DIMENSIONING SURVIVABLE CAPACITATED NETWORKS

vorgelegt von
Diplom-Informatiker

ROLAND WESSÄLY

Vom Fachbereich 3 Mathematik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades eines

DOKTOR DER NATURWISSENSCHAFTEN

genehmigte Dissertation

Promotionsausschuß:

Vorsitzender:	Prof. Dr. Dirk Ferus
Berichter:	Prof. Dr. Martin Grötschel
Berichter:	Prof. Dr. Rolf Möhring

Tag der wissenschaftlichen Aussprache: 25. April 2000

Berlin 2000
D 83

Deutsche Zusammenfassung

In der vorliegenden Dissertation untersuchen wir die Optimierung von ausfallsicheren Telekommunikationsnetzwerken. Wir präsentieren unterschiedliche gemischt-ganzzahlige Modelle für die diskrete Kapazitätsstruktur, sowie für die Sicherung des Netzes gegen den Ausfall einzelner Komponenten. Die Modelle wurden in einer Kooperation mit der `E-Plus` Mobilfunk GmbH verwendet. Die theoretischen Resultate wurden in Algorithmen umgesetzt und in das von uns entwickelte Netzwerksoptimierungswerkzeug DISCNET (DImensioning Survivable Capacitated NETworks) integriert, welches seit mehreren Jahren in der Planung bei `E-Plus` eingesetzt wird.

Wir betrachten das Transportnetzplanungsproblem eines Telekommunikationsanbieters. Dieses Problem setzt auf logischen Kommunikationsanforderungen zwischen den Standorten (Knoten) des zu planenden Netzes und potentiell installierbaren Verbindungen (Kanten) zwischen derselben Knotenmenge auf. Ein Kapazitätsmodell stellt die Information bereit, welche Kapazitäten auf den potentiellen Kanten verfügbar sind. Wir betrachten zwei Modelle. Entweder ist eine explizite Liste der verfügbaren Kapazitäten gegeben oder eine Menge von sogenannten Basiskapazitäten, die auf jeder Kante individuell kombiniert werden können. Die Basiskapazitäten müßen paarweise ganzzahlige Vielfache voneinander sein. Man beachte, daß diese Eigenschaft von den internationalen Standards PDH und SDH erfüllt wird. Ein Ausfallsicherheitsmodell stellt die Information bereit, wie das zu planende Netz gegen den Ausfall einzelner Netzkomponenten geschützt werden soll. Wir betrachten sinnvolle Kombinationen der Modelle DIVERSIFICATION, RESERVATION und PATH RESTORATION. Das erste Modell garantiert Ausfallsicherheit durch kommunikationsbedarfsabhängige Beschränkung des Prozentsatzes, der durch einzelne Netzkomponenten geroutet werden darf. Bei den beiden anderen Modelle können Kommunikationsbedarfe bei Ausfall einer Netzkomponente auf unterschiedliche Weise neu geroutet werden. Ziel der Planung ist eine kostenminimale Kapazitätsentscheidung, die eine Routenplanung aller Kommunikationsbedarfe gemäß den Ausfallsicherheitsanforderungen ermöglicht.

Wir entwickeln ein Schnittebenenverfahren zur Lösung der betrachteten Optimierungsprobleme. Zu diesem Zweck untersuchen wir Polyeder, die mit den verschiedenen Problemen assoziiert sind. Wir präsentieren neue Klassen von Ungleichungen, entwickeln Separationsalgorithmen und Heuristiken. Mit dem Schnittebenenverfahren werden untere und obere Schranken für den Wert von Optimallösungen berechnet, und daher ist es möglich, Qualitätsgarantien für die berechneten Löungen anzugeben. Parallel zur Beschreibung der implementierten Algorithmen präsentieren wir umfangreiche Tests mit praktisch relevanten Daten, die zu Problemen mit mehr als 2 Billionen Variablen führen.

Schlüsselworte: Ausfallsichere Telekommunikationsnetzwerke, Schnittebenenverfahren

Mathematics Subject Classification (1991): 90C11, 90C90, 90B12

Abstract

In this thesis, we develop a framework for cost-minimal survivable capacitated network design problems. We present different models both for the discrete capacity structure and the protection of the network against the failure of single network components. The mathematical models are used within a cooperation with the German mobile-phone provider `E-Plus` Mobilfunk GmbH. The theoretical and practical results have been integrated into our network dimensioning tool DISCNET (DImensioning Survivable Capacitated NETworks), which is in use at `E-Plus`.

We consider the transport network design problem of a telecommunication provider. As input is given a so-called demand graph which contains as edges the logical communication requirements in terms of channels, and a supply graph which contains as edges the potential physical transmission links. A capacity model provides information about capacities which can potentially be installed on the edges of the supply graph together with the respective costs. We consider two models. Either it is possible to choose the capacity of each edge from a finite set of capacities, or, in the other case, as a non-negative integer combination of a finite set of basic capacities. These basic capacities satisfy, as in the plesiochronous digital hierarchy (PDH) and the synchronous digital hierarchy (SDH), that each one is an integer multiple of all smaller basic capacities. The survivability model provides information on how to cope with single node or single edge failures. We consider reasonable combinations of the following models: DIVERSIFICATION, RESERVATION, and PATH RESTORATION. The first model ensures survivability through a "node-disjoint" routing in the non-failure case, while the other models ensure survivability through rerouting in case of a single network component failure. These latter models differ in the way the failure routing is performed and to which extent the non-failure routing must be respected. The objective is to choose capacities at minimal cost such that it is possible to route the communication demands simultaneously under the given survivability requirements.

We employ a cutting plane approach to solve the different network design problems. Thus, we focus on computing a lower bound for the optimal solution value and feasible solutions through heuristic algorithms. Together, the values of the lower bound and the best solution provide a guarantee for the quality of the solutions computed. We investigate the structure of associated polyhedra, develop separation algorithms for various classes of inequalities, and integrate this together with different types of heuristics. Throughout the thesis, we present computational studies of the different algorithmic parts with real-world problem instances that lead to mixed-integer programs with more than 2 trillion variables.

Keywords: Survivable Network Design, Cutting Plane Algorithm

Mathematics Subject Classification (1991): 90C11, 90C90, 90B12

Danksagung

Es ist mir ein Bedürfnis, den zahlreichen Freunden, sowie den Kollegen am ZIB und bei `E-Plus` zu danken, deren Mithilfe das Niveau der Arbeit deutlich gehoben haben. Ich bin mir dieser Hilfe aus meinem familiären und kollegialen Umfeld sehr bewusst und danke allen sehr herzlich. Drei Personen möchte ich dabei besonders hervorheben, ohne die diese Arbeit nicht zu Stande gekommen wäre.

Herrn Grötschel gilt besonderer Dank für ein außerordentlich fruchtbares Arbeitsumfeld, für dieses überaus interessante Dissertationsthema, für die vielen Ratschläge und die notwendige Freiheit zur selbstständigen Arbeit.

Meiner Mutter gebührt größter Dank. Sie hatte den Weitblick, selbst in schwierigsten Zeiten nicht kurzfristig attraktiven Zielen hinterherzulaufen, sondern mir eine gute Ausbildung zu gestatten. Vielen Dank Mama, wer weiss wie mein Weg ohne Dich verlaufen wäre?!

Tanja kann ich einfach nur für alles danken. Sie verkörpert das größte Glück, das mir neben der Geburt unserer Tochter Hanna widerfahren ist.

Berlin, den 7. Februar 2000 Roland Wessäly

Contents

Introduction

In this thesis, we present part of the research that has been motivated by an ongoing project of the Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB) with the `E-Plus` Mobilfunk GmbH. The project was initiated in July 1994 and our team at ZIB, guided by Martin Grötschel, started with Dimitris Alevras, Mechthild Stoer and myself. A few months later Mechthild was hired by the Norwegian Telecom and after two years, Dimitris left to IBM. Besides the support of some students, I have been working alone in this project since then[1].

The structure of this thesis reveals our view of solving problems arising in practice. We start with a description of the practical background, then set up mathematical models which focus on important parts of the practical problems, continue with a mathematical investigation of the structural properties and eventually, we convey the theory into an algorithmic environment which can then be used to solve the practical problems.

Along this line of thought we attack the transport network design problem of a telecommunication provider. This is a very general problem and many telecommunication providers (including mobile-communication providers) need to solve some variation of the basic version, which can be defined as follows:

> Given are the locations (nodes) of the network, communication requirements between pairs of nodes, and a set of available capacities for each potential line (link) between two nodes. A feasible solution comprises a topology consisting of a subset of the links, an individual capacity for each chosen link, and routings for all communication requirements such that the chosen link capacities suffice to accommodate the routings simultaneously. Among all feasible solutions, the target is to find an optimal solution with respect to the cost incurred by the selected capacities.

The combinatorics of this integrated topology, capacity and routing problem is enormous, since the number of potential topologies and transmission paths is typically exponential in the number of available transmission links. In fact, many special cases of this basic version of network design problem are hard optimization problems in the sense of complexity theory. Another reason for this complexity is the limited number of available capacities. In practical applications, the set of capacities has almost always a discrete structure since the standards

[1] I will use "we", whenever I make reference to this project team or to the reader and myself.

defined by the international telecommunication union (ITU) provide a well-defined set of capacities which are supported by the available network equipment. It is worth mentioning that many of the survivable capacitated network design problems (including those considered in this thesis) would be polynomially solvable if an arbitrary capacity could be installed on links between network nodes.

As we mentioned before, routings satisfying the communication requirements are one target within the transport network design. However, we did not mention yet, that the routings should be chosen such that the network is considered *survivable*. According to the standards defined in (ITUT-G.841, 1995), a network is survivable, if it "is capable of restoring traffic in the event of a failure. The degree of survivability is determined by the network's ability to survive single line system failures, multiple line system failures, and equipment failures." The equipment in modern telecommunication networks is highly reliable, but survivability is an issue since there might be a huge impact of network failures on our society. Just recall the predicted horror scenarios related to the so-called year 2000 problem, and consider the following newspaper references as a few examples of network failures:

- In May 1988 (The Wall Street Journal, 1988), an electrical fire in a switching center in Chicago was responsible for 20 percent flight reduction at O'Hare, which has been the most busy airport of the US at that time. It turned out, see (IEEE Spectrum, 1989), that "some areas had no service for a month, and dollar estimates of lost business ranged from hundreds of millions to tens of billions".
- Christmas 1994 (Der Tagesspiegel, 1994), a digger cut the main cable connecting several districts of Berlin to the rest of the city. In consequence, approximately 160 000 households were without cable TV and 3000 households without telephone. Recall, it was Christmas.
- In June 1995 (Der Spiegel, 1995), terrorists cut fiber-optic cables at strategic points close to the international airport in Frankfurt. They were successful. The complete southern part of Frankfurt, including the airport as well as the university hospital, were without data connection and telephone.
- In March 1999 (Die Presse, 1999), a digger cut an important cable in Austria. As a consequence, Vienna and the western parts of Austria were disconnected for almost a complete day.

Modern telecommunication networks are based on fiber-optic cables, which have the potential to transport huge amounts of data using the necessary equipment at the end-nodes. Hence, networks with such high capacity links tend to be very sparse and each link carries a substantial part of the overall traffic. The bad news within this context are that a substantial amount of traffic is lost in case of such a link failure, if the network has not been prepared for the particular failure situation. Since networks which are designed to be survivable tend to be considerably more expensive, the network planner has to find the right balance between cost and quality of the network. This is exactly the topic of this thesis: How to design a *cheapest survivable* network?

A few years ago, it has been common practice to solve the transport network design problem through iteration over the following subproblems.

- Select the topology of the network.
- Choose a routing for the communication requirements using the selected topology.
- Choose capacities able to accommodate the chosen routings.

It has been proven by experience that this sequence of decisions leads to feasible but very expensive solutions. The reason is the following. If the network planner chooses a cheapest among all connected topologies, the result will be a minimal spanning tree. Obviously, such a decision completely determines the other two steps. The routing is fixed since there is exactly one path between every pair of nodes and the capacities are determined by the routing decision anyway. Even if survivability issues have been included in the topology planning, the cheapest sufficiently connected topology will be very sparse, and therefore the routings as well as the capacities are almost predetermined.

During the last years, computing power has been rapidly increasing and solution methodologies have been considerably improved by several researchers. Hence, it appears natural that modern telecommunication providers are now demanding for a more sophisticated integration of planning problems as well as solution methods.

We examined many commercial network planning tools, which usually provide an elaborate graphical user interface and whose solution approaches are also becoming more sophisticated. Typically, some sort of a randomized heuristic such as simulated annealing, genetic algorithms or tabu search has been implemented as solution approach. In all tools we got aware of, however, at least two features are missing which are important for *optimization* network design tools in a mathematical sense:

- First, an algorithm to compute a lower bound for the cost of an optimal solution. Without this feature, it is impossible to prove that a solution is optimal or to provide a quality guarantee for the solutions.
- Second, an exact algorithm to verify whether a given set of capacities can accommodate routings of the communications demands which satisfy all imposed requirements. Without this feature, even the enumeration of all feasible capacity selections becomes a non-trivial task.

Unless $\mathcal{P} = \mathcal{NP}$ (which is not believed by the majority of researchers in this field), we cannot expect to find an efficient algorithm that computes provably optimal solutions. Therefore, we decided to develop problem dependent theory for a *cutting plane approach*, a solution methodology which has proven to be successful for many $\mathcal{NP}$-hard combinatorial optimization problems. This approach potentially yields *provably* optimal solutions, since lower bounds (we assume a minimization problem) are calculated for the value of an optimal solution throughout the optimization process. This is done in addition to the computation of feasible solutions. With the knowledge of such a lower bound, a *quality guarantee* for the value of the best solution found can be provided. In more detail, if *up* denotes the value of the best solution found and *low* the computed lower bound, the quality guarantee (gap) is defined as $(up - low)\ /\ low$. For example, if some heuristic yields a solution of value 11 million and the lower bound calculation yields 10 million, then it is proven that an optimal solution is at most 10 percent cheaper. Notice that the best solution found might be optimal but this just cannot be proven. If the lower bound, however, is equal to the best solution value, then this establishes a *proof of optimality* for the best solution found.

In this thesis, we integrate different models for the discrete capacity structure and different models to deal with single network component failures. Our mathematical models integrate topology decisions, capacity decisions, routing decisions and survivability issues. This integration leads to huge mixed-integer programming models which are extremely difficult to solve. Just as an example, several of the practical problem instances used within this thesis have more than a trillion (1 000 000 000 000(!)) variables. This indicates the necessity of sophisticated algorithms. We present problem dependent theory including the investigation and classification of polyhedra related to different network design problems together with separation algorithms for all classes of inequalities employed in the cutting plane algorithm. We develop several starting and improvement heuristics, and extend and adapt column generation based algorithms to solve the decision problem whether given capacities suffice to accommodate a routing that satisfies all capacity and survivability requirements. We developed and implemented an optimization tool, called DISCNET (DImensioning Survivable Capacitated NETworks), that has been in use within the network planning process at `E-Plus` for more than three years now. It serves as a tool for short-term network expansion as well as for long-term network and budget planning. Beside the potential cost reduction of $10-20\%$, in other words millions of Euro, DISCNET puts the network designer into position to analyze different network scenarios. It becomes easier to decide how much capital one is willing to invest in order to achieve a *well-defined* quality in the network.

Outline of the thesis

The preliminaries, which follow this introduction, serve as a short reference to the notation and concepts used. We cover parts from linear algebra, polyhedral theory, linear programming and graph theory. The subsequent chapters reveal, as we already mentioned, our view of solving practical problems.

In Chapter 1, we describe the practical background of the overall planning process for a mobile-communication network. We give an overview of the structure of such a network and outline different parts of the planning process. Beside the transport network design, these are node location problems, traffic forecast, and switching network planning.

In Chapter 2, we focus on a mathematical view of the considered problem. We formulate linear mixed-integer programming models for the transport network design problem, covering two models for the discrete capacity structure, and three models to deal with single network component failures. At the end of the second chapter, we give a survey on capacitated network design problems in which the problem of routing the demands is defined as a multicommodity-flow problem.

In Chapter 3, we present our theoretical knowledge about the polyhedra defined as the convex hull of feasible solutions. For each combination of a model for the discrete capacity structure and a model for survivability, we investigate the facial structure of the respective polyhedra. We provide a coherent presentation of the knowledge about these polyhedra including several *new* classes of valid and facet-defining inequalities.

In Chapter 4, we describe how to practically solve transport network design problems on the basis of the research presented in the preceding chapter. Starting from an overview of the particular cutting plane algorithm, we discuss in the subsequent sections details concerning preprocessing, linear programming relaxations, feasibility problems of capacity vectors, separation algorithms, and heuristic algorithms. Eventually, we present the results of thorough computational experiments with real-world data.

These four chapters build the core of the thesis. Afterwards, we conclude with some remarks on possible future research directions and we briefly describe the graphical user interface, which has been implemented (in JAVA) to support the basic work of the network planner.

Preliminaries

In the following, we give an overview of well-known notions and concepts from linear algebra, polyhedral theory, linear programming and graph theory. Our description is rather compact and does not serve as an introduction to the respective areas. Hence, we refer the reader not familiar with the particular theory to one of the following excellent books and overviews. An introduction to linear algebra can be found in any basic textbook on this topic. A good overview of polyhedral theory can be found in (Pulleyblank, 1983), (Padberg and Grötschel, 1985), (Schrijver, 1986) and (Nemhauser and Wolsey, 1988). The books (Schrijver, 1986) and (Padberg, 1995) give excellent introductions to the theory of linear programming. A more elementary introduction to this topic can be found in (Chvátal, 1983). There exists a huge number of introductory books for graph theory. (Bondy and Murty, 1976) might serve as a first reference. Finally, we refer to the excellent book (Ahuja et al., 1993) which provides a coherent introduction to the theory of network flows.

Linear Algebra

We denote by $\mathbb{R}$, $\mathbb{Q}$, and $\mathbb{Z}$ the sets of real, rational and integer numbers, respectively. For the positive part of these sets (including zero) we use the symbols $\mathbb{R}_+$, $\mathbb{Q}_+$, and $\mathbb{Z}_+$. To make definitions short we use $\mathbb{K}$ (or $\mathbb{K}_+$) if any of these three sets can be applied. We distinguish between the positive integer numbers without zero by $\mathbb{N}$. That is, $\mathbb{N} = \mathbb{Z}_+ \backslash \{0\}$. For $n \in \mathbb{N}$ the set of n-dimensional vectors over $\mathbb{K}$ is $\mathbb{K}^n$. Special vectors are $\mathbf{0}$ and $\mathbf{1}$, the vectors of all 0's and all 1's, respectively. The *transposition* of a vector x is x^T. The n-dimensional vectors $\{x_1, ..., x_k\} \subseteq \mathbb{K}^n$ are *linear independent*, if $\lambda_i = 0, i = 1, \ldots, k$, is the unique solution of the equation $\sum_{i=1}^k \lambda_i x_i = 0$ with $\lambda_i \in \mathbb{K}, i = 1, \ldots, n$.

For $x \in \mathbb{K}$, we denote by $\lceil x \rceil$ the smallest integer number larger than or equal to x and by $\lfloor x \rfloor$ the largest integer number smaller than or equal to x. We also say $\lceil x \rceil$ and $\lfloor x \rfloor$ are the *ceiling* and the *floor of* x, respectively. If $x \in \mathbb{Z}\backslash\{0\}$ divides $y \in \mathbb{Z}\backslash\{0\}$, that is, if $y/x \in \mathbb{Z}$, we write $x|y$. The *greatest common divisor* $gcd(Y)$ of a set $Y = \{y_1, \ldots, y_n\} \in (\mathbb{Z}\backslash\{0\})^n$ is $\max\{x : x|y_i \text{ for } i = 1, \ldots, n\}$.

For any finite set E, we identify a function $x\colon E \to \mathbb{K}$ with the corresponding $|E|$-dimensional vector $x = (x_e)_{e\in E} \in \mathbb{K}^E := \mathbb{K}^{|E|}$. The *incidence vector* $\chi^F \in \{0,1\}^E$ of $F \subseteq E$ is defined by $\chi^F(e) := 1$ if $e \in F$ and $\chi^F(e) := 0$ if $e \notin F$. Conversely, a vector $x \in \{0,1\}^E$ defines the *incidence set* $F_x := \{e \in E : x_e = 1\}$. More generally, the *support* of a vector $x \in \mathbb{K}^E$ is defined by $\operatorname{supp}(x) := \{e \in E : x_e \neq 0\}$. Thus, $F_x = \operatorname{supp}(x)$ for a $\{0,1\}$-vector x.

A set $X \subseteq \mathbb{K}^n$ is *bounded*, if $M \in \mathbb{K}_+$ exists with $\|x\| \leq M$ for all $x \in X$ and some norm $\|\cdot\| : \mathbb{K}^n \to \mathbb{K}_+$. (A norm satisfies $(i) \|x\| \geq 0, \forall x \in \mathbb{K}^n; (ii) \|\lambda x\| = |\lambda| \|x\|, \forall \lambda \in \mathbb{K}, x \in \mathbb{K}^n; (iii) \|x+y\| \leq \|x\| + \|y\|, \forall x, y \in \mathbb{K}^n$.) For $X \subseteq \mathbb{K}^n$, we define

$$\begin{array}{lll}
\operatorname{lin}(X) & := \{x \in \mathbb{K}^n : & \exists \lambda_1, \ldots, \lambda_t \in \mathbb{K} \text{ and } \exists x_1, \ldots, x_t \in X, t \in \mathbb{N} \text{ s.t.} \\
 & & x = \sum_{i=1}^t \lambda_i x_i\}, \\
\operatorname{aff}(X) & := \{x \in \mathbb{K}^n : & \exists \lambda_1, \ldots, \lambda_t \in \mathbb{K} \text{ and } \exists x_1, \ldots, x_t \in X, t \in \mathbb{N} \text{ s.t.} \\
 & & \sum_{i=1}^t \lambda_i = 1 \text{ and } x = \sum_{i=1}^t \lambda_i x_i\}, \\
\operatorname{conv}(X) & := \{x \in \mathbb{K}^n : & \exists \lambda_1, \ldots, \lambda_t \in \mathbb{K}_+ \text{ and } \exists x_1, \ldots, x_t \in X, t \in \mathbb{N} \text{ s.t.} \\
 & & \sum_{i=1}^t \lambda_i = 1 \text{ and } x = \sum_{i=1}^t \lambda_i x_i\}, \\
\operatorname{cone}(X) & := \{x \in \mathbb{K}^n : & \exists \lambda_1, \ldots, \lambda_t \in \mathbb{K}_+ \text{ and } \exists x_1, \ldots, x_t \in X, t \in \mathbb{N} \text{ s.t.} \\
 & & x = \sum_{i=1}^t \lambda_i x_i\},
\end{array}$$

to be the linear, affine, convex and cone hull of X, respectively. The *dimension* $\dim(X)$ of X is $\dim(\operatorname{aff}(X))$, that is, the maximum number of linearly independent vectors in $\operatorname{aff}(X)$.

Polyhedral Theory

Given $a \in \mathbb{K}^n \setminus \{\mathbf{0}\}$ and $\alpha \in \mathbb{K}$, the set $\{x \in \mathbb{K}^n : a^T x \leq \alpha\}$ is a *half-space* and $\{x \in \mathbb{K}^n : a^T x = \alpha\}$ is a *hyperplane*. The finite intersection of half-spaces given by $\{x \in \mathbb{K}^n : Ax \leq b\}$ with $A \in \mathbb{K}^{m \times n}$ and $b \in \mathbb{R}^m$ is a *polyhedron*, where $\mathbb{K}^{m \times n}$ is the space of matrices with m rows and n columns. A bounded polyhedron is a *polytope*.

The inequality $a^T x \leq \alpha$ for $a \in \mathbb{K}^n, \alpha \in \mathbb{K}$ is *valid* for a polyhedron P, if $P \subseteq \{x \in \mathbb{K}^n : a^T x \leq \alpha\}$, and it is *tight for* P, if it is valid and $\mathcal{F}_{a,\alpha} := P \cap \{x \in \mathbb{K}^n : a^T x = \alpha\} \neq \emptyset$, that is, if there exists $\overline{x} \in P$ with $a^T \overline{x} = \alpha$. We say, $\mathcal{F}_{a,\alpha}$ is the *face of* P *induced by* $a^T x \leq \alpha$. A zero-dimensional face is a *vertex*, a one-dimensional face is an *edge* and a face $F \neq P$ of a polyhedron P is a *facet of* P if it is maximal with respect to inclusion. If $a^T x \leq \alpha$ is valid for P and $F = \{x \in P : a^T x = \alpha\}$ is a facet of P, we say that $a^T x \leq \alpha$ is *facet-defining* or *facet-inducing*. An equivalent characterization of a facet is that $\dim(F) = \dim(P) - 1$. If a polyhedron P is full-dimensional, and $a^T x \leq \alpha$ and $b^T x \leq \beta$ are facet-defining with $\mathcal{F}_{a,\alpha} = \mathcal{F}_{b,\beta}$, then there exists $\lambda \in \mathbb{R}_+$ with $\lambda a = b$ and $\lambda \alpha = \beta$. Note, this property is the key to most proofs that some inequality is facet-defining for a polyhedron. It is not difficult to see that every bounded polyhedron has vertices. If a polyhedron has vertices then every face of the polyhedron has vertices.

Linear Programming

A polyhedron $P = \{x \in \mathbb{K}^n : Ax \leq b\} \subseteq \mathbb{K}^n$, for $\mathbb{K} \in \{\mathbb{R}, \mathbb{Q}\}$, and a linear function $c : \mathbb{R}^n \to \mathbb{R}$ define a *linear program*, for short LP. Minimization and maximization versions are

$$\max\{c^T x : x \in P\} \qquad \text{and} \qquad \min\{c^T x : x \in P\} . \tag{1}$$

A vector $x^* \in P$, which attains the maximum (minimum) in (1), is an *optimal solution*. The set of optimal solutions $\{c^Tx : x \in P, c^Tx = c^Tx^*\}$ of a linear program $\max\{c^Tx : x \in P\}$ is a face of the polyhedron P. Thus, if a linear program is bounded, the optimal solution value is attained at a vertex. For every linear program $\max\{c^Tx : Ax \leq b, x \geq 0\}$, the *dual linear program* is $\min\{b^Ty : A^Ty \geq c, y \geq 0\}$.

Theorem 0.1 (Duality of linear programming) *Let $A \in \mathbb{K}^{m\times n}$, $b \in \mathbb{K}^m$ and $c \in \mathbb{K}^n$. If*

$$\{x \in \mathbb{K}^n : Ax \leq b, x \geq 0\} \neq \emptyset \quad \text{and} \quad \{y \in \mathbb{K}^m : A^Ty \geq c, y \geq 0\} \neq \emptyset\ ,$$

the optimal solution values of

$$\max\left\{c^Tx : Ax \leq b, x \geq 0\right\} \quad \text{and} \quad \min\left\{b^Ty : A^Ty \geq c, y \geq 0\right\} \tag{2}$$

are finite, and

$$\overline{x} \in \{x \in \mathbb{K}^n : Ax \leq b, x \geq 0\} \quad \text{and} \quad \overline{y} \in \{y \in \mathbb{K}^m : A^Ty \geq c, y \geq 0\} \tag{3}$$

exist such that $c^T\overline{x} = b^T\overline{y}$.

Under the assumption of Theorem 0.1, the maximum and the minimum in (2) are attained.

Theorem 0.2 (Complementary slackness) *Let $A \in \mathbb{K}^{m\times n}$, $b \in \mathbb{K}^m$ and $c \in \mathbb{K}^n$. If there exist $\overline{x} \in \{x \in \mathbb{K}^n_+ : Ax \leq b\}$ and $\overline{y} \in \{y \in \mathbb{K}^m_+ : A^Ty \geq c\}$, then $\overline{x}$ and $\overline{y}$ are optimal solutions of $\max\left\{c^Tx : Ax \leq b, x \geq 0\right\}$ and $\min\left\{b^Ty : A^Ty \geq c, y \geq 0\right\}$, respectively, if and only if $\overline{y}(b - A\overline{x}) = 0$ and $\overline{x}(A^T\overline{y} - c) = 0$.*

Graph Theory

An (*undirected*) *graph* $G = (V, E, \phi)$ consists of two sets V and E and an incidence function $\phi : E \longrightarrow V^{(2)}$, where $V^{(2)}$ is the set of unordered pairs of V. The elements of V are the *nodes* of G, and the elements of E are the *edges* of G. For each $e \in E$ with $\phi(e) = \{u, v\}$, the nodes u and v are the *end-nodes* of e. We say, u is *adjacent* to v and vice versa v is *adjacent* to u, and e is *incident* to u and v, respectively. For each $v \in V$, the number $d(v)$ of incident edges is the *degree* of v in G. The node v is an *isolated* node if the degree of v in G is zero, that is, $d(v) = 0$. An edge $e \in E$ with $\phi(e) = \{u, u\}$ for some node $u \in V$ is a *loop*, and two edges with the same end-nodes, that is, two edges $e_1, e_2 \in E$ with $\phi(e_1) = \phi(e_2)$, are *parallel edges*. Notice that parallel edges are the reason for the formalism with the incidence function. If we do not have to distinguish between parallel edges, or the edges are unique within the context, we write $e = uv \in E$ or $e = \{u, v\} \in E$ for an edge $e \in E$ with $\phi(e) = \{u, v\}$, instead of using the incidence function. A graph without loops and parallel edges is a *simple* graph.

A graph $H = (W, F, \psi)$ is a *subgraph* of $G = (V, E, \phi)$, if $W \subseteq V, F \subseteq E$ and $\psi(f) = \phi(f)$ for all $f \in F$. Special subgraphs are induced by subsets of the nodes and edges, respectively, and as a special case, by removing a node or an edge.

For a subset $F \subseteq E$ of the edges the triple $G[F] := (V, F, \phi_F)$ is the *subgraph of G induced by F*, where $\phi_F : F \longrightarrow V^{(2)}$ is the incidence function ϕ restricted to F, that is, $\phi_F(f) = \phi(f)$ for all $f \in F$. For a subset $W \subseteq V$ of the nodes, $E(W) \subseteq E$ denotes the subset of edges with both end-nodes in W, that is, $e \in E(W)$ if and only if $\phi(e) \in W^{(2)}$. With this notation, the triple $G[W] := (W, E(W), \phi_{E(W)})$ is the *subgraph of G induced by W*, where $\phi_{E(W)}$ is, analogously, the incidence function ϕ restricted to the edge set $E(W)$. In the following two special cases of induced subgraphs we slightly change the common notation for reasons of brevity. If an edge $e \in E$ is removed from G, the subgraph $G_e = (V, E_e)$ of G is obtained, and if a node $v \in V$ and all edges incident to v are removed from G, the subgraph $G_v = (V_v, E_v)$ of G is obtained. A node-set $W \subseteq V$ is a *node-cover* of G, if either $u \in W$ or $v \in W$ for each $e \in E$ with $\phi(e) = \{u, v\}$.

In the following, we represent graphs by their nodes and edges whenever the incidence function is implicitly given. Therefore, we will denote a graph by $G = (V, E)$ instead of $G = (V, E, \phi)$.

Given two subsets $W_1, W_2 \subseteq V$ with $W_1 \cap W_2 = \emptyset$, the set $\delta(W_1, W_2) := \{e \in E : \exists\, w_1 \in W_1, w_2 \in W_2 \text{ with } \phi(e) = \{w_1, w_2\}\} \subseteq E$ contains all edges with exactly one end-node in W_1. A subset $F \subseteq E$ of the edges is called a *k-graph-partition of G*, $k \in \mathbb{N}$, if there exists a partition of the node set V into k subsets $V_1, \ldots, V_k$ with $V_1 \cup \cdots \cup V_k = V$ and $V_i \cap V_j = \emptyset$ for $1 \leq i < j \leq k$, such that $F = \delta_G(V_1, \ldots, V_k) := \bigcup_{1 \leq i < j \leq k} \delta_G(V_i, V_j)$. A special case of a k-graph-partition of G is a *cut*, where $F = \delta_G(W, V \backslash W) =: \delta_G(W)$ for a subset $W \subseteq V$ of the nodes. If $W = \{v\}$, we write $\delta_G(v)$ instead of $\delta_G(\{v\})$.

If it is necessary to distinguish between different graphs, we extend all notational conventions introduced above by a supplementary index. For a graph G we write, for example, $\phi_G, d_G(v), \delta_G(w)$, etc.

A *path* P in G from v_0 to v_k is a sequence of the form $P = (v_0, e_1, v_1, e_2, \ldots, e_k, v_k)$ where $v_i \in V$ for $i = 0, \ldots, k$, $e_i \in E$ for $i = 1, \ldots, k$, and v_{i-1}, v_i are the two end-nodes of edge $e_i, i = 1, \ldots, k$. The *length* of a path is its number of edges, that is, the length of P in the notation above is k. The nodes v_0 and v_k are the *end-nodes of P* and the nodes $v_1, \ldots, v_{n-1}$ are the *inner nodes* of P. We use the notation $e \in P$ or $v \in P$, if $e \in E$ is an edge of P or $v \in V$ is an inner node of P. We denote by $V(P)$ and $E(P)$ the set of inner nodes and edges, respectively. That is, for a path $P = (v_0, e_1, v_1, e_2, \ldots, e_k, v_k)$ in G we have $V(P) = \{v_1, \ldots, v_{k-1}\}$ and $E(P) = \{e_1, \ldots, e_k\}$. It is common to distinguish between paths and walks, where walks are paths without node repetition. However, we will only use paths without node repetition and only use the notion of a path. A *cycle* or *closed path* is a path where the end-nodes are identical. A cycle is *odd* if its length is an odd number, and an *odd-cycle node-cover* is a set of odd-cycles such that each node is inner-node of at least one odd-cycle. Two paths P_1 and P_2 are *node-disjoint* if $V(P_1) \cap V(P_2) = \emptyset$, that is, if the intersection of the two sets of inner-nodes is empty, and analogously, P_1 and P_2 are *edge-disjoint* if $E(P_1) \cap E(P_2) = \emptyset$, that is, if the intersection of the two sets of edges is empty.

A graph G is *connected* if for each pair of different nodes u and v there exists a path in G from u to v. More generally, a graph G is *k-node (edge)-connected* ($k \in \mathbb{N}$) if there exist k node (edge)-disjoint paths from u to v for each pair of different nodes u and v. The maximal connected subgraphs in G with respect to edge set inclusion are the *components of G*. A node $v \in V$ of a graph G is an *articulation node* if the subgraph G_v has more components than G, and analogously, edge e of G is a *bridge* if G_e has more components than G.

Chapter 1

Practical Background

The research presented in this thesis was motivated by the complex network design problem our project partner E-Plus Mobilfunk GmbH has to solve. As we will see, this network design problem contains a series of subproblems which must be solved and integrated; each complex in itself. The network planners have to make many decisions to design a network in which communication requirements between users of the network can be fulfilled with high probability at any time. These decisions are interdependent and influenced by a variety of parameters such as the available hardware including its cost, the estimated communication requirements, a given budget, or an existing network. The hardware of the network and the manpower needed to maintain the network are expensive, and therefore, it is the target of the network planner to find the right balance between investment cost and network quality. (There is no precise definition of the quality of a network, but it should be expressed in terms of customer satisfaction and the ability to fulfill communication requirements.)

This chapter provides a description of the main tasks in the design of a (mobile-) communication network, including a description of the transport network design problem, which is the problem we will focus on in the subsequent chapters.

The following point is worth mentioning: Our project partner is a mobile-communication network operator, and consequently, we developed models and algorithms to solve parts of the overall design process of a mobile-communication network. However, the described models and solution approaches can often be used in the transport network design process of other telecommunication providers. In fact, only minor details of the transport network are *particular* to mobile-communication, and the used abstract mathematical models focus on the important aspects of network planning, which appear for other types of networks as well.

The contents of this chapter is the following. We briefly explain the architecture of a *global system for mobile communication* (GSM)[1] network including its interfaces and subsystems as defined in the standards (GSM-1.02, 1993; GSM-1.04, 1994) and described less technically in (Mouly and Pautet, 1992). Based on the architecture, we describe a typical decomposition of the overall design process and present, in more detail, the node location problem, the traffic

[1]Originally, GSM was an acronym for *Groupe Spécial Mobile*, which was a working group of the *Conférence Europeénne des Administration des Postes et des Télécommunications* between 1982 and 1987.

forecast, and the switching and transport network planning problems.

Architecture of a GSM network

A GSM-900 (or GSM-1800)[2] network consists of well-defined hierarchically ordered subsystems together with interfaces between the subsystems and to the external world. The three main subsystems are the *network and switching subsystem* (NSS), the *base station subsystem* (BSS) and the *mobile stations* (MSs). Throughout our description we neglect the fourth subsystem, the *operation subsystem* (OSS), because it is responsible for maintenance issues and less important for the network design problems under consideration. There are two external interfaces: One to the customers and the other to external networks which might be the network of another mobile-communication network operator or a public switched telephone network (PSTN). Altogether, the subsystems and the interfaces define the hierarchical system shown in Figure 1.1.

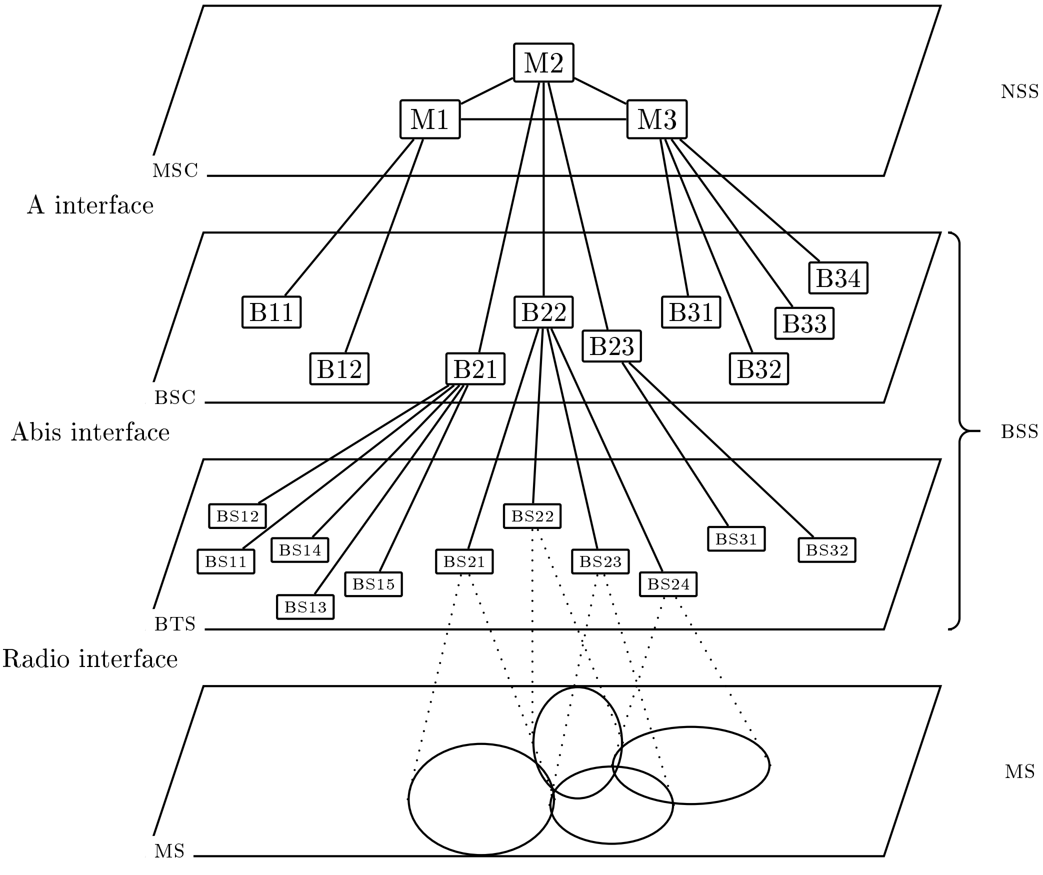


Figure 1.1: Architecture of a GSM network

[2]The values 900 and 1800 indicate the range of the frequency band in which the radio network operates. A GSM-900 network utilizes parts of the frequency band 890–960 MHz. Analogously, a GSM-1800 network utilizes parts of the frequency band 1710–1880 MHz.

The connecting lines between the nodes in Figure 1.1 represent the logical relations in a GSM network. Across the three interfaces (A, Abis and radio interface), these relations set up a collection of stars, that is, the nodes from a "lower" subsystem are partitioned such that the nodes of one partition are connected to a single node of the subsystem "above". Only in the top level subsystem, the NSS, there are logical relations between internal nodes.

MS The mobile stations build the lowest level of the hierarchy. They consist of the *mobile equipment* (ME), which everybody knows as mobile phone, and the *subscriber identity module* (SIM), which stores part of the administrative information about a particular user. The SIM is the smart card inserted into an ME. The MS*s* face on one side the users of the network and on the other side the BSS. Notice that a user can be a human being who uses the interfaces display, loudspeaker, or keyboard, but a user can also be terminal equipment like a computer or a facsimile.

BSS The base station subsystem has two interfaces to other subsystems: the so-called *radio interface* to the MS*s*, which is the only interface that definitely transmits data through the air, and the so-called *A interface* to the NSS. There are no external interfaces. The main task of the BSS is to connect the MS*s* with the NSS through its two classes of network components: the *base transceiver stations* (BTS*s*) which are linked to the MS*s*, and the *base station controllers* (BSC*s*) which are linked to the NSS. Within *a* BSS, each BTS communicates with exactly one predetermined BSC across the so-called *Abis interface*.

The BTS*s* accommodate the *radio transceivers* (TRX*s*) and handle the link protocols to communicate with the MS*s*. The service area of a single BTS is divided into 3-4 sectors, and for each sector there are up to 3 TRX*s*. The first TRX in a sector can manage 6 communications (channels[3]) and each additional TRX in the same sector manages 8 communications. (There is a difference of two channels since these are reserved for broadcast information at the first TRX in the sector.) Thus, a BTS with three sectors and one TRX within each sector can manage 18 ($= 3 \cdot 6$) communications, and with two TRX*s* in each sector 42 ($= 3 \cdot (6+8)$) communications. A typical BSC today can control up to 128 TRX*s*.

NSS The network and switching subsystem builds the highest level of the internal hierarchy and has interfaces to the BSS and external networks. The NSS consists of different classes of network components: the *mobile services switching centers* (MSC*s*) and two types of data bases: *home location registers* (HLR*s*) and *visitor location registers* (VLR*s*).

An MSC is a switching node[4] with additional functionality to operate a mobile-communication network. For instance, hand-over management, location update, and user authentication are such additional functionalities. The HLR*s* (there might be more than one if the capacity of a single HLR does not suffice to handle all users of the network) store administrative information about all registered users. This includes, for instance, the current geographical position and basic data about the contract of the user. With each MSC a VLR is associated. It stores the subset of information from the HLR*s* about

[3]A channel is the basic unit of data transmission. Its capacity is 64 kbit/s.

[4]Switching is the ability to interconnect the channels attached to each network node and to move traffic from each incoming channel to the appropriate outgoing channel whenever the requirement neither originates nor terminates at the node.

those users currently in the service area of the particular MSC. This information at the VLR is frequently updated since the users of a mobile-communication network typically move between service areas. The capacity of the VLR and the MSC, respectively, is not unbounded. In fact, in the design process the VLR must be dimensioned such that the expected number of users in the service area of the associated MSC can be handled, and the MSC must be dimensioned such that the computing power suffices to handle the expected maximum number of so-called *busy hour call attempts* (BHCA).

It is worth mentioning that the users of a network are not only the subscribers of the provider operating this network. There exist so-called *roaming* contracts between providers of different mobile-communication networks. These contracts permit a user of one network to utilize the resources of another network, if the user is not in the service area of the "home" network. Typically, providers in different countries sign such contracts to make it possible for their costumers to communicate with their mobile phone in foreign countries.

The MS*s* communicate with the geographically close BTS*s* even if no communication is requested by a user. Periodically, every MS negotiates with its close BTS*s* to choose the best one to receive or submit a communication request. Here, the best option is a compromise between the available radio channels and the quality of the received signal. The negotiation is repeated as the MS changes its geographical position and the result is stored in an HLR of the NSS and the VLR of the visited MSC.

Switching and transport network

In Figure 1.1, we presented the logical structure of a GSM network. As we have seen, there are logical links across the three interfaces (A, Abis, and radio interface) and between the MSC*s* in the NSS. Knowing this, we can now describe the logical communication path for a particular communication between users MS A and MS B (see Figure 1.2).

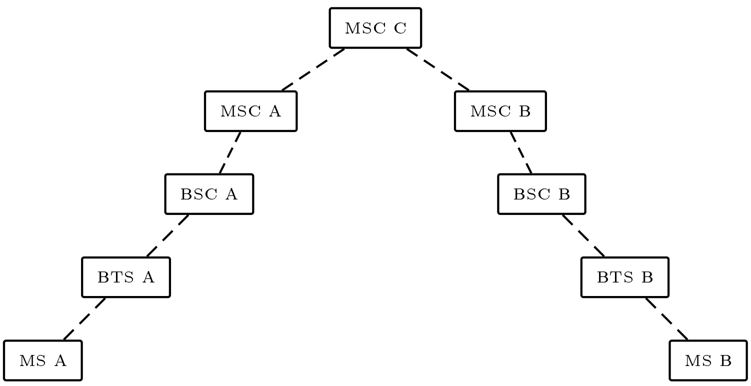


Figure 1.2: Logical communication path

MS A knows its (currently) associated BTS A, which itself is (permanently) associated with BSC A, which itself is (permanently) associated with MSC A. On the NSS level, the communication path might pass additional MSCs, like the MSC C in Figure 1.2. Eventually, the path follows again the hierarchy: from MSC B over BSC B and BTS B down to user MS B. Altogether, this is the *logical* communication path between MS A and MS B.

The logical links do not suffice to establish the communication. In addition, there must be physical transmission links with sufficient transmission capacity, and physical transmission paths which connect the end-nodes of each logical link using the transmission links. In this context, it is common to distinguish between two networks: the *switching network* and the *transport network*. The switching network consists of BSCs and MSCs as nodes and logical communication requirements as links. The transport network consists of *digital cross-connects* (DXCs) as nodes and physical transmission lines as links. DXCs are very flexible network components which automatically map ("cross-connect") digital signals from incoming to outgoing ports. This is done according to a map stored in electronic form. With each node of the switching network there is a DXC attached, which is responsible for adding signals from or to drop signals to the switching node. A logical link of the switching network is realized in the transport network over paths between the digital cross-connects attached to the end-nodes of the link.

Figure 1.3 illustrates the interworking of switching and transport network for the communication between users MS A and MS B of Figure 1.2 on the NSS level. The dashed lines in Figure 1.3 represent the logical path MSC A – MSC C – MSC B of Figure 1.2 on the NSS level. For both logical links in this path, a transmission path must be provided in the transport network. In our example these paths are represented by straight lines. One transmission path MSC A – DXC A – DXC C – MSC C for the logical link MSC A – MSC C, and another transmission path MSC C – DXC C – DXC D – DXC B – MSC B for the logical link MSC C – MSC B. Both paths start and end with the link between the MSC and its attached DXC. In between, a transmission path never "visits" an MSC. In our example, the second transmission path passes DXC D, but it does not use a link between MSC D and DXC D.

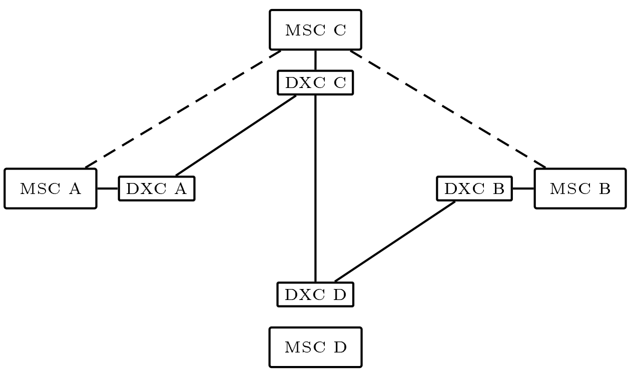


Figure 1.3: Switching and transport network

Sources of failures

In the introduction of this thesis, we described the importance of network survivability within the planning process and presented a brief list of network failures that happened in the past. Knowing technical details of the network structure and the different network components, we can now describe sources of network failures in more detail. A good network design should take care of the severe failure categories and provide strategies and resources to decrease service disruption time. We distinguish between two categories of network failures: problems due to insufficient network resources, and service disruption of network facilities.

Network outages due to overload of some network resources are the result of an under-dimensioned network. It is difficult to estimate the financial loss caused by overload of network facilities, since none of the communications in progress is affected. The network simply does not accept any further communication request that requires free resources at one of the overloaded network facilities. The main sources for this category of failures are the following:

- There is no available radio channel at the BTS to serve the communication request of a particular MS. In this case, the network planner should have assigned more radio channels to the BTS, or should have installed more BTS*s* within the bottleneck area.
- The capacity of a transmission link may not suffice to establish additional communication requests which use this link. In this case, the network planner should have increased the transmission capacity of the link, or should have routed less communication requests over paths which use this link.
- The MSC might fail to operate because of too many call attempts. Recall, an MSC can only handle a certain number of busy hour call attempts. In this case, the network planner should have chosen an MSC with a larger maximum number of busy hour call attempts, or should have installed more MSC*s* within the bottleneck area.
- The capacity of the VLR may not suffice to handle all users which are currently within the area covered by the associated MSC. In this case, the network designer should have chosen a larger capacity for the VLR, or should have installed more MSC*s* within the bottleneck area.

The second category of network failures comprises the failure of network facilities due to the following reasons:

- Human errors like a cable cut or power supply disruption.
- Natural disaster like an earth quake, a fire, a hurricane, etc.
- A bug in the software controlling the switching nodes or digital cross-connects.
- A bug in the hardware of the switching nodes or digital cross-connects.

Since 1992 it is mandatory for network operators in the United States to report on network failures affecting more than 30 000 customers to the US Federal Communications Commission.

Based on those failure reports, (Kuhn, 1997) showed that network failures due to overload of some network facilities account for 44% of all costumer-minute outages. Human errors account for 28% and natural disaster for 18%. Interestingly, software and hardware failures are rather rare. Just 2% of costumer-minute outages are due to software failures and 7% due to hardware failures.

The planning process

As we have seen so far, the design of a telecommunication network involves many decisions. These include the locations of BTSs, BSCs, MSCs, DXCs, VLRs and HLRs, the individual capacity of this equipment, the transmission capacity between these locations, and the paths in the switching and transport network which are used to establish requested communications. These decisions are not independent from each other, but from a practical point of view the overall problem is too complex to be handled within a single step. A natural approach is the decomposition of the problem into a series of subsequent problems such that each individual problem can be handled. The resulting sequence of subproblems should be iterated until a reasonable compromise between cost, maintenance effort, and quality has been found. A typical sequence of subproblems for network design is shown in Figure 1.4.

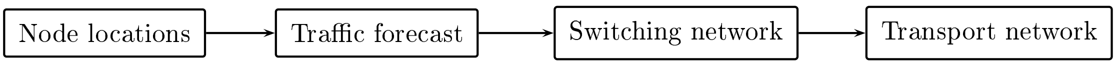


Figure 1.4: Sequence of subproblems for network design

First, a good choice of **node locations** together with an assignment of BTSs to BSCs and BSCs to MSCs must be determined such that a cost-effective and manageable network can be determined in the subsequent planning steps. This decision depends on the expected number of subscribers and the expected communication traffic. Node locations are usually fixed for a long period and are not subject to frequent redesign. In contrast, the rest of the planning process is periodically applied.

The **traffic forecast** has to be performed for each type of service within the network. Different types are, for instances, telephony, signaling, or private networks. The forecast depends on the network structure, marketing data, and data from traffic accounting in the operating network. The result of this planning step is a demand matrix (in Erlang[5]) for each different type of service.

Given a demands matrix for each different service, together with *quality of service* (QoS) requirements and routing strategies as input, the *logical* connections between the switching nodes and the *logical* paths to route the estimated traffic are determined in the **switching network** planning. As most important QoS requirement an upper bound for the probability that a communication request has to be rejected must be respected. This bound is called

[5]Agner Krarup Erlang (1878–1929) was a Danish mathematician and the pioneer for dimensioning telephone networks. He developed the fundamental Erlang-B formula (Erlang, 1917) to estimate the capacity of a transmission link for given probability distributions of call attempts and call durations. For more information on A.K. Erlang see the biography (Brockmeyer et al., 1948).

blocking probability and must be specified by the network designer. By means of some variation of the Erlang-B formula (see Section 1.3), a demand in terms of channels that satisfies this QoS requirement can be calculated.

As a result of the switching network planning, an input demand matrix in terms of Erlang has been transformed into a demand matrix in terms of channels. The **transport network** planning is in charge to provide physical transmission capacity that is either rented, or installed using microwaves or fiber-optic links. Sufficient capacity must be provided to ensure that each communication requirement can be routed according to specified survivability requirements.

1.1 Node locations and their hierarchy

In the initial setup and in the major expansion steps of a telecommunication network it is necessary to decide the locations of the permanent network equipment. In a GSM network these are the antennas (BTS*s*), nodes of the switching network (BSC*s* and MSC*s*), additional nodes of the transport network (DXC*s*), and data bases (VLR*s* and HLR*s*). Simultaneously, the logical hierarchy of the network must be determined. That is, within the same planning step the assignment of BTS*s* to BSC*s* and the assignment of BSC*s* to MSC*s* must be performed.

The decisions about the BTS locations depend on the costumers of the network. It is not necessary to cover the complete potential service area of a network which is often a whole country. Instead, it suffices to cover those areas in which the customers (or expected costumers) typically move. These areas are, for instance, the cities, the main traffic roads like highways and railways, and tourist attractions.

The decisions about the BSC locations depend on the BTS locations. It is necessary to place BSC*s* such that it is possible to assign each BTS to a BSC. This assignment is constrained by the capacity of a BSC. As we already mentioned, a typical BSC today can control up to 128 TRX*s* and there are up to 12 TRX at each BTS. Hence, one can only assign about 10–12 BTS*s* to a single BSC. An optimization problem could be to find the minimum number of BSC locations such that an assignment of BTS*s* to BSC*s* is possible and no BSC must control more than 128 TRX*s*.

The decision of the MSC locations includes the decision about the VLR locations, since one VLR is associated with each MSC. Consequently, it it is useful to decide the respective locations simultaneously. Analogous to the previous problem, it is necessary to place MSC*s* such that it is possible to assign each BSC to an MSC, and again, this decision is constrained by capacities. In this case, the capacities are the maximum number of busy hour call attempts an MSC can handle and a maximum number of users a VLR can handle. Thus, the sum of busy hour call attempts generated at the BSC*s* assigned to an MSC must not exceed the maximum the MSC can handle, and the sum of costumers of these BSC*s* must not exceed the capacity of the VLR. Typically, there are several types of MSC*s* or VLR*s* available, each type with a particular capacity at a certain cost. Thus, the choice of the capacity of the equipment belongs to the planning decisions.

The DXC locations are almost determined with the decision about the MSC locations. Each MSC is connected with a DXC in order to provide the connection between the switching and the transport network. Therefore, it is reasonable to place a DXC at any MSC location. The locations of the HLR*s* are not particularly restricted.

All mentioned location problems depend on the *potential* locations for the equipment. Not every place within the network area is a suitable location for a node. The place must either be owned by the network operator or available to rent. In any case, there must be enough physical space for the entire equipment.

1.2 Traffic forecast

Suppose that the locations of the network nodes and the assignment of BTS*s* to BSC*s* and BSC*s* to MSC*s*, respectively, have been determined in the initial planning phase. Then the communication traffic between pairs of locations has to be estimated in order to obtain a suitable basis for the decisions of the link sizes and the communication paths. The traffic is measured in the unit *Erlang* which is defined as follows:

$$1 \text{ Erlang} := (\text{utilization time}) \; / \; (\text{length of time interval}) \quad .$$

As an example, if a costumer generates 12 minutes traffic during one hour, this amounts to 0.2 (= 12 / 60) Erlang. This unit depends on the length of the time interval, and the amount of traffic depends on the considered time interval. But, for which time interval should a network be designed? To our knowledge, there are no precise models to answer this question, but experience has shown that those consecutive 60 minutes during a day are appropriate, in which the network generates the maximum amount of traffic. These 60 minutes are called *peak hour* or *busy hour*.

The traffic originating at a particular BTS (or in a certain area containing several BTS*s*) depends on regional demographic characteristics such as the number of inhabitants or costumers, how many are self-employed, employed, students, unemployed, etc. Given the traffic forecast of the BTS*s*, it is rather easy to propagate it through the hierarchy of a GSM network. For each BSC, the traffic of the controlled BTS*s* is accumulated and, similarly, for each MSC the traffic generated at the served BSC*s* is accumulated. A certain percentage remains within the BSS served by this MSC and the rest is split among the other MSC*s* and the external networks. This way, demand matrices (in terms of Erlang) for the NSS and for each BSS can be estimated.

In the setup phase of the network these rough estimations are the basis of the traffic forecast. If the network is operating for several years, the traffic forecast becomes easier in the sense that input data can be obtained from accounting at the network nodes.

1.3 Switching network

For every type of service, the traffic forecast yields a demand matrix in terms of Erlang. In the switching network planning, these matrices are transformed into demand matrices in terms of channels, where one channel corresponds to a transmission rate of 64 kbit/s. The decisions to be made during the switching network planning are the following:

- For each communication demand a set of alternative paths must be chosen together with the fraction of the demand routed over each of these paths.
- The maximum blocking probability must be specified, which is the probability that a call attempt must be rejected.
- The demands in terms of channels must be calculated such that it is guaranteed that a call cannot be rejected with a probability larger than the maximum blocking probability.

As we mentioned before, the logical communication paths in the BSS of a GSM network are predetermined. In the NSS there is need for planning of a logical routing between the MSCs. The logical communication paths are usually chosen according to deterministic rules. For example, a typical rule is to route each demand on exactly two node-disjoint paths, where the primary path carries the bigger fraction of the demand (usually about 80%) and the secondary path carries the remaining demand. For this rule, the number of node-disjoint paths and the way the demand is distributed over the paths must be determined.

Given such rules, it is still difficult to determine the particular paths to use. The reason for this is twofold. There are many alternative ways to route a communication demand on, say, two node-disjoint paths and there does not exist a well-defined cost function to guide the decision since the costs depend on the transmission facilities. In this planning step, however, it is not yet specified which transmission facilities are needed to satisfy the communication demands. In fact, this belongs to the design decision within the transport network planning. To overcome the problem of the missing cost function, it is common practice to approximate the cost function of the transport network. There, the cost of a certain capacity usually depends on the length of the physical connection and the capacity itself. Economies of scale apply since the costs per channel monotonically decrease with increasing capacity. To approximate this type of cost function one often chooses the length of a physical connection as costs per channel and attempts to bundle the (Erlang-) demands in order to take advantage of the mentioned economies of scale. Bundling means to choose the logical routings of different communication demands such that the same link is used by many paths, and such that the number of used links is small. This has several advantages. First, if several (independent) communication demands use the same link for transmission, then it is possible to gain from multiplexing since the capacity of a link is chosen such that the blocking requirement is satisfied on average. For independent communication demands it is unlikely that call attempts always arrive simultaneously, and therefore, the same transmission capacity can often be used for both demands. As further advantage of bundling the (Erlang-) demands, it becomes easier to manage the network as the number of logical communication links decreases.

Given the specification of the logical routing, the communication demands in terms of channels have to be calculated such that the blocking probability does not exceed a specified maximum value. It is possible to view a communication link of capacity N (see Figure 1.5) as a system in which calls arrive according to a Poisson process with average arrival rate λ, and calls always find free resources available until all N channels are occupied. Calls are blocked if all N channels are occupied.

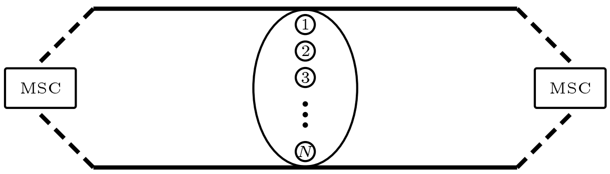


Figure 1.5: Communication link

Let such a communication link with capacity N be represented by a finite queue with states $0, \ldots, N$. (0 represents the state that no call is served by the link.) Furthermore, suppose that the call holding time process is exponential, with parameter μ. Figure 1.6 visualizes the state diagram of such a queue.

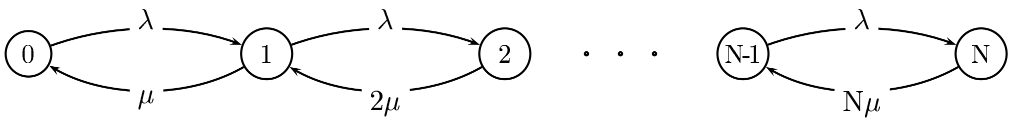


Figure 1.6: State diagram of a queue

There is a rate λ of moving to the next larger state, due to call arrivals, and there is a rate $k\mu$ of moving from state k to state $k-1$, due to call completion. Apparently, if the queue is empty, it is only possible to move to state 1, and if the queue is full, it is only possible to move to state $N-1$. Let $p_k, k = 0, \ldots, N$, be the probability that exactly k calls are in the queue. Then the probability p_N is the blocking probability since the queue in state N is full, and any further arriving call must be blocked. These probabilities must satisfy the normalization equation

$$\sum_{k=0}^{N} p_k = 1 . \tag{1.1}$$

Carrying out a more detailed analysis, the validity of the following equations can be shown:

$$\begin{aligned} \mu p_1 &= \lambda p_0 , && (1.2)\\ (\mu k + \lambda) p_k &= \lambda p_{k-1} + \mu(k+1) p_{k+1} , \qquad k = 1, \ldots, N-1 , && (1.3)\\ \mu N p_N &= \lambda p_{N-1} . && (1.4) \end{aligned}$$

Notice, however, these equations can also be derived by inspection from the state diagram in Figure 1.6. Just consider each state of the queue individually, and apply that for each state the sum of the outgoing and ingoing rates (weighted by the respective state probabilities) must be zero. Now, calculating $p_k, k = 1, \ldots, N$, in dependence of p_0 from (1.2) – (1.4) yields

$$p_k = \frac{1}{k!} \left(\frac{\lambda}{\mu} \right)^k p_0 ,$$

and substituting p_k for $k = 1, \dots, N$ in (1.1), we obtain

$$p_0 = \left(\sum_{k=1}^{N} \frac{1}{k!} \left(\frac{\lambda}{\mu} \right)^k \right)^{-1},$$

and thus the blocking probability p_N is

$$p_N = \frac{1}{N!} \left(\frac{\lambda}{\mu} \right)^N \left(\sum_{k=1}^{N} \frac{1}{k!} \left(\frac{\lambda}{\mu} \right)^k \right)^{-1}.$$

Let us consider an example. Suppose that the capacity of a (small) link with call arrival rate of $\lambda = 720$ calls/hour and an average holding time of $1/\mu = 3$ minutes/call should be determined. This gives a demand of $\lambda/\mu = 720 \cdot 3/60 = 36$ Erlang. How many channels guarantee a maximum blocking of 1%, for instance? Table 1.1 shows that 48 channels suffice to achieve a blocking probability less than 1% and with decreasing number of channels the blocking probability increases much faster than linearly. This indicates that the blocking probability for fixed demand in Erlang is strongly dependent on the chosen number of channels.

Number of channels	24	30	36	42	48	54
Blocking probability (in %)	37.40	23.66	12.19	4.45	0.96	0.11

Table 1.1: Pairs of blocking probabilities and channels for 36 Erlang

Furthermore, Table 1.2 shows the required number of channels for various blocking probabilities and demands.

	Erlang								
Blocking	20	40	60	80	100	150	200	300	400
1%	30	53	75	96	117	170	221	324	426
5%	26	47	67	86	105	154	202	298	394
10%	23	43	62	80	97	188	142	279	370

Table 1.2: Channel values for different blocking probabilities and Erlang-demands

The ratio between required channels and Erlang-demands is decreasing for increasing Erlang-demands and fixed blocking probabilities. For instance, for blocking probability 1% and a demand of 20 Erlang this ratio is $30/20 = 1.5$, and for 400 Erlang the ratio decreases to $426/400 = 1.06$. A blocking probability of 10% yields similar values; the ratio decreases from $23/20 = 1.17$ for 20 Erlang to $370/400 = 0.93$ for 400 Erlang. This indicates that it might pay to design a sparse network concentrating the Erlang-demands on a small number of logical links.

1.4 Transport network

The transport network consists of the physical equipment and the physical links to transfer the digital signals. Given logical demands in terms of channels (as a result of the switching network planning), the topology, the transmission capacities, and the transmission paths to route each of these demands have to be determined in the transport network planning. The nodes of the transport network are digital cross-connects, and the links (connections between transport network nodes) can be leased lines, microwave links, fiber-optic cables, etc. The task is to design the transport network in the most cost-efficient way, while satisfying the following constraints:

- Provide a set of transmission paths for each demand and assign a channel value to each of these paths such that the sum of channels over all paths is at least the value of the demand.
- Provide alternative transmission paths for those scenarios of network failures which are considered important.
- Provide capacity for data transmission such that for each network component the capacity is at least as big as the sum of channels over all passing transmission paths through the component. This must also be satisfied in the considered failure scenarios.

The cost parameters in this planning phase are rather accurate. The network nodes have prices given by the different vendors, leased lines have a given cost structure, and the cost of microwave connections depends on the number of required repeaters. Hence, it is possible to optimize with respect to the cost of the network infrastructure in the transport network planning.

Today, the available transmission capacities are defined in two different hierarchies: the *plesiochronous digital hierarchy* (PDH) and the *synchronous digital hierarchy* (SDH). Before we describe these hierarchies, we briefly review the origins of digital signal transmission. For long time, the analog signal was the basis for voice transmission. In the early 1960s, the first networks based on digital transmission were installed and since then there has been a clear drift towards the more reliable digital transmission. However, there has been a need to integrate the digital transmission technologies and the analog terminal equipment (e.g., telephones of the end-users). Therefore, the principle of *pulse code modulation* (PCM) has been introduced (see (ITUT-G.711, 1988)). It transforms an analog signal into a digital signal, which is then transmitted to the receiver and there retransformed into an analog signal. The technical terms for these transformations are modulation and demodulation. Already at the early stages of digital signal transmission development, the standardization organizations agreed to sample data at a rate of 8 kHz, that is, 8000 times per second, and to allocate 8 bits to digitize each sample. These values have been standardized in (ITUT-G.711, 1988). Therefore, the length of one sample is 125μs ($= 1/8000$s), and sampling rate and length determine the size of a channel to be 64 kbit/s $= 8000 \cdot 1/\text{s} \cdot 8$ bit. From the beginning of digital transmission it was clear that there is a need for transmission rates higher than 64 kbit/s and therefore the principle of multiplexing was introduced through the PDH.

Plesiochronous Digital Hierarchy (PDH)

In general, a *multiplexer* (MUX) receives a certain number of equal rate data streams as input and yields one data stream at a higher transmission rate as output. The associated *demultiplexer* (DEMUX) recovers the original rate data streams from the high transmission rate data stream, see Figure 1.7. Digital multiplexing is based on the principle of *time division multiplexing* (TDM). The high transmission rate data stream is divided into successive intervals, each carrying information of successive channels. The intervals associated with a particular channel appear periodically and thus the demultiplexer is able to interpret the data stream correctly. Obviously, the transmission rate depends on the length of a time interval and, in consequence, smaller time intervals lead to higher transmission rates.

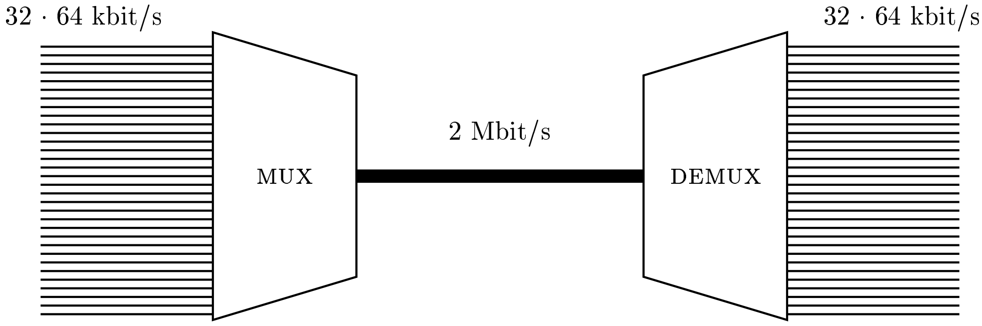


Figure 1.7: (De-) Multiplexing of 64 kbit/s into a primary rate digital signal

The PDH was developed in the 1960s and standardized in (ITUT-G.702, 1988). However, it was not possible to achieve a common agreement about the multiplexing hierarchy among the three main parties North America, Europe and Japan, and therefore, there exist three different hierarchies with different data transmission rates at the respective digital signal levels. Table 1.3 shows the values of the digital signal levels.

Level	Europe	North America	Japan
0	64	64	64
1	2048	1544	1544
2	8448	6312	6312
3	34368	44736	32064
4	139264	139264	97728

Table 1.3: International plesiochronous digital hierarchies (in kbit/s)

The primary rate digital signal in Europe is 2048 kbit/s[6]. This translates to $32 \cdot 64$ kbit/s. However, the capacity for data transmission is only 30 channels since two channels are reserved for frame synchronization and signaling. For instance, the information which bits belong to which stream must be conveyed to the receiving demultiplexer, and synchronization of the data

[6]We write 2048 instead of 2,048 or 2.048 to avoid misinterpretation, as recommended in (ITUT-G.701, 1993).

streams must be performed. In general, the aggregated rate must be a little bit higher than the sum of the input rates to accommodate the overhead multiplexing information. Rewriting the number of channels in the higher level of the hierarchy as sum of user data and overhead data yields

$$
\begin{array}{rrcrcr}
30\cdot & 64\ \text{kbit/s} & + & 2\cdot 64\ \text{kbit/s} & = & 2048\ \text{kbit/s}\ , \\
4\cdot & 2048\ \text{kbit/s} & + & 4\cdot 64\ \text{kbit/s} & = & 8448\ \text{kbit/s}\ , \\
4\cdot & 8448\ \text{kbit/s} & + & 9\cdot 64\ \text{kbit/s} & = & 34368\ \text{kbit/s}\ , \\
4\cdot & 34368\ \text{kbit/s} & + & 28\cdot 64\ \text{kbit/s} & = & 139264\ \text{kbit/s}\ .
\end{array}
$$

With different values this also applies to the PDH in North America and Japan, respectively. We wish to point out a property of the available user channels of the PDH (which also applies to the SDH, as we will see soon). Cutting out the overhead channels, the following one-to-one correspondence between transmission rates and capacities in channels can be seen:

$$
\begin{array}{rcr}
2048\ \text{kbit/s} & \longleftrightarrow & 30\ \text{channels}\ , \\
8448\ \text{kbit/s} & \longleftrightarrow & 120\ \text{channels}\ , \\
34368\ \text{kbit/s} & \longleftrightarrow & 480\ \text{channels}\ , \\
139264\ \text{kbit/s} & \longleftrightarrow & 1920\ \text{channels}\ .
\end{array}
$$

The number of user channels of each digital signal level is an *integer* multiple of the number of user channels of *each* smaller digital signal level. This is important to note since it is possible to take mathematically advantage of this property. We will see this in more detail in the subsequent chapters.

Synchronous Digital Hierarchy (SDH)

The synchronous digital hierarchy (SDH) was developed in the 1980s and standardized in (ITUT-G.803, 1997). The driving forces were the need for higher transmission rates than those provided by the PDH and some deficiencies of the PDH. The SDH is again a multiplexing hierarchy with the levels shown in Table 1.4.

Level	STM-1	STM-4	STM-16	STM-64
kbit/s	155520	622080	2488320	9953280

Table 1.4: Synchronous digital hierarchy

The basic unit is the so-called *synchronous transport module* (STM) which is able to transport so-called *virtual containers* (VCs). The capacity of an STM-4, STM-16, and STM-64 is exactly four times the capacity of an STM-1, STM-4, and STM-16, respectively. There is no overhead since the multiplexing information is contained in the VCs which build themselves a hierarchy of transmission rates as shown in Table 1.5.

The values of the virtual containers reflect the need to integrate the digital signal levels of the different PDH's. For instance, the digital signal level 1 of the North American PDH fits into

Level	VC-11	VC-12	VC-2	VC-3	VC-4
kbit/s	1664	2240	6848	48960	150336

Table 1.5: Virtual Container Hierarchy of the SDH

an VC-11, and the European digital signal level 1 fits into an VC-12. Four VC-11s, three VC-12s, or one digital signal level 2 of the North American PDH can be multiplexed into an VC-2 which itself fits seven times into an VC-3. Finally, three VC-3s are multiplexed into an VC-4 which then is transported over an STM-1.

One of the big improvements of the SDH over the PDH is the elimination of the requirement to demultiplex the complete hierarchy in order to get access to a particular channel. Instead, it is possible to extract a low bit-rate virtual container from a higher bit-rate digital signal with so-called *add-drop multiplexers* (ADMs).

The cost structure

The available capacities of the transport network come at different cost. The information about the costs of these capacities is rather accurate; for every particular type of communication link, the structure of the cost can be described by length and capacity dependent cost functions. These functions are structurally different on microwave links, leased lines or fiber-optic cables. In this section, we present some typical cost functions for two important types of links: leased lines and microwave links.

Leased lines

A mobile-communication network operator may rent part of its network from a leased line provider. As illustrative example, Figure 1.8 shows a typical cost structure of DEUTSCHE TELEKOM, the only leased line provider in Germany when we started working on this problem. The cost structure of new leased line providers (who started business after the German telecommunication market has been deregulated in 1998) is similar.

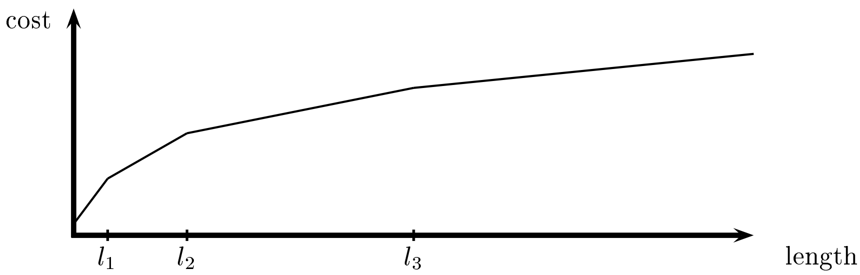


Figure 1.8: Typical cost structure for leased lines

Figure 1.8 illustrates that the typical cost structure for a particular capacity such as 30, 480, or 1920 channels (2, 34, or 140 Mbit/s), is piecewise linear and monotonically increasing with the length of the link. The slope, however, decreases with the length of the link. In more detail, it decreases at specified lengths such as l_1, l_2, l_3 in Figure 1.8. Changing the view from a particular capacity to available capacities on a particular link, as illustrated in Figure 1.9 for the digital signal levels 1, 3, and 4 of the PDH, it is easy to see that economies of scale are large. It depends on the distance between two end-nodes of a link, but as a rule of thumb, a capacity of six to eight times 30 channels is more expensive than a capacity of 480 channel, and three times 480 channels are more expensive than 1920 channel.

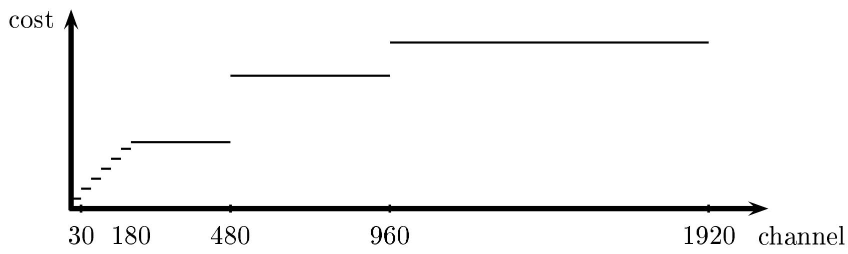


Figure 1.9: Typical cost structure for leased lines on a link

Microwave

The cost structure for microwaves is different since the maximum transmission distance through the air is limited. To guarantee a specified quality of the signal, it is necessary to periodically amplify the digital signal. So-called repeaters are needed after every interval of constant distance. For instance, if the maximum distance without amplification of the signal is 50 kilometer, two repeaters are necessary on a link of length 130 kilometer. The necessary amplification of the digital signals dominates the structure of the cost function for microwaves. Figure 1.10 illustrates a typical cost structure of microwaves for a particular capacity. The cost in dependence of the length is a staircase function with equal width intervals of constant cost. The width l_1 is the distance at which a repeater becomes necessary, and $l_i = i \cdot l_1$.

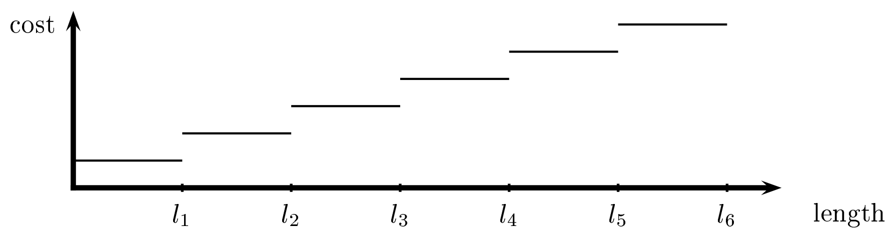


Figure 1.10: Typical cost structure for microwave connections

Figure 1.11 illustrates the cost structure of a microwave connection for a particular link. Similar to leased lines, it is a staircase function with considerable economies of scale.

Figure 1.11: Typical cost structure for microwaves on a link

The illustrated capacity and cost structures add significant complexity to the design of a transport network. In the sense of complexity theory, the problem becomes difficult because of the discrete structure of the available capacities. It is not possible, for instance, to install 30.5 channels. Even if this is a required value, the network designer must choose between 30 and 60 channels. The illustrated economies of scale cause further difficulties since it is not clear at which point it is appropriate to choose a 480 channel link instead of several 30 channel links. Of course, as shown in Figure 1.9, there exists a break-even point from which on it is cheaper to use the higher capacity link, but it might pay to choose this higher capacity even below the break-even point because of the additional capacity. Using the larger capacity of 480 channels instead of six to eight times 30 channels, additional $240-300$ channels are available at relatively small extra cost. Because of this additional capacity on one link it might be possible to decrease capacities on other links, and thus the overall network cost might decrease.

Routing and survivability

Besides the capacity decisions, transmission paths to satisfy logical communications demands are settled within the transport network planning. For these demands in terms of channels, which are given as output of the switching network planning, one or several transmission paths must be provided such that the value of each communication demand is less than or equal to the sum of channels assigned to the individual paths of a demand.

Several issues are related to the routing decisions. Obviously, the transmission paths must only use links with available transmission capacity, and if one considers a link between two particular nodes of the transport network, and sums up the values of those transmission paths using the link, then the capacity of the link must be at least as large as this sum. (Notice that additional capacity might be necessary to satisfy survivability requirements.) In fact, the capacity and the routing decision should simultaneously be performed, since only discrete units of capacities are available and economies of scale can be huge. It might be less expensive to route some communication demands on transmission paths which are, at first glance, expensive in the sense that these paths are longer in terms of used links and geographical distances. However, the overall solution might be less expensive.

We already described in the introduction of this thesis that the impact of a network component failure can be tremendous. Hence, as additional planning requirement, todays communication networks must be survivable in the sense that it is possible to deal with single component failures. If only a single transmission path is used to satisfy a particular communication demand, then no communication is possible if any network component (node or link) of this path does not operate. The strategies to deal with this problem fall in two main categories. First, it is possible to choose transmission paths which use different network components. This provides partial survivability since the failure of a single network component cannot result in complete loss of communication. However, as drawback of such an approach, it is not possible to protect all communication. As alternative approach, the network planner can provide routings for individual failure scenarios. In this case, the network management is more complex, but it can be handled in modern communication networks since the equipment is controlled by software.

Preview

In the rest of this thesis, we focus on the problem of planning the transport network of a telecommunication provider. We present mathematical models integrating the described topology and capacity decisions together with routing planning under survivability requirements. Afterwards, these models and the sets of solutions for the different problems are investigated, and eventually, algorithms to solve the problems are developed. We focus on the transport network design problem because of its relevance within the overall network design process. The optimization target in this planning step is minimization of costs that can be specified with high accuracy. Furthermore, the structure of solutions is too complicated to be handled without the support of sophisticated mathematical models and algorithms.

Chapter 2

Mathematical Models

As we have seen in the previous chapter, the design of a (mobile-) communication network contains a series of complex problems. In the remainder of the thesis we focus on one particular subproblem: the transport network design problem. That is, given the result of the switching network planning, the target is to choose *simultaneously* the topology of the transport network, the capacities to install on the transmission links, and the routings of all logical communication requirements for normal operation and all single network component failures, such that the cost of the network is minimal.

In this chapter we first present the mathematical models developed in cooperation with `E-Plus` Mobilfunk GmbH. We consider two ways to model the *discrete capacity structure* and three ways to achieve *survivability* in the network. Any combination of a capacity and a survivability model leads to a different mixed-integer programming formulation. (All combinations have been integrated in our network dimensioning tool DISCNET.) We integrate the routing planning in the normal operation *and* all single network component failures. In contrast to other models described in the related literature, we do not *assume* a given routing for the normal operation and deal with the capacity expansion and routing planning for failure situations only. The integrated planning makes considerable investment cost reductions possible, and with our models, it is possible to accomplish these cost reductions. At the end of this chapter, we survey the related research that focuses on theoretical and practical results for mathematical models which integrate the topology decision, the capacity planning and the routing problem for the communication demands.

2.1 Supply graph, demand graph, and operating states

The considered problems have the following in common. The input consists of two graphs on the same node set V, the

- *supply graph* $G = (V, E)$ and the
- *demand graph* $H = (V, D)$.

The set V consists of the nodes of the transport network. In our case, V often corresponds to the locations of the digital cross-connects (see page 15). The edge set E of the supply graph G is the set of all physical links which may be used (in the planning period). Different transmission links (representing different technologies, e.g., microwave connections, fiber optic cables, leased lines of different providers) are represented by *parallel edges.* The demand graph H (for the planning period) contains an edge whenever there is a positive demand in terms of channels between its two end-nodes. For each edge $uv \in D$ of the demand graph, the value

- $d_{uv} \in \mathbb{Z}_+$ is the *communication demand* between nodes u and v.

We denote by $V_D := \{v \in V : \exists u \in V \text{ and } uv \in D \text{ with } d_{uv} > 0\}$ the subset of the nodes V with at least one positive emanating demand. While the characteristics of the supply graph are relatively stable (they change, e.g., with hardware and suppliers), demand predictions are based on statistical analysis and forecasting. They are revised frequently, and scenario analysis has to be made to take different possible evolvements of the market into account.

In practice, different service classes arise such as voice traffic, signaling traffic, data traffic, etc. In a model appropriate to cover different classes of services, it is necessary to allow parallel edges in the demand graph as well. From a modeling point of view this is no problem. However, to reduce the size of the problem instances and the complexity of the model, we decided in the initial phase of the project to aggregate the demands of different service classes between the same pairs of end-nodes into a single demand.

As a major planning requirement, the network to design should be well-prepared against failures of single nodes or edges. For each communication demand, a routing in all *operating states* S must be provided. These are the *normal operating state*, where all nodes and supply edges are operational, and a subset of the *failure states*, in which a single node $w \in V$ or a single supply edge $e \in E$ is non-operational. We denote these operating states by $s = 0$, $s = w$, and $s = e$, respectively. By definition,

$$S \subseteq \{0\} \cup V \cup E\ .$$

Note, for each edge and each node of the supply graph the network designer can specify whether its failure state should be considered, that is, whether routing tables must be provided for the state in which this particular node or edge fails. For every $s \in S$, we denote by $G_s = (V_s, E_s)$ the supply graph in operating state s, where V_s is the set of nodes that are still operational in operating state s, and, likewise, E_s is the set of the operational edges in operating state s. Similarly, the demand graph in operating state $s \in S$ is $H_s = (V_s, D_s)$, where D_s is the set of surviving communication demands. (A demand $uv \in D$ is *surviving* in operating state $s \in S$, if none of its end-nodes failed, that is, if $s \notin \{u, v\}$.)

Example 2.1 See Figure 2.1 for an example of a supply and a demand graph. The demand graph has three edges with associated communication demands $d_{ac} = 120$, $d_{ce} = 80$, and $d_{cd} = 30$. The eight edges of the supply graph represent the potential physical links. The available capacities are not specified, yet. We assume that the set of operating states is $S = \{0\} \cup V$; no edge failure situations have to be considered. Notice that it is possible to reduce the set S to S

$= \{0\} \cup (V \setminus \{c\})$ in this example, since the demand graph $H_c = (V_c, D_c)$ has an empty demand set D_c. □

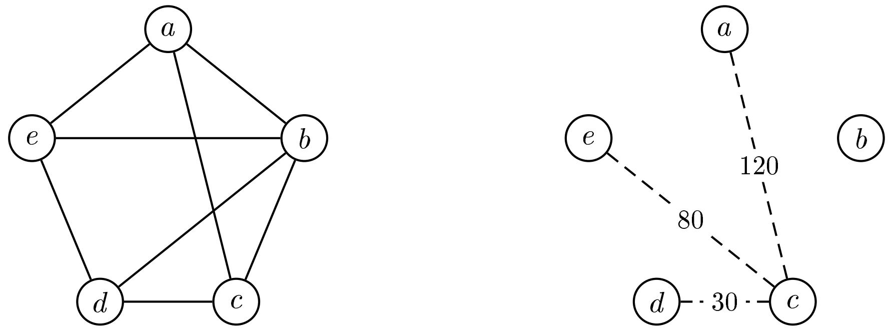


Figure 2.1: Example of a supply and a demand graph

2.2 Capacity models

In the target network, sufficient capacity must be installed on the edges of the supply graph such that these can accommodate a feasible routing of the communication demands. In principle, a leased line provider might offer arbitrary capacities and in this case we would introduce *continuous* variables to model the capacity decision. However, not a single provider offers arbitrary capacities and, in fact, it is common to offer capacities of the hierarchies PDH and SDH. From DEUTSCHE TELEKOM, for instance, it is possible to rent as PDH capacities multiples of 2 Mbit/s links (30 channels), multiples of 34 Mbit/s links (480 channels) and multiples of 140 Mbit/s links (1920 channels). Consequently, we decided to model a discrete capacity structure.

In the following, we distinguish between two different capacity models to cope with this discrete structure. In the first case, the set of possible capacities for each edge of the supply graph is given as a finite set. This model was introduced in (Dahl and Stoer, 1998). In the second model, a small set of "basic capacities" is given. These basic capacities must satisfy the property that each one is an integral multiple of all smaller basic capacities. As we have seen in Chapter 1, this is a reasonable assumption for many network design problems. Special cases of the second capacity model (without survivability requirements) have been investigated by several researchers as we will see in Section 2.6.

Existing network

Designing a telecommunication network is a dynamic process. Changes in the demand forecast or new offers of transmission capacity suppliers make redesigning the network necessary once in every planning period. Although it is theoretically possible to design the network in every planning period from scratch, it is reasonable to assume that parts of the network exist and cannot be changed in the short run.

There are two main reasons for this assumption. First, there might exist long term contracts with a supplier which have to be fulfilled. Second, but equally important, it is necessary to guarantee a certain stability in the network in order to reduce the maintenance effort for switching to a new network topology. It is not desired to change the whole network just because of small changes in the demand forecast.

To model this planning requirement, we assume that every edge $e \in E$ of the supply graph is already equipped with an initial capacity $C_e^0 \in \mathbb{Z}_+$ (possibly $C_e^0 = 0$), the so-called *free capacity*. This assumption applies to all capacity models formulated in this thesis. Of course, the free capacity has a certain cost. However, since this cost can be predetermined, it is not part of the optimization and can be ignored. We set the cost K_e^0 to install capacity C_e^0 on edge e to zero, that is, $K_e^0 := 0$.

Similarly, it might be desired to fix part of the routings of the logical communication demands in order to reduce the maintenance effort when the network has to be reconfigured. However, we do not consider such a planning requirement here.

Maintenance Costs

It is an option to include maintenance costs in the mathematical model. Such an issue can be modeled with cost coefficients either on individual routing paths or on the flow through transmission links. However, we do not add such cost coefficients to our models, since we believe that it is problematic to model maintenance costs in this way. It is too difficult to provide accurate data.

2.2.1 Discrete Capacities

As we already noted, the available capacities in many practical applications have a discrete structure, since the equipment (multiplexer or digital cross-connect) is technologically restricted to certain capacities. The capacity model Discrete Capacities provides the most general form to deal with such an underlying capacity structure. For every supply edge, the set of capacities that might be installed on this particular edge is given as a finite set. Installation costs are associated with each capacity.

Data

For each $e \in E$, there is a finite set of capacities specified by the following data:

- $T_e \in \mathbb{Z}_+$ is the *number of possible capacities* that can be installed in addition to the free capacity,
- $C_e^t \in \mathbb{Z}_+$, $1 \le t \le T_e$, are the *potential capacities* (we assume $C_e^0 < C_e^1 < \cdots < C_e^{T_e}$), and
- $K_e^t \in \mathbb{Q}_+$, $1 \le t \le T_e$, are the respective *cost* of installing capacity C_e^t.

Instead of the original values, it is useful to consider the *incremental capacity* and *cost* values and values

- $c_e^t := C_e^t - C_e^{t-1}$, $1 \leq t \leq T_e$,
- $k_e^t := K_e^t - K_e^{t-1}$, $1 \leq t \leq T_e$.

For notational convenience, we set $c_e^0 := C_e^0$ and $k_e^0 := K_e^0$. The capacities $C_e^1, \dots, C_e^{T_e}$ are the *breakpoint capacities* and T_e is the number of *breakpoints*.

Variables

For each edge $e \in E$, we introduce an ordered set of *integer capacity variables*, the *0/1-variables* $x(e,0) \geq x(e,1) \geq \cdots \geq x(e,T_e)$. Since we assume that a free capacity C_e^0 is always installed, we set $x(e,0) := 1$. Choosing capacity C_e^τ for some breakpoint $0 \leq \tau \leq T_e$, is equivalent to setting $x(e,0) = x(e,1) = \cdots = x(e,\tau) = 1$ and $x(e,\tau+1) = \cdots = x(e,T_e) = 0$.

Mathematical formulation

The objective is to minimize the total cost of installing the necessary capacities on the edges of the supply graph. This is formulated as

$$\min \sum_{e \in E} \sum_{t=1}^{T_e} k_e^t x(e,t) \ . \tag{2.1}$$

For every supply edge $e \in E$, the associated 0/1-variables must satisfy the *ordering constraints*

$$1 = x(e,0) \geq x(e,1) \geq \cdots \geq x(e,T_e) \geq 0 \ , \tag{2.2}$$

and the *integrality constraints*

$$x(e,t) \in \{0,1\} \ , \tag{2.3}$$

for all $t = 1, \dots, T_e$. For notational convenience, we introduce *continuous capacity variables*

$$y(e) := \sum_{t=0}^{T_e} c_e^t x(e,t) \ , \tag{2.4}$$

for all $e \in E$. These (auxiliary) variables represent the capacity installed on supply edges.

2.2.2 Divisible Basic Capacities

The number of capacity variables for Discrete Capacities becomes large if it is possible to install any combination of the capacities of the PDH. Even if one only considers those capacities which are not dominated by larger capacities (at smaller cost), one typically needs more than 20 integer variables to model the potential capacities of a single supply edge. To overcome this problem of too many integer capacity variables, we introduce the second capacity model Divisible Basic Capacities which can be employed if a set of “divisible” capacities is given.

Data

We denote by $T = \{\tau_1, \dots, \tau_n\} \neq \emptyset$ the set of *technologies*, one for each different type that can be installed on a supply edge. Associated with each technology $\tau \in T$ are a *basic capacity* C^τ and edge dependent installation cost which include a fixed cost and a length-dependent cost which varies with the total length of an edge. For this capacity model, the basic capacities must satisfy the *divisibility property*. That is, the increasingly ordered basic capacities $C^{\tau_1} \leq C^{\tau_2} \leq \dots \leq C^{\tau_n}$, must satisfy

$$\frac{C^{\tau_{i+1}}}{C^{\tau_i}} \in \mathbb{Z}_+ ,$$

for all $i = 1, \dots, n-1$. We refer to the smallest basic capacity C^{τ_1} as the *unit capacity*. The available capacities for a particular supply edge are the integer combinations of the basic capacities of the technologies. For each supply edge $e \in E$,

- $K_e^\tau \in \mathbb{Q}_+$, $\tau \in T$, is the cost of installing one unit of capacity C^τ.

For notational convenience, we assume that all technologies $\tau \in T$ are available for all supply edges. This assumption is not necessarily satisfied in practical application, but it can be assumed without loss of generality. If a technology $\tau \in T$ is not available for a supply edge $e \in E$, one can easily overcome this problem by setting $K_e^\tau := \infty$.

Variables

We introduce a non-negative *integer capacity variable* $x(e, \tau)$ for every supply edge $e \in E$ and every technology $\tau \in T$ to denote the integer multiples of C^τ combined into the capacity of edge e.

Mathematical formulation

Again, the objective is to minimize the total cost of installing the necessary capacities on the edges of the supply graph. This is formulated as

$$\min \sum_{e \in E} \sum_{\tau \in T} K_e^\tau x(e, \tau) \ . \tag{2.5}$$

The constraints that must be satisfied for every supply edge $e \in E$ and every technology $\tau \in T$ are the *integrality constraints*

$$x(e, \tau) \in \mathbb{Z}_+ \ . \tag{2.6}$$

For every supply edge $e \in E$, the continuous capacity $y(e)$ is calculated from

$$y(e) = C_e^0 + \sum_{\tau \in T} C^\tau x(e, \tau) \ . \tag{2.7}$$

2.2.3 A note on the capacity models

Both capacity models have advantages and disadvantages. With DISCRETE CAPACITIES a finite set of available capacities is given for every supply edge. This model is very general and provides the flexibility to cover every practical situation with a discrete capacity structure. It is possible to employ this model even if the network designer incorporates different transmission capacity suppliers which provide different capacities that do not satisfy the divisibility property. As a further advantage of DISCRETE CAPACITIES, it is possible to model the capacity structure of future networks. It is worth mentioning, that the capacity structure of links with *wavelength division multiplexers* at both end-nodes can only be covered with this capacity model, and *not* with the model DIVISIBLE BASIC CAPACITIES.

If many capacities are available, however, the size of the resulting problem instance may become too large to be handled within a cutting plane algorithm (our solution approach). In order to obtain reasonable upper bounds on the optimal objective function value in acceptable running times, it might be necessary to considerably reduce the number of available capacities beforehand. As a drawback of such a step, the lower bounds for the optimal solution value of the restricted problem instance are not necessarily lower bounds for the optimal solution value of the original problem. For instance, if an optimal solution for the restricted problem is not optimal for the original problem.

The second model, DIVISIBLE BASIC CAPACITIES, is a special case of the first capacity model. If the available capacities have a particular structure, it provides a way to handle large numbers of capacities without reducing them artificially in a preprocessing step. As we already mentioned, we encountered such a structure in many (but not all) problem instances. Despite the advantages of DIVISIBLE BASIC CAPACITIES, there are practical situations where the available capacities do not satisfy the divisibility property. This happens, for instance, if one incorporates transmission capacities from the PDH and the SDH in the same network design problem. In this case, the model DISCRETE CAPACITIES has to be used. We shall further note that even if the capacities satisfy the divisibility property network designers sometimes prefer to use DISCRETE CAPACITIES since any integer combination of the basic capacities might not be appropriate.

2.3 Combining capacities, demands and routings

In addition to the selection of a capacity for each supply edge, a feasible routing for each communication demand must be computed for a solution. For each operating state all demands must be routed simultaneously, and the capacities must suffice to accommodate the routings. This suggests a multicommodity-flow formulation of the routing problem with variables expressing the flow on a path or over a supply edge.

Even though the routings are inherently integral, we model them with continuous variables, since a model with integer routing variables cannot be solved to a satisfactory degree with the available mathematical methods. It is possible to formulate the problem with integer variables, but not to solve it. Instead, we suggest to solve the network design problem with continuous path variables and to employ a postprocessing algorithm to deal with the non-integral routing variables. Our computational experiments revealed that the required changes in the postprocessing are not too extensive since the routings are often “almost” integral for “practical” parameter selections.

Combining any of the two capacity models with multicommodity-flow conditions for the non-failure situation, we now state the basic mixed-integer programming formulations. These do not include survivability requirements beside a restriction on the number of supply edges in the paths that can be used to route a particular demand. Such restrictions should be employed, if the transmission time depends on the number of edges in the transmission path, or if one wants to avoid long paths in order to decrease the probability of a failure of a path component. Typically, the length of valid paths is only restricted in the normal operating state. In exceptional cases, like failure situations, it is satisfactory to provide any routing.

Data

For each demand $uv \in D$, the *path-length restriction parameter*

- $\ell_{uv} \in \mathbb{N}$ is the maximum number of supply edges allowed in any uv-path on which parts of the demand between the end-nodes u and v is routed in the normal operating state.

For each operating state $s \in S$ and each demand edge $uv \in D_s$, let $\mathcal{P}_s^{uv}$ denote the set of *valid uv-paths* in G_s. If s is the normal operating state ($s = 0$), a uv-path in $G = G_0$ is valid if its length (number of edges) is at most ℓ_{uv}. We call such a path *short*. If s is a failure state then any uv-path in G_s is valid.

Variables

For each operating state $s \in S$, each demand edge $uv \in D_s$, and each path $P \in \mathcal{P}_s^{uv}$, we define a variable $f_s^{uv}(P)$, called *flow* or *path variable*, representing the communication demand between the nodes u and v routed on path P in operating state s.

Mathematical formulation

The constraints for the routings in the normal operating state are the capacity, demand and non-negativity constraints. The *capacity constraints*

$$\sum_{uv \in D} \sum_{P \in \mathcal{P}_0^{uv} : e \in P} f_0^{uv}(P) \leq y(e) \ , \tag{2.8}$$

for each supply edge $e \in E$, express that the sum of the flow values over all paths containing edge e in the normal operating state must not exceed the capacity $y(e)$. For each demand $uv \in D$, the values of the path variables must sum up to the value d_{uv} in the normal operating state. This yields the *demand constraints*

$$\sum_{P \in \mathcal{P}_0^{uv}} f_0^{uv}(P) = d_{uv} \ , \tag{2.9}$$

for every demand $uv \in D$. Additionally, for every demand $uv \in D$ and every path $P \in \mathcal{P}_0^{uv}$, the path variables must satisfy the *non-negativity constraints*

$$f_0^{uv}(P) \geq 0 \ . \tag{2.10}$$

We now formally define the two network design models that do not cover survivability requirements. The model for DISCRETE CAPACITIES is

$$\begin{aligned}
\min \sum_{e \in E} \sum_{t=1}^{T_e} k_e^t x(e,t) & & \\
x(e,t) - x(e,t-1) &\leq 0 \ , & e \in E,\ t = 1, \ldots, T_e \ , \\
x(e,t) &\in \{0,1\} \ , & e \in E,\ t = 1, \ldots, T_e \ , \\
C_e^0 + \sum_{t=1}^{T_e} c_e^t x(e,t) &= y(e) \ , & e \in E \ , \\
\sum_{uv \in D} \sum_{P \in \mathcal{P}_0^{uv} : e \in P} f_0^{uv}(P) &\leq y(e) \ , & e \in E \ , \\
\sum_{P \in \mathcal{P}_0^{uv}} f_0^{uv}(P) &= d_{uv} \ , & uv \in D \ , \\
f_0^{uv}(P) &\geq 0 \ , & uv \in D,\ P \in \mathcal{P}_0^{uv} \ ,
\end{aligned}$$

and the model for DIVISIBLE BASIC CAPACITIES is

$$\begin{aligned}
\min \sum_{e \in E} \sum_{\tau \in T} K_e^\tau x(e,\tau) & & \\
x(e,\tau) &\in \mathbb{Z}_+ \ , & e \in E,\ \tau \in T \ , \\
C_e^0 + \sum_{\tau \in T} C^\tau x(e,\tau) &= y(e) \ , & e \in E \ , \\
\sum_{uv \in D} \sum_{P \in \mathcal{P}_0^{uv} : e \in P} f_0^{uv}(P) &\leq y(e) \ , & e \in E \ , \\
\sum_{P \in \mathcal{P}_0^{uv}} f_0^{uv}(P) &= d_{uv} \ , & uv \in D \ , \\
f_0^{uv}(P) &\geq 0 \ , & uv \in D,\ P \in \mathcal{P}_0^{uv} \ .
\end{aligned}$$

The path formulation of the continuous multicommodity-flow problem has, in general, an exponential number of path variables. However, it provides an easy way to model path-length restrictions and other survivability constraints, as we will see in the following section. In Chapter 4, we describe the column generation approach suggested by (Minoux, 1981) to solve non-simultaneous continuous multicommodity-flow problems with path variables. In some cases, this approach can be time-consuming and therefore, we also make use of a polynomial formulation of the multicommodity-flow problem for the normal operating state (without path-length restrictions). This formulation contains edge-flow variables, and demands are aggregated with respect to their end-nodes. The following paragraph describes this in more detail.

A set $Q \subseteq V$ is an *aggregation* of the demands if there exists an assignment of the demands $uv \in D$ to either u or v such that the union of the assigned end-nodes equals Q. The nodes in Q are called *commodities*. (For simplicity, we assume that $uv \in D$ is always assigned to u.) Furthermore, for every commodity $k \in Q$ and every supply edge $e = ij \in E$, the two *edge-flow variables* $f^k(ij)$ and $f^k(ji)$ represent the directed flow over edge e to satisfy part of the demand of commodity k. With this notation, constraints (2.8) – (2.10) of the multicommodity-flow problem can be substituted by the constraints

$$\sum_{j \in V} f^k(ji) - \sum_{j \in V} f^k(ij) = \begin{cases} d_{uv}\ , & uv \in D,\ k = u,\ i = v\ , \\ 0\ , & \text{else}\ , \end{cases} \qquad k \in Q,\ i \in V\ , \tag{2.11}$$

$$\sum_{k \in Q} (f^k(ij) + f^k(ji)) \le y(e)\ , \qquad e = ij \in E\ , \tag{2.12}$$

$$y(e), f^k(ij), f^k(ji) \ge 0\ , \qquad k \in Q,\ e = ij \in E\ , \tag{2.13}$$

where the capacity $y(e)$, $e \in E$, in (2.12) is calculated from (2.4) or (2.7) for integer x-variables. This formulation is polynomial in the size of the input since it contains $2|Q||E|$ edge-flow variables and $|Q||V| + |E|$ constraints. In consequence, the linear relaxation of this mixed-integer program can be solved in polynomial time (see (Khachyan, 1979)). Note, this mixed-integer program is a formulation of the capacitated network design problem *without* survivability requirements, if no path-length restrictions for the demands have to be satisfied.

2.4 Survivability models

In the capacitated network design models presented in the previous section we have ignored survivability requirements. Now, we focus on survivable network design. As we already described, the available transmission capacities increase while the respective costs of capacity substantially decrease. Hence, optimal networks with respect to network cost tend to be very sparse with huge capacities on a few number of transmission links and, in consequence, the failure of a network component causes severe losses in terms of money and costumer confidence. It is necessary to cope with network component failures at the planning stage, and over the past ten years different models to increase the quality of a network through protection and restoration mechanisms have been proposed in the literature. Survivability can be introduced in two steps: in the normal operating state with constraints imposed on the routing, and in case of a single node or edge failure through (partial) rerouting of surviving demands.

The survivability model DIVERSIFICATION imposes restrictions on the routing in the normal operating state. For every demand, the routing is diversified on several paths such that the failure of any network component does not affect more than a specified percentage of the demand. The two survivability models RESERVATION and PATH RESTORATION use different strategies to reroute part of the demand in a failure state. In both models, no restrictions on the normal operating state routing are imposed. RESERVATION makes use of complete rerouting since it is admissible to change parts of the routing of a demand even if not affected by the particular failure. In contrast, PATH RESTORATION partially reroutes demands affected by a particular failure, and maintains the unaffected routings of the normal operating state. To the best of our knowledge, the survivability model DIVERSIFICATION has been introduced in (Dahl and Stoer, 1998), the model RESERVATION in (Minoux, 1981), and the model PATH RESTORATION in (Wu, 1992).

2.4.1 DIVERSIFICATION

The survivability model DIVERSIFICATION is based on the following idea. If the maximum percentage of a demand value that is allowed to flow through any network component is restricted, then the maximum loss of this demand is restricted if any single network component is not operating.

Data

For every demand edge $uv \in D$, the *diversification parameter*

- δ_{uv}, $0 < \delta_{uv} \leq 1$, is the maximum fraction of the demand d_{uv} allowed to flow through any edge or node (other than nodes u and v) of the supply graph.

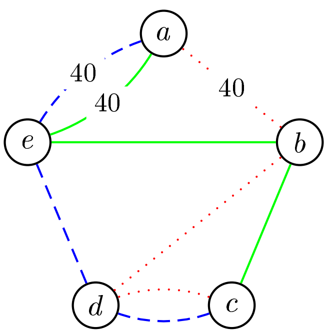


Figure 2.2: Example routing for DIVERSIFICATION

Figure 2.2 shows a feasible routing of the demand edge ac from Example 2.1 with diversification parameter $\delta_{ac} = 2/3$. The demand $d_{ac} = 120$ is routed on three paths, each carrying 40 channels. No two paths are node-disjoint, but there is also no component with more than $2/3 \cdot 120 = 80$ channels flowing through it. The flow through node b, for instance, is equal to $66.\overline{6}\%$ of the demand since only two paths, both with flow value 40, have b as inner node.

As in the example above, it is not required to route a demand on pairwise node-disjoint paths. The diversification parameter only imposes a restriction on the sum of the flow values of all paths passing through a network component. However, if a demand $uv \in D$ is routed on exactly $\lceil 1/\delta_{uv} \rceil$ paths then these paths are node-disjoint. Therefore, we sometimes say that DIVERSIFICATION implements survivability by means of a "node-disjoint" routing.

Mathematical formulation

For every demand $uv \in D$ and every node $w \in V \backslash \{u, v\}$, the *node-flow constraints* are

$$\sum_{P \in \mathcal{P}_0^{uv} : w \in P} f_0^{uv}(P) \leq \delta_{uv} d_{uv} \ , \tag{2.14}$$

and for every demand $uv \in D$ and every path $P \in \mathcal{P}_0^{uv}$ with $|P| = 1$, the *edge-flow constraints* are

$$f_0^{uv}(P) \leq \delta_{uv} d_{uv} \ . \tag{2.15}$$

For a demand $uv \in D$ and a node $w \in V \backslash \{u, v\}$, the summation in the node-flow constraints is over all short paths between the demand end-nodes u and v that contain node w as inner node. These constraints restrict the amount of flow dedicated to a particular demand that goes through a particular node, that is, they ensure a flow of no more than a fraction δ_{uv} of the total demand d_{uv} through a single node w in the normal operating state. The node-flow constraints imply that every edge $e \in E$ carries no more than $\delta_{uv} d_{uv}$ of the demand $uv \in D$, unless it is a supply edge between the demand end-nodes u and v. To cover the latter case, the edge-flow constraints are used. These only exist, of course, if E contains edges between u and v. The constraints (2.14) and (2.15) ensure that the flow between u and v is *diversified*, that is, routed on at least $\lceil 1/\delta_{uv} \rceil$ paths.

2.4.2 RESERVATION

No restrictions on the routing in the normal operating state are imposed for the survivability model RESERVATION. Instead, spare capacity is used (that has to be provided) to reroute the surviving demands if a single node or single edge of the supply graph is not operating. For every demand edge, the network designer specifies the percentage of the demand value that should still be routable in case of such a failure. A feasible solution must contain enough spare capacity to accommodate the routings of all demands $uv \in D_s$ in all failure situations $s \in S \backslash \{0\}$.

Data

For every demand edge $uv \in D$, the *reservation parameter*

- ρ_{uv}, $0 \leq \rho_{uv} \leq 1$, is the fraction of the demand d_{uv} that must be satisfied if a single node or a single edge of the supply graph fails.

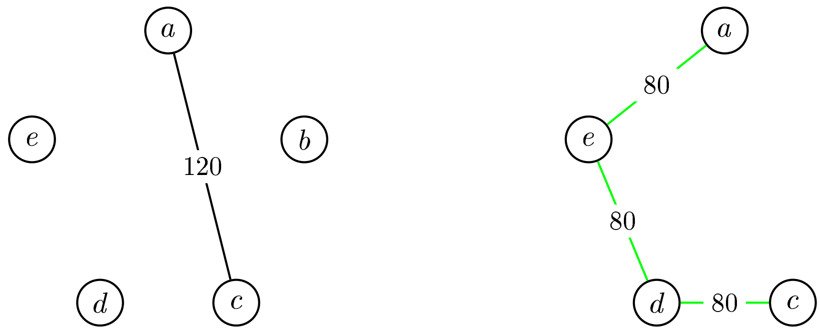


Figure 2.3: Example routing for RESERVATION

Figure 2.3 shows a feasible routing of the demand edge ac from Example 2.1 for the normal operating state and the failure state $s = b$ with reservation parameter $\rho_{ac} = 2/3$. The routing of this demand in the failure state $s = b$ is different from the routing in the normal operating state, even though no path of the normal operating state routing is affected by this failure situation.

Mathematical formulation

For every failure state $s \in S\backslash\{0\}$ and every supply edge $e \in E_s$, the *capacity constraints* are

$$\sum_{uv \in D_s} \sum_{P \in \mathcal{P}_s^{uv} : e \in P} f_s^{uv}(P) \leq y(e) \; . \tag{2.16}$$

For every failure state $s \in S\backslash\{0\}$ and every demand $uv \in D_s$, the *demand constraints* are

$$\sum_{P \in \mathcal{P}_s^{uv}} f_s^{uv}(P) = \rho_{uv} d_{uv} \; , \tag{2.17}$$

and, additionally, for every valid path $P \in \mathcal{P}_s^{uv}$, the *non-negativity constraints* are

$$f_s^{uv}(P) \geq 0 \; . \tag{2.18}$$

No path-length restrictions are imposed in failure situations. Therefore, the summation in inequalities (2.16) is over *all* variables that correspond to paths containing a particular supply edge. Notice that only surviving demands of operating state s have to be routed and that the paths only use supply edges operating in state s. Recall, if $s = w$ for some $w \in V$, the demand and supply edges emanating from w in the demand graph H and the supply graph G, respectively, are *not* surviving. Inequalities (2.16) guarantee in failure situations that the flow through surviving supply edges does not exceed its capacity. Inequalities (2.17) guarantee for all surviving demand edges that the specified percentages of the demands survive the failure with state dependent routings, and inequalities (2.18) formulate the necessary non-negativity of the flow variables in all failure situations.

2.4.3 PATH RESTORATION

The survivability model PATH RESTORATION can be viewed as a compromise between the two extreme models DIVERSIFICATION and RESERVATION. Similar to the model RESERVATION, no restrictions on the routings in the normal operating state are imposed, and a specified percentage of each demand value should still be routable in case of a single component failure. However, in contrast to the previous model, the normal operating state routing is linked with the failure state routing. For every failure state $s \in S \backslash \{0\}$, the routings that do not contain s have to be maintained, and only those routings that are affected by the failure of s can be rerouted.

The capacity of those paths affected by a failure are released and can be used for rerouting. Therefore, the spare capacity of an edge $e \in E_s$ in a failure situation $s \in S \backslash \{0\}$ is the sum of the spare capacity from the normal operating state plus the released capacity of those paths which include e and s, that is,

$$y(e) - \sum_{uv \in D} \sum_{P \in \mathcal{P}_0^{uv} : e \in P} f_0^{uv}(P) \quad + \sum_{uv \in D} \sum_{P \in \mathcal{P}_0^{uv} : s \in P, e \in P} f_0^{uv}(P) \ ,$$

is the spare capacity of $e \in E_s$ in $s \in S \backslash \{0\}$. Recall, a failure state $s \in S \backslash \{0\}$ is a component of the supply graph, and thus we can write $s \in P$ to denote that the failing component s is contained in path P.

Data

For every demand edge $uv \in D$, the *path restoration parameter*

- σ_{uv}, $0 \leq \sigma_{uv} \leq 1$, is the fraction of the demand d_{uv} that must be satisfied end-to-end in a single node or single edge failure, *without* rerouting the paths not affected by the particular failure situation.

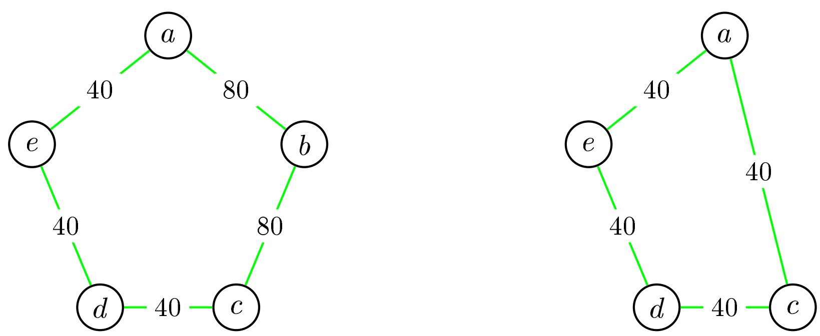


Figure 2.4: Example routing for PATH RESTORATION

Figure 2.4 provides a feasible routing of the demand edge ac from Example 2.1 with restoration parameter $\sigma_{ac} = 2/3$. The left part shows a routing for the normal operating state. Since the "surviving flow" in the failure states $s = d$, $s = e$, $s = ae$, $s = ed$, and $s = dc$ is greater than or equal to $\sigma_{ac}d_{ac}$, the normal operating state routing already provides a feasible routing for these failure states. However, in the failure states $s = b$, $s = ab$, and $s = bc$, at least 40 channels of the demand must be rerouted to satisfy a minimum of $66,\overline{6}\%$. This is satisfied through the additional path $P = \{ac\}$ with the flow value $f_b^{ac}(P) = 40$, as illustrated on the right-hand side of Figure 2.4.

Mathematical formulation

Here, the *capacity* and *demand constraints* for failure situations have to respect the unaffected normal operating state routings. Since a restriction on the length of the paths used to route a demand in the normal operating state is possibly imposed, we denote by $\mathcal{P}_0^{uv} \cap \mathcal{P}_s^{uv}$ the set of surviving short paths in $\mathcal{P}_s^{uv}$ for each failure situation $s \in S\backslash\{0\}$ and each demand $uv \in D_s$. That is, $P \in \mathcal{P}_0^{uv} \cap \mathcal{P}_s^{uv}$ if and only if $P \in \mathcal{P}_s^{uv}$ and P has at most ℓ_{uv} supply edges. The following constraints are needed for every failure state $s \in S\backslash\{0\}$. For every surviving supply edge $e \in E_s$, the *capacity constraints* are

$$\sum_{uv \in D_s} \left(\sum_{P \in \mathcal{P}_0^{uv} \cap \mathcal{P}_s^{uv} : e \in P} f_0^{uv}(P) + \sum_{P \in \mathcal{P}_s^{uv} : e \in P} f_s^{uv}(P) \right) \leq y(e) \; . \tag{2.19}$$

These constraints express that the installed capacities must suffice to accommodate all short paths of all surviving demands which are not affected by the failure *and* those paths used to reroute part of the failing flow. For every surviving demand $uv \in D_s$, the *demand constraints* are

$$\sum_{P \in \mathcal{P}_0^{uv} \cap \mathcal{P}_s^{uv}} f_0^{uv}(P) + \sum_{P \in \mathcal{P}_s^{uv}} f_s^{uv}(P) \geq \sigma_{uv} d_{uv} \; , \tag{2.20}$$

and, additionally, the *non-negativity constraints* for every $P \in \mathcal{P}_s^{uv}$ are

$$f_s^{uv}(P) \geq 0 \; . \tag{2.21}$$

Constraints (2.20) express the guaranteed survivability for each demand. The sum of the values of the unaffected short paths together with the paths used to reroute part of the failing flow must be at least the specified percentage of the demand value. Notice that no rerouting of a demand $uv \in D$ in failure state $s \in S\backslash\{0\}$ is necessary, if the surviving part of the normal operating state routings suffices in this particular failure situation. In this case, it holds $\sum_{P \in \mathcal{P}_0^{uv} \cap \mathcal{P}_s^{uv}} f_0^{uv}(P) \geq \sigma_{uv} d_{uv}$.

2.4.4 A note on the survivability models

The survivability models aim at different kind of protection against the failure of a single network component. Setting the diversification parameter δ_{uv} for the demand edge $uv \in D$, we require that at most $100 \cdot \delta_{uv}\%$ of the demand value d_{uv} is routed through any node (other than u and v) or any edge of the network. This implies routings which provide "node-disjoint" paths, each of them carrying at most $\delta_{uv}d_{uv}$, and therefore, only this part of the demand can be lost if a single node or single edge of the supply graph fails. Hence, $(1 - \delta_{uv})d_{uv}$ channels "survive" without any rerouting effort. There are two drawbacks, however. First, setting the diversification parameter to δ_{uv} implies that the demand will be routed on at least $\lceil 1/\delta_{uv} \rceil$ paths. Second, we cannot achieve 100% survivability with this parameter. In practice, diversification values below 1/3 are undesirable, because this would force at least four paths each of them carrying only a small fraction of the demand.

Using RESERVATION the network designer takes advantage of possible redundancy in the network by allowing rerouting in failure situations. The advantage of this method is the design of low cost networks, but as obvious disadvantage, there is need for rerouting in case of a failure situation. In fact, this rerouting may be extensive and the network management is rather difficult for this survivability model. Furthermore, the available equipment and software cannot change the normal operating state routings appropriately fast to considerably different failure state routings. Hence, the cost of the solutions obtained using RESERVATION currently serve as a lower bound to the necessary capital investment to achieve a specified survivability in the network. As soon as such equipment will be available, however, it will be possible to practically take advantage of the low cost solutions using RESERVATION.

The best compromise between cost and maintenance effort can be obtained with the survivability model PATH RESTORATION. As we will see in Chapter 4, the solutions are comparable to the respective RESERVATION solutions, and the rerouting effort is relatively small. However, there is a drawback of this method in the network planning stage. Much more computation time is needed to achieve good solutions, since it is much more difficult to test whether a given set of capacities permits a feasible routing in all operating states. We describe this in more detail in Section 4.4.

Finally, the survivability model LINK RESTORATION (see (Wu, 1992)) is worth mentioning. This model is similar to PATH RESTORATION and can be employed to deal with single edge failures. In contrast to PATH RESTORATION, the failing flow is not rerouted end-to-end (between the end-nodes of the affected demands), but between the end-nodes of the failing edge. In other words, a failing edge generates a demand between its end-nodes. Obviously, capacities that suffice to accommodate routings with respect to these requirements also suffices to accommodate routings with respect to the requirements imposed by PATH RESTORATION. Thus, solutions for LINK RESTORATION tend to be more expensive. The practical advantage is, however, that faster restoration is possible since only communication between the end-nodes of the failing edge is necessary to establish a failure routing.

In practice, it is reasonable to combine DIVERSIFICATION as survivability model for the normal operating state with either RESERVATION or PATH RESTORATION. In these cases, a minimum survivability is achieved by the diversification parameter setting, with the advantage of easy network management. Additional survivability is introduced by the respective "failure" parameter setting. In case of a failure situation, the operator has to decide whether to reconfigure the network or not. This decision depends on various aspects, e.g., on the affected demands, the expected recovery time, and the required effort to reconfigure the network.

2.5 Valid model combinations

A particular instance of a survivable capacitated network design problem contains a combination of the capacity and survivability models presented in the previous sections. Figure 2.5 illustrates the valid combinations. Any instance contains either Discrete Capacities or Divisible Basic Capacities as capacity model, and the routing constraints for the normal operating state (Nos Routing in the figure). This is shown in the upper part of Figure 2.5. Optionally, the survivability model Diversification can be used to constraint the normal operating state routing and, in addition, one of the failure state survivability models Reservation or Path Restoration can be used to reroute flow in single component failures.

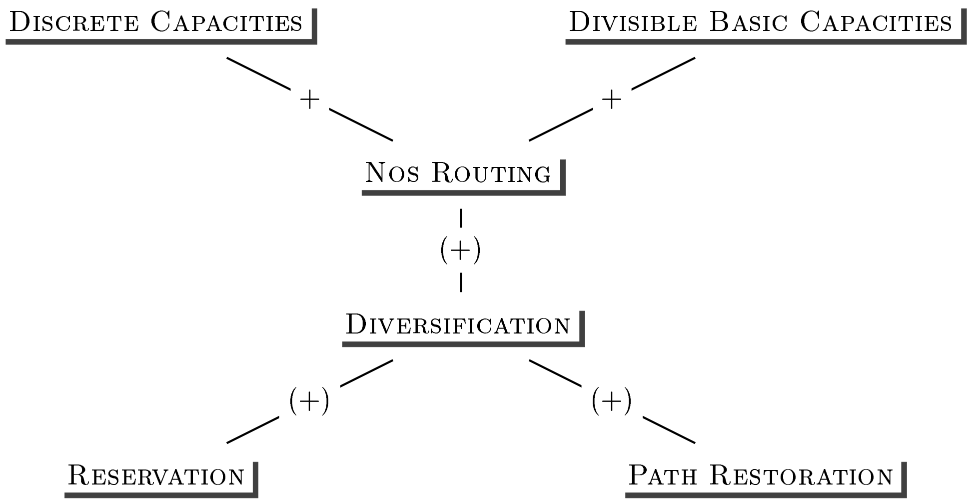


Figure 2.5: Valid combinations of capacity and survivability models

2.6 Survivable capacitated network design: A survey

In the following chapters we will investigate the network design models described in the previous sections. The research community considered, of course, alternative models that are closely related to ours. In this section, we give a brief overview on the research in the area of survivable capacitated network design, where the inherent routing problem can be modeled as multicommodity-flow problem. We omit other interesting and closely related research areas, since there already exist excellent surveys.

A very special case of network design problem is the Steiner-tree problem. Many researchers have been attracted by this problem and therefore the huge amount of existing practical and theoretical results in this area is not surprising. For a general overview of this topic, we refer the reader to the book (Wang et al., 1992), and for polyhedral methods to solve this problem exactly to the overview in (Koch and Martin, 1998). More general problems arise in the design of uncapacitated survivable networks which are interesting for fiber-optic cable networks if no routing and capacity issues are considered. Uncapacitated network design problems are often modeled by means of k-node (k-edge) connected networks and surveys on related optimization problems from a theoretical and practical perspective can be found in (Stoer, 1992) and (Grötschel et al., 1995).

The design problem for self-healing-ring networks fits into the category of capacitated survivable network design. In this context, however, the important subproblems are ring-covering and ring-loading problems (see (ITUT-G.841, 1995)), but not multicommodity-flow problems. The interring-routing problem (ITUT-G.842, 1997) could be considered as multicommodity-flow problem, but we are not aware of optimization models which integrate interring-routing, ring-covering and ring-loading.

2.6.1 Computational complexity

Very basic versions of capacitated network design problems are already $\mathcal{NP}$-complete. The following complexity results do not even take survivability constraints or path-length restrictions into account, and the capacity model is always DIVISIBLE BASIC CAPACITIES with $|T| \leq 2$.

We start with a polynomially solvable special case. (Magnanti and Mirchandani, 1993) showed for the single-demand case with one technology, that is, for $|D| = 1$ and $|T| = 1$, that the network design problem reduces to a shortest-path problem, if there are no routing costs. However, if routing costs are incorporated, even this single-demand case with one technology is theoretically difficult.

Theorem 2.2 (Chopra et al., 1998) *The capacitated network design problem for* DIVISIBLE BASIC CAPACITIES *with $|D| = 1$, $|T| = 1$ and routing costs is $\mathcal{NP}$-hard.*

Theorem 2.3 (Chopra et al., 1998) *The capacitated network design problem for* DIVISIBLE BASIC CAPACITIES *with $|D| = 1$ and $|T| \leq 2$ is $\mathcal{NP}$-hard even if the routing costs are always zero.*

Both preceding results have been proven through reduction from the minimal cover problem, see (Garey and Johnson, 1979).

In practice, economies of scale often apply for the basic capacities, that is, usually the inequality $K_e^{\tau_i} \cdot C^{\tau_j}/C^{\tau_i} \geq K_e^{\tau_j}$ is satisfied for all edges $e \in E$ and $\tau_i, \tau_j \in T$ with $C^{\tau_j} > C^{\tau_i}$. If this inequality holds at equality for all supply edges then a result of (Magnanti and Mirchandani, 1993) shows that the capacitated network design problem in the single-demand case reduces again to a shortest-path problem.

Turning to capacitated network design problems with more than just a single demand, we already know from the previously mentioned results that these optimization problems are $\mathcal{NP}$-hard. However, this already holds for *series-parallel* graphs. (A graph is series-parallel, if it can be constructed from a single edge by adding parallel edges and substituting edges by simple paths.) The following theorem has been proven through reduction from the knapsack problem.

Theorem 2.4 (Bienstock et al., 1998) *The capacitated network design problem for the model* DIVISIBLE BASIC CAPACITIES *with* $|T| = 1$ *is (weakly)* $\mathcal{NP}$*-hard on series-parallel supply graphs.*

2.6.2 Continuous capacities

In this section, we abstract from a particular discrete capacity model and assume the possibility to install arbitrary (continuous) capacities on the supply edges. Under this assumption the network design problem for the normal operating state without path-length restrictions is polynomially solvable. It can be considered as a continuous multicommodity-flow problem, which can be solved in polynomial time with either the ellipsoid method of (Khachyan, 1979) or the interior-point algorithm of (Karmakar, 1984), since it can be formulated as linear program with a polynomial number of variables and constraints. (For instance, the constraints (2.11) – (2.13) provide such a formulation.) Furthermore, there exists an exact characterization under which conditions a capacity vector suffices to accommodate a feasible (continuous) routing.

Theorem 2.5 (Iri, 1971), (Kakusho and Onaga, 1971) *A capacity vector* $\overline{y}$ *is feasible for the continuous capacitated network design problem if and only if*

$$\sum_{e \in E} \mu_e \overline{y}(e) \quad \geq \quad \sum_{uv \in D} \pi_{uv} d_{uv} \tag{2.22}$$

for all edge weights $\mu_e \geq 0, e \in E$*, where, for every* $uv \in D$*,* π_{uv} *is the value of a shortest* uv*-path in* G *with respect to these edge weights.*

The proof of this theorem is a simple application of the duality theorem of linear programming (see Theorem 0.1). Inequalities (2.22) are called *metric inequalities* and can be interpreted as follows. If the weight μ_e for a supply edge $e \in E$ defines the cost for one unit of capacity on this edge then the total network costs for capacities $\overline{y}(e)$, $e \in E$, are $\sum_{e \in E} \mu_e \overline{y}(e)$. The right-hand side of a metric inequality sums up the cost if we could route each demand $uv \in D$ on a cheapest path between u and v. Thus, this sum provides a lower bound to the cost of a feasible solution.

Some special cases of metric inequalities are important. Suppose that a k-graph-partition $V_1, \ldots, V_k$ of the supply graph $G = (V, E)$ is given (see page 10) and define supply edge weights by setting

$$\mu_e \quad := \quad \begin{cases} 1\,, & e \in \delta_G(V_1, \ldots, V_k)\,, \\ 0\,, & \text{else}\,. \end{cases}$$

Furthermore, let the subset of the demand edges with end-nodes in two different shores of the k-graph-partition be

$$\delta_H(V_1, \ldots, V_k) \quad := \quad \bigcup_{1 \leq i < j \leq k} \delta_H(V_i, V_j)\,.$$

The values of all shortest uv-paths are one, if $\delta_G(V_i, V_j) \neq \emptyset$ for every $uv \in \delta_H(V_1, \ldots, V_k)$ with $u \in V_i, v \in V_j$. In this case, the corresponding metric inequality reads as

$$\sum_{e \in \delta_G(V_1,\ldots,V_k)} y(e) \quad \geq \quad \sum_{uv \in \delta_H(V_1,\ldots,V_k)} d_{uv} \; . \tag{2.23}$$

For general k, (2.23) is a *k-graph-partition inequality*, and for $k = 2$ a *cut inequality*. Several researchers investigated the interesting question under which conditions a capacity vector is feasible if and only if all cut inequalities (or k-graph-partition inequalities, $k \leq l$, for some fixed $l \in \mathbb{N}$) are satisfied. Well-known results are the famous Max-Flow-Min-Cut Theorem and its extension to two demands.

Theorem 2.6 (Ford and Fulkerson, 1962) *For $|D| = 1$, a capacity vector $\overline{y}$ is feasible for the (continuous) capacitated network design problem if and only if $\overline{y}$ satisfies all cut inequalities.*

Theorem 2.7 (Hu, 1963) *For $|D| = 2$, a capacity vector $\overline{y}$ is feasible for the (continuous) capacitated network design problem if and only if $\overline{y}$ satisfies all cut inequalities are satisfied.*

In addition, there exists a huge number of characterizations for which structure of demand and supply graph a capacity vector provides a feasible routing if and only if all cut inequalities are satisfied. We do not further extend this list of results and refer to the excellent overviews (Frank, 1990; Frank, 1995) and (Schrijver, 1990) for a thorough presentation of related results.

In Section 2.3, we formulated the continuous capacitated network design problem with path variables and, as we already noted, there might be an exponential number of them. In general, it is therefore impossible to solve this linear program if it includes all path variables. (Minoux, 1981) suggested a column generation approach which is applicable to non-simultaneous multicommodity-flow problems such as given for the survivability models DIVERSIFICATION and RESERVATION. Minoux presented computational results with instances up to 40 nodes.

Large network design problem instances (e.g., more than 100 nodes) are difficult to solve by means of linear programming. However, for these cases, there is the interesting ϵ-approximation algorithm proposed by (Leighton et al., 1991). The running time of this algorithm scales up to a logarithmic factor linear with the number of demands. ((Leighton et al., 1991): "... giving the surprising result that approximately computing a $|D|$-commodity maximum flow is not much harder than computing about $|D|$ single commodity maximum flow problems.")

Theorem 2.8 (Leighton et al., 1991) *For any fixed $\epsilon > 0$, a $(1-\epsilon)$-approximation algorithm to the continuous capacitated network design problem for supply graph $G = (V, E)$ and demand graph $H = (V, D)$ can be found by a randomized algorithm in $\mathcal{O}(|D||E||V|log|D|log^3|V|)$ time and a deterministic algorithm in $\mathcal{O}(|D|^2|E||V|log|D|log^3|V|)$ time, where the constant depends on ϵ.*

This is a theoretical result. Several researchers, however, put emphasis on fast and practical implementations of this ϵ-approximation algorithm. First successful computational experiments on randomly generated data have been presented in (Leong et al., 1993). More recently, (Bienstock, 1999) solved this problem for a large-scale problem instance with more than 200 nodes. In fact, he employed the ϵ-approximation to compute solutions for a DIVISIBLE BASIC CAPACITIES with $T = \{\tau\}$ and $C^\tau = 1$. This problem size is not tractable otherwise.

2.6.3 Discrete capacities

In this section, we consider network design problems with a discrete capacity structure. As the complexity results reveal, these problems are much more difficult than their continuous counterparts. Even single-demand cases are $\mathcal{NP}$-hard, and hence it appears natural to devise heuristic algorithms. However, we do not know of any elaborate combinatorial heuristic with proven quality guarantee. The only "heuristics" we know are based on linear programming relaxations and are (partial) branch&bound or branch&cut algorithms. These are also the most successful approaches we are aware of.

Many researchers identified valid or even facet-defining inequalities for polyhedra associated with some version of a capacitated survivable network design problem. It is worth mentioning that in some sense, all known inequalities can be viewed as either an application of (mixed-)integer-rounding (see (Gomory, 1969; Chvátal, 1973; Nemhauser and Wolsey, 1990)) to an adaption of the k-graph-partition inequalities (2.23), or as an adaption of the minimal-cover inequalities for the knapsack polyhedron (see (Balas, 1975; Hammer et al., 1975; Padberg, 1975; Wolsey, 1975)). There are not many special cases of network design problems for which a complete description of the associated polyhedron is known. We cite two results about complete descriptions. In both cases, the supply graph contains not more than *three* nodes.

In the next chapter, we provide a coherent presentation of the polyhedra associated with capacitated network design problems. We will present several classes of inequalities, some of them are new, others are from the literature. Notice that most polyhedral investigations are concerned with the capacity model DIVISIBLE BASIC CAPACITIES with one or two basic capacities only, even though the capacity model DISCRETE CAPACITIES is more flexible (see Section 2.2). Only Dahl and Stoer investigated the latter model.

Models without survivability

As we already mentioned, not much is known about the structure of optimal solutions or the quality of heuristics even if one does not include survivability considerations. However, for a problem defined on three nodes there exists a complete description of the associated polyhedron.

Theorem 2.9 (Magnanti et al., 1993) *For a network design problem on a supply graph $G = (V, E)$ and demand graph $H = (V, D)$ with $|V| = 3$ for* DIVISIBLE BASIC CAPACITIES *with $|T| = 1$, the convex hull of all feasible integer capacity vectors is completely described by cut inequalities, 3-graph-partition inequalities and the non-negativity constraints.*

(Bienstock et al., 1998) proved a similar result for *directed* supply graphs on three nodes. In this case, so-called *total capacity inequalities* are needed to provide a complete description.

How about practically solving this type of problem? We briefly summarize the literature of computational experiments with linear programming based approaches; all of them for the model DIVISIBLE BASIC CAPACITIES with one or two basic capacities. (Bienstock and Günlük, 1995) solved sparse ATM network design problems with real-life data for instances of up to 16 nodes to optimality. Their model included flow costs and the capacities are given as combinations of the two basic capacities OC3 and OC12. They presented many classes of facet-defining inequalities, including different versions of cut inequalities, flow-cut-set inequalities, and 3-graph-partition inequalities.

(Magnanti et al., 1995) investigated the network design problem with two basic capacities. They compared a Lagrangian-relaxation approach with a cutting plane approach. For the cutting plane approach, (Magnanti et al., 1995) included cut inequalities, arc residual inequalities, and 3-graph-partition inequalities. The computational experiments on randomly generated problem instances with up to 15 nodes revealed that the cutting plane approach provides better lower bounds. The integrality gaps were about 5–20% for the 15 node problems and the integrality gap of the Lagrangian-relaxation approach was about 4% worse on average. However, their experiments indicated that the running time of the cutting plane approach is more sensitive to the problem size. With increasing problem size the time to solve the problem instances with Lagrangian-relaxation increased more slowly than those of the cutting plane approach. (Barahona, 1996) considered a linear relaxation for a network design problem with one basic capacity that includes only cut- and k-graph-partition inequalities. The flow variables were projected out, and instead of separating all metric inequalities only the mentioned subclasses were employed. No algorithm was mentioned to test the feasibility of a capacity vector (e.g., by means of separation of metric inequalities). Instead, Barahona described sophisticated separation algorithms for the two inequality classes and used these in a cut&branch algorithm. (In contrast to a branch&cut algorithm, a cut&branch algorithm uses cutting planes only at the root node and uses a pure branch&bound algorithm afterwards.) Problem instances up to 64 nodes with complete supply and demand graphs were solved with an accuracy of 5-10%. (Günlük, 1999) presented a branch&cut algorithm to solve the network design problem with two basic capacities. Beside the previously known classes of metric and k-graph-partition inequalities he proved that inequalities based on the principle of mixing mixed-integer inequalities (see (Günlük and Pochet, 1997)) are facet-defining for the associated polyhedron. Besides, a new branching rule was presented, the so-called knapsack branching. The computational tests on three real-world data sets ((i) 15 nodes, 22 supply edges, (ii) 16 nodes, 49 supply edges, (iii) 27 nodes, 51 supply edges) revealed the strength of branch&cut compared to cut&branch or branch&bound for this type of problem. (Bienstock et al., 1998) studied network design problems with one basic capacity. An interesting point in this paper is the computational comparison of two different formulations of the problem: one with edge-flow variables (see page 40), and one in terms of metric inequalities. The tests on two data sets, the "New York area" problem with 15 nodes, 44 supply edges, 210 demand edges, and "Norwegian" problems with 27 nodes, 102 supply edges, 19 demand edges (supplied by Mechthild Stoer) showed that the problems can be solved to optimality with both formulations. The formulation without flow-variables, however, outperformed the other in terms of branch&bound nodes and computation time.

Models with survivability

First, we consider the case of a single basic technology of unit capacity, that is, $T = \{\tau\}$ and $C^\tau = 1$. Let $G = (V, E)$ be the supply graph, $H = (V, D)$ the demand graph, and let $\delta_G(W), \delta_H(W)$ be cuts for $W \subseteq V$ in G and H, respectively. Furthermore, the value d represents the demand across the cut in the normal operating state, and for the supply edges $g \in \delta_G(W)$, the value d_g represents the demand across the cut in failure state g. The definition of these values depends on the survivability model for failure situations. For the model RESERVATION these are defined as

$$\begin{aligned} d &:= \sum_{uv \in \delta_H(W)} d_{uv} , \\ d_g &:= \sum_{uv \in \delta_H(W)} \rho_{uv} d_{uv} , \quad g \in \delta_H(W) . \end{aligned}$$

For the model PATH RESTORATION, suppose that a *fixed* normal operating state routing is given, Hence, a value $f_0^{uv}(e)$ is given for each supply edge $e \in \delta_H(W)$ and each demand edge $uv \in \delta_H(W)$, which represents the flow through e that is dedicated to the demand uv. In this setting, the definition of the above mentioned demand values is the following:

$$\begin{aligned} d &:= 0 , \\ d_g &:= \sum_{uv \in \delta_H(W)} \max\{0, (\sigma_{uv} - 1) d_{uv} + f_0^{uv}(g)\} , \quad g \in \delta_H(W) . \end{aligned}$$

Under these assumptions, the following constraints provide a general formulation of a survivable network design problem on a cut $W \subseteq V$ for the models RESERVATION and PATH RESTORATION.

$$\sum_{e \in \delta_G(W) \setminus \{g\}} x(e, \tau) \geq d_g , \quad g \in \delta_G(W) , \tag{2.24}$$

$$\sum_{e \in \delta_G(W)} x(e, \tau) \geq d , \tag{2.25}$$

$$x(e, \tau) \in \mathbb{Z}_+ , \quad e \in \delta_G(W) . \tag{2.26}$$

For the first case, RESERVATION, Bienstock and Muratore characterized inequalities through lifting (with simple lifting coefficients) such that the derived inequalities provide a complete description of the convex hull of all feasible solutions of (2.24)–(2.26), if all values $d_g, g \in \delta_G(W)$, are equal. In the other case, for PATH RESTORATION with fixed normal operating state routing, Magnanti and Wang provided a complete description.

Theorem 2.10 (Bienstock and Muratore, 1999) *For constant $d_g, g \in \delta_G(W)$, the convex hull of all solutions of the system* (2.24) – (2.26) *is completely described by inequalities* (2.24) *and inequalities of the form*

$$\sum_{e \in F_1} x(e, \tau) + \alpha \sum_{e \in F_2} x(e, \tau) \geq R , \tag{2.27}$$

for $F_1 \cup F_2 = \delta_G(W), F_1 \cap F_2 = \emptyset$ *and appropriate* $\alpha \geq 1$ *and* R*.*

Inequalities (2.27) can be obtained from sequential lifting applied to one particular (non-trivial) type of inequality. We do not want to go into further detail, but Bienstock and Muratore described the inequality to start the sequential lifting procedure with, *exactly* calculated the lifting coefficient, and proved that all facets can be obtained this way.

Theorem 2.11 (Magnanti and Wang, 1997) *For* $d = 0$*, the convex hull of all solutions of the system* (2.24) – (2.26) *is completely described by non-negativity inequalities, inequalities* (2.24) *and so-called* Q-subset *inequalities.*

Q-subset inequalities can be interpreted as the result of mixed-integer rounding applied to the sum of inequalities (2.24) for the edges in $Q \subset \delta_G(W)$. Magnanti and Wang also proved a generalization of this theorem. With slight modifications the same result is true for series-parallel graphs.

Next, we review some computational experiments for survivable capacitated network design with different combinations of models. Interestingly, integrated approaches which attempt to find a cheapest network such that a routing for the normal operating state *and* the failure states is possible are not often considered. Such integrated approaches are more complex and practically more difficult to solve. However, it has also been noted in (Murakami and Kim, 1995) and (Poppe and Demeester, 1997) that solutions of an integrated approach tend to be considerably cheaper.

(Stoer and Dahl, 1994; Dahl and Stoer, 1998) investigated the capacity model DISCRETE CAPACITIES and the survivability models DIVERSIFICATION and RESERVATION without path-length restrictions for the *Norwegian Telecom Research*. These models have been the basis for our work. Dahl and Stoer identified several classes of inequalities like band inequalities (see (3.8)) and k-graph-partition-band inequalities (see (3.10)), and solved a large number of instances (from 37 up to 118 nodes with a very sparse supply graph) to optimality. However, they also report on difficulties for problem instances with denser supply graph. Beside Dahl and Stoer and ourselves, there are only a few other references to integrated approaches, all for DIVISIBLE BASIC CAPACITIES and one basic capacity. (Poppe and Demeester, 1997) investigated the joint problem of installing capacities (one technology) such that a normal operating state routing *and* a routing for all single link failure is possible with respect to the survivability model LINK RESTORATION. Their mixed-integer programming model contains continuous path variables for the flow in the normal operating state and integer variables for the capacity decisions. They proposed a branch&cut approach with inequalities arising from the uncapacitated network design problems of (Grötschel et al., 1992) and sum-and-divide procedures. Nothing is proven about the strength of the inequalities. The problem sizes in the computational studies ranged from 8 nodes, 13 supply edges, 13 demand edges to 20 nodes, 54 supply edges, 79 demands. Optimality was proven for the small problem instances and the maximal integrality gap was 37%. For PATH RESTORATION, (Xiong, 1998) formulated the integrated optimization of the normal operating state and the failure states with path variables. The routing is supposed to be non-bifurcated, that is, for each demand and each operating state there is a unique path. Since no column generation for the path variables and no cutting planes to strengthen the linear relaxation were employed, the author preselected for each demand 10 alternative paths. The tests on two sparse problem instances (11 nodes, 23 supply edges and 28 nodes, 45 supply edges) yield gaps below 5%. (Filho and Tavares, 1998) considered the integrated problem for LINK RESTORATION. On two sparse test problems with large demands (compared to the basic capacity STM-4) the gap between the initial linear programming relaxation and the solution of a simple rounding heuristic was below 2%.

All of the following references are based on the capacity model DIVISIBLE BASIC CAPACITIES for one or two basic capacities, and the survivability model LINK RESTORATION or PATH RESTORATION with *fixed* normal operating state routing.

(Sakauchi et al., 1990) considered LINK RESTORATION. Their optimization target was to minimize the total number of spare capacity units (one basic capacity) needed to accommodate the failure routings. No costs were involved. As solution method they used a linear programming relaxation based on cut inequalities which were separated with a maximum flow algorithm. No extension to metric inequalities was shown for PATH RESTORATION. To compute feasible integral solutions the authors employed a rounding procedure to make the variables of the final linear programming solution integral, followed by an improvement algorithm to find a solution with less spare capacity. Nothing was reported about the criteria used and the quality guarantee obtained. (Lee et al., 1995) considered the capacity expansion for LINK RESTORATION. Combinations of two basic capacities (STM-1 and STM-4) are allowed and, in addition to other models, capacity constraints for the nodes of the supply graph have been added to the model. Cut inequalities are proven to be facet-defining for the associated polyhedra and these inequalities are then used in a branch&cut algorithm which was tested on two networks stemming from the literature. The problem sizes were 11 nodes, 23 supply edges, and 26 nodes, 42 supply edges. The number of demands was not reported. The variants with one basic capacity could be solved within a few seconds and those with two basic capacities within an hour. (Balakrishnan et al., 1998) used the polyhedral investigations of (Magnanti and Wang, 1997) for LINK RESTORATION, presented separation algorithms and reported on computational experiments with a cut&branch algorithm. The size of the real-world problem instance was 41 nodes, 61 supply edges (unknown demands) and the size of the random instance ranged from 20 up to 50 nodes with an average connectivity of 4, with random demands between any pair of nodes. The value of the initial linear programming relaxation was almost always below 10% and the cut&branch algorithm solved all problem instance within a few minutes to optimality.

Chapter 3

Polyhedral Investigations

The convex hull of all solutions that satisfy the constraints for a particular combination of a capacity and a survivability model is a polyhedron. In fact, these solutions provide an "inner" description of this polyhedron. By a well-known theorem in polyhedral combinatorics (Weyl, 1935), an "outer" description in terms of linear inequalities exists as well. Thus, if such an outer description is known, it is in principle possible to solve the problem by means of linear programming. However, in general it is not possible to provide such a description, and even if it is known, its number of inequalities is too large large for state-of-the-art linear programming solvers. Despite these problems, an optimal solution in terms of integer capacity variables can uniquely be described by a small number of linearly independent inequalities, and thus, it suffices to identify inequalities that determine such an optimal solution.

A cutting plane approach seeks a partial description which approximates the (complete) description as well as possible. The success of this approach depends on the extent to which the structure of the polyhedra is known, and therefore we focus in this chapter on the investigation of the polyhedra associated with the survivable capacitated network design described in Chapter 2. We present classes of valid or even facet-defining inequalities for the original polyhedra, for projections to subspaces and for particular relaxations.

The polyhedra depend on the structure of the supply and demand graphs, the capacity model, and the survivability model. Given a supply graph G and a demand graph H, a capacity model CAP and a survivability model SURV, we denote the polyhedron by

$$P(G, H, \textsc{cap}, \textsc{surv}) \ .$$

To keep the exposition simple, we ignore edge dependent parameters such as individual capacities of the supply edges, demand values or survivability parameters in the format. These are implicitly given.

The variables are the integer capacity variables x, the continuous path variables f, and the auxiliary continuous capacity variables y. We denote by FIN and BAS the capacity models DISCRETE CAPACITIES and DIVISIBLE BASIC CAPACITIES, respectively, and by DIV, RES, and PATH the survivability models DIVERSIFICATION, RESERVATION, and PATH RESTORATION, respectively. Additionally, we denote the normal operating state by NOS. We consider the following polyhedra in the subsequent sections.

$$\begin{aligned}
P(G,H,\textsc{Fin},\textsc{Nos}) &:= \text{conv}\,\{(x,y,f) : (x,y) \text{ satisfies } (2.2),(2.3),(2.4),\\
&\qquad (y,f) \text{ satisfies } (2.8),(2.9),(2.10)\},\\
P(G,H,\textsc{Fin},\textsc{Div}) &:= \text{conv}\,\{(x,y,f) : (x,y,f) \in P(G,H,\textsc{Fin},\textsc{Nos}),\\
&\qquad f \text{ satisfies } (2.14),(2.15)\},\\
P(G,H,\textsc{Fin},\textsc{Res}) &:= \text{conv}\,\{(x,y,f) : (x,y,f) \in P(G,H,\textsc{Fin},\textsc{Nos}),\\
&\qquad (y,f) \text{ satisfies } (2.16),(2.17),(2.18)\},\\
P(G,H,\textsc{Fin},\textsc{Path}) &:= \text{conv}\,\{(x,y,f) : (x,y,f) \in P(G,H,\textsc{Fin},\textsc{Nos}),\\
&\qquad (y,f) \text{ satisfies } (2.19),(2.20),(2.21)\},\\
P(G,H,\textsc{Bas},\textsc{Nos}) &:= \text{conv}\,\{(x,y,f) : (x,y) \text{ satisfies } (2.6),(2.7),\\
&\qquad (y,f) \text{ satisfies } (2.8),(2.9),(2.10)\},\\
P(G,H,\textsc{Bas},\textsc{Div}) &:= \text{conv}\,\{(x,y,f) : (x,y,f) \in P(G,H,\textsc{Bas},\textsc{Nos}),\\
&\qquad f \text{ satisfies } (2.14),(2.15)\},\\
P(G,H,\textsc{Bas},\textsc{Res}) &:= \text{conv}\,\{(x,y,f) : (x,y,f) \in P(G,H,\textsc{Bas},\textsc{Nos}),\\
&\qquad (y,f) \text{ satisfies } (2.16),(2.17),(2.18)\},\\
P(G,H,\textsc{Bas},\textsc{Path}) &:= \text{conv}\,\{(x,y,f) : (x,y,f) \in P(G,H,\textsc{Bas},\textsc{Nos}),\\
&\qquad (y,f) \text{ satisfies } (2.19),(2.20),(2.21)\}.
\end{aligned}$$

For notational convenience, we do not distinguish between the flow vector f and its projection to the normal operating state, and we omit the dimension of the vector space of the solutions (x,y,f), since it depends too much on the structure of a particular problem instance. Without enumeration of all valid paths for all operating states, it is difficult to determine the number of path variables. Obvious relations between these polyhedra are

$$P(G,H,\textsc{Fin},\textsc{Div}) \subseteq P(G,H,\textsc{Fin},\textsc{Nos}),\ P(G,H,\textsc{Bas},\textsc{Div}) \subseteq P(G,H,\textsc{Bas},\textsc{Nos}).$$

The above polyhedra are very high-dimensional, because of the huge number of path variables. As we already pointed out, the objective function coefficients of all path variables are zero (see (2.1) and (2.5)), and only cost coefficients of integer capacity variables are positive. In the space of the continuous capacity variables, we define the following polyhedra.

$$\begin{aligned}
Y(G,H,\textsc{Nos}) &:= \{y \in \mathbb{R}^E_+ : \exists f \text{ such that } (y,f) \text{ satisfies } (2.8),(2.9),(2.10)\},\\
Y(G,H,\textsc{Div}) &:= \{y \in \mathbb{R}^E_+ : \exists f \text{ such that } (y,f) \text{ satisfies } (2.8)-(2.10),(2.14),(2.15)\},\\
Y(G,H,\textsc{Res}) &:= \{y \in \mathbb{R}^E_+ : \exists f \text{ such that } (y,f) \text{ satisfies } (2.8)-(2.10),(2.16)-(2.18)\},\\
Y(G,H,\textsc{Path}) &:= \{y \in \mathbb{R}^E_+ : \exists f \text{ such that } (y,f) \text{ satisfies } (2.8)-(2.10),(2.19)-(2.21)\}.
\end{aligned}$$

For each survivability model, the above polyhedron is the sum of the positive orthant $\mathbb{R}^E_+$ and the respective projection to y-variables. Notice that these polyhedra are independent from the capacity model and defined by the set of feasible continuous capacity vectors. Obvious relations between these polyhedra are

$$Y(G, H, \textsc{Div}) \subseteq Y(G, H, \textsc{Nos}) \text{ and } Y(G, H, \textsc{Path}) \subseteq Y(G, H, \textsc{Res}).$$

Furthermore, we define polyhedra in the space of x-variables. Let $T(E) := \{(e,t) : e \in E, 1 \leq t \leq T_e\}$ be the index set of the integer capacity variables for the model DISCRETE CAPACITIES, and define

$$\begin{aligned}
X(G, H, \textsc{Fin}, \textsc{Nos}) &:= \text{conv}\ \{x \in \{0,1\}^{T(E)} : x(e,1) \geq \cdots \geq x(e,T_e), e \in E, \\
&\qquad (C_e^0 + \sum_{t=1}^{T_e} c_e^t x(e,t))_{e \in E} \in Y(G, H, \textsc{Nos})\}, \\
X(G, H, \textsc{Fin}, \textsc{Div}) &:= \text{conv}\ \{x \in \{0,1\}^{T(E)} : x(e,1) \geq \cdots \geq x(e,T_e), e \in E, \\
&\qquad (C_e^0 + \sum_{t=1}^{T_e} c_e^t x(e,t))_{e \in E} \in Y(G, H, \textsc{Div})\}, \\
X(G, H, \textsc{Fin}, \textsc{Res}) &:= \text{conv}\ \{x \in \{0,1\}^{T(E)} : x(e,1) \geq \cdots \geq x(e,T_e), e \in E, \\
&\qquad (C_e^0 + \sum_{t=1}^{T_e} c_e^t x(e,t))_{e \in E} \in Y(G, H, \textsc{Res})\}, \\
X(G, H, \textsc{Fin}, \textsc{Path}) &:= \text{conv}\ \{x \in \{0,1\}^{T(E)} : x(e,1) \geq \cdots \geq x(e,T_e), e \in E, \\
&\qquad (C_e^0 + \sum_{t=1}^{T_e} c_e^t x(e,t))_{e \in E} \in Y(G, H, \textsc{Path})\}.
\end{aligned}$$

Similarly, let $T(E) := \{(e,\tau) : e \in E, \tau \in T\}$ for the DIVISIBLE BASIC CAPACITIES and define

$$\begin{aligned}
X(G, H, \textsc{Bas}, \textsc{Nos}) &:= \text{conv}\ \{x \in \mathbb{Z}_+^{T(E)} : (C_e^0 + \sum_{\tau \in T} C^\tau x(e,\tau))_{e \in E} \in Y(G, H, \textsc{Nos})\}, \\
X(G, H, \textsc{Bas}, \textsc{Div}) &:= \text{conv}\ \{x \in \mathbb{Z}_+^{T(E)} : (C_e^0 + \sum_{\tau \in T} C^\tau x(e,\tau))_{e \in E} \in Y(G, H, \textsc{Div})\}, \\
X(G, H, \textsc{Bas}, \textsc{Res}) &:= \text{conv}\ \{x \in \mathbb{Z}_+^{T(E)} : (C_e^0 + \sum_{\tau \in T} C^\tau x(e,\tau))_{e \in E} \in Y(G, H, \textsc{Res})\}, \\
X(G, H, \textsc{Bas}, \textsc{Path}) &:= \text{conv}\ \{x \in \mathbb{Z}_+^{T(E)} : (C_e^0 + \sum_{\tau \in T} C^\tau x(e,\tau))_{e \in E} \in Y(G, H, \textsc{Path})\}.
\end{aligned}$$

For each combination of a capacity and a survivability model these are the projections to x-variables. Again, the obvious inclusions are

$$X(G, H, \textsc{Fin}, \textsc{Div}) \subseteq X(G, H, \textsc{Fin}, \textsc{Nos}), X(G, H, \textsc{Fin}, \textsc{Path}) \subseteq X(G, H, \textsc{Fin}, \textsc{Res}),$$

$$X(G, H, \textsc{Bas}, \textsc{Div}) \subseteq X(G, H, \textsc{Bas}, \textsc{Nos}), X(G, H, \textsc{Bas}, \textsc{Path}) \subseteq X(G, H, \textsc{Bas}, \textsc{Res}).$$

If the particular survivability model is not important, we use the notation

$$P(G, H, \textsc{Fin}, \cdot), P(G, H, \textsc{Bas}, \cdot), \quad X(G, H, \textsc{Fin}, \cdot), X(G, H, \textsc{Bas}, \cdot), \quad Y(G, H, \cdot),$$

respectively. If the particular capacity model is also not important, we use

$$P(G, H, \cdot, \cdot), \quad X(G, H, \cdot, \cdot),$$

respectively. These three types of polyhedra are related with each other. The polyhedra $X(G, H, \cdot, \cdot)$ are the projection of the respective polyhedra $P(G, H, \cdot, \cdot)$ to the space of integer capacity variables. Furthermore, there is a canonic way to embed the polyhedra $X(G, H, \cdot, \cdot)$ and $P(G, H, \cdot, \cdot)$ into the (continuous) polyhedra $Y(G, H, \cdot)$. This implies the following lemma, the proof of which is obvious.

Lemma 3.1

(a) $\min\{k^T x : (x, y, f) \in P(G, H, \cdot, \cdot)\} = \min\{k^T x : x \in X(G, H, \cdot, \cdot)\}$,

(b) $a^T x \geq \alpha$ *valid for* $P(G, H, \cdot, \cdot) \iff a^T x \geq \alpha$ *valid for* $X(G, H, \cdot, \cdot)$,

(c) $a^T y \geq \alpha$ *valid for* $Y(G, H, \cdot) \implies$
$\sum_{e \in E} a_e \sum_{t=1}^{T_e} c_e^t x(e, t) \geq \alpha - \sum_{e \in E} a_e C_e^0$ *valid for* $X(G, H, \text{FIN}, \cdot)$ *and* $P(G, H, \text{FIN}, \cdot)$,

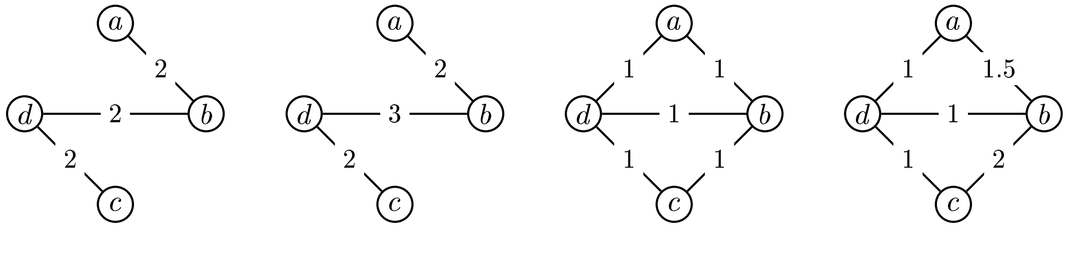
(d) $a^T y \geq \alpha$ *valid for* $Y(G, H, \cdot) \implies$
$\sum_{e \in E} a_e \sum_{\tau \in T} C^\tau x(e, \tau) \geq \alpha - \sum_{e \in E} a_e C_e^0$ *valid for* $X(G, H, \text{BAS}, \cdot)$ *and* $P(G, H, \text{BAS}, \cdot)$.

Notice that in statements (c) and (d) of Lemma 3.1 a valid inequality for a polyhedron $Y(G, H, \cdot)$ is transformed into a valid inequality for the respective capacity model dependent polyhedra $X(G, H, \text{FIN}, \cdot)$, $X(G, H, \text{BAS}, \cdot)$, $P(G, H, \text{FIN}, \cdot)$, and $P(G, H, \text{BAS}, \cdot)$.

3.1 Continuous capacities: $Y(G, H, \cdot)$

In this section, we investigate the polyhedra $Y(G, H, \cdot)$ for the different survivability models. We present in each case a complete description of $Y(G, H, \cdot)$ by means of some variation of so-called metric inequalities. The original results of (Iri, 1971) and (Kakusho and Onaga, 1971) state that metric inequalities suffice to describe the polyhedron $Y(G, H, \text{NOS})$. The proofs are based on linear programming duality applied to the appropriate formulation of the decision problem whether some capacity vector $\overline{y}$ satisfies $\overline{y} \in Y(G, H, \text{NOS})$. This result can be adapted for the survivability models DIVERSIFICATION, RESERVATION, and PATH RESTORATION.

At the end of this section, we illustrate the difference between the survivability model dependent classes of metric inequalities based on the following small examples. Let a complete supply graph on four nodes a, b, c, and d be given, and suppose that there are two demands ac and bd with values $d_{ac} = 2$ and $d_{bd} = 1$. Furthermore, let the survivability dependent parameters be $\delta_{ac} = \delta_{bd} = 0.5$, $\rho_{ac} = \rho_{bd} = 0.5$, and $\sigma_{ac} = \sigma_{bd} = 0.5$ for DIVERSIFICATION, RESERVATION, and PATH RESTORATION, respectively. In this example, no path-length restrictions are considered. The four capacity vectors (the labels at the edges represent the capacity) shown in Figure 3.1 satisfy



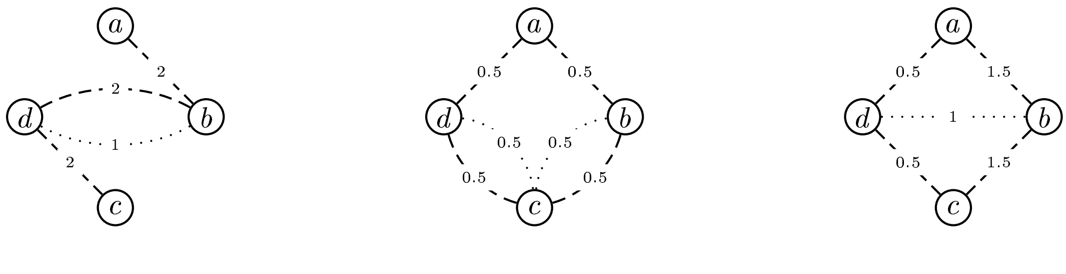
(a) (b) (c) (d)

Figure 3.1: Example capacities

3.1(a)	is infeasible for $Y(G, H, \text{NOS})$,
3.1(b)	is feasible for $Y(G, H, \text{NOS})$, but infeasible for $Y(G, H, \text{RES})$,
3.1(c)	is feasible for $Y(G, H, \text{RES})$, but infeasible for $Y(G, H, \text{PATH})$,
3.1(d)	is feasible for $Y(G, H, \text{PATH})$, but infeasible for $Y(G, H, \text{DIV})$.

To verify the claimed feasibility for Figure 3.1(b)-(d), see Figure

- 3.2(a) for a feasible routing for the normal operating state and the capacities of Figure 3.1(b).
- 3.2(b) for a feasible routing for RESERVATION in edge failure state $s = bd$ for the capacities of Figure 3.1(c). The routings in all other operating states are obvious.
- 3.2(c) for a feasible routing in the normal operating state for the capacities of Figure 3.1(d). For PATH RESTORATION, this routing is feasible for the failure states $s \in \{a, c, d, ad, ac, cd\}$ without any changes. For the edge failure states $s = ab$ and $s = bc$, and the node failure state $s = b$, a feasible routing is obtained by rerouting 0.5 units of demand ac over the path $a - d - c$. Finally, in the edge failure state $s = bd$, a feasible routing is obtained by rerouting 0.5 units of demand bd over the path $b - c - d$.



(a) (b) (c)

Figure 3.2: Example routings

At the end of this section, we will prove the claimed infeasibilities of the capacity vectors of Figures 3.1(a)-(d). For each of the four cases, we present a metric inequality which is valid for the particular polyhedron $Y(G, H, \cdot)$, but which is violated by the particular capacity vector.

3.1.1 No survivability restrictions

The polyhedron $Y(G,H,\text{NOS})$ consists of all continuous capacity vectors $\overline{y} \in \mathbb{R}_+^E$ which suffice to accommodate a continuous multicommodity-flow. Theorem 2.5 states that $\overline{y} \in Y(G,H,\text{NOS})$ if and only if all metric inequalities (2.22) are satisfied. This result is formulated without path-length restrictions, but can easily be extended to cover this case.

3.1.2 DIVERSIFICATION

Next, we consider the polyhedron $Y(G,H,\text{DIV})$ for the survivability model DIVERSIFICATION. In this case, a capacity vector $\overline{y}$ is *feasible*, if there exists a routing vector $\overline{f}$ such that $(\overline{y},\overline{f})$ satisfies the constraints (2.14) and (2.15), in addition to the normal operating state constraints. The following theorem is an extension of Theorem 2.5 that has been proven in (Dahl and Stoer, 1998), rephrased here to include path-length restrictions.

Theorem 3.2 *A capacity vector $\overline{y}$ is feasible for* DIVERSIFICATION with *path-length restrictions, that is, $\overline{y} \in Y(G,H,\text{DIV})$, if and only if*

$$\sum_{e\in E} \overline{y}(e)\mu_e \geq \sum_{uv\in D} d_{uv}\pi_{uv} - \sum_{uv\in D} \delta_{uv} d_{uv} \left(\sum_{e\in\delta_G(u)\cap\delta_G(v)} \gamma_{uv}^e + \sum_{w\in V\setminus\{u,v\}} \gamma_{uv}^w \right) , \tag{3.1}$$

for all $\mu_e \geq 0, e \in E$, $\gamma_{uv}^w \geq 0, uv \in D, w \in V\setminus\{u,v\}$, and $\gamma_{uv}^e \geq 0$, $uv \in D, e \in \delta_G(u)\cap\delta_G(v)$ with π_{uv} is defined as follows: Given $uv \in D$, we assign to each edge $e \in E\setminus\{\delta_G(u)\cap\delta_G(v)\}$ the weight μ_e, to edges $e \in \delta_G(u)\cap\delta_G(v)$ the weight $\mu_e + \gamma_{uv}^e$, and to each node $w \in V\setminus\{u,v\}$ the weight γ_{uv}^w. Then π_{uv} is the value of a shortest among all uv-paths in G with at most ℓ_{uv} edges.

We refer to inequalities (3.1) as metric inequalities, too, since these are the obvious extension for the survivability model DIVERSIFICATION.

3.1.3 RESERVATION

Next, we consider the polyhedron $Y(G,H,\text{RES})$ for the survivability model RESERVATION. In this case, a capacity vector $\overline{y}$ is *feasible*, if there exists a routing vector $\overline{f}$ such that $(\overline{y},\overline{f})$ satisfies the constraints (2.16), (2.17), and (2.18), in addition to the constraints for the normal operating state. These constraints nicely decompose into a separate set of constraints for each operating state. Hence, the extension of the previous results to $Y(G,H,\text{RES})$ is the following.

Proposition 3.3 *A capacity vector $\overline{y}$ is feasible for the survivability model* RESERVATION, *that is, $\overline{y} \in Y(G,H,\text{RES})$, if and only if $\overline{y} \in Y(G,H,\text{NOS})$ and for all failure states $s \in S\setminus\{0\}$*

$$\sum_{e\in E_s} \overline{y}(e)\mu_e^s \geq \sum_{uv\in D_s} \rho_{uv} d_{uv} \pi_{uv}^s , \tag{3.2}$$

for all $\mu_e^s \geq 0, e \in E_s$, where π_{uv}^s is the value of a shortest uv-path in G_s with respect to the edge weights μ_e^s for every $uv \in D_s$.

We refer to inequalities (3.2) as metric inequalities, too, since these are the obvious extension for the survivability model RESERVATION.

3.1.4 PATH RESTORATION

Finally, we consider the polyhedron $Y(G, H, \text{PATH})$ for the survivability model PATH RESTORATION. In this case, a capacity vector $\overline{y}$ is *feasible*, if there exists a routing vector $\overline{f}$ such that $(\overline{y}, \overline{f})$ satisfies the constraints (2.19) and (2.20), in addition to the constraints for the normal operating state.

The previously introduced metric inequalities (2.22), (3.1), and (3.2), are of the following form. Given are supply edge weights (for individual operating states), which define the coefficients for the continuous capacity variables. The value of the right-hand side is then defined as the sum of weighted demand values, where the weights are the values of shortest paths between the demand end-nodes. This construction is not possible for PATH RESTORATION, but the set of feasible solutions can also be described by a single class of inequalities. In this case, the coefficients reflect that the routing for the normal operating state is linked with the routings of the failure states.

Proposition 3.4 *A capacity vector $\overline{y}$ is feasible for the survivability model* PATH RESTORATION, *that is, $\overline{y} \in Y(G, H, \text{PATH})$, if and only if*

$$\sum_{s \in S} \sum_{e \in E_s} \overline{y}(e)\, \mu_e^s \;\; \geq \;\; \sum_{uv \in D} d_{uv} \pi_{uv}^0 + \sum_{s \in S \setminus \{0\}} \sum_{uv \in D_s} \sigma_{uv} d_{uv} \pi_{uv}^s \tag{3.3}$$

is satisfied, for all $\mu_e^s \geq 0$, $s \in S$, $e \in E_s$, and π_{uv}, $\pi_{uv}^s \geq 0, s \in S\setminus\{0\}$, $uv \in D_s$ that satisfy

$$\pi_{uv}^s - \sum_{e \in P} \mu_e^s \;\; \leq \;\; 0\,, \; s \in S\setminus\{0\},\; uv \in D_s,\; P \in \mathcal{P}_s^{uv}\,, \tag{3.4}$$

$$\pi_{uv}^0 - \sum_{e \in P} \mu_e^0 + \sum_{s \in S\setminus\{0\}: s \notin P} \left(\pi_{uv}^s - \sum_{e \in P} \mu_e^s \right) \;\; \leq \;\; 0\,, \; uv \in D,\; P \in \mathcal{P}_0^{uv}\,. \tag{3.5}$$

Proof. We just sketch the proof, which is a simple application of linear programming duality. Suppose that the decision problem whether a capacity vector is feasible for PATH RESTORATION is formulated as linear program with path variables such that the additional capacity needed on a single supply edge is minimized. See (4.14) – (4.20) on page 110 for such a formulation including DIVERSIFICATION. Then, the constraints (3.4) and (3.5) are the non-trivial constraints of the associated dual linear program, and (3.3) corresponds to the dual objective function. The result follows, since (3.3) is violated if and only if the optimal dual objective is strictly positive, that is, if and only if additional capacity is needed on at least one supply edge. □

In the small examples at the beginning of this section, we claimed infeasibility for the capacity vectors in Figure 3.1(a),(b),(c), and (d), for the normal operating state and the survivability models Reservation, Path Restoration, and Diversification, respectively. For each example, we prove the infeasibility by presenting a metric inequality that is valid for the particular polyhedron $Y(G,H,\cdot)$, but which is violated by the capacity vector.

- Let $\mu_{ac} = \mu_{ad} = \mu_{bc} = \mu_{bd} = 1$ and $\mu_e = 0$, otherwise. The values of shortest paths between the demand end-nodes are $\pi_{ac} = 1$ and $\pi_{bd} = 1$, and hence, the capacity vector in Figure 3.1(a) is not feasible for normal operating state, since the metric inequality

$$y(ac) + y(ad) + y(bc) + y(bd) \quad \geq \quad 1 \cdot 2 + 1 \cdot 1 \;=\; 3$$

  is valid for $Y(G,H,\textsc{Nos})$. However, this inequality is violated for the capacity vector in Figure 3.1(a). The left-hand side evaluates to 2.

- For the node failure state $s = b$, let $\mu^b_{ac} = \mu^b_{ad} = 1$. Otherwise, let $\mu^s_e = 0$. For these supply edge weights, the value of a shortest ac-path in G_b is $\pi^b_{ac} = 1$. The capacity vector in Figure 3.1(b) is not feasible for Reservation since the cut inequality

$$y(ac) + y(ad) \quad \geq \quad 0.5 \cdot 2 \cdot 1 \;=\; 1$$

  is valid $Y(G,H,\textsc{Res})$, but violated for this capacity vector. The left-hand side evaluates to 0.

- For the next case, let

$$\mu^s_e := \begin{cases} 4\,, & s = 0\,, \quad e = ac\,, \\ 2\,, & s = 0\,, \quad e = bd\,, \\ 1\,, & s = 0\,, \quad e \neq ac,\ bd\,, \\ 1\,, & s = bd\,, \quad e = ab,\ ad,\ bc,\ cd \\ 0\,, & \text{else}\,. \end{cases} \qquad \text{and} \qquad \pi^s_{uv} := \begin{cases} 4\,, & s = 0\,, \quad uv = ac\,, \\ 2\,, & s = 0\,, \quad uv = bd\,, \\ 2\,, & s = bd\,, \quad uv = bd\,, \\ 0\,, & \text{else}\,. \end{cases}$$

  It is straightforward to verify that these values satisfy the constraints (3.4) and (3.5) of Proposition 3.4 and thus the inequality

$$\begin{aligned} 2y(bd) + 4y(ac) + (1+1)(y(ab) + y(ad) + y(bc) + y(cd)) \quad &\geq \quad 4 \cdot 2 + 2 \cdot 1 + 2 \cdot 0.5 \cdot 1 \\ &= \quad 11 \end{aligned}$$

  is valid for $Y(G,H,\textsc{Path})$. However, this inequality is violated for the capacity vector in Figure 3.1(c). The left-hand side evaluates to 10.

- Finally, let $\mu_{ad} = \mu_{cd} = 1, \mu_{ac} = 2$, and $\mu_e = 0$, otherwise. Furthermore, let $\gamma^{bd}_{bd} = 1$, $\gamma^b_{ac} = 2$, and $\gamma^{ac}_{ac} = \gamma^d_{ac} = \gamma^a_{bd} = \gamma^c_{bd} = 0$. For these edge and node weights, the values of shortest paths between the demand end-nodes are $\pi_{bd} = 1$ and $\pi_{ac} = 2$. Hence, the capacity vector in Figure 3.1(d) is infeasible for the survivability model Diversification since the inequality

$$y(ad) + y(cd) + 2y(ac) \quad \geq \quad 2 \cdot 2 + 1 \cdot 1 - 2 \cdot 0.5 \cdot 2 - 1 \cdot 0.5 \cdot 1 \;=\; 2.5$$

  is valid for $Y(G,H,\textsc{Div})$, but is violated for the capacity vector in Figure 3.1(d). The left-hand side evaluates to 2.

3.2 Discrete Capacities: $X(G, H, \text{Fin}, \cdot)$

For each survivability model, we investigate in this section the projection $X(G, H, \text{Fin}, \cdot)$ of the respective polyhedron $P(G, H, \text{Fin}, \cdot)$ to the space of integer capacity variables. We present classes of valid and facet-defining inequalities for $X(G, H, \text{Fin}, \cdot)$. As we already pointed out in Lemma 3.1, there is a canonical way to derive a valid inequality for $X(G, H, \text{Fin}, \cdot)$ from a valid inequality for $Y(G, H, \cdot)$, and obviously, each of these inequalities induces the following relaxation of $X(G, H, \text{Fin}, \cdot)$.

Definition 3.5 Let $\mu \in \mathbb{R}_+^E$, $\sum_{e \in E} \mu_e y(e) \geq d$ be a valid inequality for $Y(G, H, \cdot)$ and set $F := \text{supp}(\mu)$. Then we can define the *induced knapsack-relaxation* for Discrete Capacities as

$$Q_{\text{Fin}}(\mu, d) := \text{conv}\{x \in \{0,1\}^{T(F)} : \begin{array}{l} \sum_{e \in F} \mu_e \sum_{t=1}^{T_e} c_e^t x(e,t) \geq d - \sum_{e \in F} \mu_e C_e^0, \\ 1 \geq x(e,1) \geq \cdots \geq x(e, T_e) \geq 0, \; e \in F\} . \end{array}$$

Notice that we assume an implicitly given valid inequality $\sum_{e \in E} \mu_e y(e) \geq d$ for $Y(G, H, \cdot)$, whenever we write $Q_{\text{Fin}}(\mu, d)$. Often, we focus on such a relaxation, derive classes of valid inequalities for it and attempt to prove that these are facet-defining for this relaxation. Setting all coefficients in $T(E \backslash F)$ to zero, valid inequalities for this relaxation can be extended to valid inequalities for $X(G, H, \text{Fin}, \cdot)$. Under additional conditions, the resulting inequality is even facet-defining for $X(G, H, \text{Fin}, \cdot)$. Table 3.1 shows the classes of valid inequalities presented in the sequel together with the reference of its first publication.

Model	Inequality class	Reference
Nos	Strengthened metric inequalities	(Alevras et al., 1996)
	Band inequalities	(Dahl and Stoer, 1998)
	2-Band inequalities	Section 3.2.1
	k-graph-partition band-inequalities	(Stoer and Dahl, 1994)
Diversification	Diversification-band inequalities	Section 3.2.2
Reservation	Strengthened band inequalities	(Dahl and Stoer, 1998)
	Strengthened 2-band inequalities	Section 3.2.3
Path Restoration	see Reservation	

Table 3.1: Classes of non-trivial valid inequalities for $X(G, H, \text{Fin}, \cdot)$

When looking at the classes of valid inequalities presented Table 3.1 it is important to recall from Lemma 3.1 that valid inequalities for $X(G, H, \text{Fin}, \text{Nos})$ are valid for the survivability model dependent polyhedra $X(G, H, \text{Fin}, \text{Div})$, $X(G, H, \text{Fin}, \text{Res})$, and $X(G, H, \text{Fin}, \text{Path})$, respectively, and valid inequalities for $X(G, H, \text{Fin}, \text{Res})$ are valid for $X(G, H, \text{Fin}, \text{Path})$. For the survivability model Path Restoration, no other classes of inequalities than those for Reservation are presented, since it turned out to be difficult to identify new classes reflecting that the normal operating state routings must be preserved in failure situations. In the following, we assume that $C_e^{T_e}$ is for each $e \in E$ appropriately large. Furthermore, the incidence vector $\chi^{g,k} \in \{0,1\}^{T(E)}$ for $(g,k) \in T(E)$ is defined by $\chi^{g,k}(e,t) = 1$, $(e,t) \in T(E)$, if and only if $e = g$ and $k = t$.

Strengthened metric inequalities

The first class of inequalities is the result of a divide-and-round procedure applied to metric inequalities (2.22), (3.1) and (3.2). These inequalities are called strengthened metric inequalities and they are valid for the respective polyhedron $X(G,H,\cdot,\cdot)$. However, no general conditions are known under which they are facet-defining.

Proposition 3.6 *Let $\mu \in \mathbb{R}_+^E$, $F := \text{supp}(\mu)$ and $\sum_{e\in F} \mu_e y(e) \geq d$ be valid for $Y(G,H,\cdot)$ with $\mu_e c_e^t \in \mathbb{N}$ for all $e \in F$, $1 \leq t \leq T_e$. Furthermore, set*

$$\begin{aligned}
\overline{d} &:= d - \sum_{e\in F} \mu_e C_e^0 \ , \\
t_e &:= \max\{t : 0 \leq t \leq T_e, \mu_e(C_e^t - C_e^0) < \overline{d}\}, \ \textit{for all } e \in E \ , \\
g &:= gcd\{\mu_e c_e^t : e \in F, 1 \leq t \leq t_e\} \ .
\end{aligned}$$

*Then the **strengthened metric inequality***

$$\sum_{e\in F} \left(\left\lceil \frac{\overline{d} - \mu_e(C_e^{t_e} - C_e^0)}{g} \right\rceil x(e, t_e+1) + \sum_{t=1}^{t_e} \frac{\mu_e c_e^t}{g} \, x(e,t) \right) \quad \geq \quad \left\lceil \frac{\overline{d}}{g} \right\rceil \tag{3.6}$$

is valid for $X(G,H,\text{Fin},\cdot)$.

Notice that for every supply edge $e \in E$ the breakpoint t_e is largest, such that its weighted capacity $\mu_e C_e^t$ does not suffice to satisfy the underlying inequality for $Y(G,H,\cdot)$.

Proof. Let $\overline{x} \in \{0,1\}^{T(E)} \cap X(G,H,\text{Fin},\cdot)$. We distinguish between two cases.

- If there exists a supply edge $e \in F$ with $\overline{x}(e, t_e+1) = 1$, then inequality (3.6) is satisfied because

$$\begin{aligned}
& \sum_{t=1}^{t_e} \frac{\mu_e c_e^t}{g} \overline{x}(e,t) + \left\lceil \frac{\overline{d} - \mu_e(C_e^{t_e} - C_e^0)}{g} \right\rceil \overline{x}(e, t_e+1) \\
= \ & \frac{\mu_e}{g} \sum_{t=1}^{t_e} c_e^t + \left\lceil \frac{\overline{d} - \mu_e(C_e^{t_e} - C_e^0)}{g} \right\rceil \\
= \ & \left\lceil \frac{\mu_e}{g} \left(\sum_{t=1}^{t_e} c_e^t - \left(C_e^{t_e} - C_e^0\right) \right) + \frac{\overline{d}}{g} \right\rceil \quad = \quad \left\lceil \frac{\overline{d}}{g} \right\rceil \ .
\end{aligned}$$

- Otherwise, if $\overline{x}(e, t_e + 1) = 0$ for all $e \in F$, we define $\overline{y}(e) := C_e^0 + \sum_{t=1}^{t_e} c_e^t \overline{x}(e, t)$ for every $e \in F$, and $\overline{y}(e) := C_e^{T_e}$ for every $e \in E \backslash F$. Clearly, $\overline{y} \in Y(G, H, \cdot)$, and

$$\begin{aligned}
& \sum_{e \in F} \left(\left\lceil \frac{\overline{d} - \mu_e (C_e^{t_e} - C_e^0)}{g} \right\rceil \overline{x}(e, t_e + 1) + \sum_{t=1}^{t_e} \frac{\mu_e c_e^t}{g} \, \overline{x}(e, t) \right) \\
= \; & \sum_{e \in F} \sum_{t=1}^{t_e} \frac{\mu_e c_e^t}{g} \overline{x}(e, t) \quad = \quad \frac{1}{g} \sum_{e \in F} \mu_e (\overline{y}(e) - C_e^0) \\
\geq \; & \frac{1}{g} \left(d - \sum_{e \in F} \mu_e C_e^0 \right) \quad = \quad \frac{\overline{d}}{g} \, .
\end{aligned}$$

Now, the validity of inequality (3.6) follows from with the integrality of $\overline{x}$. □

If the underlying inequality is a *k-graph-partition* or a *cut inequality*, we refer to the associated inequality (3.6) as a *strengthened k-graph-partition* or a *strengthened cut inequality*.

It is difficult to provide sufficient conditions such that an inequality (3.6) is facet-defining for $X(G, H, \text{FIN}, \cdot)$, even if the underlying inequality is a *cut inequality*. However, by definition there exist feasible solutions in the face induced by a strengthened cut inequality if the two shores of the cut are connected, and therefore, strengthened cut inequalities induce non-trivial faces of $X(G, H, \text{FIN}, \cdot)$ under these conditions.

From a computational point of view, it is interesting to note that strengthened metric inequalities have dense support. Often almost every integer capacity variable $x(e, t)$ of the supply edges e in the support F of the underlying valid inequality for $Y(G, H, \cdot)$ appears in the strengthened metric inequality. As a consequence, we observed stronger relaxations and numerical instabilities whenever we employed these inequalities in the cutting plane algorithm.

3.2.1 No survivability restrictions

We start the investigation of $X(G, H, \text{FIN}, \text{NOS})$ with its dimension and the property that the ordering constraints are indeed facet-defining. Then we derive several classes of inequalities valid for $X(G, H, \text{FIN}, \text{NOS})$ which are based on valid inequalities for $Y(G, H, \text{NOS})$. The class of *band inequalities* is similar to minimal cover inequalities for the knapsack problem (see, e.g., (Padberg, 1975)) and was first presented in (Stoer and Dahl, 1994) for network design problems based on the capacity model DISCRETE CAPACITIES. It was also proven in this paper that band inequalities are under rather natural conditions facet-defining for $Q_{\text{FIN}}(\mu, d)$. In addition, we present sufficient conditions for band inequalities to be facet-defining for $X(G, H, \text{FIN}, \text{NOS})$, and introduce *2-band inequalities* and *k-graph-partition band-inequalities* as generalizations of band inequalities.

The following basic properties of $X(G, H, \text{FIN}, \text{NOS})$ have been proven in (Stoer and Dahl, 1994) for the case $C_e^0 = 0$ for all $e \in E$.

Proposition 3.7 *The polytope $X(G,H,\textsc{Fin},\textsc{Nos})$ is full-dimensional if and only if for all $e \in E$ the polytope*

$$\{x \in X(G,H,\textsc{Fin},\textsc{Nos}) : x(e,1) = 0\}$$

is non-empty.

Proposition 3.8
- *For every $e \in E$, the inequalities $1 \geq x(e,1) \geq \cdots \geq x(e,T_e)$ are facet-defining for $X(G,H,\textsc{Fin},\textsc{Nos})$.*
- *For every $e \in E$, the inequality $x(e,T_e) \geq 0$ is facet-defining for $X(G,H,\textsc{Fin},\textsc{Nos})$ if and only if the polytope $X((V,E\backslash\{g\}),H,\textsc{Fin},\textsc{Nos}) \cap \{x \in \{0,1\}^{T(E)} : x(e,T_e) = 0\}$ is non-empty for all $g \in E\backslash\{e\}$.*

Assumption 3.9 We assume that $X(G,H,\textsc{Fin},\textsc{Nos})$ is full-dimensional throughout the remainder of Section 3.2.1.

The following proposition provides sufficient conditions under which a facet-defining inequality for $Q_{\textsc{Fin}}(\mu,d)$ is facet-defining for $X(G,H,\textsc{Fin},\textsc{Nos})$ as well.

Proposition 3.10 *Let $W \subseteq V$, $\sum_{e\in\delta_G(W)} y(e) \geq \sum_{uv\in\delta_H(W)} d_{uv} =: d$ be the associated cut inequality, and $a^T x \geq \alpha$ be facet-defining for $Q_{\textsc{Fin}}(\chi^{\delta_G(W)}, d)$. Then $a^T x \geq \alpha$ is facet-defining for $X(G,H,\textsc{Fin},\textsc{Nos})$, if for all supply edges $g \in E(W) \cup E(V\backslash W)$*

$$\big\{x \in X(G,H,\textsc{Fin},\textsc{Nos}) : a^T x = \alpha,\ x(g,1) = 0\big\} \ \neq \ \emptyset\,. \tag{3.7}$$

Proof. Let $b^T x \geq \beta$ be facet-defining for $X(G,H,\textsc{Fin},\textsc{Nos})$ and suppose

$$\begin{aligned}\mathcal{F}_{b,\beta} \ &:= \ \{x \in X(G,H,\textsc{Fin},\textsc{Nos}) : b^T x = \beta\} \\ &\supseteq \ \{x \in X(G,H,\textsc{Fin},\textsc{Nos}) : a^T x = \alpha\} \quad =: \quad \mathcal{F}_{a,\alpha}.\end{aligned}$$

We show that $b^T x \geq \beta$ is a positive multiple of $a^T x \geq \alpha$. Let $g \in E$.

- $g \notin \delta_G(W)$: We show $b_g^t = 0$ for all $t = 1,\ldots,T_g$. W.l.o.g. $g \in E(W)$. Since there exists a solution $\overline{x} \in \mathcal{F}_{a,\alpha}$ with $\overline{x}(g,1) = 0$ the claim follows from $\overline{x} + \sum_{t=1}^{k} \chi^{g,t} \in \mathcal{F}_{a,\alpha}$ for all $k = 1,\ldots,T_g$.
- $g \in \delta_G(W)$: Follows from the fact that $a^T x \geq \alpha$ is facet-defining for $Q_{\textsc{Fin}}(\chi^{\delta_G(W)}, d)$ and $b_e^t = 0$ for all $e \notin \delta_G(W)$ and $t = 1,\ldots,T_e$. □

Condition (3.7) is very general. For sufficiently large capacities this condition is satisfied, for instance, if both subgraphs $G[W]$ and $G[V\backslash W]$ are two-edge connected. Furthermore, the requirements are weaker, if there exists a bridge $e \in E\backslash\delta_G(W)$ such that there exists a solution needing the free capacity of this edge only.

Band inequalities

(Stoer and Dahl, 1994) first introduced the class of band inequalities. These inequalities are valid for $Q_{\text{FIN}}(\mu, d)$ and under rather weak conditions also facet-defining for $Q_{\text{FIN}}(\mu, d)$.

Proposition 3.11 (Stoer and Dahl, 1994) *Let $\mu \in \mathbb{R}_+^E$, $F := \text{supp}(\mu)$, $d \in \mathbb{R}_+$, and $a^T x \geq \alpha$ be facet-defining for $Q_{\text{FIN}}(\mu, d)$. If $a^T x \geq \alpha$ is not a non-negative multiple of one of the ordering constraints, then*

- $a_e^t \geq 0$, *for all* $e \in F, t = 1, \ldots, T_e$,
- $\sum_{t=1}^{T_e} a_e^t = \alpha$, *for all* $e \in F$.

That is, all non-redundant inequalities for $Q_{\text{FIN}}(\mu, d)$ have non-negative coefficients, if these are not equivalent to one of the ordering constraints, and the sum of the coefficients of all edges in the support of such an inequality is equal to its right-hand side.

Definition 3.12 Let $\mu \in \mathbb{R}_+^E$, $F := \text{supp}(\mu)$ and $\sum_{e \in F} \mu_e y(e) \geq d$ be a valid inequality for $Y(G, H, \text{NOS})$. Then an assignment $B : F \to \mathbb{Z}_+$ with $B(e) \in \{0, \ldots, T_e - 1\}$ is a *band*. We often write t_e or t_e^B to denote the breakpoint $B(e)$ of a band B. Furthermore, a band B is *valid for* (μ, d) if $\sum_{e \in F} \mu_e C_e^{t_e} < d$.

Figure 3.3 visualizes the idea of a band. A band valid for (μ, d) is simply a selection of breakpoints for all supply edges in the support of μ such that the sum of the weighted breakpoint capacities is less than d, that is, does not suffice to satisfy the underlying valid inequality for $Y(G, H, \text{NOS})$. The fact that we have to increase the capacity of at least one supply edge is the interpretation of a band inequality (3.8).

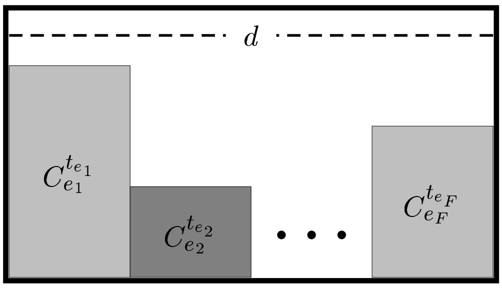


Figure 3.3: A band

Lemma 3.13 (Stoer and Dahl, 1994) *Let $\mu \in \mathbb{R}_+^E$, $F := \text{supp}(\mu)$ and $\sum_{e \in F} \mu_e y(e) \geq d$ be a valid inequality for $Y(G, H, \text{NOS})$. If a band B is valid for (μ, d), then the* ***band inequality***

$$\sum_{e \in F} x(e, t_e + 1) \quad \geq \quad 1 \tag{3.8}$$

is valid for $X(G, H, \text{FIN}, \text{NOS})$.

Band inequalities are similar to cover inequalities for the knapsack problem with generalized upper bounds (see (Wolsey, 1990)). As proven in (Stoer and Dahl, 1994), band inequalities are facet-defining for $Q_{\text{FIN}}(\mu, d)$, under conditions similar to those of cover inequalities for the knapsack polytope.

Definition 3.14 Let $\mu \in \mathbb{R}^E_+$, $F := \operatorname{supp}(\mu)$, and $d \in \mathbb{R}_+$. A band valid B for (μ, d) is *maximal* if there does not exist a valid band B_1 for (μ, d) with $C_e^{t_e^B} \leq C_e^{t_e^{B_1}}$, for all $e \in F$, and $\sum_{e \in F} \mu_e C_e^{t_e^B} < \sum_{e \in F} \mu_e C_e^{t_e^{B_1}}$.

By definition, maximal bands are maximal with respect to the componentwise order of the capacities associated with the breakpoints of the band.

Proposition 3.15 *Let* $\mu \in \mathbb{R}^E_+$, $F := \operatorname{supp}(\mu)$ *and* $\sum_{e \in F} \mu_e y(e) \geq d$ *be a valid inequality for* $Y(G, H, \text{NOS})$. *If a band* B *is valid for* (μ, d) *and* B *is maximal, then the band inequality* (3.8) *is facet-defining for* $Q_{\text{FIN}}(\mu, d)$.

2-band inequalities

The idea of band inequalities can be generalized to k-band inequalities. Instead of exactly one breakpoint, exactly k breakpoints are assigned to each supply edge. The formalism to prove that a k-band inequality is facet-defining for $Q_{\text{FIN}}(\mu, d)$ is very technical for a general k. We decided to consider the case $k \leq 2$ only.

Definition 3.16 Let $\mu \in \mathbb{R}^E_+$, $F := \operatorname{supp}(\mu)$ and $\sum_{e \in F} \mu_e y(e) \geq d$ be a valid inequality for $Y(G, H, \text{NOS})$. An assignment $B : F \to \mathbb{Z}^2_+$ with $B(e) = (t_e^1, t_e^2) \in \{0, \ldots, T_e - 1\}^2$ is a *2-band*, if $t_e^1 \leq t_e^2$ for all $e \in F$. A 2-band B is *valid for* (μ, d) if $\mu_g C_g^{t_g^2} + \sum_{e \in F \setminus \{g\}} \mu_e C_e^{t_e^1} < d$, for all $g \in F$.

Figure 3.4 visualizes the idea of a 2-band. A 2-band is simply a selection of exactly two breakpoints for all supply edges in the support of μ. It is valid for (μ, d), if for each supply edge g in the support of μ, the band $B_g := \{(g, t_g^2) \cup \{(e, t_e^1) : e \in F \setminus \{g\}$ is valid for (μ, d).

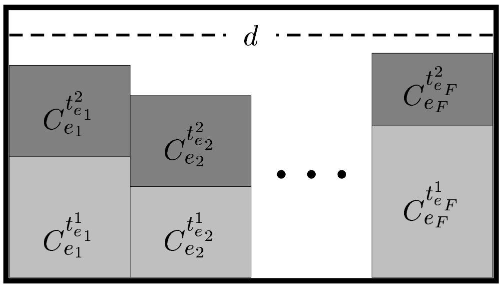


Figure 3.4: A 2-band

Lemma 3.17 *Let* $\mu \in \mathbb{R}^E_+$, $F := \operatorname{supp}(\mu)$ *and* $\sum_{e \in F} \mu_e y(e) \geq d$ *be a valid inequality for* $Y(G, H, \text{NOS})$. *For every valid 2-band* $B = \{(t_e^1, t_e^2) : e \in F\}$ *for* (μ, d), *the* ***2-band inequality***

$$\sum_{e \in F} \left(x(e, t_e^1 + 1) + x(e, t_e^2 + 1) \right) \quad \geq \quad 2 \tag{3.9}$$

is valid for $Q_{\text{FIN}}(\mu, d)$.

Proof. Let $\overline{x} \in Q_{\text{FIN}}(\mu, d) \cap \{0, 1\}^{T(F)}$. If there exists a $g \in F$ with $\overline{x}(g, t_g^2 + 1) = 1$, then (3.9) is satisfied since the ordering constraints imply $\overline{x}(g, t_g^1 + 1) = 1$. Otherwise, if $\overline{x}(g, t_g^2 + 1) = 0$, suppose that $\overline{x}(g,t_g^2) = 1$ for at most one $g \in F$, and $\overline{x}(e,t_e^1+1) = 0$ for all other $e \in F\backslash\{g\}$. Then $\overline{y} \notin Y(G,H,\text{NOS})$, for $\overline{y}(e) := C_e^0 + \sum_{t=1}^{T_e} c_e^t \overline{x}(e,t)$, $e \in F$ and $\overline{y}(e) := C_e^{T_e}$, $e \in E\backslash F$. $\square$

As for bands, we now define conditions for 2-bands such that the associated inequality (3.9) is facet-defining for $Q_{\text{FIN}}(\mu,d)$.

Definition 3.18 Let $\mu \in \mathbb{R}_+^E$, $F := \text{supp}(\mu)$, $d \in \mathbb{R}_+$. A (μ,d)-valid 2-band B is *maximal* if

(a) $\mu_g C_g^{t_g^2+1} + \sum_{e\in F\backslash\{g\}} \mu_e C_e^{t_e^1} \geq d$, for all $g \in F$, and

(b) for all $g \in F$ there exists an $h \in F\backslash\{g\}$ with $\mu_h C_h^{t_h^2} + \mu_g C_g^{t_g^1+1} + \sum_{e\in F\backslash\{g,h\}} \mu_e C_e^{t_e^1} \geq d$.

Definition 3.19 Let $\mu \in \mathbb{R}_+^E$, $F := \text{supp}(\mu)$, $d \in \mathbb{R}_+$, and suppose that B is a maximal 2-band valid for (μ,d). An *odd-cycle cover* of B is defined as an odd-cycle node-cover (see page 10) in the associated directed graph $G(B) = (V(B),A(B))$. The nodes $V(B)$ are "identical" to the edges F, and there is a directed arc $(h,g) \in A(B)$ between the nodes h and g of $V(B)$, whenever g and h satisfy condition (b) of Definition 3.18.

Proposition 3.20 *Let $\mu \in \mathbb{R}_+^E$, $F := \text{supp}(\mu)$, and $\sum_{e\in F} \mu_e y(e) \geq d$ be a valid inequality for $Y(G,H,\text{NOS})$. Furthermore, let $B = \{(t_e^1,t_e^2) : e \in F\}$ be a valid and maximal 2-band for (μ,d). If there exists an odd-cycle cover of B as defined in 3.19, then the 2-band inequality* (3.9) *is facet-defining for $Q_{\text{FIN}}(\mu,d)$.*

Proof. Let $a^T x \geq \alpha$ be valid for $Q_{\text{FIN}}(\mu,d)$ and suppose

$$\begin{aligned}\mathcal{F}_{a,\alpha} &:= \{x \in Q_{\text{FIN}}(\mu,d) : a^T x = \alpha\} \\ &\supseteq \{x \in Q_{\text{FIN}}(\mu,d) : x \text{ satisfies (3.9) at equality}\} \;=:\; \mathcal{F}.\end{aligned}$$

Let $g \in F$. We distinguish between four cases.

- $t \leq t_g^1$: We show $a_g^t = 0$. Let us define $\overline{x} \in \mathcal{F}$ by setting $\overline{x}(h,T_h) := 1$ for some $h \in F\backslash\{g\}$, and $\overline{x}(e,1) := 0$ for all $e \in F\backslash\{h\}$. Clearly, $\overline{x} \in \mathcal{F}$ and thus the claim follows from $\overline{x} + \sum_{t=1}^k \chi^{g,k} \in \mathcal{F}$ for $k = 1,\ldots,t_g^1$.

- $t > t_g^2+1$: We show $a_g^t = 0$. Let us define $\overline{x}(e,t_e^1) := 1, \overline{x}(e,t_e^1+1) := 0$ for all $e \in F\backslash\{g\}$, and $\overline{x}(g,t_g^2+1) := 1, \overline{x}(g,t_g^2+2) := 0$. We conclude $\overline{x} \in \mathcal{F}$ from condition (a) of Definition 3.19. Now, the claim follows from $\overline{x} + \sum_{t=t_g^2+2}^k \chi^{g,k} \in \mathcal{F}$ for $k = t_g^2+2,\ldots,T_g$.

- $t_g^1+1 < t \leq t_g^2$: We show $a_g^t = 0$. Due to condition (b) of Definition 3.18 there exists an $h \in F$ such that $\overline{x} \in \mathcal{F}$ for $\overline{x}(h,t_h^2) := 1, \overline{x}(g,t_h^2+1) := 0$, $\overline{x}(g,t_g^1+1) := 1, \overline{x}(g,t_g^1+2) := 0$, and $\overline{x}(e,t_e^1) := 1, \overline{x}(e,t_e^1+1) := 0$ for all $e \in F\backslash\{g,h\}$. Again, the claim follows from $\overline{x} + \sum_{t=1}^k \chi^{g,k} \in \mathcal{F}$ for $k = t_g^1+2,\ldots,t_g^2$.

- $t = t_g^2 + 1$ or $t = t_g^1 + 1$: We show $2a_g^{t_g^1+1} = 2a_g^{t_g^2+1} = \alpha$. For every $g \in F$, the vector $\overline{x}$ defined by $\overline{x}(g, T_g) := 1$ and $\overline{x}(e, 1) := 0$, for all $e \in F\backslash\{g\}$, satisfies $\overline{x} \in \mathcal{F}$. Therefore, $a_g^{t_g^1+1} + a_g^{t_g^2+1} = \alpha$, and it remains to show that $a_g^{t_g^1+1} = a_g^{t_g^2+1}$.

  Let $g_1, \ldots, g_{2k+1}, k \in \mathbb{N}$, be an odd cycle in the graph $G(B)$ of Definition 3.19. If the coefficients of the edges $g_i, 1 \leq i \leq 2k+1$, are all equal, the claim follows from the existence of an odd-cycle cover.

  To show that all these coefficients are equal, notice that for any $g, h \in F$ satisfying condition (b) of Definition 3.18, the vector $\overline{x}$ defined by $\overline{x}(h, t_h^2) := 1, \overline{x}(g, t_h^2 + 1) := 0$, $\overline{x}(g, t_g^1 + 1) := 1, \overline{x}(g, t_g^1 + 2) := 0$, and $\overline{x}(e, t_e^1) := 1, \overline{x}(e, t_e^1 + 1) := 0$ satisfies $\overline{x} \in \mathcal{F}$. Furthermore, $\overline{\overline{x}} := \overline{x} + \chi^{h, t_h^2+1} - \chi^{g, t_g^1+1} \in \mathcal{F}$, according to condition (a) of Definition 3.18, and thus, $a_g^{t_g^1+1} = a_h^{t_h^2+1}$, since $a^T(\overline{\overline{x}} - \overline{x}) = 0$.

  In consequence, $a_g^{t_{g_i}^1+1} = a_g^{t_{g_{i+1}}^2+1} = a_g^{t_{g_{i+2}}^1+1}$, for all $i = 1, \ldots, k$, and in particular, $a_g^{t_{g_1}^1+1} = a_g^{t_{g_1}^2+1} (= a_g^{t_{g_{2k+1}}^1+1})$, since the cycle is odd. □

k-graph-partition band inequalities

(Stoer and Dahl, 1994) presented inequalities based on k-graph-partitions (see Definition on page 10). Let $(V_1, \ldots, V_k), i = 1, \ldots, k$, be a partition of the node set V and denote by $F := \delta_G(V_1, \ldots, V_k)$ the edges between the shores of the partition. Let $\overline{G} = (\overline{V}, \overline{E})$ be the graph obtained by identifying the nodes of each $V_i, i = 1, \ldots, k$. That is, each node $\overline{v} \in \overline{V}$ represents one of the k node sets in the partition, and an edge $\overline{e} \in \overline{E}$ represents the edges between pairs of nodes in the node sets of the two end-nodes. Similarly, the demand graph $\overline{H} = (\overline{V}, \overline{D})$ is defined. Notice that edges between nodes of the same node set are not represented in $\overline{G}$ and $\overline{H}$, respectively. A band $B := \{(e, t_e) : e \in F\}$ is called a $\mathcal{P} - band$, and B is valid if

$$\sum_{e \in \delta_{\overline{G}}(W)} C_e^{t_e} \quad < \quad \sum_{uv \in \delta_{\overline{H}}(W)} d_{uv}$$

for every $W \subset \overline{V}$.

Lemma 3.21 *Let $\mathcal{P}$ be a k-partition and B a valid $\mathcal{P} - band$. Then the **k-graph-partition band inequality***

$$\sum_{e \in F} x(e, t_e + 1) \quad \geq \quad k - 1 \tag{3.10}$$

is valid for $X(G, H, \text{FIN}, \text{NOS})$.

Proposition 3.22 *Let $\mathcal{P}$ be a k-graph-partition and B a valid $\mathcal{P} - band$. Then the k-graph-partition band inequality* (3.10) *is facet-defining for $X(G, H, \text{FIN}, \text{NOS})$ if*

- *$G[V_i]$ is edge-connected for all $i = 1, \dots, k$,*
- *$G[V_D]$ is 2-edge-connected,*
- *$\overline{G}$ is 2-node-connected,*
- *There is no valid $\mathcal{P}$ – band $\overline{B}$ with $t_e^B \le t_e^{\overline{B}}$ for all $e \in F$, and $\sum_{e\in F} C_e^{t_e^B} < \sum_{e\in F} C_e^{t_e^{\overline{B}}}$.*

3.2.2 DIVERSIFICATION

The investigation of the polyhedron $X(G, H, \text{FIN}, \text{DIV})$ is based on relaxations defined by cut inequalities. Let $W \subset V$ be a cut with $\delta_H(W) \neq \emptyset$ and suppose the two shores of the cut are shrunken into two nodes, all supply edges with end-nodes in distinct shores are kept as parallel edges, and the demand edges with end-nodes in distinct shores are aggregated into a single demand with parameters

$$d := \sum_{uv \in \delta_H(W)} d_{uv}, \qquad \delta := \left(\sum_{uv \in \delta_H(W)} \delta_{uv} d_{uv} \right) / d\,.$$

Then, the problem can be viewed as shown in Figure 3.5.

Figure 3.5: Supply and demand graph to represent a cut

The supply graph is $(\{u, v\}, \delta_G(W))$. The nodes u and v represent the shores of the cut and the edges are exactly those from the cut (just with redefined end-nodes). Analogously, the demand graph is $(\{u, v\}, uv)$. This supply and demand graph is given in the remainder of this section and, as usual, these two graphs are denoted by $G = (V, E)$ and $H = (V, D)$. Furthermore, since the edge set of the demand graph contains only one edge, we omit unnecessary indices in the following: d is the demand value, δ is the diversification parameter and $f(e)$, $e \in E$, are the flow variables associated with the supply edges. We are interested in the situation, where the diversification parameter of the demand edge induces a diversified flow, that is, $\delta < 1$. In this case, the flow through any of the parallel supply edges between u and v is bounded from above by δd and, therefore, it is interesting to investigate the polytope

$$X_{\text{FIN}}(\delta, d) := \text{conv} \left\{ x \in \{0,1\}^{T(E)} : \begin{array}{l} \exists f \in \mathbb{R}_+^E : \sum_{e\in E} f(e) = d \text{ and for all } e \in E \;: \\ f(e) \le \min\{c_e^0 + \sum_{t=1}^{T_e} c_e^t x(e,t), \delta d\}, \\ 1 \ge x(e,1) \ge \cdots \ge x(e, T_e) \ge 0 \end{array} \right\}.$$

We start our discussion of the polytope $X_{\text{FIN}}(\delta, d)$ with its dimension and present afterwards valid inequalities.

Lemma 3.23 $|E| < \lceil 1/\delta \rceil \Rightarrow X_{\text{FIN}}(\delta, d) = \emptyset$.

Proof. For every supply edge $e \in E$, it holds $f(e) \leq \delta d$. Thus, the maximal flow is $|E| \cdot f(e) \leq |E| \cdot \delta d < d$, which contradicts the existence of a solution. □

Lemma 3.24 $\sum_{e \in E} \min\{C_e^{T_e}, \delta d\} \geq d \;\Rightarrow\; X_{\text{FIN}}(\delta, d) \neq \emptyset$.

Proof. The vector $\overline{x}$ with $\overline{x}(e,t) = 1$, for all $e \in E$, $t = 1, \ldots, T_e$, is in $X_{\text{FIN}}(\delta, d)$ if the condition of this lemma is satisfied. □

Lemma 3.25 *Let $g \in E$ and $0 \leq t_g < T_g$. If*

$$\min\{C_g^{t_g}, \delta d\} + \sum_{e \in E \setminus \{g\}} \min\{C_e^{T_e}, \delta d\} \quad < \quad d$$

then

$$X_{\text{FIN}}(\delta, d) \quad \subseteq \quad \operatorname{conv}\{x \in \{0,1\}^{T(E)} : x(g, t_g + 1) = 1\} \;.$$

Proof. A capacity less than or equal to $C_g^{t_g}$ on edge g implies $X_{\text{FIN}}(\delta, d) = \emptyset$. □

Lemma 3.25 can be used in the preprocessing. If there exists a supply edge $g \in E$ and a breakpoint $0 \leq t_g < T_g$ with $\sum_{e \in E \setminus g} \min\{C_e^{T_e}, \delta d\} + \min\{C_g^{t_g}, \delta d\} < d$, then breakpoint t_g can be removed. However, the cost and capacity coefficients of this supply edge must be redefined by setting $k_g^{t_g+1} := k_g^{t_g+1} + k_g^{t_g}, c_g^{t_g+1} := c_g^{t_g+1} + c_g^{t_g}$, removing breakpoint t_g, and shifting all breakpoints greater than t_g to the next smaller breakpoint.

Lemma 3.26 *The polytope $X_{\text{FIN}}(\delta, d)$ is full-dimensional if and only if*

$$\min\{C_g^0, \delta d\} + \sum_{e \in E \setminus \{g\}} \min\{C_e^{T_e}, \delta d\} \quad \geq \quad d,$$

for all $g \in E$.

Proof. The necessity follows from Lemma 3.25. To see sufficiency, we observe that for all $g \in E$ and $t_g = 1, \ldots, T_g$, the vector $\overline{x}$ defined by

$$\overline{x}(e, T_e) = 1, e \in E \setminus \{g\}, \quad \overline{x}(g, t_g) = 1, \quad \overline{x}(g, t_g + 1) = 0,$$

is in $X_{\text{FIN}}(\delta, d)$. □

Assumption 3.27 Throughout the remainder of Section 3.2.2, we assume that $X_{\text{FIN}}(\delta, d)$ is full-dimensional.

Lemma 3.28 *For every $(g, t_g) \in T(E)$, the ordering constraint $x(g, t_g) \geq x(g, t_g - 1)$ is facet-defining for $X_{\text{FIN}}(\delta, d)$.*

Proof. For every $(g, t_g) \in T(E)$ we define the vector $\overline{x}_{g,t_g}$ by setting

$$\overline{x}_{g,t_g}(e,t) \quad := \quad \begin{cases} 1\,, & \text{if } e \neq g \text{ or } e = g, t < t_g\,, \\ 0\,, & \text{else}\,. \end{cases}$$

Together with the vector of all 1's this yields to $|T(E)| + 1$ affinely independent vectors, where each ordering constraint is satisfied by $|T(E)|$ vectors at equality. $\square$

Lemma 3.29 *Let $a^T x \geq \alpha$ be facet-defining for $X_{\text{FIN}}(\delta, d)$ and $(g, t_g) \in T(E)$ with $C_g^{t_g - 1} \geq \delta d$. Then $a_g^{t_g} = 0$.*

Proof. Suppose $a_g^{t_g} > 0$ and $C_g^{t_g-1} \geq \delta d$. Let $\overline{x} \in \{x \in X_{\text{FIN}}(\delta, d) : a^T x = \alpha\}$ with $\overline{x}(g, t_g) = 1$ (such an $\overline{x}$ exists since otherwise the equality $\overline{x}(g, t_g) = 0$ is implied). We define a second solution $\overline{\overline{x}}$ by $\overline{\overline{x}} := \overline{x} - \chi^{g,t_g}$. Then, $\overline{\overline{x}} \in X_{\text{FIN}}(\delta, d)$, because $\overline{x} \in X_{\text{FIN}}(\delta, d)$ and $C_g^{t_g-1} \geq \delta d$ (this implies that every flow vector feasible for $\overline{x}$ is also feasible for $\overline{\overline{x}}$). However, this yields the contradiction $0 = a^T(\overline{\overline{x}} - \overline{x}) = a_g^{t_g} < 0$. $\square$

Next we introduce diversification-band inequalities, which are a generalization of band inequalities (3.8) from Section 3.2.1. The idea is the following. Let us assume for each supply edge $e \in E$ an installed capacity $C_e^{t_e}$, where $0 \leq t_e \leq T_e$. Then there are two remarks in order. First, even if $C_e^{t_e} > \delta d$, only δd units can flow over e to satisfy part of the demand d. Second, if the sum of $\min\{C_e^{t_e}, \delta d\}$ over all supply edges is smaller than d, the capacity of some supply edges must be increased.

Definition 3.30 $B = \{(e, t_e) : e \in E, 0 \leq t_e < T_e\}$ is a *diversification-band*, if $C_e^{t_e} < \delta d$ for all $e \in E$ and if the *residual band-demand* $r(B) := d - \sum_{e \in E} C_e^{t_e}$ is greater than zero.

The residual band-demand is exactly the part of the demand that still has to be satisfied if the capacity $C_e^{t_e^B}$ is installed on each supply edge $e \in E$. In principle, it is not necessary to insist on a positive residual band-demand in the definition above, but without this condition, we yield trivially satisfied inequalities with right-hand side zero. Furthermore, the condition $C_e^{t_e^B} < \delta d$ in 3.30 is not necessary to prove the validity of the associated diversification-band inequality. However, due to Lemma 3.29 this condition is necessary to get a facet-defining inequality.

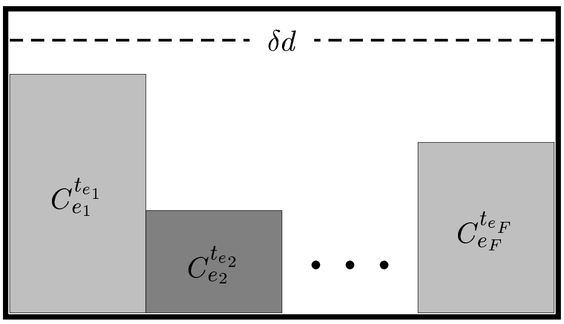


Figure 3.6: Structure of a diversification-band

As we already mentioned, the maximum flow through any of the supply edges is δd. Thus, it follows immediately that the capacity of at least $\lceil r(B)/\delta d \rceil$ edges must be strictly greater than $C_e^{t_e^B}$ in a feasible solution.

Lemma 3.31 *Let B be a diversification-band. Then the inequality*

$$\sum_{e\in E} x(e, t_e + 1) \quad \geq \quad \left\lceil \frac{r(B)}{\delta d} \right\rceil \tag{3.11}$$

is valid for $X_{\text{FIN}}(\delta, d)$.

For some diversification-bands, the right-hand side of inequality (3.11) can be further increased. No matter which breakpoints are chosen in a diversification-band, an upper bound on the flow through a supply edge remains δd. If the chosen breakpoint capacity $C_e^{t_e}$ is strictly greater than zero for some supply edge $e \in E$, then the possible additional flow $\delta d - C_e^{t_e}$ through this supply edge is less than δd. In such a case, additional capacity on $\lceil r(B)/\delta d \rceil$ edges (as required by the right-hand side of inequality (3.11)) might not suffice to satisfy the demand d. An example of this situation is the following.

Example 3.32 Suppose that $|E| = 10, d = 30, \delta = 0.1, C_e^0 = 0, C_e^1 = 2$, and $C_e^2 = 4$, for all $e \in E$. Then inequality (3.11) for diversification-band $B := \{(e, 1) : e \in E\}$ with $r(B) = 30 - 10 \cdot 2 = 10$ reads as

$$\sum_{e\in E} x(e, 2) \quad \geq \quad 4 \,,$$

but obviously, the capacity on all supply edges $e \in E$ must be increased to C_e^2. Hence, the stronger inequality

$$\sum_{e\in E} x(e, 2) \quad \geq \quad 10$$

is satisfied by all solutions. □

This observation results in the definition of a minimal residual band-demand cover, which is the minimum number of supply edges needed to satisfy the residual band-demand.

Definition 3.33 Let B be a diversification-band with residual band-demand $r(B)$ as in Definition3.30. Then the optimal solution value $\text{cov}(B)$ of

$$\begin{aligned} \min \sum_{e\in E} h(e) & \\ \sum_{e\in E} (\delta d - C_e^{t_e})\, h(e) \quad &\geq \quad r(B) \quad , \\ h(e) \quad &\in \quad \{0, 1\} \,, \quad \text{for all } e \in E \,, \end{aligned}$$

is the *minimal residual band-demand cover.* Every solution $h = (h(e))_{e\in E}$ is called residual band-demand cover for B.

With this definition, the strength of the inequalities (3.11) can be improved.

Lemma 3.34 *Let B be a diversification-band. Then the **diversification-band inequality***

$$\sum_{e \in E} x(e, t_e + 1) \quad \geq \quad \text{cov}(B) \tag{3.12}$$

is valid for $X_{\text{FIN}}(\delta, d)$.

Proof. Let $\overline{x} \in X_{\text{FIN}}(\delta, d) \cap \{0,1\}^{T(E)}$ and suppose that $\sum_{e \in E} \overline{x}(e, t_e + 1) =: k^* < \text{cov}(B)$. Through each edge $e \in E$ with $\overline{x}(e, t_e + 1) = 0$ at most $C_e^{t_e}$ can be routed, and through each edge $e \in E$ with $\overline{x}(e, t_e + 1) = 1$ at most δd can be routed. Summing up these values over all supply edges yields

$$\begin{aligned} \sum_{e \in E: \overline{x}(e, t_e+1)=0} C_e^{t_e} + \sum_{e \in E: \overline{x}(e, t_e+1)=1} \delta d \quad &= \quad \sum_{e \in E} C_e^{t_e} + \sum_{e \in E: \overline{x}(e, t_e+1)=1} (\delta d - C_e^{t_e}) \\ < \quad d - r(B) + r(B) \quad &= \quad d \,, \end{aligned}$$

where the strict inequality follows from $k^* < \text{cov}(B)$. This contradicts $\overline{x} \in X_{\text{FIN}}(\delta, d)$. □

Next, we show that diversification-band inequalities are under rather weak conditions facet-defining for $X_{\text{FIN}}(\delta, d)$. The key notion is a generalization of the maximality condition for bands, introduced in Section 3.2.1. A diversification-band $B^{'}$ *dominates* a diversification-band B, if $B^{'} \neq B$ and $C_e^{t_e^{B^{'}}} \geq C_e^{t_e^{B}}$ for every $e \in E$.

Lemma 3.35 *Let $B^{'}, B$ be diversification-bands such that $B^{'}$ dominates B. Then* $\text{cov}(B^{'}) \leq \text{cov}(B)$.

Proof. By definition, $r(B) - r(B^{'}) = \sum_{e \in E} (C_e^{t_e^{B^{'}}} - C_e^{t_e^{B}})$ holds, and therefore, every diversification cover $\overline{h} \in \{0,1\}^{T(E)}$ for B is a diversification cover for $B^{'}$, since

$$\begin{aligned} \sum_{e \in E} (\delta d - C_e^{t_e^{B^{'}}}) \overline{h}(e) \quad &= \quad \sum_{e \in E} \left((\delta d - C_e^{t_e^{B^{'}}}) + (C_e^{t_e^{B}} - C_e^{t_e^{B}}) \right) \overline{h}(e) \\ &= \quad \sum_{e \in E} (\delta d - C_e^{t_e^{B}}) \overline{h}(e) - \sum_{e \in E} (C_e^{t_e^{B^{'}}} - C_e^{t_e^{B}}) \overline{h}(e) \\ &\geq \quad r(B) - (r(B) - r(B^{'})) \quad = \quad r(B^{'}) \,. \end{aligned}$$

□

Suppose that two diversification-bands $B, B^{'}$ are given, where $B^{'}$ dominates B and $\text{cov}(B^{'}) = \text{cov}(B)$. In this case, the diversification-band inequality (3.12) for B is the sum of inequality (3.12) for $B^{'}$ and the appropriate ordering constraints. Thus, inequality (3.12) cannot be facet-defining if there exists a dominating diversification-band with the same minimal residual band-demand cover. This gives rise to the following definition of maximality.

Definition 3.36 A diversification-band B is *maximal* if $\text{cov}(B^{'}) < \text{cov}(B)$ holds for every diversification-band $B^{'}$ that dominates B.

This definition of maximality is a generalization of the maximality for bands, since the minimal residual band-demand cover is always equal to one for a band valid for d. The following lemma states a property of maximal diversification-bands needed to characterize facet-defining diversification-band inequalities. For every $e \in E$ with capacity $C_e^{t_e}$, there exists a solution in the face induced by the associated inequality (3.12).

Lemma 3.37 *Let B be a maximal diversification-band, and $g \in E$ with $C_g^{t_g+1} < \delta d$ and $t_g+1 < T_g$. Then*

$$\left\{ x \in X_{\text{FIN}}(\delta, d) : \sum_{e \in E} x(e, t_e + 1) = \text{cov}(B),\ x(g, t_g + 1) = 1,\ x(g, t_g + 2) = 0 \right\} \neq \emptyset\ .$$

Proof. Let us define a diversification-band $B^{'}$ that dominates B by setting

$$B^{'} := B \cup \{(g, t_g + 1)\} \setminus \{(g, t_g)\}\ ,$$

and choose an optimal residual band-demand cover $h^{'}$ for $B^{'}$. Since B is maximal, we know that $\text{cov}(B^{'}) < \text{cov}(B)$ which implies $\sum_{e \in E}(\delta d - C_e^{t_e^B})h^{'}(e) < r(B)$. First, we prove that $h^{'}(g) = 0$. To the contrary, suppose that $h^{'}(g) = 1$. Then

$$\begin{aligned} r(B) &= (C_g^{t_g^B+1} - C_g^{t_g^B}) + r(B^{'}) \\ &\leq (C_g^{t_g^B+1} - C_g^{t_g^B}) + (\delta d - C_g^{t_g^B+1}) + \sum_{e \in E \setminus \{g\}} (\delta d - C_e^{t_e^B})h^{'}(e) \\ &= (\delta d - C_g^{t_g^B}) + \sum_{e \in E \setminus \{g\}} (\delta d - C_e^{t_e^B})h^{'}(e) \\ &= \sum_{e \in E} (\delta d - C_e^{t_e^B})h^{'}(e) \\ &< r(B)\ . \end{aligned}$$

Hence, $h^{'}(g) = 0$. In this case, a minimal residual band-demand cover h for B is defined by

$$h(e) \quad := \quad \begin{cases} 1\ , & e = g\ , \\ h^{'}(e)\ , & \text{else}\ . \end{cases}$$

Obviously, $\sum_{e \in E}(\delta d - C_e^{t_e^B})h(e) \geq r(B)$. We define a solution $\overline{x}$ in the considered set by setting $\overline{x}(e, T_e) = 1$, if $h^{'}(e) = 1$, $\overline{x}(g, t_g + 1) = 1, \overline{x}(g, t_g + 2) = 0$, and $\overline{x}(e, t_e) = 1, \overline{x}(e, t_e + 1) = 0$ else. Obviously, $\overline{x} \in X_{\text{FIN}}(\delta, d)$ and $\sum_{e \in E} \overline{x}(e, t_e + 1) = \text{cov}(B)$, which proves the lemma. □

Using this lemma, we now state a sufficient condition such that a maximal diversification-band induces a facet-defining inequality.

Proposition 3.38 *Let B be a maximal diversification-band. The associated diversification-band inequality* (3.12) *is facet-defining for $X_{\text{FIN}}(\delta, d)$, if*

$$\left\{ x \in X_{\text{FIN}}(\delta, d) : \sum_{e \in E} x(e, t_e + 1) = \text{cov}(B),\ x(g, 1) = 0 \right\} \neq \emptyset$$

for every $g \in E$.

Proof. We proved the validity of (3.12) in Lemma 3.34. Suppose that a valid inequality $a^T x \geq \alpha$ for $X_{\text{FIN}}(\delta, d)$ exists with

$$\begin{aligned} \mathcal{F}_{a,\alpha} &:= \{x \in X_{\text{FIN}}(\delta, d) : a^T x = \alpha\} \\ &\supseteq \{x \in X_{\text{FIN}}(\delta, d) : \sum_{e \in E} x(e, t_e + 1) = \text{cov}(B)\} =: \mathcal{F}. \end{aligned}$$

Let $(e, t) \in T(E)$. To show that $a^T x \geq \alpha$ is a positive multiple of (3.12), we distinguish between three cases:

- $t < t_e + 1$: We show $a_e^t = 0$. By hypothesis there exist solutions $\overline{x}, \overline{\overline{x}} \in \mathcal{F}$ with

  $$\overline{x}(e, t-1) = 1, \quad \overline{x}(e, t) = 0, \qquad \overline{\overline{x}}(e, t) = 1, \quad \overline{\overline{x}}(e, t+1) = 0,$$

  and $\overline{x} = \overline{\overline{x}}$ otherwise. Thus, it follows $a_e^t = a^T(\overline{\overline{x}} - \overline{x}) = \alpha - \alpha = 0$.

- $t > t_e + 1$: We show $a_e^t = 0$. If $C_e^{t-1} \geq \delta d$, this follows from Lemma 3.29. Thus, we assume $C_e^{t-1} < \delta d$. Because of the maximality of B and Lemma 3.37, there exist solutions $\overline{x}, \overline{\overline{x}} \in \mathcal{F}$ with

  $$\overline{x}(e, t-1) = 1, \quad \overline{x}(e, t) = 0, \qquad \overline{\overline{x}}(e, t) = 1, \quad \overline{\overline{x}}(e, t+1) = 0,$$

  and $\overline{x} = \overline{\overline{x}}$ otherwise. Again, we conclude $a_e^t = a^T(\overline{\overline{x}} - \overline{x}) = \alpha - \alpha = 0$.

- $t = t_e + 1$: We show $\text{cov}(B) a_e^t = \alpha$. By hypothesis, for every two edges $e_1, e_2 \in E$ there exist two solutions $\overline{x}, \overline{\overline{x}} \in \mathcal{F}$ with

  $$\overline{x}(e_1, 1) = 0, \quad \overline{x}(e_2, T_{e_2}) = 1, \qquad \overline{\overline{x}}(e_2, 1) = 0, \quad \overline{\overline{x}}(e_1, T_{e_1}) = 1,$$

  and $\overline{x} = \overline{\overline{x}}$ otherwise. To see this, recall that we can exchange e_1 and e_2 in a solution since the largest breakpoint capacity is greater than d. Therefore, we get $a_{e_1}^t = a_{e_2}^t$ from $\overline{\overline{x}} - \overline{x}$. Now, the definition of the diversification-band inequality implies $\text{cov}(B) a_e^t = \alpha$. □

If there exists a solution $\overline{x}$ with $\overline{x}(g, 1) = 0$ and $\sum_{e \in E} \overline{x}(e, t_e + 1) = \text{cov}(B)$ for every $g \in E$, then the equality $\overline{x}(g, 1) = 1$ is not implied, Obviously, a necessary condition to get a facet-defining diversification-band inequality. Proposition 3.38, states that this condition is sufficient if the diversification-band is maximal.

3.2.3 RESERVATION

For the survivability model RESERVATION, the entire routing of a demand can be affected by a failure, but a minimum percentage must be reroutable. For this case, we present valid inequalities for $X(G, H, \text{FIN}, \text{RES})$ based on the ideas of (Gomory, 1969; Chvátal, 1973). Suppose that for each failure state $s \in S\backslash\{0\}$ an inequality $a_s^T x \geq \alpha_s$ is given, which is valid for the particular failure state. (In some failure states, this inequality might be $0^T x \geq 0$.) Then, the sum of these inequalities is $\sum_{s\in S\backslash\{0\}} a_s^T x \geq \sum_{s\in S\backslash\{0\}} \alpha_s$, which is by definition valid for $X(G, H, \text{FIN}, \text{RES})$. If these inequalities are well-chosen, the non-zero coefficients are all equal or an integer multiple of the smallest non-zero coefficient. In this case, it is possible to take advantage of the integrality. For example, if there exists a $k \in \mathbb{N}$ with $\sum_{s\in S\backslash\{0\}} (a_s)_e \in \{0, k\}$, for all $e \in E$, then the inequality

$$\sum_{s\in S\backslash\{0\}} \frac{1}{k} \cdot a_s^T x \quad \geq \quad \left\lceil \frac{1}{k} \cdot \sum_{s\in S\backslash\{0\}} \alpha_s \right\rceil$$

is valid for $X(G, H, \text{FIN}, \text{RES})$. The classes of *strengthened band inequalities* and *strengthened 2-band inequalities* are the result of such an argument.

Strengthened band inequalities

The class of strengthened band inequalities has been introduced in (Stoer and Dahl, 1994). In the way described above, inequalities of this class can be obtained as the sum of band inequalities.

Proposition 3.39 *Let $\mu \in \mathbb{R}_+^E$ such that $F := supp(\mu) \subseteq S$, that is, all supply edges in the support of μ are in the set of operating states. Suppose that for all $g \in F$, the inequality*

$$\sum_{e\in F\backslash\{g\}} \mu_e y(e) \quad \geq \quad \sum_{uv\in D} \pi_{uv}^g \rho_{uv} d_{uv} \quad =: \; d^g$$

*is valid for $Y(G, H, \text{RES})$, where π_{uv}^g is the value of a shortest uv-path w.r.t. μ in G_g for every $uv \in D$. Furthermore, let $B = \{(e, t_e) : e \in F\}$ be a band such that for every $g \in F$, the band $B^g := \{(e, t_e) : e \in F\backslash\{g\}\}$ is valid for (μ, d^g). Then the **strengthened band inequality***

$$\sum_{e\in F} x(e, t_e + 1) \quad \geq \quad 2 \tag{3.13}$$

is valid for $X(G, H, \text{FIN}, \text{RES})$.

Strengthened 2-band inequalities

The application of the same idea to the class of 2-band inequalities yields the strengthened 2-band inequalities. Let $\mu \in \mathbb{R}_+^E$ and $rd \in \mathbb{R}_+$ be given such that $F := supp(\mu) \subseteq S$, and suppose that for all $g \in F$, the inequality

$$\sum_{e \in F \setminus \{g\}} \mu_e y(e) \quad \geq \quad rd$$

is valid for $Y(G, H, \text{RES})$. Furthermore, let $B = \{(e, t_e^1, t_e^2) : e \in F,\ 0 \leq t_e^1 < t_e^2 < T_e\}$ be a 2-band such that for every $g \in F$ the 2-band $B^g := \{(e, t_e^1, t_e^2) : e \in F \setminus \{g\},\ 0 \leq t_e^1 < t_e^2 < T_e\}$ is valid for (μ, rd) (according to Definition 3.16). Then, the sum of the respective 2-band inequalities (3.9) is

$$\begin{aligned} \sum_{g \in F} \left(\sum_{e \in F \setminus \{g\}} (x(e, t_e^1 + 1) + x(e, t_e^2 + 1)) \right) \quad &\geq \quad \sum_{g \in F} 2 \\ \Longrightarrow \quad (|F| - 1) \sum_{e \in F} (x(e, t_e^1 + 1) + x(e, t_e^2 + 1)) \quad &\geq \quad |F| \cdot 2 \, , \end{aligned}$$

and therefore

$$\sum_{e \in F} (x(e, t_e^1 + 1) + x(e, t_e^2 + 1)) \quad \geq \quad \left\lceil \frac{|F|}{|F| - 1} \cdot 2 \right\rceil \quad \geq \quad 3 \tag{3.14}$$

is valid for $X(G, H, \text{FIN}, \text{RES})$. Inequalities (3.14) can be improved with a different definition of the validity of a 2-band.

Definition 3.40 Let $\mu \in \mathbb{R}_+^E$ such that $F := supp(\mu) \subseteq S$. Suppose that for all $g \in F$, the inequality

$$\sum_{e \in F \setminus \{g\}} \mu_e y(e) \quad \geq \quad rd$$

is a valid inequality for $Y(G, H, \text{RES})$. Then a 2-band $B = \{(e, t_e^1, t_e^2) : e \in F, 0 \leq t_e^1 < t_e^2 < T_e\}$ is *valid* for (μ, rd), if

$$(\mu_{g_1} C_{g_1}^{t_{g_1}^2} + \mu_{g_2} C_{g_2}^{t_{g_2}^2}) + \sum_{e \in F \setminus \{g, g_1, g_2\}} \mu_e C_e^{t_e^1} \quad < \quad rd \, ,$$

for all $g \in F$ and all $g_1, g_2 \in F \setminus \{g\}$ with $g_1 \neq g_2$.

Proposition 3.41 *Let μ, F, rd and the 2-band B be valid for (μ, rd) as in Definition 3.40. Then the* ***strengthened 2-band inequality***

$$\sum_{e \in F} (x(e, t_e^1 + 1) + x(e, t_e^2 + 1)) \quad \geq \quad 4 \tag{3.15}$$

is valid for $X(G, H, \text{FIN}, \text{RES})$.

Proof. First, notice that for any $g, g_1, g_2 \in F$ the capacity vector $\overline{c} \in \mathbb{Z}_+^E$, defined by

$$\overline{c}(e) \quad := \quad \begin{cases} C_e^{t_e^1}, & e \in F \backslash \{g, g_1, g_2\}, \\ C_e^{t_e^2}, & e \in \{g_1, g_2\}, \\ C_e^{T_e}, & e \in E \backslash F \cup \{g\}, \end{cases}$$

is *not* feasible, that is, $\overline{c} \notin Y(G, H, \text{RES})$, since it violates the inequality $\sum_{e \in F \backslash \{g\}} \mu_e y(e) \geq rd$, which is by assumption valid for $Y(G, H, \text{RES})$. Let $\overline{x} \in X(G, H, \text{FIN}, \text{RES}) \cap \{0,1\}^{T(E)}$ be given and let $\overline{y}$ be the associated capacity vector, that is, $\overline{y}(e) = \sum_{t=0}^{T_e} c_e^t \overline{x}(e,t)$ for all $e \in E$. For the rest of the proof, we assume that $\overline{x}$ violates (3.15) and distinguish between the following two cases.

First, suppose that there exists $g \in F$ with $\overline{x}(g, t_g^2 + 1) = 1$. Since $\overline{x}$ violates (3.15), there exists at most one supply edge, say g_1, in F with $\overline{x}(g_1, t_{g_1}^1 + 1) = 1$ and $\overline{x}(g_1, t_{g_1}^2 + 1) = 0$. In this case, $\overline{x} \notin Y(G, H, \text{RES})$, since $\overline{y} \leq \overline{c}$ for the $\overline{c}$ defined by the chosen g, g_1 and an arbitrary $g_2 \in F$.

Now, suppose that $\overline{x}(g, t_g^2 + 1) = 0$ for all $g \in F$. In this case, there exist $g, g_1, g_2 \in F$ with $\overline{y}(e) \leq C_e^{t_e^2}$ for $e \in \{g, g_1, g_2\}$ and $\overline{y}(e) \leq C_e^{t_e^1}$ for $e \in F \backslash \{g, g_1, g_2\}$. Again, $\overline{x} \notin Y(G, H, \text{RES})$, since $\overline{y} \leq \overline{c}$ for the $\overline{c}$ defined by these edges g, g_1, g_2. □

3.3 DIVISIBLE BASIC CAPACITIES: $X(G, H, \text{BAS}, \cdot)$

For the capacity model DIVISIBLE BASIC CAPACITIES, we proceed in the same way as in Section 3.2. That is, we investigate for each survivability model the projection $X(G, H, \text{BAS}, \cdot)$ of the respective polyhedron $P(G, H, \text{BAS}, \cdot)$ to the space of integer capacity variables and present classes of valid and facet-defining inequalities for $X(G, H, \text{BAS}, \cdot)$. Similar to Section 3.2, we start with the definition of a relaxation induced by a valid inequality for $Y(G, H, \cdot)$.

Definition 3.42 Let $\mu \in \mathbb{R}_+^E$, $F := \operatorname{supp}(\mu)$, and $\sum_{e \in E} \mu_e y(e) \geq d$ be a valid inequality for $Y(G, H, \cdot)$. Then the polyhedron

$$Q_{\text{BAS}}(\mu, d) := \operatorname{conv}\{x \in \mathbb{Z}_+^{T(F)} : \sum_{e \in F} \mu_e \sum_{\tau \in T} C^\tau x(e, \tau) \geq d - \sum_{e \in F} \mu_e C_e^0\},$$

is called the *induced knapsack-relaxation* for DIVISIBLE BASIC CAPACITIES.

Again, we assume an implicitly given valid inequality $\sum_{e \in E} \mu_e y(e) \geq d$ for $Y(G, H, \cdot)$, whenever we write $Q_{\text{BAS}}(\mu, d)$. In what follows, we first prove that some inequality is valid for $Q_{\text{BAS}}(\mu, d)$. This implies that the same inequality is valid for $X(G, H, \text{BAS}, \cdot)$, if we set the coefficients in $T(E \backslash F)$ to zero. Then we attempt to prove that the inequality is facet-defining for $Q_{\text{BAS}}(\mu, d)$ and try to find conditions such that the inequality is facet-defining for $X(G, H, \text{BAS}, \cdot)$. Table 3.2 shows known classes of valid inequalities for each survivability model.

Model	Inequality class	Reference
NOS	Strengthened metric inequalities Knapsack-partition inequalities	(Alevras et al., 1998b) (Pochet and Wolsey, 1995)
DIVERSIFICATION	Diversification-cut inequalities Diversification-partition inequalities Lifted diversification-cut inequalities	Section 3.3.2 Section 3.3.2 Section 3.3.2
RESERVATION	Strengthened partition inequalities	Section 3.3.3
PATH RESTORATION	see RESERVATION	

Table 3.2: Classes of non-trivial valid inequalities for $X(G, H, \text{BAS}, \cdot)$

Again, any valid inequality for $X(G, H, \text{BAS}, \text{NOS})$ is valid for the survivability model dependent polyhedra $X(G, H, \text{BAS}, \text{DIV})$, $X(G, H, \text{BAS}, \text{RES})$, and $X(G, H, \text{BAS}, \text{PATH})$, respectively, and any valid inequality for $X(G, H, \text{BAS}, \text{RES})$ is valid for $X(G, H, \text{BAS}, \text{PATH})$. For the survivability model PATH RESTORATION, no other classes of inequalities than those for RESERVATION are presented. It turned out to be difficult for this capacity model as well, to identify new classes reflecting that the normal operating state routings must be preserved in failure situations. In this section, the incidence vector $\chi^{g,t} \in \{0,1\}^{T(E)}$ for $(g,t) \in T(E)$ is defined by $\chi^{g,t}(e,\tau) = 1$, $(e,\tau) \in T(E)$, if and only if $e = g$ and $t = \tau$.

Strengthened metric inequalities

Before we consider the polyhedra for a particular survivability model, let us mention a canonical way to derive a valid inequality for $X(G, H, \text{BAS}, \cdot)$ through a divide-and-round procedure.

Proposition 3.43 *Let $\mu \in \mathbb{R}_+^E$, $F := \text{supp}(\mu)$, and $\sum_{e \in F} \mu_e y(e) \geq d$ be valid for $Y(G, H, \cdot)$ with $\mu_e C^\tau \in \mathbb{N}$ for all $e \in F$ and $\tau \in T$. Furthermore, set*

$$\begin{aligned} \overline{d} &:= d - \sum_{e \in F} \mu_e C_e^0 , \\ g &:= gcd\{\mu_e C^\tau : e \in F, \tau \in T, \text{with } \mu_e C^\tau < \overline{d}\} . \end{aligned}$$

Then the ***strengthened metric inequality***

$$\sum_{e \in F} \sum_{\tau \in T} \min\left\{ \frac{\mu_e C^\tau}{g}, \left\lceil \frac{\overline{d}}{g} \right\rceil \right\} x(e,\tau) \quad \geq \quad \left\lceil \frac{\overline{d}}{g} \right\rceil \tag{3.16}$$

is valid for $X(G, H, \text{BAS}, \cdot)$.

Proof. Analogous to inequality (3.6) for the capacity model DISCRETE CAPACITIES. □

Notice that C^{τ_1} is by definition the greatest common divisor, if the underlying valid inequality $\sum_{e \in E} \mu_e y(e) \geq d$ in Proposition 3.43 is a k-graph-partition inequality (2.23).

3.3.1 No survivability restrictions

Next, we present basic properties of the polyhedron $X(G, H, \text{Bas}, \text{Nos})$ such as its dimension and a characterization of the existence of a solution, followed by the class of *knapsack-partition inequalities*. These latter inequalities are based on valid inequalities for $Y(G, H, \text{Nos})$.

Proposition 3.44 *Let $H[V_i] = (V_i, D_i), i = 1, \ldots, k$, be the connected components of the demand graph $H_D = (V_D, D)$ (see page 32 for the definition of V_D). Then*

$$X(G, H, \text{Bas}, \text{Nos}) \neq \emptyset \iff \begin{array}{l} G[V_i] \text{ is subgraph of an edge-connected subgraph of } G \\ \text{for all } i = 1, \ldots, k. \end{array}$$

Proof. Let $i \in \{1, \ldots, k\}$ such that $G[V_i]$ is *not* subgraph of an edge-connected subgraph of G. Then a demand $uv \in D$ exists such that *no* path between its end-nodes $u, v \in V_i \subseteq V_D$ exists. Thus, it is not possible to satisfy uv, which implies $X(G, H, \text{Bas}, \text{Nos}) = \emptyset$. For the reverse direction, suppose all $G[V_i], i \in \{1, \ldots, k\}$, are subgraph of an edge-connected subgraph of G. Then, it is obvious that $\overline{x} \in X(G, H, \text{Bas}, \text{Nos})$ for $\overline{x}(e, \tau) = \lceil (\sum_{uv \in D} d_{uv}) / C^\tau \rceil$, for $e \in E$ and $\tau \in T$. $\square$

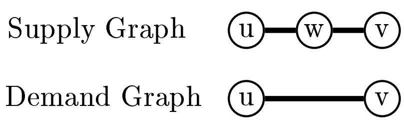


Figure 3.7: $G[V_D]$ not edge-connected, but $X(G, H, \text{Bas}, \text{Nos}) \neq \emptyset$

Note, if $X(G, H, \text{Bas}, \text{Nos}) \neq \emptyset$, the subgraphs $G[V_i]$, $i = 1, \ldots, k$, are not necessarily edge-connected. As an example consider the supply and demand graph in Figure 3.7. The demand graph $H = (V_D, uv)$ contains a single component only, which is identical to H. The associated subgraph in G is $G[V_D] = (V_D, \emptyset)$, that is, it has an empty set of supply edges and is *not* edge-connected. However, obviously it holds $X(G, H, \text{Bas}, \text{Nos}) \neq \emptyset$.

Proposition 3.45 $X(G, H, \text{Bas}, \text{Nos}) \neq \emptyset \iff \dim(X(G, H, \text{Bas}, \text{Nos})) = |T| \cdot |E|$.

Proof. Obviously, $X(G, H, \text{Bas}, \text{Nos}) \neq \emptyset$, if it is full-dimensional. For the reverse direction, let $\overline{x} \in X(G, H, \text{Bas}, \text{Nos})$. Then, $\overline{x}$ and $\overline{x} + \chi^{g,\tau} \in X(G, H, \text{Bas}, \text{Nos})$, for $e \in E$ and $\tau \in T$, are $|T(E)| + 1$ affinely independent vectors in $X(G, H, \text{Bas}, \text{Nos})$. Thus, $X(G, H, \text{Bas}, \text{Nos})$ is full-dimensional, that is, $\dim(X(G, H, \text{Bas}, \text{Nos})) = |T(E)| = |T| \cdot |E|$. $\square$

Proposition 3.46 *Let $\mu \in \mathbb{R}^E_+$, $F := \operatorname{supp}(\mu)$, and $\sum_{e \in F} \mu_e y(e) \geq d$ be a valid inequality for $Y(G, H, \text{Nos})$. If $a^T x \geq \alpha$ is facet-defining for $Q_{\text{Bas}}(\mu, d)$ and if*

$$\left\{ x \in X(G, H, \text{Bas}, \text{Nos}) : a^T x = \alpha,\ x(g, \tau) = 0 \right\} \neq \emptyset \tag{3.17}$$

for all $g \in E \backslash F$ and $\tau \in T$, then $a^T x \geq \alpha$ is facet-defining for $X(G, H, \text{Bas}, \text{Nos})$.

Proof. Let $b^T x \geq \beta$ be facet-defining for $X(G, H, \text{BAS}, \text{NOS})$ and suppose that

$$\mathcal{F}_{a,\alpha} := \{x \in X(G, H, \text{BAS}, \text{NOS}) : a^T x = \alpha\} \subseteq \{x \in X(G, H, \text{BAS}, \text{NOS}) : b^T x = \beta\} \ .$$

The claim follows, if $b^T x \geq \beta$ is a positive multiple of $a^T x \geq \alpha$. To see this, we first consider $g \notin F$ and $\tau \in T$. In this case, $b_g^\tau = 0$ follows, since $\overline{x} \in \mathcal{F}_{a,\alpha}$ exists and $\overline{x} + \chi^{g,\tau} \in \mathcal{F}_{a,\alpha}$. Thus, $b_g^\tau = b^T(\overline{x} + \chi^{g,\tau} - \overline{x}) = 0$. Knowing this, the claim easily follows since $a^T x \geq \alpha$ is facet-defining for $Q_{\text{BAS}}(\chi^F, d)$, and $b_e^\tau = 0$ for all $e \notin F$ and $\tau \in T$. □

Condition (3.17) is easily satisfied. For instance, suppose that the underlying valid inequality for $Y(G, H, \text{NOS})$ is a cut inequality with $W \subseteq V, F = \delta_G(W)$. Then, this condition is satisfied, if either the two subgraphs $G[W]$ and $G[V \backslash W]$ are two-edge connected, or if these two subgraphs are edge-connected and $|T| > 1$.

Knapsack-partition inequalities

The class of knapsack-partition inequalities is based on the work of (Pochet and Wolsey, 1995). Let

$$M \quad := \quad \{c_1, \ldots, c_n\} \subseteq \mathbb{N}$$

be satisfying the divisibility property (see page 36), that is, $c_1 \leq \cdots \leq c_n$ and $c_{i+1}/c_i \in \mathbb{N}$ for every $i = 1, \ldots, n-1$. Furthermore, for $d \in \mathbb{N}$, let $r(d) := \max\{i : c_i \leq d, 1 \leq i \leq n\}$ be the maximum index such that the associated coefficient is less than or equal to d. A partition of the index set of M w.r.t. d consists of t consecutive ordered blocks

$$\{l_1, \ldots, j_1\}, \ldots, \{l_t, \ldots, j_t\} \ ,$$

such that $l_1 = 1, l_t \leq r(d), j_t = n$, and $l_k - 1 = j_{k-1}$ for $k = 2, \ldots, t$. For this partition of the index set M, set $d_t := d$ and define

$$\kappa_k := \left\lceil \frac{d_k}{c_{l_k}} \right\rceil \ , \qquad d_{k-1} := d_k - (\kappa_k - 1) c_{l_k} \ , \tag{3.18}$$

for $k = t, \ldots, 1$. With this notation, we now formulate two important results presented in (Pochet and Wolsey, 1995).

Proposition 3.47 *The inequality*

$$\sum_{p=1}^{t} \left(\prod_{s=1}^{p-1} \kappa_s \right) \sum_{i=l_p}^{j_p} \min\left\{ \frac{c_i}{c_{l_p}}, \kappa_p \right\} x_i \quad \geq \quad \prod_{s=1}^{t} \kappa_s \tag{3.19}$$

is valid for $\operatorname{conv}\{x \in \mathbb{Z}_+^n : \sum_{i=1}^n c_i x_i \geq d\} =: Q(M, d)$.

Theorem 3.48 *The polyhedron $Q(M,d)$ is completely described by the set of all inequalities* (3.19) *and the non-negativity constraints $x_i \geq 0$ for $i = 1,\ldots,n$.*

With Proposition 3.47, we are in position to present knapsack-partition inequalities.

Proposition 3.49 *Let $V_1,\ldots,V_k$ be a k-graph-partition of G, set $F := \delta_G(V_1,\ldots,V_k)$ and $d := \sum_{uv\in\delta_H(V_1,\ldots,V_k)} d_{uv}$. For a partition $\{l_1,\ldots,j_1\},\ldots,\{l_t,\ldots,j_t\}$ of the index set of $\{C^{\tau_1},\ldots,C^{\tau_n}\}$ w.r.t. d, let $\kappa_1,\ldots,\kappa_t$ be defined as in* (3.18)*, and for $i = 1,\ldots,n$, let $p(i)$ be the partition index of technology τ_i, that is, $i \in \{l_{p(i)},\ldots,j_{p(i)}\}$. Then the **knapsack-partition inequality***

$$\sum_{e\in F}\sum_{i=1}^{n} \min\left\{\kappa_{p(i)}, \frac{C^{\tau_i}}{C^{\tau_{l_{p(i)}}}}\right\} \cdot \left(\prod_{s=1}^{p(i)-1} \kappa_s\right) x(e,\tau_i) \quad\geq\quad \prod_{i=1}^{t} \kappa_i \tag{3.20}$$

is valid for $X(G,H,\text{BAS},\text{NOS})$.

3.3.2 DIVERSIFICATION

Similar to Section 3.2.2, the investigation of the polyhedron $X(G,H,\text{BAS},\text{DIV})$ is based on a relaxation defined by a cut inequality. We consider a polyhedron associated with the particular supply and demand graph structure shown in Figure 3.5 (page 73). For the capacity model DIVISIBLE BASIC CAPACITIES this polyhedron is defined as follows:

$$\begin{aligned} X_{\text{BAS}}(\delta,d) \quad := \quad \text{conv}\{x \in \mathbb{Z}_+^{T(E)} : \exists f \in \mathbb{R}_+^E \text{ s.t.} \\ f(e) \leq \min\{\delta d, C_e^0 + \textstyle\sum_{\tau\in T} C^\tau x(e,\tau)\}, \text{for all } e \in E\}. \end{aligned}$$

As shown in the following two lemmata, this polyhedron is non-empty if there are at least $\lceil 1/\delta \rceil$ supply edges, and it is full-dimensional if it is non-empty.

Lemma 3.50 $|E| < \lceil 1/\delta \rceil \iff X_{\text{BAS}}(\delta,d) = \emptyset$.

Proof. The maximum flow over the supply edges in E to satisfy the demand d is $|E| \cdot \delta d$. Consequently, $X_{\text{BAS}}(\delta,d) = \emptyset$ if $|E| < \lceil 1/\delta \rceil$ since $|E| \cdot \delta d \leq (\lceil 1/\delta \rceil - 1)\delta d < d$. Conversely, if $|E| \geq \lceil 1/\delta \rceil$, a solution $\overline{x} \in X_{\text{BAS}}(\delta,d)$ can easily be defined by setting $\overline{x}(e,\tau) := \lceil \delta d / C^\tau \rceil$ for all $e \in E$ and $\tau \in T$. □

Lemma 3.51 $|E| \geq \lceil 1/\delta \rceil \iff X_{\text{BAS}}(\delta,d)$ *is full-dimensional.*

Proof. Clearly, $|E| \geq \lceil 1/\delta \rceil$ is necessary due to Lemma 3.50. To see sufficiency, we observe that $X_{\text{BAS}}(\delta,d) = X_{\text{BAS}}(\delta,d) + \mathbb{R}_+^{T(E)}$. □

For every $e \in E$ and $\tau \in T$, the trivial inequalities $x(e,\tau) \geq 0$ are almost always facet-defining for $X_{\text{BAS}}(\delta,d)$. The only exception to this rule occurs if $|E| = \lceil 1/\delta \rceil$ and $|T| = 1$. In this case, the right-hand side must be appropriately increased such that the face induced by the trivial inequality is non-empty.

Proposition 3.52 *Let $|E| \geq \lceil 1/\delta \rceil$ and $T = \{\tau\}$. Then*

$$x(e,\tau) \quad \geq \quad \max\left\{0, \left\lceil \frac{d-(|E|-1)\delta d}{C^\tau} \right\rceil \right\} \tag{3.21}$$

is, for every $e \in E$, facet-defining for $X_{\text{BAS}}(\delta, d)$. If $|E| \geq \lceil 1/\delta \rceil$ and $|T| \geq 2$, then

$$x(e,\tau) \quad \geq \quad 0 \tag{3.22}$$

is, for every $e \in E$ and every $\tau \in T$, facet-defining for $X_{\text{BAS}}(\delta, d)$.

Proof. Set $k := \max\{0, \lceil (d - (|E|-1)\delta d)/C^\tau \rceil\}$ and let $g \in E$.

(3.21): Let $T = \{\tau\}$ and set $\mathcal{F} := \{x \in X_{\text{BAS}}(\delta, d) : x(g,\tau) = k\}$. The validity of (3.21) is obvious. We define $\overline{x} \in X_{\text{BAS}}(\delta, d)$ by setting $\overline{x}(g,\tau) := k$ and $\overline{x}(e,\tau) := \lceil \delta d / C^\tau \rceil$ for every $e \in E \backslash \{g\}$. By definition, $\overline{x} \in \mathcal{F}$. The same is obviously true for every $\overline{x} + \chi^{e,\tau} \in X_{\text{BAS}}(\delta, d), e \in E\backslash\{g\}$. Thus, there are $|T(E)|$ affinely independent solutions contained in $\mathcal{F}$.

(3.22): Assume that $|T| \geq 2$, choose $\tau_1 \in T$, and set $\mathcal{F} := \{x \in X_{\text{BAS}}(\delta, d) : x(g, \tau_1) = 0\}$. The validity of (3.22) is again obvious. Similar to the the previous case, we identify sufficiently many affinely independent solutions in $\mathcal{F}$. First, we define $\overline{x} \in X_{\text{BAS}}(\delta, d)$ by setting $\overline{x}(e,\tau_1) := \lceil \delta d / C^{\tau_1} \rceil$, $e \in E\backslash\{g\}$, $\overline{x}(g,\tau_1) := 0$, and $\overline{x}(e, \tau_2) := 0$, $e \in E, \tau_2 \in T\backslash\{\tau_1\}$. By definition, $\overline{x} \in \mathcal{F}$. In addition, $\overline{x} + \chi^{e,\tau_1} \in X_{\text{BAS}}(\delta, d) \cap \mathcal{F}$ for every $e \in E\backslash\{g\}$, and $\overline{x} + \chi^{e,\tau_2} \in X_{\text{BAS}}(\delta, d) \cap \mathcal{F}$ for every $e \in E$ and $\tau_2 \in T\backslash\{\tau_1\}$. Again, there are $|T(E)|$ affinely independent solutions contained in $\mathcal{F}$. □

To exclude technical details, we make a few (practically reasonable) assumptions.

Assumption 3.53 In the remainder of Section 3.3.2, we assume that

- $X_{\text{BAS}}(\delta, d)$ is full-dimensional,
- $C_e^0 = 0$, for all $e \in E$, and
- $|E| > \lceil 1/\delta \rceil$ or $|T| \geq 2$.

The following lemma accumulates some results about the structure of coefficients of facet-defining inequalities for $X_{\text{BAS}}(\delta, d)$.

Lemma 3.54 *Let $a^T x \geq \alpha$ be facet-defining for $X_{\text{BAS}}(\delta, d)$ and $e \in E$:*

1. *$a_e^\tau \geq 0$, for every $\tau \in T$.*
2. *If $\alpha \neq 0$ then either $a_e^\tau > 0$ for all $\tau \in T$, or $a_e^\tau = 0$ for all $\tau \in T$.*
3. *If $\alpha \neq 0$ then $a_e^{\tau_1} = a_e^{\tau_2}$, for all $\tau_1, \tau_2 \in T$ with $C^{\tau_1} > C^{\tau_2} \geq \delta d$.*

Proof. Let $\mathcal{F} := \{x \in X_{\mathrm{BAS}}(\delta, d) : a^T x = \alpha\}$, $\overline{x} \in \mathcal{F}$, and $e \in E$.

1. Suppose that $\tau \in T$ with $a_e^\tau < 0$ exists. Then we get the following contradiction: $\alpha = a^T(\overline{x} + \chi^{e,\tau}) = \alpha + a_e^\tau < \alpha$.
2. Suppose that $\tau_1, \tau_2 \in T$ such that $a_e^{\tau_1} = 0$ and $a_e^{\tau_2} > 0$ exists. W.l.o.g. we can assume $\overline{x}(e, \tau_2) \geq 1$. (Otherwise, $\overline{x}(e, \tau_2) = 0$ is implied.) Then we get with
$$\overline{\overline{x}} := \overline{x} - \chi^{e,\tau_2} + \lceil C^{\tau_2}/C^{\tau_1} \rceil \chi^{e,\tau_1} \in X_{\mathrm{BAS}}(\delta, d),$$
a contradiction since $0 \leq a^T(\overline{\overline{x}} - \overline{x}) = -a_e^{\tau_2} < 0$.
3. The inequality $a_e^{\tau_1} \geq a_e^{\tau_2}$ follows immediately from $C^{\tau_1} > C^{\tau_2}$. To prove the other inequality ($a_e^{\tau_1} \leq a_e^{\tau_2}$) let us assume w.l.o.g. $\overline{x}(e, \tau_1) \geq 1$. (We can assume this, since otherwise $\overline{x}(e, \tau_1) = 0$ is implied.) Since $C^{\tau_2} \geq \delta d$, it holds
$$\overline{\overline{x}} := \overline{x} + \chi^{e,\tau_2} - \chi^{e,\tau_1} \in X_{\mathrm{BAS}}(\delta, d),$$
and thus the claim follows from $a_e^{\tau_2} - a_e^{\tau_1} = a^T(\overline{x} + \chi^{e,\tau_2} - \chi^{e,\tau_1} - \overline{x}) \geq 0$. □

The central notion needed to define different classes of valid and facet-defining inequalities for $X_{\mathrm{BAS}}(\delta, d)$ is a *minimal diversification cover*, which is based on the following question: how many units of a capacity C are needed to satisfy a demand d, if there are k supply edges and if the diversification parameter is δ? The value of the minimal diversification cover is important, since it often defines coefficients or right-hand side of valid and facet-defining inequalities.

Definition 3.55 Let $0 < \delta < 1, C \in \mathbb{N}, k \in \mathbb{N}$ with $k \geq \lceil 1/\delta \rceil > 1$. A solution to

$$\min \sum_{i=1}^{k} \sum_{t=0}^{\lceil \delta d/C \rceil} t z_i^t$$

$$\sum_{i=1}^{k} \left(\delta d z_i^{\lceil \delta d/C \rceil} + \sum_{t=0}^{\lceil \delta d/C \rceil - 1} t C z_i^t \right) \geq d \,, \tag{3.23}$$

$$\sum_{t=0}^{\lceil \delta d/C \rceil} z_i^t = 1 \,, \quad i = 1, \ldots, k \,, \tag{3.24}$$

$$z_i^t \in \{0, 1\} \,, \qquad i = 1, \ldots, k \,,\ t = 0, \ldots, \lceil \delta d/C \rceil \,, \tag{3.25}$$

is a *diversification cover*. An optimal solution and its value are called *minimal diversification cover*, and abbreviated by $(d, \delta d, C, k)$.

Obviously, every solution $\overline{z}$ for the $\{0, 1\}$-program in Definition 3.55 is completely determined through the unique $t_i, i = 1, \ldots, k$ with $\overline{z}_i^{t_i} = 1$. This justifies the notation $\overline{z} = (t_1, \ldots, t_k)$ for such a solution. Let $\overline{\tau} \in T$ such that $C^{\overline{\tau}} = C$ and let $\overline{z} = (t_1, \ldots, t_k)$ be a diversification cover. Then, the *associated $C^{\overline{\tau}}$-solution* $\overline{x} \in X_{\mathrm{BAS}}(\delta, d)$ is defined by

$$\overline{x}(e_i, \tau) := \begin{cases} t_i \,, & \tau = \overline{\tau} \,, \\ 0 \,, & \text{else} \,, \end{cases}$$

for every $e_i \in E = \{e_1, \ldots, e_k\}$ and $\tau \in T$. Thus, the capacity for each supply edge $e \in E$ is $C^{\overline{\tau}}\overline{x}(e, \overline{\tau})$, and the flow vector $\overline{f} \in \mathbb{R}_+^E$ defined by

$$\overline{f}(e) \quad := \quad \min\{C^{\overline{\tau}}\overline{x}(e, \overline{\tau}), \lceil \delta d \rceil\},$$

for every $e \in E$, satisfies the capacity and flow bound constraints. Hence, $\overline{x} \in X_{\text{BAS}}(\delta, d)$ follows, if $\sum_{e \in E} \overline{f}(e) \geq d$ (justifying the notion "associated C^τ-solution"), which is true since

$$\sum_{e_i \in E} \overline{f}(e_i) \; = \; \sum_{e_i \in E} \min\{C^{\overline{\tau}}\overline{x}(e_i, \overline{\tau}), \lceil \delta d \rceil\} \; = \; \sum_{e_i \in E} \left(\delta d z_i^{\lceil \delta d / C \rceil} + \sum_{t=1}^{\lceil \delta d / C \rceil - 1} t C z_i^t \right) \; \geq \; d\,.$$

The following lemma is important for the forthcoming proofs.

Lemma 3.56 *Let $C = C^\tau$. If there is a unique minimal diversification cover $(d, \delta d, C^\tau, |E|)$, then this is $\overline{z} = (\lceil \delta d / C^\tau \rceil, \ldots, \lceil \delta d / C^\tau \rceil)$. Otherwise, two different minimal diversification cover exist for $(d, \delta d, C^\tau, |E|)$ and every $e_1, e_2 \in E$, whose associated C^τ-solutions $\overline{x}$ and $\overline{\overline{x}}$ satisfy $\overline{\overline{x}} = \overline{x} - \chi^{e_1, \tau} + \chi^{e_2, \tau}$.*

Proof. First, suppose that the minimal diversification cover $\overline{z}$ is unique and $\overline{z}_i < \lceil \delta d / C^\tau \rceil$ for some $i \in \{1, \ldots, k\}$. Then, choose $j \in \{1, \ldots, k\}$ with $i \neq j$ and $\overline{z}_j > 0$. (Such a j exists, since $\delta d < d$.) It is easy to see that $\overline{z} - \chi^{\{j\}} + \chi^{\{i\}}$ satisfies constraints (3.23) – (3.25) and is thus a minimal diversification cover; contradicting the uniqueness. Knowing this, the other claim follows easily. Just notice that the same exchange argument can be applied since the supply edges are interchangeable. □

A lower bound on the value of a minimal diversification cover is obviously $\lceil d/C \rceil$, since this number of units of C is even without diversification restriction needed.

Lemma 3.57 $(d, \delta d, C, k) \geq \lceil d/C \rceil$.

Proof. If we relax in (3.23) the coefficient δd to $\lceil \delta d / C \rceil C$, we get

$$(d, \delta d, C, k) \geq \min \left\{ \sum_{i=1}^{k} \sum_{t=0}^{\lceil \delta d / C \rceil} t z_i^t : \begin{array}{rcll} \sum_{i=1}^{k} \sum_{t=0}^{\lceil \delta d / C \rceil} t C z_i^t & \geq & d, & \\ \sum_{t=0}^{\lceil \delta d / C \rceil} z_i^t & = & 1, & i = 1, \ldots, k, \\ z_i^t & \in & \{0, 1\}, & i = 1, \ldots, k, \\ & & & t = 0, \ldots, \lceil \delta d / C \rceil \end{array} \right\}.$$

Thus, the lemma follows from

$$\sum_{i=1}^{k} \sum_{t=0}^{\lceil \delta d / C \rceil} t z_i^t \; = \; \frac{1}{C} \sum_{i=1}^{k} \sum_{t=0}^{\lceil \delta d / C \rceil} C t z_i^t \; \geq \; \frac{d}{C}$$

and the integrality of z. □

In some cases, the value $(d, \delta d, C, k)$ can immediately be determined. For instance, if the flow bound δd is an integral multiple of the capacity C and if $C \geq \delta d$.

Lemma 3.58 $\delta d/C \in \mathbb{N} \Rightarrow (d, \delta d, C, k) = \lceil d/C \rceil$.

Proof. To define a minimal diversification cover $\overline{z} = (t_1, \ldots, t_k)$ of value $\lceil d/C \rceil$, we set

$$t_i := \begin{cases} \lceil \delta d/C \rceil \ , & i \in \{1, \ldots, \lfloor 1/\delta \rfloor\} \ , \\ \lceil (d - \lfloor 1/\delta \rfloor \delta d)/C \rceil \ , & i = \lfloor 1/\delta \rfloor + 1 \ , \\ 0 \ , & \text{else} \ , \end{cases}$$

for $i = 1, \ldots, k$. By definition constraints (3.24) and (3.25) are satisfied. The same holds for (3.23) since

$$\sum_{i=1}^{\lfloor 1/\delta \rfloor} \delta d + \left\lceil \frac{d - \lfloor 1/\delta \rfloor \, \delta d}{C} \right\rceil \cdot C \ \geq \ \lfloor 1/\delta \rfloor \, \delta d + d - \lfloor 1/\delta \rfloor \, \delta d \ = \ d \ .$$

Thus, the claim follows from

$$\sum_{i=1}^{k} \sum_{t=0}^{\lceil \delta d/C \rceil} t z_i^t \ = \ \sum_{i=1}^{k} t_i \ = \ \left\lfloor \frac{1}{\delta} \right\rfloor \cdot \left\lfloor \frac{d}{C} \right\rfloor + \left\lceil \frac{d - \lfloor 1/\delta \rfloor \delta d}{C} \right\rceil \ = \ \left\lceil \frac{d}{C} \right\rceil \ . \qquad \square$$

The following lemmata provide obvious bounds for the minimal diversification cover and a formula to recursively calculate its value. These will be used in the validity proofs for some classes of inequalities.

Lemma 3.59 $C \geq \delta d \Rightarrow (d, \delta d, C, k) = \lceil 1/\delta \rceil$.

Lemma 3.60 $(d, \delta d, C, k) \leq (d - \delta d, \delta d, C, k - 1) + \lceil \delta d/C \rceil$.

Lemma 3.61 *If $C \leq \delta d$, we have*

$$(d, \delta d, C, k) = \begin{cases} \lceil d/C \rceil \ , & \text{if } kC \lfloor \delta d/C \rfloor \geq d \ , \\ k \lfloor \delta d/C \rfloor + (d - kC \lfloor \delta d/C \rfloor, \delta d - C \lfloor \delta d/C \rfloor, C, k) \ , & \text{else} \ . \end{cases}$$

Proof. The first case is obvious. If we use on each of the k edges the capacity $\lfloor \delta d/C \rfloor C$ then there remains a demand of $d - kC \lfloor \delta d/C \rfloor > 0$ and the remaining bound on the flow is $\delta d - C \lfloor \delta d/C \rfloor$. $\square$

Note, the calculation rule in Lemma 3.61 is satisfied for $C > \delta d$ as well, but in this case the equality is not very informative since it reduces to $(d, \delta d, C, k) = (d, \delta d, C, k)$. Summarizing the previous results, the value of a minimal diversification cover satisfies

$$\left\lceil \frac{d}{C} \right\rceil \ \leq \ (d, \delta d, C, k) \ \leq \ \left\lceil \frac{1}{\delta} \right\rceil \left\lceil \frac{\delta d}{C} \right\rceil \ , \tag{3.26}$$

where the left inequality is tight for $\delta d/C \in \mathbb{N}$, and the right inequality is tight for $C \geq \delta d$. For $C < \delta d$ (the usual case in our practical application), we construct a small example showing that the upper bound can be attained.

Example 3.62 Let $d = 560$ and $\delta = 0.5$ be the demand parameters, and let $C = 30$ and $k = 2$. In this case,

$$(d, \delta d, C, k) = (560, 280, 30, 2) = 20 = \lceil 1/\delta \rceil \lceil \delta d / C \rceil \ .$$

It is not possible to achieve $(d, \delta d, C, k) = 19$, since the best combination of 19 units of 30 is

$$2 \cdot 9 \cdot 30 + 1 \cdot (280 - 270) = 550 < 560 \ ,$$

since the flow through any of the supply edges is bounded from above by $280 = 0.5 \cdot 560$. □

After the characterization of the minimal diversification cover, we now turn to valid inequalities for the polyhedron $X_{\text{BAS}}(\delta, d)$.

Lemma 3.63 *The* ***diversification-cut inequality***

$$\sum_{e \in E} \sum_{\tau \in T} \min \left\{ \frac{C^\tau}{C^{\tau_1}}, \left\lceil \frac{\delta d}{C^{\tau_1}} \right\rceil \right\} x(e, \tau) \quad \geq \quad (d, \delta d, C^{\tau_1}, |E|) \tag{3.27}$$

is valid for $X_{\text{BAS}}(\delta, d)$.

Proof. Let $\overline{x} \in X_{\text{BAS}}(\delta, d) \cap \mathbb{Z}_+^{T(E)}$. We define a vector $\overline{z} = (t_1, \ldots, t_k)$, $k = |E|$, which satisfies for $C = C^{\tau_1}$ the constraints (3.23) – (3.25), and which further satisfies

$$\sum_{\tau \in T} \min \left\{ \frac{C^\tau}{C^{\tau_1}}, \left\lceil \frac{\delta d}{C^{\tau_1}} \right\rceil \right\} \overline{x}(e_i, \tau) \quad \geq \quad t_i \tag{3.28}$$

for all $e_i \in E$. In this case, the claim follows since the sum of inequalities (3.28) for all $e_i \in E$ satisfies

$$\sum_{e_i \in E} \sum_{\tau \in T} \min \left\{ \frac{C^\tau}{C^{\tau_1}}, \left\lceil \frac{\delta d}{C^{\tau_1}} \right\rceil \right\} \overline{x}(e_i, \tau) \ \geq \ \sum_{e_i \in E} t_i \ \geq \ (d, \delta d, C^{\tau_1}, |E|) \ .$$

Let $\overline{z}$ be defined by

$$t_i \quad := \quad \min \left\{ \sum_{\tau \in T} \frac{C^\tau}{C^{\tau_1}} \overline{x}(e, \tau), \left\lceil \frac{\delta d}{C^{\tau_1}} \right\rceil \right\} \ ,$$

for $e_i \in E$. By definition, constraints (3.24) and (3.25) are satisfied. To see that constraint (3.23) is satisfied, let $I_1 := \{i : t_i = \lceil \delta d / C^{\tau_1} \rceil, 1 \leq i \leq k\}$ and $I_2 := \{1, \ldots, k\} \backslash I_1$. Then (3.23) reduces to

$$|I_1| \cdot \delta d + \sum_{i \in I_2} t_i C \geq d \ ,$$

since otherwise $\overline{x} \notin X_{\text{BAS}}(\delta, d)$. (Recall, the maximum flow through a supply edge is δd.) □

Example 3.64 Inequalities (3.27) are not facet-defining, in general. A counter example is the following: Let $d = 564$, $\delta = 0.75$, $|E| = 2$, $T = \{\tau_1, \tau_2\}$ and $C^{\tau_1} = 30, C^{\tau_2} = 480$. It is easy to verify that $(d, \delta d, C^{\tau_1}, |E|) = 19$, and thus inequality (3.27) reads as

$$\sum_{e \in E} (x(e, \tau_1) + \min\{16, 15\} x(e, \tau_2)) \quad \geq \quad 19 \; .$$

However, this inequality is not facet-defining, since it is dominated by the valid inequality

$$\sum_{e \in E} (x(e, \tau_1) + 14 x(e, \tau_2)) \quad \geq \quad 19 \; .$$

To see the validity of the latter inequality, consider the possible choices of a feasible solution. Let $\overline{x}$ be such a solution with $\overline{x}(g, \tau_2) \geq 1$ for exactly one $g \in E$ (the other cases are obvious). Then, a flow of $\delta d = 423$ can be routed over g, and the remaining demand of value $564 - 423 = 141$ can be satisfied by installing five units of capacity 30 or one unit of capacity 480 on the other supply edge. In both cases the inequality is satisfied. □

Suppose that a capacity larger than or equal to δd is chosen on $k \leq \lceil 1/\delta \rceil - 1$ supply edges. The maximum flow over these edges is $k \cdot \delta d$ and the remaining demand is $d_k := d - k\delta d \leq \delta d$. This must be satisfied with the edges in E_k, which are all supply edges but those with a capacity larger than δd. Then, any inequality valid for the polyhedron $\operatorname{conv}\{x \in \mathbb{Z}_+^{T(E_k)} : \sum_{e \in E_k} C^\tau x(e, \tau) \geq d_k\}$ is valid for $X_{\mathrm{BAS}}(\delta, d)$. Furthermore, every partition of the index set of $\{C^{\tau_1}, \ldots, C^{\tau_n}\}$ w.r.t. d_k gives rise to a knapsack-partition inequality (3.20) which is valid for $\operatorname{conv}\{x \in \mathbb{Z}_+^{T(E_k)} : \sum_{e \in E_k} C^\tau x(e, \tau) \geq d_k\}$. This proves the following proposition.

Proposition 3.65 *Let $k \leq \lceil 1/\delta \rceil - 1, E_k \subseteq E$ with $|E_k| = |E| - k$ and set $d_k := d - k\delta d$. For a partition $\{l_1, \ldots, j_1\}, \ldots, \{l_t, \ldots, j_t\}$ of the index set of $\{C^{\tau_1}, \ldots, C^{\tau_n}\}$ w.r.t. d_k, let $\kappa_1, \ldots, \kappa_t$ be defined as in (3.18), and for $i = 1, \ldots, n$, let $p(i)$ be the partition index of technology τ_i, that is, $i \in \{l_{p(i)}, \ldots, j_{p(i)}\}$. Then, the **diversification-partition inequality***

$$\sum_{e \in E_k} \sum_{i=1}^{n} \min\left\{\kappa_{p(i)}, \frac{C^{\tau_i}}{C^{\tau_{l_{p(i)}}}}\right\} \cdot \left(\prod_{s=1}^{p(i)-1} \kappa_s\right) x(e, \tau_i) \quad \geq \quad \prod_{i=1}^{t} \kappa_i \tag{3.29}$$

is valid for $X_{\mathrm{BAS}}(\delta, d)$.

For a particular knapsack-partition inequality, it is possible to prove that the diversification-partition inequality is facet-defining for $X_{\mathrm{BAS}}(\delta, d)$.

Proposition 3.66 *Let $k := \lceil 1/\delta \rceil - 1, E_k \subseteq E$ with $|E_k| = |E| - k$ and set $d_k := d - k\delta d$. Then*

$$\sum_{e \in E_k} \sum_{\tau \in T} \min\left\{\frac{C^\tau}{C^{\tau_1}}, \left\lceil \frac{d_k}{C^{\tau_1}} \right\rceil\right\} x(e, \tau) \quad \geq \quad \left\lceil \frac{d_k}{C^{\tau_1}} \right\rceil \quad (= (d_k, \delta d, C^{\tau_1}, |E_k|)) \tag{3.30}$$

is facet-defining for $X_{\mathrm{BAS}}(\delta, d)$.

Proof. The validity of (3.30) follows from Lemma 3.57 and Lemma 3.63 applied to the supply graph $G_k = (V, E_k)$ with demand d_k. To prove that it is facet-defining for $X_{\text{BAS}}(\delta, d)$, let us assume the existence of a valid inequality $a^T x \geq \alpha$ for $X_{\text{BAS}}(\delta, d)$ with

$$\begin{aligned}\mathcal{F}_{a,\alpha} &:= \{x \in X_{\text{BAS}}(\delta, d) : a^T x = \alpha\} \\ &\supseteq \{x \in X_{\text{BAS}}(\delta, d) : \sum_{e \in E_k} \sum_{\tau \in T} \min\{C^\tau / C^{\tau_1}, \lceil d_k / C^{\tau_1} \rceil\} x_e^\tau = \lceil d_k / C^{\tau_1} \rceil\} \;=:\; \mathcal{F}.\end{aligned}$$

Notice that $\mathcal{F} \neq \emptyset$ since the demand d_k can be satisfied by setting $\overline{x}(g, \tau_1) = \lceil d_k / C^{\tau_1} \rceil$, for a single $g \in E_k$, and $\overline{x}(e, \tau) = 0$ for $e \in E_k \setminus \{g\}$ or $\tau_1 \neq \tau \in T$. We distinguish between two cases:

- $\overline{e} \in E \setminus E_k$ and $\overline{\tau} \in T$: In this case we get $a_{\overline{e}}^{\overline{\tau}} = 0$, because $\overline{x} + \chi^{\overline{e},\overline{\tau}} \in \mathcal{F}$ for every $\overline{x} \in \mathcal{F}$. $(a_{\overline{e}}^{\overline{\tau}} = a^T(\overline{x} + \chi^{\overline{e},\overline{\tau}}) - a^T \overline{x} = \alpha - \alpha = 0.)$
- $\overline{e} \in E_k$ and $\overline{\tau} \in T$: We define a solution $\overline{x} \in X_{\text{BAS}}(\delta, d)$ by setting

$$\overline{x}(e, \tau) := \begin{cases} \lceil \delta d / C^\tau \rceil \,, & e \in E \setminus E_k \,, \\ \lceil d_k / C^{\tau_1} \rceil \,, & e = \overline{e}, \tau = \tau_1 \,, \\ 0 \,, & \text{else} \,. \end{cases}$$

(Such a solution exists since it is possible to route $d - d_k$ over the edges $E \setminus E_k$ and d_k over edge $\overline{e} \in E_k$.) Obviously, $\overline{x} \in \mathcal{F}$ since

$$\sum_{e \in E_k} \sum_{\tau \in T} \min\left\{\frac{C^\tau}{C^{\tau_1}}, \left\lceil \frac{d_k}{C^{\tau_1}} \right\rceil\right\} \overline{x}(e, \tau) = \min\left\{1, \left\lceil \frac{d_k}{C^{\tau_1}} \right\rceil\right\} \cdot \left\lceil \frac{d_k}{C^{\tau_1}} \right\rceil = \left\lceil \frac{d_k}{C^{\tau_1}} \right\rceil \,.$$

Since $\mathcal{F} \subseteq \mathcal{F}_{a,\alpha}$, it follows $\lceil d_k / C^{\tau_1} \rceil a_{\overline{e}}^{\tau_1} = \alpha$. To complete the proof, it remains to show that $a_{\overline{e}}^{\overline{\tau}} = \min\{C^\tau / C^{\tau_1}, \lceil d_k / C^{\tau_1} \rceil\} a_{\overline{e}}^{\tau_1}$. Let $\overline{\overline{x}} \in X_{\text{BAS}}(\delta, d)$ by defined by

$$\overline{\overline{x}} := \overline{x} + \chi^{\overline{e},\overline{\tau}} - \min\{C^{\overline{\tau}} / C^{\tau_1}, \lceil d_k / C^{\tau_1} \rceil\} \chi^{\overline{e},\tau_1} \,.$$

$\overline{\overline{x}}$ defines a solution since the induced capacity on each supply edge is at least the capacity induced by $\overline{x}$. Furthermore, $\overline{x} \in \mathcal{F}$ since

$$\begin{aligned}&\sum_{e \in E_k} \sum_{\tau \in T} \min\left\{\frac{C^\tau}{C^{\tau_1}}, \left\lceil \frac{d_k}{C^{\tau_1}} \right\rceil\right\} \overline{\overline{x}}(e, \tau) = \\ &\qquad \left\lceil \frac{d_k}{C^{\tau_1}} \right\rceil + \min\left\{\frac{C^{\overline{\tau}}}{C^{\tau_1}}, \left\lceil \frac{d_k}{C^{\tau_1}} \right\rceil\right\} - \min\left\{\frac{C^{\overline{\tau}}}{C^{\tau_1}}, \left\lceil \frac{d_k}{C^{\tau_1}} \right\rceil\right\} = \left\lceil \frac{d_k}{C^{\tau_1}} \right\rceil \,.\end{aligned}$$

This implies

$$a_{\overline{e}}^{\overline{\tau}} - \min\{C^{\overline{\tau}} / C^{\tau_1}, \lceil d_k / C^{\tau_1} \rceil\} a_{\overline{e}}^{\tau_1} = a^T(\overline{\overline{x}} - \overline{x}) = \alpha - \alpha = 0 \,.$$

Thus, $a^T x \geq \alpha$ is an integral multiple of (3.30), which proves the proposition. $\square$

The support of inequalities (3.30) is defined on $|E| - \lceil 1/\delta \rceil + 1$ supply edges and therefore the coefficients of at least one supply edge are zero. The next task is to extend the support of these inequalities to some of the "missing" edges. There are two remarks in order. Since each inequality (3.30) is facet-defining, the lifting coefficient of any of the variables not in the support is zero, as shown in the proof of Proposition 3.66. Second, if we extend the inequality to an edge (not in the support) we have to find positive coefficients for all available technologies of the edge, as we know from Lemma 3.54. Before we formalize the construction of a new valid inequality, let us consider an example.

Example 3.67 Let $|E| = 3$ and $T = \{\tau_1, \tau_2, \tau_3\}$, $d = 190$, $\delta = 0.5$, $C^{\tau_1} = 30, C^{\tau_2} = 60$, and $C^{\tau_3} = 120$. Due to Proposition 3.66,

$$\sum_{e \in \{e_1, e_2\}} (x(e,\tau_1) + 2x(e,\tau_2) + 4x(e,\tau_3)) \geq 4$$

is facet-defining for $X_{\mathrm{BAS}}(\delta, d)$. Furthermore, due to Lemma 3.63,

$$\sum_{e \in \{e_1, e_2, e_3\}} (x(e,\tau_1) + 2x(e,\tau_2) + 4x(e,\tau_3)) \geq 7$$

is valid for $X_{\mathrm{BAS}}(\delta, d)$. However, this inequality can be improved. Let us consider the valid inequality

$$x(e_3,\tau_1) + 2x(e_3,\tau_2) + \sum_{e \in \{e_1, e_2\}} (x(e,\tau_1) + 2x(e,\tau_2) + 4x(e,\tau_3)) \geq 4$$

and lift the variable $x(e_3, \tau_3)$. The lifting coefficient is defined by

$$\begin{aligned} & \min\{ && x(e_3,\tau_1) + 2x(e_3,\tau_2) + \textstyle\sum_{e \in \{e_1,e_2\}} (x(e,\tau_1) + 2x(e,\tau_2) + 4x(e,\tau_3)) : \\ & && x \in X_{\mathrm{BAS}}(\delta, d), x(e_3,\tau_3) = 0\} - 4 \\ = \; & 7 - 4 && = \; \mathbf{3} \, . \end{aligned}$$

Thus, we have proven that

$$x(e_3,\tau_1) + 2x(e_3,\tau_2) + \mathbf{3}x(e_3,\tau_3) + \sum_{e \in \{e_1, e_2\}} (x(e,\tau_1) + 2x(e,\tau_2) + 4x(e,\tau_3)) \geq 7$$

is valid for $X_{\mathrm{BAS}}(\delta, d)$ in this example. In this particular case, the last inequality is also facet-defining. This will follow from Proposition 3.70. $\square$

Following the reasoning of the previous example, we now construct a new inequality which is often facet-defining for $X_{\mathrm{BAS}}(\delta, d)$. Let $k = \lceil 1/\delta \rceil - 1$, $E_k \subseteq E$ with $|E_k| = |E| - k$, and $d_k := d - k\delta d$. Proposition 3.66 implies that

$$\sum_{e \in E_k} \sum_{\tau \in T} \min\left\{ \frac{C^{\tau}}{C^{\tau_1}}, \left\lceil \frac{d_k}{C^{\tau_1}} \right\rceil \right\} x(e,\tau) \quad \geq \quad (d_k, \delta d, C^{\tau_1}, |E_k|)$$

is facet-defining for $X_{\text{BAS}}(\delta, d)$. If we relax the coefficients $\lceil d_k/C^{\tau_1}\rceil$ to $\lceil \delta d/C^{\tau_1}\rceil$ and extend the resulting inequality to some variables $x(\overline{e}, \overline{\tau})$ of an edge $\overline{e} \notin E_k$ and all technologies $\overline{\tau} \in T$ with $C^\tau < \delta d$, it remains true that

$$\sum_{\tau \in T: C^\tau < \delta d} \frac{C^\tau}{C^{\tau_1}} x(\overline{e}, \tau) + \sum_{e \in E_k} \sum_{\tau \in T} \min\left\{\frac{C^\tau}{C^{\tau_1}}, \left\lceil \frac{\delta d}{C^{\tau_1}} \right\rceil\right\} x(e, \tau) \quad \geq \quad (d - k\delta d, \delta d, C^{\tau_1}, |E_k|)$$

is valid for $X_{\text{BAS}}(\delta, d)$. This inequality is, of course, not facet-defining and thus we try to lift the missing coefficients for $\tau \in T$ with $C^\tau \geq \delta d$ into this inequality. First we note that the coefficients for all these variables $x(\overline{e}, \tau)$ are equal, as we know from Lemma 3.54. Thus, it suffices to consider the case, where all these variables are set to zero and we have to cover a demand of $d - (k-1)\delta d$. Let us define

$$L_k \quad := \quad (d - (k-1)\delta d, \delta d, C^{\tau_1}, |E_k| + 1) - (d - k\delta d, \delta d, C^{\tau_1}, |E_k|). \tag{3.31}$$

By definition, $0 < L_k \leq \lceil \delta d/C^{\tau_1}\rceil$, for every $k \in \mathbb{N}$ with $k \leq \lceil 1/\delta \rceil - 1$.

Lemma 3.68 $(d - (k-1)\delta d, \delta d, C^{\tau_1}, |E_k| + 1) =$

$$\min\left\{\sum_{e \in E_k \cup \overline{e}} \sum_{\tau \in T} \min\left\{\frac{C^\tau}{C^{\tau_1}}, \left\lceil \frac{\delta d}{C^{\tau_1}} \right\rceil\right\} x(e, \tau) : x \in X_{\text{BAS}}(\delta, d), \sum_{\tau \in T: C^\tau \geq \delta d} x(\overline{e}, \tau) = 0\right\}.$$

Proof. Every optimal solution to $(d - (k-1)\delta d, \delta d, C^{\tau_1}, |E_k| + 1)$ defines a solution satisfying the conditions of the right-hand side. Thus, '$\geq$' is proven. For the reverse direction we remark that every solution of right-hand side can easily be transformed into a minimal diversification cover for $d - (k-1)\delta d$, δd, C^{τ_1}, and $|E_k| + 1$. $\square$

Proposition 3.69 *For every $k \in \mathbb{N}$ with $k \leq \lceil 1/\delta \rceil - 1$, $E_k \subseteq E$ with $|E_k| = |E| - k$ and $\overline{e} \in E \backslash E_k$, the **lifted diversification-cut inequality***

$$\sum_{\tau \in T} \min\left\{\frac{C^\tau}{C^{\tau_1}}, L_k\right\} x(\overline{e}, \tau) \quad + \quad \sum_{e \in E_k} \sum_{\tau \in T} \min\left\{\frac{C^\tau}{C^{\tau_1}}, \left\lceil \frac{\delta d}{C^{\tau_1}} \right\rceil\right\} x(e, \tau)$$
$$\geq \quad (d - (k-1)\delta d, \delta d, C^{\tau_1}, |E_k| + 1) \tag{3.32}$$

is valid for $X_{\text{BAS}}(\delta, d)$.

Proof. Let $\overline{x} \in X_{\text{BAS}}(\delta, d)$. We distinguish between two cases:

- For $\sum_{\tau \in T: C^\tau \geq \delta d} \overline{x}(\overline{e}, \tau) = 0$, the validity follows immediately from Lemma 3.68.

- In the other case, there exists $\tau \in T$ with $C^\tau \geq \delta d$ and $\overline{x}(\overline{e},\tau) \geq 1$. Then, it is possible to route δd over $\overline{e}$ and, in consequence, at most a flow of value $k\delta d$ over the edges in $E \backslash E_k$. The remaining demand $(d - k\delta d)$ must be routed over the edges in E_k. Thus,

$$\begin{array}{rll} & \sum_{\tau \in T} \min\left\{\frac{C^\tau}{C^{\tau_1}}, L_k\right\} \overline{x}(\overline{e},\tau) & + \sum_{e \in E_k} \sum_{\tau \in T} \min\left\{\frac{C^\tau}{C^{\tau_1}}, \left\lceil \frac{\delta d}{C^{\tau_1}} \right\rceil\right\} \overline{x}(e,\tau) \\ \geq & L_k & + (d - k\delta d, \delta d, C^{\tau_1}, |E_k|) \\ = & (d-(k-1)\delta d, \delta d, C^{\tau_1}, |E_k|+1) & \end{array}$$

by definition of L_k and Proposition 3.66. □

We observe that $\overline{x} \in X_{\mathrm{BAS}}(\delta,d)$ satisfying (3.32) at equality exists, since the C^{τ_1}-solution associated with a minimal diversification cover for $(d-(k-1)\delta d, \delta d, C^{\tau_1}, |E_k|+1)$ satisfies (3.32) at equality. Under further assumptions, inequalities (3.32) are even facet-defining for $X_{\mathrm{BAS}}(\delta,d)$.

Proposition 3.70 *For $k = \lceil 1/\delta \rceil - 1$, $E_k \subseteq E$ with $|E_k| = |E| - k$ and $\overline{e} \in E \backslash E_k$, inequality (3.32) is facet-defining for $X_{\mathrm{BAS}}(\delta,d)$, if there exist two different optimal solutions for $(d-(k-1)\delta d, \delta d, C^{\tau_1}, |E_k|+1)$, and an optimal solution $\overline{z}$ for $(d-k\delta d, \delta d, C^{\tau_1}, |E_k|)$ with $\overline{z}_i^{\lceil \delta d/C^{\tau_1} \rceil} = 1$, for some $e_i \in E_k$.*

Proof. Suppose that $a^T x \geq \alpha$ is a valid inequality for $X_{\mathrm{BAS}}(\delta,d)$ with

$$\begin{array}{rcl} \mathcal{F} & := & \{x \in X_{\mathrm{BAS}}(\delta,d) : x \text{ satisfies (3.32) at equality}\} \\ & \subseteq & \{x \in X_{\mathrm{BAS}}(\delta,d) : a^T x = \alpha\} . \end{array}$$

Let $g \in E$. We distinguish between five cases to show that $a^T x = \alpha$ is a positive integer multiple of (3.32).

- $g \notin E_k \cup \{\overline{e}\}, \tau \in T$: Obviously, $a_g^\tau = 0$, since $\overline{x} + \chi^{g,\tau} \in \mathcal{F}$ for every $\overline{x} \in \mathcal{F}$.
- $\tau = \tau_1$: Choose $e_1 \neq e_2 \in E$. We show $a_{e_1}^{\tau_1} = a_{e_2}^{\tau_1}$. Lemma 3.56 implies that two minimal diversification cover $\overline{z} \neq \overline{\overline{z}}$ for $(d-(k-1)\delta d, \delta d, C^{\tau_1}, |E_k|+1)$ exist such that the associated C^{τ_1}-solutions $\overline{x}, \overline{\overline{x}} \in \mathcal{F}$ satisfy $\overline{\overline{x}} = \overline{x} - \chi^{e_1,\tau_1} + \chi^{e_2,\tau_1}$. Therefore, it follows $a_{e_2}^{\tau_1} - a_{e_1}^{\tau_1} = a^T(\overline{\overline{x}} - \overline{x}) = \alpha - \alpha = 0$.
- $g = \overline{e}, \tau \in T \backslash \{\tau_1\}$: We show $a_{\overline{e}}^\tau = \min\{C^\tau/C^{\tau_1}, L_k\} a_{\overline{e}}^{\tau_1}$. Let $\overline{z}$ be a minimal diversification cover for $(d-(k-1)\delta d, \delta d, C^{\tau_1}, |E_k|+1)$ with $\overline{z}_{i_{\overline{e}}}^t = 1$ ($i_{\overline{e}}$ is the index of edge $\overline{e}$) and let $\overline{x} \in \mathcal{F}$ be the associated C^{τ_1}-solution. By definition, $\overline{x}(\overline{e},\tau_1) > 0$. Furthermore, define $\overline{\overline{x}} := \overline{x} - \min\{C^\tau/C^{\tau_1}, L_k\} \chi^{\overline{e},\tau_1} + \chi^{\overline{e},\tau} \in \mathbb{Z}_+^{T(E)}$. Obviously, $\overline{\overline{x}}$ satisfies (3.32) at equality and $\overline{\overline{x}} \in X_{\mathrm{BAS}}(\delta,d)$ since $C^\tau \geq \min\{C^\tau/C^{\tau_1}, L_k\} \cdot C^{\tau_1}$. Thus, the claim follows from $a_{\overline{e}}^\tau - \min\{C^\tau/C^{\tau_1}, L_k\} a_{\overline{e}}^{\tau_1} = a^T(\overline{\overline{x}} - \overline{x}) = \alpha - \alpha = 0$.
- $g \in E_k, \tau \in T, C^\tau \leq \delta d$: We show $a_g^\tau = C^\tau/C^{\tau_1} a_g^{\tau_1}$. Since $C^\tau \leq \delta d$ there exists a minimal diversification cover $\overline{z}$ for $(d-(k-1)\delta d, \delta d, C^{\tau_1}, |E_k|+1)$ with $\overline{z}_{i_g}^t = 1$ (i_g is the index of edge g) for $t = C^\tau/C^{\tau_1}$. For the associated C^{τ_1}-solution $\overline{x} \in \mathcal{F}$ we define $\overline{\overline{x}} := \overline{x} - C^\tau/C^{\tau_1} \chi^{g,\tau_1} + \chi^{g,\tau}$. As in the previous case, $\overline{\overline{x}} \in \mathcal{F}$, and thus the claim follows from $a_g^\tau - C^\tau/C^{\tau_1} a_g^{\tau_1} = a^T(\overline{\overline{x}} - \overline{x}) = \alpha - \alpha = 0$.

- $g \in E_k, \tau \in T, C^\tau > \delta d$: We show $a_g^\tau = \lceil \delta d / C^{\tau_1} \rceil a_g^{\tau_1}$. Choose a minimal diversification cover $\overline{z}$ for $(d - k\delta d, \delta d, C^{\tau_1}, |E_k|)$ with $\overline{z}_{i_g}^{\lceil \delta d / C^{\tau_1} \rceil} = 1$. Such a solution exists by assumption, since it exists for at least one edge in E_k. With the associated C^{τ_1}-solution $\overline{x} \in \mathcal{F}$ and $\overline{\overline{x}} := \overline{x} - \lceil \delta d / C^{\tau_1} \rceil \chi^{e,\tau_1} + \chi^{e,\tau} \in \mathcal{F}$, the result follows as in the previous cases. □

The assumption of two optimal solutions for the minimal diversification cover $(d, \delta d, C^{\tau_1}, |E|)$ in Proposition 3.70 is necessary as we see in the following example.

Example 3.71 Let $|E| = 2$, $T = \{\tau_1, \tau_2\}$, $d = 220, \delta = 0.5$, $C^{\tau_1} = 30$, and $C^{\tau_2} = 90$. In this case $(d, \delta d, C^{\tau_1}, |E|) = 8$ and $\overline{z} = \{4, 4\}$ is the minimal diversification cover. Due to Proposition 3.66 the inequalities $x(e_1, \tau_1) + 3x(e_1, \tau_2) \geq 4$ and $x(e_2, \tau_1) + 3x(e_2, \tau_2) \geq 4$ are both facet-defining for $X_{\text{BAS}}(\delta, d)$, and therefore, inequality (3.32), which reads as $x(e_1, \tau_1) + 3x(e_1, \tau_2) + x(e_2, \tau_1) + 3x(e_2, \tau_2) \geq 8$, cannot be facet-defining for $X_{\text{BAS}}(\delta, d)$. □

3.3.3 RESERVATION

Similar to Section 3.2.3, we derive valid inequalities for the polyhedron $X(G, H, \text{BAS}, \text{RES})$ based on the ideas of (Gomory, 1969; Chvátal, 1973). In this case, these are the classes of *strengthened metric inequalities* (3.35) and *strengthened knapsack-partition inequalities* (3.36).

Lemma 3.72 *Suppose that $F \subseteq S \cap E$, $a^\tau \in \mathbb{Z}_+$, for every $\tau \in T$, and $\alpha \in \mathbb{Z}_+$ is given such that for every $h \in F$, the inequality*

$$\sum_{e \in F \setminus \{h\}} a^\tau x(e, \tau) \quad \geq \quad \alpha \tag{3.33}$$

is valid for $X(G, H, \text{BAS}, \text{RES})$. Then the inequality

$$\sum_{e \in F} a^\tau x(e, \tau) \quad \geq \quad \left\lceil \frac{|F|}{|F| - 1} \cdot \alpha \right\rceil \tag{3.34}$$

is valid for $X(G, H, \text{BAS}, \text{RES})$.

Proof. Sum up (3.33) for all $h \in F$ and divide the resulting inequality by $|F| - 1$. The validity of (3.34) for $X(G, H, \text{BAS}, \text{RES})$ follows from the integrality of x. □

Strengthened metric inequalities

Applying Lemma 3.72 to metric inequalities yields the class of strengthened metric inequalities.

Proposition 3.73 *Let $\mu \in \mathbb{R}_+^E$ with $F := \text{supp}(\mu) \subseteq S$, and for every $h \in F$ and $uv \in D$, let π_{uv}^h be the value of a shortest uv-path in $G_h = (V, E_h)$ with respect to the edge weights μ. Then, the* ***strengthened metric inequality***

$$\sum_{e \in F} \sum_{\tau \in T} \min\left\{ \frac{\mu_e C^\tau}{g}, \left\lceil \frac{d}{g} \right\rceil \right\} x(e,\tau) \quad \geq \quad \left\lceil \frac{|F|}{|F|-1} \cdot \left\lceil \frac{d}{g} \right\rceil \right\rceil \tag{3.35}$$

is valid for $X(G, H, \textsc{Bas}, \textsc{Res})$, where g is the greatest common divisor of $\{\mu_e C^\tau : e \in F, \tau \in T\}$ and $d := \min\{\sum_{uv \in D} \pi_{uv}^h \rho_{uv} d_{uv} - \sum_{e \in F \setminus \{h\}} \mu_e C_e^0 : h \in F\}$.

Proof. Set $\alpha := d$, and $a^\tau := \min\{\mu_e C^\tau / g, \lceil d/g \rceil\}, \tau \in T$, in Lemma 3.72. □

Strengthened knapsack-partition inequalities

Similarly, the class of strengthened knapsack-partition inequalities is the result of applying Lemma 3.72 to the class of knapsack-partition inequalities (3.20).

Proposition 3.74 *Suppose that a k-graph-partition $V_1, \ldots, V_k$ is given, for some $k \in \mathbb{N}$, such that $F := \delta_G(V_1, \ldots, V_k) \subseteq S$. Let*

$$d \quad := \quad \min\{ \sum_{uv \in \delta_H(V_1,\ldots,V_k)} \rho_{uv} d_{uv} - \sum_{e \in F \setminus \{h\}} C_e^0 : h \in F\} \ .$$

For a partition $\{l_1, \ldots, j_1\}, \ldots, \{l_t, \ldots, j_t\}$ of the index set of $\{C^{\tau_1}, \ldots, C^{\tau_n}\}$ w.r.t. d, let $\kappa_1, \ldots, \kappa_t$ be defined as in (3.18), and for $i = 1, \ldots, n$, let $p(i)$ be the partition index of technology τ_i, that is, $i \in \{l_{p(i)}, \ldots, j_{p(i)}\}$. Then the ***strengthened knapsack-partition inequality***

$$\sum_{e \in F} \sum_{i=1}^{n} \min\left\{ \kappa_{p(i)}, \frac{C^{\tau_i}}{C^{\tau_{l_{p(i)}}}} \right\} \cdot \left(\prod_{s=1}^{p(i)-1} \kappa_s \right) x(e,\tau_i) \quad \geq \quad \left\lceil \frac{|F|}{|F|-1} \prod_{s=1}^{t} \kappa_s \right\rceil \ . \tag{3.36}$$

is valid for $X(G, H, \textsc{Bas}, \textsc{Res})$.

Proof. The partition of the index set of $\{C^{\tau_1}, \ldots, C^{\tau_n}\}$ w.r.t. d is independent from the supply edges. Hence, for every $h \in F$, the knapsack-partition inequality

$$\sum_{e \in F \setminus \{h\}} \sum_{i=1}^{n} \min\left\{ \kappa_{p(i)}, \frac{C^{\tau_i}}{C^{\tau_{l_{p(i)}}}} \right\} \cdot \left(\prod_{s=1}^{p(i)-1} \kappa_s \right) x(e,\tau_i) \quad \geq \quad \prod_{s=1}^{t} \kappa_s$$

is valid for $X(G, H, \textsc{Bas}, \textsc{Res})$. The result now follows from Lemma 3.72 for $\alpha := \prod_{s=1}^{t} \kappa_s$, and $a^\tau := \min\{\kappa_{p(i)-1}, C^{\tau_i}/C^{\tau_{l_{p(i)}}}\} \cdot \prod_{s=1}^{p(i)-1} \kappa_s$. □

Chapter 4

Algorithms and Computational Results

The last two chapters focused on theoretical aspects of network design problems: setting up appropriate mathematical models and describing the convex hull of the set of feasible solutions. This chapter concentrates on algorithmic aspects. There are several theoretical results contained in this chapter, but practically computing cost-minimal solutions is the driving force now. The general approach integrates a *cutting plane algorithm* with *heuristic algorithms*, and therefore, a *quality guarantee* for the solutions computed with the heuristic algorithms can be provided since the cutting plane algorithm yields a lower bound for the cost of an optimal solution.

The structure of this chapter is as follows. In this introductory part, we describe the practical problem instances used to evaluate the algorithms. Afterwards, we present an overview of the general approach to solve the survivable capacitated network design problems described in Chapter 2, followed by a description of algorithmic aspects of five important subproblems: preprocessing, initialization of the linear programming relaxation, feasibility of a continuous capacity vector, separation algorithms for the different classes of valid inequalities, and eventually heuristic algorithms to compute feasible solutions.

The performance of the algorithms to solve the above mentioned subproblems depends on various parameter selections. As a common testing platform for the evaluation of these parameter dependencies, nine network design problem instances are used, all based on real-world instances supplied by `E-Plus`. The structure of the original instances is maintained, but the problem instances are slightly perturbed such that disclosure agreements are not violated. Furthermore, values such as lower bounds and solution cost are scaled such that it is impossible to estimate current network costs from the results presented.

The nine problem instances fall into three different classes. Three problem instances, m1, m2 and m3, are medium-sized, three problem instances, p1, p2 and p3, contain many parallel supply edges, and three problem instances, l1, l2 and l3, are considered large-sized. The problem sizes in terms of nodes, supply edges, demand edges, integer capacity variables, and continuous path variables are shown in Table 4.1. The supply graphs range from 15 to 36 nodes and 46 to 135 edges. The number of demand edges ranges from 48 to 135. The total number of integer capacity variables and continuous path variables depends on the capacity and survivability models. Different types of communication links are used between pairs of nodes. These represent multiple leased line providers and microwave connections. The capacity choices reflect the available capacities of the PDH (see page 24). For the capacity model Discrete Capacities, the capacities 30, 60, 480, 960 and 1920 are used for supply edges corresponding to leased lines, and the capacities 60, 120, 240, 480 are used for supply edges corresponding to microwave links. Similarly, for the capacity model Divisible Basic Capacities, the basic capacities 30, 480, 1920 are used for supply edges corresponding to leased lines, and the capacities 60, 120, and 480 are used for supply edges corresponding to microwave links. With this selection, the number of capacity variables is, of course, larger for Discrete Capacities, but, these are $\{0,1\}$-variables instead of general integer variables for Divisible Basic Capacities. The number of path variables is computed for unrestricted length of valid paths. For the normal operating state, the number of path variables ranges between 3.9 million and more than 200 billion(!). Summing up over all operating states, that is, over the normal operating state, all supply node failures, and all supply edge failures, the number of path variables ranges between 153 million and more than 20000 billion(!). (We computed these values with an upper bound of 10 billion paths for the normal operating state, and roughly estimated the correct number based on the number of demands. Hence, not for all instances the exact numbers of path variables is known.) The survivability model dependent parameters are 0.5 and 0.75 for Diversification, and 0.5 and 1.0 Reservation and Path Restoration, leading to the problem instances m1d50, m1d75, p1d50, p1d75, l1d50, l1d75 for Diversification, and m1r50, m1r100, p1r50, p1r100, l1r50, l1r100 for Reservation and Path Restoration. (In the latter set of problem instances, the interpretation of the failure parameter depends on the survivability model.) As already mentioned, the set of operating states S comprises the normal operating state, all supply node failures, and all supply edge failures.

Problem	$\|V\|$	$\|E\|$	$\|D\|$	$\|x\|$ variables		$\|f\|$ variables	
				Fin	Bas	Div	Res/Path
m1	15	46	78	220	132	3 898 238	153 028 236
m2	16	51	77	243	153	19 771 917	846 193 483
m3	17	48	135	234	144	7 575 728	321 459 696
p1	21	135	48	643	405	$> 50 \cdot 10e09$	$> 8 \cdot 10e12$
p2	28	122	67	577	366	$> 50 \cdot 10e09$	$> 8 \cdot 10e12$
p3	24	97	67	461	291	$> 60 \cdot 10e09$	$> 5 \cdot 10e12$
l1	36	107	79	512	421	4 319 021 173	464 782 082 567
l2	34	100	131	480	300	$> 200 \cdot 10e09$	$> 20 \cdot 10e12$
l3	36	123	123	582	369	$> 20 \cdot 10e09$	$> 2 \cdot 10e12$

Table 4.1: The test problem instances

The used linear programming solver is Cplex 6.5.3, and all computational experiments are performed on SUN-Ultra60's (359 MHz SUNW,UltraSPARC-II) with 512MB or 1024MB main memory.

4.1 Overview

Suppose that a fixed capacity model and a survivability model are given. From an abstract point of view, solving the associated survivable capacitated network design problems reduces to solving a mixed-integer programming problem of the form

$$\min\{kx : Cx + Df \geq d, x \text{ integer}, f \geq 0\} , \tag{4.1}$$

where x represents the integer capacity variables, f the continuous path variables, C and D are appropriate matrices, and kx is the linear cost function. In Table 4.1, we have seen that even for moderately sized supply and demand graphs the number of variables in the associated mixed-integer program is huge[1]. The reason is the number of paths which is potentially exponential in the input size of the supply graph. As a consequence, it is impossible to follow the typical linear programming approach of relaxing the integrality constraints, and solving and strengthening the linear relaxation. Instead, we follow an approach which integrates decomposition techniques with cutting plane techniques and linear programming based heuristic algorithms.

The objective function coefficients of all path variables are zero. Therefore, (4.1) is equivalent to the optimization problems

$$\min\{kx : x \text{ integer}, \{f : Df \geq d - Cx, f \geq 0\} \neq \emptyset\} , \tag{4.2}$$

$$\min\{kx : x \text{ integer}, Ax \geq a, \{f : Df \geq d - Cx, f \geq 0\} \neq \emptyset\} , \tag{4.3}$$

where $\{x : Ax \geq a\}$ is a relaxation of the polyhedron $X(G, H, \cdot, \cdot)$, that is, $X(G, H, \cdot, \cdot) \subseteq \{x : Ax \geq a\}$. (Benders, 1960; Benders, 1962) suggested to decompose problem (4.3) and to solve alternately the relaxed integer program $\min\{kx : x \text{ integer}, Ax \geq a\}$ and the decision problem $\{f : Df \geq d - Cx, f \geq 0\} \neq \emptyset$. The important point of Benders approach is the following. Whenever an optimal solution of the relaxed integer program is given then either optimality can be proven or a so-called Benders cut can be derived which is violated by this optimal solution. Thus, one yields a new and stronger relaxed integer program when adding the Benders cut. Our approach is similar. In contrast to Benders decomposition, however, we do not solve *integer* programs. Instead, we seek for a good linear approximation of the polyhedron $\{x \text{ integer}, Ax \geq a\}$. That is, we compute a lower bound to the optimal solution value of the relaxed integer program. Notice that a lower bound to the relaxed integer program is a lower bound to the optimal solution value of the considered network design problem.

In the context of network design, the above mentioned decision problem is the following: given a solution $\overline{x}$ of a linear programming relaxation of $X(G, H, \cdot, \cdot)$, it is necessary to decide whether a feasible routing exists. That is, given a (continuous) capacity vector $\overline{y}$ obtained from $\overline{x}$ with (2.4) or (2.7), the following feasibility problem must be solved.

Definition 4.1 (Feasibility problem)
Given the survivability model RESERVATION or PATH RESTORATION and a (continuous) capacity vector $\overline{y}(e)$, $e \in E$: Decide whether $\overline{y} \in Y(G, H, \text{DIV}) \cap Y(G, H, \text{RES})$ or $\overline{y} \in Y(G, H, \text{DIV}) \cap Y(G, H, \text{PATH})$, respectively, that is, decide whether there exist routings for all demands in all operating states satisfying the particular routing and survivability constraints.

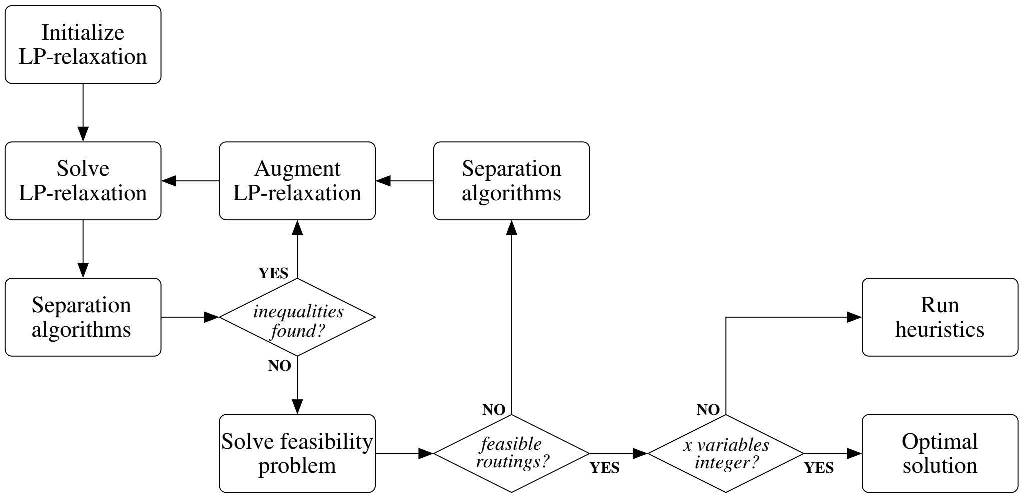


Figure 4.1: Flow chart of the algorithm.

Our algorithmic approach, sketched in the flow chart of Figure 4.1, utilizes this way of decomposing the network dimensioning problem as follows. Starting point is an initial relaxation $\{x : A_0 x \geq a_0\}$ of $X(G, H, \cdot, \cdot)$, for which an optimal solution $\overline{x}$ can be identified by means of linear programming. In the next step, the following separation problem for the optimal solution $\overline{x}$ of the current relaxation, and some of the classes of valid inequalities for the particular polytope $X(G, H, \cdot, \cdot)$ is solved.

Definition 4.2 (Separation problem)
Let $\mathcal{C}$ be a class of valid inequalities for the polytope $X \subseteq \mathbb{R}^n$, that is, $X \subseteq \{x \in \mathbb{R}^n : c^T x \geq \gamma, \text{for all } (c, \gamma) \in \mathcal{C}\}$, and $\overline{x} \in \mathbb{R}^n$. Decide whether $c^T \overline{x} \geq \gamma$ for all $(c, \gamma) \in \mathcal{C}$, or identify $(c, \gamma) \in \mathcal{C}$ with $c^T \overline{x} < \gamma$.

If one or several separation algorithms identify violated inequalities, then these are added to the current relaxation. This way, a larger and stronger relaxation is obtained. As most important property, the new relaxation does not contain $\overline{x}$ and thus the optimal solution value of the new relaxation provides an improved lower bound for the optimal solution value of the considered problem instance. These two steps of solving the linear relaxation and the separation problem are iterated, until no further violated inequality can be identified. Then, a capacity vector $\overline{y}$ is calculated according to either (2.4) or (2.7), and in the next step, the feasibility problem for $\overline{y}$ is solved.

Often, if the capacity vector $\overline{y}$ is *not* feasible, a valid metric inequality for the respective polyhedron $Y(G, H, \cdot)$ can be derived. In fact, for DIVERSIFICATION and RESERVATION a violated metric inequality can always be derived, and for PATH RESTORATION this is under further restrictions possible. Given a metric inequality that is violated by $\overline{y}$, it is always possible to identify a valid inequality for the respective polyhedron $X(G, H, \cdot, \cdot)$ that is violated by $\overline{x}$; see Sections 3.2 and 3.3. Summarizing, the algorithm to solve the feasibility problem for

[1]The memory of todays workstations does not even suffice to store the associated mixed-integer programs.

a capacity vector $\overline{y}$ serves as a separation algorithm for $\overline{x}$ as well. Again, identified violated inequalities are added to the current relaxation and the new stronger relaxation is resolved. There are two possibilities if the capacity vector $\overline{y}$ turns out to be feasible. If $\overline{x}$ is integral, the solution is optimal. In this case, feasible routings can be computed with the algorithm to solve the feasibility problem for the capacity vector $\overline{y}$. If $\overline{x}$ is not integral, heuristic algorithms are applied to find “reasonable” integral solutions.

The cutting plane phase provides a lower bound $z_{\text{LP}} = k\overline{x}$, where $\overline{x}$ is an optimal solution of the final linear programming relaxation; and the best heuristic solution provides an upper bound z_{IP} to the unknown optimal solution value. Thus, a quality guarantee for the best solution found by the algorithm can be provided, which is an upper bound on the gap between the values of an optimal solution and the best solution found. This quality guarantee is given by the quantity

$$\frac{z_{\text{IP}} - z_{\text{LP}}}{z_{\text{LP}}} 100\% \ .$$

4.2 Preprocessing

The purpose of preprocessing is twofold. Given a particular problem instance, the first target is to decide whether there exists a feasible solution, or not. If infeasibility of the problem can be proven, the overall algorithm terminates. The second target is to reduce the problem size.

4.2.1 Detecting infeasibility

The initial step of the preprocessing consists of solving the feasibility problem from Definition 4.1 for a particular capacity vector. For the capacity model DISCRETE CAPACITIES, this capacity vector consists of the largest capacities $C_e^{T_e}$ for all supply edges $e \in E$. For the capacity model DIVISIBLE BASIC CAPACITIES, this capacity vector consists of sufficiently large capacities (e.g., the smallest valid capacity larger than the sum over all demand values). This task is difficult, however, since a subproblem of the feasibility problem is already $\mathcal{NP}$-complete. As we will see in Section 4.4, the algorithm to solve the feasibility problem also solves the following *uncapacitated* network design problems.

Definition 4.3 (Uncapacitated network design problems)
Let the topology of a supply graph $G = (V, E)$, and a demand graph $H = (V, D)$ together with the demand edge dependent survivability parameters be given. Then, the following decision problems are uncapacitated network design problems:

- Do there exist $\lceil 1/\delta_{uv} \rceil$ node-disjoint paths of length at most ℓ_{uv}?
- Does there exist a $uv-$path in G_s for all $s \in S \backslash \{0\}$ with $uv \in D_s$ and $\rho_{uv} > 0$?

Definition 4.3 asks for the existence of sufficiently many node-disjoint paths in all operating states for all demands. Obviously, the existence of a solution for this uncapacitated network design problem is a necessary condition for the existence of a feasible solution for the survivable capacitated network design problems presented in Chapter 2. As shown in (Itai et al., 1982), the first problem in Definition 4.3 is $\mathcal{NP}$-complete. Instead of an exact method, we employ algorithms from the software library developed by (Bley, 1997), which contains a set of heuristic algorithms to solve various "node-disjoint length-restricted path problems".

An obvious algorithm to solve the second problem of Definition 4.3 is the following: For each failure state $s \in S\backslash\{0\}$ and each demand $uv \in D_s$ with $\rho_{uv} > 0$, test whether u and v belong to the same component of the supply graph G_s. This can obviously be done in polynomial time, but it is rather time consuming. The following sufficient criteria for the existence of a $uv-$path in G_s for all $s \in S\backslash\{0\}$ with $uv \in D_s$ and $\rho_{uv} > 0$ help improving the computation time:

- G is two-node connected.
- For all $uv \in D$ there exist two node-disjoint paths in G between u and v.
- $V \cap S = \emptyset$ and G is two-edge connected.
- $V \cap S = \emptyset$ and for all $uv \in D$ there exist two edge-disjoint paths in G between u and v.

Given the existence of a solution for the uncapacitated network design problems from Definition 4.3, the algorithms described in Section 4.4 are used to solve the feasibility problem for the above mentioned vector of large capacities. As we will see in Section 4.4, the feasibility algorithms for the survivability models Diversification and Reservation are exact, that is, this capacity vector *is* feasible if these algorithms do not prove the infeasibility. For Path Restoration, however, the algorithm is not exact, that is, in some cases the algorithm terminates without proof of feasibility or infeasibility.

Assumption 4.4 Throughout the rest of this chapter, we assume the existence of a feasible solution for the considered survivable capacitated network design problem instances.

4.2.2 Problem reduction

There are various ways to reduce the problem size. By means of decomposition, it might be possible to obtain a series of smaller subproblems which are easier to solve, and the parameters of the supply and demand edges might allow to reduce the set of operating states, to remove some of the valid capacities, or to decide the routing of a demand beforehand.

It turned out, however, that none of the following reduction techniques applies in our test problem instances. Obviously, the `E-Plus` engineers had already done a good preprocessing themselves.

Decomposition

- Suppose that the supply graph of the problem instance has the structure illustrated in Figure 4.2, that is, there exists an articulation node $w \in V$. Hence, there exist $V_1, V_2 \subseteq V$ with $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \{w\}$ and $\delta_G(V_1\backslash\{w\}) \cup \delta_G(V_2\backslash\{w\}) = \delta_G(w)$.

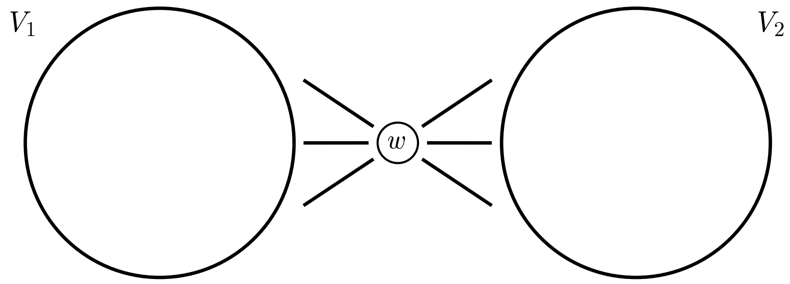


Figure 4.2: Decomposition w.r.t. an articulation node

If all demand edges in $\delta_H(V_1)$ are *not* path-length restricted, the problem decomposes into two subproblems:

$$\begin{aligned} G_1 &= (V_1, E(V_1)) \ , & H_1 &= (V_1, D(V_1) \cup \delta_H(V_1)) \ , \\ G_2 &= (V_2, E(V_2)) \ , & H_2 &= (V_2, D(V_2) \cup \delta_H(V_2)) \ . \end{aligned}$$

Each supply edge appears in exactly one subproblem, the parameters of each supply edge are exactly those from the original problem, and the parameters of a demand are those of the associated demand in the original problem.

- Suppose that the supply graph of the problem instance has the structure illustrated in Figure 4.3, that is, the node set V of the supply graph $G = (V, E)$ can be partitioned into subsets $V_1, V_2 \subseteq V$ with $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$ such that two articulation nodes $v_1 \in V_1, v_2 \in V_2$ exist, which satisfy $\delta_G(V_1) = \delta_G(V_2) = \delta_G(v_1) \cap \delta_G(v_2)$. The only supply edges in the cut $\delta_G(V_1)$ are those between the nodes v_1 and v_2.

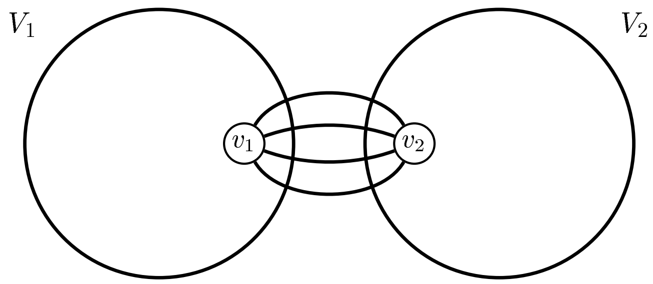


Figure 4.3: Decomposition w.r.t. a multi-bridge

Again, if all demand edges in $\delta_H(V_1)$ are *not* path-length restricted, the problem decomposes into three subproblems with the following supply and demand graphs:

$$\begin{aligned} G_1 &= (V_1, E(V_1)) \ , & H_1 &= (V_1, D(V_1) \cup (\delta_H(V_1) \backslash \delta_H(v_1))) \ , \\ G_2 &= (V_2, E(V_2)) \ , & H_2 &= (V_2, D(V_2) \cup (\delta_H(V_2) \backslash \delta_H(v_2))) \ , \\ G_3 &= (\{v_1, v_2\}, \delta_G(v_1) \cap \delta_G(v_2)) \ , & H_3 &= (\{v_1, v_2\}, \delta_H(V_1)) \ . \end{aligned}$$

Each supply edge appears in exactly one of the three subproblems, and the parameters of each supply edge are exactly those of the original problem. The three demand graphs are a little bit more complicated since every demand that crosses the cut appears in two or three of the subproblems. For each demand edge of one of the three subproblems there is a unique associated demand edge in the original problem and each demand inherits its parameters from this associated demand.

If some demand edges in $\delta_H(V_1)$ are path-length restricted, the problem still decomposes, if (w.l.o.g.) $V_1 = \{v_1\}$. In this case, the decomposition is defined as above with the small change that the length parameter for the demand edges $uv_2 \in \delta_H(v_2)\backslash\delta_H(V_2)$ has to be set to $\ell_{uv_1} - 1$, where ℓ_{uv_1} is the length parameter of the original demand edge between v_1 and u which has been propagated to node v_2.

Parameter dependent reductions

- If the demand graph $H = (V, D)$ is a star, that is, if there is a node $v \in V$ which is end-node of all demand edges, then $D_v = \emptyset$, and therefore we can set

  $$S := S\backslash\{v\} \ .$$

  Practically, this situation comes up in the planning of BSS-networks. Recall from Chapter 1 that all BSCs within a BSS-region are logically connected to a single MSC.

- If the respective diversification and failure parameters satisfy $1 - \delta_{uv} \geq \rho_{uv}$ for all demand edges $uv \in D$, then it suffices to consider the normal operating state. In this case,

  $$S := \{0\} \ ,$$

  since any feasible routing for the normal operating state is feasible for all failure states.

- If the capacity model is DISCRETE CAPACITIES, it is possible to remove breakpoints $t \in \{1, \dots, T_e - 1\}$, if either

  $$C_e^t \leq C_e^{t+1} \text{ and } K_e^t \geq K_e^{t+1} \ , \quad \text{or} \quad C_e^{t-1} \geq \max\{\sum_{uv\in D} \delta_{uv} d_{uv}, \sum_{uv\in D} \rho_{uv} d_{uv}\} \ ,$$

  that is, either if there is a larger capacity at smaller cost (modulo equality) or if there is a smaller capacity that is already larger than the maximum possible flow through any supply edge. If the capacity model is DIVISIBLE BASIC CAPACITIES, a similar argument applies. It is possible to remove technology $\overline{\tau} \in T$, if $m \cdot K_e^\tau < K_e^{\overline{\tau}}$, for some $\tau \in T$ and some $1 \leq m \cdot C^\tau < C^{\overline{\tau}}$, where $m \cdot C^\tau$ must be larger than the maximum possible flow through any supply edge.

- If there are exactly $1/\delta_{uv} \in \mathbb{N}$ supply edges between u and v, the routing of all demands $uv \in D$ with length restriction $\ell_{uv} = 1$ and $\rho_{uv} = 0$ can be decided immediately. More generally, if there is only a unique routing for a demand, then these routings can by fixed and the demand edge can be removed from the demand graph. Notice that it is necessary to appropriately update the free capacity of all supply edges used in the routing.

4.3 Linear programming relaxation

The running time of the cutting plane algorithm depends on the selection and the initialization of the linear programming relaxation. In this section, we present our choice of the linear programming relaxation. The mathematical models presented in Chapter 2 contain an exponential number of variables. Hence, it is not practical to consider the canonic linear programming relaxation, which comprises all constraints and all variables, with relaxed integrality constraints. This would simply be too large.

The relaxed variables $x(e,t)$, $e \in E$, $t \in \{1,\ldots,T_e\}$, together with the ordering constraints are part of the used linear programming relaxation for the capacity model DISCRETE CAPACITIES, and similarly, the relaxed variables for the capacity model DIVISIBLE BASIC CAPACITIES are $x(e,\tau)$, $e \in E$, $\tau \in T$. Using only these variables, the initial relaxation yields a very weak lower bound, since the 0-vector is feasible for both capacity models. Therefore, additional variables and constraints are needed to obtain a relaxation with a stronger initial lower bound. The following relaxation is a compromise between the size in terms of variables and constraints, and the quality of the initial lower bound. It is based on the formulation for capacitated network design without survivability requirements presented in Section 2.3 (page 40).

Let $Q \subseteq V$ be an aggregation of the demands (each demand $uv \in D$ is assigned to one of its end-nodes in Q), and let $f^k(ij)$ and $f^k(ji)$ be edge-flow variables for all aggregated demands $k \in Q$ and all supply edges $e = ij \in E$. The relaxation is enlarged by these variables and the constraints are

$$\sum_{j\in V} f^k(ji) - \sum_{j\in V} f^k(ij) \;=\; \begin{cases} d_{uv}\,, & uv \in D, k=u, i=v\,, \\ 0\,, & \text{else}\,, \end{cases} \qquad k \in Q,\; i \in V, \tag{4.4}$$

$$\sum_{k\in Q} (f^k(ij) + f^k(ji)) \;\le\; y(e)\,, \qquad e = ij \in E\,, \tag{4.5}$$

$$y(e), f^k(ij), f^k(ji) \;\ge\; 0\,, \qquad k \in Q,\; e = ij \in E\,, \tag{4.6}$$

where

$$\begin{aligned} y(e) &= C_e^0 + \textstyle\sum_{t=1}^{T_e} c_e^t x(e,t)\,, && \text{for DISCRETE CAPACITIES}\,, \\ y(e) &= C_e^0 + \textstyle\sum_{\tau\in T} C^\tau x(e,\tau)\,, && \text{for DIVISIBLE BASIC CAPACITIES}\,. \end{aligned}$$

The node-set Q can be obtained in various ways. As one and only requirement, at least one of the two end-nodes of each demand $uv \in D$ must be contained in Q, that is, Q must define a node-cover of the demand graph. Obviously, a smaller node-cover leads to a smaller number of constraints and coefficients in the linear programming relaxation. However, since the problem of finding a minimal cardinality node-cover is an $\mathcal{NP}$-complete optimization problem, see (Garey and Johnson, 1979), a simple greedy heuristic is employed, which utilizes the degree of the nodes in the demand graph.

Notice that the variables $y(e)$, $e \in E$, are only auxiliary variables needed for notational convenience. These are not included in the relaxation. As we mentioned before, a feasible solution of this relaxation provides capacities which are feasible for the normal operating state without path-length restrictions and diversification constraints. Hence, Theorem 2.5 implies that these capacities satisfy all metric inequalities and all cut inequalities defined on the supply and demand graph of the normal operating state.

4.4 Feasibility of a capacity vector

In this section, we discuss the most time-consuming subproblem: solving the feasibility problem (see page 101) for a capacity vector $\overline{y}(e)$, $e \in E$, that is, deciding whether there exist routings for all demands in all operating states that satisfy the routing and survivability constraints. Depending on the survivability model, we could distinguish between the feasibility problems for

$$\overline{y} \in Y(G, H, \textsc{Div}), \quad \overline{y} \in Y(G, H, \textsc{Res}), \quad \overline{y} \in Y(G, H, \textsc{Path}),$$

but we consider two combinations only. One for the normal operating state and the other for the failure states. We distinguish between the two versions

$$\overline{y} \in Y(G, H, \textsc{Div}) \cap Y(G, H, \textsc{Res}) \quad \text{and} \quad \overline{y} \in Y(G, H, \textsc{Div}) \cap Y(G, H, \textsc{Path})$$

of the feasibility problem. The particular capacity model is not important for the feasibility problems. Only the absolute values of the capacities of the supply edges are important. However, these capacities are implicitly defined in dependence of the capacity model. For DISCRETE CAPACITIES with variables $\overline{x}(e,t), e \in E, t \in \{1, \dots, T_e\}$, or DIVISIBLE BASIC CAPACITIES with variables $\overline{x}(e,\tau), e \in E, \tau \in T$, the capacity vector is given by

$$\overline{y}(e) := C_e^0 + \sum_{t=1}^{T_e} c_e^t \overline{x}(e,t) \quad \text{or} \quad \overline{y}(e) := C_e^0 + \sum_{\tau \in T} C^\tau \overline{x}(e,\tau) \ ,$$

respectively. We formulate the feasibility problems as linear optimization problems in terms of path variables (see Section 2.3), and therefore, it is in principle possible to solve these problems with any commercial software package for linear programming. This approach, however, has a major drawback: the linear programs are extremely large since the number of path variables might be exponential in the size of the input. To solve these linear programs despite the number of variables, we adapt the column generation approach for continuous multicommodity-flow problems described in (Minoux, 1981). In the following sections, we present the formulations of the feasibility problems, the column generation procedure, implementation details, and results of computational experiments.

4.4.1 Formulation of the feasibility problems

We formulate both versions of the feasibility problem as linear optimization problems with an auxiliary variable α. This variable measures the minimum amount of additional capacity needed, on at least one of the supply edges, to make the capacity vector $\overline{y}$ feasible. The minimal value $\overline{\alpha}$ of this auxiliary variable will satisfy

$$\overline{y} \text{ feasible} \iff \overline{\alpha} \leq 0 \ .$$

DIVERSIFICATION **and** RESERVATION

In this case, the decision problem $\overline{y} \in Y(G, H, \text{DIV}) \cap Y(G, H, \text{RES})$ can be formulated as the following linear program:

$$
\begin{array}{rcll}
 & \min & \alpha & \\
\displaystyle\sum_{uv \in D_s} \sum_{P \in \mathcal{P}_s^{uv} : e \in P} f_s^{uv}(P) & \leq & \alpha + \overline{y}(e) \ , & s \in S,\ e \in E_s \ , \qquad (4.7) \\
\displaystyle\sum_{P \in \mathcal{P}_0^{uv}} f_0^{uv}(P) & = & d_{uv} \ , & uv \in D \ , \qquad (4.8) \\
\displaystyle\sum_{P \in \mathcal{P}_s^{uv}} f_s^{uv}(P) & = & \rho_{uv} d_{uv} \ , & s \in S \backslash \{0\},\ uv \in D_s \ , \qquad (4.9) \\
\displaystyle\sum_{P \in \mathcal{P}_0^{uv} : w \in P} f_0^{uv}(P) & \leq & \delta_{uv} d_{uv} \ , & uv \in D,\ w \in V \backslash \{u, v\} \ , \qquad (4.10) \\
f_0^{uv}(P) & \leq & \delta_{uv} d_{uv} \ , & uv \in D,\ P \in \mathcal{P}_0^{uv},\ |P| = 1 \ , \qquad (4.11) \\
f_s^{uv}(P) & \geq & 0 \ , & s \in S,\ uv \in D_s,\ P \in \mathcal{P}_s^{uv} \ . \qquad (4.12)
\end{array}
$$

Constraints (4.7) differ from those in the mathematical model in Chapter 2. Here, the capacity on a supply edge $e \in E$ is the sum of the constant value $\overline{y}(e)$ and the variable α. All other constraints are identical to those in the mathematical model. The linear program above has the block diagonal form illustrated in Figure 4.4.

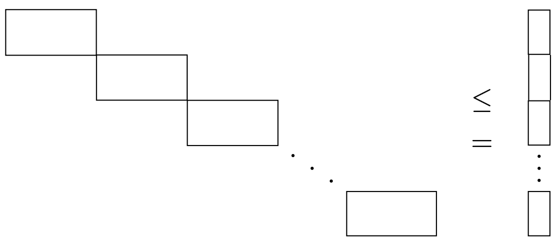


Figure 4.4: Structure of the feasibility LP (DIVERSIFICATION and RESERVATION)

Each block corresponds to one operating state, that is, each block in Figure 4.4 represents the subset of constraints for a particular operating state. Obviously, this feasibility problem decomposes with respect to the operating states and it suffices to solve the smaller feasibility problems for all operating states. The capacity $\overline{y}$ is feasible if and only if it is feasible for all operating states.

DIVERSIFICATION **and** PATH RESTORATION

In this case, the decision problem $\overline{y} \in Y(G, H, \text{DIV}) \cap Y(G, H, \text{PATH})$ can be formulated as the following linear program:

$$
\begin{aligned}
& & \min \alpha & & & (4.13)\\
\sum_{uv \in D} \sum_{P \in \mathcal{P}_0^{uv}: e \in P} f_0^{uv}(P) &\le \alpha + \overline{y}(e), & & e \in E, & & (4.14)\\
\sum_{uv \in D_s} \Big(\sum_{P \in \mathcal{P}_0^{uv} \cap \mathcal{P}_s^{uv}: e \in P} f_0^{uv}(P) + \sum_{P \in \mathcal{P}_s^{uv}: e \in P} f_s^{uv}(P)\Big) &\le \alpha + \overline{y}(e), & & s \in S \backslash \{0\},\ e \in E_s, & & (4.15)\\
\sum_{P \in \mathcal{P}_0^{uv}} f_0^{uv}(P) &= d_{uv}, & & uv \in D, & & (4.16)\\
\sum_{P \in \mathcal{P}_s^{uv}} (f_0^{uv}(P) + f_s^{uv}(P)) &\ge \sigma_{uv} d_{uv}, & & s \in S \backslash \{0\},\ uv \in D_s, & & (4.17)\\
\sum_{P \in \mathcal{P}_0^{uv}: w \in P} f_0^{uv}(P) &\le \delta_{uv} d_{uv}, & & uv \in D,\ w \in V \backslash \{u, v\}, & & (4.18)\\
f_0^{uv}(P) &\le \delta_{uv} d_{uv}, & & uv \in D,\ P \in \mathcal{P}_0^{uv},\ |P| = 1, & & (4.19)\\
f_s^{uv}(P) &\ge 0, & & s \in S,\ uv \in D_s,\ P \in \mathcal{P}_s^{uv}. & & (4.20)
\end{aligned}
$$

Figure 4.5 illustrates that the linear program above has again sort of a block diagonal form. However, this linear program is more complicated to solve since the failure states are linked with the normal operating state and, consequently, it is not sufficient to solve linear programs individually for all operating states.

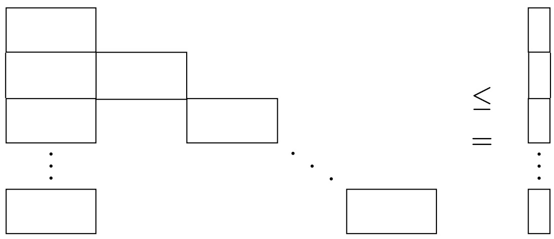


Figure 4.5: Structure of the feasibility LP (DIVERSIFICATION and PATH RESTORATION)

The first "row" in Figure 4.5 consists of the constraints (4.14) and (4.16). Each of the following "rows" consists of the constraints (4.15) and (4.17), which connect the normal operating state with the particular failure state. Notice that this linear program again decomposes into a sequence of smaller linear programs, if the normal operating state routing is fixed.

4.4.2 Solving the feasibility problems

The number of path variables is, even for moderately sized network design problems, quite large. Thus, it is not practical to solve the linear programs from scratch, that is, to enumerate the variables for all valid paths in all operating states for all demands, and to execute a commercial software package like CPLEX. As we already noted, we adapt the column generation approach described in (Minoux, 1981).

Suppose that the primal feasibility problem for a subset of the variables (valid paths) has been solved and $\overline{\alpha}$ is the optimal objective function value for this *restricted* problem. Obviously, the true optimal value of α (involving all path variables) is at most as large as the one obtained with the restricted number of variables, and thus $\overline{y}$ is feasible if $\overline{\alpha} \leq 0$. Otherwise, if $\overline{\alpha} > 0$, it might be necessary to identify missing path variables in order to find the optimal value of α.

For this purpose, all constraints of the complete dual program (including all path variables) must implicitly be evaluated for the dual variables of the optimal solution of the restricted primal problem. Optimality of $\overline{\alpha} > 0$ and hence infeasibility of $\overline{y}$ is proven, if all dual constraints are satisfied. In the other case, if the dual variables do not define a dual feasible solution, $\overline{\alpha}$ is not necessarily optimal for the primal problem including all path variables. If violated dual constraints can be identified, the associated primal path variables are added to the restricted primal problem and this enlarged linear program is resolved. The whole procedure is iterated until either $\overline{\alpha} \leq 0$, or no violated dual constraint can be identified. For the latter case there are two alternatives. Either the method to identify such dual constraints is *exact* and the infeasibility of $\overline{y}$ is proven, or the method is *heuristic* and it cannot be decided whether $\overline{y}$ is feasible.

DIVERSIFICATION and RESERVATION

As we observed, the feasibility problem for this combination of survivability models can be decomposed and it suffices to solve the feasibility problem for each operating state individually. Hence, let $s \in S$ be a fixed operating state, and let μ_e^s, for all supply edges $e \in E_s$, and π_{uv}^s, for all demand edges $uv \in D_s$, be the dual variables associated with constraints (4.7) and (4.8) or (4.9), respectively. The dual program for a particular failure state $s \in S\backslash\{0\}$ reads as follows:

$$
\begin{array}{llll}
\max & -\sum\limits_{e \in E_s} \overline{y}(e)\mu_e^s \;+\; \sum\limits_{uv \in D_s} \rho_{uv} d_{uv} \pi_{uv}^s & & (4.21) \\
& \sum\limits_{e \in P} \mu_e^s \;\geq\; \pi_{uv}^s, & uv \in D_s,\ P \in \mathcal{P}_s^{uv}, & (4.22) \\
& \sum\limits_{e \in E_s} \mu_e^s \;=\; 1, & & (4.23) \\
& \mu_e^s \;\geq\; 0, & e \in E_s\,. & (4.24)
\end{array}
$$

Constraints (4.22) are important. For those paths which are in the restricted primal problem, the corresponding dual constraints are satisfied. But how about the missing paths? All constraints (4.22) for a particular demand $uv \in D_s$ can be rewritten as

$$
\pi_{uv}^s \;\leq\; \min\left\{ \sum_{e \in P} \mu_e^s : P \in \mathcal{P}_s^{uv} \right\}.
$$

Thus, it suffices to calculate the shortest uv-path P^* in G_s. If

$$\pi^s_{uv} \quad > \quad \sum_{e \in P^*} \mu^s_e \ ,$$

then the associated dual constraint (4.22) is violated and, in this case, the column for the path variable $f^{uv}_s(P^*)$ must be added to the restricted primal formulation.

For the normal operating state, all paths must satisfy the path-length restriction, and the primal problem contains additionally the diversification constraints (4.10) and (4.11). Let γ^w_{uv}, for all demands $uv \in D$ and all nodes $w \in V \backslash \{u, v\}$, and γ^e_{uv}, for all demands $uv \in D$ and all supply edges $e \in \delta_G(u) \cap \delta_G(v)$, be the dual variables associated with constraints (4.10) and (4.11), respectively. Then the dual objective reads as

$$\max \qquad - \sum_{e \in E} \overline{y}(e)\, \mu_e + \sum_{uv \in D} d_{uv} \pi_{uv} - \sum_{uv \in D} \delta_{uv} d_{uv} \left(\sum_{e \in \delta_G(u) \cap \delta_G(v)} \gamma^e_{uv} + \sum_{w \in V \backslash \{u,v\}} \gamma^w_{uv} \right) .$$

Furthermore, for every demand $uv \in D$ and $P = \{e\} \in \mathcal{P}^{uv}_0$ with $e \in \delta_G(u) \cap \delta_G(v)$, the constraints (4.22) change to

$$\gamma^e_{uv} + \mu_e \geq \pi_{uv} \ , \tag{4.25}$$

and for all other $P \in \mathcal{P}^{uv}_0$ to

$$\sum_{w \in P} \gamma^w_{uv} + \sum_{e \in P} \mu_e \geq \pi_{uv} \ . \tag{4.26}$$

Similar arguments as above show how to identify violated dual constraints and therefore missing path variables. In the normal operating state, for every demand $uv \in D$ a length-restricted shortest path with respect to the following supply edge weights must be calculated:

$$w_e \quad := \mu_e + \begin{cases} \gamma^e_{uv} \ , & u = v_1, v = v_2, \text{ or } u = v_2, v = v_1 \ , \\ \frac{1}{2}(\gamma^{v_1}_{uv} + \gamma^{v_2}_{uv}) \ , & \text{else} \ , \end{cases} \tag{4.27}$$

for all $e = v_1 v_2 \in E$.

Remark 4.5 *If the capacity vector $\overline{y}$ is infeasible for a failure state $s \in S \backslash \{0\}$, in which case the optimal value α^* of the primal feasibility problem is strictly positive, the inequality*

$$\sum_{uv \in D_s} \rho_{uv} d_{uv} \pi^s_{uv} \quad > \quad \sum_{e \in E_s} \overline{y}(e) \mu^s_e$$

is satisfied. Thus, the dual objective function induces a metric inequality (3.2) that is violated by the capacity vector $\overline{y}$. In other words, the algorithm to solve the feasibility problem for $\overline{y}$ is also a separation algorithm for metric inequalities. The same is true for the normal operating state.

DIVERSIFICATION **and** PATH RESTORATION

As before, suppose that dual variables $\mu_e, e \in E$, $\mu_e^s, s \in S\backslash\{0\}, e \in E_s$, $\pi_{uv}, uv \in D$, $\pi_{uv}^s, s \in S\backslash\{0\}; uv \in D_s$, $\gamma_{uv}^w, uv \in D, w \in V\backslash\{u,v\}$, and $\gamma_{uv}^e, uv \in D, e \in \delta_G(u) \cap \delta_G(v)$, are associated with constraints (4.14), (4.15), (4.16), (4.17), (4.18), and (4.19), respectively. Furthermore, suppose that $\gamma_{uv}(P)$ denotes the sum of those dual variables associated with inner nodes and edges in the path $P \in \mathcal{P}_0^{uv}$ between the end-nodes of demand $uv \in D$. That is, $\gamma_{uv}(P) = \sum_{w \in P} \gamma_{uv}^w$, if $|P| > 1$, and $\gamma_{uv}(P) = \gamma_{uv}^e$, if $P = \{e\}$ for $e \in \delta_G(u) \cap \delta_G(v)$. Then, the dual linear program is the following:

$$\max \sum_{uv \in D} d_{uv}\pi_{uv} - \sum_{e \in E} \overline{y}(e)\mu_e - \sum_{uv \in D} \delta_{uv} d_{uv} \left(\sum_{e \in \delta_G(u) \cap \delta_G(v)} \gamma_{uv}^e + \sum_{w \in V\backslash\{u,v\}} \gamma_{uv}^w \right) + \sum_{s \in S\backslash\{0\}} \sum_{uv \in D_s} \sigma_{uv} d_{uv} \pi_{uv}^s - \sum_{s \in S\backslash\{0\}} \sum_{e \in E_s} \overline{y}(e)\mu_e^s$$

$$\pi_{uv}^s - \sum_{e \in P} \mu_e^s \le 0 \,,\; s \in S\backslash\{0\},\; uv \in D_s,\; P \in \mathcal{P}_s^{uv} \,, \tag{4.28}$$

$$\gamma_{uv}(P) + \pi_{uv} - \sum_{e \in P} \mu_e + \sum_{s \in S\backslash\{0\} : s \notin P} \left(\pi_{uv}^s - \sum_{e \in P} \mu_e^s \right) \le 0 \,,\; uv \in D,\; P \in \mathcal{P}_0^{uv} \,, \tag{4.29}$$

$$\sum_{s \in S\backslash\{0\}} \sum_{e \in E_s} \mu_e^s + \sum_{e \in E} \mu_e = 1 \,, \tag{4.30}$$

$$\mu_e \ge 0 \,,\; e \in E \,, \tag{4.31}$$

$$\mu_e^s \ge 0 \,,\; s \in S\backslash\{0\},\; e \in E_s \,. \tag{4.32}$$

The dual constraints (4.28), (4.29) and (4.30) are associated with the primal variables $f_s^{uv}(P)$, $f_0^{uv}(P)$ and α, respectively. Again, the primal solution is optimal if the dual variables define a dual feasible solution. That is, to prove optimality of the primal solution it is necessary to implicitly evaluate the dual constraints for all path variables. By linear programming duality, the non-negativity constraints (4.31) and (4.32), as well as the 'α' constraint (4.30), are satisfied. Therefore, it remains to solve the "separation problem" for constraints (4.28) and (4.29).

The separation problem for constraints (4.28) is exactly solvable. For each failure state $s \in S\backslash\{0\}$ and each demand $uv \in D_s$, it reduces to computing a shortest uv-path in G_s w.r.t. non-negative edge weights μ_e^s. If the dual constraint (4.28) for the shortest uv-path $P^* \in \mathcal{P}_s^{uv}$ is violated, the variable $f_s^{uv}(P^*)$ must be added to the primal linear program. Since this is an exact method, we can assume in the following that all constraints (4.28) are satisfied.

The separation problem for the constraints (4.29) is decomposable into a sequence of separation problems: one problem for each demand $uv \in D$. However, we are not aware of an efficient method that solves these problems exactly.

Proposition 4.6 *Let $uv \in D$, and assign a weight w_e to each supply edge $e \in E$ according to* (4.27)*. Suppose that $P^* \in \mathcal{P}_0^{uv}$ is a shortest path-length restricted uv-path in G w.r.t. these edge weights. If $\pi_{uv} \leq \sum_{e\in P^*} w_e$, then the dual constraints* (4.29) *are satisfied for this demand uv.*

Proof. Let P be a path-length restricted uv-path in G. The associated dual constraint (4.29) is satisfied, because of

$$\pi_{uv} \;\leq\; \sum_{e\in P^*} w_e \;\leq\; \sum_{e\in P} w_e \;\leq\; \gamma_{uv}(P) + \sum_{e\in P}\mu_e - \sum_{s\in S\setminus\{0\}:s\notin P} \underbrace{(\pi^s_{uv} - \sum_{e\in P}\mu^s_e)}_{\leq\,0} \;.$$

□

Proposition 4.6 provides a sufficient criterion to state that for a demand edge $uv \in D$ there is no violated constraint (4.29). Next, we provide a sufficient criterion to state violation, which is based on a shortest path with respect to the following supply edge weights. For every demand $uv \in D$ and every supply edge $e = v_1v_2 \in E$, let the edge weight w_e be defined by

$$w_e \;:=\; \mu_e + \pi^e_{uv} + \sum_{s\in S\setminus\{0\}:e\in E_s} \mu^s_e + \begin{cases} \frac{1}{2}(\pi^{v_2}_{uv} + \gamma^{v_2}_{uv})\;, & v_1 = u,\; v_2 \neq v\;,\\ \frac{1}{2}(\pi^{v_1}_{uv} + \gamma^{v_1}_{uv})\;, & v_2 = v,\; v_1 \neq u\;,\\ \gamma^e_{uv}\;, & v_1 = u,\; v_2 = v\;,\\ \frac{1}{2}(\pi^{v_1}_{uv} + \pi^{v_2}_{uv} + \gamma^{v_1}_{uv} + \gamma^{v_2}_{uv})\;, & \text{else}\;,\end{cases}$$

Proposition 4.7 *Let $uv \in D$ and $P \in \mathcal{P}_0^{uv}$ be a shortest uv-path in G w.r.t. the edge weights w_e, $e \in E$, defined above. If*

$$\pi_{uv} + \sum_{s\in S\setminus\{0\}:uv\in D_s} \pi^s_{uv} \;>\; \sum_{e\in P} w_e,$$

then the dual constraint (4.29) *for demand edge $uv \in D$ and path $P \in \mathcal{P}_0^{uv}$ is violated.*

Proof. Follows immediately from

$$\begin{aligned}
w(P) \;&=\; \sum_{e\in P} w_e\\
&=\; \sum_{e\in P}\mu_e + \gamma_{uv}(P) + \sum_{e\in P}\pi^e_{uv} + \sum_{w\in P}\pi^w_{uv} + \sum_{e\in P}\sum_{s\in S\setminus\{0\}:e\in E_s}\mu^s_e\\
&=\; \sum_{e\in P}\left(\mu_e + \sum_{s\in S\setminus\{0\}:e\in E_s}\mu^s_e\right) + \gamma_{uv}(P) + \sum_{s\in P}\pi^s_{uv}\\
&=\; \sum_{e\in P}\left(\mu_e + \sum_{s\in S\setminus\{0\}:e\in E_s}\mu^s_e\right) + \gamma_{uv}(P) + \sum_{s\in P}\pi^s_{uv} + \sum_{s\in S\setminus\{0\}:uv\in D_s}(\pi^s_{uv} - \pi^s_{uv})\\
&\geq\; \sum_{e\in P}\mu_e + \sum_{s\in S\setminus\{0\}:s\notin P}\sum_{e\in P}\mu^s_e + \gamma_{uv}(P) - \sum_{s\in S\setminus\{0\}:s\notin P}\pi^s_{uv} + \sum_{s\in S\setminus\{0\}:uv\in D_s}\pi^s_{uv}
\end{aligned}$$

□

4.4.3 Implementation issues

There are several subproblems to discuss for an efficient implementation of an algorithm to test the feasibility of a capacity vector. The important issues for the column generation are the initialization of the restricted problem, the algorithm to solve the shortest-path problem with path-length restrictions, and the column generation strategy. Furthermore, we discuss in this section the order of the operating states in which the feasibility problems for the individual operating states are solved. The computational experiments revealed that solving these linear programs with column generation belongs to the most time-consuming parts of the overall algorithm. Thus, there is need for alternative criteria to decide the feasibility of a capacity vector in a single operating state which are faster to evaluate than the linear programs. We present some of these criteria at the end of this section.

Initialization of the linear program

The respective linear programs used to solve the feasibility problems are initialized with a small subset of the path variables. For the running time of the column generation algorithm, it is crucial to find the the right balance between the number of added path variables and the number of iteration steps needed to generate the missing columns. We aim at two targets.

- The initial path variables must guarantee the existence of a solution for the *initial* restricted problem.

- The number of "unnecessary" reoptimizations due to missing path variables in the restricted problem formulation should be small.

It is difficult to accomplish the first target for the normal operating state, since an $\mathcal{NP}$-complete subproblem must be solved. Recall, the problem of finding the maximum number of path-length restricted node-disjoint paths between two nodes is $\mathcal{NP}$-complete (see (Itai et al., 1982)). For each demand $uv \in D$, the initial set of paths is computed with Algorithm 1, and different strategies to accomplish the second target are evaluated in Section 4.4.4.

Algorithm 1 INITIALPATHVARIABLES

Require: $uv \in D$
  $k := \max\{\lceil \frac{1}{\delta_{uv}} \rceil, \lceil \rho_{uv} \rceil + 1\}$
  if $\ell_{uv} \geq |V| - 1$ **then**
    k-node-disjoint path algorithm (see (Suurballe, 1974))
  else if $\ell_{uv} \leq 4$ **then**
    exact algorithm (see (Itai et al., 1982))
  else
    heuristic framework (see (Bley, 1997))
  end if

Shortest path problem with path-length restrictions

One auxiliary problem of the column generation procedure is the shortest path problem with path-length restrictions. The problem is defined as follows. Given a graph $G = (V, E)$, a node $u \in V$, weights $\mu_e \in \mathbb{R}_+$ and lengths $\lambda_e \in \mathbb{R}_+$ for each edge $e \in E$, find the minimum-weight path of length at most ℓ_{uv} from u to every other node $v \in V$. The problem of deciding whether a path of weight at most M and length at most L exists between two specified nodes is $\mathcal{NP}$-complete. However, it is polynomially solvable if all weights or all lengths are equal; see (Garey and Johnson, 1979). In our case, $\lambda_e = 1$ for all $e \in E$, and thus the problem is polynomially solvable.

Algorithm 2 PathLengthRestrictedShortestPath

> $U := V - \{u\}, R := \{v \in V : (u, v) \in E\}$
> **for all** $v \in V$ **do**
> $\quad d_1(v) := \begin{cases} 0 & \text{if } v = u \\ \mu_e & \text{if } e = (u, v) \in E \\ \infty & \text{otherwise} \end{cases}$
> **end for**
> **while** $R \neq \emptyset$ **do**
> $\quad$ find $v \in R$ and $1 \leq i \leq \ell$ such that $\forall w \in R, 1 \leq j \leq \ell$
> $\quad\quad$ (i) $d_i(v) \leq d_j(w)$ and
> $\quad\quad$ (ii) $d_i(v) = d_j(w) \Rightarrow j \geq i$
> $\quad U := U - \{v\}, R := R - \{v\}$
> $\quad$ **for all** neighbors $w \in U$ **do**
> $\quad\quad$ **for all** j with $i + 1 \leq j \leq \ell$ **do**
> $\quad\quad\quad d_j(w) = \min\{d_j(w), d_{j-1}(v) + \mu_{(v,w)}\}$
> $\quad\quad$ **end for**
> $\quad\quad$ **if** $d_\ell(w) < \infty$ **then**
> $\quad\quad\quad R := R \cup \{w\}$
> $\quad\quad$ **end if**
> $\quad$ **end for**
> **end while**

Algorithm 2 shows the modification of the well-known algorithm presented in (Dijkstra, 1959), used to solve the shortest path problem with path-length restrictions. The target is a shortest path subtree such that each node in the tree can be reached from u on a path with at most $\ell := \min\{\ell_{uv} : uv \in D\}$ supply edges. In every iteration of the algorithm, a shortest path that satisfies the path-length restriction is determined for exactly one node. This node will be called *labeled*. The algorithm terminates, if all nodes are labeled that can be reached from u on a path with at most ℓ edges.

In more detail, for every $v \in V$, let $d_i(v)$ be the shortest distance from u to v using at most i edges in the current iteration of the algorithm. Denote by U the set of all unlabeled nodes and define $R := \{v \in U : \exists k \leq \ell \text{ with } d_k(v) < \infty\} \subseteq U$. That is, R is the set of unlabeled nodes that can be reached from u using only labeled intermediate nodes. Initially, set $U := V - \{u\}$. In each iteration of the algorithm, the node with the smallest distance from u is labeled, breaking ties by selecting the one corresponding to the path with the fewest edges. At the end of each iteration, the distance labels $d_i(v)$ of its unlabeled neighbors are updated. At the end of the algorithm, a length-restricted shortest paths from u to $v \in V \setminus \{u\}$ can be easily determined by keeping track of the predecessors for each $v \in V$ and each $1 \leq i \leq \ell$.

Column generation strategies

Suppose that the feasibility problem for a particular operating state must be solved. As we noted in Section 4.4.2, the primal feasibility problem has not yet been solved to proven optimality, whenever a violated dual constraint has been identified, In this case, it is necessary to add columns corresponding to path variables, and to reoptimize the primal problem. The number of necessary reoptimizations depends obviously on the way the new columns are generated. The question arises, how many missing path variables should be added for how many demands in a single iteration. No computational experiments for this runtime parameter are presented in Section 4.4.4, we only note that it turned out to be advantageous to solve the shortest path problem (with path-length restrictions in the normal operating state) for *all* surviving demands in the particular operating state and to add all identified path variables corresponding to dual constraints which are violated by the computed shortest paths.

Order of operating states

As described in Section 4.4.2, the algorithm to solve the feasibility problem for a capacity vector is based on solving feasibility (sub-)problems for individual operating states. The number of solved subproblems depends on the sequence in which the operating states are considered. In more detail, the subproblems of *all* operating states must be solved, if a capacity vector is feasible. However, if the capacity vector is *not* feasible, that is, if it is not feasible for some operating states, the algorithm terminates as soon as the first infeasible operating state has been determined. Hence, efficient implementations of this algorithm should manipulate the sequence of considered operating states such that infeasibility is determined as soon as possible. For this purpose, the (adaptive) sequences described in the next paragraph are compared in Section 4.4.4.

Suppose that a capacity vector $\overline{y}$ and a list of operating states S are given. The operating states are increasingly ordered with respect to weights $w(s) \in \mathbb{Q}_+$, $s \in S$, which represent a measure of the feasibility of the particular operating state. To achieve the described effect that infeasible operating states are more likely to be considered early, these weights are frequently updated. The weights are initialized by setting $w(s) := 0$, for all $s \in S$. As initial sequence, the normal operating state is first, followed by all nodes $v \in S$ and all supply edges $e \in S$. The initial order of the nodes and edges is arbitrary.

- ARO Set $w(s) := w(s) + 1$, if s is infeasible.
- ASO Set $w(s) := w(s) + 1$, if s is feasible and $w(s) := w(s) + 2$, if s is infeasible.
- DEF Set $w(s) := w(s)$.

ARO is called adaptive reuse order since the weight of an infeasible operating state is not changed, implying that this operating state will be reused in the next application of the algorithm to solve the feasibility problem. In contrast, ASO is called adaptive sequential order since the weights of infeasible operating states are increased as well, implying that operating states with equal weight (such as all operating states after the initialization of the weights) are considered in sequence. Only if there are no other operating states with equal weight, an infeasible operating state will be reused in the next application of the algorithm to solve the feasibility problem. Finally, the default order DEF maintains the initial weights. Since the weights are never changed, the order does not change as well.

Alternative criteria to decide feasibility of the capacity vector

Solving the linear programs with column generation is the most time-consuming part of the overall algorithm since it is frequently necessary to test the feasibility of a capacity vector and since the respective linear programs are considerably large. Thus, whenever possible other criteria should be applied to determine feasibility or infeasibility of a capacity vector in all or some operating states. In the following, we present some simple criteria that can be applied for this decision.

- Suppose that a feasible capacity vector $c(e)$, $e \in E$, is given. A capacity vector $\overline{y}$ is *feasible*, if $\overline{y}(e) \geq c(e)$ for all $e \in E$. Furthermore, $\overline{y}$ is *infeasible*, if c is a minimal feasible capacity vector with respect to the componentwise order and if a supply edge $g \in E$ with $\overline{y}(g) < c(g)$ and $\overline{y}(e) = c(e)$ for all $e \in E \backslash \{g\}$ exists. It is worth mentioning that these trivial tests speed up the improvement heuristics.
- Suppose that $\overline{y}$ is feasible for the normal operating state, and let f_0 be a corresponding feasible routing for this operating state. Furthermore, for $e \in E$, $w \in V$, and $uv \in D$, set

$$\begin{array}{rclcrcl} f_0^{uv}(e) & := & \sum_{P \in \mathcal{P}_0^{uv}: e \in P} f_0^{uv}(P) \;, & \qquad & f_0(e) & := & \sum_{uv \in D} f_0^{uv}(e) \;, \\ f_0^{uv}(w) & := & \sum_{P \in \mathcal{P}_0^{uv}: w \in P} f_0^{uv}(P) \;, & \qquad & f_0(w) & := & \sum_{uv \in D} f_0^{uv}(w) \;. \end{array}$$

  That is, $f_0(e)$ and $f_0(w)$ are the flow through $e \in E$ and $w \in V$, respectively, while $f_0^{uv}(e)$ and $f_0^{uv}(w)$ are the respective flow values for a particular demand $uv \in D$. For $e \in E$ and $w \in V$ the respective values $f_0(e)$ and $f_0(w)$ are easily calculated from (4.7) or (4.14) by

$$f_0(e) := \alpha + \overline{y}(e) - \text{slack}(e) \quad \text{and} \quad 2 \cdot f_0(w) := \sum_{e \in \delta_G(w)} f_0(e) - \sum_{uw \in D} d_{uw} \;,$$

  where slack(e) is the slack of constraint (4.7) or (4.14) for supply edge e. Furthermore,

$$f_0^{uv}(w) = \delta_{uv} d_{uv} - \text{slack}(uv, w) \;,$$

  can be obtained from (4.18) for all demands $uv \in D$ with $\delta_{uv} < 1$ and all $w \in V \backslash \{u, v\}$, where slack$(uv, w)$ is the slack of constraint (4.18). For $f_0^{uv}(e)$, $e \in E$, there is no fast computation. Now, $\overline{y}$ is feasible for an operating state $s = e \in E \cap S$ if

$$f_0(e) \quad \leq \quad (1 - \rho_{uv}) d_{uv} \;, \quad uv \in D \;, \tag{4.33}$$

  and feasible for an operating state $s = w \in V \cap S$ if

$$f_0^{uv}(w) \quad \leq \quad (1 - \rho_{uv}) d_{uv} \;, \quad uv \in D \;. \tag{4.34}$$

4.4.4 Computational tests

We report in this section on computational experiments with different parameter selections for the algorithms solving the feasibility problem. As main performance measures,

- the accumulated time to solve the feasibility problems,
- the number of linear programs solved, and
- the size of the linear programs in terms of columns and coefficients

are used. The tests are performed for the problem instances m1, l1, and p1 with survivability parameters $\delta = 0.5$ and $\delta = 0.75$ for DIVERSIFICATION, and $\rho = 0.5$ and $\rho = 1.0$ for RESERVATION and PATH RESTORATION. The respective problem names associated with these parameters are m1d50, m1d75, l1d50, l1d75, p1d50, and p1d75, for DIVERSIFICATION, and m1r50, m1r100, l1r50, l1r100, p1r50, and p1r100 for RESERVATION and PATH RESTORATION. It is not necessary to consider different problem instances for the two capacity models, since the feasibility problem is independent from the particular model. Only absolute capacity values are important and not how a particular capacity has been obtained.

It is difficult to choose a good testing environment, since the performance of the algorithms to solve the feasibility problem depends on the applications using this problem as a subproblem. In particular, the sequence of capacity vectors tested for feasibility matters, and this sequence depends on other algorithms applied, such as separation algorithms, or starting and improvement heuristics. Further difficulties arise since the employed linear programs are also used within separation algorithms for metric inequalities. (We already indicated this relation in Remark 4.5 and we will further discuss this in Section 4.5.) To provide an environment that *only* evaluates the influence of different run-time parameter selections for the algorithms solving the feasibility problem, the following tests are performed on a fixed complete sequence of algorithms. (This sequence comprises the lower bound calculation, and a fixed subset of all starting and improvement heuristics.) Furthermore, a special implementation has been used which ensures that the linear programs used for solving the feasibility problem are independent from other algorithms such as separation algorithms for metric inequalities.

The computational results are presented in Table 4.2 which has the following format. Column 1 (NAME) gives the name of the problem instance which also encodes the survivability parameters. Columns 2 and 3 provide the selection of run-time parameters. In more detail, column 2 (ORD) gives the applied strategy in which the operating states are ordered. The considered alternatives are the adaptive reuse ordering ARO, the adaptive sequential ordering ASO, and the default ordering DEF. Column 3 (INI) encodes the initial set of path variables for the related linear programs. This initial set always contains a set of node-disjoint length-restricted paths, ensuring a feasible routing using only these paths if sufficiently large capacities are installed on the supply edges. NDB encodes that such a minimal set of paths is used, while NDBTWO and NDBTHREE encode that all variables corresponding to paths up to length 2 and 3, respectively, are added to the initial linear programs. Five to seven columns with statistics are following for each of the three survivability models. From left to right, these are a subset of the accumulated time to solve the feasibility problems (TIME), the number of linear programs solved (LP), the accumulated number of operating states decided by one of the alternative criteria described in the previous section (ALT), the number of rows in the final linear programs (ROWS), the number of columns in the final linear programs (COLS), and the number of non-zero coefficients in the final linear programs (COEFFS). For Reservation, there are two times the columns COLS and COEFFS. The reason is that two independent linear programs are needed for Reservation. One for the normal operating state (including path-length restrictions) and the other for the failure states.

The main observation is the following: The running-times strongly depend on the initialization of the linear programs used to solve the feasibility problems. In most cases, the times for NDB are smaller than those for NDBTWO, which are themselves much smaller than those of NDBTHREE. In fact, the times for NDBTHREE are orders of magnitudes worse than those of the other two alternatives. Considering the p1 instances, we observe that the times for NDB are between 20 and 50 times smaller than those for NDBTHREE, and even worse, for Path Restoration it is impossible to compute a single solution within 48 hours since the intermediate linear programs are extremely difficult to solve. Summarizing, the column generation algorithm is powerful enough to compute the missing columns. It pays to start with a small set of columns and to let the algorithm decide which ones to add. This makes the difference between finding a solution or not.

Closely related to the previous observations, the final linear programs are surprisingly small. For NDB, the best initialization strategy, the number of path variables is approximately 1000 for Diversification, approximately 10000 for Reservation, and between 20000 and 250000 for Path Restoration. according to Table 4.1, there are between 4 million and 5 billion path variables for Diversification, and between 100 billion and 20000 billion path variables for the survivability models Reservation and Path Restoration. In other words, in some cases fewer than 10^{-6} percent of the path variables are needed.

In general, for fixed initialization of the linear programs, the running times are almost independent from the order in which the operating states are considered. The number of successful decisions due to alternative criteria only matters for the survivability Reservation. For this case, we can observe from Table 4.2 that these are independent from both the initialization of the linear programs and the order of operating states. Alternative criteria apply for the l1 and m1 instances for approximately $1/4$ of the decisions, and for approximately $1/2 - 2/3$ of the decisions for the p1 instances. Finally, we observe that the number of solved linear programs is almost independent from the considered parameters.

In the following, the operating states are solved in the order defined by ARO and the linear programs are initialized with NDB.

NAME	ORD	INI	DIVERSIFICATION TIME	LP	COEFFS	COLS	ROWS	RESERVATION TIME	LP	ALT	COEFFS	COLS	COEFFS	COLS	PATH RESTORATION TIME	LP	COEFFS	COLS	ROWS
m1 d50/r50	ARO	NDBTHREE	50	173	13781	1791	996	53	**1030**	**293**	5594	1332	7962	1732	5:39:56	115	392854	22412	7394
	ARO	NDBTWO	52	186	11953	1354	1061	31	**1030**	**293**	3291	718	5106	1061	3:27:14	130	216927	22078	7394
	ARO	NDB	**22**	**149**	13769	1491	1078	**29**	**1030**	**293**	3617	744	5030	1021	25:17	**77**	152840	16399	7394
	ASO	NDBTHREE	43	173	13781	1791	996	53	**1030**	**293**	5594	1332	7962	1732	5:39:38	132	392854	22412	7394
	ASO	NDBTWO	40	186	11953	1354	1061	31	**1030**	**293**	3291	718	5106	1061	2:37:51	104	216927	22078	7394
	ASO	NDB	25	**149**	13769	1491	1078	**29**	**1030**	**293**	3617	744	5030	1021	18:58	79	151475	16336	7394
	DEF	NDBTHREE	47	173	13781	1791	996	53	1037	274	5603	1333	7854	1715	5:27:17	125	392854	22412	7394
	DEF	NDBTWO	39	186	11953	1354	1061	32	1037	274	3342	723	5238	1083	2:37:30	104	216927	22078	7394
	DEF	NDB	**22**	**149**	13769	1491	1078	30	1037	274	3708	757	4944	1006	**18:48**	79	151475	16336	7394
m1 d75/r100	ARO	NDBTHREE	36	203	9985	1487	848	2:09	1949	515	4215	1104	5764	1361	3:37:57	65	353136	23312	7394
	ARO	NDBTWO	37	**190**	7813	977	931	**1:15**	1948	516	1881	489	3698	814	4:24:13	135	161094	19971	7394
	ARO	NDB	34	**190**	9509	1110	1020	1:16	**1942**	515	1890	453	3809	806	48:01	86	132582	16965	7394
	ASO	NDBTHREE	34	203	9985	1487	848	2:37	2174	615	4236	1108	5807	1371	5:05:34	**44**	334384	19949	7394
	ASO	NDBTWO	**28**	**190**	7813	977	931	1:29	2171	**618**	1878	489	3693	810	4:21:32	141	161094	19971	7394
	ASO	NDB	33	**190**	9509	1110	1020	1:28	2166	616	1890	453	3893	822	31:23	97	112551	16384	7394
	DEF	NDBTHREE	35	203	9985	1487	848	2:42	2154	392	4269	1114	5899	1382	5:10:39	46	334384	19949	7394
	DEF	NDBTWO	30	**190**	7813	977	931	1:32	2153	392	1851	483	3634	800	4:24:11	130	161094	19971	7394
	DEF	NDB	36	**190**	9509	1110	1020	1:37	2144	395	1943	457	3819	806	**31:03**	97	112551	16384	7394
l1 d50/r50	ARO	NDBTHREE	12:40	443	35485	3378	1991	12:11	4528	1830	9526	2003	13145	2532	22:45:42	87	1360239	64832	26305
	ARO	NDBTWO	11:06	427	28047	2499	1971	10:25	4527	1825	8330	1437	12311	2070	10:16:41	71	720415	54813	26305
	ARO	NDB	**7:33**	**424**	30091	2547	2055	**8:23**	**4509**	1838	8641	1452	10461	1775	**2:05:46**	**66**	627268	45861	26305
	ASO	NDBTHREE	9:15	443	35485	3378	1991	12:48	4611	1880	9526	2003	13145	2532	22:40:09	87	1360239	64832	26305
	ASO	NDBTWO	8:21	427	28047	2499	1971	10:27	4610	1875	8330	1437	12311	2070	10:55:28	71	720415	54813	26305
	ASO	NDB	7:54	**424**	30091	2547	2055	8:31	4592	1888	8641	1452	10461	1775	3:53:53	**66**	571135	45785	26305
	DEF	NDBTHREE	8:49	443	35485	3378	1991	21:30	6516	2889	9384	1979	13164	2530	22:39:29	87	1360239	64832	26305
	DEF	NDBTWO	8:11	427	28047	2499	1971	20:44	6520	2879	8331	1437	12180	2043	10:46:10	71	720415	54813	26305
	DEF	NDB	8:00	**424**	30091	2547	2055	14:36	6483	**2911**	8289	1395	10456	1773	3:50:36	**66**	571135	45785	26305
l1 d75/r100	ARO	NDBTHREE	4:15	**327**	20573	2436	1619	21:06	5109	1983	7238	1657	15263	2725	16:46:40	39	1272028	62490	26305
	ARO	NDBTWO	3:58	384	17925	1777	1670	**14:37**	**5061**	2020	4040	821	12371	2032	36:04:43	**38**	692517	59828	26305
	ARO	NDB	3:25	380	20007	1968	1683	16:43	5149	1998	4948	911	13909	2215	**2:47:32**	75	544108	45751	26305
	ASO	NDBTHREE	3:12	**327**	20573	2436	1619	21:31	5385	2210	7367	1670	15151	2701	16:46:56	39	1272028	62490	26305
	ASO	NDBTWO	**3:03**	384	17925	1777	1670	16:42	5342	**2238**	4297	851	12949	2110	36:15:44	**38**	692517	59828	26305
	ASO	NDB	3:47	380	20007	1968	1683	16:55	5292	2153	4807	887	13275	2145	3:26:20	68	509916	45716	26305
	DEF	NDBTHREE	3:24	**327**	20573	2436	1619	20:27	5286	1669	7418	1691	15267	2732	17:08:07	39	1272028	62490	26305
	DEF	NDBTWO	3:19	384	17925	1777	1670	16:14	5240	1687	4125	833	12311	2017	35:26:13	**38**	692517	59828	26305
	DEF	NDB	3:18	380	20007	1968	1683	16:49	5078	1613	4967	916	13382	2156	3:21:23	69	509916	45716	26305
p1 d50/r50	ARO	NDBTHREE	19:05	273	106545	17748	842	1:18:42	1542	2953	67305	17224	69093	17504	⋆[a]				
	ARO	NDBTWO	53	**231**	12795	2296	818	3:30	1542	2953	5740	1801	8905	2305	⋆				
	ARO	NDB	57	262	12921	1520	874	**1:40**	**1530**	**2959**	3763	947	5073	1122	35:30	**83**	822811	246668	28230
	ASO	NDBTHREE	14:13	273	106545	17748	842	1:15:45	1542	2953	67305	17224	69093	17504	⋆				
	ASO	NDBTWO	49	**231**	12795	2296	818	3:31	1542	2953	5740	1801	8905	2305	⋆				
	ASO	NDB	56	262	12921	1520	874	**1:40**	**1530**	**2959**	3763	947	5073	1122	9:11	93	892984	246764	28230
	DEF	NDBTHREE	15:10	273	106545	17748	842	1:28:23	1542	2953	67305	17224	69093	17504	⋆				
	DEF	NDBTWO	**48**	**231**	12795	2296	818	3:18	1541	2954	5740	1801	8922	2307	⋆				
	DEF	NDB	58	262	12921	1520	874	1:41	**1530**	**2959**	3753	941	5073	1122	**9:05**	93	892984	246764	28230
p1 d75/r100	ARO	NDBTHREE	10:52	**209**	102561	17461	624	2:51:40	2987	3831	67200	17204	68790	17473	⋆				
	ARO	NDBTWO	48	**209**	11845	2227	635	10:52	3046	4276	5196	1691	9202	2428	⋆				
	ARO	NDB	25	**209**	7453	1019	814	8:24	**2894**	4214	2307	637	8036	1737	22:10:39	308	929046	246965	28230
	ASO	NDBTHREE	10:46	**209**	102561	17461	624	3:02:36	3132	4257	67200	17204	68740	17462	⋆				
	ASO	NDBTWO	47	**209**	11845	2227	635	10:39	3083	**4480**	5196	1691	8912	2389	⋆				
	ASO	NDB	**24**	**209**	7453	1019	814	**8:08**	**2894**	4317	2372	659	7984	1740	**50:18**	**123**	853108	246704	28230
	DEF	NDBTHREE	10:12	**209**	102561	17461	624	3:24:17	3436	3626	67200	17204	68800	17474	⋆				
	DEF	NDBTWO	46	**209**	11845	2227	635	12:00	3363	3787	5536	1759	8799	2354	⋆				
	DEF	NDB	25	**209**	7453	1019	814	10:18	3288	3820	3064	779	7874	1703	51:39	**123**	853108	246704	28230

Table 4.2: Computational results for the feasibility problem

[a] A ⋆ symbolizes that no solution has been identified within 48 hours of CPU time

4.5 Separation algorithms

In Chapter 3, we presented polyhedra associated with different models for survivable capacitated network design problems. For each polyhedron, we listed known and developed new classes of valid or even facet-defining inequalities. In principle, all these inequalities are useful to strengthen the linear programming relaxation, but it is necessary to devise for each individual class of inequalities an algorithm which solves the separation problem (see Definition 4.2). In this section, we describe our separation algorithms.

The classes of valid inequalities for the polyhedra $X(G, H, \text{FIN}, \cdot)$ and $X(G, H, \text{BAS}, \cdot)$ are based on valid inequalities for the corresponding polyhedra $Y(G, H, \cdot)$. In Sections 3.2 and 3.3, we derived valid inequalities for the knapsack-relaxations induced by a valid inequality $\mu y \geq d$ for $Y(G, H, \cdot)$ (see Definitions 3.5 and 3.42). Let $\mathcal{C}(\mu, d)$ be such a class of valid inequalities and define the class $\mathcal{C}$ to be the union of all classes $\mathcal{C}(\mu, d)$ of all inequalities $\mu y \geq d$ that are valid for $Y(G, H, \cdot)$. In principle, we seek for a separation algorithm for such a class $\mathcal{C}$ which *simultaneously* finds an inequality $\mu y \geq d$ and a violated inequality in $\mathcal{C}(\mu, d)$. However, even for classes $\mathcal{C}(\mu, d)$ the separation problems are difficult and, in fact, some of them are $\mathcal{NP}$-complete. Therefore, we decided to use the approach sketched in Algorithm 3.

Algorithm 3 BASICSEPARATION

Require: pool of valid inequalities for $Y(G, H, \cdot)$
Require: separation algorithms for some classes $\mathcal{C}(\mu, d)$
  while separation algorithms are sufficiently successful **do**
    for all inequalities $\mu y \geq d$ in the pool **and all** separation algorithms **do**
      apply separation algorithm for $\mathcal{C}(\mu, d)$
      update and solve the relaxation
    end for
    update the pool (add and delete)
  end while

The basic version of the separation algorithm maintains a pool of inequalities valid for $Y(G, H, \cdot)$. These are inequalities which have proven to be useful at run-time. In addition to the pool, a set of separation algorithms for some of the classes $\mathcal{C}(\mu, d)$ is given. The main loop of the algorithm iterates in some order (to be made precise) over the inequalities in the pool and these separation algorithms. The current separation algorithm is applied to the current class $\mathcal{C}(\mu, d)$. Identified inequalities are added to the linear programming relaxation, if these are violated by the current optimal solution. After each iteration of the main loop, the pool of valid inequalities for $Y(G, H, \cdot)$ is updated. This means, inequalities for which the separation algorithms were too often unsuccessful are removed, and new inequalities for the pool are generated with a separation algorithm for metric inequalities. The algorithm terminates, if the objective function value of the linear programming relaxation increased less than a certain threshold during the last iteration of the main loop.

Apparently, the efficiency of Algorithm 3 is influenced by the selection of run-time parameters such as the applied separation algorithms, the number of violated inequalities added in each iteration, the way the pool is updated, etc. We discuss the influence of these parameters in

Section 4.5.4, where we describe our computational experiments. Beforehand, we present in Sections 4.5.1, 4.5.2, and 4.5.3 the separation algorithms for individual classes of valid inequalities for $Y(G,H,\cdot)$, $X(G,H,\text{FIN},\cdot)$, and $X(G,H,\text{BAS},\cdot)$, respectively.

4.5.1 Inequalities for $Y(G,H,\cdot)$

We present separation algorithms for *metric* inequalities (2.22) (including its variations (3.1), (3.2) and (3.3)), and for the subclass of *k-graph-partition* inequalities (2.23). Suppose that a capacity vector $\overline{y}(e)$, $e \in E$, is given.

Metric inequalities

In Section 4.4.2, we described an algorithm to solve the feasibility problem for a capacity vector $\overline{y}$, and we already mentioned in Remark 4.5 that this algorithm serves as a separation algorithm for metric inequalities as well. The metric inequalities (2.22) for the normal operating state, and (3.2) for a failure state $s \in S\backslash\{0\}$, are special cases of the metric inequalities (3.1). Thus, we first describe the separation algorithm for the latter class, and then the one for metric inequalities (3.3).

Suppose that the algorithm to test feasibility of $\overline{y}$ for the normal operating state including the diversification constraints terminates with objective function value $\overline{\alpha} \leq 0$. In this case, $\overline{y}$ is feasible, and no metric inequality (3.1) is violated due to Theorem 3.2. In the other case, if $\overline{\alpha} > 0$, the dual variables $\mu_e, e \in E$, $\pi_{uv}, uv \in D$, $\gamma_{uv}^w, uv \in D, w \in V\backslash\{u,v\}$, and $\gamma_{uv}^e, uv \in D, e \in \delta_G(u) \cap \delta_G(v)$ which are associated with the constraints (4.7) for $s = 0$, (4.8), (4.10), and (4.11), respectively, satisfy

$$\sum_{uv \in D} \left(d_{uv}\pi_{uv} - \sum_{e \in \delta_G(u) \cap \delta_G(v)} \delta_{uv} d_{uv} \gamma_{uv}^e - \sum_{w \in V\backslash\{u,v\}} \delta_{uv} d_{uv} \gamma_{uv}^w \right) - \sum_{e \in E} \overline{y}(e)\mu_e \quad > \quad 0 \; .$$

This follows immediately from linear programming duality (see Theorem 0.1). According to Theorem 3.2, the dual variables define a violated metric inequality (3.1), if all dual variables π_{uv}, $uv \in D$, are the values of shortest uv-paths in G with at most ℓ_{uv} edges and with respect to the supply edge weights

$$w_e \quad := \mu_e + \begin{cases} \gamma_{uv}^e \; , & u = v_1, v = v_2, \text{ or } u = v_2, v = v_1 \; , \\ \frac{1}{2}(\gamma_{uv}^{v_1} + \gamma_{uv}^{v_2}) \; , & \text{else} \; , \end{cases}$$

for all $e = v_1 v_2 \in E$. This follows, since all dual constraints (4.25) and (4.26) are satisfied by the dual variables.

The separation algorithm for metric inequalities (3.3) is based on similar arguments. The capacity vector $\overline{y}$ is feasible for PATH RESTORATION, if the respective algorithm to test feasibility of $\overline{y}$ terminates with objective function value $\overline{\alpha} \leq 0$. In this case, no metric inequality (3.3) is violated. The other case, $\overline{\alpha} > 0$, is more complicated. Let $\mu_e, e \in E$, $\mu_e^s, s \in S\backslash\{0\}, e \in E_s$, $\pi_{uv}, uv \in D$, $\pi_{uv}^s, s \in S\backslash\{0\}, uv \in D_s$ be the optimal dual variables associated with (4.14), (4.15), (4.16), and (4.17), respectively. (We ignore possible diversification constraints here.) If all dual constraints (4.28) and (4.29) are satisfied, then $\overline{\alpha}$ is optimal and the dual objective function satisfies

$$\sum_{uv\in D} d_{uv}\pi_{uv} - \sum_{e\in E} y(e)\,\mu_e + \sum_{s\in S\backslash\{0\}} \sum_{uv\in D_s} \sigma_{uv} d_{uv}\pi_{uv}^s - \sum_{s\in S}\sum_{e\in E_s} y(e)\,\mu_e^s \quad > \quad 0 \; .$$

Thus, the dual objective function induces a violated metric inequality (3.3), since the dual constraints (4.28) and (4.29) are exactly the conditions of Proposition 3.4. (Again, we ignore possible diversification constraints.) The main difficulty is the verification that all dual constraints (4.29) are satisfied. Proposition 4.6 provides a sufficient condition for the dual variables to satisfy constraints (4.29). Hence, if these conditions are satisfied the separation algorithm terminates with a violated metric inequality (3.3). Otherwise, the algorithm fails to solve this separation problem.

k-graph-partition inequalities

The class of k-graph-partition inequalities (2.23) is the most important subclass of metric inequalities. In particular, if these are cut inequalities, that is, if $k = 2$. In principle, there is no need for another separation algorithm for these subclasses since an exact one for metric inequalities is known. There are, however, at least two arguments why one should be willing to apply a different separation algorithm. First, a metric inequality is nasty in the sense that its support is often almost the entire set of supply edges, and even worse, metric inequalities often have "wild" coefficients which can cause numerical instabilities. Second, the above separation algorithms are time-consuming in practice, since the feasibility problem for a capacity vector must be solved as a subproblem.

We employ a heuristic separation algorithm (see Algorithm 4) for k-graph-partition inequalities suggested by Dan Bienstock. The algorithm is based on two arguments. First, if the number of nodes in the graph is small, e.g. less than 10, then the time to enumerate all k-graph-partitions for some fixed small k is small as well, and therefore, it is practically possible to apply complete enumeration as separation algorithm. Second, it is likely that supply edges $e \in E$ with large capacities $\overline{y}(e)$ are not in the support of violated k-graph-partition inequalities. Thus, it is reasonable to shrink supply edges with large capacities and to enumerate all k-graph-partitions afterwards.

Typically, we use the values $k = 2, 3$ and $p = 8, 9, 10$. As supply edge weights $w(e), e \in E$, we consider the following alternatives:

$$w(e) \;:=\; \overline{y}(e) \;, \quad \text{or} \quad w(e) \;:=\; \overline{y}(e) - \sum_{uv\in D: e\in \overline{P_{uv}}} d_{uv} \;,$$

where $\overline{P_{uv}}$ is a shortest uv-path with respect to geographical distances for demand edge $uv \in D$. In the first step of Algorithm 4, the supply edges are sorted in decreasing order with

Algorithm 4 k-GRAPHPARTITIONSEPARATION

Require: $p, k \in \mathbb{N}, k \geq 2, p > k$
  sort edges in decreasing order with respect to weights $w(e), e \in E$.
  set $\overline{G} = (\overline{V}, \overline{E}) := G = (V, E)$.
  while $|\overline{V}| > p$ **do**
    shrink end-nodes of the *next* edge with largest weight
  end while
  for all k-graph-partitions $\overline{V}_1, \ldots, \overline{V}_k$ in $\overline{G}$ **do**
    evaluate the corresponding k-graph-partition inequality (2.23) in G
  end for

respect to these weights. If the weights of two different supply edges are equal, the geographical distance between the end-nodes is the tie-breaker. Then, the end-nodes of supply edges with the largest weights are identified, until the number of nodes in the resulting multi-graph (each node represents a set of nodes and each edge the set of edges between the two sets of nodes) is equal to p. Eventually, all k-graph-partitions in the final multi-graph are enumerated and for each k-graph-partition the corresponding k-graph-partition inequality (2.23) is evaluated in the original supply and demand graph.

4.5.2 Inequalities for $X(G, H, \textsc{Fin}, \cdot)$

In this section, we present separation algorithms for classes of valid inequalities for the polyhedra $X(G, H, \textsc{Fin}, \cdot)$. In particular, for the classes of *strengthened-metric* inequalities (3.6), *band* inequalities (3.8), *2-band* inequalities (3.9), *3-graph-partition band* inequalities (3.10), *diversification-band* inequalities (3.12), *strengthened band* inequalities (3.13), and *strengthened 2-band* inequalities (3.15). Throughout this section, a (fractional) vector $\overline{x}(e,t), e \in E, t = 1, \ldots, T_e$, is given.

Strengthened metric inequalities

The separation algorithm for strengthened metric inequalities (3.6) is completely determined by the separation algorithms for metric and k-graph-partition inequalities presented in the previous section. The first step attempts to separate $\overline{y}$ from $Y(G, H, \textsc{Nos})$, where $\overline{y}(e) := \sum_{t=0}^{T_e} c_e^t x(e,t), e \in E$ with a separation algorithm presented in the previous section. However, with a small change of the output: the separation algorithm always returns a metric inequality. If there is no violated metric inequality, the algorithm yields one with minimal slack. Eventually, the associated strengthened metric inequality is evaluated for $\overline{x}$.

Band inequalities

In the separation algorithm for band inequalities (3.8), we assume a given valid inequality $\sum_{e \in F} \mu_e y(e) \geq d$ for $Y(G, H, \textsc{Nos})$ with $\mu_e > 0$ for all supply edges $e \in F \subseteq E$. This means, the separation algorithm identifies only those band inequalities which are valid for a particular induced knapsack-relaxation (see page 65). The separation problem for band inequalities can be formulated as a multiple-choice knapsack problem. A formulation as an integer program with strict inequalities is the following:

$$\begin{aligned}
\min \quad & \sum_{e\in F}\sum_{t=1}^{T_e} \overline{x}(e,t)\beta_e^t && && \\
& \sum_{e\in F}\sum_{t=1}^{T_e} \mu_e C_e^{t-1}\beta_e^t && < && d\,, && \qquad (4.35)\\
& \sum_{t=1}^{T_e} \beta_e^t && = && 1\,, \quad e\in F\,, && \qquad (4.36)\\
& \beta_e^t && \in && \{0,1\}\,, \quad e\in F\,,\ 1\le t\le T_e\,. &&
\end{aligned}$$

This optimization problem is $\mathcal{NP}$-complete (see (Garey and Johnson, 1979)). Given a solution $\overline{\beta}_e^t$, $e\in F$, $t=1,\ldots,T_e$, there is a unique breakpoint t_e for each supply edge $e\in F$ with $\overline{\beta}_e^{t_e}=1$ and therefore we can define the corresponding band

$$B(\overline{\beta}) \quad := \quad \{(e,t_e-1) : e\in F, \overline{\beta}_e^{t_e}=1\}\ .$$

The band inequality (3.8) for band $B(\overline{\beta})$ is violated if and only if

$$1 \ > \ \sum_{e\in F}\sum_{t=1}^{T_e}\overline{x}(e,t)\overline{\beta}_e^t \ = \ \sum_{e\in F}\overline{x}(e,(t_e-1)+1)\ .$$

Due to Proposition 3.15, maximal band inequalities are facet-defining for the induced knapsack-relaxation. Thus, the question arises whether the band $B(\overline{\beta})$ corresponding to an *optimal* solution $\overline{\beta}_e^t$, $e\in F$, $t=1,\ldots,T_e$, is maximal. In general, this is not true. For instance, if a supply edge $g\in F$ with $\overline{x}(g,t_g)=\overline{x}(g,t_g+1)$ exists such that the weighted incremental capacity $\mu_g c_g^{t_g}$ is smaller than the slack in inequality (4.35), that is, if $\mu_g c_g^{t_g}+\sum_{e\in F}\mu_e C_e^{t_e-1}<d$, then the band $B(\overline{\beta})$ corresponding to the optimal solution $\overline{\beta}$ is not maximal.

We apply dynamic programming (see (Bellman and Dreyfus, 1962)) to solve the multiple-choice knapsack problem. Assuming integer data, in a straightforward implementation (see (Martello and Toth, 1990)), one would define an order $F=\{e_1,\ldots,e_{|F|}\}$, and then iteratively calculate for $i=1,\ldots,|F|$ and $c=0,\ldots,d$, the value $f_i(c)$ which is defined as the optimal solution value of the multiple-choice knapsack problem defined above for the restricted edge set $\{e_1,\ldots,e_i\}$ (instead of F) and right-hand side c (instead of d) in (4.35). This value can recursively be calculated through

$$f_i(c) \quad := \quad \min\{f_{i-1}(c-\mu_{e_i}C_{e_i}^{t-1})+\overline{x}(e_i,t) : t=1,\ldots,T_e, c\ge \mu_{e_i}C_{e_i}^{t-1}\}$$

where $f_0(c):=0$, for $c=0,\ldots,d$. Eventually, the optimal solution value of the multiple-choice knapsack problem is $f_{|F|}(d)$. In our case, this algorithm is very time consuming since it depends on the value d, which is potentially a large value.

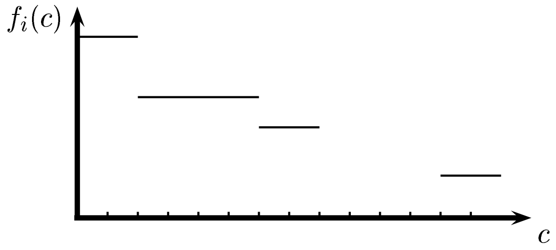


Figure 4.6: Multiple-choice knapsack : Objective function for fixed $i \in \{1, \ldots, |F|\}$

Our implementation, which is sketched in Algorithm 5, relies on the following observation. For fixed $i \in \{1, \ldots, |F|\}$, the values $f_i(c)$, $c \in \mathbb{R}_+$, define a staircase function as illustrated in Figure 4.6. That is, a finite set of break-even capacities exists such that the objective function value is monotonically decreasing and *constant* between two succeeding break-even capacities. Therefore, it suffices to calculate the break-even capacities. The outer for-loop of Algorithm 5 iterates over the supply edges in some order, say $F = \{e_1, \ldots, e_{|F|}\}$, and in the two inner for-loops, the list of break-even capacities is maintained and updated. In the $i-$th iteration, for $i = 1, \ldots, |F|$, the list of break-even capacities of the $(i-1)-$th iteration is given. Together with the breakpoint capacities $C^0_{e_i}, \ldots, C^{T_{e_i}}_{e_i}$ new break-even capacities are calculated. A new break-even capacity is added to the current list, if it is not dominated by an existing break-even capacity. It is not described in Algorithm 5, but further data associated with break-even capacities must be maintained such that an optimal solution can easily be constructed after the $|F|-$th iteration. As we noted above, the computed band is not necessarily maximal. Hence, the band-breakpoints $t_e, e \in E$, are increased until the constructed band is maximal. Notice that the objective function value of the maximal band is equal to $f_{|F|}(d)$.

Algorithm 5 BANDSEPARATION

Require: $\sum_{e \in F} \mu_e y(e) \geq d$ and $\overline{x}(e,t), e \in F, t = 1, \ldots, T_e$
  Define order $F = \{e_1, \ldots, e_{|F|}\}$
  $f_0(c) := 0, c \in \mathbb{R}_+$
  for all $i := 1, \ldots, |F|$ **do**
    for all $t := 1, \ldots, T_e$ **do**
      for all break-even capacities c **do**
        if $c + \mu_{e_i} C^{t-1}_{e_i} < d$ **and** $f_{i-1}(a) > f_{i-1}(c) + \overline{x}(e_i, t)$, for all a with $a < c + \mu_{e_i} C^{t-1}_{e_i}$ **then**
          add break-even capacity $c + \mu_{e_i} C^{t-1}_{e_i}$, and remove all break-even capacities a with $a > c + \mu_{e_i} C^{t-1}_{e_i}$ and $f_{i-1}(a) > f_{i-1}(c) + \overline{x}(e_i, t)$.
        end if
      end for
    end for
  end for
  if $f_{|F|}(d) < 1$ **then**
    initialize band $B(\overline{\beta})$ (corresponding to $f_{|F|}(d)$)
    construct maximal band above $B(\overline{\beta})$
  end if

2-band inequalities

Our heuristic separation algorithm for 2-band inequalities (3.9) is based on Algorithm 5. Given a valid inequality $\sum_{e\in F}\mu_e y(e) \geq d$ for $Y(G,H,\mathrm{Nos})$ as input, the application of Algorithm 5, yields a band $B = \{(e,t_e) : e \in F\}$. For each $e \in F$ with $t_e > 0$, the construction described in the next paragraph is performed. If no such supply edge exists, that is, if $t_e = 0$ for all $e \in F$, the separation algorithm fails to identify a violated 2-band inequality (3.9). Output of the separation algorithm is the maximal violated among all constructed 2-band inequalities.

For $g \in F$ with $t_g > 0$, interpret B as $\{(g,t_g^2)\} \cup \{(e,t_e^1) : e \in F\backslash\{g\}\}$, that is, $t_g^2 := t_g$ and $t_e^1 := t_e$ for all $e \in F\backslash\{g\}$. Now, for each breakpoint $t_g^1 < t_g^2$ and each supply edge $h \in F\backslash\{g\}$, let $t_h^2 > t_h^1$ be the maximal breakpoint of h with

$$\mu_h C_h^{t_h^2} + \sum_{e\in F\backslash\{h\}} \mu_e C_e^{t_e^1} \quad < \quad d \, .$$

If there is no such t_h^2 for a supply edge $h \in F\backslash\{g\}$, then no 2-band is constructed for this choice of t_g^1. Notice that t_h^2 is defined such that the validity requirements of Definition 3.16 are satisfied and hence this choice of breakpoints defines a valid 2-band for (μ, d).

3-graph-partition band inequalities

Suppose that a 3-graph-partition with shores V_1, V_2, V_3 is given as input of the separation algorithm for 3-graph-partition band inequalities (3.10). The target of this algorithm is to construct a valid $\mathcal{P} - band$ (see Definition 3.2.1 on page 72) such that the corresponding inequality (3.10) is violated by $\overline{x}$. In the first step, Algorithm 5 is applied for each pair (V_i, V_j), $1 \leq i < j \leq 3$, of shores with inequality

$$\sum_{e\in\delta_G(V_i,V_j)} y(e) \quad \geq \quad \sum_{uv\in\delta_H(V_i,V_j)} d_{uv}$$

and $\overline{x}$ restricted to the supply edges $\delta_G(V_i, V_j)$ between the two shores as input. Each of these applications yields a (partial) band

$$B_{ij} \quad := \quad \{(e,t_e) : e \in \delta_G(V_i,V_j)\} \, .$$

Notice that each supply edge $e \in \delta_G(V_1, V_2, V_3)$ appears in exactly one of these partial bands. Thus, the union $B := B_{12} \cup B_{13} \cup B_{23}$ defines a $\mathcal{P} - band$, which is by construction valid. The remaining steps of the separation algorithm for 3-graph-partition band inequalities (3.10) maximize this initial $\mathcal{P} - band$.

Diversification-band inequalities

Suppose that a cut inequality $\sum_{e\in F} y(e) \geq d$ with $F := \delta_G(W)$ for $\emptyset \subset W \subset V$ and $d := \sum_{uv\in\delta_H(W)} d_{uv}$ is given as input of the separation algorithm for diversification-band inequalities

(3.12). Furthermore, let $\delta := \left(\sum_{uv\in\delta_G(W)} \delta_{uv} d_{uv}\right) / \left(\sum_{uv\in\delta_G(W)} d_{uv}\right)$ and assume $0 < \delta < 1$. For this setting, the target of this separation algorithm is to construct a diversification-band

$$B = \{(e, t_e) : e \in F,\ 0 \leq t_e < T_e \text{ s.t. } C_e^{t_e} < \delta d\} ,$$

such that the associated diversification-band inequality

$$\sum_{e\in F} x(e, t_e + 1) \geq \operatorname{cov}(B)$$

is violated. Recall, $\operatorname{cov}(B)$ is the minimal number of edges $e \in F$ on which a capacity larger than $C_e^{t_e}$ is needed to satisfy the underlying cut inequality. For band inequalities, $\operatorname{cov}(B) = 1$ always holds, but, in general, $\operatorname{cov}(B)$ attains a value in $\{1, \dots, \lceil 1/\delta \rceil\}$, since the maximum flow through any supply edge is bounded from above by δd. In the main loop, this separation algorithm iterates over the possible values of $\operatorname{cov}(B)$ and attempts to identify a violated diversification-band inequality (3.12). As a subproblem, an adaption of the multiple-choice knapsack problem is solved which can be formulated for fixed $k = 1, \dots, \lceil 1/\delta \rceil$ as the following integer program.

$$\begin{array}{llll}
\min \displaystyle\sum_{e\in F} \sum_{t=1,\dots,T_e : C_e^{t-1} < \delta d} & \overline{x}(e,t)\beta_e^t & & \\
\displaystyle\sum_{e\in F} \sum_{t=1,\dots,T_e : C_e^{t-1} < \delta d} & C_e^{t-1}\beta_e^t & < \; d - (k-1)\delta d \,, & \\
\displaystyle\sum_{t=1,\dots,T_e : C_e^{t-1} < \delta d} & \beta_e^t & = \; 1 \,, & e \in F \,, \\
 & \beta_e^t & \in \; \{0,1\} \,, & e \in F \,,\ 1 \leq t \leq T_e \,.
\end{array}$$

This integer program is similar to the one formulated for the separation problem for band inequalities. Just the right-hand side d in (4.35) is substituted by $d-(k-1)\delta d$ and the admissible breakpoints for a supply edge $e \in F$ are restricted to those breakpoints $t \in \{1, \dots, T_e\}$ with capacities $C_e^{t-1} < \delta d$. Let $\overline{\beta}$ be an optimal solution of this integer program and denote by $B(\overline{\beta}) := \{(e, t_e - 1) : e \in F, \overline{\beta}_e^{t_e} = 1\}$ the associated diversification-band. (Note, $B(\overline{\beta})$ is by definition a diversification-band, since $C_e^{t_e} < \delta d$ for all $e \in F$.) The right-hand side $d-(k-1)\delta d$ is chosen such that the residual band-demand satisfies $r(B(\overline{\beta})) > (k-1)\delta d$, and thus

$$\sum_{e\in F} x(e, (t_e - 1) + 1) \quad \geq \quad k$$

is a valid diversification-band inequality due to Lemma 3.31. This might be violated by $\overline{x}$, but there are several opportunities to improve it. As we illustrated in Example 3.32 (see page 76), $\operatorname{cov}(B(\overline{\beta})) > k$ might hold and thus $\operatorname{cov}(B(\overline{\beta}))$ needs to be computed to obtain the inequality

$$\sum_{e\in F} x(e, (t_e - 1) + 1) \quad \geq \quad \operatorname{cov}(B(\overline{\beta})) \,.$$

Due to Proposition 3.38, maximal diversification-bands (see Definition 3.36) are candidates to induce facet-defining inequalities. Thus, band-breakpoints $t_e, e \in F$, are increased in the next step, until a maximal diversification-band with respect to $\operatorname{cov}(B(\overline{\beta}))$ is constructed.

Strengthened band inequalities

Let a cut inequality $\sum_{e\in F} y(e) \geq d$ with $F := \delta_G(W) \subseteq S$ for some $\emptyset \subset W \subset V$, and $d = \sum_{uv\in\delta_H(W)} d_{uv}$ be given as part of the input of the separation algorithm for strengthened band inequalities (3.13). For every $g \in F$, the inequality $\sum_{e\in F\setminus\{g\}} y(e) \geq \sum_{uv\in\delta_H(W)} \rho_{uv} d_{uv} =: rd$ is valid for $Y(G, H, \textsc{Res})$, and by definition of (3.13), the target of this separation algorithm is a band $B = \{(e, t_e) : e \in F\}$ such that, for every $g \in F$, the (partial) band $B_g := B\setminus\{(g, t_g)\}$ is valid for $(\chi^{F\setminus\{g\}}, d)$, that is,

$$\sum_{e\in F\setminus\{g\}} C_e^{t_e} \quad < \quad rd \ ,$$

for all $g \in F$. The separation algorithm proceeds as follows. First, a supply edge $g \in F$ is selected such that $C_g^0 \leq C_e^0$ for all $e \in F$. Then, Algorithm 5 is applied to the edge set $e \in F\setminus\{g\}$, the failure demand rd, and the vector $\overline{x}$. The result is a band $B_g = \{(e, t_e) : e \in F\setminus\{g\}\}$ which induces the valid strengthened band inequality

$$x(g, 1) + \sum_{e\in F\setminus\{g\}} x(e, t_e + 1) \quad \geq \quad 2 \ ,$$

since $C_g^0 \leq C_e^0 \leq C_e^{t_e}$ for all $e \in F$. Again, the target is a maximal band and therefore a maximal $t_g \in \{0, \ldots, T_g - 1\}$ is chosen, such that the band $B := B_g \cup \{(g, t_g)\}$ remains valid. This procedure is iteratively applied to all supply edges $g \in F$ with $C_g^0 \leq C_e^0$ for all $e \in F\setminus\{g\}$.

Strengthened 2-band inequalities

Let a cut inequality $\sum_{e\in F} y(e) \geq rd$ with $F := \delta_G(W) \subseteq S$ for some $\emptyset \subset W \subset V$, and $rd = \sum_{uv\in\delta_H(W)} \rho_{uv} d_{uv}$ be given as part of the input of the separation algorithm for strengthened 2-band inequalities (3.15). The target of this algorithm is to construct a 2-band $B = \{(e, t_e^1, t_e^2) : e \in F,\ 0 \leq t_e^1 < t_e^2 < T_e\}$ satisfying

$$(C_{g_1}^{t_{g_1}^2} + C_{g_2}^{t_{g_2}^2}) + \sum_{e\in F\setminus\{g, g_1, g_2\}} C_e^{t_e^1} \quad < \quad rd \ ,$$

for all $g \in F$ and all $g_1, g_2 \in F\setminus\{g\}$ with $g_1 \neq g_2$, such that the corresponding inequality (3.15) is violated by $\overline{x}$. In the first step, Algorithm 5 is applied to the cut inequality $\sum_{e\in F} y(e) \geq rd$ and $\overline{x}$. The resulting band $B = \{(e, t_e) : e \in F\}$ is interpreted as partial 2-band. Two supply edges g and h with strictly positive band breakpoints are chosen such that the difference $\overline{x}(e, t-1) - \overline{x}(e, t+1)$ is minimal. With this choice, the band B is interpreted as partial 2-band $\{(g, t_g^2), (h, t_h^2)\} \cup \{(e, t_e^1) : e \in F\setminus\{g, h\}\}$, that is, $t_g^2 := t_g$, $t_h^2 := t_h$, and $t_e^1 := t_e$ for all $e \in F\setminus\{g, h\}$. Now, let $r(B)$ be the residual band-demand, that is,

$$r(B) \quad := \quad rd - \sum_{e\in F} C_e^{t_e} \ > \ 0 \ .$$

Given this interpretation of the band B and a choice of the lower breakpoints for g and h, the next step of the algorithm attempts to identify breakpoints t_e^2 for all supply edges $e \in F\backslash\{g,h\}$ such that

$$(C_{e_1}^{t_{e_1}^2} - C_{e_1}^{t_{e_1}^1}) + (C_{e_2}^{t_{e_2}^2} - C_{e_2}^{t_{e_2}^1}) \quad < \quad r(B) \; + \; \min\{C_e^{t_e^1} : e \in F\backslash\{g,h\}\}$$

is satisfied for all $e_1, e_2 \in F$, and

$$\sum_{e \in F\backslash\{g,h\}} \overline{x}(e, t_e^2)$$

is minimal. A greedy heuristic is applied for this purpose.

4.5.3 Inequalities for $X(G, H, \mathrm{BAS}, \cdot)$

In this section, we present separation algorithms for classes of valid inequalities for the polyhedra $X(G, H, \mathrm{BAS}, \cdot)$. In particular, for the classes of *strengthened metric* inequalities (3.16), *knapsack-partition* inequalities (3.20), *diversification-cut* inequalities (3.27), *diversification-partition* inequalities (3.29), *lifted diversification-cut* inequalities (3.32), and *strengthened knapsack-partition* inequalities (3.36). Throughout this section, a vector $\overline{x}(e,\tau)$, $e \in E$, $\tau \in T$, is given.

Strengthened metric inequalities

The separation algorithm for strengthened metric inequalities (3.16) is almost identical to the one presented for strengthened metric inequalities (3.6) in Section 4.5.2. Instead of (3.6), inequality (3.16) is evaluated for $\overline{x}$.

Knapsack-partition inequalities

Suppose that a k-graph-partition $V_1, \ldots, V_k$ is given as part of the input for the separation algorithm for the class of knapsack-partition inequalities (3.20). The separation problem for this class of inequalities reduces to finding a violated inequality (3.19) for

$$Q(M,d) \quad := \quad \mathrm{conv}\{z \in \mathbb{Z}_+^n : \sum_{i=1}^n c_i z_i \geq d\} \; ,$$

and $\overline{z}$, where

$$\begin{aligned} M &:= \{c_1, \ldots, c_n\} \; := \; \{C^{\tau_1}, \ldots, C^{\tau_n}\} \; , \\ d &:= \sum_{uv \in \delta_H(V_1,\ldots,V_k)} d_{uv} \; , \\ \overline{z}_i &:= \sum_{e \in \delta_G(V_1,\ldots,V_k)} \overline{x}(e,\tau_i) \; , \; i = 1, \ldots, n \; . \end{aligned}$$

(Pochet and Wolsey, 1995) proposed an exact separation algorithm to solve the separation problem for the vector $\overline{z}$ and the class of inequalities (3.19). We briefly summarize this algorithm, but refer the reader to the original paper for details. The algorithm is based on decomposition arguments and the following proposition. Let $r(d) := \max\{i : c_i \leq d, 1 \leq i \leq n\}$ be the maximal index such that the capacity c_i is less than or equal to d.

Proposition 4.8 *If $d/c_{r(d)} \in \mathbb{N}$, then $Q(M,d) = \{z \in \mathbb{R}^n_+ : \sum_{i=1}^{r(d)} c_i z_i + d \sum_{i=r(d)+1}^{n} z_i \geq d\}$.*

Thus, if $d/c_{r(d)} \in \mathbb{N}$, the separation problem reduces to the evaluation of the inequality

$$\sum_{i=1}^{r(d)} c_i z_i + d \sum_{i=r(d)+1}^{n} z_i \;\; \geq \;\; d$$

for $\overline{z}$. In the other case, if $d/c_{r(d)} \notin \mathbb{N}$, and if

$$\sum_{j=r(d)+1}^{n} \overline{z}_i \;\; \geq \;\; 1 \; ,$$

then $\overline{z} \in Q(M,d)$. If both these arguments do not apply, then $d/c_{r(d)} \notin \mathbb{N}$ and $\sum_{j=r(d)+1}^{n} \overline{z}_i < 1$. For this case, (Pochet and Wolsey, 1995) proved that $\overline{z} \notin Q(M,d)$, if

$$\sum_{j=1}^{r(d)} c_i \overline{z}_i \;\; < \;\; \left(1 - \sum_{j=r(d)+1}^{n} \overline{z}_i\right) \left\lfloor \frac{d}{c_{r(d)}} \right\rfloor c_{r(d)} \; .$$

Now, suppose that the separation problem cannot be solved with one of the preceding arguments. Then, $\sum_{j=r(d)+1}^{n} \overline{z}_i < 1, d/c_{r(d)} \notin \mathbb{N}$ and $\sum_{j=1}^{r(d)} c_i \overline{z}_i \geq (1 - \sum_{j=r(d)+1}^{n} \overline{z}_i) \lfloor d/c_{r(d)} \rfloor c_{r(d)}$. In this case, the vector $\overline{z}$ can be decomposed such that the separation problem can be reduced to a smaller subproblem. Therefore, let $\overline{z}$ be decomposed into

$$\overline{\alpha} \;\; := \;\; (0, \ldots, 0, \overline{\alpha}_v, \overline{z}_{v+1}, \ldots, \overline{z}_n) \text{ and} \tag{4.37}$$

$$\overline{\gamma} \;\; := \;\; (\overline{z}_1, \ldots, \overline{z}_{v-1}, \overline{z}_v - \overline{\alpha}_v, 0, \ldots, 0, \overline{z}_{r(d)+1}, \ldots, \overline{z}_n) \; , \tag{4.38}$$

where

$$v := \min \left\{ k : \sum_{j=k+1}^{r(d)} c_i \overline{z}_i \;\; < \;\; \left(1 - \sum_{j=r(d)+1}^{n} \overline{z}_i\right) \left\lfloor \frac{d}{c_{r(d)}} \right\rfloor c_{r(d)}, 1 \leq k \leq n \right\}$$

and $\overline{\alpha}_v$ is chosen such that $\sum_{j=1}^{n} c_i \overline{\alpha}_i = \lfloor d/c_{r(d)} \rfloor c_{r(d)}$. According to the following proposition it then suffices to solve the separation problem for the "smaller" vector $\overline{\gamma}$ and reduced right-hand side.

Algorithm 6 KNAPSACKPARTITIONSEPARATION

Require: $\{c_1, \ldots, c_n\}, \{\overline{z}_1, \ldots, \overline{z}_n\}$, and $d \in \mathbb{N}$

  repeat
    $r(d) := \max\{i : c_i \leq d\}$
    if $d/c_{r(d)} \in \mathbb{N}$ **then**
      if $\sum_{i=1}^{r(d)} c_i\overline{z}_i + d\sum_{i=r(d)+1}^{n} \overline{z}_i \geq d$ **then**
        stop : $\overline{z}$ satisfies all inequalities (3.19)
      else
        stop : construct violated inequality
      end if
    else if $\sum_{j=r(d)+1}^{n} \overline{z}_i \geq 1$ **then**
      stop : $\overline{z}$ satisfies all inequalities (3.19)
    else if $\sum_{j=1}^{r(d)} c_i\overline{z}_i < (1 - \sum_{j=r(d)+1}^{n} \overline{z}_i)\lfloor d/c_{r(d)}\rfloor c_{r(d)}$ **then**
      stop : construct violated inequality
    else
      set $v := \min\{k : \sum_{j=k+1}^{r(d)} c_i\overline{z}_i < (1 - \sum_{j=r(d)+1}^{n} \overline{z}_i)\lfloor d/c_{r(d)}\rfloor c_{r(d)}, 1 \leq k \leq n\}$
      set $\overline{\alpha}_v$ according to (4.37) such that $\sum_{j=1}^{n} c_i\overline{\alpha}_i = \lfloor d/c_{r(d)}\rfloor c_{r(d)}$
      set $\overline{z} := (\overline{z}_1, \ldots, \overline{z}_{v-1}, \overline{z}_v - \overline{\alpha}_v, 0, \ldots, 0, \overline{z}_{r(d)+1}, \ldots, \overline{z}_n)$ $\quad$ ($\overline{z} = \overline{\gamma}$ of (4.38))
      set $d := d - \lfloor d/c_{r(d)}\rfloor c_{r(d)}$
    end if
  until stop

Proposition 4.9 *$\overline{\gamma} \in Q(M, d - \lfloor d/c_{r(d)}\rfloor c_{r(d)})$ if and only if $\overline{z} \in Q(M, d)$.*

Algorithm 6 summarizes the separation algorithm for inequalities (3.19).

Suppose that Algorithm 6 yields a partition $\{l_1, \ldots, j_1\}, \ldots, \{l_t, \ldots, j_t\}$ of the index set of $\{C^{\tau_1}, \ldots, C^{\tau_n}\}$ w.r.t. d such that the corresponding inequality (3.19) is violated. Let $\kappa_1, \ldots, \kappa_t$ be defined as in (3.18), and for $i = 1, \ldots, n$, let $p(i)$ be the partition index of technology τ_i, that is, $i \in \{l_{p(i)}, \ldots, j_{p(i)}\}$. Then, the knapsack-partition inequality (3.20)

$$\sum_{e \in F}\sum_{i=1}^{n} \min\left\{\kappa_{p(i)}, \frac{C^{\tau_i}}{C^{\tau_{l_{p(i)}}}}\right\} \cdot \left(\prod_{s=1}^{p(i)-1} \kappa_s\right) x(e, \tau_i) \quad \geq \quad \prod_{i=1}^{t} \kappa_i$$

is violated by $\overline{x}$.

Diversification-related inequalities

Suppose that a cut inequality $\sum_{e \in F} y(e) \geq d$, for $W \subseteq V$ and $F := \delta_G(W)$, is given such that the diversification parameter is less than one for some demand edges with end-nodes in two different shores, that is,

$$\delta d \;:=\; \sum_{uv \in \delta_H(W)} \delta_{uv} d_{uv} \;<\; \sum_{uv \in \delta_H(W)} d_{uv} =: d\,.$$

For $\delta := d/\delta d$, the target of the following separation algorithm is to identify a violated inequality among all inequalities (3.27), (3.29), and (3.32). The separation algorithm for these classes of inequalities requires the calculation of minimal diversification covers. For $d \in \mathbb{N}$, $\delta d \in \mathbb{Q}_+$, $C \in \mathbb{N}$, $k \in \mathbb{N}$, the minimal diversification cover is calculated as follows. According to (3.26) and Lemma 3.58, if $\lceil d/C \rceil = \lceil 1/\delta \rceil \lceil \delta d/C \rceil$ or $\delta d/C \in \mathbb{N}$, then

$$(d, \delta d, C, k) = \left\lceil \frac{d}{C} \right\rceil ,$$

and for $C \geq \delta d$, it follows from Lemma 3.59 that

$$(d, \delta d, C, k) = \left\lceil \frac{1}{\delta} \right\rceil .$$

If none of the previous arguments can be applied, then $d - k\lfloor \delta d/C \rfloor > 0$, and Lemma 3.61 implies

$$(d, \delta d, C, k) = k \left\lfloor \frac{\delta d}{C} \right\rfloor + \left\lceil \left(d - k \cdot C \left\lfloor \frac{\delta d}{C} \right\rfloor \right) / \left(\delta d - C \left\lfloor \delta \frac{d}{C} \right\rfloor \right) \right\rceil .$$

In the separation algorithm to identify violated inequality among all inequalities (3.27), (3.29), and (3.32) the following steps are applied. In the first step, inequality (3.27) is evaluated. Afterwards, for every $k \in \{1, \ldots, \lceil 1/\delta \rceil\}$ and every $F_k \subseteq F$ with $|F_k| = |F| - k$, one inequality (3.29) and one inequality (3.32) is evaluated. The inequality (3.29) is the result of Algorithm 6, the separation algorithm for knapsack-partition inequalities (3.20), applied to

$$\begin{array}{rcl} c_i & := & C^{\tau_i},\ i = 1, \ldots, n , \\ \overline{z}_i & := & \sum_{e \in F_k} \overline{x}(e, \tau_i),\ i = 1, \ldots, n , \quad \text{and} \\ d & := & d - k\delta d . \end{array}$$

Eventually, setting $L_k := (d - (k-1)\delta d, \delta d, C^{\tau_1}, |F_k| + 1) - (d - k\delta d, \delta d, C^{\tau_1}, |F_k|)$, the inequality (3.32) can be evaluated.

Strengthened knapsack-partition inequalities

Suppose that a k-graph-partition $V_1, \ldots, V_k$ is given such that the reservation parameter is greater than zero for some demand edges with end-nodes in different shores, that is,

$$\sum_{uv \in \delta_H(V_1, \ldots, V_k)} \rho_{uv} d_{uv} \quad > \quad 0 .$$

The separation algorithm for strengthened knapsack-partition inequalities (3.36) is based on Algorithm 6, which is applied with

$$\begin{array}{rcl} c_i & := & C^{\tau_i},\ i = 1, \ldots, n , \\ \overline{z}_i & := & \sum_{e \in \delta_G(V_1, \ldots, V_k)} \overline{x}(e, \tau_i),\ i = 1, \ldots, n , \quad \text{and} \\ d & := & \min\{\sum_{uv \in \delta_H(V_1, \ldots, V_k)} \rho_{uv} d_{uv} - \sum_{e \in \delta_G(V_1, \ldots, V_k) \setminus \{h\}} C_e^0 : h \in \delta_G(V_1, \ldots, V_k)\} . \end{array}$$

as input. As result of Algorithm 6, let $\{l_1, \ldots, j_1\}, \ldots, \{l_t, \ldots, j_t\}$ be a partition of the index set of $\{C^{\tau_1}, \ldots, C^{\tau_n}\}$ w.r.t. d, let $\kappa_1, \ldots, \kappa_t$ be defined as in (3.18), and for $i = 1, \ldots, n$, let $p(i)$ the partition index of technology τ_i, that is, $i \in \{l_{p(i)}, \ldots, j_{p(i)}\}$. If

$$\sum_{e\in F} \sum_{i=1}^{n} \min\left\{\kappa_{p(i)}, \frac{C^{\tau_i}}{C^{\tau_{l_{p(i)}}}}\right\} \cdot \left(\prod_{s=1}^{p(i)-1} \kappa_s\right) \overline{x}(e,\tau_i) \quad < \quad \left\lceil \frac{|F|}{|F|-1} \prod_{s=1}^{p} \kappa_s \right\rceil ,$$

then a violated inequality (3.36) has been identified.

4.5.4 Computational tests

In the preceding sections, we presented separation algorithms for particular classes of inequalities. Now, we focus on the problem how to combine these algorithms in order to compute a lower bound with the cutting plane algorithm efficiently with respect to the following performance measures:

- the *value* of the final lower bound,
- the *time* needed to compute this lower bound, and
- the *size* of the final linear programming relaxation in terms of rows and coefficients.

As we sketched in Algorithm 3 (see page 122), the general strategy is to maintain a pool of valid inequalities for $Y(G, H, \cdot)$ and to use them to identify valid inequalities for $X(G, H, \cdot, \cdot)$, which are violated by the current optimal solution of the linear programming relaxation. In each iteration of this algorithm, the pool of valid inequalities for $Y(G, H, \cdot)$ is updated, and a sequence of separation algorithms is applied for valid inequalities of the knapsack-relaxations induced by the inequalities in the pool (see Definitions 3.5 and 3.42). At the end of each iteration, inequalities are removed from the pool if a specified maximum number of separation algorithms failed for the respective induced knapsack-relaxation, and identified violated inequalities are added to the linear programming relaxation which is then reoptimized.

The influence of different run-time parameter selections on the mentioned performance measures is evaluated for the two main components of the cutting plane algorithm: the separation algorithms for valid inequalities for $Y(G, H, \cdot)$, and the separation algorithms for valid inequalities for $X(G, H, \cdot, \cdot)$. We present results of computational experiments for combinations of the capacity models Discrete Capacities and Divisible Basic Capacities, and the survivability models Diversification and Reservation. Additional experiments for Path Restoration are not necessary, since no inequalities other than those presented for Reservation are known for the polyhedra $X(G, H, \text{Fin}, \text{Path})$ and $X(G, H, \text{Bas}, \text{Path})$. The tests are performed for the problem instances m2, l2, and p2 with the survivability parameters $\delta = 0.5$ and $\delta = 0.75$ for Diversification, and $\rho = 0.5$ and $\rho = 1.0$ for Reservation. The respective problem names associated with these parameters are m2d50, m2d75, l2d50, l2d75, p2d50, and p2d75 for Diversification, and m2r50, m2r100, l2r50, l2r100, p2r50, and p2r100 for Reservation.

Run-time parameter for separation of metric inequalities

First, run-time parameters for separation algorithms related to classes of valid inequalities for $Y(G, H, \cdot)$ are examined. For metric inequalities (3.1) and (3.2) (the respective versions for the survivability models Diversification and Reservation), we presented in Section 4.5.1 an exact separation algorithm based on the algorithm to test feasibility of a capacity vector, and for k-graph-partition inequalities (2.23) we presented Algorithm 4, a heuristic separation algorithm based on shrinking. Both algorithms are applied, the one for metric inequalities every M iterations, and the one for k-graph-partition inequalities every P iterations. For both these parameters, the values $0, 5$, and 10 are compared. The target number of nodes in the shrunken supply and demand graphs within Algorithm 4 is set to 10 and k is set to 2. A generated k-graph-partition inequality is added to the pool if its right-hand side is not more than 3 percent larger than the left-hand side evaluated with the current capacity vector $\overline{y}$ (which is obtained from the current linear programming relaxation). The initial pool of valid inequalities for $Y(G, H, \cdot)$ comprises all minimal cut inequalities with no more than two nodes in the smaller shore, and all minimal 3-graph-partition inequalities with no more than two nodes in all but the largest shore.

For classes of valid inequalities for $X(G, H, \cdot, \cdot)$, the applied separation algorithms are listed in Tables 4.3 and 4.4. The left and right column of Table 4.3 show the applied separation algorithms for the survivability model Diversification in combination with the capacity models Discrete Capacities and Divisible Basic Capacities, respectively.

Discrete Capacities	Divisible Basic Capacities
strengthened metric inequalities (3.6)	strengthened metric inequalities (3.16)
diversification-band inequalities (3.12)	diversification-cut inequalities (3.27) (3.32)
3-graph-partition band inequalities (3.10)	knapsack-partition inequalities (3.20), (3.29)
2-band inequalities (3.9)	

Table 4.3: Diversification: Applied separation algorithms for $X(G, H, \cdot, \mathrm{Div})$

Analogously, the left and right column of Table 4.4 show the applied separation algorithms for the survivability model Reservation in combination with the capacity models Discrete Capacities and Divisible Basic Capacities, respectively.

Discrete Capacities	Divisible Basic Capacities
strengthened metric inequalities (3.6)	strengthened metric inequalities (3.16)
strengthened band inequalities (3.13)	strengthened knapsack-partition inequalities (3.36)
strengthened 2-band inequalities (3.15)	

Table 4.4: Reservation: Applied separation algorithms for $X(G, H, \cdot, \mathrm{Res})$

These separation algorithms are applied in the order implicitly defined in the Tables 4.3 and 4.4 (from top to bottom). In this first test series, the number of possibly identified inequalities in a single iteration is unbounded, and every identified violated inequality is added to the linear programming relaxation. Furthermore, inequalities are never removed from the pool. The cutting plane algorithm terminates, if no improvement in the lower bound calculation can be obtained.

Name	M	P	Discrete Capacities						Divisible Basic Capacities					
			init	lower	time	iter	rows	coeffs	init	lower	time	iter	rows	coeffs
m2d50	5	5	36.84	102.27	39	92	780	16464	36.84	107.00	18	45	406	10733
	5	0	36.84	101.71	48	121	715	17301	36.84	108.26	58	178	436	16829
	5	10	36.84	102.04	33	82	761	15842	36.84	**112.29**	30	87	436	13442
	0	5	36.84	93.43	3	18	644	9686	36.84	97.50	**1**	9	352	7901
	0	0	36.84	86.95	**1**	13	604	8398	36.84	89.44	**1**	3	328	7010
	0	10	36.84	93.63	4	29	672	9875	36.84	97.50	**1**	10	352	7901
	10	5	36.84	101.42	32	112	777	19024	36.84	111.09	33	100	429	13391
	10	0	36.84	101.71	52	131	719	17334	36.84	108.26	58	188	436	16829
	10	10	36.84	**102.33**	48	144	788	21106	36.84	110.64	28	90	432	12935
m2d75	5	5	36.84	94.38	33	112	796	20273	36.84	100.23	13	55	371	9770
	5	0	36.84	93.61	38	158	717	20643	36.84	99.24	18	94	372	11033
	5	10	36.84	**94.52**	33	111	801	19756	36.84	100.72	16	61	384	10625
	0	5	36.84	85.24	**1**	8	612	8746	36.84	94.32	1	10	338	7457
	0	0	36.84	83.06	**1**	10	600	8301	36.84	88.74	**1**	3	320	6818
	0	10	36.84	89.27	5	30	689	10380	36.84	94.32	**1**	11	338	7457
	10	5	36.84	94.45	34	137	816	22621	36.84	**101.31**	14	55	380	9770
	10	0	36.84	93.61	39	174	725	20731	36.84	99.24	18	99	372	11033
	10	10	36.84	94.46	32	119	752	18526	36.84	100.23	13	58	371	9770
l2d50	5	5	38.55	123.46	12:57	496	1663	114149	38.73	**133.52**	11:01	313	1095	52288
	5	0	38.55	122.11	18:41	470	1553	98739	38.73	128.79	10:18	207	968	35167
	5	10	38.55	**124.12**	11:40	295	1573	67882	38.73	131.34	5:12	141	997	31951
	0	5	38.55	101.78	11	20	1277	16678	38.73	105.15	7	11	867	18304
	0	0	38.55	97.09	**5**	11	1243	15371	38.73	99.65	**3**	3	842	17128
	0	10	38.55	100.80	11	22	1280	17039	38.73	105.15	7	13	867	18304
	10	5	38.55	123.60	13:35	542	1625	109432	38.73	131.95	9:56	225	1030	38014
	10	0	38.55	121.21	13:45	386	1487	75571	38.73	128.79	10:10	219	968	35167
	10	10	38.55	122.46	9:47	337	1509	71532	38.73	**133.52**	11:40	331	1095	52288
l2d75	5	5	38.55	**107.44**	8:50	303	1484	63544	38.73	121.46	4:53	249	1021	43486
	5	0	38.55	104.81	10:55	407	1473	84883	38.73	118.14	5:49	254	975	40300
	5	10	38.55	105.93	9:54	341	1463	69474	38.73	120.73	5:33	247	1011	40891
	0	5	38.55	89.06	8	16	1226	15571	38.73	107.13	7	9	852	17899
	0	0	38.55	86.34	**4**	9	1195	14661	38.73	99.22	**3**	3	824	16654
	0	10	38.55	89.07	7	14	1224	15608	38.73	107.13	7	10	852	17899
	10	5	38.55	106.43	7:07	368	1511	77978	38.73	**122.02**	5:35	297	1044	47401
	10	0	38.55	104.40	7:35	284	1399	55123	38.73	118.14	5:41	268	975	40300
	10	10	38.55	105.63	6:40	237	1411	46207	38.73	121.46	4:55	263	1021	43486
p2d50	5	5	38.70	**105.65**	49	50	1329	27532	38.70	**112.95**	7	7	522	17312
	5	0	38.70	100.24	14	19	1200	20889	38.70	103.85	2	3	499	15854
	5	10	38.70	104.08	1:15	78	1283	29715	38.70	**112.95**	7	7	522	17312
	0	5	38.70	95.83	8	17	1178	20477	38.70	112.48	3	5	519	17081
	0	0	38.70	94.71	**5**	13	1155	19638	38.70	103.85	**1**	2	498	15818
	0	10	38.70	95.67	9	22	1180	20751	38.70	112.48	3	5	519	17081
	10	5	38.70	101.77	22	31	1252	22776	38.70	**112.95**	7	7	522	17312
	10	0	38.70	101.45	15	25	1196	20666	38.70	103.85	2	3	499	15854
	10	10	38.70	104.30	39	48	1269	25049	38.70	**112.95**	7	7	522	17312
p2d75	5	5	38.70	96.81	19	26	1217	22881	38.70	**108.47**	7	10	503	16229
	5	0	38.70	94.40	10	14	1156	19895	38.70	103.11	2	4	482	14879
	5	10	38.70	96.12	17	26	1212	22600	38.70	**108.47**	7	10	503	16229
	0	5	38.70	90.87	6	12	1156	20164	38.70	107.04	3	6	499	15959
	0	0	38.70	90.00	**4**	9	1140	19466	38.70	103.11	**1**	3	481	14843
	0	10	38.70	91.12	7	16	1170	21131	38.70	107.04	3	7	499	15959
	10	5	38.70	**97.92**	25	30	1236	24212	38.70	**108.47**	7	11	503	16229
	10	0	38.70	95.72	29	48	1209	25134	38.70	103.11	2	4	482	14879
	10	10	38.70	96.05	20	26	1188	21873	38.70	**108.47**	7	11	503	16229

Table 4.5: Diversification: Separation algorithms for $Y(G, H, \cdot)$

Name	M	P	Discrete Capacities						Divisible Basic Capacities					
			init	lower	time	iter	rows	coeffs	init	lower	time	iter	rows	coeffs
m2r50	5	5	36.84	**92.56**	7	15	630	9119	36.84	**86.39**	4	7	304	6383
	5	0	36.84	86.88	3	9	592	8240	36.84	80.63	2	3	295	6074
	5	10	36.84	91.61	9	20	629	9269	36.84	**86.39**	4	7	304	6383
	0	5	36.84	**92.56**	5	15	630	9119	36.84	**86.39**	**1**	7	303	6374
	0	0	36.84	86.88	**2**	8	592	8240	36.84	80.63	**1**	2	294	6065
	0	10	36.84	91.61	6	20	629	9269	36.84	**86.39**	**1**	7	303	6374
	10	5	36.84	**92.56**	7	15	630	9119	36.84	**86.39**	4	7	304	6383
	10	0	36.84	86.88	3	9	592	8240	36.84	80.63	2	3	295	6074
	10	10	36.84	91.61	8	20	629	9269	36.84	**86.39**	4	7	304	6383
m2r100	5	5	36.84	127.42	44	40	924	15545	36.84	**133.22**	27	30	432	10160
	5	0	36.84	127.19	34	34	860	14609	36.84	132.69	24	32	437	10250
	5	10	36.84	127.16	43	38	904	15940	36.84	**133.22**	27	30	432	10160
	0	5	36.84	106.40	31	32	836	11911	36.84	87.49	**1**	7	304	6374
	0	0	36.84	103.16	**10**	13	755	10113	36.84	81.59	**1**	2	297	6134
	0	10	36.84	106.40	28	31	847	11862	36.84	87.49	**1**	7	304	6374
	10	5	36.84	**127.90**	1:04	51	995	16606	36.84	**133.22**	27	32	432	10160
	10	0	36.84	126.92	44	48	908	15505	36.84	132.69	25	34	437	10250
	10	10	36.84	127.36	53	48	941	16081	36.84	**133.22**	27	32	432	10160
l2r50	5	5	38.55	99.38	27	10	1222	15187	38.73	89.56	36	7	712	12616
	5	0	38.55	98.04	25	9	1216	15076	38.73	**92.03**	46	8	713	12535
	5	10	38.55	100.57	1:06	19	1237	15837	38.73	89.56	36	7	712	12616
	0	5	38.55	99.55	17	15	1240	15801	38.73	82.83	2	5	699	12481
	0	0	38.55	96.55	**12**	11	1234	15319	38.73	81.84	**1**	2	697	12388
	0	10	38.55	100.05	21	19	1259	16631	38.73	82.83	2	5	699	12481
	10	5	38.55	99.99	48	15	1241	15754	38.73	89.56	37	7	712	12616
	10	0	38.55	98.04	28	12	1234	15327	38.73	**92.03**	47	9	713	12535
	10	10	38.55	**102.33**	51	18	1268	16898	38.73	89.56	36	7	712	12616
l2r100	5	5	38.55	145.33	6:13	60	1685	32696	38.73	148.77	7:47	52	935	22984
	5	0	38.55	144.92	5:23	54	1625	33200	38.73	147.18	6:05	46	915	21967
	5	10	38.55	145.32	5:29	52	1636	31683	38.73	**150.44**	6:23	48	930	22453
	0	5	38.55	119.65	**30**	13	1377	18187	38.73	82.83	2	5	699	12481
	0	0	38.55	119.18	40	16	1378	17685	38.73	81.84	**1**	2	697	12388
	0	10	38.55	120.22	1:05	21	1419	18417	38.73	82.83	2	5	699	12481
	10	5	38.55	**145.78**	7:08	71	1713	32194	38.73	149.54	7:43	54	926	22936
	10	0	38.55	144.97	6:50	62	1662	34257	38.73	147.18	6:06	49	915	21967
	10	10	38.55	145.58	6:54	66	1673	31968	38.73	148.77	7:52	55	935	22984
p2r50	5	5	38.70	**95.22**	39	17	1116	19345	38.70	**96.67**	19	10	472	13898
	5	0	38.70	94.10	20	10	1091	18499	38.70	94.02	16	10	462	13472
	5	10	38.70	94.11	26	11	1095	18771	38.70	**96.67**	19	10	472	13898
	0	5	38.70	95.21	29	17	1113	19294	38.70	93.98	2	5	451	13544
	0	0	38.70	94.10	**13**	9	1090	18487	38.70	91.01	**1**	2	445	13148
	0	10	38.70	94.99	22	14	1108	18918	38.70	93.98	**1**	5	451	13544
	10	5	38.70	95.21	41	18	1113	19294	38.70	**96.67**	20	11	472	13898
	10	0	38.70	94.10	20	10	1090	18487	38.70	94.02	16	11	462	13472
	10	10	38.70	94.99	33	14	1108	18918	38.70	**96.67**	19	11	472	13898
p2r100	5	5	38.70	**129.19**	8:23	77	1701	52981	38.70	**133.83**	2:34	49	690	30071
	5	0	38.70	129.09	7:42	85	1707	61078	38.70	131.82	2:27	43	654	25457
	5	10	38.70	128.55	7:31	78	1693	58276	38.70	**133.83**	2:35	49	690	30071
	0	5	38.70	107.92	2:16	20	1369	24463	38.70	96.56	3	10	474	14933
	0	0	38.70	107.63	3:19	22	1402	24402	38.70	91.43	**1**	2	450	13340
	0	10	38.70	107.93	**2:11**	20	1386	24389	38.70	96.56	3	11	474	14933
	10	5	38.70	128.64	10:33	95	1804	59871	38.70	**133.83**	2:34	52	690	30071
	10	0	38.70	128.85	7:58	79	1753	57495	38.70	131.82	2:29	45	654	25457
	10	10	38.70	129.05	8:21	88	1766	60334	38.70	**133.83**	2:35	52	690	30071

Table 4.6: Reservation: Separation algorithms for $Y(G, H, \cdot)$

The computational results are presented in Tables 4.5 and 4.6. Both tables have the following format. Column 1 (NAME) gives the name of the problem which also encodes the survivability parameters. The next two columns provide run-time parameters. The number in the second and third column specify after how many iterations the separation algorithm for metric inequalities (M) and k-graph-partition inequalities (P) is applied. Six columns with statistics are following for each of the two capacity models. From left to right, these columns give the scaled[2] value of the initial (INIT) and the final lower bound (LOWER), the time to compute this lower bound (TIME), the number of iterations (ITER), the number of rows in the final linear programming relaxation (ROWS), and the number of non-zero coefficients in this relaxation (COEFFS).

The main observation is the following: Without application of the separation algorithm for metric inequalities ($M = 0$), the lower bound is $10 - 20$ percent smaller than in the other cases ($M = 5, 10$). The influence of the separation algorithm for k-graph-partition inequalities is of minor importance. For all problem instances, the value of the lower bound is larger for $P > 0$, but the improvement is only about 1 percent. Interestingly, the lower bound obtained with the initial pool of k-graph-partition inequalities ($M = 0, P = 0$) is often close to the lower bound obtained using the separation algorithm for k-graph-partition inequalities only ($M = 0, P > 0$). This indicates that the initial pool is well-chosen.

Large improvements of the initial lower bound can be observed. In all cases shown in Tables 4.5 and 4.6, the final lower bound is more than 250 percent larger than the initial lower bound. Since the initial linear programming relaxation contains only normal operating state constraints, it is not surprising that the improvements for the $\delta = 0.5$ and $\rho = 1.0$ instances are larger than those of the corresponding $\delta = 0.75$ and $\rho = 0.5$ instances. The running times are small; $1 - 10$ minutes are needed to compute the best lower bounds. For each problem instance and for $M > 0$, the times are almost identical. As expected, this indicates that most time is spent in the separation algorithm for metric inequalities. The other values, the number of iterations and the number of rows and coefficients of the final linear programming relaxation, are smaller if $M = 0$, but for $M > 0$ all these values are in the same range.

The following parameter setting is the consequence of these tests: in all subsequent computations, the separation algorithm for metric inequalities is applied every 5 iteration for DIVERSIFICATION, and every 10 iterations for RESERVATION (and PATH RESTORATION). Furthermore, the separation algorithm for k-graph-partition inequalities is applied every 10 iterations (independent from the survivability model).

General run-time parameters for separation

The second test series aims at the influence of more general run-time parameters. The separation algorithms for valid inequalities for $Y(G, H, \cdot)$ are again applied in the order that is implicitly defined in Tables 4.3 and 4.4 (from top to bottom). In this test series, the number of possibly identified violated inequalities in a single iteration attains the values 10 and 50. An identified violated inequality is only added to the linear programming relaxation, if the violation is larger than a specified minimum percentage of its right-hand side. The values 1 percent and 0.1 percent are compared. The cutting plane algorithm terminates, if the improvement of the lower

[2]All lower bounds and solution values are scaled to comply with disclosure agreements.

bound in a single iteration is smaller than a specified percentage, which attains the values 0.1 percent and 0.5 percent.

The results of this test series are presented in Tables 4.7 and 4.8, which have a format similar to that of Tables 4.5 and 4.6. In contrast to the latter tables, columns 2 and 3 are replaced by three new columns. These show the maximum number of violated inequalities added in a single iteration (VIO), the required minimum percentage violation of added inequalities (SLACK), and the required minimum percentage of the lower bound improvement in a single iteration (IMP).

The main observations are the following: The running times to compute a lower bound for the optimal solution value can be further decreased by appropriate parameter settings. For DIVERSIFICATION, all lower bounds are calculated in less than 1 minute, and for RESERVATION in less than 3 minutes in all instances but l2r100, where almost 8 minutes are needed. Important to note is that the quality of the computed lower bounds is the same as presented for in Tables 4.5 and 4.6. The lower bounds are in some cases larger and in some cases smaller than the previous ones. Comparing the lower bounds in dependence of the capacity model, we observe that the lower bounds for DIVISIBLE BASIC CAPACITIES tend to be better than those for DISCRETE CAPACITIES. This is interesting since there are more possible capacity choices for DIVISIBLE BASIC CAPACITIES. This indicates that the (mixed-integer rounding) inequalities for DIVISIBLE BASIC CAPACITIES are practically more successful than the (knapsack-cover) inequalities for DISCRETE CAPACITIES.

The influence of the number of inequalities added in one iteration is small. The same holds for the minimal slack such that an inequality is considered violated. The minimum required improvement of the lower bound in a single iteration has the largest influence on the quality of the computed lower bound. The cutting plane algorithm terminates too fast, if the respective parameter is set to 0.5 percent. The other values, the number of iterations and the number of rows and coefficients of the final linear programming relaxation, are independent from these run-time parameters.

The following parameter setting is the consequence of these tests: in all subsequent computations, the cutting plane algorithm terminates if the improvement in a single iteration is less than 0.1 percent, an identified violated inequality is only added to the linear programming relaxation, if the violation is larger than 0.1 percent, and a maximum of 10 inequalities is added in each iteration.

NAME	VIO	SLACK	IMP	Discrete Capacities						Divisible Basic Capacities					
				INIT	LOWER	TIME	ITER	ROWS	COEFFS	INIT	LOWER	TIME	ITER	ROWS	COEFFS
m2d50	50	0.10	0.5	36.84	100.03	10	16	746	12058	36.84	103.57	**5**	14	370	8687
	50	0.10	0.1	36.84	100.86	18	32	760	13165	36.84	109.60	13	30	404	10283
	50	1.00	0.5	36.84	100.11	11	18	722	11537	36.84	103.57	**5**	14	370	8687
	50	1.00	0.1	36.84	**102.21**	21	33	771	13823	36.84	109.60	14	30	404	10283
	10	0.10	0.5	36.84	98.42	**9**	34	760	11018	36.84	110.18	7	25	376	8915
	10	0.10	0.1	36.84	99.88	17	44	806	12456	36.84	**111.06**	9	28	382	9284
	10	1.00	0.5	36.84	98.73	**9**	27	696	10113	36.84	102.82	**5**	19	360	8246
	10	1.00	0.1	36.84	101.62	22	44	809	12983	36.84	106.64	11	32	382	9263
m2d75	50	0.10	0.5	36.84	92.46	8	14	738	12087	36.84	96.42	**3**	9	341	7643
	50	0.10	0.1	36.84	93.56	10	23	770	12979	36.84	99.16	7	24	359	8675
	50	1.00	0.5	36.84	92.69	**6**	13	715	12024	36.84	96.42	**3**	9	340	7622
	50	1.00	0.1	36.84	93.61	10	21	759	13226	36.84	99.16	7	24	358	8654
	10	0.10	0.5	36.84	91.36	7	24	662	10258	36.84	97.97	4	16	328	7121
	10	0.10	0.1	36.84	**94.44**	16	42	756	12846	36.84	97.97	4	16	328	7121
	10	1.00	0.5	36.84	93.64	14	34	766	12685	36.84	98.38	5	19	337	7598
	10	1.00	0.1	36.84	94.19	17	48	766	13370	36.84	**100.43**	9	34	355	8447
l2d50	50	0.10	0.5	38.55	116.49	44	19	1333	19408	38.73	122.60	23	16	868	18595
	50	0.10	0.1	38.55	115.83	1:16	29	1348	21313	38.73	122.67	25	18	869	18661
	50	1.00	0.5	38.55	113.29	36	17	1317	18852	38.73	122.60	**21**	16	872	18676
	50	1.00	0.1	38.55	118.38	1:38	31	1356	21336	38.73	122.67	25	18	873	18742
	10	0.10	0.5	38.55	111.53	**33**	38	1368	17406	38.73	125.07	28	30	873	19345
	10	0.10	0.1	38.55	117.11	1:27	56	1398	19231	38.73	**125.22**	31	32	884	19864
	10	1.00	0.5	38.55	**120.22**	59	49	1428	20054	38.73	122.41	30	30	881	19804
	10	1.00	0.1	38.55	119.32	1:21	56	1434	21251	38.73	123.63	25	30	872	19012
l2d75	50	0.10	0.5	38.55	97.10	18	13	1294	17005	38.73	115.83	**19**	15	840	17497
	50	0.10	0.1	38.55	**105.42**	1:15	45	1368	22573	38.73	**117.37**	38	29	856	18688
	50	1.00	0.5	38.55	97.08	**17**	13	1274	16622	38.73	115.83	20	15	841	17491
	50	1.00	0.1	38.55	103.14	47	31	1305	18240	38.73	**117.37**	39	29	857	18685
	10	0.10	0.5	38.55	100.26	36	33	1315	17456	38.73	115.86	20	29	844	17599
	10	0.10	0.1	38.55	100.00	38	39	1320	17334	38.73	116.39	26	33	847	17695
	10	1.00	0.5	38.55	97.98	32	29	1286	16830	38.73	115.64	22	29	851	18052
	10	1.00	0.1	38.55	104.75	1:18	56	1370	21210	38.73	116.69	29	32	853	18172
p2d50	50	0.10	0.5	38.70	100.88	22	16	1240	22720	38.70	109.76	**7**	7	512	16601
	50	0.10	0.1	38.70	101.07	28	25	1256	23424	38.70	**111.88**	15	17	527	17522
	50	1.00	0.5	38.70	101.23	20	14	1209	21551	38.70	109.76	**7**	7	512	16601
	50	1.00	0.1	38.70	101.01	22	18	1233	22849	38.70	**111.88**	14	17	527	17522
	10	0.10	0.5	38.70	103.20	39	54	1373	24116	38.70	111.22	15	22	525	17240
	10	0.10	0.1	38.70	**105.70**	53	71	1420	28707	38.70	111.22	16	22	525	17240
	10	1.00	0.5	38.70	93.19	**16**	30	1121	18527	38.70	106.59	11	17	502	15965
	10	1.00	0.1	38.70	104.94	43	59	1331	25278	38.70	106.59	11	17	502	15965
p2d75	50	0.10	0.5	38.70	94.84	13	12	1229	22867	38.70	109.47	**7**	8	506	16106
	50	0.10	0.1	38.70	94.95	14	15	1223	21562	38.70	**110.71**	9	12	508	16343
	50	1.00	0.5	38.70	94.91	14	14	1210	21232	38.70	109.47	**7**	8	506	16106
	50	1.00	0.1	38.70	94.91	13	14	1210	21232	38.70	**110.71**	9	12	508	16343
	10	0.10	0.5	38.70	90.26	16	30	1119	18082	38.70	107.97	10	19	496	15434
	10	0.10	0.1	38.70	**97.30**	31	53	1283	24309	38.70	109.25	12	24	499	15617
	10	1.00	0.5	38.70	84.76	**9**	16	978	15869	38.70	110.37	14	23	507	16214
	10	1.00	0.1	38.70	96.00	31	50	1293	22590	38.70	110.37	15	24	507	16214

Table 4.7: Diversification: Influence of separation run-time parameters

NAME	VIO	SLACK	IMP	Discrete Capacities						Divisible Basic Capacities					
				INIT	LOWER	TIME	ITER	ROWS	COEFFS	INIT	LOWER	TIME	ITER	ROWS	COEFFS
m2r50	50	0.10	0.5	36.84	92.36	**7**	13	637	9089	36.84	86.39	4	8	300	6257
	50	0.10	0.1	36.84	**93.55**	9	17	652	9560	36.84	86.39	4	10	300	6257
	50	1.00	0.5	36.84	92.49	**7**	13	632	9122	36.84	86.39	4	8	300	6257
	50	1.00	0.1	36.84	93.50	9	17	640	9360	36.84	86.39	5	10	300	6257
	10	0.10	0.5	36.84	91.64	10	22	638	8594	36.84	**88.47**	**3**	12	287	5708
	10	0.10	0.1	36.84	91.91	9	24	648	8523	36.84	**88.47**	4	14	287	5708
	10	1.00	0.5	36.84	92.90	10	24	642	8691	36.84	86.85	4	11	286	5714
	10	1.00	0.1	36.84	91.62	8	24	646	8654	36.84	86.85	4	11	286	5714
m2r100	50	0.10	0.5	36.84	121.55	**16**	17	870	13787	36.84	130.13	**14**	20	406	9599
	50	0.10	0.1	36.84	126.93	26	27	916	15555	36.84	**133.19**	25	28	427	10367
	50	1.00	0.5	36.84	126.51	25	26	906	15474	36.84	130.13	15	20	406	9599
	50	1.00	0.1	36.84	126.85	29	30	912	15786	36.84	**133.19**	24	28	427	10367
	10	0.10	0.5	36.84	126.74	32	50	924	16141	36.84	132.10	22	29	377	8585
	10	0.10	0.1	36.84	**127.52**	38	60	963	16601	36.84	131.14	23	28	381	8723
	10	1.00	0.5	36.84	127.19	33	50	897	14699	36.84	132.58	26	34	409	9722
	10	1.00	0.1	36.84	127.23	35	52	911	14785	36.84	133.03	25	34	406	9968
l2r50	50	0.10	0.5	38.55	97.84	**43**	11	1248	15865	38.73	94.40	55	12	773	14719
	50	0.10	0.1	38.55	**102.11**	1:06	18	1277	17052	38.73	96.28	1:20	15	774	14740
	50	1.00	0.5	38.55	100.33	**43**	11	1248	16119	38.73	93.64	**44**	8	770	14578
	50	1.00	0.1	38.55	100.40	46	14	1253	16157	38.73	93.64	51	10	770	14578
	10	0.10	0.5	38.55	99.78	1:30	30	1289	16684	38.73	97.55	1:01	15	745	13858
	10	0.10	0.1	38.55	101.26	1:30	34	1311	17049	38.73	97.55	58	15	745	13858
	10	1.00	0.5	38.55	91.62	1:03	16	1156	14973	38.73	**98.87**	55	15	757	14131
	10	1.00	0.1	38.55	100.36	1:34	31	1287	16236	38.73	**98.87**	1:17	18	757	14131
l2r100	50	0.10	0.5	38.55	137.99	1:19	18	1544	22817	38.73	131.76	**1:36**	21	872	19582
	50	0.10	0.1	38.55	141.88	2:14	27	1628	26879	38.73	**150.20**	7:58	57	1054	27790
	50	1.00	0.5	38.55	137.53	**1:19**	18	1517	22589	38.73	131.93	1:40	23	876	20281
	50	1.00	0.1	38.55	145.08	4:22	43	1652	31669	38.73	134.66	2:27	28	906	21745
	10	0.10	0.5	38.55	144.58	5:50	85	1815	34917	38.73	148.99	6:03	58	1012	25957
	10	0.10	0.1	38.55	144.88	6:42	91	1835	36686	38.73	149.53	7:09	60	1002	25315
	10	1.00	0.5	38.55	125.87	2:33	52	1533	26601	38.73	149.64	6:30	52	975	23740
	10	1.00	0.1	38.55	**145.34**	6:59	92	1868	38104	38.73	149.70	6:36	54	985	23761
p2r50	50	0.10	0.5	38.70	95.63	29	11	1121	19540	38.70	97.39	22	10	465	13880
	50	0.10	0.1	38.70	95.76	31	13	1130	19864	38.70	**100.17**	31	19	481	15143
	50	1.00	0.5	38.70	95.34	**27**	11	1111	19117	38.70	97.39	22	10	465	13880
	50	1.00	0.1	38.70	94.76	28	12	1109	19125	38.70	**100.17**	33	19	481	15143
	10	0.10	0.5	38.70	92.62	38	32	1141	18481	38.70	94.09	**16**	11	439	12212
	10	0.10	0.1	38.70	**97.39**	49	42	1185	20745	38.70	94.09	**16**	11	439	12212
	10	1.00	0.5	38.70	88.43	**27**	21	1028	16593	38.70	94.09	**16**	11	439	12179
	10	1.00	0.1	38.70	95.70	39	36	1146	19255	38.70	94.09	**16**	11	439	12179
p2r100	50	0.10	0.5	38.70	121.11	**1:01**	15	1425	29356	38.70	123.71	**30**	16	548	19487
	50	0.10	0.1	38.70	124.71	1:38	28	1562	40385	38.70	**134.66**	2:26	40	666	26951
	50	1.00	0.5	38.70	121.02	1:12	15	1413	28841	38.70	123.71	**30**	16	547	19352
	50	1.00	0.1	38.70	124.45	2:01	30	1545	38467	38.70	134.62	3:34	49	659	26651
	10	0.10	0.5	38.70	113.22	1:36	53	1346	32437	38.70	132.59	2:28	44	627	24800
	10	0.10	0.1	38.70	128.31	4:15	103	1788	56310	38.70	133.33	2:39	47	637	25334
	10	1.00	0.5	38.70	113.10	1:43	53	1351	32644	38.70	131.73	2:33	47	643	25931
	10	1.00	0.1	38.70	**128.49**	3:49	94	1625	50127	38.70	134.42	3:17	58	660	26996

Table 4.8: Reservation: Influence of separation run-time parameters

4.6 Heuristics

With todays computing power and mathematical methodology, it is impossible to solve real-world problem instances for the survivable capacitated network design problems described in Chapter 2 to proven optimality. Hence, it is necessary to develop heuristic algorithms to compute feasible solutions. Almost all heuristics described in this section depend on the cutting plane algorithm, since the value of the (fractional) capacity variables is exploited to guide run-time decisions. It is out of scope to guarantee optimality of the best solutions computed, but as good news, the employed cutting plane algorithm terminates with a lower bound for the optimal solution value, implying that a quality guarantee can be provided.

We do not devise any randomized heuristic algorithms, like *simulated annealing*, *genetic algorithms*, *tabu search*, etc., since the application of such algorithms is most successful if not much about structural properties of the underlying problem is known. The theory developed in Chapter 3, however, particularly focuses on the structure of survivable capacitated network design problem. By itself, this is not a sufficient reason to neglect this type of randomized heuristics, but, as a further difficulty for the network design problems under consideration, it is time consuming to test feasibility of capacity vectors, see Section 4.4. This is a serious drawback, since these randomized heuristics can only provide good solutions if it is possible to search large neighborhoods of solutions in short time.

We proceed now as follows. First, the *starting* heuristics used to calculate initial feasible solutions are presented in Section 4.6.1. The subsequent Section 4.6.2, contains the description of *improvement* heuristics, which obtain as input a feasible solution and modify this solution, until it is locally optimal with respect to a given neighborhood. Finally, results of computational experiments are reported in Section 4.6.3.

4.6.1 Starting heuristics

We implemented one class of starting heuristics, the so-called *branch&cut path heuristics*. These heuristics depend the linear programming relaxation and the separation algorithms. In fact, these heuristics are based on the ideas of a branch&cut algorithm. A heuristic of this class follows a specific path in the branch&bound tree and applies at each node of the tree a cutting plane algorithm. In contrast to a branch&cut algorithm, the heuristic does not examine alternative subtrees. A general description of the branch&cut path heuristic is given in Algorithm 7.

In every iteration, (fractional) capacity variables $\overline{x}$ are given as the solution of the current linear programming relaxation. According to some criterion, a supply edge $e \in E$ with at least one fractional capacity variable $\overline{x}(e,t)$ is selected. Here, the interpretation of t depends on the capacity model. Either $t \in \{1, \ldots, T_e\}$ for DISCRETE CAPACITIES, or $t \in T$ for DIVISIBLE BASIC CAPACITIES. Then, an integer lower bound $l(e,t) \in \mathbb{N}$ for at least one capacity variable of edge e is determined such that the current solution $\overline{x}$ is no longer valid, that is, $\overline{x}(e,t) < l(e,t)$. These lower bounds for the integer capacity variables are then set in the current relaxation $\overline{lp}$. At the end of each iteration, separation algorithms are used to identify valid inequalities for $X(G, H, \cdot, \cdot)$ which are violated by the current $\overline{x}$. However, in order to reduce the overall running time of these starting heuristics, not all separation algorithms are used in this step.

Algorithm 7 BRANCHANDCUTPATHHEURISTIC

Require: capacity variables $\overline{x}$ as solution of the current relaxation $\overline{lp}$
  while $\overline{x}$ **not** integer **do**
    choose supply edge $e \in E$ with fractional capacity variable
    choose lower bounds for the capacity variables of e
    set the bounds of these variables in the relaxation $\overline{lp}$ accordingly
    run the cutting plane algorithm and update $\overline{x}$
  end while
  compute capacities $\overline{y}$ from $\overline{x}$ (according to (2.4) or (2.7))
  if $\overline{y}$ **not** feasible **then**
    postprocessing
  end if

Eventually, all capacity variables $\overline{x}$ are integer. If the associated capacity vector $\overline{y}$, which is calculated from equation (2.4) or (2.7), is feasible, the heuristic terminates. However, the final integer solution may not be feasible, since the identified violated inequalities do not necessarily suffice to describe the respective polyhedron $Y(G, H, \cdot)$. This may happen, for instance, if no separation algorithm for metric inequalities is used during the cutting plane algorithm, or if PATH RESTORATION is the survivability model. To overcome the problem of an infeasible solution at the end of these starting heuristics, a postprocessing algorithm is utilized to compute a feasible solution from the final integer, but infeasible, solution.

In what follows, we describe the selection criteria for the supply edges with fractional capacity variables, and the different strategies to set integer lower bounds for the capacity variables of the chosen edge. We present these criteria separately for the two capacity models, and describe afterwards the postprocessing and the results of computational experiments.

DISCRETE CAPACITIES

Suppose that capacity variables $\overline{x}(e,t)$, $e \in E$, $t = 1, \ldots, T_e$, as the solution of the current relaxation are given, and let $\overline{y}(e)$, $e \in E$, be the corresponding capacity vector. Furthermore, let F be the subset of supply edges with at least one fractional capacity variable, that is,

$$F \quad := \quad \{e \in E : \exists t \in \{1, \ldots, T_e\} \text{ with } 0 < \overline{x}(e,t) < 1\} \; .$$

The selection criteria for a supply edge from F are the following:

MAXFRAC Choose a supply edge with the largest fractional variable, that is, $(\epsilon, \tau) \in T(F)$ such that

$$\overline{x}(\epsilon, \tau) \quad = \quad \max_{(e,t) \in T(F)} \{\overline{x}(e,t) : 0 < \overline{x}(e,t) < 1\} \; .$$

MAXSUMFRAC Choose a supply edge with the largest sum of fractional variables, that is,

$\epsilon \in F$ such that

$$\sum_{t=1,\ldots,T_\epsilon : \overline{x}(\epsilon,t)<1} \overline{x}(\epsilon,t) \quad = \quad \max_{e\in F} \left\{ \sum_{t=1,\ldots,T_e : \overline{x}(e,t)<1} \overline{x}(e,t) \right\} .$$

MINCAP Choose a supply edge with fractional variable such that the additional capacity needed to make $\overline{y}(e)$ a breakpoint capacity is minimal, that is, $(\epsilon,\tau) \in T(F)$ such that

$$C_\epsilon^\tau - \overline{y}(\epsilon) \quad = \quad \min_{(e,t)\in T(F)} \left\{ C_e^t - \overline{y}(e) : C_e^t \geq \overline{y}(e) \right\} .$$

MINCOST Choose a supply edge with fractional variable such that the additional cost incurred from increasing to the respective capacity is minimal, that is, $(\epsilon,\tau) \in T(F)$ such that

$$K_\epsilon^\tau - \sum_{i=1}^{T_\epsilon} k_\epsilon^i \overline{x}(\epsilon,i) \quad = \quad \min_{(e,t)\in T(F)} \left\{ K_e^t - \sum_{i=1}^{T_e} k_e^i \overline{x}(e,i) : C_e^t \geq \overline{y}(e) \right\} .$$

MINRELCOST Choose a supply edge with fractional variable such that the additional cost relative to the additional capacity is minimal, that is, $(\epsilon,\tau) \in T(F)$ such that

$$(K_\epsilon^\tau - \sum_{i=1}^{T_\epsilon} k_\epsilon^i)/(C_\epsilon^\tau - \overline{y}(\epsilon)) \quad = \quad \min_{(e,t)\in T(F)} \left\{ (K_e^t - \sum_{i=1}^{T_e} k_e^i)/(C_e^t - \overline{y}(e)) : C_e^t > \overline{y}(e) \right\} .$$

MININCCOST Choose a supply edge with fractional variable such that the additional incremental cost is minimal, that is, $(\epsilon,\tau) \in T(F)$ such that

$$(1 - \overline{x}(\epsilon,\tau))k_\epsilon^\tau \quad = \quad \min_{(e,t)\in T(F)} \left\{ (1 - \overline{x}(e,t))k_e^t : 0 < \overline{x}(e,t) < 1 \right\} .$$

MINRELCAP Choose a supply edge with fractional variable such that the incremental capacity for the smallest breakpoint capacity greater than or equal to $\overline{y}(e)$ relative to this breakpoint capacity is minimal, that is, $(\epsilon,\tau) \in T(F)$ such that

$$(C_\epsilon^\tau - \overline{y}(\epsilon))/C_\epsilon^\tau \quad = \quad \min_{(e,t)\in T(F)} \{ (C_e^t - \overline{y}(e))/C_e^t : C_e^t \geq \overline{y}(e) > C_e^{t-1} \} .$$

Suppose that the supply edge $e \in E$ is decided. Then, the integer lower bounds for the capacity variables of e must be chosen such that the current solution $\overline{x}$ does not satisfy these bounds. This is done according to one of the following two strategies.

GREEDY Set $x(e,t) = 1$ for the largest breakpoint $t \in \{1,\ldots,T_e\}$ with $C_e^t \leq \overline{y}(e)$. If the current solution already satisfies $\overline{x}(e,t) = 1$, then set $x(e,t+1) = 1$.

CONSERVATIVE Set $x(e,t) = 1$ for the smallest breakpoint $t \in \{1,\ldots,T_e\}$ with $x(e,t) < 1$.

Divisible Basic Capacities

Suppose that capacity variables $\overline{x}(e,\tau)$, $e \in E$, $\tau \in T$, as the solution of the current relaxation are given, and let $\overline{y}(e)$, $e \in E$, be the corresponding capacity vector. Again, a supply edge with fractional capacity variable is selected in the first step, and afterwards integer lower bounds are decided according to two strategies. In the following, $\lceil \overline{y}(e) \rceil$ denotes the smallest valid capacity larger than $\overline{y}(e)$ for every supply edge $e \in E$, that is,

$$\lceil \overline{y}(e) \rceil \quad := \quad \min \left\{ \sum_{\tau \in T} C^\tau m(e,\tau) : m(e,\tau) \in \mathbb{Z}_+ \text{ for } \tau \in T \right\} .$$

Furthermore, $\overline{m}(e,\tau)$ denotes for every $e \in E$ and every $\tau \in T$ this number of multiples of capacity C^τ combined into the ceiling capacity $\lceil \overline{y}(e) \rceil$, that is, $\lceil \overline{y}(e) \rceil = \sum_{\tau \in T} C^\tau \overline{m}(e,\tau)$ for every $e \in E$. Finally, let F be again the subset of supply edges with at least one fractional capacity variable, that is,

$$F \quad := \quad \{ e \in E : \exists\ \tau \in T \text{ with } \overline{x}(e,\tau) \notin \mathbb{Z}_+ \} .$$

MaxFrac Choose a supply edge with the largest fractional variable, that is, $(\epsilon, \overline{\tau}) \in T(F)$ such that

$$\overline{x}(\epsilon, \overline{\tau}) \quad = \quad \max_{e \in F} \{ \overline{x}(e,\tau) : \tau \in T,\ \overline{x}(e,\tau) \notin \mathbb{Z}_+ \} .$$

MinCap Choose the fractional variable such that the additional capacity needed to make $\overline{y}(e)$ a valid capacity is minimal, that is, $\epsilon \in E$ such that

$$\lceil \overline{y}(\epsilon) \rceil - \overline{y}(\epsilon) \quad = \quad \min_{e \in F} \{ \lceil \overline{y}(e) \rceil - \overline{y}(e) : e \in F \} .$$

MinCost Choose a supply edge with fractional variable such that the additional cost incurred from increasing to the ceiling capacity is minimal, that is, $\epsilon \in E$ such that

$$\sum_{\tau \in T} K^\tau_\epsilon \left(\overline{m}(\epsilon,\tau) - \overline{x}(\epsilon,\tau) \right) \quad = \quad \min_{e \in F} \left\{ \sum_{\tau \in T} K^\tau_e \left(\overline{m}(e,\tau) - \overline{x}(e,\tau) \right) \right\} .$$

MinRelCost Choose a supply edge with fractional variable such that additional cost incurred from increasing to the ceiling capacity relative to the additional capacity is minimal, that is, $\epsilon \in E$ such that

$$\sum_{\tau \in T} K^\tau_\epsilon (\overline{m}(\epsilon,\tau) - \overline{x}(\epsilon,\tau)) \ / \ (\lceil \overline{y}(\epsilon) \rceil - \overline{y}(\epsilon)) =$$

$$\min_{e \in F} \left\{ \sum_{\tau \in T} K^\tau_e (\overline{m}(e,\tau) - \overline{x}(e,\tau)) \ / \ (\lceil \overline{y}(e) \rceil - \overline{y}(e)) \right\} .$$

Suppose the supply edge $e \in F$ is decided. Then, integer lower bounds for the capacity variables of e must be determined such that the current solution $\overline{x}$ does not satisfy these bounds. This is done according to one of the following two strategies.

GREEDY Set the integer lower bounds for $x(e,\tau)$, $\tau \in T$, according to the multipliers of the ceiling capacity $\lceil \overline{y}(e) \rceil$, that is, $x(e,\tau) \geq \overline{m}(e,\tau)$, for all $\tau \in T$.

CONSERVATIVE Set the integer lower bound for the maximal $\tau \in T$ (w.r.t. its capacity C^τ) with strictly positive multiplier $\overline{m}(e,\tau)$ $\overline{x}(e,\overline{\tau}) > \overline{m}(e,\overline{\tau})$.

Postprocessing

If the capacity vector $\overline{y}$ that corresponds to the final integer capacity vector $\overline{x}$ is *not* feasible, a postprocessing step is necessary in order to find a feasible integer capacity vector. We distinguish between two different versions. One for the survivability models DIVERSIFICATION and RESERVATION, and the other for PATH RESTORATION.

It is known from Theorem 3.2 and Proposition 3.3 that the metric inequalities (3.1) and (3.2) suffice to describe the polyhedra $Y(G,H,\text{DIV})$ and $Y(G,H,\text{RES})$ for the survivability models DIVERSIFICATION and RESERVATION, respectively. Furthermore, the separation algorithm for these classes is exact, that is, whenever there exists a violated metric inequality, the algorithm identifies one. Thus, the heuristic eventually yields an integer *feasible* solution, if this separation algorithm is added to those applied in the cutting plane algorithm.

If PATH RESTORATION is the chosen survivability model, the postprocessing proceeds as before in the sense that the separation algorithms for metric inequalities (2.22) and (3.2) are added to those applied in the cutting plane algorithm. Again, the heuristic might end up with an integer, but infeasible solution $\overline{x}$. In this case, the postprocessing makes also use of separation algorithms for inequalities that are “almost” metric inequalities. As described before, the feasibility test serves as separation algorithm for the different versions of metric inequalities. For the survivability models DIVERSIFICATION and RESERVATION this separation algorithm is exact since the column generation procedure always identifies missing path variables. For PATH RESTORATION, the column generation might fail to identify such variables.

Suppose that the path variables in the linear program of the feasibility test are fixed and no column generation is applied to identify missing path variables. For the capacity vector $\overline{y}$ that corresponds to the current integer capacity variables $\overline{x}$, it does not exist a feasible routing using only paths that correspond to variables in the linear program. (Otherwise, we would have stated that $\overline{y}$ and thus $\overline{x}$ is feasible.) Therefore, the optimal objective function value of this linear program is strictly greater than zero, and the dual variables define an inequality (3.3) that is violated by $\overline{y}$. If no path variable is missing (and this fact could just not be proven), this inequality is also a valid inequality. The postprocessing of a branch&cut path heuristic for PATH RESTORATION ignores that some of these variables might be missing and that the inequality is not necessarily a valid metric inequality. The inequality (3.3) defined by the dual variables is constructed, and then the corresponding strengthened metric inequality (3.6) or

(3.16) is added to the current relaxation. This way, the integral solution $\overline{x}$ is cut off. Iterating these steps, the postprocessing eventually yields an integer feasible solution.

Computational tests

We report in this section on computational experiments with different parameter selections for the branch&cut path heuristics. As performance measures,

- the value of the computed solution, and
- the time to execute the heuristic

are used. For both capacity models, the tests are performed on the problem instances m3, l3, and p3 with the survivability parameters $\delta = 0.5$ and $\delta = 0.75$ for Diversification, and $\rho = 0.5$ and $\rho = 1.0$ for Reservation and Path Restoration. The respective problem names associated with these parameters are m3d50, m3d75, l3d50, l3d75, p3d50, and p3d75 for Diversification, and m3r50, m3r100, l3r50, l3r100, p3r50, and p3r100, for Reservation and Path Restoration.

Every k iterations, the cutting plane algorithm is partially executed. The parameter M, which determines after how many iterations the time-consuming separation algorithm for metric inequalities is executed, is increased to 10. (Recall, this parameter has been set to 5 for the lower bound calculation.) Furthermore, only a restricted number of iterations of the cutting plane algorithm is performed. Other separation algorithms for valid inequalities of $X(G, H, \cdot, \cdot)$ are used with the default parameter settings defined at the end of Section 4.5.4.

The computational results are presented in Tables 4.9 and 4.10 in the following format. Column 1 (name) gives the name of the problem which also encodes the survivability parameters. Columns 2 and 3 provide the selection of run-time parameters. In more detail, column 2 (edge) gives the applied strategy to select the next edge for which some of the fractional capacity variables will be fixed. Column 3 (fix) provides whether the fixing strategy is Greedy or Conservative. For each of the three survivability models are three columns with statistics following. From left to right, these columns give the running time of the starting heuristic (time), the scaled[3] value of the computed solution (val), and the number of iterations (iter).

From Tables 4.9 and 4.10 can be seen that the cost of the starting solutions depends on the applied criterion to select the supply edge, but not on the strategy to fix integer capacity variables of the chosen supply edge. The winning strategies to select the next supply edge are MaxFrac, MaxSumFrac, and MinCap for Discrete Capacities, and MinCap and MinCost for Divisible Basic Capacities. The values of the starting solutions obtained with these criteria are in 2/3 of all test instances at most 5 percent worse than the best starting solution.

Finally, we observe that the running times and the number of iterations needed to compute the starting solution are independent from both the criterion to select the supply edge and the strategy to fix variables.

[3] All lower bounds and solution values are scaled to comply with disclosure agreements.

NAME	EDGE	FIX	DIVERSIFICATION			RESERVATION			PATH RESTORATION		
			VAL	TIME	ITER	VAL	TIME	ITER	VAL	TIME	ITER
m3 d50/r50	MAXFRAC	GREEDY	226	53	58	193	45	39	219	12:50	40
	MAXSUMFRAC	GREEDY	234	1:05	54	**182**	37	40	203	20:22	40
	MINCAP	GREEDY	226	**48**	40	192	22	31	209	13:13	28
	MINCOST	GREEDY	239	58	40	195	**17**	32	223	9:19	36
	MINRELCOST	GREEDY	296	56	38	212	23	32	218	8:47	30
	MININCCOST	GREEDY	256	1:09	55	204	34	41	222	10:03	44
	MINRELCAP	GREEDY	235	59	**34**	196	28	**29**	204	15:12	**27**
	MAXFRAC	CONSERVATIVE	222	1:09	51	194	34	43	215	30:18	44
	MAXSUMFRAC	CONSERVATIVE	**214**	56	50	188	27	43	216	26:01	43
	MINCAP	CONSERVATIVE	232	1:15	53	186	30	43	**200**	8:35	43
	MINCOST	CONSERVATIVE	238	1:15	54	196	21	47	215	15:39	46
	MINRELCOST	CONSERVATIVE	272	1:18	59	204	23	45	241	**6:29**	46
	MININCCOST	CONSERVATIVE	232	1:20	58	190	29	46	223	31:25	45
	MINRELCAP	CONSERVATIVE	238	1:03	54	200	26	41	215	10:57	44
m3 d75/r100	MAXFRAC	GREEDY	216	30	49	253	1:44	42	267	10:26	42
	MAXSUMFRAC	GREEDY	203	31	47	253	1:14	44	**262**	16:55	44
	MINCAP	GREEDY	213	**22**	**34**	269	48	29	280	**8:42**	31
	MINCOST	GREEDY	222	38	43	272	38	35	281	24:00	36
	MINRELCOST	GREEDY	285	35	41	267	47	29	318	44:17	35
	MININCCOST	GREEDY	229	41	53	266	1:00	49	275	21:35	49
	MINRELCAP	GREEDY	207	34	36	264	38	**26**	265	29:24	**24**
	MAXFRAC	CONSERVATIVE	223	38	54	258	47	39	294	1:14:00	49
	MAXSUMFRAC	CONSERVATIVE	**201**	39	44	252	49	43	285	1:47:12	46
	MINCAP	CONSERVATIVE	205	36	51	268	1:00	45	275	32:12	45
	MINCOST	CONSERVATIVE	223	48	55	276	52	52	300	1:00:17	54
	MINRELCOST	CONSERVATIVE	259	53	59	265	48	48	275	12:12	46
	MININCCOST	CONSERVATIVE	231	51	60	260	50	51	282	1:08:12	55
	MINRELCAP	CONSERVATIVE	215	38	49	**247**	**34**	39	267	17:40	39
l3 d50/r50	MAXFRAC	GREEDY	219	1:04	60	203	59	52	215	3:41	52
	MAXSUMFRAC	GREEDY	222	1:06	60	205	1:01	57	218	**3:34**	57
	MINCAP	GREEDY	230	1:05	50	211	**48**	47	**213**	3:51	44
	MINCOST	GREEDY	**214**	**56**	53	222	54	47	239	14:12	50
	MINRELCOST	GREEDY	254	1:03	**42**	222	1:07	**37**	231	15:53	**36**
	MININCCOST	GREEDY	224	1:22	63	214	1:25	57	242	10:05	62
	MINRELCAP	GREEDY	232	1:15	54	214	1:24	45	217	16:32	42
	MAXFRAC	CONSERVATIVE	230	1:27	61	205	1:17	52	220	14:16	52
	MAXSUMFRAC	CONSERVATIVE	232	1:26	64	204	1:27	55	221	10:32	58
	MINCAP	CONSERVATIVE	222	1:26	64	**200**	1:39	51	219	14:36	52
	MINCOST	CONSERVATIVE	234	1:28	63	227	1:39	64	235	17:31	60
	MINRELCOST	CONSERVATIVE	257	1:42	62	221	1:14	47	226	25:56	44
	MININCCOST	CONSERVATIVE	236	1:44	67	218	1:41	59	236	17:56	59
	MINRELCAP	CONSERVATIVE	239	1:42	64	211	1:42	49	223	11:58	49
l3 d75/r100	MAXFRAC	GREEDY	202	44	60	237	4:35	65	251	12:31	65
	MAXSUMFRAC	GREEDY	202	**35**	61	**229**	3:01	64	246	7:05	64
	MINCAP	GREEDY	203	37	57	243	2:56	59	260	**6:07**	56
	MINCOST	GREEDY	204	47	60	245	2:27	62	259	7:18	57
	MINRELCOST	GREEDY	218	40	**43**	256	**1:35**	**46**	284	21:29	**51**
	MININCCOST	GREEDY	207	1:02	67	234	3:16	72	247	27:54	71
	MINRELCAP	GREEDY	215	47	58	251	2:29	53	271	33:44	53
	MAXFRAC	CONSERVATIVE	201	47	62	234	2:26	65	244	44:10	65
	MAXSUMFRAC	CONSERVATIVE	**189**	42	58	233	2:40	64	**243**	15:55	65
	MINCAP	CONSERVATIVE	207	53	65	239	2:38	68	257	16:21	73
	MINCOST	CONSERVATIVE	198	51	62	250	2:44	74	254	14:14	73
	MINRELCOST	CONSERVATIVE	205	47	54	271	2:43	70	283	18:17	69
	MININCCOST	CONSERVATIVE	204	1:01	66	233	2:44	72	248	35:50	72
	MINRELCAP	CONSERVATIVE	200	53	60	243	2:37	65	262	32:29	67
p3 d50/r50	MAXFRAC	GREEDY	**194**	1:28	67	165	2:01	54	181	2:57	54
	MAXSUMFRAC	GREEDY	232	1:56	80	**161**	1:35	53	**177**	1:33	53
	MINCAP	GREEDY	207	1:45	69	166	1:18	46	187	1:20	50
	MINCOST	GREEDY	209	1:52	67	172	1:20	52	186	1:26	52
	MINRELCOST	GREEDY	223	1:26	**50**	192	**41**	40	196	58	**37**
	MININCCOST	GREEDY	213	2:19	80	170	1:32	61	186	1:28	61
	MINRELCAP	GREEDY	200	**1:12**	51	175	51	**39**	191	**56**	39
	MAXFRAC	CONSERVATIVE	206	1:55	71	165	46	51	183	1:13	52
	MAXSUMFRAC	CONSERVATIVE	212	2:21	74	163	1:25	53	**177**	1:21	53
	MINCAP	CONSERVATIVE	220	2:25	79	169	1:22	59	184	1:05	59
	MINCOST	CONSERVATIVE	194	2:23	70	166	1:05	59	181	1:12	59
	MINRELCOST	CONSERVATIVE	214	1:50	67	192	45	54	192	1:35	51
	MININCCOST	CONSERVATIVE	209	2:32	80	170	1:09	61	186	1:27	61
	MINRELCAP	CONSERVATIVE	211	1:55	71	185	1:05	56	201	1:27	56
p3 d75/r100	MAXFRAC	GREEDY	188	57	63	198	4:57	65	209	8:42	65
	MAXSUMFRAC	GREEDY	188	52	64	**198**	3:34	66	208	**4:40**	66
	MINCAP	GREEDY	**184**	50	58	205	2:50	58	222	8:12	60
	MINCOST	GREEDY	186	51	59	211	3:16	58	222	9:15	61
	MINRELCOST	GREEDY	213	**41**	**46**	233	**2:38**	57	289	47:54	59
	MININCCOST	GREEDY	200	1:18	76	212	4:31	79	288	1:31:43	89
	MINRELCAP	GREEDY	203	54	56	204	3:18	**51**	234	28:18	**54**
	MAXFRAC	CONSERVATIVE	207	1:03	67	203	3:43	67	**204**	14:38	64
	MAXSUMFRAC	CONSERVATIVE	184	1:06	63	212	4:06	70	206	25:22	64
	MINCAP	CONSERVATIVE	200	1:10	72	206	3:14	74	220	11:30	77
	MINCOST	CONSERVATIVE	184	1:11	67	210	3:42	76	281	2:06:24	88
	MINRELCOST	CONSERVATIVE	203	1:04	62	245	3:24	78	259	58:15	79
	MININCCOST	CONSERVATIVE	197	1:19	74	221	4:13	82	242	59:31	80
	MINRELCAP	CONSERVATIVE	192	58	64	216	4:06	70	248	50:41	76

Table 4.9: DISCRETE CAPACITIES: Computational results for branch&cut path heuristics

NAME	EDGE	FIX	DIVERSIFICATION			RESERVATION			PATH RESTORATION		
			TIME	VAL	ITER	TIME	VAL	ITER	TIME	VAL	ITER
m3 d50/r50	MAXFRAC	CONSERVATIVE	283	**21**	26	201	31	15	210	**10:14**	**15**
	MINCAP	CONSERVATIVE	**230**	23	**21**	196	**23**	15	235	54:21	27
	MINCOST	CONSERVATIVE	243	28	35	195	52	27	238	1:24:25	33
	MINRELCOST	CONSERVATIVE	307	39	30	209	28	28	213	19:52	25
	MAXFRAC	GREEDY	265	31	24	203	29	**14**	228	36:10	20
	MINCAP	GREEDY	243	25	23	**189**	31	19	230	1:35:29	20
	MINCOST	GREEDY	239	35	31	196	28	26	**209**	29:07	26
	MINRELCOST	GREEDY	252	54	24	207	34	27	285	1:34:45	36
m3 d75/r100	MAXFRAC	CONSERVATIVE	212	17	27	267	58	18	285	**6:34**	21
	MINCAP	CONSERVATIVE	**211**	**11**	**21**	266	27	10	285	10:56	13
	MINCOST	CONSERVATIVE	226	15	27	**262**	**25**	12	284	43:29	19
	MINRELCOST	CONSERVATIVE	243	23	26	263	29	10	**262**	21:28	13
	MAXFRAC	GREEDY	212	28	26	273	42	20	276	12:13	19
	MINCAP	GREEDY	214	19	24	264	35	**9**	275	17:44	20
	MINCOST	GREEDY	218	21	26	265	39	20	270	49:50	**11**
	MINRELCOST	GREEDY	239	28	35	270	42	19	269	44:54	**11**
l3 d50/r50	MAXFRAC	CONSERVATIVE	309	46	42	222	3:39	**31**	226	**6:00**	**31**
	MINCAP	CONSERVATIVE	**231**	**45**	**31**	223	**2:51**	36	230	10:50	35
	MINCOST	CONSERVATIVE	257	1:00	43	226	5:00	38	243	8:53	39
	MINRELCOST	CONSERVATIVE	272	1:40	46	252	4:35	45	266	14:08	43
	MAXFRAC	GREEDY	286	1:13	39	**220**	3:13	33	231	14:31	34
	MINCAP	GREEDY	259	1:18	36	232	4:35	37	227	16:05	33
	MINCOST	GREEDY	246	1:12	36	225	3:04	38	**225**	18:06	36
	MINRELCOST	GREEDY	276	2:01	45	247	3:08	43	234	14:11	38
l3 d75/r100	MAXFRAC	CONSERVATIVE	279	28	36	256	3:11	34	264	**25:01**	34
	MINCAP	CONSERVATIVE	242	**21**	34	248	3:47	29	264	1:33:25	**27**
	MINCOST	CONSERVATIVE	**223**	25	**31**	251	4:15	32	264	1:29:35	31
	MINRELCOST	CONSERVATIVE	256	34	40	260	2:52	33	276	2:15:12	35
	MAXFRAC	GREEDY	255	36	34	263	3:02	33	262	1:07:45	30
	MINCAP	GREEDY	230	27	**31**	**245**	2:33	**28**	255	2:35:37	29
	MINCOST	GREEDY	238	31	34	248	**2:23**	31	**254**	1:57:12	29
	MINRELCOST	GREEDY	264	49	41	257	3:02	32	266	58:52	35
p3 d50/r50	MAXFRAC	CONSERVATIVE	224	45	**32**	166	1:28	29	182	2:07	29
	MINCAP	CONSERVATIVE	234	48	40	**157**	**56**	**25**	178	1:33	**25**
	MINCOST	CONSERVATIVE	205	**40**	38	178	1:38	35	184	1:47	32
	MINRELCOST	CONSERVATIVE	205	46	37	173	1:26	32	200	1:32	36
	MAXFRAC	GREEDY	250	49	36	164	1:22	27	183	1:49	30
	MINCAP	GREEDY	200	44	34	162	1:00	27	182	1:29	30
	MINCOST	GREEDY	**194**	57	36	183	1:33	34	188	1:37	34
	MINRELCOST	GREEDY	201	51	35	192	1:50	40	199	1:49	37
p3 d75/r100	MAXFRAC	CONSERVATIVE	190	**25**	**37**	204	2:43	34	226	3:40	35
	MINCAP	CONSERVATIVE	**178**	27	**37**	**189**	2:04	**30**	**204**	8:44	**26**
	MINCOST	CONSERVATIVE	202	31	48	190	1:32	33	213	7:37	32
	MINRELCOST	CONSERVATIVE	203	38	46	206	1:58	43	227	24:48	39
	MAXFRAC	GREEDY	201	35	**37**	212	1:59	33	245	18:24	33
	MINCAP	GREEDY	180	27	38	190	1:34	**30**	212	30:57	29
	MINCOST	GREEDY	191	31	43	200	2:04	37	233	23:24	32
	MINRELCOST	GREEDY	203	39	46	205	**1:19**	37	237	18:10	40

Table 4.10: DIVISIBLE BASIC CAPACITIES: Computational results for branch&cut path heuristics

4.6.2 Improvement heuristics

We implemented one class of improvement heuristics, the so-called *decrease heuristics*. Given a feasible integer solution, these heuristics produce locally optimal solution which are minimal with respect to the canonical partial order $\leq_E$ on $|E|$-dimensional vectors. This partial order is defined by

$$y^1 \leq_E y^2 \quad :\Longleftrightarrow \quad y^1(e) \leq y^2(e) \ , \qquad e \in E \ ,$$

and every $y^1, y^2 \in \mathbb{R}_+^E$. In Algorithm 8, a general description of a decrease heuristic is given.

Algorithm 8 DECREASEHEURISTIC

Require: feasible solution $\overline{C}(e), e \in E$
  identify the set R of reducible edges
  while $R \neq \emptyset$ **do**
    select $e \in R$
    select temporary valid capacity C for e with $C^0(e) \leq C < \overline{C}(e)$
    test feasibility of temporary capacity vector
    if feasible **then**
      update $\overline{C}$
    end if
    update R
  end while
  postprocessing

The input of a decrease heuristic consists of integer capacity variables $\overline{x}$ such that the corresponding capacity vector $\overline{y}$ (calculated according to (2.4) or (2.7)) is *feasible*. Let $\overline{C} \in \mathbb{R}_+^E$ be the capacity vector defined by $\overline{C}(e) := \overline{y}(e)$ for all $e \in E$. A preprocessing step identifies the set $R \subseteq E$ of *reducible edges*, where a supply edge is defined to be reducible if and only if it has not been proven, yet, that it is impossible to reduce its capacity. (Initially, $R = \{e \in E : \overline{C}(e) > C^0(e)\}$.) In the main loop, the heuristic attempts to decrease the capacity of reducible edges until R is empty. According to some criterion, a supply edge $e \in R$ is selected in each iteration. For this supply edge, the capacity is temporarily set to some capacity C with $C^0(e) \leq C < \overline{C}(e)$, and to this new capacity vector the algorithm to test feasibility is applied. If it is feasible, this capacity is set, that is, $\overline{C}(e) := C$. Furthermore, e is removed from R either if $C = C^0(e)$ or if the capacity vector has not been feasible. Eventually, R is empty, since the heuristic never adds an edge to R, and in every iteration either the capacity for one supply edge in R decreases or one supply edge is removed from R.

The algorithm to test feasibility of a capacity vector for the survivability model PATH RESTORATION deserves additional remarks. As described in Section 4.4, this algorithm can be time-consuming since very large-scale linear programs must be solved. In consequence, the column generation procedure is not used during a decrease heuristic in order to reduce the computation time. That is, during such a heuristic the set of path variables in the linear program remains unchanged, implying that the routing in a feasible solution can only use the paths associated with those path variables which have either been initially added to the linear program or which have already been generated with the column generation algorithm.

In the remainder of this section on decrease heuristics, we describe the selection criteria for the reducible supply edges and the different strategies to choose the temporary capacity. We describe these criteria separately for the two capacity models, and present afterwards results of computational experiments.

Discrete Capacities

Suppose that a feasible capacity vector $\overline{C}(e), e \in E$, is the given, and let $0 \leq t_e \leq T_e$ be the breakpoint with $C_e^{t_e} = \overline{C}(e)$ for every supply edge $e \in E$. Furthermore, let $\overline{x}(e,t)$, $e \in E$, $t = 1, \ldots, T_e$, be the (fractional) solution of the linear programming relaxation. The criteria to choose the supply edge are the following:

MinFrac Choose a reducible supply edge with smallest fractional capacity variable, that is, $(\epsilon, \tau) \in T(F)$ with $\epsilon \in R$ such that

$$\overline{x}(\epsilon, \tau) \quad = \quad \min\{\overline{x}(e,t) : e \in R,\ 1 \leq t \leq t_e\} \ .$$

MinSumFrac Choose a reducible supply edge with smallest sum of fractional capacity variables, that is, $\epsilon \in R$ such that

$$\sum_{t=1,\ldots,T_\epsilon : \overline{x}(\epsilon,t)<1} \overline{x}(\epsilon,t) \quad = \quad \min\{ \sum_{t=1,\ldots,T_e : \overline{x}(e,t)<1} \overline{x}(e,t) : e \in R\} \ .$$

MaxCost Choose a reducible supply edge whose capacity reduction potentially incurs the largest cost reduction, that is, $\epsilon \in R$ such that

$$K_\epsilon^{t_\epsilon} \quad = \quad \max\{K_e^{t_e} : e \in R\} \ .$$

MaxRelCost Choose a reducible supply edge whose capacity reduction potentially incurs the largest cost reduction relative to its capacity reduction, that is, $\epsilon \in R$ such that

$$K_\epsilon^{t_\epsilon} \ / \ C_\epsilon^{t_\epsilon} \quad = \quad \max\{K_\epsilon^{t_e} \ / \ C_e^{t_e} : e \in R\} \ .$$

MaxIncCost Choose a reducible supply edge whose capacity reduction to the next smaller breakpoint capacity potentially incurs the largest (incremental) cost reduction, that is, $\epsilon \in R$ such that

$$k_\epsilon^{t_\epsilon} \quad = \quad \max\{k_e^{t_e} : e \in R\} \ .$$

MaxRelIncCost Choose a reducible supply edge whose capacity reduction to the next smaller breakpoint capacity potentially incurs the largest (incremental) cost reduction relative to its capacity reduction, that is, $\epsilon \in R$ such that

$$k_\epsilon^{t_\epsilon} \ / \ c_\epsilon^{t_\epsilon} \quad = \quad \max\{k_e^{t_e} \ / \ c_e^{t_e} : e \in R\} \ .$$

MAXCAP Choose a reducible supply edge with maximal potential capacity reduction, that is, $\epsilon \in R$ such that

$$C_\epsilon^{t_\epsilon} - C_\epsilon^0 \quad = \quad \max\{C_e^{t_e} - C_e^0 : e \in R\} \; .$$

Suppose, the supply edge $e \in R$ is chosen. Then, one of the following two strategies is applied to choose the breakpoint which defines the temporary capacity on e.

GREEDY Set in increasing order $C(e) := C_e^t$, $t = 0, \ldots, t_e - 1$, until the capacity vector with temporary capacity $C(e)$ on edge e is feasible. If none of the capacity vectors is feasible, then remove e from R.

CONSERVATIVE Set $C(e) := C_e^{t_e - 1}$. If the capacity vector with this temporary capacity on edge e is not feasible, remove e from R.

DIVISIBLE BASIC CAPACITIES

Similar to the definitions on page 146, let $\overline{m}(e,\tau)$ be the number of multiples of technology $\tau \in T$ needed to combine the capacity $\overline{C}(e)$ for supply edge $e \in E$, that is, $\overline{C}(e) = \sum_{\tau \in T} C^\tau \overline{m}(e,\tau)$. Associated with this capacity, $\overline{K}(e)$ denotes the cost of capacity $\overline{C}(e)$, that is, $\overline{K}(e) = \sum_{\tau \in T} K_e^\tau \overline{m}(e,\tau)$. Furthermore, $\lfloor \overline{C}(e) \rfloor$ denotes for every $e \in E$ with $\overline{C}(e) > C_e^0$ the largest capacity smaller than $\overline{C}(e)$, and $\lfloor \overline{K}(e) \rfloor$ denotes the costs associated with capacity $\lfloor \overline{C}(e) \rfloor$. Let $\overline{x}(e,\tau)$, $e \in E$, $\tau \in T$, be the (fractional) solution of the linear programming relaxation. The criteria to select a supply edge are:

MINFRAC Choose a reducible supply edge with smallest fractional capacity variable, that is, $(\epsilon, \overline{\tau}) \in T(R)$ such that

$$\overline{x}(\epsilon, \overline{\tau}) \quad = \quad \min\{\overline{x}(e,\tau) : e \in R, \; \overline{x}(e,\tau) \neq \mathbb{Z}_+\} \; .$$

MINSUMFRAC Choose a reducible supply edge with smallest sum of fractional capacity variables, that is, $\epsilon \in R$ such that

$$\sum_{\tau \in T : \overline{x}(\epsilon,\tau) \neq \mathbb{Z}_+} \overline{x}(\epsilon,\tau) \quad = \quad \min \left\{ \sum_{\tau \in T : \overline{x}(e,\tau) \neq \mathbb{Z}_+} \overline{x}(e,\tau) : e \in R \right\} \; .$$

MAXCOST Choose a reducible supply edge whose capacity reduction potentially incurs the largest cost reduction, that is, $\epsilon \in R$ such that

$$\overline{K}(\epsilon) \quad = \quad \max\left\{\overline{K}(e) : e \in R\right\} \; .$$

MAXRELCOST Choose a reducible supply edge whose capacity reduction potentially incurs the largest cost reduction relative to its capacity reduction, that is, $\epsilon \in R$ such that

$$\overline{K}(\epsilon) \ / \ \overline{C}(\epsilon) \quad = \quad \max\left\{\overline{K}(e) \ / \ \overline{C}(e) : e \in R\right\} \ .$$

MAXINCCOST Choose a reducible supply edge whose capacity reduction to the next smaller breakpoint capacity potentially incurs the largest (incremental) cost reduction, that is, $\epsilon \in R$ such that

$$\overline{K}(\epsilon) - \lfloor \overline{K}(\epsilon) \rfloor \quad = \quad \max\{\overline{K}(e) - \lfloor \overline{K}(e) \rfloor : e \in R\} \ .$$

MAXRELINCCOST Choose a reducible supply edge whose capacity reduction to the next smaller breakpoint capacity potentially incurs the largest (incremental) cost reduction relative to its capacity reduction, that is, $\epsilon \in R$ such that

$$\begin{aligned}(\overline{K}(\epsilon) - \lfloor \overline{K}(\epsilon) \rfloor) \ / \ (\overline{C}(\epsilon) - \lfloor \overline{C}(\epsilon) \rfloor) = \\ \max\{(\overline{K}(e) - \lfloor \overline{K}(e) \rfloor) \ / \ (\overline{C}(e) - \lfloor \overline{C}(e) \rfloor) : e \in R\} \ .\end{aligned}$$

MAXCAP Choose a reducible supply edge with maximal potential capacity reduction, that is, $\epsilon \in R$ such that

$$\overline{C}(\epsilon) - C^0_\epsilon \quad = \quad \max\{\overline{C}(e) - C^0_e : e \in R\} \ .$$

Suppose, the supply edge $e \in R$ is chosen. Then, one of the following two strategies is applied to choose the temporary capacity on e.

GREEDY Increase $C(e)$ temporarily until the capacity vector is feasible. Afterwards, remove e from R.

CONSERVATIVE Set temporarily $C(e) := \lfloor \overline{C}(e) \rfloor$. If the capacity is feasible then set $\overline{C}(e) := \lfloor \overline{C}(e) \rfloor$, else remove e from R.

4.6.3 Computational tests

In this section, we report on computational experiments with different parameter selections for the improvement heuristics. As performance measures,

- the value of the computed solution, and
- the time to execute the improvement heuristic

are used. The tests are performed for the problem instances m3, l3, and p3 with the survivability parameters $\delta = 0.5$ and $\delta = 0.75$ for DIVERSIFICATION, and $\rho = 0.5$ and $\rho = 1.0$ for RESERVATION and PATH RESTORATION. The respective problem names associated with these parameters are m3d50, m3d75, l3d50, l3d75, p3d50, and p3d75 for DIVERSIFICATION, and m3r50, m3r100, l3r50, l3r100, p3r50, and p3r100 for RESERVATION and PATH RESTORATION. The performance of the improvement heuristics is evaluated for the best starting solution computed by one of the branch&cut path heuristics.

The computational results are presented in Tables 4.11 and 4.12 in the following format. Column 1 (NAME) gives the name of the problem which also encodes the survivability parameters. Columns 2 and 3 provide the selection of run-time parameters. In more detail, column 2 (EDGE) gives the applied strategy to select the next edge for which some of the fractional capacity variables are fixed. Column 3 (FIX) provides whether the applied fixing strategy is GREEDY or CONSERVATIVE. For each of the three survivability models are four columns with statistics following. From left to right, these columns give the scaled[4] values of starting solution (START), computed solution (FINAL), the running time of the starting heuristic (TIME), and the number of iterations (ITER).

The main observation is the following: Given the best starting solution obtained with a branch&cut path heuristic, only minor improvements are possible with the decrease heuristics. The largest improvement is 10 percent and can be observed for l3r50 for the combination DISCRETE CAPACITIES and RESERVATION. Often, the improvement is less than 1 percent. In other words, the starting solutions obtained with a branch&cut path heuristic are almost locally optimal with respect to neighbors obtained by changing the capacity on a single supply edge only.

All performance measures, the final solution value, the number of iterations, and the running times, are independent from both the criterion to select the next supply edge and the strategy how a smaller capacity is selected for the chosen supply edge. The running times for DIVERSIFICATION and RESERVATION instances are small, ranging between a few seconds and 10 minutes. In most case, the time is about 1 minute. For all PATH RESTORATION instances, however, no improvement is possible within the time limit of 60 minutes.

[4]All lower bounds and solution values are scaled to comply with disclosure agreements.

NAME	EDGE	CAP	DIVERSIFICATION FINAL	START	TIME	ITER	RESERVATION FINAL	START	TIME	ITER	PATH RESTORATION FINAL	START	TIME	ITER
m3 d50/r50	MINFRAC	GREEDY	**214**	214	1:02	**33**	**181**	182	38	**30**	**200**	200	1:01:14	23
	MINSUMFRAC	GREEDY	**214**	214	**6**	**33**	**181**	182	32	**30**	**200**	200	1:01:33	**21**
	MAXCOST	GREEDY	**214**	214	57	**33**	**181**	182	29	**30**	**200**	200	1:02:28	**21**
	MAXRELCOST	GREEDY	**214**	214	56	**33**	**181**	182	28	**30**	**200**	200	1:01:56	22
	MAXINCCOST	GREEDY	**214**	214	55	**33**	**181**	182	29	**30**	**200**	200	1:10:42	25
	MAXRELINCCOST	GREEDY	**214**	214	1:00	**33**	**181**	182	29	**30**	**200**	200	1:02:28	28
	MAXCAP	GREEDY	**214**	214	7	**33**	**181**	182	31	**30**	**200**	200	1:00:06	25
	MINFRAC	CONSERVATIVE	**214**	214	39	34	**181**	182	28	32	**200**	200	1:04:07	30
	MINSUMFRAC	CONSERVATIVE	**214**	214	7	34	**181**	182	29	32	**200**	200	1:02:16	42
	MAXCOST	CONSERVATIVE	**214**	214	51	34	**181**	182	29	32	**200**	200	1:01:17	22
	MAXRELCOST	CONSERVATIVE	**214**	214	55	34	**181**	182	30	32	**200**	200	1:02:23	32
	MAXINCCOST	CONSERVATIVE	**214**	214	50	34	**181**	182	**28**	32	**200**	200	1:01:02	27
	MAXRELINCCOST	CONSERVATIVE	**214**	214	53	34	**181**	182	29	32	**200**	200	**1:00:05**	33
	MAXCAP	CONSERVATIVE	**214**	214	7	34	**181**	182	28	32	**200**	200	1:02:09	46
m3 d75/r100	MINFRAC	GREEDY	**201**	201	**37**	**30**	**247**	247	**20**	**29**	**258**	258	1:26:48	5
	MINSUMFRAC	GREEDY	**201**	201	0	0	**247**	247	0	0	**258**	258	1:13:01	**3**
	MAXCOST	GREEDY	**201**	201	0	0	**247**	247	0	0	**258**	258	1:12:12	4
	MAXRELCOST	GREEDY	**201**	201	0	0	**247**	247	0	0	**258**	258	1:19:19	6
	MAXINCCOST	GREEDY	**201**	201	0	0	**247**	247	0	0	**258**	258	1:01:03	5
	MAXRELINCCOST	GREEDY	**201**	201	0	0	**247**	247	0	0	**258**	258	1:26:35	5
	MAXCAP	GREEDY	**201**	201	0	0	**247**	247	0	0	**258**	258	1:08:31	4
	MINFRAC	CONSERVATIVE	**201**	201	0	0	**247**	247	0	0	**258**	258	**1:00:02**	7
	MINSUMFRAC	CONSERVATIVE	**201**	201	0	0	**247**	247	0	0	**258**	258	1:07:08	4
	MAXCOST	CONSERVATIVE	**201**	201	0	0	**247**	247	0	0	**258**	258	1:26:06	5
	MAXRELCOST	CONSERVATIVE	**201**	201	0	0	**247**	247	0	0	**258**	258	1:16:03	5
	MAXINCCOST	CONSERVATIVE	**201**	201	0	0	**247**	247	0	0	**258**	258	1:06:33	5
	MAXRELINCCOST	CONSERVATIVE	**201**	201	0	0	**247**	247	0	0	**258**	258	1:04:55	4
	MAXCAP	CONSERVATIVE	**201**	201	0	0	**247**	247	0	0	**258**	258	1:08:51	**3**
l3 d50/r50	MINFRAC	GREEDY	210	214	3:42	**77**	**190**	200	10:46	**74**	**213**	213	1:01:19	55
	MINSUMFRAC	GREEDY	210	214	3:13	**77**	**190**	200	7:32	**74**	**213**	213	1:02:10	54
	MAXCOST	GREEDY	**209**	214	3:54	**77**	190	200	9:42	**74**	**213**	213	1:04:20	**11**
	MAXRELCOST	GREEDY	**209**	214	2:51	**77**	**190**	200	**5:46**	**74**	**213**	213	1:00:49	14
	MAXINCCOST	GREEDY	**209**	214	**2:25**	**77**	190	200	6:06	**74**	**213**	213	1:05:06	14
	MAXRELINCCOST	GREEDY	**209**	214	2:52	**77**	**190**	200	6:04	**74**	**213**	213	1:03:57	13
	MAXCAP	GREEDY	211	214	2:58	**77**	191	200	8:36	**74**	**213**	213	1:01:23	14
	MINFRAC	CONSERVATIVE	210	214	2:28	79	**190**	200	7:32	75	**213**	213	1:05:47	25
	MINSUMFRAC	CONSERVATIVE	210	214	2:42	79	**190**	200	7:39	75	**213**	213	1:02:16	18
	MAXCOST	CONSERVATIVE	**209**	214	3:00	79	190	200	9:32	76	**213**	213	**1:00:46**	**11**
	MAXRELCOST	CONSERVATIVE	**209**	214	3:08	79	**190**	200	5:47	75	**213**	213	1:05:19	14
	MAXINCCOST	CONSERVATIVE	**209**	214	2:44	79	190	200	7:07	76	**213**	213	1:03:13	13
	MAXRELINCCOST	CONSERVATIVE	**209**	214	3:10	79	**190**	200	6:01	75	**213**	213	1:02:26	13
	MAXCAP	CONSERVATIVE	211	214	2:43	79	191	200	7:58	76	**213**	213	1:03:55	14
l3 d75/r100	MINFRAC	GREEDY	**183**	189	1:26	**72**	**229**	229	**1:24**	**72**	**243**	243	**1:00:10**	18
	MINSUMFRAC	GREEDY	**183**	189	1:04	**72**	**229**	229	0	0	**243**	243	1:01:17	10
	MAXCOST	GREEDY	**183**	189	1:18	**72**	**229**	229	0	0	**243**	243	1:23:39	8
	MAXRELCOST	GREEDY	**183**	189	1:23	**72**	**229**	229	0	0	**243**	243	1:02:08	14
	MAXINCCOST	GREEDY	**183**	189	1:20	**72**	**229**	229	0	0	**243**	243	1:25:03	8
	MAXRELINCCOST	GREEDY	**183**	189	1:20	**72**	**229**	229	0	0	**243**	243	1:12:15	10
	MAXCAP	GREEDY	184	189	1:21	**72**	**229**	229	0	0	**243**	243	1:03:06	15
	MINFRAC	CONSERVATIVE	**183**	189	**59**	73	**229**	229	0	0	**243**	243	1:04:19	11
	MINSUMFRAC	CONSERVATIVE	**183**	189	59	73	**229**	229	0	0	**243**	243	1:02:59	8
	MAXCOST	CONSERVATIVE	**183**	189	1:19	73	**229**	229	0	0	**243**	243	1:02:44	**6**
	MAXRELCOST	CONSERVATIVE	**183**	189	1:17	73	**229**	229	0	0	**243**	243	1:01:12	**6**
	MAXINCCOST	CONSERVATIVE	**183**	189	1:18	73	**229**	229	0	0	**243**	243	1:05:13	11
	MAXRELINCCOST	CONSERVATIVE	**183**	189	1:19	73	**229**	229	0	0	**243**	243	1:30:10	7
	MAXCAP	CONSERVATIVE	184	189	1:24	**72**	**229**	229	0	0	**243**	243	1:22:24	10
p3 d50/r50	MINFRAC	GREEDY	**192**	194	1:36	**53**	156	161	2:51	**50**	**177**	177	**4:02**	**53**
	MINSUMFRAC	GREEDY	**192**	194	1:31	**53**	156	161	2:55	**50**	**177**	177	0	0
	MAXCOST	GREEDY	**192**	194	1:41	**53**	**155**	161	3:04	**50**	**177**	177	0	0
	MAXRELCOST	GREEDY	**192**	194	1:37	**53**	**155**	161	2:42	**50**	**177**	177	0	0
	MAXINCCOST	GREEDY	**192**	194	1:36	**53**	156	161	2:04	**50**	**177**	177	0	0
	MAXRELINCCOST	GREEDY	**192**	194	1:34	**53**	**155**	161	2:23	**50**	**177**	177	0	0
	MAXCAP	GREEDY	**192**	194	1:39	**53**	156	161	2:06	**50**	**177**	177	0	0
	MINFRAC	CONSERVATIVE	**192**	194	**1:17**	54	156	161	2:18	**50**	**177**	177	0	0
	MINSUMFRAC	CONSERVATIVE	**192**	194	1:32	54	156	161	2:58	**50**	**177**	177	0	0
	MAXCOST	CONSERVATIVE	**192**	194	1:43	54	**155**	161	3:01	51	**177**	177	0	0
	MAXRELCOST	CONSERVATIVE	**192**	194	1:36	54	**155**	161	2:42	51	**177**	177	0	0
	MAXINCCOST	CONSERVATIVE	**192**	194	1:41	54	156	161	2:05	**50**	**177**	177	0	0
	MAXRELINCCOST	CONSERVATIVE	**192**	194	1:35	54	**155**	161	2:20	51	**177**	177	0	0
	MAXCAP	CONSERVATIVE	**192**	194	1:30	54	156	161	**2:01**	**50**	**177**	177	0	0
p3 d75/r100	MINFRAC	GREEDY	179	184	51	**61**	**196**	198	5:21	**48**	**204**	204	1:07:41	24
	MINSUMFRAC	GREEDY	**178**	184	52	**61**	**196**	198	4:18	**48**	**204**	204	**1:00:46**	17
	MAXCOST	GREEDY	**178**	184	51	**61**	**196**	198	**4:09**	**48**	**204**	204	1:02:27	16
	MAXRELCOST	GREEDY	179	184	48	**61**	**196**	198	4:52	**48**	**204**	204	1:01:12	20
	MAXINCCOST	GREEDY	**178**	184	53	**61**	**196**	198	5:57	**48**	**204**	204	1:09:13	15
	MAXRELINCCOST	GREEDY	179	184	51	**61**	**196**	198	4:36	**48**	**204**	204	1:01:02	25
	MAXCAP	GREEDY	**178**	184	47	**61**	**196**	198	5:53	**48**	**204**	204	1:02:29	**10**
	MINFRAC	CONSERVATIVE	179	184	**29**	**61**	**196**	198	5:28	49	**204**	204	1:01:41	11
	MINSUMFRAC	CONSERVATIVE	**178**	184	49	62	**196**	198	5:01	49	**204**	204	1:06:02	**10**
	MAXCOST	CONSERVATIVE	**178**	184	51	62	**196**	198	4:41	49	**204**	204	1:05:39	**10**
	MAXRELCOST	CONSERVATIVE	179	184	49	**61**	**196**	198	5:01	49	**204**	204	1:04:31	11
	MAXINCCOST	CONSERVATIVE	**178**	184	53	62	**196**	198	5:29	49	**204**	204	1:02:23	11
	MAXRELINCCOST	CONSERVATIVE	179	184	51	**61**	**196**	198	5:01	49	**204**	204	1:02:28	14
	MAXCAP	CONSERVATIVE	**178**	184	47	62	**196**	198	4:23	49	**204**	204	1:04:23	11

Table 4.11: DISCRETE CAPACITIES: Computational results for improvement heuristics

name	edge	cap	Diversification final	start	time	iter	Reservation final	start	time	iter	Path Restoration final	start	time	iter
m3 d50/r50	MinFrac	Greedy	**228**	230	56	**31**	**182**	189	27	**28**	**209**	209	**1:00:01**	8
	MinSumFrac	Greedy	**228**	230	33	**31**	**182**	189	25	**28**	**209**	209	1:09:21	**4**
	MaxCost	Greedy	**228**	230	44	**31**	**182**	189	26	**28**	**209**	209	1:05:49	11
	MaxRelCost	Greedy	**228**	230	47	**31**	**182**	189	26	**28**	**209**	209	1:03:56	13
	MaxIncCost	Greedy	**228**	230	49	**31**	**182**	189	27	**28**	**209**	209	1:09:29	11
	MaxRelIncCost	Greedy	**228**	230	46	**31**	**182**	189	26	**28**	**209**	209	1:08:41	8
	MaxCap	Greedy	**228**	230	8	**31**	**182**	189	25	**28**	**209**	209	1:08:53	6
	MinFrac	Conservative	**228**	230	34	36	**182**	189	25	30	**209**	209	1:01:56	17
	MinSumFrac	Conservative	**228**	230	33	36	**182**	189	24	30	**209**	209	1:07:28	19
	MaxCost	Conservative	**228**	230	48	36	**182**	189	26	30	**209**	209	1:03:00	16
	MaxRelCost	Conservative	**228**	230	46	36	**182**	189	26	30	**209**	209	1:02:47	13
	MaxIncCost	Conservative	**228**	230	48	36	**182**	189	28	30	**209**	209	1:00:43	6
	MaxRelIncCost	Conservative	**228**	230	46	36	**182**	189	27	30	**209**	209	1:00:05	11
	MaxCap	Conservative	**228**	230	**6**	36	**182**	189	**24**	30	**209**	209	1:30:36	14
m3 d75/r100	MinFrac	Greedy	202	211	32	**32**	**262**	262	**17**	**31**	**262**	262	1:31:56	7
	MinSumFrac	Greedy	**202**	211	33	**32**	**262**	262	**0**	0	**262**	262	1:03:15	11
	MaxCost	Greedy	202	211	27	**32**	**262**	262	**0**	0	**262**	262	1:01:24	8
	MaxRelCost	Greedy	**202**	211	30	**32**	**262**	262	**0**	0	**262**	262	1:03:59	14
	MaxIncCost	Greedy	**202**	211	29	**32**	**262**	262	**0**	0	**262**	262	1:03:27	8
	MaxRelIncCost	Greedy	**202**	211	29	**32**	**262**	262	**0**	0	**262**	262	1:22:57	14
	MaxCap	Greedy	202	211	26	**32**	**262**	262	**0**	0	**262**	262	1:07:39	16
	MinFrac	Conservative	202	211	24	36	**262**	262	**0**	0	**262**	262	1:00:16	22
	MinSumFrac	Conservative	**202**	211	25	36	**262**	262	**0**	0	**262**	262	1:03:58	17
	MaxCost	Conservative	202	211	27	36	**262**	262	**0**	0	**262**	262	1:04:37	**3**
	MaxRelCost	Conservative	**202**	211	30	36	**262**	262	**0**	0	**262**	262	**1:00:14**	9
	MaxIncCost	Conservative	**202**	211	28	36	**262**	262	**0**	0	**262**	262	1:00:22	8
	MaxRelIncCost	Conservative	**202**	211	28	36	**262**	262	**0**	0	**262**	262	1:19:05	13
	MaxCap	Conservative	202	211	**24**	36	**262**	262	**0**	0	**262**	262	1:01:49	17
l3 d50/r50	MinFrac	Greedy	**231**	231	3:18	59	**220**	220	**54**	**56**	**225**	225	**1:00:06**	17
	MinSumFrac	Greedy	**231**	231	0	0	**220**	220	0	0	**225**	225	1:08:37	6
	MaxCost	Greedy	**231**	231	0	0	**220**	220	0	0	**225**	225	1:02:00	7
	MaxRelCost	Greedy	**231**	231	0	0	**220**	220	0	0	**225**	225	1:06:27	9
	MaxIncCost	Greedy	**231**	231	0	0	**220**	220	0	0	**225**	225	1:05:07	8
	MaxRelIncCost	Greedy	**231**	231	0	0	**220**	220	0	0	**225**	225	1:05:24	10
	MaxCap	Greedy	**231**	231	0	0	**220**	220	0	0	**225**	225	1:10:07	10
	MinFrac	Conservative	**231**	231	0	0	**220**	220	0	0	**225**	225	1:04:38	12
	MinSumFrac	Conservative	**231**	231	0	0	**220**	220	0	0	**225**	225	1:06:39	12
	MaxCost	Conservative	**231**	231	0	0	**220**	220	0	0	**225**	225	1:07:08	7
	MaxRelCost	Conservative	**231**	231	0	0	**220**	220	0	0	**225**	225	1:03:38	8
	MaxIncCost	Conservative	**231**	231	0	0	**220**	220	0	0	**225**	225	1:02:28	**5**
	MaxRelIncCost	Conservative	**231**	231	0	0	**220**	220	0	0	**225**	225	1:03:45	8
	MaxCap	Conservative	**231**	231	0	0	**220**	220	0	0	**225**	225	1:06:57	8
l3 d75/r100	MinFrac	Greedy	207	223	1:48	**59**	**245**	245	**51**	**63**	**254**	254	1:11:41	2
	MinSumFrac	Greedy	207	223	1:10	**59**	**245**	245	0	0	**254**	254	1:07:51	3
	MaxCost	Greedy	**205**	223	1:43	**59**	**245**	245	0	0	**254**	254	1:12:21	3
	MaxRelCost	Greedy	207	223	1:29	**59**	**245**	245	0	0	**254**	254	1:08:33	3
	MaxIncCost	Greedy	207	223	1:33	**59**	**245**	245	0	0	**254**	254	1:28:36	4
	MaxRelIncCost	Greedy	207	223	1:28	**59**	**245**	245	0	0	**254**	254	1:45:55	5
	MaxCap	Greedy	207	223	11	**59**	**245**	245	0	0	**254**	254	**1:00:01**	**1**
	MinFrac	Conservative	207	223	1:07	66	**245**	245	0	0	**254**	254	1:32:18	3
	MinSumFrac	Conservative	207	223	1:04	66	**245**	245	0	0	**254**	254	1:19:05	4
	MaxCost	Conservative	**205**	223	1:30	66	**245**	245	0	0	**254**	254	1:20:48	3
	MaxRelCost	Conservative	207	223	1:25	64	**245**	245	0	0	**254**	254	1:49:30	3
	MaxIncCost	Conservative	207	223	1:26	64	**245**	245	0	0	**254**	254	1:40:37	4
	MaxRelIncCost	Conservative	207	223	1:28	64	**245**	245	0	0	**254**	254	1:09:21	3
	MaxCap	Conservative	207	223	**8**	66	**245**	245	0	0	**254**	254	1:00:06	2
p3 d50/r50	MinFrac	Greedy	**194**	194	1:25	47	**157**	157	17	37	**178**	178	8:03	41
	MinSumFrac	Greedy	**194**	194	0	0	**157**	157	0	0	**178**	178	0	0
	MaxCost	Greedy	**194**	194	0	0	**157**	157	0	0	**178**	178	0	0
	MaxRelCost	Greedy	**194**	194	0	0	**157**	157	0	0	**178**	178	0	0
	MaxIncCost	Greedy	**194**	194	0	0	**157**	157	0	0	**178**	178	0	0
	MaxRelIncCost	Greedy	**194**	194	0	0	**157**	157	0	0	**178**	178	0	0
	MaxCap	Greedy	**194**	194	0	0	**157**	157	0	0	**178**	178	0	0
	MinFrac	Conservative	**194**	194	0	0	**157**	157	0	0	**178**	178	0	0
	MinSumFrac	Conservative	**194**	194	0	0	**157**	157	0	0	**178**	178	0	0
	MaxCost	Conservative	**194**	194	0	0	**157**	157	0	0	**178**	178	0	0
	MaxRelCost	Conservative	**194**	194	0	0	**157**	157	0	0	**178**	178	0	0
	MaxIncCost	Conservative	**194**	194	0	0	**157**	157	0	0	**178**	178	0	0
	MaxRelIncCost	Conservative	**194**	194	0	0	**157**	157	0	0	**178**	178	0	0
	MaxCap	Conservative	**194**	194	0	0	**157**	157	0	0	**178**	178	0	0
p3 d75/r100	MinFrac	Greedy	**178**	178	33	40	**189**	189	**51**	**43**	**204**	204	1:02:21	24
	MinSumFrac	Greedy	**178**	178	0	0	**189**	189	0	0	**204**	204	1:00:50	25
	MaxCost	Greedy	**178**	178	0	0	**189**	189	0	0	**204**	204	**1:00:20**	27
	MaxRelCost	Greedy	**178**	178	0	0	**189**	189	0	0	**204**	204	1:07:01	17
	MaxIncCost	Greedy	**178**	178	0	0	**189**	189	0	0	**204**	204	1:01:00	9
	MaxRelIncCost	Greedy	**178**	178	0	0	**189**	189	0	0	**204**	204	1:03:12	9
	MaxCap	Greedy	**178**	178	0	0	**189**	189	0	0	**204**	204	1:04:12	11
	MinFrac	Conservative	**178**	178	0	0	**189**	189	0	0	**204**	204	1:00:35	9
	MinSumFrac	Conservative	**178**	178	0	0	**189**	189	0	0	**204**	204	1:05:02	9
	MaxCost	Conservative	**178**	178	0	0	**189**	189	0	0	**204**	204	1:02:09	9
	MaxRelCost	Conservative	**178**	178	0	0	**189**	189	0	0	**204**	204	1:08:17	8
	MaxIncCost	Conservative	**178**	178	0	0	**189**	189	0	0	**204**	204	1:04:36	**7**
	MaxRelIncCost	Conservative	**178**	178	0	0	**189**	189	0	0	**204**	204	1:07:31	12
	MaxCap	Conservative	**178**	178	0	0	**189**	189	0	0	**204**	204	1:03:34	16

Table 4.12: Divisible Basic Capacities: Computational results for improvement heuristics

4.7 Computational results

In the previous sections of this chapter, we described various algorithmic parts and evaluated the influence of several parameters. After each computational test, we fixed a default parameter setting which seems to be a good compromise between computation time, and quality of the computed solutions and lower bounds. With this default parameter setting, we now evaluate lower bounds, solution values, and computation times of the different algorithmic parts for all test problem instances described in the introduction to this chapter (see page 100).

For both capacity models, this final test series is performed on all nine problem instances with survivability parameters $\delta = 0.5$ and $\delta = 0.75$ for DIVERSIFICATION, and $\rho = 0.5$ and $\rho = 1.0$ for RESERVATION and PATH RESTORATION. Altogether these are 54 problem instances for each capacity model. Table 4.13 shows results for the two capacity models DISCRETE CAPACITIES and DIVISIBLE BASIC CAPACITIES in combination with the three survivability models DIVERSIFICATION, RESERVATION and PATH RESTORATION. The table has the following format. The first column (NAME) gives the name of the problem instance which also encodes the survivability parameters. For both capacity models are seven columns with statistics following. From left to right, these columns give the lower bound (LOWER), the value of the best solution (VAL), the gap in percent computed as (VAL $-$ LOWER)/LOWER $\cdot$ 100 (GAP), and the running times to compute the lower bound (L-TIME), to solve all feasibility problems (F-TIME), and the average times to execute a starting heuristic (S-TIME) and an improvement heuristic (I-TIME).

There are two main observations. First of all, it is possible to compute good solutions for such large-scale mixed-integer programming problems within a few minutes for the "easier" survivability models DIVERSIFICATION and RESERVATION, and within a few hours for the "difficult" survivability model PATH RESTORATION. Second, a lot of additional research is necessary to accomplish the final target of proven optimal solutions.

From a practical point of view, the computation time to provide a good solution is important. With our approach this implies that the average times to compute a lower bound, a starting solution, and an improved solution must be added. The sum of these times is very small for the problem instances with DIVERSIFICATION. The smallest and largest times can be observed for m1d75 and l2d50, respectively. These are 33 seconds and 18 minutes for DISCRETE CAPACITIES, and 11 seconds and 13 minutes for DIVISIBLE BASIC CAPACITIES. The times for RESERVATION are only a little bit larger. This is surprising, if one recalls that the number of variables for the problem instances with RESERVATION is two orders of magnitude larger than those for DIVERSIFICATION. (The largest numbers were approximately 20 trillion and 200 billion, respectively; see Table 4.1). The times for PATH RESTORATION are considerably larger. The smallest and largest times can be observed for p3r100 and l2r100, respectively. These are 16 minutes and 12 hours for DISCRETE CAPACITIES, and 5 minutes and 10 hours for DIVISIBLE BASIC CAPACITIES. The important point to make is the following. These times are larger than those for the other survivability models, but they are small enough such that problem instances with PATH RESTORATION can be used within the network design process in practice.

The gaps are large. On the average, the smallest gaps can be observed for RESERVATION (between 10 and 47 percent) and the largest gaps for DIVERSIFICATION (between 11 and 79 percent). This is surprising, since we expected that the new inequalities for the polyhedra associated with the survivability model Diversification have a better impact on the quality of the lower bound. Very promising is that gaps below 20 percent are possible for Path Restoration. Recall, for the polyhedra associated with this survivability model, there are no inequalities other than those inherited from the respective polyhedra for Reservation. Furthermore, the linear programs to test feasibility for Path Restoration are extremely large, and therefore, it is out of scope to run our improvement heuristics. Thus, despite all compromises, these relatively small gaps for a very large-scale problem can be computed. It remains the question whether the lower bound or the best heuristic solution is the reason for the gap. There is no serious answer to this question. However, tests on smaller problem instances (with weeks of computation time and a lot of interactive manipulation) indicate that the lower bound is responsible. Hence, further research on the polyhedral structure of survivable capacitated network design problems is necessary.

It is interesting to compare the best solutions for varying capacity model. For the chosen problem instances, feasible solutions for Discrete Capacities are feasible for Divisible Basic Capacities. Therefore, one would expect that the best solutions for Divisible Basic Capacities are the better ones, and that the lower bounds for Discrete Capacities are the better ones. To some extent, the results are the other way around. In 2/3 of the considered problem instances, the best solution for Discrete Capacities is better, and in most cases, the lower bound for Divisible Basic Capacities is better. Our conclusion is that it pays to consider a model with a restricted number of available capacities if this is easier to handle. For Divisible Basic Capacities there are much more capacity choices, and it seems to be more difficult to guide the heuristics in the right direction. On the other hand, the better lower bounds with the capacity model Divisible Basic Capacities indicate that the (mixed-integer rounding) inequalities for Divisible Basic Capacities are practically more successful than the (knapsack-cover) inequalities for Discrete Capacities.

Even more interesting is a comparison of the best solutions for varying survivability model. Figure 4.7 shows the average over the best solutions presented.

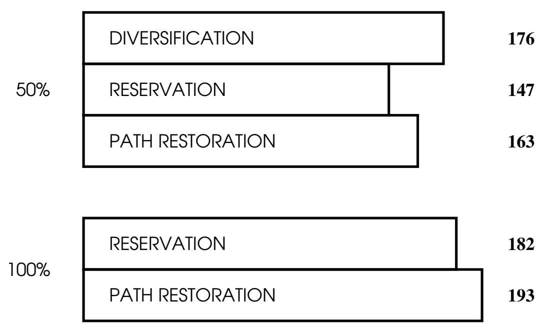


Figure 4.7: Comparison of the survivability models.

For a minimum survivability of 50 percent ($\rho = 0.5, \delta = 0.5$), the average value of a best solution for Reservation is 147. The average for Path Restoration is about 11 percent larger and for Diversification about 20 percent larger. The result for the maximum survivability of 100 percent ($\rho = 1.0$), which can only be accomplished with Reservation and Path Restoration, are very promising. On average, the best solutions for Path Restoration are less than 5 percent larger than the respective best solution for Reservation. That is, we accomplished to compute solutions for the practical relevant survivability model Path Restoration, which are only 5 percent more expensive than solutions computed for Reservation.

	NAME	DISCRETE CAPACITIES							DIVISIBLE BASIC CAPACITIES						
		LOWER	VAL	GAP	L-TIME	F-TIME	S-TIME	I-TIME	LOWER	VAL	GAP	L-TIME	F-TIME	S-TIME	I-TIME
DIVERSIFICATION	m1d50	81.30	116.69	43.53	14	1:55	18	7	87.69	109.12	24.43	8	23	9	1
	m1d75	74.22	109.62	47.69	14	2:15	10	9	81.65	104.60	28.11	5	10	5	1
	l1d50	140.62	184.28	31.05	30	30:23	1:33	2:08	151.51	203.66	34.42	20	48:13	1:17	3:24
	l1d75	120.83	170.81	41.36	27	14:09	57	1:00	148.88	204.81	37.57	8	24:30	38	1:44
	p1d50	78.38	123.90	58.08	34	6:37	41	27	89.82	131.05	45.90	12	3:05	12	12
	p1d75	74.32	113.05	52.14	38	3:56	37	16	81.96	124.73	52.19	9	3:10	10	13
	m2d50	99.88	146.05	46.23	16	6:18	25	26	111.06	141.50	27.41	8	25	14	1
	m2d75	94.44	131.16	38.88	16	1:47	14	7	97.97	135.41	38.22	3	1:06	6	4
	l2d50	117.11	204.43	74.56	1:28	2:15:30	7:03	9:28	125.22	197.88	58.03	27	2:00:19	3:48	8:31
	l2d75	100.00	179.98	79.98	38	1:05:50	3:38	4:38	116.39	174.44	49.88	23	43:40	1:38	3:05
	p2d50	105.70	166.33	57.36	50	45:19	2:54	3:11	111.22	168.07	51.11	14	25:34	1:24	1:48
	p2d75	97.30	152.47	56.70	31	16:35	1:28	1:09	109.25	154.04	41.00	11	14:09	38	59
	m3d50	146.03	214.48	46.87	24	9:59	1:05	40	157.73	228.10	44.61	15	9:26	32	39
	m3d75	127.39	201.81	58.41	21	49	38	2	139.56	202.26	44.92	9	6:47	20	28
	l3d50	161.89	209.39	29.34	29	42:22	1:20	2:59	187.64	231.02	23.12	9	3:43	1:15	14
	l3d75	134.57	183.06	36.03	15	18:03	48	1:16	184.79	205.18	11.03	10	17:47	31	1:15
	p3d50	122.60	192.21	56.78	1:09	22:39	1:57	1:35	130.66	194.50	48.86	16	1:40	47	6
	p3d75	106.45	178.24	67.44	25	11:43	1:01	49	120.83	178.26	47.53	10	46	32	2
RESERVATION	m1r50	72.75	93.42	28.41	10	3:28	21	7	72.79	96.71	32.86	6	2:29	17	8
	m1r100	102.66	126.25	22.98	22	1:48	21	1	112.85	125.22	10.97	17	50	19	1
	l1r50	131.00	175.11	33.67	37	8:49	2:09	4	129.43	180.40	39.38	23	5:28	3:08	1
	l1r100	165.92	193.05	16.35	2:02	9:45	3:53	4	166.81	202.31	21.28	2:10	2:01:48	2:58	8:20
	p1r50	73.31	101.31	38.19	46	1:08:55	1:46	4:09	78.19	99.48	27.23	14	2:45	1:11	1
	p1r100	96.57	119.52	23.77	2:20	41:57	1:18	2:40	107.05	120.51	12.57	1:39	3:50	1:11	1
	m2r50	91.91	115.77	25.96	9	1:57	23	1	88.47	125.55	41.91	4	3:10	15	11
	m2r100	127.52	158.71	24.46	39	7:24	20	27	131.14	163.53	24.70	24	38	22	1
	l2r50	101.26	139.91	38.17	1:34	4:28:59	7:41	17:18	97.55	144.16	47.78	54	1:38:19	5:10	6:12
	l2r100	144.88	180.10	24.31	5:49	7:25:26	11:38	29:34	149.53	185.96	24.36	6:32	14:05	8:53	15
	p2r50	97.39	143.45	47.29	47	20:25	5:29	9	94.09	139.18	47.92	16	8:48	3:22	4
	p2r100	128.31	164.93	28.54	4:02	1:38:31	7:33	5:50	133.33	174.80	31.10	2:37	6:06	5:02	4
	m3r50	137.60	181.77	32.10	18	9:05	32	30	139.74	182.11	30.32	25	7:50	39	25
	m3r100	199.12	247.48	24.29	1:06	4:59	1:03	1	197.95	262.46	32.59	52	2:12	44	1
	l3r50	147.61	190.78	29.25	28	1:53:00	1:33	7:35	154.70	220.37	42.45	37	5:49	4:15	4
	l3r100	198.62	229.57	15.58	1:56	8:14	3:01	6	201.72	245.05	21.48	1:48	6:59	3:36	3
	p3r50	109.05	155.86	42.92	29	41:28	1:24	2:32	100.32	157.24	56.73	6	4:09	1:38	1
	p3r100	149.65	196.28	31.16	2:20	1:20:56	4:00	5:01	146.53	189.37	29.23	1:35	3:28	2:01	2
PATH RESTORATION	m1r50	72.75	105.97	45.66	10	25:17	2:11	14:04	72.79	106.43	46.21	5	32:04	2:52	22:37
	m1r100	102.66	130.77	27.38	21	48:01	3:59	26:51	112.85	137.78	22.08	15	1:04:26	5:59	46:55
	l1r50	131.00	188.34	43.77	25	2:05:46	11:12	1:00:54	129.43	191.48	47.94	22	2:01:20	15:35	1:03:08
	l1r100	165.92	194.70	17.35	1:58	2:47:32	16:59	1:01:24	166.81	205.47	23.17	1:56	4:31:04	1:32:37	1:04:08
	p1r50	73.31	121.69	65.99	43	35:30	3:34	20:33	78.19	128.09	63.82	13	4:24	1:13	2:56
	p1r100	96.57	137.23	42.10	2:16	22:10:39	5:33:35	1:03:24	107.05	127.18	18.80	1:26	30:55	3:37	22:10
	m2r50	91.91	138.58	50.78	8	29:06	3:24	12:43	88.47	139.24	57.38	3	41:37	3:27	28:36
	m2r100	127.52	185.42	45.40	36	2:10:10	18:09	55:18	131.14	166.22	26.75	21	2:03:46	25:42	1:00:20
	l2r50	101.26	147.97	46.13	1:43	13:51:34	2:01:33	1:01:03	97.55	159.22	63.22	55	6:00:51	1:25:17	1:28:03
	l2r100	146.58	197.80	34.94	5:23	98:47:22	9:30:12	1:54:43	149.53	195.96	31.05	6:40	47:63:18	9:02:12	1:00:15
	p2r50	97.39	151.75	55.82	48	4:33:35	1:01:44	1:07:28	94.09	160.76	70.86	15	15:59	3:45	12:26
	p2r100	128.31	181.06	41.11	4:11	7:22:43	1:02:15	1:10:08	133.33	182.31	36.73	2:26	4:53:45	1:00:03	1:32:34
	m3r50	137.60	200.76	45.90	18	2:32:53	15:56	1:01:28	139.74	209.33	49.80	25	3:57:27	1:03:39	1:04:00
	m3r100	199.12	258.98	30.06	1:06	4:15:25	41:32	1:06:15	197.95	262.46	32.59	51	4:12:01	51:25	1:18:52
	l3r50	147.61	213.89	44.90	51	1:37:08	7:59	1:01:22	154.70	225.29	45.63	24	1:54:12	15:55	1:00:24
	l3r100	198.62	242.62	22.15	1:56	2:45:33	12:44	1:00:05	201.72	254.19	26.01	2:20	3:15:41	30:28	1:12:32
	p3r50	109.05	177.93	63.15	32	17:54	2:09	12:47	100.32	178.69	78.12	6	6:59	1:50	5:11
	p3r100	149.65	204.72	36.78	2:42	4:06:46	31:17	1:12:13	146.53	204.74	39.72	1:36	2:17:27	21:18	1:01:39

Table 4.13: Computational results

Conclusions

In this thesis, we investigated the problem of dimensioning survivable capacitated networks. We started with the practical problem, introduced linear mixed-integer programming models, investigated the structure of polyhedra associated with the solution sets of such network design problems, and developed an integrated algorithmic environment, which can now be used to solve such problems.

For all combinations of two capacity and three survivability models, we presented theory and algorithms. We developed new classes of inequalities for several polyhedra related to the considered network design problems, and for these, as well as for previously known classes of inequalities, we developed separation algorithms to identify violated inequalities at runtime. In addition, we extended and adapted previously know algorithms to solve the decision problem whether given capacities suffice to accommodate a routing that satisfies all capacity and survivability requirements. These algorithms are based on column generation. To complete the set of algorithms, we developed several starting and improvement heuristics.

We implemented the software tool DISCNET (DImensioning Survivable Capacitated NETworks) in C++ and JAVA. It provides an integrated environment that includes a graphical user interface and all developed algorithms. With DISCNET it is possible to analyze necessary investments in the network infrastructure by variation of models and parameters. The cost reduction obtained by using DISCNET is difficult to estimate since its first prototype implementations have been applied more than four years ago. At that time, the first applications on real-world problem instances with about 10 transport network nodes revealed a potential cost reduction of about $10 - 20\%$. These values have been obtained for DISCRETE CAPACITIES in combination with DIVERSIFICATION or RESERVATION; the model combinations which have been implemented in the initial prototype. Today, an improved version of DISCNET is permanently used at E-Plus, and since no other tools are used for the same purpose, an estimation of the cost reduction for the current networks is not possible. In this context, it is worth mentioning that current networks are considerably larger. As we have seen, practical problem instances often lead to the linear mixed-integer programs with more than a trillion variables. Hence it seems obvious that even a very smart human being is not able to dimension survivable capacitated networks in a cost-effective way without tools such as DISCNET.

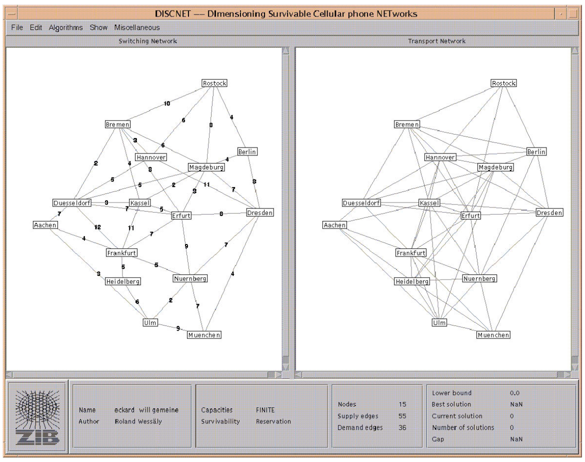
Regarding the survivability model PATH RESTORATION, there are two remarks in order. First, our approach of utilizing RESERVATION as relaxed variant of PATH RESTORATION revealed the opportunity to design low-cost survivable networks for this model. This is particularly the case, if a high level of survivability is desired. Second, we are not aware of another implementation that integrates the optimization of the network for the non-failure case *and* all single network component failures. To our knowledge, our results provide the first integration for PATH RESTORATION which also includes a lower bound calculation.

The overall problem cannot be solved to proven optimality with the current theoretical knowledge (e.g., through branch&cut). Additional research is necessary to get closer to this target. We see two main directions. First, the integer programming problem defined by the linear programming relaxation obtained as result of the cutting plane algorithm seems to be considerably easier to solve and, in consequence, this problem might be approachable by an exact algorithm. Closing the gap by improving the lower bound is the main focus of such an approach, but it might even help to improve the upper bound, since the resulting optimal solution of the relaxed problem can be used to obtain a solution which is feasible with respect to *all* constraints. Second, additional classes of facet-defining inequalities together with fast separation algorithms are needed. In particular, for the survivability model PATH RESTORATION. In this context, it might be worth considering the concept of mixing mixed-integer inequalities presented in (Günlük and Pochet, 1997).

There are at least three major directions for model extensions and variations; one for capacities, one for demands and one for survivability. As we already mentioned in Chapter 2, DISCRETE CAPACITIES is a very flexible capacity model which even suffices to model SDH networks in which wavelength division multiplexers are used on point-to-point links (no routing of wavelengths). However, if additional hardware at the nodes such as digital cross-connects or add-drop-multiplexers should be considered in the optimization, then models for node capacities become necessary. Furthermore, the input for the transport network planning usually contains demand requirements for different types of traffic such as user traffic, signaling traffic, or data traffic. With our model it is necessary to aggregate for each pair of nodes the requirements of the different types. This reduces the problem sizes considerably, but might be insufficient if different traffic types need different protection against network component failures. If this is desired, a model extension using parallel demands between pairs of nodes should be used. Finally, the survivability model LINK RESTORATION (see page 46) might be a possible extension. As described, it is a variant of the model PATH RESTORATION for single edge failures.

Other interesting research directions arise in the design of self-healing rings (ITUT-G.841, 1995). This topic attracted many researchers, but as far as we know, there are no implementations of exact solutions methods of the overall network design problem including the interworking of the self-healing rings (see (ITUT-G.842, 1997)). Typically, only a subproblem on a single ring is reflected in mathematical models for the optimization of self-healing ring networks. Closely related problems arise in the design of optical networks. As soon as optical cross-connects and optical add-drop multiplexers become commercially available, these can be used to route wavelengths generated by wavelength division multiplexers. From a mathematical point of view, very complex, but interesting, coloring problems (see (Mukherjee, 1997)) must be solved in this context.

DISCNET - Graphical User Interface



In the main parts of this thesis, we described mathematical models, polyhedral theory and optimization algorithms to solve the transport network design problem of a mobile-communication provider. The fact that the developed theory and the implemented algorithms to support the planning process are used in a real-world environment has another consequence. There is no doubt regarding the advantage of having the opportunity to run such state-of-the-art algorithms, but in addition, network planners call for graphical user interfaces which support the basic work related to the planning process. To fulfill such non-mathematical requirements, we supplemented DISCNET with a graphical user interface implemented in JAVA. Some of its features are the following:

- Import and export of switching and transport network.
- Graphical representation of switching and transport network.
- Dialogs to modify all parameters related to costs, capacities and demands.
- Execution of all optimization algorithms presented in this thesis.
- Visualization of solutions in terms of capacities and routings.
- Postscript representation of switching and transport network, solutions and routings.

We implemented a graphical representation of the switching and the transport network (in our terminology demand and supply graph). Both networks consist of graphical objects which can be moved, deleted, or selected to either edit the properties or to display related routings.

A typical application of DISCNET starts with the import of switching and transport network as data files. The switching network is part of the output of the planning step preceding the transport network planning (see Chapter 1), and the input transport network consists of potential communication links including possibly preinstalled capacities and a specification of available capacities (according to one of the capacity models presented in Chapter 2). All this input data can easily be manipulated; the network planner can add or delete links and nodes in both networks, and can change parameters of all or individual network components. As an important feature of DISCNET, the input can be analyzed regarding feasibility. As described in Chapter 4, beside some exceptional cases when the survivability model PATH RESTORATION is used, it is possible to decide whether there exists a solution for the particular problem instance. (Recall, we pointed out in Section 4.2 that the decision problem whether a specific number of node-disjoint length-restricted paths exists between a pair of nodes is $\mathcal{NP}$-complete.)

Parameters

The network planner can manipulate all data related to the optimization process. This includes the cost and capacity structure of different providers, capacity and survivability model, and individual parameters for nodes, communication demands and potential transport edges.

Cost parameter

Part of the input of a problem instance are cost values for all available capacities on all potential transport network edges. It would be exhausting and a source of mistakes if the network planner needed to provide all these values. To overcome this problem, cost functions are employed in DISCNET, from which individual cost values can be computed. As single requirement, such a cost functions for a particular capacity is piecewise linear and monotonically increasing with respect to the length.

In DISCNET, the usage of many different providers is permitted. Each one offering different technologies or transmission capacities with different cost functions. To maintain these, the dialog shown in Figure 4.8 is provided. For a particular technology (basic capacity) of a particular provider, the network planner needs to specify the shown values; names of technology and provider, basic capacity, basic costs (which are fixed costs independent of the length), a maximum number of available units between a single pair of nodes and additional length dependent costs. These latter costs are specified for subsequent length intervals. Each one consists of the boundaries of the length interval together with fixed cost and length dependent cost which are incurred every kilometer. Notice that it is possible to maintain cost functions of leased lines as well as microwaves with this dialog.

Figure 4.8: Example of the dialog to maintain cost functions

Node parameters

The parameters of a transport network node are maintained with the dialog shown in Figure 4.9. Every node has a unique name and it is possible to specify its geographical coordinates. These are used to calculate the distance (in kilometer) between pairs of network nodes and thus needed to compute cost values of available capacities for potential transport edges. Additionally, the network planner can specify for each individual node whether its failure should be considered in the optimization. If yes, then every feasible solution contains routings for the respective failure state.

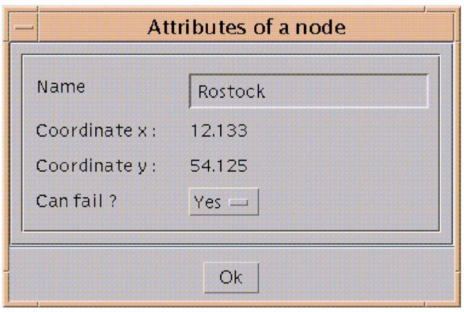


Figure 4.9: Example of the dialog to maintain nodes

Supply edge parameters

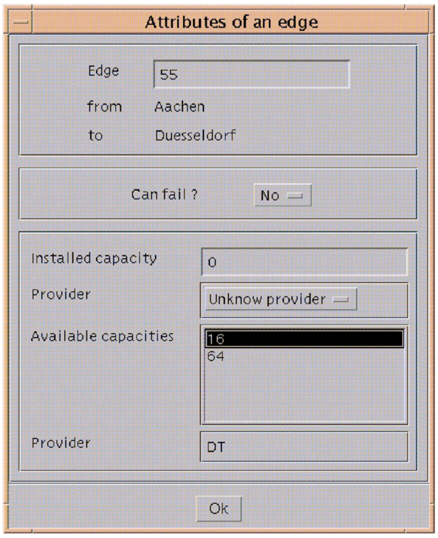


Figure 4.10: Example of the dialog to maintain supply edges

The parameters of a potential edge of the transport network are maintained with the dialog shown in Figure 4.10. Every edge has a unique name and two end-nodes (specified by name). As for nodes, one parameter specifies whether the network planner decided to consider the failure of this particular edge. Again, if yes, then every feasible solution contains routings for the respective failure state. The dialog shows, furthermore, a section to specify the available capacities of such an edge. There might be a positive preinstalled capacity and alternatives to expand the capacity of the edge. The interpretation of the shown capacities depends on the capacity model. For DISCRETE CAPACITIES, these capacities are exactly those considered in the optimization (see Section 2.2.1) and for DIVISIBLE BASIC CAPACITIES, the capacities represent the basic capacities (see Section 2.2.2).

Demand edge parameters

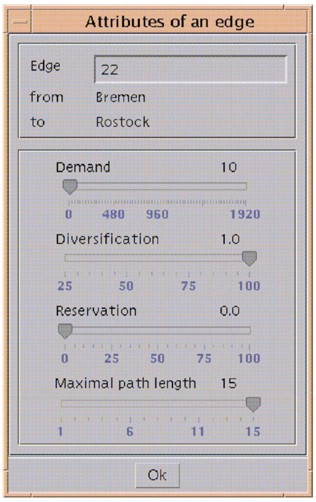


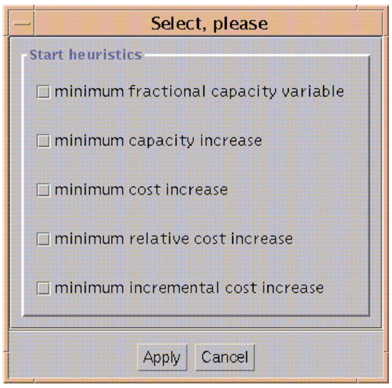
Figure 4.11: Example of the dialog to maintain demand edges

The parameters of a communication demand are maintained with the dialog shown in Figure 4.11. Every edge representing a demand has a unique name, two end-nodes (specified by name) and a value. We shall note, that this value does not have a unit. The network planner is responsible for synchronization of capacity and demand unit. Additionally, the network planner can specify for each individual demand the survivability parameters. These are for the normal operating state a value for DIVERSIFICATION and path-length restriction, and for the failure states a parameter which is interpreted as either RESERVATION or PATH RESTORATION parameter (dependent on the planners choice of the survivability model).

Algorithms & Solutions

The network planner can choose among several algorithms to optimize the cost of the transport network. A typical sequence of algorithms comprises

- the calculation of a lower bound (see Section 4.5),
- the execution of several starting heuristics (see Section 4.6.1), and
- the execution of several improvement heuristics (see Section 4.6.2), with one or several of the previously calculated starting solutions as input.



As described in Section 4.6, the particular choice of starting and improvement heuristics depends on the selected combination of capacity and survivability model. These can be selected with the shown dialog. After the execution of this sequence of algorithms there exists a set of feasible solutions. Each consists of a choice of capacities for all potential edges of the transport network and it is *proven* that these capacities suffice to accommodate a feasible routing in all operating states. Furthermore, a *lower* and an *upper bound* for the value of an optimal solution have been calculated and thus a *quality guarantee* for the computed solutions can be provided.

In addition, feasibility can be tested for every admissible selection of capacities on all supply edges, and if feasible, routings can be computed for such capacities. The latter option is particularly important if the planner imported solutions from other projects or changed the value of some capacities. Recall, beside some exceptional cases when the survivability model PATH RESTORATION is used, the implemented algorithm to test feasibility yields an *exact* decision. That is, if there exist feasible routings for a given choice of admissible capacities then DISCNET will compute such routings. In the other case, if no routings can be computed, then it is *proven* that no feasible routings exist. (We are not aware of any other tool that implemented this feature.) For given routings, DISCNET provides different views such as the table shown in Figure 4.12.

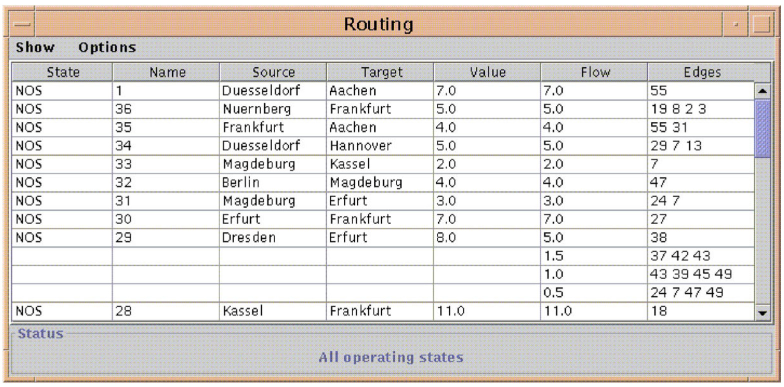


Figure 4.12: Example of a visualization of routings

List of Figures

List of Tables

List of Algorithms

Bibliography

Ahuja, R., Magnanti, T., and Orlin, J. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc.

Alevras, D., Grötschel, M., Jonas, P., Paul, U., and Wessäly, R. (1998a). Survivable mobile phone network architectures: Models and solution methods. *IEEE Communications Magazine*, 36(3):88–93.

Alevras, D., Grötschel, M., and Wessäly, R. (1996). A network dimensioning tool. Technical Report SC 96-49, Konrad-Zuse-Zentrum für Informationstechnik, Berlin.

Alevras, D., Grötschel, M., and Wessäly, R. (1998b). Cost–efficient network synthesis from leased lines. *Annals of Operations Research*, 76:1–20.

Balakrishnan, A., Magnanti, T., Sokol, J., and Wang, Y. (1998). Modeling and solving the single facility line restoration problem. Technical report, MIT, Operations Research Center.

Balas, E. (1975). Facets of the knapsack polytope. *Mathematical Programming*, 8:146–164.

Barahona, F. (1996). Network design using cut inequalities. *SIAM Journal on Optimization*, 6(3):823–837.

Bellman, R. E. and Dreyfus, S. E. (1962). *Applied Dynamic Programming*. Princeton University Press, Princeton, New Jersey.

Benders, J. (1960). *Partitioning in mathematical programming*. PhD thesis, University of Utrecht.

Benders, J. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252.

Bienstock, D. (1999). Experiments with a network design algorithm using ϵ-approximate linear programs. Technical Report 3, Columbia University. CORC Report.

Bienstock, D., Chopra, S., Günlük, O., and Tsai, C.-Y. (1998). Minimum cost capacity installation for multicommodity network flows. *Mathematical Programming*, 81:177–199.

Bienstock, D. and Günlük, O. (1995). Computational experience with a difficult mixed-integer multicommodity flow problem. *Mathematical Programming*, 68:213–237.

Bienstock, D. and Muratore, G. (1999). Strong inequalities for capacitated survivable network design problems. Technical Report 3, Columbia University. CORC Report, Mathematical Programming (to appear).

Bley, A. (1997). Node-disjoint length-restricted paths. Diplomarbeit, TU Berlin, Germany.

Bondy, J. and Murty, U. (1976). *Graph Theory with Applications.* American Elsevier, New York, and Macmillan, London.

Brockmeyer, E., Halstrom, H., and Jensen, A. (1948). *The life and works of A.K. Erlang.* Copenhagen: The Copenhagen Telephone Company.

Chopra, S., Gilboa, I., and Sastry, S. (1998). Source sink flows with capacity installation in batches. *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 85.

Chvátal, V. (1973). Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4:305 – 337.

Chvátal, V. (1983). *Linear Programming.* Freeman Press, New York.

Dahl, G. and Stoer, M. (1998). A cutting plane algorithm for multicommodity survivable network design problems. *INFORMS Journal on Computing*, 10(1):1–11.

Der Spiegel (1995). Stummer Rebell. 6/1995.

Der Tagesspiegel (1994). Graue Mattscheiben und stille Telephone. 24th December.

Die Presse (1999). „Funkstille“ zwischen Wien und Salzburg. 5th March.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

Erlang, A. (1917). Solution of some problems in the theory of probabilities of significance in automatic telephone exchanges. *Elektrotkeknikeren*, 13.

Filho, A. and Tavares, H. (1998). Sizing of meshed networks with line restoration. In *Proceedings of the 8th Telecommunication Network Planning Symposium*, pages 399–404, Sorrento, Italy.

Ford, L. and Fulkerson, D. (1962). *Flows in Network.* Princeton University Press, Princeton, NJ.

Frank, A. (1990). Packing paths, circuits, and cuts – A survey. In Korte, B., Lovász, L., Prömel, H. J., and Schrijver, A., editors, *Paths, Flows, and VLSI-Layout*, volume 9 of *Algorithms and Combinatorics*, chapter 4, pages 47–100. Springer Verlag.

Frank, A. (1995). Connectivity and network flows. In Graham, R., Grötschel, M., and Lovász, L., editors, *Handbook of Combinatorics*, chapter 2, pages 111–179. North-Holland.

Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-completeness*. Freeman, San Francisco.

Gomory, R. E. (1969). An algorithm for integer solutions to linear programming. In Graves, R. L. and Wolfe, P., editors, *Recent Advances in Mathematical Programming*, pages 269 – 302, New York. McGraw-Hill.

Grötschel, M., Monma, C., and Stoer, M. (1992). Facets for polyhedra arising in the design of communication networks with low-connectivity constraints. *SIAM Journal on Optimization*, 2(3):474–504.

Grötschel, M., Monma, C., and Stoer, M. (1995). *Design of survivable networks*, volume Network Models of *Handbooks in Operations Research and Management Science*, chapter 10, pages 617–672. North-Holland.

GSM-1.02 (1993). European digital cellular telecommunications system (Phase 2); General description of a GSM Public Land Mobile Network (PLMN). In *series* GSM. European Telecommunications Standards Institute.

GSM-1.04 (1994). European digital cellular telecommunications system (Phase 2); Abbreviations and acronyms. In *series* GSM. European Telecommunications Standards Institute.

Günlük, O. (1999). A branch-and-cut algorithm for capacitated network design. *Mathematical Programming*, A 86:17–39.

Günlük, O. and Pochet, Y. (1997). Mixing mixed-integer inequalities. Technical report, Cornell University.

Hammer, P. L., Johnson, E. L., and Peled, U. N. (1975). Facets of regular 0-1 polytopes. *Mathematical Programming*, pages 179–206.

Hu, T. (1963). Multi-commodity network flows. *Operations Research*, 11:344–360.

IEEE Spectrum (1989). Keeping the phone lines open. June.

Iri, M. (1971). On an extension of the maximum-flow minimum-cut theorem to multicommodity flows. *Journal of the Operations Research Society of Japan*, 13(3):129–135.

Itai, A., Perl, Y., and Y.Shiloach (1982). The complexity of finding maximum disjoint paths with length constraints. *Networks*, 12:277–286.

ITUT-G.701 (1993). Vocabulary of digital transmission and multiplexing, and pulse code modulation (PCM) terms. In *series G*. ITUT Recommendation.

ITUT-G.702 (1988). Digital hierarchy bit rates. In *series G*. ITUT Recommendation.

ITUT-G.711 (1988). Pulse code modulation (PCM) of voice frequencies. In *series G*. ITUT Recommendation.

ITUT-G.803 (1997). Architecture of transport networks based on the synchronous digital hierarchy (SDH). In *series G*. ITUT Recommendation.

ITUT-G.841 (1995). Types and characteristics of SDH network protection architectures. In *series G.* ITUT Recommendation.

ITUT-G.842 (1997). Interworking of SDH network protection architectures. In *series G.* ITUT Recommendation.

Kakusho, O. and Onaga, K. (1971). On feasibility conditions of multicommodity flows in networks. *Transactions on Circuit Theory*, 18:425–429.

Karmakar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica*, pages 373–395.

Khachyan, L. (1979). A polynomial algorithm in linear programming. *Doklady Akad. Nauk SSSR*, 244:1093–1096. (Russian) (engl. translation: Soviet Mathematics Doklady 20 (1979), 191–194).

Koch, T. and Martin, A. (1998). Solving steiner tree problems in graphs to optimality. *Networks*, 32(3):207–232.

Kuhn, D. (1997). Sources of failure in the public switched telephone network. *IEEE Computer*, 30(4).

Lee, K., Park, K., and Park, S. (1995). Spare channel assignment for DCS mesh-restorable networks. In *Proceedings of the 3rd International Conference on Telecommunication Systems: Modeling and Analysis*, pages 296–307.

Leighton, T., Makedon, F., Plotkin, S., Stein, C., Tardos, E., and Tragoudas, S. (1991). Fast approximation algorithms for multicommodity flow problems. In *Proceedings of the 23th Annual ACM Symposiumon Theory of Computing*, pages 101–111.

Leong, T., Shor, P., and Stein, C. (1993). Implementation of a combinatorial multicommodity flow algorithm. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 12:387–405.

Magnanti, T. and Mirchandani, P. (1993). Shortest paths, single origin-destination network design, and associated polyhedra. *Networks*, 23:103–121.

Magnanti, T., Mirchandani, P., and Vachani, R. (1993). The convex hull of two core capacitated network design problems. *Mathematical Programming*, 60(2):233–250.

Magnanti, T., Mirchandani, P., and Vachani, R. (1995). Modeling and solving the two–facility capacitated network loading problem. *Operations Research*, 43(1):142–157.

Magnanti, T. and Wang, Y. (1997). Polyhedral properties of the network restoration problem – with the convex hull of a special case. Technical report, MIT, Operations Research Center.

Martello, S. and Toth, P. (1990). *Knapsack Problems – Algorithms and Computer Implementations.* Discrete Mathematics and Optimization. John Wiley & Sons.

Minoux, M. (1981). Optimum synthesis of a network with non-simultaneous multicommodity flow requirements. In Hansen, P., editor, *Studies on Graphs and Discrete Programming*, pages 269–277. North–Holland Publishing Company.

Mouly, M. and Pautet, M.-B. (1992). *The* GSM *System for Mobile Communications.* Cell & Sys, France.

Mukherjee, B. (1997). *Optical Communication Networks.* Series on Computer Communications. McGraw-Hill.

Murakami, K. and Kim, H. (1995). Joint optimization of capacity and flow-assignment for self-healing ATM networks. In *Proceedings of ICC'95*, pages 216–220.

Nemhauser, G. and Wolsey, L. (1988). *Integer and Combinatorial Optimization.* John Wiley & Sons.

Nemhauser, G. L. and Wolsey, L. A. (1990). A recursive procedure to generate all cuts for $0-1$ mixed integer programs. *Mathematical Programming*, 46:379 – 390.

Padberg, M. (1995). *Linear Optimization and Extensions.* Springer, Berlin, Heidelberg.

Padberg, M. and Grötschel, M. (1985). *Polyhedral Theory*, chapter 8, pages 251–306. John Wiley & Sons.

Padberg, M. W. (1975). A note on zero-one programming. *Operations Research*, 23(4):833–837.

Pochet, Y. and Wolsey, L. (1995). Integer knapsack and flow covers with divisible coefficients: Polyhedra, optimization and separation. *Discrete Applied Mathematics*, 59(1):57–74.

Poppe, F. and Demeester, P. (1997). The design of SDH mesh-restorable networks: Problem formulation and algorithm. In *Proceedings of the 5th International Conference on Telecommunication Systems, Modeling and Design*, pages 88–97.

Pulleyblank, W. (1983). *Mathematical Programming: State of the Art*, chapter Polyhedral Combinatorics, pages 312–345. Springer, Berlin.

Sakauchi, H., Nishimura, Y., and Hasegawa, S. (1990). A self-healing network with an economical spare-channel assignment. In *Proceedings IEEE Global Telecommunication Conference*, pages 403.1.1–403.1.6.

Schrijver, A. (1986). *Theory of Linear and Integer Programming.* John Wiley & Sons.

Schrijver, A. (1990). Homotopic routing methods. In Korte, B., Lovász, L., Prömel, H. J., and Schrijver, A., editors, *Paths, Flows, and VLSI-Layout*, volume 9 of *Algorithms and Combinatorics*, chapter 12, pages 329–371. Springer Verlag.

Stoer, M. (1992). *Design of Survivable Networks*, volume 1531 of *Lecture Notes in Mathematics.* Springer.

Stoer, M. and Dahl, G. (1994). A polyhedral approach to multicommodity survivable network design. *Numerische Mathematik*, 68:149–167.

Suurballe, J. (1974). Disjoint paths in a network. *Networks*, 4:125–145.

The Wall Street Journal (1988). Fire in fiber-optic 'gateway' sparks flight delays, problems at brokerages. 11th May.

Wang, F., Richards, D., and Winter, P. (1992). *The Steiner Tree Problem*, volume 53 of *Annals of Discrete Mathematics*. North-Holland.

Weyl, H. (1935). Elementare Theorie der konvexen Polyeder. *Comentarii Mathematici Helvetici*, 7:290–306.

Wolsey, L. (1975). Faces of linear inequalities in 0-1 variables. *Mathematical Programming*, 8:165–178.

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

Wu, T.-H. (1992). *Fiber Network Service Survivability*. Telecommunications Library. Artech House.

Xiong, Y. (1998). Optimal design of restorable ATM mesh networks. In *Proceedings of IEEE ATM'98 Workshop*, pages 394–399.

Index

Lebenslauf

Roland Wessäly

geboren am 12. Juni 1967 in Berlin

1973–1977	Brüder-Grimm-Grundschule in Bielefeld-Sennestadt
1977–1986	Gymnasium Hans-Ehrenberg-Schule in Bielefeld-Sennestadt
1986	Einzel- und Mannschaftseuropameister im Bahnengolf
1986–1987	Grundwehrdienst in der Bundeswehr-Sportschule in Warendorf
1988–1994	Studium der Informatik an der Technischen Universität Berlin
1989–1994	Studium der Mathematik an der Technischen Universität Berlin
1991	Vordiplom in Mathematik und Informatik
1994	Diplom in Informatik
seit 1994	Wissenschaftlicher Mitarbeiter am Konrad-Zuse-Zentrum für Informationstechnik Berlin