L_{B_1}) \supseteq \mathrm{supp}(L_{B_2})$, $T_x(B_2)$ is a forest in $(V_2, \mathrm{supp}(L_{B_1}))$ as well. Hence, $x(L_{B_2} \upharpoonright T_x(B_2)) \leqslant x(L_{B_1} \upharpoonright T_x(B_1))$.

Additionally, it is possible to construct a forest in $(V_2, \mathrm{supp}(L_{B_2}))$ from $T_x(B_1)$: Let $T := T_x(B_1) \cap \mathrm{supp}(L_{B_2})$. Because B_1 and B_2 differ only on u, the edge weights $x_{B_1}(\cdot)$ and $x_{B_2}(\cdot)$ differ only on $\delta(u)$, where B_1-weights are dominating, and restricting T to $\mathrm{supp}(L_{B_2})$ only cuts off edges $e \in \mathrm{supp}(L_{B_1})$ that have a B_2-weight of $x_{B_2}(e) = 0$. This new forest T has a B_2-weight of $x(L_{B_2} \upharpoonright T) = x(L_{B_1} \upharpoonright T_x(B_1)) - t_1'$, where

$$t_1' := \sum_{e \in \delta(u)} x_{B_1}(e) - x_{B_2}(e) \ .$$

T is a feasible candidate in determination of $T_x(B_2)$, and hence $x(L_{B_2} \upharpoonright T_x(B_2)) \geqslant x(L_{B_1} \upharpoonright T_x(B_1)) - t_1'$. Together with the prior result this gives

$$x(L_{B_1} \upharpoonright T_x(B_1)) - t_1' \quad \leqslant \quad x(L_{B_2} \upharpoonright T_x(B_2)) \quad \leqslant \quad x(L_{B_1} \upharpoonright T_x(B_1))$$

and, by definition of t_1, $-t_1' \leqslant t_1 \leqslant 0$. Define

$$\begin{aligned} \underline{z}(B_1, B_2) &:= x(H_{B_1} \setminus H_{B_2}) - \sum_{e \in \delta(u)} (x_{B_1}(e) - x_{B_2}(e)) \\ \overline{z}(B_1, B_2) &:= x(H_{B_1} \setminus H_{B_2}) \end{aligned}$$

This leads to the estimate

$$\underline{z}(B_1, B_2) \quad \leqslant \quad z(B_1, B_2) \quad \leqslant \quad \overline{z}(B_1, B_2)\ .$$

Notice that $\underline{z}(B_1, B_2)$ can be easily determined if $T_x(B_1)$ is known, and does not depend on $T_x(B_2)$. Using $\underline{z}$ as an estimate for the objective gain caused by choosing one-node extension bands, we can use a heuristic for (4.35) that resembles the greedy heuristic for 0-1-knapsack problems. This heuristic is shown in Algorithm 1. At its core, it starts by choosing a band B consisting of the least-capacitated node designs on V_2, and determines $T_x(B)$ exactly. Then it checks all possible one-node extensions that stay below the knapsack capacity b, and assigns each such extension a density value $\frac{z}{c}$, where z is an estimate for the objective gain and c is the weight difference between the old and the new band. If an extension band with positive density exists, the algorithm chooses the band with highest density, determines a new $T_x(B)$, and repeats the process. This leads to a band where $z(B)$ is known and of potentially large value. If $z(B) > -1$, a violated inequality is identified.

Algorithm 1: Generate a potentially violated inequality (4.17)

Input : Node cut V_2, according bound b, point $x \geqslant 0$
Output : An inequality (4.17), or nothing.

Initialize $B \in \mathcal{B}$ to choose the smallest node designs: $\forall v \in V_2,\ d \in \mathcal{D}(v) : C^d \geqslant C^{B(v)}$
if $C(B) \geqslant b$ **then return** *'no inequality found'*

Determine maximal forest $T_x(B)$
repeat
  $a^* \leftarrow 0$, $B_2^* \leftarrow B$.
1.1 **for** *every one-node extension* B_2 *of* B *with* $C(B_2) < b$ **do**
    Determine density value $a(B_2) := \frac{\underline{z}(B,B_2)}{C(B_2)-C(B)}$
    if $a(B_2) > a^*$ **then**
      $a^* \leftarrow a(B_2)$, $B_2^* \leftarrow B_2$
1.2 **if** $a^* > 0$ **then**
    $B \leftarrow B_2^*$.
    Determine maximal forest $T_x(B)$.
until $a^* \leqslant 0$
return *valid inequality* $x(H_B) - x(L_B \restriction T_x(B)) \geqslant 1$.

4.1.4 $\mathbf{L_B \restriction T}$ Estimate

The heuristic assumes that $L_B \restriction T$ can be determined quickly, which is not always true. However, because any mapping $L : E_{22} \to 2^{\mathcal{L}}$ with $L \subseteq L_B \restriction T$ suits for Proposition 4.9 and the heuristic works well with L instead of $L_B \restriction T$, a predicate that identifies some chunky link design for an edge is sufficient to apply the heuristic.

Let $\ell \in \mathcal{L}$ and $d \in \mathcal{D}$. Define

$$\begin{aligned} S(\ell, d) := \quad \min\{ \sum_{m \in \mathcal{M}} S^m n_m \mid \quad & \sum_{m \in \mathcal{M}} I^{i,m} n_m \geqslant I^{i,\ell} && \forall i \in \mathcal{I}\ , \\ & 0 \leqslant n_m \leqslant M^{d,m} && \forall m \in \mathcal{M}\ , \\ & n_m \in \mathbb{Z}_+ && \forall m \in \mathcal{M}\ \}\ . \end{aligned} \tag{4.36}$$

if the minimum exists and $S(\ell, d) := +\infty$ otherwise.

Observation 4.11 *Let $B \in \mathcal{B}$ be a band. Let $v \in V_2$, $e \in E_{22} \cap \delta(v)$. Let $\ell \in \mathcal{L}(e)$ be a link design available on e. If $C^{B(v)} < C^\ell$ or $S^d < S(\ell, d)$ for all nonheavy node designs $d \in \mathcal{D}(v) \setminus H_B(v)$, then $\ell \in L_B(e)$.*

Correctness of this observation follows directly from the switching capacity resource inequality (3.11) and the fact that $S(\ell, d)$ is a lower bound on the number of slots occupied by modules for link design ℓ if d is installed on an incident node.

4.2 Single-Link Inequalities

In this section, we study polyhedra associated with link designs and commodity set capacities on a single edge: Let $e \in E$ be an edge and assume $\mathcal{K} = \{k_1, \ldots, k_n\}$ with $n \geqslant 1$ and $\mathcal{L}(e) = \{\ell_2, \ldots, \ell_m\}$.

Let $u \in \mathbb{Z}_+^n$ and $c \in \mathbb{Z}_+^m$ defined by

$$u_i := U^{k_i} \quad \forall i \in \{1, \ldots, n\} \ , \tag{4.37}$$

$$c_1 := C_e^0 \ , \quad c_j = C_e^0 + C^{\ell_j} \quad \forall j \in \{2, \ldots, m\} \ . \tag{4.38}$$

W.l.o.g. we assume $0 < u_1 \leqslant u_2 \leqslant \ldots \leqslant u_n$ and $0 \leqslant c_1 \leqslant c_2 \leqslant \ldots \leqslant c_m$. Now u contains all commodity set capacity units in nondecreasing order, and c contains all possible link capacities for e, also nondecreasing, where c_1 is the preinstalled capacity.

The routing capacity inequality (3.13) can now be written as

$$\sum_{i=1}^{n} u_i z_i \leqslant \sum_{j=1}^{m} c_j y_j \ , \tag{4.39}$$

using z and y as replacements for the component variables. There are only two local resources for edges: the above one and the link design GUB inequality (3.12). To model both, define the integer set Z as

$$Z := \{\ (z,y) \mid \ (z,y) \text{ fulfills (4.39)} \ , \quad \sum_{j=1}^{m} y_j = 1 \ , \quad z \in \mathbb{Z}_+^n,\ y \in \mathbb{B}^m \ \} \ . \tag{4.40}$$

Obviously, Z can be seen as relaxation of X_{HC} projected onto the space of variables for installable components on e, with y_1 being the slack of (3.12). We now study the structure of Z and $\text{conv}Z$, with obvious implications for X_{HC}.

Notice that a variable z_i, $i \in \{1, \ldots, n\}$, is fixed to zero unless $u_i \leqslant c_m$. Hence, we assume that $u_n \leqslant c_m$ holds.

First, a class of inequalities that are valid for Z is introduced. We will then attempt to identify class members that are facet-defining for $\text{conv}Z$.

Observation 4.12 *Let $s \in \{1, \ldots, n\}$. Then*

$$\sum_{i=1}^{n} \left\lfloor \frac{u_i}{u_s} \right\rfloor z_i \leqslant \sum_{j=1}^{m} \left\lfloor \frac{c_j}{u_s} \right\rfloor y_j \tag{4.41}$$

is valid for Z.

Verification of this observation is obvious: (4.41) can be derived from (4.39) by first scaling with $\frac{1}{u_s} > 0$, then relaxing the inequality by rounding down the left hand side, after which it is always integral and the right hand side can be rounded down as well.

The following proposition identifies facet-defining inequalities if each u_i divides its successors. According to [PW92], these results are a generalization of [Mar85], where a complete description of $\text{conv}Z$ for $m = 1$ with divisible commodity set capacity components is given.

For two integers $a, b \in \mathbb{Z}_+$ with $b > 0$, we denote the division remainder of $\frac{a}{b}$ by

$$r(a, b) := a - b \left\lfloor \frac{a}{b} \right\rfloor \ . \tag{4.42}$$

Proposition 4.13 *Let $s \in \{1, \dots, n\}$. If*

(i) $u_i | u_{i+1}$ for each $i \in \{1, \dots, n-1\}$,

and there exists a $t \in \{1, \dots, m\}$ such that

(ii) $u_s \leqslant c_t$, and

(iii) $u_{s-1} \leqslant r(c_t, u_s)$ if $s > 1$,

then (4.41) *defines a facet of* $\operatorname{conv} Z$.

Notice that (4.41) does not depend on the choice of t in Proposition 4.13, but on s only.

Proof Let s, t and u fulfill (i), (ii) and (iii). Let $I := \{1, \dots, n\}$, $I' := I \backslash \{s\}$ and $J := \{1, \dots, m\}$. Validity of (4.41) follows from observation 4.12. Let F be the set of all points in $\operatorname{conv} Z$ that fulfill (4.41) at equality.

Notice that $\left\lfloor \frac{u_i}{u_s} \right\rfloor = 0$ for $u_i < u_s$ and $\left\lfloor \frac{u_i}{u_s} \right\rfloor = \frac{u_i}{u_s}$ for $u_i \geqslant u_s$.

Define points p^i $(i \in I')$ and q^j $(j \in J)$ as follows:

1. For $i \in I'$ with $u_i < u_s$ let $p^i = (z^i, y^i)$ with $y^i_t = 1$, $z^i_s = \left\lfloor \frac{c_t}{u_s} \right\rfloor$, $z^i_i = 1$ and all other coefficients being 0. Then $p^i \in F$, p^i is integer, and

$$z^i_i = 1 \overset{\text{(iii)}}{\leqslant} \frac{1}{u_i} r(c_t, u_s) = \frac{1}{u_i} \left(c_t y^i_t - u_s z^i_s \right) ,$$

thus $p^i \in Z$.

2. For $i \in I'$ with $u_i \geqslant u_s$ define $p^i = (z^i, y^i)$ with $y^i_m = 1$, $z^i_i = 1$, $z^i_s = \left\lfloor \frac{c_m}{u_s} \right\rfloor - \left\lfloor \frac{u_i}{u_s} \right\rfloor$, remaining coefficients being zero. Since $u_i \leqslant c_m$, $z^i_s \geqslant 0$. Thus $p^i \in F$, and $p^i \in Z$, because

$$u_s z^i_s + u_i z^i_i = u_s \left(\left\lfloor \frac{c_m}{u_s} \right\rfloor - \left\lfloor \frac{u_i}{u_s} \right\rfloor \right) + u_i = u_s \left\lfloor \frac{c_m}{u_s} \right\rfloor \leqslant c_m = c_m z^i_m .$$

3. For $j \in J$ with $c_j \geqslant u_s$, let $q^j = (\tilde{z}^j, \tilde{y}^j)$, where $\tilde{y}^j_j = 1$, $\tilde{z}^j_s = \left\lfloor \frac{c_j}{u_s} \right\rfloor \geqslant 1$ and all other coefficients being 0. Then $q^j \in F$ and q^j is integer. Since $u_s \left\lfloor \frac{c_j}{u_s} \right\rfloor \leqslant c_j$, $q^j \in Z$.

4. For $j \in J$ with $c_j < u_s$, let $q^j = (\tilde{z}^j, \tilde{y}^j)$, where $\tilde{y}^j_j = 1$ is the only nonzero coefficient. Then $q^j \in F \cap Z$.

These points are affine independent: Each p^i is the only point with a nonzero coefficient for z_i $(i \in I')$. On the other hand, q^j $(j \in J)$ has only zero-coefficients for z_i $(i \in I')$ while being the only point in $\{q^j : j \in J\}$ with a nonzero coefficient for y_j.

Thus there are $(n + m - 1)$-many affine independent points in $F \cap Z$, and since $\dim \operatorname{conv} Z \leqslant n + m - 1$, $F \cap \operatorname{conv} Z$ is either a facet of $\operatorname{conv} Z$, or $\operatorname{conv} Z$ itself. The latter case is impossible, since the point $r = (z, y)$ with $z = 0$, $y_j = 0$ for $j < m$ and $y_m = 1$ is contained in $Z \backslash F$, and thus (4.41) is facet defining. $\square$

Chapter 5

Implementation

This chapter combines routings, as introduced in Section 2.2, the HC model from Section 3.2, and knowledge about associated polyhedra from Chapter 4 with related results from other sources in order to develop an integrated algorithm for network dimensioning problems. In Chapter 6, computational results will prove that the algorithm framework proposed here is in fact useful for real-world network planning problems.

We consider the problem of finding a feasible component installation whose capacities are sufficient for demand routings at minimal component cost, i.e.,

$$\min \left\{ Q^\mathsf{T} x \mid x \in X_{\mathrm{HC,R}} \right\} \ , \tag{5.1}$$

where $X_{\mathrm{HC,R}}$ is the set defined in (4.3), i.e.,

$$X_{\mathrm{HC,R}} := \{ x \in X_{\mathrm{HC}} \mid y_{x,k} \text{ allows for a feasible routing } \forall k \in \mathcal{K} \} \ .$$

In this chapter, an algorithm to improve variable bounds, eliminate variables and strengthen problem parameters is introduced. Then, a decomposition scheme to separate the problems of finding a routing and constructing a component installation is presented. Section 5.3 lists a series of preprocessing steps, which is followed by a detailed introduction to our central algorithm, which is a branch-and-cut variant on a MIP relaxation. The chapter is concluded by presenting cut inequalites and routing variations.

For ease of notation, we assume throughout this chapter that the two model extensions "tributary deprivation" and "paramater localization", as introduced in Section 3.2.3, are used.

5.1 Bound Strengthening

It is advisable to have the relaxation obtained by dropping the integrality constraints in $X_{\mathrm{HC,R}}$ as close as possible to $\mathrm{conv} X_{\mathrm{HC,R}}$, if LP-based algorithms such as branch-and-cut are used. In this section, a heuristic to strengthen inequalities is introduced. We attempt to refine variable bounds and problem parameters by different procedures, leading to a potentially better relaxation.

5.1.1 IP Rules

Here, four integer sets S_1 to S_4 which are similar to certain relaxations of $X_{\mathrm{HC,R}}$ are introduced. We show how solving IPs over these sets yields better, yet feasible, values for local component bounds and problem parameters.

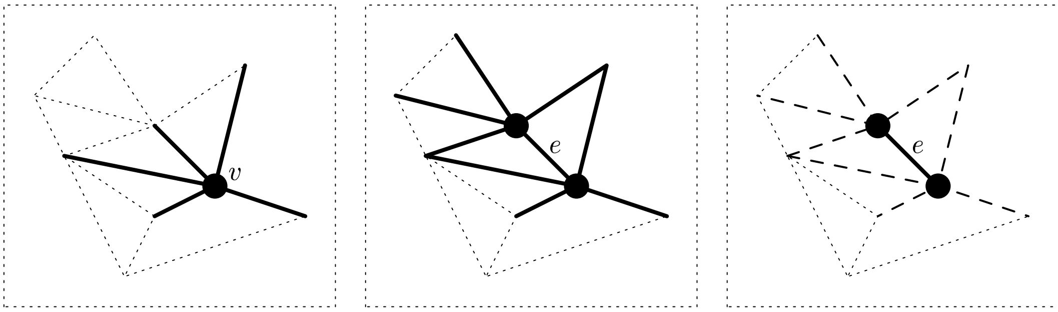


Figure 5.1: Graph elements for bound strengthening relaxations. *Left to right: $S_1(v)$ consists of a node and its incident edges, with all components and resources, as does $S_2(v,d)$ (same image); $S_3(e)$ contains edge e and the incident nodes and adjacent edges of e; $S_4(e)$ uses the same graph elements as $S_3(e)$, but limits adjacent edges to carry preinstalled capacities only.*

Node Set $\mathbf{S_1(v)}$

We begin by introducing an integer set that can be seen as a relaxation of $X_{\mathrm{HC,R}}$. It contains component installations for a node $v \in V$ and its incident edges. Let $N_1(v) \subseteq \mathcal{C} \times \mathcal{G}$ be defined by

$$
\begin{aligned}
N_1(v) := \quad & \big(\ \mathcal{D}(v) \times \{v\}\ \big) && \cup \\
& \big(\ \mathcal{M}(v) \times \{v\}\ \big) && \cup \\
& \big\{\ (\ell, e) \quad \mid e \in \delta(v),\ \ell \in \mathcal{L}(e)\ \big\} && \cup \\
& \big\{\ (k, e) \quad \mid e \in \delta(v),\ k \in \mathcal{K}\ \big\} && \cup \\
& \big\{\ (g, g) \quad \mid g \in \{v\} \cup \delta(v)\ \big\} && .
\end{aligned}
\tag{5.2}
$$

This set forms the variable index set for $S_1(v)$, which consists of component installations on $v \cup \delta(v)$ that satisfy the node resource inequalities on v and the local resource inequalities on $v \cup \delta(v)$, and provide sufficient commodity set capacities for the emanating demand of v. Figure 5.1 visualizes this set.

$$
\begin{aligned}
S_1(v) := \{\ s \in \mathbb{Z}_+^{N_1(v)} \mid\ & s \in R(r, v) && \forall r \in \mathcal{R}^V\ , \\
& s \in R(r, g) && \forall r \in \mathcal{R}^L,\ g \in \{v\} \cup \delta(v)\ , \\
& \sum_{e \in \delta(v)} s_{k,e} \geqslant \mathfrak{d}_k(v) && \forall k \in \mathcal{K}\ , \\
& \underline{x}_{c,g} \leqslant s_{c,g} \leqslant \overline{x}_{c,g} && \forall (c, g) \in N_1(v) \qquad \}.
\end{aligned}
\tag{5.3}
$$

An important property of $S_1(v)$ is the following fact:

Observation 5.1 *Let $x \in X_{\mathrm{HC,R}}$ be a component installation that allows a feasible demand routing, $v \in V$ and $c \in \mathcal{C}$ a component for which $(c, v) \in N_1(v)$. Then there exists a point $s \in S_1(v)$ such that $s_{c,v} = x_{c,v}$.*

For verification, notice that the construction of $S_1(v)$ consists of inequalities that are either valid for X_{HC} or a routing model, and $N_1(v)$ merely consists of all variables appearing in these constraints. $S_1(v)$ is thus a relaxation of the projection of X_{HC} onto $\mathbb{Z}_+^{N_1(v)}$. For completeness, we explicitly state the following obvious corollary:

Observation 5.2 *Let $v \in V$ and $c \in \mathcal{C}(v)$. Then*

$$
\min \{s_{c,v} \mid s \in S_1(v)\} \quad \leqslant \quad x_{c,v} \quad \leqslant \quad \max \{s_{c,v} \mid s \in S_1(v)\}
$$

for all $x \in X_{\mathrm{HC,R}}$. *Furthermore, setting*

$$\underline{x}_{c,v} \leftarrow \min_{s \in S_1(v)} s_{c,v} \,, \tag{5.4}$$

$$\overline{x}_{c,v} \leftarrow \max_{s \in S_1(v)} s_{c,v} \tag{5.5}$$

does not change $X_{\mathrm{HC,R}}$.

Hence, for a node $v \in V$ and a component $c \in \mathcal{C}(v)$, it is possible to set $\underline{x}_{c,v}$ and $\overline{x}_{c,v}$ to the values determined in (5.4) and (5.5). While this does not change $X_{\mathrm{HC,R}}$, it might very well change the relaxation of $X_{\mathrm{HC,R}}$ obtained by dropping the integrality constraints. This is precisely how $S_1(v)$ will be used: Solving the IPs in (5.4) and (5.5) to obtain better variable bounds during a preprocessing step.

Node Design Set $\mathbf{S_2(v,d)}$

$S_1(v)$ allows getting information on variable bounds for all nodes. We now use it to sharpen problem parameters of a node design. Let $v \in V$ and $d \in \mathcal{D}(v)$, and let

$$S_2(v,d) := \{ s \in S_1(v) \mid s_{d,v} = 1 \} \tag{5.6}$$

be the integer set consisting of component installations in $S_1(v)$ that have node design d installed. Then $S_2(v,d)$ is a projected relaxation of $\{x \in X_{\mathrm{HC,R}} \mid x_{d,v} = 1\}$. Hence, the set can be used to get accurate values for a node design's parameters:

Observation 5.3 *Let* $v \in V$ *and* $d \in \mathcal{D}(v)$. *Setting*

$$C_v^d \leftarrow \max_{s \in S_2(v,d)} C_v^0 + \sum_{e \in \delta(v)} \left(C_e^0 + \sum_{\ell \in \mathcal{L}} C^\ell s_{\ell,e} \right) \tag{5.7}$$

$$M_v^{d,m} \leftarrow \max_{s \in S_2(v,d)} s_{m,v} \quad \forall m \in \mathcal{M} \tag{5.8}$$

does not change $X_{\mathrm{HC,R}}$.

This observation is valid because applying these rules just closes the gap of some model inequality: (5.7) for (3.11) resp. (3.11b) on page 33, and (5.8) for (3.9).

Large Edge Set $\mathbf{S_3(e)}$

For an edge $e \in E$, we construct a set $S_3(e)$ by a means similar to $S_1(v)$. It consists of component installations on e, the end-nodes of e and all adjacent edges of e. Hence, its variable index set is

$$\begin{aligned} N_3(e) := \; & \big(\mathcal{D}(u) \times \{u\} \big) \;\cup\; \big(\mathcal{D}(v) \times \{v\} \big) && \cup \\ & \big(\mathcal{M}(u) \times \{u\} \big) \;\cup\; \big(\mathcal{M}(v) \times \{v\} \big) && \cup \\ & \big\{ (\ell, e') \mid e' \in \delta(u) \cup \delta(v),\ \ell \in \mathcal{L}(e') \big\} && \cup \\ & \big\{ (k, e') \mid e' \in \delta(u) \cup \delta(v),\ k \in \mathcal{K} \big\} && \cup \\ & \big\{ (g, g) \mid g \in \{u\} \cup \{v\} \cup \delta(u) \cup \delta(v) \big\} && . \end{aligned} \tag{5.9}$$

We define $S_3(e)$ to contain component installations that are valid w.r.t. resource inequalities and component bounds for the considered graph elements, and provide sufficient commodity set capacity components for the emanating demand of the two nodes:

$$\begin{aligned} S_3(e) := \{ \, s \in \mathbb{Z}_+^{N_3(e)} \mid \; & s \in R(r,g) && \forall r \in \mathcal{R}^V,\ g \in \{u,v\} \,, \\ & s \in R(r,g) && \forall r \in \mathcal{R}^L,\ g \in \{u,v\} \cup \delta(u) \cup \delta(v) \,, \\ & \sum_{e' \in \delta(g)} s_{k,e'} \geqslant \mathfrak{d}_k(g) && \forall k \in \mathcal{K},\ g \in \{u,v\} \,, \\ & \underline{x}_{c,g} \leqslant s_{c,g} \leqslant \overline{x}_{c,g} && \forall (c,g) \in N_3(e) \quad \} \,. \end{aligned} \tag{5.10}$$

By an argumentation analoguous to Observation 5.2, the following holds as well:

Observation 5.4 *Let $e \in E$ and $c \in \mathcal{C}(e)$. Setting*

$$\underline{x}_{c,v} \leftarrow \min_{s \in S_3(e)} s_{c,v} \ , \tag{5.11}$$

$$\overline{x}_{c,v} \leftarrow \max_{s \in S_3(e)} s_{c,v} \tag{5.12}$$

does not change $X_{\mathrm{HC,R}}$.

Small Edge Set $\mathbf{S_4(e)}$

In case solving IPs over $S_3(e)$ becomes too time-consuming, another integer set for $e \in E$ is introduced here. It is constructed by first relaxing the constraints for the emanating demands and all local resources on all adjacent edges of e. Because link designs and commodity set capacity components do not provide any resource to other components, their variables can be omitted. We keep only link designs on e, components on the end-nodes of e and the graph elements themselves. The resulting relaxation's index set is thus

$$\begin{aligned} N_4(e) := \ & (\ \mathcal{D}(u) \times \{u\}\ ) \quad \cup \quad (\ \mathcal{D}(v) \times \{v\}\ ) \quad \cup \\ & (\ \mathcal{M}(u) \times \{u\}\ ) \quad \cup \quad (\ \mathcal{M}(v) \times \{v\}\ ) \quad \cup \\ & (\ \mathcal{L}(e) \times \{e\}\ ) \quad \cup \\ & \{\ (g,g) \quad \mid\ g \in \{u\} \cup \{v\} \cup \delta(u) \cup \delta(v)\ \} \quad . \end{aligned} \tag{5.13}$$

Using this index set, we define

$$\begin{aligned} S_4(e) := \{\ s \in \mathbb{Z}_+^{N_4(e)} \mid\ & s \in R(r,g) && \forall r \in \mathcal{R}^V,\ g \in \{u,v\}\ , \\ & s \in R(r,g) && \forall r \in \mathcal{R}^L,\ g \in \{e,u,v\}\ , \\ & \underline{x}_{c,g} \leqslant s_{c,g} \leqslant \overline{x}_{c,g} && \forall (c,g) \in N_4(e) \qquad \}\ . \end{aligned} \tag{5.14}$$

By construction, we can adopt Observation 5.4 to this set:

Observation 5.5 *Let $e \in E$ and $c \in \mathcal{C}(e)$. Setting*

$$\underline{x}_{c,v} \leftarrow \min_{s \in S_4(e)} s_{c,v} \ , \tag{5.15}$$

$$\overline{x}_{c,v} \leftarrow \max_{s \in S_4(e)} s_{c,v} \tag{5.16}$$

does not change $X_{\mathrm{HC,R}}$.

5.1.2 Combinatorial Rules

The classic approach to bound improving is using a fixed set of combinatorial rules that specify conditions under which certain parameters can be changed. Five such rules are presented here:

1. A link design cannot be installed if there exists no available node design, which has a switching capacity that is large enough for the link design and the preinstalled capacities on an edge:

$$\overline{x}_{\ell,e} \leftarrow 0 \quad \forall v \in V,\ e \in \delta(v),\ \ell \in \mathcal{L}:\ C^\ell > \max_{d \in \mathcal{D}(v)} C^d - \sum_{e' \in \delta(v)} C^0_{e'} - C^0_v\ . \tag{5.17}$$

2. Commodity set capacity components have infinite upper bounds according to the model, but since they are bound by the provided capacities, we set

$$\overline{x}_{k,e} \leftarrow \max_{\ell \in \mathcal{L}(e)} \left\lfloor \frac{C^0_e + C^\ell}{U^k} \right\rfloor \quad \forall k \in \mathcal{K},\ e \in E\ . \tag{5.18}$$

			BSH–1	BSH–2	BSH–3
Once	$M^{d,m}$	$(\forall d \in \mathcal{D}, m \in \mathcal{M})$	Apply (5.20)	Apply (5.20)	Apply (5.20)
$\forall e \in E$	$\overline{x}_{\ell,e}$	$(\forall \ell \in \mathcal{L}(e))$	Apply (5.17)	$\max_{s \in S_4(e)} s_{\ell,e}$	$\max_{s \in S_3(e)} s_{\ell,e}$
	$\overline{x}_{k,e}$	$(\forall k \in \mathcal{K})$	Apply (5.18)	Apply (5.18)	Apply (5.18)
$\forall v \in V$	$\overline{x}_{d,v}$	$(\forall d \in \mathcal{D}(v))$	Apply (5.19)	$\max_{s \in S_1(v)} s_{d,v}$	$\max_{s \in S_1(v)} s_{d,v}$
	$M_v^{d,m}$	$(\forall d \in \mathcal{D}(v), m \in \mathcal{M})$	—	—	$\max_{s \in S_2(v,d)} s_{m,v}$
	$\overline{x}_{m,v}$	$(\forall m \in \mathcal{M}(v))$	Apply (5.21)	$\max_{s \in S_1(v)} s_{m,v}$	Apply (5.21) †
	$\underline{x}_{m,v}$	$(\forall m \in \mathcal{M}(v))$	—	$\min_{s \in S_1(v)} s_{m,v}$	$\min_{s \in S_1(v)} s_{m,v}$
	C_v^d	$(\forall d \in \mathcal{D}(v))$	—	—	Solve (5.7) ‡

Table 5.1: Levels of the bound strengthening heuristic. *In each level, first* (5.20) *is evaluated. Second, different strengthening rules are applied, first for edges, then for nodes. With increasing level, more IP-based rules are applied.*

3. It is impossible to install a node design at a given node unless its switching capacity suffices for the preinstalled capacities on incident edges. Thus we set

$$\overline{x}_{d,v} \leftarrow 0 \quad \forall d \in \mathcal{D},\ v \in V :\ C^d < \sum_{e \in \delta(v)} C_e^0 + C_v^0 \ . \tag{5.19}$$

4. If a module $m \in \mathcal{M}$ does consume slots (i.e. $S^m > 0$), it is useless to raise module bounds above full slot utilization. We thus assume

$$M^{d,m} \leftarrow \min \left\{ M^{d,m}, \left\lfloor \frac{S^d}{S^m} \right\rfloor \right\} \quad \forall d \in \mathcal{D},\ m \in \mathcal{M} :\ S^m > 0 \ . \tag{5.20}$$

5. Albeit $\overline{x}_{m,v} = \infty$ for a module $m \in \mathcal{M}$ and a node $v \in V$, it is impossible to install more modules than supported by the node designs installable at v. Since at most one node design can be chosen at v, we can set

$$\overline{x}_{m,v} \leftarrow \max_{d \in \mathcal{D}(v)} M_v^{d,m} \quad \forall m \in \mathcal{M},\ v \in V \ . \tag{5.21}$$

5.1.3 Heuristics

To perform a feasible strengthening of bounds and problem parameters, any combination of the IP-based rules presented in Observations 5.2 to 5.5 and the combinatorial rules (5.17) to (5.21) can be used.

It is obvious that the IP-based rules yield better or equal results, because the combinatorials simply exploit a few resource inequalities that are contained in the IPs as well. On the other hand, the combinatorial rules can be evaluated in polynomial time, while nothing is known about the former's runtimes. As will be seen in Section 5.2, our algorithmic approach is based on the assumption that the IP $\min \left\{ Q^\mathsf{T} x \mid x \in X_{\mathrm{HC}} \right\}$ can be solved in reasonable time. Because the IPs S_1 to S_4 are projected relaxations of X_{HC}, we assume that they can be solved quickly, but fail to provide evidence other than computational observations.

We picked three rule sets to create the three bound strengthening heuristics BSH–1 to BSH–3. Each of them first applies (5.20), then applies some rules for all edges, and finally applies some

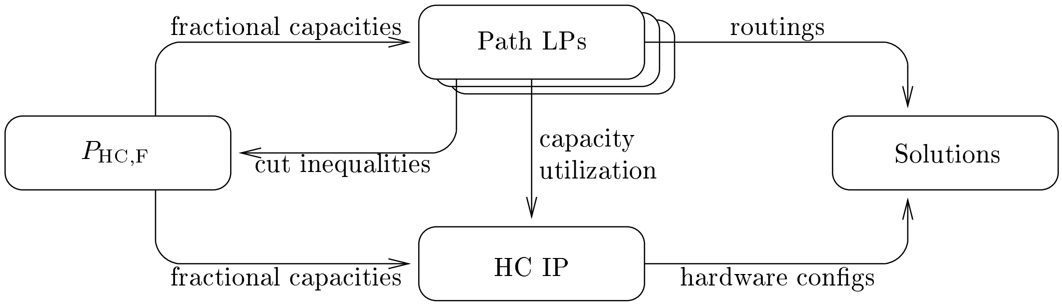


Figure 5.2: Decomposition. *The central relaxation $P_{\mathrm{HC,F}}$ provides fractional capacities on the edges. The path LPs produce routings, fractional edge capacity utilizations and cut inequalities. The HC IP computes feasible component installations based on given edge capacity requirements.*

rules for all nodes. A detailed overview on which rule is applied by what heuristic is given in Table 5.1. BSH–1 is supposed to be fast while producing the worst results, BSH–3 to provide the best results while being very slow, and BSH–2 is a compromise. Most of Table 5.1 is self-explanatory, additional notes are needed for two entries only:

†) Application of (5.21) gives the same results as running the IP-based rule applied by BSH–2, because $\max\{s_{m,v} \mid s \in S_1(v)\}$ appears as $M_v^{d,m}$ for some $d \in \mathcal{D}(v)$ due to the construction of $S_2(v,d)$.

‡) There is nothing special about this entry that distinguishes it from the other IP-based rules, except for the long objective function, which does not fit into the table.

5.2 Problem Decomposition

Our central algorithm is based on a decomposition of the problem into three parts. In this section, these parts are introduced and their interaction is shown. See Figure 5.2 for an overview.

For each commodity set capacity component $k \in \mathcal{K}$, let $\mathfrak{D}_k$ denote the according commodity set as introduced in Section 2.2.

Main LP Relaxation $\mathbf{P_{HC,F}}$

The first part consists of the LP relaxation obtained by combining the HC model with a multi-commodity-flow formulation and relaxing integrality requirements. Let

$$\begin{aligned} P_{\mathrm{HC,F}} := \{(x,f) \mid\ & x \in \mathbb{R}^{\mathcal{C}\times\mathcal{G}} \ , \\ & x \in R(r,g)\ \forall r \in \mathcal{R},\ g \in \mathcal{G} \ , \\ & \underline{x} \leqslant x \leqslant \overline{x} \ , \\ & f = (f^{(k)})_{k\in\mathcal{K}} \ , \\ & f^{(k)} \text{ fulfills (2.4) to (2.6) for } \mathfrak{D}_k \text{ and } y_{x,k},\ \forall k \in \mathcal{K}\} \ . \end{aligned} \tag{5.22}$$

The according LP minimizes component installation cost over $P_{\mathrm{HC,F}}$. Its optimal solution value is denoted by

$$z_{\mathrm{HC,F}} := \min_{(x,f)\in P_{\mathrm{HC,F}}} Q^{\mathsf{T}} x \ . \tag{5.23}$$

Notice that each point $(x,f) \in P_{\mathrm{HC,F}}$ provides fractional edge capacities $y_{x,k}$ for each $k \in \mathcal{K}$.

Path LPs

The second decomposition piece are path LPs (2.1). For a point $(x, f) \in P_{\mathrm{HC,F}}$ and a commodity set $\mathfrak{D}_k$, there is an according path LP that computes a routing using at most $y_{x,k}(e) + \alpha$ on each edge $e \in E$, with minimal α. Additionally, it provides violated cut inequalities to $P_{\mathrm{HC,F}}$ and a fractional capacity vector by the routing's edge flows, by Theorem 2.1.

HC IP

For given fractional edge capacities

$$\tilde{y} = (\tilde{y}_k)_{k \in \mathcal{K}}, \quad \tilde{y}_k : E \to \mathbb{R}_+ \quad \forall k \in \mathcal{K} ,$$

consider the IP

$$\begin{aligned} \min\{\, & Q^{\mathsf{T}} x \mid x \in X_{\mathrm{HC}}, \\ & y_{x,k}(e) \geqslant \tilde{y}_k(e) \ \forall k \in \mathcal{K},\ e \in E \} . \end{aligned} \tag{5.24}$$

For fractional edge capacities as delivered by $P_{\mathrm{HC,F}}$ or the path LPs, this problem can be used to test whether a feasible component installation exists, which provides these commodity set capacities, and to provide such an installation of minimal cost.

5.3 Preprocessing

We apply a series of preprocessing steps to improve $P_{\mathrm{HC,F}}$ and X_{HC} prior to running our central algorithms. These steps are introduced here.

Pseudo Node Design

A drawback of the relaxation $P_{\mathrm{HC,F}}$ is that optimal solutions typically use small node design fractions at nodes where small capacities are sufficient on incident edges. Hence, an artificial node design $d^0 \in \mathcal{D}$ is introduced. It is installable on all nodes, i.e., $d^0 \in \mathcal{D}(v)$ for each node $v \in V$, but does not provide anything:

$$C^{d^0} = 0, \quad S^{d^0} = 0, \quad M^{d^0,m} = 0 \ \forall m \in \mathcal{M}, \quad Q_{d^0,v} = 0 \ \forall v \in V .$$

Now, the node design GUB inequalities (3.7) are updated to read

$$\sum_{d \in \mathcal{D}} x_{d,v} = 1 \quad \forall v \in V . \tag{5.25}$$

However, this step does not help preventing fractional node design installations by itself. This step is done in the hope that the following preprocessing steps will eliminate the pseudo node design on some nodes, thereby updating (5.25).

Bound Strenghtening

As second preprocessing step, one of the bound strengthening heuristics from Section 5.1 is applied.

Single-Link Inequalities

Proposition 4.13 identifies facets for a polyhedron associated with one edge of the graph. The preprocessing checks for all edges and commodity set capacity components whether the preconditions for that proposition are met, and adds all resulting inequalities (4.41) to the formulation.

Variable and Constraint Suppression

The HC model contains way more variables and constraints than needed. E.g., there are variables for node designs on edges, which are fixed to zero by the component bounds, and node design GUB inequalities for edges, which read $0 \leqslant 0$. This preprocessing step gets rid of them.

First, all variables are replaced by variables relative to lower bounds. Each model inequality of the form

$$\sum_{c \in \mathcal{C}} \sum_{g \in \mathcal{G}} \alpha_{c,g} x_{c,g} \geqslant \beta \tag{5.26}$$

is instead written as the equivalent

$$\sum_{c \in \mathcal{C}} \sum_{g \in \mathcal{G}} \alpha_{c,g} (x_{c,g} - \underline{x}_{c,g}) \geqslant \beta - \sum_{c \in \mathcal{C}} \sum_{g \in \mathcal{G}} \alpha_{c,g} \underline{x}_{c,g} \ . \tag{5.27}$$

Then, for each pair $(c, g) \in \mathcal{C} \times \mathcal{G}$, a variable for $x_{c,g} - \underline{x}_{c,g}$ is added. This variable always has a lower bound of zero and a upper bound of $\overline{x}_{c,g} - \underline{x}_{c,g}$. Then, $P_{\mathrm{HC,F}}$ and X_{HC} are formulated using the new variables. Now, variables that are fixed to zero and some trivial constraints $0 \leqslant 0$ are omitted.

To avoid stacking a series of dependent notations for each preprocessing step, we continue to use the original variables and constraints in this work, as the transition is obvious and can be dealt with implicitly.

Incremental Link Designs

In [SD94], the concept of *incremental capacities* was introduced: To model a set of discrete capacities on an edge, a series of nonincreasing binary variables is used, where each variable provides some expansion capacity w.r.t. its predecessors. This concept turns out to be very useful in a variable-based branch-and-bound, as will become clear later. Hence, we adopt it to the link design components of the HC model.

Let $e \in E$ and let $T_e := |\mathcal{L}(e)|$ be the number of link designs available on e. Let $(\ell_e^1, \ldots, \ell_e^{T_e})$ be a list of these link designs sorted in order of nondecreasing capacity, i.e.,

$$\mathcal{L}(e) = \left\{\ell_e^1, \ldots, \ell_e^{T_e}\right\} , \quad C^{\ell_e^1} \leqslant C^{\ell_e^2} \leqslant \cdots \leqslant C^{\ell_e^{T_e}} \ .$$

Instead of using the component variables $x_{\ell_j,e}$ directly, incremental variables are used by introducing variables

$$\hat{x}_e^t := \sum_{j=t}^{T_e} x_{\ell_e^j,e} \quad \forall t \in \{1, \ldots, T_e\} \ . \tag{5.28}$$

In reverse, this is equivalent to

$$x_{\ell_e^{T_e},e} = \hat{x}_e^{T_e} \ , \quad x_{\ell_e^t,e} = \hat{x}_e^t - \hat{x}_e^{t+1} \quad \forall t \in \{1, \ldots, T_e - 1\} \ . \tag{5.29}$$

Some important properties of the new variables become obvious by explicitly translating inequalities to the new space. For $t \in \{1, \ldots, T_e - 1\}$, $\underline{x}_{\ell_e^t,e} = 0$ gives that variable bounds are $x_{\ell_e^t,e} \geqslant 0$, which translates to

$$\hat{x}_e^t \geqslant \hat{x}_e^{t+1} \quad \forall t \in \{1, \ldots, T_e - 1\} \ . \tag{5.30}$$

These inequalities are called *ordering constraints.* The last bound $x_{\ell_e^{T_e},e} \geqslant 0$ translates, using (5.30), to

$$\hat{x}_e^t \geqslant 0 \quad \forall t \in \{1, \ldots, T_e\} \ . \tag{5.31}$$

The link design GUB inequality $\sum_{t=1}^{T_e} x_{\ell_e^j,e} \leqslant 1$ (3.12) translates to $\hat{x}_e^1 \leqslant 1$. Together with (5.30), this gives

$$\hat{x}_e^t \leqslant 1 \quad \forall t \in \{1,\ldots,T_e\} \quad . \tag{5.32}$$

So the incremental variables are binary variables in nonincreasing order, hence consisting of a leading sequence of ones followed by a trail of zeroes, where the index of the last 1 corresponds to the index of the link design that is chosen in the original variable space.

An important point is the effect of fixing such a variable. For $t \in \{1,\ldots,T_e\}$, fixing $\hat{x}_e^t := 1$ is equivalent to limiting the link choice on e to ℓ_e^t to $\ell_e^{T_e}$, and setting $\hat{x}_e^t := 0$ equals forbidding them. Because capacities are sorting in nondecreasing order, the former specifies a lower bound on the capacity installed at e, while the latter produce an upper bound.

For the central relaxation $P_{\mathrm{HC,F}}$, incremental link design variables are used on all edges. We actually discard the original variables $x_{\ell,e}$ and translate all inequalities, including cut inequalities, to the new variable space. Hence, inequalities (5.30) to (5.32) are explicitly contained in the relaxation. For simplicity of notation, we will neglect the existence of this preprocessing step unless properties of the incremental variables are explicitly needed.

Capacity Bounds

If commodity set capacity components are free w.r.t. the objective function, i.e., $Q_{k,e} = 0$ for $k \in \mathcal{K}$ and $e \in E$, it is always possible to raise them as far as possible without changing the objective value. On the other hand, chances are that low values for these variables do suffice for the routing relaxation contained in $P_{\mathrm{HC,F}}$, but are not sufficient for a real routing. Hence, adding a lower bound constraint for these variables seems advisable. The preprocessing adds all inequalities

$$\sum_{k\in\mathcal{K}} U^k x_{k,e} \quad \geqslant \quad C_e^0 + \sum_{\ell\in\mathcal{L}(e)} C^\ell x_{\ell,e} - \left(\min_{k\in\mathcal{K}} U^k - 1\right) \quad \forall e \in E \tag{5.33}$$

to the formulation, if the commodity set capacity components are free.

5.4 Branch-and-Cut

We employ a customized branch-and-cut algorithm to solve problem (5.1),

$$\min\left\{Q^\mathsf{T} x \mid x \in X_{\mathrm{HC,R}}\right\} \; .$$

This section covers details about this algorithm.

First, let us review 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

$$\begin{aligned}\min\{c^\mathsf{T} x \mid\; & x \in P\;,\\ & \underline{x} \leqslant x \leqslant \overline{x}\;,\\ & x_i \text{ integer } \;\forall i \in Z\;\}\;.\end{aligned} \tag{5.34}$$

The branch-and-bound algorithm solves this problem using a modified divide-and-conquer approach. The complete algorithm is shown as Algorithm 2. It solves the relaxation of (5.34) 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 sub-problems such that the current point is feasible for neither of the two, but any solution of the original problem is for one of them.

This algorithm is widely known, so we restrain from going into depth here. Rather, a few facts which become important later in this work are discussed now.

Algorithm 2: Branch-and-bound

Input : Bounded problem (5.34): $\min\left\{c^\mathsf{T}x \mid x \in R, \underline{x} \leqslant x \leqslant \overline{x}, x \in \mathbb{R}^N, x_i \in \mathbb{Z}\ \forall i \in Z\right\}$

Output : Optimal solution for (5.34)

$\mathcal{T} \leftarrow \{(\underline{x},\overline{x})\}$, $\mathcal{A} \leftarrow \{(\underline{x},\overline{x})\}$, $\{(\underline{x},\overline{x})\}$, $z^* \leftarrow +\infty$

while $\mathcal{A} \neq \varnothing$ **do**

2.1 Select a $(\underline{x}',\overline{x}') \in \mathcal{A}$

$\mathcal{A} \leftarrow \mathcal{A} \backslash \{(\underline{x}',\overline{x}')\}$

Solve relaxation $\min\left\{c^\mathsf{T}x \mid x \in R, \underline{x}' \leqslant x \leqslant \overline{x}', x \in \mathbb{R}^N\right\}$.

2.2 **if** *there exists an optimal solution* x' *with* $c^\mathsf{T}x' < z^*$ **then**

if *there exists an* $i \in Z$ *such that* x'_i *is fractional* **then**

2.3 Branch on i: Create two sub-problems $(\underline{x}^\downarrow,\overline{x}^\downarrow)$ and $(\underline{x}^\uparrow,\overline{x}^\uparrow)$ that equal $(\underline{x},\overline{x})$ with the exceptions that $\overline{x}_i^\downarrow = \lfloor x'_i \rfloor$ and $\underline{x}_i^\uparrow = \lceil x'_i \rceil$.

$\mathcal{T} \leftarrow \mathcal{T} \cup \left\{(\underline{x}^\downarrow,\overline{x}^\downarrow),(\underline{x}^\uparrow,\overline{x}^\uparrow)\right\}$

$\mathcal{A} \leftarrow \mathcal{A} \cup \left\{(\underline{x}^\downarrow,\overline{x}^\downarrow),(\underline{x}^\uparrow,\overline{x}^\uparrow)\right\}$

else

2.4 x' is a valid solution for the original problem (5.34).

if $c^\mathsf{T}x' < z^*$ **then**

$z^* \leftarrow c^\mathsf{T}x'$, $x^* \leftarrow x'$.

if $z^* = \infty$ **then**

return *'problem* (5.34) *has no solution'*

else

return *optimal solution* x^* *with objective value* z^*.

- 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 nodes 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 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 2.2 could be omitted without breaking algorithm correctness. It exploits the fact that $c^\mathsf{T}x'$ is a lower bound for all integral (w.r.t. Z) solutions of the node's local problem. Hence, if $c^\mathsf{T}x' \geqslant z^*$, the node can only contain an optimal solution to (5.34) if the already known x^* is optimal as well. This method of removing sub-trees is called bounding.
- The branching step 2.3 may use different rules which fractional x_i to choose. This will be further explored in Section 5.4.2.

The branch-and-cut algorithm improves branch-and-bound by merging a cutting plane algorithm into it. If cutting plane separators for (5.34) are available, their inequalities are obviously valid for each node's problem as well. Hence, at each branch-and-bound node, it is attempted to add cutting planes that are valid for the original MIP and cut off the local fractional points to P. This algorithm was introduced in [PR91] for the traveling salesman problem. An adoption for network design problems is proposed for example in [Gün99].

There are two benefits in adding cutting planes that cut off a fractional solution: First, integral solutions are found more quickly. Second, local and global lower bounds rise faster, thereby increasing the chance that sub-trees can be discarded from the branch-and-bound tree.

Our implementation handles the root node other than the other nodes: At the root node, all available separation algorithms are iterated until they no longer find violated inequalities. On the other nodes, separators are run just once.

For the node selection step 2.1, we employ a *lower-and-dive* strategy where two phases are alternated: First, a "lower" phase, where the node with the worst lower bound is chosen, in the hope that its offspring will yield better bounds, thereby increasing the global lower bound. This phase always lasts ten branch-and-bound iterations. Second, a "dive". Here, 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 infeasible, or a solution is found.

5.4.1 Decomposed Branch-and-Cut

We are now in the position to put the pieces introduced so far together, resulting in an algorithm that solves the hardware configuration and routing problems at once.

We run a branch-and-cut algorithm on $\min\left\{Q^\mathsf{T}x \mid x \in P_{\mathrm{HC,F}},\ x_i \in \mathbb{Z}_+\ \forall i \in Z\right\}$ where Z contains the indices of all commodity set capacity components and incremental link designs on all edges. While enumerating the according branch-and-bound tree eventually yields solutions with integral capacities, it is not clear that these will contain feasible component installations or allow a feasible routing. Alas, the properties of the problem decomposition that are introduced in Section 5.2 suffice to modify the branch-and-bound procedure to solve the original problem (5.1). The new procedure to handle a node of the branch-and-bound tree is Algorithm 3.

Algorithm 3: Node procedure for decomposed problem

Input : $P_{\mathrm{HC,F}}$, branch-and-cut node $(\underline{x}, \overline{x})$, Z.

Solve relaxation $\min\left\{c^\mathsf{T}x \mid x \in P_{\mathrm{HC,F}}, \underline{x} \leqslant x \leqslant \overline{x}\right\}$. Let x' be an optimal solution if it exists.
if *there exists no optimal solution, or* $c^\mathsf{T}x' \geqslant z^*$ **then return**

foreach $k \in \mathcal{K}$ **do**
3.1 Solve according external path LP (2.1) using $y_{x',k}$ to obtain optimal α_k and a routing for the commodity set corresponding to k.

Solve HC IP to find an optimal hardware configuration $x \in X_{\mathrm{HC}}$, using the routings' capacity utilization as lower bounds for commodity set capacity components.
if *HC solution exists* **then**
$\{x$ *and the routings form a solution*$\}$
3.2 Postprocess solution if required.
Update x^* and z^*.

3.3 **if** *there exists a* $(c, g) \in Z$ *with* $\underline{x}_{c,g} < \overline{x}_{c,g}$ **then**
Branch on (c, g).

In general, the branch-and-bound algorithm only needs to branch on fractional variables: If all variables in Z are integral, the current solution is an optimal solution for the MIP of the current node. Here, we branch on link designs and commodity set capacity components only, and use a relaxation of the routing formulation. A solution of a node's local problem can be integral w.r.t. Z without allowing for a routing that respects all routing conditions. It may also be impossible to find a fully integral solution with the same capacities. Therefore, it is required to branch on variables which have an integral value but whose local bounds do not fix the variable.

Step 2.3 of Algorithm 2 has to be updated to deal with this situation. If $i \in Z$ is the index of a variable chosen for branching, and $x'_i \in \mathbb{Z}$, there are two possibilities to construct $(\underline{x}^{\downarrow}, \overline{x}^{\downarrow})$ and

$(\underline{x}^{\uparrow}, \overline{x}^{\uparrow})$. First, using

$$\overline{x}_i^{\downarrow} = x_i', \quad \underline{x}_i^{\uparrow} = x_i' + 1 ,$$

and second with

$$\overline{x}_i^{\downarrow} = x_i' - 1, \quad \underline{x}_i^{\uparrow} = x_i' .$$

If $\underline{x}_i = \overline{x}_i - 1$, exactly one of the two possibilities always produces an empty sub-problem in one child node and the original problem in the other. In this case, the other possibility is realized. If $\underline{x}_i < \overline{x}_i - 1$, both are useful.

In either case, one of the child problems contains the fractional solution of the parent node. However, the branch-and-cut algorithm is still guaranteed to terminate within finite time, because it is impossible to repeat the branching step infinite times, because variable bounds of the sub-problems are getting closer.

5.4.2 Branching Rules

There are many possible branching strategies. Our implementation uses five rules, two classic and three new ones, which are introduced here.

The rules follow a common scheme: First, a weight function is evaluated. It assigns a weight ω_e^j to each incremental link design $j \in \{1, \ldots, T_e\}$ for all edges $e \in E$, and a weight $\omega_{k,e}$ to each commodity set capacity component $k \in \mathcal{K}$ for each $e \in E$. A variable that is fractional and has a largest weight ω is chosen for branching. If no such variable exists, a nonfixed (w.r.t. $\underline{x}$ and $\overline{x}$) integral variable of largest weight is chosen. If such a variable doesn't exist either, all variables indexed by Z are fixed, hence branching is not possible at all.

The five rules and their weight functions are:

max.inf.: This is one of the two classic rules: A variable gets a high priority if it is nearly integral, i.e., if f is the fractional part of a variable's value, the variable is chosen if f is close to 1 or 0. The weight function hence is

$$\begin{aligned} \omega_e^j &:= \max\{\hat{x}_e^j - \lfloor \hat{x}_e^j \rfloor, \quad \lceil \hat{x}_e^j \rceil - \hat{x}_e^j \quad\} \quad \forall e \in E,\ j \in \{1, \ldots, T_e\} ,\\ \omega_{k,e} &:= \max\{x_{k,e} - \lfloor x_{k,e} \rfloor, \lceil x_{k,e} \rceil - x_{k,e}\} \quad \forall e \in E,\ k \in \mathcal{K} . \end{aligned}$$

min.inf.: This is the other classic rule. It is the opposite of the first: The variables whose fractional parts are farthest away from 0 and 1 get the highest priority. The according weight function is

$$\begin{aligned} \omega_e^j &:= \min\{\hat{x}_e^j - \lfloor \hat{x}_e^j \rfloor, \quad \lceil \hat{x}_e^j \rceil - \hat{x}_e^j \quad\} \quad \forall e \in E,\ j \in \{1, \ldots, T_e\} ,\\ \omega_{k,e} &:= \min\{x_{k,e} - \lfloor x_{k,e} \rfloor, \lceil x_{k,e} \rceil - x_{k,e}\} \quad \forall e \in E,\ k \in \mathcal{K} . \end{aligned}$$

up.inf.: This rule works on the following assumption: In a "dive", the "up" node of the an iteration is processed next. This node should be an improvement. Hence, variables get a high priority if their fractional part is low.

$$\begin{aligned} \omega_e^j &:= \lceil \hat{x}_e^j \rceil - \hat{x}_e^j \qquad \forall e \in E,\ j \in \{1, \ldots, T_e\} ,\\ \omega_{k,e} &:= \lceil x_{k,e} \rceil - x_{k,e} \quad \forall e \in E,\ k \in \mathcal{K} . \end{aligned}$$

min.cutoff: This rule exploits the ordering constraints (5.30). For an edge $e \in E$ and a $j \in \{1, \ldots, T_e\}$, setting $\hat{x}_e^j$ to 1 does the same to $\hat{x}_e^1, \ldots, \hat{x}_e^{j-1}$ and setting $\hat{x}_e^j := 0$ zeroes the variables $\hat{x}_e^{j+1}, \ldots, \hat{x}_e^{T_e}$ as well. The following weight function is similar to min.inf., except that it considers the implicitly fixed variables for incremental link designs as well:

$$\begin{aligned} \omega_e^j &:= \min\left\{ \sum_{n=j}^{T_e} \hat{x}_e^n, \qquad \sum_{n=1}^{j} (1 - \hat{x}_e^j) \right\} \quad \forall e \in E,\ j \in \{1, \ldots, T_e\} ,\\ \omega_{k,e} &:= \min\{\ x_{k,e} - \lfloor x_{k,e} \rfloor, \lceil x_{k,e} \rceil - x_{k,e}\} \quad \forall e \in E,\ k \in \mathcal{K} . \end{aligned}$$

		inc. link designs				c.s.c.c.		
		$\hat{x}_e^1$	$\hat{x}_e^2$	$\hat{x}_e^3$	$\hat{x}_e^4$	$x_{k_1,e}$	$x_{k_2,e}$	$x_{k_3,e}$
		.95	.85	.60	.20	17.90	2.55	65.25
max.inf.	ω	.95	.85	.60	.80	.90	.55	.75
	order	1st	2nd		3rd			4th
min.inf.	ω	.05	.15	.40	.80	.10	.45	.25
	order			2nd	4th		1st	3rd
up.inf.	ω	.05	.15	.40	.80	.10	.45	.75
	order			4th	1st		3rd	2nd
min.cutoff	$\sum_j^{T_e} x$	2.60	1.65	.80	.20			
	$\sum_1^j 1 - x$	.05	.20	.60	1.40			
	ω	.05	.20	.60	.20	.10	.45	.25
	order		4th	1st			2nd	3rd
ld.min.cutoff	ω	2.05	2.20	2.60	2.20	.10	.45	.25
	order	4th	2nd	1st	3rd			

Table 5.2: Branching rule examples. *For four incremental link design and three commodity set capacity components and the shown fractional values, the weight function of all branching rule is evaluated. The four variables of highest branching priority are marked.*

ld.min.cutoff: We assume that constantly priorizing link designs over commodity set capacity components can benefit the branching process, because it quickly fixes a network topology and capacities before alloting the available capacity to the commodity sets. This priorizing can be added to all of the previous four rules. Running prestudies revealed that extending min.cutoff looks most promising. Hence, the according weight function is

$$\omega_e^j := \min\left\{ \sum_{n=j}^{T_e} \hat{x}_e^n, \qquad \sum_{n=1}^{j} (1 - \hat{x}_e^j) \right\} + 2 \quad \forall e \in E,\ j \in \{1, \dots, T_e\}\ ,$$

$$\omega_{k,e} := \min\{\ x_{k,e} - \lfloor x_{k,e} \rfloor\, , \lceil x_{k,e} \rceil - x_{k,e} \} \qquad \forall e \in E,\ k \in \mathcal{K}\ .$$

Table 5.2 shows an example for these five rules. In Section 6.3.2, they are analyzed using the data sets that are presented there.

5.5 Cutting Planes

5.5.1 Node Cut Inequalities

In Section 4.1.3, we proposed a heuristic to separate node cut inequalities (4.17) for a fixed node cut. Our implementation uses a fixed pool of minimal node cuts that are determined in advance. In, [BBC00] an enumeration algorithm for minimal node cuts in connected graphs is proposed. This algorithm is shown as Algorithm 4, and runs in $\mathcal{O}(|V|^3)$ time per cut. This algorithm is executed until the node cut pool reaches a certain size or the algorithm terminates. Prestudies hinted that a maximum size of $3|V|$ gives good results, hence this value is used. The separation algorithm iterates through the node cut pool until either the separation heuristic from Section 4.1.3 finds a violated inequality or all known node cuts are checked without result.

5.5.2 Other Inequalities

Metric Inequalities

In Section 5.2, we introduced a decomposition with external path LPs. We showed how fractional capacities derived from points in $P_{\mathrm{HC,F}}$ can be used in conjunction with a path LP to either create a feasible routing or a violated metric inequality (2.3).

Algorithm 4: Generate set $\mathcal{S}(G)$ of all minimal node cuts in graph G

Input : connected graph G
Output : set $\mathcal{S}$ of all minimal node cuts in G

$\mathcal{S} \leftarrow \varnothing$
for *every* $v \in V$ **do**
4.1   **for** *every connected component* $C \subseteq V$ *of* $G - (\{v\} \cup \Gamma(v))$ **do**
        $T \leftarrow \Gamma(C) \setminus C$
        if $T \notin \mathcal{S}$ **then**
            { T is new minimal node cut }
            $\mathcal{S} \leftarrow \mathcal{S} \cup \{T\}$
            mark T as "new"

4.2 **while** $S \in \mathcal{S}$ *with "new" mark exists* **do**
    clear mark of S
    for *every* $v \in S$ **do**
4.3   **for** *every connected component* $C \subseteq V$ *of* $G - (S \cup \Gamma(v))$ **do**
            $T \leftarrow \Gamma(C) \setminus C$
            if $T \notin \mathcal{S}$ **then**
                { T is new minimal node cut }
                $\mathcal{S} \leftarrow \mathcal{S} \cup \{T\}$
                mark T as "new"

{ $\mathcal{S}$ contains all minimal node cuts in G. }

Hence, testing a solution for feasibility using path LPs is a separation algorithm as a byproduct. Our implementation uses inequalities found in that matter as cutting planes and adds them to $P_{\mathrm{HC,F}}$ in the obvious way.

Strengthened Metric Inequalities

This class of inequalities was proposed in [AGW96]. We give a short summary here: Let

$$\sum_{e \in F} \mu_e y_x(e) \geqslant b \tag{5.35}$$

be a valid inequality for $X_{\mathrm{HC,R}}$, where $F \subseteq E$ and $\mu_e > 0$ for each $e \in F$. Assume that $\mu_e C_e^0$ and $\mu_e C^\ell$ are positive integrals for each $e \in E$ and $\ell \in \mathcal{L}(e)$. For rational μ_e, this can be achieved by scaling μ and b. For each $e \in F$, let

$$\hat{c}_e^0 := C_e^0 \ , \quad \hat{c}_e^t := C^{\ell_e^t} - \sum_{j=1}^{t-1} \hat{c}_e^j \quad \forall t \in \{1, \ldots, T_e\} \ . \tag{5.36}$$

Using incremental link designs, (5.35) translates to

$$\sum_{e \in F} \left(\mu_e \hat{c}_e^0 + \sum_{t=1}^{T_e} \mu_e \hat{c}_e^t \hat{x}_e^t \right) \geqslant b \ . \tag{5.37}$$

Let $b' := b - \sum_{e \in F} \mu_e \hat{c}_e^0$. Then (5.37) reads

$$\sum_{e \in F} \left(\sum_{t=1}^{T_e} \mu_e \hat{c}_e^t \hat{x}_e^t \right) \geqslant b' \ . \tag{5.38}$$

Now let g be the greatest common divisor of the integrals $\mu_e \hat{c}_e^t$ for $e \in E$ and $t \in \{1, \ldots, T_e\}$. Scaling (5.38) by $\frac{1}{g}$ preserves coefficient integrality on the left hand side, hence rounding up the right hand side is feasible. Because the variables are binary, coefficients can be capped at the resulting right hand side. The resulting inequality

$$\sum_{e \in F} \left(\sum_{t=1}^{T_e} \min \left\{ \frac{\mu_e \hat{c}_e^t}{g}, \left\lceil \frac{b'}{g} \right\rceil \right\} \hat{x}_e^t \right) \geqslant \left\lceil \frac{b'}{g} \right\rceil \tag{5.39}$$

is called *strengthened metric inequality* and valid for $X_{\mathrm{HC,R}}$ (w.r.t. the variable transformation for incremental link designs). In [Wes00], a separator for these inequalities using a pool of valid inequalities (5.35) is introduced, and we use the implementation proposed there.

Band Inequalities

These inequalities were introduced in [SD94]. Again, let

$$\sum_{e \in F} \mu_e y_x(e) \geqslant b \tag{5.40}$$

be a valid inequality for $X_{\mathrm{HC,R}}$ with $F \subseteq E$ and $\mu_e > 0$ for each $e \in F$. Let $(t_e)_{e \in F}$, $t_e \in \{0, \ldots, T_e\}$ be a choice of incremental capacities that do not fulfill for (5.40), i.e.,

$$\sum_{e \in F} \mu_e \left(\sum_{j=0}^{t_e} \hat{c}_e^j \right) < b \; . \tag{5.41}$$

Then at least one larger capacity has to be chosen, which gives the *band inequality*

$$\sum_{e \in F : t_e < T_e} \hat{x}_e^{t_e + 1} \geqslant 1 \; . \tag{5.42}$$

Again, we use the implementation proposed in [Wes00] to separate such inequalities based on a pool of valid inequalities.

5.6 Routing Mode Impact

Our implementation offers three modes to deal with routings. While we abstracted from routing details throughout this work and hence are not in the position to discuss routing issues in depth, these routing modes have an impact on component installations and our branch-and-cut algorithm. They are briefly presented here.

Plain Routing

This is the default mode, and we implicitly assumed this mode in the previous chapters. In this mode, a routing for a commodity set consists of flow paths with flow values, such that the commodity set capacities are not exceeded. The flow values may be fractional, but commodity set capacity components are required to be integral, thereby providing each commodity set a sub-graph with integral capacities for private use.

Fractional Routing

This mode differs from the previous one in that commodity set capacities may be fractional as well. Hence, scaling commodities such that the according routing unit U^k becomes 1 is always possible. Then, there is no need anymore to keep the distinction between different commodity sets. Instead, all commodities are aggregated in a single commodity set. In the end, there is hence a single commodity set with a routing unit of 1, where fractional values for the commodity set capacity component are feasible. The preprocessing detects this case and ensures that the commodity set capacity component is removed, and constraints are updated to use the capacities provided by link designs and the preinstalled capacities directly. This mode is assumed to be easier to solve than the "plain" mode, because there are

- potentially less commodities,
- less variables to deal with in the branch-and-cut, and
- metric inequalities that reference link designs directly.

Integral Routing

In this mode, the flow values are required to be integral, everything else being as in the "plain" mode. While on first sight this seems to be unrelated to this work, there are in fact implications on the algorithms presented in this chapter.

There is no practicable algorithm known to us that can produce an integral routing. Instead, we employ a heuristic that hooks into the postprocessing step 3.2 of our branch-and-cut node procedure (Algorithm 3). This heuristics gets a fractional routing as input. It first rounds the fractional path flow values of each demand to integral ones with the same sum, which is the according demand value. The resulting routing may exceed the capacities provided by the component installations.

Then, flow paths are rerouted: The heuristic picks an edge whose capacity is exceeded, and a path that crosses this edge. It then attempts to compute a new path that uses edges with spare capacity only using a shortest path algorithm. This procedure is iterated until it stalls, i.e., it failed to remove an edge overload for a fixed number of times.

The resulting routing is integral, but may still exceed the installed capacities. Now, the HC IP is used to compute a new component installation that suffices for the routing. If this is successfull, a feasible solution has been identified.

This approach cannot guarantee that a feasible solution is always found if one exists. Hence, there is no guarantee that the optimal solution of the original problem (5.1) is found, even if the whole branch-and-cut tree is enumerated. The branch-and-cut algorithm can only use feasible solutions for the sub-tree pruning, and the heuristic may produce solutions of higher cost than the one it gets as input. Hence, the algorithm often fails to detect that a sub-problem is empty or contains no better solution than already known ones. This further slows down the algorithm. Consequently, this mode is assumed to be the most difficult to solve.

Chapter 6

Applications

In this chapter, we show how to apply our models to the planning scenarios that were introduced in Chapter 1. We present results from running our algorithms with several data sets, all of which are rooted in network planning studies that were conducted for several telecommunication companies. This proves that our models can be put into practice. Additionally, we use these data sets for further analysis of certain settings for algorithms and heuristics.

Test Environment

We implemented the models and algorithms on top of the existing optimization tool DISCNET [Wes00]. In the backend, we used CPLEX 8.0 to solve MIPs and LPs, and LEDA 4.1 for graph data structures and graph algorithms.

All tests were conducted on a Linux host with a 1.0 GHz processor and 256 Megabytes of RAM. The machine was dedicated to these tests, and did not run any other software but the basic operating system. For all tests, a time limit of 60 minutes was imposed. Cost values are scaled w.r.t. a reference solution: The best known solution to a problem with default algorithm settings always has a value of 100.0.

These default settings include the following: The bound strengthening heuristic runs at level 3, min.cutoff is used as branching rule, all four separators (node cut, metric, strengthened metric, and band) are active, and routing mode is "plain".

Result Presentation

When presenting results, we will use similar tables for the different planning scenarios, showing the same properties for different applications. Table 6.1 briefly lists all such properties and their exact meanings to avoid repetition.

6.1 SDH Scenario

Scenario 1.1 deals with the capacity hierarchy defined by the SDH standard, and according transmission and switching hardware. On the links, fiber pairs are installed and operated using predetermined capacities, which are split up into virtual containers for the actual routing. Node hardware includes ADMs and DXCs as routers. These are equipped with router cards, to which the fibers connect.

6.1.1 Application

It is possible to solve problem instances of this scenario using only the algorithmic framework introduced in Chapter 5. The transition from SDH hardware elements to HC components is straightforward:

Problem-related	
Nodes	$\lvert V \rvert$, number of nodes.
Edges	$\lvert E \rvert$, number of edges.
Preinstalled capacities	$\lvert \left\{ e \in E \mid C_e^0 > 0 \right\} \rvert$, number of edges with a preinstalled capacity.
Zero-cost capacity edges	Number of edges on which capacities of zero cost are available.
Avg. link designs / edge	$\frac{1}{\lvert E \rvert} \sum_{e \in E} \lvert \mathcal{L}(e) \rvert$, average number of link designs per edge, before preprocessing.
Avg. node designs / node	$\frac{1}{\lvert V \rvert} \sum_{v \in V} \lvert \mathcal{D}(v) \rvert$, average number of node designs per node, before preprocessing.
Modules	$\lvert \mathcal{M} \rvert$, number of modules, before preprocessing. These modules are available on all nodes.
Commodity sets	$\lvert \mathcal{K} \rvert$, number of commodity sets, number of commodity set capacity components.
Demands	Total number of demands.
Survivability conditions	Whether there are additional survivability conditions.
Components in global pool	Number of components for which global bounds exist.
Components under expansion tracking	Number of components for which change limits exist.
Solutions and Bounds	
Best solution value	Cost of best known solution.
Final lower bound	Lower bound on algorithm terminantion.
Root node lower bound	Lower bound after adding cuts at the branch-and-cut's root node.
Initial lower bound	Lower bound of the initial relaxation.
Final gap	$100 \frac{\text{solution} - \text{lowerbound}}{\text{solution}}$, gap between solution and final lower bound, in percent.
Relative node hardware cost	Part of the best known solution's cost for hardware equipment for the nodes, in percent.
Behaviour	
Branch-and-cut iterations	Number of nodes processed during the branch-and-cut algorithm.
Iteration of best solution	Branch-and-cut iteration number in which the best solution was identified.
Total runtime	Total runtime, as minutes:seconds.
Time for bound strengthening	Time spent for bound strengthening, as minutes:seconds.
Identified node	Number of identified violated node cut inequalities (4.17).

Table 6.1: Identifiers in result tables. *For each of the short row headers in the tables of this chapter, an explanation is given here.*

	sdh1	sdh1b	sdh2	sdh3	sdh3b	sdh4	sdh5
Nodes	18	18	22	54	54	94	127
Edges	33	33	34	81	81	178	200
Preinstalled capacities	13	0	25	17	0	0	148
Zero-cost capacity edges	0	22	0	0	44	0	0
Avg. link designs / edge	6.0	6.0	6.0	7.0	7.0	3.0	15.0
Avg. node designs / node	5.0	5.0	5.2	7.2	7.2	2.5	8.0
Modules	4	4	4	4	4	13	4
Commodity sets	2	2	1	2	2	1	3
Demands	101	101	135	290	290	166	1475
Survivability conditions	No	No	Yes	No	No	No	Yes

Table 6.2: SDH problem instances. *This tables shows for the five data sets, both basic and the two "b" variants, some characteristic properties.*

- Each link design $\ell \in \mathcal{L}$ models a certain setup of fiber pairs and SDH capacities. The cost of such a setup depends solely on the amount of fibers, because capacity is determined by router cards in the nodes, assuming all used fibers are of sufficiently high quality.
- Each module $m \in \mathcal{M}$ represents a pair of router cards. The ports of the cards are modelled by interfaces in $\mathcal{I}$. The slot count S^m specifies the number of router slots occupied by the cards.
- A node design $d \in \mathcal{D}$ models a particular pre-determined setup of multiplexers. Such a setup consists of a chain of ADMs and DXCs that are interconnected at their aggregates. The aggregates of the first and the last ADM in the chain as well as all tributary router card slots are available for use. The cost of a node design is the total cost for the ADMs and the linking hardware.

For each virtual container VC–N, a commodity set and according commodity set capacity component is used. Hence, all demands that are specified for a particular virtual container belong to the according commodity set. The fact that ADMs and DXCs are limited in which virtual containers they are able to switch is ignored, as are tributary connections at gateway nodes.

6.1.2 Results

Data Sets

We ran tests using five data sets, all of which come from real-world application of our implementation. An overview over size and structure of these problems is given in Table 6.2. The sets have a common cost structure: Fibers are the most expensive part, because this scenario assumes that new fibers have to be laid physically into ground. It turns out that the hardware required for an average switching center is worth less than five kilometers of fiberoptic cabling. The cost of a link design scales linear with length of the link on which it is to be installed, and with the number of fiber pairs that have to be laid. Operating four fiber pairs, each with a capacity of STM–4, costs four times the cost of one fiber pair with this capacity. The same bandwidth can be achieved by operating one pair with a capacity of STM–16, which costs the same as one STM–4-operated pair. This leads to a sawtooth-like capacity/cost relation. Figure 6.1 shows these coherences.

Two of the data sets have an additional "b" variant with a different cost structure. These variants are a special form of expansion planning problems. They are based on a solution for the basic variant where only half of the demands are routed. All fibers that are present in this reference solution are available for free in the "b" problem. This does not use preinstalled capacities, it just changes the cost structure such that the first few fibers on each edge come at zero cost, independent of the SDH capacity operated over them.

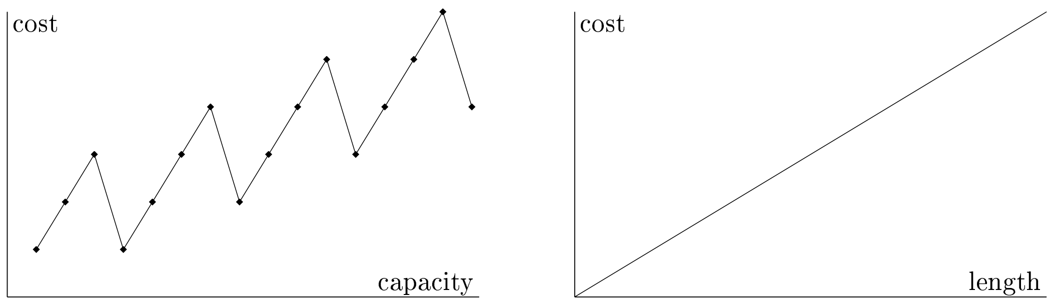


Figure 6.1: SDH cost structure. *Left: The relationship between SDH capacity and link design cost exhibits a strong sawtooth-like structure. Right: Cost scales linear with fiber length.*

Results

We ran our implementation three times on each data set, with different demand scalings. In the default setting "100%", demands are in their original state as determined by the traffic forecasting process of a telecommunication company. In "50%" and "200%", the demand values are halfed resp. doubled to get some insight on the relation between available and required capacities.

Table 6.3 shows the results of the test runs. At 200%, there are three problems where no optimal solution value is given. These problems are infeasible in the sense that there exists no hardware configuration that supports a feasible demand routing.

A few observations can be made:

- Because cost values at 50% are not 100, the demands are not misscaled such that a capacity assignment using the cheapest link design on all edges suffices. For `sdh1`, `sdh1b`, `sdh2` and `sdh5`, the demands do not necessarily require that the largest available capacities are installed, because the cost values at 200% are not 100 either. The infeasibility of `sdh3`, `sdh3b`, and `sdh4` hints that for these instances nearly the largest capacities are needed to fulfill the original demands, so that they do not suffice with doubled demands.

- For the basic variants, hardware equipment takes a minor part of solution cost. However, hardware still plays a key role because of the limits it imposes on the capacities.

- The results for the "b" variants further emphasize the high cost of fibers. At 50%, the best known solutions to both `sdh1b` and `sdh3b` use no fibers in addition to the free ones. At 100%, there are two additional fibers in the solution for `sdh1b`, and at 200%, there are 14. Installation of these additional fibers comes together with an exploding solution cost.

- For the small problems `sdh1`, `sdh1b`, and `sdh2`, good solutions can be found within the time limit of one hour. A gap of about 10% is sufficient for most practical applications.

- The problems `sdh4` and `sdh5` are very large. Finding any solution within an hour of computing time is good for itself, and finding solutions with a gap below 50% has to be considered a success.

- Feasible solutions are found quickly. Afterwards, there is a long time where branch-and-cut nodes are explored to raise the lower bound without finding new solutions. This behaviour is best to be seen with `sdh2`, where the best known solutions are in fact optimal solutions and these were found within the first few iterations of the branch-and-cut algorithm.

- Solving problems of Scenario 1.1 is possible. Solutions can be produced within reasonable time, although lower bound and gap could stand some improvement.

		sdh1	sdh1b	sdh2	sdh3	sdh3b	sdh4	sdh5
Best solution	50%	84.2	69.2	99.8	78.9	20.8	93.3	88.4
value	100%	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	200%	140.9	1859.4	114.8	—*	—*	—*	122.7
Final lower	50%	70.3	63.7	99.7	74.1	15.1	55.9	71.1
bound	100%	90.3	91.0	99.9	89.0	61.9	60.9	72.7
	200%	124.5	1607.7	114.5	—	—	—	85.2
Root node	50%	60.9	59.9	96.4	73.0	14.6	53.8	70.9
lower bound	100%	79.3	85.3	96.5	88.5	61.4	60.7	72.7
	200%	120.9	1349.3	104.8	—	—	—	85.2
Initial lower	50%	57.0	56.6	77.1	63.6	14.0	20.9	70.1
bound	100%	77.0	74.8	79.4	83.1	61.0	35.2	71.1
	200%	120.2	1337.5	90.5	—	—	—	80.7
Final gap	50%	16.5	8.0	0.1	6.1	27.2	40.1	19.6
	100%	9.7	9.0	0.1	11.0	38.1	39.1	27.3
	200%	11.6	13.5	0.3	—	—	—	30.6
Relative node	50%	1.6	100.0	5.8	8.5	100.0	11.7	6.0
hardware cost	100%	1.7	96.6	6.0	9.9	34.8	10.7	6.0
	200%	2.2	8.1	6.7	—	—	—	7.0
Branch-and-cut	50%	10397	9752	1536	885	874	380	4
iterations	100%	13491	15762	590	501	987	173	2
	200%	14470	16132	1456	—	—	—	2
Iteration of	50%	844	1623	4	391	837	57	2
best solution	100%	4901	163	2	386	221	63	1
	200%	11113	1722	8	—	—	—	2
Total runtime	50%	60:00	60:00	33:29	60:00	60:00	60:00	60:00
(min:sec)	100%	60:00	60:00	14:24	60:00	60:00	60:00	60:00
	200%	60:00	60:00	55:30	—	—	—	60:00

Table 6.3: Results for SDH problems. *For the seven SDH problems, each with half, full, and doubled demands, some properties of the according test run are given.*
**) These problems are infeasible at 200%.*

	`opt1`	`opt2`	`opt3`
Nodes	14	17	24
Edges	21	26	36
Preinstalled capacities	0	0	0
Zero-cost capacity edges	21	26	0
Avg. link designs / edge	23.0	47.0	12.0
Avg. node designs / node	1.0	1.0	1.0
Modules	4	4	4
Commodity sets	1	1	1
Demands	33	58	91
Survivability conditions	No	No	Yes

Table 6.4: Optical problem instances. *There are three data sets. They all have only one commodity set due to the nature of the problem. All of these data sets feature a large amount of link designs.*

6.2 Optical Network Scenario

In Scenario 1.2, placement of fibers, WDM systems, optical switching devices, wavelength converters, and regenerators is discussed. Planning task is to satisfy demands by a lightpath configuration.

6.2.1 Application

An approach to solve planning problems of this scenario was proposed and implemented in [ZKW02]. Our models and algorithms are used in an intermediate step of a larger algorithm. The overall approach to solve problems of Scenario 1.2 is:

1. For each edge, generate the set of all feasible and usable pairs of fibers and WDM systems.
2. Using sets of these pairs as link designs, determine a solution that allows a routing, ignoring the necessary wavelength assignment.
3. Determine wavelength assignment, add converters where required.

Here, we concentrate on step 2 only, because it is the one that is solved using our implementation. The link design set resulting from step 1 can become very large. There, all combinations that may appear in solutions have to be generated, which are all feasible combinations of fibers and WDM system within reasonable bounds. The heuristic mentioned in step 3 might add cost to solutions by placement of converters. A lower bound obtained in step 2 is however valid for the overall problem.

We allow placement of multiple OXCs at each node. While equipment needed to interconnect them and resulting effects are neglected, the model extension “tributary deprivation” is used to model connections to other network layers at all nodes.

There exists a single node design that is available on all nodes. It is simply a placeholder allowing modules to be installed. For each OXC and OADM, there is a single module type, each offering a certain amount of the single existing interface, which models an optical port.

There is always only one commodity set, because all demands specify an amount of lightpaths resp. channels. The routing unit of the commodity set is hence 1.

6.2.2 Results

Data Sets

For computational tests, three data sets were used. Again, they are stemming from real-world applications. An overview over these data sets is given in Table 6.4.

The cost of a link design consists of three independent parts, namely the cost for the fibers, the two WDM systems in a link’s end nodes and the regenerators at regular distances. Although it

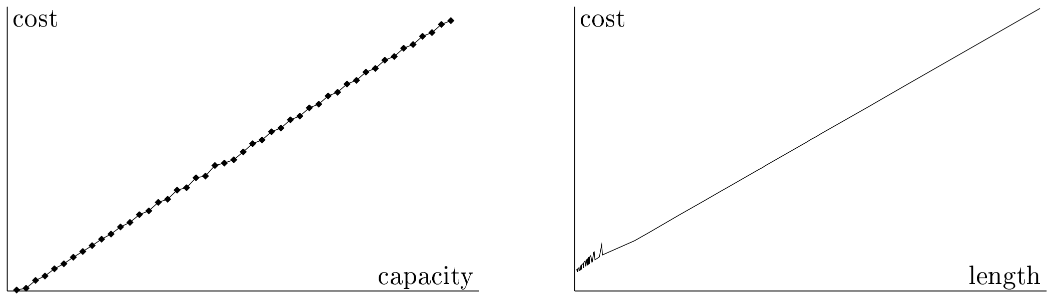


Figure 6.2: Optical Cost Structure. *The cost of a link design roughly scales affine with both capacity and edge length.*

name		opt1	opt2	opt3
Best solution	50%	79.5	73.8	62.8
value	100%	100.0	100.0	100.0
Final lower	50%	34.9	39.8	49.7
bound	100%	63.4	70.2	87.4
Root node	50%	32.2	39.0	48.5
lower bound	100%	61.8	69.4	85.8
Initial lower	50%	32.1	37.8	36.6
bound	100%	61.8	68.1	64.0
Final gap	50%	56.1	46.1	20.8
	100%	36.6	29.8	12.6
Relative node	50%	95.7	90.2	60.1
hardware cost	100%	76.1	75.9	54.8
Branch-and-cut	50%	17334	510	342
iterations	100%	12668	474	373
Iteration of	50%	498	451	242
best solution	100%	2034	101	125
Total runtime	50%	60:00	60:00	60:00
(min:sec)	100%	60:00	60:00	60:00

Table 6.5: Results for optical problems. *For the optical problems* `opt1`, `opt2`, *and* `opt3`, *results are given for test runs with half and full demands.*

seems that the resulting cost structure is very complex, it turns out to be nearly an affine function on the product of capacity and length. Figure 6.2 visualizes these cost functions. Preinstalled capacities do not appear in the data sets, but preinstalled fibers do. They are reflected by reduced link design costs on corresponding edges.

The placeholder node design comes at no cost. The modules carry the cost of the OXCs and OADMs. Because wavelength converters do not appear in this step, they do not contribute to a solution's cost value.

Results

Our implementation was run on the three instances, again with scaled demands. The preprocessing that produces the data first generates an upper bound for required capacities from the demands. It does not produce link designs with a capacity larger than this bound, hence running a test with doubled demand values does not provide anything useful and was thus skipped. The results can be found in Table 6.5. The following conclusions seem adequate:

- Judging from the bad gap and the fact that no problem could be solved to optimality, these problems seem more difficult to solve than those of Scenario 1.1, even though the networks are smaller. This is not surprising considering the large amount of link designs available on each edge. Branching on a link design variable of some edge is unlikely to have a great impact, because it leaves many possibilities on the edge itself, and other edges can compensate many fixations, until there are many variable fixations on practically all edges.

- These thoughts also explain why the lower bound is that low and does not improve perceptibly during the algorithm. Difficulties with the lower bound can be observed best with problem `opt1`: The best known solution was found within 3% resp. 16% of the branch-and-cut iterations. Then, the algorithm spent a long time exploring branch-and-cut nodes without any success.

- The instances with scaled demands have an average solution cost value of about 72 percent of the unscaled instances. Hence, the demands are not that small, compared to the capacities, such that no real decisions are to be made, and therefore hardware decisions at the nodes influences solutions. Otherwise, the average cost would be around 100 or 50.

- It seems that the large amount of link designs is the major reason why our algorithms perform so poor on these data sets. Therefore, it is advisable to seek alternatives. Analysis of the data reveals that both cost and interface consumption are nearly affine functions of capacity, hence using a single integral variable instead of many binary ones could greatly improve results. This is no contra against our model, but rather a plea for another model extension: By allowing a link design $\ell \in \mathcal{L}$ to have a upper bound $\overline{x}_{\ell,e} > 1$ for each edge $e \in E$, this change could be simply incorporated into the model. Considering the limited time, and because this would invalidate every result that depends on the assumption that link design variables are binary, we renounced implementing this extension.

- Solving problems of Scenario 1.2 is possible. Solutions can be produced within reasonable time, although lower bound and gap could stand some improvement.

6.3 Algorithm Analysis

Now that we introduced two sets of problem instances that have a different structure, while both can be solved by the algorithms introduced in Chapter 5, we use these data sets to further analyze the behaviour of algorithms and heuristics.

6.3.1 Bound Strengthening Impact

In Section 5.1, we introduced three levels of a bound strengthening heuristic. This heuristic attempts to strengthen variable bounds and problem parameters in order to get a better LP relaxation and to speed up the branch-and-cut algorithm.

We ran our algorithm on all basic SDH data sets as well as `opt1` and `opt2`, once for each heuristic level and once without bound strengthening. The results are summarized in Table 6.6. While the heuristic attempts to improve many bounds, we only list successful updates for upper bounds of node and link design variables, as well as lower bounds of module variables. The first eliminate variables in the central LP relaxation, and the latter should improve the LP lower bounds.

A few conclusions can be drawn:

- For the problems from the optical network scenario, the heuristic does nothing. This is not surprising, since these data sets have no preinstalled capacities and allow unlimited numbers of OXCs and OADMs. Therefore, all components can be installed where available, and none are fixed implicitly.

		sdh1	sdh2	sdh3	sdh4	sdh5	opt1	opt2
Best solution value	None	100.6	100.0	100.9	98.9	101.0	100.0	100.2
	BSH–1	101.1	100.0	99.5	100.7	100.1	100.0	100.0
	BSH–2	100.0	100.0	99.2	103.3	98.1	100.0	100.0
	BSH–3	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Final lower bound	None	89.5	98.8	88.9	60.8	71.5	63.3	70.2
	BSH–1	89.4	99.5	88.9	61.0	72.7	63.4	70.2
	BSH–2	89.8	99.9	89.0	60.9	72.7	63.4	70.2
	BSH–3	90.3	99.9	89.0	60.9	72.7	63.4	70.2
Root node lower bound	None	78.9	95.4	88.4	60.6	71.5	61.8	69.4
	BSH–1	78.9	96.1	88.4	60.6	72.7	61.8	69.4
	BSH–2	79.1	96.5	88.5	60.7	72.7	61.8	69.4
	BSH–3	79.3	96.5	88.5	60.7	72.7	61.8	69.4
Initial lower bound	None	76.8	78.2	83.0	34.6	69.8	61.8	68.1
	BSH–1	76.8	79.0	83.0	34.6	71.1	61.8	68.1
	BSH–2	77.0	79.4	83.1	35.2	71.1	61.8	68.1
	BSH–3	77.0	79.4	83.1	35.2	71.1	61.8	68.1
Final gap	None	11.0	1.2	11.9	38.5	29.2	36.7	29.9
	BSH–1	11.7	0.5	10.7	39.4	27.3	36.6	29.8
	BSH–2	10.2	0.1	10.3	41.1	25.9	36.6	29.8
	BSH–3	9.7	0.1	11.0	39.1	27.3	36.6	29.8
Branch-and-cut iterations	None	10379	2256	517	226	2	12334	473
	BSH–1	10555	1058	399	286	2	12710	473
	BSH–2	12355	590	427	235	2	12676	473
	BSH–3	13491	590	501	173	2	12668	474
Iteration of best solution	None	6257	2	391	63	2	2109	345
	BSH–1	1196	2	211	64	2	2034	101
	BSH–2	6847	2	359	66	2	2034	101
	BSH–3	4901	2	386	63	1	2034	101
Total runtime (min:sec)	None	60:00	60:00	60:00	60:00	60:00	60:00	60:00
	BSH–1	60:00	26:06	60:00	60:00	60:00	60:00	60:00
	BSH–2	60:00	14:26	60:00	60:00	60:00	60:00	60:00
	BSH–3	60:00	14:24	60:00	60:00	60:00	60:00	60:00
Time for bound strengthening	None	0:00	0:00	0:00	0:00	0:00	0:00	0:00
	BSH–1	0:00	0:00	0:00	0:00	0:00	0:00	0:00
	BSH–2	0:00	0:00	0:02	0:08	0:09	0:00	0:01
	BSH–3	0:01	0:01	0:07	0:11	0:24	0:00	0:01
Link design eliminations	None	0	0	0	0	0	0	0
	BSH–1	0	0	0	0	0	0	0
	BSH–2	0	0	0	0	0	0	0
	BSH–3	19	0	0	0	0	0	0
Node design eliminations	None	0	0	0	0	0	0	0
	BSH–1	0	15	18	0	72	0	0
	BSH–2	15	20	24	1	72	0	0
	BSH–3	15	20	24	1	72	0	0
Module lower bound improves	None	0	0	0	0	0	0	0
	BSH–1	0	0	0	0	0	0	0
	BSH–2	27	24	34	48	82	0	0
	BSH–3	27	24	34	48	82	0	0

Table 6.6: Impact of the bound strengthening heuristic. *The seven data sets were run without the heuristic, and with the heuristics, once for each of the available levels. While successfully eliminating variables, there is no improvement on the initial lower bound. The heuristic does reduce the size of the branch-and-cut tree, but this can only be observed for one data set because the others did not finish within the time limit.*

- For the SDH problems, the heuristic successfully eliminates some variables, and is able to modify problem parameters. BSH–2 and BSH–3 reveal differences only in eliminating link design variables in `sdh1`. This is due to the larger edge relaxations with commodity set capacity component constraints for emanating demands.

- There is practically no effect of the heuristic on any lower bound. This is surprising, but probably due to the fact that the heuristic can only cut off small fractions of single variables, thereby increasing the lower bound by the cost of these small fractions, which happen to be too small to show effect.

- The heuristic runs in reasonable time, compared to the total runtime of the full application. Solutions and gap do not get worse by applying the heuristic. Therefore, running it at least does not harm.

- The problem `sdh2` shows the behaviour that we expected for all data sets according to prestudies: By reducing the size of the branch-and-cut tree, the time until the algorithm terminates with an optimal solution is greatly reduced. Given that `sdh2` is in fact the only instance for which the branch-and-cut terminated within the time limit, evidence for this assumption is poor.

6.3.2 Branching Rule Comparison

We tested the branching rules introduced in Section 5.4.2 using the basic data sets, except for `sdh5`. The latter was left out because the branch-and-cut algorithm did only two iterations within the given time limit, hence comparing application of different branching rules is useless for this data set.

Table 6.7 gives an overview over using the different branching rules. In the runs marked with *, our implementation did not find any feasible solution within the time limit of one hour. Notice that the lower bound at the root node is always the same, independent of the branching rule, because no rule is applied yet at that point. These values are listed separately merely for consistency of the table layout.

The following observations seem appropriate:

- The influence of the branching rules in solving the optical problems is minimal, if not nonexisting. It was already seen that these problems have a structure that is difficult to attack using branch-and-bound methods, and this observation is confirmed here again.

- The branching rule up.inf. was expected to ensure that good solutions are found very fast. This seems to be not true, given that this rule does not generally find the best solution in less iterations than the other rules.

- For the SDH problems, the two rules that employ cutoff weights, min.cutoff and ld.min.cutoff, perform significantly better than the three others. For `sdh1` and `sdh3`, the final gap is much better. They are also the only rules with which a solution to `sdh4` can be found. With `sdh2`, all rules allow to enumerate the branch-and-cut tree within time and find the same best solution right at the beginning, so no ranking is possible here.

6.3.3 Node Cut Inequalities

In Sections 4.1 and 5.5.1, we introduced node cut inequalities (4.17) and a separation heuristic. Table 6.8 lists how many violated inequalities the heuristic identified under several settings, and compares this amount to the inequalities identified by the strengthened metric and band separators.

Considering the results, three conclusions can be made:

		sdh1	sdh2	sdh3	sdh4	opt1	opt2	opt3
Best solution value	min.inf.	110.0	100.0	107.6	∞^*	100.0	100.4	103.2
	max.inf.	108.6	100.0	106.9	∞^*	99.9	100.8	101.7
	up.inf.	111.3	100.0	109.0	∞^*	99.9	102.1	102.8
	ld.min.cutoff	100.0	100.0	98.6	100.0	100.0	100.0	100.0
	min.cutoff	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Final lower bound	min.inf.	84.0	99.9	88.7	62.6	63.3	70.4	87.5
	max.inf.	82.4	99.9	88.5	61.1	70.0	69.9	87.2
	up.inf.	82.1	99.9	88.5	61.1	70.0	69.9	87.2
	ld.min.cutoff	90.2	99.9	89.0	60.9	63.4	70.2	87.4
	min.cutoff	90.3	99.9	89.0	60.9	63.4	70.2	87.4
Root node lower bound	min.inf.	79.3	96.5	88.5	60.7	61.8	69.4	85.8
	max.inf.	79.3	96.5	88.5	60.7	61.8	69.4	85.8
	up.inf.	79.3	96.5	88.5	60.7	61.8	69.4	85.8
	ld.min.cutoff	79.3	96.5	88.5	60.7	61.8	69.4	85.8
	min.cutoff	79.3	96.5	88.5	60.7	61.8	69.4	85.8
Final gap	min.inf.	23.6	0.1	17.6	—	36.7	29.8	15.2
	max.inf.	24.2	0.1	17.2	—	29.9	30.7	14.3
	up.inf.	26.2	0.1	18.8	—	29.9	31.5	15.1
	ld.min.cutoff	9.8	0.1	9.8	39.1	36.6	29.8	12.6
	min.cutoff	9.7	0.1	11.0	39.1	36.6	29.8	12.6
Branch-and-cut iterations	min.inf.	12921	590	1148	486	12933	445	186
	max.inf.	11963	590	370	240	12950	418	202
	up.inf.	12780	590	382	240	12707	385	198
	ld.min.cutoff	13802	590	427	168	12671	472	373
	min.cutoff	13491	590	501	173	12668	474	373
Iteration of best solution	min.inf.	6720	2	403	—	5865	54	176
	max.inf.	2287	2	223	—	605	395	74
	up.inf.	4992	2	370	—	196	330	187
	ld.min.cutoff	10491	2	217	63	2034	101	125
	min.cutoff	4901	2	386	63	2034	101	125
Total runtime (min:sec)	min.inf.	60:00	14:36	60:00	60:00	60:00	60:00	60:00
	max.inf.	60:00	14:47	60:00	60:00	60:00	60:00	60:00
	up.inf.	60:00	14:27	60:00	60:00	60:00	60:00	60:00
	ld.min.cutoff	60:00	14:25	60:00	60:00	60:00	60:00	60:00
	min.cutoff	60:00	14:24	60:00	60:00	60:00	60:00	60:00

Table 6.7: Branching rule comparison. *Results of running five branching rules from Section 5.4.2 on seven data sets.*

*) *No solution was found within the time limit.*

		sdh1	sdh2	sdh3	sdh4	sdh5	opt1	opt2	opt3
Identified node cut inequalities	Default	1	0	3	4	0	0	0	0
	50%	1	0	0	4	0	0	0	0
	200%	1	0	0	4	0	—*	—*	—*
	No b.str.	10	2	3	4	5	0	0	0
	BSH–1	10	0	10	4	0	0	0	0
	BSH–2	0	0	3	4	0	0	0	0
Inequalities by other separators	Default	40	8	393	825	111	5	16	112
	50%	20	3	493	908	35	5	16	166
	200%	20	3	493	908	35	—	—	—
	No b.str.	40	8	393	820	106	5	16	0
	BSH–1	40	8	417	815	111	5	16	112
	BSH–2	40	8	393	825	111	5	16	0

Table 6.8: Node cut separator success. *Comparing the number of violated inequalities that were found by the node cut separator to those found by the strengthened metric and band separators in various settings, the former does not seem to be a success.*
*) *The optical problems were not run at 200%.*

- The single node design used in the optical problems always has a switching capacity that is as large as the whole demand. Hence, there exists no valid node cut band, and therefore the node cut separator can never find a violated inequality.
- For the SDH problems, the separator does work, and it successfully identifies violated inequalities.
- It does however identify such a small amount of inequalities, especially if compared to the other two separators, that using the separator seems a waste of time.

To confirm the last assumption, we picked two settings where the separator did find inequalities and ran our implementation in these settings with disabled node cut separator. The first setting is the default, with data sets `sdh1`, `sdh3`, and `sdh4`. Results for these runs can be found in Table 6.9. The other setting is with disabled bound strengthening for all SDH problems, results being listed in Table 6.10. In fact, the solution cost and gap improve when the separator is disabled, and there is barely a difference at the other properties. But surprisingly, our implementation seems to be inable to find any solution for `sdh4` without the node cut separator. Therefore, although the separator usually does nothing useful, there are situations where it is key to finding solutions.

6.3.4 Routing Modes

In Section 5.6, we introduced three routing modes: With aggregated commodity sets and fractional commodity set capacity components, with fractional routing but integral capacity components, and fully integral.

The results of running our implementation in each of the three modes are listed in Table 6.11. Two of the entries need further explanation:

*) Solving problem `sdh4` in the fractional mode did not produce any solution within the time limit. We fail to deliver a reason. Analysis of this run does not give any hint for what caused this behaviour.

†) When solving `sdh5` in the fractional mode, our implementation did not finish the cutting plane phase of the branch-and-cut root node within an hour. It did in fact identify a huge amount of inequalities during that time. It did not identify a solution, because it never reached any situation where production of a solution was attempted.

The test results allow for the following conclusions:

		sdh1	sdh3	sdh4
Best solution value	With separator	100.0	100.0	100.0
	Without	99.9	98.6	∞^*
Final lower bound	With separator	90.3	89.0	60.9
	Without	90.1	89.0	60.7
Root node lower bound	With separator	79.3	88.5	60.7
	Without	79.3	88.5	60.5
Initial lower bound	With separator	77.0	83.1	35.2
	Without	77.0	83.1	35.2
Final gap	With separator	9.7	11.0	39.1
	Without	9.8	9.7	—
Branch-and-cut iterations	With separator	13491	501	173
	Without	13450	401	194
Iteration of best solution	With separator	4901	386	63
	Without	5455	323	—
Total runtime (min:sec)	With separator	60:00	60:00	60:00
	Without	60:00	60:00	60:00
Identified node cut inequalities	With separator	1	3	4
	Without	0	0	0
Inequalities by other separators	With separator	40	393	825
	Without	40	393	827

Table 6.9: Node cut separator success at default settings. *For the three problems where the node cut separator identified violated inequalities at default settings, competitive tests were run with the separator turned off.*
**) Without separator, no solution could be found for* `sdh4` *within the time limit.*

		sdh1	sdh2	sdh3	sdh4	sdh5
Best solution value	With separator	100.6	100.0	100.9	98.9	101.0
	Without	100.0	100.0	98.3	∞^*	100.6
Final lower bound	With separator	89.5	98.8	88.9	60.8	71.5
	Without	89.7	98.8	88.9	60.7	71.4
Root node lower bound	With separator	78.9	95.4	88.4	60.6	71.5
	Without	78.8	95.4	88.4	60.5	71.4
Initial lower bound	With separator	76.8	78.2	83.0	34.6	69.8
	Without	76.8	78.2	83.0	34.6	69.8
Final gap	With separator	11.0	1.2	11.9	38.5	29.2
	Without	10.3	1.2	9.6	—	29.0
Branch-and-cut iterations	With separator	10379	2256	517	226	2
	Without	11004	2242	436	275	2
Iteration of best solution	With separator	6257	2	391	63	2
	Without	5852	2	209	—	1
Total runtime (min:sec)	With separator	60:00	60:00	60:00	60:00	60:00
	Without	60:00	60:00	60:00	60:00	60:00
Identified node cut inequalities	With separator	10	2	3	4	5
	Without	0	0	0	0	0
Inequalities by other separators	With separator	40	8	393	820	106
	Without	40	8	389	819	111

Table 6.10: Node cut separator success without bound strengthening. *Because the separator identified some violated inequalities for all SDH problems, it is compared to runs with the same settings but disabled node cut separator.*
**) Without separator, no solution could be found for* `sdh4` *within the time limit.*

name		sdh1	sdh2	sdh3	sdh4	sdh5	opt1	opt2	opt3
Best solution	Frac.	99.9	100.0	97.5	∞^*	$\infty^\dagger$	100.0	100.0	100.0
value	Plain	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Int.	99.9	100.0	97.8	100.0	92.4	100.1	100.0	100.1
Final lower	Frac.	92.6	99.9	89.9	60.8	—	63.4	70.2	87.4
bound	Plain	90.3	99.9	89.0	60.9	72.7	63.4	70.2	87.4
	Int.	89.2	99.9	89.1	60.9	72.7	63.2	70.2	87.4
Root node	Frac.	81.3	99.9	88.5	60.7	—	61.8	69.4	85.8
lower bound	Plain	79.3	96.5	88.5	60.7	72.7	61.8	69.4	85.8
	Int.	79.3	96.5	88.5	60.7	72.7	61.8	69.4	85.8
Initial lower	Frac.	77.0	79.4	83.1	35.2	71.1	61.8	68.1	64.0
bound	Plain	77.0	79.4	83.1	35.2	71.1	61.8	68.1	64.0
	Int.	77.0	79.4	83.1	35.2	71.1	61.8	68.1	64.0
Final gap	Frac.	7.3	0.1	7.8	—	—	36.6	29.8	12.6
	Plain	9.7	0.1	11.0	39.1	27.3	36.6	29.8	12.6
	Int.	10.8	0.1	9.0	39.1	21.4	36.9	29.8	12.7
Branch-and-cut	Frac.	15381	594	941	134	0	12673	474	354
iterations	Plain	13491	590	501	173	2	12668	474	373
	Int.	6471	590	530	163	2	8083	472	388
Iteration of	Frac.	14473	1	463	—	—	2034	101	125
best solution	Plain	4901	2	386	63	1	2034	101	125
	Int.	5662	2	379	63	1	4012	95	40
Total runtime	Frac.	60:00	25:56	60:00	60:00	60:00	60:00	60:00	60:00
(min:sec)	Plain	60:00	14:24	60:00	60:00	60:00	60:00	60:00	60:00
	Int.	60:00	15:33	60:00	60:00	60:00	60:00	60:00	60:00

Table 6.11: Routing mode comparison. *For the three routing modes fractional, plain, and integral, the basic SDH and the optical problems are compared.*

*) *For* `sdh4` *in the fractional mode, no solution could be identified within the time limit.*

†) *For* `sdh5` *in the fractional mode, the root node could not be processed in time.*

- For the problems with only one commodity set of routing unit 1 (`sdh2`, `sdh4`, `opt1`, `opt2`, and `opt3`), we expected the fractional and the plain mode to be equivalent, because in this situation the capacity bound inequalities (5.33) ensure that the commodity set capacity components are installed up to the available capacity on each edge, which simulates the effect of the fractional mode. Except for `sdh4`, which is mentioned above, this assumption turns out to be true.

- It is however surprising that for `sdh2`, it takes in fact longer to solve the problem to optimality in the fractional mode than the plain mode. This is probably due to the fact that in the plain mode the variables for the commodity set capacity component are available for branching, but we cannot provide evidence for this assumption.

- For two of the problems with more than one commodity set, solutions in the fractional mode were found. These are `sdh1` and `sdh3`. The assumption that the aggregation happening in the fractional mode makes these problems easier to solve than in plain mode holds: Better solutions with a smaller gap are identified, and more branch-and-cut nodes can be processed within time.

- Using the integral mode turns the whole algorithm framework into a heuristic. Because of the potential gap between fractional solutions from the LPs to solutions for an integral routing obtained by rounding, we assumed that solutions would be of high cost and large gap in this mode. Because of the simplicity of the rerouting heuristic that is employed in this mode, we assumed that it cannot undo the damages of the rounding process, and hence believed that this mode performs so poor that it cannot be put into practice. These assumptions turn out to be wrong. In fact, this mode produced better solutions with a smaller gap, and sometimes it even increased the number of branch-and-cut node that could be processed within the time limit. Hence, it is certainly advisable to explore further improvements to this mode.

6.4 OSPF Scenario

In Scenario 1.3, expansion steps of an IP-over-SDH network over some years are discussed. On top of a SDH network similar to Scenario 1.1, which is rented from a capacity provider, an IP network running the OSPF protocol is built.

6.4.1 Application

This scenario describes one aspect of a long-term project dealing with a single backbone network, which is described in [BK00]. A bunch of applications were built for different parts of the project. One of these applications uses our models and the data structures of our implementation, but different algorithms. Here, we focus on this application.

The construction of components and parameters for the HC model follows the same scheme as the SDH scenario: Fiber setups with SDH capacities are modelled by link designs, and router cards become modules. There is a second set of modules, which consists of IP router cards. Each link design needs both SDH and IP interfaces, which are provided by the two module types. There is just one node design, which is an IP router. SDH multiplexers are not modelled, because they are owned by the capacity provider and responsibility to acquire and maintain these multiplexers is up to the provider.

The OSPF protocol requires provision of routing weights. Demands are then sent along a weighted shortest path to their destination. The implementation uses an LP-based algorithm to compute routing weights such that the resulting routing equals a given set of flow paths. An additional requirement states that routing weights are to be chosen in such a manner that the resulting shortest paths are unique, to ensure that the switching hardware can not choose different

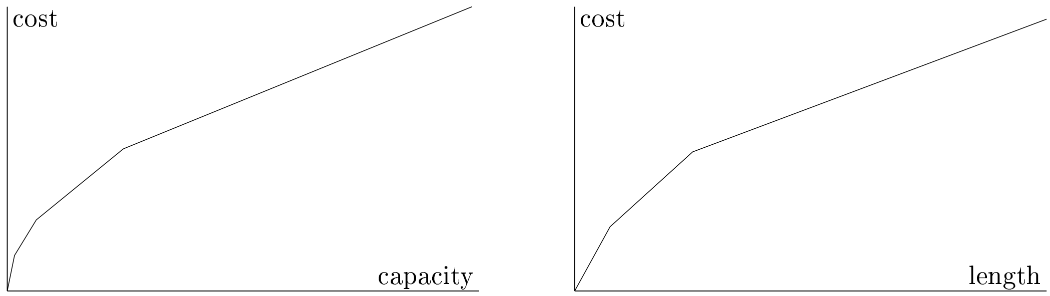


Figure 6.3: OSPF scenario cost structure. *For the leased SDH lines, economies of scale apply for both capacity and fiber length.*

paths than planned. There are additional constraints to ensure that the capacity utilization of links and switching hardware stays below predefined limits.

There are cost values assigned to components, and the optimization's objective is minimization of overall cost. For all link designs, there are limits on the total amount that may be used in a solution. These limits are called *global pool*. Additionally, there are limits on the amount of link design changes w.r.t. the previous network state. The "expansion tracking" HC model extension, as introduced in Section 3.2.3, is used to model these constraints.

The implementation that was used employs a branch-and-cut algorithm with a different decomposition than the scheme introduced in Chapter 5. The algorithm solves a relaxation consisting of the HC model IP and a path-based IP. The LP to compute routing weights is used as an external oracle. The algorithm branches on all variables until it finds an integral solution. More details about the techniques used can be found in [BK02].

There exists another implementation that uses a more advanced approach. It relaxes the constraints linking capacities provided by the HC model and edge flows by the routing IP using a Lagrangian relaxation, thereby decomposing the problem into a HC model part and a routing part. Although this approach is heuristical, it is able to produce good solutions in short time. An introduction to this implementation is given in [Ble03].

6.4.2 Results

Data Sets

We use four data sets, all of which deal with the same network. They belong to expansion problems of that network in chronological order. Each data set has three variants: First, "Free", with global pools for link designs but no change restriction. The second is "restricted", where pools and change limitations apply. The third variant, where link designs are fully fixed, is called "fixed". An overview over the resulting twelve instances is given in Table 6.12. Notice that the "fixed" variant is not necessarily within the bounds of the "restricted" one. The global pools of `ospf3` in the "free" variant are different to those of the other variants. It is also noteworthy that capacities are evolving over the four sets: The three available capacities in `ospf1` are STM–1, STM–4, and STM–16. In `ospf4`, only STM–16 and STM–64 are available.

Links are rented from a capacity provider. The cost structure for capacity is typical for leased lines, and exhibits economies of scale. Influence of capacity and link length on fees is shown in Figure 6.3.

Results

We attempted to solve all twelve instances using the implementation introduced above. The "fixed" variant of `ospf2` turned out to be infeasible. An overview over the remaining runs is given in Table 6.13.

		ospf1	ospf2	ospf3	ospf4
Nodes	Free	11	11	11	11
	Restricted	11	11	11	11
	Fixed	11	11	11	11
Edges	Free	47	47	47	47
	Restricted	47	47	47	47
	Fixed	47	47	47	47
Preinstalled capacities	Free	0	0	0	0
	Restricted	0	0	0	0
	Fixed	0	0	0	0
Zero-cost capacity edges	Free	0	0	0	0
	Restricted	0	0	0	0
	Fixed	0	0	0	0
Avg. link designs / edge	Free	2.9	2.0	2.0	2.0
	Restricted	2.9	2.0	2.0	2.0
	Fixed	1.0	1.0	1.0	1.0
Avg. node designs / node	Free	1.0	1.0	1.0	1.0
	Restricted	1.0	1.0	1.0	1.0
	Fixed	1.0	1.0	1.0	1.0
Modules	Free	9	9	9	9
	Restricted	9	9	9	9
	Fixed	9	9	9	9
Commodity sets	Free	1	1	1	1
	Restricted	1	1	1	1
	Fixed	1	1	1	1
Demands	Free	110	110	110	109
	Restricted	110	110	110	109
	Fixed	110	110	110	109
Survivability conditions	Free	No	No	No	No
	Restricted	No	No	No	No
	Fixed	No	No	No	No
Components in global pool	Free	3	2	2	2
	Restricted	3	2	2	2
	Fixed	0	0	0	0
Components under expansion tracking	Free	0	0	0	0
	Restricted	3	2	0	2
	Fixed	0	0	0	0

Table 6.12: OSPF problem instances. *There are four data sets, one for each expansion phase. Each set exists in three variants: Free link designs within global limits, link designs restricted to a certain amount of changes, and all link designs fully fixed.*

		`ospf1`	`ospf2`	`ospf3`	`ospf4`
Best solution value	Free	50.2	96.5	123.1	77.9
	Restricted	100.0	100.0	100.0	100.0
	Fixed	115.0	—*	252.0	73.7
Final lower bound	Free	50.2	68.6	123.1	66.1
	Restricted	96.4	88.7	74.7	100.0
	Fixed	115.0	—	252.0	65.6
Final gap	Free	0.0	28.8	0.0	15.1
	Restricted	3.6	11.6	25.3	0.0
	Fixed	0.0	—	0.0	11.0
Relative node hardware cost	Free	27.1	20.4	19.6	20.1
	Restricted	23.7	18.6	15.6	20.9
	Fixed	23.7	—	21.0	19.5
Branch-and-cut iterations	Free	384	1506	1152	1014
	Restricted	4095	1964	1426	810
	Fixed	34	—	2	1177
Total runtime (min:sec)	Free	11:47	60:00	60:00	60:00
	Restricted	60:00	60:00	60:00	14:10
	Fixed	0:03	—	0:01	60:00

Table 6.13: Results for OSPF problems. *For the four OSPF problems in each of the three variants, results of the test runs are listed.*
*) *The fixed variant of* `ospf2` *is infeasible.*

The following conclusions seem valid:

- Adding the constraints for the expansion tracking extension can make a problem more difficult to solve. For two of the data sets, the “restricted” variant is the only one that could not be solved to optimality. On the other hand, two instances finish this variant with the best gap. Because some of the additional constraints have a very large support, chances are that solutions have more fractional coefficients. This could explain why sometimes no solution of small gap is found.
- Although only modules carry cost, node hardware makes up about 20% of solution’s cost values. Therefore, the hardware configuration part not only adds complex constraints to the problem, but also plays an important role for the objective.
- Considering the complex problem structure, the sharp time limit, and the lack of advanced decomposition schemes in the branch-and-cut algorithm, results are of surprisingly high quality.
- Solving problems of Scenario 1.3 is possible. Solutions can be produced within reasonable time.

Chapter 7

Conclusion

In this thesis, we proposed means to integrate decisions about hardware configuration with capacity dimensioning and routing models. We introduced the component-resource model, which is a novel approach to discuss a broad variety of network optimization problems. We developed a hardware abstraction consisting of node designs, link designs, modules, and commodity set capacity components. This abstraction can be formulated using the previous model, and is hence an application of the component-resource model. It is the second model proposed in this work, the hardware-configuration model. We introduced three optional model extension that allow to include additional conditions, and showed how incorporation of these extension reduces to simple changes in the underlying component-resource model. Additionally, the model supports multiple parallel demand sets that can be separately dealt with, and might use any routing model, including different ones for the demand sets.

We examined the polyhedra associated with the hardware-configuration model. We introduced a class of inequalities that are facet-defining under reasonable conditions, and introduced facet-defining inequalities for the polyhedron associated with a single edge of the network. We then developed a separation heuristic for the former inequalities. We then showed how to solve instances of the model using a branch-and-cut algorithm on a problem decomposition. We introduced a bound strengthening heuristic that runs as a preprocessor. This heuristic uses IP relaxations that are small compared to the full problem to deduce better variable bounds and problem parameters.

We presented our implementation of the models and algorithms. A brief introduction to preprocessing steps, model transformations, branching rules, separators and different routing modes was given. Using three planning scenarios that act as representatives for the large amount of application possibilities, we showed how to put our models and algorithms into practice. We used data sets originating in real-world application of our implementation to test how well our algorithms perform and what influence certain settings have.

Considering hardware decisions in network dimensioning problems is not a new issue. However, employing linear and integer programming and the ability to solve such problems to optimality is in fact new, as nearly all previous results are heuristics. To our knowledge, there exists only one work where hardware decisions are employed in exact algorithms, and these decisions cover just one aspect of hardware configuration, and are equivalent to the "switching capacity" property of our node designs. Because we can successfully produce solutions for networks of reasonable size, we consider our approach a success for all three planning scenarios. Hence, we are confident that our models can also be applied to many more similar problems.

Unfortunately, we were not able to solve all of our test instances to optimality within the time limit we imposed on the algorithms. Although we are confident that by enlarging this limit, optimal solutions for all instances could be identified, the failure to reach optimality limits the conclusions that can be drawn. It seems valid to assume that the bound strengthening heuristic should always be used where practical problems are to be solved, as it does not consume too much time while sometimes successfully eliminating variables and branch-and-cut nodes. Our separation heuristic consumes a lot of time without identifying many violated inequalities, although it does work. Comparison of the different routing modes revealed a surprising advantage of employing our heuristic to obtain integral routings. It is obvious that this heuristic should be further investigated and improved, for it is so successful in improving solution quality.

Altogether, we proposed a framework that successfully extends the possibilities of telecommunication network optimization with hardware configuration features, and proved that this framework can be put easily into practice. It does produce good solutions in reasonable time. There is room for improvement in finding additional cut inequalities and branch-and-cut enhancements to faster increase the lower bound, so that proven optimality can be reached in less time.

List of Figures

List of Tables

List of Algorithms

Index

Bibliography

[AGW96] D. Alevras, M. Grötschel, and R. Wessäly. A network dimensioning tool. Technical Report SC 96-49, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 1996.

[AGW97] D. Alevras, M. Grötschel, and R. Wessäly. Capacity and survivability models for telecommunication networks. Technical Report SC 97-24, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 1997.

[BBC00] A. Berry, J.-P. Bordat, and O. Cogis. Generating all the minimal separators of a graph. *International Journal of Foundations of Computer Science*, 11(3):397–403, 2000.

[BG96] D. Bienstock and O. Günlük. Capacitated network design – Polyhedral structure and computation. *INFORMS Journal on Computing*, 8:243–259, 1996.

[BK00] A. Bley and T. Koch. Optimierung in der planung und beim aufbau des g-win. Technical Report ZR 00-48, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 2000.

[BK02] A. Bley and T. Koch. Integer programming approaches to access and backbone IP-network planning. Technical Report ZR 02-41, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 2002.

[Ble03] A. Bley. A lagrangian approach for integrated network configuration and routing planning in ip networks. In *Proceedings of the International Network Optimization Conference, Evry/Paris, October 2003*, volume (to appear), 2003.

[BS01] D. Bienstock and I. Saniee. ATM network design: Traffic models and optimization-based heuristics. *Telecommunication Systems*, 16:399–421, 2001.

[Gün99] O. Günlük. A branch-and-cut algorithm for capacitated network design problems, 1999.

[Iri71] M. Iri. On an extension of the maximum-flow minimum-cut theorem to multicommodity flows. *Journal of the Operations Research Society of Japan*, 13:129–135, 1971.

[ITU95] ITU-T Recommendation G.841: Types and characteristics of sdh network protection architectures. 1995.

[ITU97] ITU-T Recommendation G.803: Architecture of transport networks based on the synchronous digital hierarchy. 1997.

[Mar85] O. Marcotte. The cutting stock problem and integer rounding. *Mathematical Programming*, 33:82–92, 1985.

[OK71] K. Onaga and O. Kakusho. On feasibility conditions of multicommodity flows in networks. *IEEE Transactions on Circuit Theory*, 18:425–429, 1971.

[PR91] M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1):60–100, 1991.

[PW92] Y. Pochet and L.A. Wolsey. Network design with divisible capacities: Aggregated flow and knapsack formulation. In *Proceedings IPCO2*, pages 150–164, Pittsburgh, 1992. Carnegie Mellon University.

[SD94] M. Stoer and G. Dahl. A polyhedral approach to multicommodity survivable network design. *Numerische Mathematik*, 68(1):149–167, 1994.

[Wes00] R. Wessäly. *DImensioning Survivable Capacitated NETworks*. PhD thesis, Technische Universität Berlin, 2000.

[ZKW02] A. Zymolka, A. Koster, and R. Wessäly. Transparent optical network design with sparse wavelength conversion. Technical Report ZR 02-34, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 2002.

d4 c∅∅13z7 6ir1 ∅f 411 |-|31p3|> m3 41∅7. 1∅∅∅ thx |<47! j∅∅ r∅x∅r!