Designing capacitated survivable networks: Polyhedral analysis and algorithms

by

Deepak Rajan

B.Tech. (Indian Institute of Technology - Madras, India) 1999
M.S. (University of California, Berkeley) 2001

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy
in

Engineering-Industrial Engineering
and Operations Research

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Alper Atamtürk, Chair
Professor Ilan Adler
Professor Thomas A. Henzinger
Professor Dorit S. Hochbaum

Fall 2004

The dissertation of Deepak Rajan is approved:

Chair	Date
	Date
	Date
	Date

University of California, Berkeley

Fall 2004

Designing capacitated survivable networks: Polyhedral analysis and algorithms

by

Deepak Rajan

Abstract

Designing capacitated survivable networks: Polyhedral analysis and algorithms

by

Deepak Rajan

Doctor of Philosophy in Engineering-Industrial Engineering and Operations Research

University of California, Berkeley

Professor Alper Atamtürk, Chair

In this dissertation, we develop new methodologies and efficient algorithms to solve the capacitated survivable network design problem. Given the graph and demands between pairs of nodes, we wish to install integer multiples of capacity on the edges and route the demand while minimizing costs. A network is said to be survivable if all demands can be rerouted under the failure of any one of its edges. Traditionally, one uses some variant of the following two approaches: protection or restoration. Protection schemes can be further classified into dedicated protection and shared protection.

We propose a method that uses failure-flow patterns for rerouting of disrupted flow. A hybrid between dedicated and shared protection schemes, our method imposes no restrictions on the network, but explicitly introduces slack on the directed cycles used as failure-flow patterns for rerouting disrupted flow. Using failure-flow patterns results in a much smaller formulation than other approaches in terms of number of constraints; we handle the exponential number of directed cycle variables using column generation.

We study the arc-set and cut-set polyhedra associated with the problem to generate strong valid inequalities. We develop various families of inequalities, and describe efficient separation algorithms for these and other classes of inequalities. By pricing out directed cycle variables and separating the valid inequalities in a branch-and-cut framework, we show that it is possible to solve much larger problem instances using directed cycles (than shared protection), without significant loss in capacity efficiency. Further-

more, the cuts added improve overall performance by an order of magnitude.

The following two directions of research show significant promise in solving capacitated survivable network design problems more effectively. The first considers directed p-cycles as failure-flow patterns for obtaining higher capacity efficiency. Preliminary results are very encouraging; however, pricing sub-problems and polyhedral structure change significantly. The second, approached from two complementary directions, involves the development of stronger classes of inequalities. In the former direction, we develop problem-specific metric-type inequalities for design of survivable networks using various failure-flow patterns. In the latter, we develop problem-independent strong valid inequalities for the mixed-integer knapsack set, a relaxation of any mixed-integer program.

Professor Alper Atamtürk, Chair

Dedication

To my parents, for their unconditional love and support in all my endeavors.

Contents

List of Figures

List of Tables

Acknowledgments

First of all, I wish to thank my adviser Alper Atamtürk. His expert guidance is probably the most significant contribution to this dissertation. At the same time, he has been a friendly and supportive person to work with. I thank my dissertation committee members Dorit Hochbaum and Ilan Adler for all the thought-provoking questions raised at various stages of this dissertation. They are primarily responsible for all the reality checks that are so crucial in any theoretical research. I also thank committee member Thomas Henzinger for the unconditional support and approval of every direction of research I attempted, irrespective of whether it was fruitful.

I will always have fond memories of Brewed Awakening. I loved the coffee; and many of my research ideas germinated while working there. Furthermore, the owners and staff are incredibly friendly, and it is no surprise than many Ph.D. students at Cal spend a significant portion of their life at Brewed.

All other work was carried out in the dungeons (my office in the basement), which was a pleasant place to work in primarily because of the presence of Simge Küçükyavuz and Kerem Bülbül. Kerem has always been a source of inspiration - his work ethic and wry sense of humor ensured that it was fun to be at school, even when I was overloaded with work. I can not have asked for a nicer and smarter person than Simge as an office-mate. I shall always cherish the lunches at North Gate, and the discussions on research, especially during the home stretch. It was great to have her around.

Special thanks to Theresa Roeder, friend and fellow Ph.D. student, for always being there, for everything. Among other things, she has been a huge influence in my tastes and interests, and I believe that I am a much better person because of them. Thanks for getting me hooked on NPR and manual transmission cars.

This acknowledgment would not be complete without Sumitra Ganesh, friend and confidante. I enjoyed the frequent coffee trips, the late nights at work, and all the long discussions on everything under the sun. Thanks for valiantly trying to force me to see

the bigger picture, and even succeeding sometimes.

Many other friends have played a significant part in making my Ph.D. an enjoyable experience. In no particular order, they are Arun Subramanian, Srinath Avadhanula, Arvind Rangarajan, Vidya Varadarajan, Rahul Khatod, Harshal Dedhia, David Phillips, Debbie Pederson, Francisa Cazares, Anne Robinson, and Priya Santhanam.

Special thanks to IEOR Ph.D. students Pablo Durango, Elizabeth Junquiera and Andrew Ross for making me feel welcome in the department when I was still a newbie. Many thanks to the IEOR staff, who ensured that I never had any administrative problems.

The friendly people at IBM Research also merit a special mention. Each and every one of them in the Mathematical Sciences group made me feel at home while I was a summer student. In particular, I thank Samer Takriti, Jon Lee, and Joel Wolf for facilitating the wonderful opportunity to work there.

Finally, I wish to acknowledge Bhavana, my fiancée. I have never been happier in my entire life than now, and look forward to the rest of our life together. Thanks for being patient with me whenever I had to work.

Chapter 1

Introduction

The goal of this dissertation is to develop new methodologies and efficient algorithms to solve the capacitated survivable network design problem. The ever increasing demand for high-speed communication networks is one of the major motivations for research in modeling and solving network design problems. We are concerned only with the planning problem; a study of the real-time routing decisions merits another dissertation all by itself.

Given a set of origin-destination vertex pairs (commodities) and demand data for the commodities, we are interested in installing integer multiples of capacity on the edges and routing the demand through the network while minimizing total (capacity installation and flow) costs. For example, installing or leasing fiber-optic cables on a telecommunication network, determining the capacities of production lines or warehouses in a production-distribution system, determining the number of engines to power a set of trains on a railroad network can all be viewed as installing capacities on the edges of a network and routing the flow of commodities on the network.

A telecommunications network may be represented as a graph where the edges correspond to transmission cables carrying signals between nodes that represent users. In this case, the capacity types are fiber-optic transmission cables of various capacities and installation costs; whereas the commodities are pairwise bandwidth requirements between users, who are represented as nodes. Associated with each commodity is a triple corresponding to the source user, the destination user, and the amount of bandwidth needed (demand), respectively.

Since equipment failures and accidents can not be avoided entirely, the network must be designed so as to survive failure. A network is said to be survivable if all of the demands on the network can be met under the failure of any one of its edges. This is done by judiciously installing sufficient spare capacity over the working edges of the network; all traffic may be diverted by way of this spare capacity in case of a failure. Since this dissertation is motivated by applications in the telecommunications industry, we focus on the design of survivable telecommunication networks. However, most of the mathematical models and solution procedures are also equally applicable to any other problem on physical and/or time-expanded networks that can be abstracted as a capacitated survivable network.

Since capacitated survivable network design problems are modeled as mixed-integer programs, we study several polyhedra of relaxations associated with the problem, in an attempt to generate strong (facet-defining) valid inequalities that explicitly use the survivability requirements. This is because optimization problems over complicated sets can be solved more efficiently by using strong inequalities from simple structured relaxations of these sets; see Section 1.5 for a review of polyhedral techniques in mixed-integer programming. Another advantage of this approach is that valid inequalities to these simpler sets may be useful in solving optimization problems over other complicated sets. We also investigate the complexity of separating these valid inequalities. In particular, we focus on the arc-set and cut-set polyhedra.

To motivate this study, we discuss the network design problem with no survivability requirements (NDP) in Chapter 2. This problem is also referred to as the network loading problem when there is no pre-existing capacity on the edges of the network. In this chapter, we first review the literature summarizing existing results on the cut-set polyhedra, and then present our new results on the arc-set polyhedra. In particular, we develop a linear-time separation algorithm for the residual capacity inequalities that completely describe the splittable arc set. We also introduce two new classes of inequalities that generalize previous results for the unsplittable arc set. We conclude Chapter 2 by presenting computational results that underscore the effectiveness of these inequalities in

solving the NDP.

Survivable networks with minimum capacity requirements can be designed by formulating the problem as a capacitated network design problem for each failure scenario, linked by common integral capacity variables. In fact, this is the same as global rerouting. However, such a framework is not implemented in practice. Therefore, a number of practical models and strategies have been developed for designing survivable networks. In Chapter 3, we review these methodologies for designing survivable networks. To evaluate these strategies, we measure the capacity requirements and compare them with the capacity requirements of global rerouting. We say that a framework is more capacity-efficient if it requires less capacity to ensure survivability.

We also propose a new method for designing survivable networks using predetermined structures for rerouting of disrupted flow. We refer to these structures that reroute all disrupted flow as failure-flow patterns. We explicitly introduce slack on these failure-flow patterns on the network to reroute all disrupted flow under failure, see Chapter 3 for the mixed-integer programming formulation for this framework using directed cycles as the failure-flow patterns. In this approach, a hybrid of dedicated and shared protection, we impose no restrictions on the network, but use these failure-flow patterns for rerouting all failure flows. In particular, using directed cycles as failure-flow patterns yields survivable networks with dedicated protection-like ease-of-implementation and shared protection-like capacity efficiency.

In Chapter 4, we study the polyhedra of several relaxations of the survivable network design problem using directed cycles (SDC), and develop cut-set inequalities. Furthermore, we extend them to k-partition inequalities; $k = 2$ for a cut set. We also discuss how the arc-set inequalities developed for NDP in Chapter 2 can be extended to SDC. We see that using predetermined failure-flow patterns such as directed cycles allows us to deal with formulations with fewer constraints, but an exponential number of directed cycle variables. We study the pricing complexity of the directed cycle variables and handle them using a column-generation scheme.

Computational experiments to solve SDC, summarized at the end of Chapter 4, are

highly encouraging for two main reasons. Firstly, we are able to solve much larger problem instances using failure-flow patterns, as compared with shared protection schemes such as global rerouting. Furthermore, in terms of capacity efficiency, using directed cycles as the patterns is better than dedicated protection schemes which require at least twice as much capacity as NDP. Secondly, the cuts added reduce the integrality gap by about 30% at the root node of the branch-and-cut search tree, and improve overall performance by an order of magnitude. To design more capacity-efficient survivable networks with even less computational effort, we consider two major directions of research.

The former is work in progress, and shows significant promise. In this direction of research, we look at other methods for ensuring survivability. In Chapter 5, we present preliminary computations comparing these models, and motivate the use of directed p-cycles as failure-flow patterns (SDP). This approach appears to perform almost as well as global rerouting, which provides a lower bound on capacity requirements for survivable networks. The models, pricing sub-problems, and valid inequalities however change when compared with SDC.

The latter direction involves the development of stronger classes of inequalities. We approach this objective from two complementary directions.

First, we develop metric-type inequalities for the design of survivable networks. These problem-specific inequalities for the class of network design problems generalize the k-partition inequalities. In Chapter 6, we review metric inequalities for the NDP, and develop new metric-type inequalities for various frameworks that ensure survivability. We present computational results indicating the effectiveness of these inequalities in solving SDP.

Second, we describe strong valid inequalities for the mixed-integer knapsack set. We develop our inequalities using sequence independent lifting of facets of the restriction of this set with one continuous and two integer variables. In Chapter 7, we review earlier work in this direction, and then present our results. This is a problem-independent approach, because the mixed-integer knapsack set is a relaxation of any mixed-integer program; these inequalities can be used to solve many classes of problems beyond capacitated survivable network design. Furthermore, we can develop more valid inequal-

ities for survivable network design problems by generating facets to the knapsack set described by any metric-type inequality.

Finally, in Chapter 8, we conclude by summarizing our contribution and outlining directions for future research.

Next, we present the assumptions and notation which will be used throughout this dissertation. All the mathematical notation is repeated in Appendix A. Since we study several polyhedra associated with the problem in an attempt to generate strong (facet-defining) valid inequalities, we present an overview of mixed-integer programming and polyhedral theory in Section 1.5, and formally define the arc-set and cut-set polyhedra in Section 1.6. We also present a brief introduction to computational complexity in Section 1.3, and linear programming theory in Section 1.4.

1.1 Notation

Throughout this dissertation, we assume that all data is rational.

We use $\mathbb{Z}, \mathbb{R}$ and $\mathbb{Q}$ to denote the set of integers, reals and rational numbers, respectively. The set of non-negative and positive integers are denoted by $\mathbb{Z}_+$ and $\mathbb{Z}_{++}$, respectively; other sets are defined similarly.

We use $\mathbb{B}$ as shorthand to represent the set $\{0, 1\}$.

Throughout this dissertation, we use $\log$ to denote the logarithm to base 2.

For any set N, we use $|N|$ to denote the number of elements in it.

For any vector function v defined on a set N, we let $v(H) = \sum_{i \in H} v_i$ for $H \subseteq N$.

Let $\alpha \in \mathbb{R}$. We define $\lfloor \alpha \rfloor = \max_x \{x \in \mathbb{Z} : x \leq \alpha\}$ and $\lceil \alpha \rceil = \min_x \{x \in \mathbb{Z} : x \geq \alpha\}$.

Let $\beta \in \mathbb{R}$. We define $r(\alpha, \beta) = \alpha - \lfloor \alpha / \beta \rfloor \beta$, and use $r(\alpha)$ to denote $r(\alpha, 1)$.

Let $[i, k]$ be the set of integers $\{j \in \mathbb{Z} : i \leq j \leq k\}$. We use IntUni$[\alpha, \beta]$ to indicate an integer-uniform distribution with minimum α and maximum β.

We define $(\cdot)^+ = \max\{(\cdot), 0\}$.

We define the indicator function $I\{(\cdot)\}$ to take the value 1 if $(\cdot)$ is true, and 0 otherwise.

The i^{th} element of a list $\mathcal{L}$ is denoted by $\mathcal{L}[i]$.

We use ϵ to represent a small strictly positive constant.

We use the following notation for all the network-based formulations in this dissertation. Let $G = (V, E)$ be an undirected graph with node set V and edge set E. Let F be the set of all ordered pairs (arcs) from E, i.e., $F = \{(ij), (ji) : [ij] \in E\}$. We use (ij) to denote the arc from node i to node j, and $[ij]$ to denote the edge between nodes i and j.

Thus, arcs in F are directed links between nodes on a network, and the undirected links between nodes are edges in E. We use $G' = (V, F)$ to denote the directed graph.

For the sake of notational convenience, we use $F \setminus [ij]$ to represent $F \setminus \{(ij), (ji)\}$. For $H_1 \in F$, we use $[H_1]$ to represent the set $\{[ij] \in E : ij \in H_1\}$.

Let S be the set of failure states. A failure state is defined as the set of edges that fail simultaneously. The set S is defined to include the no-failure state, denoted by 0.

In the case of single-edge failures, $S = E \cup \{0\}$.

Let $K = [1, |K|]$ be the set of commodities.

Let $\{(s^k, t^k, d^k)\}_{k \in K}$ be the commodity triples of source and destination nodes s^k and t^k, and d^k be the supply at s^k for t^k, $k \in K$. Let b_i^k be the supply of commodity k at node i, i.e., $b_{s^k}^k = d^k$, $b_{t^k}^k = -d^k$, and $b_i^k = 0$ for $i \in V \setminus \{s^k, t^k\}$.

We define variable y_{ij}^{ks} as the fraction of commodity k routed through arc $(ij) \in F$ in failure state s. Let e_{ij}^k be the cost associated with routing each unit of commodity $k \in K$.

In path-based formulations, we use y_p to indicate the fraction of commodity routed on path p. We use P_k^s to denote the set of paths from s^k to t^k in failure state s.

For any path p, we set $\delta_{ij}^p = 1$ if it includes arc (ij), and 0 otherwise.

Also, for any path p, we set $\zeta_s^p = 1$ if it is affected by failure state $s \in S \setminus \{0\}$. When all flow variables in a formulation only model no-failure routing, we drop the superscript s.

Let T be the set of installable capacity types.

The capacity q^t for type $t \in T$ is the maximum amount of demand that can be routed using one unit of capacity type t.

We define the capacity variable $x_{[ij]}^t$ as the amount of capacity of type $t \in T$ installed on edge $[ij]$. Let $h_{[ij]}^t$ be the cost of installing unit capacity of type $t \in T$ on edge $[ij] \in E$.

If only one capacity type exits, then we drop the superscript t. Furthermore, we can

then without loss of generality assume that all demand data are scaled such that the capacity unit is 1.

For all practical purposes, the cost of sending flow on arc (ij) is insignificant when compared with the cost of installing capacity on edge $[ij]$, $e_{ij} \ll h_{[ij]}$; we keep e_{ij} in the formulation for generality.

Let C and $\mathcal{C}$ be the sets of undirected and directed cycles of G and G', respectively.

For cycle c, we define the variable z_c to denote the amount of slack reserved on cycle c, for both undirected and directed cycles. We use slack to refer to fractional capacity that is reserved to cover no-failure flows; z are modeled as continuous variables.

Let α_{ij}^c be 1 if directed cycle c includes arc (ij), and 0 otherwise. For undirected cycles, we say that $\alpha_{[ij]}^c$ is 1 if undirected cycle c includes edge $[ij]$, and 0 otherwise.

Let $\rho_{[ij]}^c$ be 1 if edge $[ij]$ is a chord to cycle c, and 0 otherwise.

For all edges covered by a failure-flow pattern, we refer to failures of edges used by the pattern as arc failures, and those not used by the pattern as chord failures.

We use g_{ij}^0 to denote the pre-existing amount of demand routed through arc $(ij) \in F$.

We use $w_{[ij]}^0$ to denote the pre-existing capacity on edge $[ij] \in E$.

1.2 Assumptions

For all the models discussed in this dissertation, we assume that capacity installed on an edge can be used to send flow up to capacity in both directions. Thus, the capacity is undirected even though flow is directed. This assumption is motivated from many kinds of telecommunications networks, for instance asynchronous transfer mode networks. The results can be easily adapted to the case where capacity must be greater than the sum of flows in both direction, and to the case where capacity is directed.

Network design problems can be differentiated into splittable flow (when there are no restrictions on how the commodities must be routed) and unsplittable flow (when flow of a commodity is restricted to run through a single path along the network). The unsplittable flow case is modeled by constraining the flow variables y to be binary.

It is possible to define commodities by aggregating all of the flow originating from a node. However, we use pairs of nodes to describe commodities in the formulations because the column generation algorithms for path-based formulations of multi-commodity flow and network design problems converge to an optimal LP solution faster when commodities are disaggregated (Jones et al. 1993).

The formulations of most network design problems in this dissertation primarily consist of three types of constraints. The first class is the flow balance constraints, which ensure that all commodities are routed in the no-failure state. The second class is the capacity constraints, which ensure that sufficient capacity exists on each edge for each failure state. The third class is the survivability constraints, which ensure that all commodities are routed in every failure state. This class of constraints can look very different depending on the methodology used for ensuring survivability.

1.3 Computational complexity

We present some of the definitions and notations used to measure the complexity of the algorithms presented in this dissertation; see Papadimitriou and Steiglitz (1998) for a good introduction to complexity theory.

Definition 1.1 Let f and g be real-valued functions that are defined on the same set of real numbers. Then f, is $\mathcal{O}(g(x))$ if and only if there exist $c, x_0 \in \mathbb{R}_{++}$ such that $|f(x)| \leq c \cdot |g(x)|$ for all $x \geq x_0$. □

Let P be a problem, A be an algorithm that solves P, and X be an instance of problem P. Before we discuss properties of algorithms, we define the notions of decision problems and certificates.

Definition 1.2 A *decision problem* is a problem with a *yes* or *no* answer. □

Definition 1.3 A *certificate* is the extra information so the correctness of an answer to a decision problem can be checked. □

We denote the length (in a reasonable encoding) of the instance by $L(X)$, and the number of elementary calculations required to run algorithm A on instance X by $f_A(X)$.

Definition 1.4 The *running time* of algorithm A is defined as the maximum number of elementary operations required to solve any instance X of P, as a function of its length, and is denoted by $f_A^*(\ell)$. Thus, $f_A^*(\ell) = \sup_X \{f_A(X) : L(X) = \ell\}$. □

Definition 1.5 A is said to be a *polynomial-time* algorithm if there exists $p \in \mathbb{Z}_{++}$ such that $f_A^*(\ell) = \mathcal{O}(\ell^p)$. □

Polynomial algorithms can be further classified into strongly polynomial and weakly polynomial, depending on whether the running time of the algorithm depends on the magnitude of the numbers describing the instance.

Definition 1.6 Algorithm A is said to be *strongly polynomial* if $f_A^*(\ell)$ is bounded by a polynomial function that does not involve the magnitude of the numbers in the description of the instance. □

Definition 1.7 Algorithm A is *weakly polynomial* if it is polynomial and not strongly polynomial. In other words, $\mathcal{O}(\ell^p)$ contains terms involving $\log \theta$, where θ is the largest number in the input. □

We are mainly interested in polynomial algorithms since they scale well with the size of the input. An algorithm that is not polynomial is called an non-polynomial-time algorithm, and is defined formally as follows.

Definition 1.8 An algorithm A is said to be an *non-polynomial-time algorithm* if $f_A^*(\ell) \neq \mathcal{O}(\ell^p)$, for any $p \in \mathbb{Z}_{++}$. □

Some non-polynomial-time algorithms are better than others. One such class of non-polynomial-time algorithms that is of interest is the class of pseudo-polynomial algorithms.

Definition 1.9 An algorithm A is *pseudo-polynomial* if it is polynomial in the length of the data when encoded in unary. This means that A is polynomial in the parameters and the magnitude of the instance data θ. □

An simple example of an problem with a pseudo-polynomial algorithm is the pure-integer knapsack problem: Given n integers $c_i, w_i,\ i \in [1, n]$, and b, maximize $\sum_{i=1}^{n} c_i x_i$

for all integers $x_i \geq 0,\ i \in [1, n]$ such that $\sum_{i=1}^{n} w_i x_i \leq b$. There is an $\mathcal{O}(nb)$ algorithm for this problem. This is a polynomial-time algorithm only when b is bounded by a polynomial function of n.

We also classify problems depending upon the best known algorithm for solving the problem. First, we define the class of problems which have polynomial algorithms.

Definition 1.10 $\mathcal{P}$ is defined as the class of all problems for which there exist some polynomial algorithm. □

Second, we classify those problems for which no known polynomial algorithms exist.

Definition 1.11 $\mathcal{NP}$ is the class of decision problems that can be solved in polynomial time on a non-deterministic turing machine. □

Definition 1.11 is not particularly useful for checking whether a problem $P \in \mathcal{NP}$. The following definition, which is equivalent, allows us to classify problems in $\mathcal{NP}$.

Definition 1.12 A decision problem $P \in \mathcal{NP}$ if for every instance X of P for which the answer is yes, there exists a certificate of polynomial length that can be checked in polynomial time. □

$P \in \mathcal{NP}$ does not imply that no polynomial algorithm exists for P. In fact, $\mathcal{P} \subseteq \mathcal{NP}$. A simple example for the class $\mathcal{NP}$ is the decision version of a binary program: $\exists x \in \mathbb{B}^n$ such that $Ax \leq b,\ cx \geq k$? If the answer is yes, then the binary vector $\bar{x}$ is a polynomial certificate; whether $A\bar{x} \leq b, c\bar{x} \geq k$ can be checked in polynomial time.

Some problems in $\mathcal{NP}$ may be harder to solve than others. We call this class of problems $\mathcal{NP}$-complete. To determine whether a problem is $\mathcal{NP}$-complete, we need the notion of polynomial reducibility.

Definition 1.13 If every instance of P can be converted in polynomial time to an instance of Q, then P is *polynomially reducible* to Q. □

Definition 1.14 Problem is *$\mathcal{NP}$-complete* if $P \in \mathcal{NP}$ and every other problem $Q \in \mathcal{NP}$ is polynomially reducible to P. □

For any optimization problem, we can pose a closely related decision problem, which is in fact a question, and can be answered by *yes* or *no*.

Definition 1.15 Given an instance X of a maximization problem P, such that the set of feasible solutions is $\mathcal{H}$, and the cost of any $h \in \mathcal{H}$ is $c(h)$, we define the following *decision version* for all $k \in \mathbb{R}$: Is there a feasible solution $h \in \mathcal{H}$ such that $c(h) \geq k$? □

It is easy to see that we can obtain the solution of the decision version trivially from the solution of the optimization problem. Thus, the optimization problem is at least as hard to solve as its decision version. The class of problems $\mathcal{NP}$-hard extends $\mathcal{NP}$-complete to include problems that are not in $\mathcal{NP}$.

Definition 1.16 Problem $P \in \mathcal{NP}$*-hard* if every problem $Q \in \mathcal{NP}$ is polynomially reducible to P. □

Such a definition now allows us to classify difficult optimization problems (which, not being decision problems, are not in $\mathcal{NP}$) whose decision versions are $\mathcal{NP}$-complete. Thus, we can say that an optimization problem is $\mathcal{NP}$-hard if its decision version is $\mathcal{NP}$-complete.

1.4 Linear programming theory

We review some aspects of linear programming theory that are used later in this dissertation, see Chvátal (1983) for more details.

Let $|N| = n$, and A be an $m \times n$ matrix. In standard linear programming theory, the constraints of a linear program can be represented in many equivalent forms. For the purpose of this discussion, we formulate an LP as follows.

$$z_{LP} = \max\{cx \,:\, x \in S_{LP}\},$$

where $S_{LP} = \{x \in \mathbb{R}^n_+,\ Ax \leq b\}$.

Observe that S_{LP} is a polyhedron, since it is defined as a set of points that satisfy a finite number of linear inequalities; see Definition 1.29. For this polyhedron, we define extreme points and extreme rays.

Definition 1.17 $x \in S_{LP}$ is an *extreme point* of S_{LP} if there do not exist $x^1, x^2 \in S_{LP}, x^1 \neq x^2$ such that $x = \frac{1}{2}x^1 + \frac{1}{2}x^2$. □

Definition 1.18 Let $S^0_{LP} = \{x \in \mathbb{R}^n : Ax \leq 0\}$. If $S_{LP} \neq \emptyset$, then $x \in S^0_{LP} \setminus \{0\}$ is called a *ray* of S_{LP}. □

Since S^0_{LP} is a cone, it is often referred to as the polyhedral cone of S_{LP}.

Definition 1.19 A ray x of S_{LP} is an *extreme ray* of S_{LP} if there do not exist rays $x^1, x^2 \in S^0_{LP}, x^1 \neq \alpha x^2$ for any $\alpha > 0$ such that $x = \frac{1}{2}x^1 + \frac{1}{2}x^2$. □

To determine whether an LP is infeasible, a certificate is obtained using Farkas' Lemma. This key result is also used to derive much of linear programming theory.

Proposition 1.20 (Farkas' Lemma) For all A, b, exactly one of the following holds: either $\exists x \geq 0$ such that $Ax \leq b$, or $\exists w \geq 0$ such that $wA \geq 0$ and $wb < 0$. □

The dual problem to an LP is derived using the following function. Consider the function $g : \mathbb{R}^m_+ \mapsto \mathbb{R}$ defined as

$$g(w) = \max_{x \in \mathbb{R}^n_+} \{cx + w(b - Ax)\}.$$

For every vector $w \in \mathbb{R}^m_+$, $g(w)$ is a upper bound on the optimal value of the original LP. To achieve the best bound, we minimize $g(w)$; this maximization problem is the dual to the original LP.

Definition 1.21 The *dual* to the original LP is the optimization problem

$$z_D = \min_{w \geq 0} \{wb : wA \geq c\}.$$ □

Similar to the original LP, a cone can be defined for the dual.

Definition 1.22 The *dual cone* of the LP is set of points $w \in \mathbb{R}^m_+$ that satisfy $wA \geq 0$. □

When an LP contains a large set of constraints or variables, it is often solved by starting with a reduced set, and adding the remaining constraints/variables when needed. In fact, to solve a particular instance, we really only need constraints that have positive

dual values and variables that have positive values at optimality. The problem is that we don't know which variables and constraints these are.

In problems with large numbers of variables, we can address this problem with a delayed column generation approach. To do so, we start with a subset of variables, and solve the LP with these variables. By adding artificial variables, we can assume without loss of generality that there exists a feasible solution to this LP. Given the restricted LP optimum, the problem of finding a variable with a positive reduced cost is referred to as the pricing problem.

Let A_i denote column i of matrix A. Consider the restricted LP obtained by considering only the subset of the columns indexed by set I.

$$z_{LP}^I = \max\{cx \,:\, x \in \mathbb{R}_+^{|I|},\ \sum_{i \in I} A_i x_i = b\}$$

We solve this LP and calculate an optimal primal and dual solution pair $(\bar{x}, \bar{w})$. Now, we need to generate a new variable j for which the reduced cost is positive. This can be done by maximizing the reduced cost,

$$\max_{j \in N \setminus I} \{c_j - \bar{w} A_j\}.$$

However, we do not need to find the optimal solution to this problem; any solution with a positive reduced cost will suffice. If there exists a variable with positive cost, then we add the new column to the set I and re-solve the restricted LP. If there exists no such variable, the restricted LP gives us the optimal solution to the original LP.

1.5 Mixed-integer programming: An overview

Definition 1.23 Let N, P be sets such that $|P| = p$ and $|N| = n$. For $m \in \mathbb{Z}_+$, let A be an $m \times n$ matrix, G an $m \times p$ matrix, and b an m dimensional column vector. Let $S = \{x \in \mathbb{Z}_+^n, y \in \mathbb{R}_+^p,\ Ax + Gy \leq b\}$. Then, a *mixed-integer program (MIP)* can be formulated as

$$z_{MIP} = \max\{cx + hy \,:\, (x, y) \in S\}.$$ □

Set S is called the feasible region, and any $(x, y) \in S$ is called a feasible solution; see Schrijver (1987), Nemhauser and Wolsey (1988), and Wolsey (1998) for a comprehensive study on mixed-integer programming.

Since MIPs (and other optimization problems) are represented using matrices, the variables are often referred to as columns, and constraints as rows.

If the mixed-integer program has only integer variables ($P = \emptyset$), then it is referred to as a pure-integer program (IP). On the other hand, if there are continuous variables and all integer variables are constrained to be 0 or 1 in any feasible solution, then the program is referred to as a mixed-binary program. The special case with exclusively binary variables is referred to as a binary program. A single constraint mixed-integer program is also called a mixed-integer knapsack. The pure-integer, mixed-binary and binary knapsacks are defined analogously as the single constraint special cases of pure-integer program, mixed-binary program, and binary programs, respectively.

The mixed-integer set S is not convex. However, one can define the smallest convex set that contains S, the convex hull of S.

Definition 1.24 The *convex hull* of a mixed-integer set S, denoted by $\mathrm{conv}(S)$, is the set of all points that are convex combinations of elements of S. □

Definition 1.25 The inequality $\pi_N x + \pi_P y \leq \pi_0$ (denoted (π_N, π_P, π_0)) is called a *valid inequality* for a mixed-integer set S if it is satisfied by all elements of S. □

Definition 1.26 For any inequality $\pi_N x + \pi_P y \leq \pi_0$ that is valid for mixed-integer set S, the set $F = \{(x, y) \in S : \pi_N x + \pi_P y = \pi_0\}$ is called a *face* of $\mathrm{conv}(S)$. □

Definition 1.27 Let $a_1, \ldots, a_n$ be scalars not all equal to 0. Then, the set H consisting of all $x \in \mathbb{R}^n$ such that $\sum_{i=1}^{n} a_i x_i = 0$ is an $(n-1)$-dimensional *hyperplane*. □

Valid inequalities are often referred to as cutting planes (or cuts) since they define hyperplanes in $(n + p)$-dimensional space. An alternate definition of valid inequalities uses the concept of half-spaces.

Definition 1.28 A *half-space* is that portion of an n-dimensional space obtained by removing the part lying on one side of an $(n-1)$-dimensional hyperplane. □

Now, inequality is said to be valid for the set S if the half-space defined by the inequality contains S.

Definition 1.29 A *polyhedron* $P \subseteq \mathbb{R}^{n+p}$ is a set of points that satisfy a finite number of linear inequalities. □

For any polyhedron, we can define its extreme points and extreme rays. We do so in Section 1.4 for the feasible regions of linear programs; however, the definitions are valid for all polyhedra.

Definition 1.30 A set of vectors $(x,y)^1, \ldots, (x,y)^k$ is affinely independent if the unique solution of $\sum_{i=1}^{k} \lambda_i x^i = 0$, $\sum_{i=1}^{k} \lambda_i y^i = 0$, $\sum_{i=1}^{k} \lambda_i = 0$ is $\lambda_i = 0,\ \forall i \in [1,k]$. □

Definition 1.31 A polyhedron P is of *dimension* k (denoted by $\dim(P) = k$) if the maximum number of affinely independent elements in P is $k+1$. □

Definition 1.32 Face F of polyhedron P defines a *facet* of P if $\dim(F) = \dim(P) - 1$. The inequality describing face F is then called a *facet-defining inequality*, or a facet. □

The strength of an inequality is measured by the dimension of the face defined by it. An inequality defining a face of higher dimension is deemed a stronger inequality. Thus, the strongest inequalities define facets of the polyhedron.

1.5.1 Cutting-plane algorithm

The mixed-integer programming problem is $\mathcal{NP}$-hard (Garey and Johnson 1979). One approach to solving mixed-integer programs is using a relaxation algorithm. While any relaxation may be used, the most commonly-used variant is the linear programming (LP) relaxation; this is called a cutting-plane algorithm.

Definition 1.33 A *relaxation* of MIP is any maximization problem $z_R = \max\{z_R(x,y) : (x,y) \in S_R\}$ such that $S \subseteq S_R$ and $cx + hy \leq z_R(x,y),\ \forall (x,y) \in S$. □

The strength of a relaxation is usually measured by the proximity of z_R to z_{MIP}, either as a ratio or in terms of the difference. A relaxation R_1 is said to be stronger than relaxation R_2 if z_{R_1} is closer to z_{MIP} than z_{R_2}.

We obtain the linear programming relaxation by dropping the integrality constraints on x to obtain the set S_{LP}. The linear programming relaxation is often used since it is polynomially solvable (Khachian 1979), and is easy to implement computationally.

We define a collection of linear programs $\{LP^i\}$, with corresponding feasible sets $\{S^i_{LP}\}$. We use z^i_{LP} to denote the value of the optimal solution to LP^i, and $(x,y)^i_{LP} \in S_{LP}$ to denote the feasible solution that attains this optimum.

Definition 1.34 The *cutting-plane algorithm* works as follows.

Step 1. (Initialization) $i = 1$.

Step 2. (Termination) If $(x,y)^i_{LP} \in S$, then $(x,y)^i_{LP}$ is an optimal solution to MIP. Stop.

Step 3. (Refinement) Let $(\pi^i_N, \pi^i_P, \pi^i_0)$ be a valid inequality for S such that $\pi^i_N x^i_{LP} + \pi^i_P y^i_{LP} > \pi^i_0$. Define $S^{i+1}_{LP} = S^i_{LP} \cap \{(x,y) \in S^i_{LP} : \pi^i_N x + \pi^i_P y \leq \pi^i_0\}$. Set $i = i+1$, and go to Step 2. □

One attempts to solve successively stronger LP relaxations of S. The crucial part of the cutting-plane algorithm is the generation of strong valid inequalities that cut off fractional solutions. Therefore, the focus of any polyhedral study is to develop these valid inequalities that can be utilized in a cutting-plane algorithm.

It is known that the convex hull of any mixed-integer set $\mathrm{conv}(S)$ is a polyhedron (Meyer 1974), and thus can be described by a finite set of inequalities. If we know the polyhedral description of the convex hull of a mixed-integer set S, then we can solve the mixed-integer program as a linear program over the set $\mathrm{conv}(S)$. This description may be exponential in size; however, the optimization problem can still be solved in polynomial time by using the ellipsoid algorithm if there exists a polynomial separation algorithm for the inequalities in this description (Grötschel et al. 1981).

Definition 1.35 The *separation problem* for the polyhedron $\mathrm{conv}(S)$ is to determine whether any rational vector $(\bar{y}, \bar{x})$ is an element of S, and if not, to construct a valid inequality $\pi_N x + \pi_P y \leq \pi_0$ for S such that $\pi_N \bar{x} + \pi_P \bar{y} > \pi_0$. □

The separation problem is the refinement step of the cutting-plane algorithm. We attempt to separate the facets of $\mathrm{conv}(S)$, which are the strongest valid inequalities that

one may describe. However, since mixed-integer programming is $\mathcal{NP}$-hard, there exists no polynomial algorithm for the separation problem unless $\mathcal{P} = \mathcal{NP}$. Thus, $\text{conv}(S)$ is usually not known, and we approximate it as well as we can.

Fortunately, valid inequalities for any relaxation S_R are also valid for $\text{conv}(S)$. We study the convex hulls of the relaxations of S, namely $\text{conv}(S_R)$, and attempt to describe their facets.

We wish to study relaxations that are complex; thus yielding a stronger relaxation than the LP relaxation. At the same time, the relaxation must be easier to study than the mixed-integer program. Two relaxations often studied in the context of network design problems are the arc set and the cut set. These relaxations are formally defined in Section 1.6, and studied in detail throughout this dissertation.

An intuitive relaxation for any mixed-integer program, not just survivable network design problems, is the relaxation described by any one of its constraints. This is the mixed-integer knapsack set, and is the focus of our study in Chapter 7.

1.5.2 Strong valid inequalities

An effective method for computing strong valid inequalities for any polyhedron P is by lifting; where P is either $\text{conv}(S)$ or some relaxation $\text{conv}(S_R)$. We wish to obtain the facets of the polyhedron P.

Definition 1.36 A *restriction* of mixed-integer set S_R is the set $S_T \subseteq S_R$. □

Lifting refers to the process of extending simple valid inequalities of low-dimensional restrictions of P into valid inequalities for the polyhedron P. Introduced by Gomory (1969) in the context of the group problem, it has been used extensively to develop strong valid inequalities for mixed-integer programs. We limit our discussion to low-dimension restrictions obtained by setting a strict subset of the variables to their (upper or lower) bounds.

In particular, we consider the mixed-integer set $S_{L,U}$, defined as follows. For a subset T of $[1, n]$, let A_T denote the matrix of columns $A_i,\ i \in T$ of A. Let l_i and u_i be the smallest and largest values that any integer variable $x_i,\ i \in [1, n]$ may take in a feasible solution to

S_R; u_i may be infinite. Let (L, U, T) be a partition of N such that $u_i < \infty,\ \forall i \in U$, $t = |T|$. We use $L \subseteq N$ and $U \subseteq N$ to denote sets of variables fixed to their lower bounds and upper bounds, respectively. Then,

$$S_{L,U}(d) = \{x_T \in \mathbb{Z}^t_+, y \in \mathbb{R}^p_+ \ : \ A_T x_T + Gy \leq d\}.$$

describes the restriction obtained by fixing all variables in L to their lower bounds, and all variables in U to their upper bounds, when $d = b - A_L l_L - A_U u_U$.

While the restriction is defined for this particular value of d, we parametrize the definition of $S_{L,U}$ to describe all mixed-integer sets with the same coefficient matrix, but different constant terms on the right hand side.

Definition 1.37 For any valid inequality

$$\pi_T x_T + \pi_P y \leq \pi_0 \tag{1.1}$$

of $S_{L,U}(d)$, and $a \in \mathbb{R}^m$, we define its *lifting function* $\Phi : \mathbb{R}^m \mapsto \mathbb{R} \cup \{\infty\}$ as

$$\Phi(a) = \pi_0 - \max\{\pi_T x_T + \pi_P y \ : \ (x_T, y) \in S_{L,U}(d-a)\}.$$ □

For any valid inequality to a mixed-integer set, the lifting function can also be defined in terms of the value function $\omega : \mathbb{R}^m \mapsto \mathbb{R}$ to the set. The value function is a parametric mixed-integer optimization problem over the set, and is defined as follows.

Definition 1.38 Given any $\pi_T \in \mathbb{R}^t,\ \pi_P \in \mathbb{R}^p,\ a \in \mathbb{R}^m$, we define the *value function* $\omega : \mathbb{R}^m \mapsto \mathbb{R}$ for $S_{L,U}$ as

$$\omega(a) = \max\{\pi_T x_T + \pi_P y \ : \ (x_T, y) \in S_{L,U}(a)\}.$$ □

Thus $\Phi(a) = \pi_0 - \omega(d - a)$. The lifting function Φ is useful in describing strong valid inequalities for S_R in terms of the inequalities of $S_{L,U}$.

Definition 1.39 Any function $\phi : \mathbb{R}^m \mapsto \mathbb{R}$ is *super-additive* on $D \subseteq \mathbb{R}^m$ if $\phi(a_1) + \phi(a_2) \leq \phi(a_1 + a_2)$ for all $a_1, a_2 \in D$ such that $a_1 + a_2 \in D$. □

Suppose that $\phi : \mathbb{R}^m \mapsto \mathbb{R}$ is also a lower bound of Φ. Then, the valid inequality

(π_T, π_P, π_0) for $S_{L,U}$ can be extended to the valid inequality

$$\pi_T x_T + \sum_{i \in L} \phi(A_i)(x_i - l_i) + \sum_{i \in U} \phi(-A_i)(u_i - x_i) + \pi_P y \leq \pi_0 \tag{1.2}$$

for S_R. Furthermore, if $\phi(A_i) = \Phi(A_i)$ for all $i \in L$, $\phi(-A_i) = \Phi(-A_i)$ for all $i \in U$, and inequality (1.1) defines a k-dimensional face of $\mathrm{conv}(S_{L,U})$, then inequality (1.2) defines a face of $\mathrm{conv}(S_R)$ of dimension at least $k + |L| + |U|$ (Atamtürk 2004). Thus, facets for the restriction $S_{L,U}$ yield facets of S_R when this property is satisfied; this technique is often used to obtain facet-defining inequalities.

Super-additivity of ϕ allows us to lift all restricted variables in one iteration, in a technique known as sequence independent lifting. If we do not use a super-additive lower bound, then the coefficient of any restricted variable depends on the sequence in which it is lifted. Furthermore, the lifting problem needs to be solved at each iteration, for each variable. Thus, super-additive lifting functions reduce the computational burden in lifting.

Naturally, there is a trade-off in the strength of the inequality obtained if Φ is not super-additive, forcing us to use a super-additive lower bound. If $i \in L$ is binary, then the lifting function Φ gives us the coefficient binary variable x_i has when lifted first. In Chapter 7, we develop strong valid inequalities of mixed-integer knapsack set by lifting the facets of its restriction with one continuous and two integer variables, obtained by fixing the other variables to zero. An often-used technique for obtaining simple yet strong valid inequalities of knapsack sets is by the procedure of rounding. This procedure usually does not result in facets of the knapsack sets; except in very simple cases. Nevertheless, these inequalities are often strong (high dimensional faces) and are easily obtained.

Integer rounding (IR) is based on the simple principle that if a is integer and $a \leq b$, then $a \leq \lfloor b \rfloor$. For any pure-integer knapsack set

$$S_p = \{x \in \mathbb{Z}^n_+ \, : \, \sum_{i=1}^{n} a_i x_i \leq b\},$$

we can assume that $a_i \in \mathbb{Z}$ $i \in [1, n]$, by scaling. Thus, by integer rounding,

$$\sum_{i=1}^{n} a_i x_i \leq \lfloor b \rfloor.$$

Similarly, strong valid inequalities for mixed-integer sets can be obtained by the pro-

cess of mixed-integer rounding (MIR), as follows. Consider the mixed-integer set

$$S_m = \{x \in \mathbb{Z}^n_+,\ y \in \mathbb{R}^p_+ \ :\ \sum_{i=1}^{n} a_i x_i + \sum_{j=1}^{p} g_j y_j \le b\}.$$

Here, we do not assume that any of the data is integral. By mixed-integer rounding, we obtain the strong valid inequality

$$\sum_{i\in[1,n]\,:\,r(a_i)\le r(b)} \lfloor a_i \rfloor x_i + \sum_{i\in[1,n]\,:\,r(a_i)>r(b)} (\lfloor a_i \rfloor + \frac{r(a_i)-r(b)}{1-r(b)}) x_i + \sum_{j\in[1,p]\,:\,g_j<0} \frac{g_j}{1-r(b)} y_j \le \lfloor b \rfloor.$$

1.5.3 Branch-and-cut algorithm

Mixed-integer programs are usually solved by using a branch-and-bound algorithm. While the branch-and-bound algorithm may use any relaxation, we use the linear programming LP relaxation since it is the focus of our study.

We define $\mathcal{L}$ as a collection of mixed-integer programs, $\mathcal{L} = \{MIP^i\}$. Let z^i_{LP} denote the optimal solution to the LP relaxation of MIP^i (denoted by LP^i), and $(x,y)^i_{LP}$ denote the element of S^i_{LP} that attains this optimum. Let $\overline{z}^i_{MIP}$ represent the upper bound to MIP^i, and $\underline{z}_{MIP}$ indicate the current best solution to MIP.

Definition 1.40 The *branch-and-bound algorithm* using the LP relaxation is as follows.

Step 1. (Initialization) $\mathcal{L} = \{MIP\},\ S^0 = S,\ \overline{z}^0 = \infty,\ \underline{z}_{MIP} = -\infty$.

Step 2. (Termination) If $\mathcal{L} = 0$, then the solution (x^*, y^*) that yielded $\underline{z}_{MIP} = cx^* + hy^*$ is optimal. Stop.

Step 3. (Problem Selection and Relaxation) Choose any problem MIP^i from $\mathcal{L}$. Solve LP^i, and delete MIP^i from the list.

Step 4. (Pruning) If $z^i_{LP} \le \underline{z}_{MIP}$, then go to Step 2. Else, if $(x,y)^i_{LP} \notin S^i$, then go to Step 5. Else if $(x,y)^i_{LP} \in S^i$ and $z^i_{LP} > \underline{z}_{MIP}$, then let $\underline{z}_{MIP} = z^i_{MIP}$, delete from $\mathcal{L}$ all problems with $z^i_{MIP} \le \underline{z}_{MIP}$, and go to Step 2.

Step 5. (Division) Let $S^j = S \cap \{x_\ell \le \lfloor x^*_\ell \rfloor\}$ and $S^k = S \cap \{x_\ell \ge \lceil x^*_\ell \rceil\}$, for some variable $\ell \in N$ such that $x^*_\ell \notin \mathbb{Z}$. Divide S^i into S^j and S^k. Add problems MIP^j and MIP^k to $\mathcal{L}$, and set $\overline{z}^j = \overline{z}^k = z^i_{LP}$. □

There are other techniques of division, but variable dichotomy is the most used in

practice. There are many other open issues that are implementation-dependent. For instance, one may employ a variety of techniques to select the next problem, MIP^i.

The effectiveness of the branch-and-bound algorithm depends on the strength of its relaxation. Thus, we are interested in LP relaxations that approximate $\mathrm{conv}(S)$ well. Subsequently, branch-and-bound algorithms can be improved by adding cutting planes at each node of the branch-and-bound tree. This implementation results in the commonly used branch-and-cut algorithm. The effectiveness of these cutting planes can be measured empirically by the improvement in the LP relaxation solution at the root node. We refer to this quantity as the root improvement, and calculate it as $100 \times \frac{zroot - z^0_{LP}}{\underline{z}MIP - z^0_{LP}}$, where $zroot$ is the value of the LP relaxation at the root node before branching.

Crowder et al. (1983) demonstrated that the use of valid inequalities for single constraint relaxations of pure-binary programs were effective in solving them. Cutting plane methods have been successfully used in solving many mixed-integer programming problems. Another advantage of this approach is that valid inequalities to these relaxations may be useful in solving optimization problems over other complicated sets. For a recent survey on cutting planes in mixed-integer programming, see Marchand et al. (2002). One approach for generating such inequalities has its roots in disjunctive programming (Balas 1979, Jeroslow 1980). Based on these ideas, lift-and-project-cuts for mixed-binary programs have been developed by Balas et al. (1993) and Sherali and Adams (1994), and extended to mixed-integer programming recently by Owen and Mehrotra (2001).

Mixed-integer programs are usually solved using the LP relaxation in a branch-and-bound algorithm, see Definition 1.40. If we solve restricted LPs at each node of the tree, and price out variables as needed, then this implementation is called the branch-and-price algorithm.

For some successful applications of cutting-plane algorithms to several classes of network design problems, see Magnanti et al. (1993, 1995), Barahona (1996), Bienstock and Günlük (1996), Brockmüller et al. (1996), Bienstock and Muratore (2000), Atamtürk (2002), and Atamtürk and Rajan (2002). These works mainly study the polyhedron of the cut-set and arc-set relaxations of the problem.

1.6 Cut sets and arc sets

Let (A, B) be a non-empty partition of the nodes V, and let $G'_A = (A, F_A), G'_B = (B, F_B)$ be the sub-graphs defined by them. Let $[AB]$ be the edges with one end in A, the other in B; corresponding to these edges, let AB be the arcs directed from A to B, and BA be the arcs directed from B to A. Let K' be the set of commodities that have source and destination nodes s^k and t^k in the same sub-graph. Let $K_A = \{k \in K : s^k \in A,\ t^k \notin A\}$ and $K_B = \{k \in K : s^k \in B,\ t^k \notin B\}$. We also define $d_A = \sum_{k \in K_A} d^k$, $d_B = \sum_{k \in K_B} d^k$, and new flow variables $y^A_{ij} = \sum_{k \in K_A} y^k_{ij}$, $y^B_{ij} = \sum_{k \in K_B} y^k_{ij}$.

We obtain the cut-set relaxation as follows. We aggregate all nodes in A into a single master node, and similarly for B. Mathematically, this is done by adding all the flow balance constraints for all nodes in each partition. On doing this, all flow balance constraints for commodities in K' are eliminated. We also substitute y^A_{ij} and y^B_{ij} wherever possible in both the flow balance and capacity constraints. Commodities in K_A and K_B are aggregated into single flow balance constraints; $\sum_{ij \in AB} y^A_{ij} - \sum_{ij \in BA} y^A_{ij} = d_A$ and $\sum_{ij \in AB} y^B_{ij} - \sum_{ij \in BA} y^B_{ij} = -d_B$.

Any optimal solution to the relaxation will set flow variables y^k_{ij} to zero, for all $(ij) \in AB \cup BA,\ k \in K'$. Thus, we can remove them from the formulation. Finally, we assume that enough capacity exists on all the edges in $E \setminus [AB]$, and thus we can remove the capacity constraints corresponding to these edges from the formulation. Mathematically, we can do this by adding a sufficiently large positive constant to the right hand sides of the capacity constraints for these arcs. This gives us the 2-partition relaxation of the NDP with two commodities K_A and K_B. The single-commodity relaxation can be obtained by aggregating commodities K_A and K_B into a single commodity.

This procedure can be extended to yield k-partitions, for $k > 2$. Now, for a subset U of the set of partitions, we define $K_U = \{k \in K : s^k \in U,\ t^k \notin U\}$, and $d_U = \sum_{k \in K_U} d^k$. As for 2-partitions, AB is used to denote the set of arcs directed from A to B

In the presence of variables for failure-flow patterns in SDC and SDP, all cover constraints other than those corresponding to arcs across the various partitions (AB and BA

when $k = 2$) can be eliminated. This is because these variables have no costs in the objective, and since the flows on all other arcs can be set to zero in any optimal solution to the k-partition relaxation. We use $\bar{\mathcal{C}}$ to denote set of directed cycles that cross the partition; let $\mathcal{C}' = \mathcal{C} \setminus \bar{\mathcal{C}}$.

The arc-set relaxation is obtained by considering the polyhedron for the capacity constraint of any arc.

Chapter 2

Network design problem: A polyhedral study

Since the capacitated network design problem with no survivability requirements (NDP) can be formulated as a mixed-integer program; we focus our attention on polyhedral approaches to solve this formulation. In particular, we review existing research on the cut-set polyhedron in Section 2.2, and then present our work on the arc-set polyhedra. Most of these results, presented in Section 2.3, have been published in Atamtürk and Rajan (2002). In Section 2.4, we report computational results that utilize both the cut-set and arc-set inequalities. The experiments show that adding these inequalities in a branch-and-cut framework often reduced our computation time in solving the NDP by an order of magnitude. In subsequent chapters, we incorporate survivability.

2.1 Problem formulation

Before we introduce the NDP, we define some related but simpler problems. We first present the multi-commodity flow problem, which is subsequently extended to the fixed-charge network flow problem, and finally to NDP.

Definition 2.1 *Multi-commodity flow problem (MFP)*: Given a directed network, flow costs, a capacity for each edge, and a set of commodities (given in terms of their origin-

destination pairs and demands[1]), we wish to route the commodities so that the flow on any arc is at most the capacity on the edge and all demands are met, at minimum total flow cost. □

In this problem, we are only concerned with the minimum cost routing of commodities on an existing network. MFP is a linear programming problem, and hence can be solved in polynomial time. An extension of this problem is when we incur a fixed cost for sending flow on a particular arc, presented next.

Definition 2.2 *Fixed-charge network flow problem (FCP)*: Given a directed network, flow costs, fixed costs, a capacity for each edge, and a set of commodities (given in terms of their origin-destination pairs and demands), we wish to route the commodities so that the net flow on any arc that is used is at most the capacity on the edge and all demands are met, at minimum total cost; in addition to the flow costs, we incur a fixed cost for each edge that we use to route flow on. □

In FCP, we incur costs not only for routing flows but also for the edges used. Hence some design aspects are incorporated into the problem. FCP is $\mathcal{NP}$-hard because it contains as a special case the $\mathcal{NP}$-hard Steiner Tree Problem (Garey and Johnson 1979). The capacities of the edges we choose are predefined. We can extend the problem by including capacity selection in the decision-making process. We enforce that we can add only integer multiples of capacity, else the problem is trivial; we simply solve the shortest path problems for each commodity and install exactly as much capacity as required.

In NDP, we incur a cost for every unit of flow routed, and a capacity installation cost for each integer unit of capacity installed. We no longer incur a fixed cost for using an edge though it is possible to define the problem so. We may also have more than one capacity type, each with its own cost, presumably with economies of scale. In a telecommunications network, the capacity types are fiber-optic cables with different bandwidth capacities and installation costs. We are now ready to define NDP.

Definition 2.3 *Network design problem (NDP)*: Given a directed network, a set of installable capacity types, flow costs, capacity installation costs for each edge, and a set

[1]Demand refers to quantity of the commodity to be routed between its origin and destination nodes.

of commodities (given in terms of their origin-destination pairs and demands), we wish to route the commodities so that the net flow on any arc is at most the capacity installed on that edge and all demands are met, at minimum total cost; in addition to the flow costs, we incur a capacity installation cost for every unit of capacity we install. □

Let $G = (V, E)$ be an undirected graph with node set V and edge set E. Let F be the set of all ordered pairs (arcs) from E, i.e., $F = \{(ij), (ji) : [ij] \in E\}$. We use (ij) to denote the arc from node i to node j, and $[ij]$ to denote the edge between nodes i and j. We use $G' = (V, F)$ to denote the directed graph. Let K be the set of commodities. Let $\{(s^k, t^k, d^k)\}_{k \in K}$ be the commodity triples of source and destination nodes s^k and t^k, and d^k be the supply at s^k for t^k, $k \in K$. Let b_i^k be the supply of commodity k at node i, i.e., $b_{s^k}^k = d^k$, $b_{t^k}^k = -d^k$, and $b_i^k = 0$ for $i \in V \setminus \{s^k, t^k\}$.

We define variable y_{ij}^k as the fraction of commodity k routed through arc $(ij) \in F$. Let e_{ij}^k be the cost associated with routing each unit of commodity $k \in K$. Let T be the set of installable capacity types. The capacity q^t for type $t \in T$ is the maximum amount of demand that can be routed using one unit of capacity type t. We define the capacity variable $x_{[ij]}^t$ as the amount of capacity of type $t \in T$ installed on edge $[ij]$. Let $h_{[ij]}^t$ be the cost of installing unit capacity of type $t \in T$ on edge $[ij] \in E$. We use $w_{[ij]}^0$ to denote the pre-existing capacity on edge $[ij] \in E$. NDP can be formulated as follows.

$$\min \sum_{(ij) \in F} \sum_{k \in K} e_{ij}^k y_{ij}^k + \sum_{[ij] \in E} \sum_{t \in T} h_{[ij]}^t x_{[ij]}^t$$

$$s.t.: \quad \sum_{(ij) \in F} d^k y_{ij}^k - \sum_{(ji) \in F} d^k y_{ji}^k = b_i^k \qquad \forall i \in V,\ \forall k \in K \qquad (2.1)$$

$$\sum_{k \in K} d^k y_{ij}^k \leq \sum_{t \in T} q^t x_{[ij]}^t + w_{[ij]}^0 \quad \forall (ij) \in F \qquad (2.2)$$

$$x_{[ij]}^t \in \mathbb{Z}_+ \qquad \forall [ij] \in E,\ \forall t \in T$$

$$y_{ij}^k \in \mathbb{R}_+ \qquad \forall (ij) \in F,\ \forall k \in K.$$

Constraints (2.1) ensure that all demands are met, and constraints (2.2) ensure that capacity installed on edge $[ij]$ is large enough to accommodate the flow routed on arc (ij). We will use this formulation to compare with formulations of other survivability frame-

works in subsequent chapters. For the rest of this dissertation, we assume that there is only one capacity type, and drop the capacity subscript t. This allows us to assume that all demand data are scaled such that the capacity unit is 1.

The problem can be differentiated into the splittable flow network design problem (when there are no restrictions on how the commodities must be routed) and unsplittable flow network design problem (when flow of a commodity is restricted to run through a single path along the network).

For a review of NDP, see Balakrishnan et al. (1997). NDP is $\mathcal{NP}$-Hard even for the single commodity case (Chopra et al. 1998). Furthermore, it is difficult to solve NDP optimally; branch-and-bound algorithms are not effective in solving even small instances of NDP. In Section 2.4, we see that many of the sample instances with 25 nodes and 41 edges, on average, are not solved to optimality within an hour of computation; we also present some results that illustrate the effectiveness of cutting planes in solving NDP.

2.2 Cut-set polyhedra

For a precise definition of the cut set, see Section 1.6. Inequalities developed from the cut-set polyhedra are known to improve the LP relaxations of network design problems significantly (Magnanti and Mirchandani 1993, Bienstock and Günlük 1996, Günlük 1999, Atamtürk 2002); however, their separation problem is $\mathcal{NP}$-hard (Bienstock 2001).

Magnanti et al. (1993) were the first to develop partition inequalities for NDP. The authors study a two-node problem, and show that the cut-set inequality is facet-defining for this special case. In fact, it completely describes the convex hull of the projection on to the space of the integral capacity variable. This inequality can be generalized to any 2-partition (A, B) of the network. For the partition (A, B) of the nodes V, let $[AB]$ be the edges with one end in A, the other in B. Let $K_A = \{k \in K : s^k \in A, t^k \in B\}$. Defining $d_A = \sum_{k \in K_A} d^k$, we obtain the cut-set inequality

$$\sum_{[ij] \in [AB]} x_{[ij]} \geq \lceil d_A \rceil. \tag{2.3}$$

For the 3-node problem, the authors describe the 3-partition inequality, and show that the 2-partition (cut-set) inequalities and the 3-partition inequality completely describe the convex hull of the projection on the space of the capacity variables x.

In Magnanti et al. (1995), the authors extend this work to the case with two capacity types, where the capacity of the larger type is an integral multiple of the other. In Magnanti and Mirchandani (1993), the authors consider network design problems with a single commodity and one, two, or three capacity types. Barahona (1996) present a cutting-plane algorithm based on the cut-set inequalities, where they formulate its separation problem as a max-cut problem.

Bienstock and Günlük (1996) also investigate the polyhedral structure of NDP and develop a cutting-plane algorithm. Using mixed-integer rounding, they generalize the cut-set inequalities to include some continuous flow variables. Corresponding to edges in the partition $[AB]$, let AB be the arcs directed from A to B, and BA be the arcs directed from B to A. For $H \subseteq AB$, they present the flow cut-set inequality

$$r(d_A) \sum_{ij \in H} x_{[ij]} + \sum_{k \in K_A} d^k \sum_{(ij) \in AB \setminus H} y_{ij}^k \geq r(d_A) \lceil d_A \rceil. \tag{2.4}$$

They also consider the case with two capacity types. Chopra et al. (1998) generalize these inequalities further to include flow variables in both directions across a cut set.

Bienstock et al. (1998) compare cutting-plane algorithms based on the natural formulation of NDP, and a reformulation in terms of the capacity variables, respectively. They show that the two formulations are comparable and yield effective algorithms for solving real-life problems when used in conjunction with cutting planes in a branch-and-cut framework. Günlük (1999) introduces new families of partition inequalities with continuous variables for the problem with two capacity types.

Atamtürk (2002) studies the cut-set polyhedron in detail, and presents a complete description of the single-commodity single-capacity cut set. He extends the analysis to multi-capacity and multi-commodity cut-set polyhedra, and reports computational experiments that underscore the effectiveness of these cut-set inequalities in solving NDP.

2.3 Arc-set polyhedra

In this section, we study the polyhedra of splittable and unsplittable single arc-set relaxations of the NDP, defined precisely in Section 1.6. We investigate the optimization problems over these sets and the separation and lifting problems of valid inequalities for them. In particular, we present a linear-time separation algorithm for the residual capacity inequalities (Magnanti et al. 1993) and show that the separation problem of c-strong inequalities (Brockmüller et al. 1996) is $\mathcal{NP}$-hard, but can be solved over the subspace of fractional variables only. We introduce two classes of inequalities for the unsplittable flow problems that generalize the c-strong inequalities. In Section 2.4, we present a summary of computational experiments with a branch-and-cut algorithm for the NDP to empirically test the effectiveness of the results presented here.

2.3.1 Introduction

In many applications of the NDP, the flow of a commodity is restricted to run through a single path along the network. This is the case, for instance, in telecommunication networks running asynchronous transfer mode (ATM) protocol, production-distribution with single sourcing, and express package shipping networks (Gavish and Altinkemer 1990, Barnhart et al. 2000). These problems are generally formulated using a binary flow variable y^k_{ij} for each commodity-arc pair $(k,(ij))$ that takes on a value of 1 if the commodity uses the arc, 0 otherwise. If the flow of commodities is allowed to be split among several paths, then the binary restriction on the flow variables is dropped, and we have $0 \leq y^k_{ij} \leq 1$. In either case, for each arc (ij) of the network there is a capacity constraint of the form

$$\sum_{k \in K} d^k y^k_{ij} \leq w^0_{[ij]} + x_{[ij]}. \tag{2.5}$$

In this section, we study the convex hull of solutions to constraints of the form (2.5). We drop the index for arcs and edges for the rest of the chapter. We investigate optimization problems over the splittable and unsplittable arc-set polyhedra and the separation and lifting problems of valid inequalities for them. Defining the feasible regions for these

polyhedra formally,

$$\mathcal{D}_S = \{0 \leq y^k \leq 1 \; k \in K, \; x \in \mathbb{Z}\} \text{ and } \mathcal{D}_U = \{y \in \{0,1\}^{|K|}, \; x \in \mathbb{Z}\},$$

the sets we consider are defined as

$$\text{Splittable flow arc set:} \quad \mathcal{F}_S = \{(y,x) \in \mathcal{D}_S \; : \; \sum_{k \in K} d^k y^k \leq w^0 + x\},$$

$$\text{Unsplittable flow arc set:} \quad \mathcal{F}_U = \{(y,x) \in \mathcal{D}_U \; : \; \sum_{k \in K} d^k y^k \leq w^0 + x\}.$$

The unsplittable flow arc set $\mathcal{F}_U$ is a relaxation of the feasible region of the more familiar binary knapsack set obtained by introducing an integer variable x with no bounds. Although the formulation of $\mathcal{F}_U$ is quite similar to the formulation of the binary knapsack set, we show in Section 2.3.4 that its structure is quite different from the latter. The splittable flow arc set $\mathcal{F}_S$ is the relaxation of $\mathcal{F}_U$ obtained by allowing the binary variables to take on any real value between 0 and 1. Finally, we let $\mathcal{F}_L$ denote the relaxation of $\mathcal{F}_S$ obtained by dropping the integrality restriction on y as well. $\mathcal{F}_L$ is the LP relaxation of both $\mathcal{F}_S$ and $\mathcal{F}_U$. The following proposition presents an important property of the extreme points of $\mathcal{F}_L$.

Proposition 2.4 For any extreme point $(\bar{y}, \bar{x})$ of $\mathcal{F}_L$, let $\bar{H} = \{k \in K \; : \; 0 < \bar{y}^k < 1\}$. Then, $|\bar{H}| \leq 1$.

Proof See Appendix B. □

Without loss of generality, we assume that $d^k > 0$ for all $k \in K$, since if $d^k < 0$, y^k can be complemented and if $d^k = 0$, y^k can be dropped. We do not impose a sign restriction on the constant term w^0, as this term may take on negative or non-negative values for the separation and lifting problems defined in following sections.

2.3.2 Related work and our contributions

The polyhedral structure of the binary knapsack set, a restriction of $\mathcal{F}_U$, has been studied extensively (Balas 1975, Hammer et al. 1975, Wolsey 1975, Padberg 1979, Zemel 1989, Weismantel 1997, Gu et al. 1998). Another set related to $\mathcal{F}_U$, the binary knapsack set

with a single continuous variable, obtained by replacing integral variable x with a non-negative continuous variable (Marchand and Wolsey 1999, Richard et al. 2002).

Magnanti et al. (1993) study the facial structure of $\mathrm{conv}(\mathcal{F}_S)$ when $w^0 = 0$. They define an exponential class of valid inequalities, called the residual capacity inequalities, and show that the residual capacity inequalities and the constraints of $\mathcal{F}_L$ are sufficient to describe $\mathrm{conv}(\mathcal{F}_S)$. However, no exact polynomial-time separation algorithm for these inequalities was known until now. In Section 2.3.3 we describe a linear-time algorithm for separating the residual capacity inequalities.

For $\mathcal{F}_U$, Brockmüller et al. (1996) introduce the c-strong inequalities and characterize the necessary and sufficient conditions under which the c-strong inequalities are facet-defining. van Hoesel et al. (2002) study $\mathcal{F}_U$ when $w^0 = 0$ as well. In Section 2.3.4, we prove that the c-strong inequalities constitute all facet-defining inequalities of $\mathrm{conv}(\mathcal{F}_U)$ of the form $\sum_{k\in K} \pi^k y^k \leq \pi^0 + x$ with integral coefficients. We show that the separation problem of c-strong inequalities is $\mathcal{NP}$-hard and that it is sufficient to solve this separation problem over the subspace of fractional variables only. Furthermore, we introduce two classes of inequalities, both of which include the c-strong inequalities as a special case.

2.3.3 Splittable flow arc set

2.3.3.1 Optimization problem

To motivate the separation problem of the splittable flow arc set $\mathcal{F}_S$, we start with the related optimization problem. Magnanti et al. (1993) state that the optimization of a linear function over $\mathcal{F}_S$ can be solved efficiently using an incremental strategy. Here, we present another simple algorithm, which is also used in Section 2.4 for approximate lifting of valid inequalities for $\mathcal{F}_U$. We consider a maximization problem and without loss of generality assume that the objective coefficient of the capacity variable x is negative since otherwise the problem is unbounded. Furthermore, we can assume that this coefficient is -1 by scaling; i.e., we consider the problem

$$(\text{SFP}) \quad \zeta = \max\{\sum_{k\in K} c^k y^k - x \,:\, \sum_{k\in K} d^k y^k \leq w^0 + x,\ (y,x) \in \mathcal{D}_S\}.$$

If $c^k \leq 0$ for some $k \in K$, then given any optimal solution with $y^k > 0$, there exists another optimal solution that is identical to the former except that $y^k = 0$. Therefore, we may assume that $c^k > 0$ for all $k \in K$.

Suppose the variables are indexed so that $\frac{c^1}{d^1} \geq \frac{c^2}{d^2} \geq \cdots \geq \frac{c^{|K|}}{d^{|K|}}$, ties broken arbitrarily. Let $D^k = \sum_{h=1}^{k} d^h - w^0$ for $k \in K$, $D^0 = -w^0$, and i be the largest index with $\frac{c^k}{d^k} \geq 1$. If $\frac{c^1}{d^1} < 1$, then let $i = 0$. There exists an optimal solution to the LP relaxation of SFP with all positive y^k in the above order and $x = D^i$. Then, by concavity of $\zeta(x)$, there exists an optimal solution (y^*, x^*) to SFP such that $x^\star = \lfloor D^i \rfloor$ or $x^* = \lceil D^i \rceil$. Hence the computational burden of finding an optimal solution to SFP is sorting the variables in non-increasing order of $\frac{c^k}{d^k}$, which can be done in $\mathcal{O}(|K| \log |K|)$. We state this as the following proposition.

Proposition 2.5 The optimization problem SFP can be solved in $\mathcal{O}(|K| \log |K|)$. □

Because of the polynomial equivalence of optimization and separation for a polyhedron (Grötschel et al. 1981), the separation problem of $\mathrm{conv}(\mathcal{F}_S)$ must also be solvable in polynomial time. Next, we show that $\mathrm{conv}(\mathcal{F}_S)$ can be separated in linear time.

2.3.3.2 Separation problem

For $H \subseteq K$ let $\eta_H = \lceil d(H) - w^0 \rceil$ and $r_H = r(d(H) - w^0)$. Magnanti et al. (1993) show that for any $H \subseteq K$ the residual capacity inequality

$$\sum_{k \in H} d^k (1 - y^k) \geq r_H (\eta_H - x) \tag{2.6}$$

is valid for $\mathcal{F}_S$ when $w^0 = 0$. Inequality (2.6) is valid for $\mathcal{F}_S$ when $r_H = 0$ or $x \geq \eta_H$ since $y^k \leq 1$ for all $k \in K$. It is also valid otherwise, since

$$\begin{aligned}
\sum_{k \in H} d^k (1 - y^k) \geq d(H) - w^0 - x &= \eta_H - (1 - r_H) - x \\
&= (1 - r_H)(\eta_H - 1) + r_H \eta_H - x \\
&\geq (1 - r_H) x + r_H \eta_H - x \\
&= r_H (\eta_H - x).
\end{aligned}$$

The residual capacity inequality can also be viewed as a mixed-integer rounding inequality from a suitable relaxation of $\mathcal{F}_S$ (Marchand and Wolsey 2001).

Magnanti et al. (1993) prove that the residual capacity inequalities together with the constraints of $\mathcal{F}_L$ are sufficient to describe $\mathrm{conv}(\mathcal{F}_S)$ when $w^0 = 0$. This result extends trivially to the case $w^0 \neq 0$ as well.

Proposition 2.6 (Magnanti et al. (1993)) The residual capacity inequalities along with the constraints of $\mathcal{F}_L$ completely describe $\mathrm{conv}(\mathcal{F}_S)$ for all values of w^0. □

Since the constraints of $\mathcal{F}_L$ can be checked for violation in linear time, the separation problem of $\mathrm{conv}(\mathcal{F}_S)$ reduces to either finding a residual capacity inequality violated by $(\bar{y}, \bar{x})$, or concluding that $(\bar{y}, \bar{x}) \in \mathrm{conv}(\mathcal{F}_S)$, given a point $(\bar{y}, \bar{x}) \in \mathcal{F}_L$.

Without loss of generality, we may assume that $\bar{x} \notin \mathbb{Z}$. No $(\bar{y}, \bar{x}) \in \mathcal{F}_L$ with $\bar{x} \in \mathbb{Z}$ violates a residual capacity inequality as residual capacity inequalities are valid for $\mathcal{F}_S$.

We therefore look for $H \subseteq K$ such that $\sum_{k \in H} d^k(1 - \bar{y}^k) < r_H(\eta_H - \bar{x})$. We are interested in only H with $r_H > 0$ and $\eta_H \geq \bar{x} + 1$. Defining $y \in \{0,1\}^{|K|}$ to be the characteristic vector of H, the separation problem can be formulated as

$$
\begin{aligned}
\varsigma = \min \ & \sum_{k \in K} d^k(1 - \bar{y}^k) y^k - r_H(\eta_H - \bar{x}) \\
s.t.: \ & \sum_{k \in K} d^k y^k = w^0 + (\eta_H - 1) + r_H \\
& 0 < r_H < 1 \\
& \bar{x} + 1 \leq \eta_H, && \eta_H \in \mathbb{Z} \\
& y^k \in \{0,1\} && k \in K.
\end{aligned}
$$

The point $(\bar{y}, \bar{x})$ violates the residual capacity inequality corresponding to an optimal (y, η_H, r_H) if $\varsigma < 0$. Otherwise, $(\bar{y}, \bar{x})$ violates no residual capacity inequality. This is a nonlinear mixed-integer optimization problem, which is hard to solve in general.

Lemma 2.7 A point $(\bar{y}, \bar{x}) \in \mathcal{F}_L$ does not violate any residual capacity inequality (2.6) with $\eta_H \leq \bar{x}$ or $\eta_H \geq \bar{x} + 1$.

Proof See Appendix B. □

From Lemma 2.7, any residual capacity inequality violated by $(\bar{y}, \bar{x})$ has $\eta_H = \lceil \bar{x} \rceil$. Since $\bar{x} \notin \mathbb{Z}$, we have $\lfloor \bar{x} \rfloor = \eta_H - 1$. After fixing η_H to $\lceil \bar{x} \rceil$, the separation problem can be formulated as a linear mixed-binary optimization problem:

$$\begin{aligned} \varsigma = \min \ & \sum_{k \in K} d^k (1 - \bar{y}^k) y^k - r_H (\lceil \bar{x} \rceil - \bar{x}) \\ s.t.: \ & \sum_{k \in K} d^k y^k = w^0 + \lfloor \bar{x} \rfloor + r_H \\ & r_H < 1, \ r_H \in \mathbb{R}_{++} \\ & y^k \in \{0, 1\} \qquad\qquad k \in K. \end{aligned}$$

Eliminating the bounded continuous variable r_H, we rewrite the separation problem as a binary problem with two strict inequalities

$$\begin{aligned} \varsigma = \min \ & \sum_{k \in K} d^k (1 - \bar{y}^k - \lceil \bar{x} \rceil + \bar{x}) y^k + (\lceil \bar{x} \rceil - \bar{x})(w^0 + \lfloor \bar{x} \rfloor) \\ \text{(SP)} \qquad s.t.: \ & w^0 + \lfloor \bar{x} \rfloor < \sum_{k \in K} d^k y^k < w^0 + \lceil \bar{x} \rceil \\ & y^k \in \{0, 1\} \qquad\qquad k \in K. \end{aligned}$$

Next we show that to find a violated residual capacity inequality, it is sufficient to consider only variables with a negative coefficient in the objective function. Let

$$T = \{k \in K \ : \ 1 - \bar{y}^k < \lceil \bar{x} \rceil - \bar{x}\}.$$

Lemma 2.8 If there exists a residual capacity inequality (2.6) violated by a fractional point $(\bar{y}, \bar{x}) \in \mathcal{F}_L$, then there exists one given by $H \subseteq T$.

Proof See Appendix B. □

Lemma 2.9 If $d(T) \leq w^0 + \lfloor \bar{x} \rfloor$ or $d(T) \geq w^0 + \lceil \bar{x} \rceil$, then there exists no residual capacity inequality violated by $(\bar{y}, \bar{x}) \in \mathcal{F}_L$.

Proof See Appendix B. □

Lemmas 2.7, 2.8, and 2.9 allow us to separate the residual capacity inequalities easily. Next, we develop a simple procedure that does so. Furthermore, since the residual capacity inequalities, together with the inequalities of $\mathcal{F}_L$, describe $\mathrm{conv}(\mathcal{F}_S)$, the following

procedure separates $(\bar{y}, \bar{x}) \in \mathcal{F}_L \setminus \mathcal{F}_S$ from $\text{conv}(\mathcal{F}_S)$: If $w^0 + \lfloor \bar{x} \rfloor < d(T) < w^0 + \lceil \bar{x} \rceil$ $\sum_{k \in T} d^k(1 - \bar{y}^k - \lceil \bar{x} \rceil + \bar{x}) + (\lceil \bar{x} \rceil - \bar{x})(w^0 + \lfloor \bar{x} \rfloor) < 0$, then the inequality $\sum_{k \in T} d^k(1 - y^k) \geq r_T(\eta_T - x)$ is violated by $(\bar{y}, \bar{x})$. Otherwise, there exists no residual capacity inequality violated by $(\bar{y}, \bar{x})$. This procedure can be performed in linear time, and is stated as the following theorem.

Theorem 2.10 The separation problem for the residual capacity inequalities (2.6) can be solved in $\mathcal{O}(|K|)$. □

2.3.4 Unsplittable flow arc set

2.3.4.1 Optimization problem

Even though our ultimate goal is to find strong valid inequalities for the unsplittable flow arc set $\mathcal{F}_U$, it is helpful to study the maximization of a linear function over $\mathcal{F}_U$ as a first step.. As for $\mathcal{F}_S$ we may assume that the objective coefficient of the capacity variable x is -1 and state the problem as

$$\text{(UFP)} \qquad \xi = \max\{\sum_{k \in K} c^k y^k - x \ : \ \sum_{k \in K} d^k y^k \leq w^0 + x, \ (y, x) \in \mathcal{D}_U\}.$$

UFP is a relaxation of the binary knapsack problem. Below we present properties of optimal solutions of UFP that will be useful when studying $\text{conv}(\mathcal{F}_U)$ in Section 2.3.4.3.

Proposition 2.11 states that in any optimal solution (y^*, x^*), the value of x^* is completely determined by y^*. Proposition 2.12 characterizes conditions under which the values of some of the binary y variables can be fixed.

Proposition 2.11 In any optimal solution (y^*, x^*) to UFP, $x^* = \lceil \sum_{k \in K} d^k y^{k*} - w^0 \rceil$.

Proof See Appendix B. □

Proposition 2.12 UFP has an optimal solution $(y^\star, x^\star)$ such that

$$y^{k\star} = \begin{cases} 1 & if \ \ c^k \geq \lceil d^k \rceil, \\ 0 & if \ \ c^k \leq \lfloor d^k \rfloor, \end{cases} \quad \text{for } k \in K.$$

Proof See Appendix B. □

From Proposition 2.12, all binary variables except the ones with $\lfloor d^k \rfloor < c^k < \lceil d^k \rceil$ can be eliminated from UFP since their optimal values can be determined *a priori*.

Corollary 2.13 UFP can be solved in $\mathcal{O}(|K|)$ if either $c^k \in \mathbb{Z}$ or $d^k \in \mathbb{Z}$ for all $k \in K$. □

Theorem 2.14 UFP is $\mathcal{NP}$-hard for any fixed value of w^0.

Proof See Appendix B. □

Remark 2.15 Theorem 2.14 states that UFP remains $\mathcal{NP}$-hard for any fixed value of w^0. In Section 2.3.4.3 we will see that w^0 may take on negative values in the separation problem of c-strong inequalities. □

Now we present a canonical form of UFP. Without loss of generality, we assume that $\lfloor d^k \rfloor < c^k < \lceil d^k \rceil$ for all $k \in K$, since all other variables can be eliminated from the problem by Proposition 2.12. We can further simplify the problem such that the data consists of only the fractional parts of d^k and c^k. For $k \in K$ let $f^k = r(d^k) = d^k - \lfloor d^k \rfloor$ and $g^k = r(c^k) = c^k - \lfloor c^k \rfloor$. Now, let $f^0 = w^0 - \lfloor w^0 \rfloor$ and define

$$(\text{UFP}_f) \quad \xi_f = \max\{\sum_{k \in K} g^k y^k - x \ : \ \sum_{k \in K} f^k y^k \leq f^0 + x, \ (y, x) \in \mathcal{D}_U\}.$$

Proposition 2.16 UFP$_f$ is related to UFP in the sense that $H \subseteq K$ maximizes UFP if and only if H maximizes UFP$_f$. Furthermore, $\xi_f = \xi - \lfloor w^0 \rfloor$.

Proof See Appendix B. □

Proposition 2.16 allows us to study UFP$_f$ instead of UFP.

2.3.4.2 Optimization algorithm

Next, we present a pseudo-polynomial algorithm for solving UFP$_f$, which is used to show strongly polynomial-time lifting of a subclass of the lifted knapsack cover inequalities in Theorem 2.29. Let λ be a common multiple of the denominators of the rational numbers $f^k, \ k \in [0, |K|]$. By multiplying f^k by λ, the constraint of UFP$_f$ can be written with only integral coefficients. Since $f^k < 1$, x can be at most $\lceil f(K) - f^0 \rceil \leq |K|$ in any optimal solution to UFP$_f$. For $\nu \in [0, \lceil f(K) - f^0 \rceil]$, consider an optimal solution $y^\star$ to the binary

knapsack problem

$$(\text{KP1}(\nu)) \quad \varsigma(\nu) = \max\{\sum_{k\in K} g^k y^k \ : \ \sum_{k\in K} \lambda f^k y^k \leq \lambda(f^0+\nu),\ y \in \{0,1\}^{|K|}\}.$$

Since $g^k > 0$ and $f^k < 1$ for all $k \in K$, we have $\lambda f^0 + \lambda(\nu-1) < \sum_{k\in K} \lambda f^k y_k^\star \leq \lambda f^0 + \lambda\nu$. Hence, $\xi_f = \max_{\nu\in[0,\lceil f(K)-f^0\rceil]}\{\varsigma(\nu)-\nu\}$. There are at most $|K|+1$ of these related knapsack problems. They can be solved in a total of $\mathcal{O}(\lambda|K|^2)$ by dynamic programming, since we complete the computations required for solving KP1(ν) for all $\nu \in [0,|K|-1]$ when solving KP1($|K|$). Alternatively, let μ be a common multiple of the denominators of the fractional numbers $g^k,\ k \in [1,|K|]$. Instead of solving KP1(ν), we may solve the dual knapsack problem

$$(\text{KP2}(t)) \quad \omega(t) = \min\{\sum_{k\in K} f^k y^k \ : \ \sum_{k\in K} \mu g^k y^k \geq t,\ y \in \{0,1\}^{|K|}\}.$$

so that $\varsigma(\nu) = \omega(t)$ if $\omega(t) \leq \lambda(f^0+\nu) < \omega(t+1)$. Since KP2($t$) is infeasible for $t \geq \mu|K| > \mu g(K)$, $\varsigma(\nu)$ for all $\nu \in [0,|K|]$ can be computed in $\mathcal{O}(\mu|K|^2)$. Thus, we have proved the following theorem about the complexity of UFP.

Theorem 2.17 UFP can be solved in $\mathcal{O}(\min\{\lambda,\mu\}|K|^2)$. □

2.3.4.3 Valid inequalities

In Section 2.3.4.3, we discuss three classes of valid inequalities for $\mathcal{F}_U$. The first class is the c-strong inequalities introduced by Brockmüller et al. (1996). The next two classes are new and both of them subsume the c-strong inequalities. Before describing specific valid inequalities, we present some general properties of $\text{conv}(\mathcal{F}_U)$ that will be useful in the analysis. First, the convex hull of $\mathcal{F}_U$, $\text{conv}(\mathcal{F}_U)$, is full-dimensional and inequalities $y^k \geq 0$ and $y^k \leq 1$ for all $k \in K$ are facet-defining for $\text{conv}(\mathcal{F}_U)$. We call these inequalities the trivial valid inequalities of $\text{conv}(\mathcal{F}_U)$. Next, we provide bounds on the coefficients of non-trivial facet-defining valid inequalities of $\text{conv}(\mathcal{F}_U)$.

Proposition 2.18 (Atamtürk and Rajan (2002), van Hoesel et al. (2002)) Any non-trivial facet-defining inequality $\sum_{k\in K} \pi^k y^k \leq \pi^0 + x$ of $\text{conv}(\mathcal{F}_U)$ has $\lfloor d^k \rfloor \leq \pi^k \leq \lceil d^k \rceil$ for all $k \in K$ and $\pi^0 = \max\{\sum_{k\in K} \pi^k y^k - x \ : \ (y,x) \in \mathcal{F}_U\}$. □

In terms of the fractional components of d and w^0, we define the set

$$\mathcal{F}_{Uf} = \{(y, x) \in \mathcal{D}_U : \sum_{k \in K} f^k y^k \leq f^0 + x\}.$$

The following proposition follows trivially from Propositions 2.16 and 2.18.

Proposition 2.19 An inequality $\sum_{k \in K} \pi^k y^k \leq \pi^0 + x$ with $\lfloor d^k \rfloor \leq \pi^k \leq \lceil d^k \rceil$ is valid for $\mathcal{F}_U$ if and only if $\sum_{k \in K} (\pi^k - \lfloor \pi^k \rfloor) y^k \leq \pi^0 - \lfloor w^0 \rfloor + x$ is valid for $\mathcal{F}_{Uf}$.

Proof See Appendix B. □

Remark 2.20 From Proposition 2.18 when looking for strong valid inequalities for $\mathcal{F}_U$, we can restrict our attention to inequalities $\sum_{k \in K} \pi^k y^k \leq \pi^0 + x$ with $\lfloor d^k \rfloor \leq \pi^k \leq \lceil d^k \rceil$ for all $k \in K$. But then, from Proposition 2.19, instead of working with $\mathcal{F}_U$, we can work with $\mathcal{F}_{Uf}$ defined using the fractional parts of the data. □

From Propositions 2.18 and 2.19, we assume that $0 < d^k < 1$ for all $k \in K$ and $0 \leq w^0 < 1$; so $\mathcal{F}_U = \mathcal{F}_{Uf}$ for the rest of Chapter 2. Consequently, from Proposition 2.18, $0 \leq \pi^k \leq 1$ for all $k \in K$ for all non-trivial facet-defining inequalities of $\text{conv}(\mathcal{F}_U)$.

2.3.4.3.1 c-strong inequalities

For $H \subseteq K$ let $c_H = |H| - \lceil d(H) - w^0 \rceil$. H is said to be maximal c-strong if $c_{H \setminus \{i\}} = c_H$ for all $k \in H$ and $c_{H \cup \{i\}} = c_H + 1$ for all $k \in K \setminus H$. Brockmüller et al. (1996) show that for any $H \subseteq K$ the c-strong inequality

$$\sum_{k \in H} y^k \leq c_H + x \tag{2.7}$$

is valid for $\mathcal{F}_U$ when $w^0 = 0$. A c-strong inequality is facet-defining for $\text{conv}(\mathcal{F}_U)$ if and only if H is maximal c-strong.

Theorem 2.21 The maximal c-strong inequalities (2.7) constitute all facet-defining inequalities $\sum_{k \in K} \pi^k y^k \leq \pi^0 + x$ of $\text{conv}(\mathcal{F}_U)$ with integral $\pi^k,\ k \in [0, |K|]$.

Proof See Appendix B. □

There might be other facet-defining inequalities of $\mathcal{F}_U$. In fact, we present a facet-defining inequality $\sum_{k \in K} \pi^k y^k \leq \pi^0 + x$ of $\text{conv}(\mathcal{F}_U)$ with fractional coefficients in Sec-

tion 2.3.4.3.4. We now discuss the separation problem for the c-strong inequalities. Given a point $(\bar{y}, \bar{x})$, there exists a c-strong inequality violated by $(\bar{y}, \bar{x})$ if and only if there exists $H \subseteq K$ such that $\sum_{k \in H} \bar{y}^k - c_H > \bar{x}$. Let λ be the least common multiple of the denominators of the rational numbers $(1 - d^k)$ and w^0. Then, a c-strong inequality is violated if and only if

$$\begin{aligned}
&\max_{H \subseteq K} \{ \sum_{k \in H} \bar{y}^k - \lfloor w^0 + \sum_{k \in H} (1 - d^k) \rfloor \} \\
&\qquad = \max \{ \sum_{k \in K} \bar{y}^k y^k - x \; : \; \sum_{k \in K} (1 - d^k) y^k + w^0 + 1/\lambda \leq x, (y, x) \in \mathcal{D}_U \} + 1 \\
&\qquad > \bar{x}.
\end{aligned}$$

From Theorem 2.14, this maximization problem with the constant term $-w^0 - 1/\lambda$ is $\mathcal{NP}$-hard. Although the separation problem of c-strong inequalities is $\mathcal{NP}$-hard, from Proposition 2.12, it has an optimal solution (y^*, x^*) such that $y^*_k = 1$ if $\bar{y}^k = 1$, and $y^*_k = 0$ if $\bar{y}^k = 0$.

Therefore, we can fix such variables to their optimal values and solve the separation problem over $k \in K$ such that $0 < \bar{y}^k < 1$, which in practice can be done very efficiently even by enumeration, as most variables take on values either 0 or 1 in the LP relaxations of network design problems. Proposition 2.4 provides insight as to why this is true.

2.3.4.3.2 j-split c-strong inequalities

In Section 2.3.4.3.2, we describe new valid inequalities for $\mathcal{F}_U$, motivated by Proposition 2.12. An inequality $\sum_{k \in K} \pi^k y^k \leq \pi^0 + x$ is valid for $\mathcal{F}_U$ if and only if $\max\{\sum_{k \in K} \pi^k y^k - x \; : \; (y, x) \in \mathcal{F}_U\} \leq \pi^0$. As shown in Section 2.3.4.1, solving this maximization problem is $\mathcal{NP}$-hard. However, if the maximum of $\sum_{k \in K} \pi^k y^k - x$ over a suitable relaxation of $\mathcal{F}_U$ is no more than π^0, then we can deduce that $\sum_{k \in K} \pi^k y^k \leq \pi^0 + x$ is valid for this relaxation and hence for $\mathcal{F}_U$.

The relaxation of $\mathcal{F}_U$ that we consider for this purpose is obtained by splitting the integer capacity variable. In a j-split relaxation, the capacity variable x is allowed to take values that are integer multiples of $1/j$, where j is a positive integer. Let $\mathcal{F}^j_U =$

$\{\sum_{k\in K} d^k y^k \leq w^0 + z/j,\ (y,z) \in \mathcal{D}_U\}$. We define an infinite set of relaxations for $\mathcal{F}_U$ with $\text{conv}(\mathcal{F}_U^1) = \text{conv}(\mathcal{F}_U)$ and $\lim_{j\to\infty} \text{conv}(\mathcal{F}_U^j) = \mathcal{F}_L$. The last equation follows from the fact that $y \in \{0,1\}^{|K|}$ for all extreme points (y,x) of $\mathcal{F}_L$.

For a given instance of $\mathcal{F}_U$, there exists a finite j for which $jc^k \leq \lfloor jd^k \rfloor$ if $c^k \leq d^k$ and $jc^k \geq \lceil jd^k \rceil$ if $c^k \geq d^k$ for all $k \in K$. Thus, From Proposition 2.12, there exists a j-split relaxation of $\mathcal{F}_U$, for which the optimization problem is trivial to solve, and hence the validity of a given inequality $\sum_{k\in K} \pi^k y^k \leq \pi^0 + x$ can be checked easily for this relaxation. Alternatively, for each j, we can define an inequality that can be easily verified to be valid for the corresponding j-split relaxation.

Proposition 2.22 Let $H = \{k \in K : \pi^k \geq \lceil jd^k \rceil\}$. Any inequality $\sum_{k\in K} \pi^k y^k \leq \pi^0 + jx$ with $j \in \mathbb{Z}_{++}$ and $\pi^k \leq \lfloor jd^k \rfloor$ or $\pi^k \geq \lceil jd^k \rceil$ for all $k \in K$ is valid for $\mathcal{F}_U$ for $\pi^0 = \pi(H) - \lceil jd(H) - jw^0 \rceil$.

Proof See Appendix B. □

Thus, for any positive integer j and any $H \subseteq K$, we can define strong valid inequalities by letting π^k equal either $\lfloor jd^k \rfloor$ or $\lceil jd^k \rceil$, for all $k \in K$. Let $c_H^j = \sum_{k\in H} \lceil jd^k \rceil - \lceil jd(H) - jw^0 \rceil$ and define the j-split c-strong inequality as

$$\sum_{k\in H} \lceil jd^k \rceil y^k + \sum_{k\in K\setminus H} \lfloor jd^k \rfloor y^k \leq c_H^j + jx. \tag{2.8}$$

A j-split c-strong inequality (2.8) is a c-strong inequality for $\mathcal{F}_U^j$. Since $\mathcal{F}_U^j$ is a relaxation of $\mathcal{F}_U$, a necessary condition for inequality (2.8) to be facet-defining for $\mathcal{F}_U$ is that H is maximal c-strong in the j-split relaxation $\mathcal{F}_U^j$. In Proposition 2.23, we present a sufficient condition for a j-split c-strong inequality to be facet-defining for $\mathcal{F}_U$. As the example in Section 2.3.4.3.4 illustrates, j-split c-strong inequalities may be facet-defining more generally.

Proposition 2.23 Let $f_H = r(d(H) - w^0) = d(H) - w^0 - \lfloor d(H) - w^0 \rfloor$. Inequality (2.8) is facet-defining for $\text{conv}(\mathcal{F}_U)$ if either H is maximal c-strong in the j-split relaxation; $f_H > (j-1)/j$ and $w^0 \geq 0$; or $d^k > f_H$ for all $k \in H$ and $d^k < 1 - f_H$ for all $k \in K \setminus H$.

Proof See Appendix B. □

2.3.4.3.3 Lifted knapsack cover inequalities

Let K_0 and K_1 be two disjoint subsets of K and ν be a non-negative integer. Consider the binary knapsack set $\mathcal{F}_U(\nu, K_0, K_1)$ obtained by projecting the capacity variable x to ν, all binary variables indexed with K_0 to 0 and all binary variables indexed with K_1 to 1, i.e., $\mathcal{F}_U(\nu, K_0, K_1) = \{(y, x) \in \mathcal{F}_U \, : \, x = \nu, \; y^k = 0 \text{ for all } k \in K_0 \text{ and } y^k = 1 \text{ for all } k \in K_1\}$.

Definition 2.24 Let $C = K \setminus (K_0 \cup K_1)$. Then, the set C is called a *cover* if $\lambda = d(C) + d(K_1) - w^0 - \nu > 0$. C is said to be a *minimal cover* if $d^k \geq \lambda$ for all $k \in C$. □

For any cover C, the knapsack cover inequality $\sum_{k \in C} y^k \leq |C| - 1$ is facet-defining for $\text{conv}(\mathcal{F}_U(\nu, K_0, K_1))$ if and only if C is a minimal cover (Nemhauser and Wolsey 1988). By lifting the knapsack cover inequalities of minimal covers with the projected variables, one can obtain facet-defining inequalities of $\text{conv}(\mathcal{F}_U)$, see Section 1.5.2 for an introduction to lifting.

One practical way of lifting inequalities is sequential lifting, in which projected variables are introduced to an inequality one at a time in some sequence. van Hoesel et al. (2002) have independently lifted knapsack cover inequalities to also describe strong valid inequalities for $\mathcal{F}_U$. Here we show that given a minimal cover, a lifted knapsack cover inequality can be constructed in $\mathcal{O}(|K|^3)$ if the capacity variable x is lifted first. We further show that inequalities obtained in this manner subsume all c-strong inequalities.

Now, we describe the lifting procedure. We introduce the capacity variable x to the cover inequality first. Let $\mathcal{F}_U(K_0, K_1) = \{(y, x) \in \mathcal{F}_U \, : \, y^k = 0 \text{ for all } k \in K_0 \text{ and } y^k = 1 \text{ for all } k \in K_1\}$ and C be a cover. Inequality $\sum_{k \in C} y^k + \alpha(\nu - x) \leq |C| - 1$ is valid for $\mathcal{F}_U(K_0, K_1)$ if and only if

$$\alpha \leq \overline{\alpha} = \min \left\{ \frac{|C| - 1 - \sum_{k \in C} y^k}{\nu - x} \, : \, x < \nu, \; (y, x) \in \mathcal{F}_U(K_0, K_1) \right\},$$

and

$$\alpha \geq \underline{\alpha} = \max \left\{ \frac{\sum_{k \in C} y^k - |C| + 1}{x - \nu} \, : \, x > \nu, \; (y, x) \in \mathcal{F}_U(K_0, K_1) \right\}.$$

If C is a minimal cover and α equals either $\underline{\alpha}$ or $\overline{\alpha}$, then $\sum_{k \in C} y^k + \alpha(\nu - x) \leq |C| - 1$ is facet-defining for $\text{conv}(\mathcal{F}_U(K_0, K_1))$, which follows from Wolsey (1976). The existence of

a valid lifting coefficient α follows from the next proposition.

Proposition 2.25 (Atamtürk and Rajan (2002), van Hoesel et al. (2002)) For any cover C, $\underline{\alpha} \leq 1 \leq \overline{\alpha}$ holds.

Proof See Appendix B. □

For any cover, $\underline{\alpha} = 1/(\lceil d(C) + d(K_1) - w^0 \rceil - \nu) = 1/\lceil \lambda \rceil$, which is computed in linear time For a minimal cover C, since $0 < \lambda \leq d^k < 1$, we have $\underline{\alpha} = 1$. The upper bound $\overline{\alpha}$ can be computed efficiently as well: Suppose $d^1 \leq d^2 \leq \cdots \leq d^{|C|}$. Let $D^0 = d(K_1) - w^0$ and $D^k = D^{k-1} + d^k$ for $k \in [1, |C|]$. Since the coefficients in the cover inequality are the same, we have

$$\overline{\alpha} = \min_{k \in [0, |C|-2]} \left\{ \frac{|C| - 1 - k}{\nu - \lceil D^k \rceil} \ : \ \lceil D^k \rceil < \nu \right\}.$$

Therefore $\overline{\alpha}$ is computed by selecting the minimum of at most $|C| - 1$ terms after sorting $d^k,\ k \in C$ in non-decreasing order, which can be done in $\mathcal{O}(|K| \log |K|)$.

Next, we introduce the projected binary variables to the inequality $\sum_{k \in C} y^k + \alpha(\nu - x) \leq |C| - 1$ one at a time in some arbitrary sequence. As shown in the example in Section 2.3.4.3.4, different sequences may lead to different lifted inequalities. Let $L_0 \subseteq K_0$ and $L_1 \subseteq K_1$ be the index sets of variables that have already been lifting and the current lifted inequality be

$$\sum_{k \in C} y^k + \sum_{k \in L_0} \alpha^k y^k + \sum_{k \in L_1} \alpha^k (1 - y^k) + \alpha(\nu - x) \leq |C| - 1. \tag{2.9}$$

Then the lifting coefficient of a variable y^i, $i \in (K_0 \setminus L_0) \cup (K_1 \setminus L_1)$ is computed by solving the lifting problem

$$\begin{aligned} \alpha^i = |C| - 1 - \max & \sum_{k \in C} y^k + \sum_{k \in L_0} \alpha^k y^k + \sum_{k \in L_1} \alpha^k (1 - y^k) + \alpha(\nu - x) \\ (BLP) \qquad s.t. : & \sum_{k \in C \cup L_0 \cup L_1} d^k y^k \leq w^0 - d(K_1 \setminus L_1) \mp d^i + x \qquad (2.10) \\ & y^k \in \{0, 1\} \qquad k \in C \cup L_0 \cup L_1 \\ & x \in \mathbb{Z}. \end{aligned}$$

In the right hand side of the constraint (2.10), we have $-d^i$ if $i \in K_0$ and $+d^i$ if $i \in K_1$.

Proposition 2.26 The maximal c-strong inequalities are equivalent to the lifted minimal cover inequalities with $\alpha = \underline{\alpha}$.

Proof See Appendix B. □

Remark 2.27 The proof of Proposition 2.26 also shows that projecting binary variables to 1 does not lead to new inequalities when $\underline{\alpha}$ is used as the lifting coefficient for the capacity variable. This is because by Proposition 2.26 and Theorem 2.21, all facet-defining inequalities $\sum_{k \in K} \pi^k y^k \leq \pi^0 + x$ of $\text{conv}(\mathcal{F}_U)$ with integer coefficients can be obtained by lifting minimal cover inequalities using $\alpha = \underline{\alpha}$ and $K_1 = \emptyset$. On the other hand, letting $K_1 \neq \emptyset$ does not lead to fractional lifting coefficients. So when $\alpha = \underline{\alpha}$, the lifted minimal cover inequalities take the simple form $\sum_{k \in C} y^k \leq |C| - 1 - \nu + x = c_C + x$. □

Lemma 2.28 If C is a minimal cover and $\alpha = \overline{\alpha}$, then the lifting coefficients of inequality (2.9) satisfy $\alpha^k \leq |C| - 1$ for all $k \in K_0$ and $-\alpha^k \leq |C| - 1$ for all $k \in K_1$.

Proof See Appendix B. □

Theorem 2.29 For a minimal cover, a lifted knapsack cover inequality with $\alpha = \overline{\alpha}$ can be constructed in $\mathcal{O}(|K|^3)$.

Proof See Appendix B. □

Remark 2.30 For the binary knapsack set, Zemel (1989) presents an $\mathcal{O}(|K|^2)$ algorithm to compute the lifting coefficients of the minimal cover inequality when $K_1 = \emptyset$. However, no polynomial-time algorithm is known for constructing a lifted cover inequality for the binary knapsack set if some of the variables are projected to 1, i.e., $K_1 \neq \emptyset$. For the binary knapsack set, Gu et al. (1994) present an example where the lifting coefficients are bounded from below by an exponential function of n. In the case of the unsplittable flow arc set $\mathcal{F}_U$, we are able to bound the coefficients of the lifted knapsack cover inequality from above by $|K|$ in Lemma 2.28 by lifting the integer capacity variable y first. □

2.3.4.3.4 Illustrative example

Let $\mathcal{F}_U = \{(y, x) \in \{0,1\}^5 \times \mathbb{Z} : \frac{1}{3}y_1 + \frac{1}{3}y_2 + \frac{1}{3}y_3 + \frac{1}{2}y_4 + \frac{2}{3}y_5 \leq x\}$. In Table 2.1, we list the lifted knapsack cover inequalities of $\mathcal{F}_U$ that are not c-strong inequalities. They are

all the facet-defining inequalities that can be obtained by the lifting procedure described in Section 2.3.4.3.3 for minimal cover inequalities.

Table 2.1: Lifted knapsack cover inequalities: A small example

ν	(C, K_0, K_1)	inequalities
1	$(\{2,3,4\},\{1,5\},\emptyset)$	$y_2 + y_3 + y_4 + y_5 \leq 2x$
1	$(\{1,4,5\},\{2,3\},\emptyset)$	$y_1 + y_2 + y_4 + y_5 \leq 2x$ and $y_1 + y_3 + y_4 + y_5 \leq 2x$
2	$(\{1,2,3,4\},\emptyset,\{5\})$	$y_1 + y_2 + y_3 + y_4 + 2y_5 \leq 2x + 1$
2	$(\{1,2,3,5\},\emptyset,\{4\})$	$y_1 + y_2 + y_3 + 2y_4 + y_5 \leq 2x + 1$

The last two inequalities can only be obtained by lifting cover inequalities (including non-minimal covers) where $K_1 \neq \emptyset$. Thus, this example illustrates that with $\alpha = \overline{\alpha}$, projecting binary variables to 1 does lead to inequalities that can not be obtained otherwise by the lifting of minimal cover inequalities.

Except the last inequality, all the inequalities above are also 2-split c-strong inequalities. The 3-split c-strong inequality $y_1 + y_2 + y_3 + 2y_4 + 2y_5 \leq 3x$ with $H = \{1,2,3,4,5\}$, which is facet-defining for $\mathrm{conv}(\mathcal{F}_U)$ can not be obtained by lifting any cover, not necessary minimal. This example shows that the j-split c-strong inequalities and the lifted knapsack cover inequalities are indeed different classes of inequalities.

2.4 Computational results

To empirically test the effectiveness of the results in the preceding sections on network design arc sets, we develop a branch-and-cut algorithm for solving the NDP; implemented using CPLEX[2] callable library (version 6.5.1). The branch-and-cut algorithm generates cutting planes from the splittable and unsplittable arc sets. We also add certain families of cut-set inequalities to the formulation, as discussed shortly. We do all computations on a Sun Ultra 5 workstation with a one hour time limit, and using a best-bound node selection strategy in the branch-and-bound search tree.

We base our data set on the unsplittable multi-commodity flow problem instances used in Barnhart et al. (2000). In these instances, capacity is fixed and demand for com-

[2]CPLEX is a trademark of ILOG, Inc.

modities ranges between 5 and 60. We create four sets of capacitated network design problems by introducing capacity variables with unit capacities 4, 25, 60, and 120 and unit installation costs 50, 250, 450, and 720, respectively. We report on our computations with 5 problems that CPLEX has the most difficulty in solving. These problem instances are available at http://ieor.berkeley.edu/∼atamturk/data; we summarize their basic properties in Table 2.2.

Table 2.2: Problem instance sizes

problem	commodities	nodes	arcs	variables		constraints
				flow	capacity	
1	70	29	132	8540	61	2181
2	58	18	58	3364	29	1120
3	93	27	74	7178	37	2612
4	87	24	84	7308	42	2196
5	81	27	72	5832	36	2284

Cut-set inequalities (see Section 2.2) are known to improve the LP relaxations of network design problems; however, their separation problem is $\mathcal{NP}$-hard. Therefore, before solving the problems, we add the cut-set inequalities defined for one and two-node subsets of the network to the formulations, and use these formulations as the basis for our comparisons for the arc-set polyhedra. The We can significantly improve the performance of the cut-set inequalities by a more effective heuristic scheme, see Günlük (1999) for some ideas. However, we are more concerned with the performance of the arc-set inequalities developed in Section 2.3.

2.4.1 Experiments with splittable flow problems

The first set of experiments tests the effectiveness of the exact separation algorithm for the residual capacity inequalities described in Section 2.3.3.2. Since these inequalities are developed for splittable flow network design problems, we relax the binary flow variables of our data set to continuous variables for these experiments.

Table 2.3 summarizes the computations with the branch-and-cut algorithm using residual capacity inequalities. In this table, for each problem (problem) and capacity (ca-

pacity) combination, we report the number of residual capacity inequalities added (cuts), percentage improvement in the integrality gap obtained by the cuts at the root node of the search tree (root improvement), and the number of nodes evaluated in the search tree (b&b nodes). Let $zprep$ be the objective value of the LP relaxation after pre-processing, $zroot$ the value of the LP relaxation at the root node before branching, and zub the value of the best feasible solution known for an instance. Then, the root improvement is calculated as $100 \times \frac{zroot-zprep}{zub-zprep}$. We also report the elapsed CPU time in seconds (time) or percentage gap between the best upper bound and the best lower bound at termination if the time limit is reached (endgap). Since we report either time or the integrality gap at termination for each instance, we present them in the same column. To distinguish between them, we report the integrality gap at termination in parenthesis, and follow this convention throughout this dissertation.

Table 2.3: Computational results with splittable flow arc sets

capacity	problem	cuts	root improvement		b&b nodes		time (endgap)	
			(1)	(2)	(1)	(2)	(1)	(2)
4	1	20	35.7	44.3	27630	22310	(0.1)	(0.1)
	2	17	25.8	43.2	6040	2918	177.4	85.9
	3	12	48.9	62.3	15736	17720	437.5	435.3
	4	24	81.8	81.8	29683	23282	(0.0)	(0.0)
	5	17	47.3	80.7	1052	1172	38.5	41.4
25	1	80	37.7	56.2	6858	2929	(0.8)	(0.8)
	2	68	44.9	61.3	5840	2967	266.7	277.4
	3	31	63.1	75.2	7177	5476	206.4	157.5
	4	57	56.9	59.3	9554	7441	(0.7)	(0.2)
	5	29	32.3	68.6	9037	8384	285.4	260.4
60	1	262	17.0	55.1	5403	967	(3.9)	(2.7)
	2	206	19.2	58.5	1788	483	133.6	161.5
	3	114	59.8	72.4	7753	5893	245.6	224.5
	4	257	22.4	55.3	8250	1832	(1.6)	(1.4)
	5	77	48.7	79.5	2166	1047	78.8	49.5
120	1	625	23.2	52.1	3650	363	(17.9)	(13.8)
	2	500	26.4	58.1	39465	9662	2118	3411
	3	340	66.3	83.7	797	626	39.2	38.4
	4	629	12.9	47.6	6665	624	(10.8)	(7.4)
	5	209	55.4	75.9	1056	1616	52.4	78.4

(1) base formulation, (2) residual capacity inequalities.

To reflect the improvement obtained by the cut-set inequalities as well, we use the $zprep$ values of formulations before adding the cut-set inequalities. In all tables, columns with heading (1) show the performance of the branch-and-bound algorithm for the base formulation that includes the cut-set inequalities.

In any branch-and-cut algorithm, the frequency of applying a separation routine in the search tree has an important effect on the computations. In this experiment, we run the separation routine for the residual capacity inequalities only at the root node. We base this choice on our observation that most of the effective cuts are found at the root node of the search tree. We see that a large number of cuts are added compared to the number of arcs in the problems and the number of cuts added increases with the capacity.

On average, the cut-set inequalities reduce the integrality gap at the root node by 41%. On the other hand, the addition of residual capacity inequalities reduces the integrality gap at the root node significantly (64%), and for almost all problems decreases the total number of nodes evaluated. For the problems that are not solved within the time limit, the integrality gap at termination is smaller for all of the problems when we add the residual capacity inequalities. Thus, we see that the residual capacity inequalities are effective in solving splittable network design problems.

Since the residual capacity inequalities describe the convex hull of the splittable arc sets $\mathcal{F}_S$ and we use an exact algorithm to separate them, the integrality gap improvement shown under heading (2) of Table 2.3 is the best that can be achieved by using cutting planes from individual arc sets for these instances.

2.4.2 Experiments with unsplittable flow problems

The second set of experiments is on the unsplittable flow network design problem. First, we test the impact of the inequalities described in Section 2.3.4 in reducing the integrality gap at the root node of the search tree. Under headings (2), (3), and (4) of Table 2.4, we report the number of cuts added (cuts) and the integrality gap improvement at the root node (impr) for c-strong inequalities, j-split c-strong inequalities, and lifted knapsack cover inequalities, respectively. As in Table 2.3, we use $zprep$ values of formulations

before adding the cut-set inequalities to calculate the improvement at the root node so that the improvement from these cuts can be compared with the improvement obtained by cut-set inequalities, shown under heading (1) of Table 2.4.

Table 2.4: Root integrality gap improvement with unsplittable flow sets

capacity	problem	(1)	(2)		(3)		(4)	
		impr	cuts	impr	cuts	impr	cuts	impr
4	1	4.3	161	53.8	358	55.2	140	54.1
	2	6.4	74	69.2	153	72.6	63	70.7
	3	37.7	70	79.6	137	83.6	70	56.6
	4	9.8	157	42.4	324	44.5	143	43.0
	5	38.4	52	82.9	77	88.4	44	84.5
25	1	0.00	270	32.2	633	34.6	225	32.7
	2	22.8	119	57.4	385	64.6	106	62.1
	3	54.2	84	62.0	218	63.6	79	62.0
	4	0.00	204	12.8	531	14.4	165	13.2
	5	17.2	126	54.1	218	56.3	115	54.1
60	1	4.7	317	25.7	854	28.4	297	26.9
	2	14.0	185	44.5	462	48.0	153	47.7
	3	53.7	153	61.2	355	61.6	127	61.1
	4	0.6	267	17.5	683	19.7	203	19.0
	5	38.0	201	68.5	431	69.3	196	68.6
120	1	9.4	486	30.7	1291	31.2	438	30.7
	2	28.0	379	52.8	809	51.6	301	52.7
	3	62.0	420	71.1	732	71.6	316	71.4
	4	2.4	531	39.3	1094	39.7	437	38.9
	5	35.5	539	63.0	746	63.3	498	63.0

(1) base formulation, (2) c-strong inequalities,
(3) j-split c-strong inequalities, (4) lifted knapsack cover inequalities.

To find violated c-strong inequalities, given a fractional point $(\bar{y}, \bar{x})$, we use the fact that there exists an optimal solution $H^\star$ to the separation problem such that $k \in H^\star$ if $\bar{y}^k = 1$ and $k \in K \setminus H^\star$ if $\bar{y}^k = 0$. Therefore, after fixing the variables with integral LP values, for each potential value of c_H, we choose the elements of H in non-decreasing order of $\bar{y}^k d^k$ in a greedy fashion for all other variables. In the separation problem, usually more than 90% of the variables are fixed by the optimality criteria of Proposition 2.12. We use a greedy heuristic to separate the knapsack cover inequalities (Gu et al. 1998) after letting $\nu = \lceil \bar{x} \rceil$.

When lifting the knapsack cover inequalities, we let the lifting coefficient of the capac-

ity variable $\alpha = \overline{\alpha}$, since c-strong inequalities correspond to lifted cover inequalities with $\alpha = \underline{\alpha}$. To find the lifting coefficients for the projected binary variables, rather than solving the lifting problems exactly, we solve their splittable relaxation as described in Section 2.3.3.1. To generate j-split c-strong cuts, we use the separation routine for c-strong inequalities, after multiplying the coefficients of the arc-set inequality by k. A preliminary test showed that the quality of j-split c-strong cuts degrade for high values of j, especially for $j > 4$. Therefore, in these computations, we set the maximum value of j to 4 for all our reported computations.

Comparing headings (3) and (4) with (2), although a good number of lifted knapsack cover and j-split c-strong cuts are generated, further improvement of the integrality gap is limited. The root improvement is slightly better with j-split c-strong inequalities for most of the instances.

We generate a lifted knapsack cover inequality in two steps: we find a violated knapsack cover inequality and then lift the projected variables. Even though it is a heuristic, the one step separation routine for the j-split c-strong inequalities may find more cuts than the knapsack cover separation that does not take into account the lifting coefficients.

Next, we compare the overall performance of the branch-and-cut algorithm for c-strong cuts; for j-split c-strong cuts with $1 \leq j \leq 4$; and for all cuts, including the lifted knapsack covers; see Table 2.5. For each capacity (cap) and problem (problem) combination, we report the number of nodes evaluated (b&b nodes), and elapsed CPU time in seconds (time) or percentage gap between the best known upper bound and the best lower bound at termination if the time limit is reached (endgap).

In the separation routine for j-split c-strong inequalities, for each arc we increase the value of j only if no cut is found with the current value. Separation routines are run in the first 50 nodes, which correspond to nodes that are high in the search tree, because we use a best-bound node selection strategy.

Comparing columns with headings (1) and (2) we see that generating c-strong cuts reduces the number of nodes and the overall CPU time significantly; we solve twelve instances to optimality, as opposed to seven. Generating j-split c-strong inequalities

Table 2.5: Computational results with unsplittable flow arc sets

cap	problem	b&b nodes				time (endgap)			
		(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)
4	1	18536	4269	2521	2282	(2.4)	(1.0)	(1.0)	(0.9)
	2	61101	757	458	494	(1.4)	133.8	93.4	117.1
	3	78659	40359	18723	16898	(0.3)	1701	819.2	728.0
	4	17494	5273	7572	5585	(1.6)	(1.0)	(1.2)	(1.1)
	5	99686	3345	936	1318	(0.1)	160.0	56.8	72.1
25	1	11283	2084	1351	1268	(12.7)	(9.5)	(9.3)	(9.3)
	2	43535	6791	2362	5396	(8.6)	2838	1767	(0.0)
	3	71023	2399	4877	1896	2687	133.4	268.9	125.1
	4	14798	2469	1191	1392	(8.8)	(7.9)	(7.8)	(7.9)
	5	95088	32464	28679	21012	(0.1)	1832	(0.1)	1878
60	1	9949	4154	2561	1817	(26.8)	(25.5)	(25.0)	(24.8)
	2	12008	5050	7902	6246	842.9	1397	1835	1890
	3	9400	9062	6632	6050	330.1	403.4	293.6	288.6
	4	12050	6043	2952	2653	(11.2)	(11.3)	(11.1)	(10.7)
	5	7210	2533	4208	1935	256.8	134.3	202.4	106.4
120	1	4869	968	932	720	(27.2)	(24.6)	(24.7)	(24.9)
	2	19082	12431	13208	16321	1147	2878	2371	(0.1)
	3	835	680	606	570	42.8	60.0	53.0	67.3
	4	6261	1128	1817	1332	(14.7)	(10.0)	(8.8)	(9.5)
	5	22827	2511	2888	2820	933.2	149.0	189.9	151.9

(1) base formulation, (2) c-strong inequalities,
(3) j-split c-strong inequalities, (4) all inequalities.

and lifted knapsack cover inequalities in addition to c-strong inequalities has a positive effect; no further instances are solved to optimality, but the CPU time and integrality gap at termination usually reduce. Thus, the additional improvement is not as significant as adding c-strong inequalities to the base formulation.

2.5 Conclusions

Based on these experimental results, we see that the residual capacity inequalities and the c-strong inequalities reduce the integrality gap at the root node, and solve the NDP more effectively. The new classes of inequalities improve the performance of the algorithm further, but not significantly.

Thus, we conclude that inequalities from the arc and cut sets of the NDP strengthen

the LP relaxations and improving the performance of LP based search algorithms. In particular, the residual capacity inequalities and the cut-set inequalities reduce the integrality gap at the root node by 64% for the splittable flow problems. For unsplittable flow, cut-set inequalities and the c-strong inequalities reduce the integrality gap at the root node by 51%, on average. However, inequalities that capture additional structures of the network design problems seem to be necessary for solving them more effectively. One such class of inequalities are the metric inequalities which generalize the cut-set inequalities. In Chapter 6, we develop these inequalities in the context of survivable network design problems. Alternatively, one can use problem-independent inequalities derived from the mixed-integer knapsack set, see Chapter 7.

For the splittable flow problems, the exact separation algorithm for residual capacity inequalities empirically provides the maximum possible integrality gap improvements at the root node based on inequalities from arc sets. This is because the residual capacity inequalities completely describe the convex hull of the splittable flow arc set. To know the value of the maximum possible improvement that can be obtained by using inequalities from arc sets for the unsplittable flow problems, we can solve LP relaxations of Dantzig-Wolfe reformulations of the unsplittable flow network design problem by relaxing the demand constraints. This LP relaxation can be solved by generating columns over the pricing sub-problems consisting of individual arc sets with the dynamic programming algorithm given in Section 2.3.4.1.

In subsequent chapters, we discuss various techniques for ensuring survivability, and present new frameworks for survivability that uses directed cycles. We study the arc-set and cut-set polyhedra of the mixed-integer formulations for these methodologies, and develop strong valid inequalities for such polyhedra.

Chapter 3

Models for designing survivable networks

3.1 Introduction

Network design problems become much more complicated when the networks have to be designed to survive failures. A network is said to be survivable if flow of commodities disrupted by the failure (removal) of some of the elements of the graph can be rerouted. Existence of two edge-disjoint paths between every pair of source and destination nodes (two-edge connectedness) is a necessary condition for survivability of the network, but is not sufficient.

To ensure that the flow on the network can be rerouted in case of a failure, sufficient spare capacity must be available on the working edges of the network. Several heuristic and exact approaches have been developed for designing survivable networks; Soriano et al. (1998) presents an excellent overview of survivable network design problems and a synthesis of related literature.

Since over-provisioning of capacity is undesirable due to the high capacity installation cost, designing capacity-efficient survivable networks is a critical problem in the telecommunications industry. The most capacity-efficient networks can be designed by formulating the problem as capacitated network design problems (NDPs) for each failure

scenario, but linked by common integral capacity variables. In fact, this is global rerouting (GNP), discussed in Section 3.3.1. Although global rerouting is not implemented in practice, we use it as a benchmark to measure the capacity efficiency of other frameworks.

One reason why GNP is not implemented is that solutions with minimal changes to no-failure flow are preferred because it is undesirable in practice to manipulate unaffected flow while restoring affected flows. A number of practical models and strategies have been developed for designing survivable networks that admit local rerouting of flow on a failed edge (Altinkemer 1994, Xiong and Mason 1999, Balakrishnan et al. 2002, Goldschmidt et al. 2003). Any framework, whether dedicated protection, shared protection or a hybrid, can include methodologies that implement local rerouting.

Failures affecting a large number of network elements at the same time are generally considered improbable; we define a restricted set of failures for which the network will need to be survivable. We include the no-failure state in the set of failure states, and define a failure state as the set of edges that fail simultaneously. An edge failure is the event of decreasing the capacity of the edge to 0. Using this definition of failure states, a node failure is a failure state that includes all the edges incident to the node.

We focus on single-edge failures. A node failure can be modeled a single-edge failure by splitting the node into two pseudo-nodes connected by a pseudo-edge. Denoting the no-failure state by $\{0\}$, the set of failures $S = E \cup \{0\}$ for single-edge failures. If the set of failure states has one element (no-failure state), then the survivable network design problem reduces to the NDP.

Definition 3.1 *Survivable network design problem (SNP)*: Given a directed network, flow costs, capacity installation costs for each edge, a set of commodities (given in terms of their origin-destination pairs and demands), and a set of failure states, we wish to route the commodities so that the net flow on any arc for any failure state is at most the capacity installed on that edge and all demands are met in all failure states, at minimum total flow routing (under no failure) and capacity installation costs. □

For the rest of this dissertation, we shall present the splittable flow versions of all formulations. Unless mentioned otherwise, these formulations can be easily modified to

the unsplittable case by enforcing the flow variables y to be binary.

We distinguish SNP from the uncapacitated survivable network design problem. In the uncapacitated problem, the demand quantities associated with the commodities are integral. A often studied special case of this uncapacitated problem is defined as follows.

Definition 3.2 *Network design problem with connectivity requirements (NDC)*: Given an undirected graph, flow costs on the edges, a set of commodities (given in terms of their origin-destination pairs and number of edge-disjoint paths required), we wish to route the commodities such that the connectivity requirements are met, at minimum flow cost. □

For comprehensive reviews of research on the NDC and its specializations, see Frank (1994), Grötschel et al. (1995), and Raghavan and Magnanti (1997).

Global rerouting shared protection, which provides the most capacity-efficient survivable networks, is extremely difficult to solve, often more so than NDP by an order of magnitude for the same network instance. In recent years, SNP has received considerable attention from both electrical engineering/computer science (EECS) and operations research (OR) communities. Most of the work has focused on the development of powerful heuristic techniques; see Herzberg et al. (1995), Balakrishnan et al. (1998), and Luss et al. (1998) for some examples. Others have developed cutting-plane algorithms (Alevras et al. 1998, Dahl and Stoer 1998, Bienstock and Muratore 2000).

In practice, to make a telecommunications network survivable, one uses either of the following two different strategies: protection or restoration. Protection techniques completely identify ahead of time the routes that disrupted flows will take and the capacities that will be used. Restoration techniques determine which available capacity will be used for a specific failure (and the routes that will be used for each affected demand) at the time of failure. We do not discuss any restoration schemes, and focus on the distinction between dedicated protection and shared protection schemes.

Dedicated protection techniques install and assign spare capacity specifically dedicated to a particular commodity (Altinkemer 1994, Chung et al. 1996, Goldschmidt et al. 2003). In a shared protection scheme, instead of pre-assigning spare capacity to protect each commodity of the network independently, spare capacity is shared by more than

one commodity, and used as required to restore the disrupted flow.

The simplest and most widely used dedicated protection technique is called 1+1 diverse protection (DP) switching (Chung et al. 1996). It uses one dedicated backup route for each commodity, the backup being physically different (on a node-disjoint path) than the no-failure route. This provides a very fast reconfiguration mechanism (under failure, all affected commodities are simply rerouted on their designated backup routes), but is very expensive in terms of capacity, more than doubling the capacity of the non-survivable network (the backup route is generally longer than the no-failure one).

With the advent of optical networks and sophisticated routing equipment (add/drop multiplexers), a new dedicated protection technique known as Self-Healing Rings (SHR) was introduced (Slevinsky et al. 1993, Altinkemer 1994, Cosares et al. 1995, Luss et al. 1998, Soriano et al. 1998, Sutter et al. 1998, Armony et al. 2000, Hochbaum and Olinick 2001, Goldschmidt et al. 2003). In networks using SHRs, nodes of a network are covered by rings (closed loops or undirected cycles) consisting of edges of the same capacity. Within such a ring architecture, there exist two arc-disjoint paths between any pair of nodes in the ring, by construction. SHR networks are thus inherently survivable, since any flow of a commodity through a ring is protected against any edge failure by sending the flow back in the reverse direction along the ring.

SHRs maintain very fast reconfiguration times in the event of failure. At the same time, they achieve lower spare capacity requirements than 1+1 DP, since spare capacity on a ring is shared by all demand flows using that ring. For this reason, ensuring survivability using SHRs is not dedicated protection in the truest sense. However, since the spare capacity on a ring is dedicated to protect failures on that ring, SHRs have been traditionally considered a dedicated protection technique.

There are several types of SHRs, depending on the protocol used for their implementation. We can broadly classify them as unidirectional and bidirectional, depending on the way the flow is routed under no failure.

In a unidirectional ring, no-failure flow is only sent in one direction; the other direction is reserved for failure flows. The ring capacity required for a unidirectional SHR is thus

determined by the largest possible failure flow, which will be the maximum flow carried by any arc on the ring. In a bidirectional ring, both directions may be used for no-failure flow; on failure, this flow is sent in the other direction. Thus, the maximum flow on any arc in a particular direction is the sum of the no-failure flow on that arc and the maximum flow in the other direction. Thus, for bidirectional SHRs, the ring capacity required is equal to the sum of the maximum flows carried in each direction.

As a result, bidirectional SHRs do not provide any capacity savings over unidirectional SHRs, but may result in lower routing costs. Even though SHRs provide excellent survivability characteristics and extremely fast reconfiguration of flow, they still result in a huge increase in capacity utilization (since we enforce a ring topology on the network), and a high cost.

A significant reduction in the amount of spare capacity can be achieved by using a general network topology and a shared protection strategy to deal with failures (Venables et al. 1993, Herzberg et al. 1995, Lisser et al. 1995, Alevras et al. 1998, Dahl and Stoer 1998, Iraschko et al. 1998, Xiong and Mason 1998, 1999, Bienstock and Muratore 2000, Balakrishnan et al. 2002). Instead of pre-assigning spare capacity to protect each commodity of the network independently, the shared protection approach spreads the redundant capacity over the whole network and uses it to restore disrupted flow. Under a shared protection scheme, the spare capacity is shared by more than one commodity.

This makes shared protection much more capacity-efficient, since the same spare capacity can be used by several rerouting paths to recover from different failures. However, which spare capacity is to be used for a specific failure, and what shared protection route will be used for each affected commodity have to be determined *a priori*. Thus, implementation of reconfiguration in the event of failure is inherently slower than dedicated protection, and requires much more complex mechanisms and software packages.

For shared protection, three different rerouting policies are considered in the literature: link rerouting, where the disrupted flow is rerouted between the end nodes of the failed edge(s); path rerouting, where the disrupted flow is rerouted all the way from the origin to the destination nodes of the commodities; and global rerouting, where the flow of all

commodities, whether disrupted or not, may be rerouted under failure. In the telecommunications literature, shared protection networks are often referred to as mesh-networks since they do not make any assumptions on the structure of the network.

In practice, ring and mesh networks are usually deployed in complete isolation from one another. Recently, there has been an increased effort in designing hybrid networks that have the reconfiguration times of ring networks and the capacity efficiency of general/mesh networks. In Section 3.4, we review the several ways in which hybrid networks are being implemented in telecommunication networks (Grover and Martens 2000).

One such step toward achieving this objective is the development of hybrid networks with predefined failure-flow patterns. In this approach, we still use a mesh network for no-failure routing, but use the predefined failure-flow patterns for all failure flows. We only reroute those demands that are disrupted by failure. For instance, using predefined undirected cycles as failure-flow patterns has been shown to be capacity-efficient, with fast reconfiguration times (Grover and Stamatelakis 1998, Stamatelakis and Grover 2000, Schupke et al. 2002). In Section 3.4.3, we present our framework that uses directed cycles as failure-flow patterns to design survivable networks.

Before discussing dedicated protection and shared protection schemes in detail in Sections 3.2 and 3.3, we define a related but simpler problem, the spare capacity assignment problem (SCP). In the SCP, given no-failure flow and network capacity, we need to determine the minimum cost allocation of spare capacity to provide survivability under the chosen scheme.

Definition 3.3 *Spare capacity assignment problem (SCP)*: Given a directed network, capacity installation costs, no-failure routing for the commodities, installed capacity on the network, and a set of failure states, we wish to reroute these commodities so that the net flow on any arc for any failure state is at most the capacity installed on that edge and all demands are met in all failure states, at minimum spare capacity installation costs. □

Solving NDP first and then SCP for the routings suggested by NDP is less capacity-efficient than solving SNP directly for the corresponding shared protection scheme, see Murakami and Kim (1995) for an example. However, this hierarchical approach may give

a good starting solution for SNP, and is used by several researchers (Herzberg et al. 1995, Iraschko et al. 1998, Sakauchi et al. 1990).

Alternatively, we can employ an iterative approach. After solving NDP and then SCP, we solve the multi-commodity flow problem (MFP) on the resulting network for the no-failure flow state. From the solution to the MFP, we install as much capacity as required on the network, and solve SCP again. This process may be repeated until costs reduce no further.

The definitions of SCP and SNP are quite generic; they do not discuss how survivability is ensured. SCP and SNP are also referred to as the hierarchical (non-joint) and integrated (joint) approaches to solving survivable network design problems. We focus on the integrated approach, except to illustrate the ideas first with the hierarchical approach, or to review work that utilizes the hierarchical approach.

In a telecommunications network, the choice of survivability strategies has a great impact on network architecture and equipment cost. In Sections 3.2 and 3.3, we take a closer look at the specific design problems arising from dedicated protection and shared protection, and review the relevant literature. In Section 3.4, we discuss the design of hybrid networks using predetermined failure-flow patterns. We review earlier work that uses undirected p-cycles in a hierarchical approach, and then present our methodology that uses directed cycles as failure-flow patterns. In Chapter 4, we study the mixed-integer formulation of our approach in detail.

3.2 Dedicated protection schemes

3.2.1 1+1 diverse protection

In 1+1 DP switching, one dedicated backup route on a node-disjoint path is allocated for each commodity. Capacity is dedicated to the backup route of each commodity and used to reroute the commodity under failure. Backup routes of different commodities do not share capacity. This scheme provides a very fast reconfiguration mechanism that is simple to implement. However, it is very expensive in terms of capacity, since we provide

enough capacity for at least twice the amount of demand for each commodity.

Definition 3.4 *1+1 diverse protection network design problem (DPP)*: Given a network, a set of commodities (in terms of their origin-destination pairs, and the demands), and a set of installable capacity types, we wish to install integer multiples of capacity on the arcs of the network so that each commodity can be simultaneously routed through two different node-disjoint paths while minimizing the total capacity installation and flow routing costs. □

We consider only unsplittable flow DPP, where the primary and backup routes can not be split into multiple paths. Splittable flow DP switching is more difficult to model, since each of the primary paths must be node-disjoint from every backup path. This forces us to use indicator binary variables for each commodity and arc combination, in addition to the continuous flow variables.

We present an arc formulation for DPP. Let $G = (V, E)$ be an undirected graph with node set V and edge set E. Let F be the set of all ordered pairs (arcs) from E, i.e., $F = \{(ij), (ji) : [ij] \in E\}$. We use (ij) to denote the arc from node i to node j, and $[ij]$ to denote the edge between nodes i and j. We use $G' = (V, F)$ to denote the directed graph with node set V and arc set F.

Let K be the set of commodities. Let $\{(s^k, t^k, d^k)\}_{k \in K}$ be the commodity triples of the source and destination nodes s^k and t^k, and d^k be the supply at s^k for t^k, for all $k \in K$. We define b_i^k as the supply of commodity k at node i, i.e., $b_{s^k}^k = d^k$, $b_{t^k}^k = -d^k$, and $b_i^k = 0$ for all $i \in V \setminus \{s^k, t^k\}$.

We define variable y_{ij}^k as the fraction of commodity k routed through arc $(ij) \in F$. Let e_{ij}^k be the cost associated with routing each unit of commodity $k \in K$. We define the capacity variable $x_{[ij]}$ as the amount of capacity installed on edge $[ij]$. Let $h_{[ij]}$ be the cost of installing unit capacity on edge $[ij] \in E$. We use $w_{[ij]}^0$ to denote the pre-existing capacity on edge $[ij] \in E$.

For this formulation, we assume that z_{ij}^k is used to denote the backup route for commodity k. Thus, $z_{ij}^k = 1$ if arc (ij) is used in the backup path for commodity k, and 0

otherwise. Then, the mathematical formulation for DPP is:

$$\min \quad \sum_{(ij)\in F}\sum_{k\in K} d^k e^k_{ij} y^k_{ij} + \sum_{[ij]\in E} h_{[ij]} x_{[ij]}$$

$$s.t.: \quad \sum_{(ij)\in F} d^k y^k_{ij} - \sum_{(ji)\in F} d^k y^k_{ji} = b^k_i \qquad \forall i \in V, k \in K \qquad (3.1)$$

$$\sum_{(ij)\in F} d^k z^k_{ij} - \sum_{(ji)\in F} d^k z^k_{ji} = b^k_i \qquad \forall i \in V, k \in K \qquad (3.2)$$

$$\sum_{(ij)\in F} (y^k_{ij} + z^k_{ij}) \leq 1 \qquad \forall i \in V \setminus \{s^k\}, k \in K \qquad (3.3)$$

$$\sum_{k\in K} d^k (y^k_{ij} + z^k_{ij}) \leq w^0_{[ij]} + x_{[ij]} \qquad \forall (ij) \in F \qquad (3.4)$$

$$z^k_{ij}, y^k_{ij} \in \mathbb{R} \qquad \forall (ij) \in F, k \in K \qquad (3.5)$$

$$x_{[ij]} \in Z_+ \qquad \forall [ij] \in E$$

In the above formulation, constraints (3.1) and (3.2) guarantee that all commodities are assigned both no-failure and backup routes. Constraints (3.3) enforce that these two routes be node-disjoint. Constraints (3.4) ensure that capacity assigned to edge $[ij]$ is large enough to accommodate demand routed on arc (ij).

To ensure that the routing of commodities is unsplittable, we replace continuous flow variables y, z by binary variables in (3.5). Interestingly, if we have no flow costs, then we can formulate DPP using only one set of flow variables y to indicate both primary and backup flow. The new formulation is obtained by removing variables z, removing constraints (3.2), and replacing the right hand side of constraints (3.1) by $2b^k_i$. This form of dedicated protection has extremely poor capacity utilization, because we allocate capacity to route twice the demand for each commodity.

3.2.2 Self-healing rings

Definition 3.5 *Self-healing ring network design problem (RNP)*: Given a two-connected network and a set of commodities (origin-destination pairs) and demand of each commodity, we wish to determine a set of feasible SHRs such that all demands are met in all failure states, at minimum cost. □

An SHR is a cluster of nodes that are connected in a loop by edges of the same capacity; a node can be connected to more than one ring. The following formulation (Soriano et al. 1998) ignores the problem of connecting the nodes in a ring after grouping them. Since this formulation is significantly different from other formulations in this dissertation, we do not follow all of the notational conventions introduced in Section 1.1; Section 3.2.2 is self-contained in terms of its notation.

Let c_r represent the cost of connecting a node to ring r, f_{rs} the inter-ring flow unit cost from ring r to ring s, u_r the capacity of ring r, h the unit cost of installing capacity on any edge, L the set of inter-ring transfer nodes, and R the set of possible SHRs, L_r the set of ring-transfer nodes on ring r. Let $x_{ir} = 1$ if node i is connected to ring r and 0 otherwise; and $y_r = 1$ if ring r is used, and 0 otherwise. We define v^k_{ir},v^k_{rj}, and $w^{k\ell}_{rs}$ as the the quantity of commodity k that enters ring r at node i, the quantity of commodity k that leaves ring r at node j, and the quantity of inter-ring flow of commodity k from ring r to ring s at inter-ring transfer node ℓ, respectively. Now, we can formulate RNP as follows.

$$\min \sum_{r\in R}\sum_{i\in V} c_r x_{ir} + \sum_{r\in R}\sum_{s\in R, s\neq r} f_{rs} \sum_{k\in K}\sum_{\ell\in L} w^{k\ell}_{rs} + h\sum_{r\in R} u_r y_r \tag{3.6}$$

$$s.t.: \quad \sum_{r\in R} v^k_{s_k r} = d^k \qquad \forall k \in K \tag{3.7}$$

$$\sum_{r\in R} v^k_{r t_k} = d^k \qquad \forall k \in K \tag{3.8}$$

$$v^k_{s_k r} + \sum_{s\in R, s\neq r}\sum_{\ell\in L_r\cap L_s} w^{k\ell}_{sr} - v^k_{r t_k} = \sum_{s\in R, s\neq r}\sum_{\ell\in L_r\cap L_s} w^{k\ell}_{rs} \qquad \forall r\in R,\ \forall k\in K \tag{3.9}$$

$$\sum_{k\in K}(v^k_{s_k r} + \sum_{s\in R, s\neq r}\sum_{\ell\in L_r\cap L_s} w^{k\ell}_{sr}) \le u_r \qquad \forall r\in R \tag{3.10}$$

$$\sum_{k\in K: s_k = i} v^k_{ir} + \sum_{k\in K: t_k = i} v^k_{ri} \le \sum_{k\in K: s_k = i \text{ or } t_k = i} d^k x^k_{ir} \qquad \forall i\in V,\ \forall r\in R \tag{3.11}$$

$$x_{ir} \in \{0,1\} \qquad \forall i\in V,\ \forall r\in R$$

$$y_r \in \{0,1\} \qquad \forall r\in R$$

$$v^k_{ir}, v^k_{ri} \in \mathbb{R}_+ \qquad \forall i\in V,\ \forall r\in R$$

$$w^{k\ell}_{rs} \in \mathbb{R}_+ \quad \forall r,s\in R\ (r\neq s),\ \forall k\in K,\ \forall \ell\in L\cap r\cap s$$

The objective (3.6) minimizes the sum of node, inter-ring flow, and capacity installation costs. The node cost corresponds to the equipment cost (c_r) of connecting a node to ring r. For each unit of flow from ring r to ring s, we incur a inter-ring flow cost f. The cost of installing capacity u_r on ring r is hu_r.

Constraints (3.7) and (3.8) ensure that for each commodity k, all the demand enters some ring at its source node, and leaves some ring at its destination node, respectively. Constraints (3.9) are flow conservation constraints ensuring that for all commodities k, the total flow entering each ring r is equal to the total flow leaving that ring. Constraint (3.10) is the capacity constraint for the SHRs and enforces that the flow on ring r is less than its capacity u_r. Finally, (3.11) is a design constraint, ensuring that no demand enters or leaves a given SHR at an origin or destination node if that node is not connected to that particular SHR.

As we can see, this formulation is already quite complex. However, this formulation does not quite model RNP completely. For instance, we have no control over the capacities of the rings, though the formulation does allow us to build up rings of different capacities by using various combinations of available capacity types. Furthermore, this formulation allows us to connect a node to a ring only if flow enters or leaves the ring through that node. Also, the formulation forces us to index rings by unique variables; the number of rings used is bounded by $|R|$.

Even more importantly, the rings have not been completely defined yet. The solution to this formulation merely tells us which nodes are connected to form a SHR. We still must find the best undirected cycle that passes through each one of the nodes assigned to a particular ring, which corresponds to solving an instance of the classical traveling salesperson problem (TSP).

RNP is an extremely difficult problem to solve. As expected, the problem can be approached from several angles, giving rise to quite different design problems, depending on the simplifying assumptions considered and the characteristics desired in the resulting survivable network. In the literature, the following special cases have been considered.

One of the simplest versions of the problem addresses a situation where the network

contains a single designated hub node where all SHRs interconnect, allowing demands to flow from one ring to another. Each node can be connected to several SHRs. In this stacked ring network design, one needs to determine which nodes connect to each SHR, and route the commodities through the rings to minimize costs.

Sutter et al. (1998) solve one such problem in which the only costs considered are the node costs. They present heuristics based on simulated annealing and an exact algorithm based on integer column generation. All rings are unidirectional SHRs of identical capacity. Armony et al. (2000) study the same problem, but use a cost function that includes both node costs and inter-ring flow costs. They develop a genetic algorithm heuristic and compare their results to optimal solutions obtained by a commercial IP solver.

A second type of network design problems using SHR protection arises when the architecture of the resulting network is required to contain multiple rings corresponding to several different topological cycles. A variant of this type of multi-ring network design problem requires that the network to be designed must have a two-level architecture consisting of a set of node disjoint rings connected by a higher level (federal) ring that carries all inter-ring traffic.

Altinkemer (1994) studies this problem in the context of computer networks. He assumes that the hub nodes composing the federal ring are given and that the only capacity constraints are that a ring can connect no more than a pre-specified maximum number of nodes (i.e., hop constraints). He presents simple heuristics exploiting the similarity of the problem with the multi-depot vehicle routing problem, and provides worst case performance bounds.

Goldschmidt et al. (2003) study the same problem in the context of optical networks. Now, in addition to the hop limits, there are capacity constraints on the rings. The authors consider only unidirectional SHRs where all rings have identical capacities. They study two versions of the problem, one where the number of lower level rings is unconstrained and the other where this value is fixed to some value λ. As in the previous work, the authors present heuristics with worst case performance guarantees.

The third group considers spare capacity assignment (SCP) using bidirectional SHRs to solve RNP. In this context, the commodities have already been routed on a meshed network, and one needs to find a set of capacitated rings such that every edge on the network is covered by at least one ring of adequate capacity.

Slevinsky et al. (1993) propose a greedy heuristic based on 7 different measures of capacity efficiency to solve this problem. They test these measures on sparse networks ranging from 6 to 39 nodes, and present better results than any known before for the same instances. For the smaller instances, they obtain the optimal solution for at least one of the metrics used.

More recently, a set covering approach is used by Luss et al. (1998) to develop a heuristic procedure. They solve a version of the problem in which capacities are not explicitly taken into account. They test their heuristic on two large sparse instances (50 and 509 nodes, respectively) and obtain feasible solutions that are 12% from the optimal, on average.

The final group of research attempts to incorporate no assumptions on the ring topology; node clusters defining rings are not required to be disjoint and the resulting architecture is not hierarchical by definition. Some of the earlier and better known work on this topic was carried out at Bellcore and resulted in the planning software known as SONET Toolkit, see Cosares et al. (1995).

Toolkit employs a greedy approach that sequentially constructs a survivable network design. In the first stage, clusters of nodes that possibly make good SHRs are identified. In the second stage, a heuristic procedure determines if the network contains a cycle connecting the cluster nodes to form a SHR. While there are still commodities to protect, the procedure tries to identify other interesting clusters. Once all commodities have been protected, the capacities of the SHRs are determined.

The first application of Toolkit to a real network planning scenario from Bellcore provided a solution with a cost 0.9 million dollars less than the current best solution (manually prepared) of 3.7 million dollars. In most trial networks, the authors report a cost saving of around 10-25%.

3.3 Shared protection schemes

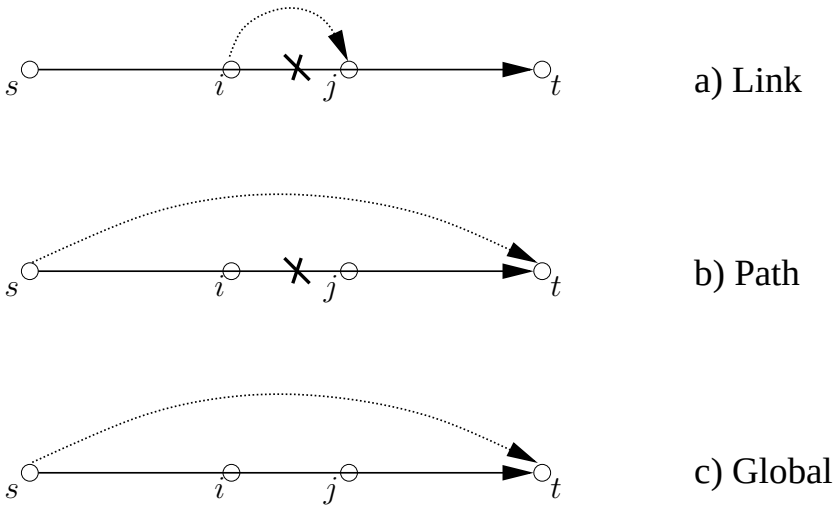
Networks with minimum capacity requirements can be obtained by shared protection schemes using a global rerouting policy. However, implementation of global rerouting of all flows (whether disrupted or not) in the event of a failure requires much more complex hardware and software packages and is inherently slower than dedicated protection schemes such as SHR and 1+1 DP. Furthermore, the size of failure scenario based shared protection models grows very rapidly with the size of the graph and render such models unfit for tackling practical problems.

Hierarchical shared protection schemes are popular for designing survivable networks in practice. In the first stage, working flows and edge capacities are determined for the no-failure scenario without survivability concerns. In the second stage, sufficient spare capacity is assigned to the edges of the network so that the disrupted flow can be safely rerouted in the case of failures (Herzberg et al. 1995, Grover and Stamatelakis 1998, Iraschko et al. 1998, Balakrishnan et al. 2001, 2002). Shared protection schemes can be primarily differentiated based on the rerouting policy considered: link, path, or global rerouting, see Figure 3.1.

We present the formulations for these shared protection schemes and discuss related literature; also see Xiong and Mason (1999) for a detailed comparison of the shared protection strategies. In Figure 3.1, we illustrate the differences between global, path and link rerouting. We consider a commodity that is routed from node s to node t using the arc (ij). In link rerouting (a), when edge $[ij]$ fails, the commodity is rerouted from node i to node j. In path rerouting (b), when edge $[ij]$ fails, the commodity is rerouted from source node s to destination node t. In global rerouting, the commodity may be rerouted from source s to destination t even if no edge on the no-failure route fails.

We model diversification, which is a requirement enforced by telecommunication companies in the design of survivable networks. Under diversification requirements, no more than a certain fraction (γ_k) of commodity k can be routed on a particular arc under no failure. This ensures that only a certain fraction (γ_k) of any commodity will be affected

Figure 3.1: Shared protection strategies



under single-edge failures. By enforcing that every commodity is routed on several edge-disjoint paths, not all paths are affected in the case of a component failure. As we shall see, this can be easily enforced in shared protection and hybrid schemes. However, for all the computations in this dissertation, we will assume that $\gamma_k = 1$.

3.3.1 Global rerouting

In global rerouting shared protection (GNP), all demands may be rerouted under any failure, even those that are not disrupted. We use $F \setminus [ij]$ to represent $F \setminus \{(ij), (ji)\}$. Let S be the set of failure states. The mathematical formulation for GNP is:

$$\min \quad \sum_{(ij)\in F} \sum_{k\in K} e_{ij}^k d^k y_{ij}^{k0} + \sum_{[ij]\in E} h_{[ij]} x_{[ij]}$$

$$s.t.: \quad \sum_{(ij)\in F} d^k y_{ij}^{ks} - \sum_{(ji)\in F} d^k y_{ji}^{ks} = b_i^k \qquad \forall i \in V,\ \forall k \in K,\ \forall s \in S \tag{3.12}$$

$$\sum_{k\in K} d^k y_{ij}^{ks} \le w_{[ij]}^0 + x_{[ij]} \qquad \forall (ij) \in F \setminus \{s\},\ \forall s \in S \tag{3.13}$$

$$0 \le y_{ij}^{ks} \le 1 \qquad \forall (ij) \in F \setminus \{s\},\ \forall k \in K,\ \forall s \in S \setminus \{0\}$$

$$0 \le y_{ij}^{k0} \le \gamma_k \qquad \forall (ij) \in F,\ \forall k \in K$$

$$x_{[ij]} \in \mathbb{Z}_+ \qquad \forall [ij] \in E$$

In the above formulation, constraints (3.12) guarantee that all demands are satisfied in all failure states. Constraints (3.13) ensure that capacity assigned to edge $[ij]$ is large enough to accommodate demand routed on arc (ij) for all possible failure states. In this formulation, the flow is splittable.

For the unsplittable flow formulation, we simply force the x_{ij}^{ks} variables to be binary. Obviously, unsplittable flow requires that all diversification parameters γ_k be 1. The definition of failure states easily allows us to model failures more complex than edge failures.

As mentioned earlier, GNP yields the most capacity-efficient capacitated survivable networks (by setting flow costs e to zero), if it can be solved to optimality. This is because GNP imposes no restrictions on either the network structure or the routing of flow.

This formulation is also known as the arc formulation since there is a flow variable y_{ij}^{ks} for each arc in each failure state. It requires $|V||K||S|+|F||S|$ constraints. For a complete network of $|V|$ nodes with complete demand, $|F| = |K| = |V|(|V|-1)$. If we consider only single-edge failures, then $|S| = |V|(|V|-1)/2+1$. When $|V| = 10$, this formulation has 45540 constraints and 372690 variables. When $|V| = 20$, it has 148980 constraints and 1451790 variables.

Even when the network is not as dense, this formulation can easily run into tens of thousands of constraints and millions of variables even for medium-sized networks, making it very hard to load into computer memory, let alone solve optimally. The size of GNP is larger than NDP by an order of magnitude $|S|$, since NDP is the special case of GNP with the single no-failure scenario $S = \{0\}$.

To alleviate this problem, one introduces a variable to indicate the fraction of a commodity that is routed through a certain path. Naturally, this formulation is often referred to as the path formulation. The number of path variables is exponential in the number of arcs. As in the arc formulation, we define γ_k as the maximum fraction of commodity k that may be routed by any path in the no-failure state.

In path-based formulations, y_p indicates the fraction of commodity routed on path p. P_k^s denotes the set of paths from s^k to t^k in failure state s. For any path p, we set $\delta_{ij}^p = 1$

if it includes arc (ij), and 0 otherwise. The path formulation for GNP is:

$$\min \quad \sum_{k\in K}\sum_{p\in P_k^0}\sum_{(ij)\in F} d^k e_{ij}^k \delta_{ij}^p y_p + \sum_{[ij]\in E} h_{[ij]} x_{[ij]}$$

$$s.t.: \quad \sum_{p\in P_k^s} y_p = 1 \qquad \forall k \in K,\ \forall s \in S$$

$$\sum_{k\in K}\sum_{p\in P_k^s} d^k \delta_{ij}^p y_p \le w_{[ij]}^0 + x_{[ij]} \qquad \forall (ij) \in F \setminus \{s\},\ \forall s \in S$$

$$0 \le y_p \le 1 \qquad \forall p \in P_k^s,\ \forall k \in K,\ \forall s \in S \setminus \{0\}$$

$$0 \le y_p \le \gamma_k \qquad \forall p \in P_k^0,\ \forall k \in K$$

$$x_{[ij]} \in \mathbb{Z}_+ \qquad \forall [ij] \in E$$

The path formulation requires only $|K||S| + |F||S|$ constraints, far fewer than the arc formulation. However, this is offset by the fact that there are an exponential number of path variables in this formulation.

The formulation for the spare capacity assignment problem (SCP) with global rerouting is slightly simpler. In the SCP, the no-failure flow and network capacity are known *a priori*; we only need to determine the minimum cost allocation of spare capacity to provide survivability under the chosen scheme. In the formulation for SCP, we do not have the no-failure flow variables ($y_{ij}^{k0},\ k \in K$ for arc formulations, and $y_p,\ p \in P_k^0$ for path formulations), and the constraints for no-failure flow. Instead, we are given the flow quantities g_{ij}^0 indicating the total no-failure flow on arc (ij). We minimize only the capacity installation costs. The size of the formulation for SCP is of the same magnitude as that of GNP.

In Section 5.4, we show that the formulations for GNP and SCP with global rerouting are too big to solve to optimality with current state-of-the-art solvers. Therefore, researchers have adopted various heuristic approaches to solve the problem.

Alevras et al. (1998) and Dahl and Stoer (1998) use the path formulation to study GNP. They first solve the LP relaxation, and then run heuristics to obtain a feasible solution. Both study the splittable flow case, and also model diversification (where no path may carry more than a certain fraction of the flow). Both develop valid inequalities and incorporate them in a cutting-plane framework to improve the LP relaxation before applying

the heuristics. They differ in the classes of inequalities generated.

Alevras et al. (1998) develop three classes of inequalities; strengthened partition inequalities, strengthened metric inequalities, and diversification inequalities. They use four real-life networks from a German telecommunication services provider to test their algorithm. They are sparse networks (around $3|V|$ edges), where the number of nodes $|V|$ varies from 11 to 17. Their running times vary from 7 minutes all the way up to 900 minutes on a SUN Ultra-1. Further, the gaps between the strengthened LP relaxation and best feasible solution obtained at termination are quite large (50% on average). In other words, the inequalities added reduce the integrality gap only by around 50%. The authors attribute the weak lower bounds to the fact that some of the classes of inequalities are useful only under particular parameter combinations.

On the other hand, Dahl and Stoer (1998) study the cut sets for each failure state to develop two new classes of inequalities; diversified metric inequalities and band inequalities. They use two very sparse networks (27 nodes, 51 edges; and 8 nodes, 13 edges) to test their algorithm, but obtain much better gaps at termination (12% and 8%)

Lisser et al. (1995) solve the LP relaxation of the arc formulation using lagrangian relaxation; dualizing the capacity constraints (3.13). The authors attempt to solve large instances (60 nodes, 120 arcs). These instances have more than $200,000$ constraints in the formulation and were not solvable using the commercial LP solvers available then. They solve the lagrangian dual by generating cutting planes. This technique allows them to solve the LP relaxations optimally. While the authors present a technique for solving really large LP relaxations optimally, they do not attempt to obtain feasible integer solutions from the LP solutions.

Bienstock and Muratore (2000) study a problem where only a fraction of each demand has to be satisfied in any failure state. They study the corresponding cut set and present several classes of facet-defining inequalities to strengthen the cut-set polyhedra, which arise as subsystems of the GNP. They extend the basic model to incorporate cut sets that arise in the context of node failures. Finally, they present some heuristics to separate these inequalities, but provide no computational results.

3.3.2 Path rerouting

We now present a path-based formulation that models the path rerouting scheme. In path rerouting shared protection (PNP), the disrupted demand is rerouted all the way from the origin to the destination nodes of the commodities. To model PNP, we need to know exactly which commodities were affected under failure. This can be easily formulated using a path-based formulation, which we present here. The arc-based formulation to model PNP is more complex; since it requires us to introduce an indicator variable for each commodity, arc and failure state combination to track what flows are effected.

For any path p, we set $\zeta_s^p = 1$ if it is affected by failure state $s \in S \setminus \{0\}$, and 0 otherwise. The path formulation for path rerouting shared protection (PNP) is:

$$\min \sum_{k\in K}\sum_{p\in P_k^0}\sum_{(ij)\in F} d^k c_{ij}^k \delta_{ij}^p y_p + \sum_{[ij]\in E} h_{[ij]} x_{[ij]}$$

$$s.t.: \quad \sum_{p\in P_k^0} y_p = 1 \qquad \forall k \in K$$

$$\sum_{k\in K}\sum_{p\in P_k^0} d^k \delta_{ij}^p y_p \le w_{[ij]}^0 + x_{[ij]} \qquad \forall (ij) \in F$$

$$\sum_{p\in P_k^s} y_p = \sum_{p\in P_k^0} \zeta_s^p y_p \qquad \forall k \in K, \forall s \in S \setminus \{0\} \tag{3.14}$$

$$\sum_{k\in K}\sum_{p\in P_k^0} d^k \delta_{ij}^p (1-\zeta_s^p) y_p + \sum_{k\in K}\sum_{p\in P_k^s} d^k \delta_{ij}^p y_p \le w_{[ij]}^0 + x_{[ij]} \qquad \forall (ij) \in F \setminus \{s\}, \forall s \in S \setminus \{0\} \tag{3.15}$$

$$0 \le y_p \le 1 \qquad \forall p \in P_k^s, \forall k \in K, \forall s \in S \setminus \{0\}$$

$$0 \le y_p \le \gamma_k \qquad \forall p \in P_k^0, \forall k \in K,$$

$$x_{[ij]} \in \mathbb{Z}_+ \qquad \forall [ij] \in E$$

In the above formulation, constraints (3.14) ensure that each affected demand will take alternative routes in case of failure. Constraints (3.15) ensure that capacity assigned to edge $[ij]$ is large enough to accommodate demand routed on arc (ij) for all possible failure states; first and second terms on the left hand side correspond to no-failure flow

and failure flows, respectively.

Both global and path rerouting require $|K||S| + |F||S|$ constraints in the path formulation. Hence, global rerouting is more capacity-efficient than path rerouting, and at no additional cost. However, path rerouting is easier to implement in practice since we only need to reroute those demands that were disrupted by failure.

Again, the formulation for SCP under path rerouting is simpler than PNP, but not by much. We do not require the no-failure flow variables (y_p), and the constraints for no-failure flow. Instead we treat no-failure routing as given data (g^0_{ij}). We minimize only the capacity installation costs. The formulation is much sparser, but the size of the problem does not decrease significantly. Interestingly, since we now have complete information about no-failure flow, we can easily model the SCP under path rerouting using an arc formulation, as opposed to PNP. However, all shared protection schemes are equally hard to solve in the hierarchical framework since one now has complete information on what demands to reroute for each failure state.

Iraschko et al. (1998) study SCP with path rerouting. They also discuss how their formulation is modified to accommodate link rerouting, and provide lower bounds on spare capacity requirements in both path and link rerouting. They extend their formulation to model PNP. Finally, they evaluate the capacity efficiency obtained from solving PNP directly in an integrated framework, as opposed to solving NDP followed by SCP on a test set of 5 problem instances (sparse networks, number of nodes varying from 10 to 30).

The authors consider only single-edge failures. For their problem instances, solving PNP directly gives us a 7% reduction (on average) in capacity utilization as compared with solving NDP followed by SCP in a hierarchical framework. The authors do not present any specific solution techniques; they directly use CPLEX to solve the formulations.

Xiong and Mason (1998) also study PNP and compare it with SCP with path rerouting. They evaluate the capacity efficiency obtained from solving PNP directly in an integrated framework, as opposed to solving NDP followed by SCP on a test set of 4 problem instances (sparse networks, number of nodes varying from 11 to 20). For their problem instances, solving PNP directly gives us a 4% reduction (on average) in capacity utiliza-

tion as compared with solving NDP followed by SCP in a hierarchical framework. However, they consider only 5 possible routes for each commodity. The authors also develop improvement heuristics to solve both approaches.

3.3.3 Link rerouting

We now present a mathematical formulation that models link rerouting shared protection (LNP). In LNP, the disrupted demand is rerouted between the end nodes of the failed edge; we present the arc formulation for LNP.

$$
\begin{aligned}
\min \sum_{(ij)\in F}\sum_{k\in K} c_{ij}^k d^k y_{ij}^{k0} + \sum_{[ij]\in E} h_{[ij]} x_{[ij]} & \\
s.t.: \sum_{(ij)\in F} d^k y_{ij}^{ks} - \sum_{(ji)\in F} d^k y_{ji}^{ks} = b_i^k & \quad \forall i \in V,\ \forall k \in K,\ \forall s \in S \\
\sum_{k\in K} d^k y_{ij}^{ks} - w_{[ij]}^0 \le x_{[ij]} & \quad \forall (ij) \in F \setminus \{s\},\ \forall s \in S \\
y_{ij}^{k0} \le y_{ij}^{ks} \le 1 & \quad \forall (ij) \in F \setminus \{s\},\ \forall k \in K,\ \forall s \in S \setminus \{0\} \qquad (3.16) \\
0 \le y_{ij}^{k0} \le \gamma_k & \quad \forall (ij) \in F,\ \forall k \in K \\
x_{[ij]} \in \mathbb{Z}_+ & \quad \forall [ij] \in E
\end{aligned}
$$

Constraints (3.16) ensure that under failure, the disrupted demand is rerouted around the endpoints of the failed edge. This formulation is obtained by adding constraints (3.16) to the arc formulation for GNP. As a result, this forces us to use $|K||S||F|$ more constraints than in the GNP. At the same time, it also gives us a less capacity-efficient solution than GNP. However, the path formulation of LNP is smaller than the path formulations for GNP and PNP.

Let B_s be the set of node-pairs affected by the failure state s. A single-edge failure s corresponds to exactly two affected node-pairs that correspond to the arcs on the edge; node-pairs are defined to be directed. Let R_b denote the set of paths from the first node to the second node of node-pair $b \in B_s$. These paths exclude arcs affected by the failure state s. Demands disrupted by the failure of flow between node-pair b are rerouted among the paths on R_b. We use new variables y'_p to denote the amount of flow rerouted on path

$p \in R_b$. The path formulation for LNP is:

$$\min \quad \sum_{k \in K} \sum_{p \in P_k^0} \sum_{(ij) \in F} d^k c_{ij}^k \delta_{ij}^p y_p + \sum_{[ij] \in E} h_{[ij]} x_{[ij]}$$

$$s.t.: \quad \sum_{p \in P_k^0} y_p = 1 \qquad \forall k \in K$$

$$\sum_{k \in K} \sum_{p \in P_k^0} \zeta_s^p d^k y_p = \sum_{p \in R_b} y'_p \qquad \forall b \in B_s,\ \forall s \in S \tag{3.17}$$

$$\sum_{k \in K} \sum_{p \in P_k^0} d^k \delta_{ij}^p y_p + \sum_{k \in K} \sum_{p \in R_b} \delta_{ij}^p y'_p \leq w_{[ij]}^0 + x_{[ij]} \qquad \forall (ij) \in F \setminus \{s\},\ \forall b \in B_s,\ \forall s \in S \tag{3.18}$$

$$0 \leq y'_p \qquad \forall p \in R_b,\ \forall b \in B_s$$

$$0 \leq y_p \leq \gamma_k \qquad \forall p \in P_k^0,\ \forall k \in K$$

$$x_{[ij]} \in \mathbb{Z}_+ \qquad \forall [ij] \in E$$

In the above formulation, constraints (3.17) ensure that flow on each arc is restored in case the arc fails. Constraints (3.18) ensure that capacity assigned to edge $[ij]$ is large enough to accommodate demand routed on arc (ij) for all possible failure states; first and second terms on the left hand side correspond to no-failure flow and failure flows, respectively. LNP requires $|K| + |F||S| + |F|$ constraints in the path formulation, fewer than in GNP and PNP. By redefining the commodities, the arc and path formulations for SCP under link rerouting are the same as the corresponding formulations for SCP using path rerouting. For each failed node pair b, we define a commodity k with source and destination points as the node pair, and demand d^k which is equal to the no-failure flow disrupted by failure of b.

Venables et al. (1993) compare 2 heuristics to solve SCP under link rerouting. The first one, which they call Spare Link Placement Algorithm (SLPA), is a greedy algorithm that adds spare capacity to the edges of the network. At each iteration, it adds spare capacity to the edge that has highest span restorability; a measure they define for this algorithm. After the network is fully survivable, the algorithm attempts to minimize redundancy while maintaining survivability. The second heuristic, Iterative Cutsets Heuristic

(ICH), formulates spare capacity placement as an LP subject to constraints based on a subset of cut sets of the network. On solving this LP formulation, if the resultant network is not survivable, more cut sets are generated and added for the arcs not restored. This process is repeated until a survivable network is obtained. Finally, the capacities are rounded up to integer values. The authors test both heuristics on large problem instances with 20 to 100 nodes, and $|F|/|V|$ between 3 and 6. ICH performed better for all instances, with an average gap of 4% at termination, as compared with 12% for SLPA. However, ICH was unable to solve instances larger than 50 nodes, and took much more time than SLPA even for the smaller instances.

Herzberg et al. (1995) use the path formulation to study the splittable flow SCP with link rerouting. They attempt to minimize the maximum spare capacity installed on any edge. They first solve the LP relaxation, round up the LP optimal solution to obtain feasible solutions and then use a series of Max-Flow tests to tighten the feasible solution. They restrict the set of feasible failure-flow paths by imposing a hop-limit (length limit), but show by means of a numerical example on a large sparse network (100 nodes, 200 edges) that efficient spare capacity solutions can still be found with relatively small hop-limit values (7 for this instance).

3.3.4 Comparison of shared protection schemes

As mentioned above, the most capacity-efficient networks can be designed by global rerouting. However, there are at least two reasons as to why such a framework is not used in practice. The first one is that it has $\mathcal{O}(|V|^5)$ number of variables and constraints for a complete network and is, therefore, impractical for designing networks except for very small instances. The second reason is that its solution involves globally rerouting flow on the network whether or not the flow of a commodity is disrupted by the failed edge. Solutions with minimal changes to no-failure flow are preferred because it is highly undesirable in practice to manipulate unaffected flow while restoring affected flows. Therefore, a number of practical models and strategies have been developed for designing survivable networks that admit path or link rerouting of flow on a failed edge.

All shared protection schemes are inherently more difficult to implement than dedicated protection. In Table 3.1, we summarize the advantages and disadvantages of the three shared protection schemes with respect to the following properties.

Table 3.1: Comparison of shared protection schemes

Capacity utilization	
GNP	Most capacity-efficient since it is a relaxation of path and link rerouting schemes, as we may reroute any commodity, not just those affected by a failure.
PNP	More capacity-efficient than LNP since any solution to link rerouting is also a solution to path rerouting.
LNP	Least capacity-efficient. In fact, link rerouting often forces us to backtrack, sending the same commodity in both directions on an edge.
Computational efficiency	
GNP	Path-based formulations of the same size as PNP.
PNP	Less capacity-efficient solutions when compared with GNP, but no easier computationally.
LNP	Modeled in a more compact formulation, and is easier to solve than path and global rerouting.
Ease of implementation	
GNP	Hardest to implement since all routings might change under a failure, even if they are not disrupted due to a particular failure.
PNP	Easier to implement, and has faster reconfiguration times, than GNP.
LNP	Faster than path and global rerouting since decentralized failure routing is much faster.

3.4 Hybrid network design

3.4.1 Introduction

In both path and arc formulations of all shared protection schemes, the introduction of failure states explodes the size of the formulation by order of magnitude of $|S|$. It is therefore not surprising that these problems are exceedingly difficult to solve, and most of the existing work involves the development of specialized heuristic techniques. They are also more difficult to implement in practice than dedicated protection, and result in much slower rerouting under failure. In particular, rerouting of disrupted as well as undisrupted flow in case of a failure, as is the case in GNP, is harder to implement than rerouting only

disrupted flow. Furthermore, shared protection networks are much more expensive to set up and maintain, as they require sophisticated hardware and software.

Rings, on the other hand, require much simpler equipment and control mechanisms at the nodes to reroute disrupted demands, and are much easier to implement. Both ring networks and 1+1 DP are implemented using specialized type of equipment called add/drop multiplexers (ADMs) at each node. ADMs insert/remove signals to/from edges at source/destination points. On the other hand, shared protection on mesh networks require the use of digital cross-connect systems (DCSs) at each node to sort and reroute signals. DCSs are much more complicated than ADMs and are naturally more expensive.

Shared protection schemes are much more capacity-efficient than dedicated protection schemes; GNP described in Section 3.3.1 achieves the lowest capacity requirement of all schemes. As a result, there has been a recent interest in developing hybrid networks that are nearly as capacity-efficient as shared protection and at the same time are comparable to dedicated protection in reconfiguration times, equipment costs, and ease of implementation.

There are two limited senses in which hybrids are already being implemented in telecommunication networks. One is the principle of ring access and mesh transport. This is a widely used form of hybrid where shared protection is applied for backbone networks even if switching times are longer because it offers significant savings in capacity requirements. On the other hand, SHRs are used for local area networks (LANs) since the reduced demand requirements do not provide capacity savings to justify the use of more expensive routing equipment. In this approach, the main issue is about deciding the spatial boundary defining the mesh core and access periphery. Once that is done, ring and mesh networks are designed separately. The LANs remain spatially distinct from the mesh backbone networks.

A second practice that is implemented is meshed-rings, which are ring-based networks with inter-ring transitions being DCS-managed by shared protection. This requires that DCSs also function as ring terminals, which has led to integration of the ADM terminals as DCS shelf equipment. This virtually eliminates any cost for a demand to transit

between ring and mesh environments in the hybrids that we may consider or design.

The third type of hybrid networks allows for the allocation of capacity on several undirected cycles to act as SHRs. Any fraction of a commodity may now be routed on these rings, and is inherently survivable. The rest of the commodity is routed as in a mesh-network, and shared protection must be provided using any of the schemes discussed in Section 3.3. For any given edge, a part of the capacity installed may be utilized as a ring, and the rest as a mesh. However, any capacity allocated as rings will behave like an SHR; this capacity is dedicated to each ring, and is not to be shared with other rings or with any shared protection scheme. Typically, in an optimal solution, some of the commodities (or a fraction thereof) will be routed on the rings.

In Grover and Martens (2000), the authors introduce this variant and present an IP formulation. They present 2 heuristics and evaluate their performance against the optimal IP solution for 6 test problems. They show that even when capacity used as rings is 40% cheaper than capacity used as a mesh, the optimal solution contains both ring and mesh components, and not all-ring. This illustrates the effectiveness of a hybrid network that allows the user to choose ring and mesh protection simultaneously.

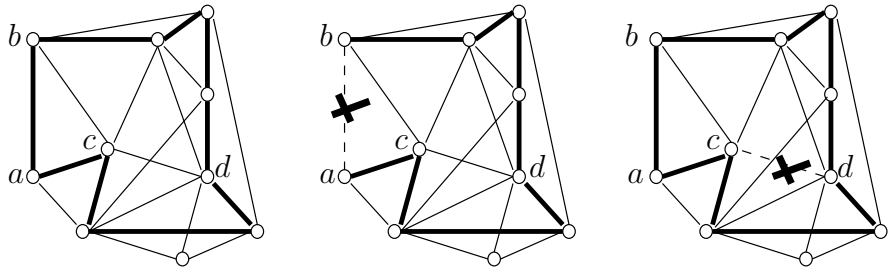
In the rest of Section 3.4, we focus on the fourth (and final) method of designing hybrid networks, which involves the use of failure-flow patterns for restoration of disrupted flow. This approach is different from the previous methods in the sense that we no longer try to incorporate shared protection and SHRs into a single network. We still use a mesh network for no-failure routing, but use the failure-flow patterns for all failure flows. Only those commodities that are disrupted by failure are rerouted. More than one failure-flow pattern may be used to restore disrupted flow on an arc.

Using these predetermined failure-flow patterns also allows us to deal with formulations that are much smaller than those required by shared protection, since we do not have to solve an NDP for each failure state. This is an added motivation to study these structures; not only do they have the advantages of both dedicated protection (ease of implementation) and shared protection (capacity efficiency), but they are also much easier to solve. This allows us to attack much larger instances than before.

3.4.2 Predefined undirected cycles

Grover and Stamatelakis (1998) introduced the concept of utilizing predefined undirected cycles for pre-configuration of spare capacity for the design of hybrid networks with ring-like speed and mesh-like capacity. They call these pre-configured failure-flow structures p-cycles. In this scheme sufficient spare capacity is installed on the undirected p-cycles of the graph so that the working capacity on any edge from the solution of the NDP is covered by the p-cycles that pass through both the nodes defining the edge. Next, we illustrate how these structures work, and present a mathematical formulation for designing survivable networks with undirected p-cycles.

Figure 3.2: An undirected p-cycle



(a) A p-cycle (b) Link $[ab]$ fails (c) Link $[cd]$ fails

Figure 3.2 illustrates the way in which an individual undirected p-cycle may be used for restoration. In (a), an example of a undirected p-cycle is shown in bold edges. In (b), edge $[ab]$ on the undirected p-cycle fails, and the remaining edges of the p-cycle are used for rerouting the flow on edge $[ab]$. In (c), we see how the undirected p-cycle can also be used for restoring the flow on an edge that is a chord of the p-cycle. Here edge $[cd]$ fails, and the undirected p-cycle provides two restoration paths between c and d. Thus installing half the working capacity of $[cd]$ on the undirected p-cycle as spare capacity covers edge $[cd]$. The use of an undirected p-cycle as in case (c) is the most advantageous since no capacity is directly utilized to provide this survivability in both directions; it is shared with capacity required to support failure on the p-cycle.

The undirected p-cycle in Figure 3.2 provides a restoration path for nine on-cycle

failures and for ten off-cycle failures. For a given undirected p-cycle, we refer to these failures as edge and chord failures, respectively, for obvious reasons. Also, more than one undirected p-cycle may be used to restore disrupted flow on an edge. Grover and Stamatelakis (1998) approach the survivable network design problem using undirected p-cycles in an hierarchical manner. In the first stage, a capacity-efficient solution (flows and edge capacities) is found for the NDP. In the second stage, given the working capacities, a minimum cost allocation of spare capacity on the edges is determined so that flow on each arc can be routed, in case of single-edge failure.

Let $(\bar{g}^0, \bar{w}^0)$ be a solution for the NDP, and C be the set of simple undirected cycles of G. Grover and Stamatelakis (1998) generate all undirected p-cycles in G with a certain length restriction. We refer to this hierarchical spare capacity assignment formulation as HUP, since it uses undirected p-cycles. For undirected p-cycles, we say that $\alpha^c_{[ij]}$ is 1 if undirected p-cycle c includes edge $[ij]$, and 0 otherwise. Let $\rho^c_{[ij]}$ be 1 if edge $[ij]$ is a chord to p-cycle c, and 0 otherwise. The decision variable z_c denotes the number of spare capacity units assigned to undirected p-cycle $c \in C$. Then, $\sum_{c \in C} \alpha^c_{[ij]} z_c$ is the spare capacity installed on edge $[ij]$. The authors give the following integer set covering model for HUP:

$$
\begin{aligned}
\min \quad & \sum_{[ij] \in E} h_{[ij]} \sum_{c \in C} \alpha^c_{[ij]} z_c & & \\
& \sum_{c \in C} (\alpha^c_{[ij]} + \rho^c_{[ij]}) z_c \geq w^0_{[ij]} & \forall [ij] \in E & \qquad (3.19) \\
& z_c \in \mathbb{Z}_+ & \forall c \in C &
\end{aligned}
$$

Naturally, solving the NDP first and then assigning spare capacity to cover the working edges is less capacity-efficient than solving an integrated framework directly. However, this hierarchical approach breaks the task of designing a survivable network into two problems that are much easier to tackle computationally than any shared protection scheme (even a hierarchical scheme); it is often preferred in practice. HUP has been reported to achieve much better capacity efficiency than ring architectures as well as very quick recovery times by several authors (Grover and Stamatelakis 1998, Stamatelakis

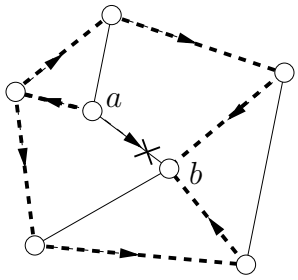
and Grover 2000, Schupke et al. 2002).

In an attempt to explain why using undirected p-cycles results in capacity-efficient networks, Stamatelakis and Grover (2000) study the effect of using various predetermined failure-flow structures for uncapacitated networks. As a criterion of measurement, they use the ratio of number of edges covered to the number of edges in the structure, and prove that no other structure can be more capacity-efficient for this measure. They do so by calculating an upper bound on the maximum ratio that any generic failure-flow structure may attain and then showing that undirected p-cycles realize the same value.

3.4.3 Survivable design using directed cycles

In our methodology, we use directed cycles as failure-flow patterns to design survivable networks. In Figure 3.3 we show two directed cycles, drawn in dashed arcs, that cover disrupted flow on arc (ab) in the reverse direction. If edge $[ab]$ fails, then the flow along arc (ab) can be rerouted from node a to node b along these two directed cycles if the sum of reserved slack on the cycles is at least the flow quantity on (ab). These directed cycles are used to reroute not only flow on arc (ab), but also on the other arcs they consist of.

Figure 3.3: Two directed cycles protecting the flow



3.4.3.1 Formulations

Let $\mathcal{C}$ be the set of directed cycles of G'. For directed cycle c, we define variable z_c to denote amount of slack reserved on all arcs of cycle c. We use slack to refer to fractional capacity that is reserved to cover no-failure flows; z are modeled as continuous variables.

The integrated approach of designing survivable networks using directed cycles (SDC) is formulated as

$$\min \sum_{(ij)\in F} \sum_{k\in K} d^k e^k_{ij} y^k_{ij} + \sum_{[ij]\in E} h_{[ij]} x_{[ij]}$$

$$s.t.: \quad \sum_{(ij)\in F} d^k y^k_{ij} - \sum_{(ji)\in F} d^k y^k_{ji} = b^k_i \qquad \forall i \in V,\ \forall k \in K \tag{3.20}$$

$$\sum_{k\in K} d^k y^k_{ij} - \sum_{c\in\mathcal{C}} \alpha^c_{ji} z_c \le 0 \qquad \forall (ij) \in F \tag{3.21}$$

$$\sum_{k\in K} d^k y^k_{ij} + \sum_{c\in\mathcal{C}} \alpha^c_{ij} z_c \le w^0_{[ij]} + x_{[ij]} \qquad \forall (ij) \in F \tag{3.22}$$

$$x_{[ij]} \in \mathbb{Z}_+ \qquad \forall [ij] \in E$$

$$z_c \in \mathbb{R}_+ \qquad \forall c \in \mathcal{C}$$

$$y^k_{ij} \in \mathbb{R}_+ \qquad \forall (ij) \in F,\ \forall k \in K$$

Constraints (3.21) ensure that for each arc (ij), the total slack reserved on the directed cycles using the reverse arc (ji) is at least the total flow on (ij). Constraints (3.22) ensure that capacity installed on edge $[ij]$ is large enough to accommodate the flow routed on arc (ij) as well as the slack reserved for directed cycles using the arc. If constraints (3.21) and the directed cycle variables z are dropped from SDC, then we obtain NDP. No cost is associated with the directed cycle variables, since we wish to compare the cost of SDC with that of NDP, and flow is routed using directed cycle variables only if there is a failure.

Since the SDC has exponentially many directed cycle variables, not all variables can be included in the model when solving large instances. Selecting a small subset of the variables *a priori* and solving the model with these variables can result in suboptimal solutions. In Section 4.1.2.1, we develop a column generation approach (see Section 1.4) that introduces the directed cycle variables into the SDC as they are needed. To solve larger instances efficiently, we reformulate the SDC using path variables rather than arc variables to model flow. This reduces the number of constraints, and introduces exponentially many path variables; the new variables can also be priced as needed via column generation (Section 4.1.2.2).

Defining u, v, w are the corresponding dual variables of the LP relaxation of SDC, the path formulation is as follows.

$$\min \quad \sum_{(ij)\in F} \sum_{k\in K} \sum_{p\in P_k} d^k \delta_{ij}^p e_{ij}^k y_p + \sum_{[ij]\in E} h_{[ij]} x_{[ij]}$$

$$(w^k) \qquad \sum_{p\in P_k} y_p = 1 \qquad \forall k \in K \qquad (3.23)$$

$$(u_{ij}) \qquad \sum_{k\in K} \sum_{p\in P_k} d^k \delta_{ij}^p y_p - \sum_{c\in\mathcal{C}} \alpha_{ji}^c z_c \leq 0 \qquad \forall (ij) \in F \qquad (3.24)$$

$$(v_{ij}) \qquad \sum_{k\in K} \sum_{p\in P_k} d^k \delta_{ij}^p y_p + \sum_{c\in\mathcal{C}} \alpha_{ij}^c z_c \leq w_{[ij]}^0 + y_{[ij]} \qquad \forall (ij) \in F \qquad (3.25)$$

$$y_p \in \mathbb{R}_+ \qquad \forall p \in P_k,\ \forall k \in K$$

$$x_{[ij]} \in \mathbb{Z}_+ \qquad \forall [ij] \in E$$

$$z_c \in \mathbb{R}_+ \qquad \forall c \in \mathcal{C}$$

Constraints (3.23) ensure the demand for each commodity is satisfied for the no-failure scenario. Constraints (3.24) ensure that sufficient slack is allocated to directed cycles to cover all no-failure flow on each arc. Constraints (5.6) ensure that for each arc, sufficient capacity is installed on the edge for the slack introduced on directed cycles and the no-failure flow on the arcs.

3.4.3.2 Related work

In Grover and Stamatelakis (1998), the authors first determine edge capacities for the no-failure scenario, and then add sufficient capacity on undirected p-cycles to protect working edge capacities, see Section 3.4.2. Their approach differs from ours (SDC) in the following aspects. Firstly, they protect working capacities and not flows; hence their model does not utilize existing slack on the edges of the network. Secondly, their model allows the undirected p-cycles to reroute disrupted flow on the chords. Thirdly, they propose a hierarchical scheme which is less capacity-efficient that our integrated scheme; see Chapter 5 for a comparison.

A related combinatorial survivable graph problem is studied in Ellinas et al. (2000),

where the edges of a graph are covered with directed cycles. The authors show that a bridge-less undirected planar graph[1] can be decomposed into directed cycles where each edge is used exactly twice (once in each direction). They conjecture that this result also applies to any bridge-less graph and give a heuristic for finding directed cycle covers of undirected graphs. In Médard et al. (2002), the authors cover the arcs of a directed graph using an overlay graph[2]. These works do not take into consideration demands, flows, capacities, or costs.

3.5 Conclusions

In this chapter, we formally defined the capacitated survivable network problem, and introduced both hierarchical and integrated frameworks. We reviewed the literature on designing survivable networks, focusing on several protection schemes for designing survivable networks. We also presented our framework that uses directed cycles as failure-flow patterns. In Chapter 4, we study its mixed-integer formulation in detail, and develop facet-defining inequalities for its cut-set and arc-set polyhedra.

[1] A graph is bridge-less if its nodes can not be partitioned into two groups such that only one edge connects them.

[2] A graph superimposed on the original graph.

Chapter 4

Capacitated survivable network design using directed cycles

In this chapter, we study the mixed-integer programming formulation for SDC. Most of these results have been published in Rajan and Atamtürk (2004). In this methodology, sufficient slack is explicitly introduced on the directed cycles of the network when flow routing decisions are made. In case of a failure, flow is rerouted along the slacks reserved on directed cycles.

We focus on a polyhedral approach. In particular, we study the cut-set and arc-set polyhedra for SDC, and describe strong valid inequalities that use the survivability requirements. We present a computational study with a branch-and-cut algorithm for SDC. This algorithm also prices the path and directed cycle variables using a column generation approach.

We first present the hierarchical optimization problem, and then extend it to the integrated optimization framework. The proposed models, and the corresponding pricing problems, are discussed in Section 4.1. In Section 4.2, valid inequalities that explicitly consider the survivability requirements are described. These inequalities are used in a branch-and-cut algorithm to strengthen the linear programming relaxations of the formulations. In Section 4.3, we present computational results with the branch-and-cut algorithm, which also prices the exponential class of variables. We compare the capacity

efficiency of the models and the impact of the valid inequalities in reducing the computation time when used as cutting planes. In Section 4.4, we conclude with a summary and directions for future research.

4.1 Models

In this section, we consider two mixed-integer programming models for designing capacitated survivable networks using directed cycles, see Section 3.4.3 for an introduction. The integrated approach was introduced in Section 3.4.3.1; we discuss the hierarchical approach here for the purpose of comparison.

4.1.1 Hierarchical approach

In the first stage of the hierarchical approach, the capacitated network design problem without survivability requirements (NDP) is solved. In the second stage, an optimal solution to the NDP (a vector of flows and edge capacities) is used as input to a spare capacity assignment model; sufficient slack is reserved on directed cycles so the flow on each arc can be safely rerouted along these cycles. Thus, this scheme is the spare capacity assignment problem for designing survivable networks using directed cycles; we present the formulation here for the sake of completeness.

Let $G = (V, E)$ be an undirected graph with node set V and edge set E. Let F be the set of all ordered pairs (arcs) from E, i.e., $F = \{(ij), (ji) : [ij] \in E\}$. We use (ij) to denote the arc from node i to node j, and $[ij]$ to denote the edge between nodes i and j. Let $G' = (V, F)$ denote the directed graph.

Define the capacity variable $x_{[ij]}$ as the amount of capacity installed on edge $[ij]$. Let $h_{[ij]}$ be the cost of installing unit capacity on edge $[ij] \in E$. Let $\mathcal{C}$ be the set of directed cycles of G'. For directed cycle c, we define the variable z_c to denote the amount of slack reserved on all arcs of cycle c. We use slack to refer to fractional capacity that is reserved to cover no-failure flows; z are modeled as continuous variables. Let α_{ij}^c be 1 if directed cycle c includes arc (ij), and 0 otherwise.

Let g^0_{ij} denote the pre-existing amount of demand routed through arc $(ij) \in F$. We use $w^0_{[ij]}$ to denote the pre-existing capacity on edge $[ij] \in E$.

The hierarchical survivable network design problem using directed cycles (HDC) is formulated as

$$\min \sum_{[ij]\in E} h_{[ij]} x_{[ij]}$$

$$s.t.: \quad \sum_{c\in\mathcal{C}} \alpha^c_{ji} z_c \geq g^0_{ij} \qquad \forall (ij) \in F \tag{4.1}$$

$$g^0_{ij} + \sum_{c\in\mathcal{C}} \alpha^c_{ij} z_c \leq w^0_{[ij]} + x_{[ij]} \qquad \forall (ij) \in F \tag{4.2}$$

$$x_{[ij]} \in \mathbb{Z}_+ \qquad \forall [ij] \in E$$

$$z_c \in \mathbb{R}_+ \qquad \forall c \in \mathcal{C}$$

In HDC, spare capacity on a directed cycle provides coverage for flows in the reverse direction to the arcs on the cycle; see Figure 3.3. Constraints (4.1) ensure that for each arc (ij) the total slack reserved on the directed cycles using the reverse arc (ji) is at least the total flow on (ij). Constraints (4.2) ensure that total capacity installed on edge $[ij]$ is large enough to accommodate the flow routed on arc (ij) as well as the slack reserved for directed cycles containing the arc.

4.1.2 Integrated approach

We use the path formulation of the integrated optimization model for SDC, see Section 3.4.3.1. This model makes flow routing and capacity installation decisions for no-failure and failure cases simultaneously. Thus, the integrated model gives a network with cost that is at most the optimal cost of the hierarchical scheme.

SDC (see Section 3.4.3.1 for the formulation) has only one more constraint (3.21) for each arc than the NDP. This is a big advantage of the model in being able to solve large problem instances.

However, the number of directed cycle variables is exponential in the number of arcs, and all of the variables can not be included in the formulation when solving large in-

stances. Therefore, we develop a column generation method to include the directed cycle variables in the formulation as they are needed when solving its linear programming (LP) relaxation. Furthermore, in a path formulation, the number of path variables is also exponential in the number of arcs. Again, they are handled by a column generation scheme. We next discuss the pricing problem for the directed cycle and path variables.

4.1.2.1 Pricing directed cycle variables

Given an LP relaxation solution $(\overline{y}, \overline{z}, \overline{x})$ to SDC (see Section 3.4.3.1), we search for a directed cycle c in G' that has at least three arcs such that z_c has a negative reduced cost, or prove that no such directed cycle exists. Let (u, v) be the dual variables corresponding to constraints (3.24) and (3.25), respectively.

Each arc (ij) on directed cycle c causes the variable z_c to appear twice in the formulation: in constraint (3.24) for arc (ji) with coefficient -1, and in constraint (3.25) for arc (ij) with coefficient $+1$. Hence, the reduced cost of directed cycle c is $\sum_{ij \in F}(u_{ji} - v_{ij})$. Let $q_{ij} = u_{ji} - v_{ij}$ be the cost of arc $(ij) \in F$. Notice that since $u, v \leq 0$, q is unrestricted in sign.

Ideally, we wish to find a directed cycle with minimum reduced cost. This can be formulated as $\min_{c \in \mathcal{C}}\{\sum_{(ij) \in F}((u_{ji} - v_{ij})\alpha_{ij}^c\}$, where $\mathcal{C}$ is the set of all directed cycles of G' of length greater than two. We refer to the problem of finding the minimum reduced-cost directed cycle, with weights as defined above, as the minimum cost cycle problem. Since $q_{ij} = u_{ji} - v_{ij}$ may be positive or negative, this problem is equivalent to finding the minimum cost simple directed cycle with more than two arcs on a network with unrestricted costs. This problem is known to be $\mathcal{NP}$-hard by reduction from the hamiltonian path problem (HPP), see Garey and Johnson (1979).

To price directed cycles, it is sufficient to identify negative reduced-cost directed cycles, rather than find one with minimum cost.

Definition 4.1 *Directed cycle pricing problem (CPP)*: Given a directed graph $G' = (V, F)$ and a cost function $q : F \mapsto \mathbb{R}$, either find a negative-cost directed cycle in G' with at least three arcs, or conclude that no such directed cycle exists. □

CPP can solved with a variant of the Floyd-Warshall algorithm (Ahuja et al. 1993) that ignores directed cycles with two arcs in $\mathcal{O}(|V|^3)$. A delayed column generation approach that finds negative cost directed cycles with at least three arcs is developed and used to add the cycle variables into the formulation as needed.

4.1.2.2 Pricing path variables

The pricing problems of the flow variables are disjoint for each commodity $k \in K$ and therefore can be solved separately. Given a dual solution (u, v, w) to the LP relaxation of the path formulation of SDC (see Section 3.4.3.1), the reduced cost of a path variable $y_p,\ p \in P_k$ is

$$\sum_{(ij)\in F} (e_{ij}^k - u_{ij} - v_{ij}) d^k \delta_{ij}^p - w^k$$

Since $u, v \leq 0$, $\zeta = \min\{\sum_{(ij)\in F}(e_{ij}^k - u_{ij} - v_{ij})d^k\delta_{ij}^p \ :\ p \in P_k\}$ is an s^k, t^k shortest path problem with non-negative weights, and can be solved efficiently using Dijkstra's algorithm (Ahuja et al. 1993). If $\zeta < w^k$, then the path variable corresponding to an optimal s^k, t^k path has a negative reduced cost, and is added to the restricted formulation.

4.2 Strong valid inequalities

In this section, we describe inequalities that utilize the directed cycle variables for ensuring survivability of the network. For an overview of cutting-plane algorithms, see Section 1.5. Since the integrated model (SDC) produces solutions with lower cost than the hierarchical model HDC, we develop inequalities for SDC.

We study the cut-set and arc-set polyhedra of SDC to develop strong valid inequalities. We develop various classes of inequalities for the cut-set polyhedra, and show that one class is facet-defining for the polyhedron of SDC under mild conditions. We extend these inequalities to k-partition inequalities ($k = 2$ for a cut set).

Before we present these results, we briefly discuss how the arc-set inequalities developed for NDP in Chapter 2 can be extended to the arc-set polyhedra of SDC. This

class of valid inequalities is derived from the the capacity constraints (3.22). Inequalities that completely describe the arc set of NDP, $\text{conv}\{y_{ij} \in \mathbb{R}_+^K, x_{[ij]} \in \mathbb{Z}_+ : \sum_{k\in K} d^k y_{ij}^k \leq x_{[ij]},\ y_{ij}^k \leq 1,\ \forall k \in K\}$, are given in Magnanti et al. (1993). A linear-time separation algorithm for these inequalities is presented in Chapter 2. These inequalities are the residual capacity inequalities, and are of the form

$$\sum_{k\in H} d^k(1 - y_{ij}^k) \geq r(d(H))(\lceil d(H)\rceil - x_{[ij]}), \tag{4.3}$$

for all $H \subseteq K$.

It can be shown that all non-trivial facets of $\text{conv}\{y_{ij} \in \mathbb{R}_+^K, z \in \mathbb{R}_+^{\mathcal{C}}, x_{[ij]} \in \mathbb{Z}_+ : \sum_{k\in K} d^k y_{ij}^k + \sum_{c\in\mathcal{C}} z_c \leq x_{[ij]},\ y_{ij}^k \leq 1,\ \forall k \in K\}$ have zero coefficient for the unbounded continuous variables z_c (Atamtürk 2003b). Therefore, residual capacity inequalities (4.3) are the only class of cutting planes for SDC that can be derived from the arc set.

4.2.1 Example to illustrate partition inequalities

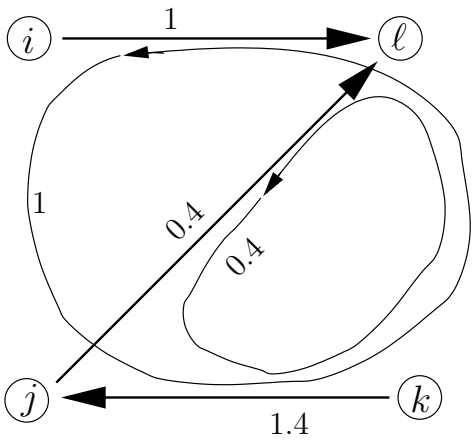
We present a small example that we will use to illustrate various partition inequalities developed in Sections 4.2.2 and 4.2.3. Consider the graph G with $V = \{i, j, k, \ell\}$ and $E = \{[ij], [jk], [k\ell], [i\ell], [j\ell]\}$, see Figure 4.1. In this example, there are 3 commodities: $(s^1, t^1, d^1) = (i, \ell, 1);\ (s^2, t^2, d^2) = (j, \ell, 0.4);\ (s^3, t^3, d^3) = (k, j, 1.4)$.

Suppose that, in the LP solution, all demands are satisfied using the arcs directly connecting the source nodes to the destination nodes, i.e., $y_{i\ell}^1 = 1$, $y_{j\ell}^2 = 0.4$, $y_{kj}^3 = 1.4$; and survivability is ensured using directed cycle $ijk\ell i$ with slack 1 and cycle $jk\ell j$ with slack 0.4. Using z_{ij} to indicate the total slack reserved on arc (ij), and defining z accordingly for other arcs, we have $z_{ij} = 1$, $z_{jk} = 1.4$, $z_{k\ell} = 1.4$, $z_{\ell i} = 1$, and $z_{\ell j} = 0.4$.

4.2.2 2-partition inequalities

We derive the following valid inequalities for SDC from its 2-commodity 2-partition relaxations, see Section 1.6 for a precise definition of cut sets. Let (A, B) be a non-empty partition of the nodes of G. Let $[AB]$ be the edges with one end in A, and the other in B; corresponding to these edges, let AB be the arcs directed from A to B, and BA be the

Figure 4.1: Small example



arcs directed from B to A.

Let d_A denote the total supply in A for nodes B (referred to as commodity A), and d_B denote the total supply in B for nodes A (referred to as commodity B). We assume without loss of generality that $d_A \geq d_B$, since A and B were chosen arbitrarily. Let y^A_{ij} and y^B_{ij} denote flow on arc (ij) for commodities A and B, respectively; z_{ij} the slack reserved for directed cycle variables on arc (ij); and $x_{[ij]}$ the capacity installed on edge $[ij]$. We model the 2-commodity 2-partition relaxation of SDC as

$$y^A(AB) - y^A(BA) = d_A \tag{4.4}$$

$$y^B(BA) - y^B(AB) = d_B \tag{4.5}$$

$$y^A_{ij} + y^B_{ij} + z_{ij} \leq x_{[ij]} \qquad \forall (ij) \in AB \cup BA \tag{4.6}$$

$$0 \leq y^A_{ij} + y^B_{ij} \leq z_{ji} \qquad \forall (ij) \in AB \cup BA \tag{4.7}$$

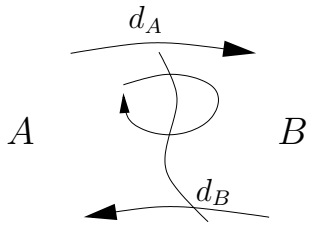
$$z(AB) = z(BA) \tag{4.8}$$

Constraints (4.4) and (4.5) are obtained by aggregating the flow balance constraints across the partition. Constraints (4.6) and (4.7) are the capacity and survivability constraints for the arcs in the partition. Constraint (4.8) states that the total slack reserved for directed cycles across the partition is the same in either direction, since any directed cycle that goes across the partition (using arcs in AB) has to come back across the

partition (using arcs in BA), see Figure 4.2.

Let $\mathcal{F}_2$ denote the convex hull of all $(y \in \mathbb{R}^{4|AB|}, z \in \mathbb{R}^{2|AB|}, x \in \mathbb{Z}^{|AB|})$ satisfying (4.4), (4.5), (4.6), (4.7), and (4.8). All feasible points of SDC are contained in $\mathcal{F}_2$.

Figure 4.2: A 2-partition



Before presenting the general form of the 2-partition inequalities, we motivate and explain the simplest version of the inequalities. The total flow of commodity A on arcs in AB must be at least d_A. Furthermore, total slack reserved for directed cycles on the arcs AB (going from A to B and back) must be sufficient to cover this flow. Consequently, the net capacity across this partition must be at least the sum of these two values, each of which is at least d_A. Finally, since the capacity variables are integral, capacity across the partition is lower bounded by $\lceil 2d_A \rceil$, i.e.,

$$x([AB]) \geq \lceil 2d_A \rceil. \tag{4.9}$$

The following theorem states that this lower bound is not only tight, but also that inequality (4.9) is strong. Let $G'_A = (A, F_A), G'_B = (B, F_B)$ be the sub-graphs defined by the node sets A and B, respectively. Define $\bar{r}_A = r(d_A) = d_A - \lfloor d_A \rfloor$.

Theorem 4.2 For any non-empty 2-partition (A, B) of G with $|[AB]| \geq 3$, the 2-partition inequality (4.9) is facet-defining for the convex hull of feasible solutions of SDC if the two sub-graphs G'_A and G'_B are 2-connected, and either $\bar{r}_A > 1/2$ or $d_A > \max\{d_B, 2\}$.

Proof See Appendix C. □

Defining $\eta_i = \lceil 2d_i \rceil$ and $r_i = r(2d_i) = 2d_i - \lfloor 2d_i \rfloor$ for $i \in \{A, B\}$, we can generalize (4.9) to include flow and directed cycle variables in both directions of the 2-partition.

Theorem 4.3 For $H_1 \subseteq AB$, $H_2 \subseteq BA$, the 2-partition inequalities

$$r_A x([H_1]) + (1 - r_A)x([H_2]) + y^A(AB \setminus H_1) + z(AB \setminus H_1) - y^A(H_2) - y^A(BA \setminus [H_1]) \geq r_A \eta_A \tag{4.10}$$

$$r_B x([H_2]) + (1 - r_B)x([H_1]) + y^B(BA \setminus H_2) + z(BA \setminus H_2) - y^B(H_1) - y^B(AB \setminus [H_2]) \geq r_B \eta_B \tag{4.11}$$

are valid for $\mathcal{F}_2$.

Proof See Appendix C. □

When $H_1 = AB$ and $H_2 = \emptyset$, inequality (4.10) reduces to (4.9). For a fixed 2-partition (A, B), the separation problem of (4.10) is easily solved as follows. Given $(\overline{y}, \overline{z}, \overline{x})$, if $r_A \overline{x}_{[ij]} < \overline{y}^A_{ij} + \overline{z}_{ij} - \overline{y}^A_{ji}$ for $(ij) \in AB$, then we include (ij) in H_1; if $(1 - r_A)\overline{x}_{[ij]} < \overline{y}^A_{ij}$ for $(ij) \in BA$, then we include (ij) in H_2. The separation for inequality (4.11) is similar. Inequalities (4.10) are not necessarily facet-defining for $\mathcal{F}_2$; however, they are always facet-defining for the convex hull of the 1-commodity 2-partition relaxation of SDC, i.e.,

$$y^A(AB) - y^A(BA) = d_A \tag{4.12}$$

$$y^A_{ij} + z_{ij} \leq x_{[ij]} \qquad \forall (ij) \in AB \cup BA \tag{4.13}$$

$$0 \leq y^A_{ij} \leq z_{ji} \qquad \forall (ij) \in AB \cup BA \tag{4.14}$$

$$z(AB) = z(BA) \tag{4.15}$$

under mild conditions. This is stated as Theorem 4.4.

Theorem 4.4 Let $\mathcal{F}_1$ denote the convex hull of points satisfying the 1-commodity 2-partition relaxation; i.e., all $(y \in \mathbb{R}^{2|AB|}, z \in \mathbb{R}^{2|AB|}, x \in \mathbb{Z}^{|AB|})$ satisfying (4.12)-(4.15). The inequality

$$r_A x([H_1]) + y^A(AB \setminus H_1) + z(AB \setminus H_1) - y^A(BA \setminus [H_1]) \geq r_A \eta_A$$

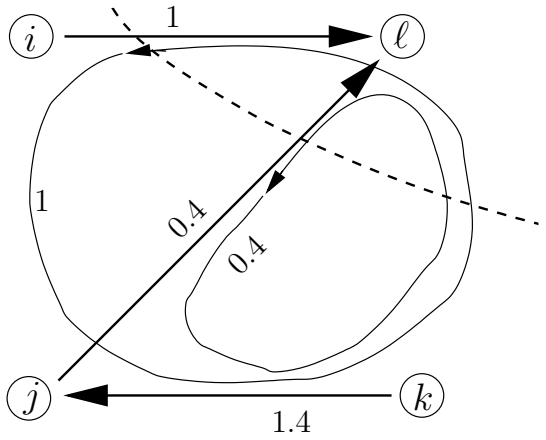
is facet-defining for $\mathcal{F}_1$ if and only if $r_A > 0$ and $H_1 \neq \emptyset$.

Proof See Appendix C. □

We now illustrate these inequalities using the example introduced in Section 4.2.1. Consider the fractional solution illustrated in Figure 4.3: $y^1_{i\ell} = 1,\ y^2_{j\ell} = 0.4,\ y^3_{kj} = 1.4$;

and $z_{ij} = 1$, $z_{jk} = 1.4$, $z_{k\ell} = 1.4$, $z_{\ell i} = 1$, $z_{\ell j} = 0.4$. Installing capacities fractionally gives $x_{[ij]} = 1,\ x_{[i\ell]} = 1,\ x_{[jk]} = 1.4,\ x_{[j\ell]} = 0.4,\ x_{[k\ell]} = 1.4$. This solution satisfies all of the constraints (3.20)-(3.22).

Figure 4.3: Example: 2-partition inequality



Now consider the 2-partition defined as $A = \{i, j, k\}$, $B = \{\ell\}$. For this partition, $d_A = 1.4$ and $d_B = 0$. Thus, we have $\eta_A = 3$, $r_A = 0.8$ and the corresponding inequality (4.9) is

$$x_{[i\ell]} + x_{[j\ell]} + x_{[k\ell]} \geq 3, \tag{4.16}$$

which is violated by the given fractional solution. Suppose we increase $x_{[j\ell]}$ until inequality (4.16) is no longer violated. Now, the allocation to capacity variables is $x_{[ij]} = 1,\ x_{[i\ell]} = 1,\ x_{[jk]} = 1.4,\ x_{[j\ell]} = 0.6,\ x_{[k\ell]} = 1.4$. By enumerating all 2-partitions of the graph, it can be seen that this fractional solution is not violated by any 2-partition inequality (4.9). For the same 2-partition $A = \{i, j, k\}$, $B = \{\ell\}$, inequality (4.10) with $H_1 = \{(i\ell), (k\ell)\}$, $H_2 = \emptyset$ is

$$0.8x_{[i\ell]} + 0.8x_{[k\ell]} + y^1_{j\ell} + y^2_{j\ell} - y^1_{\ell j} - y^2_{\ell j} + z_{j\ell} \geq 2.4 \tag{4.17}$$

This inequality with continuous variables is violated by the new fractional point. Now, suppose we increase $x_{[k\ell]}$ until inequality (4.17) is no longer violated. Now, the capacity variables take the values $x_{[ij]} = 1,\ x_{[i\ell]} = 1,\ x_{[jk]} = 1.4,\ x_{[j\ell]} = 0.6,\ x_{[k\ell]} = 1.5$. We re-

sume this example in Section 4.2.3 after introducing a more general class of inequalities.

4.2.3 3-partition inequalities

In this section we show how to generalize the 2-partition inequalities for 3-partitions of the graph. The ideas presented here can be extended to k-partitions for $k > 3$ as well.

4.2.3.1 Mixed-integer inequalities

Consider a non-empty 3-partition (A, B, C) of the nodes of G. As before, AB is defined as the arcs directed from A to B; the other arc and edge sets are defined in the same way. Further, we divide each set of edges into two groups; e.g., $[AB]$ into $[AB_1]$ and $[AB_2]$. For each proper subset U of $\{A, B, C\}$, we again let d_U denote the total supply of U, i.e., $d_U = \{\sum_k d_k \,:\, s^k \in U,\, t^k \in V \backslash U\}$. As before, $\eta_U = \lceil 2d_U \rceil$ and $r_U = r(d_U) = 2d_U - \lfloor 2d_U \rfloor$.

Now, each of the six proper subsets of $\{A, B, C\}$ results in a non-empty 2-partition $(U, V \setminus U)$ of the nodes of G. For each of them, we obtain a subclass of intermediate 2-partition inequalities as follows. Choose $H_1 = \{(ij) \,:\, i \in U, j \in V \setminus U, [ij] \in [AB_1] \cup [BC_1] \cup [AC_1]\}$, and $H_2 = \emptyset$. Below we present two such inequalities, corresponding to ($U = \{A\}, H_1 = AB_1 \cup AC_1$) and ($U = \{B\}, H_1 = BA_1 \cup BC_1$), respectively.

$$\begin{aligned} r_A x([AB_1] \cup [AC_1]) + y^A(AB_2 \cup AC_2) \\ + z(AB_2 \cup AC_2) - y^A(BA_2 \cup CA_2) \geq r_A \eta_A \end{aligned} \tag{4.18}$$

$$\begin{aligned} r_B x([BA_1] \cup [BC_1]) + y^B(BA_2 \cup BC_2) \\ + z(BA_2 \cup BC_2) - y^B(AB_2 \cup CB_2) \geq r_B \eta_B \end{aligned} \tag{4.19}$$

Define $\eta = \lceil \frac{r_A \eta_A + r_B \eta_B}{r_A + r_B} \rceil$ and $\bar{r} = \frac{r_A \eta_A + r_B \eta_B}{r_A + r_B} - \lfloor \frac{r_A \eta_A + r_B \eta_B}{r_A + r_B} \rfloor$. Adding (4.18) and (4.19), and applying mixed-integer rounding (Section 1.5.2) to the resulting inequality, we obtain the 3-partition inequality

$$\begin{aligned} \bar{r}(r_A + r_B)\, x([AB_1]) + r_A x([AC_1]) + r_B x([BC_1]) \\ + y^A(AB_2 \cup AC_2) + y^B(BA_2 \cup BC_2) - y^A(BA_2 \cup CA_2) - y^B(AB_2 \cup CB_2) \\ + z(AB_2 \cup BA_2 \cup AC_2 \cup BC_2) \;\geq\; \bar{r}\, \eta(r_A + r_B). \end{aligned} \tag{4.20}$$

There are exponentially many inequalities of the form (4.20) depending on how $[AB]$, $[BC]$, and $[AC]$ are partitioned into groups. Nevertheless, for a fixed 3-partition (A, B, C), the separation problem for (4.20) is simple. Given $(\overline{y}, \overline{x}, \overline{z})$, $(ij) \in AB$ is included in AB_1 if $\bar{r}(r_A + r_B)\overline{x}_{[ij]} < \overline{y}^A_{ij} - \overline{y}^A_{ji} - \overline{y}^B_{ij} + \overline{y}^B_{ji} + \overline{z}_{ij} + \overline{z}_{ji}$; and in AB_2 otherwise. On the other hand, $(ij) \in BC$ is included in BC_1 if $r_B\overline{x}_{[ij]} < \overline{y}^B_{ij} - \overline{y}^B_{ji} + \overline{z}_{ij}$; and in BC_2 otherwise. Finally, $(ij) \in AC$ is included in AC_1 if $r_A\overline{x}_{[ij]} < \overline{y}^A_{ij} - \overline{y}^A_{ji} + \overline{z}_{ij}$, in AC_2 otherwise.

Since there are six proper subsets of $\{A, B, C\}$ which can be used to form the intermediate 2-partition inequalities, we can describe fourteen more 3-partition inequalities similar to (4.20) for the 3-partition (A, B, C).

4.2.3.2 Pure-integer inequalities

We present another class of 3-partition inequalities written in terms of the integral capacity variables; that is, we let $[AB_1] = [AB]$, $[AC_1] = [AC]$ and $[BC_1] = [BC]$. Defining H_1 and H_2 as before, we have the three 2-partition inequalities with only capacity variables

$$x([AB]) + x([AC]) \geq \lceil 2\max\{d_A, d_{BC}\}\rceil,$$
$$x([AB]) + x([BC]) \geq \lceil 2\max\{d_B, d_{AC}\}\rceil,$$
$$x([AC]) + x([BC]) \geq \lceil 2\max\{d_C, d_{AB}\}\rceil.$$

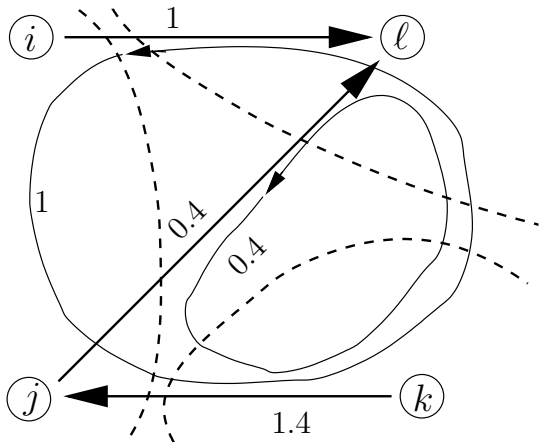
Let $\theta = \lceil 2\max\{d_A, d_{BC}\}\rceil + \lceil 2\max\{d_B, d_{AC}\}\rceil + \lceil 2\max\{d_C, d_{AB}\}\rceil$. By adding these three inequalities, dividing the sum by two and rounding up the right hand side, we obtain the 3-partition inequality

$$x([AB]) + x([AC]) + x([BC]) \geq \lceil \frac{\theta}{2} \rceil \tag{4.21}$$

We now illustrate inequalities (4.21) using the example in Section 4.2.1. Consider the fractional solution $y^1_{i\ell} = 1,\ y^2_{j\ell} = 0.4,\ y^3_{kj} = 1.4$; $z_{ij} = 1,\ z_{jk} = 1.4,\ z_{k\ell} = 1.4,\ z_{\ell i} = 1,\ z_{\ell j} = 0.4$; and $x_{[ij]} = 1,\ x_{[i\ell]} = 1,\ x_{[jk]} = 1.8,\ x_{[j\ell]} = 0.6,\ x_{[k\ell]} = 1.5$; see Figure 4.4.

By enumerating among all subsets $H_1,\ H_2$ for all the 2-partitions, it can be seen that no 2-partition inequality violates this solution. However, for 3-partition $A = \{i, j\}$, $B = \{k\}$, $C = \{\ell\}$, we have $d_A = 1.4$, $d_B = 1.4$, $d_C = 0$, $d_{AB} = 1.4$, $d_{AC} = 0$, $d_{BC} = 1.4$.

Figure 4.4: Example: 3-partition inequality



Correspondingly, $\theta = 9$ and the 3-partition inequality (4.21)

$$x_{[i\ell]} + x_{[j\ell]} + x_{[jk]} + x_{[k\ell]} \geq \lceil 9/2 \rceil$$

is violated by the fractional point.

4.3 Computational results

In Section 4.3 we present computational results with a branch-and-cut generation algorithm for solving the models introduced in Section 4.1. This algorithm also prices the path and directed cycle variables in a column generation scheme. We compare the capacity requirements of the models and test the effectiveness of the valid inequalities given in Section 4.2 in reducing the solution times when used as cutting planes.

The algorithm is implemented using the callable library of CPLEX Version 8.1 Beta. All experiments are done on a 2GHz Intel Pentium 4 Linux workstation with 1GB RAM. The data set consists of random instances of networks with number of nodes $(|V|)$ ranging from 5 to 12. The instances have 75% edge density and 50% demand density, i.e., each of the $|V|(|V|-1)/2$ edges (and both the directed arcs corresponding to that edge) exists with probability $.75$, and each of the $|V|(|V|-1)$ demand pairs exists with probability 0.5. The demand values are chosen from $0.1\times$ IntUni$[1, 20]$.

The directed cycle variables with negative reduced cost are generated as explained in Section 4.1.2.1. Since CPLEX does not allow addition of variables to the formulation in the branch-and-bound tree, directed cycle variables are generated only at the root node of the tree. Since HDC is the hierarchical counterpart of SDC, directed cycle variables are priced even in HDC. In Section 5.4, we present a study on the effect of column generation of directed p-cycles as failure-flow pattern variables in the context of SDP.

In the first experiment we compare NDP with and the hierarchical and integrated models of survivable network design models using directed cycles, HDC and SDC, respectively. In Table 4.1 we present the time taken to solve the three models and the total capacity installed with each model. If a problem is solved within one hour, then we report the objective value of the solution and the elapsed CPU time in seconds (time); otherwise, we report the objective value of the best known feasible solution (zub), and the gap (endgap) between this solution and the best lower bound at termination (zlb) as a percentage of the best lower bound, i.e., endgap $= 100 \times (zub - zlb)/zlb$.

Table 4.1: Hierarchical and integrated approaches

size	time (endgap)			best feasible solution (zub)		
$\|V\|$	NDP	HDC	SDC	NDP	HDC	SDC
5	0.03	0.01	0.06	50.5	110.3	103.5
6	0.16	0.02	0.23	129.5	251.7	235.6
7	0.27	0.03	2.29	103.4	222.3	189.4
8	2.88	0.04	14.0	146.6	286.2	259.6
9	517.1	0.15	20.9	172.7	364.9	311.6
10	(1.6)	0.21	(0.3)	235.8	457.6	432.4
11	1216	0.16	(0.8)	289.3	562.7	527.1
12	(3.1)	0.30	(2.7)	326.0	641.2	592.1

Comparing the capacity requirements of the models, HDC needs about 100% more capacity than the no-survivability model NDP, whereas SDC requires on the average 80% more capacity. Comparing the models in terms of ease of solvability, we see that HDC takes the least amount of time. However, we first need to solve NDP before we can use its solution as an input to HDC. Interestingly, SDC is not any harder to solve than NDP, see Table 4.1. This is important, since SDC incorporates survivability. Later in this

section, we will also see that SDC scales well with increasing network size when solved in a branch-and-cut framework using the cutting planes proposed in Section 4.2.

In the next experiment, we investigate the effect of the valid inequalities described in Section 4.2 in reducing the number of branch-and-bound nodes and solution times, when used as cutting planes to improve the linear programming relaxations. Residual capacity inequalities are added using the linear-time separation method given in Atamtürk and Rajan (2002) for each arc at the root node of the branch-and-bound tree. We enumerate all 2-partitions with at most three nodes in one partition and all 3-partitions with at most two nodes in two of the partitions; and add the corresponding inequalities with only capacity variables whenever they are violated in the tree.

In Table 4.2 we report the number of cuts added (cuts added), improvement of the integrality gap at the root node (root improvement), the number of branch-and-bound nodes (b&b nodes), and the solution times (time) or gap at termination (endgap), for runs with and without the polyhedral cuts. The default CPLEX cuts are added in both runs. The results for experiments using only the CPLEX cuts are reported under heading (1) and results for experiments using both CPLEX and polyhedral cuts are reported under heading (2). All runs have a time limit of ten hours.

Table 4.2: Effect of cutting planes

Size	cuts added		root improvement		b&b nodes		time (endgap)	
$\|V\|$	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)
5	24	20	76.5	77.1	47	36	0.06	0.06
6	22	24	30.6	100	279	0	0.23	0.01
7	34	35	24.0	72.0	2298	203	2.29	0.35
8	46	58	34.6	78.7	7060	547	14.0	1.76
9	50	47	38.1	77.3	7122	196	20.9	1.33
10	67	90	31.4	74.6	2009857	88145	4876	267
11	116	164	41.2	67.6	2177271	65744	12296	552
12	142	188	22.3	50.0	2111101	1762640	(2.2)	(1.2)

We see in Table 4.2 that adding polyhedral cuts more than doubles the integrality gap improvement at the root node. This leads to significant reduction in the number of nodes and solution time. In general, the improvement in the solution time is more than an

order of magnitude. Based on these computations, we conclude that the cutting planes developed in Section 4.2 improve the performance of the algorithm significantly.

In the final experiment, we tested how well the branch-and-cut generation algorithm scales for large instances. This is significant since we are pricing an exponential class of variables using a column generation scheme. For this experiment, we ran the algorithm for instances ranging from 20 to 70 nodes for ten hours. In Table 4.3, we report the number of directed cycles added (cycles added), the number of cuts added (cuts added), percentage improvement at the root node (root improvement), total number of nodes in the branch-and-bound tree (b&b nodes) and the gap at termination (endgap).

Table 4.3: Computations on large instances

size	cycles added	cuts added	root improvement	b&b nodes	(endgap)
20	458	225	42.9	498668	(0.58)
30	614	248	33.3	315777	(0.44)
40	842	219	34.3	123080	(0.21)
50	1450	279	26.4	27791	(0.26)
60	1248	276	25.9	13328	(0.18)
70	1546	292	20.8	1100	(0.21)

Although none of the instances are solved to optimality, the gap at termination was less than 1% for all instances. To some extent, the drop in the gap at termination for larger instances can be attributed to the fact that the LP relaxations of the formulation seem to strengthen with increasing problem size. This fact reiterates the scalability of our model (and methodology). At the same time, the effect of our cutting planes can not be discounted, even as problem size increases. These experiments suggest that the proposed methodology is a computationally effective way for designing capacitated survivable networks.

4.4 Conclusions

We introduced a new methodology for designing capacitated survivable networks that explicitly reserves slack on directed cycles. We first presented the hierarchical optimization model, and then extended it to an integrated model that makes all routing and capacity

decisions simultaneously. Even though the number of variables in the models is exponential in the size of the input graph, they were generated in polynomial time in a column generation framework.

We also developed strong polyhedral cutting planes for the integrated SDC model. Finally, we compared the models and the effectiveness of the cutting planes computationally using a branch-and-cut generation algorithm. The integrated approach yielded survivable networks with about 10% less capacity requirements than the hierarchical scheme, and the polyhedral cuts reduced the solution times by an order of magnitude.

Our experiments suggest that the proposed methodology is a computationally effective way for designing capacitated survivable networks. At the same time, we can reduce the capacity requirements further by considering other failure-flow patterns; for instance, directed p-cycles to route disrupted flows when their chords fail as well. In Chapter 5, we present the mixed-integer programming formulation for this framework, and study the corresponding pricing sub-problem. Preliminary computations indicate that this framework results in survivable networks that are almost as capacity-efficient as global rerouting. The polyhedral structure when using this more complicated failure-flow pattern is a direction of future research.

Chapter 5

Capacitated survivable network design using directed p-cycles

In this chapter, we present the mixed-integer programming model for designing survivable networks using directed p-cycles (SDP). We compare this method with network design problem without survivability requirements (NDP), global rerouting (GNP), spare capacity assignment using undirected p-cycles (HUP) and survivable network design using directed cycles (SDC).

We compare all the frameworks for designing survivable networks introduced in this chapter with the NDP (Section 2.1) since we wish to measure the amount of extra capacity required to enforce survivability. We include GNP (Section 3.3.1) since this is the best one can do in terms of capacity efficiency. We incorporate HUP (Section 3.4.2) to compare our framework with the work of Grover and Stamatelakis (1998), in which the authors use undirected p-cycles as failure-flow patterns in a hierarchical framework. We compare with SDC (Section 3.4.3.1) to measure improvements obtained by using directed p-cycles over directed cycles. In Section 5.4, we present results of computational experiments that compare the capacity efficiency and ease of solvability of these models.

Most of the work in this chapter has been published in Rajan and Atamtürk (2002). In contrast to the failure scenario models, the hybrid models (HUP, SDC and SDP) have almost the same number of constraints as NDP, which makes them effective for large

instances. Even though the complexity of pricing the exponentially many directed p-cycle variables in SDP is $\mathcal{NP}$-hard (as opposed to directed cycle variables in SDC), in our computational experiments, we are able to produce capacity-efficient survivable networks with dense graphs up to 70 nodes.

5.1 Introduction

In Section 5.2, we introduce the new mixed-integer programming model for designing survivable networks using directed p-cycles (SDP). This model considers routing of no-failure flows and failure flows simultaneously by installing slacks on directed p-cycles of the network so as to ensure survivability in the case of edge failures. Thus, this model is very similar to SDC, except that it uses directed p-cycles as opposed to directed cycles, as failure flow patterns. As in SDC, since only disrupted flow is rerouted, reconfiguration of the network can be done quickly.

SDP delivers survivable networks with capacity efficiency very close to GNP, as we will see in Section 5.4. The number of the constraints of the formulation is almost the same as NDP, which makes the model effective for large instances. The number of the variables is exponential in the number of edges of the graph; however, the variables are treated implicitly by column generation. In Section 5.3, we show that the pricing complexity of the variables is $\mathcal{NP}$-hard, and discuss a polynomial-time heuristic. Using this heuristic, we solve SDP by column generation and report successful computational experiments with dense graphs up to 70 nodes. We conclude in Section 5.5 by summarizing this chapter, and motivate directions for future research in Section 5.6.

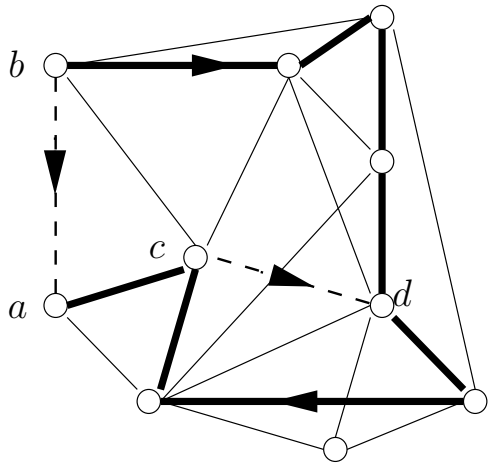
5.2 Survivable network design using directed p-cycles

In this section, we present a mixed-integer programming model for SDP. Rather than using undirected p-cycles, as in the case of HUP (Grover and Stamatelakis 1998), we utilize directed p-cycles to introduce sufficient slack on top of the no-failure flows. As opposed to HUP, which is hierarchical, this scheme is integrated, in the sense that it

makes all decisions simultaneously.

A directed p-cycle provides coverage for flows in the reverse direction for the arc on the p-cycle, see arc (ba) in Figure 5.1. A directed p-cycle provides only one recovery path for the flow on a chord, see arc (cd) in Figure 5.1. However, slack introduced for the directed p-cycle in the reverse direction also provides coverage for the flow on arc (cd).

Figure 5.1: A directed p-cycle



Let $G = (V, E)$ be an undirected graph with node set V and edge set E; and let F be the set of all ordered pairs (arcs) from E, i.e., $F = \{(ij), (ji) : [ij] \in E\}$. We use (ij) to denote the arc from node i to node j, and $[ij]$ to denote the edge between nodes i and j. We use $G' = (V, F)$ to denote the directed graph. Let K be the set of commodities. Let $\{(s^k, t^k, d^k)\}_{k \in K}$ be the commodity triples of source and destination nodes s^k and t^k, and d^k be the supply at s^k for t^k, $k \in K$. Let b_i^k be the supply of commodity k at node i, i.e., $b_{s^k}^k = d^k$, $b_{t^k}^k = -d^k$, and $b_i^k = 0$ for $i \in V \setminus \{s^k, t^k\}$.

We define variable y_{ij}^k as the fraction of commodity k routed through arc $(ij) \in F$. Let e_{ij}^k be the cost associated with routing each unit of commodity $k \in K$. We define the capacity variable $x_{[ij]}$ as the amount of capacity installed on edge $[ij]$; and let $h_{[ij]}$ be the cost of installing unit capacity on edge $[ij] \in E$. Let $\mathcal{C}$ be the set of directed p-cycles of G'. For $c \in \mathcal{C}$, we define the variable z_c to denote the amount of slack reserved on directed p-cycle c. We use slack to refer to fractional capacity that is reserved to cover no-failure

flows; z are modeled as continuous variables. Let α_{ij}^c be 1 if directed p-cycle c includes arc (ij), and 0 otherwise. Let $\rho_{[ij]}^c$ be 1 if edge $[ij]$ is a chord to directed p-cycle c, and 0 otherwise. We use $w_{[ij]}^0$ to denote the pre-existing capacity on edge $[ij] \in E$. SDP can now be formulated as follows.

$$\min \sum_{(ij)\in F} \sum_{k\in K} d^k e_{ij}^k y_{ij}^k + \sum_{[ij]\in E} h_{[ij]} x_{[ij]}$$

$$\sum_{j:(ij)\in F} d^k y_{ij}^k - \sum_{j:(ji)\in F} d^k y_{ji}^k = b_i^k \qquad \forall i \in V,\ \forall k \in K \tag{5.1}$$

$$\sum_{k\in K} d^k y_{ij}^k - \sum_{c\in\mathcal{C}} \rho_{[ij]}^c z_c - \sum_{c\in\mathcal{C}} \alpha_{ji}^c z_c \le 0 \qquad \forall (ij) \in F \tag{5.2}$$

$$\sum_{k\in K} d^k y_{ij}^k + \sum_{c\in\mathcal{C}} \alpha_{ij}^c z_c \le w_{[ij]}^0 + x_{[ij]} \qquad \forall (ij) \in F \tag{5.3}$$

$$x_{[ij]} \in \mathbb{Z}_+ \qquad \forall [ij] \in E$$

$$z_c \in \mathbb{R}_+ \qquad \forall c \in \mathcal{C}$$

$$y_{ij}^k \in \mathbb{R}_+ \qquad \forall (ij) \in F,\ \forall k \in K.$$

Constraints (5.2) ensure that for each arc (ij), the total flow is no more than the total slack installed on the directed p-cycles which include arc (ji) or chord $[ij]$. Constraints (5.3) ensure that capacity installed on edge $[ij]$ is large enough to accommodate the flow routed on arc (ij) as well as the slack introduced on the arc by directed p-cycles.

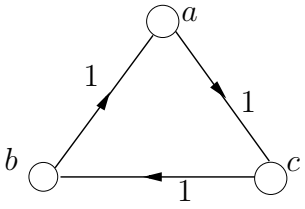
Interestingly, in contrast to GNP, SDP requires only one constraint more for each arc than NDP. We assume that $\mathcal{C}$ is a restricted set of directed p-cycles from graph $G' = (V, F)$. With this restricted set, we denote the formulation as SDPb. However, the number of cycles in a graph, and hence the number of directed p-cycle variables in the formulation is exponential in the number of the arcs. In Section 5.3, we discuss how all the directed p-cycle variables can be handled.

The example in Figure 5.2 emphasizes the importance of determining slacks on directed p-cycles, rather than covering working edge capacities with undirected p-cycles as in Grover and Stamatelakis (1998). Here, the flow on each arc (ac), (cb), and (ba) equals 1. Since installed capacity on an edge allows flow in both directions up to capacity. Rout-

ing one unit using a directed p-cycle in the counter-clockwise direction (from a to b to c) covers all of the flow on the network and requires no additional capacity. Thus, using SDP, the network in Figure 5.2 is survivable with a total capacity of 3. On the other hand, covering installed capacities with undirected p-cycles requires installing one additional unit on each edge, thus doubling the installed capacity.

Routing slacks on directed p-cycles to cover failure flows will lead to lower capacity than covering no-failure capacity, even without the assumption that installed capacity on an edge allows flow in both directions up to capacity.

Figure 5.2: A small survivable network



Consequently, SDP produces survivable networks that are more capacity-efficient than HUP for two reasons: SDP determines slack on directed p-cycles to cover flows rather than using undirected p-cycles to cover working edge capacities; and SDPb considers the routing of no-failure flows when determining excess capacity installation in an integrated framework. Indeed our computational experiments indicate that the capacity efficiency delivered by SDP is very close to GNP, see Section 5.4.

5.3 A column generation approach

Since there exist exponentially many directed p-cycle variables, not all variables can be included in the model when solving large instances. Selecting a small subset of the variables *a priori* and solving the model with these variables can result in suboptimal solutions, as we show in Section 5.4. Therefore, we develop a column generation approach and introduce the directed p-cycle variables into SDPb as they are needed (see Section 1.4). We refer to this formulation, in which all directed p-cycles have been added

as needed, as SDP. To solve larger instances efficiently, we reformulate SDP using path-based flow variables, rather than arc-based flow variables. This reduces the number of constraints, but introduces exponentially many path variables; which can also be generated, as needed, via column generation.

In path-based formulations, y_p indicates the fraction of commodity routed on path p, and P_k denotes the set of paths from s^k to t^k. For any path p, $\delta_{ij}^p = 1$ if it includes arc (ij), and 0 otherwise. We reformulate SDP using a path formulation as follows.

$$\min \quad \sum_{(ij)\in F} \sum_{k\in K} \sum_{p\in P_k} d^k \delta_{ij}^p e_{ij}^k y_p + \sum_{[ij]\in E} h_{[ij]} x_{[ij]}$$

$$(w^k) \qquad \sum_{p\in P_k} y_p = 1 \qquad \forall k \in K \tag{5.4}$$

$$(u_{ij}) \qquad \sum_{k\in K} \sum_{p\in P_k} d^k \delta_{ij}^p y_p - \sum_{c\in\mathcal{C}} \rho_{[ij]}^c z_c - \sum_{c\in\mathcal{C}} \alpha_{ji}^c z_c \le 0 \qquad \forall (ij) \in F \tag{5.5}$$

$$(v_{ij}) \qquad \sum_{k\in K} \sum_{p\in P_k} d^k \delta_{ij}^p y_p + \sum_{c\in\mathcal{C}} \alpha_{ij}^c z_c \le w_{[ij]}^0 + x_{[ij]} \qquad \forall (ij) \in F \tag{5.6}$$

$$y_p \in \mathbb{R}_+ \qquad \forall p \in P_k,\ \forall k \in K$$

$$x_{[ij]} \in \mathbb{Z}_+ \qquad \forall [ij] \in E$$

$$z_c \in \mathbb{R}_+ \qquad \forall c \in \mathcal{C},$$

where u, v, w are the corresponding dual variables of the LP relaxation of SDP. Constraints (5.4) ensure the demand for each commodity is satisfied for the no-failure scenario. Constraints (5.5) ensure that sufficient slack is allocated to directed p-cycles to cover all no-failure flow on each arc. Constraints (5.6) ensure that, for each arc, sufficient capacity is installed on the edge for the slack introduced on the directed p-cycle and the no-failure flow on the arc.

This formulation has $2|F| + |K|$ constraints, as compared with the $2|F| + |K||V|$ constraints in the arc formulation. We can now solve much larger instances because of the smaller number of constraints. However, the reduction in number of constraints comes at a price: We now have an exponential number of flow variables. In Sections 5.3.1 and 5.3.2, we study the pricing problems for directed p-cycle and flow variables, respectively.

5.3.1 Pricing of directed p-cycle variables

Our SDP formulation needs to include all directed p-cycles that are used in the optimal solution. Since we have no way of knowing which directed p-cycles they are *a priori*, we need to price out any directed p-cycles with negative reduced cost, when solving the LP relaxation. Given an LP-relaxation solution to SDP $(\overline{y}, \overline{z}, \overline{x})$, we are interested in finding a directed p-cycle variable z_c with negative reduced cost, or prove that no such directed p-cycle exists. Variable z_c appears four times in the formulation, for each arc (ij) on directed p-cycle c: in constraint (5.5) for arc (ji) with coefficient $-\alpha^c_{ij}$; in constraint (5.5) for arcs (ij) and (ji) with coefficient $-\rho^c_{[ij]}$; and in constraint (5.6) with coefficient α^c_{ij}. Therefore, for an optimal dual solution (u, v, w), the reduced cost of any variable z_c is

$$\sum_{(ij)\in F} ((u_{ji} - v_{ij})\alpha^c_{ij} + (u_{ij} + u_{ji})\rho^c_{[ij]})$$

A negative reduced cost z_c can be identified by finding a negative weight directed cycle that has at least three arcs on $G' = (V, F)$, where weight of arc (ij) on the directed cycle is $f^a_{ij} = u_{ji} - v_{ij}$ and the weight of edge $[ij]$ that is a chord of the directed cycle is $f^h_{ij} = u_{ij} + u_{ji}$. Since $u, v \leq 0$, f^a_{ij} is unrestricted in sign, but $f^h \leq 0$.

Finding a minimum reduced-cost directed p-cycle problem on the directed network $G' = (V, F)$ is trivially $\mathcal{NP}$-hard, since the special case with $f^h = 0$ for all edges corresponds to the problem of finding a minimum cost directed cycle on a network that may have negative arc costs, see Section 4.1.2.1. Fortunately, we are interested in finding any negative reduced-cost directed p-cycle, not necessarily the one with the minimum reduced cost.

Definition 5.1 *Pricing problem of directed p-cycles (PPP)*: Given arc weights $f^a \in \mathbb{R}^{|F|}$ and chord weights $f^h \in \mathbb{R}^{|F|}$, either find a negative cost directed p-cycle with at least three arcs, or conclude that no such directed p-cycle exists. □

Theorem 5.2 The pricing problem of directed p-cycles (PPP) is $\mathcal{NP}$-hard.

Proof See Appendix D. □

Next, we discuss some efficient heuristics for pricing these directed p-cycle variables.

When all chord weights f^h of directed p-cycles are zero, PPP reduces to finding a negative weight (f^a) directed cycle with at least three arcs, and is the same as CPP, see Section 4.1.2.1. This can be accomplished in $\mathcal{O}(|V|^3)$ with a simple modification to the Floyd-Warshall algorithm for finding shortest paths in a directed graph. Thus, the presence of weights for the chords (f^h) complicates the problem significantly.

However, we can use CPP as a basis for designing heuristics to PPP. Since chord weights f^h are non-positive, if we find a negative weight cycle c by assuming that f^h is zero, then directed p-cycle c has negative reduced cost. Therefore, to find negative reduced cost directed p-cycle variables, we solve CPP using f^a and add the corresponding directed p-cycle variables to the restricted formulation of SDP. When we exhaust all such negative weight directed p-cycles, there can still be other negative reduced-cost directed p-cycle variables with $f^a > 0$.

The longer the directed p-cycle is, greater the number of chords it has, especially for dense graphs. Thus, longer directed p-cycles have a higher tendency of having a negative weight since $f^h \leq 0$. We can potentially obtain good p-cycles by changing the weights f^a in such a way that we get longer cycles when we solve the polynomial-time negative weight directed cycle problem. One possible way of accomplishing this is by reducing all arc weights f^a by a certain constant so that the longer cycles will be in favor. We incorporate this idea in the column generation algorithm for pricing directed p-cycle variables that do not correspond to negative weight cycles.

However, we still price directed p-cycle variables only at the root node of the branch-and-bound tree. It is not necessary that the integer optimal solution to SDP will include only those directed p-cycles in the LP relaxation solution. We need to develop a branch-and-price algorithm that solves the pricing problem for directed p-cycles at each node in the branch-and-bound search tree.

5.3.2 Pricing of flow variables

We need to price out any path variable with a negative reduced cost, given an LP relaxation solution $(\overline{y}, \overline{z}, \overline{x})$ to SDP. The pricing problems of the path-based flow variables

are disjoint for each commodity $k \in K$ and therefore can be solved separately. This is exactly the same as the pricing problem for the flow variables of SDC in Section 4.1.2.2; we repeat here for completeness.

The path variable y_p appears once in constraint (5.4) with coefficient 1. Also, for each arc (ij), the variable y_p appears twice in the formulation; with coefficient δ_{ij}^p in constraint (5.5) for arc (ij) and in constraint (5.6) for arc (ij). Since u, v, w are the dual variables for constraints (5.5), (5.6) and (5.4), respectively, the reduced cost of a path variable $y_p,\ p \in P_k$ is

$$\sum_{(ij)\in F} (e_{ij}^k - u_{ij} - v_{ij}) d^k \delta_{ij}^p - w^k$$

Since $u, v \leq 0$, $\zeta = \min\{\sum_{(ij)\in F}(e_{ij}^k - u_{ij} - v_{ij}) d^k \delta_{ij}^p \ :\ p \in P_k\}$ is an $(s^k - t^k)$ shortest path problem with non-negative weights, and can be solved efficiently using Dijkstra's algorithm (Ahuja et al. 1993). If $\zeta < w^k$, then the path variable corresponding to the optimal $(s^k - t^k)$ path has a negative reduced cost, and is added to the restricted formulation.

5.4 Computational results

We present our computational experiments comparing the capacity efficiency and ease of solvability of the models NDP, GNP, HUP, SDC, SDPb, and SDP. Our goal is to determine whether the new method of designing survivable networks using directed p-cycles (SDP) is a viable alternative for designing survivable telecommunication networks.

While GNP produces the most capacity-efficient survivable networks, the size of the formulation, which essentially carries a copy of the NDP for each failure scenario, makes it unfit for tackling problems unless they are very small. Nevertheless, in our experiments we solve the GNP model for small instances to find the lowest capacity requirement.

For these experiments we create small instances of randomly generated graphs (of size $|V|$ from 5 to 15) with 75% edge density and 50% demand density; the same instances as in Section 4.3. These networks are thus quite dense ($\mathcal{O}(|V|^2)$ arcs) and are much more difficult that any real-world problem instances.

For a fair comparison of the capacity efficiency of HUP and SDPb, we solve these models with the same set of cycles that are selected *a priori*. For SDC, SDPb, and HUP, we chose the restricted set of cycle variables as follows. First, we calculate the minimum cost spanning tree (T - using edge weights proportional to h_e). Then, for every edge $[ij]$ on the tree, we find the minimum cost cycle that uses edge $[ij]$ and exactly one edge $\notin T$. We add these undirected cycles as undirected p-cycles to HUP. We add directed p-cycle variables (both directions) corresponding to these undirected cycles *a priori* to the formulation of SDC and SDPb.

In Table 5.1, we report the number of constraints and variables in the five formulations for the graph with 14 nodes; we use the arc formulations. In this comparison, we see that the formulations of NDP, SDC and SDPb are about the same size. By choice of the restricted set of cycles, SDC and SDPb have exactly the same number of constraints. Thus, using failure-flow patterns to design survivable networks allows us to work with formulations of roughly the same size as the NDP. On the other hand, the formulation of GNP is more than an order of magnitude bigger than the rest of the models. In fact, the formulation for GNP with $|V| = 5$ is larger than the formulations for NDP, SDC and SDPb with $|V| = 15$; see Table 5.1.

Table 5.1: Formulation sizes

$\|V\| = 14$	NDP	GNP	HUP	SDC	SDPb
constraints	342	27352	133	488	488
variables	2117	151329	105	2173	2173

We solve all of the models with CPLEX7.5 MIP solver on an Intel Pentium4 2GHz Linux workstation with 1GB RAM with one hour CPU time limit. We do all of the computations using a best-bound node selection strategy in the branch-and-bound search tree, and recover the best feasible solution if we do not obtain the optimal solution at the time limit. In Table 5.2, for each instance (of size $|V|$), we report the solution times in CPU seconds (time) and the best MIP solutions found (best feasible soln) for the five models. Now, we no longer use a restricted set of directed cycle variables for SDC. We report the optimality gap (endgap) of the best MIP solution in place of time if the branch-and-bound

computation does not finish in one hour time limit.

Table 5.2: Solution times and capacity efficiency

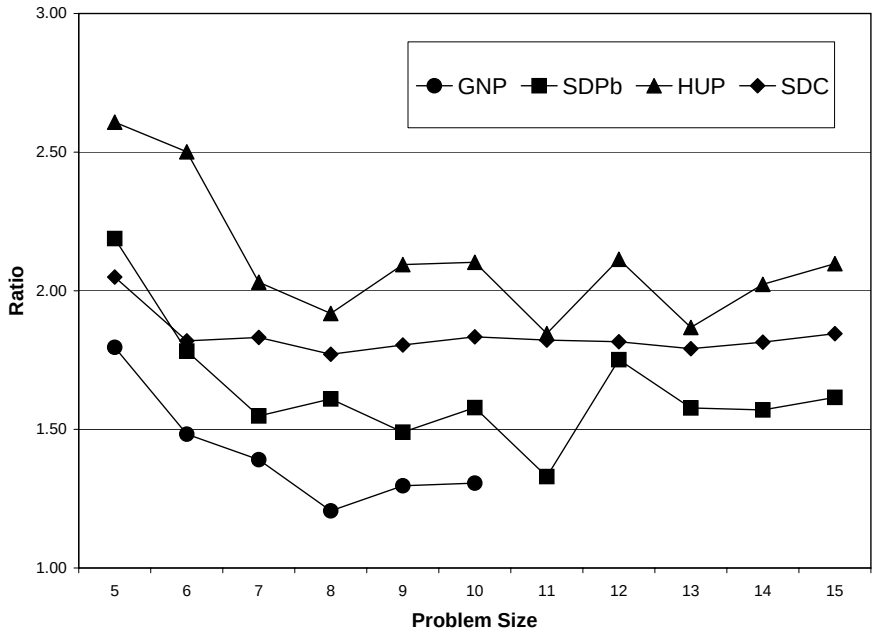
$\|V\|$	time (endgap)					best feasible soln				
	NDP	GNP	HUP	SDC	SDPb	NDP	GNP	HUP	SDC	SDPb
5	0.03	0.57	0.00	0.06	0.04	50.5	90.7	131.7	103.5	110.5
6	0.16	1.27	0.00	0.23	0.19	129.5	192.0	323.9	235.6	230.8
7	0.27	23.6	0.00	2.29	1.00	103.4	143.8	210.0	189.4	160.1
8	2.88	577.1	0.00	14.0	4.18	146.6	176.8	281.2	259.6	236.0
9	517.1	(3.5)	0.00	20.9	2.88	172.7	223.9	361.7	311.6	257.3
10	(1.6)	(13)	0.00	(0.3)	2.47	235.8	308.0	495.8	432.4	372.2
11	1216	(-)	0.00	(0.8)	42.5	289.3	(-)	533.9	527.1	384.7
12	(3.0)	(-)	0.00	(2.7)	61.2	326.0	(-)	689.0	592.1	570.9
13	(1.6)	(-)	0.00	(1.9)	2967	366.1	(-)	683.7	655.7	577.4
14	(1.9)	(-)	0.00	(1.3)	(0.6)	443.0	(-)	896.2	803.8	695.5
15	(1.8)	(-)	0.00	(1.0)	(0.9)	554.9	(-)	1164	1024	896.4

CPLEX finds no feasible solution to GNP for instances with more than 10 nodes. In fact, it is not possible to solve even the LP relaxations of GNP for instances with more than 14 nodes within an hour of CPU time. This is because the formulation for GNP is too huge for larger instances. This further reinforces the importance of using failure-flow patterns such as directed p-cycles to design survivable networks; we can solve much larger instances.

In Figure 5.3, we report the ratio of installed capacity for the solutions provided by GNP, HUP, SDC, and SDPb to the capacity requirements of NDP. Here we see that survivable networks produced by HUP have about 100% more capacity installed compared with the NDP, whereas SDPb requires only about 50% increase in the capacity. On the other hand, GNP requires about 25% more capacity than NDP. Thus, when compared with GNP, which provisions the lowest possible capacity for survivable networks, we see that SDPb requires only an additional 20% capacity, whereas HUP provisions 60% excess capacity over GNP.

We consider only a subset of all possible directed p-cycles in SDPb. We can potentially achieve even lower costs for SDP by generating other directed p-cycles in the network. This reinforces our results that survivable networks designed using directed

Figure 5.3: Comparing capacity efficiency of the survivability models



p-cycles have comparable capacity efficiency to GNP. Interestingly, even though we generate directed cycle variables as needed, SDC performs worse than SDPb for all but one instance. In fact, SDC requires 80% more capacity than NDP, and 44% more than GNP.

When we compare the models in terms of ease of solvability, we see that the HUP model is the easiest to solve; it takes negligible time. However, HUP requires the solution of NDP as an input, and the solution time of NDP must be included for designing survivable networks with HUP. SDC takes about as much time as NDP. Thus, we can incorporate survivability using the SDC model without much loss in computation time. Surprisingly, we see in Table 5.2 that SDPb is solved more easily than NDP and we were able to obtain optimal solutions or feasible solutions within 1% of optimal (for the subset of directed cycle variables used in the formulation) for all instances in Table 5.2.

In Table 5.3 we compare SDP with SDPb to see the effect of generating variables as needed rather than solving SDPb on a subset of the variables selected *a priori*. A comparison of columns (3) indicate that the capacity efficiency of the networks improve by about 4% on average by pricing the directed p-cycle variables based on their LP reduced

costs. Furthermore, this improvement gets better with increasing problem size; 16% for networks with more than twelve nodes. SDP produces networks that have about 15% more capacity than the ones from GNP, whereas SDPb has 20% excess capacity. The LP solution times for SDP, which include the time for pricing variables, indicate that columns can be generated very efficiently. In the last two columns of Table 5.3, we see that only a small number of the variables are generated.

Table 5.3: The effect of column generation of directed p-cycles

$\|V\|$	SDPb			SDP				
	(1)	(2)	(3)	(1)	(2)	(3)	paths added	p-cycles added
5	0.00	0.04	110.5	0.01	0.05	93.8	23	8
6	0.00	0.19	230.8	0.01	0.19	222.9	36	6
7	0.01	1.00	160.1	0.01	0.41	162.8	51	4
8	0.01	4.18	236.0	0.01	2.55	221.3	107	14
9	0.02	2.88	257.3	0.02	6.03	262.8	143	14
10	0.04	2.47	372.2	0.02	4.99	358.9	182	10
11	0.10	42.5	384.7	0.06	88.0	384.5	282	26
12	0.05	61.2	570.9	0.17	2337	428.6	499	38
13	0.08	2967	577.4	0.24	(1.4)	503.4	567	50
14	0.12	(0.6)	695.5	0.31	(0.2)	598.4	702	46
15	0.19	(0.9)	896.4	0.42	(0.2)	761.3	839	46

(1) LP solution time, (2) time (endgap), (3) best feasible solution.

Finally, in Table 5.4 we report the results of the our experiments with the column generation approach for large instances of SDP. Table 5.4 demonstrates that relatively large instances are effectively solved by SDP. These computational experiments suggest that the method of using directed p-cycles to reroute disrupted flow is quite effective in designing survivable networks.

Table 5.4: Experiments with large instances

$\|V\|$	LP solution time	(endgap)	best feasible solution	paths	p-cycles
20	3.97	(0.7)	1641	2171	100
30	27.8	(1.4)	3567	3549	190
40	123.4	(2.8)	6245	7262	262
50	186.3	(1.1)	12212	10643	352
60	2315	(0.6)	28754	14251	366
70	1333	(1.4)	24121	23490	390

5.5 Conclusions

In this chapter, we presented a new method (SDP) for designing capacity-efficient survivable telecommunication networks. This method differs from the hierarchical edge capacity covering methods such as HUP (Grover and Stamatelakis 1998), in that we simultaneously route no-failure flows and determine slacks for disrupted flow through directed p-cycles of the network. Therefore, capacity efficiency achieved by SDP is always at least as high as HUP.

Our computational experiments show that SDPb delivers consistently about 25% more capacity-efficient networks than HUP does. SDP differs from SDC in the sense that it uses directed p-cycles as failure-flow patterns. Since these directed p-cycles dominate directed cycles, capacity efficiency achieved by SDP must be at least as good as SDC. In fact, SDP outperforms SDC by 20%. Surprisingly, even with a restricted set, SDPb outperforms SDC significantly (approximately 17%). In fact, SDP compares well with global rerouting (GNP), which gives theoretically the most capacity-efficient survivable networks possible (see Figure 5.3).

To solve large instances we developed a column generation approach. We showed that the pricing problem for the directed p-cycle variables problem is $\mathcal{NP}$-hard, and presented an efficient heuristic to price them effectively. Judicious selection of directed p-cycle variables seems to be very important in increasing the capacity efficiency of the networks. Pricing the variables based on their LP reduced costs, even with a heuristic method, improved capacity efficiency of the networks over selecting them *a priori* by about 4%. Our computational experiments suggest that SDP is an effective way for designing survivable telecommunication networks.

5.6 Future directions with failure-fbw patterns

There are several directions for further research. In particular, we need to develop other efficient ways of pricing the directed p-cycle variables. We plan to study the minimum-weight directed p-cycle problem in detail, which is interesting in itself.

Just as in the case of SDC, the arc-set inequalities developed for NDP in Chapter 2 can be extended to the arc-set polyhedra of SDP. The capacity constraint is unchanged from SDC and SDP, and thus all the results for the arc-set polyhedra of the SDC are valid. Therefore, residual capacity inequalities (4.3) are the only class of cutting planes for SDP that can be derived from the arc set; see Section 4.2.

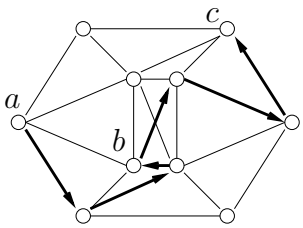
In Chapter 6, we continue the polyhedral study of survivable network design problems. In particular, we develop metric-type inequalities for NDP, SDC, and SDP. In Section 6.4, we see that these inequalities bring down the solution times for solving SDP by an order of magnitude. As a special case of these new inequalities, we obtain the cut-set inequality for SDP, see Section 6.2.3. Finally, in Chapter 7 we study the mixed-integer knapsack set. Inequalities developed for this set can be used to develop more valid-inequalities for SDP.

We mentioned in Section 3.4 that Stamatelakis and Grover (2000) study the effect of using various predetermined failure-flow structures and prove that no other structure is more efficient for uncapacitated problems. However, this does not imply that a directed p-cycle is the most capacity-efficient structure for the capacitated survivable network design problem. There may be other structures which provide lower capacity utilization. Further, it may be possible to achieve better capacity efficiency by using more than one structure at the same time. This is a subject of future research; here we present some initial thoughts on two other failure-flow patterns - paths and trees.

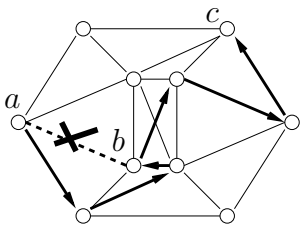
First, we discuss directed paths as failure-flow patterns. Figure 5.4 illustrates the way in which an individual path may be used for restoration. In (a), an example of a directed path is shown. In (b), edge $[ab]$ breaks, and the arcs of the path are used for restoring any disrupted flow on arc (ab). Further inspection shows that this particular directed path provides a restoration path for five off-path failures. No restoration paths are provided when edges on the path fail; unlike directed p-cycles.

Second, we discuss directed trees as failure-flow patterns. Figure 5.5 illustrates the way in which an individual tree may be used for restoration. In (a), an example of a directed tree is shown. In (b), edge $[ab]$ fails, and the arcs of the tree (from a to b through node c) are used for restoring any disrupted flow on arc (ab). Further inspection shows

Figure 5.4: Directed paths as failure-flow patterns



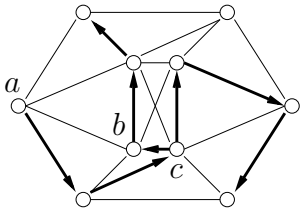
a) A directed path



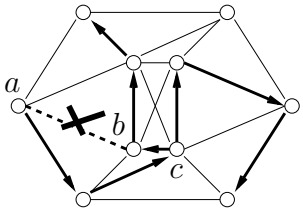
b) Edge $[a, b]$ fails

that this particular directed tree provides a restoration path for seven off-tree failures. Again, no restoration paths are provided when edges on the tree fail.

Figure 5.5: Directed trees as failure-flow patterns



a) A directed tree



b) Edge $[a, b]$ fails

Chapter 6

Metric-type inequalities for survivable network design

In this chapter, we develop metric-type inequalities for survivable network design. We first review known metric inequalities for the network design problem with no survivability (NDC), and then describe new inequalities for the survivable network design using directed cycles (SDC) and directed p-cycles (SDP), respectively.

We strengthen the metric-type inequalities by integer rounding or mixed-integer rounding, as the case may be. We also develop more valid inequalities by generating facets of the integer knapsack set (see Chapter 7) defined by these metric inequalities. Depending on the set, we refer to these new metric-type inequalities as either pure-metric inequalities or mixed-metric inequalities.

We show that known cut-set inequalities for NDP and SDC (see Sections 2.2 and 4.2) are special cases of the strengthened metric-type inequalities. We also derive new cut-set inequalities for SDP as special cases of the pure-metric and mixed-metric inequalities.

We assume that there exists no pre-installed capacity on the network, for ease of exposition. All the results in this chapter can be easily modified for the case with pre-existing capacity w^0 on the edges of the graph. We use arc-based formulations to derive the results in this chapter; path-based formulations yield the same results.

We begin by describing the general procedure used to develop metric-type inequal-

ities in Section 6.1. In Section 6.2, we derive metric-type inequalities with only pure integer variables. We extend these inequalities to include continuous variables in Section 6.3. For both these cases, we repeat the analysis for NDP, SDC, and SDP. For NDP and SDC, we show that known cut-set inequalities are special cases of strengthened metric-type inequalities. Using similar techniques, we develop 2-partition inequalities for SDP. Finally, in Section 6.4, we present computational results illustrating the effectiveness of the metric-type inequalities in solving survivable network design problems.

6.1 Overview of procedure to develop metric-type inequalities

We describe the general procedure used in this chapter to derive metric-type inequalities for NDP, SDC, and SDP. For the definitions of many of the terms used in this overview, we refer the reader to Section 1.4.

Let $|N| = n$, and A be an $m \times n$ matrix. Consider the linear program

$$z_{LP} = \max\{cx \,:\, x \in S_{LP}\},$$

where $S_{LP} = \{x \in \mathbb{R}^n_+,\ Ax \leq b\}$.

We partition the set N into sets N_0 and N_1, and refer to the corresponding variables and matrix of columns by x_0, x_1 and A_0, A_1 respectively. Let $|N_i| = n_i,\ i = 0, 1$. Suppose that we want to derive metric-type inequalities which involve only the variables x_1.

Consider the restriction obtained by fixing the variables x_1 to $\bar{x}_1$, and denote it by $S_{\bar{x}_1}$. Now, any $x_0 \in S_{\bar{x}_1}$ if and only if

$$S_{\bar{x}_1} = \{x_0 \in \mathbb{R}^{n_0}_+,\ A_0 x_0 \leq b - A_1 \bar{x}_1\}.$$

By Farkas' Lemma, $S_{\bar{x}_1} \neq \emptyset$ if and only if there exist no $w \in \mathbb{R}^m_+$ such that $wA_0 \geq 0,\ w(b - A_1\bar{x}_1) < 0$. In other words, $\bar{x}_1$ yields a feasible solution to S_{LP} for some x_0 if and only if $w(b - A_1\bar{x}_1) \geq 0$ for all w in the dual cone of $S_{\bar{x}_1}$. Therefore, $w(b - A_1 x_1) \geq 0$ for all w in the dual cone of $S_{\bar{x}_1}$, and we have proved the following.

Proposition 6.1 Metric-type inequality $wA_1x_1 \leq wb$ is valid for S_{LP} for all w in the dual cone of $S_{\bar{x}_1}$. □

This procedure can be repeated for any $H \subseteq N$ to get valid inequalities in the variables x_H. Naturally, we would like to find non-dominated inequalities. The following result states how this can be done.

Proposition 6.2 Metric-type inequality $\bar{w}A_1x_1 \leq \bar{w}b$ is not dominated by any other metric inequality if $\bar{w}$ is an extreme ray of the dual cone of $S_{\bar{x}_1}$. □

Since the metric-type inequality is derived from feasibility conditions using Farkas' Lemma, it is always valid for S_{LP}. In the context of mixed-integer programming, a metric-type inequality obtained using its LP relaxation is valid for the LP relaxation. To convert this to a valid inequality for the convex hull of the mixed-integer program such that it also cuts off parts of S_{LP}, it needs to be strengthened. The metric-type inequalities can be strengthened by mixed-integer rounding. Alternatively, we can compute the facets of the mixed-integer knapsack set described by the inequality.

This procedure was first used in the context of multi-commodity flow problems to develop metric inequalities (Iri 1971, Onaga and Kakusho 1971). Since then, the technique has been used for many classes of problems, including network design, see (Avella et al. 2004) for recent work in this direction.

6.2 Inequalities with no continuous variables

6.2.1 Metric inequalities for NDP

Let $G = (V, E)$ be an undirected graph with node set V and edge set E. Let F be the set of all ordered pairs (arcs) from E, i.e., $F = \{(ij), (ji) : [ij] \in E\}$. We use (ij) to denote the arc from node i to node j, and $[ij]$ to denote the edge between nodes i and j. We use $G' = (V, F)$ to denote the directed graph. We use $F \setminus [ij]$ to represent $F \setminus \{(ij), (ji)\}$. Let $K = [1, |K|]$ be the set of commodities. Let $\{(s^k, t^k, d^k)\}_{k \in K}$ be the commodity triples of source and destination nodes s^k and t^k, and d^k be the supply at s^k for t^k, $k \in K$. Let b_i^k be the supply of commodity k at node i, i.e., $b_{s^k}^k = d^k$, $b_{t^k}^k = -d^k$, and $b_i^k = 0$ for $i \in V \setminus \{s^k, t^k\}$.

We define variable y_{ij}^k as the fraction of commodity k routed through arc $(ij) \in F$. We define the capacity variable $x_{[ij]}$ as the amount of capacity installed on edge $[ij]$.

Given a capacity vector $\bar{x} \in \mathbb{Z}^{|E|}$, a feasible solution to the NDP exists for this $\bar{x}$ if and only if there exist $y_{ij}^k \in \mathbb{R}_+,\ (ij) \in F, k \in K$ such that

$$(w_i^k) \qquad \sum_{(ij)\in F} d^k y_{ij}^k - \sum_{(ji)\in F} d^k y_{ji}^k = b_i^k \qquad \forall i \in V,\ \forall k \in K \tag{6.1}$$

$$(u_{ij}) \qquad \sum_{k\in K} d^k y_{ij}^k \le \bar{x}_{[ij]} \qquad \forall (ij) \in F, \tag{6.2}$$

where w, u are dual variables to constraints (6.1) and (6.2), respectively. By Farkas' Lemma, this is true if and only if

$$\sum_{(ij)\in F} u_{ij}\bar{x}_{[ij]} \ge -\sum_{i\in V}\sum_{k\in K} w_i^k b_i^k \tag{6.3}$$

for all $u \in \mathbb{R}^{|F|}, w \in \mathbb{R}^{|V||K|}$ such that

$$u_{ij} \ge w_j^k - w_i^k \quad \forall k \in K,\ \forall (ij) \in F \tag{6.4}$$

$$u_{ij} \in \mathbb{R}_+ \qquad \forall (ij) \in F. \tag{6.5}$$

For all $k \in K$, $b_i^k = 0$ unless $i = s^k$ or t^k; we can simplify constraint (6.3).

Definition 6.3 For all u, w satisfying constraints (6.4) and (6.5),

$$\sum_{(ij)\in F} u_{ij} x_{[ij]} \ge \sum_{k\in K} d^k (w_{t^k}^k - w_{s^k}^k) \tag{6.6}$$

is a *metric inequality for NDP* with only integer variables. □

To simplify the following discussion, we use γ to denote the right hand side of the inequality (6.6). Given $\bar{x}$, finding the most violated inequality is the following linear program, and can be solved in polynomial time.

$$\begin{aligned} \max \sum_{k\in K} d^k (w_{t^k}^k - w_{s^k}^k) - \sum_{(ij)\in F} u_{ij}\bar{x}_{[ij]} & \\ s.t.:\ w_j^k - w_i^k \le u_{ij} \qquad & \forall k \in K,\ \forall (ij) \in F \\ u_{ij} \in \mathbb{R}_+ \qquad & \forall (ij) \in F. \end{aligned} \tag{6.7}$$

When u is fixed, the problem decomposes for each commodity, and can be solved

more efficiently since (6.7) are the dual constraints for a shortest path problem on a directed network (Ahuja et al. 1993). For the most violated inequality (6.6) when u is given, we wish to maximize γ, and thus w are the shortest path labels when the arc weights are defined as u. The contribution to γ from commodity k is exactly the weight of the shortest path from source s^k to destination t^k.

Since the metric inequalities for NDP are derived from the dual of the LP relaxation of NDP, they do not violate any fractional points. To obtain strong valid inequalities for NDP, the metric inequalities need to be strengthened.

An immediate strengthening is obtained by integer rounding (IR). We can scale the coefficients u arbitrarily; without loss of generality we assume that $u \in \mathbb{Z}$. Rounding up the right hand side, we obtain the rounded metric inequalities (Avella et al. 2004)

$$\sum_{(ij)\in F} u_{ij} x_{[ij]} \geq \lceil \gamma \rceil. \tag{6.8}$$

Next, we show that the cut-set inequality with only integral capacity variables is a special case of the rounded metric inequalities. For a non-empty partition (A, B) of the nodes V, let $[AB]$ be the edges with one end in A, the other in B; corresponding to these edges, let AB be the arcs directed from A to B, and BA be the arcs directed from B to A. Let $K_A = \{k \in K : s^k \in A,\ t^k \notin A\}$, $K_B = \{k \in K : s^k \in B,\ t^k \notin B\}$, $d_A = \sum_{k\in K_A} d^k$, and $d_B = \sum_{k\in K_B} d^k$. Without loss of generality, we assume that $d_A \geq d_B$.

If we fix $u_{ij} = 1$ for all $(ij) \in AB$, and 0 otherwise, then the left hand side of (6.8) is the same as the left hand side of the cut-set inequality (2.3). Now, for commodity $k \in K$, we have the cost of its shortest path equal to 1 if $k \in K_A$, and 0 otherwise. Thus, $\gamma = d_A$, and we have proved the following.

Proposition 6.4 For partition (A, B), the cut-set inequality for NDP

$$\sum_{[ij]\in[AB]} x_{[ij]} \geq \lceil d_A \rceil$$

is obtained by rounding the special case of the metric inequality for NDP (see Definition 6.3) where $u_{ij} = 1$ if $(ij) \in AB$, and 0 otherwise. □

6.2.2 Metric inequalities for SDC

Let $\mathcal{C}$ be the set of directed cycles of G'. We define the variable z_c to denote the amount of slack reserved on directed cycle c. Let α_{ij}^c be 1 if directed cycle c includes arc (ij), and 0 otherwise.

Given a capacity vector $\bar{x} \in \mathbb{Z}^{|E|}$, a feasible solution to SDC exists for this $\bar{x}$ if and only if there exist $y_{ij}^k \in \mathbb{R}_+,\ (ij) \in F, k \in K$ and $z_c \in \mathbb{R}_+,\ c \in \mathcal{C}$ such that

$$(w_i^k) \qquad \sum_{(ij)\in F} d^k y_{ij}^k - \sum_{(ji)\in F} d^k y_{ji}^k = b_i^k \qquad \forall i \in V,\ \forall k \in K \tag{6.9}$$

$$(u_{ij}) \qquad \sum_{k\in K} d^k y_{ij}^k \leq \bar{x}_{[ij]} \quad \forall (ij) \in F \tag{6.10}$$

$$(v_{ij}) \qquad \sum_{k\in K} d^k y_{ij}^k - \sum_{c\in\mathcal{C}} \alpha_{ji}^c z_c \leq 0 \qquad \forall (ij) \in F, \tag{6.11}$$

where w, u, v are dual variables to constraints (6.9), (6.10), and (6.11), respectively. By Farkas' Lemma, this is true if and only if

$$\sum_{(ij)\in F} u_{ij}\bar{x}_{[ij]} \geq \sum_{k\in K} d^k (w_{t^k}^k - w_{s^k}^k)$$

for all $u \in \mathbb{R}^{|F|}, v \in \mathbb{R}^{|F|}, w \in \mathbb{R}^{|V||K|}$ such that

$$u_{ij} + v_{ij} \geq w_j^k - w_i^k \quad \forall k \in K,\ \forall (ij) \in F \tag{6.12}$$

$$\sum_{(ij)\in F} \alpha_{ij}^c (u_{ij} - v_{ji}) \geq 0 \qquad \forall c \in \mathcal{C} \tag{6.13}$$

$$u_{ij}, v_{ij} \in \mathbb{R}_+ \qquad \forall (ij) \in F. \tag{6.14}$$

Definition 6.5 For all u, v, w that satisfy constraints (6.12)- (6.14),

$$\sum_{(ij)\in F} u_{ij} x_{[ij]} \geq \sum_{k\in K} d^k (w_{t^k}^k - w_{s^k}^k) \tag{6.15}$$

is a *metric inequality for SDC* with only integer variables. □

Again, we use γ to denote the right hand side of the inequality (6.15). Finding the most violated inequality is the following LP; however we have an exponential number of

directed cycle constraints (6.17).

$$\max \sum_{k \in K} d^k (w^k_{t^k} - w^k_{s^k}) - \sum_{(ij) \in F} u_{ij} \bar{x}_{[ij]}$$

$$s.t.: \qquad w^k_j - w^k_i \leq u_{ij} + v_{ij} \quad \forall k \in K,\ \forall (ij) \in F \tag{6.16}$$

$$\sum_{(ij) \in F} \alpha^c_{ij} (u_{ij} - v_{ji}) \geq 0 \qquad \forall c \in \mathcal{C} \tag{6.17}$$

$$u_{ij}, v_{ij} \in \mathbb{R}_+ \qquad \forall (ij) \in F.$$

The separation problem for inequalities (6.17) is exactly the same as the pricing of directed cycle variables in SDC, and can be done in polynomial time. Thus, this LP is polynomially solvable by the ellipsoid algorithm.

When u, v are fixed such that inequalities (6.17) are satisfied for all directed cycles in the network, the problem decomposes for each commodity. As in Section 6.2.1, the problem of finding the maximum γ can be solved more efficiently since (6.16) are the dual constraints for a shortest path problem on a directed network (Ahuja et al. 1993). Thus, w are the shortest path labels when the weight on arc (ij) is $u_{ij} + v_{ij}$, and the contribution to γ from commodity k is exactly the weight of the shortest path from source s^k to destination t^k.

Since the metric inequalities for SDC (6.15) are derived from the dual of the LP relaxation of SDC, they do not violate any fractional points. As in Section 6.2.1, they need to be strengthened to obtain strong valid inequalities for SDC.

An immediate strengthening is obtained by integer rounding. By scaling, we assume without loss of generality that $u \in \mathbb{Z}$ and then round up the right hand side to obtain

$$\sum_{(ij) \in F} u_{ij} x_{[ij]} \geq \lceil \gamma \rceil. \tag{6.18}$$

We call this class of inequalities the rounded metric-type inequalities for SDC.

Let $G'_A = (A, F_A), G'_B = (B, F_B)$ be the sub-graphs defined by the partition (A, B), and let $\bar{\mathcal{C}}$ be the set of directed cycles that cross the partition.

Proposition 6.6 For partition (A, B), the cut-set inequality for SDC (4.9)

$$x([AB]) \geq \lceil 2d_A \rceil$$

is obtained by rounding the special case of the metric inequality for SDC (see Definition 6.5) where $u_{ij} = 1$ if $(ij) \in AB$, and 0 otherwise. □

When $u_{ij} = 1$ if $(ij) \in AB$, and 0 otherwise, the left hand side of (6.18) corresponds to the cut-set inequality (4.9). The right hand side is equal to $\lceil \gamma \rceil$, where γ is the solution to the following optimization problem:

$$\max \sum_{k \in K} d^k (w^k_{t^k} - w^k_{s^k})$$

$$\begin{aligned} s.t.: \quad & w^k_j - w^k_i \leq 1 + v_{ij} && \forall k \in K,\ \forall (ij) \in AB && (6.19) \\ & w^k_j - w^k_i \leq v_{ij} && \forall k \in K,\ \forall (ij) \in F \setminus AB \\ - \sum_{(ij) \in F \setminus AB} \alpha^c_{ij} v_{ji} + \sum_{(ij) \in AB} \alpha^c_{ij} (1 - v_{ji}) & \geq 0 && \forall c \in \mathcal{C} && (6.20) \\ & v_{ij} \in \mathbb{R}_+ && \forall (ij) \in F. \end{aligned}$$

Consider the following solution to this optimization problem. Fix $v_{ij} = 1$ if $(ij) \in AB$, and 0 otherwise. Now the left hand side of constraints (6.20) reduces to $-\sum_{(ij) \in BA} \alpha^c_{ij} + \sum_{(ij) \in AB} \alpha^c_{ij}$. All directed cycles $c \notin \bar{C}$ satisfy constraints (6.20) for this allocation of u and v trivially. All directed cycles $c \in \bar{C}$ use some arc in AB exactly as many times as they use some arc in BA; thus u, v as chosen satisfy (6.20) for such cycles.

The problem reduces to shortest path problems for each commodity where the weight of arcs in AB is 2, and is 0 otherwise. For commodity $k \in K$, we have the cost of its shortest path equal to 2 if $k \in K_A$, and 0 otherwise. Thus, $\gamma \geq 2d_A$, and we have exactly the right hand side of the cut-set inequality (4.9). The following proposition proves that this solution is optimal under mild conditions.

Proposition 6.7 If the sub-graphs G'_A and G'_B are 2-connected, then the solution $v_{ij} = 1$ if $(ij) \in AB$, and 0 otherwise, is optimal when $u_{ij} = 1$ if $(ij) \in AB$, and 0 otherwise.

Proof See Appendix E. □

6.2.3 Metric inequalities for SDP

Finally, we develop metric inequalities for SDP. Let $\rho^c_{[ij]}$ be 1 if edge $[ij]$ is a chord to directed p-cycle c, and 0 otherwise. Given a capacity vector $\bar{x} \in \mathbb{Z}^{|E|}$, a feasible solution to SDP exists for this $\bar{x}$ if and only if there exist $y^k_{ij} \in \mathbb{R}_+,\ (ij) \in F, k \in K$ and $z_c \in \mathbb{R}_+,\ c \in \mathcal{C}$ such that

$$(w^k_i) \qquad \sum_{(ij)\in F} d^k y^k_{ij} - \sum_{(ji)\in F} d^k y^k_{ji} = b^k_i \qquad \forall i \in V,\ \forall k \in K \tag{6.21}$$

$$(u_{ij}) \qquad \sum_{k\in K} d^k y^k_{ij} \le \bar{x}_{[ij]} \quad \forall (ij) \in F \tag{6.22}$$

$$(v_{ij}) \qquad \sum_{k\in K} d^k y^k_{ij} - \sum_{c\in\mathcal{C}} \alpha^c_{ji} z_c - \sum_{c\in\mathcal{C}} \rho^c_{[ij]} z_c \le 0 \qquad \forall (ij) \in F, \tag{6.23}$$

where w, u, v are dual variables to constraints (6.21), (6.22), and (6.23), respectively. By Farkas' Lemma, this is true if and only if

$$\sum_{(ij)\in F} u_{ij} \bar{x}_{[ij]} \ge \sum_{k\in K} d^k (w^k_{t^k} - w^k_{s^k})$$

for all $u \in \mathbb{R}^{|F|}, v \in \mathbb{R}^{|F|}, w \in \mathbb{R}^{|V||K|}$ such that

$$u_{ij} + v_{ij} \ge w^k_j - w^k_i \quad \forall k \in K,\ \forall (ij) \in F \tag{6.24}$$

$$\sum_{(ij)\in F} \alpha^c_{ij}(u_{ij} - v_{ji}) - \sum_{(ij)\in F} \rho^c_{[ij]} v_{ij} \ge 0 \qquad \forall c \in \mathcal{C} \tag{6.25}$$

$$u_{ij}, v_{ij} \in \mathbb{R}_+ \qquad \forall (ij) \in F. \tag{6.26}$$

Definition 6.8 For all u, v, w that satisfy constraints (6.24)-(6.26),

$$\sum_{(ij)\in F} u_{ij} x_{[ij]} \ge \sum_{k\in K} d^k (w^k_{t^k} - w^k_{s^k}) \tag{6.27}$$

is a *metric inequality for SDP* with only integer variables. □

As in Sections 6.2.1 and 6.2.2, we use γ to denote the right hand side of the metric-type inequality for SDP (6.27). Finding the most violated metric inequality is the following

LP; we have an exponential number of directed p-cycle constraints (6.29).

$$\max \sum_{k \in K} d^k (w^k_{t^k} - w^k_{s^k}) - \sum_{(ij) \in F} u_{ij} \bar{x}_{[ij]}$$

$$s.t.: \qquad w^k_j - w^k_i \leq u_{ij} + v_{ij} \quad \forall k \in K,\ \forall (ij) \in F \tag{6.28}$$

$$\sum_{(ij) \in F} \alpha^c_{ij} (u_{ij} - v_{ji}) - \sum_{(ij) \in F} \rho^c_{[ij]} v_{ij} \geq 0 \qquad \forall c \in \mathcal{C} \tag{6.29}$$

$$u_{ij}, v_{ij} \in \mathbb{R}_+ \qquad \forall (ij) \in F.$$

The separation problem for constraints (6.29) is exactly the same as the pricing of directed p-cycle variables in SDP, which was shown to be $\mathcal{NP}$-hard (Theorem 5.2). Thus, finding the most violated metric-type inequality for SDP is $\mathcal{NP}$-hard.

When u, v are fixed such that constraints (6.29) are satisfied for all directed p-cycles in the network, the problem decomposes for each commodity. As before, when u, v are fixed, constraints (6.28) are the dual constraints for a shortest path problem on a directed network (Ahuja et al. 1993), and can be solved efficiently. Thus, w are the shortest path labels when the weight on arc (ij) is $u_{ij} + v_{ij}$, and the contribution to γ from commodity k is exactly the weight of the shortest path from source s^k to destination t^k.

Since the metric inequalities (6.27) are derived from the dual of the LP relaxation of SDP, they do not violate any fractional points. As in Sections 6.2.1 and 6.2.2, the metric inequalities need to be strengthened to obtain strong valid inequalities for SDP.

An immediate strengthening is obtained by integer rounding. Without loss of generality, we assume that the coefficients u are scaled such that $u \in \mathbb{Z}$. Rounding up the right hand side, we obtain

$$\sum_{(ij) \in F} u_{ij} x_{[ij]} \geq \lceil \gamma \rceil. \tag{6.30}$$

We call this class of inequalities the rounded metric-type inequalities for SDP. We next present two classes of partition inequalities for SDP that are special cases of the rounded metric-type inequalities.

6.2.3.1 Cardinality-k cut-set inequality

As for NDP and SDC, the cardinality-k cut-set inequality for SDP corresponding to the partition (A, B) is a special case of the rounded metric-type inequalities. We obtain the cardinality-k cut-set inequality as follows. Let $u_{ij} = 1$ if $(ij) \in AB$, and 0 otherwise. Then, the left hand side of (6.18) corresponds to the cardinality-k cut-set inequality, see Sections 6.2.2 and 6.2.1 for the cut-set inequality for NDP and SDC, respectively. The right hand side is equal to γ, which is the solution to the following optimization problem:

$$\begin{aligned}
&\max \sum_{k\in K} d^k(w^k_{t^k} - w^k_{s^k}) \\
s.t.: \quad & w^k_j - w^k_i - 1 \le v_{ij} && \forall k \in K,\ \forall (ij) \in AB \\
& w^k_j - w^k_i \le v_{ij} && \forall k \in K,\ \forall (ij) \in F \setminus AB \\
-\sum_{(ij)\in F\setminus AB} \alpha^c_{ij} v_{ji} + \sum_{(ij)\in AB} \alpha^c_{ij}(1 - v_{ji}) - \sum_{(ij)\in F} \rho^c_{[ij]} v_{ij} \ge 0 \quad & && \forall c \in \mathcal{C} \qquad (6.31)\\
& v_{ij} \in \mathbb{R}_+ && \forall (ij) \in F.
\end{aligned}$$

Consider the following solution. We fix $v_{ij} = \nu \ge 0$, for all $(ij) \in AB$ and 0, otherwise. Since u, v have been fixed, the problem reduces to shortest path problems for each commodity where the weight of arcs in AB is $1 + \nu$, and is 0 otherwise. For commodity k, the cost of its shortest path is equal to $1 + \nu$ if $k \in K_A$, and 0 otherwise. Thus, $\gamma \ge d_A(1 + \nu)$. To ensure feasibility of constraints (6.31), we need

$$-\sum_{(ij)\in BA} \alpha^c_{ij}\nu + \sum_{(ij)\in AB} \alpha^c_{ij} - \sum_{(ij)\in AB} \rho^c_{[ij]}\nu \ge 0 \quad \forall c \in \mathcal{C}.$$

For directed p-cycle c, let ℓ_c be the number of times it uses an arc in AB, and let ω_c be the number of chords among the edges in $[AB]$. Then, constraint (6.31) is satisfied if $\nu(\ell_c + \omega_c) \le \ell_c$ for all directed p-cycles $c \in \mathcal{C}$. This is satisfied for any ν by all directed p-cycles c that do not cross the partition (A, B). Let $\bar{\mathcal{C}}$ be the set of directed p-cycles that cross the partition. To find the most violated inequality (6.30), we set ν as high as possible. Defining $\mu_{AB} = \min_{c\in\bar{\mathcal{C}}}\{\ell_c/(\ell_c + \omega_c)\}$, we set $\nu = \mu_{AB}$. Proposition 6.9 states this allocation is optimal under mild conditions. For partition (A, B), we obtain the

cardinality-k cut-set inequality for SDP as

$$x([AB]) \geq \lceil d_A(1+\mu_{AB}) \rceil. \tag{6.32}$$

Proposition 6.9 If the sub-graphs G'_A and G'_B are 2-connected, then the solution $v_{ij} = \mu_{AB}$ if $(ij) \in AB$, and 0 otherwise, is optimal when u is fixed so that the left hand side corresponds to the cardinality-k cut-set inequality (6.32), i.e., when $u_{ij} = 1$ if $(ij) \in AB$, and 0 otherwise.

Proof See Appendix E. □

To obtain μ_{AB}, we need to solve an optimization problem over an exponential set of directed p-cycles. We define ω_{AB} as the maximum ratio of (chords)/(arcs) that any directed p-cycles may have among the edges $[AB]$, i.e., $\omega_{AB} = \max_{c \in \bar{\mathcal{C}}} \omega_c/\ell_c$. Thus, $\mu_{AB} = 1/(1+\omega_{AB})$. Given graph $G = (V, E)$ and partition (A, B), we refer to the problem of finding ω_{AB} as the maximum chord problem (MCP).

Theorem 6.10 The maximum chord problem (MCP) is $\mathcal{NP}$-hard.

Proof See Appendix E. □

Since solving for ω_{AB} is $\mathcal{NP}$-hard, we wish to find an upper bound to ω_{AB}. For directed p-cycle $c \in \bar{\mathcal{C}}$, $\omega_c \leq |AB| - 2$. Since $\ell_c \geq 1$, we get $\omega_c/\ell_c \leq |AB| - 2$, Thus, we have $\mu_{AB} \geq 1/(|AB| - 1)$. To maximize the right hand side of (6.32), we set $\nu = 1/(|AB| - 1)$, and we have proved the following.

Proposition 6.11 For partition (A, B), the cardinality-k cut-set inequality for SDP

$$x([AB]) \geq \lceil d_A \frac{|AB|}{|AB| - 1} \rceil. \tag{6.33}$$

is obtained by rounding the special case of the metric-type inequality for SDP (see Definition 6.8) where $u_{ij} = 1$ if $(ij) \in AB$, and 0 otherwise. □

Inequality (6.33) can be strengthened to

$$x([AB]) \geq \lceil \lceil d_A \rceil \frac{|AB|}{|AB| - 1} \rceil. \tag{6.34}$$

This is obtained by integer rounding from another special case of the rounded metric-type

inequalities for SDP. We call this class of inequalities the strengthened cut-set inequality, and derive them in Section 6.2.3.2.

6.2.3.2 Strengthened cut-set inequality

We call these inequalities the strengthened cut-set inequalities since the left hand side contains the capacity variables for all but one edge in the cut set.

Proposition 6.12 For partition (A, B), let $[ab] \in [AB]$. The strengthened cut-set inequality for SDP

$$x([AB] \setminus \{[ab]\}) \geq \lceil d_A \rceil, \tag{6.35}$$

is obtained by rounding the special case of the metric-type inequality for SDP (see Definition 6.8) where $u_{ij} = 1$ if $(ij) \in AB \setminus \{[ab]\}$, and 0 otherwise. □

When $u_{ij} = 1$ if $(ij) \in AB \setminus \{[ab]\}$, and 0 otherwise, the left hand side of (6.30) corresponds to the strengthened cut-set inequality (6.35).

The right hand side is equal to $\lceil \gamma \rceil$, where γ is the optimal solution to the following maximization problem:

$$\begin{aligned}
\gamma = \max & \sum_{k \in K} d^k (w^k_{t^k} - w^k_{s^k}) & & \\
s.t.: & & w^k_j - w^k_i - 1 \leq v_{ij} \quad & \forall k \in K,\ \forall (ij) \in AB \setminus \{(ab)\} \\
& & w^k_j - w^k_i \leq v_{ij} \quad & \forall k \in K,\ \forall (ij) \notin AB \setminus \{(ab)\} \\
- \sum_{(ij) \notin AB \setminus \{(ab)\}} \alpha^c_{ij} v_{ji} + \sum_{(ij) \in AB \setminus \{(ab)\}} \alpha^c_{ij} (1 - v_{ji}) - \sum_{(ij) \in F} \rho^c_{[ij]} v_{ij} \geq 0 \quad & & \forall c \in \mathcal{C} \\
& & v_{ij} \in \mathbb{R}_+ \quad & \forall (ij) \in F.
\end{aligned} \tag{6.36}$$

In order to find a lower bound on γ, we fix $v_{ab} = 1$ and $v_{ij} = 0$ for all $(ij) \in F \setminus \{(ab)\}$. Then, the problem reduces to shortest path problems for each commodity where the weight of arcs in AB is 1, and is 0 otherwise. For commodities k in K_A, we have the cost of its shortest path equal to 1, and 0 otherwise. Thus, $\gamma \geq d_A$, and we obtain (6.35).

For this allocation of v to be feasible, constraints (6.36) need to be satisfied. For directed p-cycle c, the left hand side reduces to $-\alpha^c_{ba} + \sum_{(ij) \in AB \setminus \{(ab)\}} \alpha^c_{ij} - \rho_{[ab]}$. Now,

the second term in this expression is equal to 0 if and only if the directed p-cycle only uses arc (ab) among all arcs in AB. In this case, both α_{ba} and $\rho_{[ab]}$ are zero, and (6.36) is satisfied. If directed p-cycle does not use arc (ab), but uses more than one arc in AB, then (6.36) is satisfied trivially. In the only remaining case, directed p-cycle uses only one arc in AB, say (ij). This implies that the directed p-cycle c crosses the partition (A, B) exactly once. If, to cross over from B to A, the directed p-cycle either uses arc (ba), then $\alpha^c_{ba} = 1, \rho_{[ab]} = 0$; otherwise $\alpha^c_{ba} = 0, \rho_{[ab]} \leq 1$. In either case, (6.36) is satisfied. Furthermore, it can be shown that this allocation of v maximizes γ so long as the subgraphs G'_A and G'_B are connected.

By summing up (6.35) for all $[ab] \in [AB]$, dividing the resultant inequality by $|AB| - 1$, and applying integer rounding, we obtain the cardinality-k cut-set inequality (6.34).

Interestingly, (6.35) and (6.34) are the same as the strengthened cut-set inequality and the cardinality-k cut-set inequality, respectively, derived for global rerouting shared protection (GNP) in Balakrishnan et al. (2002). This provides some insight as to why SDP yields survivable networks with capacity efficiency comparable with GNP.

We remark the validity of cut-set inequalities for NDP and SDC that are special cases of the rounded metric inequalities (6.8) and (6.18) was shown in earlier works without using the arguments in this chapter. It is similarly possible to prove the validity of the strengthened cut-set inequalities (6.35) and the cardinality-k cut-set inequalities (6.34) using other arguments. However, we believe that the framework introduced in this chapter allows us to develop many other classes of inequalities as well; even those that are not easily derived otherwise.

6.2.4 Pure-metric inequalities

For the derivations in Section 6.2, we assumed that all the capacity variables $x \in E$ were given ($\bar{x}$). If only a strict subset of $x_{[ij]}$ were fixed, then the dual variables corresponding to the capacity constraints u_{ij}, u_{ji} will be forced to zero in all feasible solutions to the dual problem obtained using Farkas' Lemma. This is true even if some of the continuous variables are treated as given, as in Section 6.3. Therefore, without loss of generality, we

assume that all x are fixed throughout this chapter.

We can develop new metric-type inequalities from the metric inequalities (6.6), (6.15), and (6.27) by generating strong valid inequalities for the pure-integer knapsack set described by the metric inequality. In Section 7.5.1, we describe how this can be done, and call these new inequalities pure-metric inequalities. The rounded metric-type inequalities are one of the many classes of pure-metric inequalities that can be obtained by this procedure. Obtaining the most violated pure-metric inequality is $\mathcal{NP}$-hard, since the separation problem for even the cut-set inequalities is known to be $\mathcal{NP}$-hard (Bienstock 2001). As a heuristic, we can find the most violated metric inequality, and then scale the u variables to obtain a pure-integer knapsack set. Unfortunately, even this procedure is $\mathcal{NP}$-hard for SDP (6.27), since the separation problem for the exponential class of constraints in the optimization problem to find the most violated metric inequality is $\mathcal{NP}$-hard; see Section 6.2.3.

No closed-form characterization exists for the convex hull of the pure-integer knapsack set, except for special cases such as divisible coefficients $u_{ij},\ (ij) \in F$ (Pochet and Wolsey 1995). To guarantee that we obtain facets of the pure-integer knapsack set, we can solve the optimization problem for γ as a mixed-integer program where the variables u are constrained to be divisible. However, this optimization problem involves the pricing of an exponential class of cycle variables for SDC and SDP, and is handled by a branch-and-price algorithm. Furthermore, for SDP, the pricing problem of directed p-cycle variables is $\mathcal{NP}$-hard.

6.3 Inequalities with continuous variables

In Section 6.3, we develop metric-type inequalities that have nonzero coefficients for the continuous variables. We derive these inequalities using the procedure described in Section 6.1, and show that they can be strengthened by mixed-integer rounding. Furthermore, we can derive new inequalities by generating facets of the mixed-integer set defined by the inequalities to obtained mixed-metric inequalities. We also show that pre-

viously known flow cut-set inequalities for NDP are special cases of the strengthened inequalities; obtained by applying mixed-integer rounding (MIR) to special cases of the metric-type inequalities.

6.3.1 Metric inequalities for NDP

Assume that we are given a capacity vector $\bar{x} \in \mathbb{Z}^{|E|}$ and a flow vector $\bar{y} \in \mathbb{R}^{|\mathcal{F}|}$. Let $F^k \subseteq F$ be the set of arcs for which the flow is given, for commodity k; and let $K_{ij} \subseteq K$ be the set of commodities for which flow is given, for arc (ij). For $(\bar{y}, \bar{x})$, a feasible solution to the NDP exists if and only if there exist $y \in \mathbb{R}_+^{|(F\times K)\setminus\mathcal{F}|}$ such that

$$(w_i^k) \qquad \sum_{(ij)\in(F\setminus F^k)} d^k y_{ij}^k - \sum_{(ji)\in(F\setminus F^k)} d^k y_{ji}^k = b_i^k - \sum_{(ij)\in F^k} d^k \bar{y}_{ij}^k + \sum_{(ji)\in F^k} d^k \bar{y}_{ji}^k \quad \forall i \in V,\ \forall k \in K \tag{6.37}$$

$$(u_{ij}) \qquad \sum_{k\in(K\setminus K_{ij})} d^k y_{ij}^k \le \bar{x}_{[ij]} - \sum_{k\in K_{ij}} d^k \bar{y}_{ij}^k \qquad \forall (ij) \in F, \tag{6.38}$$

where w, u are dual variables to constraints (6.37) and (6.38), respectively. Thus by Farkas' Lemma, $(\bar{y}, \bar{x})$ yields a feasible solution to the NDP if and only if

$$\sum_{(ij)\in F} u_{ij}(\bar{x}_{[ij]} - \sum_{k\in K_{ij}} d^k \bar{y}_{ij}^k) \ge -\sum_{i\in V}\sum_{k\in K} w_i^k (b_i^k - \sum_{(ij)\in F^k} d^k \bar{y}_{ij}^k + \sum_{(ji)\in F^k} d^k \bar{y}_{ji}^k)$$

for all $u \in \mathbb{R}^{|F|}, w \in \mathbb{R}^{|V||K|}$ such that

$$w_j^k - w_i^k \le u_{ij} \quad \forall k \in K \setminus K_{ij},\ \forall (ij) \in F \tag{6.39}$$

$$u_{ij} \in \mathbb{R}_+ \qquad \forall (ij) \in F. \tag{6.40}$$

Definition 6.13 For all u, w that satisfy constraints (6.39) and (6.40),

$$\sum_{(ij)\in F} u_{ij}(x_{[ij]} - \sum_{k\in K_{ij}} d^k y_{ij}^k) - \sum_{i\in V}\sum_{k\in K} w_i^k (\sum_{(ij)\in F^k} d^k y_{ij}^k - \sum_{(ji)\in F^k} d^k y_{ji}^k) \ge \sum_{k\in K} d^k (w_{t^k}^k - w_{s^k}^k) \tag{6.41}$$

is a *metric inequality for NDP* with integer and continuous variables. □

Finding the most violated metric inequality (6.41) is equivalent to

$$\max \sum_{k\in K} d^k(w^k_{t^k} - w^k_{s^k}) - \sum_{(ij)\in F} u_{ij}(\bar{x}_{[ij]} - \sum_{k\in K_{ij}} d^k \bar{y}^k_{ij}) + \sum_{i\in V}\sum_{k\in K} w^k_i (\sum_{(ij)\in F^k} d^k \bar{y}^k_{ij} - \sum_{(ji)\in F^k} d^k \bar{y}^k_{ji})$$

subject to the constraints (6.39) and (6.40). This is a linear program, and can be solved in polynomial time.

When u is fixed, the problem decomposes for each commodity, and can be solved more efficiently since (6.39) are the dual constraints for a minimum cost network flow problem on a directed network (Ahuja et al. 1993).

Since constraints (6.41) do not violate any LP relaxation solution, these need to be strengthened using mixed-integer rounding (MIR). We refer to these inequalities as m-rounded metric inequalities.

The m-rounded metric inequalities are one of the many classes of the mixed-metric inequalities, which are derived as valid inequalities for the mixed-integer set described by the metric inequality, see Section 6.3.4. However, the m-rounded metric inequalities deserve special attention since the flow cut-set inequality for NDP is a special case; shown as follows.

For a non-empty partition (A, B), let $H \subseteq AB$. Define K' as the set of commodities that have both source and destination nodes s^k and t^k in the same sub-graph (either G'_A or G'_B). Let $K_B = K \setminus (K' \cup K_A)$. To obtain the same variables as in (2.4), we fix flow variables y such that $F^k = AB \setminus H$ for all $k \in K$. Thus, $K_{ij} = K$ if $(ij) \in AB \setminus H$, and $\emptyset$ otherwise. Fixing $u_{ij} = 1$ if $(ij) \in H$, and 0 otherwise, the first term in the left hand sides of (6.41) and the flow cut-set inequality (2.4) are the same.

Since we fix u, the problem decomposes by commodity. For commodity k, the problem thus reduces to maximizing

$$d^k(w^k_{t^k} - w^k_{s^k}) + \sum_{i\in V} w^k_i (\sum_{(ij)\in AB\setminus H} d^k \bar{y}^k_{ij} - \sum_{(ji)\in AB\setminus H} d^k \bar{y}^k_{ji}).$$

Rearranging the second term in the objective, we want to maximize

$$(w^k_{t^k} - w^k_{s^k}) - \sum_{(ij)\in AB\setminus H} \bar{y}^k_{ij}(w^k_i - w^k_j). \tag{6.42}$$

To satisfy (6.39), we are now forced to set $w_j^k = w_i^k$ if $(ij) \notin H$; and $w_j^k - w_i^k \leq 1$, otherwise. Since we are only interested in the difference $w_{t^k}^k - w_{s^k}^k$, without loss of generality, we can set $w_{s^k}^k = w,\ \forall k \in K$. For the rest of this derivation of the optimal solution to (6.42), we assume that the sub-graphs G'_A and G'_B are connected.

Since $0 \leq \sum_{(ij)\in AB\setminus H} \bar{y}_{ij}^k \leq 1,\ \forall k \in K$, and $w_{t^k}^k - w_{s^k}^k = 0,\ \forall k \notin K_A$, the optimal solution to (6.42) is easily obtained. For $k \in K$ such that $s^k \in B$, $w_i^k = w$ for all $i \in V$, and the objective reduces to 0. For $k \in K_A$, (6.42) is maximized by setting $w_i^k = w + 1\ i \in B$, and w otherwise. Then the objective reduces to $1 - \sum_{(ij)\in AB\setminus H} \bar{y}_{ij}^k$. For $k \in K'$ such that $s^k \in A$, the first term in (6.42) reduces to zero. Since the second term is non-positive, we maximize it by setting $w_i^k = w,\ \forall i \in V$.

Thus, the most violated metric inequality with the restriction that $u_{ij} = 1,\ (ij) \in H$, and 0 otherwise, is obtained by setting $w_i^k = w + 1$ if $i \in B, k \in K_A$, and w otherwise; resulting in the metric inequality

$$\sum_{(ij)\in H} x_{[ij]} + \sum_{k\in K_A} d^k \sum_{(ij)\in AB\setminus H} y_{ij}^k \geq \lceil d_A \rceil.$$

By applying MIR, we obtain the flow cut-set inequality (2.4).

Proposition 6.14 For partition (A, B), let $H \in AB$. The flow cut-set inequality for NDP

$$r(d_A) \sum_{(ij)\in H} x_{[ij]} + \sum_{k\in K_A} d^k \sum_{(ij)\in AB\setminus H} y_{ij}^k \geq r(d_A)\lceil d_A \rceil,$$

is obtained by applying MIR to the special case of the metric inequality for NDP (see Definition 6.13) where $u_{ij} = 1,\ (ij) \in H$, and 0 otherwise. □

Interestingly, we assumed in this derivation that all commodities were fixed for arcs $(ij) \in AB$. We would have arrived at the same conclusion by fixing only the commodities in K_A; our derivation proved that the other commodities will not contribute to the flow cut-set inequality even if they were not restricted from doing so.

6.3.2 Metric inequalities for SDC

Assume that we are given a capacity vector $\bar{x} \in \mathbb{Z}^{|E|}$, a flow vector $\bar{y} \in \mathbb{R}^{|\mathcal{F}|}$, and a vector of directed cycle variables $\bar{z} \in \mathbb{R}^{|\hat{\mathcal{C}}|}$. Let $F^k \subseteq F$ be the set of arcs for which the flow

is given, for commodity k; and let $K_{ij} \subseteq K$ be the set of commodities for which flow is given, for arc (ij). For $(\bar{z}, \bar{y}, \bar{x})$, a feasible solution to the SDC exists if and only if there exist $y \in \mathbb{R}_+^{|(F\times K)\setminus\mathcal{F}|}$ and $z \in \mathbb{R}_+^{|\mathcal{C}\setminus\hat{\mathcal{C}}|}$ such that

$$(w_i^k) \quad \sum_{(ij)\in(F\setminus F^k)} d^k y_{ij}^k - \sum_{(ji)\in(F\setminus F^k)} d^k y_{ji}^k = b_i^k - \sum_{(ij)\in F^k} d^k \bar{y}_{ij}^k + \sum_{(ji)\in F^k} d^k \bar{y}_{ji}^k \quad \forall i \in V,\ \forall k \in K \tag{6.43}$$

$$(u_{ij}) \quad \sum_{k\in(K\setminus K_{ij})} d^k y_{ij}^k + \sum_{c\in\mathcal{C}\setminus\hat{\mathcal{C}}} \alpha_{ij}^c z_c \le \bar{x}_{[ij]} - \sum_{k\in K_{ij}} d^k \bar{y}_{ij}^k - \sum_{c\in\hat{\mathcal{C}}} \alpha_{ij}^c \bar{z}_c \quad \forall (ij) \in F \tag{6.44}$$

$$(v_{ij}) \quad \sum_{k\in K} d^k y_{ij}^k - \sum_{c\in\mathcal{C}} \alpha_{ji}^c z_c \le \sum_{c\in\hat{\mathcal{C}}} \alpha_{ji}^c \bar{z}_c - \sum_{k\in K_{ij}} d^k \bar{y}_{ij}^k \quad \forall (ij) \in F, \tag{6.45}$$

where w, u, v are dual variables to constraints (6.43), (6.44), and (6.45), respectively. Thus by Farkas' Lemma, $(\bar{y}, \bar{z}, \bar{x})$ yields a feasible solution to SDC if and only if

$$\sum_{(ij)\in F} u_{ij}(\bar{x}_{[ij]} - \sum_{k\in K_{ij}} d^k \bar{y}_{ij}^k - \sum_{c\in\hat{\mathcal{C}}} \alpha_{ij}^c \bar{z}_c) + \sum_{(ij)\in F} v_{ij}(\sum_{c\in\hat{\mathcal{C}}} \alpha_{ji}^c \bar{z}_c - \sum_{k\in K_{ij}} d^k \bar{y}_{ij}^k) \ge -\sum_{i\in V}\sum_{k\in K} w_i^k (b_i^k - \sum_{(ij)\in F^k} d^k \bar{y}_{ij}^k + \sum_{(ji)\in F^k} d^k \bar{y}_{ji}^k)$$

for all $u \in \mathbb{R}^{|F|}, v \in \mathbb{R}^{|F|}, w \in \mathbb{R}^{|V||K|}$ such that

$$w_j^k - w_i^k \le u_{ij} + v_{ij} \quad \forall k \in K \setminus K_{ij},\ \forall (ij) \in F \tag{6.46}$$

$$\sum_{(ij)\in F} \alpha_{ij}^c (u_{ij} - v_{ji}) \ge 0 \quad \forall c \in \mathcal{C} \tag{6.47}$$

$$u_{ij}, v_{ij} \in \mathbb{R}_+ \quad \forall (ij) \in F. \tag{6.48}$$

Definition 6.15 For all u, v, w that satisfy constraints (6.46)-(6.48),

$$\sum_{(ij)\in F} u_{ij}(x_{[ij]} - \sum_{k\in K_{ij}} d^k y_{ij}^k - \sum_{c\in\hat{\mathcal{C}}} \alpha_{ij}^c z_c) + \sum_{(ij)\in F} v_{ij}(\sum_{c\in\hat{\mathcal{C}}} \alpha_{ji}^c z_c - \sum_{k\in K_{ij}} d^k y_{ij}^k) - \sum_{i\in V}\sum_{k\in K} w_i^k (\sum_{(ij)\in F^k} d^k y_{ij}^k - \sum_{(ji)\in F^k} d^k y_{ji}^k) \ge \sum_{k\in K} d^k (w_{t^k}^k - w_{s^k}^k) \tag{6.49}$$

is a *metric inequality for SDC* with integer and continuous variables. □

Finding the most violated metric inequality (6.49) is equivalent to

$$\max \sum_{k \in K} d^k (w^k_{t^k} - w^k_{s^k}) - \sum_{(ij) \in F} u_{ij} (\bar{x}_{[ij]} + \sum_{k \in K_{ij}} d^k \bar{y}^k_{ij} + \sum_{c \in \hat{\mathcal{C}}} \alpha^c_{ij} \bar{z}_c)$$
$$- \sum_{(ij) \in F} v_{ij} (\sum_{c \in \hat{\mathcal{C}}} \alpha^c_{ji} \bar{z}_c - \sum_{k \in K_{ij}} d^k y^k_{ij}) + \sum_{i \in V} \sum_{k \in K} w^k_i (\sum_{(ij) \in F^k} d^k y^k_{ij} - \sum_{(ji) \in F^k} d^k y^k_{ji})$$

subject to the constraints (6.46)-(6.48). This is a linear program, and can be solved in polynomial time, since the separation problem for inequalities (6.47) is the same as the pricing problem for directed cycle variables in SDC.

When u, v are fixed such that inequalities (6.47) are satisfied for all directed cycles in the network, the problem decomposes for each commodity, and can be solved more efficiently since (6.46) reduce to the dual constraints for a shortest path problem on a directed network (Ahuja et al. 1993). Thus, w are the shortest path labels when the weight on arc (ij) is $u_{ij} + v_{ij}$, and the contribution to the right hand side of (6.49) from commodity k is exactly the weight of the shortest path from source s^k to destination t^k.

Since inequalities (6.49) do not violate any LP relaxation solution, they need to be strengthened using mixed-integer rounding (MIR). We refer to these inequalities as m-rounded metric inequalities.

6.3.3 Metric inequalities for SDP

Assume that we are given a capacity vector $\bar{x} \in \mathbb{Z}^{|E|}$, a flow vector $\bar{y} \in \mathbb{R}^{|\mathcal{F}|}$, and a vector of directed p-cycle variables $\bar{z} \in \mathbb{R}^{|\hat{\mathcal{C}}|}$. Let $F^k \subseteq F$ be the set of arcs for which the flow is given, for commodity k; and let $K_{ij} \subseteq K$ be the set of commodities for which flow is given, for arc (ij). For $(\bar{z}, \bar{y}, \bar{x})$, a feasible solution to SDP exists if and only if there exist

$y \in \mathbb{R}_+^{|(F\times K)\setminus\mathcal{F}|}$ and $z \in \mathbb{R}_+^{|\mathcal{C}\setminus\hat{\mathcal{C}}|}$ such that

$$(w_i^k) \quad \sum_{(ij)\in(F\setminus F^k)} d^k y_{ij}^k - \sum_{(ji)\in(F\setminus F^k)} d^k y_{ji}^k = b_i^k - \sum_{(ij)\in F^k} d^k \bar{y}_{ij}^k + \sum_{(ji)\in F^k} d^k \bar{y}_{ji}^k \qquad \forall i \in V,\ \forall k \in K \tag{6.50}$$

$$(u_{ij}) \quad \sum_{k\in(K\setminus K_{ij})} d^k y_{ij}^k + \sum_{c\in\mathcal{C}\setminus\hat{\mathcal{C}}} \alpha_{ij}^c z_c \le \bar{x}_{[ij]} - \sum_{k\in K_{ij}} d^k \bar{y}_{ij}^k - \sum_{c\in\hat{\mathcal{C}}} \alpha_{ij}^c \bar{z}_c \quad \forall (ij) \in F \tag{6.51}$$

$$(v_{ij}) \quad \sum_{k\in K} d^k y_{ij}^k - \sum_{c\in\mathcal{C}} \alpha_{ji}^c z_c - \sum_{c\in\mathcal{C}\setminus\hat{\mathcal{C}}} \rho_{[ij]}^c z_c \le \sum_{c\in\hat{\mathcal{C}}} \alpha_{ji}^c \bar{z}_c + \sum_{c\in\hat{\mathcal{C}}} \rho_{[ij]}^c \bar{z}_c - \sum_{k\in K_{ij}} d^k \bar{y}_{ij}^k \ \ \forall (ij) \in F, \tag{6.52}$$

where w, u, v are dual variables to constraints (6.50), (6.51), and (6.52), respectively. Thus by Farkas' Lemma, $(\bar{y}, \bar{z}, \bar{x})$ yields a feasible solution to SDP if and only if

$$\sum_{(ij)\in F} u_{ij}(\bar{x}_{[ij]} - \sum_{k\in K_{ij}} d^k \bar{y}_{ij}^k - \sum_{c\in\hat{\mathcal{C}}} \alpha_{ij}^c \bar{z}_c) + \sum_{(ij)\in F} v_{ij}(\sum_{c\in\hat{\mathcal{C}}} \alpha_{ji}^c \bar{z}_c + \sum_{c\in\hat{\mathcal{C}}} \rho_{[ij]}^c \bar{z}_c - \sum_{k\in K_{ij}} d^k \bar{y}_{ij}^k) \ge -\sum_{i\in V}\sum_{k\in K} w_i^k (b_i^k - \sum_{(ij)\in F^k} d^k \bar{y}_{ij}^k + \sum_{(ji)\in F^k} d^k \bar{y}_{ji}^k)$$

for all $u \in \mathbb{R}^{|F|}, v \in \mathbb{R}^{|F|}, w \in \mathbb{R}^{|V||K|}$ such that

$$w_j^k - w_i^k \le u_{ij} + v_{ij} \quad \forall k \in K \setminus K_{ij},\ \forall (ij) \in F \tag{6.53}$$

$$\sum_{(ij)\in F} \alpha_{ij}^c (u_{ij} - v_{ji}) - \sum_{(ij)\in F} \rho_{[ij]}^c v_{ij} \ge 0 \qquad \forall c \in \mathcal{C} \tag{6.54}$$

$$u_{ij}, v_{ij} \in \mathbb{R}_+ \qquad \forall (ij) \in F. \tag{6.55}$$

Definition 6.16 For all u, v, w that satisfy constraints (6.53)-(6.55),

$$\sum_{(ij)\in F} u_{ij}(x_{[ij]} - \sum_{k\in K_{ij}} d^k y_{ij}^k - \sum_{c\in\hat{\mathcal{C}}} \alpha_{ij}^c \bar{z}_c) + \sum_{(ij)\in F} v_{ij}(\sum_{c\in\hat{\mathcal{C}}} \alpha_{ji}^c z_c + \sum_{c\in\hat{\mathcal{C}}} \rho_{[ij]}^c \bar{z}_c - \sum_{k\in K_{ij}} d^k y_{ij}^k) - \sum_{i\in V}\sum_{k\in K} w_i^k (\sum_{(ij)\in F^k} d^k y_{ij}^k - \sum_{(ji)\in F^k} d^k y_{ji}^k) \ge \sum_{k\in K} d^k (w_{t^k}^k - w_{s^k}^k) \tag{6.56}$$

is a *metric inequality for SDP* with integer and continuous variables. □

Finding the most violated metric inequality (6.56) is equivalent to

$$\max \sum_{k \in K} d^k (w^k_{t^k} - w^k_{s^k}) - \sum_{(ij) \in F} u_{ij} (\bar{x}_{[ij]} - \sum_{k \in K_{ij}} d^k \bar{y}^k_{ij}) + \sum_{i \in V} \sum_{k \in K} w^k_i (\sum_{(ij) \in F^k} d^k \bar{y}^k_{ij} - \sum_{(ji) \in F^k} d^k \bar{y}^k_{ji})$$

subject to the constraints (6.53)-(6.55). This is a linear program. However, constraints (6.54) are exponential in number, and their separation is exactly the same as the pricing problem for directed p-cycle variables in SDP, which is $\mathcal{NP}$-hard (Theorem 5.2). Thus, finding the most violated inequality is $\mathcal{NP}$-hard.

When u, v are fixed such that constraints (6.54) are satisfied, the problem decomposes for each commodity, and can be solved more efficiently since (6.39) are the dual constraints for a shortest path problem on a directed network (Ahuja et al. 1993). Then, w are the shortest path labels when the weight on arc (ij) is $u_{ij} + v_{ij}$, and the contribution to the right hand side of (6.56) from commodity k is exactly the weight of the shortest path from source s^k to destination t^k.

Since (6.49) do not violate any LP relaxation solution, these need to be strengthened using mixed-integer rounding (MIR). We refer to these strengthened inequalities as m-rounded metric inequalities.

6.3.4 Mixed-metric inequalities

We can develop new metric-type inequalities from metric inequalities (6.41), (6.49), and (6.56) by generating valid inequalities for the mixed-integer knapsack set described by the metric inequality. In Section 7.5.2, we describe how this can be done, and call these new inequalities mixed-metric inequalities. The m-rounded metric inequalities are one of the many classes of mixed-metric inequalities that can be obtained by this procedure. Obtaining the most violated mixed-metric inequality is $\mathcal{NP}$-hard, since the separation problem for even cut-set inequalities of NDP is known to be $\mathcal{NP}$-hard (Bienstock 2001).

6.4 Computational results

In this section, we present computational results that illustrate the effectiveness of the metric-type inequalities in reducing the computational effort in solving various network design problems. It is known that the cut-set inequalities improve solution times of NDP and SDC by more than an order of magnitude, see (Atamtürk 2002) and Section 4.3. These works also indicate that the most computationally effective inequalities are the cut-set or partition inequalities involving only the integer capacity variables. Therefore, we focus on SDP for our experiments, and study the effect of adding the strengthened cut-set inequalities (6.35) and the cardinality-k cut-set inequalities (6.34) in a branch-and-cut framework.

We use the same networks as in Sections 4.3 and 5.4; i.e., with 50% demand density and 75% link density, chosen at random. The branch-and-cut algorithm is implemented using CPLEX Version 8.1 Callable Library on an Intel Pentium4 2GHz Linux workstation with 1GB RAM. Each instance was run for one hour of CPU time, and the best feasible solution recovered if the optimum is not found. Strengthened cut-set and cardinality-k cut-set inequalities for certain partitions were pre-generated, and added to the formulation if violated, at root node. We repeat the experiments with these metric-type inequalities, and without; CPLEX default cuts are added for both.

In Table 6.1, we report the improvement of the integrality gap at the root node (root improvement), the number of branch-and-bound nodes (b&b nodes), and the solution times (time) or gap at termination (endgap), for runs with and without the metric-type cuts. The results for experiments using only the CPLEX cuts are reported under heading (1) and results for experiments using both CPLEX and metric-type cuts are reported under heading (2).

We see in Table 6.1 that adding strengthened cut-set and cardinality-k cut-set inequalities more than doubles the improvement in the integrality gap at root node. On average, the metric-type inequalities improve the integrality gap at the root node by 59% on average, as compared with 22% for CPLEX default. This naturally manifests itself in

Table 6.1: Performance of strengthened and cardinality-k cut-set inequalities

Size	root improvement		b&b nodes		time (endgap)	
$\|V\|$	(1)	(2)	(1)	(2)	(1)	(2)
6	14	70	155	29	0.11	0.04
7	17	69	698	96	0.60	0.16
8	32	53	2171	375	2.19	0.53
9	47	73	1017	276	1.22	0.50
10	17	68	3769	432	4.48	0.71
11	27	54	89669	8291	275	34.6
12	12	43	40916	4882	90.3	13.2
13	17	57	282470	10343	793	37.2
14	15	62	263989	2870	762	11.2
15	18	58	714969	21743	3451	135
16	22	57	483079	465627	(1.8)	3466
17	20	46	242806	262387	(1.6)	(1.0)

the computational performance of the branch-and-cut algorithm; we obtain an order of magnitude reduction in the number of branch-and-bound nodes and solution times.

6.5 Conclusions

In this chapter, we developed metric-type inequalities for various network design problems using feasibility conditions derived from Farkas' Lemma. Since these inequalities do not violate any fractional solutions, we discussed techniques to strengthen them using integer and mixed-integer rounding. We also developed new metric-type inequalities using results from the polyhedron of the mixed-integer knapsack set. We showed that many classes of partition inequalities for NDP and SDC are special cases of these strengthened metric inequalities. Using similar techniques, we developed new partition inequalities for SDP. Finally, we presented computational results that illustrated the effect of these new partition inequalities in solving SDP. When added as cutting planes in branch-and-cut algorithm, these inequalities improved the computations by an order of magnitude.

This line of research using metric inequalities is by no means complete. We would like to develop theoretical results on the strength of these new partition inequalities. We also wish to develop other known inequalities as special cases of the metric-type inequalities,

whenever possible. Finally, we would like to generalize this technique into a general framework for developing metric-type inequalities for all network design problems, with or without survivability requirements.

Chapter 7

New valid inequalities for mixed-integer knapsack sets

In this chapter, we present new inequalities for the mixed-integer knapsack set. First, we analyze the mixed-integer knapsack set with two integer variables and one continuous variable, and develop a polynomial algorithm that enumerates all the facets in its convex hull. Then, we study the exact lifting function for the facets of the convex hull of this set, and describe super-additive lower bounds for it. These super-additive lower bounds are obtained from partial LP relaxations of the exact lifting function, and used in a sequence independent lifting framework to develop strong valid inequalities for the mixed-integer knapsack polyhedron. We present some sufficient conditions under which these lifted inequalities define facets of mixed-integer knapsack sets with at most one continuous variable. Finally, we summarize our computational experience with these inequalities.

7.1 Introduction

Let N be the index set of integer variables, and P the index set of continuous variables. The mixed-integer knapsack set can be represented as

$$M(b) = \{x \in \mathbb{Z}_+^{|N|}, w \in \mathbb{R}_+^{|P|} \ : \ \sum_{i \in N} a_i x_i + \sum_{i \in P} g_i w_i \leq b, \ x_i \leq u_i, \ i \in N, \ w_i \leq v_i, \ i \in P\}.$$

We assume that the data is rational, except that v_i may be infinite; u_i are finite. No assumptions are made on the sign of a,g, and b. Defining $g_i w_i$ as a new continuous variable, without loss of generality we assume that $g_i \in \{-1, 1\}$ $i \in P$. Let $P^+ = \{i \in P : g_i = 1\}$, $P^- = P \setminus P^+$, and $P_B = \{i \in P : v_i < \infty\}$. Finally, by complementing variables if necessary, we can assume without loss of generality that $P_B \subseteq P^+$.

Since the optimization problem over all points satisfying any given constraint of a mixed-integer program is its relaxation, any inequalities derived for the mixed-integer set are valid, and can be used in a branch-and-cut framework to solve the mixed-integer program; many successful attempts have been made to study several special cases of the mixed-integer knapsack set.

Most seminal works have dealt with the pure-binary knapsack set, see Balas (1975), Hammer et al. (1975), Wolsey (1975), and Zemel (1989) for a few examples. More recently, others have studied the mixed-binary knapsack set (Marchand and Wolsey 1999, Richard et al. 2002), the binary knapsack set with one integer variable (Brockmüller et al. 1996, Atamtürk and Rajan 2002, van Hoesel et al. 2002), and the pure-integer knapsack set (Pochet and Wolsey 1995, Pochet and Weismantel 1998). In Ceria et al. (1998), the authors extend the theory of knapsack covers to the integer knapsack set.

For a common presentation of recent research in cover (and pack) inequalities, and new results for integer knapsack sets, see Atamtürk (2003a). However there is not much work on the mixed-integer knapsack set, except the works on mixed-integer cuts (Gomory 1960), or the equivalent mixed-integer rounding cuts (Nemhauser and Wolsey 1990).

An alternative methodology for generating valid inequalities utilizes super-additive functions; see Section 1.5.2. In this approach, one studies the shape of the lifting functions of facet-defining inequalities of various restrictions of the set being analyzed. If these are super-additive, then it is known that the restricted variables can be lifted independent of sequence to obtain facet-defining inequalities for the set (Atamtürk 2004). This approach has been successfully adapted in obtaining facet-defining inequalities of certain mixed-binary sets (Gu et al. 1999, Marchand and Wolsey 1999).

In Atamtürk (2003b), the author extends this approach to the mixed-integer knapsack

set. He studies the restriction with exactly one integer variable and one continuous variable, obtained by fixing all other variables to zero. The facet-defining inequality for this restriction is the well-known mixed-integer rounding (MIR) inequality, see Section 1.5.2. The author analyzes the lifting function for this inequality, and shows that it is super-additive under certain conditions. Many known inequalities for special structured sets are seen to be special cases of the inequalities in Atamtürk (2003b) that use super-additive exact lifting functions. In general, however, the exact lifting function is not super-additive; the author presents lower bounds that may be used instead.

7.1.1 Motivation

Theorem 7.1 (Atamtürk (2003b)) Any non-trivial facet-defining inequality $\pi x + \mu w \leq \pi_0$ of $\mathrm{conv}(M(b))$ satisfies $\mu_i = 0,\ \forall i \in P^+ \setminus P_B$, and $\mu_i = \alpha,\ \forall i \in P^-$, for some $\alpha \in \mathbb{R}_-$.

In other words, all continuous variables in $P^+ \setminus P_B$ have coefficient equal to zero in any facet-defining inequality of $\mathrm{conv}(M(b))$. This allows us to ignore these variables while studying the polyhedron of the set $M(b)$. Furthermore, all continuous variables in P^- have the same coefficient in all facets of $\mathrm{conv}(M(b))$. Thus, we can aggregate these variables into a single non-negative unbounded continuous variable.

Obtaining facets of the mixed-integer knapsack polyhedron with a small number of integer variables and lifting them could possibly lead to new classes of strong inequalities for the mixed-integer knapsack polyhedron. One strong reason to suggest this is that few of the facets of the convex hull of even the restriction with two integer variables and one continuous variable can be obtained by the procedure studied by Atamtürk (2003b). Formally, this restriction is defined as

$$M_2^{\leq}(b) = \{x_1, x_2 \in \mathbb{Z}_+\ y \in \mathbb{R}_+ \,:\, a_1x_1 + a_2x_2 - y \leq b,\ x_1 \leq u_1,\ x_2 \leq u_2\}.$$

The following result allows us to obtain facets of $\mathrm{conv}(M(b))$ in terms of facets of $\mathrm{conv}(M_n^{\leq}(b'))$, for specific values of b'.

Theorem 7.2 Let $H \subseteq P^+$ and $n = |N|$. If $\pi x - y \leq \pi_0 - v(H)$ defines a facet of $\mathrm{conv}(M_n^{\leq}(b - v(H)))$, then $\pi x + w(H) - w(P^-) \leq \pi_0$ defines a facet of $\mathrm{conv}(M(b))$.

Proof See Appendix F. □

Furthermore, lifting a facet-defining inequality $\pi x + w(H) - w(P^-) \leq \pi_0$ over $M(b)$ is equivalent to lifting $\pi x - y \leq \pi_0 - v(H)$ over $M_n^{\leq}(b - v(H))$; by the variable substitution $y = w(P^-) + v(H) - w(H)$. This motivates our focus on the restriction with two integer variables and one continuous variable ($M_2^{\leq}(b)$) in this study.

The converse of Theorem 7.2 is true when $n = 1$ (Atamtürk 2003b). In fact, when $n = 1$, $\mathrm{conv}(M(b))$ can be completely described and separated in linear time for all $|P|$ (Magnanti et al. 1993, Atamtürk and Rajan 2002). However, for any $n > 1$, no closed-form description of $\mathrm{conv}(M(b))$ is known when $P^+ \neq \emptyset$. The optimization of a linear function over $M(b)$ can be done in polynomial time, for fixed n (Lenstra Jr. 1983).

Independently, Agra and Constantino (2003) develop valid inequalities for pure and mixed-integer knapsack sets by studying a restriction with two integer variables and one continuous variable. However, they do not consider upper bounds to the variables in the set. Their work differs from ours primarily in that they use a group relaxation of the exact lifting function to derive super-additive lower bounds.

7.1.2 Notation

Throughout this chapter, we will use $M_n^{\leq}(b)$ to refer to the mixed-integer knapsack set with n integer variables, one continuous variable, and right hand side b; $K_n^{\leq}(b)$ ($K_n^{\geq}(b)$) to refer to the pure-integer knapsack set with n variables, a "$\leq$" ("$\geq$") constraint, and right hand side b.

The instance of $K_n^{\geq}(b)$ parametrized by (a, u) is the same as the instance of $K_n^{\leq}(\sum_{i=1}^n a_i u_i - b)$ parametrized by (a, u); obtained by the variable substitution $x_i' = u_i - x_i,\ i \in [1, n]$. We introduce $K_n^{\leq}(b)$ purely for ease of exposition.

For any $p \in \mathbb{Z}_+^n \times \mathbb{R}_+$, we denote its integral components by $p_i,\ i \in [1, n]$ and continuous component by p_0. We denote the components of any element $q \in \mathbb{Z}^n$ by $q_i,\ i \in [1, n]$. We assume that $a, u, \pi \geq 0$; by complementing variables, if necessary. Furthermore, for $K_n^{\leq}(b)$, we assume without loss of generality that $a_i \leq b,\ i \in [1, n]$; since otherwise $q_i = 0$ in all feasible solutions.

7.1.3 Outline

In Section 7.2, we study the pure-integer restriction $K_2^{\leq}(b)$ with two integer variables. We first present a bound on the number of extreme points in its convex hull, and then present a polynomial-time algorithm that enumerates these points. In Kannan (1980), the author presents an algorithm to optimize a linear function over $K_2^{\leq}(b)$. Our algorithm calls Kannan's algorithm once, and has the same complexity bound. These results serve as crucial building blocks for the study of $M_2^{\leq}(b)$ in Section 7.3, where we present a key result that allows us to enumerate all extreme points (and hence facet-defining inequalities) of $\mathrm{conv}(M_2^{\leq}(b))$ in polynomial-time.

In Section 7.5, we study the exact lifting function for the facet-defining inequalities of $\mathrm{conv}(K_2^{\leq}(b))$ and $\mathrm{conv}(M_2^{\leq}(b))$, and describe lower bounds for them using their partial LP relaxations. We generate super-additive lower bounds for the exact lifting function from special cases of the two families of super-additive functions introduced in Section 7.4. The super-additive lifting functions are then used to define valid inequalities to the the pure-integer knapsack set $K_n^{\leq}(b)$ and the mixed-integer knapsack set $M_n^{\leq}(b)$. In Section 7.6, we present some computational results that illustrate the effectiveness of these inequalities. We conclude in Section 7.7 by summarizing the main results of the chapter.

7.2 Two integer variables

We first study the restriction of $M_n^{\leq}(b)$ with two integer variables and no continuous variable; denoted by $K_2^{\leq}(b)$. This is a simpler structure than $M_2^{\leq}(b)$; yet it provides significant insight toward the polyhedral structure of $M_2^{\leq}(b)$. Let

$$K_2^{\leq}(b) = \{x_1, x_2 \in \mathbb{Z}_+ \,:\, a_1x_1 + a_2x_2 \leq b,\ x \leq u\}.$$

We state the assumptions made in this section. Some of them are repeated in later sections, and are referred to using the labels introduced here.

(A.0) We assume that all data is integral, and $\gcd(a1, a2) = 1$, by scaling.

(A.1) We assume that $a_1 \geq a_2$, by reordering the variables, if necessary.

(A.2) We assume that $a > 0$, $u > 0$.

(A.3) We assume that $a_1 + a_2 < b < a_1u_1 + a_2u_2$.

(A.4) We assume without loss of generality that $u_i \leq \lfloor b/a_i \rfloor$ $i = 1, 2$, since we can replace u_i by $\lfloor b/a_i \rfloor$ otherwise.

Assumptions (A.2) and (A.3) are made to eliminate trivial cases. Non-zero lower bounds can be easily handled by redefining variables as usual. We define $t_1 = (b - a_2u_2)/a_1$ and $t_2 = (b - a_1u_1)/a_2$. Assumption (A.4) can be strengthened to

(A.4') We assume that $u_1 = \lfloor b/a_1 \rfloor$ and $u_2 = \lfloor b/a_2 \rfloor$.

To see why this is true, we first assume the contrary, and then propose a translation that results in $u_i = \lfloor b/a_i \rfloor$ $i = 1, 2$. In Figure 7.1, we present the set $K_2^{\leq}(b)$; t_1 and t_2 are the intercepts of $a_1x_1 + a_2x_2 \leq b$ at the upper bound constraints. For $i \in [1, 2]$, let q^i denote the solution of the axis $x_i = 0$ and the upper bound constraint $x_{2-i} = u_{2-i}$; thus $q^1_1 = 0, q^1_2 = u_2$, and $q^2_1 = u_1, q^2_2 = 0$. Let q^0 be the solution of the upper bound constraint $x_2 = u_2$ and $x_1 = \lfloor t_1 \rfloor$, and define q^3 similarly. Thus, we have $q^0_1 = \lfloor t_1 \rfloor, q^0_2 = u_2$, and $q^3_1 = u_1, q^3_2 = \lfloor t_2 \rfloor$.

Let $\bar{q}$ be the origin; i.e., $\bar{q}_1 = \bar{q}_2 = 0$. We also define q^4 as the point $q^4_1 = \lceil t_1 \rceil, q^4_2 = u_2 - 1$. Since $a_1 \geq a_2$ by assumption (A.1), using $\lfloor t_1 \rfloor$ may not ensure that the upper bound constraint $x_2 \leq u_2$ is dominated; we use $\lceil t_1 \rceil$ instead. Thus, the variable substitution $x'_1 = x_1 - \lceil t_1 \rceil$, $x'_2 = x_2 - \lfloor t_2 \rfloor$ results in a smaller set with redundant upper bound constraints; see Figure 7.1.

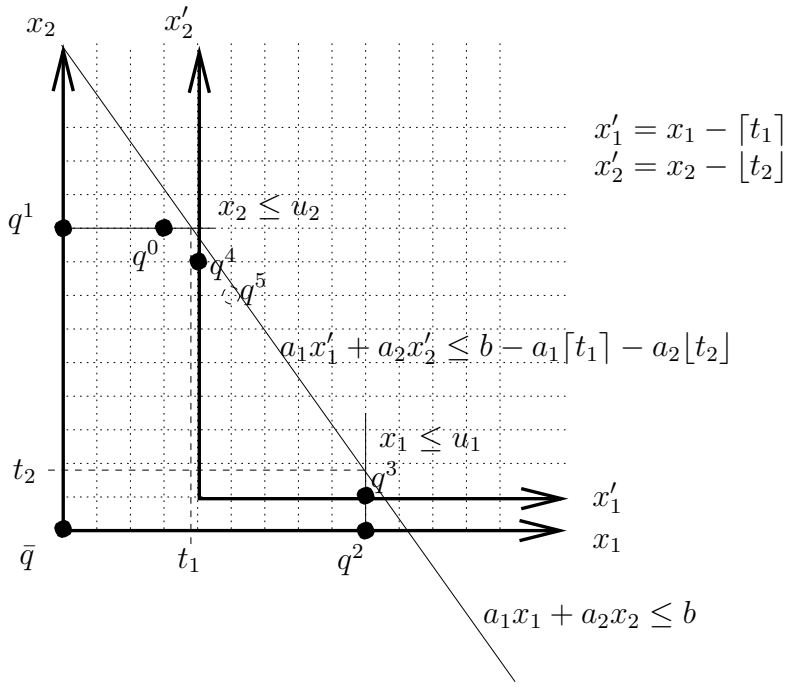
Some of the extreme points and facet-defining inequalities of $\mathrm{conv}(K_2^{\leq}(b))$ are thus lost in the translation from $K_2^{\leq}(b)$ to K'; illustrated in Figure 7.1. We list these points and facet-defining inequalities in the following propositions.

Proposition 7.3 The inequalities $x_1 \geq 0$, $x_2 \geq 0$ define facets of $\mathrm{conv}(K_2^{\leq}(b))$. Upper bound constraints $x_1 \leq u_1$ and $x_2 \leq u_2$ define facets of $\mathrm{conv}(K_2^{\leq}(b))$ if and only if $t_2 \geq 1$ and $t_1 \geq 1$, respectively.

Proof See Appendix F. □

Proposition 7.4 The points q^1, q^2, $\bar{q}$, q^0 (distinct from q^1 if $t_1 > 1$), and q^3 (distinct from q^2 if $t_2 > 1$) are extreme points of $\mathrm{conv}(K_2^{\leq}(b))$.

Figure 7.1: Sets $K_2^{\leq}(b)$ and K'



Proof See Appendix F. □

Proposition 7.5 The point q^4 is an extreme point of $\text{conv}(K_2^{\leq}(b))$ if and only if $q^5 \notin K_2^{\leq}(b)$; where q^5 is defined as $q_1^5 = q_1^4 + 1$ and $q_2^5 = q_2^4 - (q_2^0 - q_2^4)$.

Proof See Appendix F. □

The remaining extreme points (and facet-defining inequalities) can be obtained by studying the set $K' = \{x_1', x_2' \in \mathbb{Z}_+ : a_1x_1' + a_2x_2' \leq b - a_1\lceil t_1 \rceil - a_2\lfloor t_2 \rfloor\}$. Thus, assumption (A.4') can be made without loss of generality.

We present a polynomial bound in Section 7.2.1 on the number of extreme points of $\text{conv}(K_2^{\leq}(b))$, and a polynomial algorithm in Section 7.2.2 to enumerate all extreme points of $\text{conv}(K_2^{\leq}(b))$. The algorithm is also easily modified to enumerate the facets of $\text{conv}(K_2^{\leq}(b))$, since it gives us the extreme points in sequence. This algorithm calls the algorithm presented by Kannan (1980) (which optimizes a linear function over $K_2^{\leq}(b)$) once, and then solves a sequence of diophantine approximations. However, our algorithm still has the same complexity bound as in Kannan (1980).

Alternatively, Kannan's algorithm can be extended to enumerate all the extreme points of $\mathrm{conv}(K_2^{\leq}(b))$ in sequence. This extension is due to comprehensive book-keeping, resulting in an algorithm that is not easy to describe. Nevertheless, this variant provides a constructive proof for an alternative polynomial bound on the number of extreme points of $K_2^{\leq}(b)$. In Weismantel (1995), the author presents an enumerative algorithm for obtaining on the facets of $\mathrm{conv}(K_2^{\leq}(b))$ using related Hilbert bases. It is closely related to the polynomial vertex enumeration algorithm introduced in Section 7.2.2; however, the author performs no complexity analysis of his algorithm.

7.2.1 Number of extreme points

Definition 7.6 Extreme point q of $\mathrm{conv}(K_2^{\leq}(b))$ is *non-trivial* if $q > 0$; extreme point q of $\mathrm{conv}(K_2^{\geq}(b))$ is *non-trivial* if $q < u$. □

We use $\pi_1 x_1 + \pi_2 x_2 \leq \pi_0$ to denote facets of $\mathrm{conv}(K_2^{\leq}(b))$; by scaling, we assume without loss of generality that $\pi_i \in \mathbb{Z}\ i = [0, 2]$. Let $\gamma = \min\{a_2 - 1, u_1, u_2\}$. We define extreme points q^D and q^U of $\mathrm{conv}(K_2^{\geq}(b))$ as follows; ϵ is a small strictly positive constant.

Definition 7.7 Let $q^D = \arg\max_x\{(a_1 + \epsilon)x_1 + a_2 x_2 \ : \ x \in K_2^{\leq}(b)\}$, and $q^U = \arg\max_x\{a_1 x_1 + (a_2 + \epsilon)x_2 \ : \ x \in K_2^{\leq}(b)\}$. □

Then, for all the facets of $\mathrm{conv}(K_2^{\leq}(b))$ defined by extreme points q^a and q^b such that $q_1^a \leq q_1^U$ and $q_1^b \leq q_1^U$, we have $\pi_1/\pi_2 < a_1/a_2$; see Figure F.1. Similarly, for all the facets of $\mathrm{conv}(K_2^{\leq}(b))$ defined by extreme points q^a and q^b such that $q_1^a \geq q_1^D$ and $q_1^b \geq q_1^U$, we have $\pi_1/\pi_2 > a_1/a_2$.

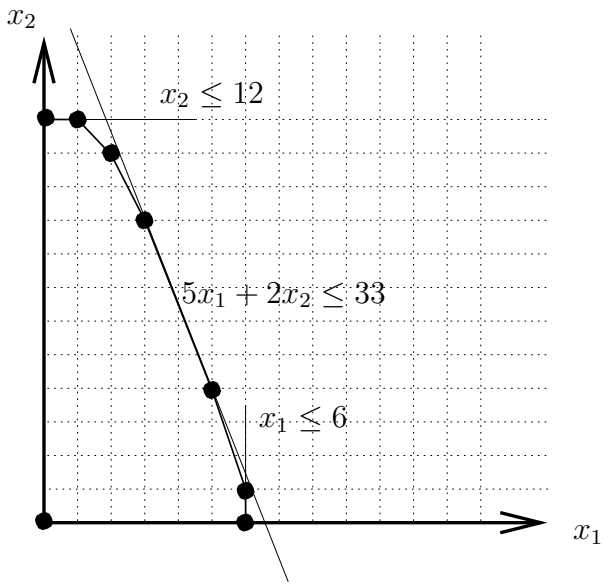
Theorem 7.8 The number of non-trivial extreme points of $K_2^{\leq}(b)$ is bounded from above by $2\lfloor \log(\gamma + 1) \rfloor + 2$.

Proof See Appendix F. □

Since we assumed that (A.4') is true, in general, we also have to include the points lost in the translation that validates this assumption: q^1, q^2, $\bar{q}$ and q^0 (if distinct from q^1); see Figure 7.1. Thus, we can bound from above the number of extreme points of $\mathrm{conv}(K_2^{\leq}(b))$ by $2\lfloor \log(\gamma + 1) \rfloor + 6$, where γ is defined for the reduced set K'.

This bound is often weak; however, tight examples exist, even with small coefficients. For instance, the convex hull of the following instance of $K_2^{\leq}(33)$ has eight extreme points, see Figure 7.2. When $K_2^{\leq}(33) = \{x_1, x_2 \in \mathbb{Z}_+ \,:\, 5x_1 + 2x_2 \leq 33,\ x_1 \leq 6,\ x_2 \leq 12\}$, we have $t_1 = 1.8, t_2 = 1.5$, and the reduced set $K' = \{x_1, x_2 \in \mathbb{Z}_+ \,:\, 5x_1 + 2x_2 \leq 21,\ x_1 \leq 4,\ x_2 \leq 10\}$. Thus, $\gamma = 1$; therefore the bound is tight.

Figure 7.2: Tight example



7.2.2 Polynomial vertex enumeration algorithm

Next, we present a polynomial-time algorithm that enumerates all the extreme points of $\mathrm{conv}(K_2^{\leq}(b))$. This algorithm first calls the algorithm by Kannan (1980), and then solves a polynomial number of diophantine approximations.

7.2.2.1 Diophantine approximation

Definition 7.9 The *diophantine approximation* is the optimization problem

$$\Omega(c, d, \kappa) = \min_{\delta_1, \delta_2} \{|\frac{c}{d} - \frac{\delta_1}{\delta_2}| \,:\, \delta_1, \delta_2 \in \mathbb{Z}_{++},\ \delta_2 \leq \kappa\}, \tag{7.1}$$

where $c, d, \kappa > 0$.

Since c and d are chosen arbitrarily, we assume without loss of generality that $c \geq d$; thus $\delta_1 \geq \delta_2$. We define two problems that require the lower and upper approximations, and denote the solutions by $\underline{\Omega}(c, d, \kappa)$ and $\overline{\Omega}(c, d, \kappa)$. Formally, $\underline{\Omega}(c, d, \kappa) = \min_{\delta_1, \delta_2}\{\frac{c}{d} - \frac{\delta_1}{\delta_2} \ : \ \delta_1, \delta_2 \in \mathbb{Z}_{++}, \ \delta_2 \leq \kappa, \ \frac{\delta_1}{\delta_2} \leq \frac{c}{d}\}$; and $\overline{\Omega}(c, d, \kappa) = \min_{\delta_1, \delta_2}\{\frac{\delta_1}{\delta_2} - \frac{c}{d} \ : \ \delta_1, \delta_2 \in \mathbb{Z}_{++}, \ \delta_2 \leq \kappa, \ \frac{\delta_1}{\delta_2} \geq \frac{c}{d}\}$.

We describe how the diophantine approximation problem can be solved in polynomial time, (Nemhauser and Wolsey 1988, Section I.7). First, we run the Euclidean algorithm on c and d. The Euclidean algorithm computes the greatest common divisor of two given integers. As a by-product, it gives us a sequence of positive integers $\{s_i, t_i, \ i \in [1, T]\}$, such that $\gcd(c, d) = -ct_i + ds_i, \ i \in [1, T]$. Since $c \geq d$, we can assume that $s_i \geq t_i, \ i \in [1, T]$. Let $\mathbb{F}_i$ be the i^{th} number of the Fibonacci series, which grows exponentially fast. Since $t_T \geq \mathbb{F}_{T-1}$, the Euclidean algorithm runs in $\mathcal{O}(\log d)$ time.

This sequence of numbers also define a sequence of ratios $s_i/t_i, \ i \in [1, T]$ that are successively stronger approximations to c/d. In fact, any two adjacent terms in this sequence provide lower and upper approximations to c/d; i.e., either $s_i/t_i \leq c/d \leq s_{i+1}/t_{i+1}$ or $s_i/t_i \geq c/d \geq s_{i+1}/t_{i+1}$. Then, the optimal solution to (7.1) can be obtained from this sequence, as described by Proposition 7.10.

Proposition 7.10 (Nemhauser and Wolsey (1988)) Let $j = \max\{i \ : \ t_i \leq \kappa\}$ and $k = \lfloor(\kappa - t_{j-1})/t_j\rfloor$. Then $\Omega(c, d, \kappa)$ is either s_j/t_j or $(ks_j + s_{j-1})/(kt_j + t_{j-1})$. □

If $s_j/t_j \geq c/d$, then $(ks_j + s_{j-1})/(kt_j + t_{j-1}) \leq c/d$, and vice versa. Thus, Proposition 7.10 also describes the solutions to $\underline{\Omega}(c, d, \kappa)$ and $\overline{\Omega}(c, d, \kappa)$.

7.2.2.2 Description of algorithm

We begin by solving $q^* = \arg\max_x\{a_1x_1 + a_2x_2 \ : \ a_1x_1 + a_2x_2 \leq b, \ x \in \mathbb{Z}^2_+\}$ in $\mathcal{O}(\log a_2)$ time using the algorithm in Kannan (1980).

The point q^* is on some facet of $\text{conv}(K^{\leq}_2(b))$ such that $q^U_1 \leq q^*_1 \leq q^D_1$; see Definition 7.7. Starting with q^*, we obtain all extreme points of $\text{conv}(K^{\leq}_2(b))$ by solving a sequence of diophantine approximations.

First, we enumerate the extreme points q such that $q_1 \leq q_1^U$. as follows. The point q^{i-1} can be obtained from q^i. Let $\delta^i = \arg\min \underline{\Omega}(c_i, d_i, \kappa_i)$, where $c_i = a_1$, $d_i = a_2$, and $\kappa_i = q_1^i$. Then,

$$q_1^{i-1} = r(q_1^i, \delta_2^i), \;\; q_2^{i-1} = (q_1^i - q_1^{i-1})\frac{\delta_1^i}{\delta_2^i} + q_2^i.$$

If $q_1^{i-1} = q_1^i - \delta_2^i$, then $q_2^{i-1} = q_2^i + \delta_1^i$. This operation is repeated until $q_1^{i-1} = 0$.

Next, given extreme point q^i such that $q_1^i \geq q_1^D$, we can obtain q^{i-1}. Let $\delta^i = \arg\min \overline{\Omega}(c_i, d_i, \kappa_i)$, where $c_i = a_1$, $d_i = a_2$, and $\kappa_i = \lfloor b/a_1 \rfloor - q_1^i$. Then,

$$q_2^{i-1} = r(q_2^i, \delta_1^i), \;\; q_1^{i-1} = (q_2^i - q_2^{i-1})\frac{\delta_2^i}{\delta_1^i} + q_1^i.$$

If $q_2^{i-1} = q_2^i - \delta_1^i$, then $q_1^{i-1} = q_1^i + \delta_2^i$. This operation is repeated until $q_2^{i-1} = 0$. Next, we discuss the complexity of this algorithm.

From Theorem 7.8, the number of extreme points of $\mathrm{conv}(K_2^{\leq}(b))$ is $\mathcal{O}(\log a_2)$, since $a_2 \geq \gamma$ by definition. For each extreme point, we need to solve (7.1) once.

The first two parameters of the optimization problems $\overline{\Omega}$ and $\underline{\Omega}$ never change, since $c_i = a_1, d_i = a_2$. This implies that we need to execute the Euclidean algorithm only once, to obtain the sequence $\{s_i/t_i,\ i \in [1, T]\}$.

On the other hand, κ_i changes with i. However, κ_i is monotonic in i; decreasing when $q_1^i \leq q_1^U$ and when $q_1^i \geq q_1^D$. We calculate the extreme points from both sets simultaneously by comparing the next κ from the two sets, and first solving (7.1) for the set with larger κ. Thus, we need to traverse the list $\{t_i,\ i \in [1, T]\}$ only once to calculate $j = \max\{i : t_i \leq \kappa\}$ for all extreme points.

Given j, (7.1) can be solved in constant time. Thus, all extreme points of $\mathrm{conv}(K_2^{\leq}(b))$ can be enumerated in $\mathcal{O}(\log a_2)$ if we are given q^* and $\{s_i, t_i,\ i \in [1, T]\}$. We store the extreme points obtained from the two sets in separate lists since we want them in sequence.

The point q^* can be obtained in $\mathcal{O}(\log a_2)$ by the algorithm in Kannan (1980). Furthermore, the Euclidean algorithm runs in $\mathcal{O}(\log a_2)$ time. We have proved the following.

Proposition 7.11 All extreme points of $K_2^{\leq}(b)$ can be enumerated in $\mathcal{O}(\log a_2)$ time. □

7.2.2.3 Enumerating facets of $\mathrm{conv}(K_2^{\leq}(b))$ in polynomial time

The algorithm can be easily extended to enumerate the facets of $\mathrm{conv}(K_2^{\leq}(b))$. Since it obtains adjacent extreme points in sequence, we need to compute the equalities that are defined by adjacent pairs of extreme points; this can be done in constant time per pair, leading to an $\mathcal{O}(\log a_2)$ algorithm to enumerate all facets.

Obtaining these extreme points in sequence also allows us to compute the facets of $\mathrm{conv}(M_2^{\leq}(b))$ much more efficiently; in Section 7.3.2.

7.3 Two integer variables and one continuous variable

In this section, we study the mixed-integer set with two integer variables and one continuous variable

$$M_2^{\leq}(b) = \{x_1, x_2 \in \mathbb{Z}_+\, y \in \mathbb{R}_+ \,:\, a_1x_1 + a_2x_2 - y \leq b,\; x_1 \leq u_1,\; x_2 \leq u_2\}.$$

We repeat assumptions (A.0)-(A.1) from Section 7.2. We also make assumption (A.2); else the set reduces to the set studied by Atamtürk (2003b). Again, note that non-zero lower bounds can be easily handled by redefining variables as usual.

We present a key result that relates the extreme points of $\mathrm{conv}(M_2^{\leq}(b))$ to the extreme points of $\mathrm{conv}(K_2^{\leq}(b))$ and $\mathrm{conv}(K_2^{\geq}(b))$. The result is much stronger; it describes a one-to-one correspondence between the extreme points of $\mathrm{conv}(M_n^{\leq}(b))$ and the extreme points of $\mathrm{conv}(K_n^{\leq}(b))$ and $\mathrm{conv}(K_n^{\geq}(b))$.

This result immediately gives us an upper bound on the number of extreme points of $\mathrm{conv}(M_2^{\leq}(b))$, and an algorithm to enumerate them in polynomial time. We also present an efficient polynomial-time algorithm to calculate all the facets of $\mathrm{conv}(M_2^{\leq}(b))$.

7.3.1 Extreme points

Before we present the main result of this section, we prove the following lemma, which characterizes one component of any extreme point of $\mathrm{conv}(M_2^{\leq}(b))$.

Lemma 7.12 If p is an extreme point of $\mathrm{conv}(M_n^{\leq}(b))$, then either $p_0 = 0$ or $p_0 = \sum_{i=1}^{n} a_i p_i - b$.

Proof See Appendix F. □

Now, we are ready for the main result; we prove a one-to-one correspondence between the extreme points of $\mathrm{conv}(M_n^{\leq}(b))$ and that of $\mathrm{conv}(K_n^{\leq}(b))$ and $\mathrm{conv}(K_n^{\geq}(b))$. By Lemma 7.12, we only need to consider $p \in M_n^{\leq}(b)$ such that $p_0 = 0$ or $p_0 = \sum_{i=1}^{n} a_i p_i - b$.

Theorem 7.13 Let $p \in M_n^{\leq}(b)$ and $q \in \mathbb{R}_+^n$ such that $q_i = p_i,\ i \in [1, n]$. If $p_0 = 0$, then p is an extreme point of $\mathrm{conv}(M_n^{\leq}(b))$ if and only if q is an extreme point of $\mathrm{conv}(K_n^{\leq}(b))$. On the other hand, if $p_0 = \sum_{i=1}^{n} a_i p_i - b$, then p is an extreme point of $\mathrm{conv}(M_n^{\leq})$ if and only if q is an extreme point of $\mathrm{conv}(K_n^{\geq}(b))$.

Proof See Appendix F. □

Theorem 7.13 gives us a one-to-one correspondence between the extreme points of $\mathrm{conv}(M_2^{\leq}(b))$, and those of $\mathrm{conv}(K_2^{\leq}(b))$ and $\mathrm{conv}(K_2^{\geq}(b))$. This allows us to use the results in Section 7.2 to present similar results for $M_2^{\leq}(b)$. For instance, from Section 7.2.1 and Theorem 7.13, we can bound the number of extreme points of $\mathrm{conv}(M_2^{\leq}(b))$.

Theorem 7.14 The number of extreme points in $\mathrm{conv}(M_2^{\leq}(b))$ is no greater than $4\lfloor \log(\gamma + 1) \rfloor + 12$, where $\gamma = \min\{a_2 - 1, u_1, u_2\}$. □

Similarly, using Theorem 7.13 and the vertex enumeration algorithm presented in Section 7.2.2, we can obtain all extreme points of $\mathrm{conv}(M_2^{\leq}(b))$ in polynomial time.

Proposition 7.15 All the extreme points of the polyhedron $\mathrm{conv}(M_2^{\leq}(b))$ can be enumerated in $\mathcal{O}(\log a_2)$ time. □

Next, we present some structural results that allow us to enumerate the facets of $\mathrm{conv}(M_2^{\leq}(b))$ efficiently. Given a sorted list of extreme points (see Section 7.3.2 for the sorting criteria), our algorithm obtains all facets of $\mathrm{conv}(M_2^{\leq}(b))$ in time linear in the number of extreme points of $\mathrm{conv}(M_n^{\leq}(b))$. This is much more efficient than running a generic convex hull algorithm such as Graham Scan (Cormen et al. 2001), which runs in cubic time in the number of extreme points.

Once we enumerate all the extreme points of $\mathrm{conv}(M_2^{\leq}(b))$, optimizing a linear function over $M_2^{\leq}(b)$ can be done trivially in time linear in the number of extreme points by calculating the objective value at each extreme point. Summarizing, we can obtain the facets of $\mathrm{conv}(M_2^{\leq}(b))$ and optimize over the set $M_2^{\leq}(b)$ in polynomial time, since we can enumerate the extreme points in polynomial time.

7.3.2 Facets

Let $X_{\leq}$ and $X_{\geq}$ be the non-trivial extreme points of $\mathrm{conv}(K_2^{\leq}(b))$ and $\mathrm{conv}(K_2^{\geq}(b))$, respectively; see Definition 7.6. Let $Z_{\leq}$ and $Z_{\geq}$ be the corresponding extreme points of $\mathrm{conv}(M_2^{\leq}(b))$. From Lemma 7.12, $p_0 = 0$ for $p \in Z_{\leq}$ and $p_0 = a_1 p_1 + a_2 p_2 - b$ for $p \in Z_{\geq}$.

We first present structural results about the non-trivial facets of $\mathrm{conv}(M_n^{\leq})$, and then present the algorithm that obtains these facets efficiently; we refer to all facets other than those defined by the upper and lower bounds on the variables as non-trivial facets.

We assume that the sets $Z_{\leq}$ and $Z_{\geq}$ are sorted in increasing order of p_2, for $p \in Z_{\leq}$ and $p \in Z_{\geq}$, respectively; breaking ties in decreasing order of p_1. Henceforth, we treat them as ordered sets; referring to elements as "adjacent to", "previous to", "next to", "between", "before", and "after" other elements. We use the following notation for each of these terms.

$\hat{p}$ before $\bar{p}$	$\hat{p} \prec \bar{p}$	$\hat{p}_2 \leq \bar{p}_2$ and $\hat{p}_1 \geq \bar{p}_1$
$\hat{p}$ after p	$\hat{p} \succ \bar{p}$	$\hat{p}_2 \geq \bar{p}_2$ and $\hat{p}_1 \leq \bar{p}_1$
$\hat{p}$ previous to $\bar{p}$	$\hat{p} \preceq \bar{p}$	$\hat{p} \prec \bar{p}$ and $\nexists p$ such that $\hat{p} \prec p \prec \bar{p}$
$\hat{p}$ next to $\bar{p}$	$\hat{p} \succeq \bar{p}$	$\hat{p} \succ \bar{p}$ and $\nexists p$ such that $\hat{p} \succ p \succ \bar{p}$
$\hat{p}$ adjacent to $\bar{p}$	$\hat{p} \asymp \bar{p}$	$\hat{p} \preceq \bar{p}$ or $\hat{p} \succeq \bar{p}$
p between $\bar{p}$ and $\hat{p}$	$p \in \mho(\hat{p}, \bar{p})$	p such that $\hat{p} \prec p \prec \bar{p}$ or $\hat{p} \succ p \succ \bar{p}$

For any point $p^{(\cdot)}$ in $Z_{\leq}$ ($Z_{\geq}$), we denote the corresponding point in $X_{\leq}$ ($X_{\geq}$) by $q^{(\cdot)}$. The preceding relations are also defined for the elements of $X_{\leq}$ and $X_{\geq}$.

We use triples of extreme points to define a half-space in $\mathbb{R}^3$. By definition, the ex-

treme points in such a triple are affinely independent. Furthermore, they either define a facet or are invalid, in which case the hyperplane defined by them passes through the interior of $\mathrm{conv}(M_n^{\leq})$. There might be more extreme points on the hyperplane defining this half-space; however, three points are sufficient to define it. In this discussion, we mostly ignore the degenerate case where more than three points lie on the same hyperplane.

By scaling, any half-space containing $\mathrm{conv}(M_2^{\leq}(b))$ can be written as $\pi_1 x_1 + \pi_2 x_2 - y \leq \pi_0$ without loss of generality, for $\pi_1, \pi_2, \pi_0 \in \mathbb{R}_+$. Lemma 7.17 shows that π_1, π_2, and $\pi_0 \in \mathbb{Z}_+$ if this half-space is a non-trivial facet of $\mathrm{conv}(K_2^{\leq}(b))$. This result does not in any way influence the analysis in this chapter, but is interesting nonetheless; independently shown by Agra and Constantino (2003).

The following Lemmas provide an insight into the structure of $\mathrm{conv}(M_n^{\leq}(b))$. They characterize several properties that any triple of extreme points defining a facet must satisfy. We assume throughout that $p^1, p^3 \in Z_{\leq}$ and $p^2, p^4 \in Z_{\geq}$. Since these pairs of points are chosen arbitrarily, we assume without loss of generality that p^1 is before p^3 and p^2 is before p^4; i.e., $p^1 \prec p^3$ and $p^2 \prec p^4$.

Lemma 7.16 Let $p^1, p^3 \in Z_{\leq}$ and $p^2, p^4 \in Z_{\geq}$ such that $p^1 \prec p^3$ and $p^2 \prec p^4$. Then, no non-trivial facet of $\mathrm{conv}(M_n^{\leq}(b))$ can be defined by p^1, p^3, and some $p \in Z_{\leq}$; or by p^2, p^4, and some $p \in Z_{\geq}$.

Proof See Appendix F. □

From Lemma 7.16, any non-trivial facet of $\mathrm{conv}(M_2^{\leq}(b))$ belongs to one of two possible families, depending on whether it is defined by p^1, p^3, and p^2; or p^1, p^2, and p^4. For the rest of this discussion, we state the results for both families; however, we only prove the lemmas for the former.

To obtain the hyperplane defined by three points, we need to calculate the equation of the plane that passes through these points. This can be done in constant time. However, we do this more efficiently using information from the analysis of $\mathrm{conv}(K_2^{\leq}(b))$ and $\mathrm{conv}(K_2^{\geq}(b))$; from Lemma 7.17. This lemma characterizes the half-spaces defined by these points in terms of the half-spaces defined by their counterparts in $X_{\leq}$ (or $X_{\geq}$), in two-dimensions. Lemma 7.17 also proves that $\pi_i \in \mathbb{Z}_+,\ i \in [0, 2]$.

Lemma 7.17 Let $\Gamma(\bar{p},\hat{p}) = (a_1\bar{p}_1 + a_2\bar{p}_2 - b)/(\hat{p}_1\bar{p}_1 + \hat{p}_2\bar{p}_2 - \hat{p}_0)$. Let $p^1, p^3 \in Z_{\leq}$ and $p^2, p^4 \in Z_{\geq}$ such that $p^1 \prec p^3$ and $p^2 \prec p^4$. Consider the half-space $\pi_1 x_1 + \pi_2 x_2 - y \leq \pi_0$. If it is defined by p^1, p^3, and p^2, then $\pi_i = \Gamma(p^2, \pi')\pi'_i,\ i \in [0,2]$, where q^1, q^3 define the half-space $\pi'_1 x_1 + \pi'_2 x_2 \leq \pi'_0$. On the other hand, if it is defined by p^1, p^2, and p^4, then $\pi_i = a_i - \Gamma(p^1, \pi')\pi'_i,\ i = 1,2$ and $\pi_0 = b - \Gamma(p^1, \pi')\pi'_0$, where q^2, q^4 define the half-space $\pi'_1 x_1 + \pi'_2 x_2 \geq \pi'_0$. Furthermore, $\Gamma \in \mathbb{Z}_+$ in either case if the hyperplane is a facet of $\mathrm{conv}(M_2^{\leq}(b))$.

Proof See Appendix F. □

Lemmas 7.18, 7.19 and 7.18 present conditions under which pairs of extreme points may define faces of $\mathrm{conv}(M_2^{\leq}(b))$. The latter two lemmas consider pairs belonging to the same set (either $Z_{\leq}$ or $Z_{\geq}$), whereas Lemma 7.18 considers one point in $Z_{\leq}$ and the other in $Z_{\geq}$.

Lemma 7.18 proves that if $p^1 \in Z_{\leq}$ and $p^4 \in Z_{\geq}$ define a face of $\mathrm{conv}(M_2^{\leq}(b))$, then no face can be defined by using elements after p^1 from $Z_{\leq}$ and elements before p^4 from $Z_{\geq}$. This is equivalent to proving that no two faces of $\mathrm{conv}(M_2^{\leq}(b))$ may intersect when projected onto the space of x_1 and x_2.

Lemma 7.18 Let $p^1, p^3 \in Z_{\leq}$ and $p^2, p^4 \in Z_{\geq}$ such that $p^1 \prec p^3$ and $p^2 \prec p^4$. Unless p^1, p^3, p^2, and p^4 define the same hyperplane, either the line joining p^1 and p^4, or the line joining p^2 and p^3 is not a non-trivial face of $\mathrm{conv}(M_2^{\leq}(b))$.

Proof See Appendix F. □

Lemma 7.19 proves that for any non-trivial facet, the two extreme points belonging to the same set must be adjacent points in the set. It does this by proving that for any half-space defined by $p^1, p^3 \in Z_{\leq}$ and some $p \in Z_{\geq}$ ($p^2, p^4 \in Z_{\geq}$ and some $p \in Z_{\leq}$) does not contain any points between p^1 and p^3 (p^2 and p^4).

Lemma 7.19 Let $p^1, p^3 \in Z_{\leq}$ and $p^2, p^4 \in Z_{\geq}$ such that $p^1 \prec p^3$ and $p^2 \prec p^4$. The half-space $\pi_1 x_1 + \pi_2 x_2 - y \leq \pi_0$ defined by p^1, p^3, and p^2 does not contain any $p \in Z_{\leq}$ such that $p \in \mho(p^1, p^3)$. Similarly, the half-space defined by p^1, p^2, and p^4 does not contain $p \in Z_{\geq}$ such that $p \in \mho(p^2, p^4)$.

Proof See Appendix F. □

Since we are only interested in facets, we are only interested p^1, p^3 that are adjacent elements of $Z_{\leq}$, and p^2, p^4 that are adjacent elements of $Z_{\geq}$. Therefore, we can assume that p^1 is previous to p^3, and that p^2 is previous to p^4. Lemma 7.20 proves that we need to consider all pairs of adjacent elements; and that each pair defines a non-trivial facet with some element in the other set.

Lemma 7.20 Let $p^1, p^3 \in Z_{\leq}$ and $p^2, p^4 \in Z_{\geq}$ such that $p^1 \preceq p^3$ and $p^2 \preceq p^4$. Then, p^1, p^3 define a facet with some $p \in Z_{\geq}$, and p^2, p^4 define a facet with some $p \in Z_{\leq}$.

Proof See Appendix F. □

Now, we know that facets can be obtained by considering all adjacent points in $Z_{\leq}$ and $Z_{\geq}$. We still need to determine which point in $Z_{\geq}$ defines a facet with adjacent points $p^1, p^3 \in Z_{\leq}$, and similarly for $p^2, p^4 \in Z_{\geq}$.

Two questions need to be addressed here: which combinations of adjacent points to consider, and how to check whether the hyperplane defined by them is a facet. The following two lemmas resolve these two issues, thus proving correctness of the algorithm presented subsequently.

Lemma 7.21 allows us to parse through these pairs of adjacent points in sequence. It proves that if we have already considered all pairs of points before $p^1 \in Z_{\leq}$ and before $p^2 \in Z_{\geq}$, then either p^1, p^3, and p^2; or p^1, p^2, and p^4 must define a facet. We also denote the elements previous to p^1 and p^2 by $p^5 \in Z_{\leq}$ and $p^6 \in Z_{\geq}$, respectively.

Lemma 7.21 Let $p^5, p^1, p^3 \in Z_{\leq}$ and $p^6, p^2, p^4 \in Z_{\geq}$ such that $p^5 \preceq p^1 \preceq p^3$ and $p^6 \preceq p^2 \preceq p^4$. If either p^1, p^6, and p^2; or p^5, p^1, and p^2 define a facet, then so do either p^1, p^3, and p^2; or p^1, p^2 and p^4.

Proof See Appendix F. □

Thus, Lemma 7.21 answers the first question: which combinations of extreme points to check. On the other hand, Lemma 7.22 proves that any half-space defined by two adjacent points in a set contains all other points in that set. Furthermore, it shows that if it contains the next (previous) element in the other set, it contains all elements after

(before) it. Therefore, it answers the second question: how to check if a hyperplane defines a facet.

Lemma 7.22 Let $p^5, p^1, p^3 \in Z_{\leq}$ and $p^6, p^2, p^4 \in Z_{\geq}$ such that $p^5 \preceq p^1 \preceq p^3$ and $p^6 \preceq p^2 \preceq p^4$. The half-space $\pi_1 x_1 + \pi_2 x_2 - y \leq \pi_0$ defined by p^1, p^3, and p^2 contains all $p \in Z_{\leq},\ p \neq p^1, p^3$. Furthermore, if it contains p^4 (p^6), then it contains all points $p\ (p') \in Z_{\geq}$ such that $p \succ p^4$ ($p' \prec p^6$). Similarly, the half-space defined by p^2, p^4, and p^1 contains all $p \in Z_{\geq},\ p \neq p^2, p^4$. Furthermore, if it contains p^3 (p^5), then it contains all points $p\ (p') \in Z_{\leq}$ such that $p \succ p^3$ ($p' \prec p^5$).

Proof See Appendix F. □

This gives us an algorithm to enumerate all the facets of $\mathrm{conv}(M_2^{\leq}(b))$. We sketch the pseudo-code, which we refer to as $Facet_2$. All special cases, such as trivial facets, $p \in Z_{\leq} \cap Z_{\geq}$, four points on a hyperplane, etc., are handled appropriately; they are not discussed here. Since $Z_{\leq}$ and $Z_{\geq}$ are sorted, we treat them as arrays (lists); we denote the i^{th} element by $Z_{(.)}[i]$ We add the facets we obtain to list $\mathcal{L}$.

$Facet_2(Z_{\leq}, Z_{\geq})$

$i = 1$, $j = 1$

While $i < |Z_{\leq}|$ or $j < |Z_{\geq}|$

If the half-space defined by $Z_{\leq}[i], Z_{\geq}[j]$, and $Z_{\leq}[i+1]$ contains $Z_{\geq}[j+1]$ and $Z_{\geq}[j-1]$, then

$Z_{\leq}[i], Z_{\geq}[j]$, and $Z_{\leq}[i+1]$ define a facet.

Add it to $\mathcal{L}$

$i = i + 1$

else

$Z_{\leq}[i], Z_{\geq}[j]$, and $Z_{\geq}[j+1]$ define a facet.

Add it to $\mathcal{L}$

$j = j + 1$

end if

end while

If $i = |Z_{\leq}|$,

then

$Z_{\leq}[i], Z_{\geq}[k]$, and $Z_{\geq}[k+1]$ define a facet for all $j \leq k < |Z_{\geq}|$.

Add them to $\mathcal{L}$.

else

$Z_{\leq}[k], Z_{\leq}[k+1]$, and $Z_{\geq}[j]$ define a facet for all $i \leq k < |Z_{\leq}|$.

Add them to $\mathcal{L}$. end if

In Section 7.2.2, we obtain the extreme points of $\text{conv}(K_2^{\leq}(b))$ in sequence. Thus, the extreme points of $\text{conv}(M_2^{\leq}(b))$ are already available as two separate sorted lists $Z_{\leq}$ and $Z_{\geq}$. Hence, using $Facet_2$, from the extreme points of $\text{conv}(M_2^{\leq}(b))$ using the algorithm in Section 7.2.2; we enumerate all facets of $\text{conv}(M_2^{\leq}(b))$ in $\mathcal{O}(\log \gamma)$, since we have $\mathcal{O}(\log \gamma)$ extreme points. This is important since we can now compute all the facets of $\text{conv}(M_2^{\leq}(b))$ efficiently.

Next, we study certain classes of two-slope super-additive functions that will be used in Section 7.5 to characterize in closed form super-additive lower bounds for the facets of $\text{conv}(K_2^{\leq}(b))$ and $\text{conv}(M_2^{\leq}(b))$.

7.4 Super-additive functions

In this section, we introduce some properties of super-additive functions. Then, we study two closely related families of two-slope functions and describe the conditions under which they are super-additive. We also describe super-additive lower bounds to these functions when these conditions are not satisfied. These families of functions generalize the lifting functions developed in Section 7.5 and are used to generate super-additive lower bounds for them.

From Definition 1.39, a function $f : \mathbb{R}^m \mapsto \mathbb{R}$ is super-additive on $D \subseteq \mathbb{R}^m$ if $f(d_1) + f(d_2) \leq f(d_1 + d_2)$ for all $d_1, d_2 \in D$ such that $d_1 + d_2 \in D$. We restrict our attention to $D \subseteq R$. In particular, we consider $0 \leq D \leq b$ for $K_2^{\leq}(b)$, and $D \geq 0$ for $M_2^{\leq}(b)$. We state the following known results.

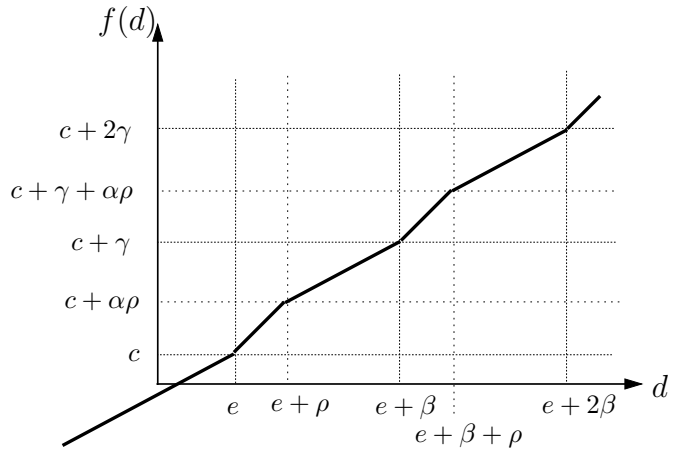
Proposition 7.23 (Nemhauser and Wolsey (1988)) If $f : \mathbb{R}^m \mapsto \mathbb{R}$ is super-additive,

then $f(0) \leq 0$. □

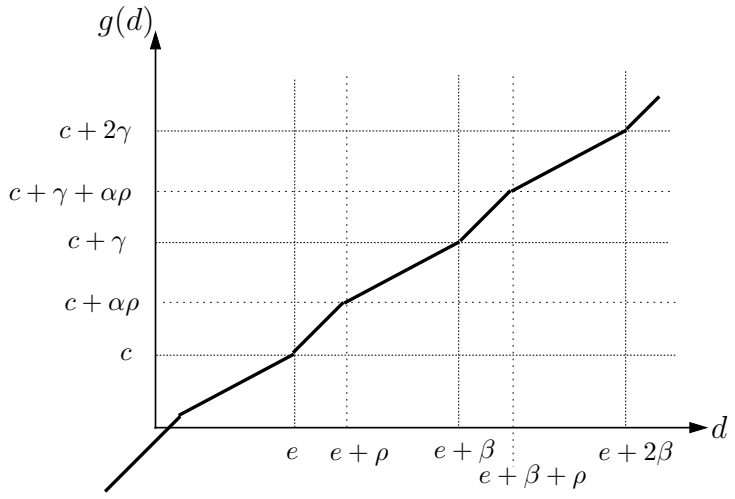
Proposition 7.24 (Agra and Constantino (2003)) If $f : \mathbb{R}_+^m \mapsto \mathbb{R}$ is convex and $f(0) \leq 0$, then f is super-additive. □

The two families of two-slope functions introduced in Section 7.4 are very closely related. In fact, $f(d)$ and $g(d)$ are piecewise linear functions that differ only in the first piece, i.e., when $d \leq e$; where f and g correspond to family 1 and family 2, respectively; see Figure 7.3.

Figure 7.3: Families of repeating two-slope functions



Family 1



Family 2

7.4.1 Family 1

Consider the function

$$f(d) = \begin{cases} c - \delta(e-d) & if\ d \le e, \\ c + \gamma k + \alpha(d - \beta k - e) & if\ e + \beta k \le d \le e + \beta(k+1) + \rho, \\ c + \gamma(k+1) - \delta(e + \beta(k+1) - d) & if\ e + \beta k + \rho \le d \le e + \beta(k+1), \end{cases}$$

for $k \ge 0$ and integer. We assume that the parameters $\beta, \gamma, e, c, \delta, \rho, \alpha$ are non-negative, and $\rho < \beta$, $\delta \le \gamma/\beta$, $\alpha = (\gamma - \delta(\beta - \rho))/\rho \ge \gamma/\beta$. Thus, α is defined in terms of the other parameters, and is introduced only for the sake of notational convenience.

Let $c_0 = e\delta$, $c_1 = (e+\beta)\delta - \gamma = e\delta - \rho(\alpha - \delta)$, and $c_2 = \alpha(e-\beta) + \gamma = e\alpha - (\beta - \rho)(\alpha - \delta)$.

We assume that $c \le c_0$ to ensure that $f(0) \le 0$, see Proposition 7.23. When $\rho = 0$, α is undefined; therefore, c_2 is undefined if $\rho = 0$. It can be shown easily $c_2 \le c_1$ if and only if $e \le \beta - 2\rho$. The proofs that follow assume that $\rho > 0$; however they can be easily modified for the case $\rho = 0$.

Proposition 7.25 If $e \ge \beta - \rho$, then f is super-additive.

Proof See Appendix F. □

For the rest of Section 7.4.1, we assume that $e < \beta - \rho$.

Proposition 7.26 f is super-additive if and only if $c \le c_1$ or $c \le c_2$.

Proof See Appendix F. □

We now present two super-additive lower bounds for f if $e < \beta - \rho$, $c > c_1$ and $c > c_2$. These two functions f_1 and f_2 are parametrized using $\beta', \gamma', e', c', \delta', \rho'$, and α'. We assume that $\rho > 0$, and treat $\rho = 0$ separately in Section 7.4.3.1.

To obtain the former, we fix δ, and modify α, ρ to get f_1. $f_1(d) = f_{\beta', \gamma', e', c', \delta', \rho', \alpha'}(d)$, where $\beta' = \beta$, $\gamma' = \gamma$, $e' = e$, $c' = c$, $\delta' = \delta$, $\alpha' = (\gamma - c)/(\beta - e)$, and $\rho' = (\gamma - \delta\beta)/(\alpha' - \delta)$. f_1 is super-additive since $c' = c'_2$. Since $c > c_2$, we have $\alpha' < \alpha$; therefore $f_1 \le f$.

For the latter, we fix α and modify δ, ρ to obtain f_2. $f_2(d) = f_{\beta', \gamma', e', c', \delta', \rho', \alpha'}(d)$, where $\beta' = \beta$, $\gamma' = \gamma$, $e' = e$, $c' = c$, $\delta' = (c + \gamma)/(e + \beta)$, $\rho' = (e\delta' - c)/(\alpha - \delta')$, and $\alpha' = \alpha$. f_2 is super-additive since $c' = c'_1$. Since $c > c_1$, we have $\delta' > \delta$; therefore $f_2 \le f$.

7.4.2 Family 2

Consider the function

$$g(d) = \begin{cases} c - \gamma + \alpha(d + \beta - e) & if\ d \le e + \rho - \beta, \\ c - \delta(e - d) & if\ e + \rho - \beta \le d \le e, \\ c + \gamma k + \alpha(d - \beta k - e) & if\ e + \beta k \le d \le e + \beta(k+1) + \rho, \\ c + \gamma(k+1) - \delta(e + \beta(k+1) - d) & if\ e + \beta k + \rho \le d \le e + \beta(k+1), \end{cases}$$

for $k \ge 0$ and integer. We assume that all parameters β, γ, e, c, δ, ρ, α are non-negative, $\rho < \beta$, $\delta \le \gamma/\beta$, $\alpha = (\gamma - \delta(\beta - \rho))/\rho \ge \gamma/\beta$. As before, α is introduced only for the sake of notational convenience.

Again, let $c_0 = e\delta$, $c_2 = \alpha(e - \beta) + \gamma = e\alpha - (\beta - \rho)(\alpha - \delta)$, and $c_3 = \alpha(e - 2\beta) + 2\gamma = e\alpha - 2(\beta - \rho)(\alpha - \rho)$. When $\rho = 0$, α is undefined, and g is not precisely defined for $d \le e - \beta$. Hence, we do not consider $\rho = 0$. Furthermore, we only consider $e > \beta - \rho$, since g is the same as f if $e \le \beta - \rho$. We assume that $c \le c_2$ to ensure that $g(0) \le 0$, see Proposition 7.23. It is easy to see that $c_3 \le c_0$ if and only if $e \le 2(\beta - \rho)$.

Proposition 7.27 g is super-additive if and only if $c \le c_0$ or $c \le c_3$.

Proof See Appendix F. □

Next, we present two super-additive lower bounds of g if $c > c_0$ and $c > c_3$.

For the first, we fix δ and modify α, ρ to obtain g_1. $g_1(d) = g_{\alpha',\beta',\delta',\gamma',e',c',\rho'}(d)$, where $\beta' = \beta$, $\gamma' = \gamma$, $\delta' = \delta$ $e' = e$, $c' = c$, $\alpha' = (2\gamma - c)/(2\beta - e)$, and $\rho' = (\gamma - \delta\beta)/(\alpha' - \delta)$. g_1 is super-additive since $c' = c'_3$. Since $c > c_3$, we have $\alpha' < \alpha$; therefore $g_1 \le g$.

For the second, we fix α and modify δ, ρ to get g_2. $g_2(d) = g_{\alpha',\beta',\delta',\gamma',e',c',\rho'}(d)$, where $\beta' = \beta$, $\gamma' = \gamma$, $\alpha' = \alpha$, $e' = e$, $c' = c$, $\delta' = c/e$, and $\rho' = (\gamma - \beta\delta')/(\alpha - \delta')$. g_2 is super-additive since $c' = c'_0$. Since $c > c_0$, we have $\delta' > \delta$; therefore $g_2 \le g$.

7.4.3 Important special cases

7.4.3.1 Special case of family 1 with $\rho = 0$

Next, we discuss the important special case of Family 1 where $\rho = 0$. In this case, α is undefined. The following results can be obtained easily from the general exposition;

however the special case merits special attention since the lifting function often belongs to this sub-family. To be precise, we consider the function

$$f'(d) = \begin{cases} c - \delta(e-d) & if\ d \leq e, \\ c + \gamma(k+1) - \delta(e + \beta(k+1) - d) & if\ e + \beta k < d \leq e + \beta(k+1), \end{cases}$$

for $k \geq 0$ and integer. Again, $\delta \leq \gamma/\beta$, and all parameters are non-negative. As before, we let $c_0 = e\delta$ and assume that $c \leq c_0$. Again, we define $c_1 = (e+\beta)\delta - \gamma$. c_2 is undefined if $\rho = 0$ (since α is undefined). From Propositions 7.25 and 7.26, we get the following results as corollaries.

Corollary 7.28 If $e \geq \beta$, then f' is super-additive. □

Corollary 7.29 When $e < \beta$, f' is super-additive if and only if $c \leq c_1$. □

Next, we characterize in closed form two super-additive lower bounds to f' when $e < \beta$ and $c > c_1$.

For the first, we fix δ and modify α, ρ to get f'_1. $f'_1(d) = f_{\beta',\gamma',e',c',\delta',\rho',\alpha'}(d)$, where $\beta' = \beta$, $\gamma' = \gamma$, $e' = e$, $c' = c$, $\delta' = \delta$, $\alpha' = (\gamma - c)/(\beta - e)$, and $\rho' = (\gamma - \delta\beta)/(\alpha' - \delta)$. f'_1 is super-additive since $c' = c'_2$ (as defined for f). Since $\alpha' < \alpha$; therefore $f'_1 \leq f'$.

For the second super-additive lower bound, we fix γ and modify δ, ρ as follows to obtain f'_2. $f'_2(d) = f'_{\beta',\gamma',e',c',\delta',\rho',\alpha'}(d)$, where $\beta' = \beta$, $\gamma' = \gamma$, $e' = e$, $c' = c$, $\delta' = (c + \gamma)/(e + \beta)$, and $\rho' = 0$. f_2 is super-additive since $c' = c'_1$. Since $c > c_1$, we have $\delta' > \delta$; therefore $f'_2 \leq f'$.

7.4.3.2 Special case of family 2 with $\delta = 0$

Next, we discuss the important special case of Family 2 where $\delta = 0$. Again, this special case merits special attention since the lifting function often belongs to this sub-family. We consider the function

$$g'(d) = \begin{cases} c - \gamma + \alpha(d + \beta - e) & if\ d \leq e + \rho - \beta, \\ c & if\ e + \rho - \beta \leq d \leq e, \\ c + \gamma k + \alpha(d - \beta k - e) & if\ e + \beta k \leq d \leq e + \beta(k+1) + \rho, \\ c + \gamma(k+1) & if\ e + \beta k + \rho \leq d \leq e + \beta(k+1), \end{cases}$$

for $k \geq 0$, where $\rho < \beta$, $\alpha \geq \gamma/\beta$, and all parameters are non-negative. As before, $c_0 = 0$, $c_2 = \alpha(e + \rho - \beta)$, and $c_3 = e\alpha - 2(\beta - \rho)\alpha$. We only consider $e > \beta - \rho$. (Section 7.4.1 covers $e \leq \beta - \rho$.) As before, we assume that $c \leq c_2$. The following result follows as a corollary of Proposition 7.27.

Corollary 7.30 g' is super-additive if and only if $c \leq c_3$. □

Next, we present the two super-additive lower bounds of g' if $c > c_3$.

First, we modify both α and ρ to obtain the super-additive lower bound g'_1. $g'_1(d) = g'_{\alpha',\beta',\gamma',e',c',\rho'}(d)$, where $\beta' = \beta$, $\gamma' = \gamma$, $e' = e$, $c' = c$, $\alpha' = (2\gamma - c)(2\beta - e)$, and $\rho' = \gamma/\alpha'$. g'_1 is super-additive since $c' = c'_3$. Since $c > c_3$, we have $\alpha' < \alpha$; therefore $g'_1 \leq g'$.

Next, we fix α and modify δ, ρ to obtain the super-additive lower bound g'_2. $g'_2(d) = g_{\alpha',\beta',\gamma',e',c',\rho'}(d)$, where $\beta' = \beta$, $\gamma' = \gamma$, $\alpha' = \alpha$, $e' = e$, $c' = c$, $\delta' = c/e$, and $\rho' = (\gamma - \beta\delta')/(\alpha - \delta')$. g'_2 is super-additive since $c' = c'_0$ (as defined for g). Since $c > c_0$, we have $\delta' > \delta$; therefore $g'_2 \leq g'$.

7.5 Exact lifting function

In this section, we study the exact lifting function for facets to the convex hull of set $M_2^{\leq}$. The exact lifting function is obtained by solving a parametric optimization problem, see Section 1.5.2. From Section 7.3, we know how to optimize a linear function over the set $M_2^{\leq}$ in polynomial time; however, no closed-form characterization is known.

We propose and describe various lower bounds for the exact lifting function. We use these lower bounds to develop super-additive lower bounds for the exact lifting function. We use these super-additive lifting functions in a sequence independent lifting framework to develop new classes of strong inequalities for $K_n^{\leq}(b)$ and $M_n^{\leq}(b)$.

We begin this section by studying the exact lifting function for facets of $\mathrm{conv}(K_2^{\leq}(b))$, which is a similar, yet simpler polyhedron. Then, we study the exact lifting function for facets of $\mathrm{conv}(M_2^{\leq}(b))$. For both cases, we first study the exact value function and derive upper bounds to it. Then, we use these upper bounds to describe lower bounds to the exact lifting function.

7.5.1 Pure-integer knapsack set

First, we study the exact lifting function for facets of the convex hull of the restriction with two integer variables and no continuous variable, denoted by $K_2^{\leq}(b)$, and introduced in Section 7.2. Then, we use these upper bounds to describe lower bounds to the exact lifting function. Finally, we describe super-additive lower bounds to the lifting function using the families of two-slope repeating functions studied in Section 7.4.

The assumptions associated with the set are the same as enumerated on Page 146, with one exception. We no longer assume (A.1); only assumptions (A.0) and (A.2)-(A.4).

Denoting the facets of $\mathrm{conv}(K_2^{\leq}(b))$ by

$$\pi_1 x_1 + \pi_2 x_2 \leq \pi_0, \tag{7.2}$$

we make the following assumptions:

(A.5) We assume that $\pi > 0$.

(A.6) We assume that $\pi_1/a_1 \geq \pi_2/a_2$; by reordering the variables, if necessary.

(A.7) We assume that $\pi_0 \leq \pi_1 u_1 + \pi_2 u_2$; else (7.2) is not a face of $\mathrm{conv}(K_2^{\leq}(b))$.

7.5.1.1 Value functions

The value function of (7.2) is defined as

$$v_{IP}(d) = \max\{\pi_1 x_1 + \pi_2 x_2 \, : \, x_1, x_2 \in K_2^{\leq}(d)\}.$$

In all optimal solutions to v_{IP}, $x_i = u_i$ if $a_i = 0$ or $u_i = 0$ $i = 1, 2$, and $x_i = 0$ if $\pi_i = 0,\ i = 1, 2$. In these cases, v_{IP} reduces to a problem with a single integral variable and hence can be characterized in a closed form for all d. Hence, we are justified in making assumptions (A.2) and (A.5).

From Section 7.2, we can evaluate the value function in polynomial time for a particular d. Kannan (1993) presents a polynomial-time (and polynomial-space) algorithm for parametric integer programming in fixed dimension; however, the value function can not be characterized using a closed form even for simple sets such as $K_2^{\leq}(b)$.

In the rest of Section 7.5.1, we define several upper bounds to the value function, use

these to derive lower bounds for the exact lifting function of (7.2), and finally describe super-additive lower bounds to these lifting functions.

To derive the first upper bound, we consider the linear programming relaxation of v_{IP}, and denote the corresponding value function as $v_{LP}(d)$, i.e.,

$$v_{LP}(d) = \max\{\pi_1 x_1 + \pi_2 x_2 \,:\, x_1, x_2 \in \mathbb{R}_+,\ a_1 x_1 + a_2 x_2 \leq d,\ \ x \leq u\}.$$

Upper bound v_{LP} can be stated explicitly as follows. Since $\frac{\pi_1}{a_1} \geq \frac{\pi_2}{a_2}$from assumption (A.6), the optimal solution has $x_2 = 0$ so long as $d \leq a_1 u_1$. For larger d, x_1 is set to u_1, and x_2 equals $\min\{u_2, (d - a_1 u_1)/a_2\}$. see Figure 7.4. Then, we see that

$$v_{LP}(d) = \begin{cases} \frac{\pi_1}{a_1} d & if\ 0 \leq d \leq a_1 u_1, \\ \pi_1 u_1 + \frac{\pi_2}{a_2}(d - a_1 u_1) & if\ a_1 u_1 \leq d \leq a_1 u_1 + a_2 u_2, \\ \pi_1 u_1 + \pi_2 u_2 & if\ a_1 u_1 + a_2 u_2 \leq d. \end{cases}$$

We wish to obtain stronger upper bounds to $v_{IP}(d)$ that are still easy to characterize in closed form. To develop these, we consider the partial linear programming relaxations of $v_{IP}(d)$ where only one of the variables is allowed to be continuous. For ease of presentation, we first make the following simplifying assumptions: $a_2 u_2 \geq a_1$ and $a_1 u_1 \geq \pi_2 \frac{a_1}{\pi_1}$; we will subsequently drop these assumptions.

To derive the second upper bound, consider the relaxation of v_{IP} where x_1 is constrained to be integral, but x_2 is continuous. We denote the corresponding value function as v_1, i.e.,

$$v_1(d) = \max\{\pi_1 x_1 + \pi_2 x_2 \,:\, x_1 \in \mathbb{Z}_+, x_2 \in \mathbb{R}_+,\ a_1 x_1 + a_2 x_2 \leq d,\ \ x \leq u\}.$$

Since v_{LP} is a relaxation of v_1, we have $v_{IP}(d) \leq v_1(d) \leq v_{LP}(d)$. Since we have only one integral variable, $v_1(d)$ is easy to characterize in closed form. For all d that are integral multiples of a_1 up to $a_1 u_1$, the optimal solution has $x_2 = 0$, since $\frac{\pi_1}{a_1} \geq \frac{\pi_2}{a_2}$, by assumption (A.6).

For all values of d that are not integral multiples of a_1, the remainder after setting variable x_1 to its largest possible integral value is allocated to the continuous variable x_2. For values of d greater than $a_1 u_1$, the optimal solution is the same as for v_{LP}, see

Figure 7.4. Thus, for $0 \leq k < u_1$,

$$v_1(d) = \begin{cases} \pi_1 k + \frac{\pi_2}{a_2}(d - ka_1) & if\ ka_1 \leq d < (k+1)a_1, \\ \pi_1 u_1 + \frac{\pi_2}{a_2}(d - a_1 u_1) & if\ a_1 u_1 \leq d \leq a_1 u_1 + a_2 u_2, \\ \pi_1 u_1 + \pi_2 u_2 & if\ a_1 u_1 + a_2 u_2 \leq d. \end{cases}$$

To obtain the third upper bound to v_{IP}, we consider the relaxation where x_2 is constrained to be integral, but x_1 is continuous. We denote this upper bound to the value function as $v_2(d)$, i.e.,

$$v_2(d) = \max\{\pi_1 x_1 + \pi_2 x_2 \ :\ x_1 \in \mathbb{R}_+, x_2 \in \mathbb{Z}_+,\ a_1 x_1 + a_2 x_2 \leq d,\ \ x \leq u\}.$$

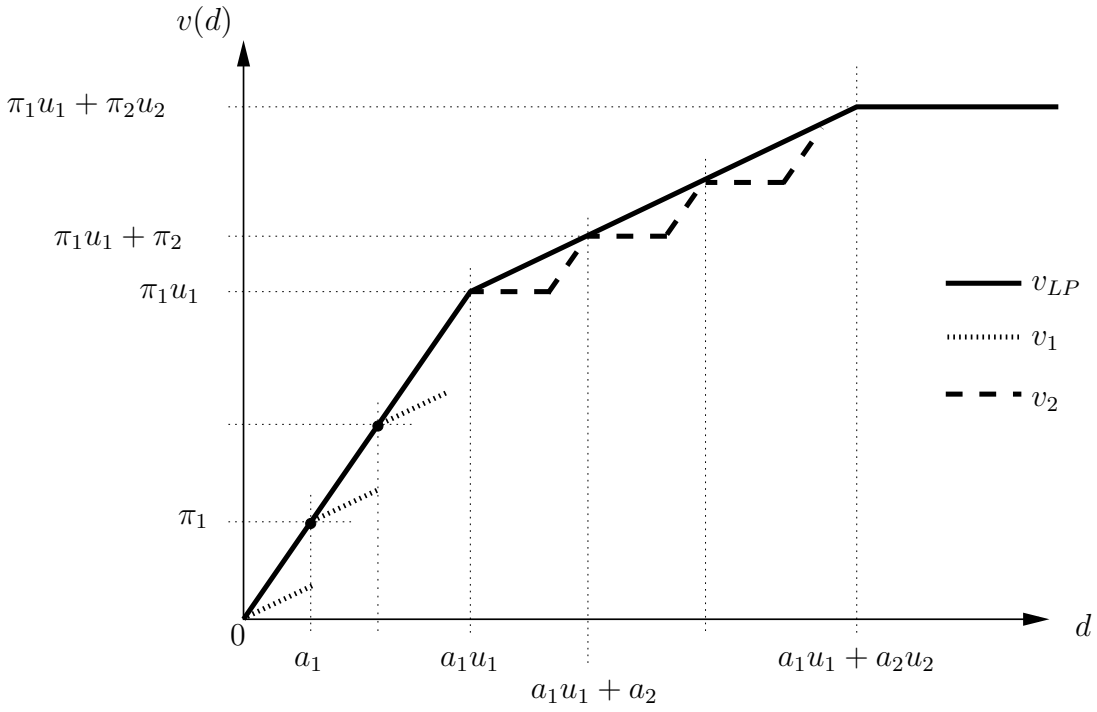
Since v_{LP} is also a relaxation of v_2, we have $v_{IP}(d) \leq v_2(d) \leq v_{LP}(d)$. Again, $v_2(d)$ is easy to characterize in closed form. Since $\frac{\pi_1}{a_1} \geq \frac{\pi_2}{a_2}$ from assumption (A.6), for all values of $d \leq a_1 u_1$, we use only x_1. For larger values of d, if $d - a_1 u_1$ is an integral multiple of a_2, the optimal value has $x_2 = (d - a_1 u_1)/a_2$, after setting x_1 to u_1. For values which do not correspond to these integral multiples, consider d such that $r(d - a_1 u_1, a_2) = a_2 - \pi_2 \frac{a_1}{\pi_1}$. For such d, there exist two optimal solutions: $x_1 = u_1, x_2 = \ell$; and $x_1 = u_1 - \frac{\pi_2}{\pi_1}, x_2 = \ell + 1$. Therefore, we can increase the objective value by reducing x_1 and incrementing x_2 if $r(d - a_1 u_1, a_2) \geq a_2 - \pi_2 \frac{a_1}{\pi_1}$, see Figure 7.4. Thus, for $0 \leq \ell < u_2$,

$$v_2(d) = \begin{cases} \frac{\pi_1}{a_1} d & if\ 0 \leq d \leq a_1 u_1 \\ \pi_1 u_1 + \pi_2 \ell & if\ \ell a_2 \leq d - a_1 u_1 \leq (\ell+1)a_2 - \pi_2 \frac{a_1}{\pi_1}, \\ \frac{\pi_1}{a_1}(d - (\ell+1)a_2) + \pi_2(\ell+1) & if\ (\ell+1)a_2 - \pi_2 \frac{a_1}{\pi_1} \leq d - a_1 u_1 \leq (\ell+1)a_2, \\ \pi_1 u_1 + \pi_2 u_2 & if\ a_1 u_1 + a_2 u_2 \leq d. \end{cases}$$

Figure 7.4 presents the form of the various upper bounds to the value functions for a pure-integer knapsack set $K_2^{\leq}(b)$. The exact value function v_{IP} can not be characterized in closed form in polynomial time. Next, we present a numerical example that illustrates these upper bounds; using the pure-integer set from Figure 7.2.

We consider $K_2^{\leq}(33) = \{x_1, x_2 \in \mathbb{Z}_+ \ :\ 5x_1 + 2x_2 \leq 33,\ x_1 \leq 6,\ x_2 \leq 12\}$, and the facet-defining inequality $3x_1 + x_2 \leq 19$. Thus, we have $\pi_1 = 3, \pi_2 = 1$; $a_1 = 5, a_2 = 2$; $u_1 = 6, u_2 = 12$; and $b = 33$. Since v_1, v_2 are different from v_{LP} for values of d on either

Figure 7.4: Value functions: Pure-integer knapsack set



side of $a_1 u_1 = 30$, we plot v_{LP}, v_1, v_2, v_{IP} for values of d from 25 to 34 in Figure 7.5. Even though v_1, v_2 may not appear to be strong upper bounds to v_{IP}, they are often the strongest super-additive upper bounds, which is the property we desire.

Both $v_1(d)$ and $v_2(d)$ are upper bounds on $v_{IP}(d)$ for all d. Furthermore, $v_1(d)$ equals v_{LP} for $d \geq a_1 u_1$, and $v_2(d)$ equals v_{LP} for $d \leq a_1 u_1$. This allows us to obtain stronger upper bounds to v_{IP} by considering

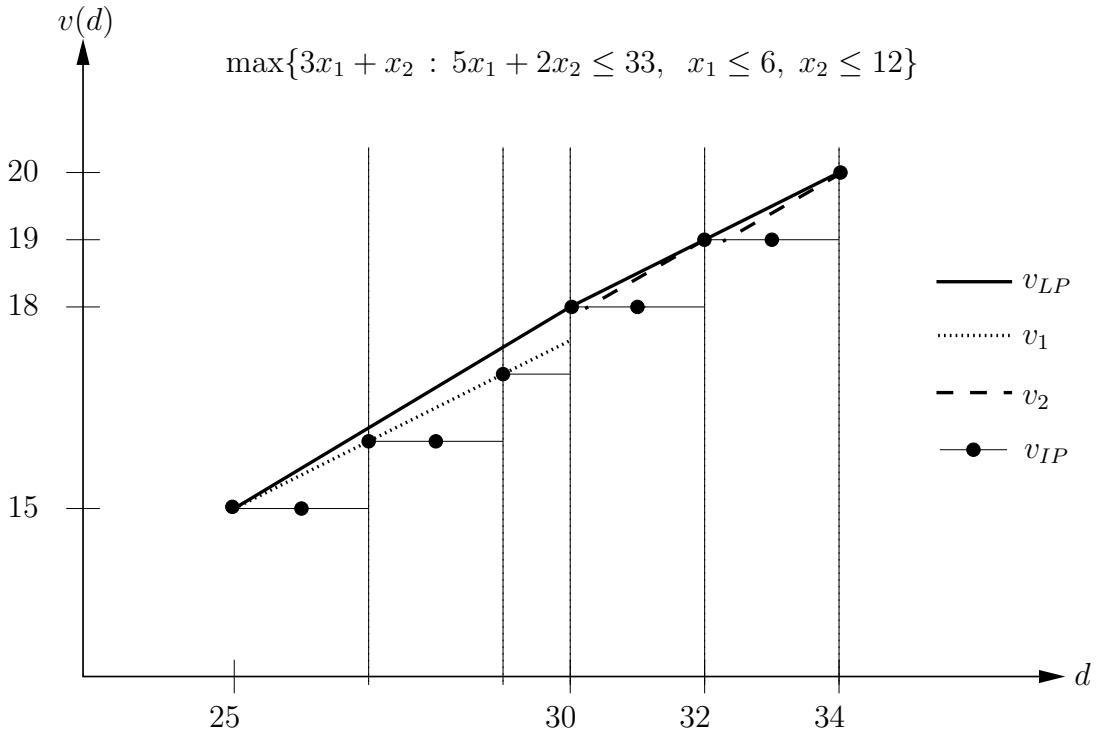
$$v(d) = \min\{v_1(d), v_2(d)\}.$$

Thus, for $0 \leq k < u_1$ and $0 \leq \ell < u_2$, we get

$$v(d) = \begin{cases} \pi_1 k + \frac{\pi_2}{a_2}(d - k a_1) & if\ k a_1 \leq d < (k+1) a_1, \\ \pi_1 u_1 + \pi_2 \ell & if\ \ell a_2 \leq d - a_1 u_1 \leq (\ell+1) a_2 - \pi_2 \frac{a_1}{\pi_1}, \\ \frac{\pi_1}{a_1}(d - (\ell+1) a_2) + \pi_2 (\ell+1) & if\ (\ell+1) a_2 - \pi_2 \frac{a_1}{\pi_1} \leq d - a_1 u_1 \leq (\ell+1) a_2, \\ \pi_1 u_1 + \pi_2 u_2 & if\ a_1 u_1 + a_2 u_2 \leq d. \end{cases}$$

In the preceding closed-form characterizations, v_1 and v_2 were simplified by the as-

Figure 7.5: Value functions for pure-integer knapsack set: Numerical example



sumptions $a_2u_2 \geq a_1$ and $a_1u_1 \geq \pi_2\frac{a_1}{\pi_1}$, respectively. To accurately model the general case, we now drop these assumptions and modify $v(d)$ to obtain $v'(d)$ as follows; some of the intervals may be empty in the following closed-form characterization. For $k < u_1$ and $ka_1 \leq d < (k+1)a_1$, the remainder after setting x_1 to the largest possible integral value is allocated to x_2 only if $d - ka_1 \leq a_2u_2$. For $\ell < u_2$ and $(\ell+1)a_2 - \pi_2\frac{a_1}{\pi_1} \leq d - a_1u_1 \leq (\ell+1)a_2$, we can increase $v_2(d)$ (and hence $v(d)$) by reducing x_1 and incrementing x_2 only if $r(d - a_1u_1, a_2) \geq a_2 - a_1u_1$. We define $\delta_1 = \min\{a_2u_2, a_1\}$ and $\delta_2 = -\min\{\pi_2\frac{a_1}{\pi_1}, a_1u_1\}$. Then, we have

$$v'(d) = \begin{cases} \pi_1 k + \frac{\pi_2}{a_2}(d - ka_1) & if\ ka_1 \leq d < ka_1 + \delta_1, \\ \pi_1 k + \pi_2 u_2 & if\ ka_1 + \delta_1 \leq d < (k+1)a_1, \\ \pi_1 u_1 + \pi_2 \ell & if\ (\ell+1)a_2 \leq d - a_1u_1 < (\ell+1)a_2 + \delta_2, \\ \frac{\pi_1}{a_1}d - (\ell+1)(a_2\frac{\pi_1}{a_1}\pi_2) & if\ (\ell+1)a_2 + \delta_2 \leq d - a_1u_1 \leq (\ell+1)a_2, \\ \pi_1 u_1 + \pi_2 u_2 & if\ a_1u_1 + a_2u_2 \leq d. \end{cases}$$

Upper bound $v'(d)$ can be strengthened for $d < a_1$ and $d > a_1u_1 + a_2(u_2 - 1)$ since v_{IP} can be characterized in closed form for these values of d. When $d < a_1$, $x_1 = 0$ and x_2 is set to the largest feasible integral value. When $d > a_1u_1 + a_2(u_2 - 1)$, any reduction of d from $a_1u_1 + a_2u_2$ decrements the value of variable x_1 from u_1 to $u_1 - \lceil \frac{a_1u_1+a_2u_2-d}{a_1} \rceil$, while variable x_2 is still set to u_2. However, this is done only if the reduction in optimal solution is less than π_2. Defining $\delta_1 = \min\{a_2u_2, a_1\}$ and $\delta_2 = -\min\{a_1u_1, a_2\}$, for these values of d, we have

$$v_{IP}(d) = \begin{cases} \pi_2\lfloor d/a_2 \rfloor & if\ 0 \le d < \delta_1, \\ \pi_2u_2 & if\ \delta_1 \le d < a_1, \\ \pi_1u_1 + \pi_2(u_2 - 1) & if\ -a_2 \le d - a_1u_1 - a_2u_2 < \delta_2, \\ \pi_1u_1 + \pi_2u_2 - \min\{\pi_1\lceil \frac{a_1u_1+a_2u_2-d}{a_1} \rceil, \pi_2\} & if\ \delta_2 \le d - a_1u_1 - a_2u_2 \le 0. \end{cases}$$

Next, we use these value functions to develop several lower bounds for the exact lifting function. Finally, we present super-additive lower bounds for the exact lifting function.

7.5.1.2 Lifting functions

We study the exact lifting function for (7.2), and denote it by $\Phi_{IP}(d)$, i.e.,

$$\Phi_{IP}(d) = \pi_0 - \max\{\pi_1x_1 + \pi_2x_2 \ :\ x_1, x_2 \in K^{\le}_2(b - d)\},$$

and $0 \le d \le b$. Defined in terms of the exact value function, we have

$$\Phi_{IP}(d) = \pi_0 - v_{IP}(b - d).$$

Unfortunately, we can not characterize v_{IP} in closed form; therefore neither can we characterize Φ_{IP}. However, this relation between the lifting function and the value function allows us to define various lower bounds for Φ_{IP} using the upper bounds for the value function. For instance, let $\Phi_{LP}(d) = \pi_0 - v_{LP}(b - d)$, and define Φ_1, Φ_2, and Φ analogously. By definition of $K^{\le}_2(b)$, we have $b \ge a_1u_1$. This allows us to easily characterize Φ_{LP}, Φ_1, Φ_2 and Φ in closed form as follows, for $0 \le k < u_1$, $0 \le \ell < u_2$, and $d \ge 0$

$$\Phi_{LP}(d) = \begin{cases} \pi_0 - \pi_1u_1 - \frac{\pi_2}{a_2}(b - a_1u_1 - d) & if\ 0 \le d \le b - a_1u_1, \\ \pi_0 - \frac{\pi_1}{a_1}(b - d) & if\ b - a_1u_1 \le d \le b. \end{cases}$$

$$\Phi_1(d) = \begin{cases} \pi_0 - \pi_1 u_1 - \frac{\pi_2}{a_2}(b - a_1 u_1 - d) & if\ 0 \le d \le b - a_1 u_1, \\ \pi_0 - \frac{\pi_2}{a_2}(b - ka_1 - d) - \pi_1 k & if\ b - (k+1)a_1 < d \le b - ka_1. \end{cases}$$

$$\Phi_2(d) = \begin{cases} \pi_0 - \pi_1 u_1 - \pi_2 \ell & if\ \ell a_2 \le b - a_1 u_1 - d \le (\ell+1)a_2 - \frac{a_1}{\pi_1}\pi_2, \\ \pi_0 - (\ell+1)(\pi_2 - a_2\frac{\pi_1}{a_1}) - \frac{\pi_1}{a_1}(b-d) & \\ & if\ -\frac{a_1}{\pi_1}\pi_2 \le b - a_1 u_1 - d - (\ell+1)a_2 \le 0, \\ \pi_0 - \frac{\pi_1}{a_1}(b-d) & if\ b - a_1 u_1 \le d \le b. \end{cases}$$

$$\Phi(d) = \begin{cases} \pi_0 - \pi_1 u_1 - \pi_2 \ell & if\ \ell a_2 \le b - a_1 u_1 - d \le (\ell+1)a_2 - \frac{a_1}{\pi_1}\pi_2, \\ \pi_0 - (\ell+1)(\pi_2 - a_2\frac{\pi_1}{a_1}) - \frac{\pi_1}{a_1}(b-d) & \\ & if\ -\frac{a_1}{\pi_1}\pi_2 \le b - a_1 u_1 - d - (\ell+1)a_2 \le 0, \\ \pi_0 - \frac{\pi_2}{a_2}(b - ka_1 - d) - \pi_1 k & if\ b - (k+1)a_1 < d \le b - ka_1. \end{cases}$$

Since (7.2) is a valid inequality for $K_2^{\le}(b)$, $\Phi_{IP}(d) \ge 0,\ d \in [0, b]$. Thus, we can strengthen all these lower bounds whenever they are negative by increasing them to 0. As in the case of $v(d)$, $\Phi(d)$ can be strengthened to $\Phi'(d)$ if $a_2 u_2 < a_1$ or $a_1 u_1 < \pi_2 \frac{a_1}{\pi_1}$, and whenever $\Phi_{IP}(d)$ can be characterized in closed form.

We separately consider the case where $\pi_2 = 0$; it merits special attention for two reasons. Firstly, this is a case where the exact lifting function can be completely characterized in closed form; we next see why this is true. Secondly, as we shall see in Section 7.5.1.3, this function is a special case of both families of super-additive functions described in Section 7.4, and one of the super-additive lower bounds turns out to be exactly the same as the well-known MIR inequality. When $\pi_2 = 0$, we have $\pi_1 = 1$ and $\pi_0 = u_1$. In other words, the facet-defining inequality is the upper bound constraint

$$x_1 \le u_1. \tag{7.3}$$

By restricting x_1 to be integral, we obtain relaxation Φ_1 of the lifting function for (7.3). For all $0 \le k < u_1$, we have

$$\Phi_1(s) = \begin{cases} 0 & if\ 0 \le s \le b - a_1 u_1, \\ u_1 - k & if\ b - (k+1)a_1 < s \le b - ka_1. \end{cases}$$

However, if $\pi_2 = 0$, all optimal solutions to the exact lifting function Φ_{IP} have $x_2 = 0$. Since Φ_1 is obtained by relaxing integrality of x_2, this relaxation is tight, and $\Phi_1 = \Phi_{IP}$.

Next, we present super-additive lower bounds for the lifting functions presented here. These super-additive lifting functions are derived from the family of super-additive functions described in Section 7.4, and can be used to obtain strong valid inequalities for the pure-integer knapsack set $K^{\leq}_n(b)$.

7.5.1.3 Super-additive lifting functions

Using the super-additive functions described in Section 7.4, we present sufficient conditions for the super-additivity of Φ_{LP}, Φ_1, and Φ_2, and present super-additive lower bounds when they are not.

First, we consider Φ_{LP}. Since (7.2) defines a facet of $K^{\leq}_2(b)$, we have $v_{IP}(b) = \pi_0$, and thus $\Phi_{IP}(0) = 0$. Therefore, $\Phi_{LP}(0) \leq 0$. Since $\Phi_{LP}(d)$ is also piecewise-linear and convex, from Proposition 7.24 Φ_{LP} is super-additive on $[0, b]$. We know that Φ_{LP} is a weak lower bound to Φ_{IP}; we want to determine whether stronger lower bounds Φ_1, Φ_2 are super-additive, and if not, describe super-additive lower bounds for them.

Second, we consider Φ_1, which belongs to the special case of Family 1 with $\rho = 0$; see Section 7.4.3.1. It is the member parametrized by $c = \pi_0 - \pi_1 u_1$, $\gamma = \pi_1$, $e = b - a_1 u_1$, $\beta = a_1$, $\delta = \frac{\pi_2}{a_2}$. Thus, we have $c_0 = \frac{\pi_2}{a_2}(b - a_1 u_1)$, and $c_1 = \frac{\pi_2}{a_2}(b - a_1(u_1 - 1)) - \pi_1$. (Since $\Phi_1(0) \leq 0$, we have $\pi_0 - \pi_1 u_1 \leq (b - a_1 u_1)\frac{\pi_2}{a_2}$.)

The following two results follow immediately from Corollaries 7.28 and 7.29.

Proposition 7.31 If $b \geq a_1(u_1 + 1)$, then $\Phi_1(d)$ is super-additive. □

Proposition 7.32 If $b < a_1(u_1 + 1)$ and $\pi_0 - \pi_1(u_1 - 1) \leq \frac{\pi_2}{a_2}(b - a_1(u_1 - 1))$, then $\Phi_1(d)$ is super-additive. □

The converse of Proposition 7.32 need not be true since Family 1 is defined on $[0, \infty)$ while $\Phi_1(d)$ is defined only for $d \in [0, b]$; $\Phi_1(d)$ may be super-additive even when the condition is not satisfied. Regardless, the following functions (derived from f'_1 and f'_2, see Section 7.4.3.1) are lower bounds of $\Phi_1(d)$ and are always super-additive.

From f_1', we obtain the super-additive lower bound $\phi_{1A}(d) = (h_1(d))^+$, where

$$h_1(d) = \begin{cases} \pi_0 - \pi_1 u_1 - \frac{\pi_2}{a_2}(b - a_1 u_1 - d) & if\ 0 \le d \le b - a_1 u_1, \\ \pi_0 - \pi_1 k - \frac{\pi_2}{a_2}(b - k a_1 - d) & if\ -a_1 + \rho' \le -b + k a_1 + d \le 0, \\ \pi_0 - \pi_1 (k+1) - \alpha'(b - (k+1)a_1 - d) & if\ 0 \le -b + (k+1)a_1 + d \le \rho', \end{cases}$$

for $0 \le k < u_1$, $\alpha' = \frac{\pi_1(u_1+1) - \pi_0}{a_1(u_1+1) - b}$, and $\rho' = \frac{\pi_1 a_2 - \pi_2 a_1}{a_2 \alpha' - \pi_2}$. From f_2', we obtain the super-additive lower bound $\phi_{1B}(d) = (h_2(d))^+$ where, for $0 \le k < u_1$,

$$h_2(d) = \begin{cases} \pi_0 - \pi_1 u_1 - \frac{\pi_2}{a_2}(b - a_1 u_1 - d) & if\ 0 \le d \le b - a_1 u_1, \\ \pi_0 - \pi_1 k - \frac{\pi_0 - \pi_1(u_1 - 1)}{b - a_1(u_1 - 1)}(b - k a_1 - d) & if\ b - (k+1)a_1 < d \le b - k a_1. \end{cases}$$

Third, we derive super-additive lower bounds for $\Phi_2(d)$. We define $k' = \max\{k : \pi_0 - \pi_1 u_1 - \pi_2 k > 0\}$; i.e., $k' = \lceil \frac{\pi_0 - \pi_1 u_1}{\pi_2} \rceil - 1$. For $d \le b - a_1 u_1$, $\Phi_2(d)$ is an instance of Family 2 parametrized by $\delta = 0$, $c = \pi_0 - \pi_1 u_1 - \pi_2 k'$, $\alpha = \frac{\pi_1}{a_1} \ge \frac{\pi_2}{a_2}$, $e = b - a_1 u_1 - k' a_2$, $\beta = a_2$, $\gamma = \pi_2$ and $\rho = \pi_2 \frac{\pi_1}{a_1}$, see Section 7.4.3.2. Since $\Phi_2(0) \le 0$, we have $\pi_0 - \pi_1 u_1 - \pi_2(k'+1) \le (b - a_1 u_1 - (k'+1)a_2)\frac{\pi_1}{a_1}$. Since $\Phi_2(d)$ is an upper bound to this instance for $d > b - a_1 u_1$, we can use all the results derived for Family 2.

Thus, from Section 7.4.3.2, we know that $\Phi_2(d)$ is super-additive if $\pi_0 - \pi_1 u_1 - \pi_2(k'+2) \le (b - a_1 u_1 - (k'+2)a_2)\frac{\pi_1}{a_1} \ge \frac{\pi_2}{a_2}$. Again, the converse is not necessarily true. Nevertheless, the following functions, which are derived from g_1' and g_2' (see Section 7.4.3.2), are super-additive lower bounds for Φ_2. From g_1', we obtain the super-additive lower bound

$$\phi_{2A}(d) = \begin{cases} (h_1'(d))^+ & if\ 0 \le d \le b - a_1 u_1, \\ \Phi_2(d) & if\ d > b - a_1 u_1, \end{cases}$$

where, for $0 \le \ell < u_2$, $\alpha' = \frac{\pi_2(k'+2) + \pi_1 u_1 - \pi_0}{(k'+2)a_2 + a_1 u_1 - b}$ and $\rho' = \pi_2/\alpha'$, we have

$$h_1'(d) = \begin{cases} \pi_0 - \pi_1 u_1 - \pi_2 \ell & if\ \ell a_2 \le b - a_1 u_1 - d \le (\ell+1)a_2 - \rho', \\ \pi_0 - \pi_1 u_1 - (\ell+1)(\pi_2 - \alpha' a_2) - \alpha'(b - d - a_1 u_1) & \\ & if\ -\rho' \le b - a_1 u_1 - d - (\ell+1)a_2 \le 0. \end{cases}$$

From g_2', we obtain the super-additive lower bound

$$\phi_{2B}(d) = \begin{cases} (h_2'(d))^+ & if\ 0 \le d \le b - a_1 u_1, \\ \Phi_2(d) & if\ d > b - a_1 u_1, \end{cases}$$

where, for $0 \leq \ell < u_2$, $e' = b - a_1u_1 - k'a_2$, $c' = \pi_0 - \pi_1u_1 - \pi_2k'$, and $\rho' = (\pi_2e' - c'a_2)/(e'\frac{\pi_1}{a_1} - c')$, we have

$$h'_2(d) = \begin{cases} \pi_0 - (k'+1)(\pi_2 - a_2\frac{\pi_1}{a_1}) - \frac{\pi_1}{a_1}\ (b-d) \\ \qquad\qquad if\ d \leq b - a_1u_1 - (k'+1)a_2 + \rho', \\ \pi_0 - \pi_1u_1 - \pi_2\ell - \frac{c'}{e'}(b - a_1u_1\ - \ell a_2 - d) \\ \qquad\qquad if\ \ell a_2 \leq b - a_1u_1 - d \leq (\ell+1)a_2 - \rho', \\ \pi_0 - (\ell+1)(\pi_2 - a_2\frac{\pi_1}{a_1}) - \frac{\pi_1}{a_1}(b-d) \\ \qquad\qquad if\ \ -\rho' \leq b - a_1u_1 - d - (\ell+1)a_2 \leq 0. \end{cases}$$

Again, we present the special case where $\pi_2 = 0$. As shown in Section 7.5.1.2, we can completely characterize Φ_{IP} in closed form since it is equal to Φ_1. The following results can easily be obtained by setting $\pi_2 = 0$ in our analysis of Φ_1. However, this case merits special attention since one of the super-additive lower bounds developed is exactly the well-known MIR inequality.

This function belongs to a special case of both Family 1 and Family 2 with $\rho = 0$ and $\delta = 0$, see Sections 7.4.3.1 and 7.4.3.2. Instead of repeating the analysis from Section 7.4, as we did for Φ_1, we present the results.

Φ_{IP} is super-additive if $b \geq a_1(u_1 + 1)$. We have proved the following.

Proposition 7.33 If $x_1 \leq u_1$ defines a facet of $\mathrm{conv}(K^{\leq}_2(b))$, then the lifted inequality defines a facet of $\mathrm{conv}(K^{\leq}_n(b))$ if $b \geq a_1(u_1 + 1)$. □

When this is not true, the function may still be super-additive. Nevertheless, the following lower bounds are super-additive.

Let $\rho' = a_1(u_1 + 1) - b$ and $\alpha' = 1/\rho'$. From f'_1 and g'_1, we obtain the super-additive lower bound

$$\phi_{1A}(d) = \begin{cases} 0 & if\ 0 \leq d \leq b - a_1u_1, \\ u_1 - k & if\ -(k+1)a_1 + \rho' \leq d - b \leq -ka_1, \\ u_1 - (k+1) + \alpha'(d - b + (k+1)a_1) & if\ 0 \leq d - b - (k+1)a_1 \leq \rho', \end{cases}$$

for $k \in [0, u_1 - 1]$. This is exactly the same as the MIR inequality if $u_1 = \lfloor b/a_1 \rfloor$; obtained by scaling the pure-integer knapsack that describes $K^{\leq}_n(b)$ by a_1.

Let $\delta' = b - a_1(u_1 - 1)$. For $k \in [0, u_1 - 1]$, from f_2' and g_2', we obtain the super-additive lower bound

$$\phi_{1B}(d) = \begin{cases} 0 & if\ 0 \leq d \leq b - a_1 u_1, \\ u_1 - k - \delta'(b - ka_1 - d) & if\ b - (k+1)a_1 < d \leq b - ka_1. \end{cases}$$

7.5.1.4 Comparing super-additive lower bounds: A numerical example

Next, we present a small example to prove that the super-additive functions in Section 7.5.1.3 are different from those presented in Agra and Constantino (2003). This example illustrates that neither function dominates the other. Recall that in Agra and Constantino (2003), the authors develop super-additive lifting functions from the group relaxation of the exact lifting function.

We consider the facet-defining inequality $3x_1 + x_2 \leq 12$ of the set $K_2^{\leq}(23) = \{x_1, x_2 \in \mathbb{Z}_+ : 5x_1 + 2x_2 \leq 21,\ x_1 \leq 4,\ x_2 \leq 10\}$. This instance is actually obtained from the example in Figure 7.2 after translating the axes by variable substitution to eliminate the upper bound constraints; see Section 7.2.1.

For this example, we have $\pi_1 = 3, \pi_2 = 1, \pi_0 = 12$; $a_1 = 5, a_2 = 2$; $u_1 = 4, u_2 = 10$; and $b = 21$. Thus, we have $b - a_1 u_1 = 1$ and $\pi_0 - \pi_1 u_1 = 0$. Therefore, $\Phi = \Phi_1'$, which can be characterized in closed form as

$$\Phi_1'(d) = \begin{cases} 0 & if\ 0 \leq d \leq 1, \\ 3k + 1/2 + (d - 5k - 1)/2 & if\ 5k + 1 < d \leq 5(k+1) + 1, \end{cases}$$

for $k \in [0, u_1 - 1]$. Since $b = 23 < 25 = a_1(u_1 + 1)$ and $\pi_0 - \pi_1(u_1 - 1) = 3 \leq 4\frac{\pi_2}{a_2}(b - a_1(u_1 - 1))$, by Proposition 7.32, $\Phi_1'(d)$ is super-additive. In Agra and Constantino (2003), the authors present a succession of super-additive lower bounds, of which the largest for this example is ψ_2', and is characterized in closed form, for $k \in [0, u_1 - 1]$, as

$$\psi_2'(d) = \begin{cases} 0 & if\ 0 \leq d \leq 1, \\ 3k + (d - 5k - 1) & if\ 5k + 1 \leq d \leq 5k + 2, \\ 3k + 1 + (d - 5k - 2)/2 & if\ 5k + 2 \leq d < 5(k+1), \\ 3(k+1) & if\ 5(k+1) \leq d \leq 5(k+1) + 1, \end{cases}$$

Observe that $\Phi_1'(d) < \psi_2'(d)$ for $5(k+1) < d < 5(k+1)+1$, $\Phi_1'(d) > \psi_2'(d)$ for $5k+1 \leq d < 5k+2$, and $\Phi_1'(d) = \psi_2'(d)$ otherwise. Thus, these two super-additive lifting functions do not dominate one another.

7.5.2 Mixed-integer knapsack set

Now, we study the exact lifting function for facets of the convex hull of the mixed-integer set with two integer variables and one continuous variable, denoted by $M_2^{\leq}(b)$; introduced in Section 7.3. Denoting facets of $\mathrm{conv}(M_2^{\leq}(b))$ by

$$\pi_1 x_1 + \pi_2 x_2 - y \leq \pi_0, \tag{7.4}$$

we repeat assumptions (A.0) and (A.2)-(A.4) from Section 7.2 and assumptions (A.5)-(A.6) from Section 7.5.1. We can also assume that $\pi_1, \pi_2, \pi_0 \in \mathbb{Z}_+$, from Lemma 7.17.

7.5.2.1 Value functions

We define the exact value function

$$w_{IP}(d) = \max\{\pi_1 x_1 + \pi_2 x_2 - y \,:\, x_1, x_2, y \in M_2^{\leq}(d)\}.$$

In all optimal solutions to w_{IP}, if $a_i = 0$ or $u_i = 0$, we have $x_i = u_i,\ i = 1, 2$; if $\pi_2 = 0$, we have $x_2 = 0$. Thus, if any of these variables is zero, then the value function can be characterized in closed form for all d since the problem reduces to a problem with one integer variable and one continuous variable, see Atamtürk (2003b). Hence, we repeat assumptions (A.2) and (A.5). We also assume that (A.7) is true, since (7.4) is not a face of $\mathrm{conv}(M_2^{\leq})$ if $\pi_0 > \pi_1 u_1 + \pi_2 u_2$.

In the rest of Section 7.5.2, we define several upper bounds to the value function and use them to derive lower bounds for the lifting function for (7.4). Whenever $\pi_i > a_i$, $x_i = u_i\ i = 1, 2$ in all optimal solutions to w_{IP}; however we treat these cases explicitly, for the sake of completeness.

Analogous to Section 7.5.1, we first consider the LP relaxation of the exact value function, which we denote by w_{LP}. Then, we describe stronger upper bounds for the

exact lifting function (w_{IP}) by considering the following relaxations of w_{IP}: x_2 continuous (w_1), and x_1 continuous (w_2). Formally, these functions are defined as

$$w_{LP}(d) = \max\{\pi_1 x_1 + \pi_2 x_2 - y : x_1, x_2, y \in \mathbb{R}_+,\ a_1 x_1 + a_2 x_2 - y \leq d,\ \ x \leq u\},$$
$$w_1(d) = \max\{\pi_1 x_1 + \pi_2 x_2 - y : x_1 \in \mathbb{Z}_+, x_2, y \in \mathbb{R}_+,\ a_1 x_1 + a_2 x_2 - y \leq d,\ \ x \leq u\},$$
$$w_2(d) = \max\{\pi_1 x_1 + \pi_2 x_2 - y : x_1, y \in \mathbb{R}_+, x_2 \in \mathbb{Z}_+,\ a_1 x_1 + a_2 x_2 - y \leq d,\ \ x \leq u\}.$$

Again, since w_1 and w_2 are upper bounds on w_{IP}, we can define a stronger upper bound

$$w(d) = \min\{w_1(d), w_2(d)\}.$$

Each of these functions can be characterized in closed form. Depending on the values of $\frac{\pi_1}{a_1}$ and $\frac{\pi_2}{a_2}$, we treat the following two cases separately. For ease of exposition, we first make two simplifying assumptions: $a_1 u_1 \geq \frac{a_1}{\pi_1}\pi_2$ and $a_2 u_2 \geq a_2 \frac{a_1 - \pi_1}{a_2 - \pi_2}$; we subsequently drop these assumptions.

7.5.2.1.1 Case 1: $\frac{\pi_1}{a_1} \leq 1$

The LP relaxation w_{LP} is easy to characterize in closed form, see Figure 7.6. When $d < 0$, $y = -d$ to ensure feasibility. Since $\frac{\pi_2}{a_2} \leq \frac{\pi_1}{a_1} \leq 1$, both x_1 and x_2 are zero. When d is non-negative, only x_1 is used in the optimal solution while $d \leq a_1 u_1$ since $\frac{\pi_1}{a_1} \geq \frac{\pi_2}{a_2}$, by assumption (A.6).

For larger values of d, we set $x_2 = d - a_1 u_1$ and x_1 to its maximum value u_1. For $d \geq a_1 u_1 + a_2 u_2$, both x_1 and x_2 are set to their upper bounds. Since $\frac{\pi_1}{a_1} \leq 1$, y is never used except to ensure feasibility if $d < 0$.

$$w_{LP}(d) = \begin{cases} d & if\ d \leq 0, \\ \frac{\pi_1}{a_1} d & if\ 0 \leq d \leq a_1 u_1, \\ \pi_1 u_1 + \frac{\pi_2}{a_2}(d - a_1 u_1) & if\ a_1 u_1 \leq d \leq a_1 u_1 + a_2 u_2, \\ \pi_1 u_1 + \pi_2 u_2 & if\ a_1 u_1 + a_2 u_2 \leq d. \end{cases}$$

Now, w_{LP} can also be characterized in closed form. As before, when $d < 0$, the optimal solution is obtained by setting $y = -d$ and $x = 0$. When $0 \leq d \leq a_1 u_1$, for all integral multiples of a_1, the optimal solution sets x_1 to the integral multiple.

For intermediate values of d, the objective can be increased in two ways, after setting x_1 to the largest possible integral multiple of a_1. First, we can let $x_2 = (d - a_1x_1)/a_2$. Second, one can increment x_1 to its next integral multiple by increasing y. This increases the objective if $r(d, a_1) \geq a_1 - \pi_1$. For d such that $r(d, a_1) = a_1 - a_2\frac{a_1-\pi_1}{a_2-\pi_2}$, let $k = \lfloor d/a_1 \rfloor$. There exist two optimal solutions for such d: $x_1 = k, x_2 = \frac{a_1-\pi_1}{a_2-\pi_2}, y = 0$; and $x_1 = k + 1, x_2 = 0, y = a_1 - a_2\frac{a_1-\pi_1}{a_2-\pi_2}$. Considering the maximum of the two cases, we obtain the value function for $0 \leq d \leq a_1u_1$.

For $d \geq a_1u_1$, the remainder after x_1 has been set to its upper bound (u_2) is allocated to x_2. When $d \geq a_1u_1 + a_2u_2$, both x_1 and x_2 are set to their upper bounds u_1 and u_2, respectively. Unless a marginal increase in y permits us to increment the integral x_1 to its next value, y is set to zero; see Figure 7.6. Thus, for $0 \leq k < u_1$,

$$w_1(d) = \begin{cases} d & if\ d \leq 0, \\ \pi_1 k + \frac{\pi_2}{a_2}(d - ka_1) & if\ ka_1 \leq d \leq ka_1 + a_2\frac{a_1-\pi_1}{a_2-\pi_2}, \\ (k+1)(\pi_1 - a_1) + d & if\ ka_1 + a_2\frac{a_1-\pi_1}{a_2-\pi_2} \leq d \leq (k+1)a_1, \\ \pi_1 u_1 + \frac{\pi_2}{a_2}(d - a_1u_1) & if\ a_1u_1 \leq d \leq a_1u_1 + a_2u_2, \\ \pi_1 u_1 + \pi_2 u_2 & if\ a_1u_1 + a_2u_2 \leq d. \end{cases}$$

w_2 is obtained by restricting x_2 to integral values, while $x_1, y \in \mathbb{R}_+$. Thus, it is a relaxation of w_{IP}. On the other hand, w_{LP} is a relaxation of w_2. As with both w_{LP} and w_1, $y = -d$ and $x = 0$ for all optimal solutions if $d \leq 0$. When d is non-negative and smaller than a_1u_1, the optimal solution is obtained by setting x_1 to the maximum possible value since $1 \geq \frac{\pi_1}{a_1} \geq \frac{\pi_2}{a_2}$.

When $d \geq a_1u_1$, for all d such that $d - a_1u_1$ is an integral multiple of a_2, we obtain the optimal solution by first setting x_1 to its maximum possible value (u_1) and then $x_2 = (d - a_1u_1)/a_2$. For intermediate values of d, the objective can be increased linearly by incrementing x_2 by one of the following methods: Decreasing x_1 if $r(d - a_1u_1, a_2) > a_2 - \pi_2\frac{a_1}{\pi_1}$; or increasing y if $r(d - a_1u_1, a_2) > a_2 - \pi_2$. Since $\frac{a_1}{\pi_1} \geq 1$, the first dominates the second. For d such that $r(d - a_1u_1, a_2) = a_2 - \pi_2\frac{a_1}{\pi_1}$, there exist two optimal solutions: $x_1 = u_1, x_2 = \ell, y = 0$; and $x_1 = u_1 - \frac{\pi_2}{\pi_1}, x_2 = \ell + 1, y = 0$. Finally, for $d \geq a_1u_1 + a_2u_2$,

both x_1 and x_2 are set to their upper bounds, see Figure 7.6. Thus, for $0 \leq \ell < u_2$, we get

$$w_2(d) = \begin{cases} d & if\ d \leq 0, \\ \frac{\pi_1}{a_1} d & if\ 0 \leq d \leq a_1 u_1, \\ \pi_1 u_1 + \pi_2 \ell & if\ \ell a_2 \leq d - a_1 u_1 \leq (\ell+1) a_2 - \pi_2 \frac{a_1}{\pi_1}, \\ \frac{\pi_1}{a_1}(d - (\ell+1)a_2) + \pi_2(\ell+1) & if\ -\pi_2 \frac{a_1}{\pi_1} \leq d - a_1 u_1 - (\ell+1) a_2 \leq 0, \\ \pi_1 u_1 + \pi_2 u_2 & if\ a_1 u_1 + a_2 u_2 \leq d. \end{cases}$$

Figure 7.6: Value functions: Mixed-integer knapsack set

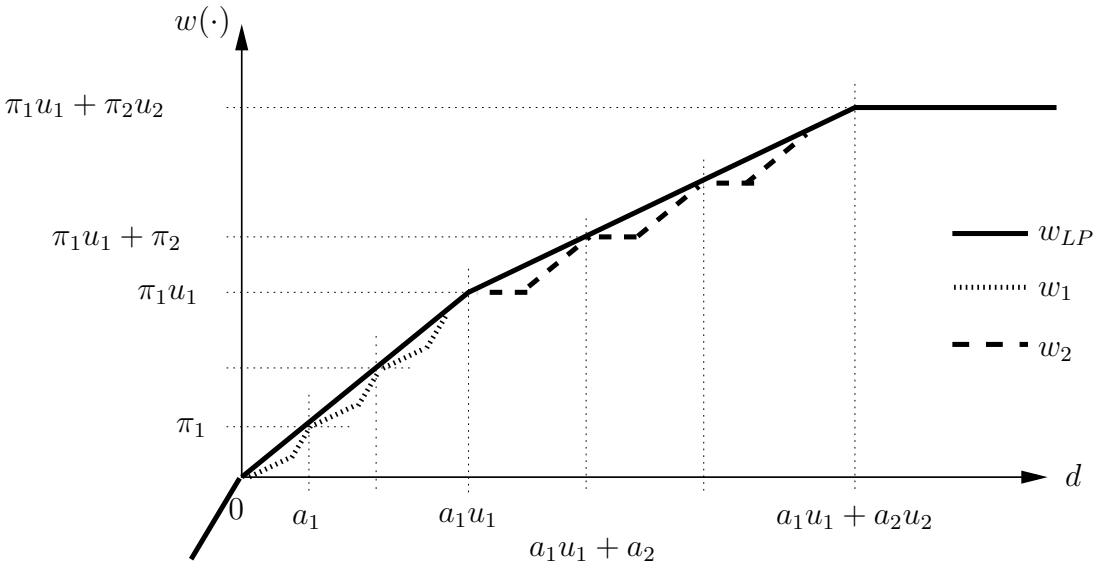
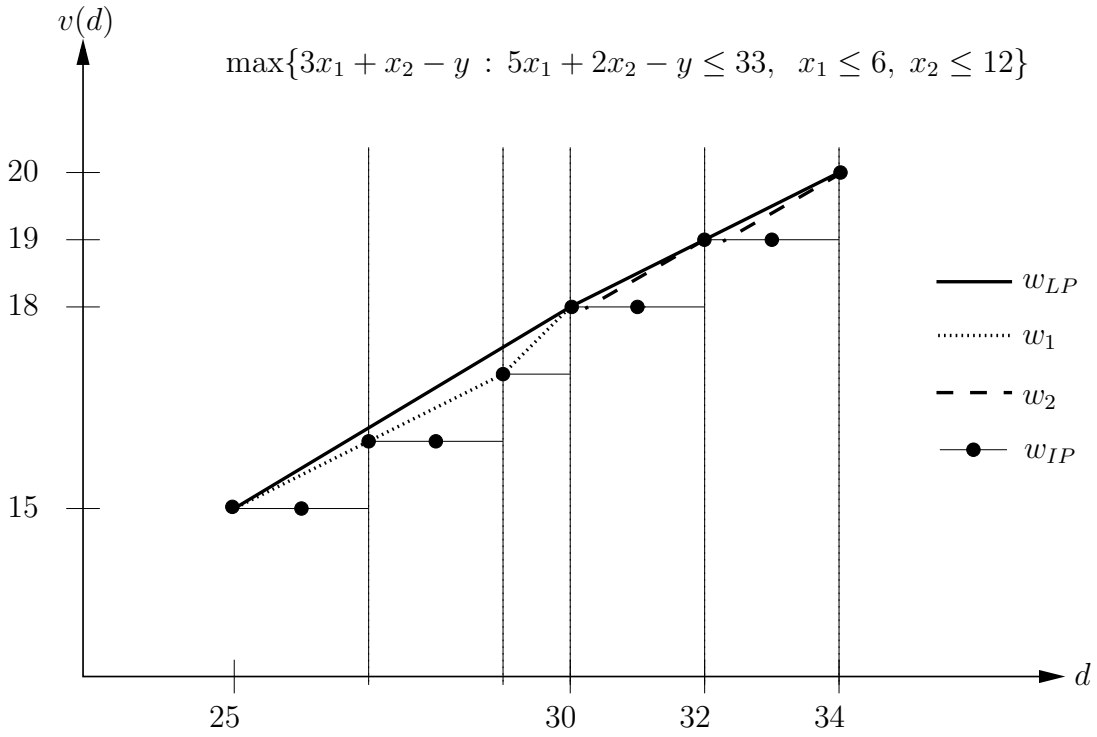


Figure 7.6 presents the form of the various upper bounds to the value functions for a generic mixed-integer knapsack set $M_2^{\leq}(b)$. Next, we present a numerical example that illustrates these upper bounds; using the following mixed-integer set, obtained by adding a continuous variable to the pure-integer set illustrated in Figure 7.2.

We consider $M_2^{\leq}(33) = \{x_1, x_2 \in \mathbb{Z}_+ : 5x_1 + 2x_2 - y \leq 33,\ x_1 \leq 6,\ x_2 \leq 12\}$, and the facet $3x_1 + x_2 - y \leq 19$. Thus, we have $\pi_1 = 3, \pi_2 = 1$; $a_1 = 5, a_2 = 2$; $u_1 = 6, u_2 = 12$; and $b = 33$. We calculated the exact lifting function for this facet. Since w_1, w_2 are different from w_{LP} for values of d on either side of $a_1 u_1 = 30$, we plot w_{LP}, w_1, w_2, w_{IP} for values of d from 25 to 34 in Figure 7.7. Even though w_1, w_2 may not appear to be strong upper bounds to w_{IP}, they are often the strongest super-additive upper bounds.

Figure 7.7: Value functions for mixed-integer knapsack set: Numerical example



Upper bound $w_1(d)$ equals $w_{LP}(d)$ for $d \geq a_1u_1$, and upper bound $w_2(d)$ equals $w_{LP}(d)$ for $d \leq a_1u_1$. Since w_1 and w_2 are both upper bounds on w_{IP}, we can stronger upper bounds by considering their maximum. For $0 \leq k < u_1$ and $0 \leq \ell < u_2$, we get

$$w(d) = \begin{cases} d & if\ d \leq 0, \\ \pi_1 k + \frac{\pi_2}{a_2}(d - ka_1) & if\ ka_1 \leq d \leq ka_1 + a_2\frac{a_1-\pi_1}{a_2-\pi_2}, \\ (k+1)(\pi_1 - a_1) + d & if\ ka_1 + a_2\frac{a_1-\pi_1}{a_2-\pi_2} \leq d \leq (k+1)a_1, \\ \pi_1u_1 + \pi_2\ell & if\ \ell a_2 \leq d - a_1u_1 \leq (\ell+1)a_2 - \pi_2\frac{a_1}{\pi_1}, \\ \frac{\pi_1}{a_1}(d - (\ell+1)a_2) + \pi_2(\ell+1) & if\ -\pi_2\frac{a_1}{\pi_1} \leq d - a_1u_1 - (\ell+1)a_2 \leq 0, \\ \pi_1u_1 + \pi_2u_2 & if\ a_1u_1 + a_2u_2 \leq d. \end{cases}$$

In the preceding closed-form characterization, w_1 and w_2 were simplified by the assumptions $a_1u_1 \geq \frac{a_1}{\pi_1}\pi_2$ and $a_2u_2 \geq a_2\frac{a_1-\pi_1}{a_2-\pi_2}$, respectively. To accurately model the general case, we now drop these assumptions and strengthen $w(d)$ to $w'(d)$, for $0 \leq k < u_1$ and $0 \leq \ell < u_2$. For $ka_1 \leq d < ka_1 + a_2\frac{a_1-\pi_1}{a_2-\pi_2}$, the remainder after setting $x_1 = k$ is

allocated to x_2 only if $d - ka_1 \leq a_2u_2$. For $(\ell+1)a_2 - \pi_2\frac{a_1}{\pi_1} \leq d - a_1u_1 \leq (\ell+1)a_2$, we can increase $w_2(d)$ (and hence $w(d)$) by incrementing x_2 and reducing x_1 only if $r(d-a_1u_1, a_2) \geq a_2 - a_1u_1$. If this is not the case, then x_2 can be incremented by increasing y. For $\delta_1 = \max\{a_2\frac{a_1-\pi_1}{a_2-\pi_2}, \pi_2u_2 + a_1 - \pi_1\}$, $\delta_2 = \min\{-\pi_2\frac{a_1}{\pi_1}, (\pi_1 - a_1)u_1 - \pi_2\}$, and $\delta_3 = -\min\{\pi_2\frac{a_1}{\pi_1}, a_1u_1\}$, we have

$$w'(d) = \begin{cases} d & if\ d \leq 0, \\ k\pi_1 + \frac{\pi_2}{a_2}\min\{d - ka_1, a_2u_2\} & if\ 0 \leq d - ka_1 \leq \delta_1, \\ (k+1)(\pi_1 - a_1) + d & if\ \delta_1 \leq d - ka_1 \leq a_1, \\ \pi_1u_1 + \pi_2\ell & if\ 0 \leq d - a_1u_1 - \ell a_2 \leq a_2 + \delta_2, \\ (d - (\ell+1)a_2) + \pi_2(\ell+1) & if\ \delta_2 \leq d - a_1u_1 - (\ell+1)a_2 \leq \delta_3, \\ \frac{\pi_1}{a_1}(d - (\ell+1)a_2) + \pi_2(\ell+1) & if\ \delta_3 \leq d - a_1u_1 - (\ell+1)a_2 \leq 0, \\ \pi_1u_1 + \pi_2u_2 & if\ a_1u_1 + a_2u_2 \leq d. \end{cases}$$

We can strengthen $w(d)$ for all d where $w_{IP}(d)$ can be characterized in closed form. When $d \leq a_2\frac{a_1-\pi_1}{a_2-\pi_2}$, we set $x_1 = 0$ and x_2 to the largest feasible integral value. Furthermore, we can increment x_2 by increasing y if this increases the objective. Thus, for $0 \leq \ell' < u_2$, we have

$$w_{IP}(d) = \begin{cases} \max\{\pi_2\ell', \pi_1 - a_1 + d\} & if\ \ell'a_2 \leq d \leq (\ell'+1)a_2 - \pi_2, \\ \max\{(\ell'+1)(\pi_2 - a_2) + d, \pi_1 - a_1 + d\} & if\ -\pi_2 \leq d - (\ell'+1)a_2 \leq 0, \\ \max\{\pi_2u_2, \pi_1 - a_1 + d\} & if\ a_2u_2 \leq d \leq \frac{a_1-\pi_1}{a_2-\pi_2}. \end{cases}$$

When $d \geq a_1u_1 + a_2u_2 - \pi_2\frac{a_1}{\pi_1}$, any reduction of d from $a_1u_1 + a_2u_2$ decrements the value of x_1 from u_1. However, x_1 can be incremented by increasing y if this improves the objective. Defining $\delta = a_1u_1 + a_2u_2$, for $0 \leq k' < u_1$, we have

$$w_{IP}(d) = \begin{cases} \pi_1u_1 + \pi_2u_2 - \min\{\pi_2, \pi_1k' + \delta - d - k'a_1\} & if\ 0 \leq \delta - d - k'a_1 \leq \pi_1, \\ \pi_1u_1 + \pi_2u_2 - \min\{\pi_2, \pi_1(k'+1)\} & if\ \pi_1 \leq \delta - d - k'a_1 \leq a_1, \\ \pi_1u_1 + \pi_2u_2 - \min\{\pi_2, \pi_1u_1 + a_2u_2 - d\} & if\ a_1u_1 \leq \delta - d \leq \pi_2\frac{a_1}{\pi_1}. \end{cases}$$

7.5.2.1.2 Case 2: $\frac{\pi_1}{a_1} > 1$

For $\frac{\pi_1}{a_1} > 1$, we first consider the sub-case $\frac{\pi_2}{a_2} \leq 1$. Again, w_{LP} is very easy to characterize

in closed form. Since $\frac{\pi_1}{a_1} > 1$, x_1 is set to its maximum value (u_1) in all optimal solutions by increasing y if $d \leq a_1 u_1$. For larger values of d, x_1 continues to be u_1, while the remainder is allocated to x_2. Since $\frac{\pi_2}{a_2} \leq 1$, y is not used to increase the value of x_2. When $d \geq a_1 u_1 + a_2 u_2$, both x_1 and x_2 are set to their maximum values u_1 and u_2, respectively. We get

$$w_{LP}(d) = \begin{cases} u_1(\pi_1 - a_1) + d & if\ d \leq a_1 u_1, \\ \pi_1 u_1 + \frac{\pi_2}{a_2}(d - a_1 u_1) & if\ a_1 u_1 \leq d \leq a_1 u_1 + a_2 u_2, \\ \pi_1 u_1 + \pi_2 u_2 & if\ a_1 u_1 + a_2 u_2 \leq d. \end{cases}$$

Now, w_{LP} is obtained as the relaxation of w_1 where x_1 is continuous. However, since $x_1 = u_1$ in the optimal solution of v_{LP} for all d, restricting x_1 to be integral does not change the value function in any way. Thus, $w_1(d) = w_{LP}(d)$ for all d.

$$w_1(d) = \begin{cases} u_1(\pi_1 - a_1) + d & if\ d \leq a_1 u_1, \\ \pi_1 u_1 + \frac{\pi_2}{a_2}(d - a_1 u_1) & if\ a_1 u_1 \leq d \leq a_1 u_1 + a_2 u_2, \\ \pi_1 u_1 + \pi_2 u_2 & if\ a_1 u_1 + a_2 u_2 \leq d. \end{cases}$$

We obtain w_2 by restricting only x_2 to be integral; thus $x_1 \in \mathbb{R}_+$. Again, since $\frac{\pi_1}{a_1} > 1$, x_1 is set to its upper bound u_1 in all optimal solutions. When $d \leq a_1 u_1$, this is achieved by increasing y. When $d \geq a_1 u_1$, x_1 is first set to its upper bound (u_1). For all d such that the remainder $d - a_1 u_1$ is an integral multiple of a_2, the optimal solution is obtained by setting x_2 to this integral multiple. For intermediate values of d, consider d such that $r(d - a_1 u_1, a_2) = a_2 - \pi_2$. For such d, there exist two optimal solutions: $x_1 = u_1, x_2 = \ell, y = 0$; and $x_1 = u_1, x_2 = \ell + 1, y = \pi_2$. Therefore, the objective value can be improved if $r(d - a_1 u_1, a_2) \geq a_2 - \pi_2$ by increasing y to increment x_2 to the next integral value. For $d \geq a_1 u_1$, both x_1 and x_2 are set at their upper bounds u_1 and u_2, respectively. Thus, for $0 \leq \ell < u_2$, we get

$$w_2(d) = \begin{cases} u_1(\pi_1 - a_1) + d & if\ d \leq a_1 u_1, \\ \pi_1 u_1 + \pi_2 \ell & if\ \ell a_2 \leq d - a_1 u_1 \leq (\ell + 1)a_2 - \pi_2, \\ u_1(\pi_1 - a_1) + (\ell + 1)(\pi_2 - a_2) + d & if\ -\pi_2 \leq d - a_1 u_1 - (\ell + 1)a_2 \leq 0, \\ \pi_1 u_1 + \pi_2 u_2 & if\ a_1 u_1 + a_2 u_2 \leq d. \end{cases}$$

Both w_1 and w_2 are upper bounds on w_{IP}; we get a stronger upper bound by considering their minimum w. In this case, $w_2(d) \geq w_1(d) = w_{LP}(d)$, $\forall d$, and thus $w(d) = w_2(d)$.

$$w(d) = \begin{cases} u_1(\pi_1 - a_1) + d & if\ d \leq a_1 u_1, \\ \pi_1 u_1 + \pi_2 \ell & if\ \ell a_2 \leq d - a_1 u_1 \leq (\ell+1)a_2 - \pi_2, \\ u_1(\pi_1 - a_1) + (\ell+1)(\pi_2 - a_2) + d & if\ -\pi_2 \leq d - a_1 u_1 - (\ell+1)a_2 \leq 0, \\ \pi_1 u_1 + \pi_2 u_2 & if\ a_1 u_1 + a_2 u_2 \leq d. \end{cases}$$

In the optimal solution to $w_2(d)$, for all d, we have x_1 set to an integral value (its upper bound). Thus, restricting x_1 also to be integral does not change the value function for any d. This implies that w_{IP} can be characterized in closed form if $\frac{\pi_1}{a_1} > 1 \geq \frac{\pi_2}{a_2}$, and is equal to $w_2(d)$.

Second, we consider the sub-case where $\frac{\pi_2}{a_2} > 1$. This is the easiest to characterize in closed form since both x_1 and x_2 are set to their upper bounds in all optimal solutions, irrespective of whether x_1 or x_2 is forced to integral. When $d \leq a_1 u_1 + a_2 u_2$, this is done by increasing y to ensure feasibility. When $d \geq a_1 u_1 + a_2 u_2$, y is set to zero. We have

$$w_{LP}(d) = \begin{cases} u_1(\pi_1 - a_1) + u_2(\pi_2 - a_2) + d & if\ d \leq a_1 u_1 + a_2 u_2, \\ \pi_1 u_1 + \pi_2 u_2 & if\ a_1 u_1 + a_2 u_2 \leq d. \end{cases}$$

As mentioned earlier, restricting x_1 to be integral does not change the value function since x_1 is set to u_1 for all values of d. Thus, we have $w_1(d) = w_{LP}(d)$.

$$w_1(d) = \begin{cases} u_1(\pi_1 - a_1) + u_2(\pi_2 - a_2) + d & if\ d \leq a_1 u_1 + a_2 u_2, \\ \pi_1 u_1 + \pi_2 u_2 & if\ a_1 u_1 + a_2 u_2 \leq d. \end{cases}$$

Similarly, restricting x_2 to take only integral values does not reduce the value function. We have $x_2 = u_2$ in all optimal solutions, and thus $w_2(d) = w_{LP}(d)$.

$$w_2(d) = \begin{cases} u_1(\pi_1 - a_1) + u_2(\pi_2 - a_2) + d & if\ d \leq a_1 u_1 + a_2 u_2, \\ \pi_1 u_1 + \pi_2 u_2 & if\ a_1 u_1 + a_2 u_2 \leq d. \end{cases}$$

We've seen that $w_2(d) = w_1(d) = w_{LP}(d)$, and thus we have $w(d) = w_{LP}(d)$.

$$w(d) = \begin{cases} u_1(\pi_1 - a_1) + u_2(\pi_2 - a_2) + d & if\ d \leq a_1 u_1 + a_2 u_2, \\ \pi_1 u_1 + \pi_2 u_2 & if\ a_1 u_1 + a_2 u_2 \leq d. \end{cases}$$

Since both $x_1 = u_1$ and $x_2 = u_2$ in the optimal solutions to $v_{LP}(d)$ for all d, restricting both x_1 and x_2 to be integral also does not change the value function. Thus, in this case w_{IP} can be characterized in closed form and is equal to w_{LP}. Next, we use these value functions to develop several lower bounds for the exact lifting function for (7.4).

7.5.2.2 Lifting functions

We denote the exact lifting function for (7.4) by $\Psi_{IP}(d)$. For $d \geq 0$, we have

$$\Psi_{IP}(d) = \pi_0 - \max\{\pi_1 x_1 + \pi_2 x_2 - y \,:\, x_1, x_2, y \in M_2^{\leq}(b-d)\}.$$

Again, the exact lifting function can be defined in terms of the exact value function as

$$\Psi_{IP}(d) = \pi_0 - w_{IP}(b-d).$$

We first study several lower bounds for Ψ_{IP}; Ψ_1, Ψ_2 and Ψ are defined similarly using the various upper bounds for w_{IP}. Again, depending on the values of $\frac{\pi_1}{a_1}$ and $\frac{\pi_2}{a_2}$, we treat the following cases separately. The following closed-form characterizations are defined for $d \geq 0$ and $b \geq a_1 u_1$. When $b < a_1 u_1$, the analysis is similar, and is omitted here.

7.5.2.2.1 Case 1: $\frac{\pi_1}{a_1} \leq 1$

We characterize in closed form the case where $\frac{\pi_1}{a_1} \leq 1$, starting with the LP relaxation Ψ_{LP}, and then Ψ_1, Ψ_2, and Ψ. Deriving these lifting functions from the corresponding value functions, for $0 \leq k < u_1$, $0 \leq \ell < u_2$, $d \geq 0$, and $\hat{b} = b - a_1 u_1$, we have

$$\Psi_{LP}(d) = \begin{cases} \pi_0 - \pi_1 u_1 - \frac{\pi_2}{a_2}(b - a_1 u_1 - d) & if\ 0 \leq d \leq (b - a_1 u_1)^+, \\ \pi_0 - \frac{\pi_1}{a_1}(b-d) & if\ b - a_1 u_1 \leq d \leq b, \\ \pi_0 + d - b & if\ b \leq d. \end{cases}$$

$$\Psi_1(d) = \begin{cases} \pi_0 - \pi_1 u_1 - \frac{\pi_2}{a_2}(b - a_1 u_1 - d) & if\ 0 \leq d \leq (b - a_1 u_1)^+, \\ \pi_0 - \pi_1 k - \frac{\pi_2}{a_2}(b - k a_1 - d) & if\ k a_1 \leq b - d \leq k a_1 + a_2 \frac{a_1 - \pi_1}{a_2 - \pi_2}, \\ \pi_0 - (k+1)(\pi_1 - a_1) + d - b & if\ k a_1 + a_2 \frac{a_1 - \pi_1}{a_2 - \pi_2} \leq b - d \leq (k+1) a_1, \\ \pi_0 + d - b & if\ b \leq d. \end{cases}$$

$$\Psi_2(d) = \begin{cases} \pi_0 - \pi_1 u_1 - \pi_2 \ell & if\ \ell a_2 \leq \hat{b} - d \leq (\ell+1)a_2 - \pi_2 \frac{a_1}{\pi_1}, \\ \pi_0 + \frac{\pi_1}{a_1}(d-b) - (\ell+1)(\pi_2 - a_2\frac{\pi_1}{a_1}) & if\ -\pi_2\frac{a_1}{\pi_1} \leq \hat{b} - d - (\ell+1)a_2 \leq 0, \\ \pi_0 - \frac{\pi_1}{a_1}(b-d) & if\ b - a_1 u_1 \leq d \leq b, \\ \pi_0 + d - b & if\ b \leq d. \end{cases}$$

$$\Psi(d) = \begin{cases} \pi_0 - \pi_1 u_1 - \pi_2 \ell & if\ \ell a_2 \leq \hat{b} - d \leq (\ell+1)a_2 - \pi_2 \frac{a_1}{\pi_1}, \\ \pi_0 + \frac{\pi_1}{a_1}(d-b) - (\ell+1)(\pi_2 - a_2\frac{\pi_1}{a_1}) & if\ -\pi_2\frac{a_1}{\pi_1} \leq \hat{b} - d - (\ell+1)a_2 \leq 0, \\ \pi_0 - \pi_1 k - \frac{\pi_2}{a_2}(b - ka_1 - d) & if\ ka_1 \leq b - d \leq ka_1 + a_2\frac{a_1-\pi_1}{a_2-\pi_2}, \\ \pi_0 - (k+1)(\pi_1 - a_1) + d - b & if\ ka_1 + a_2\frac{a_1-\pi_1}{a_2-\pi_2} \leq b - d \leq (k+1)a_1, \\ \pi_0 + d - b & if\ b \leq d. \end{cases}$$

Since (7.4) is a valid inequality for $M_2^{\leq}(b)$, we have that $\Psi_{IP} = 0$. Thus, $\Psi_{IP}(d) \geq 0,\ \forall d \geq 0$; therefore all the lower bounds can be strengthened by increasing them to 0 wherever they are negative. Furthermore, as for $\Phi(d)$, $\Psi(d)$ can be strengthened to $\Psi'(d)$ whenever $a_1 u_1 < \frac{a_1}{\pi_1}\pi_2$ or $a_2 u_2 < a_2\frac{a_1-\pi_1}{a_2-\pi_2}$, and wherever Ψ_{IP} can be characterized in closed form.

The special case of $\pi_2 = 0$ studied and characterized in closed form in Atamtürk (2003b) deserves special mention for three reasons. Firstly, the inequality when $\pi_2 = 0$ corresponds to the MIR inequality for the restriction $M_1^{\leq}(b)$ with all variables other than x_2 and y set to 0. Secondly, the exact lifting function for this case can be characterized in closed form even though $\frac{\pi_1}{a_1} \leq 1$. Finally, this exact lifting function is super-additive. Defining $\eta_b = \lceil b/a_1 \rceil$ and $r_b = r(b, a_1)$, we have $\pi_1 = a_1 - r_b$ and $\pi_0 = b - \eta_b r_b$. For $0 \leq k < u_1$, we have

$$\Psi_{IP}(d) = \begin{cases} 0 & if\ 0 \leq d \leq (b - a_1 u_1)^+, \\ b - \eta_b r_b - k(a_1 - r_b) & if\ ka_1 \leq b - d \leq (k+1)a_1 - \pi_1, \\ -\eta_b r_b + (k+1)r_b + d & if\ (k+1)a_1 - \pi_1 \leq b - d \leq (k+1)a_1, \\ d - \eta_b r_b & if\ d \geq b. \end{cases}$$

7.5.2.2.2 Case 2: $\frac{\pi_1}{a_1} > 1$

As before, we first consider sub-case $\frac{\pi_2}{a_2} \leq 1$. When $\frac{\pi_1}{a_1} > 1$, $x_1 = u_1$ in all optimal solu-

tions to Ψ_{IP}. Therefore, relaxing integrality on x_1 while calculating Ψ_2 does not improve the objective. At the same time, constraining x_1 to be integral does not worsen the objective when compared with the LP relaxation. Thus, we have $\Psi_{LP} = \Psi_1$ and $\Psi_2 = \Psi = \Psi_{IP}$. The exact lifting function is therefore characterized in closed form in this case.

$$\Psi_{LP}(d) = \Psi_1(d) = \begin{cases} \pi_0 - \pi_1 u_1 - \frac{\pi_2}{a_2}(b - a_1 u_1 - d) & if\ 0 \leq d \leq (b - a_1 u_1)^+, \\ \pi_0 - u_1(\pi_1 - a_1) + d - b & if\ d \geq b - a_1 u_1. \end{cases}$$

For $0 \leq \ell < u_2$, we have

$$\Psi_{IP}(d) = \begin{cases} \pi_0 - \pi_1 u_1 - \pi_2 \ell & if\ 0 \leq b - d - a_1 u_1 - \ell a_2 \leq a_2 - \pi_2, \\ \pi_0 - u_1(\pi_1 - a_1) - (\ell + 1)(\pi_2 - a_2) + d - b & \\ & if\ -\pi_2 \leq b - d - a_1 u_1 - (\ell + 1)a_2 \leq 0, \\ \pi_0 - u_1(\pi_1 - a_1) + d - b & if\ b - a_1 u_1 \leq d. \end{cases}$$

As before, we can strengthen $\Psi_{(\cdot)}(d)$ to $(\Psi_{(\cdot)}(d))^+$.

Now, we consider the sub-case $\frac{\pi_2}{a_2} > 1$. From Section 7.5.2.1.2, $w_{LP}(d) = w_{IP}(d)$, and therefore $\Psi_{IP}(d)$ can be characterized in closed form as follows.

$$\Psi_{IP}(d) = \begin{cases} \pi_0 - \pi_1 u_1 - \pi_2 u_2 & if\ 0 \leq d \leq b - a_1 u_1 - a_2 u_2, \\ \pi_0 - (\pi_1 - a_1)u_1 - (\pi_2 - a_2)u_2 - b + d & if\ d \geq b - a_1 u_1 - a_2 u_2. \end{cases}$$

As before, we can strengthen $\Psi_{IP}(d)$ to $(\Psi_{IP}(d))^+$ for all d.

Next, we describe the conditions under which these functions are super-additive, and construct super-additive lower bounds when these conditions fail.

7.5.2.3 Super-additive lifting functions

7.5.2.3.1 Case 1: $\frac{\pi_1}{a_1} \leq 1$

For the case where $\frac{\pi_1}{a_1} \leq 1$, we start with the LP relaxation of the exact lifting function, Ψ_{LP}. Since (7.2) defines a facet of $M_2^{\leq}(b)$, we have $w_{IP}(b) = \pi_0$, and thus $\Psi_{IP}(0) = 0$. Therefore, $\Psi_{LP}(0) \leq 0$. From Proposition 7.24, Ψ_{LP} is super-additive since $\Psi_{LP}(0)$ is also convex. However Ψ_{LP} is often a weak relaxation of the exact lifting function, and thus we develop super-additive lower bounds for the partial LP relaxations Ψ_1 and Ψ_2.

We consider Ψ_1 first. Since the slope of $\Psi_1(d)$ is unknown when $\Psi_1(d) = 0$, the function may belong to either Family 1 or Family 2 for $d \leq b$, see Section 7.4. We define $k' = \lfloor \pi_0/\pi_1 \rfloor$ if $b - a_1 u_1 < 0$, and u_1 otherwise.

When $\Psi_1'(d) = \frac{\pi_2}{a_2}$ at $\Psi_1(d) = 0$, then the function is an instance of Family 1 for $d \leq b$, parametrized by $\alpha = 1$, $\beta = a_1$, $\gamma = \pi_1$, $\delta = \frac{\pi_2}{a_2}$, $e = b - k'a_1$, $c = \pi_0 - \pi_1 k'$ and $\rho = (a_2\pi_1 - a_1\pi_2)/(a_2 - \pi_2)$, see Section 7.4.1. Since $\Psi_1(d) \leq 0$, we have $\pi_0 - \pi_1 k \leq (b - k'a_1)\frac{\pi_2}{a_2}$. Also, since $\Psi_1(d)$ is an upper bound to this instance of Family 1 for $d \geq b$, we can use all the results for this instance. Thus, Ψ_1 is super-additive if $b - k'a_1 \geq a_2 \frac{a_1 - \pi_1}{a_2 - \pi_2}$ or $\pi_0 - \pi_1(k' - 1) \leq (b - (k' - 1)a_1)\frac{\pi_2}{a_2}$ or $\pi_0 - \pi_1(k' + 1) \leq b - (k' + 1)a_1$. The converse may not be true; nevertheless the following functions derived from f_1 and f_2 are super-additive lower bounds for Ψ_1.

From f_1, we get

$$\psi_{1A}(d) = \begin{cases} (h_1(d))^+ & if\ 0 \leq d \leq b, \\ \Psi_1(d) & if\ d > b, \end{cases}$$

where, for $0 \leq k < k'$, $\alpha' = \frac{\pi_1(k'+1) - \pi_0}{k'+1)a_1 - b}$, and $\rho' = \frac{a_2\pi_1 - a_1\pi_2}{a_2\alpha' - \pi_2}$, we have

$$h_1(d) = \begin{cases} \pi_0 - \pi_1 k - \frac{\pi_2}{a_2}(b - ka_1 - d) & if\ 0 \leq b - ka_1 - d \leq a_1 - \rho', \\ \pi_0 - \pi_1(k+1) + \alpha'(d - b + (k+1)a_1) & if\ a_1 - \rho \leq b - ka_1 - d \leq a_1. \end{cases}$$

From f_2, we obtain

$$\psi_{1B}(d) = \begin{cases} (\Psi_1(d))^+ & if\ d \leq b - k'a_1, \\ h_2(d) & if\ b - k'a_1 - d \leq b, \\ \Psi_1(d) & if\ d \geq b, \end{cases}$$

where, for $0 \leq k < k'$, $\delta' = \frac{\pi_0 - \pi_1(k'-1)}{b - (k'-1)a_1}$, and $\rho' = \frac{(b - k'a_1)\delta' - (\pi_0 - \pi_1 k')}{1 - \delta'}$, we have

$$h_2(d) = \begin{cases} \pi_0 - \pi_1 k - \delta'(b - ka_1 - d) & if\ ka_1 \leq b - d \leq (k+1)a_1 - \rho', \\ \pi_0 - \pi_1(k+1) + (d - b + (k+1)a_1) & if\ (k+1)a_1 - \rho \leq b - d \leq (k+1)a_1. \end{cases}$$

When $\Psi_1'(d) = 1$ at $\Psi_1(d) = 0$, then the function is an instance of Family 2 for $d \leq b$, parametrized by $\alpha = 1$, $\beta = a_1$, $\gamma = \pi_1$, $\delta = \frac{\pi_2}{a_2}$, $e = b - k'a_1$, $c = \pi_0 - \pi_1 k'$ and $\rho = (a_2\pi_1 - a_1\pi_2)/(a_2 - \pi_2)$, see Section 7.4.2. Since $\Psi_1(d) \leq 0$, we have $\pi_0 - \pi_1(k+1) \leq$

$b-(k'+1)a_1$. Also, since $\Psi_1(d)$ is an upper bound to this instance of Family 2 for $d \geq b$, we can use all the results derived for this instance.

Thus, Ψ_1 is super-additive if $\pi_0 - \pi_1 k' \leq (b-k'a_1)\frac{\pi_2}{a_2}$ or $\pi_0 - \pi_1(k'+2) \leq b-(k'+2)a_1$. As before, the converse may not be true. Nevertheless, the following functions derived from g_1 and g_2 (see Section 7.4.2) are super-additive lower bounds for Ψ_1.

From g_1, we get

$$\psi_{1A}(d) = \begin{cases} (h_1(d))^+ & if\ 0 \leq d \leq b, \\ \Psi_1(d) & if\ d > b, \end{cases}$$

where, for $0 \leq k < k'$, $\alpha' = \frac{\pi_1(k'+2)-\pi_0}{k'+2)a_1-b}$, and $\rho' = \frac{a_2\pi_1 - a_1\pi_2}{a_2\alpha' - \pi_2}$, we have

$$h_1(d) = \begin{cases} \pi_0 - \pi_1 k - \frac{\pi_2}{a_2}(b-ka_1-d) & if\ 0 \leq b-ka_1-d \leq a_1-\rho', \\ \pi_0 - \pi_1(k+1) + \alpha'(d-b+(k+1)a_1) & if\ a_1-\rho \leq b-ka_1-d \leq a_1. \end{cases}$$

From g_2, we obtain

$$\psi_{1B}(d) = \begin{cases} (h_2(d))^+ & if\ 0 \leq d \leq b, \\ \Psi_1(d) & if\ d > b, \end{cases}$$

where, for $0 \leq k < k_1$, $\delta' = \frac{\pi_0 - \pi_1 k'}{b-k'a_1}$, and $\rho' = \frac{\pi_1 - \delta' a_1}{1-\delta'}$, we have

$$h_2(d) = \begin{cases} \pi_0 - \pi_1 k - \delta'(b-ka_1-d) & if\ 0 \leq b-ka_1 d \leq a_1-\rho', \\ \pi_0 - \pi_1(k+1) + (d-b+(k+1)a_1) & if\ a_1-\rho \leq b-ka_1-d \leq a_1. \end{cases}$$

Next, we develop super-additive lower bounds for Ψ_2. If $b \leq a_1 u_1$, then $\Psi_2 = \Psi_{LP}$, and is therefore super-additive. Otherwise, Ψ_2 is exactly the same as Φ_2, which is a lower bound to the exact lifting function for the pure-integer case, see Section 7.5.1.2. Thus, we can use the super-additive lower bounds developed in Section 7.5.1.3.

Again, we mention that for the special case of $\pi_2 = 0$, we can characterize in closed form the exact lifting function; see Section 7.5.2.2. Furthermore, the exact lifting function is super-additive (Atamtürk 2003b).

7.5.2.3.2 Case 2: $\frac{\pi_1}{a_1} > 1$

For the case where $\frac{\pi_2}{a_2} \leq 1$, Ψ_{IP} can be characterized in closed form, see Sec-

tion 7.5.2.2.2.

When $b \leq a_1u_1$, Ψ_{IP} is linear. Since $\Psi_{IP}(0) \leq 0$, Ψ_{IP} is super-additive. Let $k' = \lceil(\pi_0 - \pi_1u_1)/\pi_2\rceil - 1$; When $b > a_1u_1$, for $d \leq b - a_1u_1$, the function is an instance of the special case of Family 2 with $\delta = 0$, parametrized by $\alpha = 1$, $\beta = a_2$, $\gamma = \pi_2$, $e = b - a_1u_1 - k'a_2$, $c = \pi_0 - \pi_1u_1 - \pi_2k'$, $\rho = \pi_2$, and $\delta = 0$; see Section 7.4.3.2. Since $\Psi_{IP} \leq 0$, we have $\pi_0 - \frac{\pi_1}{a_1} \geq \frac{\pi_2}{a_2} - \pi_2(k'+1) \leq b - a_1u_1 - (k'+1)a_2$. Furthermore, $\Psi_{IP}(d)$ is an upper bound to this instance if $d > b - a_1u_1$, and we can use all the results derived for the special case of Family 2.

Thus, Ψ_{IP} is super-additive if $\pi_0 - \pi_1u_1 - (k'+2)\pi_2 \leq b - a_1u_1 - (k'+2)a_2$. We have proved the following.

Proposition 7.34 Let $k' = \lceil(\pi_0 - \pi_1u_1)/\pi_2\rceil - 1$. If $\frac{\pi_1}{a_1} > 1 \geq \frac{\pi_2}{a_2}$, then the lifted inequality is facet-defining for $\mathrm{conv}(M_n^{\leq}(b))$ if $\pi_0 - \pi_1u_1 - (k'+2)\pi_2 \leq b - a_1u_1 - (k'+2)a_2$. □

However, the converse is not necessarily true. Regardless, the following lower bounds derived from g_1 and g_2 are super-additive.

From g_1, we obtain the super-additive lower bound

$$\psi_{2A}(d) = \begin{cases} (h_1(d))^+ & if\ 0 \leq d \leq b - a_1u_1, \\ \Psi_{IP}(d) & if\ d > b - a_1u_1, \end{cases}$$

where, for $0 \leq \ell < k'$, $\alpha' = \frac{\pi_2(k'+2)+\pi_1u_1-\pi_0}{(k'+2)a_2+a_1u_1-b}$ and $\rho' = \pi_2/\alpha'$, we have

$$h_1(d) = \begin{cases} \pi_0 - \pi_1u_1 - \pi_2\ell & if\ 0 \leq b - d - a_1u_1 - \ell a_2 \leq a_2 - \rho', \\ \pi_0 - u_1\pi_1 - \pi_2(\ell+1) + \alpha'(d - b + (\ell+1)a_2 - a_1u_1) & \\ & if\ -\rho' \leq b - d - a_1u_1 - (\ell+1)a_2 \leq 0. \end{cases}$$

From g_2, we obtain the super-additive lower bound

$$\psi_{2B}(d) = \begin{cases} (\Psi_{IP}(d))^+ & if\ 0 \leq d \leq b - a_1u_1 - (k'+1)a_2, \\ (h_2(d))^+ & if\ b - a_1u_1 - (k'+1)a_2 \leq d \leq b - a_1u_1, \\ \Psi_{IP}(d) & if\ d > b - a_1u_1, \end{cases}$$

where, for $0 \leq \ell < k'$, $\delta' = \frac{\pi_0 - \pi_1 u_1 - \pi_2 k'}{b - a_1 u_1 - k' a_2}$, and $\rho' = \frac{\pi_2 - \delta' a_2}{1 - \delta'}$, we have

$$h_2(d) = \begin{cases} \pi_0 - \pi_1 u_1 - \pi_2 \ell - \delta'(b - a_1 u_1 - \ell a_2 - d) \\ \qquad\qquad if\ 0 \leq b - d - a_1 u_1 - \ell a_2 \leq a_2 - \pi_2, \\ \pi_0 - u_1(\pi_1 - a_1) - (\ell + 1)(\pi_2 - a_2) + d - b \\ \qquad\qquad if\ -\pi_2 \leq b - d - a_1 u_1 - (\ell + 1) a_2 \leq 0. \end{cases}$$

When $\frac{\pi_2}{a_2} > 1$, from Section 7.5.2.2.2, Ψ_{IP} can be characterized in closed form. Since $\Psi_{IP}(0) \leq 0$ and Ψ_{IP} is convex, it is super-additive; from Proposition 7.24. We have proved the following.

Proposition 7.35 If $\frac{\pi_1}{a_1} > \frac{\pi_2}{a_2} > 1$, then the lifted inequality is facet-defining for $\text{conv}(M_n^{\leq}(b))$. □

7.6 Computational experiments

Finally, we present computational results that illustrate the effectiveness of our lifted inequalities in a branch-and-cut framework to solve linear optimization problems over pure-integer knapsack sets. We perform these experiments on pure-integer knapsacks. We choose the data from two sources.

The first set of instances (twenty in total) are from Aardal and Lenstra (2002); we refer to them as Dataset 1. We modify the $prob1$-$prob20$ problem instances as follows. We choose only the first five variables from each set. We set the coefficient in the objective function c_i equal to $a_i - 1,\ \forall i = [1, 5]$. They are pure-integer knapsacks with five integer variables, but are still very difficult to solve.

We choose the second set of instances randomly with the same three parameter sets as in Agra and Constantino (2003); four from each set, giving us twelve in total. We choose the parameters for these sets randomly from the intervals described in Table 7.1. They are instances with one hundred integer variables; we refer to them as Dataset 2.

We implement all our computations using CPLEX 9.0 Callable Library on a 2.4GhZ machine with 1GB RAM, and running Linux. We set the maximum number of branch-and-bound nodes to $100,000$, and set the time limit to 300 CPU seconds. To ensure

Table 7.1: Parameters of instances in Dataset 2

set	a_i	c_i	b
A	[1700,2000]	[1700,2000]	[200000,300000]
B	[700,10000]	[700,10000]	[2000000,5000000]
C	[100,200]	[50,60]	[2000,5000]

consistency of results, we disable the CPLEX heuristic.

We add the lifted inequalities only at root node; while we add CPLEX cuts aggressively throughout the tree. To obtain the two-integer restriction, we consider all pairs of variables with nonzero LP solution. We use super-additive lower bounds to the lifting function (see Section 7.5.1.3) for the facets of the two-variable restriction to generate valid inequalities, and add them if they violate the current fractional solution.

First, we test our lifted inequalities in a branch-and-cut algorithm on DataSet 1. In Table 7.2, we report the improvement at the root node (root improvement), the number of branch and bound nodes (b&b nodes), and the solution times (time) or gap at termination (endgap), under headings (1) and (2) for CPLEX default and our lifted inequalities, respectively. We also report the number of added (cuts added).

We see in Table 7.2 that in most of the instances, we outperform CPLEX default in terms of the improvement at the root node. In fact, our cuts reduce the integrality gap at the root node by 7.0% on average, as compared with 3.5% for CPLEX default. Not surprisingly, this manifests itself in the number of branch-and-bound nodes to obtain the optimal solution. Interestingly, these problems are hard to solve, in spite of being very small; five variables and one constraint. Even though it does not take too much time to solve them, the number of branch-and-bound nodes is quite high.

Observe that CPLEX usually finds only two or three violated inequalities, even in its most aggressive setting, which we report here. On the other hand, we add many more of our lifted inequalities, 59 on average. Consequently, a much larger LP is solved at each node of the tree, resulting in the larger solution times (4.4 seconds) as compared with 2.5 seconds for CPLEX default even though the number of nodes is lesser. We need to perform more detailed computational experiments which maintain an active cut pool

Table 7.2: Computations on pure-integer knapsack: Dataset 1

instance	root improvement		b&b nodes		cuts added		time (endgap)	
	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)
1	4.1	33	96303	72292	33	3	3.2	4.1
2	2.9	9.3	9943	6158	60	2	0.4	0.5
3	2.4	2.4	609420	544466	20	3	22	24
4	4.2	4.2	346906	348864	128	3	11	21
5	0.5	1.6	(-)	(-)	56	3	(0.0)	(0.0)
6	1.6	2.1	23475	300122	80	3	0.8	19
7	1.0	3.2	(-)	(-)	102	3	(0.0)	(0.0)
8	5.6	6.7	72234	50657	68	3	2.6	3.0
9	14	8.1	5728	3950	20	3	0.2	0.3
10	0.0	0.1	(-)	(-)	68	3	(0.0)	(0.0)
11	0.9	14	3701	2882	53	3	0.2	0.3
12	0.1	0.1	3484	2298	72	3	0.1	0.3
13	4.2	0.8	3265	3511	30	3	0.1	0.3
14	0.8	7.1	1897	1458	82	3	0.1	0.2
15	7.3	7.4	3567	3876	42	3	0.1	0.3
16	0.9	4.0	14487	14007	41	3	0.5	0.8
17	11	19	1051	770	83	3	0.1	0.1
18	3.2	7.8	7107	7051	91	3	0.3	0.6
19	3.9	4.6	6658	6362	36	3	0.2	0.4
20	2.0	3.5	9477	8724	16	3	0.3	0.5
Average	3.5	7	71688	81026	59	3	2.5	4.4

(1) CPLEX default, (2) lifted inequalities.

depending on the strength of the inequality being added; this is a topic of future study.

Next, we test the instances in Dataset 2. In Table 7.3, we report the improvement at the root node (root improvement), the number of branch-and-bound nodes (b&b nodes), and the number of cuts added, for CPLEX default (1) and our lifted inequalities (2). We obtain the instances in Dataset 2 by a random sampling within a parameter space; we present computations for four instances (instance) for each parameter set (data set).

We see in Table 7.3 that these instances are quite easy to solve, even though they have many more variables (100) as compared with Dataset 1. In fact, CPLEX default obtains the optimal solution at the root node for five of the twelve instances, and does not need more than 16 nodes for any instance. This illustrates that hard instances not easily obtained by random sampling, and it comes as no surprise that the hard instances in Dataset 1 are highly contrived. Even for these easy instances in Dataset 2, the inequal-

Table 7.3: Computations on pure-integer knapsack: Dataset 2

data set	instance	root improvement (1)	root improvement (2)	b&b nodes (1)	b&b nodes (2)	cuts added (1)	cuts added (2)
A	1	96	97	5	2	1	811
A	2	0	100	16	0	1	3183
A	3	94	94	3	10	1	2984
A	4	100	100	0	0	1	1746
B	1	99	100	6	9	2	1305
B	2	0	100	11	0	1	3654
B	3	0	100	9	0	0	3524
B	4	100	100	0	0	1	1364
C	1	0	100	15	0	1	1375
C	2	100	100	0	0	1	966
C	3	100	100	0	0	2	954
C	4	87	97	1	2	1	709

(1) CPLEX default, (2) lifted inequalities.

ities significantly outperform CPLEX; we obtain the optimal solution at the root node for nine out of the 12 instances. Even in the other three instance, we close at least 94% of the gap at the root node; while CPLEX default does not add any cuts for four instances.

7.7 Conclusions

In this chapter, we studied the polyhedra of mixed-integer knapsacks with two integer variables and at most one continuous variable. We presented polynomial algorithms to enumerate all facets of these sets. We then analyzed the exact lifting function for these facets, and characterized in closed form super-additive lower bounds for them that can be computed trivially. These super-additive lower bounds were developed by studying two classes of super-additive functions that generalize all partial LP relaxations of the exact lifting function.

Using sequence independent lifting, we generated valid inequalities for the mixed-integer knapsack set with at most one continuous variable. We presented sufficient conditions under which these lifted inequalities define facets of mixed-integer knapsack sets with at most one continuous variable. Finally, we incorporated these inequalities in a

branch-and-cut framework, and demonstrated their effectiveness in solving linear optimization problems over these sets.

Chapter 8

Conclusion

8.1 Summary of dissertation

In this dissertation, we reviewed various methodologies for designing capacitated survivable networks. We showed that the capacitated survivable network design problem (SNP), in its myriad versions, results in very large and complex formulations and is computationally very difficult to solve. Furthermore, the current research on survivable networks is still in the early stages of infancy - most of the work deals with the construction of heuristics to solve the SNP. In Chapter 3, we presented a new approach that uses directed cycles as failure-flow patterns to reroute disrupted flow. This approach constitutes a hybrid between traditional approaches of dedicated and shared protection, and performed better than dedicated protection schemes in terms of capacity efficiency.

Other hybrid schemes are easy to implement; see Grover and Stamatelakis (1998), Schupke et al. (2002), Stamatelakis and Grover (2000). As in shared protection, no assumptions are made on the network topology, or on the no-failure flows. However, all disrupted flows are restricted (as in dedicated protection schemes such as self-healing rings) to be rerouted among predetermined failure-flow patterns. In particular, we use directed cycles as our failure-flow patterns.

We focused on the mixed-integer programming formulations for such frameworks, and studied various polyhedra related to the SNP in an attempt to describe strong valid inequalities. In particular, we focused on the arc-set and cut-set polyhedra. For both

relaxations, we first studied the network design problem without any survivability requirements (NDP). In Chapter 2, we developed new classes of inequalities and proposed more efficient separation algorithm for known families of inequalities. The inequalities developed for the NDP significantly reduced the solution times when used in a branch-and-cut framework, and provided significant insight toward the study of similar polyhedra for the survivable case.

When designing survivable networks using directed cycles (SDC), we developed k-partition inequalities for the cut-set polyhedra and extended the residual capacity inequalities for the arc-set polyhedra. In Chapter 4, one class of the k-partition inequalities was shown to be facet-defining even for the polyhedron of SDC, under mild conditions. Our mixed-integer formulation for SDC was shown to have only as many constraints as NDP, but included an exponential class of directed cycle variables. We studied the pricing problem for these variables, and proposed a polynomial-time algorithm that was used to handle these variables in a column generation approach.

Incorporated in a branch-and-cut algorithm, the strong valid inequalities reduced the computational effort in solving SDC, often by an order of magnitude. These computational results also scaled favorably to large problem instances, suggesting that using failure-flow patterns is the key to designing large scale survivable networks. The usage of failure-flow patterns allows us to design capacity-efficient networks, which can also be implemented easily.

To design more capacity-efficient networks with less computational effort, we considered two main approaches. In the former, we considered other failure-flow patterns. Based on preliminary computational efforts presented in Chapter 5, directed p-cycles seemed to result in survivable networks with capacity efficiency comparable to global rerouting. However, we see that designing survivable networks using directed p-cycles is still computationally challenging. For instance, the pricing problem for the directed p-cycle variables is $\mathcal{NP}$-Hard. Furthermore, the cut-set polyhedra now changes significantly. On the other hand, the arc-set polyhedra is unchanged, when compared with SDC.

The latter direction attempts to develop stronger inequalities for the SNP. We ap-

proached this in two complementary directions. The first direction attempts to develop problem-specific strong valid inequalities. In Chapter 6, we study the metric inequalities for the NDP, and described metric-type inequalities for survivable network design, both using directed cycles and directed p-cycles. These inequalities generalize the partition inequalities of the cut-set polyhedra, and are shown to reduce solution times for SDP by an order of magnitude. In the second direction, we developed problem-independent inequalities for any mixed-integer knapsack set. Since this set is a relaxation of any constraint of a mixed-integer program, these can be used for all problems, see Chapter 7. We developed these inequalities by sequence independent lifting of facets of the two integer and one continuous variable restriction of the mixed-integer knapsack set.

8.2 Future directions of research

There are many directions in which the research in this dissertation can be continued. In this section, we summarize some of the open questions from the preceding chapters; and then discuss long-term research directions.

In the development of strong valid inequalities for the general mixed-integer knapsack set, we considered partial LP relaxations to develop super-additive lifting functions, see Chapter 7. Other relaxations can result in stronger super-additive lower bounds for the exact lifting function, resulting in stronger valid inequalities. In the development of strong valid inequalities for the arc-set polyhedra, it will be interesting to know to measure the maximum possible improvement from these inequalities, especially for unsplittable flow problems, see Chapter 2.

Most of the future work revolves around the design of survivable networks using directed p-cycles (SDP), since this framework results in highly capacity-efficient networks. We need to develop better (faster and more effective) heuristics to price directed p-cycles. The exact pricing problem for the directed p-cycle variables is interesting in itself, and is a possible direction for future study. Successfully pricing more directed p-cycle variables with negative reduced costs will bring down the capacity requirements of SDP even closer

to global rerouting.

Furthermore, we need to develop stronger inequalities for the polyhedra of various relaxations of SDP. This will reduce the computation time even more. On the theoretical front, we wish to characterize when these inequalities describe the convex hull of these relaxations, and of SDP itself.

A challenging computational issue is how to integrate column and cut generation schemes in a branch-and-price-and-cut algorithm. Synthesis of pricing and separation requires that the separated inequalities do not change the pricing problem; and the separated inequalities can be modified to include variables that are priced later. Hence, we see that synthesizing the separation and pricing problems is non-trivial and has been successfully implemented only for simpler problems such as the multi-commodity flow problem, see Barnhart et al. (2000).

More research is necessary to develop other methodologies to design capacitated survivable network design problems, especially when extended to particular technologies, such as wavelength division multiplexing networks using either virtual wavelength paths or wavelength paths. There may exist other frameworks, or other failure-flow patterns that result in more capacity-efficient networks than SDP. Ideally, we will use a formulation that allows us to choose from different failure-flow patterns. Each of these requires us to study the corresponding pricing problem and polyhedra.

Another direction for future research addresses multiple edge failures. Some methodologies can be extended to incorporate multiple failures more easily than others; further research is required.

Bibliography

Aardal, K. and Lenstra, A. K. (2002). Hard equality constrained integer knapsacks. In Cook, W. J. and Schulz, A. S., editors, *Lecture Notes in Computer Science*, volume 2337, pages 350–366. Springer-Verlag.

Agra, A. and Constantino, M. F. (2003). Lifting 2-integer knapsack inequalities. Technical report, CIO.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Englewood Cliffs NJ.

Alevras, D., Grötschel, M., and Wessäly, R. (1998). Cost-efficient network synthesis from leased lines. *Annals of Operations Research*, 76:1–20.

Altinkemer, K. (1994). Topological design of ring networks. *Computers and Operations Research*, 21:421–431.

Armony, M., Klincewicz, J. G., Luss, H., and Rosenwein, M. B. (2000). Design of stacked self-healing rings using a genetic algorithm. *Journal of Heuristics*, 6:85–105.

Atamtürk, A. (2002). On capacitated network design cut-set polyhedra. *Mathematical Programming*, 92:425–437.

Atamtürk, A. (2003a). Cover and pack inequalities for (mixed) integer programming. Technical Report BCOL.03.02, University of California at Berkeley. To appear in *Annals of Operations Research*.

Atamtürk, A. (2003b). On the facets of the mixed-integer knapsack polyhedron. *Mathematical Programming*, 98:145–175.

Atamtürk, A. (2004). Sequence independent lifting for mixed-integer programming. *Operations Research*, 52:487–490.

Atamtürk, A. and Rajan, D. (2002). On splittable and unsplittable flow capacitated network design arc-set polyhedra. *Mathematical Programming*, 92:315–333.

Avella, P., Mattia, S., and Sassano, A. (2004). Metric inequalities and the network loading problem. In Bienstock, D. and Nemhauser, G., editors, *Lecture Notes in Computer Science*, volume 3064, pages 16–32. Springer-Verlag.

Balakrishnan, A., Magnanti, T. L., and Mirchandani, P. (1997). Network design. In Dell'Amico, M., Maffioli, F., and Martello, S., editors, *Annotated Bibliographies in Combinatorial Optimization*. John Wiley & Sons.

Balakrishnan, A., Magnanti, T. L., and Mirchandani, P. (1998). Designing hierarchical survivable networks. *Operations Research*, 46:116–136.

Balakrishnan, A., Magnanti, T. L., Sokol, J. S., and Wang, Y. (2001). Telecommunication link restoration planning with multiple facility types. *Annals of Operations Research*, 106:127–154.

Balakrishnan, A., Magnanti, T. L., Sokol, J. S., and Wang, Y. (2002). Spare-capacity assignment for line-restoration using a single-facility type. *Operations Research*, 50:617–635.

Balas, E. (1975). Facets of the knapsack polytope. *Mathematical Programming*, 8:146–164.

Balas, E. (1979). Disjunctive programming. *Annals of Discrete Mathematics*, 5:3–51.

Balas, E., Ceria, S., and Cornuéjols, G. (1993). A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58:295–324.

Barahona, F. (1996). Network design using cut inequalities. *SIAM Journal on Optimization*, 6:823–837.

Barnhart, C., Hane, C. A., and Vance, P. H. (2000). Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, 48:318–326.

Bienstock, D. (2001). Personal communication.

Bienstock, D., Chopra, S., Günlük, O., and Tsai, C.-Y. (1998). Minimum cost capacity installation for multicommodity networks. *Mathematical Programming*, 81:177–199.

Bienstock, D. and Günlük, O. (1996). Capacitated network design - Polyhedral structure and computation. *INFORMS Journal on Computing*, 8:243–259.

Bienstock, D. and Muratore, G. (2000). Strong inequalities for capacitated survivable network design problems. *Mathematical Programming*, 89:127–147.

Brockmüller, B., Günlük, O., and Wolsey, L. A. (1996). Designing private line networks - Polyhedral analysis and computation. CORE Discussion Paper 96-47, Université Catholique de Louvain. To appear in *Transactions on Operations Research*.

Ceria, S., Cordier, C., Marchand, H., and Wolsey, L. A. (1998). Cutting planes for integer programs with general integer variables. *Mathematical Programming*, 81:201–214.

Chopra, S., Gilboa, I., and Sastry, S. T. (1998). Source sink flows with capacity installation in batches. *Discrete Applied Mathematics*, 85:165–192.

Chung, S. H., King, H. G., Yoon, Y. S., and Tcha, D. W. (1996). Cost-minimizing construction of a unidirectional SHR with diverse protection. *IEEE Transactions on Networking*, 4:921–928.

Chvátal, V. (1983). *Linear Programming*. W. H. Freeman and Company, New York.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. The MIT Press, Cambridge.

Cosares, S., Deutsch, D. N., Saniee, I., and Wasem, O. J. (1995). SONET toolkit: a decision support system for designing robust and cost-effective fiber-optic network. *Interfaces*, 25:20–40.

Crowder, H., Johnson, E. L., and Padberg, M. W. (1983). Solving large-scale zero-one linear programming problems. *Operations Research*, 31:803–834.

Dahl, G. and Stoer, M. (1998). A cutting plane algorithm for multicommodity survivable network design problems. *INFORMS Journal on Computing*, 10:1–11.

Dantzig, G. B. and Eaves, B. C. (1973). Fourier-motzkin elimination and its dual. *Journal of Combinatorial Theory (A)*, 14:288–297.

Ellinas, G., Hailemariam, A. G., and Stern, T. E. (2000). Protection cycles in mesh WDM networks. *IEEE Journal on Selected Areas in Communications*, 18:1924–1937.

Frank, A. (1994). Connectivity augmentation problems in network design. In Birge, J. R. and Murty, K. G., editors, *Mathematical Programming: State of the Art 1994*. The University of Michigan.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York.

Gavish, B. and Altinkemer, K. (1990). Backbone network design tools with economic tradeoffs. *ORSA Journal on Computing*, 2:58–76.

Goldschmidt, O., Laugier, A., and Olinick, E. V. (2003). SONET/SDH ring assignment with capacity constraints. *Discrete Applied Mathematics*, 129:99–128.

Gomory, R. E. (1960). An algorithm for the mixed integer problem. Technical Report RM-2597, The Rand Corporation.

Gomory, R. E. (1969). Some polyhedra related to combinatorial problems. *Linear Algebra and Its Applications*, 2:451–558.

Grötschel, M., Lovász, L., and Schrijver, A. (1981). The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197.

Grötschel, M., Monma, C. L., and Stoer, M. (1995). Design of survivable networks. In Ball, M., Magnanti, T. L., Monma, C. L., and Nemhauser, G. L., editors, *Network Models*, pages 617–672. North-Holland.

Grover, W. and Stamatelakis, D. (1998). Cycle-oriented distributed pre-configuration: ring-like speed with mesh-like capacity for self-planning network restoration. In *Proceedings of IEEE International Conference on Communications*, pages 537–543.

Grover, W. D. and Martens, R. G. (2000). Optimized design of ring-mesh hybrid networks. In *Proceedings of IEEE/VDE Design of Reliable Communication Networks 2000*, pages 291–297.

Gu, Z., Nemhauser, G. L., and Savelsbergh, M. W. P. (1994). Lifted knapsack cover inequalities for 0-1 integer programs: Fast algorithms. Manuscript, Georgia Institute of Technology, Atlanta.

Gu, Z., Nemhauser, G. L., and Savelsbergh, M. W. P. (1998). Cover inequalities for 0-1 integer programs: Computation. *INFORMS Journal on Computing*, 10:427–437.

Gu, Z., Nemhauser, G. L., and Savelsbergh, M. W. P. (1999). Lifted flow cover inequalities for mixed 0-1 integer programs. *Mathematical Programming*, 85:439–467.

Günlük, O. (1999). A branch-and-cut algorithm for capacitated network design problems. *Mathematical Programming*, 86:17–39.

Hammer, P. L., Johnson, E. L., and Peled, U. N. (1975). Facets of regular 0-1 polytopes. *Mathematical Programming*, 8:179–206.

Herzberg, M., Bye, S. J., and Utano, A. (1995). The hop-limit approach for spare capacity assignment in survivable networks. *IEEE/ACM Transactions on Networking*, 3:775–784.

Hochbaum, D. S. and Olinick, E. (2001). The bounded cycle cover problem. *INFORMS Journal on Computing*, 13:104–119.

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

Iraschko, R., MacGregor, M., and Grover, W. (1998). Optimal capacity placement for path restoration in STM or ATM mesh survivable networks. *IEEE/ACM Transactions on Networking*, 6:325–336.

Iri, M. (1971). On an extension of the max-flow min-cut theorem to multicommodity flows. *Journal of the Operations Research Society of Japan*, 13:129–135.

Jeroslow, R. (1980). A cutting plane game for facial disjunctive programs. *SIAM Journal on Control and Optimization*, 18:264–280.

Jones, K. L., Lustig, I. J., Farvolden, J. M., and Powel, W. B. (1993). Multicommodity network flows: the impact of formulation on decomposition. *Mathematical Programming*, 62:95–117.

Kannan, R. (1980). A polynomial algorithm for the two-variable integer programming problem. *Journal of the Association for Computing Machinery*, 27:118–122.

Kannan, R. (1993). Optimal solution and value of parametric integer programs. In *Proceedings of the 3rd International Integer Programming and Combinatorial Optimization Conference*, pages 11–21.

Khachian, L. G. (1979). A polynomial algorithm for linear programming. *Soviet Mathematics Doklady*, 20:191–194.

Lenstra Jr., H. W. (1983). Integer programming with fixed number of variables. *Mathematics of Operations Research*, 8:538–547.

Lisser, A., Sarkissian, R., and Vial, J. P. (1995). Survivability in telecommunication networks. Technical Report 1995.3, University of Geneva.

Luss, H., Rosenwein, M. B., and Wong, R. T. (1998). Topological network design for SONET ring architecture. *IEEE Transactions on Systems, Man and Cybernetics*, 28:780–790.

Magnanti, T. L. and Mirchandani, P. (1993). Shortest paths, single origin-destination network design, and associated polyhedra. *Networks*, 23:103–121.

Magnanti, T. L., Mirchandani, P., and Vachani, R. (1993). The convex hull of two core capacitated network design problems. *Mathematical Programming*, 60:233–250.

Magnanti, T. L., Mirchandani, P., and Vachani, R. (1995). Modeling and solving the two-facility capacitated network loading problem. *Operations Research*, 43:142–157.

Marchand, H., Martin, A., Weismantel, R., and Wolsey, L. A. (2002). Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 123:397–446.

Marchand, H. and Wolsey, L. A. (1999). The 0-1 knapsack problem with a single continuous variable. *Mathematical Programming*, 85:15–33.

Marchand, H. and Wolsey, L. A. (2001). Aggregation and mixed integer rounding to solve MIPs. *Operations Research*, 49:363–371.

Médard, M., Barry, R. A., Finn, S. G., He, W., and Lumetta, S. S. (2002). Generalized loop-back recovery in optical mesh networks. *IEEE Transactions on Networking*, 10:153–164.

Meyer, R. R. (1974). On the existence of optimal solutions to integer and mixed-integer programming problems. *Mathematical Programming*, 7:223–235.

Murakami, K. and Kim, H. S. (1995). Joint optimization of capacity and flow assignment for self-healing ATM networks. In *Proceedings of the IEEE International Conference on Communications*, pages 216–220.

Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and Combinatorial Optimization*. John Wiley and Sons, New York.

Nemhauser, G. L. and Wolsey, L. A. (1990). A recursive procedure for generating all cuts for 0-1 mixed integer programs. *Mathematical Programming*, 46:379–390.

Onaga, K. and Kakusho, O. (1971). On feasibility conditions of multi-commodity flows in networks. *Transactions on Circuit Theory*, 18:425–429.

Owen, J. and Mehrotra, S. (2001). A disjunctive cutting plane procedure for general mixed-integer linear programs. *Mathematical Programming*, 89:437–448.

Padberg, M. W. (1979). Covering, packing and knapsack problems. *Annals of Discrete Mathematics*, 4:265–287.

Papadimitriou, C. and Steiglitz, K. (1998). *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Inc, New York.

Pochet, Y. and Weismantel, R. (1998). The sequential knapsack polytope. *SIAM Journal on Optimization*, 8:248–264.

Pochet, Y. and Wolsey, L. A. (1995). Integer knapsack and flow covers with divisible coefficients: Polyhedra, optimization, and separation. *Discrete Applied Mathematics*, 59:57–74.

Raghavan, S. and Magnanti, T. L. (1997). Network connectivity. In Dell'Amico, M., Maffioli, F., and Martello, S., editors, *Annotated Bibliographies in Combinatorial Optimization*, pages 335–354. John Wiley & Sons.

Rajan, D. and Atamtürk, A. (2002). Survivable network design : Routing of flows and slacks. In Anandalingam, G. and Raghavan, S., editors, *Telecommunications Network Design and Management*, pages 65–81. Kluwer Academic Publishers.

Rajan, D. and Atamtürk, A. (2004). A directed cycle based column-and-cut generation method for capacitated survivable network design. *Networks*, 43:201–211.

Richard, J., de Farias, I. R., and Nemhauser, G. L. (2002). Lifted inequalities for 0-1 mixed integer programming : Basic theory and algorithms. In *Integer Programming and Combinatorial Optimization, 9th International IPCO Conference, Proceedings*, pages 161–175.

Sakauchi, H., Nishimura, Y., and Hasegawa, S. (1990). A self-healing network with an economical spare-channel assignment. In *Proceedings of IEEE GLOBECOM 1990*, pages 438–443.

Scarf, H. E. (1981). Production sets with indivisibilities Part II. The case of two activities. *Econometrica*, 49:395–423.

Schrijver, A. (1987). *Theory of Linear and Integer Programming*. John Wiley and Sons, Chichester.

Schupke, D. A., Gruber, G. C., and Autenrieth, A. (2002). Optimal configuration of p-cycles in WDM networks. In *IEEE International Conference on Communications*.

Sherali, H. D. and Adams, W. P. (1994). A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems. *Discrete Applied Mathematics*, 3:83–106.

Slevinsky, J. B., Grover, W. D., and MacGregor, M. H. (1993). An algorithm for survivable network design employing multiple self-healing rings. In *Proceedings of IEEE GLOBECOM 1993*, pages 1568–1572.

Soriano, P., Wynants, C., Séguin, R., Labbé, M., Gendreau, M., and Fortz, B. (1998). Design and dimensioning of survivable SDH/SONET networks. In Sansò, B. and Soriano, P., editors, *Telecommunications Network Planning*, pages 147–168. Kluwer Academic Publishers, Netherlands.

Stamatelakis, D. and Grover, W. D. (2000). Theoretical underpinnings for the efficiency of restorable networks using pre-configured cycles ("p-cycles"). *IEEE Transactions on Communications*, 48:1262–1265.

Sutter, A., Vanderbeck, F., and Wolsey, L. A. (1998). Optimal placement of add/drop multiplexers: heuristic and exact algorithms. *Operations Research*, 46:719–728.

Venables, B., Grover, W. D., and MacGregor, M. H. (1993). Two strategies for spare capacity placement in mesh restorable networks. In *Proceedings of IEEE International Conference on Communications 1993*, pages 267–271.

Weismantel, R. (1995). Knapsack problems, test sets and polyhedra. Postdoctoral Thesis, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Technische Universität Berlin.

Weismantel, R. (1997). On the 0/1 knapsack polytope. *Mathematical Programming*, 77:49–68.

Wolsey, L. A. (1975). Faces for linear inequality in 0-1 variables. *Mathematical Programming*, 8:165–178.

Wolsey, L. A. (1976). Facets and strong valid inequalities for integer programs. *Operations Research*, 24:367–372.

Wolsey, L. A. (1998). *Integer Programming*. John Wiley and Sons, New York.

Xiong, Y. and Mason, L. G. (1998). On state-independent and state-dependent path restoration in self-healing networks. In *Proceedings of IEEE International Conference on Communications 1998*, pages 1114–1118.

Xiong, Y. and Mason, L. G. (1999). Restoration strategies and spare capacity requirements in self-healing ATM networks. *IEEE/ACM Transactions on Networking*, 7:98–110.

Zemel, E. (1989). Easily computable facets of the knapsack polytope. *Mathematics of Operations Research*, 14:760–764.

Appendix A

Miscellaneous tables

Table A.1: Miscellaneous notation

symbol	explanation
$\mathbb{Z}$	set of integers
$\mathbb{R}$	set of reals
$\mathbb{Q}$	set of rational numbers
N_+	non-negative elements of set N
N_{++}	positive elements of set N
$\mathbb{B}$	set $\{0,1\}$
$\log$	logarithm to base 2.
$\lvert N\rvert$	number of elements in set N
$v(H)$	$\sum_{i\in H} v_i$ for vector function v defined on set N and $H \subseteq N$
$\lfloor\alpha\rfloor$	$\max_x\{x \in \mathbb{Z} : x \leq \alpha\}$, for $\alpha \in \mathbb{R}$
$\lceil\alpha\rceil$	$\min_x\{x \in \mathbb{Z} : x \geq \alpha\}$, for $\alpha \in \mathbb{R}$
$r(\alpha,\beta)$	$\alpha - \lfloor\alpha/\beta\rfloor\beta$, for $\alpha,\beta \in \mathbb{R}$
$r(\alpha)$	$r(\alpha,1)$
$[i,k]$	set of integers $\{j \in \mathbb{Z} : i \leq j \leq k\}$
IntUni$[\alpha,\beta]$	integer-uniform distribution with minimum α and maximum β
$(\cdot)^+$	$\max\{(\cdot),0\}$
$I\{(\cdot)\}$	indicator function $I\{(\cdot)\}$ to take the value 1 if $(\cdot)$ is true, and 0 otherwise.
ϵ	a small strictly positive constant

Table A.2: Acronyms used

acronym	explanation
FCP	Fixed-charge network flow problem
MFP	Multi-commodity flow problem
NDP	Network design problem with no survivability requirements
DPP	1+1 diverse protection network design problem
GNP	Global rerouting shared protection problem
LNP	Link rerouting shared protection problem
NDC	Network design problem with connectivity requirements
PNP	Path rerouting shared protection problem
RNP	Self-healing ring network design problem
SCP	Spare capacity assignment problem
SNP	Capacitated survivable network design problem
HDC	Spare capacity assignment problem using directed cycles
HUP	Spare capacity assignment problem using undirected p-cycles
SDC	Survivable network design problem using directed cycles
SDP	Survivable network design problem using directed p-cycles
BLP	Lifting problem for projected variables in lifted knapsack cover inequality
CPP	Pricing problem of directed cycles
HPP	Hamiltonian path problem
MCP	Maximum chord problem
PPP	Pricing problem of directed p-cycles
SFP	Optimization problem over splittable flow arc set
SP	Separation problem over splittable flow arc set
TSP	Traveling salesperson problem
UFP	Optimization problem over unsplittable flow arc set
UFP_f	Optimization problem over fractional unsplittable flow arc set
IP	Pure-integer program
LP	Linear programming
MIP	Mixed-integer program
DP	Diverse protection
SHR	Self-healing rings

Table A.3: Notation for network design problems

symbol	explanation
$G = (V, E)$	undirected graph with node set V and edge set E
$G' = (V, F)$	directed graph with node set V and arc set F
$F \setminus [ij]$	$F \setminus \{(ij), (ji)\}$
S	set of failure states
0	no-failure state
K	set of commodities
s^k	source node of commodity k
t^k	destination node of commodity k
d^k	demand quantity for commodity $k \in K$
b_i^k	supply of commodity k at node i
$b_{s^k}^k$	d^k
$b_{t^k}^k$	$-d^k$
b_i^k	0 for $i \in V \setminus \{s^k, t^k\}$
e_{ij}^k	cost associated with routing each unit of commodity $k \in K$ on arc (ij)
P_k^s	set of paths from s^k to t^k in failure state s
δ_{ij}^p	1 if path p it includes arc (ij), and 0 otherwise
ζ_s^p	1 if path p is affected by failure state $s \in S \setminus \{0\}$
T	set of installable capacity types
q^t	maximum demand that can be routed using one unit of capacity type t
$h_{[ij]}^t$	cost of installing unit capacity of type $t \in T$ on edge $[ij] \in E$
C	set of undirected cycles of G
$\mathcal{C}$	set of directed cycles of G'
α_{ij}^c	1 if directed cycle c includes arc (ij), and 0
$\alpha_{[ij]}^c$	1 if undirected cycle c includes edge $[ij]$, and 0 otherwise
$\rho_{[ij]}^c$	1 if edge $[ij]$ is a chord to cycle c, and 0 otherwise
g_{ij}^0	pre-existing amount of demand routed through arc $(ij) \in F$
$w_{[ij]}^0$	pre-existing capacity on edge $[ij] \in E$

Table A.4: List of variables

symbol	explanation
y_{ij}^{ks}	fraction of commodity k routed through arc $(ij) \in F$ in failure state s
y_p	fraction of commodity routed on path p
$x_{[ij]}^t$	amount of capacity of type $t \in T$ installed on edge $[ij]$
z_c	amount of slack (fractional capacity) reserved on cycle c

Appendix B

Proofs of results in Chapter 2

Proposition 2.4 For any extreme point $(\bar{y}, \bar{x})$ of $\mathcal{F}_L$, let $\bar{H} = \{k \in K : 0 < \bar{y}^k < 1\}$. Then, $|\bar{H}| \leq 1$.

Proof We prove that $|\bar{H}| \leq 1$ by contradiction. Assume that there exist $i, j \in K, i \neq j$ such that $0 < \bar{y}^i < 1$ and $0 < \bar{y}^j < 1$. Consider the points $(\hat{y}, \hat{x})$and $(\tilde{y}, \tilde{x})$ obtained as follows.

$$\hat{y}^k = \begin{cases} \bar{y}^k - d^j\epsilon & if \;\; k = i, \\ \bar{y}^k + d^i\epsilon & if \;\; k = j, \\ \bar{y}^k & \text{otherwise}, \end{cases} \quad \text{for } k \in K, \qquad \tilde{y}^k = \begin{cases} \bar{y}^k + d^j\epsilon & if \;\; k = i, \\ \bar{y}^k - d^i\epsilon & if \;\; k = j, \\ \bar{y}^k & \text{otherwise}, \end{cases} \quad \text{for } k \in K,$$

and $\hat{x} = \tilde{x} = \bar{x}$. Since $\sum_{k\in K} d^k\hat{y}^k = \sum_{k\in K} d^k\tilde{y}^k = \sum_{k\in K} \bar{y}^k$, $(\hat{y}, \hat{x})$ and $(\tilde{y}, \tilde{x})$ are elements of $\mathcal{F}_L$. Furthermore, $(\bar{y}, \bar{x})$ can be written as a convex combination of $(\hat{y}, \hat{x})$ and $(\tilde{y}, \tilde{x})$. Thus, $(\bar{y}, \bar{x})$ can not be an extreme point, and we have a contradiction. □

Lemma 2.7 A point $(\bar{y}, \bar{x}) \in \mathcal{F}_L$ does not violate any residual capacity inequality (2.6) with $\eta_H \leq \bar{x}$ or $\eta_H \geq \bar{x} + 1$.

Proof Since residual capacity inequality (2.6) is the mixed-integer rounding inequality (see Section 1.5.2) for the relaxation $\sum_{k\in H} d^k(1-y^k)+x \geq d(H)-w^0$, it is dominated by $\sum_{k\in H} d^k(1-y^k) \geq 0$ and $\sum_{k\in H} d^k(1-y^k)+x \geq d(H)-w^0$ unless $\eta_H - 1 < x < \eta_H$. □

Lemma 2.8 If there exists a residual capacity inequality (2.6) violated by a fractional point $(\bar{y}, \bar{x}) \in \mathcal{F}_L$, then there exists one given by $H \subseteq T$.

Proof Suppose the residual capacity inequality given by $C \subseteq K$ is violated by $(\bar{y}, \bar{x})$. Then, $d(C \cap T) + d(C \setminus T) = w^0 + \lfloor \bar{x} \rfloor + r_H$. Consider the following two cases.
Case 1. $d(C \setminus T) < r_H$. In this case $w^0 + \lfloor \bar{x} \rfloor < d(C \cap T)$. Also, $d(C \cap T) < w^0 + \lceil \bar{x} \rceil$ and $C \cap T$ has an objective value in SP that is no more than that of C. So let $H = C \cap T$.
Case 2. $d(C \setminus T) \geq r_H$. In this case $d(C \cap T) \leq w^0 + \lfloor \bar{x} \rfloor$. Therefore, the objective value for C in SP is $\sum_{k\in C} d^k(1 - \bar{y}^k - \lceil \bar{x} \rceil + \bar{x}) + (\lceil \bar{x} \rceil - \bar{x})(w^0 + \lfloor \bar{x} \rfloor) \geq \sum_{k\in C} d^k(1 - \bar{y}^k - \lceil \bar{x} \rceil + \bar{x}) + (\lceil \bar{x} \rceil - \bar{x})d(C \cap T) = \sum_{k\in C\cap T} d^k(1-\bar{y}^k) + \sum_{k\in C\setminus T} d^k(1-\bar{y}^k - \lceil \bar{x} \rceil + \bar{x}) \geq 0$. However, this contradicts the assumption that the residual capacity given by C is violated. □

Lemma 2.9 If $d(T) \leq w^0 + \lfloor \bar{x} \rfloor$ or $d(T) \geq w^0 + \lceil \bar{x} \rceil$, then there exists no residual capacity inequality violated by $(\bar{y}, \bar{x}) \in \mathcal{F}_L$.

Proof Suppose $d(T) \leq w^0 + \lfloor \bar{x} \rfloor$. Then, there exists no $H \subseteq T$ that satisfies the constraints of SP and therefore, from Lemma 2.8, there exists no residual capacity inequality that is violated by $(\bar{y}, \bar{x})$. Now suppose $d(T) \geq w^0 + \lceil \bar{x} \rceil$ and, for contradiction, suppose there exists a set H that gives a violated inequality. From Lemma 2.8, we may assume that $H \subseteq T$. The objective value for H in SP is $\sum_{k \in H} d^k(1 - \bar{y}^k - \lceil \bar{x} \rceil + \bar{x}) + (\lceil \bar{x} \rceil - \bar{x})(w^0 + \lfloor \bar{x} \rfloor) \geq \sum_{k \in T} d^k(1 - \bar{y}^k - \lceil \bar{x} \rceil + \bar{x}) + (\lceil \bar{x} \rceil - \bar{x})(w^0 + \lfloor \bar{x} \rfloor) \geq (\lceil \bar{x} \rceil + w^0)(1 - \lceil \bar{x} \rceil + \bar{x}) - (w^0 + \bar{x}) + (\lceil \bar{x} \rceil - \bar{x})(w^0 + \lfloor \bar{x} \rfloor) = (\lceil \bar{x} \rceil + w^0) - (\lceil \bar{x} \rceil - \bar{x}) - (w^0 + \bar{x}) = 0$. This contradicts the assumption that the inequality is violated by $(\bar{y}, \bar{x})$. □

Proposition 2.11 In any optimal solution (y^*, x^*) to UFP, $x^* = \lceil \sum_{k \in K} d^k y^{k*} - w^0 \rceil$.

Proof We prove this by contradiction. Assume that (y^*, x^*) is an optimal solution to UFP such that $x^* > \lceil \sum_{k \in K} d^k y^{k*} - w^0 \rceil$. Consider the point $(\hat{y}, \hat{x})$ defined as $\hat{y} = y^*$ and $\hat{x} = x^* - 1$. Since $x^* - 1 \geq \lceil \sum_{k \in K} d^k y^{k*} - w^0 \rceil$, $(\hat{y}, \hat{x}) \in \mathcal{D}_U$. Furthermore, $\sum c^k \hat{y}^k - \hat{x} = \sum_{k \in K} c^k y^{k*} - \hat{x} > \sum_{k \in K} c^k y^{k*} - x^*$. Thus, (y^*, x^*) is not optimal, and we have a contradiction.

Proposition 2.12 UFP has an optimal solution $(y^\star, x^\star)$ such that

$$y^{k\star} = \begin{cases} 1 & if \;\; c^k \geq \lceil d^k \rceil, \\ 0 & if \;\; c^k \leq \lfloor d^k \rfloor, \end{cases} \quad \text{for } k \in K.$$

Proof For $H \subseteq K$ let $\xi(H)$ be the maximum value of the objective of UFP when $y^k = 1$ for all $k \in H$, and $y^k = 0$ otherwise, i.e., $\xi(H) = c(H) - \lceil d(H) - w^0 \rceil$. Suppose $k \notin H$. Then $\xi(H) - \xi(H \cup k) = \lceil d(H) + d^k - w^0 \rceil - \lceil d(H) - w^0 \rceil - c^k$. Since for any $a, b \in \mathbb{R}$, we have $\lceil a \rceil + \lfloor b \rfloor \leq \lceil a + b \rceil \leq \lceil a \rceil + \lceil b \rceil$, it follows that

$$\xi(H) - \xi(H \cup k) \begin{cases} \leq 0 & if \;\; c^k \geq \lceil d^k \rceil, \\ \geq 0 & if \;\; c^k \leq \lfloor d^k \rfloor. \end{cases}$$ □

Theorem 2.14 UFP is $\mathcal{NP}$-hard for any fixed value of w^0.

Proof The proof is by reduction from PARTITION (Garey and Johnson 1979): Given a set K and weights d^k $k \in K$ with $d(K) = 2$, does there exist $H \subset K$ such that $d(H) = 1$?

Let w^0 be fixed to $\bar{w}^0$ and let $\bar{f}^0 = \bar{w}^0 - \lfloor \bar{w}^0 \rfloor$. Let $\alpha > 1$ and $1 > \epsilon > 0$. To answer PARTITION, we construct the following instance of UFP with $w^0 = \bar{w}^0$.

$$\begin{aligned} \varsigma = \max \;\; & \sum_{k \in K} d^k y^k + 3\alpha y^0 - \alpha x \\ s.t. : \;\; & \sum_{k \in K} d^k y^k + (\epsilon + \bar{f}^0(1+\epsilon)) y^0 \leq (1+\epsilon)(\bar{w}^0 + x) \\ & y^k \in \{0,1\} \qquad\qquad k \in K \\ & y^0 \in \{0,1\} \\ & x \in \mathbb{Z} \end{aligned}$$

After dividing the objective by α and the constraint by $1 + \epsilon$, since $\lceil \frac{\epsilon}{1+\epsilon} + \bar{f}^0 \rceil < 3$, from Proposition 2.12 we see that $y^0 = 1$ and hence $x \geq \lceil -\bar{w}^0 \rceil + 1$ in every optimal solution

to UFP. Also since $d(K) + \epsilon + \bar{f}^0(1+\epsilon) < (1+\epsilon)(\bar{f}^0 + 2)$, from Proposition 2.11 we have $x \leq \lceil -\bar{w}^0 \rceil + 2$ in any optimal solution. The objective value $\varsigma(x)$ as a function of x satisfies the following: $\varsigma(\lceil -\bar{w}^0 \rceil + 1) \leq 1 + 2\alpha + \alpha\lfloor \bar{w}^0 \rfloor$ and $\varsigma(\lceil -\bar{w}^0 \rceil + 2) \leq 2 + \alpha + \alpha\lfloor \bar{w}^0 \rfloor$. Let $H^\star$ be the index set of binary variables at value 1 in an optimal solution. Since $\alpha > 1$, $\varsigma = 1 + 2\alpha + \alpha\lfloor \bar{w}^0 \rfloor$ if and only $d(H^\star) = 1$. Hence, the PARTITION problem has an affirmative answer if and only if the optimal value of UFP equals $1 + 2\alpha + \alpha\lfloor \bar{w}^0 \rfloor$. □

Proposition 2.16 UFP$_f$ is related to UFP in the sense that $H \subseteq K$ maximizes UFP if and only if H maximizes UFP$_f$. Furthermore, $\xi_f = \xi - \lfloor w^0 \rfloor$.

Proof For $H \subseteq K$ let $\xi(H) = c(H) - \lceil d(H) - w^0 \rceil$. Then,

$$\begin{aligned}\xi(H) &= \sum_{k\in H} \lfloor c^k \rfloor + g(H) - \sum_{k\in H} \lfloor d^k \rfloor + \lfloor w^0 \rfloor - \lceil f(H) - f^0 \rceil \\ &= g(H) + \lfloor w^0 \rfloor - \lceil f(H) - f^0 \rceil = \xi_f(H) + \lfloor w^0 \rfloor.\end{aligned}$$ □

Proposition 2.19 An inequality $\sum_{k\in K} \pi^k y^k \leq \pi^0 + x$ with $\lfloor d^k \rfloor \leq \pi^k \leq \lceil d^k \rceil$ is valid for $\mathcal{F}_U$ if and only if $\sum_{k\in K} (\pi^k - \lfloor \pi^k \rfloor) y^k \leq \pi^0 - \lfloor w^0 \rfloor + x$ is valid for $\mathcal{F}_{Uf}$.

Proof From Propositions 2.16 and 2.18,

$$\begin{aligned}&\max\{\sum_{k\in K} \pi^k y^k - x \,:\, \sum_{k\in K} d^k y^k \leq w^0 + x,\ (y,x) \in \mathcal{D}_U\} \\ &\quad = \max\{\sum_{k\in K} (\pi^k - \lfloor \pi^k \rfloor) y^k - x \,:\, \sum_{k\in K} f^k y^k \leq f^0 + x,\ (y,x) \in \mathcal{D}_U\} + \lfloor w^0 \rfloor.\end{aligned}$$

Then, for any π^0,

$$\begin{aligned}&\max\{\sum_{k\in K} \pi^k y^k - x \,:\, \sum_{k\in K} d^k y^k \leq w^0 + x,\ (y,x) \in \mathcal{D}_U\} \leq \pi^0 \\ &\iff \max\{\sum_{k\in K} (\pi^k - \lfloor \pi^k \rfloor) y^k - x \,:\, \sum_{k\in K} f^k y^k \leq f^0 + x,\ (y,x) \in \mathcal{D}_U\} \leq \pi^0 - \lfloor w^0 \rfloor.\end{aligned}$$

Thus, $\sum_{k\in K} \pi^k y^k \leq \pi^0 + x$ is valid for $\mathcal{F}_U$ if and only if $\sum_{k\in K} (\pi^k - \lfloor \pi^k \rfloor) y^k \leq \pi^0 - \lfloor w^0 \rfloor + x$ is valid for $\mathcal{F}_{Uf}$.

Theorem 2.21 The maximal c-strong inequalities (2.7) constitute all facet-defining inequalities $\sum_{k\in K} \pi^k y^k \leq \pi^0 + x$ of $\mathrm{conv}(\mathcal{F}_U)$ with integral $\pi^k,\ k \in [0, |K|]$.

Proof Let $\sum_{k\in K} \pi^k y^k \leq \pi^0 + x$ be a facet-defining inequality of $\mathrm{conv}(\mathcal{F}_U)$ with integral coefficients. From Proposition 2.18, it follows that $\pi^k \in \{0,1\}$ for $k \in K$. Let $H = \{k \in K \,:\, \pi^k = 1\}$. From Proposition 2.12, we have that $\pi^0 = c_H$. □

Proposition 2.22 Let $H = \{k \in K \,:\, \pi^k \geq \lceil jd^k \rceil\}$. Any inequality $\sum_{k\in K} \pi^k y^k \leq \pi^0 + jx$ with $j \in \mathbb{Z}_{++}$ and $\pi^k \leq \lfloor jd^k \rfloor$ or $\pi^k \geq \lceil jd^k \rceil$ for all $k \in K$ is valid for $\mathcal{F}_U$ for $\pi^0 = \pi(H) - \lceil jd(H) - jw^0 \rceil$.

Proof The proof is an immediate consequence of Proposition 2.12.

$$\begin{aligned}\pi^0 &= \pi(H) - \lceil jd(H) - jw^0 \rceil \\ &= \max\{\sum_{k\in K} \pi^k y^k - z \ : \ \sum_{k\in K} jd^k y^k \le jw^0 + z, \ (y,z) \in \mathcal{D}_U\} \\ &\ge \max\{\sum_{k\in K} \pi^k y^k - jx \ : \ \sum_{k\in K} d^k y^k \le w^0 + x, \ (y,x) \in \mathcal{D}_U\} \ \ (jx = z)\end{aligned}$$ □

Proposition 2.23 Let $f_H = r(d(H) - w^0) = d(H) - w^0 - \lfloor d(H) - w^0 \rfloor$. Inequality (2.8) is facet-defining for $\mathrm{conv}(\mathcal{F}_U)$ if either H is maximal c-strong in the j-split relaxation; $f_H > (j-1)/j$ and $w^0 \ge 0$; or $d^k > f_H$ for all $k \in H$ and $d^k < 1 - f_H$ for all $k \in K \setminus H$.

Proof Let (C, z) denote a point $(y, z) \in \mathbb{B}^{|K|} \times \mathbb{Z}$ where $y^k = 1$ for all $k \in C$, and $y^k = 0$ otherwise. Consider the following $|K| + 1$ affinely independent points of $\mathcal{F}_U^j$:

$(H, \lceil j(d(H) - w^0) \rceil)$ if $c_H^j \neq 0$, $(\emptyset, 0)$ if $c_H^j = 0$;
$(H \setminus \{k\}, \lceil j(d(H) - d^k - w^0) \rceil)$ for $k \in H$;
$(H \cup \{k\}, \lceil j(d(H) + d^k - w^0) \rceil)$ for $k \in K \setminus H$.

Since H is maximal c-strong in j-split relaxation, we have $\lceil j(d(H) - w^0) \rceil = \lceil j(d(H) - d^k - w^0) \rceil + \lceil jd^k \rceil$ for all $k \in H$ and $\lceil j(d(H) - w^0) \rceil = \lceil j(d(H) + d^k - w^0) \rceil + \lfloor jd^k \rfloor$ for all $k \in K \setminus H$. Therefore, after replacing z with $x = z/j$, these points satisfy the j-split c-strong inequality (2.8) at equality. To complete the proof, it is enough to show that z/j is integer for the points above. If $j\ r(w) > j - 1$, then $\lceil j\ r(w) \rceil = j$ since $r(w) < 1$, so $\lceil jw \rceil = j\lfloor w \rfloor + \lceil j\ r(w) \rceil = j\lceil w \rceil$. Thus, $\lceil j(d(H) - w^0) \rceil / j = \lceil d(H) - w^0 \rceil$ for the first point since $f_H > (j-1)/j$. Similarly, $\lceil j(d(H) - d^k - w^0) \rceil / j = \lfloor d(H) - w^0 \rfloor$ for $k \in H$ and $\lceil j(d(H) + d^k - w^0) \rceil / j = \lceil d(H) - w^0 \rceil$ for $k \in K \setminus H$; since $d^k > f_H$ for all $k \in H$, and $d^k < 1 - f_H$ for all $k \in K \setminus H$. □

Proposition 2.25 For any cover C, $\underline{\alpha} \le 1 \le \overline{\alpha}$ holds.

Proof We show that $\underline{\alpha} \le 1$ and $\overline{\alpha} \ge 1$.

$$\underline{\alpha} = \frac{1}{\lceil d(C) + d(K_1) - a_0 \rceil - \nu} = \frac{1}{\lceil r \rceil} \le 1$$

$$\overline{\alpha} \ge \min\left\{ \frac{\xi(\nu) - \xi(x)}{\nu - x} \ : \ x \in \{\lceil d(K_1) - w^0 \rceil, \dots, \nu - 1\} \right\}, \ \ (\text{as } \xi(\nu) \le |C| - 1)$$

$$\text{where } \xi(x) = \max\left\{ \sum_{k\in C} y^k \ : \ \sum_{k\in C} d^k y^k \le w^0 - d(K_1) + x, \ y^k \in \{0,1\} \ k \in C \right\}$$

≥ 1 (as $d^k < 1$ for all $k \in C$, $\xi(\nu) - \xi(x) \ge \nu - x$ for all x) □

Proposition 2.26 The maximal c-strong inequalities are equivalent to the lifted minimal cover inequalities with $\alpha = \underline{\alpha}$.

Proof Let H be maximal c-strong. Then, the c-strong inequality $\sum_{k\in H} y^k \le c_H + x$ is facet-defining for $\mathrm{conv}(\mathcal{F}_U)$, and H is a minimal cover with $\nu = \lfloor d(H) - w^0 \rfloor$, $K_0 = K \setminus H$ and $K_1 = \emptyset$. Consider the cover inequality lifted with the capacity variable using $\alpha = \underline{\alpha} =$

1, $\sum_{k\in H} y^k \leq |H| - 1 - \nu + x$. Since $d^k < 1$ and since H is maximal c-strong, we have $\lfloor d(H) - w^0 \rfloor < d(H) - w^0$, it follows that $|H| - 1 - \nu = c_H$. Since the c-strong inequality is facet-defining, the lifting coefficients of all of the projected binary variables must be 0. Thus, the maximal c-strong inequality is indeed a lifted minimal cover inequality. The other direction follows from Theorem 2.21 since $\underline{\alpha} = 1$ and hence the coefficients of the lifted cover inequality are integer. □

Lemma 2.28 If C is a minimal cover and $\alpha = \overline{\alpha}$, then the lifting coefficients of inequality (2.9) satisfy $\alpha^k \leq |C| - 1$ for all $k \in K_0$ and $-\alpha^k \leq |C| - 1$ for all $k \in K_1$.

Proof From the definition of $\overline{\alpha}$, $\overline{\alpha} \leq |C| - 1$ with $x = \nu - 1$. Since C is a minimal cover, the lifted inequality is facet-defining for $\mathrm{conv}(\mathcal{F}_U)$. Then from Proposition 2.18, since $0 < d^k < 1$, we have $\alpha^k \leq \overline{\alpha}$ for $k \in K_0$ and $-\alpha^k \leq \overline{\alpha}$ for $k \in K_1$. □

Theorem 2.29 For a minimal cover, a lifted knapsack cover inequality with $\alpha = \overline{\alpha}$ can be constructed in $\mathcal{O}(|K|^3)$.

Proof We have already argued that $\overline{\alpha}$ can be computed in $\mathcal{O}(|K| \log |K|)$. Since the coefficients of the objective function of the lifting problem BLP is bounded, when computing α^k $k \in K_0 \cup K_1$, it is more efficient to solve BLP with the dual knapsack formulation KP2 in Section 2.3.4.1. Scaling the objective of BLP by $1/\overline{\alpha}$, we write BLP in the form of UFP$_f$. Since $\overline{\alpha}$ is a common multiple of the coefficients $\alpha^k/\overline{\alpha}$ and $\overline{\alpha} \leq |C| - 1 < |K|$, from Theorem 2.17 the lifting problem for a single binary variable can be solved in $\mathcal{O}(|K|^3)$ by dynamic programming. Furthermore, similar to the lifting of binary knapsack cover inequalities (Zemel 1989), the lifting coefficients of all projected variables can also be computed in $\mathcal{O}(|K|^3)$ by dynamic programming, since the set of variables in the knapsack problems solved for lifting are nested. □

Appendix C

Proofs of results in Chapter 4

Theorem 4.2 For any non-empty 2-partition (A, B) of G with $|[AB]| \geq 3$, the 2-partition inequality (4.9) is facet-defining for the convex hull of feasible solutions of SDC if the two sub-graphs G'_A and G'_B are 2-connected, and either $\bar{r}_A > 1/2$ or $d_A > \max\{d_B, 2\}$.

Proof For the non-empty partition (A, B), let $G'_A = (A, F_A), G'_B = (B, F_B)$ be the sub-graphs defined by them. Define $\mathcal{C}'$ as the set of all directed cycles that do not contain any edges in $[AB]$, and let $\bar{\mathcal{C}} = \mathcal{C} \setminus \mathcal{C}'$. Define K' as the set of commodities that have source and destination nodes s^k and t^k in the same sub-graph. For each commodity k, consider an arbitrary arborescence (T_k) rooted at the source node s_k. Consider an arbitrary equation

$$\sum_{(ij)\in F} \sum_{k\in K} \pi^k_{ij} y^k_{ij} + \sum_{[ij]\in E} \beta_{[ij]} x_{[ij]} + \sum_{c\in\mathcal{C}} \alpha_c z_c = \gamma \tag{C.1}$$

on the variables (y, x, z). Since by adding appropriate multiples of (3.20) for commodity k for all nodes in the depth-first sequence of T_k, we can eliminate the coefficients of the flow variables corresponding to arcs in T_k in (C.1); without loss of generality we assume $\pi^k_{ij} = 0,\ \forall (ij) \in T_k,\ \forall k$. Let x' be an arbitrary feasible solution that satisfies (4.9) at equality. Later in the proof, we show the existence of such x' under the assumptions of the theorem. We use x' to derive other feasible points that satisfy (4.9) at equality, and prove that they define (C.1) up to a scalar multiple, and a multiple of equalities (3.20). Let ϵ be an infinitesimally small positive constant, and $d' = d_A + \bar{r}_A - 1$.

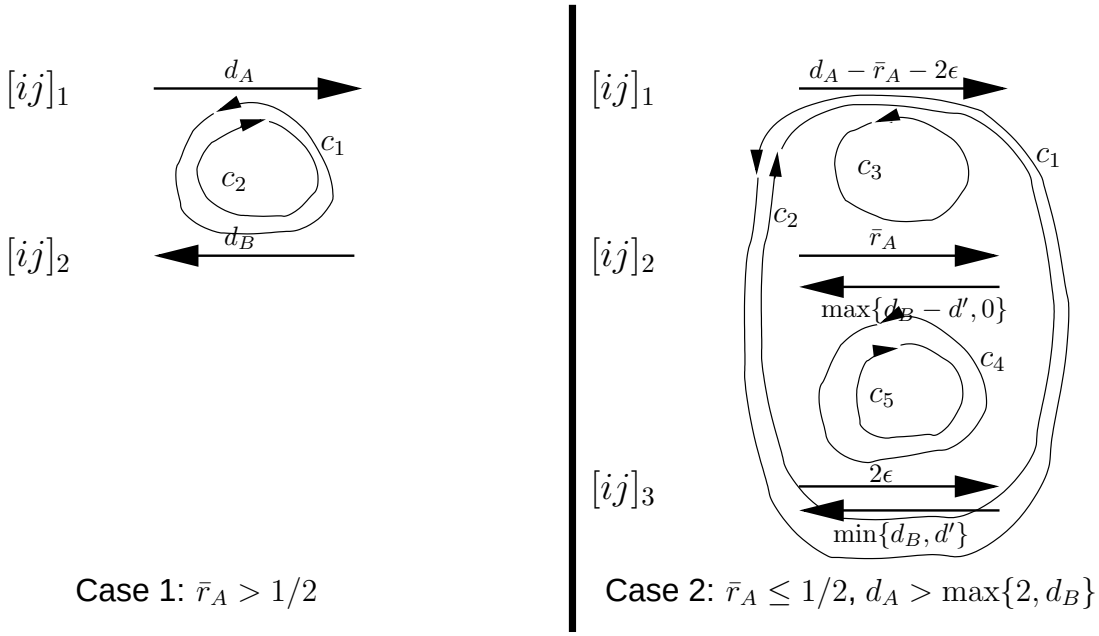
We first show that $\beta_{[ij]} = 0,\ \forall [ij] \in E \setminus [AB]$. For any edge $[ij] \in E \setminus [AB]$, we increase the capacity by one unit to obtain a new feasible solution x'' that still satisfies (4.9) at equality. Substituting x' and x'' into (C.1), $\beta_{[ij]} = 0,\ [ij] \in E \setminus [AB]$.

We now show that $\alpha_c = 0,\ \forall c \in \mathcal{C}'$. Consider the feasible solution x'' obtained by adding one unit of capacity to x' on all edges $[ij] \in E \setminus [AB]$. This new solution also satisfies (4.9) at equality. For any directed cycle $c \in \mathcal{C}'$, we increase the slack reserved for the cycle by one unit to obtain a new feasible solution x''' that satisfies (4.9) at equality. Substituting x'' and x''' into (C.1), $\alpha_c = 0,\ c \in \mathcal{C}'$.

Next we prove that $\pi^k_{ij} = 0,\ \forall (ij) \in F \setminus (AB \cup BA),\ k \in K$. Consider the feasible solution x'' obtained by adding one unit of capacity to x' on all edges $[ij] \in E \setminus [AB]$; redirecting the flow such that there exists some positive flow on all arcs in $T_k \setminus (AB \cup BA)$, for each commodity k without increasing the flow on the arcs in $AB \cup BA$; and increasing the slack reserved on directed cycles in $\mathcal{C}'$ such that the new flow is covered. This can

be done because the sub-graphs obtained by removing $[AB]$ are 2-connected. Since we only increased capacity on all edges in $E \setminus [AB]$, and increased flow and slack on arcs in $F \setminus AB$, x'' is feasible, and also satisfies (4.9) at equality. For commodity k, consider any arc $(ij) \in F \setminus (AB \cup BA \cup T_k)$. We can redirect ϵ additional units of flow through this arc by changing flow by ϵ units only among the arcs in $T_k \setminus (AB \cup BA)$ (by using the fundamental circuit for tree T_k). We satisfy (3.21) by reserving additional slack on directed cycles in $\mathcal{C}'$ that cover arcs on which flow is increased. This new solution (x''') is feasible since unused capacity exists on these edges by definition of x'', and also satisfies (4.9) at equality. Substituting x'' and x''' into (C.1), $\pi_{ij}^k = 0,\ (ij) \in F \setminus (AB \cup BA),\ k \in K$.

Figure C.1: Feasible solution



Case 1: $\bar{r}_A > 1/2$

Case 2: $\bar{r}_A \leq 1/2$, $d_A > \max\{2, d_B\}$

For the coefficients $\beta_{[ij]}$ $[ij] \in [AB]$, we treat the case $\bar{r}_A > 1/2$ separately from $d_A > \max\{2, d_B\}$. For both cases, we only need to route commodities in $K \setminus K'$ using the arcs in $AB \cup BA$. Since the two sub-graphs G'_A and G'_B are 2-connected, all other commodities can be routed using arcs in $F \setminus (AB \cup BA)$ and can be covered using directed cycles that do not cross the 2-partition ($\mathcal{C}'$). For all feasible points considered in the rest of the proof, we install sufficiently large capacity on edges in $E \setminus [AB]$ and slack on directed cycles in $\mathcal{C}'$ while still satisfying (4.9) at equality.

Case 1. $\bar{r}_A > 1/2$. Consider the feasible solution x' shown in Figure C.1. We set flow variables $y^A_{(ij)_1} = d_A$ and $y^B_{(ji)_2} = d_B$; directed cycle variables $z_{c_1} = d_A + \epsilon$ and $z_{c_2} = \epsilon$; and installed capacity variables $x_{[ij]_1} = \lceil d_A \rceil$ and $x_{[ij]_2} = \lceil d_A \rceil$. Since $\bar{r}_A > 1/2$, $2\lceil d_A \rceil = \lceil 2d_A \rceil$; thus x' satisfies (4.9) at equality. Furthermore, we can obtain some other feasible solution x'' by choosing another edge $[ij]_3$ instead of either $[ij]_1$ or $[ij]_2$ since G'_A and G'_B are connected, and there exists sufficient spare capacity on all edges in $E \setminus [AB]$. This new solution still satisfies (4.9) at equality. Substituting such pairs of solutions into (C.1), $\beta_{[ij]} = \beta,\ \forall [ij] \in [AB]$.

Case 2. $\bar{r}_A \leq 1/2$, $d_A > \max\{2, d_B\}$. Consider the solution x' shown in Figure C.1. We set flow variables $y^A_{(ij)_1} = d_A - \bar{r}_A - 2\epsilon$, $y^A_{(ij)_2} = \bar{r}_A$, $y^A_{(ij)_3} = 2\epsilon$, $y^B_{(ji)_2} = \max\{d_B - d', 0\}$ and $y^B_{(ji)_3} = \min\{d_B, d'\}$; directed cycles variables $z_{c_1} = d_A - 1 + \epsilon$, $z_{c_2} = \epsilon$, $z_{c_3} = 1 - \bar{r}_A - 2\epsilon$, $z_{c_4} = \bar{r}_A + \epsilon$, and $z_{c_5} = 3\epsilon/2$; and installed capacity variables $x_{[ij]_1} = \lceil d_A \rceil - 1$, $x_{[ij]_2} = 1$ and $x_{[ij]_3} = \lfloor d_A \rfloor$. Since $\bar{r}_A \leq 1/2$, $\lceil d_A \rceil + \lfloor d_A \rfloor = 2\lceil d_A \rceil - 1 = \lceil 2d_A \rceil$; thus x' satisfies (4.9) at equality. Furthermore, we can obtain another solution x'' that satisfies (4.9) at equality, by interchanging the values (flow, directed cycle, capacity) on edges $[ij]_2$ and $[ij]_3$. This solution is feasible since the sub-graphs G'_A and G'_B are 2-connected, and there exists sufficient spare capacity on all the edges. Substituting such pairs of solutions x' and x'' into (C.1), $\beta_{[ij]} = \beta,\ \forall [ij] \in [AB]$.

For the rest of the coefficients, we define Y_{ij} and Z_{ij} as the total flow and slack reserved for directed cycles on arc (ij), respectively. For the solution x' (for both cases), we have $Y_{ij} < Z_{ji}$ and $Y_{ij} + Z_{ij} < x_{[ij]}$ $(ij) \in AB \cup BA$, whenever $x_{[ij]} > 0$. For instance, consider edge $[ij]_2$ in Case 2. We have $Y_{(ij)_2} = \bar{r}_A$, $Z_{(ji)_2} = z_{c_4} = \bar{r}_A + \epsilon > Y_{(ij)_2}$ and $Y_{(ji)_2} = \max\{d_B - d', 0\} < 1 - \bar{r}_A$, $Z_{(ij)_2} = z_{c_3} + z_{c_5} = 1 - \bar{r}_A - \epsilon/2 > Y_{(ji)_2}$. Furthermore, $Y_{(ij)_2} + Z_{(ij)_2} = 1 - \epsilon/2 < x_{[ij]_2}$ and $Y_{(ji)_2} + Z_{(ji)_2} < 1 = x_{[ij]_2}$. Hence, we can obtain a new feasible solution (x'') by reserving ϵ additional units of slack on any directed cycle $c \in \bar{\mathcal{C}}$. This new solution (x'') satisfies (4.9) at equality. Substituting x' and x'' into (C.1), $\alpha_c = 0,\ c \in \bar{\mathcal{C}}$.

Now for commodity k, consider any arc $(ij) \in (AB \cup BA) \setminus T_k$. Starting with solution x', we can redirect ϵ additional units of flow through this arc by changing flow by ϵ units only among the arcs in T_k (using the fundamental circuit). Furthermore, we satisfy (3.21) by reserving additional slack on some directed cycle in $\mathcal{C}$ that covers arc (ij). This new solution (x''') is feasible since unused capacity exists on all edges by definition of x', and also satisfies (4.9) at equality. Substituting x' and x''' into (C.1), $\pi^k_{ij} = 0,\ (ij) \in (AB \cup BA) \setminus T_k$. Finally, plugging x' into (C.1), we obtain $\gamma = \lceil 2d_A \rceil \beta$. Dividing (C.1) by β, it reduces to (4.9). □

Theorem 4.3 For $H_1 \subseteq AB$, $H_2 \subseteq BA$, the 2-partition inequalities

$$r_A x([H_1]) + (1 - r_A)x([H_2]) + y^A(AB \setminus H_1) + z(AB \setminus H_1) - y^A(H_2) - y^A(BA \setminus [H_1]) \geq r_A \eta_A \tag{C.2}$$

$$r_B x([H_2]) + (1 - r_B)x([H_1]) + y^B(BA \setminus H_2) + z(BA \setminus H_2) - y^B(H_1) - y^B(AB \setminus [H_2]) \geq r_B \eta_B \tag{C.3}$$

are valid for $\mathcal{F}_2$.

Proof We show the validity for inequality (C.2). Relaxing the flow balance constraint for commodity A,

$$y^A(AB) \geq d_A + y^A(BA \setminus [H_1]).$$

Using (4.8) and the fact that the slack reserved for directed cycles containing arcs AB is greater than the flow in the reverse direction, we have

$$z(AB) \geq d_A + y^A(H_2).$$

Since the capacity installed on a set of edges is greater than the net flow and the slack reserved on directed cycles containing the corresponding arcs,

$$x([H_1])+ \geq y^A(H_1) + z(H_1).$$

Finally, adding these three inequalities, we have

$$x([H_1]) + y^A(AB\setminus H_1) + z(AB\setminus H_1) - y^A(H_2) - y^A(BA\setminus [H_1]) \geq 2d_A. \tag{C.4}$$

Adding $x([H_2]) - x[H_2]$ to the left-hand-side of (C.4) and applying mixed-integer rounding (see Section 1.5.2) to the resulting inequality, we obtain the inequality (C.2). Validity of (C.3) can be shown in the same way by considering commodity B instead of commodity A. □

Theorem 4.4 Let $\mathcal{F}_1$ denote the convex hull of points satisfying the 1-commodity 2-partition relaxation; i.e., all $(y \in \mathbb{R}^{2|AB|}, z \in \mathbb{R}^{2|AB|}, x \in \mathbb{Z}^{|AB|})$ satisfying (4.12)-(4.15). The inequality

$$r_A x([H_1]) + y^A(AB\setminus H_1) + z(AB\setminus H_1) - y^A(BA\setminus [H_1]) \geq r_A\eta_A \tag{C.5}$$

is facet-defining for $\mathcal{F}_1$ if and only if $r_A > 0$ and $H_1 \neq \emptyset$.

Proof First we prove that (C.5) is not facet-defining for $\mathcal{F}_1$ if $r_A = 0$ or $H_1 = \emptyset$. If $r_A = 0$, then (C.5) reduces to $z(AB\setminus H_1) + y(AB\setminus H_1) - y(BA\setminus [H_1]) \geq 0$, which is dominated by the sum of the non-negativity constraints of $y(AB\setminus H_1)$ and the survivability constraints of $(BA\setminus [H_1])$. If $H_1 = \emptyset$, then the inequality reduces to $z(AB) + y^A(AB) - y^A(BA) \geq r_A\eta_A$, which is dominated by $z(AB) = z(BA) \geq y^A(AB) \geq d_A$; since $2d_A \geq r_A\eta_A$.

Next, we prove that (C.5) is facet-defining if $r_A > 0$ and $H_1 \neq \emptyset$. Let f_{ij} and g_{ij} denote the unit vectors of flow and directed cycle variables, respectively, for $(ij) \in AB \cup BA$, and $h_{[ij]}$ denote the unit vector of the capacity variables for $[ij] \in [AB]$.

Let $\sum_{(ij)\in AB\cup BA}(\pi_{ij}y^A_{ij} + \alpha_{ij}z_{ij}) + \sum_{[ij]\in[AB]}\beta_{[ij]}x_{[ij]} = \beta_0$ define an arbitrary hyperplane that contains the face induced by (C.5). Let $(st) \in H_1$. Since all points of $\mathcal{F}_1$ satisfy $y^A(AB) - y^A(BA) = d_A$ and $z(AB) = z(BA)$, we may add multiples of these equalities to a valid inequality without changing it. Therefore, without loss of generality we assume $\pi_{st} = \alpha_{st} = 0$. Consider the following points of the face. Let $u^0 = \eta_A h_{st} + d_A g_{st} + d_A g_{ts} + d_A f_{st}$. From points $u^0 + h_{[ij]}$, we see that $\beta_{[ij]} = 0$, $\forall [ij] \in [AB]\setminus[H_1]$. From points $u^0 + \epsilon g_{st} + h_{[ij]} + \epsilon g_{ji}$, we obtain $\alpha_{ij} = 0, \forall (ij) \in BA\setminus[H_1]$. Similarly, from the point $u^0 + \epsilon g_{st} + \epsilon g_{ts}$, we obtain $\alpha_{ts} = 0$. Therefore, $\alpha_{ij} = 0 \forall (ij) \in BA$. From points $u^0 + \epsilon g_{ts} + \epsilon f_{st} + h_{[ij]} + \epsilon g_{ij} + \epsilon f_{ji}$, we obtain $\alpha_{ij} = -\pi_{ji}$, $\forall (ij) \in AB\setminus H_1$.

For the rest of the coefficients, consider points $v^{ij} = u^0 - h_{st} - r_A g_{ts} - r_A f_{st} + h_{[ij]} + r_A g_{ji} + r_A f_{ij}$ and $w^{ij} = u^0 - h_{st} - r_A g_{st} + h_{[ij]} + r_A g_{ij}$, for $(ij) \in AB\setminus\{(st)\}$. Comparing u^0 and v^{ij}, we have $\beta_{st} = r_A\pi_{ij}$, $\forall (ij) \in AB\setminus H_1$. On the other hand, comparing u^0 and w^{ij}, we have $\beta_{st} = r_A\alpha_{ij}$, $\forall (ij) \in AB\setminus H_1$. From w^{ij} and $w^{ij} + \epsilon g_{ij} + \epsilon g_{ji}$, we obtain $\alpha_{ij} = \alpha_{ji}$, $\forall (ij) \in H_1\setminus\{(st)\}$. Further, comparing v^{ij} and w^{ij}, we obtain $\pi_{ij} = 0$, $\forall (ij) \in H_1\setminus\{(st)\}$. From v^{ij} and $v^{ij} - \epsilon g_{ts} - \epsilon f_{st} + \epsilon f_{ij} + \epsilon g_{ij} - \epsilon g_{ji}$, we obtain $\alpha_{ij} = -\alpha_{ji}$, $\forall (ij) \in H_1\setminus\{(st)\}$. Thus, $\alpha_{ij} = \alpha_{ji} = 0$, $\forall (ij) \in H_1\setminus\{(st)\}$. Comparing u^0 and w^{ij}, we also see that $\beta_{st} = \beta_{[ij]}$, $\forall [ij] \in [H_1]$. Also, from points w^{ij} and $w^{ij} - \epsilon g_{st} + \epsilon g_{ts} + \epsilon f_{st} + h_{[ij]} + 2\epsilon g_{ij} + \epsilon f_{ji}$, we obtain $\pi_{ji} = 0$, $\forall (ij) \in H_1\setminus\{(st)\}$. From the point $u^0 + \epsilon g_{st} + \epsilon g_{ts} + \epsilon f_{st} + \epsilon f_{ts}$, we

see that $\pi_{ts} = 0$. Finally, plugging in these coefficients for u^0, we find that $\eta_A \beta_{[st]} = \beta_0$. Therefore, the points described above define the hyperplane up to a scalar multiple and a multiple of the two equalities. Dividing all coefficients by $\beta_{[st]}/r_A$, we arrive at (C.5). Hence, we have shown that the face of $\mathcal{F}_1$ induced by (C.5) has $5|[AB]| - 2$ affinely independent points. □

Appendix D

Proofs of results in Chapter 5

Theorem 5.2 The pricing problem of directed p-cycles (PPP) is $\mathcal{NP}$-hard.

Proof We prove the theorem by reducing PPP to the decision version of TSP (Garey and Johnson 1979): Given a complete directed graph $G' = (V, F)$, weights $d : F \mapsto \mathbb{Z}_+$, and a positive integer k, does there exist a hamiltonian cycle in G' with total weight $< k$? To answer TSP, we construct the following instance of PPP. Let $n = |V|$, and $M > nk$. For $(ij) \in F$, let $f^a_{ij} = d_{[ij]} + (M(n-3) - 2k)/(n(n-1))$ and $f^h_{ij} = -(M+k)/(n(n-1))$. This construction is polynomial in n and $\log k$. Let $d(c)$ be the weight of cycle c for the TSP. Now, any directed p-cycle on G' with ℓ arcs has exactly $\ell^2 - 3\ell$ directed chords. Hence the weight of directed p-cycle c with ℓ arcs is equal to

$$\begin{aligned}
\sum_{(ij)\in F} \alpha^c_{ij} f^a_{ij} + \sum_{[ij]\in E} \rho^c_{[ij]} f^h_{ij} &= d(c) + \frac{\ell(M(n-3) - 2k)}{n(n-1)} - \frac{\ell(\ell-3)(M+k)}{n(n-1)} \\
&= d(c) + M\frac{\ell(n-3) - \ell(\ell-3)}{n(n-1)} - k\frac{2\ell + \ell(\ell-3)}{n(n-1)} \\
&= d(c) + M\frac{\ell(n-\ell)}{n(n-1)} - k\frac{\ell(\ell-1)}{n(n-1)}. \qquad \text{(D.1)}
\end{aligned}$$

When M is chosen as above, (D.1) is positive unless $\ell = n$. Hence, PPP has an affirmative answer only if the directed p-cycle is Hamiltonian. However, since the weight of any Hamiltonian directed p-cycle c on G is $d(c) - k$ (the second term in (D.1) vanishes when $\ell = n$), PPP has an affirmative answer if there exists a Hamiltonian cycle c of weight $d(c) < k$. Hence, TSP has an affirmative answer if and only if PPP has an affirmative answer. □

Appendix E

Proofs of results in Chapter 6

Theorem 6.10 The maximum chord problem (MCP) is $\mathcal{NP}$-hard.

Proof The proof is by reduction from the $\mathcal{NP}$-hard hamiltonian path problem (HPP) (Garey and Johnson 1979): Given an undirected network $G = (V, E)$, does there exist a simple path that passes through all nodes in G?

To answer HPP, we construct the following instance of MCP. Let the network $\bar{G} = (\bar{V}, \bar{E})$, where $\bar{V} = V \cup \{u\}$, and $\bar{E} = E \cup \{[ui],\ \forall i \in V\}$. The partition is defined as $A = \{u\}, B = \bar{V} \setminus A$. Let $n = |V|$. We note that this is a polynomial construction in n since $|\bar{V}| = n + 1$ and $|\bar{E}| = |E| + n$.

For any directed p-cycle c, let ℓ_c be the number of times it uses an arc in AB, and let ω_c be the number of chords among the edges in $[AB]$. The number of edges in the cut is n, and hence the optimal solution to MCP $\leq n - 2$ since $\ell_c \geq 1$ for any directed p-cycle c. Now, any cycle that actually attains this value has to pass through all the nodes in B. This implies that there exists an HPP in G. It is also easily seen that if there exists a Hamiltonian path in G (say $i - \cdots - j$), then there exists a cycle in $\bar{G}$ ($u - i - \ldots - j - u$) that has $\omega_c = n - 2$ and $\ell_c = 1$. Hence HPP on G has an affirmative answer if and only if MCP has a solution $n - 2$ for the partition (A, B). □

Proposition 6.7 If the sub-graphs G'_A and G'_B are 2-connected, then the solution $v_{ij} = 1$ if $(ij) \in AB$, and 0 otherwise, is optimal when $u_{ij} = 1$ if $(ij) \in AB$, and 0 otherwise.

Proof First, we prove that $v_{ij} = 0$ for all $(ij) \in F \setminus (AB \cup BA)$ for all feasible solutions. Then, we show that there exists at least one optimal solution such that $v_{ij} = v_{AB},\ (ij) \in AB$ and $v_{ij} = v_{BA},\ (ij) \in BA$. Finally, we prove that $v_{BA} = 0$ and $v_{AB} = 1$ for at least one optimal solution. We define $\mathcal{C}'$ as the set of all directed cycles that do not contain any edges in $[AB]$, and let $\bar{\mathcal{C}} = \mathcal{C} \setminus \mathcal{C}'$. For directed cycle c, let ℓ_c be the number of times it uses any arc in AB.

For all $(ij) \in F \setminus [AB]$, there exists some $c \in \mathcal{C}'$ such that $\alpha^c_{ij} = 1$, since the sub-graphs G'_A and G'_B are 2-connected. Then, since (6.20) reduces to $-\sum_{(ij)\in F\setminus(AB\cup BA)} \alpha^c_{ij} v_{ji} \geq 0$ for all $c \in \mathcal{C}'$, we have $v_{ij} = 0$ for all $(ij) \in F \setminus (AB \cup BA)$.

Now, (6.20) reduces to

$$-\sum_{(ij)\in BA} \alpha^c_{ij} v_{ji} + \sum_{(ij)\in AB} \alpha^c_{ij}(1 - v_{ji}) \geq 0 \qquad \forall c \in \bar{\mathcal{C}} \qquad \text{(E.1)}$$

Since $v_{ij} = 0$ for all $(ij) \in F \setminus (AB \cup BA)$ and sub-graphs G'_A and G'_B are connected, the objective function reduces to

$$\max d_A(1 + \min_{(ij)\in AB} v_{ij}) + d_B(\min_{(ij)\in BA} v_{ij}) \tag{E.2}$$

Suppose that in all optimal solutions, there exist some $(ab), (cd) \in AB$ such that $v_{ab} > v_{cd}$. Then, we can reduce v_{ab} to v_{cd} without decreasing the objective. We have a contradiction; thus there exists at least one optimal solution such that $v_{ab} = v_{cd}$ for all $(ab), (cd) \in AB$. We restrict our discussion to such solutions. Similarly, we can prove that least one optimal solution such that $v_{ab} = v_{cd}$ for all $(ab), (cd) \in BA$. We have proved that there exists at least one optimal solution such that $v_{ij} = v_{AB},\ (ij) \in AB$ and $v_{ij} = v_{BA},\ (ij) \in BA$.

Now, (E.1) reduces to $-\ell_c v_{AB} + \ell_c(1 - v_{BA}) \geq 0$ for all $c \in \bar{\mathcal{C}}$, which is equivalent to $v_{AB} + v_{BA} \leq 1$. Since (E.2) reduces to $\max d_A(1 + v_{AB}) + d_B v_{BA}$, and $d_A \geq d_B$ without loss of generality, there exists at least on optimal solution with $v_{AB} = 1$ and $v_{BA} = 0$. □

Proposition 6.9 If the sub-graphs G'_A and G'_B are 2-connected, then the solution $v_{ij} = \mu_{AB}$ if $(ij) \in AB$, and 0 otherwise, is optimal when u is fixed so that the left hand side corresponds to the cardinality-k cut-set inequality (6.32), i.e., when $u_{ij} = 1$ if $(ij) \in AB$, and 0 otherwise.

Proof First, we prove that $v_{ij} = 0$ for all $(ij) \in F \setminus (AB \cup BA)$ for all feasible solutions. Then, we show that there exists at least one optimal solution such that $v_{ij} = v_{AB},\ (ij) \in AB$ and $v_{ij} = v_{BA},\ (ij) \in BA$. Finally, we prove that $v_{BA} = 0$ and $v_{AB} = \mu_{AB}$ for at least one optimal solution. We define $\mathcal{C}'$ as the set of all directed p-cycles that do not contain any edges in $[AB]$, and let $\bar{\mathcal{C}} = \mathcal{C} \setminus \mathcal{C}'$. For directed p-cycle c, let ℓ_c be the number of times it uses an arc in AB, and let ω_c be the number of chords among the edges in $[AB]$.

For all $(ij) \in F \setminus [AB]$, there exists some $c \in \mathcal{C}'$ such that $\alpha^c_{ij} = 1$, since the sub-graphs G'_A and G'_B are 2-connected. Then, since (6.31) reduces to $-\sum_{(ij)\in F\setminus(AB\cup BA)}(\alpha^c_{ij} v_{ji} + \rho^c_{[ij]} v_{ij}) \geq 0$ for all $c \in \mathcal{C}'$, we have $v_{ij} = 0$ for all $(ij) \in F \setminus (AB \cup BA)$.

Now, (6.31) reduces to

$$-\sum_{(ij)\in BA} \alpha^c_{ij} v_{ji} + \sum_{(ij)\in AB} \alpha^c_{ij}(1 - v_{ji}) - \sum_{(ij)\in(AB\cup BA)} \rho^c_{[ij]} v_{ij} \geq 0 \qquad \forall c \in \bar{\mathcal{C}} \tag{E.3}$$

Since $v_{ij} = 0$ for all $(ij) \in F \setminus (AB \cup BA)$ and sub-graphs G'_A and G'_B are connected, the objective function reduces to

$$\max d_A(1 + \min_{(ij)\in AB} v_{ij}) + d_B(\min_{(ij)\in BA} v_{ij}) \tag{E.4}$$

Suppose that in all optimal solutions, there exist some $(ab), (cd) \in AB$ such that $v_{ab} > v_{cd}$. Then, we can reduce v_{ab} to v_{cd} without decreasing the objective. We have a contradiction; thus there exists at least one optimal solution such that $v_{ab} = v_{cd}$ for all $(ab), (cd) \in AB$. We restrict our discussion to such solutions. Similarly, we can prove that least one optimal solution such that $v_{ab} = v_{cd}$ for all $(ab), (cd) \in BA$. We have proved that there exists at least one optimal solution such that $v_{ij} = v_{AB},\ (ij) \in AB$ and $v_{ij} = v_{BA},\ (ij) \in BA$.

Now, (E.3) reduces to $-\ell_c v_{AB} + \ell_c(1 - v_{BA}) - \omega_c(v_{AB} + v_{BA}) \geq 0$ for all $c \in \bar{\mathcal{C}}$, which is equivalent to $v_{AB} + v_{BA} \leq \ell_c/(\ell_c + \omega_c)$. for all $c \in \bar{\mathcal{C}}$. Since (E.2) reduces to

$\max d_A(1+v_{AB})+d_Bv_{BA}$, and $d_A \geq d_B$ without loss of generality, there exists at least on optimal solution with $v_{AB}=\mu_{AB}$ and $v_{BA}=0$. □

Appendix F

Proofs of results in Chapter 7

Theorem 7.2 Let $H \subseteq P^+$ and $n = |N|$. If $\pi x - y \leq \pi_0 - v(H)$ defines a facet of $\mathrm{conv}(M_n^{\leq}(b - v(H)))$, then $\pi x + w(H) - w(P^-) \leq \pi_0$ defines a facet of $\mathrm{conv}(M(b))$.

Proof We first demonstrate the validity of the inequality, and then describe the affinely independent points which prove that it defines a facet. Using the results in Atamtürk (2003b), we aggregate all variables in P^- into a single continuous variable z.

Assume that $\pi x + w(H) - z \leq \pi_0$ is not valid for $M(b)$. $\Longrightarrow$ There exist $\bar{w}, \bar{y}, \bar{x}$ such that $a\bar{x} + \bar{w}(P^+) - \bar{z} \leq b$ and $\pi\bar{x} + \bar{w}(H) - \bar{z} > \pi_0$. Setting $\hat{y} = \bar{z} + v(H) - \bar{w}(H)$, we have $a\bar{x} - \hat{y} \leq b - v(H)$ (since $\bar{w}(P^+ \setminus H) \geq 0$) and $\pi\bar{x} - \hat{y} > \pi_0 - v(H)$. Since $\bar{x}, \hat{y} \in M_n^{\leq}(b)(b - v(H))$, inequality $\pi x - y \leq \pi_0 - v(H)$ is not valid, and we have a contradiction.

Since $\pi x - y \leq \pi_0 - v(H)$ defines a facet of $\mathrm{conv}(M_n^{\leq}(b - v(H)))$, we have $n + 1$ affinely independent elements $p^1, \ldots p^{n+1}$ such that $\sum_{i=1}^n \pi_i p_i^j - p_0^j = \pi_0 - v(H)$ and $\sum_{i=1}^n a_i p_i^j - p_0^j \leq b - v(H)$ for all $j \in [1, n+1]$. From the affine independence of these points, we know that $\lambda_j = 0,\ j \in [1, n+1]$ is the only solution to

$$\sum_{j=1}^{n+1} \lambda_j p_i^j = 0,\ i \in [1, n], \qquad \sum_{j=1}^{n+1} \lambda_j p_0^j = 0, \qquad \sum_{j=1}^{n+1} \lambda_j = 0. \tag{F.1}$$

We assume without loss of generality that $\sum_{i=1}^n a_i p_i^1 - p_0^1 < b - v(H)$ and $p_0^2 > 0$; otherwise the facet is trivial. We show that the following $n + |P|$ elements of $M(b)$ are affinely independent; where the elements are listed as vectors with the components as follows.

$x_i,\ i \in [1,n]$	y	$w_k,\ k \in H$	$w_\ell,\ \ell \in P^+ \setminus H$	
$p_i^j,\ i \in [1,n]$	p_0^j	$v_k,\ k \in H$	0	$\forall j \in [1, n+1]$
$p_i^2,\ i \in [1,n]$	$p_0^2 - \epsilon$	$v_t - \epsilon I\{t = k\},\ t \in H$	0	$\forall k \in H$
$p_i^1,\ i \in [1,n]$	p_0^1	$v_k,\ k \in H$	$\epsilon I\{t = \ell\},\ t \in P^+ \setminus H$	$\forall \ell \in P^+ \setminus H$

To prove affine independence, we need to show that $\lambda = 0$ is the only solution to

$$\sum_{j=1}^{n+1} \lambda_j p_i^j + \sum_{k \in H} \lambda_k p_i^2 + \sum_{\ell \in C^+ \setminus H} \lambda_\ell p_i^1 = 0 \qquad i \in [1, n] \tag{F.2}$$

$$\sum_{j=1}^{n+1} \lambda_j p_0^j + \sum_{k \in H} \lambda_k (p_0^2 - \epsilon) + \sum_{\ell \in C^+ \setminus H} \lambda_\ell p_0^1 = 0 \tag{F.3}$$

$$(\sum_{j=1}^{n+1} \lambda_j + \sum_{t \in H} \lambda_t + \sum_{\ell \in C^+ \setminus H} \lambda_\ell) v_k - \epsilon \lambda_k = 0 \qquad k \in H \tag{F.4}$$

$$\lambda_\ell = 0 \qquad \ell \in P^+ \setminus H \tag{F.5}$$

$$\sum_{j=1}^{n+1} \lambda_j + \sum_{k \in H} \lambda_k + \sum_{\ell \in C^+ \setminus H} \lambda_\ell = 0 \tag{F.6}$$

From (F.4) and (F.6), $\lambda_k = 0,\ k \in H$. Thus, also using (F.5), (F.2) and (F.3) reduce to (F.1). Since $\lambda_j = 0,\ j \in [1, n+1]$ is the only solution to (F.1), we are done. □

Proposition 7.3 The inequalities $x_1 \geq 0$, $x_2 \geq 0$ define facets of $\mathrm{conv}(K_2^{\leq}(b))$. Upper bound constraints $x_1 \leq u_1$ and $x_2 \leq u_2$ define facets of $\mathrm{conv}(K_2^{\leq}(b))$ if and only if $t_2 \geq 1$ and $t_1 \geq 1$, respectively.

Proof For $i = 1, 2$, the inequalities $x_i \geq 0$ and $x_i \leq u_i$ are valid. Therefore, to prove that any of these inequalities define a facet, it is sufficient to show that there exist two feasible elements of $K_2^{\leq}(b)$ that satisfy it as an equality.

Since $u > 0$ by assumption (A.2), the points q^2 and q^1 are both distinct from $\bar{q}$. Now, the points $\bar{q}, q^2$ satisfy $x_1 = 0$, and the points $\bar{q}$ and q^1 satisfy $x_2 = 0$. Therefore, the inequalities $x_1 \geq 0$, $x_2 \geq 0$ define facets of $\mathrm{conv}(K_2^{\leq}(b))$.

If $t_2 \geq 1$, then points q^2 and q^3 satisfy $x_1 = u_1$. Similarly, if $t_1 \geq 1$, points q^1 and q^0 satisfy $x_2 = u_2$. Therefore, upper bound constraints $x_1 \leq u_1$ and $x_2 \leq u_2$ define facets of $\mathrm{conv}(K_2^{\leq}(b))$ if $t_2 \geq 1$ and $t_1 \geq 1$, respectively.

For $i \in [1, 2]$, point q^i is the only element of $K_2^{\leq}(b)$ that satisfies $x_{2-i} = u_{2-i}$ if $t_i < 1$. Thus, upper bound constraints $x_1 \leq u_1$ and $x_2 \leq u_2$ do not define facets of $\mathrm{conv}(K_2^{\leq}(b))$ only if $t_2 \geq 1$ and $t_1 \geq 1$, respectively. □

Proposition 7.4 The points q^1, q^2, $\bar{q}$, q^0 (distinct from q^1 if $t_1 > 1$), and q^3 (distinct from q^2 if $t_2 > 1$) are extreme points of $\mathrm{conv}(K_2^{\leq}(b))$.

Proof The points q^1, q^2, and $\bar{q}$ are elements of $K_2^{\leq}(b)$. To show that any of these points is an extreme point, it suffices to show that it is the solution of two faces of $\mathrm{conv}(K_2^{\leq}(b))$.

Point q^1 denotes the solution of $x_1 = 0$ and $x_2 = u_2$. Point q^2 denotes the solution of $x_2 = 0$ and $x_1 = u_1$. Point $\bar{q}$ denotes the solution of $x_2 = 0$ and $x_1 = 1$. Therefore, by Proposition 7.3, q^1, q^2, and $\bar{q}$ are extreme points of $\mathrm{conv}(K_2^{\leq}(b))$.

By definition of q^0, there exist no points $q \in K_2^{\leq}(b)$ such that $q_2 > q_2^0$. Also, there exist no points $q \in K_2^{\leq}(b)$ such that $q_2 = q_2^0$ and $q_1 > q_1^0$. Thus, q^0 can not be written as a convex combination of any two points in $\mathrm{conv}(K_2^{\leq}(b))$. Therefore, q^0 is an extreme point

of $\mathrm{conv}(K_2^{\leq}(b))$.

Similarly, by definition of q^3, there exist no points $q \in K_2^{\leq}(b)$ such that $q_1 > q_1^3$. Also, there exist no points $q \in K_2^{\leq}(b)$ such that $q_1 = q_1^3$ and $q_2 > q_2^3$. Thus, q^3 can not be written as a convex combination of any two points in $\mathrm{conv}(K_2^{\leq}(b))$. Therefore, q^3 is an extreme point of $\mathrm{conv}(K_2^{\leq}(b))$. □

Proposition 7.5 The point q^4 is an extreme point of $\mathrm{conv}(K_2^{\leq}(b))$ if and only if $q^5 \notin K_2^{\leq}(b)$; where q^5 is defined as $q_1^5 = q_1^4 + 1$ and $q_2^5 = q_2^4 - (q_2^0 - q_2^4)$.

Proof If $q^5 \notin K_2^{\leq}(b)$, then for $\delta \geq 0$ there exists no point $q \in K_2^{\leq}(b)$ with $q_2 = q_2^5 - \delta$ and $q_1 > q_1^5 + \delta$. Since $q_1' \leq q_1^0$ for all $q' \in K_2^{\leq}(b)$ with $q_2' \geq q_2^0$. Therefore, the point q^4 can not be written as a convex combination of any such $q, q' \in K_2^{\leq}(b)$. Hence, the point q^4 is an extreme point of $\mathrm{conv}(K_2^{\leq}(b))$ if $q^5 \notin K_2^{\leq}(b)$.

The point $q^0 \in K_2^{\leq}(b)$. By definition of q^5, the point $q^4 = (q^5 + q^0)/2$. Thus, The point q^4 is an extreme point of $\mathrm{conv}(K_2^{\leq}(b))$ only if $q^5 \notin K_2^{\leq}(b)$. □

Theorem 7.8 The number of non-trivial extreme points of $K_2^{\leq}(b)$ is bounded from above by $2\lfloor \log(\gamma + 1) \rfloor + 2$.

Proof We will use extreme points q^D and q^U in this proof; see Definition 7.7. Points q^U and q^D may be identical; in which case there exists no facet such that $\pi_1/\pi_2 = a_1/a_2$. However, there always exists a face such that $\pi_1/\pi_2 = a_1/a_2$. Without loss of generality, we can assume that $\pi \in \mathbb{Z}_+$, by scaling.

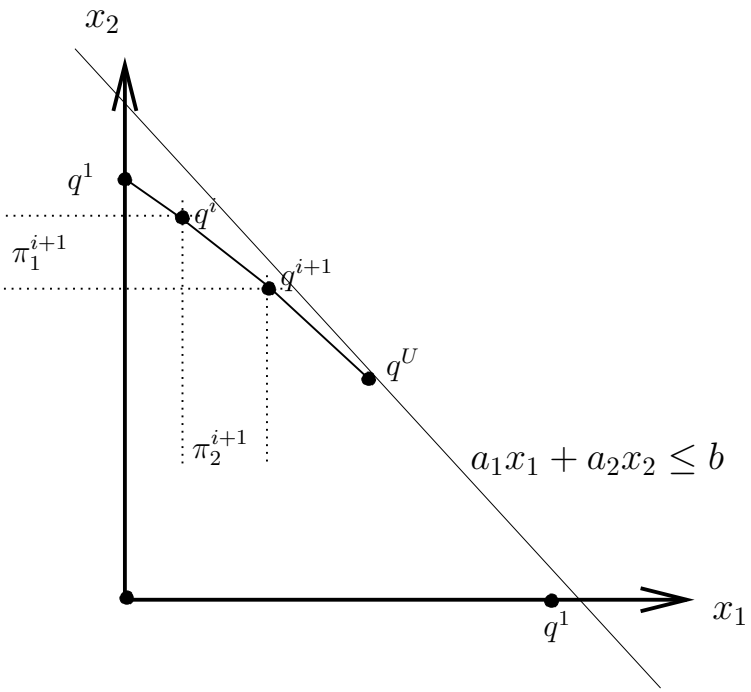
First, we present an upper bound to q_1^U, and use this to bound the number of extreme points q such that $q_1 \in [0, q_1^U]$. Counting sequentially from the extreme point on the axis $x_1 \geq 0$, such that $q_1^1 = 0, q_2^1 = u_2$, let q^i be the i^{th} extreme point. We want an upper bound to n, where $q^n = q^U$. Let $\pi_1^{i+1}x_1 + \pi_2^{i+1}x_2 \leq \pi_0^{i+1}$ be the facet described by q^i and q^{i+1}, for $i \in [1, n]$. We set $\pi_1^{n+1} = a_1$ and $\pi_2^{n+1} = a_2$; this will not define a facet if $q^U = q^D$. π_1^i/π_2^i is strictly increasing in i. For all $i \in [1, n-1]$, we scale π_1^{i+1} and π_2^{i+1} such that $q_1^{i+1} - q_1^i = \pi_2^{i+1}$, and $q_2^i - q_2^{i+1} = \pi_1^{i+1}$.

We show by contradiction that $\pi_2^{i+1} > q_1^i,\ i \in [1, n]$. Assume that this is not true. Consider $q \in \mathbb{Z}_+^2$ such that $q_1 = q_1^i - \pi_2^{i+1}$ and $q_2 = q_2^i + \pi_1^{i+1}$. Since $q_1 \geq 0$ and $a_1q_1 + a_2q_2 = a_1q_1^i - a_1\pi_2^{i+1} + a_2q_1^i + a_2\pi_1^{i+1} \leq b - a_1\pi_2^{i+1} + a_2\pi_1^{i+1} \leq b$, we have $q \in K_2^{\leq}(b)$. However, since π_1^i/π_2^i is strictly increasing in i, $\pi_1^i q_1 + \pi_2^i q_2 = \pi_0^i - \pi_1^i\pi_2^{i+1} + \pi_2^i\pi_1^{i+1} > \pi_0$, which is a contradiction. $\implies \pi_2^{i+1} > q_1^i$. For $i = n$, this proves that $q_1^n < a_2$. Trivially, $q_1^n \leq u_1$. $\implies q_1^n \leq \min\{a_2 - 1, u_1\}$.

Now, $\pi_2^{i+1} > q_1^i$ the same as $q_1^{i+1} \geq 2q_1^i + 1$, for $i \leq n-1$. Since $q_1^1 \geq 0$, we have $q_1^i \geq 2^{i-1} - 1$, for $i \in [1, n]$. Since $q_1^n \leq \gamma$, we have $2^{n-1} - 1 \leq \gamma$. Taking logarithm to base 2, and using the integrality of n, we have that $n \leq \lfloor \log(\min\{a_2, u_1 + 1\}) \rfloor + 1$.

Second, we use a similar argument for counting the number of extreme points q such that $q_1 \in [q_1^D, u_1]$. Counting sequentially from the extreme point on the axis $x_2 = 0$, such that $q_1^1 = u_1, q_2^1 = 0$, let q^i be the i^{th} extreme point and let $q^n = q^D$. It can be shown using similar arguments that $\gamma \geq u_1 - q_1^i \geq 2(u_1 - q_1^{i-1}) + 1$ for $i \in [1, n]$. Combining these result with $u_1 - q_1^1 \geq 0$, we get $u_1 - q_1^i \geq 2^{i-1} - 1$ for $i \in [1, n]$. Hence $n \leq \lfloor \log(\min\{a_2, u_1 + 1\}) \rfloor + 1$. Thus, the number of extreme points is bounded from above by $2\lfloor \log(\min\{a_2, u_1 + 1\}) \rfloor + 2$.

Figure F.1: Extreme points of $K_2^{\leq}(b)$



Repeating the arguments in this proof for the q_2 component of all extreme points of $\text{conv}(K_2^{\leq}(b))$, we bound the number of extreme points by $2\lfloor \log(\min\{a_1, u_2 + 1\}) \rfloor + 2$. □

Lemma 7.12 If p is an extreme point of $\text{conv}(M_n^{\leq}(b))$, then either $p_0 = 0$ or $p_0 = \sum_{i=1}^n a_i p_i - b$.

Proof We prove this by contradiction. Assume that there exists an extreme point $\bar{p}$ such that $\bar{p}_0 > 0$ and $\bar{p}_0 > \sum_{i=1}^n a_i \bar{p}_i - b$. Since $\bar{p} \in M_n^{\leq}(b)$, we also have $\bar{p}_i \geq 0,\ i \in [1, n]$.

Consider p^1, p^2 defined as follows. $p_0^1 = \bar{p}_0 + \epsilon$, $p_0^2 = \bar{p}_0 - \epsilon$, and $p_i^1 = p_i^2 = \bar{p}_i,\ i \in [1, n]$.

Now, $p^1 \in M_n^{\leq}(b)$ since $p_i^1 = \bar{p}_i \geq 0,\ i \in [1, n]$, $p_0^1 = \bar{p}_0 + \epsilon > 0$, and $\sum_{i=1}^n a_i p_i^1 - p_0^1 = \sum_{i=1}^n a_i \bar{p}_i - (\bar{p}_0 + \epsilon) < \sum_{i=1}^n a_i \bar{p}_i - \bar{p}_0 < b$. Similarly, $p^2 \in M_n^{\leq}(b)$ since $p_i^2 = \bar{p}_i \geq 0,\ i \in [1, n]$, $p_0^2 = \bar{p}_0 - \epsilon \geq 0$, and $\sum_{i=1}^n a_i p_i^2 - p_0^2 = \sum_{i=1}^n a_i \bar{p}_i - (\bar{p}_0 - \epsilon) < b + \epsilon \leq b$.

Since $\bar{p}$ can be written as a convex combination of p^1 and p^2, we are done. □

Theorem 7.13 Let $p \in M_n^{\leq}(b)$ and $q \in \mathbb{R}_+^n$ such that $q_i = p_i,\ i \in [1, n]$. If $p_0 = 0$, then p is an extreme point of $\text{conv}(M_n^{\leq}(b))$ if and only if q is an extreme point of $\text{conv}(K_n^{\leq}(b))$. On the other hand, if $p_0 = \sum_{i=1}^n a_i p_i - b$, then p is an extreme point of $\text{conv}(M_n^{\leq})$ if and only if q is an extreme point of $\text{conv}(K_n^{\geq}(b))$.

Proof First, we show that p is an extreme point of $\text{conv}(M_n^{\leq}(b))$ if and only if q is an extreme point of $\text{conv}(K_n^{\leq}(b))$ when $p_0 = 0$. We prove this by showing that $K_n^{\leq}(b)$ is the projection of the $y = 0$ face of $M_n^{\leq}(b)$ on the space of $x \in \mathbb{Z}$.

Let M' be the face of $M_n^{\leq}(b)$ such that $y = 0$. Thus, $M' = \{y = 0, \sum_{i=1}^n a_i x_i -$

$y \leq b, y \geq 0, x \in \mathbb{Z}^n_+\}$. By Fourier-Motzkin elimination (Dantzig and Eaves 1973), the projection of M' on $x \in \mathbb{Z}^n_+$ is $\{\sum_{i=1}^n a_i x_i \leq b, x \in \mathbb{Z}^n_+\}$, which is the same as $K^{\leq}_n(b)$.

Next, we show that p is an extreme point of $\text{conv}(M^{\leq}_n(b))$ if and only if q is an extreme point of $\text{conv}(K^{\geq}_n(b))$ when $p_0 = \sum_{i=1}^n a_i p_i - b$. We prove this by showing that $K^{\geq}_n(b)$ is the projection of the $y = \sum_{i=1}^n a_i x_i - b$ face of $M^{\leq}_n(b)$ on the space of $x \in \mathbb{Z}$.

Let M'' be the face of $M^{\leq}_n(b)$ such that $y = \sum_{i=1}^n a_i x_i - b$. Thus, $M'' = \{y = \sum_{i=1}^n a_i x_i - b, \sum_{i=1}^n a_i x_i - y \leq b, y \geq 0, x \in \mathbb{Z}^n_+\}$. Again, by Fourier-Motzkin elimination, the projection of M'' on $x \in \mathbb{Z}^n_+$ is $\{\sum_{i=1}^n a_i x_i - b \geq 0, x \in \mathbb{Z}^n_+\}$, which is $K^{\geq}_n(b)$. □

Lemma 7.16 Let $p^1, p^3 \in Z_{\leq}$ and $p^2, p^4 \in Z_{\geq}$ such that $p^1 \prec p^3$ and $p^2 \prec p^4$. Then, no non-trivial facet of $\text{conv}(M^{\leq}_n(b))$ can be defined by p^1, p^3, and some $p \in Z_{\leq}$; or by p^2, p^4, and some $p \in Z_{\geq}$.

Proof Since $p^1, p^3, p \in Z_{\leq}$, we have $p^1_0 = p^3_0 = p_0 = 0$. Since they are also affinely independent, they define the trivial facet $y = 0$. $\Longrightarrow$ We have a contradiction.

Similarly, if $p \in Z_{\geq}$, then p^2, p^4, and p define the trivial facet $y = a_1 x_1 + a_2 x_2 - b$. □

Lemma 7.17 Let $\Gamma(\bar{p}, \hat{p}) = (a_1\bar{p}_1 + a_2\bar{p}_2 - b)/(\hat{p}_1\bar{p}_1 + \hat{p}_2\bar{p}_2 - \hat{p}_0)$. Let $p^1, p^3 \in Z_{\leq}$ and $p^2, p^4 \in Z_{\geq}$ such that $p^1 \prec p^3$ and $p^2 \prec p^4$. Consider the half-space $\pi_1 x_1 + \pi_2 x_2 - y \leq \pi_0$. If it is defined by p^1, p^3, and p^2, then $\pi_i = \Gamma(p^2, \pi')\pi'_i,\ i \in [0, 2]$, where q^1, q^3 define the half-space $\pi'_1 x_1 + \pi'_2 x_2 \leq \pi'_0$. On the other hand, if it is defined by p^1, p^2, and p^4, then $\pi_i = a_i - \Gamma(p^1, \pi')\pi'_i,\ i = 1, 2$ and $\pi_0 = b - \Gamma(p^1, \pi')\pi'_0$, where q^2, q^4 define the half-space $\pi'_1 x_1 + \pi'_2 x_2 \geq \pi'_0$. Furthermore, $\Gamma \in \mathbb{Z}_+$ in either case if the hyperplane is a facet of $\text{conv}(M^{\leq}_2(b))$.

Proof Since $p^1_0 = p^3_0 = 0$, we have $\pi_1 p^1_1 + \pi_2 p^1_2 = \pi_0$, and $\pi_1 p^3_1 + \pi_2 p^3_2 = \pi_0$. From q^1 and q^3, we also have $\pi'_1 q^1_1 + \pi'_2 q^1_2 = \pi_0$, and $\pi'_1 q^3_1 + \pi'_2 q^3_2 = \pi_0$. Since these define the same system (up to a multiplicative factor), we have $\pi_i = \ell\pi'_i,\ i \in [0, 2]$. Substituting this for p^2, we get $\ell\pi'_1 p^2_1 + \ell\pi'_2 p^2_2 - p^2_0 = \ell\pi'_0$. Since $p^2_0 = a_1 p^2_1 + a_2 p^2_1 - b$, we have shown that $\ell = \Gamma(p^2, \pi')$.

Next, we prove that $(\pi'_1 p^2_1 + \pi'_2 p^2_2 - \pi'_0) = 1$ if p^1, p^2, p^3 define a facet. We have $(\pi'_1 p^2_1 + \pi'_2 p^2_2 - \pi'_0) = \pi'_1(p^2_1 - p^1_1) + \pi'_2(p^2_2 - p^1_2)$, since the point p^1 satisfies $\pi'_1 x_1 + \pi'_2 x_2 \leq \pi'_0$ at equality.

We assume (without loss of generality) that no integral point lies on the line joining p^1 and p^3. If p^1, p^2, p^3 define a facet, then there exist no integral points in the triangle defined by the vectors $p^3 - p^1$ and $p^2 - p^1$. $\Longrightarrow$ Area of the triangle defined by the integral vectors π' and $p^2 - p^1$ is exactly 0.5 (Scarf 1981). In other words, $\pi'_1(p^2_1 - p^1_1) - \pi'_2(p^2_2 - p^1_2) = 1$. This implies that $\Gamma(p^2, \pi') \in \mathbb{Z}$ if p^1, p^2, and p^3 define a facet.

Similarly, we can prove that $\pi_i = a_i - \Gamma(p^1, \pi')\pi'_i,\ i = 1, 2$, $\pi_0 = b - \Gamma(p^1, \pi')\pi'_0$, and $\Gamma(p^1, \pi') \in \mathbb{Z}$ when the half-space is defined by p^1, p^2, and p^4. □

Lemma 7.18 Let $p^1, p^3 \in Z_{\leq}$ and $p^2, p^4 \in Z_{\geq}$ such that $p^1 \prec p^3$ and $p^2 \prec p^4$. Unless p^1, p^3, p^2, and p^4 define the same hyperplane, either the line joining p^1 and p^4, or the line joining p^2 and p^3 is not a non-trivial face of $\text{conv}(M^{\leq}_2(b))$.

Proof Let the line joining q^1 and q^4 intersect the line joining q^2 and q^3 at q^*. Since q^* is a convex combination of distinct points, $0 < q^*_1 < u_1$ and $0 < q^*_2 < u_2$.

We define the points p^{13} and p^{24} to lie on the lines joining p^1,p^4, and p^2,p^4, respectively, as follows; $p_1^{14} = p_1^{23} = q_1^*$, and $p_2^{14} = p_2^{23} = q_2^*$. $p^{14}, p^{23} \in M_2^{\leq}(b)$ since they are convex combinations of elements of a convex set.

Now, if $p_0^{14} = p_0^{23}$, then p^{14} and p^{23} define the same point, say p^*. In this case, p^1, p^2, p^3, p^4 lie on the same hyper-plane, since p^4 lies on the line joining p^2 and p^* that lies on the hyper-plane defined by p^1, p^2, and p^3. This is a contradiction, by assumption.

If $p_0^{14} > p_0^{23}$, then the line joining p^1 and p^3 can not be a face. We show this by proving that p^{14} is not on the boundary of $\mathrm{conv}(M_2^{\leq}(b))$. From q^*, we have $0 < p_1^{14} < u_1$ and $0 < p_2^{14} < u_2$. Since $p_0^{23} \geq 0$, we have $p_0^{24} > (a_1 p_1^{14} + a_2 p_2^{14} - b)^+$.

If $p_0^{14} > p_0^{23}$, then we can similarly prove that the line joining p^1 and p^3 can not define a face. □

Lemma 7.19 Let $p^1, p^3 \in Z_{\leq}$ and $p^2, p^4 \in Z_{\geq}$ such that $p^1 \prec p^3$ and $p^2 \prec p^4$. The half-space $\pi_1 x_1 + \pi_2 x_2 - y \leq \pi_0$ defined by p^1, p^3, and p^2 does not contain any $p \in Z_{\leq}$ such that $p \in \mho(p^1, p^3)$. Similarly, the half-space defined by p^1, p^2, and p^4 does not contain $p \in Z_{\geq}$ such that $p \in \mho(p^2, p^4)$.

Proof Using the corresponding elements $q, q^1, q^3 \in X_{\leq}$, we have $(p_2 - p_2^1)/(p_1^1 - p_1) > (p_2^3 - p_2^1)/(p_1^1 - p_1^3)$ From the proof of Lemma 7.17, we get $\pi_1/\pi_2 = \pi_1'/\pi_2' = (p_2^3 - p_2^1)/(p_1^1 - p_1^3)$. Now, $\pi_1 p_1 + \pi_2 p_2 - p_0 = \pi_0 - \pi_1(p_1^1 - p_1) + \pi_2(p_2 - p_2^1) > \pi_0$, which proves the first part of the lemma.

Similarly, we can prove that the half-space defined by p^1, p^2, and p^4 does not contain $p \in Z_{\geq}$ such that $p \in \mho(p^2, p^4)$. □

Lemma 7.20 Let $p^1, p^3 \in Z_{\leq}$ and $p^2, p^4 \in Z_{\geq}$ such that $p^1 \preceq p^3$ and $p^2 \preceq p^4$. Then, p^1, p^3 define a facet with some $p \in Z_{\geq}$, and p^2, p^4 define a facet with some $p \in Z_{\leq}$.

Proof Since $p^1, p^3 \in Z_{\leq}$ are extreme points, they define a line that is a one-dimensional face. This line must be an intersection of two facets (two-dimensional faces). One of them is $y \geq 0$, which passes through all points in $Z_{\leq}$. The other facet must pass through some point in $Z_{\geq}$.

Similarly, we can prove that all adjacent points $p^2, p^4 Z_{\geq}$ define the trivial facet $a_1 x_1 + a_2 x_2 - y = b$ and a non-trivial facet with some $p \in Z_{\leq}$. □

Lemma 7.21 Let $p^5, p^1, p^3 \in Z_{\leq}$ and $p^6, p^2, p^4 \in Z_{\geq}$ such that $p^5 \preceq p^1 \preceq p^3$ and $p^6 \preceq p^2 \preceq p^4$. If either p^1, p^6, and p^2; or p^5, p^1, and p^2 define a facet, then so do either p^1, p^3, and p^2; or p^1, p^2 and p^4.

Proof From Lemma 7.20, the pair of adjacent extreme points p^1, p^3 must define a facet with some $p \in Z_{\geq}$. Thus, the line joining p and p^3 defines a face of $\mathrm{conv}(M_2^{\leq}(b))$.

If $p = p^2$, then p^1, p^3, and p^2 define a facet, and we are done.

Assume it is not so. Since p^1, p^2 define a facet with either p^6 or p^5, the line joining p^1 and p^2 is a non-trivial face of $\mathrm{conv}(M_2^{\leq}(b))$. From Lemma 7.18, we have a contradiction if $p \prec p^2$. In other words, p must be after p^2 in the ordered set $Z_{\geq}$.

Now, p^1, p^2 and p^1, p are non-trivial faces of $\mathrm{conv}(M_2^{\leq}(b))$. Since $p^4 \in Z_{\geq}$ is between p^2 and p, it can define a non-trivial face only with p^1; from Lemma 7.18. Furthermore, p^2 and p^4 must define a non-trivial facet with some $p' \in Z_{\leq}$. Therefore, $p' = p^1$, and we are done. □

Lemma 7.22 Let $p^5, p^1, p^3 \in Z_{\leq}$ and $p^6, p^2, p^4 \in Z_{\geq}$ such that $p^5 \preceq p^1 \preceq p^3$ and $p^6 \preceq p^2 \preceq p^4$. The half-space $\pi_1 x_1 + \pi_2 x_2 - y \leq \pi_0$ defined by p^1, p^3, and p^2 contains all $p \in Z_{\leq},\ p \neq p^1, p^3$. Furthermore, if it contains p^4 (p^6), then it contains all points $p\ (p') \in Z_{\geq}$ such that $p \succ p^4$ ($p' \prec p^6$). Similarly, the half-space defined by p^2, p^4, and p^1 contains all $p \in Z_{\geq},\ p \neq p^2, p^4$. Furthermore, if it contains p^3 (p^5), then it contains all points $p\ (p') \in Z_{\leq}$ such that $p \succ p^3$ ($p' \prec p^5$).

Proof We only prove the first part of the lemma here; the second is proved similarly.

Let $p, p' \in Z_{\leq},\ p, p' \neq p^1, p^3$ such that $p \succ p^3$ and $p' \prec p^1$. Using the corresponding elements $q, q', q^1, q^3 \in X_{\leq}$, we have $(p_2 - p_2^3)/(p_1^3 - p_1) < (p_2^3 - p_2^1)/(p_1^1 - p_1^3) < (p_2^3 - p_2')/(p_1' - p_1^3)$. From the proof of Lemma 7.17, we get $\pi_1/\pi_2 = \pi_1'/\pi_2' = (p_2^3 - p_2^1)/(p_1^1 - p_1^3)$. Now, $\pi_1 p_1 + \pi_2 p_2 = \pi_0 - \pi_1(p_1^3 - p_1) + \pi_2(p_2 - p_2^3) < \pi_0$. Similarly, $\pi_1 p_1' + \pi_2 p_2' = \pi_0 + \pi_1(p_1' - p_1^3) - \pi_2(p_2^3 - p_2') < \pi_0$.

Now, let $p, p' \in Z_{\geq}$ such that $p \succ p^4$ and $p' \prec p^6$. For any $p \in Z_{\geq}$ such that $p \succ p^4$, the half-space contains p if and only if $(a_1 - \pi_1)/(a_2 - \pi_2) \leq (p_2 - p_2^2)/(p_1^2 - p_1)$ Using the corresponding elements $q, q^2 \in X_{\geq}$, the right hand side in the above inequality increases in the ordered list $Z_{\geq}$. Thus, if the inequality is true for p^4, then it is true for all $p \in Z_{\geq}$ such that $p \succ p^4$. Similarly, it can be shown that if the half-space contains p^6, then it contains all $p' \in Z_{\geq}$ such that $p' \prec p^6$. □

Proposition 7.25 If $e \geq \beta - \rho$, then f is super-additive.

Proof We prove this by a case analysis of the possible values of d_1, d_2. We define $k_i = \lfloor (d_i - e)/\beta \rfloor$ and $r_i = r(d_i - e, \beta)\ i = 1, 2$. Furthermore, let $k_0 = \lfloor (e + \rho)/\beta \rfloor - 1$ and $r_0 = r(e + \rho, \beta)$. Since $e \geq \beta - \rho$, we have $k_0 \geq 0$. Now, $f(d_1 + d_2) = f(e + (k_1 + k_2 + k_0 + 1)\beta + r_1 + r_2 + r_0 - \rho)$. γ is equal to $\delta\beta + \rho(\alpha - \delta)$; we will use this repeatedly.

Case 1: $r_1 \leq \rho, r_2 \leq \rho$.
$f(d_1) + f(d_2) = 2c + (k_1 + k_2)\gamma + \alpha(r_1 + r_2)$.
When $r_1 + r_2 + r_0 - \rho \leq \rho$, we have $f(d_1 + d_2) = c + (k_1 + k_2 + k_0 + 1)\gamma + \alpha(r_1 + r_2 + r_0 - \rho)$. Therefore, we have $f(d_1 + d_2) - f(d_1) - f(d_2) = \gamma(k_0 + 1) + \alpha(r_0 - \rho) - c \geq (k_0 + 1)(\gamma - \delta\beta) + (\alpha - \delta)(r_0 - \rho) = (\alpha - \delta)(r_0 + k_0\rho)$. Since $r_0 \geq 0$ and $k_0 \geq 0$, f is super-additive. For larger values of $r_1 + r_2 + r_0 - \rho$, $f(d_1 + d_2)$ can only increase.

Case 2: $r_1 \leq \rho, r_2 \geq \rho$.
$f(d_1) + f(d_2) = 2c + (k_1 + k_2)\gamma + \alpha r_1 + \delta r_2 + (\alpha - \delta)\rho$.
When $r_1 + r_2 + r_0 - \rho \leq \rho$, we have $f(d_1 + d_2) = c + (k_1 + k_2 + k_0 + 1)\gamma + \alpha(r_1 + r_2 + r_0 - \rho)$. Therefore, we have $f(d_1 + d_2) - f(d_1) - f(d_2) = \gamma(k_0 + 1) + \alpha r_0 + (\alpha - \delta)r_2 + (\delta - 2\alpha)\rho - c \geq (k_0 + 1)(\gamma - \delta\beta) + (\alpha - \delta)(r_0 - 2\rho + r_2) = (\alpha - \delta)(r_0 + (k_0 - 1)\rho + r_2)$. Since $r_0 \geq 0$, $r_2 \geq \rho$, and $k_0 \geq 0$, f is super-additive. For larger values of $r_1 + r_2 + r_0 - \rho$, $f(d_1 + d_2)$ can only increase. The case $r_1 \geq \rho \geq r_2$ is proved analogously.

Case 3: $r_1 \geq \rho, r_2 \geq \rho$.
$f(d_1) + f(d_2) = 2c + (k_1 + k_2)\gamma + 2\alpha\rho + \delta(r_1 + r_2 - 2\rho)$.
When $\rho \leq r_1 + r_2 + r_0 - \rho \leq \beta$, we have $f(d_1 + d_2) = c + (k_1 + k_2 + k_0 + 1)\gamma + \alpha\rho + \delta(r_1 + r_2 + r_0 - 2\rho)$. Therefore, we have $f(d_1 + d_2) - f(d_1) - f(d_2) = \gamma(k_0 + 1) + \delta r_0 - \alpha\rho - c \geq (k_0 + 1)(\gamma - \delta\beta) - (\alpha - \delta)\rho = (\alpha - \delta)k_0\rho$. Since $k_0 \geq 0$, f is super-additive. For larger values of $r_1 + r_2 + r_0 - \rho$, $f(d_1 + d_2)$ can only increase. □

Proposition 7.26 f is super-additive if and only if $c \leq c_1$ or $c \leq c_2$.

Proof First, we prove that f is super-additive if $c \leq c_1$ or $c \leq c_2$. We prove this by a case analysis of the possible values of d_1, d_2. We define $k_i = \lfloor (d_i - e)/\beta \rfloor$ and $r_i = r(d_i - e, \beta)$ $i = 1, 2$. We have $f(d_1 + d_2) = f(e + (k_1 + k_2)\beta + r_1 + r_2 + e)$.

Case 1: $r_1 \leq \rho, r_2 \leq \rho$.

$f(d_1) + f(d_2) = 2c + (k_1 + k_2)\gamma + \alpha(r_1 + r_2)$.

When $r_1 + r_2 + e \leq \rho$, we have $f(d_1 + d_2) = c + (k_1 + k_2)\gamma + \alpha(r_1 + r_2 + e)$. Therefore, we have $f(d_1 + d_2) - f(d_1) - f(d_2) = \alpha e - c$. When $c \leq c_2$, f is super-additive since $c_2 \leq e\alpha$. When $c \leq c_1$, $\alpha e - c \geq e(\alpha - \delta) + \rho(\alpha - \delta) \geq 0$.

For larger values of $r_1 + r_2 + e$, $f(d_1 + d_2)$ can only increase.

Case 2: $r_1 \leq \rho$, $r_2 \geq \rho$.

$f(d_1) + f(d_2) = 2c + (k_1 + k_2)\gamma + \alpha(r_1 + \rho) + \delta(r_2 - \rho)$.

Now, $r_1 + r_2 + e \geq \rho$. When $r_1 + r_2 + e \leq \beta$, we have $f(d_1 + d_2) = c + (k_1 + k_2)\gamma + \alpha\rho + \delta(r_1 + r_2 + e - \rho)$. Therefore, we have $f(d_1 + d_2) - f(d_1) - f(d_2) = \delta(r_1 + e) - c - \alpha r_1$. When $c = c_2$, this is equal to $(\delta - \alpha)(r_1 + e - \beta + \rho) \geq 0$ (since $r_1 + e \leq \beta - r_2$). When $c = c_1$, this is equal to $(\alpha - \delta)(\rho - r_1) \geq 0$ (since $r_1 \leq \rho$).

For larger values of $r_1 + r_2 + e$, $f(d_1 + d_2)$ can only increase. The case $r_1 \geq \rho \geq r_2$ is proved analogously.

Case 3: $r_1 \geq \rho$, $r_2 \geq \rho$.

$f(d_1) + f(d_2) = 2c + (k_1 + k_2)\gamma + 2\alpha\rho + \delta(r_1 + r_2 - 2\rho)$.

Now, $r_1 + r_2 + e \geq \rho$. When $\rho \leq r_1 + r_2 + e \leq \beta$, we have $f(d_1 + d_2) = c + (k_1 + k_2)\gamma + \alpha\rho + \delta(r_1 + r_2 + e - \rho)$. Therefore, we have $f(d_1 + d_2) - f(d_1) - f(d_2) = \delta(\rho + e) - c - \alpha\rho$. When $c = c_2$, this is equal to $(\delta - \alpha)(e + 2\rho - \beta) \geq 0$ (since $e \leq \beta - r_1 - r_2$). When $c = c_1$, $f(d_1 + d_2) - f(d_1) - f(d_2) = 0$ (since $c_1 = \delta(\rho + e) - \alpha\rho$).

For larger values of $r_1 + r_2 + e$, $f(d_1 + d_2)$ can only increase.

Next, we prove that f is not super-additive if $c > c_1$ and $c > c_2$. We prove this by constructing d_1 and d_2 such that $f(d_1 + d_2) < f(d_1) + f(d_2)$. Let d_1,d_2 such that $r_1 = r_2 = \rho$. Now, $f(d_1) + f(d_2) = 2c + (k_1 + k_2)\gamma + 2\alpha\rho$.

When $\rho \leq e + 2\rho \leq \beta$, $f(d_1 + d_2) = c + \gamma(k_1 + k_2) + \alpha\rho + \delta(\rho + e)$. We have $f(d_1) + f(d_2) - f(d_1 + d_2) = c + \alpha\rho - \delta(\rho + e) > 0$ (since $c > c_1 \geq c_2$).

When $\beta \leq e + 2\rho$, $f(d_1 + d_2) = c + \gamma(k_1 + k_2) + \gamma + \alpha(2\rho + e - \beta)$ since $e + 2\rho < \beta + \rho$. We have $f(d_1) + f(d_2) - f(d_1 + d_2) = c - \gamma - \alpha(e - \beta) > \alpha e - (\beta - \rho)(\alpha - \delta) - \alpha(e - \beta) - \gamma = 0$. (The inequality is because $c > c_2 \geq c_1$, and the equality because $\gamma = \alpha\rho + (\beta - \rho)\delta$.) □

Proposition 7.27 g is super-additive if and only if $c \leq c_0$ or $c \leq c_3$.

Proof First, we prove that g is super-additive if $c \leq c_0$ or $c \leq c_3$. We prove this by a case analysis of the possible values of d_1, d_2. We define $k_i = \lfloor (d_i - e)/\beta \rfloor$ and $r_i = r(d_i - e, \beta)$ $i = 1, 2$. We have $g(d_1 + d_2) = g(e + (k_1 + k_2)\beta + r_1 + r_2 + e)$.

Case 1: $r_1 \leq \rho, r_2 \leq \rho$.

$g(d_1) + g(d_2) = 2c + (k_1 + k_2)\gamma + \alpha(r_1 + r_2)$.

When $r_1 + r_2 + e \leq \rho$, we have $g(d_1 + d_2) = c + (k_1 + k_2)\gamma + \alpha(r_1 + r_2 + e)$. Therefore, we have $g(d_1 + d_2) - g(d_1) - g(d_2) = \alpha e - c$. When $c \leq c_3$, g is super-additive since $c_3 \leq \alpha e$. When $c \leq c_0$, $\alpha e - c \geq (\alpha - \delta)e \geq 0$. For larger $r_1 + r_2 + e$, $g(d_1 + d_2)$ can only increase.

Case 2: $r_1 \leq \rho$, $r_2 \geq \rho$.

$g(d_1) + g(d_2) = 2c + (k_1 + k_2)\gamma + \alpha(r_1 + \rho) + \delta(r_2 - \rho)$.

Now, $r_1 + r_2 + e \geq \beta$. When $r_1 + r_2 + e \leq \beta + \rho$, we have $g(d_1 + d_2) = c + (k_1 + k_2)\gamma + \gamma + \alpha(r_1 + r_2 + e - \beta)$. Therefore, we have $g(d_1 + d_2) - g(d_1) - g(d_2) = \gamma - \delta(r_2 - \rho) - c + \alpha(r_2 + e - \beta - \rho) =$

$(\alpha-\delta)(r_2-\beta)+\alpha e-c$. (The last equality is because $\gamma=\alpha\rho+\delta(\beta-\rho)$.) When $c\le c_0$, this is equal to $(\alpha-\delta)(r_2+e-\beta)\ge 0$ (since $r_2+e\ge\beta$). When $c\le c_3$, this is equal to $(\alpha-\delta)(\beta-2\rho+r_2)\ge 0$ (since $r_2\ge\rho$).
For larger values of r_1+r_2+e, $g(d_1+d_2)$ can only increase. The case $r_1\ge\rho\ge r_2$ is proved analogously.

Case 3: $r_1\ge\rho$, $r_2\ge\rho$.
$g(d_1)+g(d_2)=2c+(k_1+k_2)\gamma+2\alpha\rho+\delta(r_1+r_2-2\rho)$.
Now, $r_1+r_2+e\ge\beta+\rho$. When $r_1+r_2+e\le 2\beta$, we have $g(d_1+d_2)=c+(k_1+k_2)\gamma+\gamma+\alpha\rho+\delta(r_1+r_2+e-\rho-\beta)$. Therefore, we have $g(d_1+d_2)-g(d_1)-g(d_2)=\gamma+\delta(e+\rho-\beta)-c-\alpha\rho=\delta e-c$. (The last equality is because $\gamma=\alpha\rho+\delta(\beta-\rho)$.) When $c\le c_0$, this is trivially true. When $c\le c_3$, $g(d_1+d_2)-g(d_1)-g(d_2)=(\alpha-\delta)(2\beta-2\rho-e)\ge 0$ (since $2\beta-e\ge r_1+r_2$).
For larger values of r_1+r_2+e, $g(d_1+d_2)$ can only increase.

Next, we prove that g is not super-additive if $c>c_0$ and $c>c_3$. We prove this by presenting d_1 and d_2 such that $g(d_1+d_2)<g(d_1)+g(d_2)$. Let $r_1=r_2=\rho$. Now, $g(d_1)+g(d_2)=2c+(k_1+k_2)\gamma+2\alpha\rho$.
When $\beta+\rho\le e+2\rho\le 2\beta$, $g(d_1+d_2)=c+\gamma(k_1+k_2)+\gamma+\alpha\rho+\delta(\rho+e-\beta)$. We have $g(d_1)+g(d_2)-g(d_1+d_2)=c+\alpha\rho-\delta(\rho+e-\beta)-\gamma=c-\delta e>0$. (The equality is because $\gamma=\alpha\rho+(\beta-\rho)\delta$ and the inequality is because $c>c_0\ge c_3$.)
When $2\beta\le e+2\rho\le 2\beta+\rho$, $g(d_1+d_2)=c+\gamma(k_1+k_2)+2\gamma+\alpha(2\rho+e-2\beta)$. We have $g(d_1)+g(d_2)-g(d_1+d_2)=c-2\gamma-\alpha(e-2\beta)=c-\alpha e+2(\alpha-\delta)(\beta-\rho)>0$. (The first equality is because $\gamma=\alpha\rho+(\beta-\rho)\delta$ and the inequality is because $c>c_3\ge c_0$.) □