Contributions towards Survivable Network Design with Uncertain Traffic Requirements

Stephanus Esias Terblanche

B.Sc. (Potchefstroomse Universiteit vir Christelike Hoër Onderwys)
B.Sc. Hons. (Potchefstroomse Universiteit vir Christelike Hoër Onderwys)
M.Sc. (Potchefstroomse Universiteit vir Christelike Hoër Onderwys)

Thesis submitted in the School for Computer, Statistical and Mathematical Sciences at the Potchefstroom Campus of the North-West University in fulfilment of the requirements for the degree Doctor of Philosophy in Computer Science

Supervisors:
Prof. J.M. Hattingh
Dr. R. Wessäly

Potchefstroom
November 2008

Acknowledgements

I would like to thank my supervisors, Prof Giel Hattingh and Dr Roland Wessäly, for their guidance during this study. I especially want to thank Prof Giel Hattingh for providing me with the opportunity to attend several international conferences and for his assistance that enabled me to make several trips to Berlin during the course of this study. I am grateful to Dr. Roland Wessäly for introducing me to the Survivable Network Design problem and for allowing me to use DISCNET as a platform for developing the code necessary for the experimental work in this research. I also wish to express a word of thanks to the other researchers at the Zuse Institute Berlin for their hospitality and willingness to assist me.

I am extremely grateful to my wife, Mariet, for her continued support and unconditional love during my years of study. Her words of encouragement guided me through many tough times when quiting seemed like an attractive alternative.

Finally, I attribute the ability to learn, develop and grow to the grace of God. *Soli Deo Gloria*!

Abstract

The objective in designing a communications network is to find the most cost efficient network design that specifies hardware devices to be installed, the type of transmission links to be installed, and the routing strategy to be followed. The volume of traffic that will be supported by the network is dependent on the capacity of the hardware devices and the transmission links. Different hardware devices and transmission media will have different capacities and costing structures depending on the choice of vendor. The outputs expected from solving the network design problem are, therefore, the proposed topology with a list of hardware devices and transmission technologies that should be installed at the node and link locations respectively, as well as the proposed routing strategy in order to satisfy all routing restrictions and traffic requirements.

In addition to finding the most cost efficient network design, ensuring quality of service, is considered to be another primary objective in planning communication networks. Two issues pertinent to quality of service are robustness and survivability. The contributions of this thesis are, therefore, towards solving the survivable network design problem by taking into account uncertainty in the traffic requirements for general IP networks. The model under consideration makes provision for a detailed representation of hardware devices typically found in SDH transmission networks, and both dynamic and static routing models are presented, with details on appropriate metric inequalities needed for characterising feasible capacities. Details are provided on separation strategies as well as an iterative approach for obtaining solutions that significantly reduces computing times. In order to maintain tractability a problem reduction approach is suggested based on the theory of domination.

Computational results are provided based on data collected from an operational network. From the results it is observed that the suggested approach for solving the survivable network design problem with uncertain traffic requirements successfully reduces computing times.

Opsomming

Die ontwerp van kommunikasienetwerke het ten doel om die mees koste-effektiewe netwerkontwerp te vind wat onder andere spesifiseer watter hardeware geïnstalleer moet word, die tipe transportasiekoppelings wat gebruik moet word, asook watter roeteringstrategie gevolg moet word. Die hoeveelheid verkeer wat deur die netwerk hanteer sal word, is afhanklik van die kapasiteit van die hardeware, asook die tipe transportasiekoppelings wat gebruik word. Verskillende tipes hardeware en transportasiemedia het verskillende kapasiteite, asook verskillende kostestruture afhangende van die keuse van die hardewareverskaffers. Die verwagte resultaat deur die oplos van die netwerkontwerpprobleem is dus 'n voorgestelde topologie met 'n lys van voorgestelde hardeware en transportasietegnologie wat geïnstalleer moet word, asook die voorgestelde roetering wat nodig is om te voldoen aan die dataverkeersvereistes.

As gevolg van die ontwikkeling van kompeterende markte is dit van uiterste belang dat kommunikasienetwerke so koste-effektief as moontlik ontwerp moet word. Voorts is die versekering van kwaliteit in diensverskaffing van primêre belang met spesifieke klem op robuustheid en oorleefbaarheid binne die telekommunikasiekonteks.

In hierdie proefskrif word bydraes gelewer tot die oplos van oorleefbare netwerkontwerpprobleme, deur onsekerheid in die verkeersvereistes in ag te neem. Die model wat beskou word, maak voorsiening vir 'n gedetaileerde voorstelling van hardewarevereistes wat voorkom in tipiese SDH-transportasienetwerke. Beide dinamiese en statiese roeteringsmodelle word beskou met detail oor die metrieke ongelykhede wat gepas is om toelaatbare kapasiteite mee te beskryf. Genoegsame detail word ook gegee rakende die skeidingstrategieë vir die metrieke ongelykhede, sovel as detail aangaande 'n iteratiewe benadering wat die berekeningstyd van oplossings aansienlik verminder. Daar word ook aandag geskenk aan 'n probleemreduksiebenadering wat gebaseer is op die teorie van dominansie.

Resultate is verkry deur gebruik te maak van data wat versamel is vanaf 'n operasionele netwerk. Die resultate toon dat die voorgestelde benaderings in hierdie proefskrif die berekeningstyd van die netwerkontwerpprobleem suksesvol minimeer.

Contents

Mathematical Preliminaries

In the following an overview of well known mathematical concepts and terminology in the fields of linear algebra, polyhedral theory, linear and integer programming, and graph theory are presented. This chapter is not intended as an introduction but rather as a reference to assist with notation.

For an introduction to polyhedral theory and linear programming, the reader is referred to the books by Bazaraa et al. [BJS90] and Padberg [Pad95]. For an introduction to linear integer programming see Wolsey [Wol98] and for a more complete text on integer programming and polyhedral combinatorics the books by Grötschel et al. [GLS88], and Nemhauser and Wolsey [NW98] are recommended. The book by Bondy and Murty [BM98] is a good reference for graph theoretical concepts.

0.1 Linear Algebra Concepts

The set of real, rational and integer numbers are denoted by $\mathbb{R}$, $\mathbb{Q}$, and $\mathbb{Z}$ respectively. The nonnegative parts of these sets are denoted by $\mathbb{R}_+$, $\mathbb{Q}_+$, and $\mathbb{Z}_+$. The symbol $\mathbb{N} = \mathbb{Z}_+ \backslash \{0\}$ denotes positive integer numbers.

Let $\mathbb{K}$ be a generic set for representing any of the sets $\mathbb{R}$, $\mathbb{Q}$, and $\mathbb{Z}$. For arbitrary index sets $I = \{1, 2, \ldots, m\}$ and $J = \{1, 2, \ldots, n\}$, the set $\mathbb{K}^n$ or equivalently $\mathbb{K}^{|J|}$, denotes all the vectors of size n that have components in $\mathbb{K}$, and the set $\mathbb{K}^{m \times n}$ or equivalently $\mathbb{K}^{|I| \times |J|}$, denotes a matrix space of size $m \times n$ that have components in $\mathbb{K}$. Let $j \in J$ be an index for the vector x such that $x = (x_j)_{j \in J}$. In the remainder of this thesis all vectors are treated as column vectors. The transposed of the vector $x \in \mathbb{K}^n$ is denoted by $x^T \in \mathbb{K}^n$.

A vector $x \in \mathbb{R}^n$ can be expressed as a **linear combination** of the vectors $x_1, x_2, \ldots, x_k \in \mathbb{R}^n$ if there exist some $\lambda \in \mathbb{R}^k$ such that $x = \sum_{i=1}^k \lambda_i x_i$. If in addition

$$\left.\begin{array}{ll} \lambda \geq 0 & \\ & \sum_{i=1}^k \lambda_i = 1 \\ \lambda \geq 0 & \sum_{i=1}^k \lambda_i = 1 \end{array}\right\} \text{we call } x \text{ a } \left\{\begin{array}{l} \textbf{conic} \\ \textbf{affine} \\ \textbf{convex} \end{array}\right\} \textbf{combination}$$

of the vectors $x_1, x_2, \ldots, x_k \in \mathbb{R}^n$. If $0 < \lambda_i < 1$ for all $i = 1, 2, \ldots, k$, the above combinations are called **proper**. Furthermore, for a nonempty set $X \subseteq \mathbb{R}^n$

$$\left.\begin{array}{l}\text{lin}(X)\\ \text{cone}(X)\\ \text{aff}(X)\\ \text{conv}(X)\end{array}\right\} \text{ denotes the } \left\{\begin{array}{l}\textbf{linear}\\ \textbf{conic}\\ \textbf{affine}\\ \textbf{convex}\end{array}\right\} \textbf{hull of } X$$

A set $X \subseteq \mathbb{R}^n$ is said to be **linearly dependent** or **affinely dependent** if any of its members is a proper linear or affine combination of the elements in X respectively. Otherwise the set X is considered to be either **linearly independent** or **affinely independent**. The **rank** of a set X, denoted by $\text{rank}(X)$, is equal to the maximum number of linear independent vectors in X. The **affine rank** of a set X, denoted by $\text{arank}(X)$, is equal to the maximum number of affine independent vectors in X. The **dimension** of X is given by $\dim(X) = \text{arank}(X) - 1$. If $\dim(X) = n$, X is said to be **full dimensional**.

0.2 Polyhedral Theory Concepts

The set $H = \{x \in \mathbb{R}^n : a^T x = a_0\}$ with $a_0 \in \mathbb{R}$ denotes a **hyperplane** with gradient $a \in \mathbb{R}^n$ and the set $\{x \in \mathbb{R}^n : a^T x \leq a_0\}$ denotes a **halfspace**. The intersection of a finite set of halfspaces defined by the set $\{x \in \mathbb{R}^n : Ax \leq b\}$, with $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, is called a **polyhedron**. A bounded polyhedron is called a **polytope**.

An inequality of the form $a^T x \leq a_0$ with $a_0 \in \mathbb{R}$ and $a \in \mathbb{R}^n$ is called a **valid inequality** for a polyhedron P, if $P \subseteq \{x \in \mathbb{R}^n : a^T x \leq a_0\}$. The set $F = \{x \in P : a^T x = a_0\} \subseteq P$ is called a **face** induced by the valid inequality $a^T x \leq a_0$. The inequality $a^T x \leq a_0$ is **binding** or **tight** if $F \neq \emptyset$. If in addition $\dim(F) = \dim(P) - 1$, then F is called a **facet** of P and $a^T x \leq a_0$ is said to be **facet defining**. The face F is a **vertex** of P if $\dim(F) = 0$ and is an **edge** of P if $\dim(F) = 1$.

0.3 Linear and Integer Programming Basics

A **Linear Programming Problem** (LP) entails finding a vector $x^* \in P = \{x \in \mathbb{R}^n_+ : Ax \leq b\}$ that optimises the objective function $c^T x$. The vector x^* is called a **feasible solution** if $x^* \in P$ and is called an **optimal solution** if $c^T x^* \geq c^T x$ for all $x \in P$ in the case of a **maximisation problem**, and if $c^T x^* \leq c^T x$ for all $x \in P$ in the case of a **minimisation problem**.

The standard form for representing an LP is:

$$\begin{aligned} &\min\ c^T x \\ &\text{s.t. } Ax \leq b \\ &\quad x \geq 0 \end{aligned} \tag{1}$$

Alternatively, for a short notation we either use $\min\{c^T x : Ax \leq b, x \in \mathbb{R}^n_+\}$ or $\min\{c^T x : x \in P\}$ or $\min_{x \in P}\{c^T x\}$. It should be noted that an LP with a minimisation objective function can be transformed into an LP with a maximization objective function (and vice versa), an LP with unbounded variables can be transformed to an LP with non-negative bounded variables, and an LP with inequality constraints can be transformed into an LP with equality constraints.

An optimal solution of an LP over a polyhedron P will always be at a vertex of P, provided P is bounded. Otherwise the LP may be **unbounded** with $c^T x = \pm\infty$, depending on the choice of the objective function $c^T x$.

Associated with each LP $\min\{c^T x : Ax \leq b, x \in \mathbb{R}^n_+\}$ is the **dual** problem $\max\{w^T b : w^T A \geq c, w \in \mathbb{R}^m_+\}$. The original LP $\min\{c^T x : Ax \leq b, x \in \mathbb{R}^n_+\}$ is referred to as the **primal** problem.

The relationship between the primal and dual problems is described by the following well known theorems:

Theorem 0.1 (Duality Theorem). *With regard to the primal and dual problems, exactly one of the following statements is true:*

1. *Both problems have optimal solutions $x^* \in \mathbb{R}^n_+$ and $w^* \in \mathbb{R}^m_+$ with $c^T x^* = b^T w^*$.*
2. *One problem is unbounded, in which case the other must be infeasible.*
3. *Both problems are infeasible.*

Theorem 0.2 (Complementary Slackness Theorem). *Let $x^* \in \mathbb{R}^n_+$ and $w^* \in \mathbb{R}^m_+$ be any feasible solutions to the primal and dual problems respectively. These solutions are optimal if and only if:*

$$(c_j - a_j^T w^*) x_j^* = 0 \quad j = 1, 2, \ldots, n$$

and

$$w_i^* (a^i x^* - b_i)^T = 0 \quad j = 1, 2, \ldots, n$$

with a_j the j-th column and a^i the i-th row of A respectively.

The objective of solving an **Integer Linear Programming Problem** (IP) is to find an **integer vector** $x^* \in \mathbb{Z}^n \cap P$ with $P = \{x \in \mathbb{R}^n_+ : Ax \leq b\}$, that optimises the objective function $c^T x$. The problem obtained by dropping the integrality restrictions on IP is again an LP, called the LP relaxation of IP.

Let $x^{\mathrm{IP}} \in \mathbb{Z}^n \cap P$ be a feasible solution to an IP with associated objective function value $z^{\mathrm{IP}} = c^T x^{\mathrm{IP}}$, and $x^{\mathrm{LP}} \in P$ be an optimal solution of the LP relaxation with associated objective function value $z^{\mathrm{LP}} = c^T x^{\mathrm{LP}}$. For a minimisation problem the quantities z^{IP} and z^{LP} are referred to as the **upper bound** (UB) and **lower bound** (LB) respectively. For a maximisation problem it is the converse. The quantity (UB – LB)/LB is called the **integrality gap**.

0.4 Graph Theory Concepts

A **graph** $G = (V, E)$ comprises a finite set $V \neq \emptyset$ of **nodes** and a finite set E of **edges**. Associated with each edge is an unordered node pair called its **endnodes**. The endnodes of an edge is said to be **adjacent**. An edge is said to be **incident** to its endnodes, and if a node is an end node to one or more edges, the node is said to have an **incidence set** of edges. A node that is not incident to any edges is **isolated**. If two or more edges have the same endnodes, the edges are said to be **parallel**. A graph without any parallel edges is called a **simple** graph. A simple graph is **complete** if every combination of node pairs is the endnodes of an edge.

A **walk** in a graph G is defined as the finite sequence of nodes and edges (or arcs) $W = v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k$, with $v_0 \in V$ and $v_k \in V$ the **origin** and **terminus** nodes of the walk respectively. The **length** of a walk is the number of edges (or arcs) defining the walk. A graph is **connected** if there exists a walk between every pair of nodes in G. A **path** is defined as a walk in which the nodes $v_0, v_1, v_2, \ldots, v_k$ are distinct, which implies that the edges (or arcs) $e_0, e_1, e_2, \ldots, e_k$ are distinct as well. A walk of nonzero length of which the origin and terminus are identical, is called **closed**. A closed walk with a distinct set of nodes and edges is called a **circuit**.

A **tree** is a connected graph with no circuit. A **spanning tree** of a graph G is a tree containing all the nodes of the graph G.

Chapter 1

Introduction

The internet and other communication networks have become integrated into our daily lives and the continual growth of internet usage has led to considerable efforts in research and development of network design models, and efficient algorithms for solving these models[1]. In addition, the deregulation of former government owned telecommunication operators has leveled the playing field and has forced operators, i.e. the incumbents and new operators, to place more emphasis on cost efficient planning in order to be more competitive. Furthermore, service providers of internet products are obligated to ensure quality of service to their clients and therefore requires from telecommunication operators to have in place a robust and reliable network.

Planning a telecommunications network entails making some decisions based on forecasts and future assumptions. Network design models should, therefore, take cognisance of uncertainty in planning data. It is the focus of this thesis to present contributions in the field of network design with specific reference to planning under uncertainty. The research for this thesis was directed towards finding an acceptable approach for modelling uncertainty and towards developing efficient algorithms that are computationally feasible.

1.1 Designing a Communications Network

The primary purpose of a communication network is the transportation of data between hardware devices at distinct locations. Examples of communicating devices are for instance personal computers sending and receiving email messages, a personal computer that downloads web pages from a web server, or a software application on a computer that extracts data resident on a database server, etc. Between any two communicating devices there is most likely a large collection of other hardware devices that are interconnected in order to facilitate the transportation of the data. The interconnections between hardware devices, also referred to as transmission links, could for instance range from physical media like copper cables or fibre optical cables to radio waves.

[1]The research for this thesis has partly been funded by the Telkom Centre of Excellence.

For most communication networks, specifically the internet, a packet switching approach for transporting data is followed. That is, at the source device data in the form of files are broken up into smaller pieces, called packets, are then transported across the network and reassembled again at the destination device. Network routers are responsible for passing on the individual data packets according to some routing strategy.

The objective in designing a communications network is to find the most cost efficient network design that specifies hardware devices to be installed, the type of transmission links to be installed, and the routing strategy to be followed. The volume of traffic that will be supported by the network is dependent on the capacity of the hardware devices and the transmission links. Different hardware devices and transmission media will have different capacities and costing structures depending on the choice of vendor.

In addition to selecting the appropriate hardware and transmission types, a **network topology** that specifies the location for hardware devices and transmission links, needs to be determined. For purposes of making an abstraction of the problem a topology graph comprising of **nodes** and **edges** is defined. The nodes are abstractions used to represent potential locations where hardware devices can be installed, and edges are abstractions used to represent the potential placement of transmission links.

The typical inputs to the network design problem are therefore:

- The potential topology.
- The permissable hardware devices to be installed at the nodes with capacity and cost specifications.
- The permissable transmission link types to be installed at the edges with capacity and cost specifications.
- Routing policies and restrictions.
- Traffic demands for each communicating node pair represented as a **traffic demand matrix**.

The outputs expected from solving the network design problem are the proposed topology with a list of hardware devices and transmission technologies that should be installed at the node and link locations respectively, as well as the proposed routing that will satisfy all routing restrictions and traffic requirements.

Due to the emergence of competitive markets it has become imperative for telecommunication operators to design communication networks as cost efficient as possible. Ensuring quality of service, however, is considered to be the primary objective in planning communication networks. Two issues pertinent to quality of service are **robustness** and **survivability**.

1.1.1 Robust network design

In any network design problem assumptions are being made about network traffic. For instance, when embarking on greenfield type of planning projects, traffic demand requirements for a couple of years into the future are in many cases estimated from population figures. That is, an assumption is made that some percentage of the population will make use of the planned network infrastructure, and that the eventual clients from this population will consume some capacity on the network for satisfying their bandwidth requirements. Even for projects that entail expansion of current capacities, historical data are often used for predicting future traffic demand requirements by assuming that current bandwidth usage will grow according to some rate.

To further complicate matters, variation in traffic load across a network needs to be taken into account when planning a network. For instance, during the course of a day several periods of peak network traffic may be observed on a network. During the early hours of the day as businesses commence operation, an increase in network traffic could be expected, as well as straight after lunch time when employees face the final stretch of a working day. Apart from overall peak traffic, there is also the possibility that different communicating node pairs could have noncoincident peak traffic. In the evenings a certain amount of traffic might be relocated from some part of the network to another as residential traffic increases and business related traffic decreases. Another example is where cities located in different time zones have different business hours and, consequently, will have noncoincident peak traffic.

Designing a network while following a conservative approach, that is considering the peak traffic estimates over all communicating node pairs as coincident, could result in a very expensive network design. If, on the other hand, average estimates are used for capacity planing, the bandwidth might be insufficient and fall short of customer satisfaction. The ideal solution is to have a network design that is cost efficient but at the same time robust enough to support noncoincident peak hour traffic, by dynamically routing the traffic through parts of the network that are under utilised.

1.1.2 Survivable network design

Apart from the increase in global internet usage, vital systems have become highly dependent on communication infrastructure. Therefore, sufficient capacity that will allow bandwidth intensive applications to perform optimally is alone not enough. Survivability measures need to be in place that will allow vital systems to remain operational even in the event of equipment failures within communication infrastructures. Although technical backup systems may be in place to handle equipment failures, network planners are expected to have in place backup bandwidth that will allow the continued delivery of communication services to vital systems.

1.2 Contribution Summary

The contributions made by the research presented in this thesis are towards network design with the emphasis on robustness and survivability. Since planning robust networks entails making provisions for varying traffic conditions, the network design models considered in this thesis have to address uncertainty in the traffic requirements. The following provides an overview of the main contributions of this study:

1. An existing path based routing model with survivability is extended to accommodate uncertainty in traffic requirements by considering multiple demand matrices. The philosophy followed in this thesis is that variation in network traffic can be represented by a finite set of nonsimultaneous multiple demand matrices. Both dynamic routing and static routing cases are considered.

2. The characterization of capacities in terms of metric inequalities is extended to both dynamic and static routing where multiple demand matrices are considered.

3. A strengthening procedure is proposed for metric inequalities in the case of dynamic routing.

4. Details of a robust separation implementation in the case of static routing are presented.

5. A problem reduction technique is presented as part of the algorithmic approach. The theory of domination among traffic matrices is extended to multiple demand matrices.

6. A new approach for solving the survivable network design problem with multiple demand matrices is presented. Included in this approach is a new heuristic for generating primal solutions, as well as several strategies for selecting a subset of matrices to be included in the initial phase of the approach.

7. Computational results are provided that clearly show the success of employing the proposed algorithmic approaches. The results are based on data obtained from an operational network in Germany.

1.3 Chapter Orientation

An overview of communication networks is provided in Chapter 2, focussing on the technological issues pertinent to network planning and also the typical modelling considerations required to model reality as closely as possible. The primary goal of the chapter is to describe the context in which the work for this thesis must be viewed.

In Chapter 3 the modelling approach to be followed in the remainder of this thesis regarding the treatment of uncertainty within network design, is established. The preferred modelling approach

is motivated by exploring the two primary approaches referred to in the literature as Stochastic Programming and Robust Optimisation.

The proposed mathematical models applied to the survivable network design problem with demand uncertainty are presented in Chapter 4. The complexity of the models considered in the said chapter is a result of both attempting to model the underlying technological aspects of a communication network in as much detail as possible, as well as accommodating the possibility that traffic requirements are uncertain.

In Chapter 5 a theoretical overview is provided of the algorithmic framework considered for solving the survivable network design problem with demand uncertainty. The proposed algorithmic contributions are set within the branch-and-cut framework for solving a mixed integer programming problem.

The computational results obtained by the proposed algorithmic contributions in this thesis are presented in Chapter 6. The results are based on both random data and data based on measurements obtained from an operational network in Germany. Finally, a summary and some concluding remarks are provided in Chapter 7.

Chapter 2

Technical Background

The origin of the Internet can be traced back to the late 1950s when the U.S. Department of Defence was in need of an alternative to the public telephone network that could survive a nuclear attack. Figure 2.1 depicts a typical telephone network and also points out the obvious vulnerability in the structure. Switching and toll offices are responsible for routing calls and by removing any of these the system gets fragmented into many isolated islands. An initial idea proposed by Paul Baran from the RAND Corporation, to rather deploy a meshed based telephone system utilising digital packet-switching technology, was rejected by AT&T as impractical. Several years later in 1967 Paul Baran's idea was finally adopted by the Advanced Research Projects Agency (ARPA). The ARPANET network was developed and went operational in 1969 with four universities acting as communicating nodes within the network. Since only universities that were awarded contracts with ARPA were allowed be part of ARPANET, the U.S. National Science Foundation created a successor to ARPANET that allowed any university to become part of the network. Eventually as other countries developed their own research networks and these networks got connected, people started to view the collection of networks as an internet, later referred to as the Internet.

2.1 The Internet as an Interconnection of Networks

It is unlikely that any two communicating devices could be connected directly. Take for example a researcher sending an email message from Potchefstroom, South Africa, to another researcher in Berlin, Germany. Once this message has left the source computer in Potchefstroom it will most likely be passed on by several other devices within the local access network managed by the University until it reaches a device hooked up to a bigger network that spans for instance the borders of South Africa. The message will eventually leave this national network and access an international spanning network that will eventually deliver it to a national spanning network within Germany. The message will find its way to a device that is hooked up to the University in Berlin and if all goes well will be delivered to the destination computer. Figure 2.2 gives a graphical

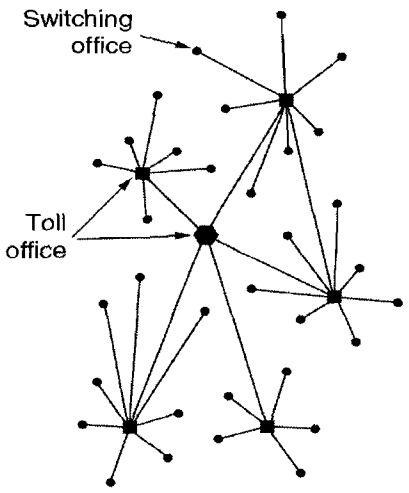

Figure 2.1: Telephone network structure

representation of the scenario and shows the concept of interconnected networks that basically constitutes the Internet.

The devices responsible for passing on the message are called **routers**. Each network in the afore mentioned example typically contains several routers providing alternative routes for a message to traverse. Figure 2.3 illustrates how a cloud representing, for instance, the network spanning South Africa contains several routers (in practice there will most likely be hundreds) that are connected with physical links. It should be noted that due to cost efficiency not all pairs of routers are typically connected to each other. This is then also part of the network design problem to determine the optimal placement of physical links between the routers. The devices and physical links encapsulated within the cloud denote what is called the **core network** or the **backbone network**. The routers within the core network are referred to as the **core routers** and the routers on the outside of the cloud giving network users and other networks access to this core network is referred to as the **edge routers**. The scope of the network design problem in this thesis is limited to the core network.

2.2 Network Protocols

Even though communication networks mostly comprise hardware devices and physical links, it would be impossible to establish communication without the appropriate software. Standardising the functionality of the software is crucial for allowing different types of network devices to communicate effectively. For this reason most network related software are viewed as a stack of **layers**. The number of layers, the functionality and contents of each layer, may differ depending on the

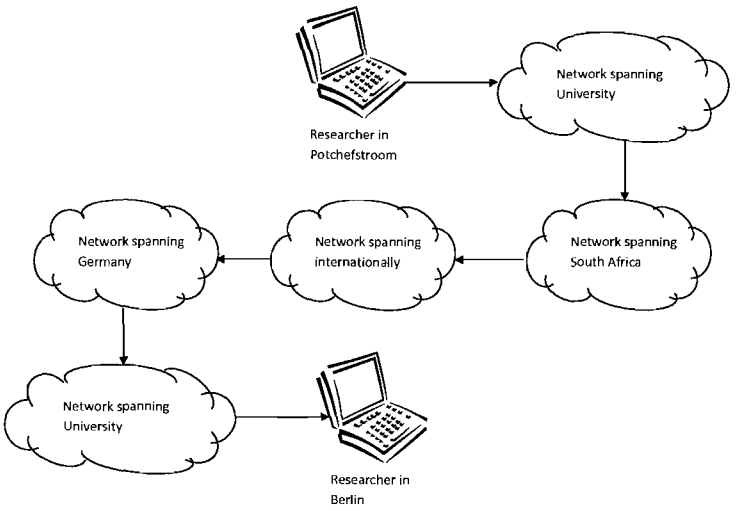

Figure 2.2: Interconnected networks

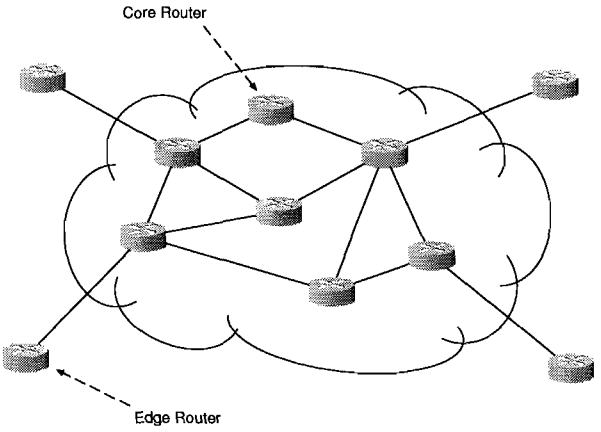

Figure 2.3: Scope for network design - core network

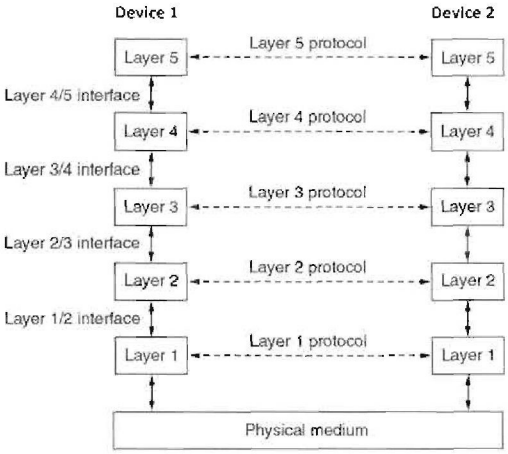

Figure 2.4: Protocol stack (taken from [Tan03])

underlying network technology, as well as the type of services provided by the network. Each layer is responsible for providing a service to the upper layers, without the upper layers needing to know the details of how the services are implemented. The services of a lower layer can be accessed through an **interface** known to the upper layers. In software terms an interface is nothing more than a list of available functions or routines. A specific layer on one device can communicate with a corresponding layer on another device according to some convention or **protocol**. Figure 2.4 depicts a five-layer protocol stack for two communicating devices.

From a logical perspective communication appears to be horizontal between corresponding layers of communicating devices. As an example, assume that the top layer in Figure 2.4 is concerned with the software applications responsible for sending and receiving email messages. The protocol on this layer will, for instance, be responsible for composing a message and identifying the receiver of the message by means of an address. On the receiving end, the same protocol will be responsible for displaying the message and revealing the address of the sender. It therefore appears as if the two email applications are "communicating" when, in fact, the flow of information mostly occurs vertical. To illustrate, consider the same scenario of sending a message between two devices as depicted by Figure 2.5. A message M is composed on layer 5 of Device 1, and is passed down to layer 4 for transmission. For our example let layer 4 be responsible for adding a **header** to the message that might contain additional control information such as size, time stamps etc. The newly packaged message is passed down to layer 3 which might be responsible for breaking up the message into packets. In most networks there is a limit imposed on message sizes within layer 3. Headers are added to the newly created packets $M1$ and $M2$ that will be passed down to layer 2. In layer 2 **data frames** are created by adding some additional headers as well as **trailers** to the packets, which are then passed down to layer 1 for transmission on the physical layer. On

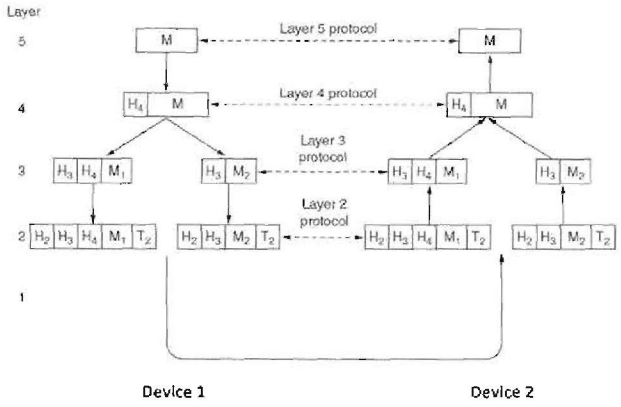

Figure 2.5: Protocol communication (taken from [Tan03])

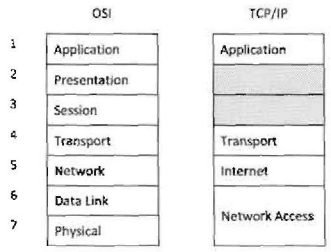

Figure 2.6: OSI vs TCP/IP

the receiving end the messages are passed from the lower layers to the top, removing headers and trailers, and merging packets into complete messages where necessary.

2.3 The TCP/IP Reference Model

The preceding section on protocols provided an abstract view of layered networks. In literature there are two models that serve as references for discussions on network protocols: the **Open System Interconnection** (OSI) reference model and the **TCP/IP** reference model, named after the two primary protocols in the model, Transmission Control Protocol and Internet Protocol. Figure 2.6 provides a comparison between the two models with respect to the protocol layers. In practice the OSI reference model is useful for theoretical discussion although the protocols thereof are not really in use anymore. On the other hand the TCP/IP reference model and protocols are relevant and widely used. An overview of the TCP/IP reference model, specifically the internet protocol, deserves some attention since it defines the context of the network design problem addressed in this thesis.

2.3.1 The application layer

The application layer within the TCP/IP reference model is primarily concerned with providing an interface between the user and the network, and includes protocols needed for most network related applications. For instance, the Simple Mail Transfer Protocol (SMTP) and Post Office Protocol (POP) are used by e-mail applications, while the Hyper Text Transfer Protocol (HTTP) are used by browser applications for fetching and displaying web pages.

2.3.2 The transport layer

One of the first functions to be performed by the transport layer is to split messages into smaller units if needed and to pass the resulting packets to the network layer. The two protocols associated with the transport layer are firstly the **Transmission Control Protocol** (TCP) that is responsible for sending messages as a stream of data that needs to be delivered to the destination device without error. This type of service is also referred to as a **connection-oriented** service. The second protocol, called the **User Datagram Protocol** (UDP), is responsible for providing a **connection-less** service. Applications that require prompt delivery rather than accurate delivery, such as transmitting audio and video, will make use of UDP.

2.3.3 The internet layer

Synonymous with the internet layer is the concept of **packet-switching**. Messages to be transported between devices are split up into smaller pieces called **packets**. For improving network scalability and congestion control the packets need not necessarily be sent along the same route in the core network. Depending on the state of the network and the routing algorithm employed, packets might be **switched** to different routers that will result in the packets following different routes. The packet-switching approach is also frequently referred to as the **store-and-forward** approach since at each router it might happen that a packet needs to be stored momentarily until resources are available that will allow the packet to be forwarded to the next router.

The official packet format used by the internet layer is defined by the **Internet Protocol** (IP). The most important aspects regarding the IP are addressing and routing. Whenever a message is sent from one device to another, the destination device is identified by means of an **IP address**. The route to be followed by the packets, however, is determined by a **routing algorithm**. The result of applying a routing algorithm to a core network is that each core router will be provided with a **routing table** having two columns. The first column is for the IP addresses of all the possible edge routers, while the second column is for the IP addresses of the next core router to be visited on route to the destination. Whenever a packet arrives at a core router, its destination address is compared to the ones listed in the first column. If a match is found, the packet is

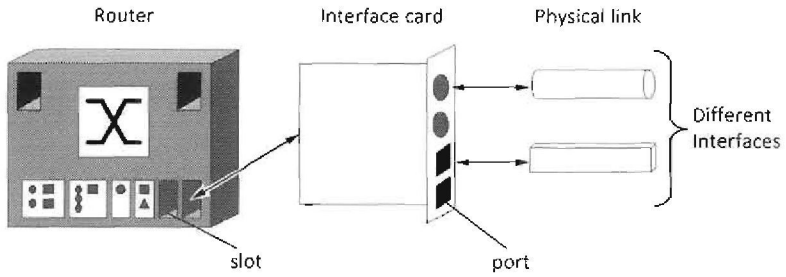

Figure 2.7: Realising link capacities through interface cards

forwarded to the next router listed in the second column.

2.3.4 The network access layer

The network access layer is not really defined within TCP/IP as a separate new layer, but is rather an encapsulation of the data link layer and physical layer found in OSI. It therefore suffices to elaborate more on the functionality of these two OSI layers. The data link layer is primarily concerned with providing the network layer in the OSI model and the internet layer in the TCP/IP model with error free data frames on the destination device. An example of a data link protocol is for instance the Asynchronous Transfer Mode (ATM) that allows the implementation of a connection-oriented service.

The physical layer within the OSI model is concerned with the actual transmission of raw bits over a physical medium. Detailed device specifications are necessary on this layer to ensure interoperability between different technologies. Other issues of importance are, for instance, **multiplexing** addressed by protocols such as the Synchronous Optical Network (SONET), the Synchronous Digital Hierarchy (SDH), and Wavelength Division Multiplexing (WDM).

The capacities of the physical links are realised by the multiplexing scheme employed as well as the type of physical media. As an example, consider an SDH network over a copper cable. The copper itself has physical properties that impose a limit on the volume of information that can be transmitted. However, depending on the level of multiplexing, this limit is significantly lifted to allow the transmission of larger volumes of information. Both SDH and SONET are based on **time division multiplexing** which allows multiple streams of data to be transmitted through a physical medium by providing each stream of data with the entire capacity of the physical medium for a short burst of time. WDM on the other hand, is based on a **frequency division multiplexing** approach. The total frequency band available on a physical medium is divided into several frequency bands with each data stream having exclusive possession of an entire band. For WDM, which is based on fibre optics, the different frequency bands are established by using different wave lengths at which light is sent through the fibre optics.

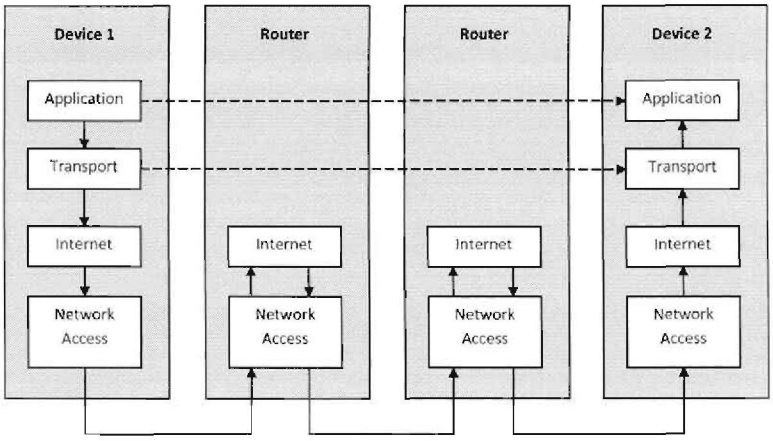

Figure 2.8: Peer-to-peer communication

Interface cards are responsible for performing multiplexing and are slotted into routers to provide the necessary interfaces that allow physical links to be connected to the routers. Figure 2.7 illustrates how an interface card will be slotted into a router and will provide ports that can accommodate different interfaces. Examples of SDH interface capacities are STM-1, STM-4, STM-16 and STM-64 with bit rates 155.52 Mbps, 622.08 Mbps, 2488.32 Mbps, and 9953,28 Mbps respectively.

2.4 Technological Considerations in Network planning

The protocols associated with the two top layers, the application and transport layers, are considered to be **peer-to-peer** or **end-to-end** protocols. That is, the exchange of data through these protocols are only between the communicating devices. Communication on the lower layers is relayed by the protocols on intermediate devices such as routers. Figure 2.8 illustrates the concept where device 1 sends a message to device 2, via two routers. It should be observed from Figure 2.8 that for the second router in the communication chain, only the network access layer is relevant. This implies that some switching of data packets may occur on a physical level without being influenced by the routing algorithm implemented on the internet layer. This provides the basis for the approach of **multi-layered** network architectures.

Figure 2.9 gives an illustration of a two-layer network perspective where **logical links** are associated with the admissible routes as defined within the internet layer, and **physical links** are associated with the admissible routes as defined within the network access layer. For this illustration an SDH multiplexing protocol is considered. This approach to network architecture is not uncommon due to the ownership with respect to the different layers. For instance, within the South African context the majority of physical network infrastructure is owned by the telecommunications operator Telkom. There are, however, several **service providers** that manage their own IP

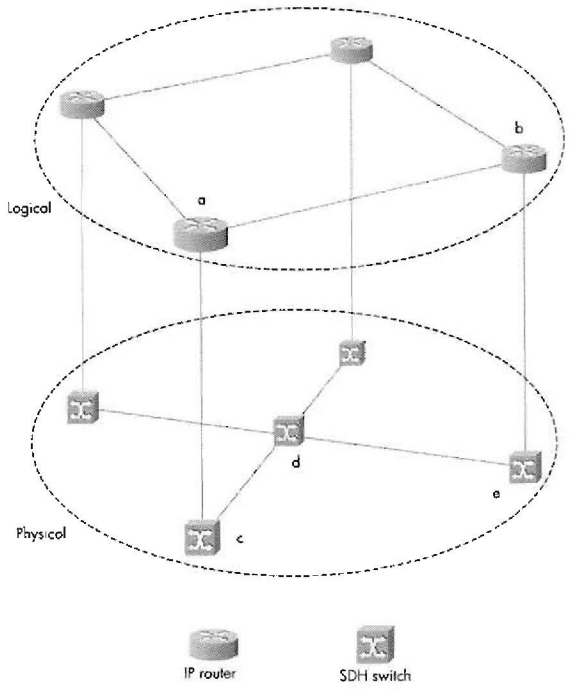

Figure 2.9: Multi-layer network architectures (adapted from [PM04])

networks which overlay the infrastructure provided by Telkom.

The capacities for the logical links are determined by the actual physical capacities and the routing that is performed on the physical layer. For example, the capacity for the logical link a-b in Figure 2.9 depends on the amount of traffic that may be routed on the path c-d-e, which in turn depends on the capacity provided by the physical links c-d and d-e.

Due to the complex nature of doing multi-layer network planning, the remainder of this thesis is devoted to the single layer network design problem that involves only the IP layer.

2.4.1 Choosing the appropriate hardware

The objective in planning a communications network, as already stated in the introductory chapter, is to find the most cost efficient network design that specifies hardware devices to be installed, the type of transmission links to be installed, and the routing strategy to be followed. The network planner is, however, faced with several challenges in achieving this objective. Consider the scenario of a high level network design specifying only the topology and capacities on the link. This type of network design might have been obtained by applying a general network design model typically found in literature. The next step for the network planner is most likely to equip the network with the appropriate technology that will realise the link capacities specified in the network design. For instance, at an edge router interface cards might be needed to multiplex user traffic onto one or more links, whereas for a core router only hardware needed for doing switching would be required.

The selection of appropriate interface cards for the routers at the node locations are typically influenced by economies of scale. For instance, instead of installing four STM-1 interfaces it might be cheaper to rather install one STM-4 interface. However, by installing a single link with high capacity instead of 4 links with lower capacity, survivability might be compromised in favour of cost efficiency. On the other hand, it might be that the only feasible configuration for the node device under consideration is to install a single STM-4 interface due to slot restrictions. These are some of the typical issues faced by the network planner and in practice there may be several other considerations to be taken into account.

2.4.2 Choosing the appropriate routing

The frequency at which routing tables within an IP network are to be updated will determine whether a **dynamic routing** or **static routing** strategy is followed. In the case of dynamic routing, routing tables are typically updated whenever there are changes in the volume of traffic or in the topology of the network. Alternatively the routing tables might also be updated at regular intervals. Static routing, however, implies that routing tables stay fairly stable and are not frequently updated.

The advantage of implementing a dynamic routing strategy is that better congestion control is achieved and reliability is improved. Routers have local information on traffic loads or failures on the adjacent links and could direct packets away from problem areas. In the case of static routing data packets tend to follow fixed routes. This might typically be required by network operators who want to have more control over the traffic on their networks.

Although packet-switching implies a connection-less type of service whereby packets are split up and routed independently, a connection-oriented service can be implemented that resembles **circuit switching**. That is, by forcing the packets to be unsplittable and to follow the same route, a **virtual circuit** is established between communicating devices. Terminology frequently used in literature to distinguish between splittable and unsplittable routing are **bifurcated** and **non-bifurcated**, respectively.

In order to have some assurance of survivability, network planners typically employ some form of bifurcated routing. That is, traffic is diversified such that routes traversing links with a high probability of failure are minimised. The result would typically be that in the unfortunate event of equipment failure traffic will still be supported for all of the communicating node pairs but with less bandwidth available. Thus, performance will decrease but total failure will at least be avoided.

Another very important aspect of routing that influences quality of service is availability of network resources. The route to be traversed by a data packet should be determined in such a way that the number of devices involved in delivering the packet is minimised. This will ensure optimal use of resources. In practice a **hop limit** is enforced that specifies the maximum number of devices

that may be utilised along the route being travelled by the data packets. Another way of achieving availability is to impose a maximum limit on the propagation delay that may be experienced due to the switching of the packets that takes place at each router.

2.5 Modelling Considerations

It should be evident that the network design process has to take into account all potential hardware configurations and routing strategies simultaneously. A specific hardware configuration will realise capacities on the links, which need to be considered when computing a routing. On the other hand, by choosing a routing differently, capacities on some links could be reduced resulting in a reduction of cost. Therefore, the simultaneous optimisation of both capacity and routing, commonly referred to as the **capacitated network design problem**, has been an active research topic ever since the first developments of communication networks. However, due to the complexity of the problem initial models only considered capacities on the links and routing as part of the problem; see for instance Gomory and Hu [GH64]. Therefore, the only variables considered as part of the early network design models were capacity variables representing the amount of bandwidth required on links, as well as flow variables that would provide the flows on links as a result of employing some routing strategy.

With the advances in computing technology and improved algorithms it became feasible to solve the capacitated network design problem by considering additional complexities. For instance, to obtain a more realistic capacity representation, models were developed that required the capacity variables to take on integral solutions. This is useful to model the installation of multiple units of a single type of transmission link, see Pochet and Wolsey [PW92], Magnanti and Mirchandani [MM95]. An alternative approach is to model the capacities as explicit technology types using binary variables. That is, from a list of potential link types having different capacities only one specific one will be selected for installation. The latter is referred to as an **explicit capacity** representation and the former as a **modular capacity** representation.

The typical formulation considered for routing within the network design problem is based on the **multicommodity flow problem**; see Minoux [Min89] for an overview. There are two approaches in formulating a multicommodity flow problem: an **edge flow** formulation could be adopted whereby the flow on each link of the network is represented by a flow variable, or a **path flow** formulation whereby a path variable would represent the amount of flow on some path in the network connecting a communicating node pair. The latter is well suited for modelling non-bifurcated routing strategies as well as imposing path length restrictions in the form of hop limits or propagation delay limits.

Depending on whether a bifurcated or non-bifurcated routing model is considered, the flow

variables will be defined as either continuous or binary. For bifurcated routing it may, however, be required in some cases that the solutions to the flow variables be integer. The data in a physical layer of a communication network, such as an SDH network, is transported in discrete units. Otherwise, for modelling a typical logical layer such as an IP network, fractional routing will suffice.

Chapter 3

Network Design with Demand Uncertainty

3.1 Introduction

Optimisation under uncertainty is nothing new and has been around from as early as the nineteen fifties [Dan55]. The application thereof, however, has only recently become an active point of discussion. The main reason might be that many deterministic optimisation problems, specifically combinatorial problems, are already difficult to solve. Therefore, much of the effort has gone into developing theories and algorithms for solving the "easier" deterministic problem. With the advances made in the field of combinatorial optimisation and new technologies, solving problems with elements of uncertainty has become more attainable. Furthermore, the improvements made towards data storage and data management have also made it easier to obtain large volumes of historical data, making uncertainty modelling more sensible.

In literature there are mainly two ways of dealing with demand uncertainty within the network design context. Firstly, if something is known about the distribution properties of the demand requirements, a stochastic programming approach can be followed. Otherwise, if such information is not available, then a box uncertainty model can be applied where demand requirements are modelled as a polyhedron of uncertainty. The latter approach is known in literature as Robust Optimisation.

3.2 Stochastic Programming

3.2.1 Basic concepts

From a very early stage in the development of linear programming models and techniques, practitioners have realised the restrictive nature of deterministic optimisation models. A **stochastic**

programming framework was developed by Dantzig [Dan55] where decision variables are split between two or more stages. The first stage variables are deterministic in the sense that their values can be determined without taking uncertainty into account. For a telecommunication network design application the first stage variables will typically correspond to infrastructure decisions such as placement of hardware components, installation of link capacities, etc. In a subsequent stage variables are subject to uncertainty and are dependent on the future values of the model parameters. For a network design application the future demand requirements are unknown and, therefore, flow variables are treated as second stage variables since the flow can only be realised later when the demand becomes known. The advantage of this modelling approach is that optimal choices can be made for the first stage variables by balancing the second stage variables against the outcome of uncertainty.

The following gives a general formulation of a two-stage stochastic linear programming problem with first stage variables $x \in \mathbb{R}^{n1}$ and second stage variables $y \in \mathbb{R}^{n2}$.

$$\begin{aligned} \min\ & cx + E[Q(x,\omega)] \\ \text{s.t. } & Ax = b \\ & x \geq 0 \end{aligned}$$

with

$$Q(x,\omega) = \min\{q(\omega)y : W(\omega)y = h(\omega) - T(\omega)x, y \geq 0\}$$

The term $E[Q(x,\omega)]$ denotes the expected value of the so called **recourse function** $Q(x,\omega)$ with respect to the random event $\omega \in \Omega$. The model parameters dependent on the outcome of the random event ω are the second stage cost vector from the mapping $q : \Omega \mapsto \mathbb{R}^{n2}$, the second stage constraint coefficients from the mappings $W : \Omega \mapsto \mathbb{R}^{m2 \times n2}$ and $T : \Omega \mapsto \mathbb{R}^{m2 \times n1}$ and the second stage right-hand side vector from the mapping $h : \Omega \mapsto \mathbb{R}^{m2}$.

The difficulty in solving a two-stage stochastic programming problem lies in finding the expected value of the recourse function. If the density function that is required for the calculation of the expected value $E[Q(x,\omega)]$ is known, the above formulation results in a non-linear programming problem. An alternative approach to remedy the problem is to assume discrete random parameters that will allow approximating the recourse function as a linear function. That is, by replacing the random parameters in $Q(x,\omega)$ with a set of discrete parameters q^s, W^s, T^s, and h^s which may realise with probability p^s for $s \in \mathcal{S}$, $\mathcal{S} = \{1, 2, \ldots, S\}$, the two-stage stochastic programming problem can be written in the following extended linear programming format:

$$
\begin{aligned}
\min\ & cx + \sum_{s \in \mathcal{S}} q^s y^s \\
\text{s.t. } & Ax = b, \\
& T^s x + W^s y^s = h^s \quad \forall s \in \mathcal{S} \\
& y^s \geq 0 \qquad\qquad\quad \forall s \in \mathcal{S} \\
& x \geq 0
\end{aligned}
$$

Several decomposition techniques have been developed that exploit the special structure that is found in the constraint matrix of the extended linear programming format. See, for instance, Ruszczyński [Rus55] for an overview of decomposition methods applied to stochastic programming problems.

Another interesting topic in stochastic programming is optimisation with probabilistic constraints, also referred to as **chance constrained programming**. If we consider the random event $\omega \in \Omega$, then the chance constrained programming problem can be stated as:

$$
\begin{aligned}
\min\ & cx \\
\text{s.t. } & Ax = b, \\
& P\{T(\omega)x \geq h(\omega)\} \geq \alpha \\
& x \geq 0
\end{aligned}
$$

Depending on the probabilistic assumptions regarding $T(\omega)$ and $h(\omega)$, it may be possible to obtain an equivalent deterministic linear programming formulation.

3.2.2 Application of stochastic programming to network design

A chance constrained programming problem has been suggested by Dempster et al. [DMT97] for simultaneously allocating capacity to virtual paths within an ATM network and assigning feasible flows to the virtual paths. The capacity and flow variables are continuous and both the set of demand constraints and capacity constraints are treated as probabilistic. The authors provided an equivalent deterministic linear programming formulation for the problem and presented results for a network problem with 30 nodes, 70 edges and nearly 300 pairs of traffic demands.

A two stage stochastic programming formulation has been suggested by Lisser et al. [LOVG99] for finding the optimal capacity assignment by taking into account penalty costs for demand requirements not met. The capacity and flow variables are continuous and as a solution approach an analytic centre cutting plane method is used. Computational results are provided for 16 test instances containing up to 26 nodes and 53 edges. A maximum of up to 80 demand pairs have been considered and the number of scenarios range between 2^8 and 3^7.

In the paper by Riss and Andersen [RA02] a stochastic integer programming model is considered for the problem of network design with uncertain demands. For this model modular capacities are considered, i.e., capacities can be installed in integer multiples of a low bandwidth type and a high bandwidth type. A bifurcated continuous flow routing model is considered. The solution approach applied is a modified L-shape method that combines Benders decomposition with a branch-and-cut scheme. Proofs are provided to generalise well known valid inequalities, such as metric and partition inequalities, to the case where multiple scenarios for modelling random demands are considered.

3.3 Robust Optimisation

3.3.1 Basic concepts

A stochastic programming approach requires knowledge of the probability distributions for the uncertain model parameters. This information is, however, rarely available in practice. Furthermore, to have an accurate representation of the problem a scenario based approach would require an enormous number of variables for a fine grained discretization of the probability distributions. A **robust optimisation** framework has, therefore, been suggested as a complementary alternative. Random variables are modelled as uncertain model parameters that belong to some uncertainty sets and the optimisation problem is to find optimal solutions that are protected against worst case values from the uncertainty sets.

One of the first well known approaches to a robust optimisation is found in the paper by Soyster [Soy73] where the following linear programming problem is considered:

$$\begin{aligned} &\min\ cx \\ &\text{s.t.}\ \sum_{j=1}^{n} a_j x_j \leq b, \quad \forall a_j \in \mathcal{U}_j, \quad j = 1, 2, \ldots, n \end{aligned}$$

with $\mathcal{U}_j$ for $j = 1, 2, \ldots, n$ convex uncertainty sets. Since the uncertainty sets are column-wise, it can be shown that the above problem is equivalent to

$$\begin{aligned} &\min\ cx \\ &\text{s.t.}\ \tilde{A}x \leq b \end{aligned}$$

with each element of $\tilde{A}$ given by $\tilde{a}_{ij} = \sup_{a_j \in \mathcal{U}_j}(a_{ij})$. Clearly, the solution to the optimisation problem suggested by Soyster is extremely conservative since all the entries in the constraint matrix $\tilde{A}$ simultaneously take on the worst case values from the uncertainty sets.

By considering an interval based uncertainty set, that is, modelling the constraint coefficient a_{ij} as symmetric bounded random variable that can take on a value within the interval $[\bar{a}_{ij} - \epsilon|\bar{a}_{ij}|, \bar{a}_{ij} + \epsilon|\bar{a}_{ij}|]$ with $\bar{a}_{ij}$ some point estimate and $\epsilon \in \mathbb{R}_+$ a scaling factor, the problem by Soyster can be formulated as the following linear programming problem:

$$\begin{aligned}
&\min\ cx \\
&\text{s.t.}\ \sum_{j=1}^{n} \bar{a}_{ij}x_j + \sum_{j\in J_i} \epsilon|\bar{a}_{ij}|y_j \leq b_i \quad i = 1,2,\ldots,m \\
&\qquad -y_j \leq x_j \leq y_j \quad j = 1,2,\ldots,n \\
&\qquad y_j \geq 0 \quad j = 1,2,\ldots,n
\end{aligned}$$

with J_i the set of indices of the coefficients in row i that are subject to uncertainty. It is clear that the above formulation is still conservative in the sense that the constraints have to satisfy the worst case values from the uncertainty sets. An approach described by Bertsimas and Sim [BS04] to counteract the conservative behaviour is to include into the above model a budget of uncertainty. That is, to specify a parameter that will restrict the number of coefficients that will be subject to uncertainty. Another approach by Ben-Tal and Nemirovsky [BN99] that allows the model to be less conservative is to make use of a constraint-wise uncertainty formulation. The resulting linear optimisation problem, called the **Robust Counterpart** (RC), is the following:

$$\begin{aligned}
&\min\ cx \\
&\text{s.t.}\ a_i x \leq b_i, \quad \forall a_i \in \mathcal{U}_i, \quad i = 1,2,\ldots,m
\end{aligned}$$

with $\mathcal{U}_i$ for $i = 1,2,\ldots,m$ convex closed uncertainty sets. The above formulation of the RC can alternatively be rewritten as:

$$\min\left\{cx : \max_{a_i\in\mathcal{U}}\{a_i x\} \leq b_i, \quad i = 1,2,\ldots,m\right\} \tag{3.1}$$

with $\mathcal{U} = \mathcal{U}_1 \times \mathcal{U}_2 \times \cdots \times \mathcal{U}_m$. It is shown in Ben-Tal and Nemirovsky [BN99] that the RC reduces to a linear programming problem provided that the uncertainty set $\mathcal{U}$ is a polyhedron. This can easily be illustrated by letting $\mathcal{U}_i = \{a_i : G_i a_i \leq g_i\}$. Then the RC can be written as

$$\min\{cx : \max\{a_i x : G_i a_i \leq g_i\} \leq b_i, \quad i = 1,2,\ldots,m\} \tag{3.2}$$

Now consider the inner optimisation problem for a fixed x. Then by associating the variables ρ_i with the set of constraints $G_i a_i \leq g_i$, the dual of the inner optimisation problem is the following:

$$\begin{aligned}
&\min\ \rho_i g_i \\
&\text{s.t.}\ \rho_i G_i = x \\
&\qquad \rho_i \geq 0
\end{aligned}$$

By substituting the above dual problem back into the RC formulation (3.2), the following linear

RC is obtained:

$$\begin{aligned} \min\ & cx \\ \text{s.t. } & \rho_i g_i \leq b_i \quad i = 1, 2, \ldots, m \\ & \rho_i G_i = x \quad i = 1, 2, \ldots, m \\ & \rho_i \geq 0 \quad i = 1, 2, \ldots, m \end{aligned}$$

By considering more complex uncertainty sets such as ellipsoidal sets, the problem becomes a second-order cone program. A two-stage robust approach as presented by Ben-Tal et al. [BGGN04] is also possible. Let $a_i = (a_i^x a_i^y)$ be the coefficients associated with the variables x and y respectively. Within the context of a RC optimisation problem, the variables x and y are considered to be non-adjustable since for all $(a_i^x a_i^y) \in \mathcal{U}_i$, the inequality $a_i^x x + a_i^y y \leq b_i$ for $i = 1, 2, \ldots, m$ must hold. That is, the RC can be stated somewhat differently as

$$\min \{cx : \exists y\ \forall (a_i^x a_i^y) \in \mathcal{U}_i : a_i^x x + a_i^y y \leq b_i, \quad i = 1, 2, \ldots, m\} \tag{3.3}$$

Now, by treating the variables y as adjustable, that is, assuming that y may be dependent on the outcome of an uncertainty event, we obtain the **Adjustable Robust Counterpart** (ARC)

$$\min \{cx : \ \forall (a_i^x a_i^y) \in \mathcal{U}_i, \ \exists y : a_i^x x + a_i^y y \leq b_i, \quad i = 1, 2, \ldots, m\} \tag{3.4}$$

3.3.2 Application of robust optimisation to network design

In the paper by Altm et. al [AABP04], a compact formulation is provided for the Virtual Private Network design problem under traffic uncertainty. The traffic demand requirements are specified as a traffic polytope and details are provided for a column generation and cutting plane algorithm.

Atamtürk and Zhang [AZ07] describe a two stage robust optimisation approach for solving the network design problem with demand uncertainty. Compared to the usual setup of having capacity variables as first stage and flow variables as second stage, the approach followed by Atamtürk and Zhang [AZ07] is to partition the graph associated with the problem into first stage and second stage links. The result of having done this is that a set of capacity variables and flow variables is jointly part of the first stage variables, and another set of capacity variables and flow variables is jointly part of the second stage variables. This setup is useful for capacity expansion type of applications. Theoretical results are provided with regard to the computational complexity of the approach and computational results are presented comparing solutions from two stage robust optimisation with solutions from two stage stochastic programming.

A robust approach for solving the capacity expansion problem with uncertain demand is described by Ordóñez and Zhao [OZ07]. An adjustable robust model is suggested along with conditions making the problem more tractable. The model only caters for continuous capacity and flow variables, and computational results they present are based on a 21-node network with 45 edges.

3.4 Choosing a Modelling Approach

The problem being considered in this thesis is solving the **survivable network design problem** taking into account a set of non-simultaneous traffic demand vectors. The philosophy is that variation in the network traffic that brings about uncertainty can be approximated by a finite set of non-simultaneous traffic demand vectors. Even though no assumptions are made about the distributional properties of the demand vectors, the problem is approached within the stochastic programming framework. Each demand vector in our set of non-simultaneous demand vectors is considered to be a realisation of a random event with probability of one. However, the true power of stochastic programming will not be exploited since the objective function of the model considered in this thesis has no revenue part that will drive the second stage flow variables. Consequently, demand requirements can only be met by implementing hard constraints. It is for this reason that many practitioners prefer to label the problem under investigation in this thesis as the **multi-hour network design problem**. It should be noted that although the main approach followed here is not based on a robust optimisation framework, a robust implementation was used to achieve tractability for a specific aspect of the problem being addressed. The remainder of this section is devoted to literature related to the multi-hour network design problem.

In the paper by Medhi [Med95] the multi-hour network design problem for reconfigurable ATM networks is considered. The model assumes unsplittable flows by considering a single virtual path (VP) between demands. The concept of different traffic classes is introduced which allows more than one VP per demand pair. Path selection variables are binary and are indexed according to a traffic class, a commodity, and a time period. A modular capacity model is considered since integer variables are used to define the number of capacity units that can be installed on an edge. A Lagrangian decomposition approach is followed by applying subgradient optimisation. In the book by Pioro and Medhi [PM04] variations of the multi-hour network design problem are presented that include link blocking for circuit-switched networks, reconfigurable splittable flows, and non-reconfigurable unsplittable flows.

The mixed-integer programming model presented by Dutta [Dut94] uses an explicit capacity model where binary variables are used for modelling different capacity types that can be installed on an edge. Integer variables are used for specifying the number of circuits carried on one or more paths in order to satisfy demand requirements. A Lagrangian based heuristic is used as a

solution approach. In the paper by Chari and Dutta [CD93] a similar model is presented, but with the modification that a modular capacity model is considered. That is, the capacity variables are treated as pure integers and as a result the number of units installable for one type of capacity is modelled instead of different capacity types. A Benders decomposition approach is followed whereby both the master and subproblems are solved heuristically.

The model suggested by Amiri and Pirkul [AP99] is a non-linear mixed integer programming model, where the non-linearity is a result of the requirement for the demand to be modelled as independent M/M/1 queues in which links are treated as servers with service rates proportional to the edge capacities. Binary variables are used to model discrete capacity types on the edges and also for assigning paths to edges for each time period. A Lagrangian based heuristic is used as a solution approach.

In the paper by Labbé et al. [LSSW99], the authors investigated the network synthesis problem with non-simultaneous multicommodity flow requirements. A Lagrangian bounding procedure is presented and heuristics for finding primal solutions are provided for both the multi-hour and the restoration problems. A modular capacity model and a bifurcated routing model is considered with integrality restrictions on the flow variables.

Chapter 4

Mathematical Models

4.1 Description

In this thesis the survivable network design problem with uncertain traffic requirements is modelled as a mixed integer programming problem. The model comprises two main parts, namely a detailed hardware part and a routing part. The hardware part of the model has to do with the equipment required for the installation of a telecommunication network. The decision variables are either integer or binary, and a solution will indicate certain hardware components that have to be installed. The typical constraints in the hardware part is responsible for expressing technological restrictions when configuring the hardware. For example, if a solution to the hardware part suggests that a router from a specific vendor should be installed at some location, then there should be enough interface cards available in the router to accommodate the physical links that have to be connected to the router. A solution to the hardware part will eventually define the capacity on the edges.

The routing part of the model is responsible for satisfying demand requirements subject to the capacity on the links that were specified by a solution of the hardware part. The typical variables found in the routing part are called flow variables. These variables represent the solution for the routing of data through the telecommunication network. Built into the routing part are also constraints responsible for satisfying survivability requirements.

It is clear that within the stochastic programming framework the variables defined for the hardware part can be considered as first stage variables and the variables defined for the routing part as the second stage variables.

For the representation of uncertainty within the demand requirements a set of non-simultaneous **demand vectors** is considered. For the sake of notation we prefer to use the term demand vector instead of demand matrix. Unlike for the case of typical stochastic programming applications, it is here required that all demand requirements be satisfied. That is, once a solution to the hardware part has been found, the routing constraints must be satisfied non-simultaneously for all demand vectors.

When considering the network planning problem where more than one demand vector has to be satisfied, two routing alternatives are distinguished. **Dynamic routing** is a routing implementation that allows multiple demand vectors to be routed independently, whereas **static routing** imposes the restriction that all demand vectors have to be routed according to the same routing.

The type of side constraint that will be imposed on the model will depend on certain model assumptions. In literature there are a variety of models motivated by specific applications. For instance, when planning the implementation of an ATM network it is required that data between two communicating locations be sent along a single route. Furthermore, all demand vectors have to be routed statically.

The approach followed in this thesis is to keep the model as general as possible. No application or technology specific assumption will be made, although specialization should only be a matter of adding more side constraints.

For ease of use all parameters and definitions related to subsequent models are listed in Table A.1 as part of Appendix A. The table provides a short description of each parameter or definition, as well as the location of where each parameter or definition was introduced.

4.1.1 General parameters

- Let $G = (V, \mathcal{E})$ be a complete graph with V the set of potential node locations where hardware may be installed and $\mathcal{E}$ the set of undirected edges where each edge $e \in \mathcal{E}$ represents a connection between two node locations.

- Let $H = (V, E)$ be the **supply graph** with $E \subseteq \mathcal{E}$ the potential edges for which capacity may be installed.

- Let $\phi(v) \subseteq E$ denote the set of incident edges to the node $v \in V$.

- Let $J = (V, K)$ be the **demand graph** with $K \subseteq \mathcal{E}$ the edges connecting the node pairs for which traffic demands exist. In the remainder of this thesis we will refer to a **commodity** $k \in K$, with K the set of commodities. Since we consider multiple demand matrices, the set of commodities K is obtained by taking the union over all commodities defined for each of the multiple demand matrices.

- The total traffic demand for a commodity $k \in K$ is denoted by d_k. The vector $d = (d_k)_{k \in K}$ is referred to as a **demand vector**.

- The set $\mathcal{T} = \{1, 2, \ldots, |\mathcal{T}|\}$ is used to index the set of non-simultaneous demand vectors $D = \{d^1, d^2, \ldots, d^{|\mathcal{T}|}\}$. The probabilities for the demand vectors to realise are all equal.

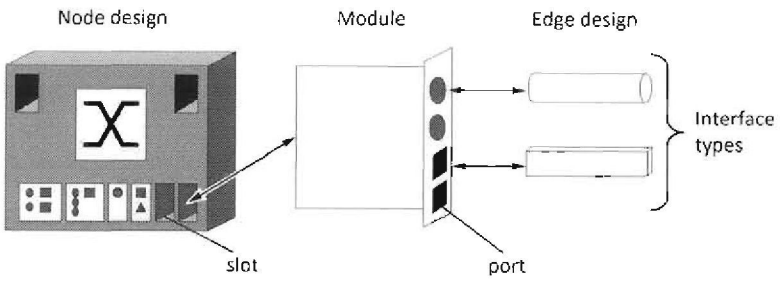

Figure 4.1: Abstracting potential hardware

4.1.2 Parameters related to hardware

The hardware model allows a detailed representation of potential hardware configurations and is based on the work by Kröller [Krö03]. Potential technologies are abstracted into so called node designs and the optimisation process is responsible for selecting the most cost effective node design for each node that will satisfy capacity requirements. Similarly, edge designs enable the modeling of different link technologies that may have different capacities, cost structures and port requirements. Figure 4.1 illustrates how potential hardware configurations are abstracted into node and edge designs.

At each node $v \in V$ a set of admissible node designs $\mathcal{N}(v)$ is defined of which not more than one node design $n \in \mathcal{N}(v)$ is allowed to be installed. Likewise, for each edge $e \in E$ no more than one edge design $l \in \mathcal{L}(e)$ are allowed to be selected for installation from the predefined set of admissible edge designs $\mathcal{L}(e)$.

The attributes of a node design $n \in \mathcal{N}(v)$ are a set of installable module types $\mathcal{M}(n)$, a limit $M^{m,n} \in \mathbb{Z}_+$ on the number of modules of type $m \in \mathcal{M}(n)$ that may be installed, the available number of slots $S^n \in \mathbb{Z}_+$ that can be occupied by one or more modules, a switching capacity $C^n \in \mathbb{Z}_+$, and a cost $c^n \in \mathbb{R}_+$. Furthermore, each type of module $m \in \mathcal{M}(n)$ is designed to occupy a number of slots $S^m \in \mathbb{Z}_+$ and can accommodate several types of interfaces $\mathcal{I}(m)$ through the availability of ports. Each module also has a cost of $c^m \in \mathbb{R}_+$. The term module is being used generically, but in practice it will typically be referred to as an interface card.

To enable the matching of edge technologies with node technologies, each edge design $l \in \mathcal{L}(e)$ for an edge $e \in E$ has a set of interface types $\mathcal{I}(l)$. Matching the interface types of a module m as part of a node design n, with the interface types of an edge e, allows us to connect the edge e with the node n. There is a limit $I_i^m \in \mathbb{Z}_+$ on the number of interfaces of type $i \in \mathcal{I}(m)$ for a module m and a limit $I_i^l \in \mathbb{Z}_+$ on the number of interfaces of type $i \in \mathcal{I}(l)$ for an edge design l.

For ease of notation let $\mathcal{M}(v) := \bigcup_{n \in \mathcal{N}(v)} \mathcal{M}(n)$ denote the set of modules installable at node $v \in V$ and $\mathcal{I}(v) := \bigcup_{m \in \mathcal{M}(v)} \mathcal{I}(m)$ the set of potential interfaces at node $v \in V$.

4.1.3 Parameters related to routing

The routing model considered in this thesis is based on a path flow formulation. That is, traffic between communicating node pairs is modelled as flow along paths linking the communicating node pairs. Let $p \in \mathcal{P}$ be a path between a communicating node pair, with $\mathcal{P}$ containing all possible paths for all possible node pairs. The subset $\mathcal{P}(k) \subseteq \mathcal{P}$ contains all paths that can route traffic between the communicating node pair corresponding to the commodity $k \in K$. The set $E(p)$ denotes the set of edges representing the path $p \in \mathcal{P}(k)$, for a commodity $k \in K$.

The advantage of a path flow formulation is the ability to impose path length restrictions. For instance, a parameter for each commodity $k \in K$ can be used to impose an upper bound on the number of edges being traversed by any path that routes (part of) the demand d_k. The idea is to avoid long paths that might incur a long propagation delay due to the switching that takes place at the routers. Path length restriction might also be useful as a way of managing survivability since long paths will have a higher probability of being affected in the case of component failures than shorter paths.

The concept of **failure states** is used to facilitate the modelling of survivability. Different failure scenarios can be modelled by letting each failure state $s \in S$ contain all the network components (nodes and edges) that fail simultaneously. In this thesis only failure states that comprise single edge failures will be considered. The set $S(p)$ is used to denote all failure states that will affect the path $p \in \mathcal{P}(k)$, for a commodity $k \in K$. A **diversification** parameter $\delta_k \in (0, 1] \subseteq \mathbb{R}_+$, for all $k \in K$, is introduced that defines a limit on the fraction of the demand d_k that can be routed through any node or edge that is affected by a failure state. The result is that traffic is distributed more evenly across the network and in the case of component failure, traffic can be redirected through surviving paths.

The set $\mathcal{P}(k, s) \subseteq \mathcal{P}(k)$ is used to index all paths for a commodity $k \in K$ that are affected by a failure state $s \in S$ and the set $\mathcal{P}(k, e) \subseteq \mathcal{P}(k)$ denotes all paths for a commodity $k \in K$ that traverse an edge $e \in E$.

4.2 Survivable Network Design with Multiple Demand Vectors

With respect to network design problems in general, the model to be applied in this thesis embeds an explicit capacity model whereby capacity levels are dictated by the type of node and edge designs installable. Furthermore, a bifurcated routing model is considered and alternative models are presented to enforce either dynamic or static routing policies.

4.2.1 Detailed hardware model

The following decision variables are required to model the installation of the node designs, the modules, and the edge designs:

- $x_v^n \in \{0,1\}$ indicates whether node design $n \in \mathcal{N}(v)$ is installed at node $v \in V$ or not.
- $x_e^l \in \{0,1\}$ indicates whether link design $l \in \mathcal{L}(e)$ is installed on edge $e \in E$ or not.
- $x_v^m \in \mathbb{Z}_+$ gives the number of modules of type $m \in \mathcal{M}$ installed at node $v \in V$.
- $y_e \in \mathbb{Z}^{|E|}$ is an auxiliary variable that represents the resulting capacity on an edge $e \in E$.

If it is assumed that each node $v \in V$ has the same set of admissible node designs $\mathcal{N}(v)$ and modules $\mathcal{M}(n)$, and each edge $e \in E$ has the same set of admissible edge designs $\mathcal{L}(e)$, then the variables for the hardware model may be written as the adjoint vector $(x,y) \in \mathbb{Z}_+^{|\mathcal{N}|\times|V|+|\mathcal{L}|\times|E|+|\mathcal{M}|\times|V|} \times \mathbb{Z}^{|E|}$. If on the other hand the set of admissible node designs and modules vary among the nodes and the set of admissible edge designs vary among the edges, then the adjoint vector (x,y) will have a smaller dimension.

The set of feasible solutions for the hardware model is defined by the following polyhedron:

$$\mathcal{H} := \Bigg\{ (x,y) \in \mathbb{Z}_+^{|\mathcal{N}|\times|V|+|\mathcal{L}|\times|E|+|\mathcal{M}|\times|V|} \times \mathbb{Z}^{|E|} :$$

$$\sum_{n\in\mathcal{N}(v)} x_v^n \leq 1 \quad \forall\, v \in V \tag{4.1}$$

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

$$\sum_{l\in\mathcal{L}(e)} C^l x_e^l = y_e \;\forall\, e \in E \tag{4.3}$$

$$\sum_{e\in\phi(v)} \sum_{l\in\mathcal{L}(e)} C^l x_e^l - \sum_{n\in\mathcal{N}(v)} C^n x_v^n \leq 0 \quad \forall\, v \in V \tag{4.4}$$

$$\sum_{m\in\mathcal{M}(v)} S^m x_v^m - \sum_{n\in\mathcal{N}(v)} S^n x_v^n \leq 0 \quad \forall\, v \in V \tag{4.5}$$

$$\sum_{e\in\phi(v)} \sum_{l\in\mathcal{L}(e)} \sum_{i\in\mathcal{I}(l)} I_i^l x_e^l - \sum_{m\in\mathcal{M}(v)} \sum_{i\in\mathcal{I}(m)} I_i^m x_v^m \leq 0 \quad \forall\, v \in V \tag{4.6}$$

$$x_v^m - \sum_{n\in\mathcal{N}(v)} M^{m,n} x_v^n \leq 0 \quad \begin{matrix} \forall\, v \in V, \\ \forall\, m \in \mathcal{M}(v) \end{matrix} \Bigg\} \tag{4.7}$$

Constraints (4.1) and (4.2) enforce the rule that only one node design be assigned to a node and only one edge design be assigned to an edge respectively. Constraint sets (4.3) and (4.4) ensure that the capacity of a node is sufficient to switch all the capacity of its incident edges. Note that y_e is an auxiliary variable that will take on a value as a result of applying valid inequalities generated during

the solution process (see Section 4.3). To ensure that the slot requirements of installed modules do not exceed the available slots of a node design, constraint set (4.5) is applied. Constraint set (4.6) will ensure that the number of interfaces of type $i \in \mathcal{I}(v)$ that will be installed at node v is sufficient to accommodate the number of interfaces of type $i \in \mathcal{I}(l)$ required by the incident edges. An upper bound on the maximum number of modules of type m at a node v is defined by the constraint set (4.7).

The objective function of the hardware model is defined by considering the minimisation of the installation cost of the node designs, the modules, and the edge designs. That is,

$$\min \; cx \;=\; \min \sum_{v \in V} \left(\sum_{n \in \mathcal{N}(v)} c^n x_v^n + \sum_{m \in \mathcal{M}(v)} c^m x_v^m \right) + \sum_{e \in E} \sum_{l \in \mathcal{L}(e)} c^l x_e^l$$

For the sake of notational ease let $\bar{x} \in \mathcal{H}$ denote the adjoint vectors $(x, y) \in \mathcal{H}$ representing all the variables found in the hardware model.

4.2.2 Single demand vector routing

For a given capacity vector $y \in \mathbb{R}_+^{|E|}$ and a demand vector $d \in \mathbb{R}_+^{|\mathcal{K}|}$, the following polyhedron $F(y, d, \delta)$ defines all feasible flows that will satisfy the demand vector d subject to the capacity y:

$$F(y, d, \delta) := \Bigg\{ f \in \mathbb{R}_+^{|\mathcal{P}|} :$$

$$\sum_{p \in \mathcal{P}(k)} f_p = d_k \;\forall\; k \in K \tag{4.8}$$

$$\sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} f_p \leq y_e \;\forall\; e \in E \tag{4.9}$$

$$\sum_{p \in \mathcal{P}(k,s)} f_p \leq \delta_k d_k \quad \begin{matrix} \forall\; k \in K, \\ \forall\; s \in \mathcal{S} \end{matrix} \Bigg\} \tag{4.10}$$

The variable $f_p \in \mathbb{R}_+$ is used to define the flow of traffic on path $p \in \mathcal{P}$. In the remainder of this thesis the vector $f \in F(y, d, \delta)$ is at times also referred to as **a routing**. For ease of notation we use $F(y, d) \supseteq F(y, d, \delta)$ to denote the flow polyhedron without any diversification requirements.

4.2.3 Dynamic routing for multiple demand vectors

For a given capacity vector $y \in \mathbb{R}_+^{|E|}$ and a set of demand vectors $D = \{d^1, d^2, \ldots, d^{|\mathcal{T}|}\}$ the following polyhedron $F(y, D, \delta)$ defines all feasible routings for the survivable network routing problem:

$$F(y, D, \delta) := \Bigg\{ f \in \mathbb{R}_+^{|\mathcal{P}| \times |\mathcal{T}|} :$$

$$\sum_{p \in \mathcal{P}(k)} f_p^t = d_k^t \;\; \forall\, k \in K,\; \forall\, t \in \mathcal{T} \tag{4.11}$$

$$\sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} f_p^t \leq y_e \quad \begin{matrix} \forall\, e \in E, \\ \forall\, t \in \mathcal{T} \end{matrix} \tag{4.12}$$

$$\sum_{p \in \mathcal{P}(k,s)} f_p^t \leq \delta_k d_k^t \quad \begin{matrix} \forall\, k \in K, \\ \forall\, s \in \mathcal{S}, \\ \forall\, t \in \mathcal{T} \end{matrix} \Bigg\} \tag{4.13}$$

The preceding definition of $F(y, D, \delta)$ implies a dynamic routing model since the demand vectors $d^1, d^2, \ldots, d^{|\mathcal{T}|}$ can be routed independently of each other according to the routings $f^1, f^2, \ldots, f^{|\mathcal{T}|}$. For ease of notation we use $F(y, D) \supseteq F(y, D, \delta)$ to denote the flow polyhedron without any diversification requirements.

4.2.4 Static routing for multiple demand vectors

In some cases, due to different planning requirements, static routing might be preferred where the same routing is required for all demand vectors. For a given capacity vector $y \in \mathbb{R}_+^{|E|}$ and a set of demand vectors $D = \{d^1, d^2, \ldots, d^{|\mathcal{T}|}\}$ the following polyhedron $R(y, D, \delta)$ defines all feasible routings for the survivable network problem with static routing:

$$R(y, D, \delta) := \Bigg\{ r \in \mathbb{R}_+^{|\mathcal{P}|} :$$

$$\sum_{p \in \mathcal{P}(k)} r_p = 1, \;\; \forall\, k \in K \tag{4.14}$$

$$\sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} r_p d_k^t \leq y_e, \quad \begin{matrix} \forall\, e \in E, \\ \forall\, t \in \mathcal{T} \end{matrix} \tag{4.15}$$

$$\sum_{p \in \mathcal{P}(k,s)} r_p \leq \delta_k, \quad \begin{matrix} \forall\, k \in K, \\ \forall\, s \in S \end{matrix} \Bigg\} \tag{4.16}$$

For ease of notation we use $R(y, D) \supseteq R(y, D, \delta)$ to denote the flow polyhedron without any diversification requirements.

4.2.5 Complete model with different routing combinations

Let d be a single demand vector, D a set of non-simultaneous demand vectors, and δ a diversification vector, then the following compact notation is used to distinguish between survivable network design problems with either single demand vector routing, dynamic routing, or static routing:

The Survivable Network Design Problem with single demand vector routing

$$\text{SNDP}(d,\,\delta): \quad \min\, cx$$
$$\text{s.t.} \bigcup_{(x,y)\in\mathcal{H}} F(y,d,\delta) \neq \emptyset$$

The Survivable Network Design Problem with Dynamic Routing

$$\text{SNDP-DR}(D,\,\delta): \quad \min\, cx$$
$$\text{s.t.} \bigcup_{(x,y)\in\mathcal{H}} F(y,D,\delta) \neq \emptyset$$

The Survivable Network Design Problem with Static Routing

$$\text{SNDP-SR}(D,\,\delta): \quad \min\, cx$$
$$\text{s.t.} \bigcup_{(x,y)\in\mathcal{H}} R(y,D,\delta) \neq \emptyset$$

The generic notations SNDP(d, δ), SNDP-DR(D, δ) or SNDP-SR(D, δ) are used in subsequent sections, to distinguish the type of survivable network design problem being solved. Alternative forms of the notations are also applied, for example the use of SNDP(d^1, δ^1) will refer to solving SNDP(d, δ) with $d := d^1$ and $\delta := \delta^1$. This in turn also implies the application of the notation $F(y, d^1, \delta^1)$ which refers to the polyhedron $F(y, d, \delta)$ by considering $d := d^1$ and $\delta := \delta^1$.

It is also convenient when referring to either the problem, SNDP-DR(D, δ) or SNDP-SR(D, δ), to make use of the notation SNDP-DR/SR(D, δ). When survivability is not considered then the notations SNDP(d), SNDP-DR(D) and SNDP-SR(D) are used to distinguish between network design problems with a single demand vector, network design problems with dynamic routing, and network design problems with static routing, respectively.

4.3 Characterising Feasible Capacities

The path flow formulation adopted for the respective routing models in this thesis has the unfortunate property that the number of flow variables in the model increases exponentially with an increase in number of nodes within the underlying supply graph. Therefore, the algorithmic approach followed, as outlined in Chapter 5, is based on a decomposition approach whereby the flow variables are projected out of the problem and are only incorporated back during the solution process in the form of inequalities. These inequalities are in terms of the auxiliary variables y defined in the hardware model and consequently, force the configuration variables in the hardware model to take on appropriate values.

4.3.1 Single demand vector routing

Consider the flow polyhedron $F(y, d)$ for the routing model of a single demand vector d. By associating the dual variables $\pi \in \mathbb{R}^{|K|}$ and $\mu \in \mathbb{R}_+^{|E|}$ with the constraint set (4.8) and (4.9) respectively, we are able to apply the results of the following theorem:

Theorem 4.1 ([Iri71], [KO71]). *For a given capacity vector y^* the polyhedron $F(y^*, d)$ will not be empty iff*

$$\sum_{e \in E} y_e^* \mu_e \geq \sum_{k \in K} d_k \pi_k \tag{4.17}$$

for all $\mu_e \geq 0$, $e \in E$ and where $\pi_k = \min_{p \in \mathcal{P}(k)} \left\{ \sum_{e \in E(p)} \mu_e \right\}$ is the length of the shortest path for each commodity $k \in K$.

The implication of Theorem 4.1 is that if the inequality (4.17) does not hold then the following inequality can be added to the hardware model to ensure feasible routing:

$$\sum_{e \in E} y_e \mu_e \geq \sum_{k \in K} d_k \pi_k \tag{4.18}$$

where y are the auxiliary variables in the hardware model. Inequality (4.18) is referred to as a **metric inequality**. By adding metric inequalities to the hardware model, enough capacity is reserved on the edges to allow a feasible routing.

By associating the dual variables $\gamma \in \mathbb{R}_+^{|K| \times |S|}$ with the diversification constraint set (4.10), the results from the following theorem allows us to derive metric inequalities for the case where survivability is considered.

Theorem 4.2 ([DS94]). *For a given capacity vector y^* and a diversification vector δ, the polyhedron $F(y^*, d, \delta)$ will not be empty iff*

$$\sum_{e \in E} y_e^* \mu_e \geq \sum_{k \in K} d_k \pi_k - \sum_{k \in K} \left(\delta_k d_k \sum_{s \in S} \gamma_k^s \right) \tag{4.19}$$

for all $\mu_e \geq 0$, $e \in E$, for all $\gamma_k^s \geq 0$, $s \in S$, $k \in K$ and where $\pi_k = \min_{p \in \mathcal{P}(k)} \left\{ \sum_{e \in E(p)} \mu_e + \sum_{s \in S(p)} \gamma_k^s \right\}$ is the length of the shortest path for each commodity $k \in K$

4.3.2 Characterising feasible capacities for dynamic routing

Consider the flow polyhedron $F(y, D)$ for the dynamic routing model with a set of demand vectors $D = \{d^1, d^2, \ldots, d^{|T|}\}$ and associate with the constraint set (4.11) and (4.12) the dual variables $\pi \in \mathbb{R}^{|K| \times |T|}$ and $\mu \in \mathbb{R}_+^{|E| \times |T|}$ respectively.

Proposition 4.1. *For a given capacity vector y^* the polyhedron $F(y^*, D)$ will not be empty iff*

$$\sum_{e \in E} y_e^* \mu_e^t \geq \sum_{k \in K} d_k^t \pi_k^t \quad \forall t \in \mathcal{T} \tag{4.20}$$

for all $\mu_e \geq 0$, $e \in E$ and $t \in \mathcal{T}$, and where $\pi_k^t = \min_{p \in \mathcal{P}(k)} \left\{ \sum_{e \in E(p)} \mu_e^t \right\}$ is the length of the shortest path for each commodity $k \in K$, for each $t \in \mathcal{T}$.

Proof. Consider the following linear program that minimises the shortfall in capacity $\alpha^t \in \mathbb{R}_+$ for all $t \in \mathcal{T}$, within the flow polyhedron $F(y^*, D)$ for a given capacity vector y^*.

$$\min \sum_{t \in \mathcal{T}} \alpha^t$$

$$\text{s.t.} \sum_{p \in \mathcal{P}(k)} f_p^t = d_k^t \quad \forall\, k \in K,\ \forall\, t \in \mathcal{T} \tag{4.21}$$

$$\sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} f_p^t - \alpha^t \leq y_e^* \quad \forall\, e \in E,\ \forall\, t \in \mathcal{T} \tag{4.22}$$

To acknowledge feasibility of a capacity vector y^* the optimal solution should yield $\alpha^t = 0$ for all $t \in \mathcal{T}$. By associating the variables $\pi \in \mathbb{R}^{|K| \times |\mathcal{T}|}$ and $\mu \in \mathbb{R}_+^{|E| \times |\mathcal{T}|}$ with the constraints (4.21) and (4.22) respectively, the corresponding dual is obtained:

$$\max \sum_{t \in \mathcal{T}} \left(\sum_{k \in K} d_k^t \pi_k^t - \sum_{e \in E} y_e^* \mu_e^t \right)$$

$$\text{s.t.} \sum_{e \in E} \mu_e^t = 1 \quad \forall\, t \in \mathcal{T} \tag{4.23}$$

$$\pi_k^t - \sum_{e \in E(p)} \mu_e^t \leq 0 \quad \forall\, p \in \mathcal{P}(k), \quad \begin{matrix} \forall\, k \in K, \\ \forall\, t \in \mathcal{T} \end{matrix} \tag{4.24}$$

For a positive objective value that signals insufficient capacity, the following valid inequality can be added to the capacitated network design problem to cut off the infeasible capacity vector y^*:

$$\sum_{t \in \mathcal{T}} \left(\sum_{k \in K} d_k^t \pi_k^t - \sum_{e \in E} y_e \mu_e^t \right) \leq 0 \tag{4.25}$$

with π_k^t for all $k \in K$, $t \in \mathcal{T}$, and μ_e^t for all $e \in E$, $t \in \mathcal{T}$, the solution values obtained by solving the above dual linear programming problem. The inequality (4.25) may be replaced with the following equivalent multiple metric inequalities:

$$\sum_{e \in E} y_e \mu_e^t \geq \sum_{k \in K} d_k^t \pi_k^t \quad \forall t \in \mathcal{T}$$

□

By associating the dual variables $\gamma \in \mathbb{R}_+^{|K| \times |S| \times |\mathcal{T}|}$ with the diversification constraint set (4.13), the results from the following theorem allows us to derive metric inequalities for the case where survivability is considered.

Proposition 4.2. *For a given capacity vector y^* the polyhedron $F(y^*, D, \delta)$ will not be empty iff*

$$\sum_{e\in E} y_e^* \mu_e^t \geq \sum_{k\in K} d_k^t \pi_k^t - \sum_{k\in K} \left(\delta_k d_k^t \sum_{s\in S} \gamma_{ks}^t \right) \quad \forall t \in \mathcal{T} \tag{4.26}$$

for all $\mu_e^t \geq 0$, $e \in E$ and $t \in \mathcal{T}$, for all $\gamma_{ks}^t \geq 0$, $s \in S$ and $k \in K$, and where $\pi_k^t = \min_{p\in\mathcal{P}(k)} \left\{ \sum_{e\in E(p)} \mu_e^t + \sum_{s\in S(p)} \gamma_{ks}^t \right\}$ is the length of the shortest path for each commodity $k \in K$, and for each $t \in \mathcal{T}$.

Proof. The proof is analogous to that of Proposition 4.1. The proof is sketched informally as part of the discussion on separation in Section 5.4.1. □

Let $W \subseteq V$ be a subset of V with $\bar{W} = V \setminus W$ its complement. The set of edges $E(W, \bar{W})$ having one endnode in W and the other in $\bar{W}$ is called a cut. Futhermore, let $K(W, \bar{W})$ be the set of commodities having one endnode in W and the other in $\bar{W}$.

Then for any subset W with $K(W, \bar{W}) \neq \emptyset$ the following inequality, called a **cutset inequality**, must hold in order for a given capacity vector y^* to be feasible for an arbitrary demand vector d.

$$\sum_{e\in E(W,\bar{W})} y_e^* \geq \sum_{k\in K(W,\bar{W})} d_k \tag{4.27}$$

Cutset inequalities are special types of metric inequalities. The metric inequality (4.18) can be transformed into the cutset inequality (4.27) by setting $\mu_e = 1$ for all $e \in E(W, \bar{W})$ and $\mu_e = 0$ for all $e \in E \setminus E(W, \bar{W})$. The shortest path lengths π_k for all $k \in K(W, \bar{W})$ will consequently all be equal to one. A cutset inequality simply states that the total demand across a cut should not exceed the total capacity available across the cut. In terms of multiple time periods this means the cut capacity should be sufficient for all of the demand vectors, non-simultaneously. That is, for an arbitrary subset of nodes W, the following set of inequalities should hold for a given capacity vector y^* to allow feasible routing of the set of demand vectors $D = \{d^1, d^2, \ldots, d^{|\mathcal{T}|}\}$

$$\sum_{e\in E(W,\bar{W})} y_e^* \geq \sum_{k\in K(W,\bar{W})} d_k^t \;\forall\, t \in \mathcal{T}$$

which is equivalent to saying

$$\sum_{e\in E(W,\bar{W})} y_e^* \geq \max_{t\in\mathcal{T}} \left\{ \sum_{k\in K(W,\bar{W})} d_k^t \right\} \tag{4.28}$$

4.3.3 Characterising feasible capacities for static routing

Consider the flow polyhedron $R(y, D)$ for the static routing model with a set of demand vectors $D = \{d^1, d^2, \ldots, d^{|\mathcal{T}|}\}$, and associate with the constraint set (4.11) and (4.12) the dual variables $\pi \in \mathbb{R}^{|K|}$ and $\mu \in \mathbb{R}_+^{|E|\times|\mathcal{T}|}$ respectively.

Proposition 4.3. *For a given capacity vector y^* the polyhedron $R(y^*, D)$ will not be empty iff*

$$\sum_{t\in\mathcal{T}}\sum_{e\in E} y_e^*\mu_e^t \geq \sum_{k\in K}\pi_k \tag{4.29}$$

for all $\mu_e^t \geq 0$, $e \in E$ and $t \in \mathcal{T}$, and where $\pi_k = \min_{p\in\mathcal{P}(k)}\left\{\sum_{t\in\mathcal{T}}\sum_{e\in E(p)} d_k^t\mu_e^t\right\}$ is the length of the shortest path for each commodity $k \in K$.

Proof. Consider the following linear program that minimises the shortfall in capacity α^t for all $t \in \mathcal{T}$ within the flow polyhedron $R(y^*, D)$ for a given capacity vector y^*:

$$\min \sum_{t\in\mathcal{T}} \alpha^t$$

$$\text{s.t.} \sum_{p\in\mathcal{P}(k)} r_p = 1 \quad \forall\, k \in K \tag{4.30}$$

$$\sum_{k\in K}\sum_{p\in\mathcal{P}(k,e)} r_p d_k^t - \alpha^t \leq y_e^* \quad \forall\, e \in E,\ \forall\, t \in \mathcal{T} \tag{4.31}$$

To acknowledge feasibility of a capacity vector y^* the optimal solution should yield $\alpha^t = 0$ for all $t \in \mathcal{T}$. By associating the variables $\pi \in \mathbb{R}^{|K|}$ and $\mu \in \mathbb{R}_+^{|E|\times|\mathcal{T}|}$ with the constraints (4.30) and (4.31) respectively, the corresponding dual is obtained:

$$\max \sum_{k\in K}\pi_k - \sum_{t\in\mathcal{T}}\sum_{e\in E} y_e^*\mu_e^t$$

$$\text{s.t.} \sum_{e\in E}\mu_e^t = 1 \quad \forall\, t \in \mathcal{T} \tag{4.32}$$

$$\pi_k - \sum_{t\in\mathcal{T}}\sum_{e\in E(p)} d_k^t\mu_e^t \leq 0 \quad \forall\, p \in \mathcal{P}(k),\ \forall\, k \in K \tag{4.33}$$

For a positive objective value that signals insufficient capacity, the following valid inequality can be added to the capacitated network design problem to cut off the infeasible capacity vector y^*:

$$\sum_{k\in K}\pi_k - \sum_{t\in\mathcal{T}}\sum_{e\in E} y_e\mu_e^t \leq 0 \tag{4.34}$$

with π_k for all $k \in K$, and μ_e^t for all $e \in E$, $t \in \mathcal{T}$, the solution values obtained by solving the above dual linear programming problem.

□

By associating the dual variables $\gamma \in \mathbb{R}_+^{|K|\times|S|}$ with the diversification constraint set (4.16), the results from the following theorem allow us to derive metric inequalities for the case where survivability is considered.

Proposition 4.4. *For a given capacity vector y^* the polyhedron $R(y^*, D, \delta)$ will not be empty iff*

$$\sum_{e \in E} y_e^* \mu_e^t \geq \sum_{k \in K} \left(\pi_k + \delta_k \sum_{s \in S} \gamma_{ks} \right) \tag{4.35}$$

for all $\mu_e^t \geq 0$, $e \in E$ and $t \in T$, and for all $\gamma_{ks} \geq 0$, $s \in S$ and $k \in K$, and where $\pi_k = \min_{p \in \mathcal{P}(k)} \left\{ \sum_{t \in T} \sum_{e \in E(p)} d_k^t \mu_e^t - \sum_{s \in S(p)} \gamma_{ks} \right\}$ is the length of the shortest path for each commodity $k \in K$.

Proof. The proof is analogous to that of Proposition 4.3. The proof is sketched informally as part of the discussion on separation in Section 5.4.2. □

Chapter 5

Algorithmic Approach

The survivable network design problem is a mixed integer programming problem (MIP). Numerous approaches exist for solving MIPs and based on solution properties, these approaches can be placed broadly into two categories. Firstly, heuristic approaches such as simulated annealing, tabu search techniques, and genetic algorithms may provide feasible solutions in reasonable time but are, however, unable to provide a certification on optimality. Exact approaches, on the other hand, guarantee optimality on completion. Furthermore, exact approaches based, for instance, on the **branch-and-cut** framework, have the property of providing a bound on potential optimal solutions that might still exist in the event of premature termination. This is very useful from a network planner's perspective since this enables one to limit computing times by compromising on optimality. For instance, a network planner may be satisfied with a feasible solution that is within 1% from the optimal solution, if it means that running time could be reduced to a couple of minutes instead of the algorithm running for several hours. In this thesis an exact approach is followed based on the branch-and-cut framework.

Irrespective of the solution approach being followed, it is worthwhile to explore ways of reducing the volume of data in an attempt to reduce computational complexity. By representing uncertainty in traffic requirements as a set of non-simultaneous traffic demand vectors, the problem can easily become unmanageable, especially if we consider a large set of demand vectors. A problem reduction approach is therefore suggested for discarding demand vectors from the problem data, without changing the feasible region of the survivable network design problem.

In the subsequent sections the proposed problem reduction approach will be described, followed by the solution approach based on the branch-and-cut framework. In order to improve readability, all acronyms or abbreviations used in referencing specific problems, procedures, or heuristics etc., are listed in Table A.2 as part of Appendix A.

5.1 Problem Reduction

The data requirements for empirical experiments with the survivable network design problem with demand uncertainty, are mainly obtained from two sources. Firstly, multiple demand vectors could be measured from an existing network over multiple time periods. The number of demand vectors obtained from measurements are dependent on available technology. If the network architecture permits, demand vectors could be measured at intervals of five minutes or even less. Secondly, demand data could be generated through simulation. If enough information is available about the traffic distributions on a network, it is even plausible to randomly generate multiple demand vectors.

In both cases, whether data has been collected through measurements or generated randomly, the outcome could be a data set containing thousands of multiple demand vectors. Network dimensioning with multiple demand vectors will consequently entail designing a minimum cost network with feasible routings that satisfy the traffic requirements for *all* of the multiple demand vectors.

It is clear that demand data with a magnitude ranging into thousands of demand vectors will render the survivable network design problem computationally infeasible. The network design problem with one traffic demand vector is already considered to be a hard problem (see [snd05]). A problem reduction approach is therefore necessary that has the ability to identify traffic demand vectors which may be removed from the problem description without changing the cost of an optimal network design.

5.1.1 Domination among traffic demand vectors

The decision to include or exclude a demand vector in the computation of a network design requires the notion of dominance. The basic idea presented by Oriolo [Ori08] for determining whether a demand vector d^1 **dominates** a demand vector d^2, is to determine whether a feasible routing can be found for d^2 by viewing d^1 as a capacity vector. Only the two demand vector case has been investigated by Oriolo with extensions towards integral and non-splittable routing models. No computational results are provided.

In the paper by Zhang and Ge [ZG06] a problem reduction approach based on a clustering algorithm is proposed for finding a critical set of traffic demand vectors. A static routing environment is assumed and a proof is provided to show that their approach is NP-hard. Several distance measures are suggested as part of the clustering objective, and computational results are provided based on the AT&T's North American backbone network. It is reported that a total of 3048 demand vectors have been used in their study, with each vector containing over 400 commodities.

The contributions presented in this thesis on domination are based on the work by Oriolo [Ori08], and part of it were presented in Terblanche et al. [TWH06].

In order to simplify notation, we shall assume for the remainder of this section that both the demand graph and supply graph are complete, that is $K = E = \mathcal{E}$. It is therefore reasonable to allow the notation d_e that denotes the demand requirement between the pair of nodes that is joined with the edge $e \in E$, since for this edge there will exist one, and only one, commodity $k \in K$. This notation can be used without loss of generality since we let $d_e = 0$ if in the normal context of the demand graph, no demand exists between the node pairs that are joined by the edge e. Another property of using complete demand and supply graphs, is that the dimensions of the demand vectors and capacity vectors will match.

Consider the polyhedron $F(y, d)$ that defines the feasible routings for the single demand vector case. If $F(y, d) \neq \emptyset$ we say the capacity vector y **supports** the demand vector d. The set $\mathcal{Y}(d)$ denotes the set of all capacity vectors supporting a demand vector d. Similarly, if $F(y, D) \neq \emptyset$, the capacity vector y supports the set of demand vectors D. The set $\mathcal{Y}(D)$ denotes the set of all capacity vectors supporting the set of demand vectors D.

For the static routing case, if $R(y, D) \neq \emptyset$, the capacity vector y supports the set of demand vectors D with the restriction that all demand vectors are routed according to the same routing $r \in R(y, D)$. The set $\mathcal{YR}(D)$ denotes the set of all pairs y and r, where y supports D with a feasible routing $r \in R(y, D)$.

Definition 5.1 ([Ori08]). *A demand vector d^1 dominates d^2 iff $\mathcal{Y}(d^1) \subseteq \mathcal{Y}(d^2)$.*

There is, however, a distinction to be made between dynamic and static routing within the context of domination. The preceding definition of domination imposes no restriction on routing and, therefore, implies that even if the same capacity vector y can support both d^1 and d^2, the two demand vectors may be routed independently.

In order to extend the notion of domination for the case of static routing we say that if every capacity vector y that supports a demand vector d^1 with routing r^1, also supports d^2 with routing r^2, and if in addition $r^1 = r^2$, then d^1 **totally dominates** d^2.

Definition 5.2 ([Ori08]). *A demand vector d^1 totally dominates d^2 iff $\mathcal{YR}(d^1) \subseteq \mathcal{YR}(d^2)$.*

In the subsequent sections the basic theory of domination w.r.t. dynamic and static routing is extended to cater for multiple demand vectors.

5.1.2 Dominance checking for dynamic routing

The main result from the work by Oriolo [Ori08] established that a demand vector d^1 will dominate a demand vector d^2, if d^1 viewed as a capacity vector is able to support d^2. This is formally stated in the following theorem:

Theorem 5.1 ([Ori08]). *A demand vector d^1 dominates d^2 iff $F(d^1, d^2) \neq \emptyset$.*

The notation $F(d^1, d^2)$ is valid within the context of this discussion since we assume a complete graph such that a demand vector $d \in \mathbb{R}_+^{|\mathcal{K}|}$ and a capacity vector $y \in \mathbb{R}_+^{|E|}$ have the same dimensions.

Proposition 5.1. *Let $y \in \mathbb{R}_+^{|E|}$ be a capacity vector that supports both the demand vectors d^1 and d^2. Then y supports the convex combination $d = \lambda d^1 + (1-\lambda)d^2$, with $0 \leq \lambda \leq 1$.*

Proof. For each commodity $k \in K$, let $f_p = \lambda f_p^1 + (1-\lambda) f_p^2$ with $p \in \mathcal{P}(k)$, where $f^1 \in F(y, d^1)$ and $f^2 \in F(y, d^2)$ then:

$$\begin{aligned}
\sum_{p \in \mathcal{P}(k)} f_p &= \sum_{p \in \mathcal{P}(k)} \left(\lambda f_p^1 + (1-\lambda) f_p^2 \right) \\
&= \lambda \sum_{p \in \mathcal{P}(k)} f_p^1 + (1-\lambda) \sum_{p \in \mathcal{P}(k)} f_p^2 \\
&= \lambda d_k^1 + (1-\lambda) d_k^2 \\
&= d_k
\end{aligned}$$

This satisfies (4.8) of $F(y, d)$. Furthermore, for each edge $e \in E$

$$\begin{aligned}
\sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} f_p &= \sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} \left(\lambda f_p^1 + (1-\lambda) f_p^2 \right) \\
&= \lambda \sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} f_p^1 \\
&\quad + (1-\lambda) \sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} f_p^2 \\
&\leq \lambda y_e + (1-\lambda) y_e \\
&= y_e
\end{aligned}$$

This completes the proof since (4.9) of $F(y, d)$ is also satisfied. □

Proposition 5.1 can be generalized such that the result is true for more than two demand vectors.

Definition 5.3. *A set of demand vectors D dominates a demand vector d iff $\mathcal{Y}(D) \subseteq \mathcal{Y}(d)$.*

Corollary 5.1. *Let $d^c = \sum_{t \in T} \lambda^t d^t$ with $\sum_{t \in T} \lambda^t = 1$, $\lambda^t \geq 0$, dominate the demand vector d. Then the set $D = \{d^1, d^2, \ldots, d^{|T|}\}$ dominates d.*

Proof. According to Proposition 5.1 each capacity vector y that supports D will also support the convex combination d^c. Furthermore, if d^c dominates d then $\mathcal{Y}(d^c) \subseteq \mathcal{Y}(d)$, therefore $\mathcal{Y}(D) \subseteq \mathcal{Y}(d^c) \subseteq \mathcal{Y}(d)$. □

The implication of Corollary 5.1 is that if we can find a demand vector $d^c \in conv(D)$ that dominates d, then d can be discarded in the network design process. Assume that a set of demand vectors $D = \{d^1, d^2, \ldots, d^{|\mathcal{T}|}\}$ is provided, and that the objective is to determine whether D dominates d. Then, by letting $f \in \mathbb{R}_+^{|\mathcal{P}|}$ be the flow variables, $\alpha \in \mathbb{R}_+$ the shortfall in capacity, $d^c \in \mathbb{R}_+^{|E|}$ a demand vector to be written as a convex combination of the set of demand vectors D, and $\lambda \in \mathbb{R}_+^{|\mathcal{T}|}$ variables to be used for expressing d^c as a convex combination of D, the following linear program, called the Dynamic Routing Dominance Checker (DRDC), is obtained:

$$\text{(DRDC)}: \quad \min \alpha \tag{5.1}$$

$$\text{s.t.} \sum_{p \in \mathcal{P}(k)} f_p = d_k \quad \forall\, k \in K \tag{5.2}$$

$$\sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} f_p - \alpha - d_e^c \leq 0 \quad \forall\, e \in E \tag{5.3}$$

$$d_e^c - \sum_{t \in \mathcal{T}} \lambda^t d_e^t = 0 \quad \forall\, e \in E \tag{5.4}$$

$$\sum_{t \in \mathcal{T}} \lambda^t = 1 \tag{5.5}$$

The objective function (5.1) together with the constraint sets (5.2) and (5.3) correspond to a multicommodity flow problem having the objective of finding a feasible routing such that d^c supports d. The constraint sets (5.4) and (5.5) are responsible for expressing d^c as a point in $conv(D)$.

If the optimum yields $\alpha = 0$, it can be concluded that a point in $conv(D)$ could be found that provides enough "capacity" that allows a feasible routing of d. Therefore, an optimum point given by the solution d^c will dominate d. If on the other hand, $\alpha > 0$, it is determined that no point in $conv(D)$ exists that dominates d.

5.1.3 Domination checking for static routing

Per definition a demand vector d^1 dominates a demand vector d^2 within a static routing framework, if all capacity vectors that support d^1 also support d^2 with the restriction that d^1 and d^2 should be routed according to the same routing. Clearly, if every component of d^2 is less than or equal to the corresponding component of d^1, then the same routing that enables a capacity vector y to support d^1 will also be feasible for d^2.

Theorem 5.2 ([Ori08]). *A demand vector d^1 totally dominates a demand vector d^2 iff $d_e^1 \geq d_e^2$ for all $e \in E$.*

Proposition 5.2. *Let $y \in \mathbb{R}_+^{|E|}$ be a capacity vector that supports both the demand vectors d^1 and d^2 with routing $r \in R(y, \{d^1, d^2\})$. Then y supports the convex combination $d = \lambda d^1 + (1-\lambda)d^2$, $0 \leq \lambda \leq 1$ with routing r.*

Proof. For each $k \in K$ let $f_p = r_p d_k$ with $p \in \mathcal{P}(k)$, then

$$\sum_{p \in \mathcal{P}(k)} f_p = \sum_{p \in \mathcal{P}(k)} r_p d_k = d_k$$

since $\sum_{p \in \mathcal{P}(k)} r_p = 1$ according to (4.14) of $R(y, d)$. This satisfies (4.8) of $F(y, d)$. Furthermore, for each edge $e \in E$

$$\begin{aligned}
\sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} f_p &= \sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} r_p d_k \\
&= \sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} \left(\lambda r_p d_k^1 + (1-\lambda) r_p d_k^2 \right) \\
&= \lambda \sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} r_p d_k^1 \\
&\quad + (1-\lambda) \sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} r_p d_k^2 \\
&\leq \lambda y_e + (1-\lambda) y_e \\
&= y_e
\end{aligned}$$

This completes the proof since (4.9) of $F(y, d)$ is also satisfied. □

Proposition 5.2 can be generalized such that the result is true for more than two demand vectors.

Definition 5.4. *A set of demand vectors $D = \{d^1, d^2, \ldots, d^{|\mathcal{T}|}\}$ totally dominates a demand vector d iff $\cap_{t \in \mathcal{T}} \mathcal{YR}(d^t) \subseteq \mathcal{YR}(d)$.*

Corollary 5.2. *Let $d^c = \sum_{t \in \mathcal{T}} \lambda^t d^t$ with $\sum_{t \in \mathcal{T}} \lambda^t = 1$, $\lambda^t \geq 0$, for all $t \in \mathcal{T}$, totally dominates the demand vector d. Then the set $D = \{d^1, d^2, \ldots, d^{|\mathcal{T}|}\}$ totally dominates d.*

Proof. According to Proposition 5.2 each capacity vector y that supports D with a routing r will also support the convex combination d^c with routing r. Furthermore, if d^c totally dominates d, then $\mathcal{YR}(d^c) \subseteq \mathcal{YR}(d)$, therefore $\cap_{t \in \mathcal{T}} \mathcal{YR}(d^t) \subseteq \mathcal{YR}(d^c) \subseteq \mathcal{YR}(d)$. □

The implication of Corollary 5.2 is that if we can find values for $\lambda^t \geq 0$, for all $t \in \mathcal{T}$, such that

$$d_e \leq \sum_{t \in \mathcal{T}} \lambda^t d_e^t \;\forall\; e \in E$$

with $\sum_{t \in \mathcal{T}} \lambda^t = 1$, then d can be discarded in the network design process since the set D totally dominates d. The following linear program, called the Static Routing Dominance Checker (SRDC),

solves the problem of finding values for $\lambda^t \geq 0$ by introducing a slack variable $\alpha \in \mathbb{R}_+$ that needs to be minimized:

$$\begin{aligned} & \min \ \alpha \\ \text{s.t.} \ & \sum_{t \in \mathcal{T}} \lambda^t d_e^t + \alpha \geq d_e \quad \forall\, e \in E \\ & \sum_{t \in \mathcal{T}} \lambda^t = 1 \\ & \lambda^t \geq 0 \quad \forall\, t \in \mathcal{T} \end{aligned}$$

The objective function is to minimize the error found in writing d as a convex combination of the demand vectors D. If the optimum to the linear program yields $\alpha = 0$, then a point was found in $conv(D)$ that totally dominates d, and therefore d can be discarded. If $\alpha > 0$, then no dominating point in $conv(D)$ exists and d will have to be included in the network design process.

5.1.4 Undetected domination

The work by Oriolo [Ori08] provides an easy mechanism for determining whether one demand vector dominates another. It should, however, be noted that the underlying assumption of Theorem 5.1 is that the supply graph is complete. Therefore, it is possible that for a sparse supply graph the result of Theorem 5.1 might incorrectly indicate that a demand vector d^1 does not dominate d^2, where in fact d^1 does indeed dominate d^2. Consider the following simple example with the supply graph depicted in Figure 5.1.a). Let d^1 be a demand vector having a demand of 2 units between n_1 and n_2, and having a demand of 2 units between n_1 and n_3. The demand graph for d^1 is depicted in Figure 5.1.b). By considering the supply graph, it is evident that any feasible solution supporting the demand vector d^1 would require that at least 4 units of capacity should be installed on the edge connecting n_1 and n_2, and at least 2 units of capacity should be installed on the edge connecting n_2 and n_3. This will allow the 2 units of traffic between n_1 and n_3 to be routed via n_2. The resulting capacity on the network will, evidently, also support the demand vector d^2 depicted in Figure 5.1.c) having a demand of 3 units between n_1 and n_2. In fact, any capacity on the network that will support d^1 will also support d^2, hence d^1 dominates d^2 per definition. However, according to Theorem 5.1, d^1 does not dominate d^2 since $F(d^1, d^2) = \emptyset$.

In the case of domination among multiple demand vectors there is also the possibility that domination might go undetected. Even though if we cannot conclude that the set D dominates d by applying the results from Corollary 5.1, the set of demand vectors D might still dominate d. Consider the following network $G(\{n_1, n_2, n_3\}, \{e_1, e_2, e_3\})$ with the commodity set $\mathcal{K} = \{1, 2, 3\} = \{\{n_1, n_2\}, \{n_2, n_3\}, \{n_1, n_3\}\}$. Now, according to Corollary 5.1, it cannot be confirmed that the set of demand vectors $d^1 = [1, 0, 0]$ and $d^2 = [0, 1, 0]$ does dominate the demand vector $d^3 = [0, 0, 1]$ since we cannot find a convex combination of d^1 and d^2 that dominates d^3.

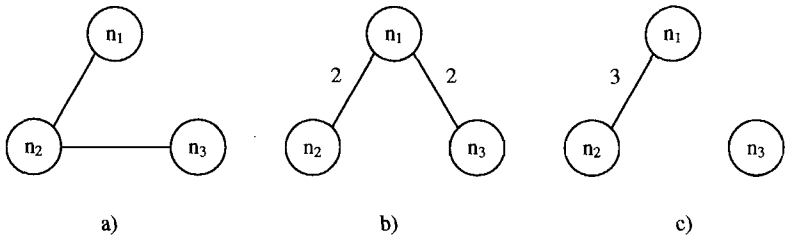

Figure 5.1: a) Supply graph. b) Demand graph for d^1. c) Demand graph for d^2.

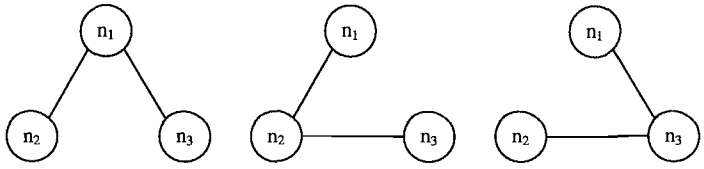

Figure 5.2: All possible solutions for routing d^1, d^2, and d^3 non-simultaneously

However, consider Figure 5.2 which shows 3 different solutions with all edges in the respective graphs having a capacity of one. Let the capacity vectors corresponding to the 3 solutions be denoted by the set $\tilde{\mathcal{Y}}(\{d^1, d^2\}) \subseteq \mathcal{Y}(\{d^1, d^2\})$, with $\mathcal{Y}(\{d^1, d^2\})$ the set of all capacity vectors supporting d^1 and d^2. For each of the 3 solutions one unit of demand can be routed from node 1 to node 3, indicating that demand vector d^3 is indeed being dominated by demand vectors d^1 and d^2 since $\tilde{\mathcal{Y}}(\{d^1, d^2\})$ also supports d^3. For this argument we only need to consider $\tilde{\mathcal{Y}}(\{d^1, d^2\})$ and not $\mathcal{Y}(\{d^1, d^2\})$, since $\tilde{\mathcal{Y}}(\{d^1, d^2\})$ contains all combinations of capacity vectors with minimum capacity that will support d^1 and d^2.

5.1.5 Dominance checking algorithms

The overall procedure for doing dominance checking is the same for both the static and the dynamic case. Let DOM_CHECK(d) denote the sub-procedure that solves either the DRDC or the SRDC problem for a demand vector d depending on whether dynamic or static routing is considered. The optimal objective value obtained by calling the sub-procedure is z^* and the vector λ^* is an optimal solution for the variables λ^t, for all $t \in \mathcal{T}$ which is present in both the problems DRDC and SRDC. If it is found that a demand vector d is being dominated, it is added to the set $\mathcal{A}$. Initially this set is empty and will eventually contain the set of dominated demand vectors. The main algorithm that calls the sub-procedure DOM_CHECK(d) is listed in Algorithm 1.

The implementation of the dominance checker that assumes dynamic routing is, however, more complex compared to the static routing case since a column generation approach is followed in

Algorithm 1 Dominance Checking Algorithm

$\mathcal{A} = \emptyset$

for all $t \in \mathcal{T}$ **do**

  for all $k \in \mathcal{K}$ **do**

    Set $d_k = d_k^t$

  end for

  Set variable bounds $\lambda^t = 0$

  Call sub-procedure DOM_CHECK(d)

  if $z^* < 0$ **then**

    Add d^t to the set $\mathcal{A}$.

  else

    Set variable bounds $\lambda^t \geq 0$

  end if

end for

order to manage the exponential number of path variables in the linear program.

Let the dual variables $\pi \in \mathbb{R}^{|E|}$, $\mu \in \mathbb{R}_+^{|E|}$, $v \in \mathbb{R}^{|E|}$, and $w \in \mathbb{R}$ be associated with the constraints (5.2), (5.3), (5.4) and (5.5) in the formulation of DRDC, respectively. Then the dual of the DRDC problem is the following:

$$\max\ w + \sum_{k \in K} d_k \pi_K$$

$$\text{s.t.} \quad \sum_{e \in E} \mu_e = 1 \tag{5.6}$$

$$\sum_{e \in E} \mu_e + \sum_{e \in E} v_e \leq 0 \tag{5.7}$$

$$w - \sum_{e \in E} d_e^t v_e \leq 0 \quad \forall\, t \in \mathcal{T}, \tag{5.8}$$

$$\pi_k - \sum_{e \in E(p)} \mu_e \leq 0 \quad \forall\, p \in \mathcal{P}(k), \quad \forall\, k \in K \tag{5.9}$$

The constraint set (5.9) of the dual problem states that each variable π_k, for a commodity $k \in K$, is less or equal to the minimum of the expression $\sum_{e \in E(p)} \mu_e$ for all paths $p \in \mathcal{P}(k)$. This is equivalent to saying that for an optimal solution to the DRDC problem, each π_k for a commodity $k \in K$ is the length of the shortest path among all the paths in the set $\mathcal{P}(k)$. However, if not all path variables have been included in the formulation of the DRDC problem, there might exist a shorter path that has a length less than the value of π_k. Therefore, the column generation approach entails the calculation of a shortest path using the the edge weights μ_e for all $e \in E$, and whenever it is found that the length of this shortest path is less than any of the π_k's, this new shortest path is added to the problem.

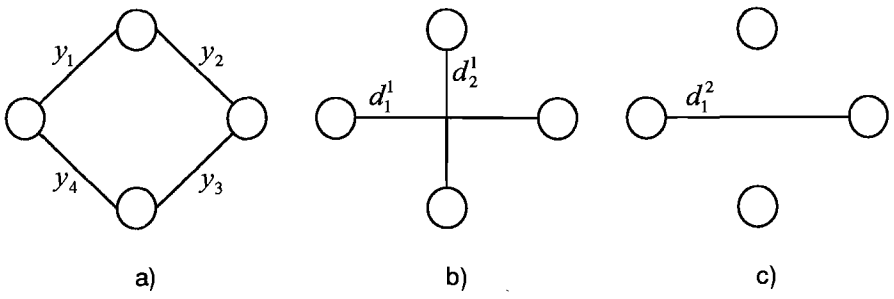

Figure 5.3: a) Supply graph for y. b) Demand graph for d^1. c) Demand graph for d^2

5.1.6 Extensions towards survivability

Extending the theory of domination to include survivability requirements poses some challenges. In fact, extending Theorem 5.1 to the case where we consider the polyhedron $F(y, d, \delta)$ is not possible due to the different meaning that failure states have within the context of domination. Consider the four-node example depicted in Figure 5.3. For this example we consider a capacity vector $y = \{y_1, y_2, y_3, y_4\}$ and two demand vectors, $d^1 = \{d_1^1, d_2^1\}$ and $d^2 = \{d_1^2, d_2^2\}$. Per definition d^1 clearly dominates d^2 since $\mathcal{Y}(d^1) \subseteq \mathcal{Y}(d^2)$. Furthermore, the demands for both d^1 and d^2 can be routed along 2 disjoint paths on the supply graph, therefore implying that, d^1 dominates d^2 by taking into account a diversification requirement of $\delta = 0.5$. However, if we want to extend Theorem 5.1 to take into account survivability, failure states will need to be defined that comprise the edges in Figure 5.3.b). That is, the demand vector d^1 is viewed as the capacity vector. Now, in order to satisfy a diversification requirement of $\delta = 0.5$, the demand vector d_1^2 would need to be split into two equal parts and be routed along two disjoint routes on the edges of the demand graph in Figure 5.3.b). This is clearly impossible with the result that $F(d^1, d^2, 0.5) = \emptyset$ even though we have shown that d^1 dominates d^2.

There is, however, the possibility to extend the theory of domination to include survivability with respect to static routing models. Let $\mathcal{YR}(D, \delta)$ denotes the set of all pairs y and r, where y supports D with a feasible routing $r \in R(y, D, \delta)$.

Definition 5.5. *A demand vector d^1 with diversification requirements δ, dominates d^2 with diversification requirements δ, iff $\mathcal{YR}(d^1, \delta) \subseteq \mathcal{YR}(d^2, \delta)$.*

Theorem 5.3. *A demand vector d^1 with diversification requirements δ totally dominates a demand vector d^2 with diversification requirements δ iff $d_k^2 \leq d_k^1$ for all $k \in K$.*

Proof. For the forward case we know that each capacity vector y that supports d^1 with a routing r, will also support d^2 with the same routing. Now if we assume that $d_k^2 > d_k^1$ for some $k \in K$, then

for the case where y supports d^1 without any spare capacity, we have that for each edge $e \in E$

$$\begin{aligned} y_e &= \sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} r_p d_k^1 \\ &< \sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} r_p d_k^2 \end{aligned}$$

which is a contradiction. Therefore, the inequality $d_k^2 \leq d_k^1$ for all $k \in K$ must hold.

For the converse case let $d_k^2 \leq d_k^1$ for all $k \in K$. Then for each edge $e \in E$

$$\begin{aligned} y_e &\geq \sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} r_p d_k^1 \\ &\geq \sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} r_p d_k^2 \end{aligned} \tag{5.10}$$

□

Proposition 5.3. *Let $y \in \mathbb{R}_+^{|E|}$ be a capacity vector that supports both the demand vectors d^1 and d^2, with routing $r \in R(y, \{d^1, d^2\}, \delta)$ and common diversification parameter δ. Then y supports the convex combination $d = \lambda d^1 + (1-\lambda)d^2$, $0 \leq \lambda \leq 1$ with routing r and diversification δ.*

Proof. For each $k \in K$ let $f_p = r_p d_k$ with $p \in \mathcal{P}(k)$, then

$$\sum_{p \in \mathcal{P}(k)} f_p = \sum_{p \in \mathcal{P}(k)} r_p d_k = d_k$$

since $\sum_{p \in \mathcal{P}(k)} r_p = 1$ according to (4.14) of $R(y, d)$. This satisfies (4.8) of $F(y, d, \delta)$. Furthermore, for each edge $e \in E$

$$\begin{aligned} \sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} f_p &= \sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} r_p d_k \\ &= \sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} \left(\lambda r_p d_k^1 + (1-\lambda) r_p d_k^2 \right) \\ &= \lambda \sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} r_p d_k^1 \\ &\quad + (1-\lambda) \sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} r_p d_k^2 \\ &\leq \lambda y_e + (1-\lambda) y_e \\ &= y_e \end{aligned}$$

This satisfies (4.9) of $F(y, d, \delta)$. Furthermore, for each $k \in K$ and each $s \in S$

$$\begin{aligned}
\sum_{p \in \mathcal{P}(k,s)} f_p &= \sum_{p \in \mathcal{P}(k,s)} r_p d_k \\
&= \sum_{p \in \mathcal{P}(k,s)} \left(\lambda r_p d_k^1 + (1-\lambda) r_p d_k^2 \right) \\
&= \lambda \sum_{p \in \mathcal{P}(k,s)} r_p d_k^1 + (1-\lambda) \sum_{p \in \mathcal{P}(k,s)} r_p d_k^2 \\
&\leq \lambda \delta_k d_k^1 + (1-\lambda) \delta_k d_k^2 \\
&= \delta_k d_k
\end{aligned}$$

This completes the proof since (4.10) of $F(y, d, \delta)$ is satisfied.

□

Proposition 5.3 can be generalised such that the result is true for more than two demand vectors.

Definition 5.6. *A set of demand vectors $D = \{d^1, d^2, \ldots, d^{|\mathcal{T}|}\}$ totally dominates a demand vector d with a common diversification vector δ iff $\cap_{t \in \mathcal{T}} \mathcal{YR}(d^t, \delta) \subseteq \mathcal{YR}(d, \delta)$.*

Corollary 5.3. *Let $d^c = \sum_{t \in \mathcal{T}} \lambda^t d^t$ with $\sum_{t \in \mathcal{T}} \lambda^t = 1$, $\lambda^t \geq 0$ totally dominate the demand vector d with a common diversification vector δ. Then the set $D = \{d^1, d^2, \ldots, d^{|\mathcal{T}|}\}$ totally dominates d with δ the common diversification vector.*

Proof. According to Proposition 5.3 each capacity vector y that supports D with a routing r will also support the convex combination d^c with routing r taking δ as the common diversification vector. Furthermore, if d^c totally dominates d with common diversification vector δ then $\mathcal{YR}(d^c, \delta) \subseteq \mathcal{YR}(d, \delta)$, therefore $\cap_{t \in \mathcal{T}} \mathcal{YR}(d^t, \delta) \subseteq \mathcal{YR}(d^c, \delta) \subseteq \mathcal{YR}(d, \delta)$. □

The implication of Corollary 5.3 is that if we can find values for $\lambda^t \geq 0$, for all $t \in \mathcal{T}$ such that

$$d_k \leq \sum_{t \in \mathcal{T}} \lambda^t d_k^t \; \forall \; k \in K$$

with $\sum_{t \in \mathcal{T}} \lambda^t = 1$, then d can be discarded in the network design process provided that the same diversification vector δ is associated with all demand vectors. Therefore, it suffices to employ the Static Routing Dominance Checker (SRDC) as described in Section 5.1.3 in the case where survivability is considered.

5.1.7 Domination among multiple demand vectors - an alternative

In the preceding section it was shown that the suggested approach for doing domination checking is easily extended to cater for survivability, but only in the case of static routing. In order to incorporate survivability into domination checking for dynamic routing, we have to go back to the

basic meaning of domination within the context of survivability. Recall that the set $\mathcal{Y}(d,\delta)$ denotes all capacity vectors y for which $F(y,d,\delta) \neq \emptyset$, and $\mathcal{Y}(D,\delta)$ denotes all capacity vectors y for which $F(y,D,\delta) \neq \emptyset$

Definition 5.7. *A set of demand vectors D dominates a demand vector d with a common diversification vector δ iff $\mathcal{Y}(D,\delta) \subseteq \mathcal{Y}(d,\delta)$.*

The approach proposed here for doing dominance checking is a direct application of the above definition. If it is possible to identify a capacity vector y that supports D but that does not support d, then we can conclude that D definitely does **not** dominate d. That is, the approach attempts to find a capacity vector $y \in \mathcal{Y}(D,\delta) \backslash \mathcal{Y}(d,\delta)$ such that $\{y\} \cap \mathcal{Y}(d,\delta) = \emptyset$. If such a y cannot be found, then it can be concluded that D dominates d.

In addition to having the ability to cater for survivability, this new approach for doing dominance checking will be capable of addressing the issues of undetected domination as discussed in Section 5.1.4.

In order to facilitate the remainder of the discussion, it is necessary to formulate a simple edge flow survivable network design problem with continuous capacity variables.

Edge flow survivable network design problem

Let the variable $f^t_{ijk} \in \mathbb{R}_+$ denote the flow of traffic from node $i \in V$ to node $j \in V$ for a commodity $k \in K$ and for a demand vector d^t, $t \in \mathcal{T}$. Furthermore, let $\overrightarrow{n}(e) = i$ and $\overleftarrow{n}(e) = j$ be functions that provide the source node and target node respectively, for an edge $e \in E$. Using this we are able to make use of the notation $f^t_{\overrightarrow{n}(e)\overleftarrow{n}(e)k}$ representing the flow of traffic from the source node $i \in V$ to the target node $j \in V$ for an edge $e \in E$. The concept of failure states is used to facilitate the modelling of survivability. For this problem only single edge failures are considered and therefore the set of failure states is exactly the set of edges E. In order to model the constraints for the edge flow network design problem, the following function is required:

$$\Phi(i,t) = \begin{cases} d^t_k & \text{if } i \text{ is the source node for commodity } k \in K \\ -d^t_k & \text{if } i \text{ is the destination node for commodity } k \in K \\ 0 & \text{otherwise} \end{cases} \tag{5.11}$$

For a cost vector $c \in \mathbb{R}_+^{|E|}$, and a set of demand vectors D, the simple edge flow network design problem SIMPLE_SNDP(c,D,δ) is defined as:

$$\min \quad \sum_{e \in E} c_e y_e$$

$$\text{s.t.} \sum_{j \in V} f^t_{ijk} - \sum_{j \in V} f^t_{jik} = \Phi(i,t) \quad \forall\, i \in V \quad \forall\, k \in K \quad \forall\, t \in \mathcal{T} \tag{5.12}$$

$$\sum_{k \in K} \left(f^t_{\overrightarrow{n}(e)\overleftarrow{n}(e)k} + f^t_{\overleftarrow{n}(e)\overrightarrow{n}(e)k} \right) - y_e \leq 0 \quad \forall\, e \in E \quad \forall\, t \in \mathcal{T} \tag{5.13}$$

$$f^t_{\overrightarrow{n}(e)\overleftarrow{n}(e)k} + f^t_{\overleftarrow{n}(e)\overrightarrow{n}(e)k} \leq \delta_k d^t_k \quad \forall\, k \in K \quad \forall\, e \in E \quad \forall\, t \in \mathcal{T} \tag{5.14}$$

Constraints (5.12) are the node balancing constraints, constraints (5.13) are the capacity constraints, and (5.14) are the diversification constraints.

Obtaining a projection of the capacity variables

The approach suggested for determining whether the set of demand vectors D dominates a demand vector d, requires the construction of a polyhedron P_y that is a projection of SIMPLE_SNDP(c,{d},δ) onto the space of the variables y_e for all $e \in E$. Let A and B be the constraint coefficients associated with the variables $y \in \mathbb{R}_+^{|E|}$ and $f \in \mathbb{R}_+^{2\times|K|\times|E|\times|\mathcal{T}|}$ respectively and let b be the vector representing the right-hand side of the constraints in SIMPLE_SNDP(c,{d},δ). Then the polyhedron of feasible solutions for SIMPLE_SNDP(c,{d},δ) is

$$P = \left\{ (y,f) \in \mathbb{R}_+^{|E|} \times \mathbb{R}_+^{2\times|K|\times|E|\times|\mathcal{T}|} : Ay + Bf \leq b \right\}$$

According to Balas [Bal98], if we let

$$W = \left\{ w \in \mathbb{R}_+^{|V|\times|K|+|E|+|E|\times|\mathcal{T}|} : wB = 0 \right\}$$

then the projection

$$P_y^d = \left\{ y \in \mathbb{R}_+^{|E|} : (wa_i)y \leq wb_i \right\}$$

with a_i the i-th row of A, can be obtained by using a block elimination approach. The same approach can be followed to create a projection P_y^D for the problem SIMPLE_SNDP(c,D,δ).

Domination checking algorithm for survivable dynamic routing

With the projections P_y^d and P_y^D at hand, an approach can be followed whereby each inequality $(wa_i)y \leq wb_i$ in P_y^d can be tested for redundancy against P_y^D. Figure 5.4 illustrates the idea by showing an inequality $(wa_i)y \leq wb_i$ in P_y^d that is not redundant with respect to P_y^D, indicating that $\mathcal{Y}(D,\delta) \backslash \mathcal{Y}(d,\delta) \neq \emptyset$. Checking for redundancy with respect to P_y^D does not necessarily have to involve explicitly calculating P_y^D. By solving SIMPLE_SNDP(wa_i,{d},δ) for a specific inequality $(wa_i)y \leq wb_i$ in P_y^d, an objective function value $(wa_i)y^*$ is obtained, with y^* the optimal solution.

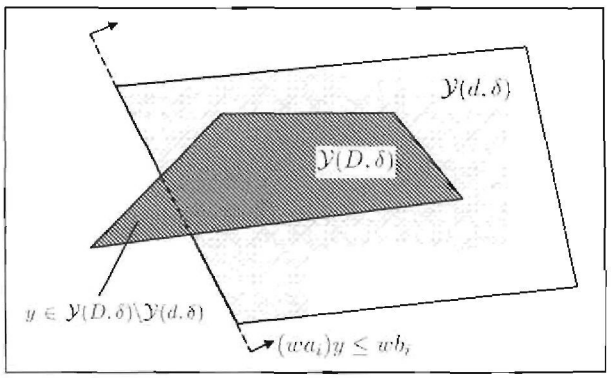

Figure 5.4: Projection unto the y variables

If $(wa_i)y^* > w_b i$, then it can be concluded that the inequality $(wa_i)y \leq wb_i$ is not redundant with respect to P_y^D and, therefore, $\mathcal{Y}(D,\delta)\backslash\mathcal{Y}(d,\delta) \neq \emptyset$ with the result that D does not dominate d.

The computational complexity of the approach will largely depend on the number of inequalities describing P_y^d, since for each of the inequalities an LP with a large number of constraints will have to be solved. Furthermore, it is not clear what the actual computation of P_y^d will entail. For computational feasibility of the suggested approach, computing of P_y^d should be done efficiently, since it has to be done for each demand vector in the set D.

5.2 A Branch-and-cut Framework

The branch-and-cut approach is an adaptation of the **branch-and-bound** method, see for instance [PR91]. A short overview of the branch-and-bound method and how it forms the basis of the branch-and-cut framework is described in the subsequent section.

5.2.1 Branch-and-bound basics

The branch-and-bound method works on the basis of systematically enumerating integer solutions. To illustrate this method consider SNDP(d, δ) that has been formulated as a MIP with $\bar{x} \in \mathcal{H}$ defined as the adjoint vectors $(x, y) \in \mathcal{H}$ representing all the variables from the hardware model. The initial step in the branch-and-bound method is to drop all integrality constraints from the MIP and to solve the LP relaxation of the problem. Let this sub-problem be denoted by LP_0. If LP_0 is discovered to be infeasible, then SNDP(d, δ) is infeasible and the process is terminated. If, on the other hand, LP_0 does have a solution, say $\bar{x}^{LP}$, and the solution is integer, i.e. $\bar{x}^{LP} \in \mathcal{H}$, then SNDP($d$, δ) is solved and the algorithm terminates. Else, if there is at least one component of $\bar{x}^{LP}$ that is not integer, the objective function value for LP_0 is taken as the lower bound $\underline{Z}^*$ for the MIP, and the upper bound $\overline{Z}^*$ is set to infinity. The next step in the process is to partition

the feasible region defined by LP_0 into two sub-problems. This is done by selecting one of the variables of $\bar{x}^{LP}$ for which the integrality constraint is violated, say $\bar{x}_r^{LP}$, and creating the two sub-problems LP_1^r and LP_2^r. This step is referred to as **branching**. To create the sub-problem LP_1^r the constraint $\bar{x}_r \leq \lfloor \bar{x}_r^{LP} \rfloor$ is added to the parent problem LP_0 and for the sub-problem LP_2^r the constraint $\bar{x}_r \geq \lceil \bar{x}_r^{LP} \rceil$ is added to the parent problem LP_0. It is convenient to represent the sub-problems as nodes in a binary tree. For our explanation, the parent node will denote the problem LP_0 and the two child nodes will denote sub-problems LP_1^r and LP_2^r respectively. Now, whenever child nodes are created, their solutions are checked for integer feasibility. If the solution to a child node satisfies all integrality constraints and its objective function value, say Z_p, is less than the current upper bound $\overline{Z}^*$, this child node becomes the incumbent integer solution with the new upper bound $\overline{Z}^* = Z_p$. Else, if not all integrality constraints are satisfied and the objective value for the node is less then the upper bound $\overline{Z}^*$, then the node is added to an open list referred to as **dangling nodes**.

The next step is to select from the set of dangling nodes the next sub-problem that will be branched on. This is referred to as **node selection**. If the list of dangling nodes is empty, then the branch-and-bound process terminates. The optimal solution to the MIP will then be the incumbent solution. If no incumbent solution however exists then the MIP has no feasible solution.

Several criteria may exist for node selection. One such a criterion could be to select the node for which the sub-problem has the minimum objective function value. Once a node has been selected, it is removed from the list of dangling nodes and the branching process is repeated. Sub-problems are created and added to the list of dangling nodes if necessary. Note that if a new incumbent has been identified, the list of dangling nodes are traversed and if a node has an objective value that exceeds that of the new incumbent (i.e. the new upper bound), it is removed from the list.

5.2.2 Improving the performance of the branch-and-bound

Substantial research has been done on several aspects of the branch-and-bound approach to improve computing times. Empirical evidence has shown that branching choices can significantly reduce computing times. State of the art solvers usually provide several branching strategies from which to choose from. In the paper by Linderoth and Savelsbergh [LS99] it was empirically found however, that no branching strategy dominates when various problem instances are considering. That is, a particular branching strategy might outperform other strategies when considering a specific problem instance. But for a different problem instance this same branching strategy might be inferior to others.

An approach that proved to significantly reduce computing times of the branch-and-bound is the application of valid inequalities. The basic idea is to strengthen the mixed integer programming formulation by attempting to shrink the feasible region of the MIP to only the convex hull of

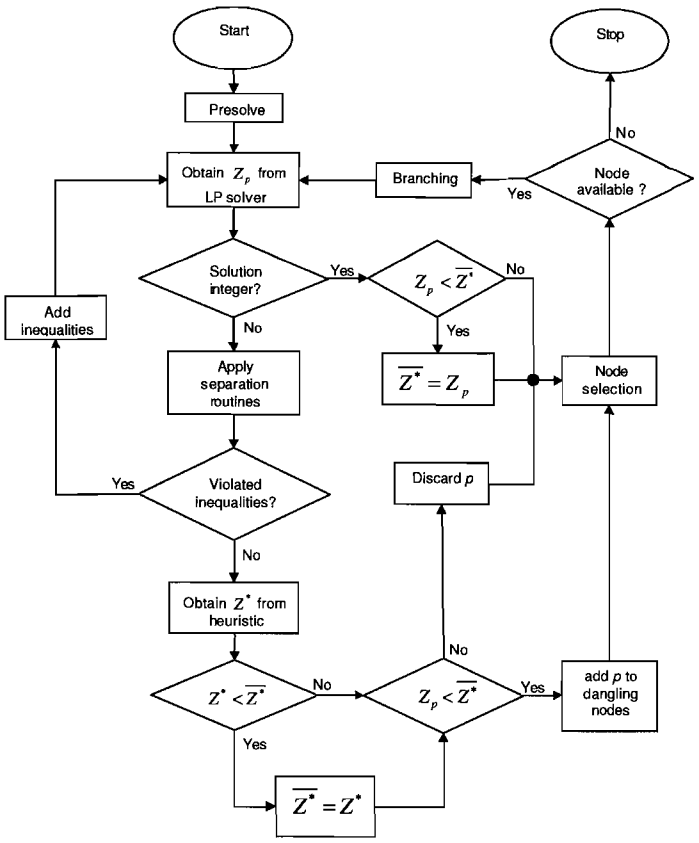

Figure 5.5: Outline of the branch-and-cut framework

integer solutions by adding facet defining valid inequalities. If this can be achieved an optimal solution for the problem can readily be found by solving only the linear programming relaxation since all extreme points will be integer solutions. It is, however, unlikely that all facet defining valid inequalities can be found in reasonable time.

For the SNDP-DR(D, δ) problem, and therefore also for the problems SNDP-DR/SR(D, δ), several classes of valid inequalities are applicable such as general upper bound cover and k-graph-partition inequalities [DS94], band inequalities [Wol90], and strengthened metric inequalities [Wes00]. The branch-and-cut approach therefore entails extending the branch-and-bound method by sporadically adding valid inequalities to the problem formulation. The valid inequalities are obtained from separation procedures that are applied after solving the LP subproblems at the nodes of the branch-and-bound tree. That is, based on the information provided by the solution of the LP relaxation, a separation procedure might provide a violated inequality that could be added to the problem formulation in order to improve the global lower bound of the problem.

Figure 5.5 gives a flow diagram of the branch-and-cut algorithm. Note that this is just a conceptual description of the branch-and-cut and some variations on this might be implemented as part of other solution approaches. Important to note from Figure 5.5 is that most commercial

MIP solvers employ some kind of a presolve technique in order to eliminate redundant constraints and/or variables. Also typical to a branch-and-cut approach is the application of heuristics for generating primal solutions. The exact sequence and frequency of applying heuristics may once again vary from one implementation of the branch-and-cut to the other.

5.2.3 Projecting out the routing model

The approach followed in this thesis for solving the survivable network design problem is based on a Benders decomposition [Ben62] whereby part of the problem is projected out and only incorporated back into the problem in the form of valid inequalities during the branch-and-cut process. For the subsequent discussion only the single demand vector routing model will be considered. This approach, however, is easily extended to the multiple demand vector routing case. Recall that $\bar{x} \in \mathcal{H}$ is defined as the adjoint vectors $(x, y) \in \mathcal{H}$ representing all the variables from the hardware model. Let c be the cost vector for $\bar{x}$ and let w be the cost vector for the flow variables $f \in F(y, D)$ which are part of the routing model. In addition, let A and W be appropriate matrices associated with the constraints found in the hardware model and the routing model respectively, then SNDP(d, δ) can be expressed according to the following notation:

$$\begin{aligned} &\min\ c\bar{x} + wf \\ &\text{s.t. } A\bar{x} + Wf \geq b \\ &\bar{x} \in \mathcal{H} \end{aligned} \tag{5.15}$$

The cost vector w is included at this stage only for purposes of illustrating the decomposition approach. Otherwise, flow costs are not considered at all for the remainder of the thesis. If a fixed hardware configuration $\bar{x}$ is considered, a minimum cost solution to the routing problem is given by:

$$\min_{f} \{wf : Wf \geq b - A\bar{x}, f \geq 0\} \tag{5.16}$$

By taking the dual of (5.16), the overall problem SNDP(d, δ) can be stated as:

$$\min_{\bar{x} \in \mathcal{H}} \left\{ c\bar{x} + \max_{u \geq 0} \left\{ (b - A\bar{x})^T u : W^T u \leq w, \right\} \right\} \tag{5.17}$$

The approach suggested by Benders is to use the outer minimisation problem of (5.17) as a master problem and the inner maximization problem as a sub-problem. The sub-problem generates cuts for a given configuration x that are added to the master problem. For the branch-and-cut approach the above decomposition applies but with a modified sub-problem since we do not take into account flow costs. Instead of minimizing flow costs, the shortfall in capacity is minimized. The cuts obtained

from the sub-problems are, consequently, exactly the metric inequalities stated in Section 4.3. Details on the separation of valid inequalities are provided in Section 5.4.

By denoting $\tilde{\mathcal{H}} = \{\bar{x} \in \mathcal{H} : (b - A\bar{x})^T u_i \geq 0, i = 1, 2, \ldots, L\} \subseteq \mathcal{H}$ as the polyhedron for the hardware part of the SNDP(d, δ) that has been augmented with L-number of metric inequalities, SNDP(d, δ) can be stated as:

$$\begin{aligned} &\min c\bar{x} \\ &\text{s.t. } \bar{x} \in \tilde{\mathcal{H}} \end{aligned} \tag{5.18}$$

Since the Benders sub-problems are solved only for purposes of generating violated metric inequalities, the problem being solved by the branch-and-cut process is simply the Benders master problem, i.e. the hardware part of SNDP(d, δ). The exponential number of flow variables that is found in the routing part, have therefore been projected out of the mixed integer programming problem. In addition, another attractive feature of the approach is that since SNDP-DR/SR(D, δ) only differs from SNDP(d, δ) with respect to routing, the approach is easily extended to cater for non-simultaneous multiple demand vectors. The master problem remains unchanged and the sub-problems will solve the routing for either SNDP-DR(D, δ) or SNDP-SR(D, δ), depending on whether dynamic or static routing is considered.

5.3 Application of Heuristics within Branch-and-cut

From an algorithmic point of view the number of demand vectors that will be considered as part of a survivable network design problem will contribute to the complexity of the problem. However, the network design problem with a single traffic demand vector is already considered to be a hard problem (see SNDLIB [snd05]). It is for this reason that heuristics have been explored that could improve computing performance for network design problems even where only a single demand vector is considered. Therefore, the heuristics presented in subsequent sections can be applied within the general context of network design problems, independent of the number of demand vectors considered.

5.3.1 K-Opt heuristic

In the paper by Van der Merwe and Hattingh [MH06], a partitioning algorithm is described for improving the solution times of a local access network design problem. The philosophy behind the partitioning approach is that information from the linear programming relaxation solution could be used in the solution process of the integer programming problem. The objective of the local access network design problem is to determine the optimal set of nodes that will be used for local access points. Therefore, associated with each potential access point is a binary variable that indicates

as part of a solution whether a node will serve as an access point or not. The basic idea of the approach is then to partition the set of binary variables representing the set of potential access points into two groups. The first group of variables, say the active set of variables, is identified as the nodes that are most likely to be in the final solution. This is achieved by examining the solution to the linear programming relaxation of the problem. Due to the special structure of the problem it may happen that a significant number of binary variables that are set in an optimal solution, will also have a value of one in the solution of the linear programming relaxation.

Two cardinality constraints are added to the local access network design problem to facilitate the partitioning procedure. The one cardinality constraint forces the model to select a subset from the active set of variables that will take on a value of one as part of a solution, and the second cardinality constraint forces a subset from the inactive set of variables to take on a value of zero. That is, the cardinality constraints enable the model to decide which variables from the active set and which variables from the inactive set will be part of the the optimal solution. This feature of the model basically defines an exchange operation whereby a subset of nodes from the active set are exchanged for a subset of nodes in the inactive set. This is similar to the *K-Opt* heuristic scheme employed in the *Traveling Salesman Problem* (TSP) whereby k-number of edges are replaced in order to try and find a new improved tour.

The above partitioning idea could be employed for solving the survivable network design problem SNDP(d, δ). Recall that $\bar{x} \in \mathcal{H}$ is defined as the adjoint vectors $(x, y) \in \mathcal{H}$ representing all the variables from the hardware model. Let $\bar{x}^{\text{IP}}$ denote the optimal solution for the hardware part of SNDP(d, δ) and $\bar{x}^{\text{LP}}$ denote the solution for the hardware model obtained by solving the linear programming relaxation of SNDP(d, δ). Now define the following vector:

$$x_j^* = \left\{ \begin{array}{ll} 1 & \text{if } \bar{x}_j^{\text{LP}} \geq \Psi; \\ 0 & \text{if } \bar{x}_j^{\text{LP}} < \Psi. \end{array} \right\} \text{for } j = 1, 2, \ldots, n \tag{5.19}$$

with Ψ some indicator level choosen arbitrarily and n the dimension of $\bar{x}$. The vector x^* is referred to as a linear programming prediction of $\bar{x}^{\text{IP}}$. A simple empirical experiment was done to get a feeling for how accurate x^* is in predicting $\bar{x}^{\text{IP}}$. By counting the number of entries in x^* that match the entries in $\bar{x}^{\text{IP}}$, divided by the size of the two vectors, a measure is obtained of how accurate the prediction is. The column labeled *Accuracy* in Table 5.1 gives the percentage of success x^* had in predicting $\bar{x}^{\text{IP}}$ for some problem instances obtained from SNDLIB [snd05]. It is interesting to note from Table 5.1 that for some problem instances a good prediction of what the optimal solution should be was obtained by solving the linear programming relaxation. For the results generated in Table 5.1 an indicator level of 0.1 was considered. It might be that for some other levels of Ψ, different levels of accuracy could be obtained.

With a linear programming prediction x^* at hand we can define the index sets

Problem Instance	Accuracy
newyork	94.8
nobel-germany	93.1
nobel-us	75.5
pdh	46.5
ta2	96.8

Table 5.1: Empirical evidence for predicting integer solutions

$X_1 = \{j : x_j^* = 1, j = 1, 2, \ldots, n\}$ and $X_0 = \{j : x_j^* = 0, j = 1, 2, \ldots, n\}$, to refer to the variables in the active set and inactive set respectively.

The following is an amended version of SNDP(d, δ) denoted by SNDP-CARD(η, ζ), that includes cardinality constraints for the purpose of facilitating a partitioning scheme:

$$\begin{aligned} &\min\ cx \\ &\text{s.t.} \bigcup_{(x,y)\in\mathcal{H}} F(y, d, \delta) \neq \emptyset \\ &\sum_{j\in X_1} x_j = \eta \qquad (5.20) \\ &\sum_{j\in X_0} x_j = \zeta \qquad (5.21) \end{aligned}$$

By solving SNDP-CARD(η, ζ) with $\eta = |X_1|$ and $\zeta = 0$, the model forces all the variables in the active set to take on a value of one and it forces all the variables in the inactive set to take on a value of zero. An exchange operation between the active and inactive sets can now easily be accomplished by changing the parameters η and ζ. For instance, by setting $\eta = |X_1| - 1$ and $\zeta = 1$, the model allows a variable in the active set to be exchanged with a variable in the in-active set.

A very attractive feature of using the cardinality constraints (5.20) and (5.21), is that SNDP-CARD(η, ζ) becomes easier to solve for values of η close to $|X_1|$ and values of ζ close to zero. That is, for a small number of allowable exchanges the search space is reduced significantly.

In contrast to the partitioning scheme by [MH06] and the K-Opt heuristic used for the TSP, a single exchange between the active set and the inactive set might not be sensible within the context of mesh based network design. Consider the exchange of an edge design between the active and inactive variable sets. An improved solution might require that an expensive edge design on a specific edge from the active set be replaced by two less expensive edge designs on two distinct edges from the inactive set. Another scenario might be that an improved solution requires that a variable in the active set be forced to zero, without increasing the levels of any of the variables in

Algorithm 2 K-Opt solution CONstruction (KOCON)

Require: An LP relaxation solution x^{LP}

Set $\Psi = 0.1$

Obtain x^* by applying indicator function (5.19).

Define index set $X_1 = \{j : x_j^* = 1, j = 1, 2, \ldots, n\}$.

Define index set $X_0 = \{j : x_j^* = 0, j = 1, 2, \ldots, n\}$.

Set $\eta = |X_1| - 2$

Set $\zeta = 2$

Obtain x if SNDP-CARD-RELAX(η, ζ) is feasible.

the inactive set. That is, there are redundant edge designs installed while for an improved solution an edge design must be removed.

Note that although the index sets X_1 and X_0 are constructed by using the vector x^* obtained by applying the indicator function (5.19), any feasible solution $\bar{x} \in \mathcal{H}$ can be used in constructing X_1 and X_0. Consequently, it might be possible to improve a feasible solution $\bar{x} \in \mathcal{H}$ by letting $X_1 = \{j : \bar{x}_j = 1, j = 1, 2, \ldots, n\}$ and $X_0 = \{j : \bar{x}_j = 0, j = 1, 2, \ldots, n\}$, and by solving SNDP-CARD(η, ζ) with appropriate values for η and ζ.

The proposed heuristic, called the SNDP-K-Opt heuristic is making use of the SNDP-CARD(η, ζ) formulation in an attempt to find primal solutions during the course of the branch-and-cut process. The approach of the heuristic is, firstly, to try and construct a feasible solution from information provided by a linear programming relaxation solution. Such a linear programming relaxation solution is readily available whenever a sub-problem has been solved at a node of the branch-and-cut tree. Secondly, if the heuristic has succeeded in constructing a feasible solution, it will then try to improve the solution. Therefore, the heuristic basically comprises two parts: the K-Opt solution CONstruction part (KOCON) and the K-Opt solution IMProvement part (KOIMP).

The solution construction part of the SNDP-K-Opt heuristic listed in Algorithm 2, does not iterate over a range of values for the parameters η and ζ. Preselected values for η and ζ are used based on empirical experiences. Furthermore, a relaxed version of the SNDP-CARD(η, ζ) problem is suggested to allow more flexibility in finding feasible solutions. This relaxed version of SNDP-CARD(η, ζ) is denoted by SNDP-CARD-RELAX(η, ζ) and is obtained by replacing the equality constraint (5.21) with the following inequality:

$$\sum_{j \in X_0} x_j \leq \zeta \tag{5.22}$$

Algorithm 3 K-Opt solution IMProvement (KOIMP)

Require: A feasible solution $\bar{x}^*$

Set $i = 1$

Set $x^i = \bar{x}^*$

Set STOP_FLAG = false

repeat

Define index set $X_1 = \{j : x_j^i = 1, j = 1, 2, \ldots, n\}$.

Define index set $X_0 = \{j : x_j^i = 0, j = 1, 2, \ldots, n\}$.

Set $\eta = |X_1| - 2$

Set $\zeta = \infty$

if SNDP-CARD-RELAX(η, ζ) is feasible **then**

i $\leftarrow$ i + 1

Obtain solution x^i

else

Set STOP_FLAG = true

end if

until STOP_FLAG = true

The solution improvement part of the SNDP-K-Opt heuristic listed in Algorithm 3 depends on an existing feasible solution $\bar{x}^*$ as input. The active and inactive sets, X_1 and X_0 respectively, are constructed relative the to $\bar{x}^*$. Once a new feasible solution has been found by solving SNDP-CARD-RELAX(η, ζ), the process is repeated by again constructing the sets X_1 and X_0 based on the new feasible solution and by solving SNDP-CARD-RELAX(η, ζ). This process is continued until no further feasible solution can be found. Values selected for η and ζ for solving SNDP-CARD-RELAX(η, ζ) are based on empirical experience.

5.3.2 Edge flow heuristic

The Edge Flow solution CONstruction (EFCON) heuristic makes use of the LP relaxation solution of a sub-problem within the branch-and-cut approach to construct a feasible solution for SNDP(d, δ) and SNDP-DR/SR(D, δ). Let (x^*, y^*) be the LP relaxation solution for the variables in the hardware model. Then, if it is found that $F(y^*, d, \delta) \neq \emptyset$, $F(y^*, D, \delta) \neq \emptyset$ or $R(y^*, d, \delta) \neq \emptyset$, depending on whether SNDP(d, δ), SNDP-DR(D, δ), or SNDP-DR(D, δ) is solved, the following problem is solved:

$$\begin{aligned} &\min\ cx \\ &\text{s.t. } y_e^* \leq y_e \ \ \forall \ \ e \in E \\ &(x, y) \in \mathcal{H} \end{aligned} \tag{5.23}$$

A feasible solution to this problem will also be feasible for the overall problem SNDP(d, δ) or SNDP-DR/SR(D, δ).

5.4 Separation of Metric Inequalities within Branch-and-cut

A separation algorithm within a mathematical optimization setting has a two-fold purpose. Firstly, for an arbitrary polyhedron $X \subseteq \mathbb{R}^n$, the separation algorithm has to determine whether $x^* \in \mathbb{R}^n$ lies within X. Secondly, if it is found that x^* is not in X, then the separation algorithm should be able to produce a hyperplane $\{x \in \mathbb{R}^n : ax \leq b\}$ that is valid for X, but violates x^*. That is, the hyperplane separates x^* from X.

Within the context of the survivable network design problem, specifically if we consider the decomposition of SNDP(d, δ) as described in Section 5.2.1, the separation problem entails finding a valid inequality that separates a point $x^* \in \mathcal{H}$ from $\tilde{\mathcal{H}} \subseteq \mathcal{H}$, with $\tilde{\mathcal{H}}$ containing all the cuts generated by the Benders sub-problems. It should be clear, by considering the proofs of Theorem 4.1 and Theorem 4.3 , that the cuts generated by the sub-problems within the Benders framework are exactly the metric inequalities described for strengthening the dynamic and static routing models SNDP-DR(D, δ) and SNDP-DR/SR(D, δ) respectively.

5.4.1 Separating metric inequalities for dynamic routing

Consider the dynamic routing polyhedron $F(y^*, D, \delta)$ for a given capacity vector y^*, a set of demand vectors D and a given diversification vector δ. By introducing the variables $\alpha^t \in \mathbb{R}_+$ for all $t \in \mathcal{T}$ representing the shortfall in capacity for $F(y^*, D, \delta)$, the following linear programming problem is obtained:

$$\min \sum_{t\in\mathcal{T}} \alpha^t$$

$$\text{s.t.} \sum_{p\in\mathcal{P}(k)} f_p^t = d_k^t \quad \forall\, k \in K,\ \forall\, t \in \mathcal{T} \tag{5.24}$$

$$\sum_{k\in K} \sum_{p\in\mathcal{P}(k,e)} f_p^t - \alpha^t \leq y_e^* \quad \forall\, e \in E,\ \forall\, t \in \mathcal{T} \tag{5.25}$$

$$\sum_{p\in\mathcal{P}(k,s)} f_p^t \leq \delta_k d_k^t, \quad \forall\, k \in K, \quad \begin{matrix}\forall\, s \in \mathcal{S},\\ \forall\, t \in \mathcal{T}\end{matrix} \tag{5.26}$$

$$\tag{5.27}$$

To indicate feasibility of a capacity vector y^* the optimal solution should yield $\alpha^t = 0$ for all $t \in \mathcal{T}$. By associating the variables $\pi \in \mathbb{R}^{|K|\times|\mathcal{T}|}$, $\mu \in \mathbb{R}_+^{|E|\times|\mathcal{T}|}$ and $\gamma \in \mathbb{R}_+^{|K|\times|S|\times|\mathcal{T}|}$ with the constraints (5.24),(5.25) and (5.26) respectively, the corresponding dual is obtained:

$$\max \sum_{t\in\mathcal{T}} \left(\sum_{k\in K} d_k^t \pi_k^t - \sum_{e\in E} y_e^* \mu_e^t - \sum_{k\in K} \left(\delta_k d_k^t \sum_{s\in S} \gamma_{ks}^t \right) \right)$$

$$\text{s.t.} \sum_{e\in E} \mu_e^t = 1 \quad \forall\, t \in \mathcal{T} \tag{5.28}$$

$$\pi_k^t - \sum_{e\in E(p)} \mu_e^t - \sum_{s\in S(p)} \gamma_{ks}^t \leq 0 \quad \forall\, p \in \mathcal{P}(k), \quad \begin{matrix}\forall\, k \in K,\\ \forall\, t \in \mathcal{T}\end{matrix} \tag{5.29}$$

For a positive optimal objective value that signals insufficient capacity, the following metric inequalities are derived:

$$\sum_{e\in E} y_e \mu_e^t \geq \sum_{k\in K} d_k^t \pi_k^t - \sum_{k\in K} \left(\delta_k d_k^t \sum_{s\in S} \gamma_{ks}^t \right) \quad \forall t \in \mathcal{T} \tag{5.30}$$

The above linear programming problem is, therefore, a separation algorithm since it is able to separate the point y^* from $\tilde{\mathcal{H}}$. In order to manage the enormous number of path variables in the linear programming problem a column generation approach is followed. Consider a single demand vector d^t, $t \in \mathcal{T}$. Then the constraint set (5.29) in the dual problem states that for a commodity $k \in K$

$$\pi_k^t = \min_{p\in\mathcal{P}(k)} \left\{ \sum_{e\in E(p)} \mu_e^t + \sum_{s\in S(p)} \gamma_{ks}^t \right\}$$

Note, however, that since we only consider single edge failures, each failure state $s \in S$ refers to a specific edge. Therefore, if we allow the notation γ_{ke}^t to be equivalent to γ_{ks}^t where the failure state s contains only the edge e, then for each commodity $k \in K$ and each $t \in \mathcal{T}$ the inequality (5.29) can be written as:

$$\pi_k^t = \min_{p \in \mathcal{P}(k)} \left\{ \sum_{e \in E(p)} (\mu_e^t + \gamma_{ke}^t) \right\}$$

This is equivalent to saying that for an optimal solution to the separation problem, π_k^t is the length of the shortest path among all the paths in the set $\mathcal{P}(k)$, $k \in K$. However, if not all path variables have been included in the formulation of the problem, there might exist a shorter path that has a length that is less than the value of π_k^t. Therefore, the column generation approach entails the calculation of a shortest path for each commodity $k \in K$ and each $t \in \mathcal{T}$ using the the edge weights $\mu_e^t + \gamma_{ke}^t$ for all $e \in E$. Whenever it is found that the length of this shortest path is less than π_k^t, this new shortest path is added to the problem.

A strengthening procedure for improving separation

For each of the metric inequalities in (5.30) a strengthening procedure can be applied. Consider a metric inequality for a demand vector d^t, $t \in \mathcal{T}$ and for an arbitrary capacity vector y. Note that although we can obtain values for μ_e^t and π_k^t from duality, the metric inequality should hold irrespective of how the edge weights have been calculated. More specifically, the edge weights could be calculated independently of the demand vector d^t. This in turn implies that for an arbitrary set of edge weights μ_e^t for all $e \in E$, an arbitrary set of failure state weights γ_{ks}^t for all $s \in S$, and shortest path lengths π_k^t for all $k \in K$, the capacity vector y will allow a feasible routing of the set of demand vectors $D = \{d^1, d^2, \ldots, d^{|\mathcal{T}|}\}$ if the following inequality holds:

$$\sum_{e \in E} y_e \mu_e^t \geq \sum_{k \in K} d_k^q \left(\pi_k^t - \delta_k \sum_{s \in S} \gamma_{ks}^t \right) \quad \forall\, q \in \mathcal{T}$$

which is equivalent to saying

$$\sum_{e \in E} y_e \mu_e^t \geq \max_{q \in \mathcal{T}} \left\{ \sum_{k \in K} d_k^q \left(\pi_k^t - \delta_k \sum_{s \in S} \gamma_{ks}^t \right) \right\} \tag{5.31}$$

It is expected that for some demand vectors the strengthening procedure might be redundant due to the magnitude of some of the demand entries. Therefore, in order to limit unnecessary processing two strategies for selecting a subset of the demand vectors are suggested. For this purpose (5.31) is rewritten below by replacing the index set $\mathcal{T}$ by $\mathcal{T}_{sub} \subseteq \mathcal{T}$. Depending on the strategy employed, $\mathcal{T}_{sub}$ will contain only the indices of the demand vectors to be used in the strengthening procedure.

$$\sum_{e \in E} y_e \mu_e^t \geq \max_{q \in \mathcal{T}_{sub}} \left\{ \sum_{k \in K} d_k^q \left(\pi_k^t - \delta_k \sum_{s \in S} \gamma_{ks}^t \right) \right\} \tag{5.32}$$

The Largest Sum of Entries-TOPx strategy (LSE-TOPx)

The approach followed in the LSE-TOPx strategy is the arranging of the demand vectors in descending order according to the largest sum of entries. For each demand vector d^t, for all $t \in \mathcal{T}$, the ordering criterion is calculated as $\sum_{k \in K} d_k^t$. During the separation of metric inequalities in the branch-and-cut process, only a percentage of the top demand vectors in the ordered list will be used for the strengthening procedure. That is, the strategies LSE-TOP2, LSE-TOP5, and LSE-TOP10 will refer to selecting only the top 2, 5, and 10 percent respectively of the demand vectors in the ordered list.

The MDE-TOPx strategy

For the MDE-TOPx strategy the ordering criterion is based on the maximum demand entry found in each demand vector d^t, for all $t \in \mathcal{T}$. For each demand vector indexed by $t \in \mathcal{T}$ the ordering criterion is $\max_{k \in K} \{d_k^t\}$. The strategies MDE-TOP2, MDE-TOP5, and MDE-TOP10 will refer to selecting only the top 2, 5, and 10 percent respectively of the demand vectors in the ordered list.

5.4.2 Separating metric inequalities for static routing

For the static routing case, consider the routing polyhedron $R(y^*, D, \delta)$ for a given capacity vector y^*, a set of demand vectors D and a given diversification vector δ. By introducing the variables $\alpha^t \in \mathbb{R}_+$ for all $t \in \mathcal{T}$ representing the shortfall in capacity for $R(y^*, D, \delta)$, the following linear programming problem is obtained:

$$\min \sum_{t \in \mathcal{T}} \alpha^t$$

$$\text{s.t.} \sum_{p \in \mathcal{P}(k)} r_p = 1 \quad \forall\, k \in K \tag{5.33}$$

$$\sum_{k \in K} \sum_{p \in \mathcal{P}(k,e)} d_k^t r_p - \alpha^t \leq y_e^* \quad \forall\, e \in E,\ \forall\, t \in \mathcal{T} \tag{5.34}$$

$$\sum_{p \in \mathcal{P}(k,s)} r_p \leq \delta_k, \quad \forall\, k \in K,\ \forall s \in \mathcal{S} \tag{5.35}$$

$$\alpha^t \geq 0 \quad \forall\, t \in \mathcal{T} \tag{5.36}$$

To indicate feasibility of the capacity vector y^* the optimal solution should yield $\alpha^t = 0$ for all $t \in \mathcal{T}$. By associating the variables $\pi \in \mathbb{R}^{|K|}$, $\mu \in \mathbb{R}_+^{|E| \times |\mathcal{T}|}$ and $\gamma \in \mathbb{R}_+^{|K| \times |S|}$ with the constraints (5.33),(5.34), and (5.35) respectively, the corresponding dual is obtained:

$$\max \sum_{k \in K} \left(\pi_k - \delta_k \sum_{s \in S} \gamma_{ks} \right) - \sum_{t \in \mathcal{T}} \sum_{e \in E} y_e^* \mu_e^t$$

$$\text{s.t.} \sum_{e \in E} \mu_e^t = 1 \quad \forall\, t \in \mathcal{T} \tag{5.37}$$

$$\pi_k - \sum_{t \in \mathcal{T}} \sum_{e \in E(p)} d_k^t \mu_e^t - \sum_{s \in S(p)} \gamma_{ks} \leq 0 \quad \forall\, p \in \mathcal{P}(k),\ \forall\, k \in K \tag{5.38}$$

For a positive optimal objective value signalling insufficient capacity, the following metric inequality is derived:

$$\sum_{t \in \mathcal{T}} \sum_{e \in E} y_e \mu_e^t \geq \sum_{k \in K} \left(\pi_k - \delta_k \sum_{s \in S} \gamma_{ks} \right)$$

The above linear programming problem is, therefore, a separation algorithm since it is able to separate the point y^* from $\tilde{\mathcal{H}}$. Similar to the dynamic case, a column generation approach is followed for managing the enormous number of path variables. The challenge is to find for a commodity $k \in K$ a shortest path p that minimizes the entire expression

$$\sum_{t \in \mathcal{T}} \sum_{e \in E(p)} d_k^t \mu_e^t + \sum_{s \in S(p)} \gamma_{ks} \tag{5.39}$$

Since only single edge failures are considered for the failure states, expression (5.39) can be rewritten as

$$\sum_{e \in E(p)} \left(\sum_{t \in \mathcal{T}} d_k^t \mu_e^t + \gamma_{ke} \right) \tag{5.40}$$

A robust separation approach to static routing

An alternative to the static routing separation approach described above is to make use of a robust separation formulation. Consider the static routing polyhedron $R(y^*, D, \delta)$ for a given capacity vector y^*. Then, if the demand vector d is treated as the variable $d \in conv(D)$, the capacity constraint (4.15) in $R(y^*, D, \delta)$ can be rewritten such that the choice of d will always denote the worst case within $conv(D)$:

$$\max_{d \in conv(D)} \left\{ \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}(k,e)} r_p d_k \right\} \leq \alpha + y_e^* \quad \forall\, e \in E \tag{5.41}$$

Constraint (5.41) can be interpreted as follows: At any point in time during the solution process, if we have a routing r^*, then the maximisation on the left hand side of the constraint attempts to increase d up to the bounds defined by the polytope of $conv(D)$. If d takes on a value such that the capacity vector y is insufficient to allow the flow $\sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}(k)} r_p^* d_k$, the variable α will take on a positive value and a violated metric inequality is found.

Thus, if the routing variables r are treated as the constants r^*, the maximisation problem becomes linear and can be written as follows:

$$\max \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}(k,e)} r_p^* d_k \tag{5.42}$$

$$\text{s.t. } d_k - \sum_{t \in \mathcal{T}} \lambda^t d_k^t = 0 \quad \forall\, k \in \mathcal{K} \tag{5.43}$$

$$\sum_{t \in \mathcal{T}} \lambda^t = 1 \tag{5.44}$$

The constraints (5.43) and (5.44) ensure that a demand vector is written as a convex combination of the extreme points of the convex hull $conv(D)$. Associating the dual variables $w_{ke} \in \mathbb{R}$ and $v_e \in \mathbb{R}$ with, respectively, the constraints (5.43) and (5.44) for all $e \in E$, the dual of the preceding maximization problem is the following:

$$\min v_e \tag{5.45}$$

$$\text{s.t. } -\sum_{k \in \mathcal{K}} w_{ke} d_k^t + v \geq 0 \quad \forall\, t \in \mathcal{T} \tag{5.46}$$

$$w_{ke} \geq \sum_{p \in \mathcal{P}(k,e)} r_p^* \quad \forall\, k \in \mathcal{K},\ \forall\, e \in E \tag{5.47}$$

Let $\alpha \in \mathbb{R}_+$ be a variable representing the shortfall in capacity for $R(y^*, D, \delta)$. Then the following linear programming problem is obtained by substituting the maximisation problem on the left hand side of (5.41) with the above dual problem, and by treating r as a variable instead of a constant:

$$\min \alpha \tag{5.48}$$

$$\text{s.t. } v_e \leq \alpha + y_e^* \quad \forall\, e \in E \tag{5.49}$$

$$-\sum_{k \in \mathcal{K}} w_{ke} d_k^t + v_e \geq 0 \quad \forall\, t \in \mathcal{T},\ \forall\, e \in E \tag{5.50}$$

$$w_{ke} \geq \sum_{p \in \mathcal{P}(k,e)} r_p \quad \forall\, k \in \mathcal{K},\ \forall\, e \in E \tag{5.51}$$

$$\sum_{p \in \mathcal{P}(k)} r_p = 1 \quad \forall\, k \in \mathcal{K} \tag{5.52}$$

$$\sum_{p \in \mathcal{P}(k,s)} r_p \leq \delta_k \quad \forall\, k \in K,\ \forall\, s \in S \tag{5.53}$$

For the purpose of deriving a metric inequality the dual of the above linear programming problem is obtained:

$$\max \sum_{k \in \mathcal{K}} \left(\pi_k - \delta_k \sum_{s \in S} \gamma_{ks} \right) - \sum_{e \in E} y_e^* \mu_e \tag{5.54}$$

$$\text{s.t.} \quad \sum_{e \in E} \mu_e = 1 \tag{5.55}$$

$$\mu_e + \sum_{t \in T} \rho_e^t = 0 \quad \forall\, e \in E \tag{5.56}$$

$$\sigma_{ke} - \sum_{t \in T} d_k^t \rho_e^t = 0 \quad \forall\, k \in \mathcal{K},\ \forall\, e \in E \tag{5.57}$$

$$\pi_k - \sum_{e \in E(p)} \sigma_{ke} - \sum_{s \in S(p)} \gamma_{ks} \leq 0 \quad \forall\, p \in \mathcal{P}(k),\ \forall\, k \in \mathcal{K} \tag{5.58}$$

For a positive optimal objective value signalling insufficient capacity, the following metric inequality is derived:

$$\sum_{e \in E} y_e^* \mu_e \geq \sum_{k \in K} \left(\pi_k - \delta_k \sum_{s \in S} \gamma_{ks} \right) \tag{5.59}$$

The above linear programming problem is, therefore, a separation algorithm since it is able to separate the point y^* from $\tilde{\mathcal{H}}$. The path generation approach for robust separation entails checking whether, for each commodity $k \in K$ the value of π_k is less than or equal to the value obtained in the shortest path computation

$$\min_{p \in \mathcal{P}(k)} \left\{ \sum_{e \in E(p)} \sigma_{ke} + \sum_{s \in S(p)} \gamma_{ks} \right\}$$

Since only single edge failures are considered for the failure states, the above shortest path computation can be restated as:

$$\min_{p \in \mathcal{P}(k)} \left\{ \sum_{e \in E(p)} (\sigma_{ke} + \gamma_{ke}) \right\}$$

5.5 An Approach for Improving Scalability

In the preceding sections heuristic and separation improvements were described for improving computing times of the branch-and-cut approach. Although the suggested improvements are aimed at improving computing times when solving SNDP-DR/SR(D, δ), none of them have good scaling characteristics. Therefore, an approach is required that will ensure that the rate of change with respect to computing time is small relative to the increase in demand vectors. The approach described in the subsequent section attempts to improve scalability through the iterative solving of SNDP-DR/SR(D, δ) by employing the branch-and-cut approach.

5.5.1 The Iterative Polyhedron Expansion Approach (IPEA)

The motivation behind the approach is that from a domination perspective, the most important demand vectors to be considered during the optimisation process are the demand vectors constitut-

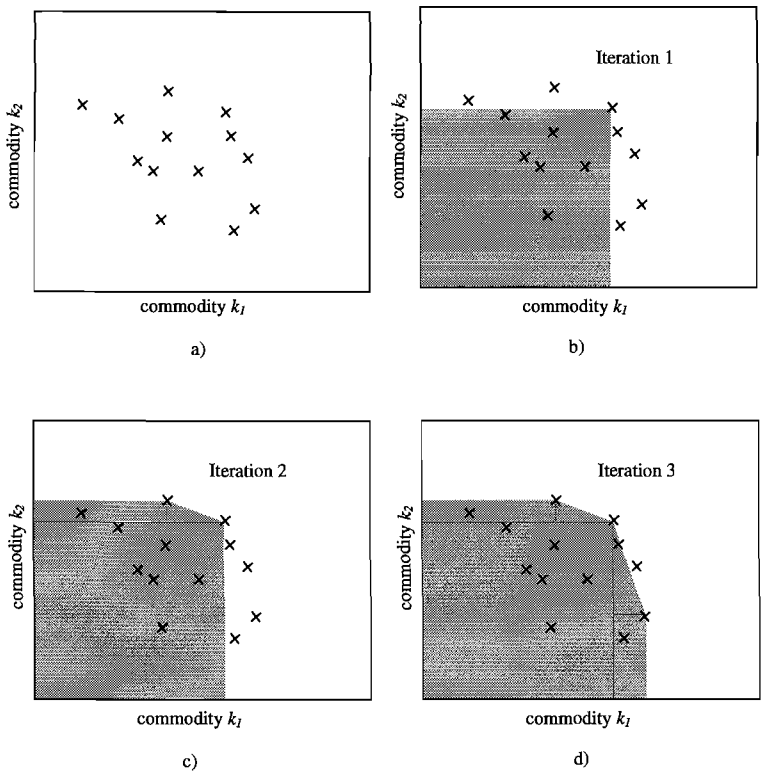

Figure 5.6: Iteratively expanding the convex hull of dominated demand vectors

ing the extreme points of the convex hull of demand vectors. It has been pointed out in the section on domination, Section 5.1, that for the current dominance checking algorithm there is some undetected domination. Furthermore, survivability requirements cannot be accommodated for the case of dynamic routing. Therefore, the basic idea behind the IPEA is to obtain solutions for SNDP-DR/SR(D, δ), by considering initially only a subset of demand vectors. During the branch-and-cut process the convex hull of all dominated demand vectors, associated with this initial set of demand vectors, is then systematically expanded by including demand vectors in the model for which the incumbent solutions are infeasible.

As an example, consider the illustration in Figure 5.6 showing in graph a) the projection of demand vectors onto the two-dimensional space of two commodities. During the first iteration, a single demand vector has been identified as part of the initial subset of demand vectors and the shaded area in graph b) shows the resulting convex hull of dominated demand vectors. As processing continues during the IPEA demand vectors are added to the initial subset of demand vectors, with the result that the convex hull of dominated demand vectors is expanded. Graphs c) and d) show the expanding convex hull of dominated demand vectors as new demand vectors are being added to the initial subset.

The subsequent outline provides a detailed description of each aspect of the IPEA. The num-

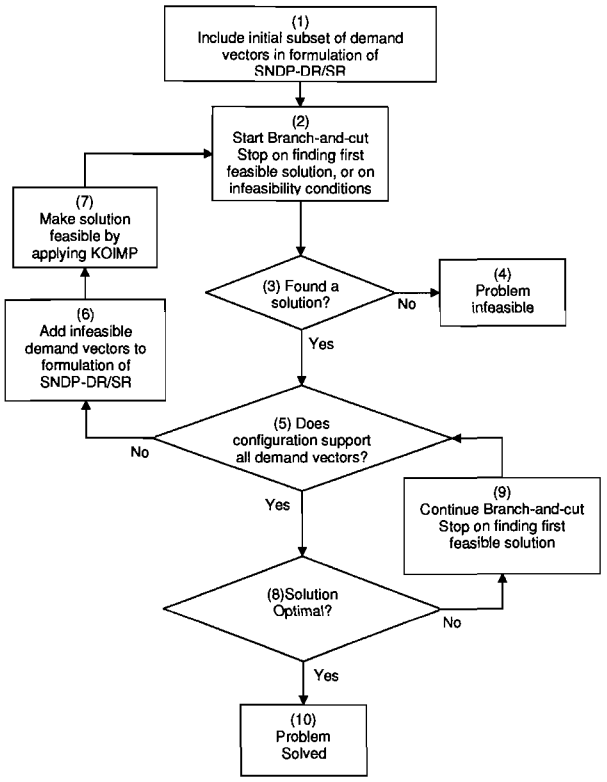

Figure 5.7: Iterative Polyhedron Expansion Algorithm (IPEA)

bering followed below corresponds to the numbering of each rectangle in the flow diagram of the IPEA as depicted in Figure 5.7.

1. The first step in the IPEA is selecting the initial subset of demand vectors denoted by $D_0 \subseteq D$, with D the entire set of demand vectors considered for the problem. Several strategies may be followed for selecting the initial set. Some suggestions are presented in Section 5.5.2.

2. The branch-and-cut algorithm is applied to SNDP-DR/SR(D_0, δ). The algorithm either stops due to infeasibility, or on finding the first feasible solution. That is, the process is terminated on finding any feasible solution, even if it has not found an optimal solution.

3. On termination of the branch-and-cut algorithm it should be clear from the underlying implementation whether a solution was found or whether the process terminated due to infeasibility.

4. Whenever this point in the IPEA approach has been reached where no feasible solution was found, it can be concluded that for the current subset of demand vectors D_0 no feasible solution exists, which implies that no feasible solution will exist when considering the entire set of demand vectors D.

5. Once a feasible solution has been found for the problem SNDP-DR/SR(D_0, δ), the set of

demand vectors $D_1 = D \setminus D_0$ is checked to see if there are any demand vectors for which the current hardware configuration is infeasible. That is, if we let y^* be the resulting capacity vector associated with the incumbent solution $(x^*, y^*) \in \mathcal{H}$, then we need to determine if $F(y^*, D_1, \delta) \neq \emptyset$, or $R(y^*, D_1, \delta) \neq \emptyset$, depending on whether dynamic or static routing is considered.

6. If it is found that $F(y^*, D_1, \delta) = \emptyset$, or $R(y^*, D_1, \delta) = \emptyset$, the demand vectors for which no feasible routing could be found are added to the set D_0. A very important choice to be made at this point is whether all of the infeasible demand vectors should be added at once, or whether only one of them should be added to the set D_0. Furthermore, if only a single infeasible demand vector is to be added, which one should it be? It was found empirically that for the dynamic routing case better results were obtained when all of the infeasible demand vectors were added at once to D_0, whereas with the static routing case performance improved when only the most infeasible demand vector was added to D_0. The most infeasible demand vector is defined as the one for which the sum of the shortfall in capacity over all the edges is maximal.

7. At this stage of the IPEA the initial set of demand vectors D_0 has been updated such that the incumbent solution $(x^*, y^*) \in \mathcal{H}$ found during the preceding steps of the approach is no longer feasible for SNDP-DR/SR(D_0, δ). However, it could be expected that (x^*, y^*) is close to a configuration that might support the newly updated D_0. Therefore, the KOIMP heuristic is applied at this point to exchange some of the solution values of (x^*, y^*) in an attempt to create a new feasible solution. If a new feasible solution was found it will be used as a starting solution for the next process in the flow diagram when the branch-and-cut is restarted.

8. Whenever it is found that the current configuration does indeed support the set of demand vectors D_0 (the result from step (8) in the flow diagram), it should be determined whether $(x^*, y^*) \in \mathcal{H}$ is optimal. This information should be readily available from the underlying implementation of the branch-and-cut algorithm.

9. If the branch-and-cut algorithm terminated with a zero percentage gap during step (2) or step (9) in the IPEA flow diagram and if the current configuration does support the entire set of demand vectors D, then surely an optimal solution has been found.

10. If, on the other hand, the optimal solution has not been found yet, the branch-and-cut process is continued until again it terminates due to finding a feasible solution.

5.5.2 Strategies for selecting the initial subset of demand vectors

Initial set based on Largest Sum of Entries (LSE)

The result of using the LSE strategy is that the initial set of demand vectors will consist of only one demand vector, the demand vector for which the sum of all entries are the maximum. That is, the initial subset of demand vectors to be used in the IPEA is

$$D_0 = \left\{ d^q : \sum_{k \in K} d_k^q = \max_{t \in \mathcal{T}} \left\{ \sum_{k \in K} d_k^t \right\} \right\}$$

Initial set based on Commodity Supremum Set (CSS)

A good approximation of the convex hull of demand vectors could be to consider a subset of demand vectors, where each demand vector has at least one entry that is the maximum among the corresponding entries over all the demand vectors. Let us index such a subset of demand vectors by the index set $\mathcal{T}_{\text{CSS}} \subseteq \mathcal{T}$. Then for all $t \in \mathcal{T}_{\text{CSS}}$ there exists a k such that

$$d_k^t = \max_{s \in \mathcal{T}} \{d_k^s\}$$

Note that it is possible for a demand vector d^t, with $t \in \mathcal{T}_{\text{CSS}}$, to have more than one maximal entry k. Therefore, $|\mathcal{T}_{\text{CSS}}| \leq |K|$. The initial subset of demand vectors to be used in the IPEA is therefore

$$D_0 = \{d^t : t \in \mathcal{T}_{\text{CSS}}\}$$

Initial set based on Gomory-Hu Dominant Tree (DT)

The following approach for selecting an initial subset of demand vectors is based on results from Gomory and Hu [GH61]

Definition 5.8 ([GH61]). *Let $J = (V, K)$ be a demand graph of a set of non-simultaneous single commodity demand requirements. Then the maximum spanning tree $\Gamma = (V, K_\Gamma)$ with $K_\Gamma \subseteq K$ is called a dominant requirement tree.*

It can be shown that a dominant requirement tree Γ is flow equivalent to the original graph G, i.e. the maximum flow between any node pair $u, v \in V$ for the graph G will be equal to the maximum flow between the corresponding nodes in Γ. The result is, therefore, that the network design problem for a set of single commodity demand requirements can be solved by taking into account only the single commodity demands that constitute Γ. That is, the number of single commodities that need to be considered is limited to $|V| - 1$.

For the survivable network design problem we consider non-simultaneous multiple demand vectors, but with each demand vector having simultaneous demand requirements. Therefore, in order

to take advantage of a dominant requirements tree within this context, we define the following single demand vector $\tilde{d}$ that has non-simultaneous demand entries, $\tilde{d}_1, \tilde{d}_2, \ldots, \tilde{d}_{|K|}$. For the purpose of assigning values to $\tilde{d}$ we consider the set of demand vectors indexed by $\mathcal{T}_{\text{CSS}}$ as defined in the proceeding section, such that for each $k \in K$:

$$\tilde{d}_k = \max_{t \in \mathcal{T}_{\text{CSS}}} \{d_k^t\}$$

The next step in the approach is to construct a maximal spanning tree by using the vector $\tilde{d}$. The resulting maximum spanning tree, denoted by $\Gamma = (V, K_\Gamma)$, is also a dominant requirements tree since $\tilde{d}$ is defined as a vector with non-simultaneous demand entries. A set of non-simultaneous demand vectors $\tilde{d}^1, \tilde{d}^2, \ldots, \tilde{d}^{|K_\Gamma|}$, indexed by the set $\mathcal{T}_{\text{DT}}$, can now be constructed from $\tilde{d}$ such that for each $t \in \mathcal{T}_{\text{DT}}$ and $k \in K_\Gamma$:

$$\tilde{d}_k^t = \left\{ \begin{array}{ll} \tilde{d}_k & \text{if} \quad t = k; \\ 0 & \text{if} \quad t \neq k. \end{array} \right\} \tag{5.60}$$

The initial subset of demand vectors is accordingly

$$D_0 = \{d^t : t \in \mathcal{T}_{\text{DT}}\}$$

The anticipated advantage of using the DT approach is that the problem data is now defined in terms of the number of nodes in the network and not in terms of the number of original demand sets in D. Secondly, solving the survivable network design problem with multiple demand vectors, where the vectors only contain a single demand entry, is expected to be easier than solving the problem with dense multiple demand vectors.

Initial set based on Extended DT (EDT)

An alternative to using single commodity non-simultaneous demand vectors as described in the preceding section, is to use the original demand vectors d^t that contributed to the creation of $\tilde{d}$, which in turn was used to construct the dominance requirements tree $\Gamma = (V, K_\Gamma)$. Recall that for each $k \in K$

$$\tilde{d}_k = \max_{t \in \mathcal{T}_{\text{CSS}}} \{d_k^t\}$$

In addition define $\Upsilon \in \mathbb{Z}^{|K|}$ such that for each $k \in K$

$$\Upsilon_k = \left\{ t : d_k^t = \max_{q \in \mathcal{T}_{\text{CSS}}} \{d_k^q\} \right\}$$

Then the set of demand vectors to be used as an initial set is indexed by $\mathcal{T}_{\text{EDT}} = \{t \in \mathcal{T}_{\text{CSS}} : t = \Upsilon_k, \forall k \in K_\Gamma\}$. The initial set of multiple demand vectors to be used in the IPEA is therefore

$$D_0 = \{d^t : t \in \mathcal{T}_{\text{EDT}}\}$$

The anticipated advantage of using the EDT initial subset strategy will be that the problem data is defined in terms of the number of nodes in the network compared to the number of demand vectors in the original set D.

5.6 Summary

The use of domination theory as a problem reduction approach, was presented in Section 5.1. The main result was a domination checking algorithm for both the dynamic and static routing cases. By applying the domination checking algorithm it is possible to determine whether a given demand vector is being dominated by the remaining demand vectors under consideration. If this is the case, then the said demand vector can be discarded from the network design process. The domination checking algorithm for static routing problems is sufficient for the case where survivability is considered. This is, however, not the case for dynamic routing since the failure states included as part of the path based formulation has a different meaning within the context of domination. An alternative approach for dominance checking that takes survivability into account has, however, been suggested. No computational results are presented in this thesis for this approach, but in theory the approach appears to be computationally feasible.

The algorithmic contributions made in this thesis is set within the framework of the branch-and-cut approach as described in Section 5.2. Two heuristics, the SNDP-K-Opt heuristic and the Edge-Flow heuristic, have been proposed in Section 5.3 for generating primal solutions within the branch-and-cut approach. It should be noted that these heuristics are not exclusively for the survivable network design problem with multiple demand vectors. They can be applied to the single demand vector network design problem as well. Furthermore, the SNDP-K-Opt heuristic is dependent on a few parameters that might have a significant influence on the performance of the heuristic. For instance, in the case of the solution construction part of the SNDP-K-Opt heuristic, called KOCON, an indicator level Ψ needs to be supplied. Since there is no clear indication what this indicator level should be, a value was decided on that showed to be promising for most problem instances. Some other parameters required for applying the primal heuristics are for instance parameters specifying the frequency at which the heuristics should be applied during the processing of the branch-and-cut approach. Since there is no theoretical evidence of what these parameters should be, values were chosen based on practical experience. The specific parameter settings for the heuristics are provided as part of the computational results chapter, Chapter 6.

Separation strategies were presented in Section 5.4 for both the dynamic and static routing cases. For dynamic routing a strengthening procedure was suggested and for the static routing

case a robust separation formulation was proposed. The anticipated advantage of the strengthening approach is that less violated metric inequalities will be added to the problem formulation, which in turn will have a positive effect on computing times since the LP relaxations at the branch-and-bound nodes should be easier to solve. The downside, however, might be that computing time will be sacrificed to perform the strengthening. Different strategies were applied for ordering and selecting subsets of demand vectors during the strengthening procedure in an attempt to balance time spent on strengthening and time spent on the actual branch-and-cut. The anticipated advantage of applying the robust separation approach is that the LP problem being solved will not grow that rapidly in the number of constraints during path generation, since the capacity constraints are not duplicated for each demand vector as in the case of the ordinary static routing separation approach. The price to be paid, however, for applying robust separation is that the LP being solved is augmented with additional variables corresponding to the number of demand vectors considered for the problem.

In Section 5.5 the IPEA was introduced as an approach for improving scalability with respect to the number of demand vectors considered for the survivable network design problem. It is anticipated that by only considering an initial subset of demand vectors solutions might be generated much faster, and by employing the solution improvement part of the SNDP-K-Opt heuristic, KOIMP, these solutions could be modified to support all the demand vectors in the problem without too much effort. Three strategies for selecting the initial subset of demand vectors were proposed, and comparative results for these strategies are provided as part of the computational results given in Chapter 6.

Chapter 6

Computational Results

In this chapter the real-world applicability of the results of this study is demonstrated by means of empirical tests. The results presented here are for the contributions made by this study towards solving the survivable network design problem, as was outlined in Chapter 5. In summary, the contributions made were towards implementing a heuristic for generating primal solutions for the branch-and-cut as presented in Section 5.3, improving the separation of metric inequalities as presented in Section 5.4, improving of scalability through IPEA as presented in Section 5.5, and finally reducing the problem complexity through domination among multiple demand vectors as presented in Section 5.1.

6.1 Implementation Details

The implementation of the theory presented in this thesis was done in C++ and is based on the network planning system DISCNET [Wes00]. The branch-and-cut approach was implemented by making use of the callback functionality provided by the commercial solver CPLEX [CPL]. There are several callback functions available to CPLEX users. For instance, a status callback function may be used allowing the user to query the status of the optimisation process, as well as to report on the optimisation process itself. A cut callback function is available providing CPLEX users with the ability to separate user defined cuts by making use of solution information from the branch-and-bound nodes. Especially useful is a heuristic callback function providing CPLEX users with a mechanism for employing their own heuristics in an attempt to generate feasible solutions or to improve on existing solutions. For a complete outline of CPLEX's functionality, the reader is advised to consult the CPLEX reference manual [CPL].

The basic code making use of CPLEX callback functions that allows the computation of the survivable network design problem with multiple demand matrices, including dynamic and static routing, is referred to as the **default implementation** of the branch-and-cut within CPLEX. For the discussion of results in subsequent sections any reference to CPLEX in graphs or tables implies

the default implementation of the branch-and-cut within CPLEX.

The usefulness of the contributions made in this thesis is illustrated by comparing the suggested approaches, namely the application of heuristics, the improving of separation strategies, the application of the IPEA, and the utilising of domination theory, with the default implementation of the branch-and-cut within CPLEX.

6.1.1 Separation of metric inequalities

During the processing of a branch-and-bound node by CPLEX, several calls are made to the cut callback function. It is within this cut callback function that the code responsible for separating metric inequalities is executed. The cut callback function is repeatedly called at a branch-and-bound node until no more violated inequalities are found. On completion of the cut callback function the violated inequalities, including violated metric inequalities, are added to the master problem.

6.1.2 Heuristics

For purposes of testing the proposed heuristics EFCON, KOCON, and KOIMP, the heuristic callback function capabilities of CPLEX has been employed. The application of the proposed heuristics depends on mainly four parameters: *min_depth*, *max_depth*, *depth_interval*, and *max_time*. The parameters *min_depth* and *max_depth* specify the allowable range of depth in the branch-and-bound tree for which the heuristics may be employed. The parameter *depth_interval* specifies the depth interval at which the heuristics are allowed to be called. For $depth\_interval = 2$, for instance, the heuristics are applied at every second depth level of the branch-and-bound tree. The parameter *max_time* imposes a maximum limit in seconds on the running time of the heuristics. For generating the subsequent results the following settings were used: $min\_depth = 0$, $max\_depth = \infty$, $depth\_interval = 20$, and $max\_time = 120$. It should be noted that these settings were based on experience. It might be that for different problem instances different parameter settings could improve computing times.

6.1.3 Generating cutset inequalities

Recall from Section 4.3.2 that a cutset inequality for a set of nonsimultaneous multiple demand vectors is of the form

$$\sum_{e \in E(W,\bar{W})} y_e^* \geq \max_{t \in T} \left\{ \sum_{k \in K(W,\bar{W})} d_k^t \right\} \tag{6.1}$$

with $W \subseteq V$ a subset of V and $\bar{W} = V \setminus W$ its complement. The set $E(W, \bar{W})$ denotes the set of edges having one endnode in W and the other in $\bar{W}$ and $K(W, \bar{W})$ the set of commodities having an endnode in W and the other in $\bar{W}$.

Although cutset inequalities are considered to be special instances of metric inequalities, they have not been generated through the application of separation routines in this work. Instead, cutset inequalities are generated and added to the problem before commencing with the branch-and-cut. Since there is an exponential number of combinations for choosing W and $\bar{W}$, an approach has been followed whereby W is chosen to consist out of no more than two nodes. That is, a cutset inequality is generated for each combination of W consisting of one node and with $\bar{W}$ its complement, as well as for each combination of W consisting of two nodes, with $\bar{W}$ its complement. It has been found from empirical tests that by generating cutset inequalities with W having more than two nodes considerably influences computing times.

6.1.4 Column generation and path initialisation

A column generation approach is followed in the separation of metric inequalities in both dynamic and static routing models. It is, however, necessary to find an initial set of paths that will allow the separation LP to find at least one feasible solution.

For the case where no survivability requirements are considered, it suffices to solve a shortest path problem for each commodity $k \in K$ by considering the supply graph $H = (V, E)$. A simple approach would be to apply Dijkstra's algorithm [Dij59] where the cost on each edge $e \in E$ could be taken as the geographical length of the edge. Alternatively, instead of generating only a single shortest path per commodity, it might be beneficial in some cases to generate a number of shortest paths, thereby reducing time spent on column generation.

When considering survivability whereby a diversification parameter leads to diversified paths, a more effective way of generating initial paths could be to make use of an algorithm like the one suggested by Suurballe [Suu74]. Solving the algorithm by Suurballe produces $\lceil 1/\delta_k \rceil$ number of disjoint shortest paths for each $k \in \mathcal{K}$.

An important aspect of managing the enormous number of path variables, in addition to column generation, is the removal of unused path variables. An approach followed in this implementation is to remove path variables from the LP formulation whenever it is found that they have not been part of the LP basis for a number of simplex iterations. For example, when it is found that a path variable has not entered the basis within the last 30 iterations, it is removed from the formulation. It is difficult to determine the optimal number of simplex iterations after which a variable must be removed. Therefore, some value other than 30 might have a different effect on computing times. If a path variable is removed to soon, unnecessary calculations have to be performed to re-introduce the variable back into the basis.

6.2 Data Sets

The data sets used in this work are divided into three groups. The first group consists of data sets with single demand vectors, the second of data sets that comprises randomly generated multiple demand vectors, and the third group of data sets that contains multiple non-simultaneous demand vectors based on measurements obtained from an operational network.

6.2.1 Single demand vectors

The data set containing single demand vector data, hereafter referred to as the SNDLIB data set, was obtained from [snd05]. The data can be accessed via the internet address provided by the reference [snd05] and several utilities are available on the web-site for testing the feasibility of solutions, adding new problem instances, obtaining resources on network design, etc. The web-site provides functionality for filtering problem instances on a variety of criteria. For instance, for purposes of performing computational tests on the proposed heuristics in Section 5.3, the following selection criteria were applied:

- Demand model: Undirected demands (U)
- Link model: Undirected links (U)
- Link capacity model: Explicit capacities (E)
- Fixed-charge model: No fixed-charge cost (N)
- Routing model: Continuous (C)
- Admissible path model: All paths (A)
- Hop-limit model: No hop-limits (N)
- Survivability model: No survivability (N)

Table 6.1 provides some statistics on the problem instances. Relaxation of survivability requirements were, however, necessary to ensure computational feasibility. Approximately 50% of the demands within each problem instance have been subjected to diversification requirements of having at least two disjoint paths. The reason for not imposing a diversification requirement on all of the demands is to limit processing times.

6.2.2 Random multiple demand vectors

The problem instances nobel-germany and nobel-us from the SNDLIB were used as a basis for generating random multiple demand vectors. These two problem instances were selected purely

Instance	|V|	|E|	|K|
cost266	37	57	1332
dfn-bwin	10	45	90
dfn-gwin	11	47	110
di-yuan	11	42	22
newyork	16	49	240
nobel-eu	28	41	378
nobel-germany	17	26	121
nobel-us	14	21	91
norway	27	51	702
pdh	11	34	24
ta1	24	55	396
ta2	65	108	1869

Table 6.1: SNDLib statistics

for the sake of computational feasibility. Both problem instances can be solved within a matter of minutes, proving it attainable to solve it in reasonable time when augmented with multiple demand vectors.

Demand vectors for both the nobel-germany and nobel-us problem instances were created randomly by using the following:

$$\hat{d}_k^i = \sigma_k^i d_k$$

where d_k^i is the k-th demand for a demand vector d^i generated for a scenario i by using the original demand vector d from the problem instances nobel-germany and nobel-us. The parameters σ_k^i were sampled from the uniform distribution $U(0.8, 1.2)$.

The randomly generated data sets will be referred to in the remainder of this chapter as SNDLIB-RANDOM.

6.2.3 Operational multiple demand vectors

The first data set containing operational multiple demand vectors, hereafter referred to as the DFN data set, was provided by DFN-Verein, the German IP network provider for universities and research institutes. The DFN data set contains three problem instances, namely DFN-12, DFN-28, and DFN-288, having respectively 12, 28, and 288 demand vectors. The problem instance with 12 demand vectors are measurements obtained over a period of one year representing monthly traffic, the problem instance with 28 demand vectors are daily measurements obtained for the month of February, and the problem instance with 288 demand vectors are measurements obtained over a period of one day representing traffic measured within five minute intervals.

The second data set, called the EIBONE data set, has its origin from the German project, Efficient Integrated Backbone [eib]. The EIBONE data set is based on the DFN data set and contains 6 problem instances, namely EIBONE17-12, EIBONE17-28, EIBONE17-288, EIBONE50-12, EIBONE50-28, and EIBONE50-288. The EIBONE17-X instances are based on a topology

Instance	|V|	|E|	|K|
DFN-12	32	64	168
DFN-28	32	64	190
DFN-288	32	64	174
EIBONE17-12	17	34	72
EIBONE17-28	17	34	72
EIBONE17-288	17	34	65
EIBONE50-12	50	100	144
EIBONE50-28	50	100	167
EIBONE50-288	50	100	163

Table 6.2: Operational data statistics

having 17 nodes and the EIBONE50-X instances are based on a topology having 50 nodes. Similarly to the DFN data set, the postfixes -12, -28, and -288 indicate the relevant number of demand vectors present. That is, 12 monthly, 28 daily, and 288 vectors measured in 5 minute intervals.

Statistics on both the DFN and EIBONE data sets are presented in Table 6.2. Modifications to the original problem instances were, however, necessary to improve computational feasibility. Minimum cutoff values of 2 Mbps and 1 Mbps were imposed on the DFN and EIBONE data sets respectively, resulting in a reduced number of commodities. Furthermore, only about 50% of the demands within each problem instance have been subjected to diversification requirements of having at least two disjoint paths.

6.3 Heuristics

The SNDLIB data set was used in the computational tests involving the EFCON, KOCON, and KOIMP heuristics. For a more compact representation only results for the heuristics and combination of heuristics, that proved to have a significant influence on computing times, are presented here. A detailed listing of the results is presented in Table B.1, as part of Appendix B. The results are presented in two parts. Firstly, Figure 6.1 shows the results for problem instances which could not be solved to optimality within a two hour time limit, and secondly, Figure 6.2 shows the results for problem instances for which the branch-and-cut terminated with an optimal solution within a two hour time limit.

From the graph in Figure 6.1 it can be observed that for most problem instances either the use of KOCON+KOIMP, EFCON+KOIMP, or KOIMP, contributed to reduced running times. For instance, the result obtained by applying the KOCON+KOIMP heuristic on the cost266 problem instance, is a solution with an integrality gap of 43%. For the same problem instance the default CPLEX implementation was unable to calculate any feasible solution within the 2 hour time limit. It should be noted that due to the definition of the integrality gap as (UB – LB)/LB, the percentage

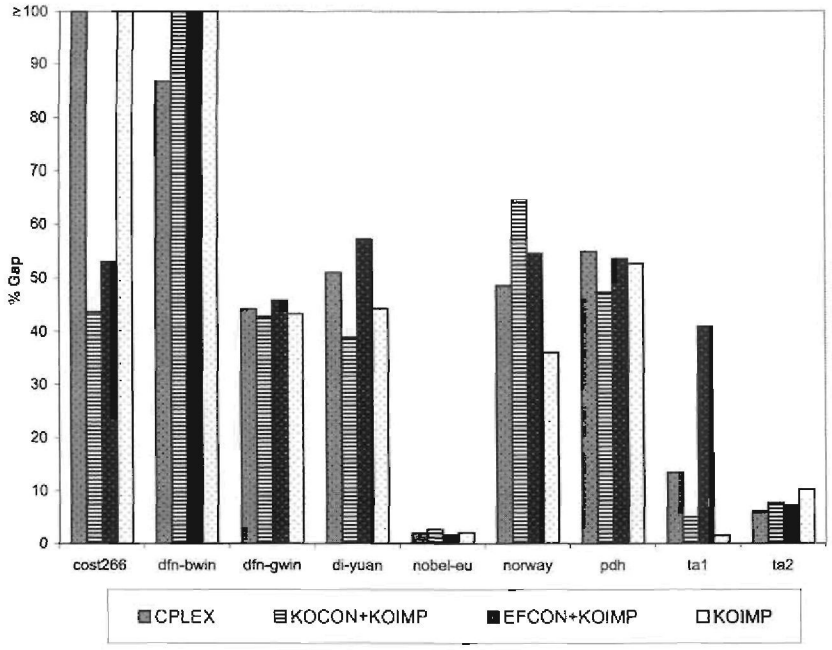

Figure 6.1: SNDLib results - percentage gap

gap when no feasible solution exists is infinite, since UB $= \infty$. Therefore, the maximum value of the vertical axis in all of the graphs showing percentage gap, such as Figure 6.1, is shown to be ≥ 100.

It is only for the dfn-bwin and ta2 problem instances for which no improvements were observed for any of the heuristics, or combination of heuristics. It is, however, not to say that the application of the heuristics for these instances was a total failure. The graph in Figure 6.3 supports the argument by showing that the KOIMP heuristic was successful in finding a better initial solution compared to the first few solutions found without applying the heuristic.

Unfortunately the results for smaller problem instances are not that promising for the heuristics, as is evident from Figure 6.2. It is suspected that the proposed heuristics are successful in generating good initial feasible solutions during the optimisation approach. Some improvement may, however, be obtained by trying different values for the parameters *min_depth*, *max_depth*, *depth_interval*, and *max_time* as described in Section 6.1.2.

6.4 Separation of Metric Inequalities

Improvements towards the separation of metric inequalities were suggested with regard to both dynamic and static routing. In the dynamic routing case a strengthening procedure was proposed and in the case of dynamic routing a robust separation strategy was suggested.

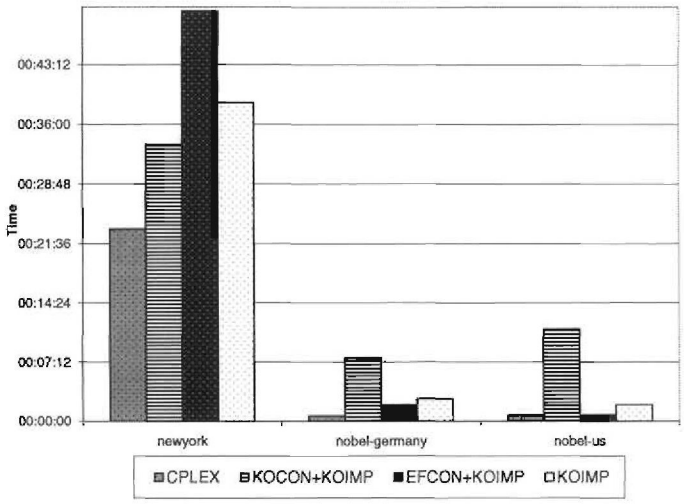

Figure 6.2: SNDLib results - execution time

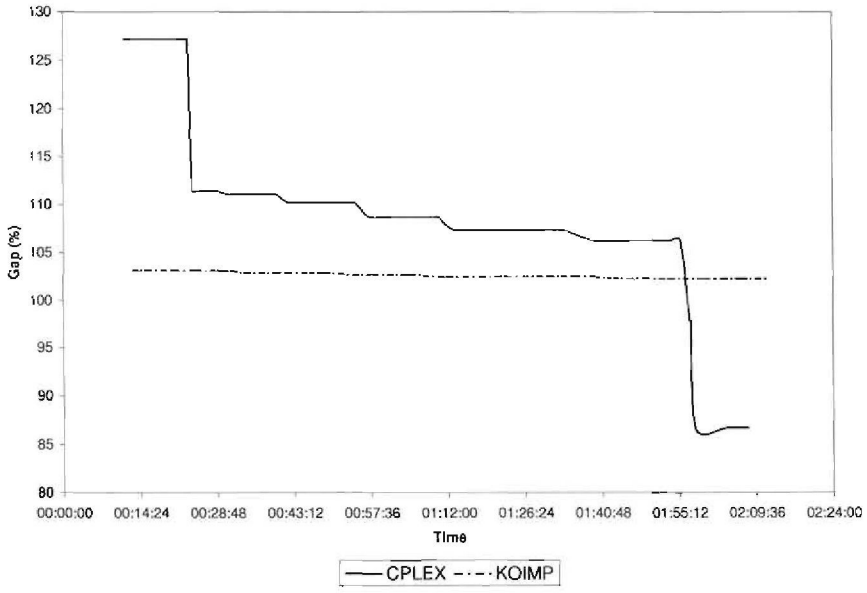

Figure 6.3: Solution progress for dfn-bwin

Instance	# Vectors	CPLEX	STR-ALL	MAX-SUM-TOP2	MAX-SUM-TOP5	MAX-SUM-TOP10	MAX-DEM-TOP2	MAX-DEM-TOP5	MAX-DEM-TOP10
nobel-germany	5	00:03:13	00:02:47	00:03:21	00:03:31	00:03:35	00:03:21	00:03:26	00:03:20
nobel-germany	10	00:06:01	00:13:54	00:05:33	00:06:16	00:06:19	00:05:33	00:05:41	00:05:31
nobel-germany	20	00:08:29	00:15:37	00:07:55	00:07:51	00:09:32	00:07:57	00:08:07	00:10:14
nobel-germany	40	00:13:58	00:16:36	00:14:15	00:12:31	00:08:34	00:14:23	00:12:17	00:08:55
nobel-germany	80	03:38:57	01:01:00	01:19:56	01:56:32	02:26:35	02:23:45	02:02:34	01:39:42
nobel-germany	160	01:14:49	02:02:21	01:42:05	01:39:48	00:59:46	00:56:39	01:10:32	01:23:10
nobel-germany	320	02:41:48	03:03:32	02:00:12	03:31:51	03:12:21	02:45:58	02:42:08	04:47:04
nobel-germany	640	06:52:29	11:34:21	12:01:17	08:37:14	11:42:48	06:16:20	08:13:21	11:57:30
nobel-us	5	00:01:18	00:01:53	00:01:19	00:01:16	00:01:18	00:01:17	00:01:18	00:01:16
nobel-us	10	00:03:21	00:03:14	00:03:22	00:03:19	00:03:19	00:03:22	00:03:28	00:03:24
nobel-us	20	00:07:26	00:05:43	00:07:59	00:07:49	00:05:43	00:07:46	00:07:44	00:07:05
nobel-us	40	00:11:04	00:12:26	00:11:03	00:09:34	00:11:57	00:11:12	00:12:15	00:10:14
nobel-us	80	00:27:27	00:25:40	00:22:00	00:23:41	00:19:00	00:36:12	00:39:22	00:27:49
nobel-us	160	00:55:27	01:16:10	00:46:36	01:00:03	01:01:04	00:51:56	00:51:34	00:47:08
nobel-us	320	01:31:28	01:32:46	01:59:25	01:33:10	01:33:28	01:17:27	01:37:18	01:40:19
nobel-us	640	04:13:43	03:31:39	04:37:19	04:10:38	03:13:12	03:41:43	02:38:21	05:11:58

Table 6.3: Strengthening of metric inequalities - computing times

6.4.1 Dynamic routing

For the purpose of demonstrating the influence of strengthening the metric inequalities in the case of dynamic routing, the SNDLIB-RANDOM data set containing the problem instances nobel-germany and nobel-us, was used. The data sets within each problem instance consist of 5,10,20,40,80,160,320, and 640 non-simultaneous demand vectors. The results for this section were accordingly generated by solving SNDP-DR(D, δ) to optimality for the different problem instances.

The results demonstrate the improvements obtained in computing times when considering the different strengthening strategies during separation as described in Section 5.4. The entire set of results is displayed in Table B.2 and Table B.3 as part of Appendix B. Table 6.3 is an extract of the results and gives the times for the different strengthening strategies over the nobel-us and nobel-germany problem instances. The minimum computing times for each row in Table 6.3 are highlighted. From the results it is not clear which strategy is superior. There is no single strategy that consistently performs well for different problem instances.

A very important point to take note of, however, is the reduction obtained in the number of metric inequalities eventually added to the problem during the branch-and-cut approach. The results displayed in Figure 6.4 and Figure 6.5 for the nobel-germany and nobel-us instances respectively, shows the number of inequalities that have been added to the problem over a range of demand vectors. The default implementation of metric separation in CPLEX is compared against all of the strengthening strategies. For the default implementation in CPLEX the number of inequalities increases almost exponentially with an increase in the number of demand vectors, whereas for the strengthening strategies the number of inequalities remains almost constant. Although it is expected that there will be a performance gain with less metric inequalities being added to the

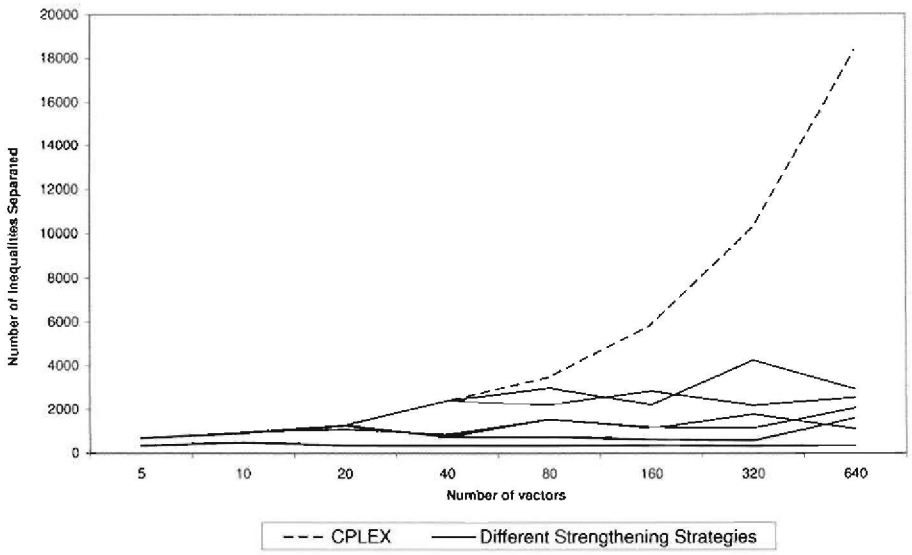

Figure 6.4: Strengthening of metric inequalities for nobel-germany

problem, the time spent on strengthening seems to have a negative effect on computing times. Table 6.3 indeed does not reflect a reduction in computing times proportional to the reduction in the number of metric inequalities.

6.4.2 Static routing

For the purpose of comparing the robust separation approach to the standard way of doing static separation as described in Section 5.4.2, the SNDLIB-RANDOM data set was used. However, due to the difficulty in solving the static routing case the number of data sets was limited to the range 2, 4, and 8.

The graphs depicted in Figure 6.6 and Figure 6.7 were generated by solving SNDP-SR(D, δ) for both the nobel-us and nobel-germany problem instances to optimality for the range of demand vectors 2, 4, and 8. As a first observation from the graphs, the extremely high computing times for relatively few demand vectors should be noted. Solving both the nobel-us and nobel-germany problem instances to optimality for a single demand vector takes no more than one minute (see Table B.1). However, solving the nobel-us and nobel-germany problem instances up to optimality with two demand vectors increases computing times significantly. For instance, it took approximately six hours to solve the nobel-us problem instance to optimality when using the default static routing separation approach, and approximately thirteen hours to solve it to optimality by using the robust separation approach. Similar results have been obtained for the nobel-germany problem instance where it required approximately five hours for the robust separation approach and seven hours for the default static routing separation approach to obtain an optimal solution. It

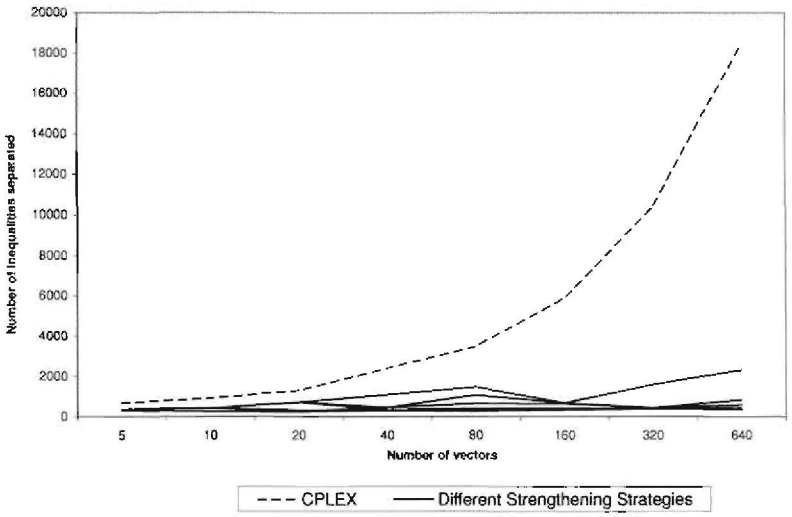

Figure 6.5: Strengthening of metric inequalities for nobel-us

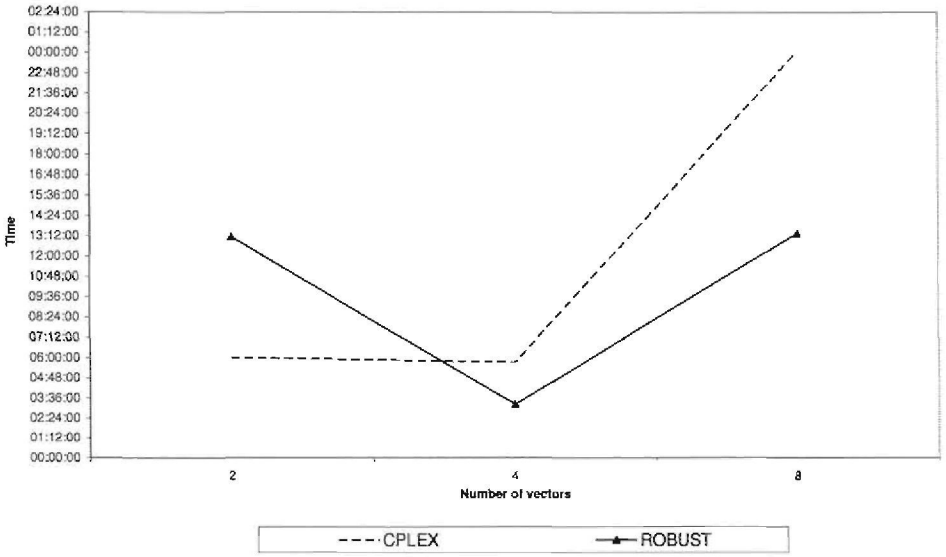

Figure 6.6: Robust separation of metric inequalities for nobel-us

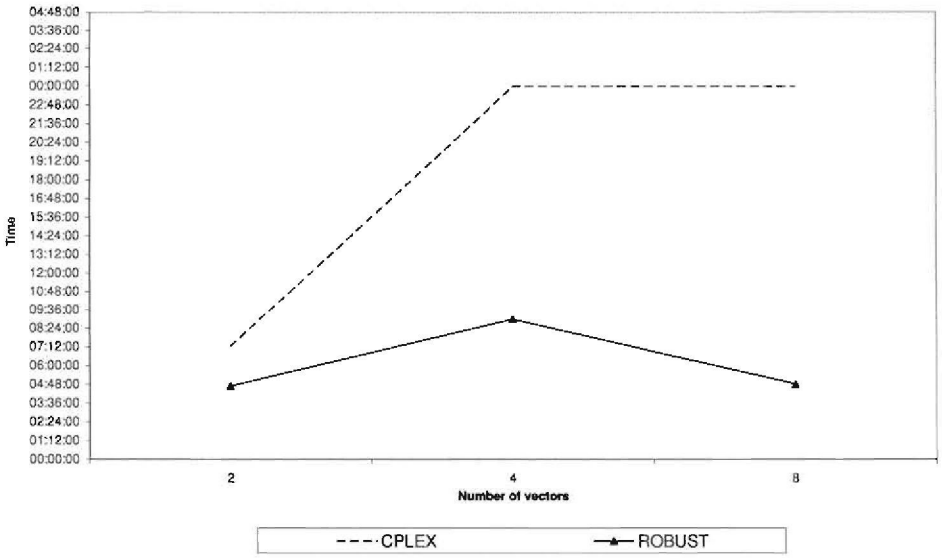

Figure 6.7: Robust separation of metric inequalities for nobel-germany

is clear that, compared to the dynamic routing case, the survivable network design problem with static routing is much harder to solve. The main reason for this inability to effectively solve the survivable network design problem with static routing can attributed to the large number of metric inequalities that are eventually being added to the problem during the branch-and-cut approach. This can be seen by looking at the complete set of results listed in Table B.4, Appendix B.

It is, however, promising to observe that significant reduction in computing times can be obtained by applying the robust separation approach instead of the default static separation approach. It should also be noted that a maximum time limit of 24 hours was imposed on running times. Therefore, the nobel-germany problem instances containing the 4 and 8 demand vectors respectively, were not solved up to optimality. In those cases the relative improvement of the robust separation approach is even more significant compared to the default static separation approach.

6.5 Iterative Polyhedron Expansion (IPEA)

6.5.1 IPEA scaling test

The SNDLIB-RANDOM data set, containing the problem instances nobel-germany and nobel-us, was used for testing the scalability of the IPEA approach. The graphs in Figure 6.8 and Figure 6.9 show the running time versus the number of demand vectors for the nobel-germany and nobel-us problem instances respectively. All problem instances were solved to optimality and the complete set of results is listed in Table B.5 as part of Appendix B.

From the results it is clear that the initial subset strategy that outperformed all the others were the Largest Sum of Entries (LSE) whereby the IPEA approach commences with only the demand

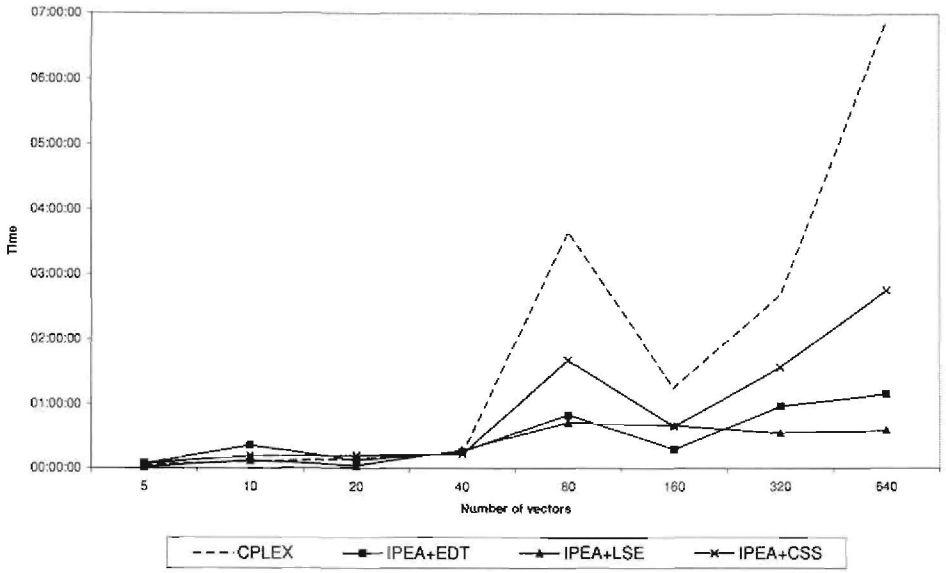

Figure 6.8: IPEA scaling test for nobel-germany

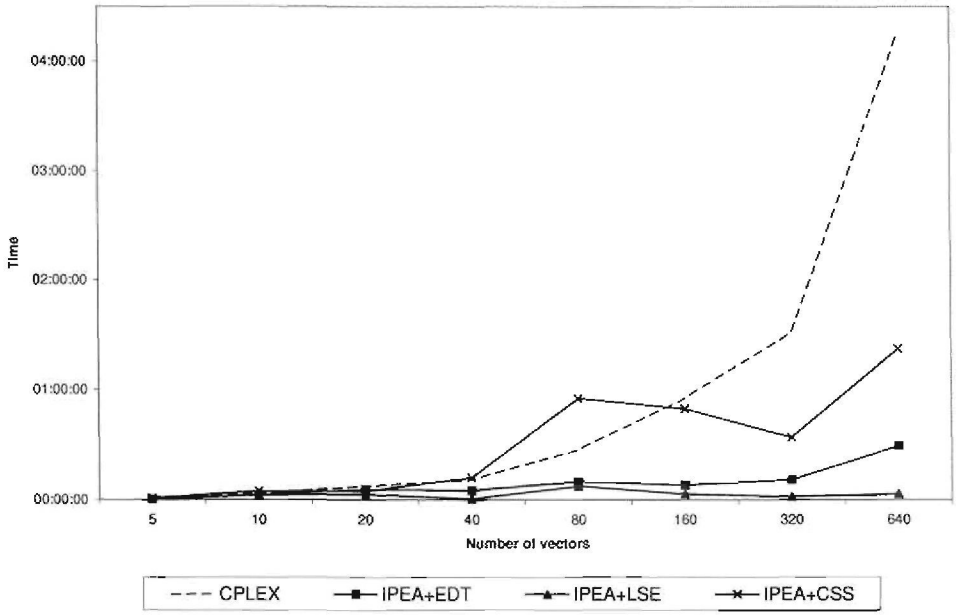

Figure 6.9: IPEA scaling test for nobel-us

Instance	# Vectors	IPEA+EDT	IPEA+LSE	IPEA+CSS
nobel-us	5	80%	20%	100%
nobel-us	10	70%	40%	100%
nobel-us	20	65%	25%	100%
nobel-us	40	33%	3%	95%
nobel-us	80	20%	14%	66%
nobel-us	160	18%	4%	45%
nobel-us	320	13%	1%	26%
nobel-us	640	22%	1%	15%
nobel-germany	5	100%	40%	100%
nobel-germany	10	90%	90%	100%
nobel-germany	20	85%	25%	100%
nobel-germany	40	43%	35%	95%
nobel-germany	80	31%	26%	76%
nobel-germany	160	16%	24%	58%
nobel-germany	320	24%	23%	34%
nobel-germany	640	13%	13%	17%

Table 6.4: Percentage of demand vectors finally added to problem formulation by IPEA

vector for which the sum of entries is a maximum. There are some instances for which the Extended Dominant Tree (EDT) approach proved to be more successful in approximating the initial set of demand vectors to be used.

It is important to take note of the total number of demand vectors eventually added to the problem formulation as a result of applying IPEA. Table 6.4 reports the percentage of demand vectors that were in total added to the problem formulation. The most significant results are for the LSE initial subset strategy where, for the larger data set containing 640 demand vectors, a total of only 1% of demand vectors were included in the final formulation of the nobel-us problem instance and a total of only 13% of demand vectors were included in the final formulation of the nobel-germany problem instance. Said differently, the size of the dominant set of demand vectors for the nobel-us problem instance is 1% of the total set of demand vectors and the size of the dominant set of demand vectors for the nobel-germany problem instance is 13% of the total set of demand vectors. From Table 6.4 it is clear that for the larger data sets, only a small percentage of demand vectors are needed for inclusion into the network design process since the remaining demand vectors are being dominated.

6.5.2 IPEA with operational data

Computational results for the DFN and EIBONE data sets are provided for both dynamic and static routing. The complete listing of results for the dynamic and the static routing cases can be found in Table B.6 and Table B.7 respectively, as part of Appendix B. Figure 6.10 shows the results for the IPEA with initial subset strategies (EDT) and (LSE) for the dynamic routing case. A maximum time limit of 2 hours was imposed for the dynamic routing case. For the static routing case, however, a maximum running time of 4 hours was allowed. The complexity

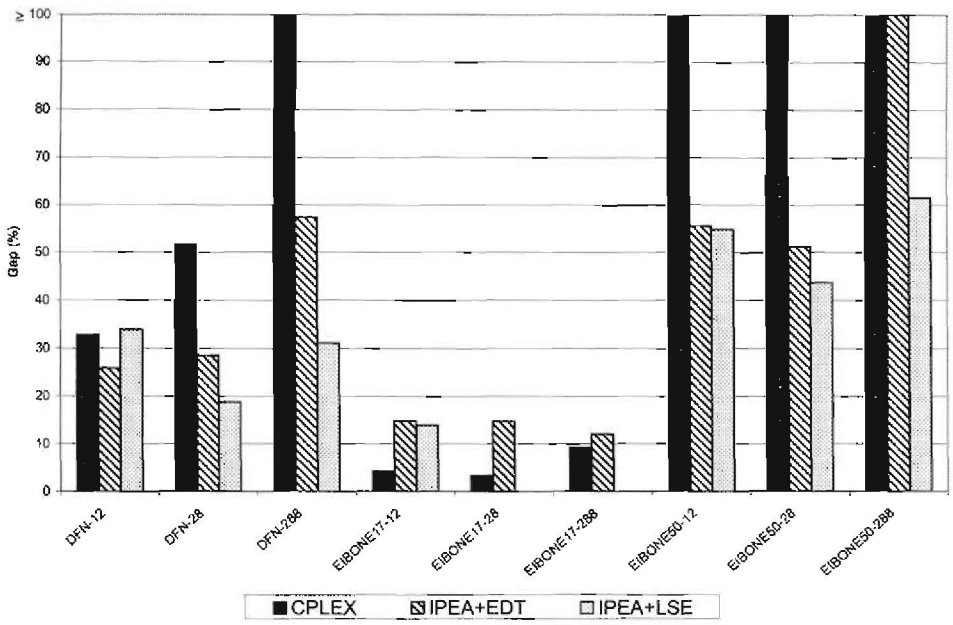

Figure 6.10: IPEA with dynamic routing

of solving the survivable network design problem with static routing is clearly observed from the results as shown in Figure 6.11 where solutions could only be found for the EIBONE17-X data sets. Unfortunately, the only instance where improvements was achieved by employing the IPEA, was for the EIBONE17-288 problem instance.

A very important observation that can be made by inspecting the complete results for dynamic routing listed in Table B.6, in Appendix B, is the small number of demand vectors finally included as part of the problem formulation. For example, by examining the results for problem instance DFN-288, it is observed that only 37 out of the 288 demand vectors, i.e. 12.8%, were taken into account by applying the LSE initial subset strategy. Similarly, for the EIBONE50-288 problem instance, only 9 out of 288 demand vectors, i.e. 3%, were used in finding solutions for the survivable network design problem.

6.6 Domination

The DFN and EIBONE data sets were used for the purpose of applying the proposed domination checking algorithms. Note that the diversification requirements for the DFN and EIBONE data sets were relaxed, since the domination algorithm for dynamic routing is only relevant for problems without any diversification requirements.

The results obtained by applying the domination checking algorithm for both dynamic and static routing cases, are presented in Table 6.5. The columns labeled *Dominated (%)* for both the dynamic and static routing, indicate the percentage of demand vectors that are dominated and

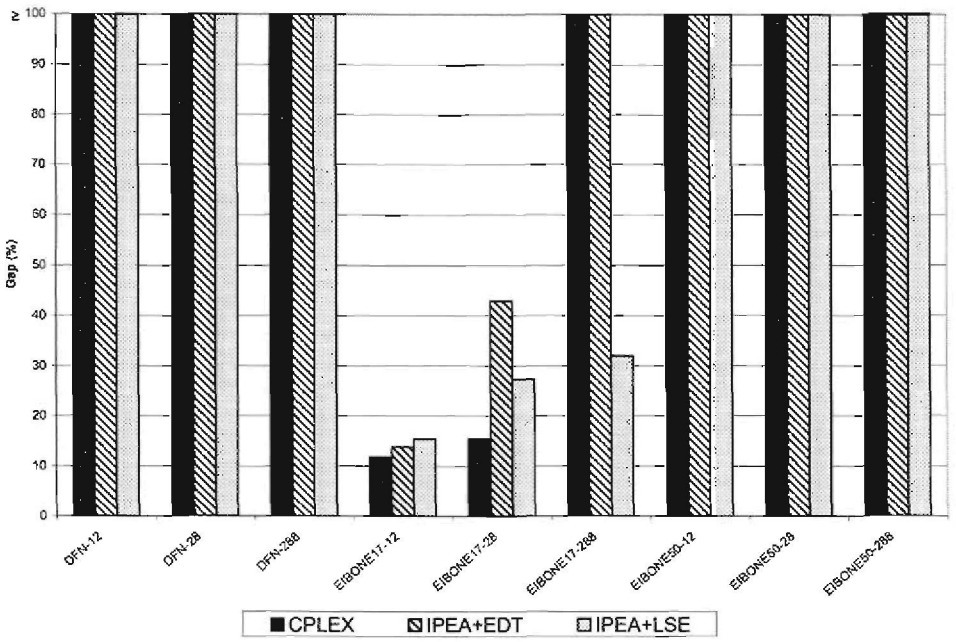

Figure 6.11: IPEA with static routing

Instance	Dynamic Routing		Static Routing	
	Dominated (%)	Time	Dominated (%)	Time
DFN-12	16.7	00:00:07	0	00:00:00
DFN-28	25.0	00:01:22	0	00:00:00
DFN-288	64.2	00:57:36	1	00:00:02
EIBONE17-12	25.0	00:00:02	0	00:00:00
EIBONE17-28	50.0	00:00:03	0	00:00:00
EIBONE17-288	78.1	00:00:27	16	00:00:00
EIBONE50-12	0.0	00:00:07	0	00:00:00
EIBONE50-28	17.9	00:00:47	0	00:00:00
EIBONE50-288	60.8	01:09:47	0	00:00:02

Table 6.5: Domination results

which can be discarded during the network design process.

The domination checking algorithm for dynamic routing was successful in identifying a large percentage of dominated demand vectors, for instance DFN-288: 64 %, EIBONE17-288: 78 %, and EIBONE50-288: 60 %. The algorithm, however, was less successful in finding any dominated demand vectors in the case of static routing. Only 1% of demand vectors were found to be dominated for the DFN-288 problem instance and only 16% of demand vectors were found to be dominated for the EIBONE17-288 problem instance. Although the computing times for doing dominance checking for some instances were considerably higher than the average, for instance about one hour for both the instances DFN-288 and EIBONE50-288, it is still worth while applying dominance checking prior to embarking on a network design exercise. The dominated set of demand vectors will never have to be part of the problem again, regardless of different problem parameters that might be considered later for the network design problem. To reinforce this statement recall that when a demand vector is being dominated by a set of demand vectors, called the dominant set, any capacity vector that will support the dominant set will also support the demand vector that is being dominated. Therefore, irrespective of the different topologies or cost functions being considered for different scenarios, the resulting network design will support all the demand vectors, including the demand vectors left out of the optimisation process due to being dominated.

Due to the inability of the domination checking algorithm to significantly reduce the number of demand vectors in the case of static routing, the remainder of this section will be devoted only to the dynamic routing case.

Computational results were subsequently obtained by using only the dominant set of demand vectors during the solution process of the network design problem. Due to the difficulty in obtaining optimal solutions for most of the problem instances, a time limit of 2 hours was imposed. The complete set of results is listed in Table B.8 as part of Appendix B. The graph depicted in Figure 6.12 is an extraction of the results and reveals what the effect is on computing times for solving the network design problem when the problem is reduced to contain only the dominant set of demand vectors. In the graph, the legend labeled CPLEX is used to indicate the default implementation of the branch-and-cut approach. The legend CPLEX+DOM is used to indicate the default implementation of the branch-and-cut approach applied to the reduced problems containing only the dominant set of demand vectors and the legend IPEA+DOM indicates the use of the IPEA with the LSE initial demand vector subset strategy applied to the network design problem with only the dominant set of demand vectors.

From the results in Figure 6.12 it is clear that by problem reduction, through the use of the dominant set of demand vectors, computing times are improved. By applying IPEA to the problem with the dominant set of demand vectors, further improvements may be observed for the larger problem instances such as DFN288, EIBONE17-288, and all of the EIBONE50-X instances. Im-

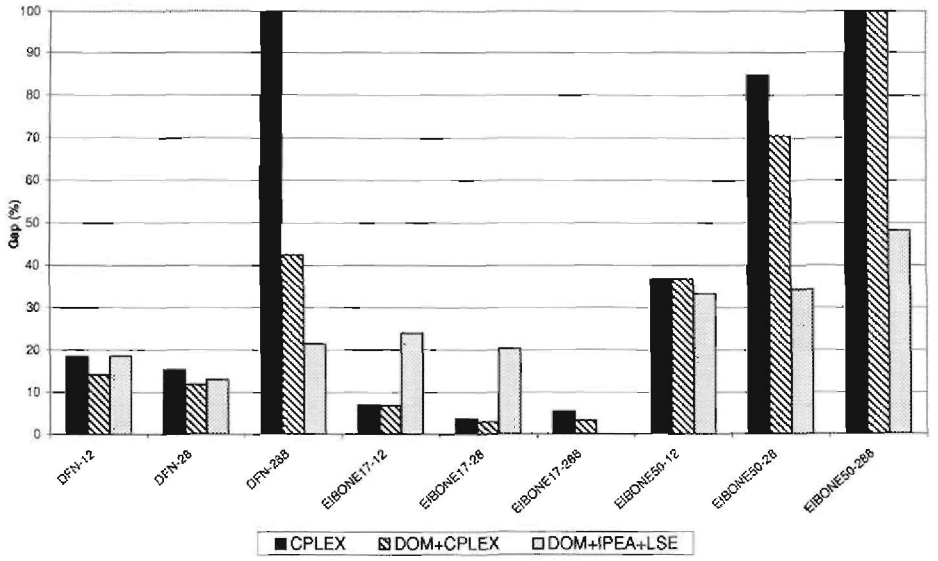

Figure 6.12: Effect of domination on network design

portant to note is that the default branch-and-cut implementation, labeled CPLEX, was unable to find any feasible solutions for the problem instances DFN288 and EIBONE50-288, whereas the IPEA computed feasible solutions with integrality gaps of 21% and 48% respectively. Less positive results are, however, observed for the smaller problem instances EIBONE17-12 and EIBONE17-28 in the case where IPEA was applied. This may be attributed to the fact that some time is spent on overheads in executing IPEA, making it less efficient when working with smaller problem instances.

Chapter 7

Summary and Conclusion

7.1 Thesis Summary

The survivable network design problem has posed many challenges for practitioners over the last decade. Although there have been some advances in both the theory of survivable network design and in computing technology, new modelling requirements have also evolved at a significant pace. Network operators require from network design tools to have the capability that will allow them to model communication networks in detail and have the capability to produce robust designs by taking into account uncertainty in the input data. Furthermore, survivability requirements have become a necessity and not a luxury in order to accommodate vital systems that are an integrated part of our lives. It has, therefore, been the objective in this thesis to address these typical requirements from a practical point of view, by making various contributions towards the design of survivable networks with demand uncertainty.

A technical background on communication networks was provided in Chapter 2. The main contribution of the chapter is to provide the context in which the research for this thesis has been done. Although there are numerous technologies that provide the building blocks of a heterogenous Internet, standard protocols allow messages to be transported from one interconnected network to the other. The Internet Protocol (IP) is one such protocol which is primarily used for facilitating packet switching in backbone networks. Therefore, the modelling approach followed in this thesis was to accommodate network design for IP networks.

Taking uncertainty into account with regard to the traffic demand requirements is the emphasis of Chapter 3. There are mainly two approaches for incorporating demand uncertainty into network design models: Firstly, practitioners might choose to follow a stochastic programming approach whereby uncertainty is represented by the realisation of possible scenarios. Associated with each scenario is a demand vector that has some probability of realising in future. The true power of stochastic programming comes from the feature that, if demand requirements are driven by some revenue model, the stochastic programming problem will optimise the expected profit but with

the risk of not satisfying some of the demand requirements. Alternatively, if satisfying all demand requirements is not negotiable, practitioners may turn to a robust network design approach. In this thesis, aspects of both modelling approaches feature. The routing part of the network design problem considered in this work, is based on a scenario formulation similar to that of stochastic programming, but with the difference that all demand vectors have equal probability of realising, and in addition, all demand requirements must be met. This kind of formulation has been adopted by many others and has been labeled as the multi-hour network design problem. A robust approach was followed in this study as an alternative for separation of metric inequalities in the case of static routing.

The models applied in this work for solving the survivable network design problem with demand uncertainty, are presented in Chapter 4. The problem is formulated as a mixed integer programming problem consisting of two parts, namely, a hardware part and a routing part. The hardware part of the problem addresses the technological requirements found in a typical SDH network. A bifurcated routing model is assumed with fractional flow variables. A path based formulation is adopted with survivability enforced by means of diversification. Two routing alternatives are presented which are in line with routing strategies followed in IP networks, namely, dynamic routing and static routing. To accommodate the solution approach followed in this thesis, a Benders decomposition scheme is applied for projecting out the flow variables, resulting in a master problem consisting of only the hardware part of the problem. Part of the discussion in Chapter 4 is therefore directed towards characterising feasibility with respect to the capacities for the edges in the hardware part, by ways of augmenting the hardware part of the problem with metric inequalities. Propositions are provided that generalise metric inequalities for both the dynamic and static routing cases.

The algorithmic approach followed in this study is based on the branch-and-cut framework and is presented in Chapter 5. The most important aspect of the approach is the ability to quantify the quality of feasible solutions. Even if an optimal solution is not found within a certain time limit, a quality gap is provided that expresses the potential increase in objective value that might still be obtained by improving the current best feasible solution. Furthermore, the branch-and-cut framework is considered primarily for improving computational tractability of the branch-and-bound through the application of valid inequalities and for incorporating the metric inequalities that characterise feasible capacities. Contributions presented in Chapter 5 include a problem reduction approach based on the theory of domination, primal heuristics for inclusion into the branch-and-cut framework, a strengthening approach for metric inequalities in the case of dynamic routing, a robust separation approach in the case of static routing, and finally, an iterative polyhedron approach (IPEA) for reducing the number of demand vectors that need to be included into the problem formulation during the optimisation process.

The computational results presented in Chapter 6 serve as a barometer for evaluating the impact

of the contributions made by this work. The proposed heuristics for generating primal solutions within the branch-and-cut framework is shown to bring about some improvements in computing times for some problem instances. However, there is no clear indication of which combination of heuristics, i.e. either KOCON+KOIMP, or EFCON+KOIMP or only KOIMP, should be used for consistent improvement. Furthermore, there is no clear indication as to what parameters must be used for specifying the frequency at which these heuristics should be applied during the branch-and-cut process.

In evaluating the strengthening of metric inequalities in the case of dynamic routing, some improvements were observed in the number of metric inequalities being added in total to the problem formulation. The anticipated advantage has been that an improvement in computing times will be obtained whenever it is possible to reduce the number of valid inequalities that will be added for each LP solved at the nodes of the branch-and-cut. The strengthening, however, did not always have a significant influence on computing times and even had a negative effect on computing times for some choices of the proposed strengthening strategies. This is attributed to the inability to identify the subset of demand vectors that will optimise the strengthening procedure. Very positive results have, however, been observed for the implementation of a robust separation approach in the case of static routing.

Positive results have been observed for applying the IPEA to both the random data sets and the operational data sets, and the application of the KOIMP heuristic as part of IPEA proved to be valuable for generating starting feasible solutions. For the random data sets, the results obtained testifies of good scalability and also shows that the Largest Sum of Entries (LSE) is superior as a strategy for selecting the initial subset of demand vectors. Furthermore, from the results it is clear that for most problem instances only a small percentage of demand vectors should be included into the network design process due to the large number of demand vectors being dominated.

In the final section on the computational results, the diversification parameters on the operational data sets are omitted due to the inability of the dominance checking algorithms in the case of dynamic routing to deal with survivability requirements. The results are very positive and clearly show that significant improvements in computing times can be achieved by firstly, reducing the number of demand vectors by applying the dominance checking algorithm, and then secondly, by applying the IPEA to the reduced set of demand vectors.

7.2 Future Work

The inability of the suggested dominance checking algorithm in the case of dynamic routing to take into account survivability, is definitely an area that needs attention. From the results where the IPEA was applied to data with survivability requirements, i.e. Section 6.5, it was shown that only

a small percentage of demand vectors are really necessary for describing the feasible region of the network design problem. An alternative approach for doing dominance checking is suggested in Section 5.1.7, however, a thorough investigation will be required to determine the computational feasibility of the approach.

The strengthening strategies suggested in Section 5.4 for the case of dynamic routing, have the attractive property of reducing the number of metric inequalities that need to be added to the problem formulation. It is unfortunate, however, that by doing so no significant reduction in computing times were observed. It is anticipated that if a more efficient way can be found of identifying a suitable subset of demand vectors to be used in the strengthening approach, reduction in computing times may be obtained.

From the empirical results it is clear that more research is needed for determining efficient ways of doing separation in the case of static routing. Even though significant improvements were observed by applying a robust separation strategy, the computing times are still dauntingly high.

Appendix A

Symbols and Notation

Parameter	Description	First Reference
V	Set of potential node locations.	Section 4.1.1
$\mathcal{E}$	Set of edges corresponding to the complete graph.	Section 4.1.1
$G = (V, \mathcal{E})$	Complete graph.	Section 4.1.1
$E \subseteq \mathcal{E}$	Set of edges corresponding to the supply graph.	Section 4.1.1
$\phi(v) \subseteq E$	Set of incident edges to node $v \in V$.	Section 4.1.1
$H = (V, E)$	Supply graph.	Section 4.1.1
$K \subseteq \mathcal{E}$	Set of commodities.	Section 4.1.1
$J = (V, K)$	Demand graph.	Section 4.1.1
d_k	Total demand for a commodity $k \in K$.	Section 4.1.1
$\mathcal{T} = \{1, 2, \ldots, \lvert\mathcal{T}\rvert\}$	Set for indexing the set of demand vectors.	Section 4.1.1
$D = \{d^1, d^2, \ldots, d^{\lvert\mathcal{T}\rvert}\}$	Set of demand vectors.	Section 4.1.1
$\mathcal{N}(v)$	Set of admissible node designs at node $v \in V$.	Section 4.1.2
$S^n \in \mathbb{Z}_+$	Slot capacity for node design $n \in \mathcal{N}(v)$.	Section 4.1.2
$C^n \in \mathbb{Z}_+$	Switching capacity for node design $n \in \mathcal{N}(v)$.	Section 4.1.2
$c^n \in \mathbb{R}_+$	Cost for node design $n \in \mathcal{N}(v)$.	Section 4.1.2
$\mathcal{M}(n)$	Set of installable modules for node design $n \in \mathcal{N}(v)$ at a node $v \in V$.	Section 4.1.2
$M^{m,n} \in \mathbb{Z}_+$	Limit on the number of modules of type $m \in \mathcal{M}(n)$	Section 4.1.2
$S^m \in \mathbb{Z}_+$	Number of slots consumed by module $m \in \mathcal{M}(n)$ for a node design $n \in \mathcal{N}(v)$ at a node $v \in V$.	Section 4.1.2
$\mathcal{I}(m)$	Set of available interfaces for module $m \in \mathcal{M}(n)$ for a node design $n \in \mathcal{N}(v)$ at a node $v \in V$.	Section 4.1.2

Table A.1: Parameters and definitions

Parameter	Description	First Reference
$I_i^m \in \mathbb{Z}_+$	Interface capacity for interface type $i \in \mathcal{I}(m)$ for a module $m \in \mathcal{M}(n)$, node design $n \in \mathcal{N}(v)$ at a node $v \in V$.	Section 4.1.2
$c^m \in \mathbb{R}_+$	Cost for module $m \in \mathcal{M}(n)$ for a node design $n \in \mathcal{N}(v)$ at a node $v \in V$.	Section 4.1.2
$\mathcal{L}(e)$	Set of admissible edge designs at edge $e \in E$.	Section 4.1.2
$\mathcal{I}(l)$	Interface requirements for edge design $l \in \mathcal{L}(e)$.	Section 4.1.2
$I_i^l \in \mathbb{Z}_+$	Interface capacity for interface type $i \in \mathcal{I}(m)$ for an edge design $l \in \mathcal{L}(e)$.	Section 4.1.2
$\mathcal{P}$	Set of all possible paths for all possible commodities.	Section 4.1.3
$\mathcal{P}(k)$	Set of paths for a commodity $k \in K$.	Section 4.1.3
$E(p) \subseteq E$	Set of edges representing a path $p \in \mathcal{P}(k)$ with $k \in K$.	Section 4.1.3
S	Set of all failure states.	Section 4.1.3
$S(p) \subseteq S$	Failure states that will affect a path $p \in \mathcal{P}(k)$,$k \in K$.	Section 4.1.3
$\delta_k \in (0, 1]$	Diversification parameter for a commodity $k \in K$.	Section 4.1.3
$\mathcal{P}(k, s) \subseteq \mathcal{P}(k)$	Paths for a commodity $k \in K$ affected by failure state $s \in S$.	Section 4.1.3
$\mathcal{H}$	Set of feasible solutions for the hardware model.	Section 4.2.1
$\tilde{\mathcal{H}} \subseteq \mathcal{H}$	Set of feasible solutions for the hardware model augmented with metric inequalities.	Section 5.2.3
$F(y, d, \delta)$	Set of feasible flows for a given capacity vector y, a demand vector d, and a diversification vector δ.	Section 4.2.2
$F(y, d)$	Set of feasible flows for a given capacity vector y, and a demand vector d, without any diversification.	Section 4.2.2
$F(y, D, \delta)$	Set of feasible flows (dynamic) for a given capacity vector y, a set of demand vectors D, and a diversification vector δ.	Section 4.2.3
$F(y, D)$	Set of feasible flows (dynamic) for a given capacity vector y, and a set of demand vectors D, without any diversification.	Section 4.2.3
$R(y, D, \delta)$	Set of feasible flows (static) for a given capacity vector y, a set of demand vectors D, and a diversification vector δ.	Section 4.2.4
$R(y, D)$	Set of feasible flows (static) for a given capacity vector y, and a demand vector d, without any diversification.	Section 4.2.4

Table A.1: Parameters and definitions (Cont.)

Parameter	Description	First Reference
$\mathcal{Y}(d)$	Set of capacity vectors supporting the demand vector d	Section 5.1.1
$\mathcal{Y}(D)$	Set of capacity vectors supporting the set of demand vectors D	Section 5.1.1
$\mathcal{Y}(D, \delta)$	Set of capacity vectors supporting the set of demand vectors D with a common diversification vector δ.	Section 5.1.7
$\mathcal{YR}(D)$	Set of all pairs y and r, where y supports D with $r \in R(y, D)$	Section 5.1.1
$\mathcal{YR}(D, \delta)$	Set of all pairs y and r, where y supports D with $r \in R(y, D, \delta)$	Section 5.1.6

Table A.1: Parameters and definitions (Cont.)

Abbreviation	Description	First Reference
SNDP(d, δ)	The survivable network design problem for a single demand vector.	Section 4.2.5
SNDP-DR(D, δ)	The survivable network design problem for a set of demand vectors assuming dynamic routing.	Section 4.2.5
SNDP-SR(D, δ)	The survivable network design problem for a set of demand vectors assuming static routing.	Section 4.2.5
SNDP-DR/SR(D, δ)	A reference to the survivable network design problem for both the dynamic and static routing cases.	Section 4.2.5
SIMPLE_SNDP(c, D, δ)	An edge flow formulation of the survivable network design problem assuming dynamic routing.	Section 5.1.7
SNDP-K-Opt heuristic	Primal heuristic to be used within the branch-and-cut.	Section 5.3.1
SNDP-CARD(η, ζ)	SNDP(d, δ) amended with cardinality constraints to be used in the SNDP-K-Opt heuristic.	Section 5.3.1
SNDP-CARD-RELAX(η, ζ)	SNDP-CARD(η, ζ) for which a cardinality constraint has been relaxed.	Section 5.3.1
KOCON	K-Opt solution CONstruction part of the SNDP-K-Opt heuristic	Section 5.3.1
KOIMP	K-Opt solution IMProvement part of the SNDP-K-Opt heuristic	Section 5.3.1
EFCON	Edge Flow solution CONstruction heuristic	Section 5.3.2

Table A.2: Acronym and abbreviation list

Abbreviation	Description	First Reference
LSE-TOPx	Largest Sum of Entries-TOPx strengthening strategy for separation in the case of dynamic routing	Section 5.4.1
MDE-TOPx	Maximum Demand Entry-TOPx strengthening strategy for separation in the case of dynamic routing	Section 5.4.1
IPEA	The Iterative Polyhedron Expansion Approach for solving SNDP-DR/SR(D, δ)	Section 5.5
LSE	Largest Sum of Entries strategy for selecting an initial subset of demand vectors to be used in IPEA	Section 5.5
CSS	Commodity Supremum Set strategy for selecting an initial subset of demand vectors to be used in IPEA	Section 5.5
EDT	Extended Dominant Tree strategy for selecting an initial subset of demand vectors to be used in IPEA	Section 5.5

Table A.2: Acronym and abbreviation list (Cont.)

Appendix B

Complete Results

	CPLEX		EFCON		KOCON+KOIMP		EFCON+KOIMP		KOIMP	
Instance	Gap(%)	Time	Gap(%)	Time	Gap(%)	Time	Gap(%)	Time	Gap(%)	Time
cost266	100.0	02:00:38	93.1	02:00:36	43.6	02:00:38	53.0	02:00:31	100.0	02:00:36
dfn-bwin	86.7	02:12:17	87.2	02:11:46	100.0	02:10:46	100.0	02:11:58	100.0	02:12:11
dfn-gwin	44.2	02:17:23	48.8	02:16:11	42.7	02:15:55	45.8	02:15:35	43.3	02:15:36
di-yuan	51.0	02:14:12	47.6	02:12:58	38.8	02:12:15	57.2	02:12:08	44.3	02:13:20
nobel-eu	2.0	02:14:07	2.0	02:11:59	2.7	02:10:19	1.6	02:12:22	2.0	02:12:16
norway	48.5	02:00:19	56.8	02:00:17	64.6	02:00:17	54.5	02:00:14	36.0	02:00:19
pdh	54.9	02:12:42	46.5	02:11:55	47.4	02:11:13	53.6	02:11:40	52.7	02:11:37
ta1	13.5	02:00:29	17.3	02:00:35	5.0	02:00:27	40.9	02:00:21	1.6	02:00:29
ta2	6.0	02:00:27	10.0	02:00:28	7.7	02:00:25	7.1	02:00:26	10.2	02:00:28
newyork	0.0	00:23:23	0.0	00:47:58	0.0	00:33:37	0.0	00:49:31	0.0	00:38:36
nobel-germany	0.0	00:00:38	0.0	00:02:06	0.0	00:07:43	0.0	00:02:02	0.0	00:02:45
nobel-us	0.0	00:00:43	0.0	00:00:50	0.0	00:11:13	0.0	00:00:45	0.0	00:02:01

Table B.1: Results for applying heuristics to SNDLIB

		CPLEX		STR-ALL		MAX-SUM-TOP2		MAX-SUM-TOP5		MAX-SUM-TOP10	
Instance	# Vectors	Time	# Inequalities	Time	# Inequalities	Time	# Inequalities	Time	# Inequalities	Time	# Inequalities
nobel-germany	5	00:03:13	685	00:02:47	342	00:03:21	685	00:03:31	685	00:03:35	685
nobel-germany	10	00:06:01	912	00:13:54	499	00:05:33	964	00:06:16	912	00:06:19	912
nobel-germany	20	00:08:29	1289	00:15:37	355	00:07:55	1289	00:07:51	1250	00:09:32	542
nobel-germany	40	00:13:58	2404	00:16:36	348	00:14:15	2404	00:12:31	749	00:08:34	591
nobel-germany	80	03:38:57	3488	01:01:00	353	01:19:56	2984	01:56:32	745	02:26:35	593
nobel-germany	160	01:14:49	5898	02:02:21	367	01:42:05	2227	01:39:48	637	00:59:46	452
nobel-germany	320	02:41:48	10361	03:03:32	335	02:00:12	4260	03:31:51	578	03:12:21	520
nobel-germany	640	06:52:29	18348	11:34:21	363	12:01:17	2935	08:37:14	1609	11:42:48	1114
nobel-us	5	00:01:18	350	00:01:53	303	00:01:19	350	00:01:16	350	00:01:18	350
nobel-us	10	00:03:21	415	00:03:14	256	00:03:22	415	00:03:19	415	00:03:19	415
nobel-us	20	00:07:26	700	00:05:43	233	00:07:59	708	00:07:49	708	00:05:43	311
nobel-us	40	00:11:04	1082	00:12:26	313	00:11:03	1082	00:09:34	321	00:11:57	312
nobel-us	80	00:27:27	1638	00:25:40	282	00:22:00	641	00:23:41	400	00:19:00	274
nobel-us	160	00:55:27	2693	01:16:10	326	00:46:36	401	01:00:03	411	01:01:04	405
nobel-us	320	01:31:28	5928	01:32:46	370	01:59:25	430	01:33:10	386	01:33:28	386
nobel-us	640	04:13:43	9393	03:31:39	364	04:37:19	480	04:10:38	409	03:13:12	368

Table B.2: Results for applying strengthening strategies to SNDLIB-RANDOM (1)

		CPLEX		STR-ALL		MAX-DEM-TOP2		MAX-DEM-TOP5		MAX-DEM-TOP10	
Instance	# Vectors	Time	# Inequalities	Time	# Inequalities	Time	# Inequalities	Time	# Inequalities	Time	# Inequalities
nobel-germany	5	00:03:13	685	00:02:47	342	00:03:21	685	00:03:26	685	00:03:20	685
nobel-germany	10	00:06:01	912	00:13:54	499	00:05:33	964	00:05:41	964	00:05:31	964
nobel-germany	20	00:08:29	1289	00:15:37	355	00:07:57	1289	00:08:07	1289	00:10:14	1093
nobel-germany	40	00:13:58	2404	00:16:36	348	00:14:23	2404	00:12:17	749	00:08:55	868
nobel-germany	80	03:38:57	3488	01:01:00	353	02:23:45	2228	02:02:34	1549	01:39:42	1557
nobel-germany	160	01:14:49	5898	02:02:21	367	00:56:39	2851	01:10:32	1213	01:23:10	1148
nobel-germany	320	02:41:48	10361	03:03:32	335	02:45:58	2196	02:42:08	1153	04:47:04	1785
nobel-germany	640	06:52:29	18348	11:34:21	363	06:16:20	2545	08:13:21	2069	11:57:30	1114
nobel-us	5	00:01:18	350	00:01:53	303	00:01:17	350	00:01:18	350	00:01:16	350
nobel-us	10	00:03:21	415	00:03:14	256	00:03:22	415	00:03:28	415	00:03:24	415
nobel-us	20	00:07:26	700	00:05:43	233	00:07:46	708	00:07:44	708	00:07:05	264
nobel-us	40	00:11:04	1082	00:12:26	313	00:11:12	1082	00:12:15	451	00:10:14	452
nobel-us	80	00:27:27	1638	00:25:40	282	00:36:12	1465	00:39:22	1071	00:27:49	650
nobel-us	160	00:55:27	2693	01:16:10	326	00:51:56	672	00:51:34	672	00:47:08	631
nobel-us	320	01:31:28	5928	01:32:46	370	01:17:27	1581	01:37:18	428	01:40:19	428
nobel-us	640	04:13:43	9393	03:31:39	364	03:41:43	2309	02:38:21	817	05:11:58	568

Table B.3: Results for applying strengthening strategies to SNDLIB-RANDOM (2)

		CPLEX		ROBUST	
Instance	# Vectors	Time	# Inequalities	Time	# Inequalities
nobel-germany	2	7:12:20	12537	4:41:52	10087
nobel-germany	4	23:59:00	21166	8:59:00	13202
nobel-germany	8	23:59:00	21323	4:49:28	10939
nobel-us	2	6:01:25	8305	13:09:05	11119
nobel-us	4	5:44:56	7918	3:14:34	6277
nobel-us	8	23:59:00	17143	13:22:04	11414

Table B.4: Results for applying static routing separation strategies to SNDLIB-RANDOM

		CPLEX	IPEA+EDT			IPEA+LSE			IPEA+CSS		
Instance	# Vectors	Time	Time	Initial $\lvert D_0 \rvert$	Final $\lvert D_0 \rvert$	Time	Initial $\lvert D_0 \rvert$	Final $\lvert D_0 \rvert$	Time	Initial $\lvert D_0 \rvert$	Final $\lvert D_0 \rvert$
nobel-us	5	0:01:18	0:00:18	4	4	0:00:18	1	1	0:00:58	5	5
nobel-us	10	0:03:21	0:02:45	7	7	0:02:33	1	4	0:04:48	10	10
nobel-us	20	0:07:26	0:05:39	9	13	0:02:56	1	5	0:04:41	20	20
nobel-us	40	0:11:04	0:05:09	12	13	0:00:32	1	1	0:12:13	38	38
nobel-us	80	0:27:27	0:10:01	12	16	0:07:29	1	11	0:55:17	50	53
nobel-us	160	0:55:27	0:08:08	13	29	0:03:12	1	7	0:49:43	71	72
nobel-us	320	1:31:28	0:11:02	13	42	0:01:42	1	4	0:34:11	83	84
nobel-us	640	4:13:43	0:29:22	13	138	0:03:16	1	5	1:22:23	88	94
nobel-germany	5	0:03:13	0:04:07	4	5	0:01:19	1	2	0:04:22	5	5
nobel-germany	10	0:06:01	0:20:52	9	9	0:06:55	1	9	0:11:12	10	10
nobel-germany	20	0:08:29	0:07:13	13	17	0:02:21	1	5	0:11:28	20	20
nobel-germany	40	0:13:58	0:15:16	11	17	0:16:38	1	14	0:13:04	38	38
nobel-germany	80	3:38:57	0:50:00	16	25	0:42:42	1	21	1:40:45	61	61
nobel-germany	160	1:14:49	0:17:20	16	26	0:39:50	1	39	0:39:09	92	92
nobel-germany	320	2:41:48	0:57:56	16	76	0:32:31	1	75	1:34:12	95	108
nobel-germany	640	6:52:29	1:09:11	16	84	0:35:34	1	81	2:45:20	107	111

Table B.5: Results for applying IPEA to SNDLIB-RANDOM

	CPLEX	IPEA+EDT				IPEA+LSE			
Instance	Gap(%)	Gap(%)	Initial	$\|D_0\|$	Final	$\|D_0\|$	Gap(%)	Initial	$\|D_0\|$	Final	$\|D_0\|$
DFN-12	32.8	25.8	8		8		33.9	1		3	
DFN-28	51.7	28.5	8		16		18.8	1		17	
DFN-288	infinity	57.3	17		73		31.1	1		36	
EIBONE17-12	4.2	14.8	4		4		13.9	1		1	
EIBONE17-28	3.3	14.7	9		17		0.0	1		5	
EIBONE17-288	9.3	12.1	10		43		0.0	1		2	
EIBONE50-12	infinity	55.6	8		8		54.9	1		3	
EIBONE50-28	infinity	51.2	15		15		43.7	1		8	
EIBONE50-288	infinity	infinity	32		32		61.6	1		9	

Table B.6: Results for applying IPEA to the DFN and EIBONE data sets - dynamic routing

	CPLEX	IPEA+EDT				IPEA+LSE			
Instance	Gap(%)	Gap(%)	Initial	$\|D_0\|$	Final	$\|D_0\|$	Gap(%)	Initial	$\|D_0\|$	Final	$\|D_0\|$
DFN-12	infinity	infinity	8		8		infinity	1		8	
DFN-28	infinity	infinity	8		8		infinity	1		27	
DFN-288	infinity	infinity	17		17		infinity	1		222	
EIBONE17-12	11.8	13.8	4		10		15.4	1		1	
EIBONE17-28	15.4	42.9	9		25		27.3	1		1	
EIBONE17-288	infinity	infinity	10		168		32.0	0		0	
EIBONE50-12	infinity	infinity	8		8		infinity	0		0	
EIBONE50-28	infinity	infinity	15		15		infinity	0		0	
EIBONE50-288	infinity	infinity	32		32		infinity	0		0	

Table B.7: Results for applying IPEA to the DFN and EIBONE data sets - static routing

	CPLEX	DOM+CPLEX	DOM+IPEA+EDT			DOM+IPEA+LSE		
Instance	Gap(%)	Gap(%)	Gap(%)	Initial D_0	Final D_0	Gap(%)	Initial D_0	Final D_0
DFN-12	18.4	14.2	17.8	9	10	18.5	1	8
DFN-28	15.3	11.9	14.6	6	8	13.0	1	6
DFN-288	infinity	42.4	19.0	8	13	21.4	1	12
EIBONE17-12	7.0	6.8	24.0	3	4	24.0	1	2
EIBONE17-28	3.7	3.0	20.3	7	8	20.3	1	7
EIBONE17-288	5.5	3.3	0.0	6	9	0.0	1	2
EIBONE50-12	36.7	36.7	45.8	8	8	33.3	1	3
EIBONE50-28	84.7	70.5	44.8	10	12	34.3	1	8
EIBONE50-288	infinity	infinity	37.6	17	21	48.2	1	1

Table B.8: Results for applying domination to the DFN and EIBONE data sets - dynamic routing

Bibliography

[AABP04] A. Altm, E. Amaldi, P. Belotti, and M.Ç. Pinar. Virtual private network design under traffic uncertainty. *Electronic Notes in Discrete Mathematics*, 17:19–22, 2004.

[AP99] A. Amiri and H. Pirkul. Routing and capacity assignment in backbone communication networks under time varying traffic conditions. *European Journal of Operational Research*, 117:15–29, 1999.

[AZ07] A. Atamtürk and M. Zhang. Two-stage robust network flow and design under demand uncertainty. *Operations Research*, 55(4):662–673, 2007.

[Bal98] E. Balas. Projection with a minimal system of inequalities. *Computational Optimization and Applications*, 210:189–193, 1998.

[Ben62] J. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[BGGN04] A. Ben-Tal, A. Goryashko, E. Guslitser, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99:351–376, 2004.

[BJS90] M. Bazaraa, J.J. Jarvis, and H.D. Sherali. *Linear Programming and Network Flows.* John Wiley & Sons, 1990.

[BM98] J. Bondy and U. Murty. *Graph Theory with Applications.* American Elsevier, 1998.

[BN99] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25:1–13, 1999.

[BN00] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88:411–424, 2000.

[BS04] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

[CD93] K. Chari and A. Dutta. Design of private backbone networks - I: time varying traffic. *European Journal of Operational Research*, 67:428–442, 1993.

[CPL] Cplex 10.1 reference manual. http://www.cplex.com.

[Dan55] G.B. Dantzig. Linear programming under uncertainty. *Management Science*, 1(3/4):197–206, 1955.

[Dij59] E. Dijkstra. A note on two problems in connextion with graphs. *Numeriche Mathematics*, 1:269–271, 1959.

[DMT97] M.A.H. Dempster, E.A. Medova, and R.T. Thompson. A stochastic programming approach to network planning. In *Proceedings of the 15th International Teletraffic Congress*, pages 329–339. North Holland, 1997.

[DS94] G. Dahl and M. Stoer. A polyhedral approach to multicommodity survivable network design. *Numerische Mathematik*, 68:149–167, 1994.

[Dut94] A. Dutta. Capacity planning of private networks using DCS under multibusy-hour traffic. *IEEE Transactions on Communications*, 42(7):2371–2374, 1994.

[eib] Efficient integrated backbone. http://www.pt-it.pt-dlr.de/en/983.php.

[GH61] R.E. Gomory and T.C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.

[GH64] R.E. Gomory and T.C. Hu. Synthesis of a communication network. *Journal of the Society for Industrial and Applied Mathematics*, 12/2:348–369, 1964.

[GLS88] M. Grötschel, L. Lovósz, and Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer-Verlag, 1988.

[Gro04] W.D. Grover. *Mesh-based Survivable Networks.* Prentice Hall, 2004.

[Iri71] M. Iri. On an extension of the maximum-flow minimum-cut theorem to multicommodity flows. *Journal of the Operations Research Society of Japan*, 13:129–135, 1971.

[KO71] O. Kakusho and K. Onaga. On feasibility conditions of multicommodity flows in networks. *Transactions on Circuit Theory*, 18:425–429, 1971.

[Krö03] A. Kröller. Network optimization: Integration of hardware configuration and capacity dimensioning. Diploma thesis, Technische Universität Berlin, June 2003.

[LOVG99] A. Lissery, A. Ouorouz, J.P. Vialz, and J. Gondzio. Capacity planning under uncertain demand in telecommunications networks, 1999.

[LS99] J. T. Linderoth and M. W. P. Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11:173–187, 1999.

[LSSW99] M. Labbé, R. Séquin, P. Soriano, and C. Wynants. Network synthesis with non-simultaneous multicommodity flow requirements: Bounds and heuristics. Technical report, Centre for Research on Transportation, Université de Montréal, 1999.

[Med95] D. Medhi. Multi-hour, multi-traffic class network design for virtual path-based dynamically reconfigurable wide-area atm networks. *IEEE/ACM Transactions on Networking*, 3:809–818, 1995.

[MH06] D.J. Van Der Merwe and J.M. Hattingh. Tree knapsack approaches for local access network design. *European Journal of Operational Research*, 174:1968–1978, 2006.

[Min89] M. Minoux. Network synthesis and optimum network design problems: Models, solution methods and applications. *Network*, 19:313–360, 1989.

[MM95] T. Magnanti and P. Mirchandani. Modeling and solving the two-facility network loading problem. *Operations Research Letters*, 43(1):142–157, 1995.

[NW98] G.L Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, 1998.

[Ori08] G. Oriolo. Domination between traffic matrices. *Mathematics of Operations Research*, 33(1):91–96, 2008.

[OZ07] F. Ordóñez and J. Zhao. Robust capacity expansion of transit networks. *Networks*, 50(2):136–145, 2007.

[Pad95] M. Padberg. *Linear Optimization and Extensions.* Springer, 1995.

[PM04] M. Pióro and D. Medhi. *Routing, flow, and capacity design in communication and computer networks.* Morgan Kaufmann, 2004.

[PR91] M. Padberg and G. Rinaldi. A branch and cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33:60–100, 1991.

[PW92] Yves Pochet and Laurence A. Wolsey. Network design with divisible capacities: Aggregated flow and knapsack subproblems. In *Proceedings of the 2nd Conference on Integer Programming and Combinatorial Optimization (IPCO 1992), Pittsburgh, PA, USA*, pages 150–164, 1992.

[RA02] M. Riis and K.A. Andersen. Capacitated network design with uncertain demand. *INFORMS Journal on Computing*, 14(3):247–260, 2002.

[Rus55] A. Ruszczyński. Some advances in decomposition methods for stochastic linear programming. *Annals of Operations Research*, (85):153–172, 1955.

[snd05] SNDlib 1.0 – Survivable network design data library. http://sndlib.zib.de, 2005.

[Soy73] A.L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21:1154–1157, 1973.

[Suu74] J. Suurballe. Disjoint paths in a network. *NETWORKS*, 4:125–145, 1974.

[SW69] R.M. Van Slyke and R.J.B Wets. *L-Shape Linear Programmings with Applications to Optimal Control and Stochastic Linear Programming.* Number 17. 1969.

[Tan03] A. Tanenbaum. *Computer Networks.* Pearson Education, 2003.

[TWH06] S.E. Terblanche, R. Wessäly, and J.M. Hattingh. Finding a dominant set of traffic demand matrices. In *Proceedings of the South African Telecommuniations and Networks Applications Conference - SATNAC*, 2006.

[TWH07] S.E. Terblanche, R. Wessäly, and J.M. Hattingh. Solution strategies for the multi-hour network design problem. In *Proceedings of the 3rd International Network Optimization Conference - INOC*, 2007.

[Wes00] R. Wessäly. *DImensioning Survivable Capacitated NETworks.* Phd thesis, Technische Universität Berlin, 2000.

[Wol90] L.A. Wolsey. Valid inequalities for 0/1 knapsacks and mips with generalised upper bound constraints. *Discrete Applied Mathematics*, 29:251–261, 1990.

[Wol98] L.A. Wolsey. *Integer Programming.* John Wiley & Sons, 1998.

[ZG06] Y. Zhang and Z. Ge. Finding critical traffic matrices. In *Proceedings of the 2005 International Conference on Dependable Systems and Networks - DSN*, 2006.