Exact Methods for Nonlinear Combinatorial Optimization

Dissertation

zur
Erlangung des Grades
eines
Doktors der Naturwissenschaften

Der Fakultät für Mathematik
der Technischen Universität Dortmund
vorgelegt von

Dipl.-Math. **Frank Baumann**

aus Stolberg (Rhld.)

Dortmund 2014

Die vorliegende Arbeit wurde von der Fakultät für Mathematik der Technischen Universität Dortmund als Dissertation zur Erlangung des Grades eines Doktors der Naturwissenschaften genehmigt.

Promotionsausschuss:

Vorsitzender:	Prof. Dr. F. Kalhoff
Erster Gutachter:	Prof. Dr. C. Buchheim
Zweiter Gutachter:	Prof. Dr. P. Mutzel
Dritter Prüfer:	Prof. Dr. C. Meyer

Tag der Einreichung:	22. April 2014
Tag der Disputation:	27. August 2014

Abstract

We consider combinatorial optimization problems with nonlinear objective functions. Solution approaches for this class of problems proposed so far are either highly problem-specific or they apply generic algorithms for constrained nonlinear optimization, which often does not yield satisfactory results in practice.

Our aim is to develop, implement and experimentally evaluate exact algorithms that address the nonlinearity of the objective function and at the same time exploit the underlying combinatorial structure of the problem. To this end we follow two approaches. The first combines good polyhedral descriptions of the objective function and the feasible set in a branch and cut-algorithm. The second approach is based on Lagrangean decomposition. By decomposing the original problem into an unconstrained nonlinear problem and a linear combinatorial problem, we are able to compute strong dual bounds for the optimal value. The computation of lower bounds is then embedded into a branch and bound-algorithm. For many applications there already exist efficient algorithms for the combinatorial subproblem, thus an important aspect of this thesis is the study of the corresponding unconstrained nonlinear subproblems.

Both approaches have the advantage that they can easily be adapted to a wide range of nonlinear combinatorial problems. We devise both polyhedral and decomposition-based algorithms for submodular applications from wireless network design and portfolio optimization and evaluate their performance experimentally. Exploiting the equivalence between unconstrained binary quadratic optimization and the maximum cut problem gives rise to a branch and cut-algorithm for quadratic combinatorial problems which we use to compute optimal layouts of tanglegrams, an application from computational biology. Additionally we study the effect of quadratic reformulation of linear constraints, both theoretically and experimentally. The last class of nonlinear combinatorial problems we consider are two-scenario problems. Here we propose a new technique to compute lower bounds in the unconstrained subproblem of the decomposition. Our computational study of the two-scenario minimum spanning tree problem shows that the new Lagrangean decomposition-based algorithm is able to solve significantly larger instances than the standard linearization approach.

Zusammenfassung

Diese Arbeit beschäftigt sich mit der exakten Lösung kombinatorischer Optimierungsprobleme mit nichtlinearen Zielfunktionen. Bisher werden für diese Art von Problemen meist entweder hochgradig problemspezifische Algorithmen entwickelt, oder es kommen Löser für allgemeine beschränkte nichtlineare Probleme zum Einsatz, was in der Praxis aber oft nicht zu befriedigenden Ergebnissen führt.

Ziel dieser Arbeit ist daher die Entwicklung, Implementierung und experimentelle Evaluation exakter Algorithmen, die sowohl die Nichtlinearität der Zielfunktion als auch die zugrunde liegende kombinatorische Struktur des Problems ausnutzen. Dazu verfolgen wir zwei Ansätze. Im ersten werden gute polyedrische Beschreibungen der Zielfunktion und der kombinatorischen Struktur in einem Branch-and-Cut Algorithmus kombiniert. Der zweite basiert auf Lagrange-Dekomposition. Hier wird das Ausgangsproblem in ein unbeschränktes nichtlineares und ein lineares kombinatorisches Problem zerlegt. Das erlaubt die Berechnung starker dualer Schranken in einem Branch-and-Bound Algorithmus. Für viele Anwendungen sind bereits effiziente Verfahren zur Lösung des kombinatorischen Teilproblems bekannt. Ein wichtiger Beitrag dieser Arbeit besteht daher in der Untersuchung der auftretenden unbeschränkten nichtlinearen Teilprobleme.

Beide Ansätze haben den Vorteil, dass sie sich ohne großen Aufwand auf eine Vielzahl nichtlinearer kombinatorischer Problemstellungen anwenden lassen. Wir betrachten drei Klassen von nichtlinearen kombinatorischen Optimierungsproblemen. Im Falle einer submodularen Zielfunktion sind beide oben beschriebenen Ansätze anwendbar. Wir stellen exakte Algorithmen für Anwendungen aus dem Bereich der mobilen Datenübertragung und der Portfoliooptimierung vor und vergleichen sie experimentell mit Standardverfahren. Bei quadratischen Zielfunktionen ergibt sich aus der Äquivalenz von unbeschränkter binärer quadratischer Optimierung und der Berechnung eines maximalen Schnittes in einem Graphen eine gute polyedrische Charakterisierung der Zielfunktion, die in einem Branch-and-Cut Algorithmus verwendet werden kann. Damit lassen sich Tanglegrams berechnen, die u.a. in der Bioinformatik Anwendung finden. Desweiteren untersuchen wir die Auswirkungen, die verschiedene quadratische Reformulierungen von linearen Nebenbedingungen auf die Güte der dualen Schranken im Branch-and-Cut Algorithmus haben. Als letzte Klasse werden Zwei-Szenario-Probleme behandelt. Wir stellen eine neue Technik zur Berechnung unterer Schranken für das unbeschränkte Teilproblem in der Lagrange-Dekomposition vor. Eine experimentelle Studie anhand des minimalen Spannbaumproblems mit zwei Szenarien zeigt, dass sich mit dem neuen dekompositionsbasierten Algorithmus deutlich größere Instanzen lösen lassen als mit dem auf ganzzahliger Programmierung basierenden Standardansatz.

Teilpublikationen

Teilergebnisse der vorliegenden Arbeit sind bereits in den folgenden Publikationen veröffentlicht worden:

Frank Baumann, Christoph Buchheim, and Anna Ilyina. Lagrangean decomposition for mean-variance combinatorial optimization. *Lecture Notes in Computer Science*, 2014. ISCO 2014 – International Symposium on Combinatorial Optimization, to appear.

Frank Baumann, Sebastian Berckey, and Christoph Buchheim. Exact algorithms for combinatorial optimization problems with submodular objective functions. In Michael Jünger and Gerhard Reinelt, editors, *Facets of Combinatorial Optimization*, pages 271–294. Springer Berlin Heidelberg, 2013.

Frank Baumann and Christoph Buchheim. Submodular formulations for range assignment problems. *Electronic Notes in Discrete Mathematics*, 36(0):239–246, 2010. ISCO 2010 – International Symposium on Combinatorial Optimization.

Frank Baumann, Christoph Buchheim, and Frauke Liers. Exact bipartite crossing minimization under tree constraints. In Paola Festa, editor, *Experimental Algorithms*, volume 6049 of *Lecture Notes in Computer Science*, pages 118–128. Springer Berlin Heidelberg, 2010.

Acknowledgements

I would like to express my thanks to everyone who supported me throughout my work on this thesis. I am particularly indebted to my supervisor Prof. Christoph Buchheim for his scientific guidance and the pleasant and stimulating working atmosphere he created in his group.

I am very grateful to Frauke Liers for introducing me to tanglegrams and the subsequent collaboration, which led to Chapter 5 of this thesis.

Many thanks go to my colleagues in Cologne, Dortmund and elsewhere for the many constructive and helpful discussions, especially to Viktor Bindewald, Anja Fischer, Anna Ilyina, Laura Klein, Jannis Kurtz, Sara Mattia, Dennis Michaels, Maribel Montenegro, Marianna De Santis, Emiliano Traversi and Long Trieu, and to Sabine Willrich for perfectly taking care of all administrative issues.

Part of my work was funded by the DFG as part of the project *Simple and Fast Implementation of Exact Optimization Algorithms with SCIL* in the *Priority Programme 1307 Algorithm Engineering* and the project *Lenkung des Güterflusses in durch Gateways gekoppelten Logistik-Service-Netzwerken mittels quadratischer Optimierung*, which is gratefully acknowledged.

Finally, I would like to thank my family for their invaluable support and encouragement.

Contents

Introduction

Combinatorial optimization deals with finding an optimal solution for problems with finite sets of feasible solutions. Often these problems are stated in graph-theoretic terms and solution methods include techniques from algorithm theory as well as polyhedral combinatorics and linear programming. For many applications that can be modeled as combinatorial optimization problems efficient algorithms are known, for others the best known algorithms have an exponential theoretical complexity.

Consider for example a task frequently encountered in the design of networks, e.g. for telecommunication. You are given a set of nodes with fixed coordinates in the plane and your job is to decide between which pairs of nodes connections are established by laying subterranean cables. In the end the network must be connected, i.e. it must be possible to reach any given node from any given starting point via the chosen links (see Figure 1 (left)). Establishing a connection is associated with costs, for example proportional to the length of the link, and the total cost of a network is given by the sum of the costs of the established connections. Which links do you choose such that the total cost is minimal? This problem is known as the minimum spanning tree problem. It is well-studied and several efficient and surprisingly simple solution algorithms are known; the earliest was proposed by Boruvka [17] in 1926.

Now imagine the task is slightly different. Nodes are not linked by laying cables in the ground but wirelessly. A connection between two nodes a and b can only be established if the signal transmitted from a is strong enough to reach b and vice versa. The goal as before is a connected network, but now your task is to choose which connections to make and to set appropriate transmission powers for the individual nodes (see Figure 1 (right)). The cost of the network in this case is determined by its total transmission power. This variant of the minimum spanning tree problem is called range assignment problem. Although both problems seem to be very similar, the second is much harder to solve than the first. In fact, no polynomial-time algorithm for the range assignment problem is known and only very small instances can be solved in practice.

Another example of an application where a slight variation in the problem statement increases its difficulty significantly is the knapsack problem. You are given

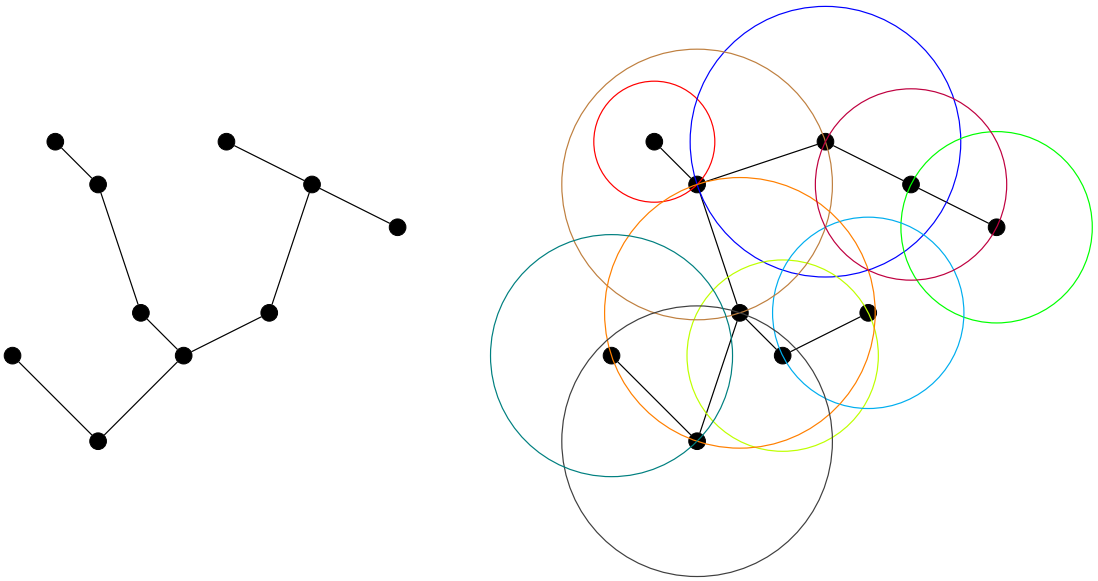

Figure 1: Optimal wired (left) and wireless (right) networks for the same set of nodes. The colored circles on the right indicate the transmission ranges of the nodes in the wireless network.

a set of items characterized by a weight and a profit and a knapsack with limited capacity. An optimal solution of the knapsack problem is a subset of the items that fits into the knapsack and gives maximum profit. The knapsack problem is closely related to a problem from portfolio theory, the so-called risk-averse capital budgeting problem. Here an investor has to choose from a set of possible investments. His budget is limited and in contrast to the knapsack problem the expected profits are not known exactly in advance. Instead they are characterized by an expected return value and a variance. The knapsack problem can be solved in pseudo-polynomial time. The exact complexity of the risk-averse capital budgeting problem is unknown, but in practice it is much harder than the knapsack problem.

In both examples given above the two problems have the same combinatorial structure, the increase in the complexity is caused by the change in the objective function. In the minimum spanning tree problem and the knapsack problem it is linear, whereas in the range assignment problem and the risk-averse capital budgeting problem it is nonlinear. Nonlinear combinatorial optimization problems are commonly solved either with highly application-specific algorithms or with generic solvers for constrained nonlinear optimization. The first approach has the advantage that the underlying combinatorial structure is exploited, but the resulting algorithms are not easily adaptable to other nonlinear applications. The second approach is more versatile but often focuses on the nonlinear aspect of the problem and disregards the combinatorial structure.

In this thesis we study exact solution techniques for nonlinear combinatorial optimization problems that address the nonlinearity of the objective function and at the same time exploit the underlying combinatorial structure of the problem.

The main idea of our approach is to treat the nonlinear objective function and the part of the problem formulation that models the set of feasible solutions independently. This allows us to combine solution techniques from unconstrained binary nonlinear optimization and linear combinatorial optimization to compute dual bounds for the original problem. Embedding the computation of dual bounds in a branch and bound-algorithm yields an algorithmic framework that can be easily customized to solve a wide range of nonlinear combinatorial applications.

For this approach to give good results in practice it is essential that the occurring subproblems can be solved to optimality quickly, or, when this is not possible, strong dual bounds can be computed efficiently. As exemplified above, in many cases the linear variant of an application is well-studied and we can use existing algorithms and polyhedral characterizations to solve the combinatorial subproblem. In contrast, the corresponding unconstrained nonlinear subproblems have often not been studied intensively so far. Therefore, we investigate several classes of nonlinear objective functions and propose efficient algorithms for some special cases.

Outline

This thesis is divided into three parts. Part I deals with general techniques for the exact solution of three classes of nonlinear combinatorial optimization problems. We start by giving the necessary basic definitions and notations used in the following chapters. In Chapter 2 we discuss the polyhedral approach to binary quadratic optimization that exploits the equivalence between unconstrained binary quadratic optimization and the MaxCut problem [28] and study the effectiveness of quadratic reformulation of linear constraints in this context. We propose a reformulation technique for assignment constraints that leads to stronger relaxations without increasing the number of variables in the problem formulation.

In Chapter 3 we study combinatorial optimization problems with submodular objective functions. We propose two exact algorithms. The first is a branch and cut-algorithm based on a polyhedral description of the convex hull of feasible points of the unrestricted problem by Edmonds [36]. The second is again a branch and bound-algorithm, but here lower bounds are computed by applying Lagrangean decomposition to the original formulation. This yields two subproblems, an unrestricted submodular minimization problem and a linear combinatorial problem.

Chapter 4 deals with two-scenario optimization. We first study the problem of

minimizing the maximum of two linear functions over bounded integer variables and propose a bounding technique that applies a transformation to two fractional knapsack problems. This approach leads to an exact branch and bound-algorithm for unconstrained two-scenario optimization problems. In the presence of additional combinatorial constraints this approach can still be used to compute lower bounds. To this end we again apply Lagrangean decomposition, as in Chapter 3. This yields an exact branch and bound-algorithm that for certain classes of combinatorial two-scenario problems does not require solving linear programs.

In Part II we use the general techniques proposed in the first part to devise exact algorithms for specific nonlinear applications. We start in Chapter 5 with computing optimal layouts of tanglegrams, which are for example used in computational biology to visualize the co-evolution of two species. This problem can be modeled as a quadratic linear ordering problem with additional constraints. In an experimental study we solve random and realistic instances to evaluate the effect of MaxCut-separation and quadratic reformulation. The experimental study is continued in Chapter 6 on the quadratic variants of the minimum spanning tree problem and the perfect matching problem.

The range assignment and mean risk optimization problems studied in Chapters 7 and 8 are generalizations of classic combinatorial problems, where the objective function is submodular instead of linear. Range assignment problems occur in wireless network design, where the overall costs are not determined by the total lengths of the established links but by the transmission power needed to establish a given network topology [122], as illustrated above. Although such problems have been studied intensively in recent years, the algorithms proposed in the literature so far do not directly exploit the submodularity of the objective function [98]. We devise a fast algorithm for the minimization of the submodular subproblem in the Lagrangean decomposition approach and evaluate the performance of the two algorithms proposed in Chapter 3 experimentally for three network topologies.

Mean risk optimization problems are used in finance to compute optimal investment strategies [24]. They are commonly solved as second-order cone problems [10]. We model the risk-averse capital budgeting problem, which served as our second example in the introduction, as a knapsack problem with a submodular objective function and compare the performance of the new approaches to the standard method.

In the last chapter of Part II we present an extensive experimental study of the algorithms for unconstrained and combinatorial two-scenario optimization presented in Chapter 4. For the combinatorial case we consider the two-scenario minimum spanning tree problem, which has applications in the design of robust telecommunication networks [77].

Part III gives a brief overview of the optimization library SCIL. For this thesis it was extended by the polyhedral methods for submodular and quadratic optimiza-

tion problems presented in Chapters 2 and 3 and used for the implementation of the branch and cut-algorithms in Part II.

We conclude this thesis with a summary of our results and a brief discussion of possible extensions and future work.

Part I

Methods

Chapter 1

Preliminaries

In this chapter we first fix some notation and give basic definitions that will be used throughout the remaining chapters. Then we give the basic theoretical background for later chapters. Notations and definitions follow the text books by Nemhauser and Wolsey [100], Wolsey [123] and Korte and Vygen [76].

1.1 Basic Definitions

Sets Sets are denoted by capital letters. For sake of legibility we write $A \cup k$ instead of $A \cup \{k\}$ for the union of a set A and a single element k.

Graphs In the following, $G = (V, E)$ denotes an *undirected graph*. $V = \{1, \ldots, n\}$ is a finite set called the set of *nodes* and the set of *edges* E consists of two-element unordered subsets of V. An edge $e = (u, v) \in E$ is said to connect nodes u and v and it is *incident* to both u and v. The nodes u an v are sometimes called the endpoints of e. Two edges with a common node are *adjacent* and an edge (u, u) that connects u to itself is called a *loop*. Graphs without loops are called *simple*. In this thesis we will only consider simple graphs. Note that in our definition of graph two nodes are connected by at most one edge, i.e. they do not have multiple edges. If the graph is *directed*, we write $G = (V, A)$. The arc set A consists of ordered pairs of nodes. The first node in the ordered pair defining an arc a is called the *tail* of a, the second the *head* of a. In a *weighted graph* – undirected or directed – each edge (arc) is assigned a weight by a *weight function* $c\colon E \longrightarrow \mathbb{Q}$ ($c\colon E \longrightarrow \mathbb{Q}$). These graphs are denoted by $G = (V, E, c)$ and $G = (V, A, c)$, respectively.

For a set $U \subseteq V$, $E(U) = \{(i, j) \mid (i, j) \in E, i \in U, j \in U\}$ is the set of edges with both endpoints in U. A subgraph G' of a graph G consists of a subset $V' \subseteq V$ of the nodes of G and a subset $E' \subseteq E(V')$ of the edges with both endpoints in V'. G' is called a *spanning subgraph* if $V' = V$.

Paths A sequence of nodes $(v_0, \ldots, v_k)$, $k \in \mathbb{N}$ of an undirected graph is called a *walk*, if $(v_i, v_{i+1}) \in E$ for all $i \in \{0, \ldots, k-1\}$. If each node occurs exactly once, the walk is called a *path* in G. A graph that contains a walk between each pair of nodes is called *connected*.

Walks and paths can also be expressed by a sequence of edges $(e_0, \ldots, e_{k-1})$. By adding the edge $e_k = (v_k, v_0)$ we obtain a *closed walk* w. If $(e_0, \cdots, e_{k-1})$ is a path containing at least two edges, w is called a *cycle*.

Trees A *tree* is a connected undirected graph without cycles. An *arborescence* is an acyclic directed graph (DAG) in which all arcs point away from a specified root node. This is equivalent to the property that for each vertex $v \in V \setminus \{v\}$ there exists a directed path from r to v. In a *Steiner arborescence* only a subset $T \subseteq V \setminus \{v\}$, called *terminals*, must be reachable via directed paths from the root node. These paths can include the *Steiner nodes* $V \setminus (T \cup \{r\})$.

Optimization problems and relaxations Given a function $f\colon \mathbb{R}^n \longrightarrow \mathbb{R}$ and a set $A \subseteq \mathbb{R}^n$, a *general optimization problem* (P) consists of finding a minimizer or maximizer of f in A. In the following we limit ourselves to discussing minimization problems and write (P) as

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & x \in A \subseteq \mathbb{R}^n\,, \end{array}$$

or shorter as

$$\min\{f(x) \mid x \in A\}\,.$$

The function f is called the *objective function* of (P) and A its *set of feasible solutions* or *feasible region*. If $A = \emptyset$, (P) is infeasible. If (P) is feasible, the minimizer of f in A is denoted by $x^\star$ and the value $f(x^\star)$ is called the *optimum value* of (P). In a slight abuse of notation, we sometimes write

$$z = \min\{f(x) \mid x \in A\}$$

to refer to the optimization problem and its optimum value z.

Given a second optimization problem

$$\min\{g(x) \mid x \in B\}\,, \tag{Q}$$

(Q) is a *relaxation* of (P) if $A \subseteq B$ and $f(x) \geq g(x)$ for each $x \in A$.

The objective function value of any feasible solution of (P) defines an *upper* or *primal bound* on the optimum value of (P), while any optimum value of a relaxation defines a *lower* or *dual bound*. For maximization problems upper bounds are dual bounds, while lower bounds are primal bounds.

In a *combinatorial optimization problem* the set of feasible solutions is finite. Combinatorial optimization problems are a special case of so-called *mixed-integer*

programs (MIPs), where some or all of the variables are required to take only integer values. They are often used to model optimization problems on graphs. Say you want to compute a minimum-cost spanning tree in an undirected graph $G = (V, E, c)$. The first step is to associate each edge $e \in E$ of the graph with a binary variable x_e. Then the set of feasible solutions, in this case the set of spanning trees in G, is modeled by a set of equations and inequalities in these variables. The result is a *binary optimization* problem. When we associate binary variables with elements of some set D, we often denote the sum of the variables associated with D as $x(D)$ instead of $\sum_{i \in D} x_i$.

If the equations and inequalities used are linear and c is a linear function, the problem is called an *integer linear program* (ILP) and generally written as

$$\begin{array}{lrcl} \min & c^\top x & & \\ \text{s.t.} & A_1 x & = & b_1 \\ & A_2 x & \leq & b_2 \\ & x & \in & \{0,1\}^n . \end{array}$$

Definition 1.1. For two optimization problems

$$\begin{array}{lrcl} \min & f(x) & & \\ \text{s.t.} & x & \in & X \subseteq \mathbb{R}^n \end{array} \tag{1.1}$$

and

$$\begin{array}{lrcl} \min & g(y) & & \\ \text{s.t.} & y & \in & Y \subseteq \mathbb{R}^m \end{array} \tag{1.2}$$

denote by $X^\star \subseteq X$ and $Y^\star \subseteq Y$ the sets of optimal solutions of (1.1) and (1.2), respectively.

Problems (1.1) and (1.2) are *equivalent*, if

1. there exists a bijective function $h\colon X^\star \longrightarrow Y^\star$, i.e. for every $y^\star \in Y^\star$ there exists a unique $x^\star \in X^\star$ with $y^\star = h(x^\star)$, and
2. $g(h(x^\star)) = f(x^\star)$ for all $x^\star \in X^\star$, i.e. equivalent optimal solutions have the same objective value.

The two problems are called *isomorphic*, if

1. there exists a bijective function $h\colon X \longrightarrow Y$, i.e. for every $y \in Y$ there exists a unique $x \in X$ with $y = h(x)$, and
2. $g(h(x)) = f(x)$ for all $x \in X$, i.e. equivalent feasible solutions have the same objective value.

Obviously, when two optimization problems are isomorphic, they are also equivalent.

Definition 1.2. Given a nonlinear optimization problem

$$\begin{array}{rcl} \min & f(x) & \\ \text{s.t.} & x & \in \quad X \subseteq \mathbb{R}^n\,, \end{array} \tag{P_1}$$

the problem

$$\begin{array}{rcl} \min & g(y) & \\ \text{s.t.} & y & \in \quad Y \subseteq \mathbb{R}^m \end{array} \tag{P_2}$$

is a *linearization* of (P_1), if (P_1) and (P_2) are equivalent and (P_2) is a linear optimization problem.

Polyhedra A *polyhedron P in $\mathbb{R}^n$* is a set of points that can be characterized by a finite number of linear inequalities: $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$. A bounded polyhedron is called a *polytope*. Given a nonempty polyhedron P and a nonzero vector $c \in \mathbb{R}^n$ for which $\delta := \max\{c^\top x \mid x \in P\}$ is finite, the set $\{x \in \mathbb{R}^n \mid c^\top x = \delta\}$ is called a *supporting hyperplane* of P. A *face* of P is the intersection of P with one of its supporting hyperplanes or P itself. Faces of maximal dimension are called *facets*, where the dimension of a set is defined as the dimension of the smallest affine space that contains the set.

A linear inequality $a^\top x \leq \delta$ is called *valid* for P, if it holds for all $x \in P$. If additionally $\{x \in P \mid a^\top x = \delta\}$ is a facet of P, the inequality is called *facet-defining*.

The *convex hull* $\operatorname{conv}(X)$ of a set X is the set of all convex combinations of points in X, i.e. the set of all points $\bar{x}$ that can be expressed as

$$\bar{x} = \sum_{i=1}^{|X|} \lambda_i x_i$$

with appropriately chosen multipliers $\lambda_1, \ldots, \lambda_{|X|} \geq 0$ with $\sum_{i=1}^{|X|} \lambda_i = 1$.

Definition 1.3. Given a set of linear inequalities $Ax \leq b$ and binarity constraints on some of the variables, relax the binarity constraints to box constraints, such that $0 \leq x \leq 1$. The set of points which are feasible for the resulting problem is called the *polytope corresponding to $Ax \leq b$*.

1.2 Lagrangean Relaxation

Consider a mixed-integer linear optimization problem of the form

$$\begin{array}{lrcll} \min\limits_{x\geq 0} & c^\top x & & & \\ \text{s.t.} & Ax & \leq & a & \\ & Bx & \leq & b & \\ & x_i & \in & \mathbb{Z} & \forall i \in I\,, \end{array} \tag{P}$$

with $A \in \mathbb{R}^{p\times n}$, $B \in \mathbb{R}^{q\times n}$, $c \in R^n$, $a \in \mathbb{R}^p$, $b \in \mathbb{R}^q$, and $p, q, n \in \mathbb{N}_+$. I is a subset of the variable indices. It is assumed that the constraints are divided into two sets such that solving problem (P) without constraints $Ax \leq a$ is significantly easier than solving the complete problem (P). The basic idea of Lagrangean relaxation is to treat the complicating constraints by moving them to the objective function. Violation of these constraints is penalized by so-called *Lagrangean multipliers* λ. The resulting problem, the *Lagrangean relaxation* with respect to $Ax \leq a$ of (P), reads

$$\begin{array}{lrcll} \min\limits_{x\geq 0} & c^\top x + \lambda^\top (Ax - a) & & & \\ \text{s.t.} & Bx & \leq & b & \\ & x_i & \in & \mathbb{Z} & \forall i \in I\,, \end{array} \tag{LR_λ}$$

with $\lambda \in \mathbb{R}^p_{\geq 0}$. For each feasible value of λ, the optimum value of (LR_λ) gives a dual bound of problem (P). The problem of determining values for λ that give the best possible dual bound z is called the *Lagrangean dual problem*:

$$\begin{array}{rlrcll} z = \max\limits_{\lambda\in\mathbb{R}^p_{\geq 0}} \min\limits_{x\geq 0} & c^\top x + \lambda^\top (Ax - a) & & & \\ \text{s.t.} & Bx & \leq & b & \\ & x_i & \in & \mathbb{Z} & \forall i \in I \end{array} \tag{D}$$

Denote by $(P^\star)$ the following relaxation of (P):

$$\begin{array}{lrcl} \min\limits_{x} & c^\top x & & \\ \text{s.t.} & Ax & \leq & a \\ & x & \in & \text{conv}\{x \geq 0 \mid Bx \leq b, x_i \in \mathbb{Z} \forall i \in I\} \end{array} \tag{$P^\star$}$$

The next theorem characterizes the relation between the relaxations (LR_λ), $(P^\star)$ and the LP relaxation $\bar{P}$ of (P).

Theorem 1.1 (Geoffrion [50])**.** *Denote by $v(\cdot)$ the optimum solution value of a problem.*

1. *For all $\lambda \geq 0$ (LR_λ) is a relaxation of* (P)*:*

$$F(LR_\lambda) \supseteq F(\text{P}) \text{ and } v(LR_\lambda) \leq v(\text{P}) \quad \forall \lambda \geq 0$$

($P^\star$) *is at least as tight as the LP relaxation:*

$$F(\bar{P}) \supseteq F(P^\star) \supseteq F(\mathrm{P}) \text{ and } v(\bar{P}) \leq v(P^\star) \leq v(\mathrm{P})$$

2. *The relaxation* ($P^\star$) *gives the same dual bound as the Lagrangean dual:*

$$z = v(P^\star)$$

If the optimal value of the relaxation (PR_λ) does not change when the integrality conditions on the variables are dropped, (PR_λ) is said to have the *integrality property*. In this case the value of the Lagrangean dual is the same as the value of the LP-relaxation ($\bar{P}$).

By Theorem 1.1 the value of the Lagrangean dual is always as least as good as the value of the LP-relaxation, but it can be better if (PR_λ) does not have the integrality property. This means that it is desirable to choose the constraints to be relaxed such that the resulting Lagrangean relaxation does *not* have the integrality property.

Theorem 1.2. *Let* ($\bar{P}$) *be feasible and* (PR_λ) *have the integrality property. Then* ($P^\star$) *is feasible and*

$$v(\bar{P}) = z\,.$$

An optimal solution of the Lagrangean relaxation LR_λ in general is not optimal for the original problem (P), unless it satisfies the following optimality conditions.

Theorem 1.3 (Geoffrion [50])**.** *Let* $x^\star$ *be an optimal solution of the Lagrangean relaxation* (LR_λ) *for a given* λ. $x^\star$ *is optimal for* (P) *if*

1. $Ax \leq a$
2. $\lambda^\top(Ax - a) = 0$.

1.2.1 Lagrangean Decomposition

Lagrangean decomposition can be considered a Lagrangean relaxation with respect to a set of artificial constraints. Its aim is to decompose the problem into auxiliary problems that can be easily computed. Starting as before from the problem

$$\begin{array}{lrcll} \min\limits_{x\geq 0} & c^\top x & & & \\ \text{s.t.} & Ax & \leq & a & \\ & Bx & \leq & b & \\ & x_i & \in & \mathbb{Z} & \forall i \in I\,, \end{array} \tag{P}$$

we introduce new variables $y \geq 0$ and artificial linking constraints and express the second set of constraints in the new variables:

$$\begin{array}{lrcl} \min\limits_{x\geq 0} & c^\top x & & \\ \text{s.t.} & Ax & \leq & a \\ & By & \leq & b \\ & x & = & y \\ & x_i & \in & \mathbb{Z} \quad \forall i \in I\,, \end{array}$$

Applying Lagrangean relaxation to the linking equations gives

$$\begin{array}{lrcl} \min\limits_{x\geq 0} & c^\top x + \lambda^\top (x-y) & & \\ \text{s.t.} & Ay & \leq & a \\ & Bx & \leq & b \\ & x_i & \in & \mathbb{Z} \quad \forall i \in I\,, \end{array}$$

Since the two sets of constraints are now independent of each other the problem decomposes into

$$\begin{array}{lrcl} \min\limits_{x\geq 0} & c^\top x + \lambda^\top x & & \\ \text{s.t.} & Bx & \leq & b \\ & x_i & \in & \mathbb{Z} \quad \forall i \in I \end{array} \quad + \quad \begin{array}{lrcl} \min\limits_{y\geq 0} & -\lambda^\top y & & \\ \text{s.t.} & Ay & \leq & a \end{array} \qquad (LD_\lambda)$$

Guignard and Kim [55] showed that the bound provided by the Lagrangean decomposition is at least as good as the bound obtained from the Lagrangean relaxation of one of the two sets of constraints.

Theorem 1.4. *Let $\lambda^\star$ be an optimal multiplier for $\max\limits_{\lambda\geq 0} v(LR_\lambda)$, $\mu^\star = \lambda^\star A$ and $(x^\star, y^\star)$ optimal for $(LD_{\mu^\star})$, i.e. optimal for*

$$\min_{x\geq 0}\{c^\top x + (a-\mu^\star)^\top x \mid Bx \leq b, x_i \in \mathbb{Z} \quad \forall i \in I\}$$

and

$$\min_{y\geq 0}\{-{\mu^\star}^\top y \mid Ay \leq a\}\,.$$

Then

1. $v(LD_{\mu^\star}) - v(LR_{\lambda^\star}) = (\lambda^\star)^\top (a - Ay^\star)$
2. $\max\limits_{\mu\in\mathbb{R}} v(LD_\mu) \geq \max\limits_{\lambda\geq 0} v(LR_\lambda)$

The proof is straight forward.

Proof. Let $\lambda^\star$ be an optimal multiplier for $\max_{\lambda \geq 0} v(LR_\lambda)$ and $\mu^\star = A^\top \lambda^\star$. Choose $(x^\star, y^\star)$ such that it is an optimal solution of $(LD_{\mu^\star})$. Then

$$\begin{aligned}
v(LD_{\mu^\star}) &= \min_{x \geq 0}\{c^\top x + (\mu^\star)^\top x \mid Bx \leq b, x_i \in \mathbb{Z} \quad \forall i \in I\} + \min_{y \geq 0}\{(-\mu^\star)^\top y \mid Ay \leq a\} \\
&= \min_{x \geq 0}\{c^\top x + ((\lambda^\star)^\top A)x \mid Bx \leq b, x_i \in \mathbb{Z} \quad \forall i \in I\} + \min_{y \geq 0}\{((-\lambda^\star)^\top A)y \mid Ay \leq a\} \\
&= c^\top x^\star + (\lambda^\star)^\top A x^\star + ((-\lambda^\star)^\top A)y^\star \qquad ((x^\star, y^\star) \text{ is optimal for } (LD_{\mu^\star})) \\
&= \big(c^\top x^\star + (\lambda^\star)^\top A)x^\star - (\lambda^\star)^\top a\big) + ((\lambda^\star)^\top a) - (\lambda^\star)^\top A y \\
&= \big(c^\top x^\star + (\lambda^\star)^\top (Ax^\star - a)\big) + (\lambda^\star)^\top (a - Ay^\star) \\
&= v(LR_{\lambda^\star}) + (\lambda^\star)^\top (a - Ay^\star)
\end{aligned}$$

The last equality holds, because, by the choice of μ as $\mu^\star = A^\top \lambda^\star$ the first part of $(LD_{\mu^\star})$ and $(LR_{\lambda^\star})$ differ only by a constant term in the objective function. Since $(x^\star, y^\star)$ is optimal for $(LD_{\mu^\star})$, $x^\star$ is also optimal for $(LR_{\lambda^\star})$.

This proves the first part of the theorem. Since both $\lambda^\star$ and $a - Ay^\star$ are non-negative, also the second part follows. □

Concluding the preliminaries, we will give a short description the branch and bound- and branch and cut-approaches for (mixed-)integer programs. These two techniques will be used throughout this thesis to compute optimal solutions of nonlinear optimization problems. We will limit the exposition in the following two sections to the basic concepts; details concerning the adaptation of the algorithms to specific problem types and applications will be discussed in later chapters. For a more detailed introduction to branch and bound- and branch and cut-algorithms, see [75] and [94]. Computational issues are discussed in [90].

1.3 Branch and Bound

Branch and bound is a technique to compute optimal solutions of optimization problems. Although it was originally proposed by Land and Doig [79] as an algorithm for MIPs, it is applicable whenever bounds on the optimal value of the optimization problem can be computed. As the name suggests, a branch and bound-algorithm has two main components, a branching procedure and a bounding procedure. The bounding procedure optimizes an objective function over a given feasible region and the branching procedure decomposes a given feasible region into two or more smaller sets. The main idea of the branch and bound-algorithm is to break up the original problem into a series of smaller *subproblems* which are easier to solve. The information obtained from the subproblems is then used to determine an optimum solution of the original problem. This approach is motivated by the following

Observation 1.5 (Wolsey [123])**.** *Consider the problem* $z = \min\{f(x) \mid x \in A\}$. *Let* $A = A_1 \cup \cdots \cup A_k$ *be a decomposition of* A *into smaller sets, and let* $z^k = \min\{f(x) \mid x \in A_k\}$ *for* $k = 1, \ldots, k$. *Then* $z = \min_k z^k$.

The branch and bound-approach has the advantage that parts of the feasible region which cannot contain a minimizer can be identified, which reduces the number of subproblems that have to be inspected:

Observation 1.6. *Consider an optimization problem*

$$\min\{f(x) \mid x \in A\} \tag{P}$$

and a relaxation

$$\min\{g(x) \mid x \in B \supseteq A\} \tag{Q}$$

of (P). *Let* $B_1, \ldots, B_k$ *such that*

$$\bigcup_{i=1,\ldots,k} B_i \supseteq B\,.$$

If for any $i \in \{1, \ldots, k\}$ *a dual bound of*

$$\min\{g(x) \mid x \in B_i\} \tag{Q_i}$$

exceeds a primal bound of P, B_i *cannot contain an optimal solution of* P. *Thus* P *is equivalent to*

$$\min\{f(x) \mid x \in A \setminus B_i\}\,.$$

This observation also implies that it is not always necessary to solve the subproblems to optimality. As soon as the dual bound exceeds a known primal bound of P, the optimization process can be terminated. Furthermore, it suffices to consider a relaxation of the subproblem.

The branch and bound-algorithm maintains a list of unprocessed subproblems, which is initialized with the original problem P. Each subproblem corresponds to a node in a tree-structure that is expanded by recursively decomposing the feasible regions of subproblems.

As long as there are unprocessed nodes in the tree, the following steps are iterated.

1. Choose an unprocessed subproblem P_i and mark its node as processed.
2. Call the bounding procedure to solve a relaxation of Q_i of P_i.
   - 2a. If Q_i is infeasible, go to Step 1.
   - 2b. If the optimum value of Q_i exceeds the best known primal bound of P, go to Step 1.

2c. If the minimizer $x^\star$ of Q_i is feasible for P_i, update the best known primal bound pb with $f(x^\star)$, if $f(x^\star) < pb$. Go to Step 1.

3. Create $k \geq 2$ new subproblems by decomposing the feasible region of P_i and insert them into the list of unprocessed subproblems.

In this thesis we will use three types of branch and bound-algorithms to solve mixed-integer formulations of nonlinear problems with bounded variables. They differ in the relaxation that is used to compute bounds in the nodes of the tree.

1.3.1 LP-Based Branch and Bound

The basis of an LP-based branch and bound-algorithm for nonlinear mixed-integer programs is a linearization of the original problem. In each subproblem P_i a relaxation Q_i is generated by replacing all integrality constraints by the corresponding box constraints. This relaxation is solved to optimality with the simplex algorithm [26].

A natural way to decompose the feasible region of a subproblem P_i is to choose one of the variables that are required to take integer values in the linearization, but have a fractional value in the optimal solution $x^\star$ of the relaxation Q_i. If none such exists, $x^\star$ is feasible for P_i and the algorithm proceeds with Step 1. If several exist the most common rules are to select either the one with the lowest index or the one whose coefficient in the objective function hast the largest absolute value. When the current node of the branch and bound-tree cannot be pruned in Step 2, two new subproblems are generated. One with the additional constraint $x_i \leq \lfloor x_i^\star \rfloor$, the other one with the additionally constraint $x_i \geq \lceil x_i^\star \rceil$, where x_i is the variable selected for branching.

1.3.2 Lagrangean Decomposition-Based Branch and Bound

In each node of the branch and bound-tree the current subproblem is decomposed as described in Section 1.2.1. Bounds are obtained by computing the Lagrangean Dual of the decomposition with a subgradient algorithm. Alternatively, one or both of the subproblems in the decomposition can first be relaxed, for example by omitting integrality constraints.

Depending on the relaxation used to obtain the dual bound, there are several choices for decomposing the feasible region of the subproblem Q_i in the branching step. When integrality constraints were relaxed, the branching variable can be chosen as in an LP-based branch and bound-algorithm, by choosing a variable with a fractional optimal value $x_i^\star$. The decomposition then works as before. The value of the fractional variable is rounded up and down to the next integer and two new subproblems are created in which the branching variable is forced to

take values below $\lfloor x_i^\star \rfloor$ or above $x_i \geq \lceil x_i^\star \rceil$. If no fractional variables exist in the optimal solutions $x^\star$ and $y^\star$ of the two subproblems of LD_λ, one can choose an index i with $x_i^\star \neq y_i^\star$ and branch by introducing the constraints $x_i \leq x_i^\star$ and $x_i \geq x_i^\star + 1$.

1.3.3 SDP-Based Branch and Bound

A classic approach to solve binary quadratic programs uses semidefinite relaxations instead of LP-relaxations to compute bounds in the nodes of the branch and bound-tree.

In order to define a semidefinite program (SDP) and derive an SDP-relaxation of a binary quadratic program we will need the following definitions.

A symmetric matrix $A \in R^{n \times n}$ is called *positive semidefinite* if

$$x^\top A x \geq 0$$

holds for all $x \in \mathbb{R}^n$. We then write $A \succeq 0$.

For two matrices $A, B \in \mathbb{R}^{m \times n}$ denote by

$$\langle A, B \rangle := \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} b_{ij}$$

the *scalar product* of A and B.

For a matrix $A \in \mathbb{R}^{m \times n}$ let $Diag(A)$ denote the matrix formed by setting all but the entries on the main diagonal of A to zero:

$$Diag(A) := \begin{cases} a_{ij}, \text{ if } i = j \\ 0 \text{ otherwise} \end{cases}$$

Given a *variable matrix* $Y \in \mathbb{R}^{n \times n}$ and coefficient matrices $Q^{(k)} \in \mathbb{R}^{n \times n}$ for $k = 1, \ldots, m$ and $b \in \mathbb{R}^m$, a semidefinite program has the form

$$\begin{array}{rlcll} \min & \langle \bar{Q}^{(0)}, Y \rangle & & & \\ \text{s.t.} & \langle \bar{Q}^{(k)}, Y \rangle & \leq & b_k & \quad k = 1, \ldots, m \\ & Y & \succeq & 0 & \\ & Diag(Y) & = & I\,, & \end{array} \tag{SDP}$$

where I denotes the identity matrix and $rank(Y)$ the rank of Y. The set of feasible solutions of SDP consists of those matrices that are symmetric and positive semidefinite, have only ones on the main diagonal and additionally satisfy the quadratic constraints.

Now consider a general binary quadratic program (BQP)

$$\begin{array}{lll} \min & x^\top Q^{(0)} x + {L^{(0)}}^\top x & \\ \text{s.t.} & x^\top Q^{(k)} x + {L^{(k)}}^\top x \;\leq b_k & k = 1, \ldots, m \\ & x \in \{0,1\}^n\,, & \end{array} \tag{BQP}$$

where $Q^{(k)} \in \mathbb{R}^{n \times n}$ for $k = 0, \ldots, m$ are the symmetric coefficient matrices of the quadratic terms, $L^{(k)}$ for $k = 0, \ldots, m$ are the coefficient vectors of the linear terms and $b \in \mathbb{R}^m$.

Reformulating BQP such that the new model can be relaxed to a semidefinite program involves two steps[46]. The first is to transform the domain of the binary variables x from $\{0, 1\}$ to $\{-1, 1\}$. This is achieved by the linear transformation

$$\bar{x}_i = 2x_i - 1\,.$$

The second step is to define appropriate coefficient matrices $\bar{Q}^{(k)}$ for $k = 0, \ldots, m$. Set

$$\bar{Q}^{(k)} = \begin{pmatrix} \frac{1}{2}\sum_{i=1}^n l_i^{(k)} + \frac{1}{4}\sum_{i=1}^n\sum_{j=1}^n q_{ij}^{(k)} & \frac{1}{4}l_1^{(k)} + \frac{1}{8}\sum_{j=1}^n q_{1j}^{(k)} & \cdots & \frac{1}{4}l_n^{(k)} + \frac{1}{8}\sum_{j=1}^n q_{nj}^{(k)} \\ \frac{1}{4}l_1^{(k)} + \frac{1}{8}\sum_{j=1}^n q_{1j}^{(k)} & \frac{1}{4}q_{11}^{(k)} & \cdots & \frac{1}{4}q_{1n}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{4}l_n^{(k)} + \frac{1}{8}\sum_{j=1}^n q_{nj}^{(k)} & \frac{1}{4}q_{n1}^{(k)} & \cdots & \frac{1}{4}q_{nn}^{(k)} \end{pmatrix}$$

BQP is then isomorphic to

$$\begin{array}{lrcll} \min & \langle \bar{Q}^{(0)}, Y \rangle & & & \\ \text{s.t.} & \langle \bar{Q}^{(k)}, Y \rangle & \leq & b_k & k = 1, \ldots, m \\ & Y & = & \begin{pmatrix} 1 \\ \bar{x} \end{pmatrix} \begin{pmatrix} 1 \\ \bar{x} \end{pmatrix}^\top & \\ & \bar{x} & \in & \{-1, 1\}^n\,. & \end{array} \tag{1.3}$$

The last two constraints ban be equivalently expressed as $Y \succeq 0$, $Diag(Y) = I$ and $rank(Y) = 1$, which gives the formulation

$$\begin{array}{lrcll} \min & \langle \bar{Q}^{(0)}, Y \rangle & & & \\ \text{s.t.} & \langle \bar{Q}^{(k)}, Y \rangle & \leq & b_k & k = 1, \ldots, m \\ & Y & \succeq & 0 & \\ & Diag(Y) & = & I & \\ & rank(Y) & = & 1\,. & \end{array} \tag{1.4}$$

Omitting the constraint $rank(Y) = 1$ yields an SDP-relaxation of BQP, which can be efficiently solved with interior point methods [61] or bundle methods [41, 60].

In the branching step two new subproblems are created by fixing one variable y_{ij} to -1 in one subproblem and to 1 in the other. Helmberg and Rendl [59] propose several branching rules. Two of them are adopted in [109]. The first chooses the variable y_{ij} which has the lowest absolute value in the solution of the current SDP. The second selects the rows i and j that have the lowest values $\sum_{k=1}^{n}(1 - y_{lk})^2$ and branches on y_{ij}.

1.3.4 Enumeration Strategies

The *enumeration strategy* determines the order in which unprocessed subproblems are selected in Step 1 of the branch and bound-algorithm. The two most commonly used strategies are *depth first* and *best first*. The depth first strategy always selects the subproblem at the back of the list, i.e. the subproblem that was most recently added. In this approach the deeper levels of the branch and bound-tree, where the feasible regions of the relaxations are small and the probability of finding a feasible solution is higher, are explored first. A disadvantage the depth first-strategy is that the global dual bound improves only slowly, since, by Theorem 1.5, it is determined by the minimum dual bound of all subproblems in the deepest level that has been completely processed. This effect is avoided in the best first-strategy, where always one of the subproblems whose parent defined the current global dual bound is selected for processing.

1.4 Branch and Cut

The idea of the *branch and cut*-approach for mixed-integer programs is to combine an LP- or SDP-based branch and bound-algorithm with a *cutting plane algorithm* to improve the quality of the dual bounds obtained in the nodes of the branch and bound-tree and thus decrease the number of nodes that have to be processed in order to find an optimal solution of the MIP.

The basic idea of a cutting plane-algorithm is to iteratively tighten a relaxation by adding valid inequalities, preferably facets of the convex hull of feasible solutions of the original MIP, to the model. Each iteration consists of two step. First, an optimal solution of the current relaxation is computed. Then an inequality that is valid for all feasible solutions of the MIP but is violated by the current optimizer is added to the model. When the new model is solved in the next iteration, the formerly optimal point will be infeasible for the improved relaxation and the new dual bound will be at least as good as the old one.

The cutting plane-algorithm for mixed-integer linear programs was first proposed

by Gomory [52], together with a *separation algorithm* that generates cutting planes from the optimal simplex tableau of the LP-relaxation. In theory, MIPs can be solved to optimality in a finite number of iterations with a cutting plane-algorithm. In practice, it is advantageous to combine this approach with a branch and bound-algorithm, since the cutting planes quickly become weaker. In a branch and cut-algorithm, a limited number of cutting plane-iterations is performed in each node of the branch and bound-tree. When no more cutting planes can be found or the improvement of the dual bound falls below a given threshold, a branching step is performed.

For combinatorial optimization problems, valid inequalities are commonly not generated with the procedure proposed by Gomory, but with problem-specific separation algorithms that exploit the combinatorial structure of the underlying problem. For some linear combinatorial problems, like the *minimum spanning tree problem* or the *perfect matching problem*, complete polyhedral descriptions of the convex hulls of feasible points are known and cutting planes can be computed efficiently, although the complete description is of exponential size [36, 34]. For other problems, such as the *travelling salesman problem*, efficient separation algorithms are only known for certain classes of valid inequalities [7]. Nevertheless, the augmentation of branch and bound-algorithms with the separation of cutting planes has proven to be very effective in practice.

Chapter 2

Binary Quadratic Optimization

In this chapter we study binary quadratic optimization problems, i.e. optimization problems defined on binary variables with quadratic terms in the objective function and/or the constraints. Since the class of quadratic functions includes the linear functions, an integer linear program (ILP) can be considered a special case of quadratic binary optimization.

We study the basic problem with n binary variables and m constraints

$$\begin{array}{lll} \min & x^\top Q^{(0)} x + {L^{(0)}}^\top x & \\ \text{s.t.} & x^\top Q^{(k)} x + {L^{(k)}}^\top x & \leq b_k \qquad k = 1, \ldots, m \\ & x \in \{0,1\}^n\,, & \end{array} \tag{2.1}$$

where $Q^{(k)} \in \mathbb{R}^{n \times n}$ for $k = 0, \ldots, m$ are the coefficient matrices of the quadratic terms, $L^{(k)}$ for $k = 0, \ldots, m$ are the coefficient vectors of the linear terms and $b \in \mathbb{R}^m$. Without loss of generality we assume that the $Q^{(k)}$ are upper triangular matrices, i.e. we have $q_{ij}^{(k)} = 0$ whenever $j \leq i$.

2.1 Standard Linearization

The basic idea of all linearization approaches is to transform the quadratic binary problem into an equivalent linear one by expressing the quadratic nature of the problem using only linear terms. Recall that, according to Definition 1.1, two problems are equivalent if there is a bijection between the sets of optimal solutions and equivalent optimal solutions have the same objective value. The transformation is achieved by adding linearization variables (binary or continuous) and linking constraints to the model that replace the quadratic objective function and constraints.

The following definition will ease notation in this chapter.

Definition 2.1. Given a quadratic constraint

$$x^\top Q x + L^\top x \leq b\,,$$

where Q is not necessarily an upper triangular matrix, substituting all square terms $x_i x_i$ with x_i and all terms $x_i x_j$ for $i < j$ with a new variable $y_{ij} \geq 0$ is called the *trivial linearization* of the quadratic constraint.

The most commonly used technique, the so-called *standard linearization*, was first proposed by Glover and Woolsey [51] and is based on earlier work by Fortet [43]. Replace each quadratic term $x_i x_j$ with a new continuous variable y_{ij} and add the following coupling constraints:

$$y_{ij} \geq 0 \tag{2.2}$$
$$y_{ij} \leq x_i \tag{2.3}$$
$$y_{ij} \leq x_j \tag{2.4}$$
$$y_{ij} \geq x_i + x_j - 1 \tag{2.5}$$

The resulting problem is linear. It reads

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n} \sum_{j=i+1}^{n} q_{ij}^{(0)} y_{ij} + \sum_{i=1}^{n} L_i^{(0)} x_i \\
\text{s.t.} \quad & \sum_{i=1}^{n} \sum_{j=i+1}^{n} q_{ij}^{(k)} y_{ij} \;+\; \sum_{i=1}^{n} L_i^{(k)} x_i \leq b_k \qquad k = 1, \ldots, m \\
& y_{ij} \;\geq\; 0 \\
& y_{ij} \;\leq\; x_i \\
& y_{ij} \;\leq\; x_j \\
& y_{ij} \;\geq\; x_i + x_j - 1 \\
& x \;\in\; \{0,1\}^n \\
& y \;\in\; \mathbb{R}^{\binom{n}{2}}\,.
\end{aligned} \tag{2.6}$$

Note that the linearization variables y need not be declared as binary explicitly. Indeed, when one of the variables x_i and x_j takes the value zero, the corresponding linearization variable y_{ij} is zero as well. When $x_i = x_j = 1$ and thus $x_i x_j = 1$, we have $y_{ij} = 1$. Therefore, Problem (2.1) and its standard linearization are not only equivalent, but also isomorphic.

It is easily checked that the standard linearization (2.6) is a linearization of (2.1) in the sense of Definition 1.2, however it requires $\mathrm{O}(n^2)$ additional continuous variables and $\mathrm{O}(n^2)$ additional linear constraints. More compact linearizations have been proposed, e.g. by Oral and Kettani [104], whose linearization only requires $\mathrm{O}(n)$ additional continuous variables and linear constraints. The common drawback of the linearizations discussed so far is that their LP-relaxations generally yield weak bounds on the optimal value of (2.1).

There are two ways to get a better description of the convex hull of the feasible solutions of (2.6) and thus stronger bounds. The first is to use an extended formulation. This increases the dimension of the problem. The other is to add valid inequalities to (2.6), which will increase the number of constraints in the LP, but not the number of variables.

One approach by De Simone [28] exploits the equivalence of binary quadratic programming and the MaxCut problem. The basic idea is to generate a better description of the convex hull of feasible solutions of (2.6) using cutting planes valid for an appropriate MaxCut polytope.

In the following sections we will study the relation between unconstrained binary quadratic programming and the MaxCut problem and discuss separation algorithms for the most important class of cutting planes for the cut polytope, the odd cycle inequalities.

We will return to the general case of binary quadratic programming with constraints in Section 2.3. All MaxCut inequalities remain valid for this case and can be used to strengthen the LP-relaxation of (2.6). These relaxations can be further improved by replacing linear constraints with equivalent quadratic ones. We study two general reformulation techniques in 2.3.1 and 2.3.2.

Section 2.3.3 presents a reformulation technique for assignment constraints, which occur in a wide range of combinatorial problems. It explicitly exploits the structure of the constraints and has the advantage of improving the effectiveness of the separation routines for the cut polytope without generating additional linearization variables.

2.2 Unconstrained Binary Quadratic Optimization

The *maximum cut problem* (MaxCut) is defined as follows.

Definition 2.2 (MaxCut)**.** Given a weighted undirected graph $G = (V, E, c)$, find a subset S of the nodes V, such that the total weight of all edges of E with exactly one endpoint in S is maximal.

To express the MaxCut problem as an IP, identify each edge $e = (u, v) \in E$ with a binary variable z_{uv}. Each cut S in V can then be expressed as a vector z with

$$z_{uv} = \begin{cases} 1 \text{ if } u \in S \text{ or } v \in S, \text{ but not both} \\ 0 \text{ otherwise.} \end{cases}$$

We get

$$\begin{array}{rcl} \max & c^\top z & \\ \text{s.t.} & z & \in \quad C(G)\,, \end{array} \tag{2.7}$$

where $C(G) \subseteq \{0,1\}^E$ is the set of all vectors corresponding to cuts in G. The convex hull of $C(G)$ is called the *cut polytope* of the graph G.

Now consider an unconstrained binary quadratic optimization problem defined on an undirected graph $H = (W, F)$ and given as

$$\begin{array}{ll} \max & a^\top x + b^\top y \\ \text{s.t.} & \binom{x}{y} \in BQ(H)\,, \end{array} \tag{2.8}$$

where $BQ(H) \subseteq \{0,1\}^{W \times F}$ is the set of all binary vectors $\binom{x}{y}$ such that $y_{vw} = x_v x_w$ for all edges $e = (v, w) \in F$. The convex hull of $BQ(H)$ is called the *boolean quadric polytope* of the graph H [106].

The binary quadratic problem (2.8) reduces to a MaxCut problem (2.7) on a modified graph [28]. Denote by $H + u$ the graph constructed from H by adding a node u and connecting all original nodes of H to u with an edge. For each z in the feasible set $C(H+u)$ corresponding to $H+u$, define $g(z)$ as $\binom{x}{y} \in \{0,1\}^{W+F}$ with

$$\begin{aligned} x_v &= z_{uv} \text{ for all } v \in W \text{ and} \\ y_{vw} &= x_v x_w \text{ for all } vw \in F\,. \end{aligned}$$

$g(z)$ is a bijective mapping between $C(H+u)$ and $BQ(H)$. By definition of g we have

$$y_{vw} = x_v x_w = z_{uv} z_{uw}$$

and since z corresponds to a cut in $H + u$, this is equivalent to

$$y_{vw} = \frac{1}{2}(z_{uv} + z_{uw} - z_{vw})\,,$$

as can be seen from Figure 2.1, and therefore

$$\tbinom{a}{b} g(z) = c^\top z\,,$$

for an appropriate vector c. This means that the binary quadratic problem on H reduces to a MaxCut problem on $H + u$.

Furthermore, the *cut polytope* of $H+u$ is mapped to the *boolean quadric polytope* of H under the linear transformation $f\colon \mathbb{R}^E \longrightarrow \mathbb{R}^{W \cup F}$ defined by

$$\begin{aligned} z_{uv} &\mapsto x_v\,, \\ z_{vw} &\mapsto x_v + x_w - 2y_{vw}\,. \end{aligned} \tag{2.9}$$

This means that all classes of valid inequalities known for the cut polytope can be used to strengthen the relaxation of the unconstrained binary quadratic problem. Moreover, inequalities inducing facets of the cut polytope remain facet-inducing for the boolean quadric polytope under the transformation f.

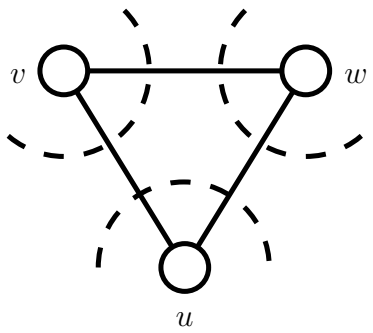

Figure 2.1: For any cut in $H + u$, there are four possible cases in the subgraph u, v and w. The empty cut contains none of the edges of the subgraph, this case corresponds to $z_{uv} = z_{uw} = z_{vw} = 0$. All other cuts contain exactly two edges.

In the presence of linear constraints, these cutting planes remain valid, although inequalities inducing facets of $C(H + u)$ not necessarily induce facets of the convex hull of the intersection of $BQ(H)$ with the additional hyperplanes under f. Nevertheless, the equivalence between the MaxCut problem and binary unconstrained quadratic optimization provides separation routines which can be used in a branch and cut-algorithm for constrained binary quadratic problems of the form (2.1), which works as follows. The standard linearization of (2.1) is solved with an LP-based branch and bound-algorithm. When the solution of an LP is fractional, it is mapped to the separation graph $H + u$ with the inverse of the linear transformation f. When a violated MaxCut-inequality is found by a separation algorithm, it is mapped back to an inequality in the original variables with f and added to the LP-relaxation.

2.2.1 Odd Cycle Inequalities

The most important class of valid inequalities for the cut polytope are the *odd cycle inequalities*. As was shown by Barahona and Mahjoub [12], they can be separated in polynomial time and under certain conditions induce facets of the cut polytope.

Definition 2.3 (odd cycle inequality)**.** Given an undirected graph $G = (V, E)$ and a set of edges $C \subseteq E$ which forms a cycle in G, let D denote a subset of C with odd cardinality. The inequality

$$z(D) - z(C \setminus D) \leq |D| - 1$$

is called an *odd cycle inequality*.

An odd cycle inequality induces a facet of the cut polytope of G if and only if its corresponding cycle C is chordless [12], i.e. C is not decomposable into two

or more shorter cycles. In fact, the MaxCut problem can be formulated as an IP using only odd cycle inequalities [31]. Problem (2.7) is equivalent to

$$\begin{array}{ll} \max & c^\top z \\ \text{s.t.} & z(D) - z(C \setminus D) \leq |D| - 1 \quad \forall (C, D) \in T(G) \\ & z \in \{0,1\}^E \,, \end{array}$$

where $T(G)$ contains all pairs of chordless cycles C of G and their subsets of odd cardinality D.

Furthermore, it was shown by Barahona and Mahjoub [12] that, when G does not contain a complete graph on five nodes as a minor, the set of odd cycle inequalities of G in combination with the inequalities $0 \leq z_e \leq 1$ for those edges $e \in E$ that belong to no triangle in G even provide a complete description of the cut polytope of G.

Now consider the graph $H + u$ from above. For each edge vw of the original edge set F, $H + u$ contains the triangle formed by uv, uw and vw. Each odd cardinality subset of the triangle defines an odd cycle inequality, which induces a facet of the convex hull of $C(H + u)$, since the triangle is a chordless cycle. These four inequalities are

$$\begin{aligned} z_{vw} - z_{uv} - z_{uw} &\leq 0 \text{ for } D = \{vw\} \,, \\ z_{uv} - z_{uw} - z_{vw} &\leq 0 \text{ for } D = \{uv\} \,, \\ z_{uw} - z_{uv} - z_{vw} &\leq 0 \text{ for } D = \{uw\} \text{ and} \\ z_{uv} + z_{uw} + z_{vw} &\leq 2 \text{ for } D = \{uv, uw, vw\} \,. \end{aligned}$$

Under the linear transformation f they are mapped to the following valid inequalities for the boolean quadric polytope of H:

$$\begin{aligned} y_{vw} &\geq 0 \,, \\ y_{vw} &\leq x_w \,, \\ y_{vw} &\leq x_v \text{ and} \\ y_{vw} &\geq x_v + x_w - 1 \,. \end{aligned}$$

This is exactly the standard linearization (2.2) – (2.5) of the quadratic term $x_v x_w$. We see that odd cycle inequalities induced by triangles containing the auxiliary node u do not strengthen the LP-relaxation of (2.6).

Triangles of original nodes of H, on the other hand, induce valid inequalities which indeed give a tighter description of the convex hull of $BQ(H)$. Choose three nodes $i, j, k \in W$ which form a cycle C in H and let $D = \{ij\}$. The resulting odd cycle inequality

$$z_{ij} - z_{ik} - z_{jk} \leq 0$$

is mapped to the inequality

$$y_{ij} + y_{ik} + y_{jk} - x_i - x_j - x_k \leq -1 \,, \tag{2.10}$$

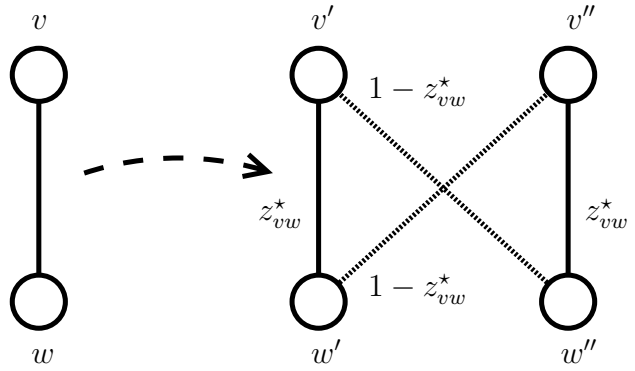

Figure 2.2: The construction of the auxiliary graph G' used in the exact separation algorithm for odd cycle inequalities, illustrated on a single edge of G. Grey edges used in the optimal path P form the odd set D.

which cannot be expressed as a conic combination of the inequalities describing the LP-relaxation of (2.6). To see why, remember that, in the standard form, all inequalities defining the standard relaxation have a non-negative right-hand side, whereas the right-hand side of (2.10) is negative.

The following observation shows that cycles including the auxiliary node u, apart from triangles, can only induce lower-dimensional faces of the cut polytope.

Observation 2.1. *Cycles which contain the auxiliary node u and consist of more than three edges cannot induce facets of the cut polytope of $H + u$.*

By construction of $H + u$ each original node is linked to u by an edge. It immediately follows that any cycle which includes u and has more than three edges must have a chord. Since odd cycle inequalities induce facets if and only if the cycle is chordless [12], long cycles including u must induce faces of lower dimension.

Observation 2.1 has an important practical consequence. For the separation of odd cycle inequalities it is not necessary to consider the extended graph $H + u$. Working with H instead results in the same LP-relaxation and at the same time speeds up the separation algorithms described in the following sections, since their running times depend on the dimension of the underlying graph.

Odd cycle inequalities can be separated in polynomial time, both by exact and heuristic algorithms.

Exact Separation of Odd Cycle Inequalities

Given a solution $z^\star$ of an LP-relaxation of (2.7), the exact separation algorithm by Barahona and Mahjoub [12] computes a cycle $C \subseteq E$ in G and a set $D \subseteq C$

of odd cardinality inducing an odd cycle inequality which is maximally violated by $z^\star$, if one exists.

It works on an auxiliary graph $G' = (V' \cup V'', E')$ constructed from $G = (V, E)$. G' has two nodes $v' \in V'$ and $v'' \in V''$ for each original node $v \in V$ and four edges for each original edge $e = (v, w) \in E$. These are (v', w') and (v'', w'') with weight $z^\star_{vw}$ and (v', w'') and (v'', w') with weight $1 - z^\star_{vw}$ (see Figure 2.2). The algorithm then computes a shortest path from v' to v'' for each $v \in G$. The shortest of these paths, P, defines the desired cycle $C \subseteq E$ and the set $D \subseteq C$ as follows. Each edge in P by construction corresponds to an edge of G. P links two copies of the same node of G and thus the original edges corresponding to edges of P form a cycle in G. To link its start and end node, P must use an odd number of edges connecting V' and V''. These define the odd subset D of the cycle C.

If the weight

$$\sum_{e \in C \setminus D} z^\star_e + \sum_{e \in D} (1 - z^\star_e)$$

of the resulting cycle is less then 1, C and D define a violated odd cycle inequality, otherwise none exists. To see this, observe that an inequality of the form

$$\sum_{e \in D} z_e - \sum_{e \in C \setminus D} z_e \leq |D| - 1$$

can be written as

$$\sum_{e \in C \setminus D} z_e + \sum_{e \in D} (1 - z_e) \geq 1$$

by multiplying with -1 and bringing $|D|$ to the left-hand side.

The algorithm by Barahona and Mahjoub [12] computes $|V|$ shortest s-t-paths in the auxiliary graph of size $O(|V|^2)$. Using an appropriate implementation of the shortest path algorithm, the overall running time is $O(|V|^3)$.

Heuristic Separation of Odd Cycle Inequalities

The exact separation algorithm for odd cycle inequalities described above runs in polynomial time, but may still be too time-consuming in some situations. An alternative is a heuristic algorithm, the *forest cycle separation* algorithm [13] (see Algorithm 1). It operates on the original graph G. Assume that G is connected, otherwise the algorithm can be applied to each connected component of G.

Given a solution $z^\star$ of an LP-relaxation of (2.7), the algorithm first computes a spanning tree of maximum weight T in G, where the edge weights are chosen as $w_e = |z^\star_e - \frac{1}{2}|$. Adding an edge $e \notin T$ to T now creates a cycle C in T, which can easily be found with a breadth-first-search in $T \cup \{e\}$. The set D is chosen as the set of edges $f \in C$ with $z^\star_f$ above a certain threshold, for example 0.5. If the

cardinality of D is odd, C and D define a candidate inequality. If this inequality is violated, it is added to the relaxation. This process is repeated for all edges $e \notin T$.

The computation of the spanning tree takes $O(|V|^2)$ time [107] and is done only once. For each non-tree edge the cycle is computed in $O(|V|)$, so that the overall running time is again $O(|V|^3)$, but in contrast to the exact algorithm up to $|E| - (|V| - 1)$ violated inequalities can be found per application. On the other hand, the heuristic algorithm will not necessarily find the most violated inequality and might fail even if violated inequalities exist.

Algorithm 1 The forest cycle separation algorithm.

---

input: undirected graph $G = (V, E)$, $z^\star \in [0, 1]^E$,
output: set $\mathcal{I}$ of odd cycle inequalities $\sum_{e \in D} z_e - \sum_{e \in C \setminus D} z_e \leq |D| - 1$

$\mathcal{I} \leftarrow \emptyset$
$T \leftarrow$ maximum spanning tree in G with edge weights $w_e = |z^\star_e - \frac{1}{2}|$
for $e \in E \setminus T$ **do**
  $C \leftarrow$ cycle in $T \cup \{e\}$
  $D \leftarrow \emptyset$
  for $f \in C$ **do**
    if $z^\star_f > 0.5$ **then**
      $D \leftarrow D \cup \{f\}$
    end if
  end for
  if $|D|$ odd and $\sum_{e \in D} z^\star_e - \sum_{e \in C \setminus D} z^\star_e \leq |D| - 1$ **then**
    $\mathcal{I} \leftarrow \sum_{e \in D} z_e - \sum_{e \in C \setminus D} z_e \leq |D| - 1$
  end if
end for
return $\mathcal{I}$

2.2.2 More Cutting Planes

The cut polytope has been intensively studied and many more classes of valid, and sometimes facet-defining, inequalities have been described [31], but for many of these classes no efficient separation algorithms are known.

The *hypermetric inequalities* [30] are of the form

$$\sum_{e=(u,v)\in E} b_u b_v z_e \leq 0\,,$$

where the b_u for $u \in V$ are integer numbers that sum to 1. They contain the triangle inequalities

$$z_{uv} - z_{uw} - z_{vw} \leq 0$$

for $u, v, w \in V$ as a special case. No exact separation algorithm for general hypermetric inequalities is known, but De Simone and Rinaldi [29] propose a heuristic separation procedure and find that hypermetric inequalities are effective in branch and cut-algorithms for the MaxCut problem.

Gap inequalities were introduced by Laurent and Poljak [80] and are a generalization of hypermetric inequalities. Let $b \in \mathbb{Z}^V$ and define

$$\sigma(b) := \sum_{v\in V} b_v \text{ and } \gamma(b) := \min_{S\subseteq V} \left| \sum_{v\in S} b_v - \sum_{v\in V\setminus S} b_v \right| .$$

The inequality

$$\sum_{e=(u,v)\in E} b_u b_v z_e \leq \frac{1}{4}\left(\sigma(b)^2 - \gamma(b)^2\right)$$

is called a gap inequality.

Theorem 2.2 (Laurent and Poljak [80])**.** *Gap inequalities are valid for the cut polytope of a complete graph.*

Proof. Let $G = (V, E)$ be a complete graph, $b \in \mathbb{Z}^V$ and $S \subseteq V$. Without loss of generality assume $|S| \geq |V \setminus S|$. By the definition of $\gamma(b)$ we then have

$$\sum_{v\in S} b_v - \sum_{v\in V\setminus S} b_v \geq \gamma(b) \geq 0$$

and therefore

$$\begin{aligned}
\frac{1}{4}\left(\sigma(b)^2 - \gamma(b)^2\right) &= \frac{1}{4}\left(\left(\sum_{v\in S} b_v + \sum_{v\in V\setminus S} b_v\right)^2 - \gamma(b)^2\right) \\
&\geq \frac{1}{4}\left(\left(\sum_{v\in S} b_v + \sum_{v\in V\setminus S} b_v\right)^2 - \left(\sum_{v\in S} b_v - \sum_{v\in V\setminus S} b_v\right)^2\right) \\
&= \left(\sum_{v\in S} b_v\right)\left(\sum_{v\in V\setminus S} b_v\right) = \sum_{(u,v)\in\delta(S)} b_u b_v\,.
\end{aligned}$$

□

The complexity of separating general gap inequalities is unknown, but Galli et al. [48] recently were able to show that it is possible in finite time. In the same paper the authors prove that already deciding whether a vector $z^\star \in [0,1]^E$ violates some gap inequality with $\gamma(b) = 1$ is *NP*-hard. Nevertheless, gap inequalities can be separated heuristically. In [47], Galli et al. devise a heuristic separation algorithm for gap inequalities based on Eigenvalue computations and show experimentally that gap inequalities yield strong relaxations of the MaxCut problem.

2.3 Quadratic Reformulation

So far we have considered binary problems with both a quadratic objective function and quadratic constraints. We have seen how these problems can be reformulated as purely linear problems and how the resulting relaxations can be strengthened with cutting planes from MaxCut. Sometimes not all constraints of a QIP are quadratic. Especially in combinatorial applications, often only the objective function is quadratic, while the set of feasible solutions is characterized by linear inequalities.

Take the *angular-metric travelling salesman problem* [1], for example, which has applications in robotics [87] and vehicle routing [91]. Like in the linear TSP, each Hamiltonian cycle C in the undirected graph $G = (V, E)$ is a feasible solution, but the costs are not determined by summing up the costs of the edges in C. Instead the costs are determined by the sum of the angles between adjacent edges in C. Denote by A the set of all unordered pairs of edges that share a node. The objective function of the angular-metric TSP can then be expressed as

$$\sum_{\{e,f\} \in A} \alpha_{\{e,f\}} x_e x_f \,,$$

where $\alpha_{\{e,f\}}$ depends on the angle between the edges e and f.

Costs induced by pairs of edges also occur in network design, where they are used to model interference costs. One example is the *quadratic minimum spanning tree problem*, which was first described by Assad and Xu [9]. Here the set A often is not restricted to pairs of adjacent edges but may include all unordered pairs of edges [23].

A natural approach to solve such quadratic combinatorial problems is to apply the linearization approach of Section 2.1. The resulting linear program can then be solved with a branch and cut-algorithm. This approach is advantageous, since the LP-relaxations can be strengthened by exploiting the equivalence between binary quadratic optimization and MaxCut described in Section 2.2.

Another technique to obtain tighter LP-relaxations, which can be combined with the cutting plane-approach, is quadratic reformulation of linear constraints. The

basic idea is to replace the set of linear constraints, or a subset, by equivalent quadratic constraints, before linearization. This has two effects: The linearized quadratic constraints often provide a better description of the convex hull of feasible points of the linearized problem and the introduction of additional quadratic terms leads to denser separation graphs when quadratic reformulation is combined with the cutting plane approach presented in this chapter.

We will investigate the properties of two quadratic reformulation techniques proposed by Helmberg et al. [62]. In this paper, the authors investigate the effectiveness of the two approaches in the context of semidefinite programming. They study the *quadratic knapsack problem* (QK). QK is a binary quadratic optimization problem with a single linear constraint

$$a^\top x \leq 1 \tag{2.11}$$

with $x \in \{0,1\}^n$ and $a_i > 0$ for all $i \in \{1,\dots,n\}$.

Similar to Helmberg et al. [62], we investigate the quadratic knapsack problem, i.e. we study the effects of quadratic reformulation of a single linear constraint of the form (2.11). In contrast to Helmberg et al. [62], we are interested in the improvement of LP-based relaxations.

Without loss of generality, in the following we can assume $a_i \leq 1$ for all $i \in \{1,\dots,n\}$, since all variables x_j with $a_j > 1$ will have value zero in all feasible solutions of (2.11).

2.3.1 SQK2

The idea of the first reformulation, called SQK2 in [62], is to square both sides of the inequality in (2.11) and thus obtain a single quadratic inequality which is equivalent to the original one:

$$\begin{aligned} & a^\top x \leq 1 \\ \overset{a,x\geq 0}{\iff} & (a^\top x)^2 \leq 1^2 \\ \iff & \sum_{i=1}^{n} a_i^2 x_i^2 + 2\sum_{i=1}^{n}\sum_{j=i+1}^{n} a_i a_j x_i x_j \leq 1 \end{aligned} \tag{2.12}$$

Applying the trivial linearization to inequality (2.12) yields the linear inequality

$$\sum_{i=1}^{n} a_i^2 x_i + 2\sum_{i=1}^{n}\sum_{j=i+1}^{n} a_i a_j y_{ij} \leq 1\,. \tag{2.13}$$

The SQK2 reformulation approach requires $\binom{n}{2}$ linearization variables, where n is the number of distinct variables in the original constraint. The following observation shows that, although the constraint (2.12) is equivalent to the original

constraint for binary variables x, assuming only the trivial linearization for the linearization variables in (2.13) does not necessarily yield an equivalent problem.

Observation 2.3. *The problem obtained from* (2.11) *by SQK2 together with the trivial linearization in general is not a linearization of* (2.11)*.*

Proof. Recall that a linear problem is a linearization of a nonlinear problem if the two problems are equivalent in the sense of Definition 1.1. To prove the statement of the proposition we will construct an instance of (2.11), for which SQK2 together with the trivial linearization does not produce an equivalent problem.
Let $n = 2$ and set $a_1 = \frac{3}{4}$ and $a_2 = \frac{1}{2}$. The SQK2 reformulation of the constraint

$$\frac{3}{4}x_1 + \frac{1}{2}x_2 \leq 1$$

is

$$\frac{9}{16}x_1^2 + \frac{1}{4}x_2^2 + \frac{3}{4}x_1x_2 \leq 1\,.$$

Applying the trivial linearization gives

$$\frac{9}{16}x_1 + \frac{1}{4}x_2 + \frac{3}{4}y_{12} \leq 1\,.$$

The points $(0,0)$, $(1,0)$ and $(0,1)$ are feasible for the original problem, $(1,1)$ is infeasible. For the linearized problem $(0,0,0)$, $(1,0,0)$, $(1,1,0)$, $(0,1,0)$ and $(0,0,1)$ are feasible, while $(1,0,1)$, $(0,1,1)$ and $(1,1,1)$ are infeasible. Since the two sets of feasible solutions have different cardinalities, there cannot exist a bijection between them. This shows that the two problems are not isomorphic. With an appropriately chosen objective function they are also not equivalent. □

If we assume the standard linearization instead of the trivial linearization for the reformulated problem, we get an integer linear problem which is equivalent to the original problem. However, the following theorem shows that this approach does not always yield a tighter LP-relaxation.

Theorem 2.4. *The polyhedron corresponding to the inequality obtained from* $a^\top x \leq 1$ *by SQK2 and the standard linearization in general is not contained in the polyhedron corresponding to* $a^\top x \leq 1$*.*

Proof. For variables $x, y, z \in \{0,1\}$ and $a, b, c > 0$ consider the inequality

$$ax + by + cz \leq 1 \tag{2.14}$$

and the corresponding polytope

$$P_1 = \{(x,y,z) \in \mathbb{R}^3 \mid ax + by + cz \leq 1,\ 0 \leq x, y, z \leq 1\}\,.$$

The SQK2 reformulation of (2.14) is

$$a^2x + b^2y + c^2z + 2abu + 2acv + 2bcw \leq 1\,. \tag{2.15}$$

Assuming the standard linearization for the new variables u, v, and w, the polyhedron corresponding to the linearization of (2.15) is

$$\begin{aligned} P_2 = \{(x,y,z,u,v,w) \in \mathbb{R}^6 \mid\, & a^2x + b^2y + c^2z + 2abu + 2acv + 2bcw \leq 1, \\ & 0 \leq x,y,z \leq 1,\ u,v,w \geq 0, \\ & u \leq x, u \leq y,\ u \geq x+y-1, \\ & v \leq x, v \leq z,\ v \geq x+z-1, \\ & w \leq y, w \leq z,\ w \geq y+z-1\}\,. \end{aligned}$$

Sequentially projecting out the variables u, v and w with Fourier-Motzkin-elimination gives the projection $\bar{P}_2$ of P_2 onto the original variable space:

$$\begin{aligned} \bar{P}_2 = \big\{(x,y,z) \in \mathbb{R}^3 \mid\, & 0 \leq x,y,z \leq 1, \\ & \frac{a}{2b}\,x + \frac{b}{2a}\,y + \frac{c^2}{2ab}\,z \leq \frac{1}{2ab}, \\ & \frac{a+2c}{2b}\,x + \frac{b}{2a}\,y + \frac{c^2+2ac}{2ab}\,z \leq \frac{1}{2ab} + \frac{c}{b}, \\ & \frac{a}{2b}\,x + \frac{b+2c}{2a}\,y + \frac{c^2+2bc}{2ab}\,z \leq \frac{1}{2ab} + \frac{c}{a}, \\ & \frac{a+2c}{2b}\,x + \frac{b+2c}{2a}\,y + \frac{c^2+2ac+2bc}{2ab}\,z \leq \frac{1}{2ab} + \frac{c}{a} + \frac{c}{b}, \\ & \Big(\frac{a}{2b}+1\Big)x + \Big(\frac{b}{2a}+1\Big)y + \frac{c^2}{2ab}\,z \leq \frac{1}{2ab} + 1, \\ & \left(\frac{a+2c}{2b}+1\right)x + \left(\frac{b}{2a}+1\right)y + \frac{c^2+2ac}{2ab}\,z \leq \frac{1}{2ab} + \frac{c}{b} + 1, \\ & \Big(\frac{a}{2b}+1\Big)x + \left(\frac{b+2c}{2a}+1\right)y + \frac{c^2+2bc}{2ab}\,z \leq \frac{1}{2ab} + \frac{c}{a} + 1, \\ & \left(\frac{a+2c}{2b}+1\right)x + \left(\frac{b+2c}{2a}+1\right)y + \frac{c^2+2ac+2bc}{2ab}\,z \leq \frac{1}{2ab} + \frac{c}{a} + \frac{c}{b} + 1\big\} \end{aligned}$$

In general $\bar{P}_2$ is not completely contained in P_1. Consider, for example, the coefficients $a = \frac{1}{2}$, $b = \frac{1}{2}$, $c = 1$. The point $(x,y,z) = (1,0,\frac{1}{2})$ is feasible for P_1, since $\frac{1}{2}\cdot 1 + \frac{1}{2}\cdot 0 + 1\cdot\frac{1}{2} \leq 1$, but it is not feasible for $\bar{P}_2$, since it violates the constraint

$$\left(\frac{a+2c}{2b}+1\right)x + \left(\frac{b}{2a}+1\right)y + \frac{c^2+2ac}{2ab}\,z \leq \frac{1}{2ab} + \frac{c}{b} + 1\,.$$

For $a = \frac{1}{2}$, $b = \frac{1}{2}$, $c = 1$ we have

$$\frac{7}{2}\cdot 1 + \frac{3}{2}\cdot 0 + 4\cdot\frac{1}{2} > 5\,.$$

□

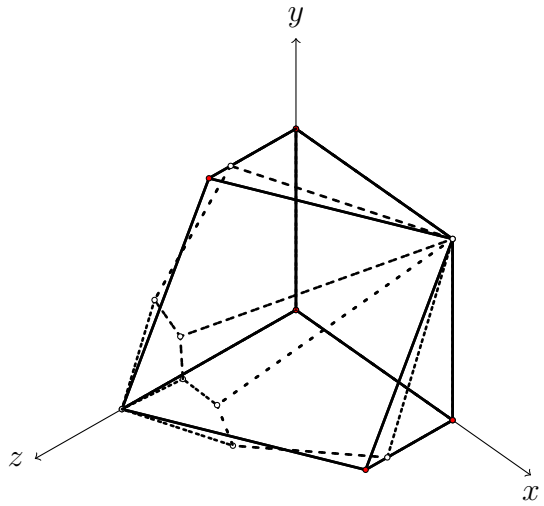

Figure 2.3: The polytopes P_1 and $\bar{P}_2$ for the example given above. The original polytope P_1 is drawn with bold edges, the polytope $\bar{P}_2$ of the reformulation with dashed edges. As can be seen, neither polytope is completely contained in the other.

Observation 2.5. *The example constructed in the previous proof, illustrated in Figure 2.3, also shows that SQK2 with standard linearization not necessarily leads to a weaker LP-relaxation, since in general also P_1 is not completely contained in $\bar{P}_2$.*

The point $(x, y, z) = (0, \frac{1}{2}, \frac{7}{8})$ satisfies all inequalities describing $\bar{P}_2$, but is not contained in P_1, since it violates the original constraint

$$\frac{1}{2}x + \frac{1}{2}y + \frac{7}{8}z \leq 1\,.$$

Theorem 2.4 and Observation 2.5 suggest that in order to achieve tighter LP-relaxations by using SQK2, at least the standard linearization should be applied to the linearization variables and the original linear constraints should be kept. This is not necessary for SDP-relaxations. As was shown by Helmberg et al. [62], SQK2 strengthens the SDP-relaxation of the quadratic knapsack problem even if the original constraint is dropped from the model.

2.3.2 SQK3

The second reformulation proposed in [62], called SQK3, is based on a reformulation by Sherali and Adams [116]. SQK3 works by multiplying (2.11) with each variable x_i in turn to produce n new inequalities and with the complement $(1 - x_j)$ for a given index j. The total number of new inequalities is $n + 1$, in

comparison to $2n$ in the reformulation by Sherali and Adams [116], where the original constraint is multiplied with all variables and all complements.

Consider again the knapsack constraint (2.11) and denote by P_3 the corresponding polytope

$$P_3 = \{x \in \mathbb{R}^n \mid a^\top x \leq 1,\, 0 \leq x \leq 1\}\,.$$

Call the polyhedron corresponding to the standard linearization of the SQK3 reformulation of (2.11) $\tilde{P}_3$ and its projection onto the original variable space $\bar{P}_3$.

It was already observed by Sherali and Adams [116] that SQK3 improves the original LP-relaxation, even when only the trivial linearization is applied:

Theorem 2.6 (Sherali and Adams [116])**.** *The polyhedron $\bar{P}_3$ corresponding to the SQK3 reformulation of inequality* (2.11) *is completely contained in the polyhedron P_3.*

Proof. Consider two of the inequalities that define $\tilde{P}_3$:

$$x_j(a^\top x) \leq x_j$$

and

$$(1 - x_j)(a^\top x) \leq 1 - x_j\,.$$

Summing up these inequalities gives the original constraint

$$a^\top x \leq 1\,.$$

Thus this inequality is valid for the polyhedron $\tilde{P}_3$ as well as for its projection $\bar{P}_3$. □

Theorem 2.7 (Helmberg et al. [62])**.** *The reformulation SQK3 gives a tighter description of the convex hull of integer points of P_3 than SQK2, independent of the linearization of the quadratic terms.*

Proof. We show that the quadratic inequality

$$(a^\top x)^2 \leq 1$$

obtained from the original constraint

$$a^\top x \leq 1$$

by SQK2 is implied by the inequalities obtained by SQK3.

For $i \in \{1, \ldots, n\}$ denote by I_i the inequality obtained by multiplying the original constraint with x_i. Then

$$\sum_{i \in \{1,\ldots,n\}} a_i I_i$$

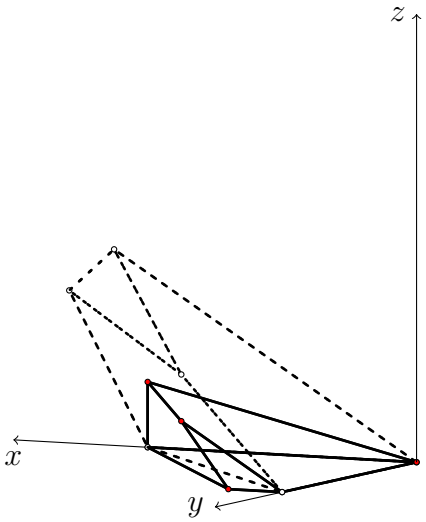

Figure 2.4: The two polyhedra P_3 and $\bar{P}_3$ from the proof of Theorem 2.8, for $a = \frac{8}{15}$ and $b = \frac{4}{5}$. P_3 is drawn with dashed lines, $\bar{P}_3$ with solid lines.

is equal to

$$(a^\top x)^2 \leq a^\top x\,.$$

By Theorem 2.6 the original constraint $a^\top x \leq 1$ is implied by the constraint system generated by SQK3 and thus $(a^\top x)^2 \leq 1$ holds for any binary vector $x \in \mathbb{R}^n$ which satisfies the inequalities of the SQK3 reformulation. □

Theorem 2.8. *In general the reformulation SQK3 of a constraint does not imply the standard linearization of the resulting quadratic terms.*

Proof. Consider a quadratic problem of the form

$$\begin{array}{lrcl} \min & -cxy & & \\ \text{s.t.} & ax + by & \leq & 1 \\ & x & \in & \{0,1\} \\ & y & \in & \{0,1\}\,, \end{array} \tag{2.16}$$

with $0 < a, b, c < 1$ and replace the term xy by the new variable $z \geq 0$. The polyhedron corresponding to the LP-relaxation of the standard linearization of (2.16) is

$$P = \{(x,y,z) \in \mathbb{R}^3 \mid ax + by \leq 1,\ 0 \leq x, y \leq 1,\ 0 \leq z \leq x,\ z \leq y,\ z \geq x + y - 1\}\,.$$

The SQK3 reformulation of $ax + by \leq 1$ consists of the inequalities

$$\begin{aligned} -x + ax^2 + bxy &\leq 0\,, \\ -y + axy + by^2 &\leq 0 \text{ and} \\ (a+1)x + by - ax^2 - bxy &\leq 1\,, \end{aligned}$$

obtained by multiplying the original inequality with the variables x and y and with the complement of one of the variables, in this case $1 - x$. Substituting the square terms with the original variables and xy with $z > 0$ before relaxing the integrality constraints yields the polyhedron

$$\begin{aligned} \bar{P} = \{(x, y, z) \in \mathbb{R}^3 \mid & 0 \leq x, y \leq 1, \\ & bz \leq (1-a)x,\ az \leq (1-b)y,\ bz + 1 \geq x + by,\ z \geq 0\}\,. \end{aligned}$$

To prove the statement of the theorem, we will show that, for some given coefficients a and b, there exists a point $(\bar{x}, \bar{y}, \bar{z})$ with $(\bar{x}, \bar{y}, \bar{z}) \in \bar{P}$, but $(\bar{x}, \bar{y}, \bar{z}) \notin P$. Let $a = \frac{8}{15}$ and $b = \frac{4}{5}$. The point $\bar{x} = \frac{1}{5}$, $\bar{y} = 1$, $\bar{z} = 0$ satisfies all inequalities defining $\bar{P}$, but violates the inequality $\bar{z} \geq \bar{x} + \bar{y} - 1$, therefore $(\bar{x}, \bar{y}, \bar{z}) \notin P$. □

Figure 2.4 shows the polyhedra P and $\bar{P}$ corresponding to the standard linearization and the SQK3 reformulation of the binary quadratic problem 2.16 for the coefficients used in the proof of the last theorem. As can be seen, the reformulation produces a polyhedron of much smaller volume, but additionally applying the standard linearization can improve the LP-relaxation of binary quadratic problems, since $\bar{P}$ is not completely contained in P.

2.3.3 Phantom Monomials

In Section 2.1 we presented a linearization approach for binary quadratic optimization problem and discussed how it can be improved by incorporating cutting planes for the cut polytope. In Sections 2.2 and 2.3 we discussed a technique to further strengthen the LP-relaxations by reformulating linear constraints in quadratic terms before linearizing the quadratic model.

A drawback of both approaches is that they increase the size of the model in comparison to the standard linearization, both in regard to the number of variables and the number of constraints. This effect generally is undesired, as it often makes the relaxations harder to solve. Certain combinatorial structures admit a quadratic reformulation that avoids additional variables without losing the benefit of a denser separation graph for MaxCut inequalities.

In many combinatorial applications exactly one or at most one element has to be selected from a set. The *Steiner arborescence problem* is an example for such a problem on a directed graph $G = (V, A, c)$. Given a root vertex r and a set

of so-called terminals $T \subseteq V$, a solution of the Steiner arborescence problem is a subgraph of G that contains a directed path from r to each terminal. A natural IP-formulation of the minimum-weight Steiner arborescence problem [42] contains the constraints

$$\sum_{(v,s)\in A} x_{(v,s)} \leq 1 \quad \forall s \in S\,,$$

where S is the set of all vertices which are not the root vertex r or one of the terminals, the so-called Steiner vertices. They ensure that the in-degree of each Steiner vertex is at most one. Constraints of the form

$$\sum_{e\in\delta(v)} x_e = 1$$

for example occur in IP-formulations of *perfect matching problems* [86]. Here they ensure that for each node of an undirected graph only one incident edge is part of a solution.

For ease of notation we will concentrate on assignment constraints of the form

$$\sum_{i\in I} x_i = 1\,, \tag{2.17}$$

with $x \in \{0,1\}^I$. All results also hold for the $\leq$-case.

An assignment constraint of the form (2.17) mandates that exactly one of the variables in I takes value one, all others value zero. This leads to the

Observation 2.9. *For any vector $x^\star \in \{0,1\}^I$ satisfying* (2.17)*,*

$$x_i^\star x_j^\star = 0$$

holds for all $i, j \in I$ with $i \neq j$.

Observation 2.9 implies that the linearization variables of all quadratic monomials $x_i x_j$ with $i, j \in I$ can be fixed to zero in the ILP-model. The same information can be obtained by applying the quadratic reformulations SQK2 and SQK3 to the assignment constraint. SQK2 produces the linearized constraint

$$\sum_{i\in I} x_i + 2\sum_{\substack{i,j\in I\\ i\neq j}} y_{ij} = 1\,.$$

Together with the original inequality and $y_{ij} \geq 0$ this implies $y_{ij} = 0$ for all $i \neq j$.

The system produced by SQK3 consists of $|I| + 1$ inequalities. In a first step the original inequality is multiplied with each of the variables in turn, the last

inequality is obtained by multiplication with $(1 - x_k)$ for some $k \in I$. In the case of assignment constraints, multiplication with x_j yields

$$\sum_{\substack{i \in I \\ i \neq j}} y_{ij} = 0 \,.$$

Again assuming non-negativity of the linearization variables, the inequalities produced in the first step of SQK3 effectively fix the linearization variables to zero. The linearization variables are present in the model, but their value is predetermined by the combinatorial structure of the assignment constraint. The monomials resulting from quadratic reformulation of assignment constraints are therefore called *phantom monomials* in the following.

Treatment of Phantom Monomials in the IP

Phantom monomials can be linearized by introducing a new variable and fixing it to zero instead of adding the standard linearization constraints (2.2) – (2.5), which leads to a reduction in the size of the linear model. Moreover, any phantom monomials which would only result from quadratic reformulation of the assignment constraint, i.e., phantom monomials which are not part of the original quadratic model, can be used to improve the LP-relaxation without being explicitly introduced.

In the cutting plane approach described earlier, each quadratic monomial corresponds to an edge in the separation graph for MaxCut inequalities. When quadratic reformulation introduces new quadratic monomials, the separation graph becomes denser. This is advantageous, because now more and stronger cutting planes can be separated. This suggests the following treatment of assignment constraints and the corresponding phantom monomials. Instead of reformulating an assignment constraint with SQK2 or SQK3, which would produce unnecessary linearization variables and constraints, the original linear constraint is kept in the model and any edges that would result from phantom monomials are added to the separation graph. This requires a slight adaptation of the cutting plane-algorithm, more specifically, the linear transformation f providing the mapping between the variables of the linearization and the separation graph. Recall that for ordinary monomials $x_i x_j$ the weight of the corresponding edge in the separation graph is defined as the LP-value of

$$x_i + x_j - 2y_{ij} \,.$$

When $x_i x_j$ is a phantom monomial, the information that its linearization variable has value zero is encoded in the transformation by setting the weight of the edge to the LP-value of

$$x_i + x_j \,.$$

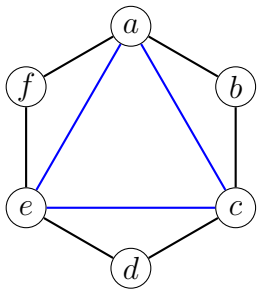

Figure 2.5: The separation graph M corresponding to the system of assignment constraints (2.18). The blue triangle induces the inequality $x_a + x_c + x_e \leq 1$, which is not implied by the original constraints.

This modified transformation also ensures a correct mapping of separated inequalities into the variable space of the ILP.

The Structure of the Separation Graph

Phantom monomials change the structure of the separation graph $H + u$. Call the subgraph of $H + u$ induced by the reformulation of a set C of assignment constraints M. In the case $|C| = 1$ M forms a clique. For $|C| > 1$ the resulting graph M consists of a set of cliques and two cliques in M share a node whenever the corresponding pair of assignment constraints shares a variable.

It is interesting to ask whether M alone can yield violated odd cycle inequalities. Consider a single assignment constraint containing at least three variables, x_a, x_b and x_c. Otherwise M consists of an isolated node or a single edge. All chordless cycles in M have length three. Each of these triangles induces four odd cycle inequalities, as explained in Section 2.2.1. Under the modified transformation f they are mapped to

$$\begin{aligned} x_a &\geq 0\,, \\ x_b &\geq 0\,, \\ x_c &\geq 0\,, \\ \text{and } x_a + x_b + x_c &\leq 1\,, \end{aligned}$$

which are trivially satisfied by any vector satisfying the assignment constraint.

For $|C| > 1$ we have to distinguish two cases. When each variable occurs only in a single constraint, the graph M decomposes into $|C|$ connected components, which are again cliques. As before, all odd cycle inequalities are trivially satisfied. When the assignment constraints have common variables the situation is different.

Consider the following system of constraints:

$$\begin{aligned} x_a + x_b + x_c &= 1 \\ x_c + x_d + x_e &= 1 \\ x_a + x_e + x_f &= 1 \end{aligned} \tag{2.18}$$

The corresponding graph M is illustrated in Figure 2.5. The nodes in M associated with x_a, x_c and x_e form a triangle which induces the odd cycle inequality

$$z_{ac} + z_{ae} + z_{ce} \leq 2\,,$$

which is mapped to

$$x_a + x_c + x_e \leq 1\,. \tag{2.19}$$

Constraint (2.19) cuts off the point $x_a = x_c = x_e = \frac{1}{2}$, $x_b = x_d = x_f = 0$, which is feasible for the system (2.18).

The previous example shows that phantom monomials can be used to generate cutting planes even if the original model does not contain quadratic terms. In this case the separation graph is purely induced by phantom monomials and the inequalities found by the separation algorithms are mapped into the original variable space.

When the original model contains quadratic terms, as in the quadratic perfect matching problem mentioned earlier, phantom monomials increase the density of the separation graph. The augmented graph potentially contains a larger number of cycles, which is beneficial for the separation of odd cycle inequalities.

2.4 Final Remarks

In this chapter we discussed techniques for solving constrained binary quadratic optimization problems. The basic idea is to improve the standard linearization approach by reformulating linear constraints and by exploiting the equivalence between unconstrained binary quadratic optimization and the MaxCut problem to generate cutting planes. The methods presented here are evaluated experimentally in Chapter 6, where we apply them to the quadratic minimum spanning tree problem and the minimum-weight quadratic perfect matching problem.

Chapter 3

Submodular Combinatorial Optimization

In this chapter we study combinatorial optimization problems with submodular objective functions. Our aim is a general approach to compute provably exact solutions for submodular combinatorial optimization problems. We do not know of any previous work that provides a general framework for constrained submodular optimization.

In some applications the problem can be modeled as a nonlinear integer program, in others it is reformulated as an integer linear program. ILP models have the advantage that they are well studied and state-of-the-art solvers are extremely efficient. A downside of considering an extended linear formulation is that the linearization often can only be achieved by introducing a large number of new variables and linear constraints to the model, reducing the advantage of using linear solvers considerably. Additionally, such reformulations often not only affect the objective function but also the original constraints, obscuring or even destroying the combinatorial structure of the problem. Our aim in this chapter is to develop generic algorithmic frameworks for submodular combinatorial optimizaton problems that however exploit the given combinatorial structure.

The first section introduces the concept of submodularity and gives a brief overview over the field of submodular function minimization. In the second section we study how submodular functions can be constructed and present some classes of submodular functions that have applications in combinatorial optimization. Afterwards we develop two exact algorithms for submodular combinatorial optimization. The first is a branch and cut-approach based on a study of the polyhedron corresponding to the submodular objective function, the second is a branch and bound algorithm that uses Lagrangean decomposition to generate lower bounds. This second approach capitalizes on the existence of efficient optimization algorithms for unconstrained submodular function minimization on the

one hand and specialized algorithms for the application-specific combinatorial constraints of the problem.

3.1 Submodularity

Submodularity is a property of set functions. Given a set S, a function $f\colon 2^S \to \mathbb{R}$ is called *submodular*, if for each pair of subsets $A, B \subseteq S$ the property

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$$

holds. If $-f$ is submodular, f is called supermodular. It is easy to see that the class of submodular functions comprises the class of linear set functions. Submodularity can be interpreted as *diminishing returns*: to see this consider the equivalent definition

$$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B),$$

for $A \subseteq B \subseteq S$ and $x \notin B$. Including the element x into a larger set generates less additional profit. A simple example of a submodular function is the maximum function. Given a weight w_s for each element s of S, the function

$$f(A) = \max_{s \in A} w_s$$

returns the weight of the heaviest element in the subset.

It is an important property of submodular functions that they can be minimized efficiently if the set of feasible solutions is not restricted. The first strongly polynomial time algorithm for submodular function minimization (SFM) was proposed by Grötschel et al. [54], using the ellipsoid method. The problem of finding a combinatorial algorithm was open for more than a decade. It was finally resolved independently by Schrijver [112] and Iwata et al. [70]. Since then, several fully combinatorial algorithms were devised [105, 69].

Despite recent progress in the development of combinatorial algorithms for SFM, their computational complexity still often prohibits their use in practical applications. The algorithm by Orlin [105], which is the fastest general-purpose algorithm currently known, takes $\mathcal{O}(n^5 EO + n^6)$ time, where EO is the time needed for evaluating the submodular function f and n is the cardinality of the ground set S. Even if f can be evaluated in linear time, the overall complexity is $\mathcal{O}(n^6)$. For some classes of submodular functions, however, specialized algorithms exist which allow efficient minimization also in practice. Any linear set function of the form $f(A) = \sum_{s \in A} c_s$ with $c_s \in \mathbb{R}$ for all $s \in S$, for example, can be minimized in linear time by a simple greedy algorithm.

In the presence of constraints on the set of feasible solutions, SFM often becomes *NP*-hard. This is the case even if optimizing a linear objective function subject to

the same constraints is easy. One example is the submodular edge cover problem. In the linear variant the aim is to find a minimum-weight subset X of the edges of a weighted undirected graph, such that every node is incident to an edge in X. This problem was shown to be solvable in polynomial time by Murty and Perin [99]. The algorithm is based on the blossom algorithm for matching problems [33]. When the linear objective is replaced by a submodular function, the problem becomes *NP*-hard. This was shown by Iwata and Nagano [68], who gave a reduction of MIN 2-SAT to the submodular edge cover problem. Another example is the minimum spanning tree problem. It is well-known to be solvable in $\mathcal{O}(|E| \log |V|)$ time, where $|E|$ is the cardinality of the edge set of the undirected weighted graph and $|V|$ its number of nodes, and even in $\mathcal{O}(|E| + |V| \log |V|)$, when more sophisticated data structures are used [107]. A special submodular variant of MST is the minimum-power symmetric connectivity problem. Here for each node only the most costly incident tree edge contributes to the overall costs of the tree. This problem was shown to be *NP*-hard by Fuchs [44]. The minimum-power symmetric connectivity problem belongs to the class of *range assignment problems*, which will be treated in detail in Chapter 7.

Submodular function minimization can be formulated as an integer program. Associate each element s_i of the ground set S with a binary variable x_i. Each incidence vector represents a subset of the ground set. The problem of minimizing a submodular function f then can be written as

$$\min_{x \in \{0,1\}^S} f(x)\,.$$

Any restrictions on the set of feasible subsets can be expressed in terms of the x-variables by restricting the set of feasible incidence vectors X:

$$\begin{array}{lll} \min & f(x) & \\ \text{s.t.} & x & \in \quad X \subseteq \{0,1\}^S\,. \end{array}$$

This formulation in general is nonlinear. Compact linearizations that exploit the structure of the function f exist for some special cases. A linearization approach that is applicable to any submodular function is studied in Section 3.3.

3.2 Constructing Submodular Functions

To check if a given set function f is submodular, it suffices to check if the submodular inequality

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$$

holds for all pairs A and B of subsets of the ground set S. By the following proposition, conic combinations of submodular functions are submodular again. Thus it suffices to check submodularity for the individual summands.

Proposition 3.1 (Nemhauser and Wolsey [100]). *Let $f_1 \dots, f_n$ be submodular functions and $\alpha_1, \dots, \alpha_n \geq 0$ non-negative scalars. Then the function*

$$f = \sum_{i=1}^{n} \alpha_i f_i$$

is submodular again.

For linear set functions equality holds in the inequality above, they are modular. Quadratic set functions are not submodular in general, but when all coefficients have the same sign, they are super-/submodular.

Theorem 3.2 (Lee et al. [82]). *Let the function f be defined on the set of subsets of S as*

$$f \colon 2^S \longrightarrow \mathbb{R}, \quad A \mapsto \sum_{i,j \in A} q_{ij} \,,$$

where $q_{ij} \geq 0$ for all $i, j \in S$. Then f is supermodular.

Proof. A function g is supermodular, if $-g$ is submodular, i.e. if for all $A \subseteq B \subseteq S$ and $k \in S \setminus B$ the property of diminishing returns holds:

$$f(A \cup k) - f(A) \leq f(B \cup k) - f(B) \,.$$

Let $A \subseteq B \subseteq S$ and $k \in S \setminus B$.

$$\begin{aligned}
&f(A \cup k) - f(A) - (f(B \cup k) - f(B)) \\
&= \sum_{i,j \in A \cup k} q_{ij} - \sum_{i,j \in A} q_{ij} - \sum_{i,j \in B \cup k} q_{ij} + \sum_{i,j \in B} q_{ij} \\
&= \sum_{i \in A \cup k} \sum_{j \in A \cup k} q_{ij} - \sum_{i \in A} \sum_{j \in A} q_{ij} - \sum_{i \in B \cup k} \sum_{j \in B \cup k} q_{ij} + \sum_{i \in B} \sum_{j \in B} q_{ij} \\
&= \sum_{i \in A \cup k} \left(\sum_{j \in A} q_{ij} + q_{ik} \right) - \sum_{i \in A} \sum_{j \in A} q_{ij} - \sum_{i \in B \cup k} \left(\sum_{j \in B} q_{ij} + q_{ik} \right) + \sum_{i \in B} \sum_{j \in B} q_{ij} \\
&= \sum_{j \in A} q_{kj} + \sum_{i \in A} q_{ik} - \sum_{j \in B} q_{kj} - \sum_{i \in B} q_{ik} \\
&= - \sum_{j \in B \setminus A} q_{kj} - \sum_{i \in B \setminus A} q_{ik} \leq 0
\end{aligned}$$

□

The composition of super-/submodular functions with other classes of functions in general does not yield submodular functions. In the following we give an overview of compositions of sub-/supermodular functions with concave/convex functions. For some cases it can be shown that the resulting functions are submodular, for other cases we give counterexamples.

Theorem 3.3. *Given a nondecreasing submodular set function $g\colon 2^S \to \mathbb{R}$ and a nondecreasing concave function $f\colon \mathbb{R} \to \mathbb{R}$, the composition $f \circ g\colon 2^S \to \mathbb{R}$, $A \subseteq 2^S \mapsto f(g(A))$ is nondecreasing submodular.*

Proof. Recall some definitions: g is submodular and nondecreasing, so for $A, B \subseteq S$ we have

$$g(A) + g(B) \geq g(A \cup B) + g(A \cap B)$$

and

$$g(A) \leq g(B)$$

whenever $A \subseteq B \subseteq S$. Concavity of f means that for arbitrary $t \in [0, 1]$ we have

$$f(tx + (1-t)y) \geq tf(x) + (1-t)f(y).$$

A function $f\colon \mathbb{R} \to \mathbb{R}$ is called nondecreasing if $x \leq y$ implies $f(x) \leq f(y)$.

The composition $f \circ g$ obviously is nondecreasing. To show that it is submodular, consider the following identity:

$$\begin{aligned} &f(g(A)) + f(g(B)) - f(g(A \cup B)) - f(g(A \cap B)) \\ &\quad = f(g(A)) - f(g(A \cup B) + g(A \cap B) - g(B)) \\ &\qquad + f(g(A \cup B) + g(A \cap B) - g(B)) - f(g(A \cup B)) - f(g(A \cap B)) + f(g(B)) \end{aligned} \tag{1}$$

To prove submodularity of $f \circ g$ it suffices to show that both parts of the right hand side of the identity are nonnegative.

For the first part we use the submodularity of g and that f is nondecreasing:

$$\begin{aligned} &g(A) \geq g(A \cup B) + g(A \cap B) - g(B) \\ \Rightarrow &f(g(A)) \geq f(g(A \cup B) + g(A \cap B) - g(B)) \\ \Rightarrow &f(g(A)) - f(g(A \cup B) + g(A \cap B) - g(B)) \geq 0 \end{aligned}$$

For the second part we start with the monotony of g:

$$\begin{aligned} A \cap B \subseteq B \subseteq A \cup B &\Rightarrow g(A \cap B) \leq g(B) \leq g(A \cup B) \\ &\Rightarrow \exists t \in [0, 1] : g(B) = tg(A \cap B) + (1-t)g(A \cup B) \end{aligned} \tag{2}$$

It follows that

$$f(g(B)) = f(tg(A \cap B) + (1-t)g(A \cup B)) \geq tf(g(A \cap B)) + (1-t)f(g(A \cup B)), \tag{3}$$

because f is concave. This gives us:

$$\begin{aligned}
&f(g(A\cup B)+g(A\cap B)-g(B))-f(g(A\cup B))-f(g(A\cap B))+f(g(B))\\
&\overset{(2)}{=} f(g(A\cup B)+g(A\cap B)-tg(A\cap B)-(1-t)g(A\cup B))\\
&\quad -f(g(A\cup B))-f(g(A\cap B))+f(g(B))\\
&= f(tg(A\cup B)+(1-t)g(A\cap B))-f(g(A\cup B))-f(g(A\cap B))+f(g(B))\\
&\overset{(3)}{\geq} f(tg(A\cup B)+(1-t)g(A\cap B))\\
&\quad -f(g(A\cup B))-f(g(A\cap B))+tf(g(A\cap B))+(1-t)f(g(A\cup B))\\
&= f(tg(A\cup B)+(1-t)g(A\cap B))-tf(g(A\cup B))-(1-t)f(g(A\cap B))\\
&\overset{(3)}{\geq} tf(g(A\cup B))+(1-t)f(g(A\cap B))-tf(g(A\cup B))-(1-t)f(g(A\cap B))\\
&= 0
\end{aligned}$$

Identity (1) together with (2) and (3) implies

$$f(g(A))+f(g(B))\geq f(g(A\cup B))+f(g(A\cap B))\,.$$

□

Example 3.1. Fix an order of the elements in S and for any $A\subseteq S$ denote by $x_A\in\{0,1\}^S$ the vector with

$$x_i=\begin{cases}1, \text{ if } i\in A\,,\\ 0 \text{ otherwise.}\end{cases}$$

For a given diagonal matrix $D\geq 0$, consider the function

$$f\colon 2^S\longrightarrow\mathbb{R},\quad A\mapsto\sqrt{x_A^\top Dx_A} \tag{3.1}$$

Since the x-variables are binary, f can be written as

$$f(A)=\sqrt{\sum_{i\in A}d_ix_i^2}=\sqrt{\sum_{i\in A}d_ix_i},$$

where d_i is the i-th entry on the main diagonal of D. f is submodular, since it is the composition of a (sub-)modular function and a nondecreasing concave function.

Functions of the form (3.1) are used in portfolio theory to model the variance of the expected returns of investments. One application is the *risk-averse capital budgeting problem*, which is studied in Chapter 8.

Similar versions of Theorem 3.3 can be proved for different combinations of sub-/supermodularity, concav-/convexity and monotonies. g is always assumed to be monotone. The following overview is taken from [120]:

f				g		$f \circ g$	
convex	concave	non-incr.	non-decr.	superm.	subm.	superm.	subm.
×			×	×		×	
×		×			×	×	
	×	×		×			×
	×		×		×		×

An obvious extension of the class of functions of the form (3.1) would be to use a covariance matrix instead of a diagonal matrix. In the context of portfolio theory this would model not only the variance of the investments, but also the correlations between the developments of the investments. The following counterexample shows that the resulting function is not always submodular.

Observation 3.4. *There exists a covariance matrix Q such that the function*

$$f_Q \colon \{0,1\}^n \to \mathbb{R}, \quad x \mapsto \sqrt{x^\top Q x}$$

is not submodular.

Proof. Let $Q \in \mathbb{R}^{3\times 3}$,

$$Q = \begin{pmatrix} 3 & -3 & -2 \\ -3 & 4 & 4 \\ -2 & 4 & 6 \end{pmatrix}$$

1) Q is a covariance matrix:
The eigenvalues of Q are $\lambda_1 = 10.6758$, $\lambda_2 = 2.2406$ and $\lambda_3 = 0.0836$. Since Q is symmetric and all eigenvalues of Q are non-negative, Q is symmetric positive definite and a matrix W exists such that $Q = WW^\top$. Consider a vector X of random variables with $X \sim \mathcal{N}(0,1)$, i.e. the entries of X are normally distributed with mean value $\mu = 0$ and variance 1. Then the covariance matrix of WX is

$$Cov(WX) = WCov(X)W^{\mathrm{T}} = W(E((X-\mu)(X-\mu^\top)))W^\top = WW^\top = Q\,.$$

2) $f_Q \colon \{0,1\}^3 \to \mathbb{R}$, $x \mapsto \sqrt{x^\top Q x}$ is not submodular:
For f_Q to be submodular, the following inequality must hold:

$$f_Q(A) + f_Q(B) \geq f_Q(A \cup B) + f_Q(A \cap B) \quad \forall A, B \subseteq S = \{x_1, x_2, x_3\}$$

Let $A = \{x_1, x_2\}$, $B = \{x_1, x_3\}$.

$$f_Q(A) + f_Q(B) = \sqrt{1} + \sqrt{5} < \sqrt{11} + \sqrt{3} = f_Q(A \cup B) + f_Q(A \cap B),$$

consequently f_Q is not submodular.

□

Observation 3.5. *The composition of a supermodular function and an nondecreasing concave function is not necessarily submodular. This holds even if the supermodular function is defined by a symmetric non-negative matrix.*

Proof. Let $Q \in \mathbb{R}^{3\times 3}$,

$$Q = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 3 & 2 \\ 0 & 2 & 3 \end{pmatrix}$$

Q is obviously symmetric and non-negative. The function

$$f_Q \colon \{0,1\}^3 \to \mathbb{R}, \quad x \mapsto \sqrt{x^\top Q x}$$

is not submodular, since for $A = \{x_1, x_2\}$ and $B = \{x_1, x_3\}$ we have:

$$f_Q(A) + f_Q(B) = \sqrt{6} + \sqrt{6} < \sqrt{13} + \sqrt{3} = f_B(A \cup B) + f_Q(A \cap B).$$

□

3.3 Polyhedral Study

In the following, we study the polyhedral structure of submodular combinatorial optimization problems. We describe a class of linear inequalities that gives a complete description of the corresponding polyhedron in the unconstrained case and a corresponding efficient separation algorithm. Combined with the polyhedral description of the set of feasible solutions X we obtain an LP-relaxation of the problem

$$\begin{array}{lll} \min & f(x) & \\ \text{s.t.} & x & \in \quad X \subseteq \{0,1\}^S, \end{array} \tag{3.2}$$

where $f \colon 2^S \to \mathbb{R}$ is a submodular function on a set S. Without loss of generality we can assume $f(\emptyset) \geq 0$. We associate each binary variable x_i with an element of S.

Starting from the unconstrained nonlinear model

$$\begin{array}{lll} \min & f(x) & \\ \text{s.t.} & x & \in \quad \{0,1\}^S \end{array}$$

we introduce a single new variable $y \in \mathbb{R}$ to replace the objective function. Clearly, the resulting model

$$\begin{array}{rlll} \min & y & & \\ \text{s.t.} & y & \geq & f(x) \\ & x & \in & \{0,1\}^S \\ & y & \in & \mathbb{R} \end{array}$$

is equivalent to the original one, since we consider a minimization problem. Now consider the convex hull of feasible points:

$$P_f = \operatorname{conv}\big\{(x,y) \in \{0,1\}^S \times \mathbb{R} \mid y \geq f(x)\big\}$$

The polyhedron P_f is the epigraph of the so-called Lovász-extension of f. The following result by Edmonds [36] and Lovász [85] gives a complete polyhedral description of P_f.

Theorem 3.6. *Let $|S| = n$ and let $f\colon 2^S \to \mathbb{R}$ be a submodular function with $f(\emptyset) \geq 0$. Then the separation problem for P_f can be solved in $O(n \log n)$ time. The facets of P_f are either induced by trivial inequalities $0 \leq x_i \leq 1$, $i \in S$, or by an inequality $a^\top x \leq y$ with*

$$a_{\sigma(i)} = f(S_i) - f(S_{i-1}) \forall i \in \{1,\dots,n\}\,, \tag{3.3}$$

where $\sigma\colon \{1,\dots,n\} \to S$ is any bijection and $S_i = \{\sigma(j) \mid j \in \{1,\dots,i\}\}$.

In the presence of constraints the above theorem does not yield a complete polyhedral description anymore, but it still provides strong dual bounds on the LP-relaxation, as we will see in the experimental evaluations in Chapters 7 and 8. The number of facets of P_f is exponential in $n = |S|$, but the separation problem can be solved efficiently by a simple greedy algorithm. Indeed, violation of the trivial facets is checked in linear time. The following algorithm produces a candidate for a separating hyperplane:

Given a fractional point $(x^\star, y^\star) \in [0,1]^S \times \mathbb{R}$, sort the elements of S in non-increasing order according to their value in $x^\star$. Starting with the empty set, iteratively construct a chain of subsets $\emptyset = S_0 \subset S_1 \subset \cdots \subset S_n = S$ by adding the elements in this order. The potentially violated inequality $a^\top x \leq y$ is then constructed by setting $a_i = f(S_i) - f(S_{i-1})$. Obviously this algorithm constructs an inequality of the form (3.3) that is most violated by the given fractional point $(x^\star, y^\star)$. Either this inequality is a separating hyperplane or none such exists. A formal description of this separation procedure is given in Algorithm 2.

In many applications the submodular objective function f can be written as a conical combination of other submodular functions f_i, i.e., we have

$$f = \sum_{i=1}^{k} \alpha_i f_i, \quad \alpha_1,\dots,\alpha_k \geq 0, \quad f_1,\dots,f_k \text{ submodular.}$$

Algorithm 2 Separation Algorithm for P_f

input: a fractional solution $(x^\star, y^\star) = (x_1^\star, \ldots, x_n^\star, y^\star)$
output: a hyperplane $a^\top x \leq y$ separating $(x^\star, y^\star)$ from P_f, if one exists

sort the elements of S into a list $\{l_1, \ldots, l_n\}$ by non-increasing value of $x^\star$
$i \leftarrow 1$
$S_0 \leftarrow \emptyset$
repeat
  $S_i \leftarrow S_{i-1} \cup \{l_i\}$
  $a_i = f(S_i) - f(S_{i-1})$
  $i \leftarrow i + 1$
until $i = n$
if $y^\star < a^\top x^\star$ **then**
  return a
else
  return *no constraint found*
end if

This situation can be exploited by modeling each function f_i separately, introducing a new continuous variable y_i modeling $f_i(x)$ for each $i \in \{1, \ldots, k\}$. Such an approach could be preferable if, e.g., the values $f_i(x)$ are used at other points in the model or if the functions f_i have much smaller domains than f. In the latter case, the total number of inequalities needed to describe the unconstrained problem can be reduced significantly.

Take, for example, the case of an optimization problem on a complete undirected graph on n nodes, where each edge corresponds to a variable and the objective function can be split into one submodular function per node. Modeled as a single function f, the size of the domain of f is $\frac{n(n-1)}{2}$ and the number of inequalities needed to describe the unconstrained problem is $(\frac{n(n-1)}{2})!$. When the problem is modeled with one submodular function per node, each function has a domain of size $n-1$. The number of inequalities then is $n(n-1)! = n!$.

When f is modeled as a conical combination of submodular functions f_i we obtain

$$\begin{array}{lrcl} \min & \displaystyle\sum_{i=1}^{k} \alpha_i y_i & & \\ \text{s.t.} & y_i & \geq & f_i(x) \text{ for all } i \in \{1, \ldots, k\} \\ & x & \in & \{0,1\}^S \\ & y & \in & \mathbb{R}^k\,. \end{array} \tag{3.4}$$

Our next aim is to show that the separation algorithm detailed above can still be

used to generate a complete description for Problem (3.4). First note that Theorem 3.6 yields a complete description of the polytope P_{f_i} for each $i \in \{1 \dots, k\}$. For the following, define

$$P = \bigcap_{i \in \{1,\dots,k\}} P_{f_i} ,$$

where each P_{f_i} is trivially extended from $\{0,1\}^S \times \mathbb{R}$ to $\{0,1\}^S \times \mathbb{R}^k$. We will show that each vertex (x, y) of P satisfies $x \in \{0,1\}^S$ and $y_i = f_i(x)$, and hence is feasible for Problem (3.4). In other words, the separation problem corresponding to (3.4) can be reduced to the single separation problems for each P_{f_i}.

Lemma 3.7. *For any submodular function $f\colon \{0,1\}^S \to \mathbb{R}$ and $j \in S$, there is a submodular function $g\colon \{0,1\}^{S\setminus\{j\}} \to \mathbb{R}$ such that $\{x \in P_f \mid x_j = 0\} = P_g$.*

Proof. For $x \in \{0,1\}^{S\setminus\{j\}}$, let $\bar{x}$ be its extension to $\{0,1\}^S$, setting $\bar{x}_j = 0$. Defining $g(x) = f(\bar{x})$ yields the desired submodular function. □

Lemma 3.8. *For any submodular function $f\colon \{0,1\}^S \to \mathbb{R}$ and $j \in S$, there is a submodular function $g\colon \{0,1\}^{S\setminus\{j\}} \to \mathbb{R}$ such that $\{x \in P_f \mid x_j = 1\} = e_j + P_g$, where e_j denotes the unit vector corresponding to x_j.*

Proof. For $x \in \{0,1\}^{S\setminus\{j\}}$, let $\bar{x}$ be its extension to $\{0,1\}^S$, setting $\bar{x}_j = 1$. Defining $g(x) = f(\bar{x})$ yields the desired submodular function. □

Lemma 3.9. *If $(x, y) \in P$ with $x \in (0,1)^S$, then (x, y) is not a vertex of P.*

Proof. Let $\mathbf{1}_S$ denote the all-ones vector in $\mathbb{R}^S$ and choose $\varepsilon > 0$ such that $x \pm \varepsilon \mathbf{1}_S \in [0,1]^S$. Define $c \in \mathbb{R}^k$ by $c_i = f_i(S) - f_i(\emptyset)$ and consider

$$z_1 = (x - \varepsilon \mathbf{1}_S, y - \varepsilon c),\ z_2 = (x + \varepsilon \mathbf{1}_S, y + \varepsilon c) .$$

As $(x, y) = \frac{1}{2}(z_1 + z_2)$, it suffices to show $z_1, z_2 \in P$. This reduces to showing $(x \pm \varepsilon \mathbf{1}_S, y_i \pm \varepsilon c_i) \in P_{f_i}$ for all $i \in \{1, \dots, k\}$. By Theorem 3.6, the polyhedron P_{f_i} is completely described by trivial inequalities and by inequalities of the type $a^\top x \le y_i$ with

$$a_{\sigma(j)} = f_i(S_j) - f_i(S_{j-1}) \forall j \in \{1, \dots, n\}$$

where $\sigma\colon \{1, \dots, n\} \to S$ is any bijection and $S_j = \{\sigma(1), \dots, \sigma(j)\}$. We obtain in particular that $a^\top \mathbf{1}_S = f_i(S) - f_i(\emptyset) = c_i$. As $(x, y) \in P$ and therefore $(x, y_i) \in P_{f_i}$, we derive

$$a^\top (x \pm \varepsilon \mathbf{1}_S) = a^\top x \pm \varepsilon a^\top \mathbf{1}_S \le y_i \pm \varepsilon c_i .$$

Hence $z_1, z_2 \in P_{f_i}$. □

Theorem 3.10. *The vertices of P are exactly the points $(x, y) \in \{0,1\}^S \times \mathbb{R}^k$ with $y_i = f_i(x)$ for all $i \in \{1, \dots, k\}$.*

Proof. It is clear that every such point is a vertex of P. We show that every vertex (x', y') of P is of this form. Since y_i is not bounded from above, every vertex must satisfy $y_i = f_i(x)$ for all $i \in \{1, \ldots, k\}$. Now assume that at least one component of x' is fractional. Define

$$S_0 = \{j \in S \mid x'_j = 0\}, \quad S_1 = \{j \in S \mid x'_j = 1\}, \quad T = S \setminus \{S_0 \cup S_1\}$$

and consider the face

$$\begin{aligned} F &= \{(x, y) \in P \mid x_j = 0 \text{ for all } j \in S_0, \;\; x_j = 1 \text{ for all } j \in S_1\} \\ &= \bigcap_{i=1}^{k} \{(x, y) \in P_{f_i} \mid x_j = 0 \text{ for all } j \in S_0, \;\; x_j = 1 \text{ for all } j \in S_1\}. \end{aligned}$$

By Lemma 3.7 and Lemma 3.8, the polyhedron F is an intersection of polyhedra $\mathbf{1}_{S_1} + P_{g_i}$ for suitable submodular functions $g_i \colon \{0,1\}^T \to \mathbb{R}$. Since $x'_i \in (0, 1)$ for all $i \in T$, the point $(x' - \mathbf{1}_{S_1}, y')$ is not a vertex of $\bigcap_{i=1}^{k} P_{g_i}$ by Lemma 3.9. It follows that (x', y') is not a vertex of F and hence not a vertex of P. □

Note that the last theorem can also be shown in a more general context [45]. It implies that the polyhedron P_f is a projection of P, given by the linear transformation $y := \sum_{i=1}^{k} \alpha_i y_i$. Moreover, it follows that each facet of P is obtained from a facet of one of the polyhedra P_{f_i}. In particular, the separation problem for (3.4) can be reduced to the respective separation problems for each polyhedron P_{f_i} as follows: to separate a point $x^\star$ from the polytope P_f it is sufficient to check if $x^\star$ violates any of the inequalities characterizing the polyhedra P_{f_i}. This can be done by applying Algorithm 2 to each P_{f_i} in turn.

Submodular Quadratic Minimization

In Chapter 1 we introduced a way to linearize a quadratic function defined on binary variables, the so-called standard linearization. Furthermore, we have seen in Theorem 3.2 that quadratic set functions with only non-positive coefficients are submodular, hence the results of the previous section are applicable. In the following we compare the quality of the relaxations obtained from these two linearization techniques.

Using an incidence vector x for the elements of the ground set S, the quadratic set function f with coefficient matrix $Q \leq 0$ can be written as $f(x) = x^\top Q x$. f is the sum of terms of the form $f_{ij}(x) = q_{ij} x_i x_j$. By Theorem 3.6 all facet-defining inequalities of the polyhedron defined by f are sums of facet-defining inequalities of the polyhedra defined by the f_{ij}. These are the trivial bound inequalities for all variables x_i and the two inequalities $y_{ij} \geq q_{ij} x_i$ and $y_{ij} \geq q_{ij} x_j$, where $y_{ij} \in \mathbb{R}$ is the linearization variable for the function f_{ij}.

Theorem 3.11. *For $Q \leq 0$, applying the submodular linearization to the problem*

$$\{\min x^\top Q x \mid x \in \{0,1\}^n\} \tag{3.5}$$

gives the same bound as the standard linearization.

Proof. Without loss of generality we need to consider only quadratic terms corresponding to entries of Q where q_{ij} is strictly less than zero, since for $q_{ij} = 0$ the quadratic term can be omitted from the function f completely. For binary variables x_i, x_j and a negative scalar q consider the standard linearization of the term qx_ix_j: we replace x_ix_j by the new variable z and add the four constraints

$$z \leq x_i\,, \tag{3.6}$$
$$z \leq x_j\,, \tag{3.7}$$
$$z \geq x_i + x_j - 1 \tag{3.8}$$
$$\text{and } z \geq 0\,. \tag{3.9}$$

Under the substitution $y := q^{-1}z$, the term qz is mapped to y and the constraints become

$$y \geq qx_i\,, \tag{3.10}$$
$$y \geq qx_j\,, \tag{3.11}$$
$$y \leq q(x_i + x_j - 1) \tag{3.12}$$
$$\text{and } y \leq 0\,. \tag{3.13}$$

Since q is negative and we consider a minimization problem, the optimal solution value does not change if we drop constraints (3.12) and (3.13). The resulting problem is exactly the same we get by applying the submodular linearization to the original quadratic term. Consequently, applying the standard linearization to each quadratic monomial in the objective function $x^\top Qx$ leads to a model which is equivalent to the model generated by the submodular linearization approach. □

Remark 3.1. Theorem 3.11 only states that the optimal solutions of both linearizations of (3.5) are the same. The sets of feasible solutions are different, even in the unconstrained case. In fact the feasible set of the relaxation of the standard linearization is contained in P_f. Consider $q < 0$ and the point $(x_i^\star, x_j^\star, y^\star) = (1, 1, 0)$. It is feasible for the submodular linearization, but violates constraint (3.12) of the standard linearization. Thus the models are equivalent but not isomorphic.

This means that in the presence of additional constraints the LP-relaxation of the standard linearization might give better bounds than the LP-relaxation of the submodular linearization. Even more, if the constrained submodular quadratic minimization problem is infeasible, standard linearization will lead to an infeasible LP-relaxation, while the submodular relaxation might be feasible, since the linearization variables y are not bounded from above.

3.4 A Branch and Cut Approach

So far we have only considered unconstrained submodular optimization problems. Recall that the original Problem (3.2) was given as

$$\begin{array}{lll} \min & f(x) & \\ \text{s.t.} & x & \in \quad X \subseteq \{0,1\}^S\,, \end{array}$$

where X is the set of feasible solutions. Assume as before that $f = \sum_{i=1}^{k} \alpha_i f_i$, with $\alpha_1, \ldots, \alpha_k \geq 0$ and $f_1, \ldots, f_k$ submodular. The constrained version of (3.4) then reads

$$\begin{array}{lll} \min & \displaystyle\sum_{i=1}^{k} \alpha_i y_i & \\ \text{s.t.} & y_i & \geq \quad f_i(x) \text{ for all } i \in \{1, \ldots, k\} \\ & x & \in \quad X \\ & y & \in \quad \mathbb{R}^k\,, \end{array}$$

which can be formulated as

$$\begin{array}{lll} \min & \displaystyle\sum_{i=1}^{k} \alpha_i y_i & \\ \text{s.t.} & (x,y) & \in \quad (X \times \mathbb{R}^k) \cap \displaystyle\bigcap_{i \in \{1,\ldots,k\}} P_{f_i} \subseteq \{0,1\}^S \times \mathbb{R}^k\,. \end{array} \tag{3.14}$$

In this case our results remain applicable but do not necessarily give a complete polyhedral description of the problem anymore. Even if the complete linear description of X (or an exact separation algorithm for X) is available, the combination of the inequalities describing X and the polyhedra P_{f_i} in general does not yield a full description of the intersection $(X \times \mathbb{R}) \cap P_f$. For an exact algorithm the generation of cutting planes can be embedded into a branch and bound-approach.

3.4.1 Cutting Planes

The first LP-relaxation of (3.14) that is solved in the root node of the branch and bound-tree consists of an initial set of inequalities from the polyhedral description of X and the variable bounds. If the solution of this LP is fractional, the separation routines for X and the P_{f_i} are used to generate cutting planes. This process is iterated until the solution is integral or no more violated inequalities are found. In this case a branching step on one of the binary variables is performed.

3.4.2 Primal Bounds

Any feasible solution that is found during the branch and bound-process gives an upper bound on the optimal solution value and can be used to prune branches

of the enumeration tree. Additionally, application-specific heuristics can be applied to generate feasible solutions from fractional solutions of the LP-relaxations. These heuristics only need to take into account the combinatorial structure of X and the fractional solution. A valid upper bound can then easily be computed by evaluating the objective function in the point found by the heuristic algorithm.

3.5 A Lagrangean Decomposition Approach

In this section we avoid the linearization of the submodular objective function and capitalize on the existence of polynomial time algorithms for SFM in the unconstrained case. We use Lagrangean decomposition as described in 1.2.1 to separate the objective function from its constraints. Starting from the constrained submodular minimization problem

$$\begin{array}{lll} \min & f(x) & \\ \text{s.t.} & x & \in \quad X \subseteq \{0,1\}^S\,, \end{array} \tag{3.15}$$

we duplicate the variable set and relax the linking constraints. The resulting problem reads:

$$\begin{array}{rlllll} Z(\lambda) = \min & f(x_1) - \lambda^T x_1 & + & \min & \lambda^T x_2 & \\ \text{s.t.} & x_1 \in \{0,1\}^S & & \text{s.t.} & x_2 \in X \subseteq \{0,1\}^S\,. \end{array} \tag{3.16}$$

The first part is an unconstrained minimization problem. The objective function is the sum of a submodular function and a modular function and thus by Proposition 3.1 submodular. As mentioned in Section 3.1 efficient general algorithms for this kind of problem are known. For some specific classes of submodular functions specialized algorithms exist, which have a significantly lower computational complexity than the general algorithms, as we will see in Chapter 7.

The second part of (3.16) is an integer linear problem which is assumed to be easier than the corresponding nonlinear problem. It might be either solved as an ILP with a general-purpose LP-solver, or, depending on the combinatorial structure of the underlying problem, with a specialized combinatorial algorithm. In Chapter 7 we will study so-called range assignment problems. In one of the applications considered there, the underlying combinatorial problem is the minimum spanning tree problem, which is solved not as an LP but directly with Kruskal's algorithm.

3.5.1 Bounds

For any given multiplier λ the value $Z(\lambda)$ gives a lower bound for the decomposition (3.16). If the combinatorial subproblem can be solved in polynomial time,

either with a combinatorial algorithm or by solving its LP-formulation, computing $Z(\lambda)$ also takes only polynomial time. Otherwise, there are two possibilities:

1. (3.16) can be solved as is. This will take exponential time, but may still be practicable, depending on the application.
2. The combinatorial subproblem can be relaxed. If the relaxation is chosen appropriately, it can be computed in polynomial time.

The advantage of the second option is that it allows the computation of lower bounds in polynomial time, even if the linear variant of the original problem is *NP*-hard. On the other hand, it will result in weaker bounds, which is disadvantageous when the Lagrangean decomposition approach is embedded into a branch and bound-algorithm.

The best lower bound that can be obtained from the Lagrangean decomposition of (3.15) is the value of the Lagrangean dual $\max_{\lambda\in\mathbb{R}^S} Z(\lambda)$, which can be computed as described in Chapter 1.

The best possible bound is obtained by computing the Lagrangean dual

$$\max_{\lambda\in\mathbb{R}^S} Z(\lambda)\,,$$

as discussed in Chapter 1. Note that, when (3.16) is solved without relaxing the combinatorial subproblem, we have

$$Z(\lambda) \quad = \quad \left\{ \begin{array}{lll} \min & z - \lambda^\top x_1 & \\ \text{s.t.} & (z, x_1) \in \operatorname{conv}(F) & \end{array} \right. \quad + \quad \begin{array}{lll} \min & \lambda^\top x_2 & \\ \text{s.t.} & x_2 & \in \operatorname{conv}(X) \end{array}$$

where $F := \{(z, x_1) \mid x_1 \in \{0,1\}^n,\ z \geq f(x_1)\}$. By general results on Lagrangean relaxation we obtain

Theorem 3.12 (Guignard and Kim [55])**.**

$$\max_{\lambda\in\mathbb{R}^n} Z(\lambda) \quad = \quad \left\{ \begin{array}{lll} \min & z & \\ \textit{s.t.} & (z, x) & \in \textit{conv}(F) \\ & x & \in \textit{conv}(X)\,. \end{array} \right.$$

Note also that

$$\begin{array}{lll} \min & z & \\ \text{s.t.} & (z, x) \in \operatorname{conv}(F) & \\ & x \in \operatorname{conv}(X) & \end{array} \quad \geq \quad \begin{array}{lll} \min & f(x) & \\ \text{s.t.} & x \in \operatorname{conv}(\{0,1\}^n) & \\ & x \in \operatorname{conv}(X) & \end{array} \quad = \quad \begin{array}{lll} \min & f(x) & \\ \text{s.t.} & x \in \operatorname{conv}(X)\,, & \end{array}$$

and that the inequality is strict in general if f is nonlinear. This is due to the fact that the objective function f is minimized over $\{0,1\}^S$ in the left-hand side

problem of (3.16), instead of over $[0,1]^S$. In other words, the bounds we obtain are potentially stronger than those obtained from convexifying the feasible set in Problem (3.16).

Besides potentially yielding stronger dual bounds than the LP-approach of Section 3.3, the Lagrangean decomposition approach has two advantages. First, using combinatorial algorithms to compute $\max_\lambda Z(\lambda)$ may in practice be faster than solving the corresponding linear program discussed in Section 3.3. The polyhedral description presented in 3.3 is not compact, since the number of facets of the polyhedron P_f is exponential. Therefore, in practice the linear programming relaxation of (3.15) cannot be solved directly. Instead, a cutting plane approach is applied, in which violated inequalities are added dynamically to the linear program. Although the separation algorithm 2 only needs polynomial running time, solving a series of linear programs and repeatedly searching for violated inequalities may in practice take more time than solving a series of combinatorial problems in a subgradient approach. The same arguments hold for the description of the underlying constraints. Even when a complete linear description of the set X is known, it might not be compact, even if efficient algorithms for the optimization of a linear objective function over X exist. Take the minimum spanning tree problem, for example. The spanning tree polyhedron described by the cycle inequalities is integral and minimum spanning trees can be computed very efficiently with combinatorial algorithms, in a linear programming context a cutting plane approach has to be applied to solve the problem to proven optimality.

Secondly, any feasible solution of the partial problem

$$\begin{array}{ll} \min & \lambda^\top x_2 \\ \text{s.t.} & x_2 \in X \subseteq \{0,1\}^S \end{array}$$

is also feasible for the original problem (3.15). This means that in each iteration of the subgradient algorithm, a valid primal bound for (3.15) can be easily computed by evaluating the current solution of the second partial problem in the original objective function f. This may prove to be useful in a branch and bound-approach, where good primal bounds are important for early pruning of subproblems.

3.5.2 Branch and Bound

To compute exact solutions for problem (3.15), the Lagrangean decomposition approach is embedded into a branch and bound-framework. In each node of the enumeration tree a problem of the form (3.16) is solved to obtain both lower and upper bounds. Further differences to a branch and bound-approach based on the LP-relaxation are described in the following.

Fixed Variables

In the LP-context fixing a variable to a value amounts to setting both its lower and upper bound to this value, which is equivalent to adding another linear constraint and does not affect the LP-solver. In the Lagrangean decomposition context, the relaxations in the subproblems are preferably solved with combinatorial algorithms, which have to be adapted to deal with fixed variables. Lemmas 3.7 and 3.8 state that the unconstrained part of (3.16) remains a submodular minimization problem when variables are fixed. Algorithms for the original function f remain applicable, only the domain of f changes; its cardinality is reduced by one for every fixed variable. Algorithms for the second part of (3.16) are application-specific, fixing a variable here is equivalent to changing the combinatorial structure of the problem. This can be achieved by manipulating the underlying graph or network, for example by deleting edges/arcs or merging nodes, or by changing the weights of nodes/edges. When a weight is set to a very big negative value $-M$, the algorithm includes the corresponding object into the solution whenever possible, when the weight is set to a very big positive value M, the algorithms tries to exclude the corresponding object whenever possible. Another possibility is to change the algorithm itself, for example by adding a preprocessing step.

Branching

In an LP-based branch and bound-algorithm a branching step is performed if there is at least one fractional variable. Otherwise the solution of the subproblem is primal feasible and the subproblem can be pruned. In the Lagrangean decomposition context there are two sets of variables, one for each part of the decomposition. When both partial problems are solved with combinatorial algorithms all variables in a solution are necessarily integral. Still, an optimal solution $(x_1^\star, x_2^\star, \lambda^\star)$ of the Lagrangean relaxation in general is not optimal for the original problem. By Theorem 1.3 this is the case only if $\lambda_i^\star((x_1^\star)_i - (x_2^\star)_i) = 0$ for all $i \in S$, i.e. if either both copies of a variable have the same value or the corresponding multiplier is zero. This gives rise to a natural selection rule for branching variables: choose an index i with $\lambda_i^\star \neq 0$ and $(x_1^\star)_i \neq (x_2^\star)_i$. When several exist, one can for example simply choose the lowest or the index i with the largest absolute value $|\lambda_i^\star|$, depending on the application. The branching itself works in the same manner as in the LP-context, by fixing variables to zero or one, except that in the two new subproblems two variables are fixed, $(x_1)_i$ and $(x_2)_i$.

3.6 Final Remarks

We presented two approaches for submodular combinatorial optimization problems which are applicable when either a polyhedral description of the combina-

torial structure is known or the linear counterpart of the problem can be optimized efficiently in practice. In Chapters 7 and 8 these algorithms are applied to submodular combinatorial problems from wireless network-design and portfolio optimization and evaluated experimentally.

Chapter 4

Two-Scenario Optimization

In classical optimization problems the input data is assumed to be known in advance and fixed. This assumption often turns out to be unrealistic. One way to safeguard against uncertainties is to consider several possible scenarios and compute a solution that gives the best possible solution in the worst-case scenario.

In the first section of this chapter we study a simplified version of this general problem, unconstrained two-scenario optimization. We propose an exact optimization algorithm, which will be evaluated experimentally in Chapter 9. In the second part of this chapter we again consider the case of two scenarios but include combinatorial side constraints. Applying the results for the unconstrained case, an exact algorithm for the two-scenario minimum spanning tree is presented. This approach will be evaluated in a computational study in Chapter 9 as well.

4.1 Unconstrained Two-Scenario Optimization

Starting from the general combinatorial optimization problem with k scenarios and bounded integer variables

$$\begin{array}{ll} \min\limits_{x} & \max\{f_1(x), \ldots, f_k(x)\} \\ \text{s.t.} & Cx \leq d \\ & l \leq x \leq u \\ & x \in \mathbb{Z}^n \end{array} \tag{4.1}$$

we simplify in two ways. First, we limit the number of scenarios to two and consider only linear objective functions f. Second, we do not allow arbitrary linear restrictions on the set of feasible solutions, but only box constraints. The resulting problem reads

$$\min_{\substack{l \leq x \leq u \\ x \in \mathbb{Z}^n}} \max\{a^\top x + a_0, b^\top x + b_0\}, \tag{4.2}$$

with $a, b \in \mathbb{R}^n$, $a_0, b_0 \in \mathbb{R}$ and $l, u \in \mathbb{Z}^n$.

4.1.1 Complexity

The two-scenario version of an optimization problem may have the same complexity as the one-scenario problem, but this is not the case in general. Take the *minimum cut problem*, for example: both the one- and two-scenario versions are solvable in polynomial time [8]. In contrast, the *shortest path problem* is easy for one scenario, but becomes *NP*-hard when a constant number of scenarios (greater than one) is considered [124]. The same holds for the *minimum spanning tree problem* [77]. For a survey of the complexity of combinatorial optimization problems with several scenarios, see [3].

We prove that the unconstrained problem with two scenarios (4.2) is *NP*-hard by reducing the subset sum problem to it, which is known to be *NP*-complete [49]. The subset sum problem is defined as follows.

Definition 4.1 (subset sum)**.** Given numbers $s_1, \ldots, s_n \in \mathbb{Z}$ and $S \in \mathbb{Z}$, is there a subset $I \subseteq \{1, \ldots, n\}$ such that the sum $\sum_{i \in I} s_i$ is exactly S?

Theorem 4.1. *The unconstrained two-scenario optimization problem* (4.2) *is* NP*-hard, even for the binary case, when* $0 \leq x_i \leq 1$ *for* $i \in \{1, \ldots, n\}$.

Proof. Obviously the decision version of problem (4.2) is in the class *NP*, since checking whether the value of a feasible solution $\bar{x}$ is below a given bound B can be done in linear time. To show that the problem is also *NP*-hard, we describe a polynomial reduction of the subset sum problem to (4.2). Given a subset sum instance $(s, S) \in \mathbb{Z}^{n+1}$ of size n, construct an instance of (4.2) by setting $a = s$, $b = -s$, $a_0 = -S$ and $b_0 = S$ and setting all variable lower bounds to zero and all upper bounds to one. We have

$$\begin{aligned} \min_{x \in \{0,1\}^n} \max\{a^\top x + a_0, b^\top x + b_0\} &= \min_{x \in \{0,1\}^n} \max\{s^\top x - S, -s^\top x + S\} \\ &= \min_{x \in \{0,1\}^n} |s^\top x - S| \end{aligned}$$

If the optimal solution value of this problem is 0, an optimal solution $x^\star$ defines the subset I that solves the subset sum instance. If the optimal value is nonzero, there is no I such that $\sum_{i \in I} s_i = S$. Consequently, problem (4.2) is at least as hard as the subset sum problem. □

4.1.2 Transformation to Fractional Knapsack Problems

In the following we present a method to efficiently compute lower bounds of problem (4.2), which can be embedded into a branch and bound-algorithm to compute exact solutions of unconstrained two-scenario problems. Starting from the nonlinear formulation (4.2), we first rewrite the unconstrained two-scenario problem as two integer linear problems with a simple structure, which can be solved independently. The LP-relaxations of these ILPs are then transformed into fractional knapsack problems, which allows us to compute lower bounds of the original problem with purely combinatorial algorithms.

For two functions $f, g\colon \{0,1\}^n \longrightarrow \mathbb{R}$ the function $h(x) = \max\{f(x), g(x)\}$ can be written as

$$h(x) = \begin{cases} f(x) \text{ if } f(x) \geq g(x) \\ g(x) \text{ otherwise.} \end{cases}$$

The minimizer $x^\star$ of h can thus be determined by computing the minimizers $x_1^\star$ and $x_2^\star$ of f and g resp. on the domains specified in the case distinction above and choosing $x^\star = \min\{x_1^\star, x_2^\star\}$.

For (4.2) we have

$$\begin{aligned}
&\min_{\substack{l \leq x \leq u \\ x \in \mathbb{Z}^n}} \max\{a^\top x + a_0, b^\top x + b_0\} \\
&= \min \left\{ \begin{array}{lll} \min & a^\top x_1 + a_0 & \\ \text{s.t.} & a^\top x_1 + a_0 \geq b^\top x_1 + b_0 & \\ & l \leq x_1 \leq u & \\ & x_1 \in \mathbb{Z}^n & \end{array} \right. , \left. \begin{array}{lll} \min & b^\top x_2 + b_0 & \\ \text{s.t.} & b^\top x_2 + b_0 \geq a^\top x_2 + a_0 & \\ & l \leq x_2 \leq u & \\ & x_2 \in \mathbb{Z}^n & \end{array} \right\} \\
&= -\max \left\{ \begin{array}{lll} \max & -a^\top x_1 - a_0 & \\ \text{s.t.} & (b-a)^\top x_1 \leq a_0 - b_0 & \\ & l \leq x_1 \leq u & \\ & x_1 \in \mathbb{Z}^n & \end{array} \right. , \left. \begin{array}{lll} \max & -b^\top x_2 - b_0 & \\ \text{s.t.} & (a-b)^\top x_2 \leq b_0 - a_0 & \\ & l \leq x_2 \leq u & \\ & x_2 \in \mathbb{Z}^n & \end{array} \right\} \qquad (4.3)
\end{aligned}$$

Both subproblems are integer linear optimization problems with a single constraint. Their LP-relaxations can be solved as fractional knapsack problems with continuous weights and profits. The fractional knapsack problem is a variant of the *knapsack problem* (KP) [74].

Definition 4.2 ((KP))**.** Given a set A of n items with profits $p \in \mathbb{Z}^n$, nonnegative weights $w \in \mathbb{Z}^n_{\geq 0}$ and a capacity $C \in \mathbb{Z}$, choose a subset $S \subseteq A$ such that the sum of the weights of the items in S does not exceed the capacity C and the sum of the profits is maximum.

KP can be expressed in linear programming terms by associating a binary variable

x with each item in A:

$$\begin{aligned} \max_{x\in\{0,1\}^n} \quad & p^\top x \\ \text{s.t.} \quad & w^\top x \leq C \end{aligned} \tag{4.4}$$

In an optimal solution $x^\star$ of (4.4) the indices i with $x_i^\star = 1$ define the optimal set S.

When the definition of KP is changed to allow partial items, the problem is known as the fractional knapsack problem (FKP).

Definition 4.3 (FKP)**.** For p in $\mathbb{R}^n$, $w \in \mathbb{R}^n$ and $C \in \mathbb{R}$ the fractional knapsack problem (FKP) is defined as

$$\begin{aligned} \max \quad & p^\top x \\ \text{s.t.} \quad & w^\top x \leq C \\ & 0 \leq x \leq 1. \end{aligned} \tag{4.5}$$

The knapsack problems is known to be *NP*-hard, but it is solvable in pseudo-polynomial time by dynamic programming. However, the fractional knapsack problem can be solved in polynomial time with a simple greedy algorithm [89]. It works by first sorting the items by their profit to weight-ratio p_i/w_i and then adding items to the set S in sorted order until adding another item would exceed the capacity limit. The first item that is not added is called the *critical item*. As was shown by Dantzig [25], adding a fraction of the critical element to S such that the capacity limit is reached exactly produces an optimal solution. The running time of this algorithm is $\mathrm{O}(n \log n)$, if an appropriate sorting algorithm is used.

Expressing the subproblems in (4.3) in the form of (4.5) takes two steps:

1. The variables in Problem (4.5) are required to take values between zero and one. The transformation from a bounded variable $l \leq x \leq u$ with $l < u$ to a box-constrained variable $0 \leq \bar{x} \leq 1$ is given by

   $$\bar{x} = (u - l)^{-1}x - l(u - l)^{-1}.$$

   Solving for x gives the substitution

   $$x = (u - l)\bar{x} + l.$$

2. The weights w in the constraint are assumed to be nonnegative. Replace those transformed variables $\bar{x}_i$ with negative weights by their complement $(1 - \bar{x}_i)$.

After applying the two substitutions, the subproblems have the form

$$\begin{aligned} \max_{x^1\in[0,1]^n} \quad & \sum_{i\in N_0\cup N_2} a_i(l_i-u_i)x_i^1 + \sum_{i\in N_1}(1-x_i^1)a_i(l_i-u_i) - \sum_{i=1}^{n} a_i l_i - a_0 \\ \text{s.t.} \quad & \sum_{i=1}^{n} |b_i-a_i|(u_i-l_i)x_i^1 \le \sum_{i\in N_1}(a_i-b_i)(u_i-l_i) + \sum_{i=1}^{n}(a_i-b_i)l_i + a_0 - b_0 \end{aligned} \tag{4.6}$$

and

$$\begin{aligned} \max_{x^2\in[0,1]^n} \quad & \sum_{i\in N_0\cup N_1} b_i(l_i-u_i)x_i^2 + \sum_{i\in N_2}(1-x_i^2)b_i(l_i-u_i) - \sum_{i=1}^{n} b_i l_i - b_0 \\ \text{s.t.} \quad & \sum_{i=1}^{n} |a_i-b_i|(u_i-l_i)x_i^2 \le \sum_{i\in N_2}(b_i-a_i)(u_i-l_i) + \sum_{i=1}^{n}(b_i-a_i)l_i + b_0 - a_0 \end{aligned} \tag{4.7}$$

where $N_0 = \{i \in \{1\ldots,n\} \mid b_i - a_i = 0\}$, $N_1 = \{i \in \{1\ldots,n\} \mid b_i - a_i < 0\}$ and $N_2 = \{i \in \{1\ldots,n\} \mid a_i - b_i < 0\}$.

The resulting problems (4.6) and (4.7) are fractional knapsack problems. By transforming the optimal solutions of (4.6) and (4.7) back into the original variable space we obtain dual (upper) bounds for the two maximization subproblems in (4.3). Taking the negative of the larger of these values gives a lower bound on the optimum value of (4.2).

Remark 4.1. Instead of transforming the LP-relaxations of the subproblems in (4.3) it is possible to express the two subproblems directly as instances of the *bounded knapsack problem* (BKP), which can then be transformed into binary knapsack problems as defined in this chapter. For details see [89, Chapter 3]. As mentioned, KP is *NP*-hard, but it can be solved in pseudo-polynomial time by dynamic programming if the coefficients are integer. Thus the two-scenario optimization problem with integer coefficients a, a_0, b and b_0 is also solvable in pseudo-polynomial time. In the general case considered here, the coefficients would need to be scaled to integer numbers first. This approach has two drawbacks: First, the rounding leads to an inexact solution. Second, the running times of the dynamic programming algorithms are only pseudo-polynomial, i.e., they depend on the size of the coefficients. Scaling the coefficients to the appropriate accuracy would lead to excessive running times.

4.1.3 An Exact Algorithm

The approach presented in the previous subsection only gives lower bounds on the optimal value of the problem. To compute an optimal integer solution, we embed it into a branch and bound-framework. In each node of the branch and bound-tree

we compute a lower bound as detailed above. Since the original problem does not have any constraints besides integrality, we can generate a feasible solution from each solution of the relaxation by rounding to the next integer.

We branch by fixing variables to integer values. To produce correct bounds in subproblems with fixed variables the greedy algorithm has to be adapted slightly to respect the fixings. This can be done in the following way: First consider all variables that are fixed. In the solution of the knapsack problem set the variable values to the transformed values of the fixings and decrease the knapsack capacity accordingly. If the capacity is exceeded, the knapsack problem is infeasible, otherwise proceed with the remaining elements in sorted order.

As the branching variable we choose the fractional variable from the knapsack problem that defined the lower bound. Note that we can have at most one fractional variable by construction of the algorithm.

4.2 Combinatorial Two-Scenario Optimization

In this section we add combinatorial side constraints to the two-scenario linear objective function. The transformation to two knapsack problems in this case cannot be applied directly anymore. Using the Lagrangean decomposition approach described in 1.2.1, we separate the objective function from the combinatorial constraints and obtain two subproblems. The first is an unconstrained two-scenario problem, the second the linear version of the combinatorial problem. This approach is analogous to the treatment of submodular combinatorial optimization problems in Section 3.5.

Let the feasible solutions of the combinatorial problem be given by the set $X \subseteq \{0,1\}^n$. The computational two-scenario problem can then be stated as

$$\begin{array}{ll} \min\limits_{x} & \max\{a^\top x + a_0, b^\top x + b_0\} \\ \text{s.t.} & x \in X\,. \end{array} \tag{4.8}$$

In the following we assume that the linear version version of the combinatorial problem can be solved in polynomial time. More precisely, we assume that a combinatorial algorithm with polynomial running time is known and/or a complete characterization of the polytope corresponding to the set X is known and efficient separation algorithms exist.

4.2.1 Lower Bounds

Consider the isomorphic linear formulation of (4.8):

$$\begin{array}{lrcl} \min & z & & \\ \text{s.t.} & z & \geq & a^\top x + a_0 \\ & z & \geq & b^\top x + b_0 \\ & x & \in & X \\ & z & \in & \mathbb{R} \end{array} \tag{4.9}$$

Assume that a complete polyhedral description of the set X is known. In this case we can write $x \in \text{conv}(X)$ instead of $x \in X$ and (4.9) is equivalent to

$$\begin{array}{lrcl} \min & z & & \\ \text{s.t.} & z & \geq & a^\top x + a_0 \\ & z & \geq & b^\top x + b_0 \\ & x & \in & \text{conv}(X) \\ & z & \in & \mathbb{R} \\ & x & \in & \{0,1\}^n\,. \end{array} \tag{4.10}$$

We have seen in Chapter 1 that the bounds for (4.8) obtained by Lagrangean decomposition are at least as good as the LP bound, i.e. the bound obtained by dropping all integrality conditions in (4.10). There are two natural ways to decompose (4.10) into an unrestricted non-linear problem and a linear combinatorial problem. Depending on whether the integrality conditions are imposed on the original or the artificial variables, the two decompositions are the following:

$$\begin{array}{lll} \min\limits_{x\in\{0,1\}^n} \max\{a^\top x + a_0, b^\top x + b_0\} + \lambda^\top x & + & \min\limits_{y} \; -\lambda^\top y \\ & & \text{s.t.} \quad y \in \text{conv}(X) \\ = \min\limits_{x\in\{0,1\}^n} \max\{(a+\lambda)^\top x + a_0, (b+\lambda)^\top x + b_0\} & + & \min\limits_{y} \; -\lambda^T y \\ & & \text{s.t.} \quad y \in \text{conv}(X) \end{array} \tag{4.11}$$

and

$$\begin{array}{lll} \min\limits_{0\leq x\leq 1} \max\{a^\top x + a_0, b^\top x + b_0\} + \lambda^\top x & + & \min\limits_{y} \; -\lambda^\top y \\ & & \text{s.t.} \quad y \in \text{conv}(X) \\ = \min\limits_{0\leq x\leq 1} \max\{(a+\lambda)^\top x + a_0, (b+\lambda)^\top x + b_0\} & + & \min\limits_{y} \; -\lambda^T y \\ & & \text{s.t.} \quad y \in \text{conv}(X)\,, \end{array} \tag{4.12}$$

where λ is the vector of Lagrangean multipliers. We see that the relaxation (4.11) is potentially stronger than (4.12), since the former does not have the integrality

property. By Theorem 1.1, these decompositions give the following dual bounds:

$$\begin{aligned} v_1 = \min \quad & z \\ \text{s.t.} \quad & y = x \\ & (x, z, y)^\top \in \operatorname{conv}\{x \in \mathbb{Z}^n, z \in \mathbb{R}, y \geq 0 \mid \\ & \qquad\qquad z \geq a^\top x + a_0, z \geq b^\top x + b_0, y \in \operatorname{conv}(X)\} \\ = \min \quad & z \\ \text{s.t.} \quad & y = x \\ & (x, z)^\top \in \operatorname{conv}\{x \in \mathbb{Z}^n, z \in \mathbb{R} \mid z \geq a^\top x + a_0, z \geq b^\top x + b_0\} \\ & y \in \operatorname{conv}(X) \end{aligned}$$

and

$$\begin{aligned} v_2 = \min \quad & z \\ \text{s.t.} \quad & y = x \\ & (x, z)^\top \in \operatorname{conv}\{x \geq 0, z \in \mathbb{R}, y \in \mathbb{Z}^n \mid z \geq a^\top x + a_0, z \geq b^\top x + b_0\} \\ & y \in \operatorname{conv}(X) \end{aligned}$$

To compute the optimal values of the two relaxations with the subgradient method, we have to solve (4.11) or (4.12) for a fixed value of λ in every iteration. Due to the decomposition, the two subproblems can be solved separately.

For the first subproblem of (4.11), the unconstrained integer two-scenario problem, we presented an exact branch and bound-algorithm algorithm in the previous section. Solving an integer two-scenario problem to optimality with a branch and bound-algorithm in each iteration of a subgradient method is expensive, but as we have the bound v_1 is potentially stronger than the bounds obtained by solving (4.12) or the LP-relaxation of (4.10).

Since in the left part of (4.12) solutions need not be integral, the two corresponding fractional knapsack problems can be solved to optimality with the fractional knapsack-algorithm by Dantzig [25]. The bound v_2 is the same as the LP bound. Still, using relaxation (4.12) might be advantageous, especially when the polyhedral description of the set X is not compact. In this case solving the LP-relaxation of (4.10) to optimality would require separating cutting planes and solving a series of linear programs. Under our assumption that an efficient algorithm for the linear combinatorial subproblem in the Lagrangean decomposition is available, computing the Lagrangean dual of (4.12) might be faster in practice than solving the LP-relaxation of (4.10) with a cutting plane-algorithm.

4.2.2 An Exact Algorithm

The two decompositions just described and their corresponding Lagrangean duals give rise to two exact algorithms for combinatorial two-scenario problems. As in the previous section we embed the computation of lower bounds into a branch

and bound-algorithm. In each node of the branch and bound-tree the Lagrangean dual is computed with a subgradient algorithm.

As in the case of submodular objective functions discussed in Section 3.5, when selecting the branching variable we only consider variables which have different values in the solutions of the two subproblems corresponding to the optimal Lagrangean multiplier $\lambda^\star$. Among those one can again choose the one with the lowest index or the one with the largest absolute value of the multiplier, depending on the application. If no such variable exists, the solution is primal feasible for the subproblem.

For the use in a branch and bound-algorithm both the method for computing the unconstrained two-scenario problem and the combinatorial algorithm have to be adapted to deal with fixed variables. For the fractional knapsack algorithm used for relaxation (4.11) the adaptation is the same as in Section 4.1.3. For (4.12) the branch and bound-algorithm described in 4.1.3 has to be modified to start with a set of fixed variables, since it is called in each node. The combinatorial algorithms used can be adapted in several ways, for example by adding a preprocessing step as in the computation of the fractional knapsack problem, by modifying the structure of the underlying graph or by modifying the weights of the graph.

Remark 4.2. For any value of λ, the solution of the combinatorial part of the decomposition is also feasible for the subproblem. Evaluating these solutions in the original objective function yields a primal bound for the original problem.

Both exact algorithms, for the unconstrained two-scenario problem and for its combinatorial variant, will be evaluated experimentally in Chapter 9. We will compare their performance to an LP-based linearization approach. We chose spanning trees in undirected graphs as the combinatorial structure of the constraint set.

4.3 Two-Scenario Min-Max Regret Problems

A problem closely related to the multi-scenario problem we studied in this chapter is the so-called *min-max regret problem*. Recall the basic combinatorial optimization problem with k scenarios (4.1):

$$\begin{aligned} \min_{x} \quad & \max\{f_1(x), \ldots, f_k(x)\} \\ \text{s.t.} \quad & Cx \leq d \\ & l \leq x \leq u \\ & x \in \mathbb{Z}^n \end{aligned}$$

The aim here is to find the best worst-case value across all scenarios. In terms of robustness this is a very conservative approach, since it does not take into

account the other possible scenarios [3]. When the min–max regret formulation of a problem is used instead the maximum deviation from the optimum solution in each scenario, the so-called *maximum regret* is minimized.

Denote by opt_i the optimum solution of (4.1) when only scenario i is considered. The regret $r(x, i)$ of a solution x for scenario i is defined as

$$r(x, i) = f_i(x) - opt_i \,,$$

that is the amount of additional cost incurred by not having chosen the optimum solution for this scenario. In the min-max regret formulation the maximum regret over all scenarios is minimized:

$$\begin{array}{ll} \min\limits_{x} & \max\{f_1(x) - opt_1, \ldots, f_k(x) - opt_k\} \\ \text{s.t.} & Cx \leq d \\ & l \leq x \leq u \\ & x \in \mathbb{Z}^n \end{array} \tag{4.13}$$

Using this formulation does not protect against the worst case as effectively as (4.1), but gives an increase of the average performance aver all scenarios [3].

The approaches we presented in this chapter can be easily adapted to also handle the regret-versions of two-scenario problems. Note that problems (4.1) and (4.13) only differ in the constant terms opt_i in the objective function. These values can be determined in advance by solving the appropriate linear (one-scenario) variants of the problem for the two scenarios. By our assumption this can be done efficiently, so the complexity does not change.

Part II

Applications

Chapter 5

Tanglegrams

In this chapter we study the *tanglegram layout problem*. We first show that computing an optimum tanglegram layout can be modeled as a binary quadratic optimization problem with side constraints. Then we use the approaches for binary quadratic optimization discussed in Chapter 2 to solve random and realistic tanglegram instances and compare their performance to the standard linearization approach and an algorithm based on semidefinite programming.

The problem of computing optimal tanglegram layouts arises in several areas of computational biology as well as in hierarchical clustering. A tanglegram consists of two trees T_1 and T_2 and a set of additional edges. These tangle edges link leaf nodes of T_1 with leaf nodes of T_2. A tanglegram layout is a drawing of the resulting graph that places the leaf sets on two parallel lines and preserves the tree structures. In an optimal tanglegram layout the number of crossings between tangle edges is minimal.

One application of tanglegrams is the comparison of phylogenetic trees in computational biology. Here the leafs represent a set of species, the inner nodes potential ancestors. Each tree represents a hypothesis of the evolutionary history of the species. A tanglegram layout with a low number of crossings between tangle edges allows an easy visual comparison of two possible evolutionary histories of a species. In this application the two leaf sets are identical. Each leaf of T_1 is linked to its counterpart in T_2 by a tangle edge.

Another application is the analysis of the coevolution of two species, e.g. of a host and its parasites. Given the phylogenetic trees of a family of hosts and a family of parasites that infest these hosts, tangle edges mark which particular parasites infest each host. An optimal tanglegram layout indicates whether the evolution of the host species directly affected the evolution of the parasites, and vice versa. In this case, a low number of crossings between tangle edges is expected. Whereas in the first application the number of tangle edges incident at each leaf is exactly one, here the number can be higher, since each parasite species can infest several

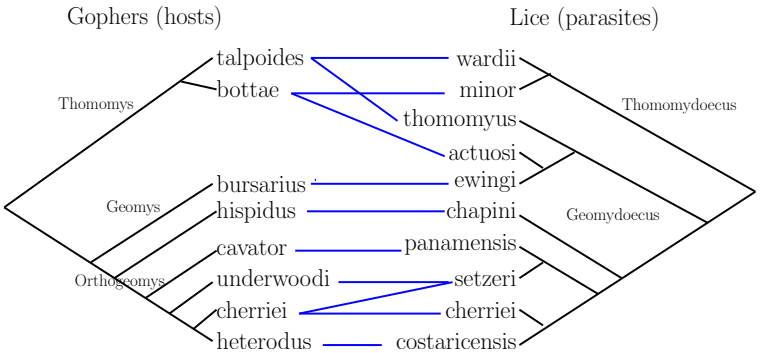

Figure 5.1: A tanglegram of two binary trees, taken from [57]. The maximum number of tangle edges incident to a leaf node in this case is two.

host species. Figure 5.1 shows a tanglegram layout from a paper by Hafner et al. [57], in which the authors study the co-evolution of pocket gophers (the hosts) and chewing lice (the parasites).

Tanglegrams also occur in hierarchical clusterings, which can be visualized by so-called *dendrograms*. Dendrograms consist of trees where the elements to be clustered are identified with the leaves. Internal nodes determine clusters that contain the elements or sub-clusters. A tanglegram layout helps comparing the results of different clustering methods. Moreover, tanglegrams occur when analyzing software projects in which a tree represents package, class, method hierarchies. Hierarchy changes are analyzed over time, or automatically generated decompositions are compared with human-made ones. This application yields tanglegrams on trees that are not binary in general [103]. Figure 5.2 shows the layout of such a general tanglegram.

In this chapter we show that the general *tanglegram layout-problem* can be modeled as a binary quadratic optimization problem, more specifically, a quadratic linear ordering problem with additional constraints. Thus the tanglegram layout-problem can be solved with the ILP-based cutting plane approach described in Chapter 2. Additionally, the linear ordering-model allows a problem-specific reformulation of the linear constraints. In an experimental study, we solve random and realistic tanglegram instances and compare the performance of the ILP algorithm to an SDP-based approach.

This chapter is organized as follows. First, we review the complexity of several variants of the tanglegram layout-problem and give an overview of existing literature on tanglegrams. In Section 5.2, we present the quadratic model. Finally, we give the results of our experimental study in Section 5.3.

5.1 Complexity and Related Work

Fernau et al. [39] showed that deciding whether a tanglegram can be drawn without tangle edge crossings can be done in polynomial time: Direct all edges from the root node of the first tree to the root node of the second tree. If the resulting graph is upward planar, a crossing free tanglegram layout exists. Testing a graph for upward planarity can be done in linear time [14].

The same authors also established the complexity of the restricted variant of the general tanglegram problem that was described as the first application above. In the so-called *two-tree-problem*, two different phylogenetic trees for the same species are considered. The tangle edges model the one-to-one correspondence of the leaf nodes, i.e. each leaf node is incident to exactly one tangle edge. The decision version of this problem is *NP*-hard. The proof given by Fernau et al. [39] works by reducing the maximum cut-problem with unit weights to two-tree in polynomial time.

When one layer order is kept fixed, the problem is called *one-tree*. It can be solved to optimality by a dynamic programming algorithm in $O(n \log^2 n)$ time, where n is the number of leaves [39].

The tanglegram problem is closely related to the *bipartite crossing minimization problem* [73, 71, 18]. Given a bipartite graph $G = (V_1, V_2, E)$, the task here is to place the nodes of V_1 and V_2 on two parallel lines, such that the number

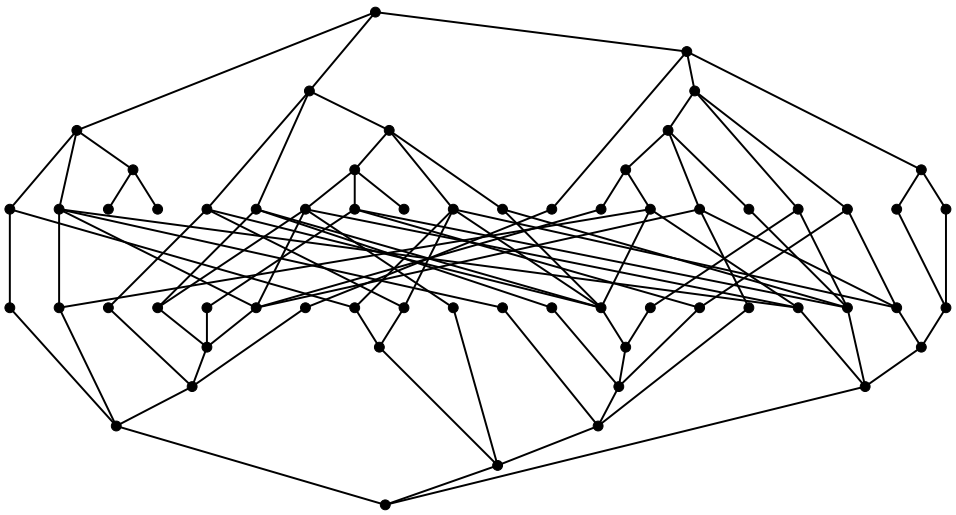

Figure 5.2: A tanglegram layout of two trees with 20 leaves each. Each internal node has at most three children and the density of the tangle edges is 10%.

of crossings between the edges in E is minimized. The edges have to be drawn as straight lines. Obviously bipartite crossing minimization can be considered a special case of the general tanglegram problem in the following sense: In the bipartite crossing minimization problem the order of the nodes on the parallel lines is not restricted. Thus this problem is equivalent to the tanglegram problem where in both trees all leafs are directly linked to the root node by an edge.

Most of the literature is concerned with the case of binary trees and leaves that are in one-to-one correspondence. Whereas several of the presented methods could easily be generalized to arbitrary tangle layers, an extension to non-binary trees is usually not possible. When allowing general trees, one extreme case would be a star, where all leaves are incident to the root node. If both T_1 and T_2 are stars, there are no constraints on the orders of leaves on either shore, so that the problem specializes to the bipartite crossing minimization problem.

We are not aware of any implementation of an exact method for drawing tanglegrams with non-binary trees. As mentioned above, Fernau et al. [40] showed the *NP*-hardness of drawing tanglegrams. They also presented a fixed-parameter algorithm for binary tanglegrams. Recently, an improved fixed-parameter algorithm was presented by Böcker et al. [16] which can solve large binary instances quickly in practice, provided that the number of crossings is not too large. Finally, while in the recent paper by Venkatachalam et al. [121] the focus is on binary instances, a fixed-parameter algorithm for general tanglegram instances is presented. According to our knowledge, this is the only algorithm that could deal with non-binary trees. However, no implementation or running times are provided making it impossible to evaluate its practical performance.

Besides analyzing the performance and quality of several heuristics in a computational study for binary tanglegrams with 1–1 tangles, Nöllenburg et al. [103] also implemented a branch-and-bound algorithm and an exact integer-programming (IP) based approach for this case.

As we will compare our approach with the exact IP-approach of [103], we describe it in more detail in the following. A feasible but not necessarily optimal tanglegram layout is given as an input. For each inner node, a binary variable x_i is introduced. In the case of complete binary trees with n leaves each, this gives rise to $2n - 2$ variables. If $x_i = 1$, the subtree rooted in node i is flipped with respect to the input drawing, otherwise it remains unchanged. As by definition there are no crossings within the trees, the number of crossings can be determined by counting the number of tangle crossings. Denote by L_1 the leafs of T_1 and by L_2 the leafs of T_2. Let (a, c) and (b, d) be tangle edges with $a, b \in L_1$ and $c, d \in L_2$. Let i be the lowest common ancestor of a, b in T_1 and j that of c, d in T_2. If the tangle edges cross each other in the input drawing, then a crossing occurs in the output drawing if and only if either both subtrees below i and j are flipped or both remain unchanged. This can be expressed as $x_i x_j = 1$ or $(1-x_i)(1-x_j) = 1$.

Similarly, if the edges do not cross each other in the input drawing, then there is a crossing in the output drawing if and only if either $(1-x_i)x_j = 1$ or $x_i(1-x_j) = 1$.

Denote by T_1° and T_2° the set of inner nodes of T_1 and T_2, respectively. For each pair of inner nodes $(i,j) \in T_1^\circ \times T_2^\circ$, let $k_{ij}^\times$ be the number of tangle edge pairs that have lowest common ancestors i and j and cross in the initial tanglegram layout, $k_{ij}^=$ be the number of tangle edge pairs that have lowest common ancestors i and j and do not cross in the initial tanglegram layout. The number of crossings in a drawing is then given by

$$\sum k_{ij}^= \left(x_i(1-x_j) + (1-x_i)x_j\right) + \sum k_{ij}^\times \left(x_i x_j + (1-x_i)(1-x_j)\right)$$

Thus minimizing the number of tangle edge crossings reduces to minimizing the sum of the given products. The latter is an instance of the unconstrained quadratic binary optimization problem. In their paper, Nöllenburg et al. [103] apply a custom linearization instead of the standard linearization for binary quadratic programming that was described in Chapter 2. They introduce a new binary variable y_{ij} for each term $x_i(1-x_j) + (1-x_i)x_j$ and add four sets of coupling constraints. The resulting model reads

$$\begin{aligned} \min \quad & \sum_{(i,j)\in T_1^\circ \times T_2^\circ} k_{ij}^= y_{ij} + k_{ij}^\times (1-y_{ij}) \\ \text{s.t.} \quad & \left.\begin{array}{rcl} y_{ij} & \leq & 2 - x_i - x_j \\ y_{ij} & \leq & x_i + x_j \\ y_{ij} & \geq & x_i - x_j \\ y_{ij} & \geq & x_j - x_i \end{array}\right\} \forall (i,j) \in T_1^\circ \times T_2^\circ \\ & x \in \{0,1\}^{T_1^\circ + T_2^\circ} \end{aligned} \tag{5.1}$$

Lemma 5.1. *The model by Nöllenburg et al. [103] is equivalent to the standard linearization of*

$$\min_{x\in\{0,1\}^{T_1^\circ+T_2^\circ}} \sum_{(i,j)\in T_1^\circ + T_2^\circ} k_{ij}^= \left(x_i(1-x_j) + (1-x_i)x_j\right) + k_{ij}^\times \left(x_i x_j + (1-x_i)(1-x_j)\right) ,$$

which is

$$\begin{aligned} \min \quad & \sum_{(i,j)\in T_1^\circ \times T_2^\circ} k_{ij}^=(x_i + x_j - 2z_{ij}) + k_{ij}^\times (2z_{ij} - x_i - x_j + 1) \\ s.t. \quad & \left.\begin{array}{rcl} z_{ij} & \geq & x_i + x_j - 1 \\ z_{ij} & \geq & 0 \\ z_{ij} & \leq & x_i \\ z_{ij} & \leq & x_j \end{array}\right\} \forall (i,j) \in T_1^\circ \times T_2^\circ \\ & x \in \{0,1\}^{T_1^\circ + T_2^\circ} \end{aligned} \tag{5.2}$$

Proof. Consider the linear transformation

$$\pi \colon y_{ij} \mapsto x_i + x_j - 2z_{ij} \,, x_i \mapsto x_i \,.$$

We will show that under this transformation each feasible point of (5.1) is mapped to a feasible point of (5.2), and vice versa.

Let $(\bar{x}, \bar{y})$ be a feasible point for (5.1) and $\pi(\bar{x}, \bar{y}) = (\bar{x}, \bar{z})$ its transformation. By definition of $(\bar{x}, \bar{y})$ and π we have for all $(i, j) \in T_1^\circ \times T_2^\circ$:

$$\begin{aligned}
&\bar{y}_{ij} \leq 2 - \bar{x}_i - \bar{x}_j \Rightarrow \bar{x}_i + \bar{x}_j - 2\bar{z}_{ij} \leq 2 - \bar{x}_i - \bar{x}_j \Rightarrow \bar{z}_{ij} \geq \bar{x}_i + \bar{x}_j - 1\,,\\
&\bar{y}_{ij} \leq \bar{x}_i + \bar{x}_j \Rightarrow \bar{x}_i + \bar{x}_j - 2\bar{z}_{ij} \leq \bar{x}_i + \bar{x}_j \Rightarrow \bar{z}_{ij} \geq 0\,,\\
&\bar{y}_{ij} \geq \bar{x}_i - \bar{x}_j \Rightarrow \bar{x}_i + \bar{x}_j - 2\bar{z}_{ij} \geq \bar{x}_i - \bar{x}_j \Rightarrow \bar{z}_{ij} \leq \bar{x}_j \text{ and}\\
&\bar{y}_{ij} \leq \bar{x}_j - \bar{x}_i \Rightarrow \bar{x}_i + \bar{x}_j - 2\bar{z}_{ij} \leq \bar{x}_j - \bar{x}_i \Rightarrow \bar{z}_{ij} \leq \bar{x}_i\,,
\end{aligned}$$

which shows that $(\bar{x}, \bar{z})$ is feasible for (5.2). To show that any feasible point for (5.2) is also feasible for (5.1), we apply the inverse transformation

$$\pi^{-1}\colon z_{ij} \mapsto \frac{1}{2}(x_i + x_j - y_{ij})\,, x_i \mapsto x_i\,.$$

Let $(\bar{x}, \bar{z})$ be a feasible point for (5.2) and $\pi^{-1}(\bar{x}, \bar{z}) = (\bar{x}, \bar{y})$ its transformation. By definition of $(\bar{x}, \bar{z})$ and π^{-1} this time we have for all $(i, j) \in T_1^\circ \times T_2^\circ$:

$$\begin{aligned}
&\bar{z}_{ij} \geq \bar{x}_i + \bar{x}_j - 1 \Rightarrow \frac{1}{2}(\bar{x}_i + \bar{x}_j - \bar{y}_{ij}) \geq \bar{x}_i + \bar{x}_j - 1 \Rightarrow \bar{y}_{ij} \leq 2 - \bar{x}_i - \bar{x}_j\,,\\
&\bar{z}_{ij} \geq 0 \Rightarrow \frac{1}{2}(\bar{x}_i + \bar{x}_j - \bar{y}_{ij}) \geq 0 \Rightarrow \bar{y}_{ij} \leq \bar{x}_i + \bar{x}_j\,,\\
&\bar{z}_{ij} \leq \bar{x}_i \Rightarrow \frac{1}{2}(\bar{x}_i + \bar{x}_j - \bar{y}_{ij}) \leq \bar{x}_i \Rightarrow \bar{y}_{ij} \geq \bar{x}_j - \bar{x}_i \text{ and}\\
&\bar{z}_{ij} \leq \bar{x}_j \Rightarrow \frac{1}{2}(\bar{x}_i + \bar{x}_j - \bar{y}_{ij}) \leq \bar{x}_j \Rightarrow \bar{y}_{ij} \geq \bar{x}_i - \bar{x}_j\,.
\end{aligned}$$

Furthermore, the transformation maps the objective functions to each other. This means that both models have the same dimension and the sets of feasible and optimal solutions are isomorphic. □

While Nöllenburg et al. used this model only for instances with 1–1 tangles, they briefly note that it could be extended to leaves of higher degree as well. However, their model cannot be generalized to instances with non-binary trees in a straightforward way, since by construction the binary variables x_i can only model flips of complete subtrees, but not arbitrary permutations in the order of the children of a node. In many applications, the trees are not necessarily binary. In the next section we will present an exact model for tanglegrams that neither restricts the degree of inner nodes in the trees nor the number of tangles incident to a leaf.

5.2 An Exact Model for General Tanglegrams

The problem of drawing tanglegrams is closely related to bipartite crossing minimization. As argued above, the latter problem can be considered a special case of

the former. Therefore, we first review approaches for drawing bipartite graphs.

5.2.1 Bipartite Crossing Minimization

Let $G = (V_1 \cup V_2, E)$ be a bipartite graph. The task is to draw G with straight line edges. The nodes in V_1 and V_2 have to be placed on parallel lines H_1 and H_2 such that the number of edge crossings is minimal. Both heuristic and exact methods [71] exist for this problem.

Assume for a moment that the nodes on the first layer H_1 are fixed, and only the nodes on layer H_2 are permuted. For each pair of nodes on H_2, we introduce a variable x_{uv} such that $x_{uv} = 1$ if u is drawn to the left of v and $x_{uv} = 0$ otherwise. For edges (i,k) and (j,l) with $i,j \in H_1$ and $k,l \in H_2$, such that i is left of j, a crossing exists if and only if l is left of k. We thus have to punish x_{lk} in the objective function. The task of minimizing the number of crossings is now equivalent to determining a minimum linear ordering on the nodes of H_2. Exploiting $x_{uv} = 1 - x_{vu}$, we can eliminate half of the variables and only keep those with $u < v$. Note that bipartite crossing minimization with one fixed layer is already *NP*-hard [32].

If the nodes on both layers are allowed to permute, the number of crossings depends on the order of the nodes on each layer. Therefore, the problem can be modeled as a quadratic optimization problem over linear ordering variables. We write the quadratic linear ordering problem (QLO) in its general form as

$$
\begin{array}{lll}
 & \min & \sum_{(i,j,k,l)\in I} c_{ijkl} x_{ij} x_{kl} \\
(QLO) & \text{s.t.} & x \in P_{LO} \\
 & & x_{ij} \in \{0,1\} \quad \forall (i,j) \in J
\end{array}
$$

where P_{LO} is the linear ordering polytope [53, 108]. The index set I consists of all quadruples (i,j,k,l) such that $x_{ij}x_{kl}$ occurs as a product in the objective function, while J is the set of all pairs (i,j) for which a linear ordering variable x_{ij} is needed. The linear ordering polytope P_{LO} can be modeled using the so-called 3-dicycle inequalities, which enforce transitivity in the ordering of the nodes. We have

$$P_{LO} = \text{conv}\{x \in \{0,1\}^J \mid 0 \le x_{ij} + x_{jk} - x_{ik} \le 1 \quad \forall (i,j),(j,k),(i,k) \in J\}$$

For the bipartite crossing minimization case, I and J are given as

$$
\begin{aligned}
I &= \{(i,j,k,l) \mid i,j \in H_1,\ i<j, \text{ and } k,l \in H_2,\ k<l\} \\
J &= \{(i,j) \mid i,j \in H_1 \text{ or } i,j \in H_2,\ i<j\}
\end{aligned}
$$

In order to linearize the objective function, we introduce a new binary variable y_{ijkl} for each $(i,j,k,l) \in I$, modeling the product $x_{ij}x_{kl}$. Applying the standard linearization, the corresponding linearized quadratic linear ordering problem

(LQLO) can be written as

$$
\begin{array}{lll}
 & \min & \sum_{(i,j,k,l)\in I} c_{ijkl} y_{ijkl} \\
(LQLO) & \text{s.t.} & x \in P_{LO} \\
 & & x_{ij} \in \{0,1\} \qquad \forall (i,j) \in J \\
 & & y_{ijkl} \leq x_{ij}, x_{kl} \qquad \forall (i,j,k,l) \in I \\
 & & y_{ijkl} \geq x_{ij} + x_{kl} - 1 \quad \forall (i,j,k,l) \in I \\
 & & y_{ijkl} \in \{0,1\} \qquad \forall (i,j,k,l) \in I.
\end{array}
$$

In [73], the above model was introduced for bipartite crossing minimization. Additionally, a quadratic reformulation of the constraints defining P_{LO} was given in [18]: it was shown that a 0/1 vector (x, y) satisfying $y_{ijkl} = x_{ij}x_{kl}$ is contained in (LQLO) if and only if

$$x_{ik} - y_{ijik} - y_{ikjk} + y_{ijjk} = 0 \ \forall (i,j,k,l) \in I. \tag{5.3}$$

Furthermore, the constraints (5.3) yield a minimum equation system for (LQLO). Note that (LQLO) is a quadratic binary optimization problem where the feasible solutions need to satisfy further side constraints, namely those restricting the set of feasible solutions to linear orderings. As unconstrained binary quadratic optimization is equivalent to the maximum cut problem (see Chapter 2), the task is to intersect a cut polytope with a set of hyperplanes.

In general, the convex hull of the corresponding feasible incidence vectors has a structure that is very different from that of a cut polytope. In the above context, however, it was shown in [18] that the hyperplanes (5.3) cut out faces of the cut polytope. Exploiting this result, both IP- and SDP-based methods originally designed for maximum cut problems were used to solve the quadratic linear ordering problem. It turned out that the SDP-based approach outperformed the IP-based techniques for dense graphs, which was also observed in [20]. For sparse graphs it is suggested in [20] to use IP-based methods.

5.2.2 Modeling Tanglegrams

Crossing minimization in tanglegrams can be seen as a generalization of bipartite crossing minimization. The set of feasible orderings is implicitly restricted by the given tree structures. Starting from the model discussed above, we formalize these restrictions as follows: let us consider a triple of leaves a, b, c in one of the trees, say T_1. In case all pairwise lowest common ancestors coincide, all relative orderings between a, b, and c are feasible. However, if the lowest common ancestor of, say, a and b is on a lower level than that of, say, a and c (in this case, the former is a descendant of the latter), then c must not be placed between a and b, as an intra-tree crossing would be induced; see Figure 5.3.

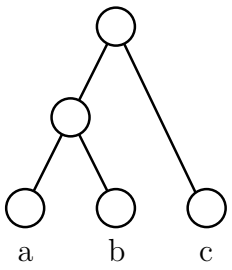

Figure 5.3: Leaf c is not allowed to lie between a and b.

Therefore, we derive a betweenness restriction for every triple of leaves such that two of the leaf pairs have different lowest common ancestors. Each such betweenness restriction of the form 'c cannot be placed between a and b' can be written in linear ordering variables as $x_{ac}x_{cb} = 0$ and $x_{ca}x_{bc} = 0$. In the linearized model (LQLO), the latter amounts to requiring

$$y_{accb} = 0 \;\text{ and }\; y_{cabc} = 0\ . \tag{5.4}$$

For binary trees with n leaves each, all triples of leaves have two different lowest common ancestors, so in this case the number of additional equations is $2\binom{n}{3}$.

In summary, we now obtain a quadratic linear ordering problem (QLO) on a smaller number of variables, with additional constraints of the form (5.4), where

$$\begin{aligned} J &= \{(i,j) \mid i,j \text{ are leaves of the same tree, } i<j\} \\ I &= \{(i,j,k,l) \mid (i,j),(k,l) \in J \text{ belong to different trees}\}\ . \end{aligned}$$

For complete binary trees with n leaves each, the total number of linear ordering variables is $2\binom{n}{2}$. The same number of variables is necessary in the corresponding bipartite crossing minimization model [18].

As mentioned above, the polytope corresponding to the linearized problem (LQLO) is isomorphic to a face of a cut polytope [18]. Since all y-variables are binary, constraints of the form (5.4) are always face-inducing for (LQLO). In summary, we derive the following result:

Theorem 5.2. *The problem of drawing tanglegrams with a minimum number of edge crossings can be solved by optimizing over a face of a suitable cut polytope.*

5.2.3 Binary Case

In the binary case, the model introduced in the last sections is closely related to the model presented in [103]. To see this, first observe that the two equations (5.4) can be written as

$$x_{ab} = x_{bc}\ . \tag{5.5}$$

Table 5.1: Average CPU time in seconds for realistic 1–1 binary trees having n leaves each [103]. Instances are grouped by their number of leaves. The nine largest instances with up to 540 leaves could not all be computed within the time and memory constraints and are omitted.

n	SDP	std	ref	std+cyc	ref+cyc
0–49	<1	<1	<1	<1	14
50–99	3	<1	2	<1	428
100–149	31	<1	7	<1	1100
150–199	125	1	32	1	1282
200–249	437	1	66	2	2704
250–299	4786	2	2529	7	10602
300–349	6483	2	80	9	8862
400–449	20508	12	12067	69	14931

This replacement does not affect the set of feasible solutions, even the corresponding LP-relaxations of (LQLO) are equivalent. Note however that introducing the y-variables allows to strengthen the model, see Theorem 5.2.

When using the linear equations (5.5) instead of the quadratic equations (5.4), we end up with a set of equivalence classes of linear ordering variables, such that all pairwise orderings corresponding to variables in the same class can only be flipped simultaneously. Two variables x_{ac} and x_{bd} belong to the same class if and only if there is a node r such that a, b and c, d are descendants of different children of r; see Figure 5.4. In the binary case, a class of linear ordering variables thus corresponds to the decision of flipping the children of node r or not, which is modeled explicitly by a single variable in the model of Nöllenburg et al. [103].

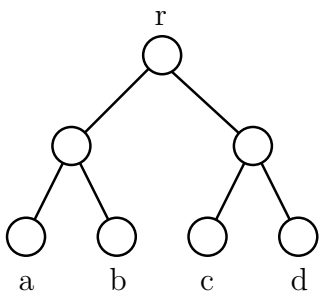

Figure 5.4: Variables x_{ac} and x_{bd} can be identified.

However, in the general case where node r has k children, there are $k!$ different orderings. As these cannot be modeled by a single binary variable, the model of Nöllenburg et al. [103] cannot be applied here.

5.3 Computational Results

We implemented the model explained in Section 5.2.2. Instead of adding equations (5.5) explicitly, we used one variable for each equivalence class of linear ordering variables, thereby significantly reducing the number of variables.

We compare four LP-based branch and bound-approaches and one that solves SDP-relaxations. The first two solve the linearized model (LQLO) and the quadratic reformulation (5.3), respectively. They are pure branch and bound-algorithms. The next two solve (LQLO) and (5.3) as before, but now odd cycle inequalities are separated with the exact algorithm discussed in Chapter 2. The last approach is described in a paper by Rendl et al. [110]. It solves the SDP-model of (5.3) with a branch and bound-algorithm and uses triangle inequalities to strengthen the relaxations. An implementation of the algorithm was made available to us by the authors of [110].

For comparison, we also implemented the IP approach [103] that only works for binary tanglegrams. For the tested binary instances, the running times for solving the latter are very comparable to the model proposed here and are omitted in the following. This behavior can be expected since our model generalizes [103], as discussed in 5.2.3.

We generated random instances on general binary, ternary and quad trees. I.e., the degree of each internal node is at most 2, 3 or 4, respectively. Each tree has n leaves, either having 1–1 tangles or a certain tangle-edge density $d\%$. Instances are generated following the description in [103], with obvious extensions to the more general cases considered here. Finally, we solved realistic binary tanglegram instances from [103] arising in applications in biology and general realistic instances from visualizing software hierarchies [63].

Average results are always computed over 5 randomly generated instances. For each instance, we imposed an upper limit of 10h of CPU time. Instances that could not be solved within this limit count with 10h in the averages. Runs were performed on Intel Xeon machines with 2.33GHz and all linear programs were solved with CPLEX 11.2 [65].

In Table 5.1, we present the average CPU time in seconds for realistic binary instances. Table 5.3 shows results for random ternary and quad trees, respectively. Figure 5.5 visualizes the results from Table 5.3 for $n = 128$. Running times for realistic general tanglegram instances are presented in Table 5.2. In case an entry is missing in the tables, we could not solve the instances by the corresponding methods due to memory allocation constraints.

The first column *SDP* shows results obtained by semidefinite optimization, whereas the remaining columns refer to IP-based approaches for solving the model from Section 5.2. *std* refers to solving the standard linearization using CPLEX default, *ref* its quadratic reformulation (5.3). In the options *std+cyc* and *ref+cyc*, cycle

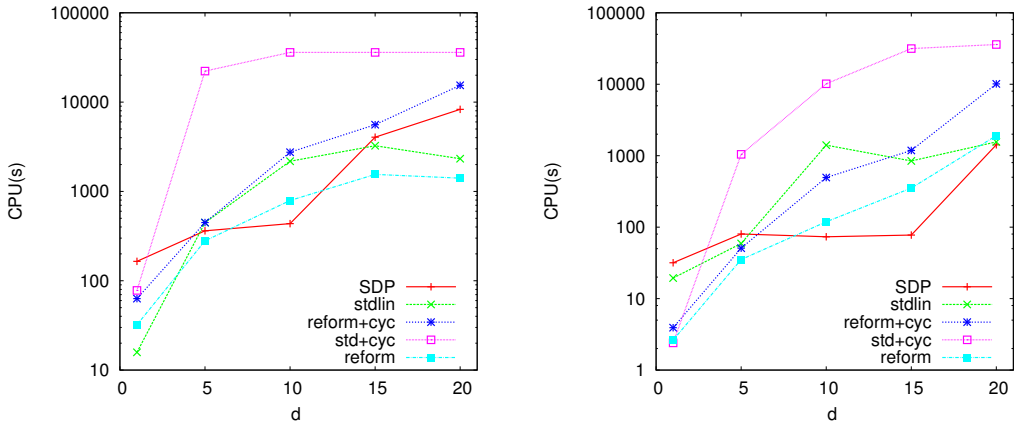

Figure 5.5: Plot showing results for $n = 128$ of Table 5.3 on page 90 for ternary (left) and quad trees (right).

inequalities for max-cut are additionally separated, all CPLEX cuts are switched off.

Table 5.2: CPU time in seconds for realistic general tanglegram instances [63].

instance	SDP	std	ref	std+cyc	ref+cyc
philips_orig_4a	27618	4725	73	4321	32692
philips_orig_4b	26093	4205	74	2756	23443
philips_orig_4c	36000	2666	122	4030	36000
philips_orig_4d	27891	3208	314	4178	28902
philips_orig_4e	36000	2789	90	5665	36000
philips_4a_4b	5238	1395	4	420	5
philips_4a_4c	4769	1858	3	637	4
philips_4a_4d	2924	1467	4	494	3
philips_4a_4e	2575	827	2	338	3
philips_4b_4c	6526	1965	8	510	7
philips_4b_4d	4872	2070	5	577	5
philips_4b_4e	2127	649	4	124	36
philips_4c_4d	4611	804	3	217	5
philips_4c_4e	6403	1074	3	396	5
philips_4d_4e	3738	891	4	811	11

Clearly, for realistic binary trees with 1–1 tangles, the SDP approach usually needs considerably more time than the IP-based methods. Furthermore, memory requirements strongly increase with system size and so the largest instances could not be solved. On average, the fastest approaches for solving the largest instances

are the pure standard linearization *std* and the quadratic reformulation *ref*.

In fact, we can optimize tanglegrams with more than 500 leaves in each tree. This is the range of sizes arising in realistic applications. The realistic instances can be solved particularly fast. Interestingly, cycle separation for max-cut usually does not pay off for binary 1–1 tanglegrams: the running time increases, even if the dual bounds are usually considerably better when cycle separation is included. Often, an optimum solution can be determined in the root node. However, although the bound is weak in the standard linearization, after few branching steps the optimum LP solution is often feasible and the program can stop. We found similar characteristics for random binary tanglegram instances.

The picture changes when varying the density of the tangle edges: for big enough tangle-edge density the SDP approach usually outperforms the IP ones. However, memory requirements usually prohibit the solution of instances with more than 500 leaf nodes and tangle-edge density of 1%. On the IP side, reformulation is often preferable. Indeed, the best performance is found when the problems are quadratically reformulated. For $n = 512$ and 1% tangle-edge density, the average solution time is 2120.42 seconds. These instances cannot be solved within the given time limits when using only the standard linearization, with or without separation of cycle inequalities.

The instances for ternary and quad trees are computationally slightly more difficult. This is mainly due to the fact that the number of betweenness restrictions decreases when compared to binary trees. Here again, the SDP approach performs well for denser instances however memory requirements strongly increase with system size. For larger instances, best performance is often found for the quadratic reformulation.

Comparing *std* with *std+cyc* and *ref* with *ref+cyc* for not necessarily binary trees, it turns out that the benefit of separating cycle inequalities increases for ternary and quad trees. The special case of a star, where the degree of the internal nodes is maximal, is equivalent to the quadratic linear ordering problem, for which we know that separation of cycle inequalities improves over *std* [18].

The realistic general instances we tested had between 371 and 414 nodes and 131 tangles. The maximum degree of an internal node was 15. We show the results in Table 5.2. Here, solving the reformulation usually yields best performance. Note that these non-binary instances could not be solved before by any other exact method.

Table 5.3: Average CPU time in seconds for random general ternary (top) and quad trees (bottom) having n leaves each, density d%.

n	d	SDP	std	ref	std+cyc	ref+cyc
64	1	3	1	<1	<1	1
	5	15	12	4	67	6
	10	18	27	19	1301	17
	15	78	54	23	3893	37
	20	41	54	48	12508	66
128	1	165	16	32	78	63
	5	362	448	280	22253	448
	10	436	2179	791	36000	2746
	15	4049	3247	1551	36000	5583
	20	8293	2326	1408	36000	15426

n	d	SDP	std	ref	std+cyc	ref+cyc
64	1	<1	<1	<1	<1	<1
	5	2	3	1	1	<1
	10	3	8	2	5	1
	15	4	16	6	57	3
	20	4	19	7	65	3
128	1	32	19	3	2	4
	5	80	59	35	1045	51
	10	73	1406	119	10147	495
	15	77	844	352	31582	1184
	20	1420	1560	1908	36000	10111

Chapter 6

Combinatorial Quadratic Optimization

In this chapter we continue the experimental evaluation of the solution techniques for binary quadratic problems presented in Chapter 2. To study the effectiveness of cutting planes for the cut polytope and quadratic reformulation when solving combinatorial quadratic problems we consider the *quadratic minimum spanning tree problem* for the reformulations SQK2 and SQK3 and the *quadratic matching problem* for the phantom monomial reformulation.

The linear variants of both problems are well-studied and solvable in polynomial time, while the quadratic variants considered here are *NP*-hard. Also, for both problems efficient separation algorithms are known, which produce good descriptions of the convex hulls of the feasible sets. In our experiments we want to find out if combining such a good description of the combinatorial structure of a quadratic problem with a good linear model of the quadratic objective function is a good approach to solve combinatorial quadratic problems. Our aim is not to compete with problem-specific algorithms but to develop an algorithmic framework that is applicable to a wide range of combinatorial quadratic problems but still gives good results in practice.

6.1 Quadratic Minimum Spanning Tree

A spanning tree in an undirected graph $G = (V, E, c)$ is a subset $T \subseteq E$ of the edges of G, such that the subgraph induced by T is cycle-free and an undirected path between each pair of nodes $u, v \in V$ exists in T. The *minimum spanning tree problem* (MST) is defined as follows.

Definition 6.1. (MST) Given an undirected Graph $G = (V, E, c)$ with edge weights $c \in \mathbb{R}^E$, find a spanning tree T in G, such that the total weight of the

edges in T is minimal.

The minimum spanning tree problem with linear costs is solvable in polynomial time, for example with the algorithms by Prim [107] and Kruskal Jr. [78]. In the *quadratic minimum spanning tree problem* (QMST) additional costs can occur for pairs of edges. Assad and Xu [9] showed that QMST is *NP*-hard.

Spanning tree problems can be modeled as integer programs using *subtour elimination constraints* and a cardinality constraint [113]. Subtour elimination constraints ensure that a solution does not contain a cycle by restricting the number of edges in all subgraphs of the solution. Given an undirected graph $G = (V, E, c)$, associate a binary variable x_e with each edge $e \in E$. A subtour elimination constraint is of the form

$$\sum_{e \in E(G[S])} x_e \leq |S| - 1\,, \tag{6.1}$$

where $S \subset V$ is a subset of the nodes of G and $E(G[S])$ denotes the edges of the subgraph $G[S]$ induced by S. Any vector $x^\star \in \{0,1\}^E$ satisfying the subtour elimination constraints for all possible subsets $S \subseteq V$ defines a cycle-free subgraph T of G, but T not necessarily spans all nodes of G. Adding the cardinality constraint

$$\sum_{e \in E} x_e = |V| - 1 \tag{6.2}$$

makes sure that T has the minimum number of edges necessary for any subgraph of G to be spanning.

There is an exponential number of subtour elimination constraints, but they can be separated in polynomial time. The separation algorithm finds violated inequalities by computing minimum cuts in a bidirected auxiliary graph [113]. As was shown by Edmonds [36], the subtour elimination constraints together with the cardinality constraint not only yield a correct IP-model for MST, they even give a complete description of the spanning tree polytope. As a consequence MST can be solved as an LP. For QMST this in general is not true anymore, but the quadratic minimum spanning tree problem can still be expressed as the following quadratic integer program:

$$\begin{aligned}
\min \quad & \sum_{e \in E} c_e x_e + \sum_{\substack{\{e,f\} \in E \times E \\ e \neq f}} q_{ef} x_e x_f \\
\text{s.t.} \quad & \sum_{e \in E} x_e = |V| - 1 \\
& \sum_{e \in E(G[S])} x_e \leq |S| - 1 \quad \forall S \subseteq V \\
& x \in \{0,1\}^E
\end{aligned} \tag{6.3}$$

6.1.1 Quadratic Reformulation

Applying the quadratic reformulation SQK2 discussed in Section 2.3.1 to the cardinality constraint (6.2) yields the constraint

$$\sum_{e \in E} x_e^2 + 2 \sum_{\substack{\{e,f\} \in E \times E \\ e \neq f}} x_e x_f \leq (|V| - 1)^2 \tag{6.4}$$

It contains all possible quadratic monomials. This means that the separation graph H for the MaxCut inequalities is complete, even if not all quadratic monomials occur in the objective function.

The SQK3 reformulation from Section 2.3.2 adds the following constraints to the model when applied to the cardinality constraint:

$$-(|V| - 2)x_f + \sum_{e \in E \setminus \{f\}} x_e x_f \leq 0 \qquad \forall f \in E \tag{6.5}$$

by multiplying with the term x_f for each edge $f \in E$ and

$$|V| x_h + \sum_{e \in E \setminus \{h\}} x_e x_h \leq |V| - 1 \tag{6.6}$$

by multiplying with the term $(1 - x_h)$, where the edge $h \in E$ is chosen arbitrarily. Again, the quadratic reformulation of the cardinality constraint leads to a complete MaxCut separation graph.

The reformulations of subtour elimination constraints are very similar to (6.5) and (6.6). The only differences are the sense of the constraints and the node sets considered.

6.1.2 Computational Results

In the following, we report computational results of a branch and cut-algorithm for the quadratic minimum spanning tree problem that incorporates the techniques presented in Chapter 2. We consider two basic settings: In the first we solve the standard linearization of (6.3) augmented with the inequalities produced by the different quadratic reformulations, but without cutting planes for the cut polytope. In the second we additionally separate odd cycle inequalities.

Instances

We used the instances with 10 and 15 nodes from the paper by Cordone and Passeri [22]. For given edge densities 33%, 67% and 100%, there are four instances with integral weights for the edges and all pairs of edges. Weights are uniformly distributed random numbers either from $\{1, \ldots, 10\}$ or $\{1, \ldots, 100\}$. There is one instance for each combination of the intervals.

Experimental Setup

The branch and cut-algorithm was implemented in SCIL and the LP-relaxations were solved with CPLEX 12.5. As proposed in Chapter 2, the separation graph for the odd cycle inequalities does not contain the auxiliary node u, because odd cycles involving u do not induce facets of the cut polytope apart from those already defined by the inequalities of the standard linearization. The separation graphs for the instances tested here are complete, therefore only odd cycle inequalities of length three are separated. All triangles in the separation graph are enumerated in a preprocessing step. The corresponding odd cycle inequalities are stored in a cut pool, which is checked for violated inequalities in each iteration of the cutting plane-algorithm.

The initial relaxation in all experiments consists of the standard linearization of the quadratic terms and the cardinality constraint (6.1). The separation routines for the odd cycle inequalities (2.3) and the subtour elimination inequalities (6.1) are called in each iteration of the cutting-plane algorithm. In the case of the SQK2 reformulation, the reformulation of the cardinality constraint as well as any violated constraints found by the separation algorithms and their SQK2 reformulation are added to the model directly. In the case of SQK3, the $|E| + 1$ inequalities of the reformulation of the cardinality constraint and all inequalities generated by reformulating constraints found by the separation algorithms are generated, but not added to the model directly, because this would lead to a significant increase in the size of the relaxation. Instead, only the original inequalities are added, the reformulation inequalities are stored in a cut pool. In each iteration of the cutting plane-algorithm this cut pool is checked for violated inequalities.

All experiments were run on an Intel Xeon CPU E5–26400@2.5GHz with 32Gb main memory. The time limit was 1h per instance.

Results

Tables 6.1 and 6.2 show the results of the computational study for the quadratic minimum spanning tree problem. The first two columns show the number of nodes and edges of the graph G, the column marked *nq* the number of quadratic terms in the objective function. In Table 6.2 this is also the number of edges in the MaxCut separation graph. *qref* is the type of quadratic reformulation used. The remaining columns give the number of subproblems solved (*subs*), the number of linear programs solved (*LPs*), the time spent by the separation algorithms in seconds (*stime/s*) and the total solution time (*ttime/s*). All numbers are averages over the four instances of each class, except for the line in italics in Table 6.1. Only two instances of this class could be solved within the time limit without MaxCut separation or quadratic reformulation.

Table 6.1: Computational results for the quadratic minimum spanning tree problem without MaxCut separation. Without quadratic reformulation only two of the instances with 45 edges could be solve to optimality within the time limit.

n	m	nq	qref	subs	LPs	stime/s	ttime/s
10	14	91	none	15.00	12.25	0.00	0.09
			SQK2	3.50	4.50	0.00	0.02
			SQK3	1.00	2.50	0.00	0.01
			SQK2+3	1.00	2.50	0.01	0.01
10	30	435	none	3734.50	3374.25	0.38	32.58
			SQK2	561.50	504.75	0.27	11.11
			SQK3	128.50	212.00	0.19	6.42
			SQK2+3	108.00	182.50	0.17	5.86
10	45	990	none	*19379.00*	*18246.00*	*4.97*	*399.73*
			SQK2	2667.00	2746.75	2.51	154.12
			SQK3	395.00	856.75	2.24	78.13
			SQK2+3	430.50	931.75	2.66	87.70
15	34	561	none	50704.50	47336.75	10.17	905.91
			SQK2	3259.00	3350.25	3.08	133.26
			SQK3	489.50	1213.00	5.08	71.81
			SQK2+3	635.50	1529.50	8.42	100.92

As can be seen from Table 6.1, quadratic reformulation leads to a significantly better performance of the branch and cut-algorithm, even when the quadratic structure of the problem is only modeled by the standard linearization. With all three reformulations tested the number of subproblems and linear programs is much lower than without quadratic reformulation. The smallest instances are even solved in the root node, when SQK2 or both SQK2 and SQK3 are applied. The reduction in the number of subproblems confirms the theoretical results of Chapter 2, that quadratic reformulation improves the LP-relaxations of constrained binary quadratic programs.

SQK2 and especially SQK3 increase the size of the relaxations and an additional separation step is required for SQK3, but this effect is compensated by the reduction in the number of LPs, so that all reformulations result in lower running times. As expected, SQK3 generally gives better results than SQK2. Applying both SQK2 and SQK3 is effective for small instances, but does not pay off for larger instances. The hardest instances for all approaches are the complete graphs on ten nodes, with 990 quadratic terms. Without quadratic reformulation only two instances are solved within one hour, while with SQK3 the average solution time is less than 80 seconds.

Table 6.2: Computational results for the quadratic minimum spanning tree problem with MaxCut separation.

n	m	nq	qref	subs	LPs	stime/s	ttime/s
10	14	91	none	1.00	3.75	0.00	0.04
			SQK2	1.00	2.75	0.00	0.01
			SQK3	1.00	2.50	0.00	0.01
			SQK2+3	1.00	2.25	0.01	0.01
10	30	435	none	1418.50	1690.50	1.96	57.46
			SQK2	10.50	50.50	0.08	1.76
			SQK3	2.00	20.00	0.05	0.69
			SQK2+3	2.00	16.25	0.03	0.68
10	45	990	none	16084.00	19579.25	42.06	943.57
			SQK2	76.50	430.50	1.27	31.07
			SQK3	40.00	296.75	1.12	26.15
			SQK2+3	42.00	300.50	1.22	28.28
15	34	561	none	5657.50	8618.50	15.75	903.12
			SQK2	50.50	259.50	0.68	20.19
			SQK3	11.00	94.50	0.31	7.37
			SQK2+3	10.00	87.00	0.30	7.26

Table 6.2 shows the effects of MaxCut separation in combination with quadratic reformulation of linear constraints. In all settings, the separation of triangle inequalities leads to a strong reduction in the number of subproblems. This effect is most pronounced for SQK2, where the number of subproblems is reduced by a factor of up to 64. At the same time, the additional separation increases the average number of LPs per subproblem, but the total number of LPs is still lower than in the case where the quadratic terms are only modeled by the standard linearization. MaxCut separation also effects the running times. As can be seen from Tables 6.1 and 6.2, without quadratic reformulation, results are mixed. While now all instances of complete graphs on ten nodes are solved within the time limit, running times for the other instances are similar or even slightly worse. When the separation of the odd cycle inequalities is combined with quadratic reformulation, the picture is different. Here, all instances are solved quicker, by a factor of up to 14.

The experimental study of the quadratic minimum spanning tree demonstrates that the combination of quadratic reformulation with a good linear description of the quadratic structure of the problem is an effective approach for solving quadratic combinatorial problems. SQK3 results in a larger increase in the size of the relaxations in comparison to SQK2, but this is compensated by the stronger

dual bounds, so that SQK3 gives the best computational results, both in regard to the number of subproblems and the overall running times.

In the final section of this chapter we investigate the effectiveness of the phantom monomial reformulation approach of Section 2.3.3 for quadratic combinatorial optimization problems with assignment constraints. We report the results of an experimental study of the quadratic perfect matching problem.

6.2 Quadratic Matching

A matching in an undirected graph $G = (V, E, c)$ is a subset $M \subseteq E$ of the edges of G, such that the edges in M are pair-wise non-adjacent. If the subgraph induced by M spans G, M is called a perfect matching. The *minimum-weight perfect matching problem* (PM) is defined as follows.

Definition 6.2. (PM) Given an undirected graph $G = (V, E, c)$ with edge weights $c \in \mathbb{R}^E$, find a perfect matching M in G, such that the total weight of the edges in M is minimal.

The minimum-weight perfect matching problem can be solved in polynomial time [33, 34] and has a simple IP-formulation:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e \\ \text{s.t.} \quad & \sum_{e \in \delta(v)} x_e = 1 \qquad \forall v \in V \\ & x \in \{0,1\}^E, \end{aligned} \tag{6.7}$$

where $\delta(v)$ denotes the set of edges incident to $v \in V$.

Adding the so-called *blossom inequalities* [34]

$$\sum_{e \in \delta(S)} x_e \geq 1 \quad \forall S \subseteq V,\ |S| \geq 3,\ |S| \text{ odd} \tag{6.8}$$

to the LP-relaxation of (6.7) gives a complete description of the *perfect matching polytope*. Here $\delta(S)$ denotes the set of edges that have exactly one end-point in S. Blossom inequalities can be separated in polynomial time, e.g. with the algorithm by Letchford et al. [84] for the slightly more general case of b-matchings.

In the *minimum-weight quadratic perfect matching problem* (QPM) additional costs occur for pairs of distinct edges, as in QMST. The IP-model of QPM is thus the same as for PM, except for additional weighted quadratic monomials in the objective function.

Theorem 6.1. *The minimum-weight quadratic perfect matching problem is* NP*-hard.*

Proof. The proof works by showing that QPM is a generalization of the *quadratic assignment problem* (QAP) [19], which is *NP*-hard [111]. In the general formulation by Lawler [81], QAP asks for a minimum-cost assignment of n facilities to n locations. All facilities must be assigned to a location and each location can only take one facility. Costs c_{ijkl} occur when facility i is assigned to location j and facility k to location l. This problem can be modeled as a quadratic perfect matching problem on a bipartite graph as follows. Define an undirected graph $G = (V \cup W, E)$ with one node in V for each facility and one node in W for each location. E contains an edge from each node in V to each node in W. A perfect matching in G now corresponds to an assignment of facilities to locations. Finally, define the objective function of the QPM on G as

$$\sum_{(i,j)\in E} \sum_{(k,l)\in E} c_{ijkl} x_{ij} x_{kl} \,.$$

The resulting QPM is a model for the QAP. □

The IP-model of QPM contains assignment constraints, which allow the introduction of phantom monomials discussed in Chapter 2. The number of phantom monomials depends on the structure of the graph G. For $v \in V$, denote by $PM(v)$ the set of phantom monomials generated by the assignment constraint

$$\sum_{e\in\delta(v)} x_e = 1 \,.$$

We have

$$|PM(v)| = \binom{|\delta(v)|}{2}$$

and the total number of phantom monomials is

$$\sum_{v\in V} |PM(v)| = \sum_{v\in V} \binom{|\delta(v)|}{2} ,$$

since $PM(v) \cap PM(w) = \emptyset$ for $v \neq w$.

When G is a complete graph on n nodes, each node has $n-1$ incident edges. In this case the total number of phantom monomials is

$$\sum_{v\in V} |PM(v)| = n\binom{n-1}{2} = \frac{1}{2} n(n-1)(n-2) = \frac{n^3 - 3n^2 + 2n}{2} \,.$$

In comparison, the total number of quadratic monomials that can be formed from pairs of edges in a complete graph is

$$\binom{\binom{n}{2}}{2} = \frac{n^4 - 2n^3 - n^2 + 2n}{8} \,.$$

Table 6.3: Computational results for the quadratic perfect matching problem without phantom monomials. Odd cycle inequalities are separated with the exact algorithm by Barahona and Mahjoub [12]. Only 4 of 5 of the instances with 20 nodes and 30% density are solved within the tine limit of 1h.

n	m	ne	subs	LPs	stime/s	ttime/s
10	45	99	5.80	6.20	0.04	0.04
		148	4.20	4.80	0.04	0.05
		198	6.20	6.40	0.09	0.11
		247	14.60	16.00	0.29	0.37
		297	7.40	10.00	0.20	0.25
14	91	409	6.20	6.00	0.32	0.36
		614	10.20	12.80	1.17	1.29
		819	13.80	22.20	2.78	3.04
		1023	32.20	65.20	11.29	12.15
		1228	25.80	55.00	11.67	12.55
18	153	1162	14.20	19.20	6.14	6.48
		1744	41.00	60.80	36.65	38.09
		2325	53.00	102.20	76.81	79.92
		2907	115.80	255.40	245.58	255.20
		3488	174.20	412.60	478.72	497.14
20	190	1795	15.40	20.40	12.65	13.20
		2693	56.60	98.40	108.67	112.17
		3591	87.80	177.40	268.25	276.66
		4488	240.20	548.20	1074.45	1107.09
		5386	*210.50*	*545.00*	*1285.94*	*1325.70*

6.2.1 Computational Results

Instances

We compute perfect matchings in complete graphs with $|V| \in \{10, 14, 18, 20\}$. Edge weights are chosen uniformly at random from the interval $[0, 100]$. In each instance a certain percentage $d \in \{10, 15, 20, 25, 30\}$ of all possible edge pairs is chosen randomly. These edge pairs are also assigned a weight in the interval $[0, 100]$. All other pairs are given weight zero and the corresponding quadratic terms are thus dropped from the model. The test set consists of 100 instances in total, five for each combination of size and number of quadratic terms.

Experimental Setup

The basic experimental setup is the same as in the previous section, except two points: In the experiments for the quadratic MST, the separation graph for odd cycle inequalities was complete, which motivated the use of triangle separation. In our experiments for the quadratic matching problem the separation graph is sparse, even with the additional edges resulting from phantom monomials. Therefore we use the two separation algorithms presented in Section 2.2.1, the exact algorithm by Barahona and Mahjoub [12] and the heuristic forest cycle separation by Barahona, Jünger, and Reinelt [13]. The second difference is that quadratic reformulation is only possible for the constraints of the basic IP-formulation (6.7). Neither the odd cycle inequalities (2.3) nor the blossom inequalities (6.8) meet the requirements for the phantom monomial reformulation. The right-hand side of an odd cycle inequality is always an even number and the sense of a blossom inequality is "$\geq$" instead of "$\leq$".

We start with the LP-relaxation of (6.7). The separation routines for blossom and odd cycle inequalities are called in each iteration of the cutting plane algorithm. We test the effectiveness of phantom monomials in combination with both the exact and the heuristic separation algorithm for odd cycle inequalities.

Results

Tables 6.3 – 6.6 report the experimental results for the minimum-weight perfect matching problem. All tables show the number of nodes (n) and edges (m) of the matching graph, the total number of edges in the separation graph for odd cycle inequalities (ne), the number of subproblems ($subs$) and linear programs (LPs) in the branch and cut-algorithm, and the time spent in the separation routines ($stime/s$) as well as the total solution time ($ttime/s$). All values are averages over five instances and times are reported in seconds. Tables 6.4 and 6.6 contain the average number of phantom monomials (PM).

It turns out that in most cases, the inclusion of phantom monomials does not lead to lower solution times. It can be observed that especially with exact MaxCut separation, the solution time is nearly completely determined by the time spent on the separation of odd cycle inequalities. The increase in the size of the separation graph caused by the quadratic reformulation of the assignment constraints naturally has a negative impact on the runtime of the separation algorithm, which cannot be compensated by the positive impact on the dual bounds. Nevertheless, the denser separation graph leads to a significant reduction in the number of subproblems. This demonstrates that phantom monomials are effective in strengthening the LP-relaxations of quadratic combinatorial problems with assignment constraints.

When the heuristic separation algorithm is used instead of the exact one, solution

Table 6.4: Computational results for the quadratic perfect matching problem with phantom monomials and exact cycle separation.

n	m	PM	ne	subs	LPs	stime/s	ttime/s
10	45	360	422.40	3.40	6.80	0.18	0.19
			456.00	3.00	4.40	0.11	0.12
			485.40	3.00	6.00	0.18	0.20
			517.00	6.60	16.00	0.56	0.62
			540.00	5.00	9.80	0.36	0.40
14	91	1092	1394.00	3.80	7.80	1.66	1.73
			1546.00	4.60	11.20	2.74	2.86
			1698.40	7.00	20.40	5.67	5.90
			1836.40	15.80	41.20	12.48	12.97
			1996.40	19.00	43.00	14.56	15.19
18	153	2448	3371.00	4.20	13.60	13.67	13.99
			3828.00	16.60	43.20	52.12	53.18
			4286.80	26.20	72.40	94.31	96.35
			4738.20	68.20	176.80	260.05	266.14
			5198.00	115.00	286.20	449.99	462.01
20	190	3420	4881.60	8.20	25.80	51.82	52.69
			5587.20	21.80	67.20	150.78	153.24
			6314.80	48.60	130.20	345.36	351.30
			7048.80	131.00	335.20	940.14	958.77
			7771.20	205.00	518.00	1527.74	1561.68

times are much lower, as can be seen from Tables 6.5 and 6.6. The heuristic separation is much faster in practice than the exact algorithm, but it is also less effective, which leads to a higher number of subproblems and LPs. The effect of phantom monomials is smaller in this setting. Especially for the larger and denser instances the reduction in the number of subproblems is lower, which indicates that the heuristic separation algorithms profits less from the additional edges in the separation graph.

The experimental study shows that the quadratic reformulation of assignment constraints in quadratic combinatorial problems has a positive impact on the quality of the LP-relaxations and leads to a significant reduction in the number of subproblems. At the same time the increase in the density of the separation graphs may lead to excessive separation times.

Table 6.5: Computational results for the quadratic perfect matching problem without phantom monomials. Odd cycle inequalities are separated with the heuristic algorithm by Barahona, Jünger, and Reinelt [13].

n	m	ne	subs	LPs	stime/s	ttime/s
10	45	99	5.80	6.20	0.00	0.02
		148	4.20	3.80	0.00	0.01
		198	6.20	6.60	0.01	0.03
		247	13.80	16.00	0.02	0.08
		297	7.40	12.80	0.02	0.07
14	91	409	6.20	6.00	0.01	0.05
		614	9.80	12.40	0.02	0.16
		819	16.20	24.00	0.10	0.39
		1023	27.80	58.20	0.34	1.10
		1228	25.80	61.00	0.41	1.37
18	153	1162	14.20	18.00	0.11	0.43
		1744	29.00	51.60	0.53	1.72
		2325	57.80	106.20	1.56	4.78
		2907	99.00	244.00	4.75	13.85
		3488	207.80	503.60	12.84	35.29
20	190	1795	22.20	25.60	0.30	0.99
		2693	60.20	103.00	1.85	5.42
		3591	81.00	177.60	4.23	12.38
		4488	209.80	548.20	17.96	50.11
		5386	327.40	906.60	37.06	101.48

Table 6.6: Computational results for the quadratic perfect matching problem with phantom monomial and heuristic cycle separation.

n	m	PM	ne	subs	LPs	stime/s	ttime/s
10	45	360	422.40	3.40	7.00	0.01	0.03
			456.00	3.00	4.60	0.01	0.02
			485.40	3.00	6.40	0.01	0.03
			517.00	7.00	16.80	0.04	0.09
			540.00	5.40	10.80	0.02	0.07
14	91	1092	1394.00	3.80	8.00	0.05	0.12
			1546.00	4.60	11.60	0.10	0.21
			1698.40	7.80	21.60	0.18	0.42
			1836.40	19.00	52.60	0.52	1.10
			1996.40	23.40	56.20	0.61	1.38
18	153	2448	3371.00	7.00	18.20	0.35	0.74
			3828.00	13.40	37.60	0.72	1.67
			4286.80	31.80	94.20	2.45	5.08
			4738.20	85.00	234.60	6.99	14.77
			5198.00	129.00	329.20	11.18	24.50
20	190	3420	4881.60	11.80	32.80	1.03	2.01
			5587.20	38.20	103.00	3.77	7.21
			6314.80	60.60	174.20	7.48	15.11
			7048.80	189.80	513.60	25.44	53.63
			7771.20	224.60	629.40	32.78	73.39

Chapter 7

Range Assignment Problems

As a first application of the two approaches to submodular combinatorial optimization problems presented in Chapter 3 we study a class of problems from wireless network design, so-called *range assignment problems*. When designing an ad-hoc wireless network one main objective is to minimize transmission costs subject to certain requirements concerning the network topology. In traditional wired networks, these transmission costs are roughly proportional to the length of all connections installed, so that the aim is to minimize the total length of all connections. In wireless networks, the transmission costs depend on the transmission ranges assigned to the nodes. The main difference lies in the so-called *multicast advantage*: if a node v reaches another node w, then it also reaches each node u that is closer to v than w, at no additional cost. Accordingly, the objective function of range assignment problems, i.e. the overall transmission power of the network needed to establish the specified connections, is nonlinear as a function of the connections.

Range assignment problems have been studied intensively in recent years and several exact algorithms have been proposed. Fuchs [44] showed that the problem of finding a minimum-power connected network with bidirectional links (the *symmetric connectivity problem*, see Figure 7.1) is *NP*-hard. Althaus et al. [5, 6] proposed an ILP formulation for this problem which is based on a linear extended formulation. For each node of the network they introduce a new binary variable for each value the transmission power of the node can take in an optimal solution and express the objective function in terms of these new variables. Montemanni and Gambardella [96] apply a very similar technique, modeling the transmission power levels of the nodes incrementally, as the sum of artificial binary variables. A comparison of different formulations for the symmetric connectivity problem can be found in [98]. Note that all models mentioned above are extended linear formulations of the original problem and do not exploit the submodularity of the objective function directly. We do not know of any approach in the literature that needs only a constant number of artificial variables.

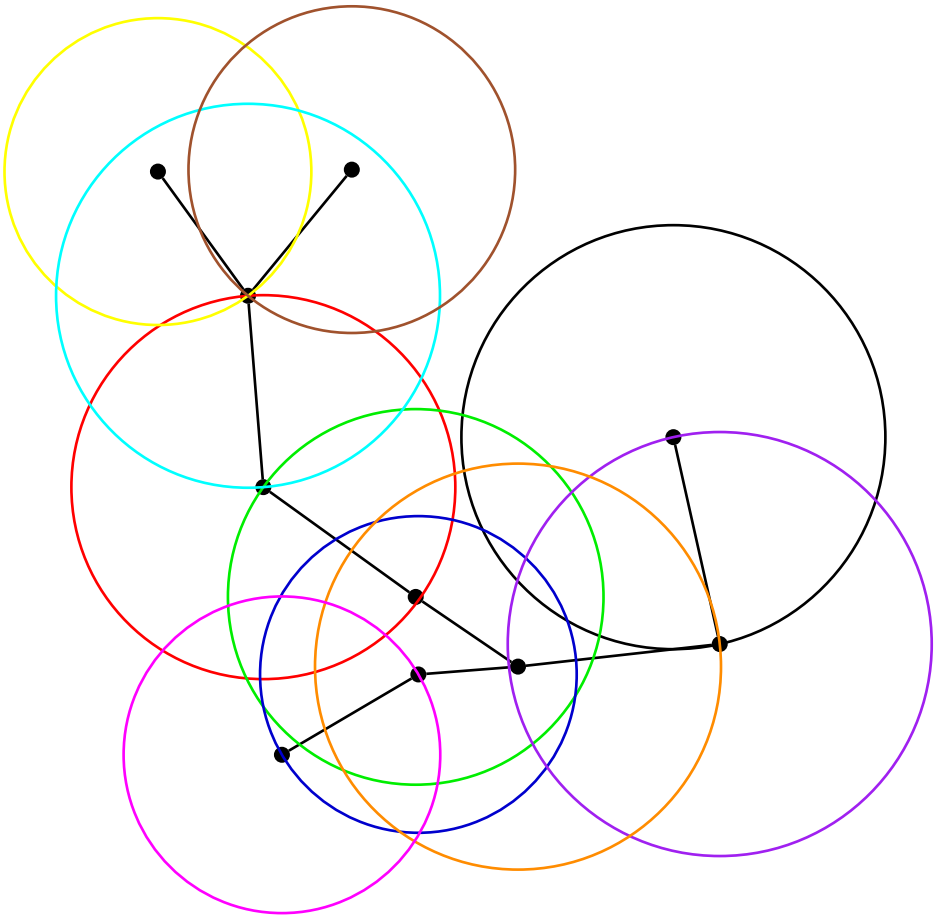

Figure 7.1: A minimum-power spanning tree on ten nodes. The colored circles indicate the transmission ranges of the nodes. Note that transmissions have to be bi-directional. The overall transmission cost is determined by the sum of the areas of the circles.

A second important variant of the range assignment problem is the *minimum power multicast problem*. Here the objective is to construct a network that allows unidirectional communication from a designated source node to a set of receiving nodes. All nodes of the network, including the receiving stations, can function as relay nodes, thereby passing on a signal on its way from the source node to the receivers, see Figure 7.2. Special cases are the *unicast problem* and the *broadcast problem*. In the former, communication is directed to a single receiving node; in the latter, all nodes except the source are addressed. The general minimum power multicast problem is *NP*-hard [44]. The unicast problem, on the other hand, is efficiently solvable. With only a single receiving station the problem reduces to

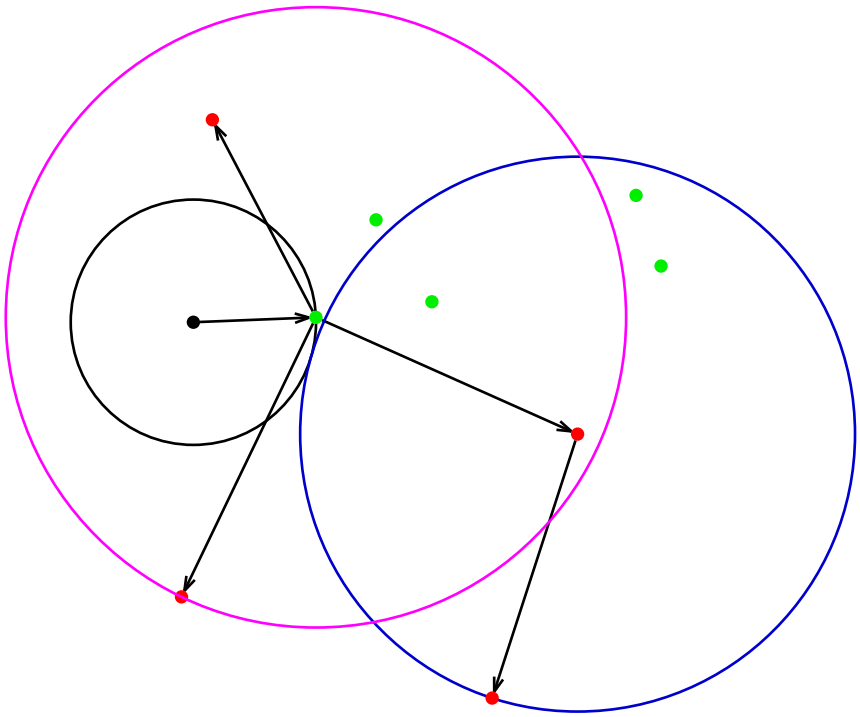

Figure 7.2: An minimum-power multicast problem on ten nodes with four terminals. The source node is black, the terminals red. The green nodes are relay nodes. Observe the *multi-cast advantage* when transmitting from the relay node. Transmissions are uni-directional. The resulting graph is a Steiner tree.

finding a shortest path through the directed network from the source to the destination node. The linear variant of the broadcast problem is also known as the *optimum branching problem*. Several authors independently presented efficient algorithms to compute an optimal solution [21, 35, 15].

Many of the algorithms for the symmetric connectivity case can be easily adapted to multicasting. Additionally, Leggieri et al. [83] investigate the multicasting problem specifically and present a set covering formulation, as well as preprocessing techniques to reduce the problem size [102]. There are also flow-based ILP-formulations. One example can be found in the paper by Min et al. [93]. In the same paper the authors present two exact iterative algorithms which use LP-relaxations to compute lower bounds. An overview over existing IP models for the multicast problem can be found in [27].

We model the general range assignment problem in graph theoretic terms. The communication stations correspond to the set of nodes V of the graph, potential

links between the stations to the set of weighted edges E. For the symmetric connectivity problem, the graph $G = (V, E, c)$ is undirected with edge costs c, for the multicast problem it is directed. The objective is to compute a subset of the edges such that certain restrictions on the topology of the network are satisfied and the overall transmission costs are minimal. Common to both models is the objective function: given a subset of edges, for each node only the most expensive incident/outgoing edge is taken into account. Summing up these values gives the overall costs. Associating a binary variable x_{vw} to each edge $e = (v, w) \in E$, the objective function can be written as

$$f(x) = \sum_{v \in V} \max \{c_{vw} x_{vw} \mid vw \in E\} . \tag{7.1}$$

The following theorem shows that f is a submodular function. Thus range assignment problems can be modeled as submodular combinatorial optimization problems and the algorithms developed in Chapter 3 are applicable.

Theorem 7.1. *For each $v \in V$ and for arbitrary $c \in \mathbb{R}^E$, the function*

$$f_v(x) = \max \{c_{vw} x_{vw} \mid vw \in E\}$$

is submodular. In particular, the function $f(x) = \sum_{v \in V} f_v(x)$ is submodular.

Proof. By definition, f_v is submodular if

$$f_v(A \cup B) + f_v(A \cap B) \leq f_v(A) + f_v(B)$$

for arbitrary sets $A, B \subseteq E$. We distinguish two cases:

(a) if $f_v(A) \geq f_v(B)$, then $f_v(A \cup B) = f_v(A)$ and $f_v(A \cap B) \leq f_v(B)$

(b) if $f_v(A) \leq f_v(B)$, then $f_v(A \cup B) = f_v(B)$ and $f_v(A \cap B) \leq f_v(A)$

In both cases, it follows that $f_v(A \cup B) + f_v(A \cap B) \leq f_v(A) + f_v(B)$. Finally, by Proposition 3.1 the function f is submodular, because it is a conic combination of submodular functions. □

The desired network topology is described by a set of feasible vectors $X \subseteq \{0, 1\}^E$. Combining objective function and constraints, the general IP formulation for range assignment problems reads

$$\begin{array}{ll} \min & \displaystyle\sum_{v \in V} \max \{c_{vw} x_{vw} \mid vw \in E\} \\ \text{s.t.} & x \in X . \end{array} \tag{7.2}$$

7.1 The Standard Model

As mentioned earlier, the standard linearization for this model found in the wireless networking literature is due to Althaus et al. [5]. They introduce new binary variables which model the possible values of the nonlinear terms in optimal solutions and add constraints linking the original variables to the new ones. The resulting problem reads

$$\begin{array}{lrcll} \min & \displaystyle\sum_{vw\in E} c_{vw}y_{vw} & & & \\ \text{s.t.} & \displaystyle\sum_{vw\in E} y_{vw} & \leq & 1 & \text{for all } v \in V \\ & \displaystyle\sum_{c_{vu}\geq c_{vw}} y_{vu} & \geq & x_{vw} & \text{for all } vw \in E \\ & x & \in & X & \\ & y & \in & \{0,1\}^E\,. & \end{array} \tag{7.3}$$

In this model, the binary variable y_{vw} is thus set to one if and only if the transmission power of node v is just enough to reach node w. Note that, depending on the network topology described by X, the first set of constraints can be strengthened to equations. This is the case when all feasible edge-induced subgraphs are connected. In this case, each node has to reach at least one other node. In general, this is not true, so that for some v all variables y_{vw} can be zero. The number of variables in this model is $2|E|$.

A closely related model appearing in the literature [97] uses binary variables in an incremental way: again, a variable $y'_{vw} \in \{0,1\}$ is used for each pair of nodes v and w, now set to one if and only if node v can reach node w. It is easy to see that the two models are isomorphic by the transformation

$$y'_{vw} = \sum_{c_{vu}\geq c_{vw}} y_{vu}\,.$$

Because of this, the two models are equivalent from a polyhedral point of view and it suffices to consider the first model in the following.

7.2 New Mixed-Integer Models

The general model (7.2) for range assignment problems we gave above is of the type of submodular combinatorial optimization problems we studied in Section 3.3, since the objective function $f(x) = \sum_{v\in V} \max\{c_{vw}x_{vw} \mid vw \in E\}$ is submodular. To apply the polyhedral results from Chapter 3, we introduce a single artificial variable $y \in \mathbb{R}$ to move the objective function into the constraints.

The corresponding model is

$$\begin{array}{lrcl} \min & y & & \\ \text{s.t.} & y & \geq & \displaystyle\sum_{v \in V} \max\{c_{vw} x_{vw} \mid vw \in E\} \\ & x & \in & X \\ & y & \in & \mathbb{R}\,. \end{array} \tag{7.4}$$

From Theorem 7.1 we know that the objective is submodular; this means that Theorem 3.6 is applicable and we have an efficient separation algorithm for (7.4).

Theorem 7.1 showed that in the case of range assignment problems the objective function is not only submodular itself but also the sum of submodular functions. We can thus use the slightly larger mixed-integer model (3.4), which in our application reads

$$\begin{array}{lrcl} \min & \sum_{v \in V} y_v & & \\ \text{s.t.} & y_v & \geq & f_v(x) \quad \text{for all } v \in V \\ & x & \in & X \\ & y & \in & \mathbb{R}^V\,. \end{array} \tag{7.5}$$

We know from Theorem 3.10 that we can again separate efficiently when ignoring the problem-specific constraint $x \in X$.

7.3 Polyhedral Relations

In the following, we investigate the polyhedral properties of the standard model and the new mixed-integer models. First, we show how the corresponding polyhedra are related to each other. For this, let $P_1(X)$, $P_2(X)$, and $P_3(X)$ denote the polyhedra given as the convex hulls of feasible solutions in the models (7.3), (7.4), and (7.5), respectively. Note that $P_1(X)$ is a convex hull of binary vectors, so in particular it is a polytope and all its integral points are vertices. On the other hand, the polyhedra $P_2(X)$ and $P_3(X)$ are unbounded by definition. It is easy to see that $P_3(X)$ arises from the convex hull of

$$\left\{(x, y) \in X \times \mathbb{R}^V \mid y_v = \max\{c_{vw} x_{vw} \mid vw \in E\} \forall v \in V\right\}$$

by adding arbitrary nonnegative multiples of unit vectors for the variables y_v. Similarly, $P_2(X)$ arises from the convex hull of

$$\left\{(x, y) \in X \times \mathbb{R} \mid y = \sum_{v \in V} \max\{c_{vw} x_{vw} \mid vw \in E\}\right\}$$

by adding arbitrary nonnegative multiples of the unit vector for y.

Theorem 7.2. *The convex hull of all vertices of $P_3(X)$ is a projection of an integer subpolytope of $P_1(X)$.*

Proof. Consider the projection π_1 given by

$$y_v := \sum_{vw \in E} c_{vw} y_{vw} \ .$$

Let $(x, y) \in X \times \mathbb{R}^V$ be a vertex of $P_3(X)$. Then $y_v = \max \{c_{vw} x_{vw} \mid vw \in E\}$ for all $v \in V$. Thus setting $y_{vw} = 1$ for exactly one w with $y_v = c_{vw}$ yields a vertex of $P_1(X)$ that is mapped to (x, y) under π_1. □

In Section 3.3, we have shown that $P_2(X)$ is a projection of the polyhedron $P_3(X)$, so that Theorem 7.2 also holds if $P_3(X)$ is replaced by $P_2(X)$. These results show that for every reasonable objective function the optimal faces of all three polyhedra are projections of each other. The first model can thus be considered an extended formulation of the second and third one, and the third model can be considered an extended formulation of the second.

Note that in general $P_1(X)$ contains vertices that are not mapped to the convex hull of vertices of $P_2(X)$ or $P_3(X)$. These vertices cannot be optimal for any of the considered objective functions.

7.4 Computational Results

In the following, we report results of branch and bound-algorithms based on the cutting plane approach of Section 3.4 and the Lagrangean decomposition approach of Section 3.5, respectively. For the implementation, we use the exact optimization software library SCIL [114]. The LP-relaxations at each node of the enumeration tree are solved with CPLEX 12.1. The bundle method for the Lagrangean decomposition approach is implemented using the ConicBundle library v0.3.8 [58]. To calculate the subgradients, i.e. to optimize the second partial problem, we used an implementation of Edmonds' algorithm [119] for the broadcast problem and the Boost Graph Library 1.46.1 for graph modeling and basic graph algorithms [117]. In all experiments with the Lagrangean decomposition approach we branch on the variable for which the corresponding multiplier has the largest absolute value among the candidates.

All experiments were run on a 2.6 GHz AMD Opteron 252 processor. We set a time limit of one hour for each instance.

7.4.1 Symmetric Connectivity

As mentioned earlier the symmetric connectivity problem is a range assignment problem on an undirected graph G. To establish a connection between nodes u and v, the transmission range of node u must be large enough to reach node v

and vice versa. The set X in the general nonlinear model (7.2) specializes to the set of spanning subgraphs of G.

In this case, all three IP-formulations (7.3), (7.4), and (7.5) can be significantly strengthened. First of all, the set X can be restricted to the set of spanning trees in G without loss of generality. This is equivalent to introducing an additional constraint $\sum_{e\in E} x_e = |V| - 1$. This stronger formulation does not change the optimum of our problem but improves the quality of the bounds obtained from the LP-relaxations and thus reduces running time. In our experiments we used the subtour formulation of the spanning tree polytope.

Another way to strengthen the model is related to the fact that in a connected subgraph (on at least two nodes) each node has at least one incident edge. For the standard model, this means that the constraints $\sum_{uv\in E} y_{uv} \leq 1$ can be strengthened to equations $\sum_{uv\in E} y_{uv} = 1$, for all $u \in V$. In the mixed-integer models (7.4) and (7.5) we can eliminate one variable from each maximum term. As the transmission power for each node v has to be at least the smallest weight $c_v^{\min}$ of the incident edges, this constant can be extracted from the corresponding maximum term. The constraints of model (7.5) become

$$y_v \geq c_v^{\min} + \max\{(c_{vw} - c_v^{\min})x_{vw} \mid vw \in E\} \text{ for all } v \in V \ ,$$

so that at least one entry in the maximum can be removed. In the compact model (7.4), the constraint that bounds the overall transmission power from below can be strengthened analogously. Both replacements lead to stronger LP-relaxations if the separation algorithms derived in Section 3.4 are now applied to the remaining maximum terms.

Turning to the Lagrangean relaxation approach, the structure of the set X, i.e. the set of all spanning trees, allows to apply fast combinatorial algorithms like Kruskal's [78] or Prim's [107] to the second problem in the decomposition (3.16). The first problem is a special submodular function minimization problem. Even though currently no specialized combinatorial algorithm for this kind of submodular function is available in the case of undirected graphs, there exists one for directed graphs first described by Miller [92]. The algorithm is based on the fact that for directed graphs the minimization of the corresponding submodular function (7.1) can be decomposed into $|V|$ smaller minimization problems

$$\sum_{v\in V} \min_{x\in\{0,1\}^{\delta(v)}} \left(\max_{e\in\delta(v)}\{c_e x_e\} - \sum_{e\in\delta(v)} \lambda_e x_e \right) ,$$

where $\delta(v) = \{vw \in E \mid w \in V\}$. This is due to the fact that each variable x_{vw} appears in only one of the minima and the variables are not linked by any constraints. The partial problems can be solved by Algorithm 3, which for each $e \in \delta(v)$ computes the optimal solution x satisfying $x_e = 1$ and $x_f = 0$ for $c_f > c_e$.

We mention that Algorithm 3 can also be implemented to run in linear time after sorting the coefficients c_e; the latter can be done in a preprocessing step, as it does

Algorithm 3 Solution of partial problem

input: objective function $f_v(x) := \max_{e \in \delta(v)} \{c_e x_e\} - \sum_{e \in \delta(v)} \lambda_e x_e$

output: optimal solution of $\min_{x \in \{0,1\}^{\delta(v)}} f_v$

```
x* ← 0
opt ← 0
for e ∈ δ(v) do
    x ← 0
    sum ← c_e
    for f ∈ δ(v) do
        if c_f ≤ c_e and λ_f > 0 then
            x_f ← 1
            sum ← sum − λ_f
        end if
    end for
    if sum < opt then
        opt ← sum
        x* ← x
    end if
end for
return x*
```

not depend on the Lagrangean multipliers λ. To take advantage of this algorithm, which only works for directed graphs, we will consider a directed version of the symmetric connectivity problem, which is originally defined on undirected graphs. To gain an equivalent directed formulation we double the variables and introduce new constraints $x_{vw} = x_{wv}$ for all $vw \in E$, where E is now the set of all directed edges between nodes in V. These new constraints will become part of the second problem in the decomposition, so that a spanning tree algorithm can still be applied to the corresponding undirected graph where the weights (Lagrangean multipliers) of two edges vw and wv are summed up.

As an alternative, one could use an algorithm for general submodular function minimization or the linear programming approach for unconstrained submodular minimization discussed in Chapter 3 to solve the first problem directly on the undirected instance. However, experiments show that the directed and the undirected models give similar bounds while the computation of the Lagrangean dual is much faster when Algorithm 3 can be applied. The following results for the Lagrangean decomposition approach are therefore based on the directed version of the symmetric connectivity problem.

Table 7.1: Results for the symmetric connectivity range assignment problem.

n	subs	LPs/LCs	t_{sep}/s	t_{tot}/s	# solved
		standard IP model (7.3)			
10	29.48	32.20	0.00	0.08	50
15	1147.96	1217.28	0.18	12.28	50
20	*4048.22*	*4461.83*	*3.63*	*114.65*	46
25	*2248.40*	*2513.51*	*4.35*	*117.99*	43
		MIP model (7.5)			
10	23.28	70.70	0.01	0.08	50
15	823.04	2597.38	0.98	7.29	50
20	*2820.51*	*11049.13*	*16.17*	*100.79*	45
25	*3353.15*	*14001.85*	*45.21*	*281.17*	41
		Lagrangean decomposition			
10	18.84	282.26	-	0.63	50
15	180.00	3007.10	-	20.43	50
20	*749.83*	*11871.20*	-	*106.39*	48
25	*1668.22*	*26067.50*	-	*324.14*	41

To speed up the bundle method, we use a warm start approach, using the best Lagrangean multipliers from the corresponding parent node as starting points. This leads to a much lower number of iterations in general. Note that in most instances over 50% of the total time was spent in the root node to compute a good initial set of Lagrangean multipliers.

We generated random range assignment instances by randomly placing points on a 10000×10000 grid, as proposed in [5]. For each size, 50 instances were created. The transmission power needed for node u to reach node v was chosen as $d(u,v)^2$, where $d(u,v)$ is the Euclidian distance between u and v. Table 7.1 summarizes our results for the symmetric connectivity problem. The first column shows the size of the instances, the second the average number of subproblems computed in the branch and cut-tree. The column *LPs/LCs* contains the average number of linear programs solved (for the cutting plane approach) and the average number of times the Lagrangean function $Z(\lambda)$ was evaluated (for the decomposition approach), respectively. The average overall time needed to solve the instance is denoted by t_{tot}/s. For the cutting plane approach we also state the time spent on separation (t_{sep}/s). The last column shows how many of the 50 instances of each size could be solved within the time limit of one hour. For the computation of averages only instances that could be solved to optimality were considered.

We do not list results for the compact model (7.4). It turned out that this model is

not competitive. Because only a single inequality of the description of the objective function can be separated per iteration, the number of LPs grows quickly in comparison to the other models. Another disadvantage of modeling the objective as a single submodular function instead of as a conic combination of submodular functions is the much higher number of cutting planes needed to generate a tight polyhedral description. As we already stated in Section 3.3, the number of inequalities modeling the objective function in the compact model is $(\frac{n(n-1)}{2})!$, compared to $(\frac{n(n-1)}{2})!$ in model (7.5). This again increases the number of LPs drastically. The medium-sized model (7.5) gives the best results for instances up to 15 nodes, also compared to the Lagrangean decomposition approach. The number of subproblems is significantly smaller than for the standard model, which compensates for the larger number of LPs. For instance size 20 the Lagrangean decomposition approach performs best. For the largest instances the standard model gives the best results, because the time spent per node in the other models becomes too large. It is remarkable that several instances could not be solved at all within the time limit, whereas the average solution time for the other instances is relatively small and only grows moderately with the instance size.

7.4.2 Multicast

We next investigate the min-power multicast problem. Recall that its objective is to send signals wirelessly from a designated source node to a set of receiving stations at minimum cost. Transmissions are unidirectional and all stations can relay signals through the network. Treating this problem as a graph optimization problem, there obviously is a one-to-one correspondence between feasible solutions and Steiner arborescences in the directed graph. The multicast advantage can again be expressed by (7.1), this time for directed graphs. We used a separation routine for the cut formulation of the Steiner arborescence polytope to model the network topology in both cutting plane models.

The given connectivity constraints can again be used to strengthen the LP-based formulations, however to a lesser extent than in the symmetric case. Only the fact that at least one edge has to leave the source node provides a way to strengthen the models.

Table 7.2 shows the results for the multicast problem. The number of terminal nodes is chosen as $\lfloor \frac{n-1}{2} \rfloor$. As mentioned before the decomposition approach is not applicable here, because there is no efficient algorithm for the Steiner tree problem. We used the same instances as for the symmetric connectivity problem. The source and terminal nodes were determined randomly. For this kind of problem exploiting the submodularity of the objective function clearly pays off. While for small instances both models give similar results, the better polyhedral description in the MIP model significantly reduces running times for larger instances. 46 of

Table 7.2: Results for the multicast range assignment problem with $|T| = \lfloor \frac{n-1}{2} \rfloor$.

n	subs	LPs	t_{sep}/s	t_{tot}/s	# solved
		standard IP model (7.3)			
10	32.68	57.10	0.00	0.08	50
15	117.92	241.84	0.09	0.93	50
20	2991.52	6444.50	8.43	71.18	50
25	*5773.97*	*27788.03*	*64.02*	*383.07*	39
		MIP model (7.5)			
10	28.92	111.92	0.01	0.10	50
15	88.24	556.38	0.28	1.14	50
20	951.32	7815.54	11.39	33.28	50
25	*5650.17*	*73373.59*	*208.51*	*571.88*	46

the largest instances could be solved to proven optimality within the time limit of one hour, compared to 39 with the standard model.

7.4.3 Broadcast

Since the problem of finding a minimal Steiner arborescence is *NP*-hard the Lagrangean decomposition approach is inefficient for general multicast problems. However, the set of feasible solutions for the broadcast problem corresponds to the set of s-arborescences for which the correspond linear minimization problem can be solved in polynomial time [35]. The first problem in the Lagrange decomposition can then again be solved by algorithm 3 that was used for the symmetric connectivity problem. In the case of the broadcast problem it is not necessary to double the variables and introduce additional constraints, since the problem is already defined on directed graphs.

Table 7.3 shows that the Lagrangean decomposition approach is able to solve the highest number of the large instances, while remaining competitive for the smaller instances. The MIP approach is slowed down by the large number of LPs and the resulting high number of calls to the separation routines.

7.5 Final Remarks

The experimental study on range assignment problems shows that none of the three algorithms is clearly superior to the others. The results for the MIP models illustrate the importance of breaking down the objective function into submodular parts for a competitive algorithm. In the Lagrangean decomposition approach

Table 7.3: Results for the broadcast range assignment problem.

n	subs	LPs/LCs	t_{sep}/s	t_{tot}/s	# solved
		standard IP model (7.3)			
10	36.16	45.02	0.00	0.09	50
15	167.24	243.04	0.07	1.31	50
20	1519.40	2801.68	4.24	36.53	50
25	*7117.19*	*16238.07*	*32.57*	*375.87*	43
		MIP model (7.5)			
10	32.88	120.70	0.02	0.12	50
15	142.52	776.78	0.43	1.79	50
20	1051.76	8796.32	13.79	42.87	50
25	*6101.98*	*69896.47*	*200.10*	*598.74*	43
		Lagrangean decomposition			
10	25.72	350.14	-	0.53	50
15	447.32	3674.34	-	7.08	50
20	2437.36	20767.40	-	55.22	50
25	*32657.40*	*245163.00*	-	*875.73*	44

the number of calls to the function oracle, i.e. the number of times the combinatorial algorithms are called to solve the subproblems, quickly grows with the instance size. This approach highly relies on problem-specific algorithms for the minimization of submodular functions and efficient algorithms for the underlying linear combinatorial problems.

Chapter 8

Mean Risk Optimization

As a second application of the approaches for submodular combinatorial optimization developed in Chapter 3 we study the risk-averse capital budgeting problem. In portfolio theory an important concept is to not only consider the expected return when choosing a set of investments but also take into account the risk associated with investments. Such *mean-risk optimization problems* can be modeled using stochastic objective functions. Potential investment decisions are given by independent random variables that have an associated mean value μ as well as a variance σ^2. The mean value stands for the expected return of the investments, σ^2 models the uncertainty inherent in the investment, i.e. the risk that the real return deviates from the expected. The case of continuous variables is well studied [88, 24], whereas the case of discrete variables has received relatively little attention yet.

We concentrate on the *risk-averse capital budgeting problem* with binary variables [10]. In this variant of the mean-risk optimization problem a set of possible investments characterized by their costs, expected return values and variances and a number ε are given as input. The number $\varepsilon > 0$ characterizes the level of risk the investor is willing to take. Investment decisions are binary, this means one can choose to make a certain investment or not. The only constraint in the risk-averse capital budgeting problem is a limit on the available budget. An optimal solution of the problem is a set of investment decisions and a solution value z. The choice of investments guarantees that with probability $1 - \varepsilon$ the portfolio will return at least a profit of z [37].

The corresponding nonlinear IP-model is

$$\begin{aligned} z = \max \quad & \sum_{i \in I} \mu_i x_i - \sqrt{\frac{1-\varepsilon}{\varepsilon} \sum_{i \in I} \sigma_i^2 x_i^2} \\ \text{s.t.} \quad & \sum_{i \in I} a_i x_i \leq b \\ & x \in \{0,1\}^I , \end{aligned} \tag{8.1}$$

where I is the set of available investments, a_i the cost of investment $i \in I$, and b the amount of capital that can be invested. The vector μ represents the expected returns of the investments and σ^2 the variance of the expected returns.

It is interesting to note that Problem (8.1) is equivalent to a special case of the *robust combinatorial optimization problem with ellipsoidal uncertainty*. Here, the set of possible manifestations of the uncertain objective function, the scenario set $\mathcal{U}$, takes the form of an ellipsoid in $\mathbb{R}^n$. The aim is to optimize the value of a solution in its worst-case scenario, i.e. to give the best possible protection against variations in costs.

The general *robust combinatorial optimization problem with ellipsoidal uncertainty* can be formulated as the min-max integer program

$$\begin{array}{lll} \min & \max\limits_{c \in \mathcal{U}} c^\top x & \\ \text{s.t.} & x \in X \subseteq \{0,1\}^n\,, & \end{array} \tag{8.2}$$

where X is the set of feasible solutions and $\mathcal{U}$ is given by an ellipsoid

$$\left\{(c - c_0)^\top Q (c - c_0) \leq 1\right\}$$

characterized by its center $c_0 \in \mathbb{R}^n$ and a positive definite matrix $Q \in \mathbb{R}^{n \times n}$. For given x, the optimum value of the inner optimization problem over $\mathcal{U}$ can be expressed in a closed form. We have

Theorem 8.1. *For fixed $x \in \mathbb{R}^n$, the value of an optimum solution $c^\star$ of*

$$\begin{array}{ll} \max & c^\top x \\ \textit{s.t.} & (c - c_0)^\top Q (c - c_0) \leq 1 \end{array} \tag{8.3}$$

is given by $c^{\star\top} x = c_0^\top x + \sqrt{x^\top Q^{-1} x}$.

Proof. Problem (8.3) optimizes a continuous linear function over a convex set. Its Lagrangean function is

$$L(c, \lambda) = c^\top x + \lambda (c - c_0)^\top Q (c - c_0) - 1,$$

where $\lambda \geq 0$ is a Lagrangean multiplier. According to the Karush-Kuhn-Tucker optimality conditions, any optimum solution $c^\star$ of (8.3) satisfies

1. $\nabla_{c^\star} L(c^\star, \lambda) = 0$ and
2. $\nabla_\lambda L(c^\star, \lambda) = 0\,.$

We have

$$\begin{aligned} &\nabla_{c^\star} L(c^\star, \lambda) = 0 \\ &\Leftrightarrow x + 2\lambda Q(c^\star - c_0) = 0 \\ &\Leftrightarrow c^\star - c_0 = -\frac{1}{2\lambda} Q^{-1} x\,, \end{aligned}$$

because Q is nonsingular and we can assume $\lambda > 0$, since otherwise $x = 0$ and the equality stated in Theorem 8.1 trivially holds.

Substituting $c^\star - c_0$ in the constraint of (8.3) gives

$$\begin{aligned} & (c^\star - c_0)^\top Q(c^\star - c_0) \leq 1 \\ \Rightarrow & x^\top Q^{-1} x = 4\lambda^2 \\ \Rightarrow & \lambda = \frac{1}{2}\sqrt{x^\top Q^{-1} x} \end{aligned}$$

and thus

$$\begin{aligned} & x^\top (c^\star - c_0) = -\frac{1}{2\lambda} x^\top Q^{-1} x \\ \Rightarrow & x^\top Q^{-1} x = 4\lambda^2 \\ \Rightarrow & {c^\star}^\top x = c_0^\top x - \frac{1}{2\lambda} x^\top Q^{-1} = c_0^\top x + \sqrt{x^\top Q^{-1} x} \end{aligned}$$

□

Theorem 8.1 shows that the *robust combinatorial optimization problem with ellipsoidal uncertainty* (8.2) is equivalent to

$$\begin{array}{rcl} \min & c_0^\top x + \sqrt{x^\top Q^{-1} x} & \\ \text{s.t.} & x \in X \subseteq \{0,1\}^n & \end{array} \tag{8.4}$$

It is now easy to see that the *risk-averse capital budgeting problem* (8.1), reformulated as a minimization problem, is the special case of (8.4) where the ellipsoid $\mathcal{U}$ is defined by the center $c_0 = -\mu$ and the matrix $Q = Diag(\sigma^2)$, and X is the set of points that satisfy the knapsack constraint $a^\top x \leq b$. Q in this case is a diagonal matrix, which means that the ellipsoid $\mathcal{U}$ is axis-parallel.

To apply the polyhedral results from Chapter 3 we need to rewrite the original model (8.1) as a minimization problem and show that the objective function is submodular. Note that since the x-variables are binary we have $x_i^2 = x_i$. The problem now reads

$$\begin{array}{rcl} z = -\min & -\sum_{i\in I} \mu_i x_i + \sqrt{\frac{1-\varepsilon}{\varepsilon} \sum_{i\in I} \sigma_i^2 x_i} & \\ \text{s.t.} & \sum_{i\in I} a_i x_i \leq b & \\ & x \in \{0,1\}^I\,. & \end{array} \tag{8.5}$$

The first part of the objective function

$$f(A) = -\sum_{i\in A} \mu_i + \sqrt{\frac{1-\varepsilon}{\varepsilon} \sum_{i\in A} \sigma_i^2}$$

is obviously modular. The second part is the composition of a nondecreasing modular function and a nondecreasing concave function. By Theorem 3.3 the composition of a nondecreasing submodular function and a nondecreasing concave function is submodular. Since all modular functions are also submodular, the following corrolary immediately follows from Theorem 3.3.

Corollary 1. *The objective function of model* (8.5) *is submodular.*

Corollary 1 shows that we can use the polyhedral results from Chapter 3 to solve the risk-averse capital budgeting problem (8.5). Indeed, Atamtürk and Narayanan [10] use the cutting planes (3.3) to strengthen the nonlinear formulation (8.1), which they then solve with a second order cone programming-approach.

Problem (8.5) can also again be decomposed into an unconstrained submodular minimization problem and a linear combinatorial problem, similarly to the Range Assignment Problem studied in Chapter 7.

The Lagrangean decomposition with multipliers $\lambda \in \mathbb{R}^I$ is

$$\begin{array}{rlcrl} -\min & -\sum_{i\in I}(\mu_i-\lambda_i)x_i+\sqrt{\frac{1-\varepsilon}{\varepsilon}\sum_{i\in I}\sigma_i^2 x_i} & + & \min & \sum_{i\in I}\lambda_i y_i \\ \text{s.t.} & x \in \{0,1\}^I & & \text{s.t.} & \sum_{i\in I} a_i y_i \le b \\ & & & & y \in \{0,1\}^I \end{array} \tag{8.6}$$

In this case the combinatorial problem is a knapsack problem with rational coefficients. The binary constraint on the y-variables can be relaxed to $0 \le y \le 1$, which is equivalent to solving a fractional knapsack problem instead of a binary knapsack problem. The obvious advantage of solving the relaxed problem is that it can be solved in linear time [11], while solving the binary knapsack problem would take pseudo-polynomial time [25]. The disadvantage is that the lower bound on the objective value of (8.5) obtained from the Lagrangean dual is weaker, as we have seen in Chapter 1.

For solving the unconstrained submodular subproblem of the decomposition we adapt an algorithm proposed by Ilyina [67]. It minimizes the unconstrained function

$$f(A) = -\sum_{i\in A}\bar{\mu}_i + \sqrt{\frac{1-\varepsilon}{\varepsilon}\sum_{i\in A}\sigma_i^2 i},$$

where $\bar{\mu} = \mu - \lambda$ and can handle fixed variables. The algorithm consists of two phases. In the first phase variables are fixed that can be included in or excluded from a solution from the start. In the second phase a set of candidates for the optimal solution is computed by first sorting the variables and then constructing a chain of solutions by iteratively adding elements in the sorted order. The candidate solution with the lowest objective value is optimal.

Algorithm 4 Minimization algorithm for $f(S) = -\sum_{i \in S} \bar{\mu}_i + \sqrt{\frac{1-\varepsilon}{\varepsilon} \sum_{i \in S} \sigma_i^2}$ with fixed variables

input: $\bar{\mu}, \sigma \in \mathbb{R}^n$, $\varepsilon \in \mathbb{R}$, $I = \{1, \ldots, n\}$, $I_0, I_1 \subseteq I$ (variables fixed to 0/1)
output: $S^\star \subseteq I$ which minimizes $f(S) = -\sum_{i \in S} \bar{\mu}_i + \sqrt{\frac{1-\varepsilon}{\varepsilon} \sum_{i \in S} \sigma_i^2}$ such that $I_0 \cap S = \emptyset$ and $I_1 \subseteq S$

$T \leftarrow \emptyset$
for $i \in I \setminus \{I_0 \cup I_1\}$ **do**
  if $\bar{\mu}_i < 0$ **then**
    $I_0 \leftarrow I_0 \cup \{i\}$ ▷ this element cannot improve the solution
  else if $\sqrt{\frac{1-\varepsilon}{\varepsilon}\sigma_i^2} < \bar{\mu}_i$ **then**
    $I_1 \leftarrow I_1 \cup \{i\}$ ▷ this element will always improve the solution
  else
    $n_i \leftarrow \left(\frac{\bar{\mu}_i^2 - \frac{1-\varepsilon}{\varepsilon}\sigma_i^2}{2\bar{\mu}_i}\right)^2$
    $T \leftarrow T \cup \{i\}$
  end if
end for

sort the elements i of T into a list $\{l_1, \ldots, l_{|T|}\}$ by nondecreasing value of n_i
$i \leftarrow 1$
$S_0 \leftarrow I_1$
repeat
  $S_i \leftarrow S_{i-1} \cup \{l_i\}$ ▷ construct candidate solutions
  $i \leftarrow i + 1$
until $i = |T|$
$S^\star \leftarrow \arg\min\{f(S_i) \mid i \in \{0, \ldots, |T|\}\}$ ▷ choose best solution
return $S^\star$

Theorem 8.2. *Algorithm 4 computes the minimizer of f in $\mathrm{O}(|I| \log |I|)$ time.*

Proof. Consider the function $f_i(d) = -\bar{\mu}_i + \sqrt{d + \frac{1-\varepsilon}{\varepsilon}\sigma_i^2} - \sqrt{d}$, where $\bar{\mu} = \mu - \lambda$. f_i describes the change in the value of f when element i is added to a set with variance d. Seen as a continuous function, f_i is convex and decreases monotonically. For each element i such that $\bar{\mu}_i < 0$, $f_i(d)$ is nonnegative for all positive d. Thus no such element can be part of a set which minimizes f. For each element i with $\sqrt{\frac{1-\varepsilon}{\varepsilon}\sigma_i^2} < \bar{\mu}_i$, $f_i(d)$ is negative for all positive d. Therefore such an element

must be part of every minimizer of f. For the remaining elements

$$n_i = \left(\frac{\bar{\mu}_i^2 - \frac{1-\varepsilon}{\varepsilon}\sigma_i^2}{2\bar{\mu}_i} \right)^2$$

is the unique zero of f_i. This means that

$$f_i(d) \leq 0 \iff d \geq n_i. \tag{8.7}$$

In the following we assume that the set F contains all fixed elements given in the input and the elements which are part of every optimal solution (as argued above), and we set $m_F := \sum_{i \in F} \bar{\mu}_i$ and $s_F := \sum_{i \in F} \frac{1-\varepsilon}{\varepsilon}\sigma_i^2 \geq 0$. Further we assume that the free elements are sorted by nondecreasing value of n_i and indexed accordingly. We show that there exists a minimizer $S^\star$ of the reduced function

$$f_F(A) := -m_F + -\sum_{i \in A} \bar{\mu}_i + \sqrt{s_F + \frac{1-\varepsilon}{\varepsilon} \sum_{i \in A} \sigma_i^2}$$

such that

$$\forall k \in S^\star \colon k > 0 \Rightarrow k - 1 \in S^\star \tag{8.8}$$

Let S be a minimizer of f_F with $i \in S$ and $i - 1 \notin S$ for some $i > 0$. Define $\bar{S} = S \setminus \{i\}$ and $S^\star = S \cup \{i-1\}$. S is optimal

$$\begin{aligned}
\Rightarrow f_F(S) - f_F(\bar{S}) &= -\bar{\mu}_i + \sqrt{s_F + \sum_{j \in S} \frac{1-\varepsilon}{\varepsilon}\sigma_j^2} - \sqrt{s_F + \sum_{j \in \bar{S}} \frac{1-\varepsilon}{\varepsilon}\sigma_j^2} \\
&= -\bar{\mu}_i + \sqrt{s_F + \sum_{j \in \bar{S}} \frac{1-\varepsilon}{\varepsilon}\sigma_j^2 + \frac{1-\varepsilon}{\varepsilon}\sigma_i^2} - \sqrt{s_F + \sum_{j \in \bar{S}} \frac{1-\varepsilon}{\varepsilon}\sigma_j^2} \leq 0 \\
&\overset{(8.7)}{\Rightarrow} s_F + \sum_{j \in \bar{S}} \frac{1-\varepsilon}{\varepsilon}\sigma_j^2 \geq n_i
\end{aligned} \tag{8.9}$$

Moreover,

$$\begin{aligned}
f_F(S^\star) - f_F(S) &= -\bar{\mu}_{i-1} + \sqrt{s_F + \sum_{j \in S^\star} \frac{1-\varepsilon}{\varepsilon}\sigma_j^2} - \sqrt{s_F + \sum_{j \in S} \frac{1-\varepsilon}{\varepsilon}\sigma_j^2} \\
&= -\bar{\mu}_{i-1} + \sqrt{s_F + \frac{1-\varepsilon}{\varepsilon}\left(\sum_{j \in \bar{S}} \sigma_j^2 + \sigma_i^2 \right) + \frac{1-\varepsilon}{\varepsilon}\sigma_{i-1}^2} - \sqrt{s_F + \frac{1-\varepsilon}{\varepsilon}\left(\sum_{j \in \bar{S}} \sigma_j^2 + \sigma_i^2 \right)} \\
&\leq 0,
\end{aligned}$$

because by (8.9) and the sorting of the elements we have

$$s_F + \sum_{j \in \bar{S}} \frac{1-\varepsilon}{\varepsilon}\sigma_j^2 + \frac{1-\varepsilon}{\varepsilon}\sigma_i^2 \geq s_F + \sum_{j \in \bar{S}} \frac{1-\varepsilon}{\varepsilon}\sigma_j^2 \geq n_i \geq n_{i-1}$$

and (8.7) holds. Since S was assumed to be a minimizer of f_F, $S^\star$ must be a minimizer too, which proves the hypothesis.

Algorithm 4 enumerates the solutions with property (8.8) and chooses the best solution among those. The running time of the algorithm is determined by the time to sort the elements, which takes $O(|I| \log |I|)$ time, when an appropriate sorting algorithm is used. The objective values of the candidate solutions $f(S_i)$ can be determined in linear time by computing the partial sums over the μ_i and the σ_i^2 incrementally. □

Remark 8.1. Shen et al. [115] also proposed an algorithm for the unconstrained problem described above. The only difference to the algorithm by Ilyina [67] is the sorting criterion in the second phase. With a similar argument they prove that the optimal solution is among the set of candidates that is obtained when the variables are sorted such that $\frac{\bar{\mu}_1}{\sigma_1^2} \leq \frac{\bar{\mu}_2}{\sigma_2^2} \leq \cdots \leq \frac{\bar{\mu}_k}{\sigma_k^2}$.

8.1 Computational Results

In the following, we report the results of a computational study of the risk-averse capital budgeting problem. We compare the performance of the branch and bound-algorithms based on the cutting plane approach of Section 3.4 and the Lagrangean decomposition approach of Section 3.5 with the performance of the commercial solver GUROBI 5.5 [56].

Experimental Setup

As in Chapter 7 the branch and cut-algorithm was implemented in SCIL and CPLEX 12.5 [64] was used to solve the LP-relaxations. We used a simple rounding heuristic to generate primal bounds in the branch and bound-algorithm. Whenever a fractional LP-solution is computed, we round all values to the nearest integer. Then we construct a feasible solution by successively setting the variables to the rounded values as long as the knapsack constraint is not violated. All remaining variables are set to zero.

The decomposition approach again used the ConicBundle library to compute the Lagrangean duals. In each node of the branch and bound-tree the left subproblem of the Lagrangean decomposition (3.16) is solved with Algorithm 4 and the right subproblem, i.e. the fractional knapsack problem, is solved with the greedy

algorithm by Dantzig [25]. The index of the branching variable is chosen as the lowest index with $\lambda_i^\star \neq 0$ and $(x_1^\star)_i \neq (x_2^\star)_i$, as proposed in Section 3.5.2.

Several commercial solvers are able to solve problems of type (8.1). According to recent benchmarks [95], CPLEX and GUROBI outperform the other solvers. We tested both CPLEX and GUROBI and found that GUROBI is slightly faster than CPLEX for our problem instances. Therefore in the following we only give the results obtained with GUROBI, which applies a second order cone programming-algorithm to directly solve the nonlinear formulation (8.1) of the problem. In our experiments we used the default settings.

All experiments were run on an Intel Xeon CPU E5–26400@2.5GHz with 32Gb main memory.

Instances

Atamtürk and Narayanan [10] describe a procedure to generate random instances of the risk-averse capital budgeting problem. The expected returns μ and the costs a are chosen as independent random numbers between 0 and 100. The variances σ are set as the expected returns multiplied by an independent random number between 0 and 1. The available budget is $\frac{1}{2}\sum_{i\in I} a_i$. This ensures the existence of nonzero solutions and at the same time excludes the trivial case where the budget is large enough to make all investments.

We solved the set of random instances from [10], which have between 25 and 100 variables. The larger instances with up to 2000 variables were generated using the same method as for the smaller instances.

We generated five instances of each size and solved each instance for the values of ε given in the tables. All tables show the average number of subproblems and the average running time for each class of instances. The fifth column of Table 8.1 contains the number of simplex iterations of the second order cone-approach, whereas in the fourth column of Table 8.2 the average number of LPs in the cutting plane algorithm is given. The fourth column of Table 8.3 shows the average number of calls to the oracles in the decomposition approach, i.e. the number of times both subproblems in (8.6) were solved.

Since with the SOCP-approach it was not possible to compute optimal solutions for all instances within a time limit of ten hours, the third column of Table 8.1 lists how many of the five instances per class could be solved to optimality within this time limit. All averages in this table, including the running time, are taken only over these instances.

Table 8.1: Results for the risk-averse capital budgeting problem, GUROBI SOCP

n	ε	#solved	subs	iter	time/s
25	0.10	5	605.20	1100.80	0.06
	0.05	5	1782.60	3471.20	0.22
	0.03	5	3646.80	8402.00	0.55
	0.02	5	2806.80	7611.80	0.45
	0.01	5	966.40	2681.60	0.26
50	0.10	5	3199.60	5085.00	0.37
	0.05	5	17202.00	25223.20	3.56
	0.03	5	39927.80	64839.60	14.72
	0.02	5	107358.60	218805.80	107.22
	0.01	5	416434.80	1370784.40	4115.62
100	0.10	5	13262.80	18518.60	2.23
	0.05	5	138222.20	201892.40	249.42
	0.03	5	1233468.40	1877895.20	13730.00
	0.02	0	–	–	–
	0.01	0	–	–	–
200	0.10	5	38140.80	48803.20	11.99
	0.05	5	340421.60	477580.00	1021.29
	0.03	3	2807775.67	5765215.33	33636.56
	0.02	0	–	–	–
	0.01	0	–	–	–
300	0.10	5	155785.40	185740.60	127.82
	0.05	3	849511.33	1112535.00	3676.86
	0.03	1	3622994.00	5556223.00	29592.24
	0.02	0	–	–	–
	0.01	0	–	–	–
400	0.10	5	134944.40	188986.40	154.18
	0.05	3	383379.67	612808.33	1316.04
	0.03	0	–	–	–
	0.02	0	–	–	–
	0.01	0	–	–	–

Table 8.2: Results for the risk-averse capital budgeting problem, polyhedral approach

n	ε	subs	LPs	time/s	n	ε	subs	LPs	time/s
25	.10	42.20	114.20	0.06	500	.10	291.00	743.80	4.89
	.05	36.20	153.40	0.08		.05	617.00	3400.40	27.97
	.03	19.40	138.80	0.07		.03	590.20	4665.80	52.45
	.02	6.20	71.60	0.03		.02	845.40	9890.80	135.26
	.01	4.60	35.60	0.01		.01	1580.60	44116.00	965.59
50	.10	98.60	259.60	0.20	600	.10	502.60	1563.60	13.48
	.05	89.00	353.00	0.31		.05	665.40	3163.60	33.56
	.03	43.80	270.40	0.25		.03	465.80	4505.20	66.80
	.02	53.40	489.20	0.49		.02	1152.20	12538.00	220.87
	.01	14.60	489.40	0.64		.01	1028.20	30214.80	899.44
100	.10	139.80	357.20	0.49	700	.10	920.20	3592.60	38.82
	.05	104.60	567.80	0.99		.05	661.40	3476.20	44.99
	.03	179.80	1468.00	3.16		.03	782.20	7568.00	132.55
	.02	188.20	1488.60	3.31		.02	943.80	12955.80	277.36
	.01	135.40	3148.20	9.20		.01	1058.60	28470.00	1062.73
200	.10	205.80	483.00	1.29	800	.10	551.80	1602.20	17.37
	.05	225.80	800.20	2.53		.05	912.60	5054.60	76.53
	.03	185.40	1249.60	5.39		.03	923.80	7090.60	129.10
	.02	245.80	2703.20	14.61		.02	1336.20	17830.60	426.56
	.01	313.80	12933.60	117.74		.01	1586.20	60431.80	2564.00
300	.10	285.00	1135.20	4.87	900	.10	542.20	1159.80	12.41
	.05	226.20	1169.20	6.15		.05	371.80	1493.60	23.64
	.03	271.40	2243.60	14.79		.03	605.00	3355.80	69.44
	.02	317.40	4397.60	38.43		.02	1686.20	7538.60	167.58
	.01	255.80	12716.80	203.86		.01	3138.60	59090.40	2200.07
400	.10	398.20	1322.40	7.56	1000	.10	840.20	1670.40	20.38
	.05	486.20	3444.40	26.48		.05	699.80	1798.60	25.75
	.03	683.80	6276.80	56.80		.03	270.60	1237.00	26.11
	.02	241.40	4735.60	56.90		.02	1133.80	6841.20	181.80
	.01	1482.20	33600.20	584.26		.01	1241.00	31882.80	1433.39

Results

Running times for the SOCP-approach increase rapidly with growing instance size and increasing values of ε. Instances with 100 variables or more can not be solved for all values of ε, even with a time limit of ten hours. Table 8.1 shows that already for $n = 100$ no instances could be solved to optimality for $\varepsilon = 0.02$ and $\varepsilon = 0.01$. For larger n the only case where all instances could be solved was $\varepsilon = 0.1$. The large number of subproblems indicates that the dual bounds obtained from the relaxations are weak. In fact, good primal bounds are found quickly, but proving optimality is difficult.

In contrast, both the polyhedral approach and the decomposition approach can solve much larger instances, as can be seen from Tables 8.2 and 8.3. The polyhedral approach from Chapter 3 that exploits the submodularity of the objective function can easily solve instances with up to 1000 variables for all values of ε that were tested. While again an increase in running times with decreasing ε can be observed, it is by far not as drastic as for the SOCP-approach. The number of subproblems and the number of LPs generally grows moderately with instance size and decreasing ε, which indicates that the dual bounds obtained with separation algorithm 2 are strong, but the approach is nevertheless sensitive to the balance between the linear and the nonlinear parts of the objective function.

This effect is nearly absent in the Lagrangean decomposition approach, as can be observed in Table 8.3. There is no clear correlation between the scaling factor ε and the number of subproblems or the average running times. In comparison to the cutting plane-algorithm the number of subproblems is higher, but this fact is compensated by the reduction in time spent in each subproblem.

For the root node, the computation time was additionally be reduced by setting the initial Lagrangean multiplier to $\lambda_0 = -\mu$, which gives slightly better results than the obvious $\lambda_0 = 0$. As for the Range Assignment Problems of Chapter 7, the other nodes inherit the best multiplier from their respective parent nodes.

If we take a closer look at the ratio between the number of calls to the combinatorial algorithms for the two parts of the decomposition (8.6) and the number of subproblems, we see that only few calls per subproblem are necessary. For the instances with $n = 2000$, for example, this ratio is less than four. Additionally, the algorithms applied in the subproblems are of low theoretical complexity and fast in practice. In combination with strong primal bounds – we get a feasible solution in each iteration of the subgradient algorithm – this leads to the low overall running times. The decomposition approach outperforms the cutting plane-algorithm in nearly all cases and can solve larger instances in the same time. The algorithm is also less sensitive to the parameter ε.

Remark 8.2. Atamtürk and Narayanan [10] present a solution approach that solves the above model using second-order cone programming embedded in a

Table 8.3: Results for the risk-averse capital budgeting problem, decomposition approach

n	ε	subs	calls	time/s	n	ε	subs	calls	time/s
25	0.10	103.40	388.60	0.01	500	0.10	3764.20	8467.00	2.86
	0.05	78.20	343.60	0.01		0.05	5337.40	12652.00	4.20
	0.03	29.80	665.40	0.06		0.03	4363.80	9731.80	3.25
	0.02	12.60	75.60	0.00		0.02	5738.20	12347.60	4.07
	0.01	11.00	65.80	0.00		0.01	7779.00	17511.40	5.86
50	0.10	268.60	1052.60	0.04	600	0.10	5566.20	12798.60	5.86
	0.05	199.80	821.00	0.04		0.05	8457.00	18898.20	8.48
	0.03	132.60	530.80	0.02		0.03	3793.80	8313.00	3.70
	0.02	142.20	2008.80	0.19		0.02	9456.60	21040.00	9.38
	0.01	33.00	8599.80	1.15		0.01	3911.00	8197.20	3.65
100	0.10	701.00	2034.60	0.12	700	0.10	9085.40	21087.20	12.36
	0.05	420.60	1313.60	0.08		0.05	8726.20	19324.20	11.20
	0.03	484.60	1547.40	0.09		0.03	7429.80	16164.40	9.13
	0.02	449.40	1475.40	0.09		0.02	8977.00	19930.40	11.32
	0.01	395.00	2388.00	0.24		0.01	4779.00	10088.20	5.79
200	0.10	1295.00	3310.00	0.36	800	0.10	5859.80	12957.20	9.38
	0.05	1557.00	3997.80	0.43		0.05	10795.00	23842.60	16.55
	0.03	1207.80	2989.00	0.33		0.03	11394.60	24199.40	16.41
	0.02	1071.80	2576.80	0.28		0.02	8179.80	16444.20	11.28
	0.01	1074.60	2772.60	0.30		0.01	6958.20	15276.00	10.36
300	0.10	2647.00	6605.20	1.15	900	0.10	3686.60	7990.40	6.54
	0.05	1446.60	3453.60	0.60		0.05	4300.20	9622.40	7.51
	0.03	1407.00	3297.60	0.59		0.03	9267.00	20492.40	15.85
	0.02	1298.60	3019.00	0.54		0.02	16574.20	36782.60	29.76
	0.01	975.80	2145.80	0.38		0.01	24477.40	53935.40	41.62
400	0.10	3933.40	8622.60	2.17	1000	0.10	16766.60	36227.20	34.47
	0.05	5303.80	12694.40	3.16		0.05	15733.80	34835.80	33.16
	0.03	5393.80	12429.20	3.18		0.03	5003.80	11018.60	10.36
	0.02	1019.00	2253.20	0.57		0.02	14013.00	31220.80	28.83
	0.01	5468.60	11785.20	3.00		0.01	13877.40	30347.80	27.83
1500	0.10	25454.20	56865.80	115.10	2000	0.10	37377.00	83695.40	282.09
	0.05	26775.80	60294.40	118.91		0.05	54840.20	126360.20	424.96
	0.03	29549.00	67257.40	126.71		0.03	13790.60	31673.40	103.93
	0.02	21182.20	46259.40	90.89		0.02	12730.60	28184.00	90.83
	0.01	40960.20	93365.40	166.26		0.01	61470.60	141114.80	449.38

branch and bound-algorithm. They use the inequalities of Theorem 3.6 to strengthen the relaxation in each node of the enumeration tree. Their results show that using the additional cutting planes significantly improves the dual bounds and leads to a much lower number of subproblems and quicker solution times. Still, their approach is not competitive with the ones presented here. Solving the instances with $n = 100$ and $\varepsilon = 0.01$ took more than 800 seconds on average.

8.2 Final Remarks

In summary we could show that both the cutting plane and the decomposition algorithm are well suited for the risk-averse capital budgeting problem. They clearly outperform the standard approach applied by general-purpose MIP solvers.

We saw in Chapter 8 that for range assignment problems no algorithm consistently dominated the others. In the case of the risk-averse capital budgeting problem the situation is different. The Lagrangean decomposition approach clearly is the best of the three methods tested. It can solve even instances with a large number of variables quickly and is largely insensitive to the scaling of the nonlinear part of the objective function.

Chapter 9

Two-Scenario Optimization

In this chapter we evaluate the exact algorithms for unconstrained and combinatorial two-scenario optimization we presented in Chapter 4 experimentally. Recall that we proposed two branch and bound-approaches. In both cases, instead of minimizing the nonlinear objective function directly, we solve two knapsack problems. In the presence of combinatorial constraints, the problem is first decomposed into an unconstrained two-scenario problem and a linear combinatorial problem. Lower bounds are then computed by solving the Lagrangean dual problem.

All computational experiments in this chapter were conducted on an Intel Xeon CPU E5–26400@2.5GHz with 32Gb main memory, running Ubuntu 12.04.1 LTS.

9.1 Unconstrained Two-Scenario Optimization

Recall that in Chapter 4 the unconstrained two-scenario optimization problem (4.2) was given as

$$\min_{\substack{l \leq x \leq u \\ x \in \mathbb{Z}^n}} \max\{a^\top x + a_0, b^\top x + b_0\},$$

with $a, b \in \mathbb{R}^n$, $a_0, b_0 \in \mathbb{R}$ and $l, u \in \mathbb{Z}^n$.

In our experiments, we consider three different domains for the variables:
$x \in \{0,1\}^n$, $x \in \{-10,10\}^n$ and $x \in \{-50,50\}^n$.

We implemented the branch and bound-algorithm proposed in Section 4.1.3. Dual bounds are computed with the modified algorithm for the fractional knapsack problem described in Section 4.1.2, primal bounds with the rounding heuristic proposed in 4.1.3.

To evaluate the approach presented above, we compare it with the direct ILP approach. Problem (4.2) can be formulated as an ILP by introducing an artificial variable y.

Table 9.1: Comparison of the knapsack and ILP approaches for the unconstrained two-scenario problem, bounds 0/1.

	FKP		ILP	
n	#nodes	t_s	#nodes	t_s
500	5159.40	0.07	270.70	0.22
1000	13265.70	0.36	422.00	0.58
1500	21924.30	0.93	503.90	1.00
2000	51360.80	2.97	1428.40	3.81
2500	50236.80	3.68	1816.70	6.13
3000	79053.10	7.19	2567.10	10.51
3500	82996.90	9.15	1827.20	8.46
4000	110991.20	14.27	2334.20	12.76
4500	104578.50	15.46	3389.40	21.73
5000	132498.80	22.20	4278.90	29.03

The maximum term can then be linearized as follows:

$$\begin{array}{lrcl} \min\limits_{y} & y & & \\ \text{s.t.} & y & \geq & a^\top x + a_0 \\ & y & \geq & b^\top x + b_0 \\ & l \leq & x & \leq u \\ & x & \in & \mathbb{Z}^n \\ & y & \in & \mathbb{R} \end{array} \tag{9.1}$$

We solved the ILP (9.1) with a branch and bound-algorithm implemented in SCIL [114]. The LP-relaxation in each node was solved with CPLEX 12.1 [66]. To find good primal bounds we used the same rounding heuristic as in the knapsack approach.

Instances

We solved ten classes of instances, ranging in size from 500 to 5000 variables. For each size, twenty instances were generated with coefficients for the two scenarios chosen randomly as real numbers from the interval $[-100, 100]$. Each instance was solved with the three different domains for the variables mentioned above.

Table 9.2: Comparison of the knapsack and ILP approaches for the unconstrained two-scenario problem, bounds −10/10.

	FKP		ILP	
n	#nodes	t_s	#nodes	t_s
500	8468.30	0.11	336.50	0.28
1000	28847.95	0.81	923.50	1.35
1500	55293.95	2.28	2506.40	5.70
2000	94947.25	5.35	3440.40	11.55
2500	92060.70	6.57	2558.40	9.51
3000	106252.40	9.38	2573.90	10.90
3500	122728.90	13.08	3555.30	18.60
4000	151346.90	18.97	3210.40	18.25
4500	191696.20	27.32	3638.90	23.40
5000	217933.10	35.44	5156.20	35.61

Computational Results

Tables 9.1 and 9.2 show the results for the binary case ($0 \leq x \leq 1$) and the case $-10 \leq x \leq 10$. The results for $-50 \leq x \leq 50$ are shown in Table 9.3 on page 136. In each table, the first column contains the number of variables n. The remaining columns contain the average number of nodes in the branch and bound-tree and the average CPU time in seconds, for the exact algorithm based on the fractional knapsack problem (FKP) and the ILP approach (ILP), respectively.

For all three types of instances the number of nodes in the branch and bound-tree is much larger in the knapsack approach than in the ILP approach. This, however, in most cases does not lead to longer running times, since solving the relaxations with the fractional knapsack-algorithm is much quicker than solving the corresponding LP. With bounds zero and one both approaches solve instances with up to 5000 variables in less than thirty seconds on average. The knapsack approach is slightly faster for most sizes. The picture is similar for bounds −10/10 and −50/50. Remarkably, running times for both algorithms do not increase significantly for larger domains. This can be explained by the fact that in the optimal solutions the vast majority of variables is at either its lower or upper bound. For example, for the instances with 5000 variables and bounds −50/50 the average number of variables not on one of the bounds in our experiments was 3.75, the maximum number was 5. This means that the transformation to bounds 0/1 does not have a negative impact on the quality of the bound given by the fractional knapsack problems.

Table 9.3: Comparison of the knapsack and ILP approaches for the unconstrained two-scenario problem, bounds $-50/50$.

	FKP		ILP	
n	#nodes	t_s	#nodes	t_s
500	10197.15	0.13	561.80	0.48
1000	24888.50	0.78	1133.70	1.59
1500	57219.45	2.32	3252.30	7.64
2000	91590.80	5.08	3236.40	9.34
2500	96099.05	6.84	2018.00	6.95
3000	106036.15	9.33	3659.40	15.73
3500	149411.25	15.82	3172.80	15.69
4000	171866.15	21.30	3860.50	22.45
4500	238739.20	34.08	4648.20	30.09
5000	226777.40	36.61	5498.20	39.51

9.2 Two-Scenario Minimum Spanning Tree

In this section we apply the general results for combinatorial two-scenario optimization of Section 4.2 to a specific application, the *two-scenario minimum spanning tree problem*. As in the linear minimum spanning tree problem, we are given an undirected graph and the task is to choose a minimum-cost subset of the edges such that the resulting subgraph includes all nodes, is connected and cycle-free. In contrast to the linear version of the problem, there is not a single set of edge costs, but two sets which determine the two scenarios in the objective function. While the linear MST is well-known to be solvable in polynomial time and several efficient combinatorial algorithms exist, the two-scenario case is *NP*-hard [77], but solvable in pseudo-polynomial time [2].

In the following we compare two branch and bound-approaches based on the two relaxations (4.11) and (4.12) with a branch and cut-approach based on the standard linearization of the non-linear objective function, i.e. we solve the ILP (4.8) with X the set of spanning trees. We use the subtour-formulation of the spanning tree polytope described in Chapter 6.1, where we used it to model the quadratic minimum spanning tree problem. Starting with the cardinality constraint, violated subtour elimination constraints are added to the model dynamically. In the ILP approach we branch on the most fractional variable and use best first-enumeration. This algorithm was implemented in SCIL.

In the decomposition approaches the combinatorial subproblem is a linear MST-problem. It is solved with a modified version of Kruskal's algorithm which can handle fixed variables. It consists of two passes of the original algorithm by

Kruskal over the graph. In the first pass the algorithm only considers edges corresponding to variables fixed to one. If the resulting subgraph contains cycles, the subproblem is infeasible. In the second pass all remaining edges which correspond to variables not fixed to zero are considered.

The Lagrangean duals are computed with the ConicBundle library [58], which calls the algorithms for MST and FKP in each iteration of the subgradient algorithm.

From the set of branching variable candidates we select the variable with the largest absolute value of the corresponding Lagrangean multiplier, since it turned out that for two-scenario MST this in most cases gives slightly better running times than choosing the variable with the lowest index.

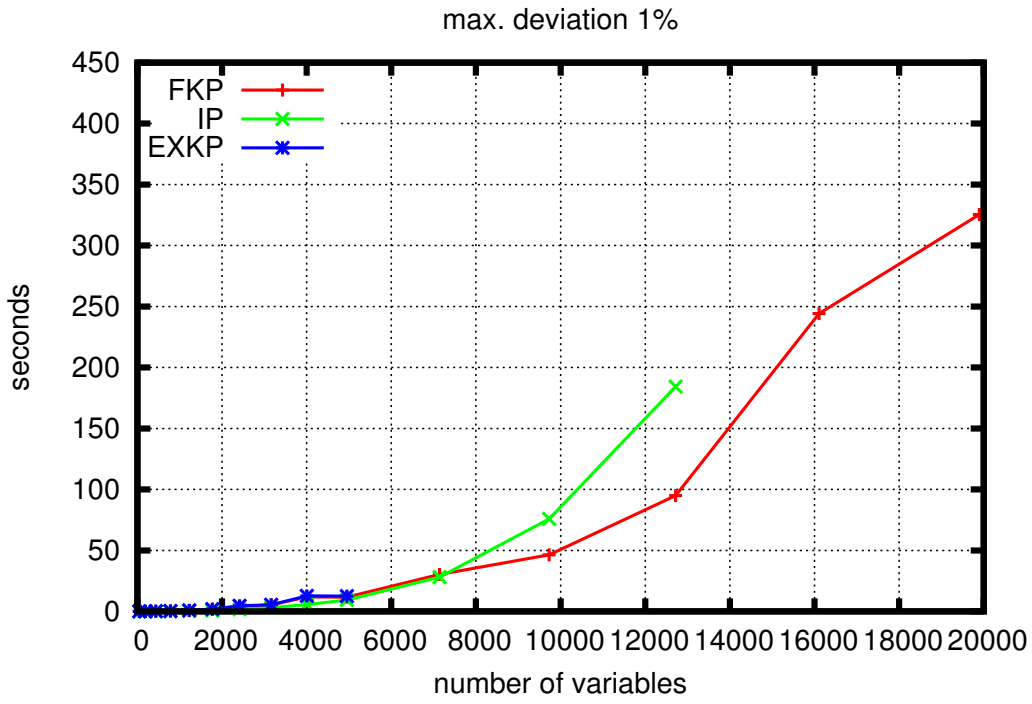

Figure 9.1: Comparison of running times for the two-scenario MST with 1% deviation.

Instances

We generated complete undirected graphs of different sizes in the following way: Nodes are placed randomly in the plane with coordinates in $[0, 100] \times [0, 100]$. Choosing the edge weights as the Euclidian distances between the end nodes of the edges induces the so-called *nominal scenario* which is the basis for five

classes of instances. Each class of instances is characterized by the percentage the edge costs in the two scenarios are allowed to diverge from the edge costs in the nominal scenario. We chose the percentages 1%, 5%, 10%, 50% and 100%. Thus, in the last case edge costs may effectively vary randomly between zero and double the edge costs in the nominal scenario. This scheme mimics the real-life situation where costs are estimated in advance, but uncertainties have to be taken into account that influence the actual costs. The percentage of deviation models the amount of uncertainty in the model.

Instance sizes range from 10 to 200 nodes. Since we consider complete graphs, the largest instances have 19900 edges. For each percentage of deviation from the nominal scenario and each size 20 instances were created.

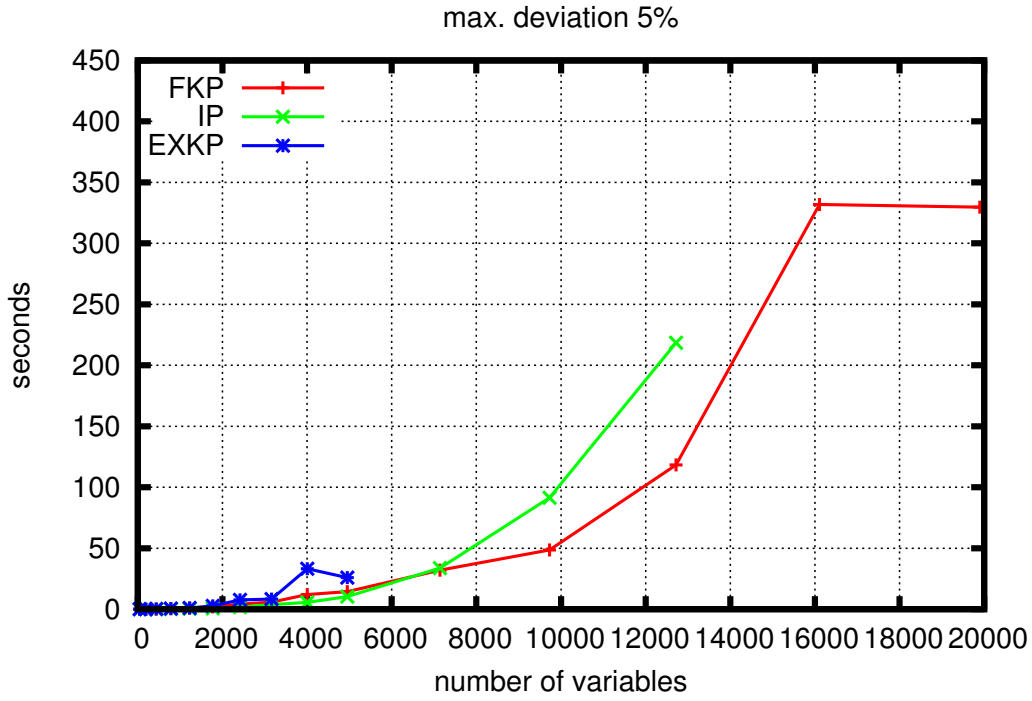

Figure 9.2: Comparison of running times for the two-scenario MST with 5% deviation.

Computational Results

The figures in this section show the running times of the three approaches in relation to the instance size. We set a time limit of 1h per instance. The decomposition approach that solves the integer unconstrained two-scenario problem solved all instances with up to 100 nodes (= 4950 edges/variables), but failed

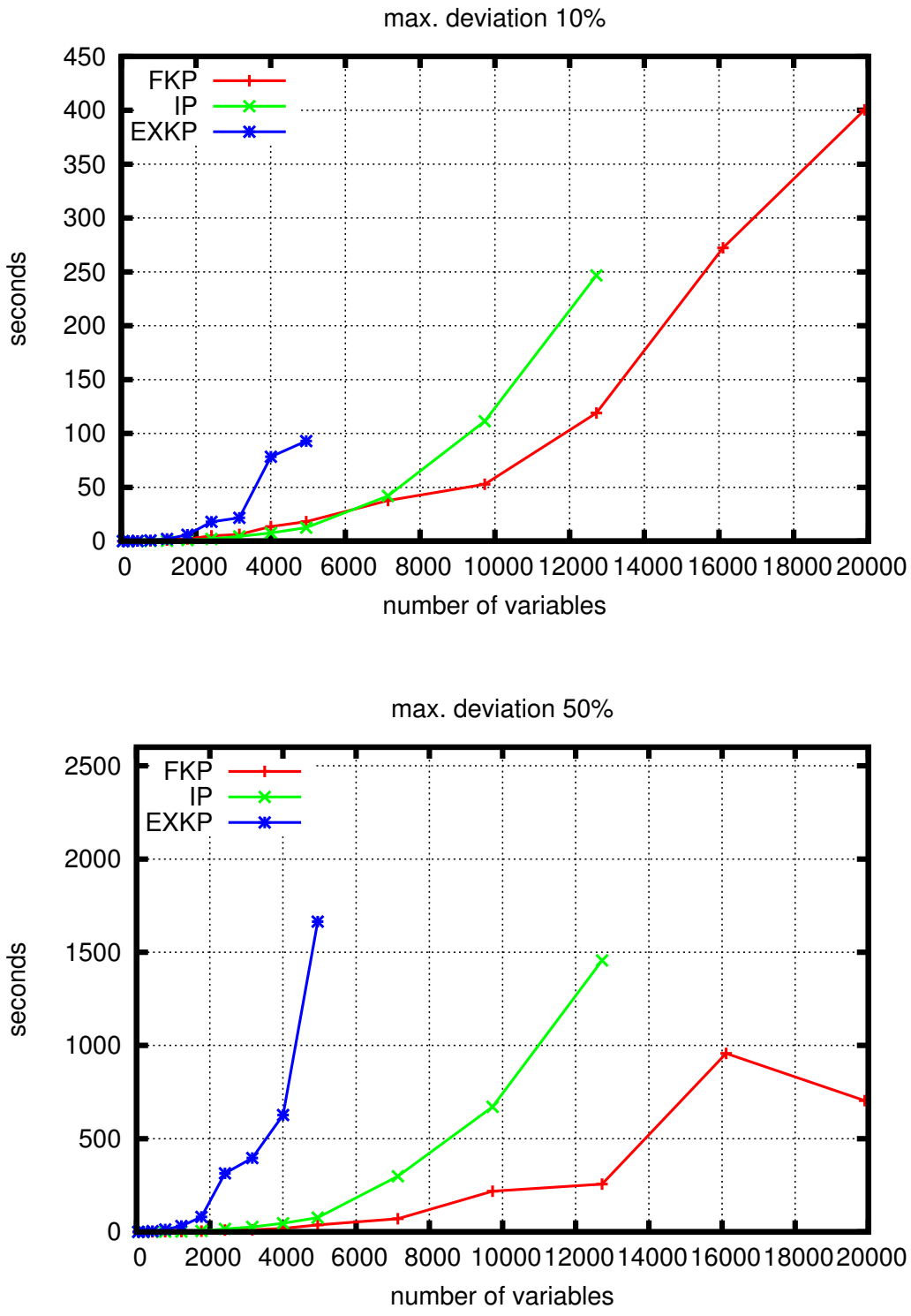

Figure 9.3: Comparison of running times for the two-scenario MST with 10% deviation (top) and 50% deviation(bottom).

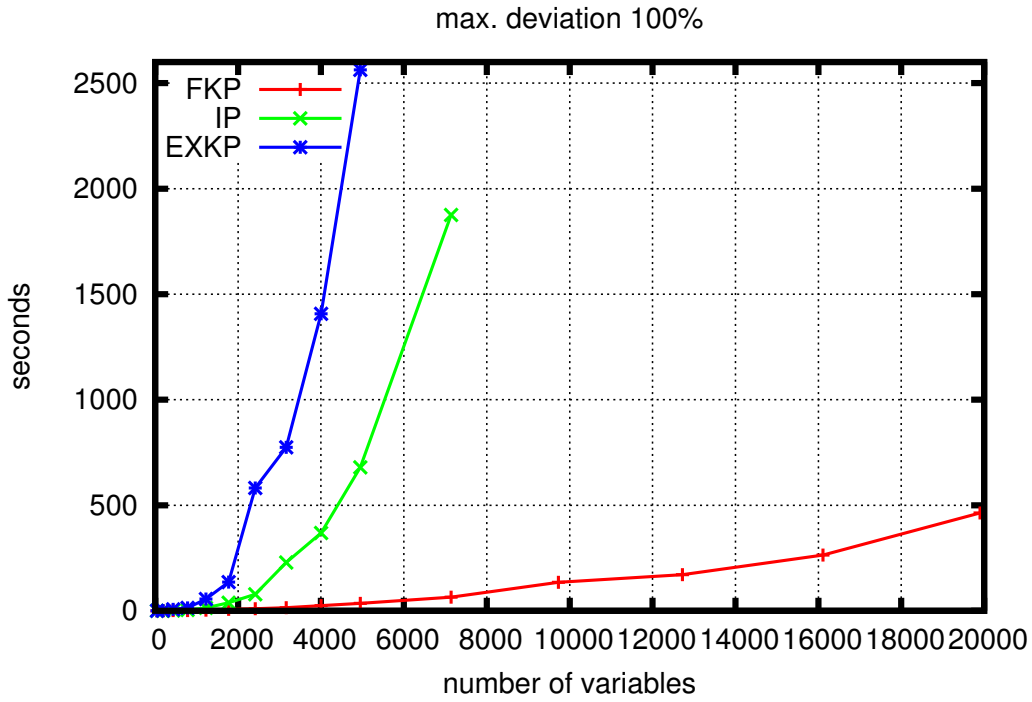

Figure 9.4: Comparison of running times for the two-scenario MST with 100% deviation.

on some of the larger instances. Similarly, the ILP approach was able to solve all instances with up to 160 nodes, except for 100% deviation. Here the largest instance size was 120.

As can be seen from the figures 9.1 – 9.4, running times for the smaller instances are very similar for all three approaches. This behavior can be observed for all classes of instances. For larger instances there are significant differences in performance. Clearly, the fractional knapsack-approach (FKP) outperforms the other two for all classes of instances. Solving the knapsack problems on integer variables in each iteration of the bundle algorithm (EXKP) theoretically gives stronger bounds, but this advantage is completely compensated by the comparatively large amount of time spent in each branch and bound-node.

Analyzing the running times for EXKP in relation to the deviation from the nominal scenario, a clear trend emerges: the larger the uncertainty in the data, the more difficult the instances become.

The ILP approach (IP) in general is faster than EXKP and the impact of the instance size on running times is not as severe. The largest difference occurs for deviation 50%. Here running times for EXKP are between 8 and 22 times as high, while for 10% and 100% the factor lies between 4–11 and 3–8, respectively.

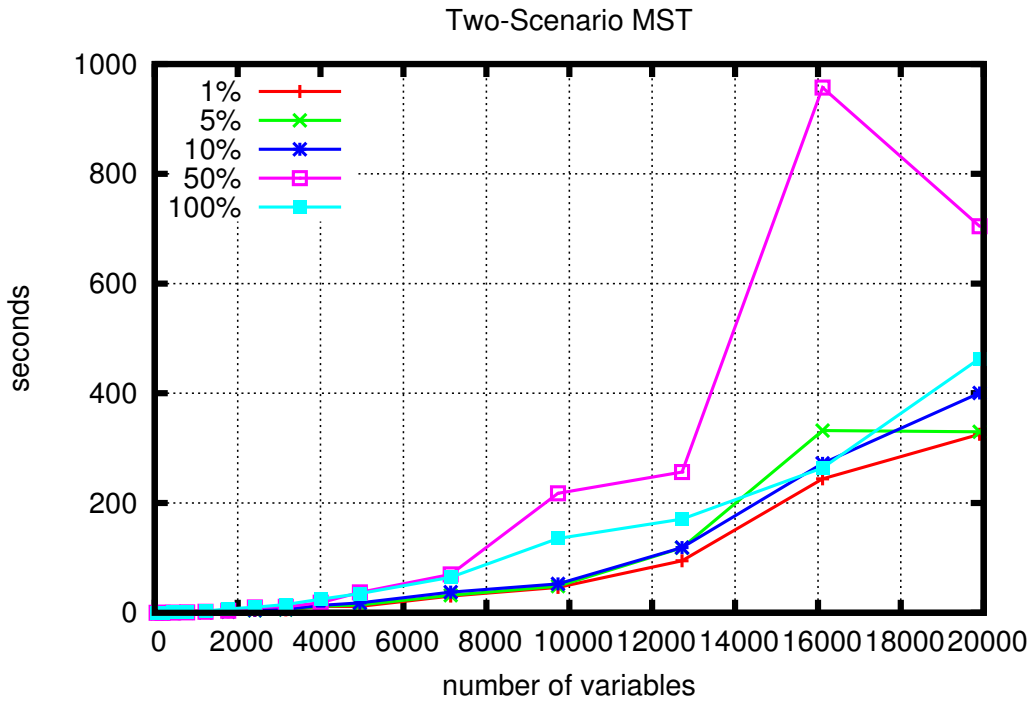

Figure 9.5: Comparison of running times for the two-scenario MST for different deviations, decomposition approach (4.12).

Clearly the fastest algorithm is FKP, which is based on the fractional knapsack algorithm. Even the largest instances of all classes are solved in less than 1000 seconds on average. Figure 9.5 shows the impact of the deviation on CPU times for FKP. In contrast to the other approaches, the most difficult instances for FKP are not the ones with the largest deviation, but those with 50%. This indicates that FKP can better handle the situation when one scenario clearly dominates the other.

The experimental evaluation of the decomposition approaches shows that, although the two-scenario minimum spanning tree problem is *NP*-hard, in practice it is possible to solve relatively large instances. Clearly, the better quality of the bounds from (4.12) is only an advantage in theory. Solving a single fractional knapsack problem instead of solving an integer knapsack problem by branch and bound in each step of the bundle algorithm leads to savings in computing time that easily compensate for the larger number of subproblems.

Part III

Nonlinear Optimization with SCIL

In the final part of this thesis we will give a brief overview of the C++-library SCIL, which was used extensively in the computational studies presented in Part II. After a short discussion of the basic principles in the design of SCIL, we will focus on the implementation of the exact polyhedral approaches for nonlinear binary optimization problems that were presented in Chapters 2, 3, 6, 7, 8 and 9.

SCIL stands for *Symbolic Constraints in Integer Linear Programming*. It is an object-oriented framework written in C++ and aims at the easy implementation of exact optimization algorithms for integer programming problems [4]. The framework provides interfaces for modeling an application as an integer program and solving the resulting model with a branch and cut-algorithm. In contrast to other optimization libraries, SCIL provides a set of pre-defined *symbolic constraints* for modeling common combinatorial structures like tours, matchings or spanning trees. Symbolic constraints can be freely combined with linear inequalities. Models can be formulated in graph-theoretic terms with the help of the boost graph library [117], which is also used in the separation algorithms provided by the symbolic constraints. Once an application has been modeled, the underlying integer program is solved with the LP-based branch and cut-system ABACUS [72, 38].

The modular design of SCIL, and especially the concept of symbolic constraints, make it a natural choice for the practical implementation of the polyhedral approaches for submodular and quadratic combinatorial optimization problems developed in this thesis. The basic idea of the branch and cut-algorithms in Chapters 2 and 3 is to combine a good polyhedral description of the combinatorial structure with a good linear model of the nonlinear objective function. In SCIL, different parts of a mathematical model, such as the (nonlinear) objective function and the combinatorial constraints, can be treated independently in different symbolic constraints. Furthermore, SCIL already provides separation routines for a wide range of combinatorial structures and, in combination with ABACUS, a complete branch and bound-framework for combinatorial optimization, which facilitates the experimental evaluation of the proposed algorithms.

Symbolic Constraints

SCIL is easily expandable. In addition to combining existing symbolic constraints to model more complex combinatorial structures, it is possible to define new symbolic constraints. The three functions that make up a symbolic constraint are the following:

Initialization This function is optional. It is used to add globally valid constraints to the initial relaxation in the root node of the branch and bound-tree.

Feasibility Test This function is called when the optimal solution of the current LP-relaxation is integer. It decides whether the solution is feasible for the symbolic constraint. If not, the separation routine of the symbolic constraint is called.

Separation This function is called in each iteration of the cutting plane-algorithm. Given a fractional LP-solution, it returns one or more inequalities that cut off the fractional point, or 'no constraint found'.

To give an example of the roles of these basic functions, consider the symbolic constraint `spanning tree`. It was used in Chapter 6 to model the *quadratic minimum spanning tree problem*, in Chapter 7 for the *symmetric connectivity range assignment problem*, and in Chapter 9 in the ILP-model of the *two-scenario minimum spanning tree problem*. The *initialization* function adds the cardinality constraint to the model. The *feasibility test* checks whether the subgraph induced by an integer solution of an LP is connected. Together with the cardinality constraint, which is satisfied by the integer solution, this ensures that the induced subgraph is a spanning tree. The *separation* function checks whether a fractional LP-solution violates any subtour elimination constraints, as described in Section 6.1. If violated subtour elimination constraints are found they are added to the ILP of the current subproblem.

Nonlinear Optimization in SCIL

The SCIL optimization library was initially designed for integer linear problems, in particular for linear combinatorial problems. For this thesis it was expanded by the polyhedral methods for nonlinear combinatorial problems discussed in previous chapters.

Submodular Combinatorial Optimization

The model for combinatorial optimization problems with submodular objective functions solved in SCIL has the form

$$\begin{array}{lrcl} \min & \displaystyle\sum_{i=1}^{k} \alpha_i y_i & & \\ \text{s.t.} & y_i & \geq & f_i(x) \text{ for all } i \in \{1, \ldots, k\} \\ & x & \in & X \\ & y & \in & \mathbb{R}^k, \end{array}$$

with submodular functions f_i and positive scalars α_i.

The description of the set X, which models the underlying combinatorial structure of the problem, can be given as any combination of linear inequalities and

symbolic constraints. Each component f_i of the submodular objective function is specified as a symbolic constraint. These problem-specific classes are derived from an abstract base class `submodular`. The purpose of the derived classes is to provide a function value oracle which returns the value $f_i(x)$ for a given binary vector x and is used by the base class `submodular` to calculate the coefficients of cutting planes in the separation algorithm 2 described in Section 3.3 and in the feasibility test, which checks if $y_i \geq f_i(\bar{x})$ for a given binary solution $\bar{x}$.

Binary Quadratic Optimization

To integrate the techniques for binary quadratic optimization discussed in Chapter 2 into SCIL, the interface for building the mathematical model was extended by two functions. The first, `add_polynomial` adds a quadratic polynomial in binary variables of the form $\sum_{i,j} a_{ij} x_i x_j$ to the objective function. This function returns the corresponding linearization variable y_{ij} for further use. The second function, `add_pol_constraint`, adds a binary quadratic inequality of the form $\sum_{i,j} a_{ij} x_i x_j \leq b$ to the set of constraints.

After the model is completed, all quadratic monomials are internally replaced by linearization variables and the separation graph for MaxCut-inequalities is constructed as discussed in Section 2.2. The *separation* itself is handled by the symbolic constraint `CUT`, which implements the exact and heuristic separation routines for odd-cycle inequalities described in Section 2.2, as well as a pool separation for triangle inequalities. The *feasibility test* checks whether $y_{ij} = x_i x_j$ holds for all quadratic monomials in a given LP-solution.

The following parameters control the treatment of quadratic terms during optimization. They can be set at runtime, before the start of the optimization process:

standard linearization If enabled, the standard linearization inequalities for all quadratic monomials are added to the model.

triangle inequalities If enabled, triangle inequalities for all cycles of length three in the separation graph are generated.

triangle pool separation Only in conjunction with **triangle inequalities**. If enabled, triangle inequalities are added to the model dynamically in the separation phase. Otherwise, all triangle inequalities are included in the initial subproblem.

Quadratic Reformulation

The quadratic reformulation techniques for linear constraints discussed in Section 2.3 are implemented in SCIL in the symbolic constraint `quadref`. The user can choose to force quadratic reformulation of individual constraints, otherwise

a linear inequality is only reformulated if all necessary quadratic monomials are already present in the model. All inequalities obtained from the reformulation SQK2 are added to the initial subproblem, inequalities from SQK3 are stored in a cut pool and are added dynamically in the separation phase.

The following parameters control the quadratic reformulation of linear constraints. They can be set at runtime, before the start of the optimization process:

quadratic reformulation type

`NONE` Quadratic reformulation is disabled, except for inequalities explicitly flagged for reformulation.

`SQK2` SQK2 is applied to all linear inequalities, unless this would require quadratic monomials not present in the initial model.

`SQK3` SQK3 is applied to all linear inequalities, unless this would require quadratic monomials not present in the initial model.

`SQK2+SQK3` Both SQK2 and SQK3 are applied.

`PM` Phantom monomial reformulation is applied to all assignment constraints added to the model. All other quadratic reformulations are disabled.

qr on separate If enabled, the quadratic reformulation specified in **quadratic reformulation type** is applied to linear inequalities found in the separation phase, unless this would require the introduction of additional linearization variables. This parameter has no effect if **quadratic reformulation type** is set to `NONE` or `PM`.

Phantom Monomials

When phantom monomial reformulation is enabled in SCIL, each linear constraint added to the model is checked whether it is an assignment constraint. The corresponding quadratic monomials are internally flagged as phantom monomials. For these monomials no linearization variables or standard linearization constraints are added to the model, but the corresponding edges are added to the separation graph for MaxCut-inequalities. When an LP-solution is mapped to the separation graph or a cutting plane is mapped to the original variable space, the modified transformation given in Section 2.3.3 is used for the phantom monomials.

Summary and Outlook

We proposed two approaches for the exact solution of combinatorial optimization problems with nonlinear objective functions. Both follow the same basic idea: Lower bounds for the nonlinear combinatorial problem are computed by decomposing it into an unconstrained binary nonlinear problem and the linear variant of the underlying combinatorial problem. In many cases these subproblems are well-studied and easier to solve than the original problem.

The first approach combines polyhedral descriptions of the convex hulls of feasible points of the two subproblems. This in general does not yield a complete polyhedral characterization of the feasible solutions of the original problem, even if complete descriptions of the subproblems are known, as for example for the symmetric connectivity range assignment problem studied in Chapter 7. Nevertheless our experiments show that this approach often yields strong lower bounds in a branch and bound-framework.

The second approach separates the nonlinear objective function from the combinatorial constraints by Lagrangean decomposition. Lower bounds are computed by solving the Lagrangean dual of the decomposition with a subgradient algorithm. This approach has the advantage that for a large class of applications both subproblems can be efficiently solved with purely combinatorial algorithms.

We used these approaches to devise branch and bound-algorithms for combinatorial problems with quadratic, submodular and min-max objective functions. Our experimental studies of the quadratic minimum spanning tree problem and the quadratic matching problem in Chapter 6 show that, especially for larger instances, using cutting planes from MaxCut as discussed in Chapter 2 significantly reduces the number of subproblems in the branch and bound-tree. Still, this does not necessarily lead to lower running times. When the MaxCut-separation is combined with quadratic reformulation, the situation is different. The number of subproblems as well as the running times are significantly lower and we conclude that the combination of MaxCut separation and quadratic reformulation is an effective tool for solving quadratic combinatorial problems. MaxCut separation is also used in Chapter 5, where we propose a quadratic model for the tanglegram layout problem. To our knowledge, this is the first model that is able to handle tanglegrams of arbitrary degree and we are able to solve large random

and realistic instances to optimality.

In the range assignment and portfolio optimization problems studied in Chapters 7 and 8, combinatorial constraints are combined with a submodular objective function. In the case of range assignment problems the objective function models the multicast advantage that occurs when nodes are connected by wireless communication links to form a given network topology. In portfolio optimization, submodularity arises when the uncertainty in the expected return value of a possible investment is characterized by its variance.

The branch and cut-algorithm is applicable to all three types of range assignment problems examined in Chapter 8, whereas the Lagrangean decomposition-based algorithm is not applicable to multicast problems, since no practically efficient algorithm for the combinatorial subproblem is known. It turns out that the performance of both algorithms in comparison to solving the standard IP model varies with the underlying network topology and the size of the instance. In the symmetric connectivity case no algorithm clearly dominates the others. In the multicast case the new MIP model gives better results than the standard IP model. Solution times are lower and a larger number of the large instances can be solved. When applied to broadcast instances, computing Lagrangean duals instead of solving LP-relaxations shows its potential. Although the algorithm is slowed down by a large number of calls to the function oracle, it is able to solve more of the largest instances than the other two. For portfolio optimization the situation is much clearer. Here Lagrangean decomposition obviously is the best choice. The standard approach already fails for small instances. While both new algorithms are able to solve large instances quickly, the decomposition approach clearly outperforms the branch and cut-algorithm. Additionally, it is nearly insensitive to the scaling of the nonlinear part of the objective function.

To evaluate the bounding techniques for two-scenario problems proposed in Chapter 4, we performed experimental studies of the unconstrained two-scenario problem and the two-scenario minimum spanning tree problem. Running times in the unconstrained case were similar to the standard LP-based branch and bound-algorithm, but in the presence of combinatorial constraints, the low computational complexity of the fractional knapsack problems and the high quality of the bounds they provide pay off. We are able to compute two-scenario spanning trees on complete graphs with up to 200 nodes quickly, largely independent of the characteristics of the objecticve function. Running times for the standard approach, on the other hand, are generally much higher and increase significantly with the deviation between the scenarios.

As we have seen in Chapter 8, there is a close connection between the mean-variance objective function in the risk-averse capital budgeting problem and robust optimization. Indeed, mean-risk problems can be seen as a special case of robust combinatorial optimization with ellipsoidal uncertainty. While it is known

that it is easy to minimize the function $f(x) = c^\top x + \sqrt{x^\top Q x}$ with $x \in \{0,1\}^n$ for diagonal matrices but NP-hard for general positive-definite Q, it is unclear at what point the problem becomes hard. Special cases of Q that have applications in practice and might be computationally tractable include doubly non-negative matrices and block diagonal matrices with constant block size. For such block diagonal matrices it is promising to search for a combinatorial algorithm similar to Algorithm 4 in Chapter 8, since the number of quadratic terms is linear in the instance size.

Doubly non-negative matrices Q occur when the uncertain costs are correlated, but all correlations are non-negative. One example is the design of robust traffic networks, where a delay on one arc of the graph often leads to delays on other arcs. Also in this case the complexity of minimizing f is unknown. An interesting approach to this problem is to approximate the ellipsoid describing the uncertainty set by an axis-parallel ellipsoid. This might yield good lower bounds, which could be computed efficiently with the combinatorial algorithm we proposed in Chapter 8. It would be interesting to study the feasibility of this approach and investigate the quality of the lower bounds both theoretically and experimentally.

While a better understanding of the structure of the unrestricted nonlinear objective function will lead to algorithms for a wide range of applications, it is also interesting to study specific combinatorial problems under uncertainty. For example, little is known about the complexity of classical combinatorial problems with a mean-variance objective function. On the one hand, Nikolova [101] showed that the mean-risk minimum spanning tree problem can be solved in polynomial time and to our knowledge there is no application which is easy with a linear objective function but proven to be NP-hard in the mean-variance case. On the other hand, many combinatorial problems are known to be hard in the general ellipsoidal uncertainty case [118].

References

[1] Alok Aggarwal, Sanjeev Khanna, Rajeev Motwani, and Baruch Schieber. The angular-metric traveling salesman problem. *In Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, 29:221–229, 1997.

[2] Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. Approximating min-max (regret) versions of some polynomial problems. In Danny Z. Chen and Der-Tsai Lee, editors, *Computing and Combinatorics*, volume 4112 of *Lecture Notes in Computer Science*, pages 428–438. Springer Berlin Heidelberg, 2006.

[3] Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. Min-max and min-max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197(2):427–438, September 2009.

[4] Ernst Althaus, Alexander Bockmayr, Matthias Elf, Michael Jünger, Thomas Kasper, and Kurt Mehlhorn. SCIL – symbolic constraints in integer linear programming. In Rolf H. Möhring and Rajeev Raman, editors, *Algorithms - ESA 2002 Rome*, volume 2461 of *Lecture Notes in Computer Science*, pages 75–87. Springer, 2002.

[5] Ernst Althaus, Gruia Calinescu, Ion I. Mandoiu, Sushil K. Prasad, Nickolay Tchervenski, and Alexander Zelikovsky. Power efficient range assignment in ad-hoc wireless networks. In *WCNC '03*, pages 1889–1894, 2003.

[6] Ernst Althaus, Gruia Calinescu, Ion I. Mandoiu, Sushil K. Prasad, Nickolay Tchervenski, and Alexander Zelikovsky. Power efficient range assignment for symmetric connectivity in static ad hoc wireless networks. *Wireless Networks*, 12(3):287–299, 2006.

[7] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA, 2007.

[8] Amitai Armon and Uri Zwick. Multicriteria global minimum cuts. *Algorithmica*, 46(1):15–26, September 2006.

[9] Arjang Assad and Weixuan Xu. The quadratic minimum spanning tree problem. *Naval Research Logistics (NRL)*, 39(3):399–417, 1992.

[10] Alper Atamtürk and Vishnu Narayanan. Polymatroids and mean-risk minimization in discrete optimization. *Operations Research Letters*, 36(5):618–622, 2008.

[11] Egon Balas and Eitan Zemel. An algorithm for large zero-one knapsack problems. *Operations Research*, 28(5):1130–1154, 1980.

[12] Francisco Barahona and Ali Ridha Mahjoub. On the cut polytope. *Mathematical Programming*, 36:157–173, 1986.

[13] Francisco Barahona, Michael Jünger, and Gerd Reinelt. Experiments in quadratic 0-1 programming. *Mathematical Programming*, 44(1–3):127–137, 1989.

[14] Paola Bertolazzi, Giuseppe Di Battista, Carlo Mannino, and Roberto Tamassia. Optimal upward planarity testing of Single-Source digraphs. *SIAM Journal on Computing*, 27(1):132–169, February 1998.

[15] Frederick Bock. An algorithm to construct a minimum directed spanning tree in a directed network. In *Developments in operations research*, pages 29–44, 1971.

[16] Sebastian Böcker, Falk Hüffner, Anke Truss, and Magnus Wahlström. A faster fixed-parameter approach to drawing binary tanglegrams. In Jianer Chen and Fedor V. Fomin, editors, *Parameterized and Exact Computation*, volume 5917 of *Lecture Notes in Computer Science*, pages 38–49. Springer Berlin Heidelberg, 2009.

[17] Otakar Boruvka. O Jistém Problému Minimálním (About a Certain Minimal Problem) (in Czech, German summary). *Práce Mor. Prírodoved. Spol. v Brne III*, 3, 1926.

[18] Christoph Buchheim, Angelika Wiegele, and Lanbo Zheng. Exact algorithms for the quadratic linear ordering problem. *INFORMS Journal on Computing*, 22(1):168–177, December 2010.

[19] Rainer E. Burkard. Quadratic assignment problems. In Panos M. Pardalos, Ding-Zhu Du, and Ronald L. Graham, editors, *Handbook of Combinatorial Optimization*, pages 2741–2814. Springer New York, January 2013.

[20] Markus Chimani, Philipp Hungerländer, Michael Jünger, and Petra Mutzel. An sdp approach to multi-level crossing minimization. *Journal of Experimental Algorithmics*, 17:3.3:3.1–3.3:3.26, September 2012.

[21] Y. Chu and T. Liu. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400, 1965.

[22] Roberto Cordone and Gianluca Passeri. Heuristic and exact approaches to the quadratic minimum spanning tree problem. In *CTW*, pages 52–55. University of Milan, 2008.

[23] Roberto Cordone and Gianluca Passeri. Solving the quadratic minimum spanning tree problem. *Applied Mathematics and Computation*, 218(23): 11597–11612, 2012.

[24] Gérard Cornuéjols and Reha Tütüncü. *Optimization methods in finance*. Mathematics, finance, and risk. Cambridge University Press, Cambridge, U.K., New York, 2006.

[25] George B. Dantzig. Discrete-Variable Extremum Problems. *Operations Research*, 5(2), 1957.

[26] George B. Dantzig, Alex Orden, and Philip Wolfe. The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5(2):183–195, 1955.

[27] Arindam K. Das, Robert J. Marks, Mohamed El-Sharkawi, Payman Arabshahi, and Andrew Gray. Minimum power broadcast trees for wireless networks: Integer programming formulations. In *INFOCOM 2003*, volume 2, pages 1001–1010, 2003.

[28] Caterina De Simone. The cut polytope and the boolean quadric polytope. *Discrete Mathematics*, 79:71–75, 1989.

[29] Caterina De Simone and Giovanni Rinaldi. A cutting plane algorithm for the max-cut problem. *Optimization Methods and Software*, 3(1–3):195–214, 1994.

[30] Michel Deza. On the hamming geometry of unitary cubes. *Doklady Akademii Nauk SSR*, 134:1037–1040, 1960. English translation in: Soviet Physics Doklady 5 (1961) 940–943.

[31] Michel Deza and Monique Laurent. *Geometry of Cuts and Metrics*. Springer, May 1997.

[32] Peter Eades and Nicholas C. Wormald. Edge crossings in drawing bipartite graphs. *Algorithmica*, 11:379–403, 1994.

[33] Jack Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[34] Jack Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards B*, 69:125–130, 1965.

[35] Jack Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240, 1967.

[36] Jack Edmonds. Submodular functions, matroids, and certain polyhedra. In Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi, editors, *Combinatorial Optimization – Eureka, You Shrink!*, volume 2570 of *Lecture Notes in Computer Science*, pages 11–26. Springer Berlin Heidelberg, 2003.

[37] Laurent El Ghaoui, Maksim Oks, and Francois Oustry. Worst-case value-at-risk and robust portfolio optimization: A conic programming approach. *Operations Research*, 51(4):543–556, 2003.

[38] Matthias Elf, Carsten Gutwenger, Michael Jünger, and Giovanni Rinaldi. Branch-and-cut algorithms for combinatorial optimization and their implementation in abacus. In Michael Jünger and Denis Naddef, editors, *Computational Combinatorial Optimization*, volume 2241 of *Lecture Notes in Computer Science*, pages 157–222. Springer, 2001.

[39] Henning Fernau, Michael Kaufmann, and Mathias Poths. Comparing trees via crossing minimization. In Ramaswamy Ramanujam and Sandeep Sen, editors, *Foundations of Software Technology and Theoretical Computer Science FSTTCS 2005*, volume 3821 of *Lecture Notes in Computer Science*, pages 457–469. Springer, 2005.

[40] Henning Fernau, Michael Kaufmann, and Mathias Poths. Comparing trees via crossing minimization. *Journal of Computer and System Sciences*, 76 (7):593–608, November 2010.

[41] Ilse Fischer, Gerald Gruber, Franz Rendl, and Renata Sotirov. Computational experience with a bundle approach for semidefinite cutting plane relaxations of max-cut and equipartition. *Mathematical Programming*, 105 (2–3):451–469, 2006.

[42] Matteo Fischetti. Facets of two Steiner arborescence polyhedra. *Mathematical Programming, Series A*, 51(3):401–419, 1991.

[43] Robert Fortet. L'algèbre de boole et ses applications en recherche opérationelle. *Cahiers du centre d'études de recherche opérationelle*, 1(4): 5–36, 1959.

[44] Bernhard Fuchs. On the hardness of range assignment problems. *Networks*, 52(4):183–195, 2008.

[45] Satoru Fujishige. *Submodular functions and optimization. 2nd ed.* Annals of Discrete Mathematics 58. Amsterdam: Elsevier. xiv, 395 p., 2005.

[46] Fabio Furini and Emiliano Traversi. Hybrid sdp bounding procedure. In Vincenzo Bonifaci, Camil Demetrescu, and Alberto Marchetti-Spaccamela, editors, *Experimental Algorithms*, volume 7933 of *Lecture Notes in Computer Science*, pages 248–259. Springer Berlin Heidelberg, 2013.

[47] Laura Galli, Konstantinos Kaparis, and Adam N. Letchford. Gap inequalities for the max-cut problem: A cutting-plane algorithm. In *Proceedings of the Second International Conference on Combinatorial Optimization*, ISCO'12, pages 178–188. Springer-Verlag, 2012.

[48] Laura Galli, Konstantinos Kaparis, and Adam N. Letchford. Complexity results for the gap inequalities for the max-cut problem. *Operations Research Letters*, 40(3):149–152, 2012.

[49] Michael Garey and David Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co Ltd, January 1979.

[50] Arthur M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Study*, 2:82–114, 1974.

[51] Fred Glover and Eugene Woolsey. Technical Note – Converting the 0–1 polynomial programming problem to a 0–1 linear program. *Operations Research*, 22(1):180–182, February 1974.

[52] Ralph E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275–278, 09 1958.

[53] Martin Grötschel, Michael Jünger, and Gerd Reinelt. Facets of the linear ordering polytope. *Mathematical Programming*, 33:43–60, 1985.

[54] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer-Verlag, 1988.

[55] Monique Guignard and Siwhan Kim. Lagrangean decomposition: A model yielding stronger lagrangean bounds. *Mathematical Programming*, 39(2): 215–228, June 1987.

[56] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2013. URL http://www.gurobi.com.

[57] Mark S. Hafner, Philip D. Sudman, Francis X. Villablanca, Theresa A. Spradling, James W. Demastes, and Steven A. Nadler. Disparate rates of molecular evolution in cospeciating hosts and parasites. *Science*, 265: 1087–1090, 1994.

[58] Christoph Helmberg. The ConicBundle Library for Convex Optimization. www-user.tu-chemnitz.de/~helmberg/ConicBundle, 2011.

[59] Christoph Helmberg and Franz Rendl. Solving quadratic (0, 1)-problems by semidefinite programs and cutting planes. *Mathematical Programming*, 82(3):291–315, 1998.

[60] Christoph Helmberg and Franz Rendl. A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 10(3):673–696, 2000.

[61] Christoph Helmberg, Franz Rendl, Robert J. Vanderbei, and Henry Wolkowicz. An interior-point method for semidefinite programming. *SIAM Journal on Optimization*, 6:342–361, 1996.

[62] Christoph Helmberg, Franz Rendl, and Robert Weismantel. A semidefinite programming approach to the quadratic knapsack problem. *Journal of Combinatorial Optimization*, 4(2):197–215, 2000.

[63] Danny Holten. personal communication, 2009.

[64] IBM. IBM ILOG CPLEX 12.5.1, 2013. www.ibm.com/software/commerce/optimization/cplex-optimizer/.

[65] ILOG, Inc. ILOG CPLEX 11.2, 2007. www.ilog.com/products/cplex.

[66] ILOG, Inc. ILOG CPLEX 12.1, 2009. www.ilog.com/products/cplex/.

[67] Anna Ilyina. personal communication, 2013.

[68] Satoru Iwata and Kiyohito Nagano. Submodular function minimization under covering constraints. In *FOCS*, pages 671–680. IEEE Computer Society, 2009.

[69] Satoru Iwata and James B. Orlin. A simple combinatorial algorithm for submodular function minimization. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 1230–1237, 2009.

[70] Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, STOC '00, pages 97–106, New York, NY, USA, 2000. ACM.

[71] Michael Jünger and Petra Mutzel. 2-layer straightline crossing minimization: performance of exact and heuristic algorithms. *Journal of Graph Algorithms and Applications*, 1:1–25, 1997.

[72] Michael Jünger and Stefan Thienel. The ABACUS system for branch-and-cut-and-price algorithms in integer programming and combinatorial optimization. *Software: Practice and Experience*, 30(11):1325–1352, 2000.

[73] Michael Jünger, Eva K. Lee, Petra Mutzel, and Thomas Odenthal. A polyhedral approach to the multi-layer crossing minimization problem. In Giuseppe DiBattista, editor, *Graph Drawing*, volume 1353 of *Lecture Notes in Computer Science*, pages 13–24. Springer Berlin Heidelberg, 1997.

[74] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer, 2004.

[75] Kiavash Kianfar. Branch-and-bound algorithms. In James J. Cochran, Louis A. Cox, Pinar Keskinocak, Jeffrey P. Kharoufeh, and J. Cole Smith, editors, *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc., 2010.

[76] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 4th edition, 2007.

[77] Panos Kouvelis and Gang Yu. *Robust Discrete Optimization and Its Applications (Nonconvex Optimization and Its Applications (closed))*. Springer, 1st edition, November 1996.

[78] Joseph B. Kruskal Jr. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[79] Ailsa H. Land and Alison G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[80] Monique Laurent and Svatopluk Poljak. Gap inequalities for the cut polytope. *European Journal of Combinatorics*, 17(2–3):233–254, 1996.

[81] Eugene L. Lawler. The quadratic assignment problem. *Management Science*, 9:586–599, 1963.

[82] Heesang Lee, George L. Nemhauser, and Yinhua Wang. Maximizing a submodular function by integer programming: Polyhedral results for the quadratic case. *European Journal of Operational Research*, 94(1):154–166, October 1996.

[83] Valeria Leggieri, Paolo Nobili, and Chefi Triki. Minimum power multicasting problem in wireless networks. *Mathematical Methods of Operations Research*, 68:295–311, 2008.

[84] Adam N. Letchford, Gerhard Reinelt, and Dirk O. Theis. A faster exact separation algorithm for blossom inequalities. In George L. Nemhauser and Daniel Bienstock, editors, *IPCO*, volume 3064 of *Lecture Notes in Computer Science*, pages 196–205. Springer, 2004.

[85] László Lovász. Submodular functions and convexity. In *Mathematical programming: the state of the art (Bonn, 1982)*, pages 235–257. Springer, Berlin, 1983.

[86] László Lovász and Michael D. Plummer. *Matching Theory (North-Holland Mathematics Studies 121)*. Elsevier Science Ltd, 1st edition, June 1986.

[87] Douglas G. Macharet, Armando A. Neto, Vilar F. da Camara Neto, and Mario F. M. Campos. Nonholonomic path planning optimization for dubins' vehicles. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4208–4213, 2011.

[88] Harry M. Markowitz. *Mean-variance analysis in portfolio choice and capital markets*. Blackwell, Oxford, 1987.

[89] Martello, Silvano, and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., New York, NY, USA, 1990.

[90] Alexander Martin. General mixed integer programming: Computational issues for branch-and-cut algorithms. In Michael Jünger and Denis Naddef, editors, *Computational Combinatorial Optimization*, volume 2241 of *Lecture Notes in Computer Science*, pages 1–25. Springer Berlin Heidelberg, 2001.

[91] Rafael Martinelli and Claudio Contardo. The quadratic capacitated vehicle routing problem. Technical Report G-2013-74, Les Cahiers du GERAD, HEC Montréal, Montréal, Quebec, Canada, October 2013.

[92] Patrick M. Miller. Exakte und heuristische Verfahren zur Lösung von Range-Assignment-Problemen. Master's thesis, Universität zu Köln, 2010.

[93] Manki Min, Oleg Prokopyev, and Panos Pardalos. Optimal solutions to minimum total energy broadcasting problem in wireless ad hoc networks. *Journal of Combinatorial Optimization*, 11:59–69(11), 2006.

[94] John E. Mitchell. Branch and cut. In James J. Cochran, Louis A. Cox, Pinar Keskinocak, Jeffrey P. Kharoufeh, and J. Cole Smith, editors, *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc., 2010.

[95] Hans Mittelmann. Benchmarks for optimization software, 2013. URL `http://plato.asu.edu/ftp/miqp.html`. (29 July 2013).

[96] Roberto Montemanni and Luca M. Gambardella. Minimum power symmetric connectivity problem in wireless networks: A new approach. In *MWCN*, pages 497–508, 2004.

[97] Roberto Montemanni and Luca M. Gambardella. Exact algorithms for the minimum power symmetric connectivity problem in wireless networks. *Computers & Operations Research*, 32:2891–2904, 2005.

[98] Roberto Montemanni, Luca M. Gambardella, and Arindam Das. Mathematical models and exact algorithms for the min-power symmetric connectivity problem: an overview. In Jie Wu, editor, *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks*, pages 133–146. CRC Press, 2006.

[99] Katta G. Murty and Clovis Perin. A 1-matching blossom-type algorithm for edge covering problems. *Networks*, 12(4):379–391, 1982.

[100] George L. Nemhauser and Laurence A. Wolsey. *Integer and combinatorial optimization*. Wiley-Interscience, New York, NY, USA, 1988.

[101] Evdokia Nikolova. Approximation algorithms for reliable stochastic combinatorial optimization. In *Proceedings of the 13th International Workshop on Approximation and the 14th International Workshop on Randomization and Computation*, APPROX/RANDOM'10, pages 338–351. Springer Berlin Heidelberg, 2010.

[102] Paolo Nobili, Simona Oprea, and Chefi Triki. Preprocessing techniques for the multicast problem in wireless networks. In *MTISD 2008*, pages 131–134, 2008.

[103] Martin Nöllenburg, Markus Völker, Alexander Wolff, and Danny Holten. Drawing binary tanglegrams: An experimental evaluation. In *Proc. of the Workshop on Algorithm Engineering and Experiments, ALENEX 2009*, pages 106–119. SIAM, 2009.

[104] Muhittin Oral and Ossama Kettani. Reformulating nonlinear combinatorial optimization problems for higher computational efficiency. *European Journal of Operational Research*, 58(2):236–249, 1992.

[105] James B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118(2):237–251, November 2007.

[106] Manfred Padberg. The boolean quadric polytope: Some characteristics, facets and relatives. *Mathematical Programming*, 45(1–3):139–172, 1989.

[107] Robert C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[108] Gerd Reinelt. *The Linear Ordering Problem: Algorithms and Applications*. Heldermann Verlag, 1985.

[109] Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele. A branch and bound algorithm for max-cut based on combining semidefinite and polyhedral relaxations. In Mattheo Fischetti and David P. Williamson, editors, *IPCO 2007*, volume 4513 of *Lecture Notes in Computer Science*, pages 295–309. Springer, 2007.

[110] Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121(2):307–335, 2010.

[111] Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23(3):555–565, July 1976.

[112] Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80:346–355, 2000.

[113] Alexander Schrijver. *Combinatorial Optimization – Polyhedra and Efficiency*. Springer, 2003.

[114] SCIL. SCIL – Symbolic Constraints in Integer Linear programming. `scil-opt.net`, 2011.

[115] Zuo-Jun Max Shen, Collette Coullard, and Mark S. Daskin. A joint location-inventory model. *Transportation Science*, 37(1):40–55, January 2003.

[116] Hanif D. Sherali and Warren P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990.

[117] Jeremy G. Siek, Lie-Quan Lee, and Andrew Lumsdaine. *The Boost Graph Library: User Guide and Reference Manual (C++ In-Depth Series)*. Addison-Wesley Professional, December 2001.

[118] Melvyn Sim. *Robust Optimization*. PhD thesis, Operations Research Center, MIT, 2004.

[119] Erik Sjölund and Ali Tofigh. Edmonds' algorithm. `edmonds-alg.sourceforge.net`, 2010.

[120] Donald Topkis. Minimizing a submodular function on a lattice. *Operations Research*, 26(2):305–321, 1978.

[121] Balaji Venkatachalam, Jim Apple, Katherine St. John, and Dan Gusfield. Untangling tanglegrams: Comparing trees by their drawings. In *Proceedings of the 5th International Symposium on Bioinformatics Research and Applications*, ISBRA '09, pages 88–99. Springer, 2009.

[122] Jeffrey E. Wieselthier, Gam D. Nguyen, and Anthony Ephremides. Algorithms for energy-efficient multicasting in static ad hoc wireless networks. *Mobile Networks and Applications*, 6(3):251–263, 2001.

[123] Laurence A. Wolsey. *Integer Programming*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 1998.

[124] Gang Yu and Jian Yang. On the robust shortest path problem. *Computers and Operations Research*, 25:457–468, 1998.