Angelika Wiegele

Nonlinear Optimization Techniques Applied to Combinatorial Optimization Problems

DISSERTATION

zur Erlangung des akademischen Grades
Doktorin der Technischen Wissenschaften

Alpen-Adria-Universität Klagenfurt

Fakultät für Wirtschaftswissenschaften und Informatik

1. Begutachter: Univ.-Prof. Dipl.-Ing. Dr. Franz Rendl
   Institut für Mathematik
2. Begutachterin: Ao. Univ.-Prof. Dipl.-Ing. Dr. Christine Nowak
   Institut für Mathematik

Oktober 2006

Ehrenwörtliche Erklärung

Ich erkläre ehrenwörtlich, dass ich die vorliegende Schrift verfasst und die mit ihr unmittelbar verbundenen Arbeiten selbst durchgeführt habe. Die in der Schrift verwendete Literatur sowie das Ausmaß der mir im gesamten Arbeitsvorgang gewährten Unterstützung sind ausnahmslos angegeben. Die Schrift ist noch keiner anderen Prüfungsbehörde vorgelegt worden.

Klagenfurt, Oktober 2006

Abstract

Combinatorial Optimization and Semidefinite Programming are two research topics that have attracted the attention of many mathematicians and computer scientists during the past two decades. Remarkable results have been achieved in both fields. This thesis is a further component in exploring the field of Semidefinite Programming and investigating Combinatorial Optimization problems.

Due to the various areas of application, one research topic of high interest is the development of algorithms for solving Semidefinite Programs. Although reliable methods are already available and widely used these algorithms are often inapplicable for large-scale programs, due to the huge memory requirements or the vast computational effort. The present work proposes methods (and implementations) that are capable of solving Semidefinite Programs of high dimensions and/or a large number of constraints. These methods are: the Bundle Method applied to solve Semidefinite Programs, the Spectral Bundle Method with second order information, and the Boundary Point Method.

Exploiting the concept of Bundle Methods allows solving problems, even if the number of constraints is rather large. By the use of Lagrange multipliers, the constraints (or some of them) are lifted into the objective function and the dual problem is then solved following the concept of Bundle Methods.

In the Spectral Bundle Method the largest Eigenvalue $\lambda_{\max}$ of a matrix is minimized. Since the second-order behavior of the $\lambda_{\max}$ function is well studied, it can be incorporated in this method. Making partial use of this second-order information improves the efficiency of the Spectral Bundle Method while keeping it computationally practical.

Another new algorithm for solving Semidefinite Programs is the Boundary Point Method. This is an augmented Lagrangian algorithm applied to solve Semidefinite Programs. For various problem classes this method is by far superior to other available algorithms.

Regarding applications, the main focus of this study is on the Max-Cut problem, one of the most challenging Combinatorial Optimization problems. The applicability of this problem is even broader than obvious at first sight, since any unconstrained quadratic (0-1) problem can be transformed to a Max-Cut problem. Apart from recalling the properties of the problem and giving a survey of relaxations and known solution methods, new relaxations based on Semidefinite

Programming are introduced. Finally, "Biq Mac" was developed, a solver for binary quadratic and Max-Cut problems. Biq Mac is an implementation of an exact solution method using a Branch & Bound algorithm with a bounding routine based on Semidefinite Programming. Detailed information on this algorithm, as well as a collection of test problems together with numerical results, can be found in the present thesis. Various test problems that have been considered in the literature for years, were solved by Biq Mac for the first time. This affirms the success of using Semidefinite Programming for Combinatorial Optimization problems.

Acknowledgements

I am grateful to a number of people who have supported me in the development of this work and it is my pleasure now to highlight them here.

I want to thank my supervisor Franz Rendl for introducing me into the field of Semidefinite Programming, for his enthusiasm about discussing mathematical issues and for the large amount of time he devoted to my concerns. His ideas and advice led me into active research and substantiated my thesis.

The Mathematics Department at the Alpen-Adria-Universität Klagenfurt provided excellent working conditions. I would like to thank my colleagues at the department and especially Christine Nowak. She served as a member of my Ph.D. committee and she was willing to encourage me at any time.

I also want to thank Giovanni Rinaldi for inviting me to IASI-CNR in Rome. Due to him and the people in his group my research stays in Rome were very fruitful and enjoyable. Furthermore, I gratefully acknowledge financial support from the EU project Algorithmic Discrete Optimization (ADONET), MRTN-CT-2003-504438. Participation at various conferences and workshops, as well as the research stay at IASI-CNR in Rome were financed by this research training network.

Having people around to have fun with, to discuss whatever has to be discussed, to fill my spare time with joyful events, but also to understand that sometimes 'spare time' is negligibly small, is very important for me. I am grateful for being surrounded by such friends and for the many occasions that proved how precious they are to me.

Above all, my thanks go to my family. Although I was away quite often, I always had a hearty welcome when returning home. I want to dedicate this work to them.

Contents

Notation

This is a short description of the symbols used throughout this thesis. Also the names of the various (semidefinite) programs are given, including the numbers of the sections where they appear for the first time.

$\mathbb{R}^n$	*space of real n-dimensional vectors*
$\mathcal{S}_n$	*space of $n \times n$ symmetric matrices*
$\mathcal{S}_n^+$	*space of $n \times n$ positive semidefinite matrices*
$\mathcal{S}_n^{++}$	*space of $n \times n$ positive definite matrices*
$\mathcal{S}_n^-$	*space of $n \times n$ negative semidefinite matrices*
$\mathcal{S}_n^{--}$	*space of $n \times n$ negative definite matrices*
$\succeq$	*Löwner partial order*
min	*minimum, minimize*
max	*maximum, maximize*
inf	*infimum*
sup	*supremum*
∇	*nabla operator*
$\frac{\partial}{\partial x_i} f(x_1, \ldots, x_n)$	*partial derivative*
$\mathcal{A}(\cdot)$	*linear operator*
$\mathcal{A}^T(\cdot)$	*adjoint of the linear operator $\mathcal{A}(\cdot)$*
$\mathrm{tr}A$	*trace of matrix A*
$\langle A, B \rangle$	$\langle A, B \rangle := \mathrm{tr}(A^T B)$
I	*identity matrix of appropriate dimension*
I_n	*identity matrix of dimension n*
J	*matrix of all ones*
e	*vector of all ones of appropriate dimension*
e_n	*vector of all ones of dimension n*
$\lambda_{\min}(A)$	*minimum eigenvalue of the symmetric matrix A*
$\lambda_{\max}(A)$	*maximum eigenvalue of the symmetric matrix A*
$\mathrm{diag}(A)$	*vector formed by the main diagonal of matrix A*

$\mathrm{Diag}(v)$	*diagonal matrix with main diagonal* v
$G = (V(G), E(G))$	*graph* G *with vertex set* $V(G)$ *and edge set* $E(G)$
$i \sim j$	*vertices* i *and* j *are in the same partition block*
$i \not\sim j$	*vertices* i *and* j *are in opposite partition blocks*
$\alpha(G)$	*stability number of graph* G
$\omega(G)$	*clique number of graph* G
$\chi(G)$	*chromatic number of graph* G
$\vartheta(G)$	*ϑ-number of graph* G
(PSDP)	*primal Semidefinite Program in standard form, 1.1*
(DSDP)	*dual Semidefinite Program in standard form, 1.1*
(EVP)	*Eigenvalue Optimization Problem, 1.1*
(MC)	*Max-Cut problem, 2.1*
(QP)	*unconstrained quadratic (0-1) problem, 3.2*
(THETA)	*SDP for computing the ϑ-number, 2.2*
(DTHETA)	*and its dual, 2.2*
(GP)	*graph partitioning problem, 2.3*
(MCSDP)	*basic SDP relaxation of* **(MC)**, *3.3.2*
(MCDSDP)	*and its dual, 3.3.2*
(MCEIG)	*basic SDP relaxation of* **(MC)** *as eigenvalue optimization problem, 3.3.2*
(SDPMET)	*SDP relaxation of* **(MC)** *strengthened by the triangle inequlities, 4.2*
(SDP3)	*lifting of Anjos and Wolkowicz, 4.3.1*
($\mathbf{SDP3_P}$)	*projected lifting of Anjos and Wolkowicz, 4.3.1*
(MCSPARSE)	*SDP relaxation of* **(MC)** *designed for sparse graphs, 4.4*

Introduction

This thesis is basically concerned with two topics:

- The Max-Cut problem: introducing new relaxations based on Semidefinite Programming to obtain tight upper bounds and developing an exact solution method.
- Methods for solving large-scale Semidefinite Programs.

In order to make this thesis self-contained, we explain in Chapter 1 the basics about Semidefinite Programming and sketch the two most popular methods for solving Semidefinite Programs, namely Interior-Point methods and the Spectral Bundle Method. In Chapter 2 an introduction to Combinatorial Optimization and some of the problems arising in this field are given.

One of these problems arising from Combinatorial Optimization is the Max-Cut problem. We concentrate in this thesis on this NP-complete problem and therefore, Chapter 3 gives a more detailed description and explains methods for finding upper bounds or solving it. Furthermore it is shown in this chapter that solving Max-Cut problems and solving unconstrained quadratic (0-1) problems is essentially the same. Therefore and since many real-world problems can be formulated as unconstrained quadratic (0-1) problems, it is even more striking to have an algorithm that solves Max-Cut problems efficiently.

Chapter 4 is concerned with the Max-Cut problem as well. Within this chapter we take a closer look on relaxations based on Semidefinite Programming. Apart from those relaxations that work with matrix variables indexed by the vertex-set of the underlying graph, we also consider methods that apply a lift-and-project strategy. The latter relaxations, although being of highly theoretical interest, are practically not computable, already for medium-sized graphs. We introduce a new relaxation, that can be viewed as 'lying between' the basic semidefinite-programming relaxation and a first lifting, and that is solvable also for graphs on a few hundred nodes.

Having various Semidefinite Programming formulations that can provide bounds for NP-hard problems leads naturally to the demand of algorithms for solving these Semidefinite Programs. This issue is addressed in Chapter 5. We sketch there the concept of Bundle Methods and apply this concept to solving Semidefinite Programs, in particular we design the algorithm for solving two of the re-

laxations introduced in Chapter 4. Furthermore, we come back to the Spectral Bundle Method, which we equip with a second order model. And as a further algorithm, the Boundary Point Method is introduced within this chapter. This new method elaborates the idea of using the augmented Lagrangian algorithm for solving Semidefinite Programs.

The final chapter of this thesis, Chapter 6, provides an exact solution method for Max-Cut problems. The *Biq Mac* solver for solving *Bi*nary *Q*uadratic and *Max-C*ut problems is explained. Apart from the ingredients of this solver, i.e. a Branch & Bound framework, branching rules, implementation issues, etc., also a wide variety of test problems are collected and detailed numerical results are given.

Summarizing, this thesis provides the following new studies:

- A relaxation for sparse Max-Cut problems based on Semidefinite Programming (Section 4.4) and implementation of the Bundle Method to solve this relaxation (Section 5.1.1). This ongoing research can partly be found in [113].

- Exploiting second order information in the spectral bundle method (Section 5.2). See also the working paper [4].

- A boundary point method for solving Semidefinite Programs (Section 5.3). This work has been published in [110].

- Developing and implementing *Biq Mac*, an exact solution algorithm for solving Max-Cut and unconstrained (0-1) problems. Furthermore, a collection of test problems has been built up and numerical results are presented (Chapter 6). A technical report [115] is available.

Chapter 1

Semidefinite Programming

Many real-world applications, although being non-linear, can be well described by linearized models. Therefore, Linear Programming (LP) became a widely studied and applied technique in many areas of science, industry and economy.

Semidefinite Programming (SDP) is an extension of LP. A matrix-variable is optimized over the intersection of the cone of positive-semidefinite matrices with an affine space. It turned out, that SDP can provide significantly stronger practical results than LP. The study of SDP goes back to the sixties, when Bellman and Fan [19] derived theoretical properties of Semidefinite Programs. A first application appeared in the work of Lovász [88]. Since then SDP turned out to be practical in a lot of different areas, like combinatorial optimization, control theory, and more recently in polynomial optimization.

Due to the numerous areas of applications, also solving SDPs became a widely studied subject. Interior-Point Methods are the most popular algorithms nowadays. Recently the concept of Bundle Methods also has been applied for solving Semidefinite Programs.

In this chapter we formulate the Semidefinite Programming problem including duality theory. A subsection is dedicated to a related problem, namely Eigenvalue Optimization. Finally, Interior-Point Algorithms and the Spectral Bundle Method, two algorithms for solving Semidefinite Programs are explained.

Most of the proofs of Theorems and Lemmas in this section are omitted, because they appear in a wide variety of text-books or survey papers. For surveys on SDP the reader is referred to e.g. Helmberg [52], Vandenberghe and Boyd [126], Laurent and Rendl [80]. More references are given in the subsequent sections.

1.1 The Semidefinite Programming Problem

A Semidefinite Program in its standard notation is given as follows. Let C and $A_1, \ldots, A_m$ be matrices in $\mathcal{S}_n$ and $b \in \mathbb{R}^m$, we obtain

$$\begin{array}{lll} \textbf{(PSDP)} & \max & \langle C, X \rangle \\ & \text{s.t.} & \mathcal{A}(X) = b \\ & & X \in \mathcal{S}_n,\ X \succeq 0 \end{array}$$

where $\mathcal{A}\colon \mathcal{S}_n \to \mathbb{R}^m$ denotes a linear operator defined as

$$\mathcal{A}(X) = \begin{pmatrix} \langle A_1, X \rangle \\ \vdots \\ \langle A_m, X \rangle \end{pmatrix}.$$

The adjoint operator $\mathcal{A}^T\colon \mathbb{R}^m \to \mathcal{S}_n$, is defined through the equation

$$\langle \mathcal{A}(X), y \rangle = \langle X, \mathcal{A}^T(y) \rangle, \text{ for all } X \in \mathcal{S}_n,\ y \in \mathbb{R}^m. \tag{1.1}$$

Therefore,

$$\langle \mathcal{A}(X), y \rangle = \sum_{i=1}^{m} y_i \langle A_i, X \rangle = \sum_{i=1}^{m} \langle y_i A_i, X \rangle = \langle \sum_{i=1}^{m} y_i A_i, X \rangle = \langle \mathcal{A}^T(y), X \rangle$$

and hence,

$$\mathcal{A}^T(y) = \sum_{i=1}^{m} y_i A_i.$$

In order to derive the dual to **(PSDP)**, we introduce $y \in \mathbb{R}^m$ to be the Lagrangian multiplier for the equations in **(PSDP)**. Then the following always hold:

$$\begin{aligned} \max\{\langle C, X \rangle \colon \mathcal{A}(X) = b, X \in \mathcal{S}_n^+\} &= \max_{X \in \mathcal{S}_n^+} \min_{y \in \mathbb{R}^m} \langle C, X \rangle - \langle \mathcal{A}(X) - b, y \rangle \\ &\leq \min_{y \in \mathbb{R}^m} \max_{X \in \mathcal{S}_n^+} \langle b, y \rangle - \langle \mathcal{A}^T(y) - C, X \rangle \\ &= \min\{\langle b, y \rangle \colon \mathcal{A}^T(y) - C \in \mathcal{S}_n^+,\ y \in \mathbb{R}^m\}. \end{aligned}$$

The first equation is true, because if $\mathcal{A}(X) = b$ is not fulfilled, the minimization yields $-\infty$ and conversely, if the equation is satisfied, $\langle C, X \rangle$ is the result, i.e.

$$\min_{y \in \mathbb{R}^m} \langle C, X \rangle - \langle \mathcal{A}(X) - b, y \rangle = \begin{cases} \langle C, X \rangle & \text{for } \mathcal{A}(X) = b \\ -\infty & \text{otherwise.} \end{cases} \tag{1.2}$$

The inequality arises because of Lemma A.11 and for the last equation similar ideas as for the first hold, namely

$$\max_{X \in \mathcal{S}_n^+} \langle b, y \rangle - \langle \mathcal{A}^T(y) - C, X \rangle = \begin{cases} \langle b, y \rangle & \text{for } \mathcal{A}^T(y) - C \in \mathcal{S}_n^+ \\ \infty & \text{otherwise.} \end{cases} \tag{1.3}$$

Therefore, the dual to **(PSDP)** can be stated as

$$\begin{array}{lll} \textbf{(DSDP)} & \min & \langle b, y \rangle \\ & \text{s.t.} & \mathcal{A}^T(y) - C = Z \\ & & y \in \mathbb{R}^m,\ Z \in \mathcal{S}_n,\ Z \succeq 0. \end{array}$$

For referring to points satisfying the constraints in **(PSDP)** or **(DSDP)**, respectively, the following definitions will be useful:

Definition 1.1 (feasibility)
Matrix $X \in \mathcal{S}_n^+$ *is* feasible *for* **(PSDP)** *if* $\mathcal{A}(X) = b$ *holds.*
The pair $(y, Z) \in \mathbb{R}^m \times \mathcal{S}_n^+$ *is* feasible *for* **(DSDP)** *if* $\mathcal{A}^T(y) - C = Z$.

Definition 1.2 (strict feasibility)
Matrix $X \in \mathcal{S}_n^{++}$ *is* strictly feasible *for* **(PSDP)** *if* $\mathcal{A}(X) = b$ *holds.*
The pair $(y, Z) \in \mathbb{R}^m \times \mathcal{S}_n^{++}$ *is* strictly feasible *for* **(DSDP)** *if* $\mathcal{A}^T(y) - C = Z$.

1.2 Duality Theory

In Section 1.1 we introduced the primal and dual formulation of an SDP in standard form by using Lagrangian multipliers. Let X be a primal feasible solution and (y, Z) a dual feasible solution, then the difference between the objective values of the primal and dual feasible solution is defined as *duality gap.*

Definition 1.3 (duality gap) *Let* $X \in \mathcal{S}_n^+$ *and* $(y, Z) \in (\mathbb{R}^m \times \mathcal{S}_n^+)$, X *being feasible for* **(PSDP)** *and* (y, Z) *being feasible for* **(DSDP)**. *The* duality gap *at* (X, y, Z) *is given by*

$$\langle b, y \rangle - \langle C, X \rangle.$$

Due to Lemma A.4, the duality gap is always non-negative:

$$\langle b, y \rangle - \langle C, X \rangle = \langle \mathcal{A}(X), y \rangle - \langle \mathcal{A}^T(y) - Z, X \rangle = \langle Z, X \rangle \geq 0. \tag{1.4}$$

This fact is called *weak duality* and we formulate it as the following

Lemma 1.4 (weak duality) *Let* $X \in \mathcal{S}_n^+$, $y \in \mathbb{R}^m$ *with* $\mathcal{A}(X) = b$ *and* $\mathcal{A}^T(y) - C \in \mathcal{S}_n^+$. *Then*

$$\langle C, X \rangle \leq \langle b, y \rangle.$$

If X, y are feasible and the duality gap is zero, then *strong duality* holds and we have a proof, that these are the optimal solutions for **(PSDP)** and **(DSDP)**, respectively. On the other hand, strong duality does not necessarily hold for SDPs, as Vandenberghe and Boyd [126] exemplify:

Example 1.5 *Consider the following SDP*

$$\begin{array}{ll} \max & x_{12} \\ s.t. & \begin{pmatrix} 0 & x_{12} & 0 \\ x_{12} & x_{22} & 0 \\ 0 & 0 & 1+x_{12} \end{pmatrix} \succeq 0. \end{array}$$

For deriving the dual to this SDP, let us rewrite it so that the matrices defining the operator $\mathcal{A}$ become more evident:

$$\begin{array}{ll} \max & x_{12} \\ s.t. & \langle \begin{pmatrix} 0 & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, X \rangle = 1 \\ & \langle \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, X \rangle = 0 \\ & \langle \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, X \rangle = 0 \\ & \langle \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, X \rangle = 0. \end{array}$$

The dual to this problem reads

$$\min y_1 + 0y_2 + 0y_3 + 0y_4 \ s.t. \ y_1A_1 + y_2A_2 + y_3A_3 + y_4A_4 - C \succeq 0,$$

i.e.

$$\begin{array}{ll} \min & y_1 \\ s.t. & \begin{pmatrix} y_2 & \frac{1-y_1}{2} & y_3 \\ \frac{1-y_1}{2} & 0 & y_4 \\ y_3 & y_4 & y_1 \end{pmatrix} \succeq 0. \end{array}$$

Because of Observation A.5, x_{12} has to be equal to zero and therefore, the optimal objective value of **(PSDP)** *is zero. In the dual, $\frac{1-y_1}{2}$ has to be equal to zero, thus the optimal value is 1 and we have a duality gap of 1 for any primal and dual feasible solution.*

Hence, contrary to linear programming, it is no longer true that the duality gap has to be zero at the optimum.

One certificate that identifies problems with zero duality gap at the optimal solution is the *Slater constraint qualification*, defined as follows.

Definition 1.6 (Slater constraint qualification)
(PSDP) *satisfies the* Slater condition *if there exists* $X \in \mathcal{S}_n^{++}$ *with* $\mathcal{A}(X) = b$.
(DSDP) *satisfies the* Slater condition *if there exists a pair* (y, Z) *with* $Z \in \mathcal{S}_n^{++}$ *and* $\mathcal{A}^T(y) - Z = C$.

We can provide the following

Theorem 1.7 *Denote*

$$p^* = \sup\{\langle C, X\rangle \colon \mathcal{A}(X) = b, X \in \mathcal{S}_n^+\}$$

and

$$d^* = \inf\{\langle b, y\rangle \colon \mathcal{A}^T(y) - C \in \mathcal{S}_n^+\}.$$

- *If* **(PSDP)** *satisfies the Slater condition with* p^* *finite, then* $p^* = d^*$ *and this value is attained for* **(DSDP)**.
- *If* **(DSDP)** *satisfies the Slater condition with* d^* *finite, then* $p^* = d^*$ *is attained for* **(PSDP)**.
- *If* **(PSDP)** *and* **(DSDP)** *both satisfy the Slater condition, then* $p^* = d^*$ *is attained for both problems.*

A proof can be found for instance in Duffin [34], Nesterov and Nemirovskii [98] or Rockafellar [117]. Obviously, these conditions do not hold for Example 1.5.

As an example for an SDP where the primal optimal solution is not attained, we cite Helmberg [52]:

Example 1.8 *Consider the following Semidefinite Program and its dual:*

$$\begin{array}{ll} \max & -x_{11} \\ s.t. & \begin{pmatrix} x_{11} & 1 \\ 1 & x_{22} \end{pmatrix} \succeq 0. \end{array}$$

$$\begin{array}{ll} \min & 2y_1 \\ s.t. & \begin{pmatrix} 1 & y_1 \\ y_1 & 0 \end{pmatrix} \succeq 0. \end{array}$$

Due to Observation A.5, y_1 *has to be equal to zero to guarantee dual feasibility and thus, the optimal dual solution value is zero.*
The primal problem has a strictly feasible solution ($x_{11} = x_{22} = 2$, *for instance). The fact, that* X *has to be positive semidefinite constrains the variables to* $x_{11} \geq 0$, $x_{22} \geq 0$ *and* $x_{11}x_{22} - 1 \geq 0$. *Thus, we have* $x_{11} \geq \frac{1}{x_{22}}$ *with* x_{22} *non-negative. With* $x_{22} \to \infty$ *we get* $x_{11} = 0$ *and thus, the optimum is not attained for the primal problem.*

We have seen in (1.4) that a zero duality gap implies $\langle Z, X \rangle = 0$ and hence, $ZX = 0$ (due to Lemma A.4). This motivates the following

Definition 1.9 (Complementary slackness) *Matrices $X \in \mathcal{S}_n^+$ and $Z \in \mathcal{S}_n^+$ are* complementary *if $ZX = 0$.*

For problems where strong duality holds, we therefore obtain the following necessary and sufficient optimality conditions:

$$\begin{array}{lll} \textbf{(OPT)} & \mathcal{A}(X) = b,\ X \in \mathcal{S}_n^+ & (\textit{primal feasibility}) \\ & \mathcal{A}^T(y) - C = Z,\ Z \in \mathcal{S}_n^+,\ y \in \mathbb{R}^m & (\textit{dual feasibility}) \\ & ZX = 0 & (\textit{complementary slackness}) \end{array}$$

These optimality conditions play an important role for the development of interior-point algorithms for solving Semidefinite Programs and will appear again in Section 1.4.1.

1.3 Eigenvalue Optimization

Many practical applications lead to problems of Eigenvalue Optimization, for a survey the reader is referred to Lewis and Overton [85]. The simple observation

$$X \in \mathcal{S}_n^+ \Leftrightarrow \lambda_{\min}(X) \geq 0$$

(where $\lambda_{\min}(X)$ denotes the smallest eigenvalue of matrix X) indicates, that Eigenvalue Optimization and Semidefinite Programming are tightly related.

Denote by $\lambda_{\max}(X)$ the maximum eigenvalue of matrix X. We consider the following Eigenvalue Optimization problem

$$\textbf{(EVP)} \qquad \min a\lambda_{\max}(C - \mathcal{A}^T(y)) + b^T y \tag{1.5}$$

or equivalently

$$\textbf{(EVP)} \qquad \min a\lambda + b^T y \text{ s.t. } \lambda I \succeq C - \mathcal{A}^T(y), \tag{1.6}$$

with $a \in \mathbb{R}$ and, as in the previous section, $C \in \mathcal{S}_n$, $\mathcal{A}\colon \mathcal{S}_n \to \mathbb{R}^m$ and $b \in \mathbb{R}^m$.

To show the relation between **(EVP)** and Semidefinite Programming, consider **(PSDP)** and **(DSDP)** defined in Section 1.1. We make the following assumption:

$$\mathcal{A}(X) = b \Rightarrow \mathrm{tr}X = a > 0, \tag{1.7}$$

called the *constant trace property.* Adding this redundant constraint $\langle I, X \rangle = a$ to **(PSDP)**, results in the following dual (λ is the Lagrangian multiplier to the newly added constraint)

$$\begin{array}{lll} \textbf{(DSDP')} & \min & \langle b, y \rangle + \langle a, \lambda \rangle \\ & \text{s.t.} & \mathcal{A}^T(y) + \lambda I - C = Z \\ & & Z \in \mathcal{S}_n^+. \end{array}$$

Assuming that strong duality holds for the underlying problem, we have

$$\langle Z, X \rangle = 0,\ Z \in \mathcal{S}_n^+,\ X \in \mathcal{S}_n^+$$

at the optimum and due to Lemma A.4 follows

$$ZX = 0.$$

Therefore the optimal Z is singular (if Z would be non-singular, we obtain $X = 0$ which contradicts $\mathrm{tr}X > 0$) and all eigenvalues of $-Z$ must be non-positive with at least one eigenvalue equal to zero. Hence,

$$\begin{aligned} \lambda_{\max}(-Z) = 0 &\Leftrightarrow \lambda_{\max}(C - \mathcal{A}^T(y) - \lambda I) = 0 \\ &\Leftrightarrow \lambda_{\max}(C - \mathcal{A}^T(y)) - \lambda = 0 \\ &\Leftrightarrow \lambda = \lambda_{\max}(C - \mathcal{A}^T(y)). \end{aligned}$$

Substituting for λ in the objective function of (**DSDP'**), we obtain

$$\min a\lambda_{\max}(C - \mathcal{A}^T(y)) + b^T y,$$

which is **(EVP)**.

For easier notation, define

$$f(y) := \lambda_{\max}(C - \mathcal{A}^T(y)) + b^T y \tag{1.8}$$

to be the function to be minimized. (To simplify matters, we assume multiplier a to be equal to one.) Due to the well-known fact

$$\lambda_{\max}(X) = \max\{\langle W, X \rangle \colon \mathrm{tr}W = 1, W \in \mathcal{S}_n^+\} \tag{1.9}$$

we can rewrite (1.8) as

$$\begin{aligned} f(y) &= \max\{\langle C - \mathcal{A}^T(y), W \rangle + b^T y \colon \mathrm{tr}W = 1, W \in \mathcal{S}_n^+\} = \\ &= \max\{\langle C, W \rangle + (b - \mathcal{A}(W))^T y \colon \mathrm{tr}W = 1, W \in \mathcal{S}_n^+\}. \end{aligned}$$

Recall that function $\lambda_{\max}(.)$ is differentiable if and only if the maximal eigenvalue has multiplicity one. Typically, the largest eigenvalue has multiplicity larger than one for eigenvalue optimization problems and therefore one has to deal with the sub-differential of $\lambda_{\max}$ at X,

$$\partial\lambda_{\max}(X) = \{W \in \mathcal{S}_n^+ \colon \langle W, X \rangle = \lambda_{\max}(X), \mathrm{tr}W = 1\}$$

(confer for instance Overton [100]). For function (1.8) we then get

$$\partial f(y) = \{b - \mathcal{A}(W) \colon \langle W, C - \mathcal{A}^T(y) \rangle = \lambda_{\max}(C - \mathcal{A}^T(y)), \mathrm{tr}W = 1, W \in \mathcal{S}_n^+\}$$

by using standard rules (see Hiriart-Urruty and Lemaréchal [59]). Note that any matrix $W = vv^T$, with $v \in \mathbb{R}^n$ being an eigenvector to the maximal eigenvalue of X, is contained in the subdifferential of $\lambda_{\max}$ at X.

Let $\bar{y}$ be an optimal solution of (**EVP**). If $\bar{\lambda} = \lambda_{\max}(C - \mathcal{A}^T(\bar{y}))$ has multiplicity k, then there exists an $n \times k$ matrix P with $P^TP = I_k$ and a matrix $U \in \mathcal{S}_k$ with $\mathrm{tr}U = 1, U \succeq 0$ satisfying $\lambda I \succeq C - \mathcal{A}^T(\bar{y})$ and $(C - \mathcal{A}^T(\bar{y}))P = \lambda P$. And since $0 \in \partial f(\bar{y})$ must hold, we have $\mathcal{A}(PUP^T) = b$.

Therefore, we can state the following optimality certificate. y is optimal for (**EVP**) if and only if there exists P, U such that

$$\begin{aligned} & P^T(C - \mathcal{A}^T(y))P = \lambda I_k \\ & \lambda I \succeq C - \mathcal{A}^T(y) \\ & \mathcal{A}(PUP^T) = b \\ & P \in \mathbb{R}^{n \times k}, P^TP = I_k \\ & U \in \mathcal{S}_k, \mathrm{tr}U = 1, U \succeq 0 \end{aligned} \tag{1.10}$$

Finally, we want to mention that assumption (1.7) is valid for many relaxations arising from problems in combinatorial optimization.

1.4 On Solving Semidefinite Programming Problems

Semidefinite Programs are convex minimization problems and can therefore be solved in polynomial time to any fixed prescribed precision, using for instance the ellipsoid method, see Grötschel, Lovász, and Schrijver [43]. In practice better running times than the ellipsoid method are obtained by Interior-Point Methods (IPMs), which have been intensively studied in the nineties.

During the last decade Bundle Methods led to an alternative way of solving SDPs. The drawback of IPMs is that they are not capable of solving SDPs with a large number of constraints. In the Spectral Bundle method the number of constraints is not an issue and therefore this method is able to solve problems which are out of reach to be solved by IPMs.

1.4.1 Interior-Point Methods

Over the last years, Interior-Point Methods turned out to be the most popular algorithms for solving Semidefinite Programs. Most of the results go back to the nineties, when Semidefinite Programming became a strong tool for solving or approximating problems for several types of applications. Many variants of IPMs have been developed, a survey can be found in the book of de Klerk [29].

It turned out that the most efficient variants are the so-called primal-dual path-following methods, which we are going to explain here. The idea is to follow

approximately a *central path* in the interior of the feasible region to reach the optimum. This central path is obtained by replacing the optimality conditions by "nearly" optimality conditions.

Throughout this section we make the following

Assumption 1.10 *The Slater constraint qualification holds for* **(PSDP)** *and* **(DSDP)**.

Let us recall the necessary and sufficient optimality conditions for **(PSDP)** and **(DSDP)**.

$$\begin{array}{lll} \textbf{(OPT)} & \mathcal{A}(X) = b,\ X \in \mathcal{S}_n^+ & (primal\ feasibility) \\ & \mathcal{A}^T(y) - C = Z,\ Z \in \mathcal{S}_n^+,\ y \in \mathbb{R}^m & (dual\ feasibility) \\ & ZX = 0 & (complementary\ slackness) \end{array}$$

The idea is to replace the last condition by

$$ZX = \mu I$$

with $\mu > 0$ and let $\mu \to 0$. In order to derive this perturbed system, we define the following auxiliary problem.

$$\begin{array}{lll} \textbf{(PSDP}_\mu\textbf{)} & \min & \langle C, X\rangle - \mu \log\det(X) \\ & \text{s.t.} & \mathcal{A}(X) = b \\ & & X \in \mathcal{S}_n^{++}. \end{array}$$

$\mu > 0$ is the so-called *barrier parameter* and $-\log\det(X)$ the *barrier function*. Dualizing the equality constraints, we get the Lagrangian

$$\mathcal{L}_\mu(X, y) = \langle C, X\rangle - \mu \log\det(X) + \langle y, b - \mathcal{A}(X)\rangle \tag{1.11}$$

and compute the gradients with respect to X and y, respectively, in order to derive the KKT-conditions, necessary for optimality.

$$\begin{aligned} \nabla_X \mathcal{L}_\mu &= C - \mu X^{-1} - \mathcal{A}^T(y) \\ \nabla_y \mathcal{L}_\mu &= b - \mathcal{A}(X). \end{aligned}$$

(Note that $\nabla_X \log\det(X) = X^{-1}$.) Setting the gradients equal to zero, we get

$$\begin{array}{ll} \textbf{(OPT}_\mu\textbf{)} & \mathcal{A}(X) = b,\ X \in \mathcal{S}_n^{++} \\ & \mathcal{A}^T(y) + Z = C,\ Z \in \mathcal{S}_n^{++},\ y \in \mathbb{R}^m \\ & XZ = \mu I. \end{array}$$

Theorem 1.11 *Under Assumption 1.10,* **(OPT$_\mu$)** *has for all $\mu > 0$ a unique solution $(X(\mu), y(\mu), Z(\mu))$.*

A proof can be found for instance in Nesterov and Nemirovskii [98], Vandenberghe and Boyd [126] or Monteiro and Todd [96]. We define

Definition 1.12 (central path) *The smooth curve* $\{(X(\mu), y(\mu), Z(\mu)) : \mu > 0\}$ *is called the* primal-dual central path.

Let $\xi = (X, y, Z), X \in \mathcal{S}_n^{++}, Z \in \mathcal{S}_n^{++}$ be any point, not necessarily lying on the central path. The goal is, to find $\Delta\xi = (\Delta X, \Delta y, \Delta Z)$, such that $\xi + \Delta\xi$ comes closer to the central path and iterate with smaller μ until μ is sufficiently small (i.e. $\mu \to 0$).

The system to be solved in order to find the appropriate $\Delta\xi$, that would bring the current point on the central path is:

$$\begin{aligned} &\mathcal{A}(X + \Delta X) = b \\ &\mathcal{A}^T(y + \Delta y) - C = Z + \Delta Z \\ &(X + \Delta X)(Z + \Delta Z) = \mu I \end{aligned} \tag{1.12}$$

This system has $m + \frac{n(n+1)}{2} + n^2$ equations in $2\frac{n(n+1)}{2} + m$ variables. Due to the fact that the product of two symmetric matrices is not symmetric in general, this system of equations is overdetermined and we cannot apply the Newton method to solve it. Many variations of system 1.12 have been proposed, to fix this and to obtain reasonable search directions. Before explaining one of these search directions, we sketch a generic primal-dual path-following algorithm.

Algorithm 1.13 (generic primal-dual interior-point algorithm) *see Monteiro and Todd [96]*

Input.
$\xi_0 := (X_0, y_0, Z_0), X_0 \in \mathcal{S}_n^{++}, Z_0 \in \mathcal{S}_n^{++}, \varepsilon > 0.$
Initialization.
$\mu_0 := \langle X_0, Z_0 \rangle / n.$
$k := 0.$
while $\mu_k > \varepsilon$ *or* $\|\mathcal{A}(X_k - b)\|_\infty > \varepsilon$ *or* $\|\mathcal{A}^T(y_k) - C - Z_k\|_\infty > \varepsilon$
determine a search direction $\Delta\xi_k$ *from a linearized model of 1.12 for* $\mu = \sigma_k\mu_k$, $\sigma_k \in [0, 1]$, *such that* ΔX_k *and* ΔZ_k *are symmetric.*
$\xi_{k+1} := \xi_k + \alpha_k \Delta\xi_k$ *where* $\alpha_k > 0$ *is chosen, such that* $X_{k+1} \in \mathcal{S}_n^{++}$ *and* $Z_{k+1} \in \mathcal{S}_n^{++}$.
$\mu_{k+1} = \langle X_{k+1}, Z_{k+1} \rangle / n.$
$k := k + 1.$
end

About twenty different search directions have been reviewed by Todd [123]. We will use the HKM-direction, that was developed independently by Helmberg,

Rendl, Vanderbei, and Wolkowicz [56], Kojima, Shindoh, and Hara [74] and Monteiro [95]. They solve the following system to obtain a search direction:

$$\begin{aligned} &\mathcal{A}(\Delta X) = b - \mathcal{A}(X) \\ &\mathcal{A}^T(\Delta y) - \Delta Z = Z + C - \mathcal{A}^T(y) \\ &Z\Delta X + \Delta Z X = \mu I - ZX \end{aligned} \tag{1.13}$$

These equations are solved for $(\Delta X, \Delta y, \Delta Z)$ and then ΔX is symmetrized. Although this idea seems quite simple, it is computationally very efficient. Theoretical convergence analysis shows, that for small $\varepsilon > 0$ and appropriately chosen μ, in each iteration the full step yields a feasible solution. Moreover, a primal and dual feasible solution pair (X, y) with duality gap less than ε can be found after $O(\sqrt{n}|\log\varepsilon|)$ iterations (see Monteiro and Todd [96]).

1.4.2 Spectral Bundle Method

In Section 1.3 we have shown the relation between Eigenvalue Optimization and Semidefinite Programming. Helmberg and Rendl [55] developed the *Spectral Bundle Method*, a machinery to solve problem **(EVP)** and therefore, use this as an alternative to Interior-Point Methods for solving SDPs. Interior-Point Methods fail for SDPs with a large number m of constraints, since in every iteration a system of order m has to be solved. For these problems the Spectral Bundle Method may still obtain solutions in reasonable time. We explain the algorithm following [55] and [51].

Recall, that in Section 1.3 we introduced

$$f(y) := \lambda_{\max}(C - \mathcal{A}^T(y)) + b^T y, \tag{1.14}$$

the function to be minimized. Two ingredients are used to minimize this function: the *bundle concept* and the *proximal point idea*. To apply the bundle method, we need to have a function $\hat{f}$, approximating f in the neighborhood of the current iterate. Introduce

$$L(W, y) := \langle C - \mathcal{A}^T(y), W\rangle + b^T y.$$

With (1.9) we can now rewrite $f(y)$ as

$$f(y) = \max\{L(W, y)\colon W \in \mathcal{S}_n^+, \mathrm{tr}W = 1\}.$$

Replacing the feasible region $\{W\colon W \in \mathcal{S}_n^+, \mathrm{tr}W = 1\}$ by a subset that is computationally more efficient to handle, we have a minorant on f, that is easier to handle than f itself. The proposed subset is

$$\widehat{\mathcal{W}} = \{\alpha\overline{W} + PVP^T\colon \alpha + \mathrm{tr}V = 1, \alpha \geq 0, V \in \mathcal{S}_k^+\}, \tag{1.15}$$

where k is the number of columns in P and hence

$$\hat{f}(y) := \max\{L(W, y)\colon W \in \widehat{\mathcal{W}}\}. \tag{1.16}$$

P is constructed in a way, that it contains subgradient information of the current iterate, but keeping r, the maximum number of columns in P, small for computational simplicity. (Note that parameter r controls the dimension of V.) To be able of using more information without increasing r, $\overline{W}$ is used as an *aggregate subgradient*. Before going into detail concerning the construction of P and $\overline{W}$, we explain the second ingredient of the Spectral Bundle Method, namely the proximal point idea.

Due to the fact, that we deal with an approximation of f which is reliable only in the neighborhood of the current iterate, one has to penalize displacement from the current point, which results in

$$\min_y \hat{f}(y) + \frac{u}{2}\|y - \hat{y}\|^2, \tag{1.17}$$

where $u > 0$ is the penalty parameter. Concerning this parameter, Helmberg and Rendl [55] state the following

Remark 1.14 *The choice of the weight u is somewhat of an art. There are several clever update strategies published in the literature, see for instance Kiwiel [71], Schramm and Zowe [118].*

An iteration of the Spectral Bundle Method consists now in finding a new trial point y_{new} and depending on how much progress is made at this point, we do a *serious step* or a *null step*. To keep notation simple, we skip the iteration counter in the subsequent description of an iteration of the Spectral Bundle Method. Let $\hat{y}$ denote the current iterate. y_{new} is obtained as the minimizer of (1.17), where $\hat{f}$ is the minorant on f in the current iteration. This minimizer is obtained by first solving

$$\max_{W \in \widehat{\mathcal{W}}} \langle C - \mathcal{A}^T(\hat{y}), W\rangle + b^T\hat{y} - \frac{1}{2u}\langle \mathcal{A}(W) - b, \mathcal{A}(W) - b\rangle \tag{1.18}$$

by an Interior-Point Algorithm (see Section 1.4.1) and from this we get $W_{new} = \alpha^*\overline{W} + PV^*P^T$. The new iterate can then be easily computed by

$$y_{new} = \hat{y} + \frac{1}{u}(\mathcal{A}(W_{new}) - b). \tag{1.19}$$

If this new iterate shows significant progress on finding the optimum, we make a *serious step* and the current iterate becomes y_{new}. Otherwise a *null step* is made, i.e. the current iterate does not change, but information obtained during this iteration is used to improve the model.

Updating matrix P is done as follows. As long as P does not contain r columns, orthogonalize the new eigenvector with respect to P and add it as a new column. If the maximum number of columns in P is attained, we exploit the information available in α^* and V^*, being the maximizers of this iteration. Let $Q\Lambda Q^T$ be the eigenvalue decomposition of V^* and $Q = [Q_1, Q_2]$, with Q_1

containing the eigenvectors associated to the 'large' eigenvalues of V^*. Thus we can rewrite the current maximizer

$$W_{new} = PQ_1\Lambda_1(PQ_1)^T + \alpha^*\overline{W} + PQ_2\Lambda_2(PQ_2)^T. \tag{1.20}$$

Then P_{new} is computed such that it contains PQ_1 and at least one eigenvector to the current maximal eigenvalue of $C - \mathcal{A}^T(y_{new})$, i.e. P_{new} is an orthonormal basis of $[PQ_1\ v_{new}]$. The remaining information contained in Q_2 is included in the new aggregate matrix $\overline{W}_{new}$ by computing

$$\overline{W}_{new} = \frac{1}{\alpha^* + \mathrm{tr}\Lambda_2}(\alpha^*\overline{W} + PQ_2\Lambda_2(PQ_2)^T). \tag{1.21}$$

In this way it is ensured, that the new aggregate matrix $\overline{W}_{new}$ is contained in $\widehat{\mathcal{W}}_{new}$.

We now have derived the necessary formulas for giving the formal description of the algorithm.

Algorithm 1.15 (Spectral Bundle Method) *Helmberg and Rendl [55]*

Input.

$y_0 \in \mathbb{R}^m$ *and eigenvector* v_0 *to* $\lambda_{\max}(C - \mathcal{A}^T(y_0))$.
$\varepsilon > 0$, *improvement parameter* $m_L \in (0, \frac{1}{2})$.
weight $u > 0$, *upper bound* $R \geq 1$ *on the number of columns of* P.

Initialization.

$k = 0, x_0 = y_0, P_0 = v_0, \overline{W}_0 = v_0(v_0)^T$.

Iteration.

1. *(Direction finding.) Solve (1.18) and obtain* y_{k+1} *from (1.19).*
   Decompose V^* *into* $V^* = Q_1\Lambda_1Q_1^T + Q_2\Lambda_2Q_2^T$ *with* $rank(Q_1) \leq R - 1$.
   Compute $\overline{W}_{k+1}$ *using (1.21).*
2. *(Evaluation.) Compute* $\lambda_{\max}(C - \mathcal{A}^T(y_{k+1}))$ *and an eigenvector* v_{k+1}.
   Compute P_{k+1} *by taking an orthonormal basis of* $P_kQ_1v_{k+1}$.
3. *(Termination.) If* $f(x_k) - \hat{f}_k(y_{k+1}) \leq \varepsilon$ *then* **stop**.
4. *(Serious step.) If* $f(y_{k+1}) \leq f(x_k) - m_L(f(x_k) - \hat{f}_k(y_{k+1}))$ *then*
   set $x_{k+1} = y_{k+1}$ *and go to step 6.*
   Otherwise continue with step 5.
5. *(Null step.) Set* $x_{k+1} = x_k$.
6. *Increase* k *by 1 and go to Step 1.*

For the proof of convergence the reader is referred to Helmberg and Rendl [55]. A set of test graphs together with computational results for the Max-Cut relaxation and the Lovász ϑ-function (see Section 2.2) are provided in their paper. For these instances the Spectral Bundle Method is by far superior than Interior-Point Algorithms.

1.4.3 Software for Solving Semidefinite Programs

Interior-Point Algorithms, as well as the Spectral Bundle Method have been implemented as open source software. Some of the Interior-Point Codes are running under Matlab, for instance SeDuMi (Sturm [122]), or SDPT3 (Toh, Todd, and Tütüncü [125]), whereas e.g. CSDP by Borchers [24] is a C-code. The implementation of the Spectral Bundle Method is SBMethod (Helmberg). A list of links to the various packages can be found on the Semidefinite Programming Website maintained by Helmberg [49]. Mittelmann [94] runs a website, providing benchmarks for many SDP-solvers.

Chapter 2

Combinatorial Optimization

As the name reveals, in Combinatorial Optimization one wants to find an element out of a set of combinatorial objects that is the optimizer for some given objective function. More specifically, we have the following setting.

- A finite set $E = \{e_1, \ldots, e_n\}$,
- a weight function $w\colon E \to \mathbb{Z}$, $w(e_i)$ being the weight of e_i,
- a finite family $\mathcal{F} = \{F_1, \ldots, F_m\}$, $F_i \subseteq E$ (*feasible solutions*),
- a cost function $f : \mathcal{F} \to \mathbb{Z}$, $f(F) = \sum_{e \in F} w(e)$ (*additive cost function*),
- a problem
$$\text{opt}\{f(F)\colon F \in \mathcal{F}\},$$
where 'opt' is replaced by either 'min' or 'max'.

Usually, such problems can be formulated as Integer Programs with binary variables, which indicate for each member of the collection, whether it belongs to the subset or not.

A lot of problems fit into this definition. For example partitioning, assignment, covering, scheduling, shortest path, travelling salesman, spanning tree, matching, etc.

Before the year 1950, problems of this kind were studied independently of each other, for a historical survey see Schrijver [120]. Then Linear and Integer Programming became a unifying research topic and thus relations between these problems were found and exploited.

Over the past years new technologies in various areas like telecommunications, VLSI-design, production planning, etc. became more rapidly changing. Combinatorial Optimization turned out to appear in all of these applications and thus the research interest grew, since knowledge about problem properties and solution algorithms led to a competitive advantage.

Many textbooks on Combinatorial Optimization appeared during the last years, for a comprehensive collection on this subject we refer to Schrijver [119]. Some Combinatorial Optimization problems that are of special interest in the context of Semidefinite Programing are explained in this chapter.

2.1 The Max-Cut Problem

Let $G = (V(G), E(G))$ denote an edge-weighted undirected graph with vertex set $V(G) = \{1, \ldots, n\}$ and m edges in the edge set $E(G)$. Let w_e denote the weight of edge $e = [ij]$, meaning edge $e \in E(G)$ links vertices $i, j \in V(G)$. The Max-Cut (MC) problem consists in finding a partition of the set of vertices into two parts so as to maximize the sum of the weights of the edges that have one end-node in each part of the partition.

Let S be a subset of V. We denote a cut by

$$\delta(S) := \{e \in E(G) \colon e = [ij], |S \cap \{i, j\}| = 1\},$$

hence $\delta(S)$ contains all edges having exactly one end-node in S, which are the edges linking S and $V(G) \backslash S$.

$$w(T) := \sum_{e \in T} w_e$$

is the sum of the weights on edges in $T \subseteq E(G)$ and therefore the value of the cut given by $\delta(S)$ is given by $w(\delta(S))$ and the Max-Cut problem can be formulated as

$$\begin{array}{lll} \textbf{(MC)} & \max & w(\delta(S)) \\ & \text{s.t.} & S \subseteq V(G). \end{array}$$

Following the general formulation of a Combinatorial Optimization problem above, set E equals the set of edges $E(G)$, and $\mathcal{F}$ is the set of all cuts of G. The cost function is the sum of the weights on the edges that form the cut and the objective is to maximize these costs.

The Max-Cut problem is known to be NP-complete and is one of the problems on the original list of NP-complete problems, investigated by Karp [66]. It is not only of highly theoretical interest, but arises also in many contexts and therefore has been well-studied over the last years. Goemans and Williamson [39] show that the ratio between the optimal cut value and the solution value of the basic SDP relaxation of Max-Cut (**MCSDP**) (see Section 3.3.2), is at least 0.878 provided there are non-negative weights on the edges. Note, that Håstad [48] showed that it is NP-complete to approximate the Max-Cut problem with a factor bigger than 0.9412.

Various heuristics for finding good solutions, and relaxations for getting tight upper bounds have been developed. We will review some of them in Chapter 3.

2.2 The Stable Set Problem

A *stable set* or *independent set* in a given graph $G = (V(G), E(G))$ is a subset I of $V(G)$ such that no two vertices in I are adjacent. The maximum stable set problem is the problem of finding a stable set of maximum cardinality. This maximum cardinality is usually referred to as the *stability number* or *independence number* of a graph and denoted by $\alpha(G)$.

$$\alpha(G) = \max\{|I| \colon I \subseteq V(G), [ij] \notin E(G)\ \forall i, j \in I\}. \tag{2.1}$$

The stable set problem is closely related to two other problems, namely the maximum clique problem and the coloring problem.

A *clique* in a graph is defined as a subset Q of $V(G)$ such that all vertices in Q are joint by an edge $e \in E(G)$. The maximum clique problem is therefore the problem of finding a clique with maximum cardinality, denoted by $\omega(G)$,

$$\omega(G) = \max\{|Q| \colon Q \subseteq V(G), [ij] \in E(G)\ \forall i, j \in Q\}. \tag{2.2}$$

With $\overline{G} = (V(G), \overline{E(G)})$ being the complementary graph of $G = (V(G), E(G))$, it is easy to observe that

$$\alpha(\overline{G}) = \omega(G).$$

A *coloring* of a graph $G = (V(G), E(G))$ is a mapping $\beta \colon V(G) \to \{1, \ldots, k\}$, where $\{1, \ldots, k\}$ is the set of "colors" used, such that no two adjacent vertices are assigned the same color. The minimum k is the so-called *chromatic number* and is denoted by $\chi(G)$,

$$\chi(G) = \min\{k \colon \beta(i) \neq \beta(j) \text{ for } i, j \in V(G) \text{ and } [ij] \in E(G)\}. \tag{2.3}$$

Since within a clique every vertex needs to be colored differently, we get the following inequality:

$$\omega(G) \leq \chi(G).$$

This inequality can be strict, for instance consider C_5, a cycle with $|V| = 5$. ($\omega(C_5) = 2$ and $\chi(C_5) = 3$.)

A graph is said to be *perfect*, if $\omega(G') = \chi(G')$ for all induced subgraphs G' of G. This definition has been introduced by Berge, who also conjectured, that a graph is perfect if and only if it does not contain an odd cycle of length ≥ 5 or its complement as an induced subgraph (Berge [20], [21]). This conjecture was proved recently by Chudnovsky, Robertson, Seymour, and Thomas [28].

Lovász [88] introduced the ϑ-number of a graph. This number is the optimum of a semidefinite program and has the following property

$$\alpha(G) \leq \vartheta(G) \leq \chi(\overline{G}). \tag{2.4}$$

To compute the ϑ-number, the SDP to be solved is:

$$\begin{array}{lllll} \textbf{(THETA)} & \vartheta(G) = & \max & e^T X e \\ & & \text{s.t.} & \operatorname{tr}(X) = 1 \\ & & & X_{ij} = 0 \text{ for } i \neq j, [ij] \in E(G) \\ & & & X \in \mathcal{S}_n^+, \end{array}$$

e being the vector of all ones. (For equivalent definitions see Grötschel et al. [43] and Knuth [72].) The dual to **(THETA)** reads

$$\begin{array}{lll} \textbf{(DTHETA)} & \min & t \\ & \text{s.t.} & tI + \sum_{ij \in E(G)} \lambda_{ij} E_{ij} - J \in \mathcal{S}_n^+, \end{array}$$

where $J = ee^T$.

The problem of deciding for a given integer k, whether $\alpha(G) \geq k$ or $\chi(G) \leq k$ is NP-complete (Karp [66]). Moreover, Lund and Yannakakis [90] show that there is a constant $\varepsilon > 0$ such that no polynomial time algorithm exists that can achieve ratio n^ε for the coloring problem unless P=NP. For the stable set problem Arora, Lund, Motwani, Sudan, and Szegedy [6] show the existence of a constant $\varepsilon > 0$ for which there is no polynomial time algorithm that can find a stable set in a graph G of size at least $n^{-\varepsilon}\alpha(G)$ unless P=NP. On the positive side, Karger, Motwani, and Sudan [64] use Semidefinite Programming for coloring a k-colorable graph with maximum degree Δ with $O(\Delta^{1-2/k}\sqrt{\log \Delta \log n})$ or $O(n^{1-3/(k+1)}\sqrt{\log n})$ colors.

The fact that the ϑ-number can be computed in polynomial time and that it satisfies the 'sandwich' inequalities (2.4) makes it valuable for many applications. For perfect graphs it leads to the fact, that the maximum stable set problem and the coloring problem can be solved in polynomial time, since equality for the chromatic number and the clique number holds on these instances. For general graphs the gap between $\vartheta(G)$ and $\alpha(G)$ can be arbitrarily large. However, Alon and Kahale [2] state positive results about approximating $\alpha(G)$ via the ϑ-number.

2.3 The Graph Partitioning Problem

A problem related to Max-Cut is the graph partitioning problem. Again, we have a graph $G = (V(G), E(G))$, $|V(G)| = n$, and edge-weights w_e, $e \in E(G)$. Furthermore, numbers k and $m_1 \geq m_2 \geq \cdots \geq m_k$ are given, such that $\sum_{i=1}^k m_i = n$. We now like to find a partition of $V(G)$ into $V_1, V_2, \ldots, V_k$ and $|V_i| = m_i$, $i \in \{1, \ldots, k\}$, with a minimum total sum of the weights on the edges that are cut:

$$\textbf{(GP)} \quad \min \ \sum_{1 \leq s < t \leq k} \sum_{i \in V_s, j \in V_t} w_{[ij]}. \tag{2.5}$$

This problem plays a major role in circuit design, for detailed applications we refer to Lengauer [84]. In the special case of $k = 2$ and $m_1 = m_2 = n/2$, the problem is called the *bisection problem*. If there are no constraints on the cardinality of the

subsets, than for $k = 2$ and maximizing the sum of the weights on the cut-edges, we obtain the Max-Cut problem, see Section 2.1.

Let the columns of matrix $X \in \{0,1\}^{n\times k}$, $X = (x_{ij})$, be the characteristic vectors of the sets of the partition, i.e.

$$x_{ij} = \begin{cases} 1 & \text{if } i \in V_j \\ 0 & \text{otherwise.} \end{cases}$$

In order that each vertex $i \in V(G)$ is in exactly one set V_j, condition

$$Xe_k = e_n$$

must be valid. (e_k, e_n being the vectors of all ones of size k and n, respectively.) Furthermore, to ensure that m_i vertices are in the set V_i, the constraint

$$X^T e_n = m,$$

$m = (m_1, m_2, \ldots, m_k)^T$, must be fulfilled.

Let $A = (a_{ij})$ be the adjacency matrix of the underlying graph. The value

$$\frac{1}{2}\text{tr}AXX^T = \frac{1}{2}\text{tr}X^TAX$$

gives the sum of the weights on all edges that are *not* cut and therefore the weight of the edges that are cut by this partition can be computed as

$$\frac{1}{2}(e^TAe - \text{tr}X^TAX).$$

With $L = \text{Diag}(Ae) - A$ being the Laplace matrix of the graph and the equality

$$\text{tr}X^T\text{Diag}(Ae)X = e^TAe,$$

we can formulate problem **(GP)** as follows.

$$\begin{array}{lll} \textbf{(GP)} & \min & \text{tr}X^TLX \\ & \text{s.t.} & Xe_k = e_n \\ & & X^Te_n = m \\ & & X \in \{0,1\}^{n\times k}. \end{array} \tag{2.6}$$

Barnes and Hoffman [15] and Donath and Hoffman [33] developed eigenvalue based relaxations for this problem. The problem is relaxed to containing only the constraint

$$X^TX = \text{Diag}(m),\ X \in \mathbb{R}^{n\times k}.$$

Through Theorem 2.1 Donath and Hoffman [33] obtain an eigenvalue based bound.

Theorem 2.1 *Let A and m be defined as above and set $M := \mathrm{Diag}(m)$. Then*

$$\begin{aligned} |w(uncut)| &\leq \max\{\frac{1}{2}\mathrm{tr}X^TAX\colon X^TX = M\} \\ &= \min\{\frac{1}{2}\mathrm{tr}MY^TAY\colon Y^TY = I_k\} = \frac{1}{2}\sum_{j=1}^{k} m_j\lambda_j(A). \end{aligned}$$

Thus we get

$$|w(cut)| \geq \frac{1}{2}(e^TAe - \sum_{j=1}^{k} m_j\lambda_j(A)).$$

The proof can be found, for instance in Donath and Hoffman [33] or Rendl and Wolkowicz [114].

Later on further SDP based bounds have been developed, confer Alizadeh [1], Rendl and Wolkowicz [114], Wolkowicz and Zhao [127], Karisch and Rendl [65].

Besides that, formulation (2.6) is similar to the *Quadratic Assignment Problem* (QAP). The latest SDP relaxations of the QAP are investigated in the paper of Rendl and Sotirov [112].

2.4 The Max-Sat Problem

In order to explain the Maximum Satisfyability problem, we first need to introduce some notation. $x_1, \ldots, x_n$ are *Boolean variables* and a *literal* z is either x_i or $\overline{x_i}$ (the negation of x_i). A *clause* C of length k is the disjunctive combination of k literals, i.e. $C = z_1 \vee \cdots \vee z_k$, a weight w_C is assigned to each clause C. Clearly, clause C is satisfied, if at least one of the literals in the clause is assigned value 1. The Max-Sat problem consists in finding an assignment of values 0 and 1 to the variables $x_1, \ldots, x_n$ such that the total sum of the weights of satisfied clauses is maximized. Given an integer $k \geq 1$, with the additional requirement that each clause has length at most k, the problem is called Max-kSat.

Max-Sat and Max-kSat are known to be NP-hard. Håstad [48] showed that there is no $(\frac{7}{8} + \varepsilon)$-approximation for any $\varepsilon > 0$, unless P=NP.

Johnson [62] constructed a $\frac{1}{2}$-approximation algorithm for Max-Sat. A linear programming relaxation leads to the $\frac{3}{4}$-approximation of Goemans and Williamson [38].

Via Semidefinite Programming, Goemans and Williamson [39] improved slightly their $\frac{3}{4}$-approximation and obtained a 0.7554-approximation for Max-Sat.

Chapter 3

The Maximum Cut Problem

In this chapter we will take a closer look on one of the NP-complete combinatorial optimization problems, the Max-Cut problem, already defined in Chapter 2. We want to state some of the important properties and give an overview on solution methods. It is easy to see, that the Max-Cut problem can be transformed to a quadratic (0-1) problem and vice versa. We explicate this transformation and point out an essential difference between Max-Cut problems and instances arising from quadratic (0-1) problems.

3.1 Properties of the Max-Cut Problem

The Max-Cut problem on a graph $G = (V(G), E(G))$, previously defined in Section 2.1, is given as

$$\begin{array}{lll} \textbf{(MC)} & \max & w(\delta(S)) \\ & \text{s.t.} & S \subseteq V(G). \end{array}$$

For several applications the following notation will be more convenient. Let $V(G) := \{1, \ldots, n\}$ be the vertex set of the given graph. The weights on the edges are expressed through the weighted adjacency matrix $A = (a_{ij})$, where

$$a_{ij} = a_{ji} = \begin{cases} w_e & \text{if } e = [ij] \in E(G) \\ 0 & \text{otherwise.} \end{cases}$$

Given A, we introduce the Laplacian matrix $L = (l_{ij})$ associated to A, which is defined as

$$l_{ii} = \sum_{k=1}^{n} a_{ik}, \forall i \in V(G)$$

$$l_{ij} = -a_{ij}, \ i \neq j, \ i, j \in V(G),$$

hence $L = \text{Diag}(Ae) - A$.

A vector $x \in \{\pm 1\}^n$ represents a cut in the graph in the sense that the sets $\{i : x_i = 1\}$ and $\{i : x_i = -1\}$ form a partition of the vertex set of the graph, i.e. $S = \{i : x_i = 1\}$ and hence $V \backslash S = \{i : x_i = -1\}$. It is easy to verify, that the weight of the cut given by S, can be computed as $w(\delta(S)) = \frac{1}{4} x^T L x$:

$$\begin{aligned} x^T L x &= \sum_{i=1}^{n} l_{ii} x_i^2 + 2 \sum_{1 \le i < j \le n} l_{ij} x_i x_j = \\ &= \sum_{i=1}^{n} (\sum_{k=1}^{n} a_{ik}) + 2 \sum_{[ij] \notin \delta(S)} (-a_{ij}) \cdot 1 + 2 \sum_{[ij] \in \delta(S)} (-a_{ij}) \cdot (-1) = \\ &= 2 \sum_{[ij] \in V(G)} a_{ij} + 2 \sum_{[ij] \notin \delta(S)} (-a_{ij}) + 2 \sum_{[ij] \in \delta(S)} a_{ij} = \\ &= 4w(\delta(S)). \end{aligned}$$

(Note that $x_i x_j = -1$ if $[ij] \in \delta(S)$ and $x_i x_j = 1$ otherwise.) Hence, Max-Cut is equivalent to

$$\begin{array}{lll} \textbf{(MC)} & \max & x^T L x \\ & \text{s.t.} & x \in \{\pm 1\}^n. \end{array} \tag{3.1}$$

Another way of specifying a cut is via its *incidence vector*, a vector indexed by the edge set of the graph and defined as follows.

$$\chi_e^S = \begin{cases} 1 & \text{if } e \in \delta(S) \\ 0 & \text{otherwise.} \end{cases}$$

Let CUT denote the *cut polytope*, i.e. the convex hull of all incidence vectors of cuts of graph G,

$$\text{CUT} = \text{conv}\{\chi^{\delta(S)} : S \subseteq V(G)\}.$$

Thus, a third version of formulating the Max-Cut problem is given by the following linear program:

$$\begin{array}{lll} \textbf{(MC)} & \max & w^T y \\ & \text{s.t.} & y \in \text{CUT}. \end{array} \tag{3.2}$$

Many theoretical results of the cut polytope are elaborated in the book of Deza and Laurent [32]. Barahona and Mahjoub [13] characterize the facet defining inequalities of the cut polytope and show different methods for constructing these inequalities from known ones. Other papers dealing with the cut polytope are for instance Barahona [10], Poljak and Tuza [108], Poljak [105].

The Max-Cut problem is known to be NP-complete (Karp [66]) and it remains NP-complete for some restricted versions, see Garey and Johnson [36]. However, several classes of graphs are known for which the solution can be obtained in polynomial time. To these classes belong graphs without long odd cycles (Grötschel and Nemhauser [41]), planar graphs (Hadlock [45],Orlova and Dorfman [99]), or more generally graphs not contractible to K_5 (Barahona [9]). More properties for certain classes of graphs are surveyed in Poljak and Tuza [109].

3.2 Quadratic (0-1) Programming and its Relation to Max-Cut

In this section we want to show, that solving a quadratic (0-1) problem and solving a Max-Cut problem is essentially the same. Given a matrix Q of order n and a vector c, define the quadratic function

$$q(y) := y^T Q y + c^T y. \tag{3.3}$$

We consider the following unconstrained quadratic (0-1) program:

$$\begin{array}{lll} \textbf{(QP)} & \min & q(y) \\ & \text{s.t.} & y \in \{0,1\}^n. \end{array} \tag{3.4}$$

This problem is equivalent to **(MC)**, which has first been pointed out by Hammer [46]. The reduction from **(QP)** to **(MC)** has also been carried out in Barahona, Jünger, and Reinelt [14], a compact table of the transformation can be found in Helmberg [50]. For completeness we show in detail in the subsequent two subsections how to transform one problem into the other.

3.2.1 (QP)→ (MC)

Define

$$W = -\begin{pmatrix} 0 & (Qe+c)^T \\ (Qe+c) & Q \end{pmatrix}$$

and consider W to be the adjacency matrix of a graph with vertex set $V = \{0, 1, \ldots, n\}$. Then the Laplacian is given by

$$\begin{aligned} L &= \mathrm{Diag}(We) - W = \\ &= \begin{pmatrix} 0 & (Qe+c)^T \\ (Qe+c) & Q \end{pmatrix} - \begin{pmatrix} e^T Qe + c^T e & 0 \\ 0 & \mathrm{Diag}(2Qe+c) \end{pmatrix}. \end{aligned}$$

Let x denote the incidence vector of a cut of this graph with value $\frac{1}{4}x^T Lx$. Without loss of generality we can assume $x_0 = 1$. Then, y defined as

$$y_i = \frac{1}{2}(x_i + 1), \quad 1 \le i \le n$$

is a vector in $\{0,1\}^n$ and therefore solution of **(QP)**.

The solution value of **(MC)** of the adjacency matrix W expressed in terms of y through the equality $x_i = 2y_i - 1$, $1 \le i \le n$ is the following:

$$
\begin{aligned}
x^T L x &= \begin{pmatrix} 1 \\ 2y-e \end{pmatrix}^T \Big(\begin{pmatrix} 0 & (Qe+c)^T \\ (Qe+c) & Q \end{pmatrix} \\
&\quad - \begin{pmatrix} q(e) & 0 \\ 0 & \mathrm{Diag}(2Qe+c) \end{pmatrix} \Big) \begin{pmatrix} 1 \\ 2y-e \end{pmatrix} = \\
&= \begin{pmatrix} 1 \\ 2y-e \end{pmatrix}^T \begin{pmatrix} 0 & (Qe+c)^T \\ (Qe+c) & Q \end{pmatrix} \begin{pmatrix} 1 \\ 2y-e \end{pmatrix} \\
&\quad - \begin{pmatrix} q(e) \\ \mathrm{Diag}(Qe) + \mathrm{Diag}(Qe+c) \end{pmatrix}^T \begin{pmatrix} 1 \\ e \end{pmatrix} = \\
&= (0 + 4(Qe+c)^T y - 2q(e) + 4y^T Q y - 4e^T Q y + e^T Q e) \\
&\quad -(q(e) + e^T Q e + q(e)) = \\
&= 4(y^T Q y + c^T y - q(e)).
\end{aligned}
$$

Therefore,

$$y^T Q y + c^T y = \frac{1}{4} x^T L x + q(e).$$

3.2.2 (MC)→ (QP)

Conversely, let be given a graph with node set $V = \{0, 1, \ldots, n\}$ and the $(n+1) \times (n+1)$ Laplacian

$$L = \begin{pmatrix} l_{11} & L_{12}^T \\ L_{12} & L_{22} \end{pmatrix}$$

where L_{22} is a $n \times n$ matrix. Let y be a solution of (**QP**) with $Q = L_{22}$ and $c = L_{12} - L_{22}e$. Then, $\bar{x} = \begin{pmatrix} x_0 \\ x \end{pmatrix}$, $x_0 \in \mathbb{R}, x \in \mathbb{R}^n$ defined as

$$x_0 = 1, \; x = 2y - e$$

is a vector $\{\pm 1\}^{n+1}$ and therefore a solution of (**MC**). The value of the cut associated to this solution, in terms of y is as follows:

$$
\begin{aligned}
q(y) &= y^T Q y + c^T y \\
&= y^T L_{22} y + (L_{12} - L_{22}e)^T y \\
&= \frac{1}{4}(x+e) L_{22} (x+e) + \frac{1}{2}(L_{12} - L_{22}e)^T (x+e) \\
&= \frac{1}{4}(x^T L_{22} x^T + 2e^T L_{22} x + e^T L_{22} e) + \frac{1}{2}(L_{12}^T x + L_{12}^T e - e^T L_{22} x - e^T L_{22} e) \\
&= \frac{1}{4}(x^T L_{22} x + 2L_{12}^T x + l_{11} - l_{11} - e^T L_{22} e + 2L_{12}^T e) \\
&= \frac{1}{4} \bar{x}^T L \bar{x} - (l_{11} - 2L_{12}^T e + e^T L_{22} e).
\end{aligned}
$$

Therefore,

$$\frac{1}{4}\bar{x}^T L\bar{x} = q(y) + e^T \begin{pmatrix} l_{11} & -L_{12}^T \\ -L_{12} & L_{22} \end{pmatrix} e.$$

3.2.3 (MC) vs. (QP)

For all algorithms available, it turned out that solving instances of (**MC**) seems to be a much harder job than solving instances arising from (**QP**). In order to investigate this behaviour, let us take a closer look on two random instances. We generate an unweighted random graph with $n = 25$ vertices and edge-probability $\frac{1}{2}$. Also, we generate a random instance of (**QP**), where all entries in Q and c are chosen from $[-100, 100]$. Due to the small size of these problems, we are able to enumerate all 2^{24} solutions, and plot the sorted and normalized objective values in Figure 3.1.

The picture nicely shows that objective values of the (**QP**) instances are quite evenly spread over the interval of possible values. Contrary, for (**MC**) the density of cut values in the top quarter of the interval is clearly much higher than in the remaining part. For the (**MC**) instance, half of the solution values are within 25% of the optimum, whereas for the (**QP**) instance only 0.5% are in that 25% region.

It is evident that the optimal solution is much harder to identify when ten-thousands of solutions lie within a 5% interval of the optimum, as it is the case of the (**MC**) instance, whereas for the (**QP**) instance only a few hundred are that close. (The bottom-plot in Figure 3.1 shows for both problems the best 10,000 objective values.) Therefore, solving problems originated from (**MC**) are obviously more challenging than (**QP**) problems.

3.3 Relaxations of the Max-Cut Problem

In this section we recall the most popular relaxations of the Max-Cut problem together with some of the recent methods for solving it to optimality. We sketch the algorithms and summarize their limits. A survey of techniques developed before 1980 can be found in Hansen [47].

3.3.1 Relaxations Based on Linear Programming

Consider the linear program (3.2). For a graph $G = (V, E)$ define $y \in \mathbb{R}^E$ as $y(E) := \sum_{e \in E} y_e$. The observation that any odd cycle intersects with a cut on an even number of vertices motivates the construction of the *odd cycle inequalities*:

$$y(F) - y(C \backslash F) \leq |F| - 1 \text{ for each cycle } C \subseteq E,\ F \subseteq C,\ |F| \text{ odd.} \qquad (3.5)$$

A special class of odd cycle inequalities are the *triangle inequalities*, which arise when C in (3.5) is a cycle of length three, i.e. a triangle. For $F = C$ (hence

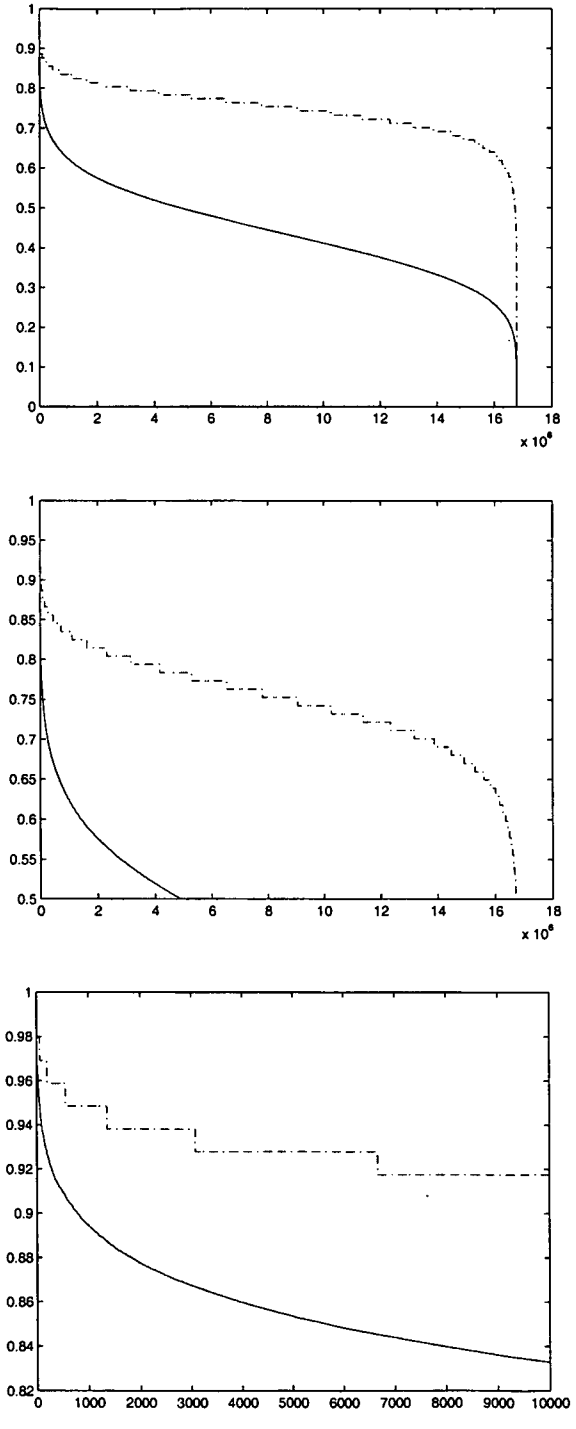

Figure 3.1: Random data, $n = 25$. Plot on top: sorted and normalized cost values for an unweighted random graph (dashed-dotted line) and a random QP instance (dashed line). Plot in the middle: plotting only those values, which are within 50% of the optimum. Plot on bottom: zooming in to the 10,000 best solutions.

$|F| = 3$), we get the inequality $y(F) \leq 2$ and for $F \subset C$ ($|F| = 1$) we obtain $y(F) - y(C \backslash F) \leq 0$. So, if C is formed by the edges $[ij], [ik], [jk]$, we obtain

$$\begin{array}{rcl} y_{ij} + y_{ik} + y_{jk} & \leq & 2 \\ y_{ij} - y_{ik} - y_{jk} & \leq & 0 \\ -y_{ij} + y_{ik} - y_{jk} & \leq & 0 \\ -y_{ij} - y_{ik} + y_{jk} & \leq & 0 \end{array} \tag{3.6}$$

The odd cycle inequalities and also the trivial inequalities $0 \leq y_e \leq 1, e \in E$ are all valid for any $y \in$ CUT (the cut polytope, see Section 3.1). Therefore, a linear programming relaxation of the Max-Cut problem can be derived by replacing the constraint $y \in$ CUT in (3.2) by the odd cycle inequalities and $0 \leq y_e \leq 1, e \in E$. Nevertheless, this LP has then an exponential number of inequalities and therefore an attempt of feeding this problem into some LP solver might already fail when specifying all the constraints. On the other hand, Grötschel, Lovász, and Schrijver [42] show that one can optimize a linear objective function over a polytope in polynomial time if and only if one can solve the separation problem for this polytope in polynomial time. Barahona and Mahjoub [13] give a polynomial time algorithm for separating the cycle inequalities and thus a cutting plane approach can be developed, where the LP relaxations can be exploited by using the cycle-inequalities in an iterative algorithm.

Barahona et al. [14] designed such a cutting plane algorithm within a Branch & Bound framework that uses these inequalities. They solve in the root node the trivial LP

$$\begin{array}{ll} \max & w^T y \\ \text{s.t.} & 0 \leq y_e \leq 1, \ e \in E \end{array}$$

and generate then cutting planes not only at the root, but also at each node of the Branch & Bound tree. They sketch the cutting plane procedure performed at each node as follows:

begin
  repeat
    solve LP;
    obtain lower bound;
    if successful **then** try to fix variables;
    try to generate cutting planes;
    revise LP;
  until no cutting planes generated;
  if LP solution feasible
    then backtrack
    else branch
end

To **obtain a lower bound** (i.e. finding a cut in the graph), a heuristic is applied to the solution obtained by solving the LP. This heuristic computes a

maximum spanning tree in the original graph with edge weights $|y_e - \frac{1}{2}|$ ($y \in \mathbb{R}^E$ is the LP solution) and assigns the vertices to one of the two subsets of the partition according to the weights on the edges of this tree. This yields a feasible solution to the Max-Cut problem.

The lower bound serves for fathoming nodes in the Branch & Cut tree, but is also used for **fixing variables**. If $y_e = 0$ and $z_{LP} - d_e < z_F$, where z_{LP} is the objective function value, $d \in \mathbb{R}^E$ the reduced cost vector and z_F the value of the best known cut in G, clearly we can fix the variable associated to this edge to 0. Similarly, if $y_e = 1$ and $z_{LP} + d_e < z_F$, we can fix the variable to 1. Furthermore, edges, that belong to a subgraph induced through the edges fixed to 0 or 1, can be fixed by logical implications.

Odd cycle inequalities are used to **generate cutting planes**. Several ideas are incorporated for finding violated odd cycle inequalities. Barahona et al. [14] proceed according to the following order, until violated inequalities are found:

1. Enumerate all 3-cycles.
2. Apply a coloring heuristic for finding violated odd-cycle inequalities. This heuristic guarantees, that in an integral solution, that is not a cut, violated odd-cycle inequalities will be found.
3. Apply a spanning tree heuristic to detect violated odd-cycle inequalities.
4. Use exact separation (see Barahona and Mahjoub [13]).

Branching is done by choosing the variable x_e with fractional value closest to $\frac{1}{2}$, and among those one with maximum absolute objective function coefficient.

Recent results on a refinement of this LP based cutting plane algorithm are due to Liers, Jünger, Reinelt, and Rinaldi [87]. They focus on solving toroidal grid graphs arising from physical applications. Since these graphs are sparse, LP based method are the proper tool for solving these instances.

Limits of this method: The computational results presented in Barahona et al. [14] show that graphs of any density up to $n = 30$ nodes can be computed in reasonable time. But with an increasing number of nodes, the limits on the density of the graphs decreases rapidly. Graphs with $n = 100$ nodes can only be solved, if the edge density is at most 10%. The algorithm of Liers et al. [87] solves 3-dimensional toroidal grid graphs with Gaussian distributed weights of size $7 \times 7 \times 7$ within minutes and 2-dimensional of size 20×20 within seconds. However, for dense instances also this algorithm is not practical.

3.3.2 A Basic SDP Relaxation

Consider (MC) formulated as (3.1) and do a transformation of variables, namely

$$X := xx^T.$$

Hence X has the properties that it is positive semidefinite, it has rank one and all diagonal elements are equal to one. Furthermore, the value of a cut associated to X can be computed as

$$\frac{1}{4}x^T Lx = \frac{1}{4}\mathrm{tr}LX = \frac{1}{4}\langle L, X\rangle.$$

Thus an equivalent formulation of the Max-Cut problem is

$$\begin{array}{lll} \textbf{(MC)} & \max & \langle L, X\rangle \\ & \text{s.t.} & \mathrm{diag}(X) = e \\ & & \mathrm{rank}(X) = 1 \\ & & X \in \mathcal{S}^n,\ X \succeq 0. \end{array} \tag{3.7}$$

A semidefinite relaxation can be obtained by simply dropping the rank-1 constraint:

$$\begin{array}{lll} \textbf{(MCSDP)} & \max & \langle L, X\rangle \\ & \text{s.t.} & \mathrm{diag}(X) = e \\ & & X \in \mathcal{S}^n,\ X \succeq 0. \end{array} \tag{3.8}$$

Its dual form

$$\begin{array}{lll} \textbf{(MCDSDP)} & \min & e^T u \\ & \text{s.t.} & \mathrm{Diag}(u) - L \succeq 0 \end{array} \tag{3.9}$$

was introduced by Delorme and Poljak [30] as the (equivalent) eigenvalue optimization problem

$$\begin{array}{lll} \textbf{(MCEIG)} & \min & n\lambda_{\max}(L - \mathrm{Diag}(u)) \\ & \text{s.t.} & u^T e = 0 \\ & & u \in \mathbb{R}^n. \end{array} \tag{3.10}$$

The primal version **(MCSDP)** can be found in Poljak and Rendl [107].

The model **(MCEIG)** is used in Poljak and Rendl [106] as the bounding routine in a Branch & Bound framework.

Limits of this method: This basic SDP bound can be computed rather cheaply by using for instance an Interior-Point algorithm. However, within a Branch & Bound scheme the progress of the bound at each node of the B&B tree is disappointingly small and therefore the number of nodes in this tree becomes rather large, already for medium sized problems. The maximum cut in graphs up to $n = 50$ nodes can be computed quite efficiently, but for larger n a solution in reasonable time can only be obtained for instances where the initial gap is already very small.

Further SDP based MC relaxations. This basic relaxation has been exploited in various ways during the past decade. For example it can be strengthened by the so-called hypermetric inequalities. Other relaxations of **(MC)** arising from SDP are the so-called lift-and-project methods. A separate chapter is dedicated to these SDP relaxations (Chapter 4).

3.3.3 Convex Quadratic Relaxations

Billionnet and Elloumi [23] came up with the idea of convexifying the objective function and then using a Mixed-Integer Quadratic Programming (MIQP) solver for solving problem (3.4). Their algorithm works in detail as follows. Consider problem **(QP)** and define for any vector $u \in \mathbb{R}^n$ the Lagrangian

$$q_u(x) := q(x) + \sum_{i=1}^{n} u_i(x_i - x_i^2) = x^T(Q - \mathrm{Diag}(u))x + (c+u)^T x.$$

It is easy to see, that an equivalent problem to **(QP)** is

$$(\mathbf{QP_u}) \quad \begin{array}{ll} \min & q_u(x) \\ \text{s.t.} & x \in \{0,1\}^n. \end{array} \tag{3.11}$$

Relaxing the integrality constraint in problem $(\mathbf{QP_u})$ gives the lower bound $\beta(u)$ on **(QP)**,

$$\beta(u) \;=\; \begin{array}{ll} \min & q_u(x) \\ \text{s.t.} & 0 \le x_i \le 1,\ i \in \{1,\dots,n\}. \end{array}$$

If the vector u is chosen, such that $Q - \mathrm{Diag}(u)$ is positive semidefinite, $\beta(u)$ is obtained by solving a convex quadratic problem, which can be done efficiently. Now, if u^* is the maximizer of $\beta(u)$, the "optimal" lower bound β^* will be obtained, i.e.

$$\beta^* = \beta(u^*) = \max\{\beta(u) : (Q - \mathrm{Diag}(u)) \succeq 0, u \in \mathbb{R}^n\}.$$

Billionnet and Elloumi [23] observe, that the dual to this SDP coincides with the basic Max-Cut relaxation **(MCSDP)**, see Section 3.3.2.

The solution of problem $(\mathbf{QP_u})$ (or $(\mathbf{QP_{u^*}})$, respectively) can be derived by using an MIQP solver, i.e. a Branch & Bound algorithm using $\beta(u)$, the continuous relaxation of $(\mathbf{QP_u})$, as bound.

The computational effort for this algorithm can be summarized as follows:

- Preprocessing phase: solve an SDP to obtain a vector u^* and a bound β^*.
- Use an MIQP solver for solving problem $(\mathbf{QP_{u^*}})$. Even though the computation of the bounds is very cheap, the number of nodes in the Branch & Bound tree typically exceeds 100,000 for problems of $n = 100$ variables, as reported in [23].

Limits of this method: Quadratic problems with some special structure can be solved up to $n = 100$ variables. But the method is not capable of solving certain classes of Max-Cut instances of this size (for example, graphs with edge weights chosen uniformly from $\{-1/0/1\}$).

3.3.4 Second-Order Cone Programming Relaxations

Kim and Kojima [68], and later on Muramatsu and Suzuki [97] use a second-order cone programming (SOCP) relaxation as bounding routine in a Branch & Bound framework to solve Max-Cut problems. Second-order cone programming is a special case of symmetric cone-programming. The second-order cone $\mathcal{K}_n$ is defined by

$$\mathcal{K}_n = \left\{ x \in \mathbb{R}^n \colon x_1 \geq \sqrt{\sum_{i=2}^{n} x_i^2} \right\}.$$

SOCP can be used to relax nonconvex quadratic problems. Muramatsu and Suzuki [97] propose an SOCP relaxation of (MC) that includes convex quadratic constraints, which reflect the structure of the graph. They are able to incorporate the triangle inequalities (see Section 3.3.1) to tighten the feasible region efficiently.

However, the basic SDP relaxation (see Section 3.3.2) performs better than their SOCP relaxation and the method works only for sparse graphs.

Limits of this method: The algorithm is capable of solving very sparse instances only. The largest graphs for which solutions are reported are random graphs (weights between 1 and 50) of $n = 120$ nodes and density 2%, and graphs, which are the union of two planer graphs up to $n = 150, d = 2\%$.

3.3.5 Branch & Bound with Preprocessing

Pardalos and Rodgers [102], [103] solve the quadratic program by Branch & Bound using a preprocessing phase where they try to fix some of the variables. The function to be minimized is (3.3). The test on fixing the variables exploits the fact, that if x^* is the global solution of

$$\min\{q(x) : x \in S\}$$

(S being a convex compact set), then x^* is also optimal for the linear program

$$\min\{(\nabla q(x^*))^T x : x \in S\}.$$

Limits of this method: Similar to the cutting plane technique in Barahona et al. [14], dense instances up to $n = 30$ and sparse instances up to $n = 100$ can be computed. Special classes of instances can be solved efficiently up to $n = 200$. These instances have off-diagonal elements in the range $[0, 100]$ and diagonal elements lying in the fixed interval $[-I, 0]$, for the case $I = 63$ (the density is 100%). For other values of I, the problem may become much more difficult to solve. However, the method fails for general dense problems with $n = 50$ variables.

3.4 A Rounding Heuristic Based on SDP

The basic SDP relaxation (**MCSDP**) can be used to obtain a feasible solution of the Max-Cut problem, i.e. to generate a cut. This method is called the Goemans-Williamson hyperplane rounding technique [39] and works as follows. Let $X = (x_{ij})$ be the optimal solution of (**MCSDP**). We have to find vectors $v_1, \ldots, v_n$, $v_i \in \mathbb{R}^k$ (for some $k \leq n$), such that $x_{ij} = v_i^T v_j$. This can be done, by computing the Cholesky Factorization $V^T V$ of X, with $V \in \mathbb{R}^{k \times n}$. Some random vector r is then used to set

$$S := \{i \colon v_i^T r \geq 0\}$$

and obtain in this way a cut $\delta(S)$. This process can be iterated with varying random vector r.

The cut obtained by this hyperplane rounding technique may be further improved by flipping single vertices. Also, instead of the solution matrix X of the SDP, a convex-combination of this matrix X with some cut-matrix xx^T used to find the Cholesky factorization may improve the result.

Summarizing, generating good cuts can be done iteratively in basically three steps:

1. Apply the Goemans-Williamson hyperplane rounding technique to the primal matrix X obtained from solving the (**MCSDP**). This gives a cut-vector x.

2. The cut x is locally improved by checking all possible moves of a single vertex to the opposite partition block.

3. Bring the rounding matrix towards a good cut by using a convex-combination of X and xx^T. With this new matrix go to 1. and repeat as long as one finds better cuts.

Chapter 4

SDP Relaxations of the Max-Cut Problem

In the previous chapter several properties and solution approaches of the Max-Cut problem have been investigated and we gave a brief description of a basic semidefinite relaxation. In this chapter we want to focus on models, that use Semidefinite Programming for obtaining upper bounds to this NP-complete problem.

4.1 The Basic Relaxation

The basic Max-Cut relaxation has already been derived in Section 3.3.2 as follows:

$$\begin{array}{lll} \textbf{(MCSDP)} & \max & \langle L, X\rangle \\ & \text{s.t.} & \operatorname{diag}(X) = e \\ & & X \in \mathcal{S}^n,\ X \succeq 0 \end{array} \tag{4.1}$$

and its dual form

$$\begin{array}{lll} \textbf{(MCDSDP)} & \min & e^T u \\ & \text{s.t.} & \operatorname{Diag}(u) - L \succeq 0. \end{array} \tag{4.2}$$

We denote the feasible set of **(MCSDP)** as

$$\mathcal{E}_n := \{X \in \mathcal{S}_n \colon \operatorname{diag}(X) = e,\ X \succeq 0\}, \tag{4.3}$$

called the *elliptope*. A study of this convex set can be found in Laurent and Poljak [78], [79].

As already mentioned in Section 2.1, for graphs with non-negative edge weights, the optimal solution of **(MCSDP)** is at most 14% above the value of the maximum cut [39].

Several strategies have been applied for strengthening this SDP-based relaxation. We will explore these methods in the subsequent sections.

4.2 Strengthening the Basic Relaxation

In Section 3.3.1 we introduced the odd-cycle inequalities, and as a special case of it, the triangle inequalities, to strengthen the LP relaxation of the Max-Cut problem.

Similar to the LP case, also the SDP bound can be improved by exploiting the observation, that in any cycle of length three, exactly zero or two edges are cut. Considering matrix $X = (x_{ij})$ representing a cut, the following inequalities must be valid for all $1 \leq i < j < k \leq n$:

$$\begin{array}{rcl} x_{ij} + x_{ik} + x_{jk} & \geq & -1 \\ x_{ij} - x_{ik} - x_{jk} & \geq & -1 \\ -x_{ij} + x_{ik} - x_{jk} & \geq & -1 \\ -x_{ij} - x_{ik} + x_{jk} & \geq & -1 \end{array} \tag{4.4}$$

The polytope containing all matrices $X \in \mathcal{S}_n$ with $\operatorname{diag}(X) = e$ and satisfying inequalities (4.4), is called the *metric polytope* and denoted by MET.

$$\begin{aligned} \text{MET} := \{X \in \mathcal{S}_n : \operatorname{diag}(X) = e,\ & x_{ij} + x_{ik} + x_{jk} \geq -1, \\ & x_{ij} - x_{ik} - x_{jk} \geq -1,\ -x_{ij} + x_{ik} - x_{jk} \geq -1, \\ & -x_{ij} - x_{ik} + x_{jk} \geq -1\}. \end{aligned} \tag{4.5}$$

This leads to the following relaxation, proposed in Poljak and Rendl [107]:

$$\begin{array}{lll} \textbf{(SDPMET)}' & \max & \langle L, X \rangle \\ & \text{s.t.} & X \in \text{MET} \\ & & X \succeq 0 \end{array} \tag{4.6}$$

The number of inequalities of **(SDPMET)** is growing rapidly with increasing dimension n. Including all these triangle inequalities and then solving the program by an Interior-Point Method (see Section 1.4.1) is intractable already for small n. Computational results of solving this SDP with successively including the $4\binom{n}{3}$ triangle inequalities can be found in Helmberg et al. [56]. Results are also given in Rendl [111], where only a limited number of these triangle inequalities is considered.

A more general class of inequalities are the *hypermetric inequalities*, studied in Deza and Laurent [31]. Let b be an integer vector with $\sum_{i=1}^{n} b_i$ is odd. This guarantees that

$$|x^T b| \geq 1 \text{ for all } x \in \{\pm 1\}^n.$$

The following equivalences always hold:

$$|x^T b| \geq 1 \Leftrightarrow (x^T b)(x^T b) \geq 1 \Leftrightarrow \langle xx^T, bb^T \rangle \geq 1.$$

And therefore the hypermetric inequalities must be valid for all matrices in the cut polytope.

The triangle inequalities can be derived as a special case of the hypermetric inequalities by setting for the triangle formed by the vertices i, j, k:

$$b_i = b_j = b_k = 1, b_l = 0, \ \forall l \notin \{i, j, k\}$$

and

$$b_i = b_j = 1, \ b_k = -1, b_l = 0, \ \forall l \notin \{i, j, k\}.$$

Helmberg and Rendl [54] use the hypermetric inequalities as cutting planes and solve the SDP by an Interior-Point Code. At the initial step of the algorithm they consider the basic semidefinite relaxation (4.1). Inequalities are added while solving the relaxation (i.e. after some Newton steps), as well as after the exact solution to the relaxation has been obtained. Then the optimization process is restarted again. Later on, Helmberg [50] improved this algorithm by fixing variables.

This algorithm has been used in a Branch & Bound framework. In Helmberg and Rendl [54] several branching rules are considered and discussed carefully. Although the relaxation produced very tight bounds, the results of the Branch & Bound code remained below the expectations of the authors. The number of nodes in the Branch & Bound tree is very small, but the computation time per node may be rather large. Most graphs up to $n = 50$ vertices can be solved in the root-node of the Branch & Bound tree. Instances up to the size $n = 100$ can still be solved, but the computational effort may be very high. Graphs with more than 100 vertices are intractable for this algorithm.

4.3 Lift-and-Project Methods

Since the nineties several approaches have been developed to construct relaxations to NP-hard problems by representing the polytope over which we want to optimize as the projection of another polytope lying in a higher dimensional space. They can be classified into the BCC method due to Balas, Ceria, and Cornuéjols [8], the SA method by Sherali and Adams [121], the LS method of Lovász and Schrijver [89], and the method of Lasserre [76]. Details and relations about these lift-and-project methods can be found in the papers of Laurent [77] or Laurent and Rendl [80].

4.3.1 The Lifting of Anjos and Wolkowicz

Anjos and Wolkowicz [3] introduced an SDP relaxation for Max-Cut via a second lifting. They obtain the relaxation by adding redundant constraints to **(MCSDP)** and then use Lagrangian duality for deriving the dual of the dual. After a second lifting they end up with a relaxation called **(SDP3)**, which is the

following:

$$\begin{array}{lll} \textbf{(SDP3)} & \max & \langle H_L, Z\rangle \\ & \text{s.t.} & \operatorname{diag}(Z) = e \\ & & Z_{0,t(i)} = 1 \ (i \in \{1,\dots,n\}) \\ & & Z_{0,T(i,j)} = Z_{T(i,k),T(k,j)} \ (\forall k,\ 1 \le i < j \le n) \\ & & Z \in \mathcal{S}_{t(n)+1},\ Z \succeq 0. \end{array} \tag{4.7}$$

Here, $t(i) = i(i+1)/2$ and

$$T(i,j) := \begin{cases} t(j-1)+i & \text{if } i \le j \\ t(i-1)+j & \text{otherwise.} \end{cases}$$

The matrix in the objective is

$$H_L = \begin{pmatrix} 0 & \frac{1}{2}\operatorname{dsvec}(L)^T \\ \frac{1}{2}\operatorname{dsvec}(L) & 0 \end{pmatrix}$$

where dsvec is the operator that forms a $t(n)$-vector columnwise from an $n \times n$ symmetric matrix while ignoring the strictly lower triangular part and multiply the off-diagonal entries by two.

Another way to derive this relaxation is as follows. Let $v \in \{\pm 1\}^n$ be a vector representing a cut and $z \in \mathbb{R}^{t(n)+1}$, indexed by $\{\emptyset\} \cup V(K_n) \cup \widetilde{E}(K_n)$, K_n being the complete graph on n vertices, and $\widetilde{E}(K_n)$ denoting the set of all subsets of $V(K_n)$ of cardinality two (thus, representing all edges in the complete graph). Define

$$\begin{array}{l} z_\emptyset := 1 \\ z_{\{i\}} := v_i v_i,\ i \in \{1,\dots,n\} \\ z_{\{i,j\}} := v_i v_j,\ 1 \le i < j \le n \end{array} \tag{4.8}$$

Thus,

$$z = \begin{pmatrix} 1 \\ v_1^2 \\ \vdots \\ v_n^2 \\ v_1 v_2 \\ \vdots \\ v_i v_j \\ \vdots \\ v_{n-1,n} \end{pmatrix},\ z \in \{\pm 1\}^{t(n)+1}$$

and zz^T yields the following matrix:

$$Z = zz^T = \left(\begin{array}{c|c|c} Z_{\emptyset,\emptyset} & Z_{\emptyset,\{i\}} & Z_{\emptyset,\{k,l\}} \\ \hline Z_{\{i\},\emptyset} & Z_{\{i\},\{i\}} & Z_{\{i\},\{k,l\}} \\ \hline Z_{\{i,j\},\emptyset} & Z_{\{i,j\},\{i\}} & Z_{\{i,j\},\{k,l\}} \\ & & \end{array}\right).$$

If this matrix corresponds to a cut, through the equalities

$$Z_{\{i,k\},\{k,j\}} = z_{\{i,k\}} z_{\{k,j\}} = v_i v_k v_k v_j = v_i v_k^2 v_j = v_i v_j = Z_{\emptyset,\{ij\}},$$

which hold for all k and $1 \leq i < j \leq n$, we obtain the following set of *triangle equalities*:

$$\begin{array}{rcll} Z_{\{i,j\},\{j,k\}} & = & Z_{\emptyset,\{i,k\}} & \\ Z_{\{i,k\},\{j,k\}} & = & Z_{\emptyset,\{i,j\}} & 1 \leq i < j < k \leq n \\ Z_{\{i,j\},\{i,k\}} & = & Z_{\emptyset,\{j,k\}} & \end{array} \tag{4.9}$$

Also, clearly all the elements in the main diagonal have to be one, since $(v_i^2)^2 = 1$ and $(v_i v_j)^2 = 1$. And via $Z_{\emptyset,\{i\}} = v_i^2 = 1$ we obtain (**SDP3**), using a slightly different way of indexing matrix Z (with rows/columns permuted in H_L and Z).

$$\begin{array}{lll} \textbf{(SDP3)} & \max & \langle H_L, Z \rangle \\ & \text{s.t.} & \operatorname{diag}(Z) = e \\ & & Z_{\emptyset,\{i\}} = 1 \ (i \in \{1,\ldots,n\}) \\ & & Z_{\emptyset,\{i,j\}} = Z_{\{i,k\},\{k,j\}} \ (\forall k,\ 1 \leq i < j \leq n) \\ & & Z \in \mathcal{S}_{t(n)+1},\ Z \succeq 0. \end{array} \tag{4.10}$$

The leading principal minor of any matrix satisfying the constraints of (**SDP3**) is $\det \binom{1\,1}{1\,1} = 0$ and therefore every feasible Z is singular (see Lemma A.2). Hence, (**SDP3**) has no strictly feasible points. Anjos and Wolkowicz [3] show that matrix $Z \in S_{t(n)+1}$ can be projected on the lower dimensional space of dimension $t(n-1)+1$ without loosing sparsity of the constraints. We want to give an alternative and maybe more intuitive proof below. Before, we make the following

Observation 4.1 *Let $Z \in \mathcal{S}_{t(n)+1}$ be feasible for* (**SDP3**) *and let Z be indexed by $\{\emptyset\} \cup V(K_n) \cup \widetilde{E}(K_n)$. Then Z is always of the form*

$$Z = \begin{pmatrix} 1 & e^T & s^T \\ e & ee^T & es^T \\ s & se^T & S \end{pmatrix},$$

where $e \in \mathbb{R}^n$ is the vector of all ones, s is a vector of length $t(n-1)$ and $S \in \mathcal{S}_{t(n-1)}$.

Proof: Consider any Z from the feasible set of (**SDP3**), with the following block structure:

$$Z = \begin{pmatrix} 1 & u^T & s^T \\ u & U & R^T \\ s & R & S \end{pmatrix},$$

with $u \in \mathbb{R}^{V(K_n)}$, $s \in \mathbb{R}^{\widetilde{E}(K_n)}$. In order to prove the observation, we have to show that

$$u = e,\ U = ee^T,\ R = se^T.$$

Because of the constraint $Z_{0,\{i\}} = 1$, we have $u = e$.

Then the leading 3×3 principal submatrix of Z reads

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & u_{12} \\ 1 & u_{12} & 1 \end{pmatrix}.$$

Since every principal minor of a positive semidefinite matrix has to be non-negative (see Theorem A.1) we get

$$u_{12} = 1.$$

Similarly we can derive, that any other u_{ij} has to be equal to one if Z is positive semidefinite, and thus we obtain $U = ee^T$.

In the same way we get, for the principal submatrix formed by the rows/columns $\{\emptyset, \{k\}, \{i,j\}\}$

$$\begin{pmatrix} 1 & 1 & s_{\{i,j\}} \\ 1 & 1 & r_{\{k\},\{i,j\}} \\ s_{\{i,j\}} & r_{\{k\},\{i,j\}} & 1 \end{pmatrix}$$

that $r_{\{k\},\{i,j\}} = s_{\{i,j\}}$ must hold and therefore we end up with $R = se^T$ and we have proved the observation. □

Theorem 4.2 *Let $Y \in S_{t(n-1)+1}$ be indexed by $\{\emptyset\} \cup \widetilde{E}(K_n)$, $\widetilde{E}(K_n)$ being the collection of all the subsets of $V(K_n)$ of cardinality two. ($V(K_n)$ is the set of vertices of the complete graph on n vertices.) Let Q_L be defined as*

$$Q_L := \begin{pmatrix} \frac{1}{2}\sum_{i=1}^{n} l_{ii} & \mathrm{usvec}(L)^T \\ \mathrm{usvec}(L) & 0 \end{pmatrix},$$

where usvec(L) *forms a vector columnwise from a symmetric matrix considering only the strictly upper triangular part of the matrix. Consider the following SDP.*

$$\begin{array}{lll} (\mathbf{SDP3_P}) & \max & \langle Q_L, Y \rangle \\ & s.t. & \mathrm{diag}(Y) = e \\ & & Y_{\emptyset,\{i,j\}} = Y_{\{i,k\},\{k,j\}}\ (\forall k \notin \{i,j\},\ 1 \le i < j \le n) \\ & & Y \in \mathcal{S}_{t(n-1)+1},\ Y \succeq 0. \end{array} \tag{4.11}$$

There is a bijection between the feasible sets of (**SDP3**) *and* (**SDP3$_P$**).

Proof: Let $\mathcal{F} \subseteq \mathcal{S}_{t(n)+1}$ be the set of matrices feasible for (**SDP3**) and $\mathcal{F}_P \subseteq \mathcal{S}_{t(n-1)+1}$ the set of matrices feasible for $(\mathbf{SDP3})_P$. Define $f\colon \mathcal{F} \to \mathcal{F}_P$, as

$$f(Z) = Z_{\{\emptyset\}\cup\widetilde{E}(K_n)},$$

meaning that $f(Z)$ is the principal submatrix of Z, that contains only the rows and columns indexed by $\{\emptyset\}\cup\widetilde{E}(K_n)$, hence we simply erase the columns indexed by $V(K_n)$. (Note, that it is easy to verify that every Y in the image of f is feasible for $(\mathbf{SDP3_P})$.)

Due to Observation 4.1, every matrix $Y \in \mathcal{F}_P$ can be extended to a $Z \in \mathcal{F}$ in a unique way. The fact that it is possible to extend Y to a feasible Z implies that f is surjective, and because of the fact, that this way of extending is unique, f is injective. Thus, f is a bijection between $\mathcal{F}$ and $\mathcal{F}_P$. □

Anjos and Wolkowicz [3] prove that relaxation $(\mathbf{SDP3_P})$ lies in the elliptope and satisfies all the triangle inequalities. Thus, the bound of this relaxation is at least as good as the bound obtained by solving (**SDPMET**), which is already a strengthening of (**MCSDP**) (see Section 4.2). In fact they prove,

$$\text{CUT} \subseteq \mathcal{F} \subseteq \mathcal{E} \cap \text{MET}$$

and show that these inclusions are strict for $n \geq 5$.

They also present numerical results for graphs up to $n = 12$ vertices and observe, that the $(\mathbf{SDP3_P})$ relaxation often yields the optimal value of (**MC**). Nevertheless, since the dimension of the matrix is $\binom{n}{2} + 1$ and the SDP has $1 + \binom{n}{2} + 3\binom{n}{3}$ linear equality constraints, it is out of reach to solve it for graphs of size $n = 50$.

4.3.2 The Lifting of Lasserre

Consider the general problem

$$\begin{array}{ll} \max & c^T x \\ \text{s.t.} & Ax \leq b \\ & x \in \{0,1\}^n \end{array} \tag{4.12}$$

and denote

$$K := \{x \in [0,1]^n \colon Ax \leq b\}, \quad P := \text{conv}(\{x \in \{0,1\}^n \colon Ax \leq b\}).$$

Lasserre [76] constructs a hierarchy of semidefinite relaxations $\mathbb{Q}_i$ of P with the property

$$K \supseteq \mathbb{Q}_0(K) \supseteq \mathbb{Q}_1(K) \supseteq \cdots \supseteq \mathbb{Q}_{n-1}(K) \supseteq \mathbb{Q}_n(K) = P.$$

The motivation for these relaxations comes from results about moment matrices. A detailed study and the proof of convergence can be found in Lasserre [76] or Laurent [77].

Applied to the Max-Cut problem, relaxation $(\mathbb{Q}_0)$ coincides with the basic SDP relaxation (**MCSDP**). The relaxation $(\mathbb{Q}_1)$ is the following:

$$\begin{array}{lll} (\mathbb{Q}_1) & \max & \langle M_L, Y\rangle \\ & \text{s.t.} & \operatorname{diag}(Y) = e \\ & & Y_{\emptyset,\{i,j\}} = Y_{\{i,k\},\{k,j\}} \quad (\forall k \notin \{i,j\}, 1 \le i < j \le n) \\ & & Y_{\{i,j\},\{h,k\}} = Y_{\{i,h\},\{j,k\}} = Y_{\{i,k\},\{j,h\}} \\ & & \qquad \text{(for all distinct } i,j,h,k \in \{1,\ldots,n\}) \\ & & Y \in \mathcal{S}^{\binom{n}{2}+1}_+,\ Y \succeq 0. \end{array} \tag{4.13}$$

Clearly, this is a strengthening of $(\mathbf{SDP3_P})$, introduced in the previous section. The constraints additional to $(\mathbf{SDP3_P})$, we call them *4-cycle equalities*, can also be motivated as follows. Consider $y \in \mathbb{R}^{\binom{n}{2}+1}$, with

$$\begin{array}{l} y_\emptyset := 1 \\ y_{\{i,j\}} := v_i v_j,\ 1 \le i < j \le n \end{array} \tag{4.14}$$

with $v \in \{\pm 1\}^n$, and Y constructed as $Y = yy^T$. Then, for all distinct $i,j,h,k \in \{1,\ldots,n\}$, we obtain

$$Y_{\{i,j\},\{h,k\}} = y_{\{i,j\}}y_{\{h,k\}} = v_i v_j v_h v_k = y_{\{i,h\}}y_{\{j,k\}} = Y_{\{i,h\},\{j,k\}},$$

and

$$Y_{\{i,j\},\{h,k\}} = y_{\{i,j\}}y_{\{h,k\}} = v_i v_j v_h v_k = y_{\{i,k\}}y_{\{j,h\}} = Y_{\{i,k\},\{j,h\}}.$$

Similar to $(\mathbf{SDP3_P})$, for various graphs, the optimal cut value can be found via $(\mathbb{Q}_1)$. Nevertheless, the number of constraints is $1 + \binom{n}{2} + 3\binom{n}{3} + 2\binom{n}{4}$ and thus the SDP is intractable to be solved already for graphs on a few vertices.

4.4 Between the Basic Relaxation and a First Lifting

Optimizing over the Anjos-Wolkowicz or the Lasserre relaxation, introduced in the previous sections, amounts to solving an SDP having a matrix variable of order $\binom{n}{2}+1$ and in case of Anjos-Wolkowicz we have to deal with $1+\binom{n}{2}+3\binom{n}{3}$ linear equalities, in the case of Lasserre with $1+\binom{n}{2}+3\binom{n}{3}+2\binom{n}{4}$ linear equalities.

In Table 4.1 some numbers are listed, giving the sizes of the SDPs to be solved. In the first column the number of vertices is given. The second column shows the number of edges of the complete graph, i.e. the dimension of the matrix variable minus one. In the third column the numbers of constraints (additional

n	$\binom{n}{2}$	$3\binom{n}{3}$	$2\binom{n}{4}$
10	45	360	420
20	190	3420	9690
30	435	12180	54810
50	1225	58800	460600
100	4950	485100	7842450
200	19900	3940200	129369900

Table 4.1: Problem sizes, complete graphs

vertices	edges	$3k$	l
40	412	3689	23020
100	581	3126	9054
200	496	891	689
216	857	1926	3831
343	1365	3069	6117
512	2041	4590	9159

Table 4.2: Problem sizes, sparse graphs

to the all-ones diagonal constraints) for Anjos-Wolkowicz are listed (i.e. three times the number of triangles) and column four shows the number of constraints to be added to the Anjos-Wolkowicz relaxation when solving the SDP proposed by Lasserre.

Because of the huge number of constraints, the relaxation of Lasserre is intractable, already for graphs on a few vertices. The Anjos-Wolkowicz relaxation can be solved for small instances, but both the dimension of the matrix variable and the number of constraints increase very fast and thus, the problem of obtaining a bound for a graph on 50 vertices by the Anjos-Wolkowicz model cannot be solved anymore.

However, in Table 4.2 the numbers of vertices, edges, triangle-constraints and 4-cycle constraints of some sparse graphs are given (k is the number of triangles, l is the number of 4-cycles). Apparently, if we would consider instead of any pair of nodes, only the support of the graph, i.e. edges e with $w_e \neq 0$, the matrix-variable and the number of constraints would be of reasonable size, even if the graph consists of a few hundred vertices.

4.4.1 Exploiting Sparsity

The idea is to formulate an SDP considering not the complete graph, but indexing the matrix variable only with those pairs of vertices, that are linked by an edge with non-zero weight, i.e. the edges in the set $E(G)$. Let y be a vector, indexed by the edge set and zero, defined by

$$\begin{aligned} y_\emptyset &:= 1 \\ y_{\{i,j\}} &:= v_i v_j,\ [ij] \in E(G), i < j \end{aligned}$$

Thus,

$$y = \begin{pmatrix} 1 \\ v_1v_2 \\ \vdots \\ v_iv_j \\ \vdots \end{pmatrix}, \; y \in \{\pm 1\}^{m+1}$$

and yy^T yields the matrix:

$$Y = yy^T = \left(\begin{array}{c|c} Y_{\emptyset,\emptyset} & Y_{\emptyset,\{k,l\}} \\ \hline Y_{\{i,j\},\emptyset} & Y_{\{i,j\},\{k,l\}} \end{array}\right).$$

If this matrix corresponds to a cut, the following equalities must hold:

$$Y_{\{i,k\},\{k,j\}} = Y_{\emptyset,\{i,j\}} \; \forall \text{ triangles } (i,j,k) \in G \tag{4.15}$$

$$Y_{\{i,j\},\{k,l\}} = Y_{\{i,l\},\{j,k\}} \; \forall \; 4-\text{cycles } (i,j,k,l) \in G \tag{4.16}$$

$$Y_{\{i,j\},\{j,k\}} = Y_{\{i,l\},\{k,l\}} \; \forall \; 4-\text{cycles } (i,j,k,l) \text{ if } [ik] \notin E(G) \tag{4.17}$$

Equalities (4.15) and (4.16) have already been proved in Section 4.3.1 to be valid for matrices Y arising from a valid cut vector v. Condition (4.17) can be justified as follows. If $[ik] \in E(G)$ then we would have the triangle-constraints: $Y_{\{i,j\},\{j,k\}} = Y_{\emptyset,\{i,k\}}$ and $Y_{\{i,l\},\{k,l\}} = Y_{\emptyset,\{i,k\}}$, thus $Y_{\{i,j\},\{j,k\}} = Y_{\{i,l\},\{k,l\}}$. Since this edge, and therefore the constraints do not exist, we can add this condition $Y_{\{i,j\},\{j,k\}} = Y_{\{i,l\},\{k,l\}}$.

Let matrix $Q_L \in \mathcal{S}_{m+1}$, $m = |E(G)|$, be defined as

$$\begin{aligned} (Q_L)_{\emptyset,\emptyset} &:= \frac{1}{4}\sum_{i=1}^{n} l_{ii} \\ (Q_L)_{\emptyset,\{i,j\}} = (Q_L)_{\{i,j\},\emptyset} &:= \frac{1}{4}l_{ij} \quad \forall [ij] \in E(G), \end{aligned}$$

where $L = (l_{ij})$ is the Laplace matrix of the graph. Hence,

$$Q_L = \frac{1}{4}\left(\begin{array}{c|ccc} \sum l_{ii} & l_{i_1j_1} & l_{i_2j_2} & \cdots \\ \hline l_{i_1j_1} & 0 & 0 & \cdots \\ l_{i_2j_2} & 0 & 0 & \cdots \\ \vdots & \vdots & & \end{array}\right).$$

Note, that if Y represents a cut $\delta(S)$, then

$$Y_{\emptyset,\{ij\}} = Y_{\{ij\},\emptyset} = \begin{cases} -1 & \text{if } [ij] \in \delta(S) \\ 1 & \text{otherwise.} \end{cases}$$

With Q_L defined in this way and if Y represents cut $\delta(S)$, $\langle Q_L, Y\rangle$ yields the value of this cut:

$$\begin{aligned}
\langle Q_L, Y\rangle &= Y_{\emptyset,\emptyset}(Q_L)_{\emptyset,\emptyset} + 2\sum_{[ij]\in E(G)} Y_{\emptyset,\{i,j\}}(Q_L)_{\emptyset,\{i,j\}} = \\
&= \frac{1}{4}(\sum_{i=1}^{n} l_{ii} + 2\sum_{[ij]\in E(G)} Y_{\emptyset,\{i,j\}} l_{ij}) = \\
&= \frac{1}{4}(2\sum_{[ij]\in E(G)} a_{ij} + \\
&\qquad 2\sum_{[ij]\notin\delta(S)} 1\cdot(-a_{ij}) + 2\sum_{[ij]\in\delta(S)} (-1)\cdot(-a_{ij})) = \\
&= w(\delta(S)).
\end{aligned}$$

Thus, the following SDP relaxation can be obtained:

$$\begin{array}{lll}
\textbf{(MCSPARSE)} & \max & \langle Q_L, Y\rangle \\
& \text{s.t.} & \operatorname{diag}(Y) = e \\
& & \mathcal{A}(Y) = b \\
& & Y \in \mathcal{S}_{m+1},\ Y \succeq 0
\end{array} \tag{4.18}$$

where $m = |E(G)|$ and $\mathcal{A}(Y) = b$ denotes the set of triangle- and possibly the 4-cycle-equalities.

For any graph that contains a star (i.e. a vertex that is adjacent to each other vertex), relaxation (**MCSPARSE**) has the important property, that the bound is at least as good as the bound obtained by solving the basic SDP relaxation. We prove this in the following

Theorem 4.3 *Suppose there exists a vertex $v \in V(G)$ with $\deg(v) = n-1$. Then the upper bound obtained from* (**MCSPARSE**) *is at least as good as the* (**MCSDP**)*-bound.*

Proof: Let Y be the optimal solution of relaxation (**MCSPARSE**) and $v \in V(G) : deg(v) = n-1$. Define $X \in \mathcal{S}_n$, $X = (x_{ij})$ to be the matrix

$$x_{ij} := \begin{cases} Y_{\{i,v\},\{j,v\}} & \text{if } i \neq v, j \neq v \\ Y_{\{i,v\},\emptyset} & \text{if } j = v, i \neq j \\ Y_{\emptyset,\emptyset} & \text{if } i = j = v \end{cases},$$

for all $i, j \in V(G)$. I.e. X is the principal submatrix of Y consisting of the rows/columns indexed by $\{\{1,v\},\{2,v\},\ldots,\{v-1,v\},\emptyset,\{v,v+1\},\ldots,\{v,n\}\}$. Since Y is a solution of (**MCSPARSE**), $\operatorname{diag}(Y) = e$ and hence, $\operatorname{diag}(X) = e$. And because of the positive semidefiniteness of Y, X is a positive semidefinite matrix (due to Theorem A.1 and the fact that X is a principal submatrix of Y).

Also, $\frac{1}{4}\langle L, X\rangle = \langle Q_L, Y\rangle$. Thus, X is feasible for (**MCSDP**) and the optimal solution of (**MCSDP**) is at least $\langle Q_L, Y\rangle$. □

This observation motivates us to add zero-cost edges to the given graph, in order to obtain at least one star (if the graph does not already contain one).

When considering complete graphs, the relaxation considering the triangle-equalities complies with the relaxation introduced by Anjos and Wolkowicz. But if the graph is sparse, the matrix variable is of order $m+1$ in comparison to $\binom{n}{2}+1$ and we have $1 + m + 3k$ linear equalities, where k is the number of triangles.

Also, considering all three types of constraints is a weaker model than the one proposed by Lasserre, but contrary to the Lasserre relaxation it is manageable for graphs on a few hundred vertices.

Solving this relaxation is discussed in Chapter 5 below.

4.4.2 Systematically Chosen Submatrices

Gvozdenović and Laurent [44] constructed a hierarchy of semidefinite approximations for the chromatic number $\chi(G)$ of a graph G (see Section 2.2). Similar to the hierarchy of, for instance, Lasserre, also their second order bound $\psi^{(2)}(G)$ is not practically computable, since the dimension of the SDP is too large. Nevertheless, they present computational results, where they use a variation of their second order bound $\psi^{(2)}(G)$. Instead of working with the matrix that is indexed by $\{\emptyset\} \cup V(K_n) \cup \widetilde{E}(K_n)$, $n = |V(G)|$, they consider only the index set $\{\emptyset\} \cup V(K_n) \cup \{\{h, i\} : i \in V(K_n)\}$, $h \in V(K_n)$ and thus have a computationally practicable strengthening of $\psi^{(1)}(G)$.

Using this idea, we can get tractable relaxations also for the Max-Cut problem. Consider the relaxation of Anjos and Wolkowicz or Lasserre, but instead of using the full matrix of dimension $\binom{n}{2} + 1$, we work with the matrix considering the index set

$$\{\emptyset\} \bigcup_{i=1}^{k} \{\{v_i, j\} : j \in V(G) \backslash \{v_i\}\},$$

$v_i \in V(G)$ $\forall i \in \{1, \ldots, k\}$. That means, we choose an integer k, $1 \leq k \leq n$, and include edges, such that the graph consists of k stars. The dimension of this matrix is $(n-1) + (n-2) + \cdots + (n-k) + 1 = kn - k(k+1)/2 + 1$.

It is easy to see, that as coefficient matrix in the objective function one can choose a matrix having the Laplace matrix of the underlying graph as a principal submatrix, namely in the rows/columns associated to one of the vertices with degree $n-1$ and the $\emptyset$-row/column.

Due to Theorem 4.3, this bound is then at least as good as the basic relaxation (**MCSDP**), already for $k = 1$. Of course, the choice of k affects the quality of the bound, but also the computational effort. Therefore, choosing the 'right' k might be a crucial task.

		(MCSPARSE) with constraints				
	(4.1)	(4.15)	(4.15),(4.16)	(4.15),(4.17)	(4.15), (4.16),(4.17)	*best cut*
tri40_20	109.3	**106.9**	**106.4**	**106.3**	**106.1**	*106*
tri50_20	170.6	167.7	166.4	165.5	**164.8**	*164*
tri_rnd40_20	304.8	289.9	**286.0**	**286.0**	**286.0**	*286*
tri_rnd50_20	339.5	312.8	305.0	305.0	305.0	*303*
grid 10, 10	142.0	135.2	133.0	133.5	**132.0**	*132*
grid 15, 15	318.7	303.8	299.1	299.7	**297.0**	*297*
grid 20, 20	574.0	546.1	540.1	541.1	**539.0**	*539*
grid 4, 4, 4	105.0	100.4	98.3	98.8	**96.1**	*96*
grid 6, 6, 4	233.2	225.2	221.3	222.1	**216.0**	*216*
grid 8, 8, 4	422.6	407.2	401.5	402.6	**388.0**	*388*
grid 8, 8, 5	520.9	503.8	497.8	498.5	**479.2**	*479*
spin5	125.3	124.2	123.8	123.7	110.8	*108*
spin6	211.8	210.2	220.0	210.4	188.7	*182*
gls	144.6	141.6	152.5	140.0	135.1	*126*

Table 4.3: Comparing bounds for (**MC**) obtained by solving (**MCSDP**), (4.1), and (**MCSPARSE**), including different types of constraints. Bold numbers indicate, that we have proved optimality of the best found cut. (Note that the weights on the edges of all graphs are integers.)

4.4.3 Numerical Results of (MCSPARSE)

In this section we present a few numerical results of solving relaxation (**MCSPARSE**) and compare it with the solution of (**MCSDP**).

Table 4.3 lists the various bounds. The SDPs have been solved using SeDuMi [122] or the Bundle Method (Section 5.1.1), depending on the size of the problem.

In the first column, the bound obtained by solving the basic relaxation (**MCSDP**) is given. The second column shows the bound when solving the sparse model and considering all the triangle-equalities. In column 3, additionally the 4-cycle constraints are included. Column 4 shows the results where the triangles and also the constraints referring to (4.17) are incorporated. And in the fifth column all constraints are taken into consideration. In the last column the best found cut is given.

As can be seen, even with the smallest set of constraints, the sparse model is always significantly better than the (**MCSDP**)-bound. Additional constraints improved the bound and considering all equations proved that the best cut found was optimal for most of the instances.

We annotate that using SeDuMi we compute the exact solution of the SDP, whereas the Bundle Method returns an upper bound. For the spin6 and gls

graphs, the relaxation with constraints (4.15),(4.16) (which is intractable to be solved by SeDuMi) was hard to solve for the Bundle Method. The progress was not satisfactory and therefore we end up with a bound that is rather far from the optimum of the SDP. Nevertheless, the Bundle Method for solving the relaxation using constraints (4.15),(4.16),(4.17) shows a more satisfying behavior and the bounds for graphs spin6 and g1s are improved significantly.

Chapter 5

Algorithms for Solving Large-Scale Semidefinite Programs

In previous chapters we investigated various Semidefinite Programs for solving or approximating NP-hard problems arising from Combinatorial Optimization. This chapter is dedicated to methods for solving these SDPs. Two prominent solution methods have already been investigated in Chapter 1, namely the Interior-Point Method and the Spectral Bundle Method. Although these two methods are widely used, for various kinds of relaxations the dimension of the matrix variable or the number of constraints is simply too large to be solved by one of these algorithms.

This means that for various NP-hard problems promising Semidefinite Programming relaxations exist, but due to the vast computational effort or the huge memory requirements, the SDPs are not solvable.

Another way to tackle the problems of solving these SDPs, is using Lagrangian duality and then solving the underlying problem by Bundle Methods, a tool for optimizing non-differentiable functions (see Schramm and Zowe [118], Kiwiel [70], Lemarechal [82]). We will focus on these methods in this chapter and will elaborate the details for solving the relaxations introduced in the previous chapter. Furthermore, we will come back to the Spectral Bundle Method, introduced in Section 1.4.2 and establish second order models to improve the performance. Finally, the Boundary Point Method is introduced in this chapter. This new algorithm uses an augmented Lagrangian algorithm applied to the dual SDP. At the end of the chapter, a brief guidance for choosing the proper algorithm to solve a given SDP is presented.

5.1 The Bundle Method in Combinatorial Optimization

Barahona and Anbil [11] and Barahona and Ladányi [12] use the volume algorithm for solving LP relaxations arising from combinatorial optimization problems. This algorithm allows solving LPs with a large number of constraints, because they are not handled directly, but are incorporated by using the Lagrangian dual. Bahiense, Maculan, and Sagastizábal [7]) show that this algorithm can be viewed as a simple variant of the Bundle Method.

Bundle Methods were first proposed by Lemarechal [82], and later on further investigated by Kiwiel [69], Lemaréchal, Nemirovskii, and Nesterov [83]. A recent survey can be found in the paper of Mäkelä [91]. A detailed presentation can be found in the textbook of Hiriart-Urruty and Lemaréchal [60].

We want to apply this method to solve semidefinite programs with a large number of constraints, for example those introduced in Sections 4.2 and 4.4.

The ingredients of the Bundle Method are basically the following:

- For a convex objective function f and a point $\hat{\gamma} \in \mathbb{R}^m$ we are able to compute
  - the objective value $f(\hat{\gamma})$
  - a subgradient $\bar{g} \in \partial f(\hat{\gamma})$, i.e. $\bar{g}$ satisfying

$$f(\gamma) \geq f(\hat{\gamma}) + \langle \bar{g}, \gamma - \hat{\gamma} \rangle, \ \forall \gamma \in \mathbb{R}^m.$$

- This information is used to construct a cutting plane model $\hat{f}$ of f, minorizing f on $\mathbb{R}^m$:

$$\hat{f}_k(\gamma) := \max_{1 \leq i \leq k} f(\hat{\gamma}_i) + \langle g_i, \gamma - \hat{\gamma}_i \rangle.$$

- Since the quality of minorant $\hat{f}$ is reasonable only in the vicinity of the current iterate, displacement is penalized by considering

$$f_k(\gamma) := \hat{f}_k(\gamma) + \frac{u}{2} \|\gamma - \hat{\gamma}_k\|^2,$$

  where $u \in \mathbb{R}$ is the *penalty parameter.*

The *bundle* (of size k) can be seen as the set of triples

$$\{\gamma_i \in \mathbb{R}^m, f_i := f(\gamma_i), g_i \in \partial f(\gamma_i), \ i = 1, \ldots, k\}.$$

We sketch the generic framework as follows.

Algorithm 5.1 (Generic Bundle Algorithm)
Input.
Function f and $\hat{\gamma} \in \mathbb{R}^m$.
Start.
Evaluate $f(\hat{\gamma})$ and obtain subgradient g.
Construct cutting plane model $\hat{f}$ of f, minorizing f on $\mathbb{R}^m$.
while *some stopping condition is not satisfied*
Solve $\min_\gamma \hat{f}(\gamma) + \frac{u}{2}\|\gamma - \hat{\gamma}\|^2$ to get γ_{test}.
Evaluate f at γ_{test} and obtain subgradient g_{test}.
if *reasonable progress is made*
Do a serious step,
update bundle, giving updated $\hat{f}$.
else
Do a null step,
update bundle, giving updated $\hat{f}$.
Check stopping condition.
end

In the two subsequent sections we want to elaborate the details for applying the Bundle Method to solve SDP relaxations of the Max-Cut problem.

5.1.1 Solving (MCSPARSE) Using the Bundle Method

We want to solve the Max-Cut relaxation introduced in Section 4.4, namely

$$\begin{array}{lll} \textbf{(MCSPARSE)} & \max & \langle Q_L, Y\rangle \\ & \text{s.t.} & \operatorname{diag}(Y) = e \\ & & \mathcal{A}(Y) = b \\ & & Y \in \mathcal{S}_{m+1},\ Y \succeq 0 \end{array} \tag{5.1}$$

where $m = |E(G)|$ and $\mathcal{A}(Y) = b$ denotes the set of triangle- and possibly the 4-cycle-equalities. Let l be the number of these constraints, i.e. $b \in \mathbb{R}^l$.

Recall that the set of positive semidefinite matrices with an all-ones diagonal is the elliptope, denoted by $\mathcal{E}$. Thus, (5.1) can equivalently be written as

$$\begin{array}{lll} \textbf{(MCSPARSE)} & \max & \langle Q_L, Y\rangle \\ & \text{s.t.} & \mathcal{A}(Y) = b \\ & & Y \in \mathcal{E}. \end{array} \tag{5.2}$$

The goal is to optimize the dual functional to **(MCSPARSE)**. Let us introduce the Lagrangian

$$\mathcal{L}(Y, \gamma) := \langle Q_L, Y\rangle + \gamma^T(b - \mathcal{A}(Y)) \tag{5.3}$$

and the dual functional

$$f(\gamma) := \max_{Y \in \mathcal{E}} \mathcal{L}(Y, \gamma) = b^T\gamma + \max_{Y \in \mathcal{E}} \langle Q_L - \mathcal{A}^T(\gamma), Y\rangle. \tag{5.4}$$

Evaluating f for some γ amounts to solving a problem of type (4.1), which can be done easily for problem sizes of our interest by Interior-Point Methods (see Section 1.4.1).

We call a pair (γ, Y) a **matching pair** for f, if $f(\gamma) = \mathcal{L}(Y, \gamma)$. A subgradient is given by $g(\gamma) = b - \mathcal{A}(Y)$. By applying Corollary A.12 we obtain the Lagrangian dual

$$\max_{Y \in \mathcal{E}} \min_{\gamma} \mathcal{L}(Y, \gamma) = \min_{\gamma} \max_{Y \in \mathcal{E}} \mathcal{L}(Y, \gamma) = \min_{\gamma} f(\gamma).$$

Following the ideas of the bundle concept, we build up a bundle of matrices to construct the minorant

$$f_{appr}(\gamma) := \max\{\mathcal{L}(Y, \gamma) : Y \in \operatorname{conv}(Y_1, \ldots, Y_k)\}.$$

Obviously, $Y \in \operatorname{conv}(Y_1, \ldots, Y_k)$ can be expressed as $Y = \lambda_1 Y_1 + \cdots + \lambda_k Y_k$, $e^T \lambda = 1, \lambda \geq 0$.

In each iteration we have some current iterate $\hat{\gamma}$ and we have to solve

$$\min_{\gamma} f_{appr}(\gamma) + \frac{1}{2t} \|\gamma - \hat{\gamma}\|^2, \tag{5.5}$$

which is convex quadratic in γ, and then we find Y (i.e. solving an SDP of dimension n with n linear constraints), such that (γ, Y) is a matching pair.

Some notation. Let us collect $Y_1, \ldots, Y_k$ symbolically in $\mathcal{Y}$ and for a vector $\lambda \in \mathbb{R}^k$ define $\mathcal{Y}\lambda := \sum_{i=1}^k \lambda_i Y_i$. Furthermore we define $\Lambda := \{\lambda \in \mathbb{R}^k : e^T \lambda = 1, \lambda \geq 0\}$. $G \in \mathbb{R}^{l \times k}$ is the collection of the k subgradients, i.e. $G := (g_1, \ldots, g_k)$ and $F \in \mathbb{R}^k$ is the vector of the primal function values of the matrices in $\mathcal{Y}$, i.e. $F_i := \langle Q_L, Y_i \rangle$, $1 \leq i \leq k$.

With this notation we rewrite

$$\begin{aligned} f_{appr}(\gamma) &= \max_{\lambda \in \Lambda} b^T \gamma + \langle Q_L - \mathcal{A}^T(\gamma), \mathcal{Y}\lambda \rangle = \\ &= \max_{\lambda \in \Lambda} \langle b - \mathcal{A}(\mathcal{Y}\lambda), \gamma \rangle + \langle Q_L, \mathcal{Y}\lambda \rangle = \\ &= \max_{\lambda \in \Lambda} \gamma^T G \lambda + F^T \lambda. \end{aligned}$$

Therefore, (5.5), the problem to be solved, reads

$$\begin{aligned} &\min_{\gamma} \max_{\lambda \in \Lambda} \gamma^T G \lambda + F^T \lambda + \frac{1}{2t} \|\gamma - \hat{\gamma}\|^2 = \\ &= \max_{\lambda \in \Lambda} \min_{\gamma} \gamma^T G \lambda + F^T \lambda + \frac{1}{2t} \|\gamma - \hat{\gamma}\|^2. \end{aligned} \tag{5.6}$$

The inner problem in (5.6) is an unconstrained convex quadratic problem in γ and therefore we can replace the minimization by insuring $\frac{\partial}{\partial \gamma}(.) = 0$:

$$\begin{aligned} \frac{\partial}{\partial \gamma}\left(\gamma^T G \lambda + F^T \lambda + \frac{1}{2t} \|\gamma - \hat{\gamma}\|^2\right) = 0 \quad &\Leftrightarrow \quad G\lambda + \frac{1}{t}(\gamma - \hat{\gamma}) = 0 \\ &\Leftrightarrow \quad \gamma = \hat{\gamma} - tG\lambda. \end{aligned} \tag{5.7}$$

Substituting (5.7) in (5.6) we obtain

$$\begin{aligned}
&\max_{\lambda\in\Lambda} \gamma^T G\lambda + F^T\lambda + \frac{1}{2t}\|\gamma-\hat{\gamma}\|^2 = \\
&= \max_{\lambda\in\Lambda} (\hat{\gamma} - tG\lambda)^T G\lambda + F^T\lambda + \frac{1}{2t}\|\hat{\gamma} - tG\lambda - \hat{\gamma}\|^2 = \\
&= \max_{\lambda\in\Lambda} \hat{\gamma}^T G\lambda - t\|G\lambda\|^2 + F^T\lambda + \frac{t}{2}\|G\lambda\|^2 = \\
&= \max_{\lambda\in\Lambda} (G^T\hat{\gamma} + F)^T\lambda - \frac{t}{2}\|G\lambda\|^2. \qquad (5.8)
\end{aligned}$$

This convex quadratic problem over Λ can be solved efficiently by an Interior-Point Method.

We can summarize now the algorithm to solve problem (5.1).

Algorithm 5.2 (Bundle Method to solve (MCSPARSE))
Input.
$Q_L, \mathcal{A}, b$ defining the problem.
Starting point $\hat{\gamma} \in \mathbb{R}^m$ (optional).
Parameter t, α, ε.
Start.
i=1, done=false.
Evaluate f at $\hat{\gamma}$:
solve $\max \langle Q_L - \mathcal{A}^T(\hat{\gamma}), Y\rangle$ *s.t.* $Y \in \mathcal{E}$ *giving* $\hat{Y}$,
$f(\hat{\gamma}) = b^T\hat{\gamma} + \langle Q_L - \mathcal{A}^T(\hat{\gamma}), \hat{Y}\rangle$,
subgradient $G = g(\hat{\gamma}) = b - \mathcal{A}(\hat{Y})$.
repeat
Solve (5.8) and obtain λ.
$\gamma_{test} = \hat{\gamma} - tG\lambda$.
$f_{appr}(\gamma_{test}) = (F + G^T\gamma_{test})^T\lambda$.
if $i < iter_{max}$ *and* $f(\hat{\gamma}) - f_{appr}(\gamma_{test}) > \varepsilon$
Evaluate f at γ_{test}:
solve $\max \langle Q_L - \mathcal{A}^T(\gamma), Y\rangle$ *s.t.* $Y \in \mathcal{E}$ *giving* Y_{test},
$f_{test} = b^T\gamma_{test} + \langle Q_L - \mathcal{A}^T(\gamma_{test}), Y_{test}\rangle$,
subgradient $g_{test} = b - \mathcal{A}(Y_{test})$.
if $f_{test} \le \alpha f_{appr}(\gamma_{test}) + (1-\alpha) f(\hat{\gamma})$
Serious step:
$\hat{\gamma} = \gamma_{test}$.
Increase t.
else
Null step:
Decrease t.
Purge bundle (*).
Append Y_{test} to $\mathcal{Y}$, g_{test} to G, $\langle Q_L, Y_{test}\rangle$ to F.

i=i+1.
else
done=true.
until *done.*

ad $(*)$: In order to keep the size of the bundle reasonably small, we *purge* it in each iteration, meaning that we eliminate points Y_i from $\mathcal{Y}$ (and corresponding F_i and the i-th column of G) whose contribution to the minorant f_{appr} is negligibly small (i.e. $\lambda_i < \rho \cdot \max_{1 \le j \le k} \lambda_j$, for small $\rho > 0$).

Computational effort. Solving problem (5.8) can be done efficiently and also the matrix and vector manipulations are of minor computational effort. The most expensive task during one iteration is the function evaluation, i.e. solving an SDP of dimension m with the m equality constraints fixing the main diagonal to all ones. This SDP has to be solved exactly once in each iteration. Therefore, one has to limit the number of iterations in order to get reasonable computation times.

Quality of the solution obtained by the Bundle Method. Since the Bundle Method returns only a "nearly" optimal solution, we want to evaluate the quality of this solution. We run the following experiment. Instances, which are solvable by Interior-Point Methods, as well as by the Bundle Method, are chosen and solved using SeDuMi (Sturm [122]) and using our Bundle code, fixing the number of iterations to 100. (In detail, we set the parameters as follows: $\varepsilon = 10^{-8}$, $\alpha = 0.2$, estimate for t: $t = 2(f(\gamma_{start}) - best)/(G^T G)$, where *best* is some best known lower bound; if serious step: $t = 1.01t$, if null step: $t = t/1.02$.) We summarize the results in Table 5.1. It turns out, that the bundle solution is always less than 1% above the solution of the Interior-Point Method and therefore it is definitely arguable to talk about a "nearly" optimal solution.

5.1.2 Solving (SDPMET) Using the Bundle Method

Fischer, Gruber, Rendl, and Sotirov [35] developed a dynamic version of the Bundle Method to solve problem

$$\begin{array}{lll} \textbf{(SDPMET)} & \max & \langle L, X \rangle \\ & \text{s.t.} & \mathcal{A}(X) \le b \\ & & X \in \mathcal{E}, \end{array} \tag{5.9}$$

where $\mathcal{A}(X) \le b$ denotes the set of triangle inequalities (see Section 4.2).

Contrary to solving **(MCSPARSE)**, explained in Section 5.1.1, we have to deal with inequalities, instead of equalities, which requires further examination.

	$\|E\|$	number of constraints	solution obtained by SeDuMi	bundle	$100\frac{bdl-SeDuMi}{SeDuMi}$
rnd40_1	412	5102	241.8254	242.6178	0.33
rnd40_2	411	4933	247.0309	247.2647	0.09
rnd40_3	413	5136	243.7674	244.2194	0.19
grid2D_1	437	1152	123.5164	124.3625	0.69
grid2D_2	595	1784	170.7619	171.2725	0.30
grid2D_3	595	1784	151.0140	151.6911	0.45
planar_1	581	3708	356.6162	357.1655	0.15
planar_2	581	3708	112.4779	113.3540	0.78
rnd400_1	1193	3594	720.7212	721.4541	0.10
rnd400_2	1188	3583	719.1141	719.7037	0.08
planar_3	609	4156	364.5891	365.5026	0.25
planar_4	605	4125	362.9813	363.6936	0.20
planar_5	602	3954	361.0765	361.7070	0.17
spin2pm_1	446	1188	134.6885	135.3837	0.52
spin2pm_2	595	1784	148.7254	148.9564	0.16
spin2pm_3	595	1784	155.3412	155.8469	0.33
spin3pm_1	655	2024	195.0179	195.4285	0.21
spin3pm_2	793	2576	197.9891	198.2795	0.15
spin3pm_3	793	2576	198.8500	199.1416	0.15
spin6	216	2784	210.7897	211.1507	0.17

Table 5.1: Solution of the IPM vs. Bundle Method. The dimension of the matrix variable is $|E| + 1$, the number of constraints is given in column three. Columns four and five give the solutions obtained by the two methods. The last column indicates how far the bundle-solution is from the optimal solution (in %).

We now have to solve the Lagrangian dual

$$\min_{\gamma \geq 0} f(\gamma),$$

with f defined in (5.4). Dualizing also the constraints $\gamma \geq 0$, we obtain

$$\max_{\lambda \in \Lambda, \eta \geq 0} \min_{\gamma} \ \gamma^T G\lambda + F^T\lambda + \frac{1}{2t}\|\gamma - \hat{\gamma}\|^2 - \gamma^T\eta \tag{5.10}$$

where in addition to (5.6) we have η, the Lagrange multipliers to the constraints $\gamma \geq 0$. As before, the inner minimization can be replaced by setting the derivative with respect to γ to zero yielding

$$\gamma = \hat{\gamma} - t(\eta - G\lambda). \tag{5.11}$$

Plugging this into (5.10) we obtain

$$\max_{\lambda \in \Lambda, \eta \geq 0} (G^T\hat{\gamma} + F)^T\lambda - \frac{t}{2}\|\eta - G\lambda\|^2 - \hat{\gamma}^T\eta. \tag{5.12}$$

This is a convex quadratic optimization problem in λ and η. Solving it directly is too time consuming and thus, we solve it approximately by alternately keeping one set of variables constant. Keeping η constant leads to a convex quadratic problem similar to (5.8). Keeping λ fixed (say, $\bar{\lambda}$), we obtain

$$\begin{aligned} &\max_{\eta \geq 0} (G^T\hat{\gamma} + F)^T\bar{\lambda} - \frac{t}{2}\|\eta - G\bar{\lambda}\|^2 - \hat{\gamma}^T\eta = \\ &= \max_{\eta \geq 0} -\frac{t}{2}\|\eta\|^2 + (tG\bar{\lambda} - \hat{\gamma})\eta + \text{const.} \end{aligned}$$

The maximizer $\bar{\eta}$ to this problem can be determined coordinate-wise:

$$\bar{\eta}_i = \begin{cases} \frac{1}{t}(t(G\bar{\lambda})_i - \hat{\gamma}_i) & \text{if } t(G\bar{\lambda})_i - \hat{\gamma}_i \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

(see for instance Helmberg, Kiwiel, and Rendl [57]). This process of keeping one set of variables constant is iterated several times, and we end up with an approximate solution of (5.12).

A further difference to the algorithm described in Section 5.1.1 is, that the function $f(\gamma)$ to be minimized is not considered to be fixed, but changes in the course of the algorithm. This is due to the fact, that we do not want to consider all the triangle-inequalities at the same time, since the number of them is of order $O(n^3)$ (n is the dimension of X).

As initial function we solve

$$\max\{\langle L, X\rangle \colon X \in \mathcal{E}\}$$

yielding an initial maximizer X^*. We could now define $f(\gamma)$ by dualizing all the constraints $\mathcal{A}(X) \leq b$, but to maintain efficiency, we are interested only in those constraints, which are likely to be active at the optimum. Therefore we look at the violation $r^* := b - \mathcal{A}(X^*)$ and set $r_{\min} := \min\{r_i^*\}$. In the unlikely event that $r_{\min} \geq 0$, X^* is optimal for (**SDPMET**) and we are done. Otherwise $r_{\min} < 0$ and we consider now only those constraints from $\mathcal{A}(X) \leq b$ which are 'badly' violated by X^* to define f.

Specifically, let $0 < \alpha < 1$ and set

$$I := \{i : r_i^* \leq \alpha\, r_{\min}\}.$$

We denote by $\mathcal{A}_I$ the operator that contains only those matrices A_i, which correspond to the triangle-inequalities in I. Similarly, we define b_I. For notational convenience we denote

$$f_I(\gamma_I) := \max_{X \in \mathcal{E}} \langle L, X \rangle + (b_I - \mathcal{A}_I(X))^T \gamma_I.$$

Thus we start by minimizing f_I using the Bundle Method described above. After a preassigned number of bundle iterations, some $\hat{\gamma}_I$ and some $\hat{X}$ are obtained. It is likely that $\hat{\gamma}_I$ is still far from the true minimizer of f_I. Before continuing with further iterations, we will update I, the index set of the triangle-inequalities under consideration, by removing those which seem to be not important and adding those, that promise progress. Thus, we need to answer the following two questions:

- Which of the constraints in I should be kept?
- Are there additional constraints that should be added to I?

To answer the first question, we use $\hat{\gamma}_I$. A large value $\hat{\gamma}_i$ indicates that the constraint i is binding and hence should not be removed. $\hat{\gamma}_i = 0$ indicates that constraint i may be inactive, and therefore could be removed. In summary, we use $\hat{\gamma}_I$ to purge I by removing constraints i with a value $\hat{\gamma}_i$ smaller than a prescribed fraction of $\max(\hat{\gamma}_I)$.

It is less obvious to decide which new constraints should be added to I. Experiments showed that using a convex combination of $\hat{X}$ and the previous point $\hat{X}_{old}$ to identify violated constraints gives the most satisfactory behaviour of the algorithm.

Numerical results for using this method to solve (**SDPMET**) can be found in [35, Table 4]. The algorithm reduces the gap of the basic SDP relaxation (4.1) by 20% up to 90% for medium sized graphs ($n \leq 512$), and by at least 10% for larger graphs ($729 \leq n \leq 2000$). No other method so far is capable of producing similar or even better bounds.

5.2 The Spectral Bundle Method with Second Order Information

What follows in this section is based on joint work with Armbruster, Overton and Rendl [4]. We will use within this section definitions and formulas from Section 1.3 (Eigenvalue Optimization) and Section 1.4.2 (The Spectral Bundle Method).

Although the Spectral Bundle Method is capable of solving quite large-scaled Semidefinite Programs, convergence is very slow. Within the method, the regularized model (1.17)

$$\min_{y\in\mathbb{R}^k}\{\hat{f}(y)+\frac{u}{2}\|y-\hat{y}\|^2\}$$

is optimized. Since the choice of the parameter u is "somewhat of an art" (see Remark 1.14), the idea is to replace uI by making partial use of second order information and thus solve the following problem:

$$\min_{y\in\mathbb{R}^k}\{\hat{f}(y)+\frac{1}{2}(y-\hat{y})^T\hat{H}(y-\hat{y})\},$$

where $\hat{H}$ is a suitable approximation of the Hessian matrix of $\hat{f}$. With this choice we hope to get a better performance of the algorithm.

To work out the details, what a "suitable approximation" of the Hessian is, recall definition (1.16)

$$\hat{f}(y):=\max_{W\in\mathcal{W}}\langle C-\mathcal{A}^T(y),W\rangle+b^Ty. \tag{5.13}$$

Here we define

$$\mathcal{W}:=\{PVP^T\colon \mathrm{tr}V=1, V\in\mathcal{S}_k^+\},$$

with integer k being significantly smaller than n. Thus (1.18), the problem to be solved in the course of the algorithm, becomes

$$\max_{\mathrm{tr}V=1,V\succeq 0}\langle C-\mathcal{A}^T(\hat{y}),PVP^T\rangle+b^T\hat{y}-\frac{1}{2u}\|\mathcal{A}(PVP^T)-b\|^2 \tag{5.14}$$

and (1.19) reads

$$y=\hat{y}+\frac{1}{u}(\mathcal{A}(PVP^T)-b). \tag{5.15}$$

In order to derive a second order model, let us explore the derivatives of the function in question. Let

$$C-\mathcal{A}^T(y)=Q\Lambda Q^T$$

be the spectral decomposition, where $Q=(q_1,q_2,\ldots,q_n)$ is an $n\times n$ matrix, $Q^TQ=I$, $\lambda_1>\lambda_2\geq\cdots\geq\lambda_n$, $\Lambda=\mathrm{Diag}(\lambda_1,\ldots,\lambda_n)$.

Assuming that the largest eigenvalue is simple, function $\lambda_{\max}(C - \mathcal{A}^T(y))$ is smooth and we obtain

$$\frac{\partial}{\partial y_i}\lambda_{\max}(C - \mathcal{A}^T(y)) = -q_1^T A_i q_1, \tag{5.16}$$

$$\frac{\partial^2}{\partial y_i \partial y_j}\lambda_{\max}(C - \mathcal{A}^T(y)) = 2q_1 A_i Q_2(\lambda_1 I_{n-1} - D_2)^{-1} Q_2^T A_j q_1, \tag{5.17}$$

where $Q_2 = (q_2, \ldots, q_n)$, $D_2 = \mathrm{Diag}(\lambda_2, \ldots, \lambda_n)$, see for instance Lancaster [75]. In general, the largest eigenvalue will have multiplicity $k > 1$. In this case, $\lambda_{\max}(C - \mathcal{A}^T(y))$ may not be written as a differentiable function in general.

Overton and Womersley [101] develop a second order model to minimize function $\lambda_{\max}(A(x))$ with $A(x)$ being a sufficiently smooth symmetric matrix. Our approach is closely related to this even more general work.

Suppose the largest eigenvalue $\lambda_{\max}(C - \mathcal{A}^T(y))$ has multiplicity k, i.e.

$$\lambda_1 = \cdots = \lambda_k > \lambda_{k+1} \geq \cdots \geq \lambda_n$$

and partition Q in Q_1 and Q_2, such that

$$Q_1 = (q_1, \ldots, q_k),\ Q_2 = (q_{k+1}, \ldots, q_n).$$

For finding a suitable approximation of the Hessian, we first need a matrix $W = PVP^T$. Recall from (1.10) that condition $\mathcal{A}(PVP^T) = b$ must hold for optimality. Now Q_1 plays the role of P in Section 1.3, and to avoid confusion let us use U instead of V. Therefore, one way to find a proper U is to solve the problem

$$\begin{array}{ll} \min & \|b - \mathcal{A}(Q_1 U Q_1^T)\|^2 \\ \text{s.t.} & \mathrm{tr}U = 1 \\ & U \succeq 0 \end{array} \tag{5.18}$$

and then define $W := Q_1 U Q_1^T$. In this way we obtain a subgradient $g(y) = b - \mathcal{A}(W) \in \partial f(y)$ of minimum norm. If $\mathcal{A}(W) = b$, then $0 \in \partial f(y)$ holds and hence, we have a certificate for optimality.

We use matrix U to define the following matrix $H(U) = (h_{ij})$,

$$h_{ij} = 2\mathrm{tr}(A_i(Q_1 U Q_1^T) A_j Q_2(\lambda_1 I_{n-k} - D_2)^{-1} Q_2^T)$$

with $D_2 = \mathrm{Diag}(\lambda_{k+1}, \ldots, \lambda_n)$. Matrix $H(U)$ captures the second order behaviour of $f(y)$ around the current point $\hat{y}$ with k being the multiplicity of the largest eigenvalue of $C - \mathcal{A}^T(\hat{y})$, see Helmberg and Oustry [53] or Overton and Womersley [101]. In the case of a simple largest eigenvalue, $H(U)$ indeed coincides with the Hessian of $f(y)$.

By using $H(U)$ instead of uI in (5.15) we get

$$y = \hat{y} + H(U)^{-1}(\mathcal{A}(W) - b). \tag{5.19}$$

Plugging this and $H(U)$ into (5.14), the program to be solved, we obtain

$$\begin{aligned}
&\max_{\substack{W=Q_1VQ_1^T \\ \mathrm{tr}V=1, V\succeq 0}} \langle C-\mathcal{A}^T(\hat{y}), W\rangle + b^T\hat{y} - \frac{1}{2}(y-\hat{y})^T H(U)(y-\hat{y}) = \\
&= \max_{\substack{W=Q_1VQ_1^T \\ \mathrm{tr}V=1, V\succeq 0}} \langle C-\mathcal{A}^T(\hat{y}+H(U)^{-1}(\mathcal{A}(W)-b)), W\rangle + \\
&\qquad +b^T(\hat{y}+H(U)^{-1}(\mathcal{A}(W)-b)) + \\
&\qquad +\frac{1}{2}(-H(U)^{-1}(\mathcal{A}(W)-b))^T H(U)(-H(U)^{-1}(\mathcal{A}(W)-b)) = \\
&= \max_{\substack{W=Q_1VQ_1^T \\ \mathrm{tr}V=1, V\succeq 0}} \langle C-\mathcal{A}^T(\hat{y})+\mathcal{A}^T(H(U)^{-1}b), W\rangle + \\
&\qquad -\frac{1}{2}\langle \mathcal{A}^T(H(U)^{-1}\mathcal{A}(W)), W\rangle + b^T\hat{y} - \frac{1}{2}b^T H(U)^{-1}b.
\end{aligned}$$

By defining

$$\begin{aligned}
C_1(W) &:= \mathcal{A}^T(H(U)^{-1}\mathcal{A}(W)) \\
C_2 &:= C-\mathcal{A}^T(\hat{y})+\mathcal{A}^T(H(U)^{-1}b) \\
c_3 &:= b^T\hat{y} - \frac{1}{2}b^T H(U)^{-1}b
\end{aligned}$$

we end up with the quadratic problem

$$\max\{-\langle C_1(W), W\rangle + \langle C_2, W\rangle + c_3 \colon W = Q_1VQ_1^T, \mathrm{tr}V = 1, V\succeq 0\}. \qquad (5.20)$$

We can now outline the algorithm as follows.

Algorithm 5.3 (Spectral Bundle Method with Second Order Information)
Input.
$C, \mathcal{A}, b$ defining the problem.
Starting point $\hat{y}\in\mathbb{R}^m$ (optional).
Parameter $iter_{\max}$ and ε.
Start.
Evaluate f at $\hat{y}$:
Compute spectral decomposition $C-\mathcal{A}^T(y) = Q\Lambda Q^T$,
$f(\hat{y}) = \lambda_1 + b^T\hat{y}$.
Estimate k, the multiplicity of λ_{max}.
$Q_1 = (q_1, \ldots, q_k)$
repeat
Solve (5.18) and obtain U.
Compute $H(U)$.
Solve (5.20) and obtain W.
Compute direction $d = H(U)^{-1}(\mathcal{A}(W)-b)$ (see (5.19)).
Do a line search and obtain t in $y = \hat{y}+td$.
$\hat{y} \leftarrow \hat{y}+td$.

Evaluate f at the new point $\hat{y}$ and obtain new Q.
Estimate k, the multiplicity of $\lambda_{\max}$.
$Q_1 = (q_1, \ldots, q_k)$.
until $iter_{\max}$ *reached or* $\|d\| < \varepsilon$.

Four items of this algorithm need further comments:

- Eigenvalue computation.
- Variation of $H(U)$ in order to gain computational efficiency.
- Guessing the multiplicity of $\lambda_{\max}$.
- Line search.

Eigenvalue computation. The eigenvalue computation can be done efficiently by iterative methods, see Parlett [104]. There exist several implementations of these methods, for instance, the *Arnoldi method* [5] can be found in the ARPACK package [81].

Variation of $H(U)$. Computing matrix $H(U)$ and the inverse of it requires substantial computational effort. A way to replace these operations by much cheaper ones, is to use only the main diagonal of $H(U)$ and ignore the off-diagonal entries. In this way we still utilize second order information, but require only cheap matrix operations and have therefore a computationally practical algorithm.

Multiplicity of $\lambda_{\max}$. The multiplicity of the largest eigenvalue is usually not known in advance and therefore we have to cleverly guess this value. We compute the differences of the normalized eigenvalues

$$\Delta\lambda_i := \frac{\lambda_i - \lambda_{i+1}}{\lambda_1}, \ 1 \leq i \leq n-1.$$

If $\Delta\lambda_1$ is a value larger than, say 0.01, we can assume the eigenvalue λ_1 to be simple. Otherwise, the multiplicity is k, if there is either a relatively to $\Delta\lambda_1$ large gap $\Delta\lambda_k$, or if the difference between λ_1 and λ_k is considerably big, i.e.

$$k = \min\{\{i\colon \Delta\lambda_i > 4\Delta\lambda_1\} \cup \{i\colon \lambda_1 - \lambda_i > \delta\}\},$$

with δ being for instance 0.05.

Line search. Lancaster [75] shows that for any matrix $C(t)$ whose elements are functions of a scalar parameter t, the function $\lambda(C(t))$ has continuous second derivatives with respect to t. We can use this in the line search, by computing the Taylor expansion up to order two of $f(\hat{y}+td)$ and therefore, have a cheap way of computing the approximate function value. I.e. if we know $\dot{\lambda}_{\max}(t)$ and $\ddot{\lambda}_{\max}(t)$, we can compute

$$\begin{aligned} f(\hat{y}+td) &= \lambda_{\max}(C-\mathcal{A}^T(\hat{y}+td))+b^T(\hat{y}+td)= \\ &= \lambda_{\max}((C-\mathcal{A}^T(\hat{y}))+t\mathcal{A}^T(d))+b^T\hat{y}+t(b^Td)\approx \\ &\approx \lambda_{\max}(C-\mathcal{A}^T(\hat{y}))+b^T\hat{y}+t((\dot{\lambda}_{\max}(t))+b^Td)+\frac{t^2}{2}\ddot{\lambda}_{\max}(t). \end{aligned}$$

Note, that eigenvalue $\lambda_{\max}(C-\mathcal{A}^T(\hat{y}))$ is already known in the current iteration.

To work out the details what $\dot{\lambda}_{\max}(t)$ and $\ddot{\lambda}_{\max}(t)$ is, consider

$$\lambda_i(C+tD),$$

with C and D being symmetric $n\times n$ matrices, and λ_i denotes the i-th eigenvalue of the matrix in question. We also assume that eigenvalues are sorted in non-increasing order, i.e. $\lambda_1\geq\cdots\geq\lambda_n$. (Applied to our function, C corresponds to $C-\mathcal{A}^T(\hat{y})$ and D to $\mathcal{A}^T(d)$.)

We rewrite the relevant theorems, applied to our problem. Note, that the theorems concerning existence of the eigenvalues (following from the theory of implicit functions and the work of Goursat [40]) and differentiability can also be found in the paper of Lancaster [75].

Theorem 5.4 *Lancaster [75, Theorem 9]*
Assume that all eigenvalues λ_i, $1\leq i\leq n$, of $B(t_0)=C+t_0D$, with spectral decomposition $B(t_0)=X\Lambda X^T$, $X^TX=XX^T=I$, are distinct. Then for an eigenvalue λ_i of $B(t_0)$ we have

$$\dot{\lambda}_i(t_0)=p_{ii}, \tag{5.21}$$

$$\ddot{\lambda}_i(t_0)=2\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{p_{ij}^2}{\lambda_i-\lambda_j}, \tag{5.22}$$

where

$$P=X^TDX. \tag{5.23}$$

Proof: First, by differentiating $X^TX=I$ we get $\dot{X}^TX+X^T\dot{X}=0$ and define the skew-symmetric matrix

$$S:=\dot{X}^TX=-X^T\dot{X}. \tag{5.24}$$

Now, we differentiate $\Lambda = X^T B X$ and by defining $P = X^T D X$ and (5.24) we obtain

$$\begin{aligned}\dot{\Lambda}(t_0) &= \dot{X}^T B X + X^T D X + X^T B \dot{X} = \\ &= X^T D X + \dot{X}^T X X^T B X + X^T B X X^T \dot{X} = \\ &= P + S\Lambda - \Lambda S.\end{aligned}$$

For the diagonal elements we obtain

$$\dot{\lambda}_i(t_0) = p_{ii} + s_{ii}\lambda_i - \lambda_i s_{ii} = p_{ii}. \tag{5.25}$$

The off-diagonal elements have to be zero and thus,

$$\begin{aligned}\dot{\lambda}_{ij}(t_0) = 0 \quad &\Rightarrow \quad p_{ij} + s_{ij}\lambda_j - \lambda_i s_{ij} = 0 \\ &\Rightarrow \quad s_{ij} = \frac{p_{ij}}{\lambda_i - \lambda_j} \quad \text{for } i \neq j.\end{aligned} \tag{5.26}$$

And since S is skew-symmetric we have

$$s_{ii} = 0. \tag{5.27}$$

To compute $\ddot{\lambda}_i(t_0)$ we differentiate P.

$$\begin{aligned}\dot{P} &= \dot{X}^T D X + X^T D \dot{X} = \\ &= \dot{X}^T X X^T D X + X^T D X X^T \dot{X} = \\ &= (\dot{X}^T X)(X^T D X) - (X^T D X)(\dot{X}^T X) = \\ &= SP - PS.\end{aligned}$$

And since P is symmetric and S skew-symmetric, we obtain

$$\dot{p}_{ii} = \sum_{j=1}^{n} s_{ij} p_{ji} - \sum_{j=1}^{n} p_{ij} s_{ji} = 2 \sum_{j=1}^{n} s_{ij} p_{ji}.$$

Combining this with (5.26) and (5.27), we finally get

$$\ddot{\lambda}_i(t_0) = \dot{p}_{ii} = 2 \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{p_{ij}^2}{\lambda_i - \lambda_j}.$$

□

Formula (5.22) will not hold in the presence of multiple eigenvalues. In this case the following theorem applies.

Theorem 5.5 *Lancaster [75, Theorem 10]*
Assume that the largest eigenvalue λ_1 of $B(t_0) = C + t_0 D$ has multiplicity k, i.e. $\lambda_1 = \cdots = \lambda_k > \lambda_{k+1} \geq \cdots \geq \lambda_n$. The spectral decomposition of $B(t_0)$ is given by $B(t_0) = X \Lambda X^T$, $X^T X = X X^T = I$, with $X = [X_k \; X_{n-k}]$, $X_k \in \mathbb{R}^{n \times k}$, $X_{n-k} \in \mathbb{R}^{n \times (n-k)}$. Furthermore, $X_k^T D X_k$ has eigenvalues $\psi_1, \ldots, \psi_k$ and denote the spectral decomposition of this product by $Y \Psi Y^T$, with $\Psi = \mathrm{Diag}(\psi_1, \ldots, \psi_k)$. Then for $1 \leq i \leq k$ it follows

$$\dot{\lambda}_i(t_0) = \tilde{p}_{ii},$$
$$\ddot{\lambda}_i(t_0) = 2 \sum_{j=k+1}^{n} \frac{\tilde{p}_{ij}^2}{\lambda_i - \lambda_j},$$

where

$$\widetilde{P} = \begin{pmatrix} (X_k Y)^T D (X_k Y) & (X_k Y)^T D X_{n-k} \\ X_{n-k}^T D (X_k Y) & X_{n-k}^T D X_{n-k} \end{pmatrix} = \\ = \begin{pmatrix} \Psi & (X_k Y)^T D X_{n-k} \\ X_{n-k}^T D (X_k Y) & X_{n-k}^T D X_{n-k} \end{pmatrix}.$$

Using these two theorems, we can do several function evaluations during the line search, without explicitly computing the largest eigenvalue.

5.3 A Boundary Point Method

The work presented in this section has been done jointly with Povh and Rendl [110].

A further method for solving Semidefinite Programs apart from the one presented so far, has been proposed by Burer and Vandenbussche [27]. They apply the Augmented Lagrangian technique to the primal SDP. Malick [92] uses a similar approach for solving semidefinite least squares problems.

We will take up the idea of using the Augmented Lagrangian technique, but will contrary to [27] apply it to the dual SDP, which is given as

$$\begin{array}{lll} \textbf{(DSDP)} & \min & \langle b, y \rangle \\ & \text{s.t.} & \mathcal{A}^T(y) - C = Z \\ & & y \in \mathbb{R}^m, \; Z \in \mathcal{S}_n, \; Z \succeq 0. \end{array}$$

Throughout this section, we make the assumption that the Slater condition (Definition 1.6) is satisfied.

We introduce a Lagrange multiplier X and consider for fixed $\sigma > 0$ the augmented Lagrangian

$$\mathcal{L}_\sigma(y, Z; X) := b^T y + \langle Z + C - \mathcal{A}^T(y), X \rangle + \frac{\sigma}{2} \| Z + C - \mathcal{A}^T(y) \|^2,$$

that has to be minimized. The Augmented Lagrangian method applied to our problem can be sketched as follows (see Bertsekas [22]).

Algorithm 5.6 (Augmented Lagrangian method to solve (DSDP))
Input.
$C, \mathcal{A}, b$.
Initialization.
$k = 0$.
Select $\sigma_k > 0$, $X_k \succeq 0$.
while *some stopping condition is not satisfied*
Keep X_k *fixed and*
solve $\min_{y,Z\succeq 0} \mathcal{L}_{\sigma_k}(y, Z; X)$ *giving* y_k, Z_k.
Update X: $X_{k+1} = X_k + \sigma(Z_k + C - \mathcal{A}^T(y_k))$.
Select $\sigma_{k+1} \geq \sigma_k$.
Check the stopping condition and increase k.
return (X_k, y_k, Z_k).

To simplify notation, we define

$$\mathcal{W}(y) := \mathcal{A}^T(y) - C - \frac{1}{\sigma}X, \tag{5.28}$$

and rewrite the augmented Lagrangian as

$$\mathcal{L}_\sigma(y, Z; X) := b^T y + \frac{\sigma}{2}\|Z - \mathcal{W}(y)\|^2 - \frac{1}{2\sigma}\|X\|^2.$$

Thus, the minimization in Algorithm 5.6 reads

$$\min_{y,Z\succeq 0} b^T y + \frac{\sigma}{2}\|Z - \mathcal{W}(y)\|^2. \tag{5.29}$$

This is a convex quadratic SDP and therefore tractable. However, when looking at the optimality conditions of this SDP, one can find a cheap way of solving it by iteratively keeping one set of variables fixed.

To derive the optimality conditions of this problem, let us introduce the Lagrange multiplier V for the constraint $Z \succeq 0$:

$$\mathcal{L}(y, Z; V) = b^T y + \frac{\sigma}{2}\|Z - \mathcal{W}(y)\|^2 - \langle V, Z\rangle.$$

Computing the derivatives of $\mathcal{L}(y, Z; V)$ with respect to y and Z, we get:

$$\begin{aligned}
\nabla_y \mathcal{L}(y, Z; V) &= b - \sigma\mathcal{A}(Z - \mathcal{A}^T(y) + C + \frac{1}{\sigma}X) = \\
&= \sigma\mathcal{A}(\mathcal{A}^T(y)) - \sigma\mathcal{A}(Z + C + \frac{1}{\sigma}X) + b, \\
\nabla_Z \mathcal{L}(y, Z; V) &= \sigma(Z - \mathcal{A}^T(y) + C + \frac{1}{\sigma}X) - V = \\
&= \sigma(Z - \mathcal{W}(y)) - V,
\end{aligned}$$

and by setting these derivatives equal to zero, we obtain the following KKT conditions, necessary for optimality:

$$\begin{aligned} \nabla_y \mathcal{L}(y, Z; V) = 0 \;&\Rightarrow\; \sigma\mathcal{A}(\mathcal{A}^T(y)) = \sigma\mathcal{A}(Z + C + \frac{1}{\sigma}X) - b, \\ \nabla_Z \mathcal{L}(y, Z; V) = 0 \;&\Rightarrow\; V = \sigma(Z - \mathcal{W}(y)), \\ V \succeq 0,\; Z \succeq 0,\; &VZ = 0. \end{aligned} \tag{5.30}$$

Since the problem is convex and the Slater condition holds, these conditions are also sufficient for optimality.

We observe that keeping y (and X) constant, Z is given by the projection onto the cone of positive semidefinite matrices and thus, Z must satisfy the projection condition

$$Z = \mathcal{W}(y)_+ := \text{argmin}_{U \succeq 0} \|U - \mathcal{W}(y)\|^2. \tag{5.31}$$

It is well known, that $\mathcal{W}(y)_+$ can be computed via the eigenvalue decomposition of $\mathcal{W}(y)$, see for instance Higham [58]. This works as follows. Let $\mathcal{W}(y) = P\Lambda P^T$, $P^T P = I$, $\Lambda = \text{Diag}(\lambda_i)$, and partition P and Λ according to its non-negative and negative eigenvalues: $\Lambda = \text{Diag}(\Lambda_+, \Lambda_-)$, $P = (P_+ \; P_-)$ with $\Lambda_1 \geq 0$, $\Lambda_2 < 0$. Therefore,

$$\mathcal{W}(y) = P_+\Lambda_+P_+^T + P_-\Lambda_-P_-^T = \mathcal{W}(y)_+ + \mathcal{W}(y)_-.$$

On the other hand, keeping Z (and X) constant, the minimization problem (5.29) is unconstrained and y can be computed by solving the linear system

$$\sigma\mathcal{A}(\mathcal{A}^T(y)) = \sigma\mathcal{A}(Z + C + \frac{1}{\sigma}X) - b. \tag{5.32}$$

Furthermore, from (5.30) we get $V = \sigma(Z - \mathcal{W}(y)) = \sigma(\mathcal{W}(y)_+ - \mathcal{W}(y)) = -\sigma\mathcal{W}(y)_-$. Thus, we can reformulate conditions (5.30) as

$$\begin{aligned} \mathcal{A}(\mathcal{A}^T(y)) &= \mathcal{A}(Z + C + \frac{1}{\sigma}X) - \frac{1}{\sigma}b, \\ Z &= \mathcal{W}(y)_+, \\ V &= -\sigma\mathcal{W}(y)_-. \end{aligned}$$

Keeping one set of the variables (y or Z) constant leads to problems that are easy to solve. Thus, we carry out minimization (5.29) by iteratively alternating between solving (5.32) and computing Z by projection (5.31).

The minimization is then followed by the update on X (see Algorithm 5.6), which can with (5.28) and (5.31) be simplified to

$$\begin{aligned} X_{new} &= X + \sigma(Z + C - \mathcal{A}^T(y)) = \\ &= -\sigma(\mathcal{W}(y) - \mathcal{W}(y)_+) = \\ &= -\sigma\mathcal{W}(y)_-. \end{aligned}$$

The Lagrange multiplier V is therefore indeed identical with the primal matrix X.

Note, that throughout the algorithm, the conditions $X \succeq 0$, $Z \succeq 0$ and $ZX = 0$ hold. Therefore, we call this algorithm a boundary point method, as both, Z and X, are on the boundary of the cone of semidefinite matrices. Once we have reached feasibility regarding the primal and dual linear equations, we have an optimal solution. Thus, primal and dual feasibility will serve as the stopping criterion in the algorithm.

Summarizing, we can now state the Boundary Point Method as the following

Algorithm 5.7 (Boundary Point Method)
Input.
$C, \mathcal{A}, b$.
Initialization.
$k = 0$.
Select $\sigma > 0$, $\varepsilon > 0$, $\{\varepsilon_k\} \to 0$.
$X_k = 0$, $Z_k = 0$.
repeat
repeat
solve $\mathcal{A}(\mathcal{A}^T(y)) = \mathcal{A}(Z_k + C + \frac{1}{\sigma}X_k) - \frac{1}{\sigma}b$ *giving* y_k.
$\mathcal{W}(y_k) = \mathcal{A}^T(y_k) - C - \frac{1}{\sigma}X_k$.
$Z_k = \mathcal{W}(y_k)_+$.
$V_k = -\sigma\mathcal{W}(y_k)_-$.
$\delta_{inner} = \|\mathcal{A}(V_k) - b\|$.
until $\delta_{inner} \leq \sigma\varepsilon_k$.
$X_{k+1} = V_k$.
$\delta_{outer} = \|Z_k - \mathcal{A}^T(y_k) + C\|$.
$k = k + 1$.
until $\delta_{outer} \leq \varepsilon$.

The highest computational costs arise in computing the spectral decomposition of matrix $\mathcal{W}(y)$ and in solving the system of linear equations (5.32). Contrary to Interior-Point Methods where the system matrix changes in each iteration, in the Boundary Point Method the matrix arising from $\mathcal{A}(\mathcal{A}^T(.))$ remains the same throughout the algorithm. By computing for instance the QR-decomposition only once, at the beginning of the iterations, one can obtain y_k by performing backsolves. Moreover, for some problems the system-matrix of (5.32) is simply a diagonal matrix and therefore the computation times for solving the equations are negligibly small.

For the theoretical convergence results the reader is referred to Povh et al. [110]. Therein numerical results of computing the ϑ-number of a graph are also given. These results show that the Boundary Point Method works very well on these instances. Comparison to the most efficient codes currently available, which

problem	n	m	ϑ	[73]	[26]	[124]	our method
theta62	300	13389	29.6412515	96	344	200	50
theta82	400	23871	34.3668929	457	695	635	87
theta83	400	39861	20.3018905	1820	852	900	70
theta102	500	37466	38.3905463	1299	1231	1300	143
theta103	500	62515	22.5285698	2317	1960	2000	110
theta104	500	87244	13.3361402	11953	2105	1440	124
theta123	600	90019	24.6686519	10538	2819	3000	205
theta162	800	127599	37.0097358	13197	6004	12000	570

Table 5.2: Comparing the Boundary Point Method with the methods presented in [73, 26, 124] on some instances of the TOH collection. Computation times are given in seconds.

are those of Kocvara and Stingl [73], Burer and Monteiro [26] and Toh [124], show that we compute the ϑ-number often ten times faster than other methods (see Table 5.2). Also, we are able to solve instances that are too big to be solved by these other algorithms. Note, that the timings in columns labeled [73] and [26] are obtained on machines, that are significantly faster than ours, whereas our machine is at most twice as fast as the machine used to compute the results in column [124].

We also tried to apply this method for solving SDP relaxations of Max-Cut, for instance (**MCSPARSE**). The practical behavior of the algorithm for these instances remained below our expectations. Contrary to computing the ϑ-number, approaching the optimum is rather laborious and there is no advantage in using the Boundary Point method. The reason for this as well as tricks to work around this is subject of ongoing work and will be reported in the paper by Malick, Povh, Rendl, and Wiegele [93]. Furthermore, connections to proximal-point methods and details regarding convergence analysis will be presented therein.

5.4 A Recipe for Choosing the Proper Solution Method

In Chapter 1 and in the present chapter we introduced several algorithms for solving Semidefinite Programs. In order to choose the best suitable method, one can check in the following order, which method applies best to the SDP to be solved.

1. Use an Interior-Point Algorithm to solve your SDP (Section 1.4.1).

   This will give you the exact solution, as long as $n \leq 1000$ and $m \leq 7000$. If m gets larger, you will run out of memory.

2. If the matrix formed by $\mathcal{A}(\mathcal{A}^T(.))$ is sparse (maybe even diagonal), try using the Boundary Point Method (Section 5.3).

3. Dualize some of the constraints and use the Bundle Method (Section 5.1).

   You will obtain a "nearly" optimal and "nearly" feasible solution. If the dimension of your problem is n with $n \leq 2000$ the computation time should be reasonable. For larger n this method is not advisable, because an SDP of dimension n, and the constraints that have not been dualized, has to be solved several times during the algorithm by an Interior-Point Code.

4. Dualize all the constraints, meaning you use the Spectral Bundle Method (Section 1.4.2).

   Computations are possible even for a very large number of constraints, but convergence may be rather slow.

5. Use the Spectral Bundle Method with second order information (Section 5.2).

   Due to the Spectral Bundle Method you can manage large-scale problems, and the second order information will improve the convergence behaviour.

Chapter 6

Biq Mac – Binary Quadratic and Max-Cut Solver

This chapter is concerned with an exact solution method for the Max-Cut problem, developed jointly with Rendl and Rinaldi [115]. The remarkable bounds obtained by using the Bundle Method for solving (**SDPMET**) motivated us to use this method within a Branch & Bound framework. We call the outcome of this work *Biq Mac Solver* – Binary Quadratic and Max-Cut solver.

In this chapter we explain the generic Branch & Bound algorithm. We discuss some branching rules and address implementation issues. In the last section numerical results are presented.

Recall the notation from Section 3.1 where we stated the Max-Cut problem as

$$\begin{array}{lll} \textbf{(MC)} & \max & x^T L x \\ & \text{s.t.} & x \in \{\pm 1\}^n, \end{array} \tag{6.1}$$

where L is the Laplace matrix of the underlying graph and x represents a cut vector.

6.1 A Branch & Bound Framework for (MC)

Even though the Branch & Bound (B&B) idea is rather simple and can be found in virtually any textbook on enumerative methods for NP-hard problems, we summarize the key features of this procedure in the context of our problem.

Given an instance of (MC) through the matrix L, we need the following three subroutines to define the B&B process.

1. **Upper Bound.** The bounding routine

   $$z_{ub} = \text{upper-bound}(L)$$

   determines an upper bound z_{ub} on the optimal value z_{MC} of (MC) for cost matrix L.

2. **Heuristic.** We also need to be able to produce some (good) feasible solution x_f with value $z_f = x_f^T L x_f$ (hopefully) close to z_{MC}.

$$x_f = \text{feasible-solution}(L)$$

3. **Branching Information.** Finally we need a procedure that tells us how to subdivide the set of feasible solutions. We follow a very simple approach and subdivide by selecting two vertices i and j and consider the two sub problems

$$X_{i\sim j} := \{x \in \{\pm 1\}^n : \ x_i - x_j = 0\},$$
$$X_{i\not\sim j} := \{x \in \{\pm 1\}^n : \ x_i + x_j = 0\}.$$

In the first case, i and j are in the same partition block, in the second case they are in separate blocks.

It is well known [106] that optimizing over both sets can be done by solving Max-Cut on graphs of size $n-1$. The two vertices that are specified to be in the same subset or not are merged into one new vertex and thus, the dimension is reduced by one. For the case $i \sim j$, all solutions satisfy $x_v x_i = x_v x_j$ for all $v \in V(G)$, and for the case $i \not\sim j$, the relation $x_v x_i = -x_v x_j$ for all $v \in V(G)$ must hold. Therefore one can eliminate variable x_j and both subproblems can be represented through their respective Laplace matrices $L_{i\sim j}$ and $L_{i\not\sim j}$. To get an idea of the construction of these matrices, we exemplify the case where vertices $n-1$ and n go into the same set, i.e. edge $[n-1, n]$ is not cut. Then we have $L_{(n-1)\sim n} = (\tilde{l}_{ij})$ with

$$\tilde{l}_{ij} = \begin{cases} l_{ij} & \text{for } 1 \le i < n-1, 1 \le j < n-1 \\ l_{i,n-1} + l_{i,n} & \text{for } 1 \le i < n-1, j = n-1 \\ l_{n-1,j} + l_{n,j} & \text{for } i = n-1, 1 \le j < n-1 \\ l_{n-1,n-1} + 2l_{n-1,n} + l_{n,n} & \text{for } i = j = n-1. \end{cases}$$

Using these three ingredients, we can summarize a generic B&B approach for (MC) as follows.

Algorithm 6.1 (Generic Branch & Bound Algorithm for (MC))
Input.
$L \in \mathcal{S}_n$.
Initialization.
$z_{ub} = \infty$ *(upper bound)*
$x_{bk} = (1, \ldots, 1)^T$ *(best known (bk) cut vector)*
$z_{bk} = x_{bk}^T L x_{bk}$ *(best known solution value)*
$Q = \{(z_{ub}, L)\}$ *(problem list)*
while $Q \neq \emptyset$
Remove problem (z', L') *(with* z' *maximal) from* Q. $(*)$

Determine new upper bound and feasible solution for L':
z_{ub} = *upper-bound*(L')
x = *feasible-solution*(L'); $z = x^T L x$.
if $z > z_{bk}$
Update (z_{bk}, x_{bk}) *and*
remove all problems (z'', L'') *from* Q *with* $z'' < z_{bk}$.
if $z_{bk} < z_{ub}$
Determine $L_{i\sim j}$, $L_{i\not\sim j}$ *from branching pair* $\{i, j\}$.
Add $\{(z, L_{i\sim j}), (z, L_{i\not\sim j})\}$ *to* Q.
return x_{bk}.

At the beginning of each iteration one has to make the choice, from which node to branch next. One possibility is a depth-first search, where always the deepest node is chosen. This strategy has the advantage that the storage demand is very small since less problems have to be memorized. However, we use the strategy of selecting the node with the largest upper bound, see (∗). In this way, the upper bound of the overall problem is monotonically decreasing and therefore, when stopping after a certain number of nodes, the bound is expected to be tighter than in the case of using a depth-first search.

Finding good feasible solutions can be done routinely in our case, as any vector $x \in \{\pm 1\}^n$ is feasible. We use the Goemans-Williamson hyperplane rounding technique, in conjunction with some simple ideas for improving this result. The procedure is described in detail in Section 3.4.

The relaxation (**SDPMET**) solved by the Bundle Method, as explained in Section 5.1.2, serves as bounding routine.

And the branching strategy to determine a pair (i, j) of vertices to be split or merged will be desribed in detail in Section 6.2 below.

6.2 Branching Rules

At each node of the Branch & Bound tree, we branch by fixing the relation between two vertices ('branching on an edge'). Either the edge will be cut, or not.

Helmberg and Rendl [54] discuss carefully five different branching strategies, called R1 – R5. R1 and R2 follow the strategy, to branch first on a pair of vertices $\{i, j\}$, where the decision whether edge $[ij]$ is cut at the optimum or not seems to be obvious. Rules R3 and R4 go for the opposite policy, namely branching first, where the decision seems to be hard. In rule R5, information offered by the triangle inequalities is used.

According to the experiments in [54], we decide to use R2 and R3, and furthermore a variation of R3 in our algorithm. Moreover, we experiment with two *strong branching rules*. In detail the branching rules work as follows.

6.2.1 Easy First

A first idea is to choose two vertices for branching, where the decision whether these two vertices should be in the same set of the partition or not seems to be obvious. We choose i and j such that their rows are 'closest' to a $\{\pm 1\}$ vector, i.e. they minimize $\sum_{k=1}^{n}(1-|x_{ik}|)^2$. We may assume, that for these two very well articulated nodes, the value $|x_{ij}|$ is also very large. Setting x_{ij} opposite to its current sign should lead to a sharp drop of the optimal solution in the corresponding sub tree. Hoping that the bound also drops as fast, we will be able to cut off this sub tree quickly. We illustrate the B&B tree created by this rule in Figure 6.1. The picture nicely shows that we obtain rather a chain than a balanced tree. The decision opposite to x_{ij} can, for this instance, always be fathomed without further branching. According to [54] we call this rule R2.

6.2.2 Difficult First

Rule R3 in [54] fixes the hard decisions first. We branch by fixing the relation between vertices i and j with minimum $|x_{ij}|$. This means, we fix the most difficult decisions and hope that the quality of the bound gets better fast. The effect is, that we obtain a very balanced B&B tree. A picture of the tree obtained by this rule is given in Figure 6.2. (The input graph is the same as in Figure 6.1.)

6.2.3 A Variant of R3

As a variation of R3, we can instead of considering the complete graph, only look at edges with non-zero weights. However, this rule did not show a significantly different behavior than rule R3.

6.2.4 Strong Branching

To make use of the dual information, available from the bundle algorithm, we consider also 2 branching rules, where we forecast on some potential branching decisions. In this section we explain, how to make use of this information to decide a priori, that a node can be omitted from the Branch & Bound tree.

Join node i and node j. If we consider the decision to join nodes i and j (i.e. $x_{ij} = x_{ji} = 1$), we would have to solve the SDP

$$z_{i\sim j} = \max\{\mathrm{tr}LX : \mathrm{diag}(X) = e, \mathcal{A}(X) \leq b, x_{ij} + x_{ji} = 2, X \succeq 0\} \qquad (6.2)$$

which has the Lagrangian

$$\mathcal{L}(X;\gamma,u) = \langle L, X\rangle + \langle \gamma, b - \mathcal{A}(X)\rangle + \langle u, \langle E_{ij} + E_{ji}, X\rangle - 2\rangle.$$

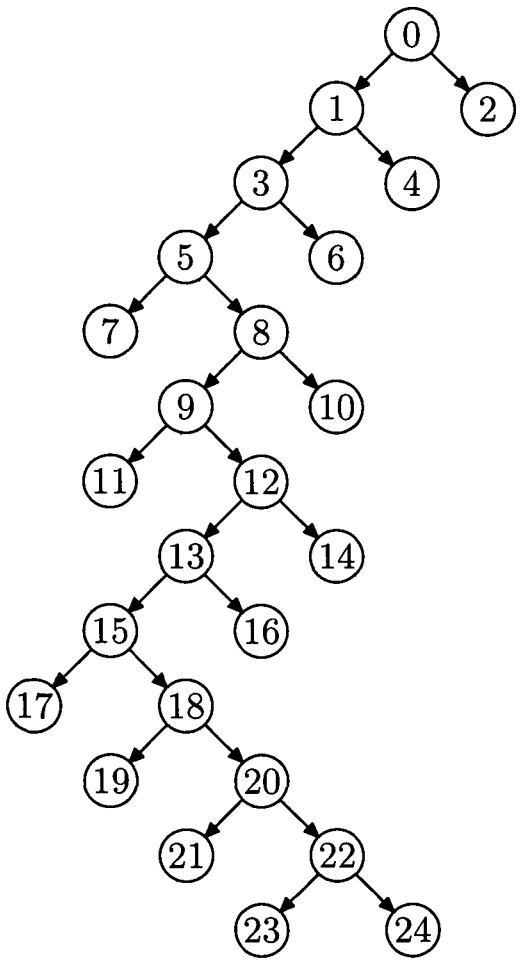

Figure 6.1: The B&B tree obtained by solving graph $G_{0.5}$, $n = 80$, using branching rule R2. The labels of the nodes correspond to the order, in which the nodes have been generated.

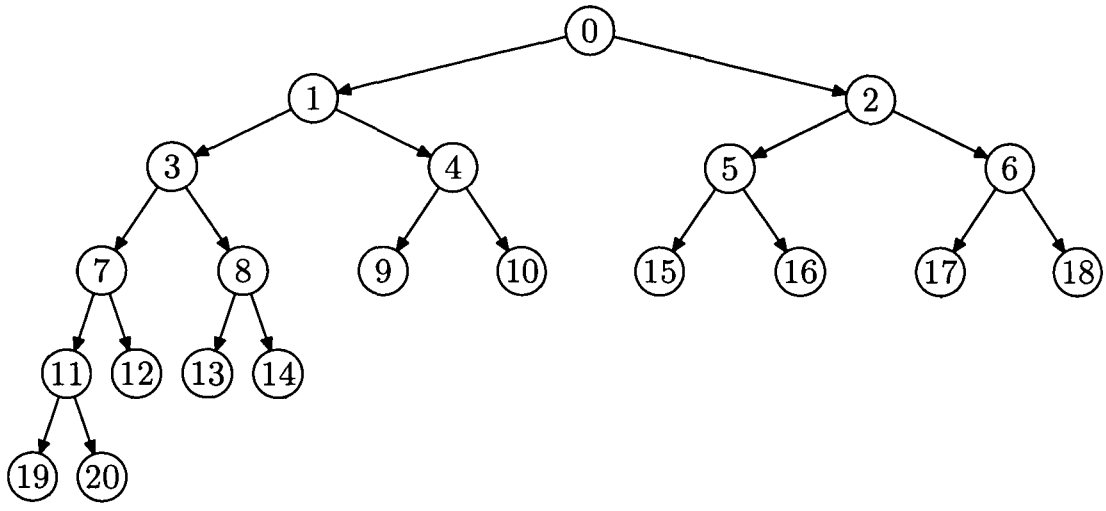

Figure 6.2: The B&B tree obtained by solving graph $G_{0.5}$, $n = 80$, using branching rule R3. The labels of the nodes correspond to the order, in which the nodes have been generated.

Thus, the dual functional reads

$$\begin{aligned} f_{i\sim j}(\gamma, u) &= \max_{\mathrm{diag}(X)=e, X\succeq 0} \mathcal{L}(X;\gamma,u) = \\ &= \max_{\mathrm{diag}(X)=e, X\succeq 0} \langle L + u(E_{ij}+E_{ji}) - \mathcal{A}^T(\gamma), X\rangle + b^T\gamma - 2u, \end{aligned}$$

where E_{ij} is the matrix having one in the (i,j)-entry and zero elsewhere. Clearly, the following inequalities always hold:

$$z_{i\sim j} = \min_{\gamma\geq 0, u\in\mathbb{R}} f_{i\sim j}(\gamma,u) \leq \min_{\gamma\geq 0} f_{i\sim j}(\gamma,\tilde{u}) \leq f_{i\sim j}(\tilde{\gamma},\tilde{u})$$

for any $\tilde{u}\in\mathbb{R}, \tilde{\gamma}\geq 0, \tilde{\gamma}\in\mathbb{R}^m$. Therefore, we choose a constant $\tilde{u} > 0$ to force the constraint $x_{ij} + x_{ji} = 2$ to hold and call the bundle routine with the objective function $\langle L+\tilde{u}(E_{ij}+E_{ji}), X\rangle$ and the dual information available from the previous calculation of the bound. In this way we obtain a valid upper bound on the sub problem.

Let z_{bk} be the best known cut value. If the condition

$$f_{i\sim j}(\tilde{\gamma},\tilde{u}) \leq z_{bk}$$

holds, the decision $i \sim j$ can be omitted from the Branch & Bound tree.

Separate node i and node j. Similarly, we can consider the case, where we want to cut edge $[ij]$, i.e. $X_{i\not\sim j}$.

How to use strong branching. We consider two different variants of incorporating strong branching.

First R2, then R3. First, find the 6 'best' potential branching edges according to rule R2. These are edges, where we expect to be able to cut off one of the sub trees quickly. We do a quick forecast on each of these branching decisions and hope to be able to fathom this node (before we actually add it), i.e. we only have to add one node to the branching tree. If we do not succeed in finding such an edge, we do a forecast on the 6 'best' branching edges according to rule R3 and branch then on the edge, where the decrease of the bound in both children is maximal.

R3 only. Do a forecast on the 6 'best' branching edges of rule R3. Branch on the edge, where the largest decrease of the bound in both children is expected.

Performance. It was disappointing to see, that strong branching does not seem to be worth the effort. There was only a slight decrease in the number of nodes, compared to the vast increase of computation time. Contrary to LP, where sensitivity analysis is at hand, for Semidefinite Programming no similar results exist and the dual information available at the end of the bundle iterations cannot be utilized. If we actually add a node to the B&B tree, that could be omitted by strong branching, the effect is, that after very little bundle iterations the node can be fathomed anyway.

6.3 Implementation of the Biq Mac Solver

We implemented the Branch & Bound Algorithm described above in C. For operations arising from linear algebra we use ATLAS (Automatically Tuned Linear Algebra Software), which is an open source implementation of BLAS (Basic Linear Algebra Subprograms) and parts of LAPACK (Linear Algebra PACKage).

To solve the SDP, which has to be done several times during the algorithm, we implemented a predictor-corrector version of an Interior-Point Algorithm using the HKM-direction (see Section 1.4.1).

We did not cover yet, when to stop the bundle algorithm. Although there is a very distinct tailing-off effect, it is not advisable to let the algorithm run, until we have reached tailing-off. The reason is, that if it is obvious, that the bounding procedure will not succeed to declare the best known feasible solution of this node to be optimal, we do not want to waste time by calculating the bound tightly. Therefore, after some bundle iterations we make a rough forecast to decide whether it is worth doing more bundle iterations or if the gap, i.e. the difference between the bound and the best known feasible solution, cannot be closed anyway.

Another issue is, after how many bundle-iterations we should purge the constraints and add newly violated ones. Let us call the bundle-iterations done

between the process of purging/separating 'inner iterations' and one round of doing the inner iterations followed by purging and separating constraints an 'outer iteration'. At the beginning of the bundle algorithm the number of inner iterations should be small, say three iterations. Then this number is increased in each outer iteration until a limit of, say ten inner iterations, is reached. This is motivated by the fact, that in the beginning it will take a while until we have found the 'right' set of triangle-inequalities and we do not waste time by trying hard to push the bound down if the current set of triangle-inequalities does not allow much progress.

Experiments show that it is definitely advisable to apply these dynamic strategies. Using the Bundle Method as described in Section 5.1.2 in a B&B framework without these features leads to significantly longer computation times. By applying the dynamic strategies, we saved up to 30% of computation time compared to an algorithm using a static number of inner and outer iterations.

We can now summarize the algorithm, performed at each node of the B&B tree as Algorithm 6.2, with subroutines described in Algorithms 6.3 and 6.4 below.

Algorithm 6.2 (Biq Mac, algorithm performed at each node of the B&B tree) *Subroutines 'function evaluation' and 'separation' can be found in Algorithms 6.3 and 6.4 below.*

Input.

Laplace matrix L.

Parameter:

$it_{inner}, it_{outer}, it_{extra}, maxit_{inner}, minit_{outer}$

$gap_{relax} = 1$ *if all weights are integers, otherwise* $gap_{relax} = 0$.

branchingule, ε.

Start.

i=1, *done=false,* $I = \emptyset$, $\hat{\gamma} = []$.

$\hat{f}, \hat{g}, \hat{X} \leftarrow$ *function evaluation with* $L, \hat{\gamma}, I$.

$G = \hat{g}, F = \langle L, \hat{X} \rangle, \mathcal{X} = \hat{X}$.

$I \leftarrow$ *separation with* $\hat{X}$.

$x_{bk}, z_{bk} \leftarrow$ *rounding heuristic, see Section 3.4.*

repeat

$\hat{f}, \hat{g}, \hat{X} \leftarrow$ *function evaluation with* $L, \hat{\gamma}, I$.

add this information to bundle $F, G, \mathcal{X}$.

j=1.

while $j < it_{inner}$ *and* $\hat{f} + \varepsilon \geq z_{bk} + gap_{relax}$

Solve (5.12) and obtain λ, η.

$\gamma_{test} = \hat{\gamma} - t(\eta - G\lambda)$, *see (5.11)*

$f_{test}, g_{test}, X_{test} \leftarrow$ *function evaluation* f *at* γ_{test}.

if $f_{test} < \hat{f}$
Serious step:
$\hat{\gamma} = \gamma_{test}$.
$t = 1.01t$.
else
Null step:
$t = t/1.02$.
Append X_{test} *to* $\mathcal{X}$, g_{test} *to* G, $\langle L, X_{test}\rangle$ *to* F.
j=j+1.
endwhile.
Purge bundle:
$B = \{i : \gamma_i > 0.01\bar{\gamma})$ *with* $\bar{\gamma} = (\sum_{i=1}^{m} \gamma_i)/m$.
$G = G(B)$, $F = F(B)$, $\mathcal{X} = \mathcal{X}(B)$.
$J \leftarrow$ *separation with* $.1\hat{X} + .9X_{old}$.
$I = I \cup J$.
if $\hat{f} + \varepsilon < z_{bk} + gap_{relax}$
/* *gap closed* */
done=true.
else if $i = minit_{outer}$
if $(f_{old} - \hat{f}) \cdot (it_{outer} - minit_{outer}) > z_{bk} + gap_{relax}$
/* *we will not be able to close the gap* */
done=true.
else if $i = it_{outer}$
/* *check whether some extra iterations could close the gap* */
if $(f_{old} - \hat{f}) \cdot it_{extra} > z_{bk} + gap_{relax}$
done=true.
else if $i = it_{outer} + it_{extra}$
/* *maximum number of iterations reached* */
done=true.
$i = i + 1$.
$it_{inner} = \min\{it_{inner} + 1, maxit_{inner}\}$.
until *done.*
$x_{bk}, z_{bk} \leftarrow$ *rounding heuristic, see Section 3.4.*
obtain branching edge $[ij]$ *according to the specified branchingrule.*
return $[ij]$ *and* $x_{bk}, z_{bk}, z_{ub} = \hat{f}$.

Algorithm 6.3 (Biq Mac, subroutine function evaluation)
Input: *Laplace matrix* L, *dual variables* γ, *index set* I.
solve $\max \langle L - \mathcal{A}_I^T(\gamma), X\rangle$ *s.t.* $X \in \mathcal{E}$ *giving* X,
$f = f(\gamma) = b^T\gamma + \langle L - \mathcal{A}_I^T(\gamma), X\rangle$,
$g = g(\gamma) = b - \mathcal{A}_I(X)$.
return *function value* f, *subgradient* g, *and matrix* X *such that* (γ, X) *is a matching pair.*

Algorithm 6.4 (Biq Mac, subroutine separation)
Input.
Matrix X.
Parameter heapsize, $maxgen_{tri}$, $mandatory_{tri}$
Create a heap of size 'heapsize' with those triangle-inequalities violated most by X.
$I \leftarrow$ indices of the $mandatory_{tr}$ most violated ones.
Choose randomly $(max_{gen} - mandatory_{tr})$ triangle-inequalities from the remaining heap and add it to I.
return *indices I of new constraints.*

In Figure 6.3 we present a typical call-graph of solving some instance. `main` is the program that administrates the B&B tree and calls the bounding procedure. In `bound` the bundle-routine is called and the branching-edge is determined. `bdl_main` calls the heuristic, starts the bundle-iterations and also separation and purging of the triangles are done in this program. Note, that the heuristic-routine as well as the separation-routine takes so little time (less than 2% each), that they are not displayed in this call-graph. The 'inner' bundle iterations are done in `bdl_mc2`, here also the routine doing the function evaluation `fct_eval` and the routine `lam_eta` for solving the λ-η-problem (5.12) are invoked. Solving the SDP in order to evaluate the function is done in `mc_psdpk`. The other routines in the graph are doing linear algebra operations, like matrix multiplications, matrix inversion, cholesky factorization, etc.

Since the programs of our interest are only the top-level routines, we zoom in Figure 6.4 into the top 11 sub-programs. We see that program `mc_psdpk`, which solves the SDP, uses up 88% of the overall time, whereas time for solving the λ-η-problem, solving the heuristic, and doing separation is negligible. This means, the best way of saving computation time, is trying to keep the number of function evaluations small. For this reason we choose the number of inner and outer iterations dynamically, as described above.

6.4 The Biq Mac Library

We have collected a wide variety of test problems of both, (**MC**) and (**QP**). All the data sets together with a description can be downloaded from `http://www.math.uni-klu.ac.at/or/Software`. The instances are taken from various sources, here we provide the characteristics of the data sets and references to papers where the problems are introduced and (possibly) solved. Throughout this section n is the dimension of the graph (for (**MC**) instances) or the dimension of matrix Q (for (**QP**) instances). d is the density of the graph or of matrix Q, respectively.

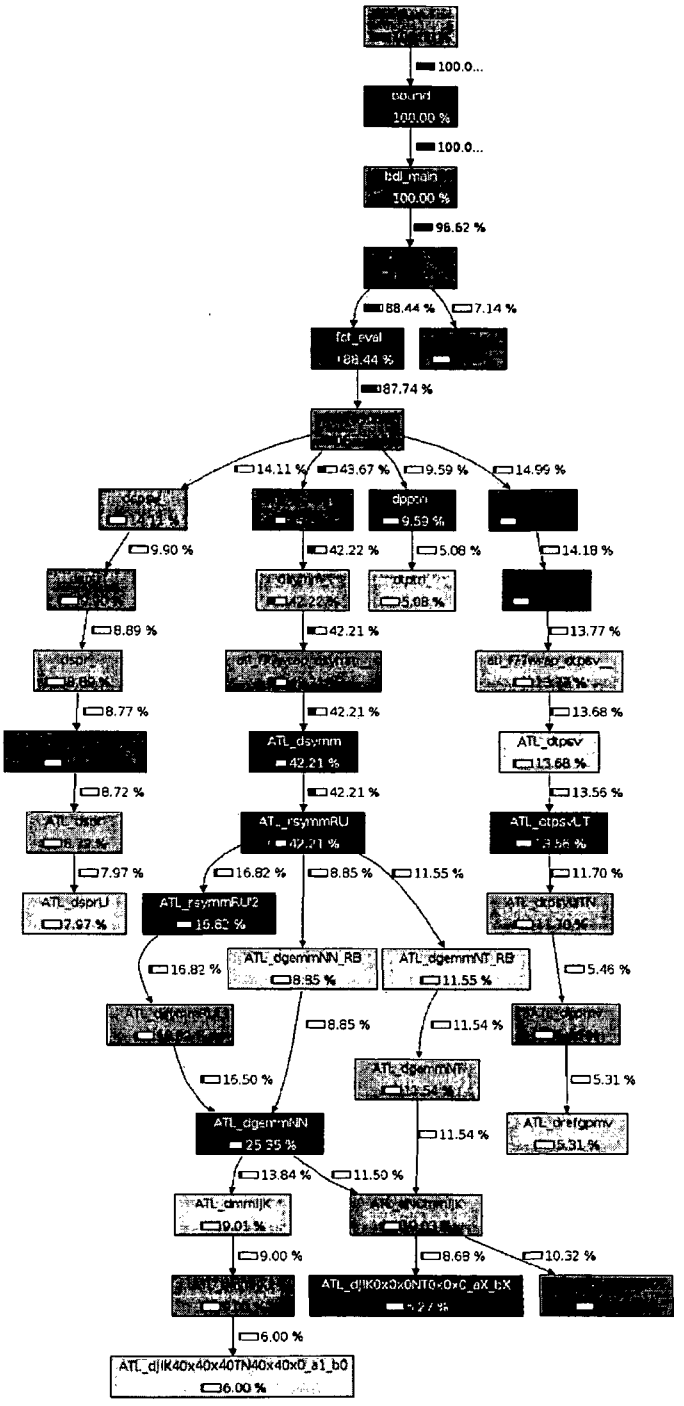

Figure 6.3: A typical call graph. Note, that the subroutine that does the separation of the triangle inequalities consumes so little time, that it does not appear in this graph.

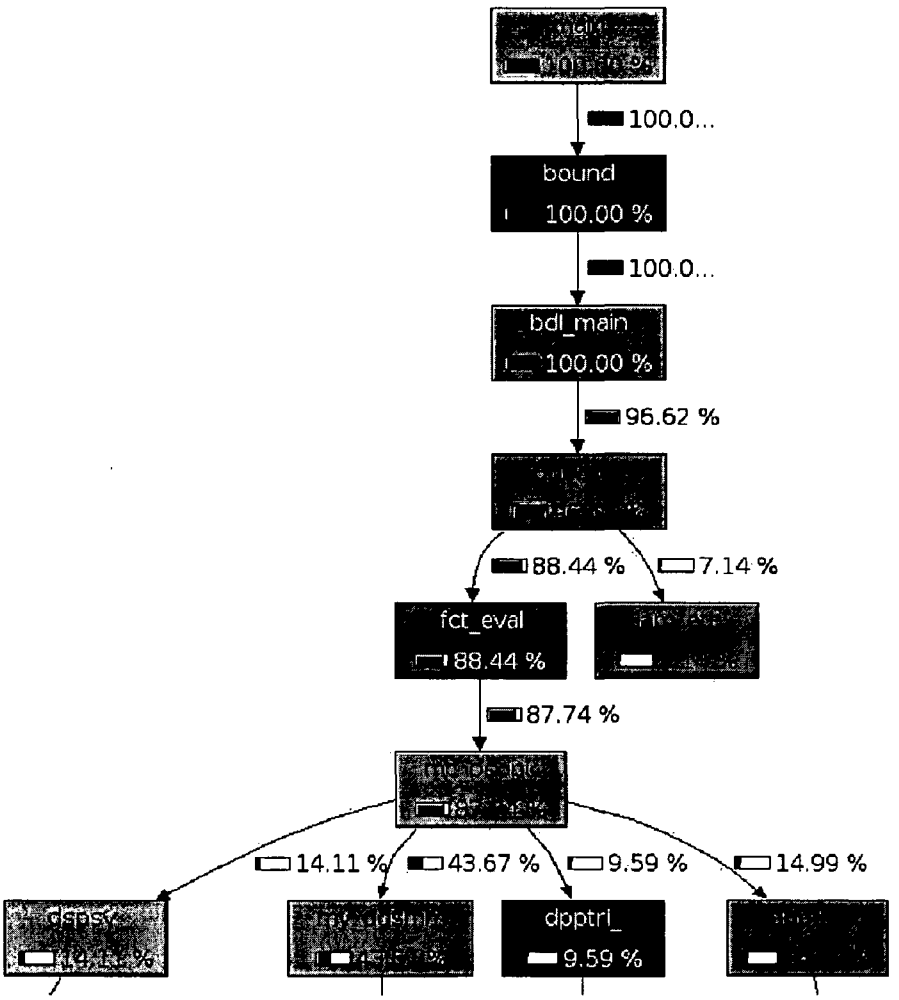

Figure 6.4: Call graph, zoomed in to the top 11 subroutines. Program `mc_psdpk` is solving the SDP, `lam_eta` solves (5.12).

6.4.1 Max-Cut

rudy-generated instances

The first group of instances is suggested by Helmberg and Rendl [54] and consists of random graphs (of specified edge density) with various types of random edge weights. All graphs were generated by the graph generator `rudy` of Rinaldi [116]. We generated ten instances each of the following types of graphs:

- $G_{0.5}$
  unweighted graphs with edge probability $1/2$, $n = 80$ and $n = 100$.

- $G_{-1/0/1}$
  weighted (complete) graphs with edge weights chosen uniformly from $\{-1, 0, 1\}$ and density $d = 0.1$ and $d = 0.99$, $n = 100$.

- $G_{[-10,10]}$
  Graphs with integer edge weights chosen from $[-10, 10]$ and density $d = 0.5$ and $d = 0.9$, $n = 100$.

- $G_{[0,10]}$
  Graphs with integer edge weights chosen from $[-10, 10]$ and density $d = 0.5$ and $d = 0.9$, $n = 100$.

Applications in physics: Ising instances

We also consider a set of test-problems from F. Liers [personal communication, Dec. 2005] deduced from physical applications. There are two kind of instances:

- two- and three-dimensional grid graphs, with Gaussian-distributed weights (zero mean and variance one). We select graphs of size 10×10, 15×15, 20×20, $5 \times 5 \times 5$, $6 \times 6 \times 6$ and $7 \times 7 \times 7$.

- dense Ising instances (one dimensional Ising chain), i.e. complete graphs with a certain structure. These instances are obtained in the following way: all nodes lie evenly distributed on a cycle. The weights of the edges depend on the Euclidean distance between two nodes and a parameter σ, such that the following proportion holds:

$$c_{ij} \sim \frac{\epsilon_{ij}}{r_{ij}^{\sigma}}$$

  where ϵ_{ij} is chosen according to a Gaussian distribution with zero mean and variance one and r_{ij} is the Euclidean distance between nodes i and j. The graphs we have chosen are of size $n \in \{100, 150, 200, 250, 300\}$.

6.4.2 Instances of (QP)

Pardalos and Rodgers [102] have proposed a test problem generator for unconstrained quadratic binary programming. Their routine generates a symmetric integer matrix Q to define the objective function for (QP), with the linear term c represented by the main diagonal of Q, and has several parameters to control the characteristics of the problem, namely:

n	the number of variables
d	the density, i.e. the probability that a nonzero will occur for any off-diagonal coefficient (q_{ij})
c^-	the lower bound of the diagonal coefficients (q_{ii})
c^+	the upper bound of the diagonal coefficients (q_{ii})
q^-	the lower bound of the off-diagonal coefficients (q_{ij})
q^+	the upper bound of the off-diagonal coefficients (q_{ij})
s	a seed to initialize the random number generator

$q_{ii} \sim$ discrete uniform $(c^-, c^+), i = 1, \ldots, n$

$q_{ij} = q_{ji} \sim$ discrete uniform $(q^-, q^+), 1 \leq i < j \leq n$.

Several test problems generated this way are provided in the OR-library maintained by Beasley [16], [17]. We have chosen all the problems of sizes of our interest, which are the data sets `bqpgka`, due to Glover, Kochenberger, and Alidaee [37] and `bqp100` and `bqp250`, see Beasley [18].

Furthermore, Billionnet and Elloumi [23] extended the sets `c` and `e` of `bqpgka`. We call these instances `bqpbe`.

In contrast to the `bqpgka` data that have varying characteristics, the data sets `beasley` and `bqpbe` consist of 10 instances for each specification.

The characteristics are as follows:

- `bqpgka`

	n	d	c^-	c^+	q^-	q^+
`bqpgka, set a`	$30, \ldots, 100$	$0.0625, \ldots, 0.5$	-100	100	-100	100
`bqpgka, set b`	$20, \ldots, 120$	1.0	0	63	-100	0
`bqpgka, set c`	$40, \ldots, 100$	$0.1, \ldots, 0.8$	-100	100	-50	50
`bqpgka, set d`	100	$0.1, \ldots, 1.0$	-75	75	-50	50
`bqpgka, set e`	200	$0.1, \ldots, 0.5$	-100	100	-50	50

- `beasley`

	n	d	c^-	c^+	q^-	q^+
`beasley100`	100	0.1	-100	100	-100	100
`beasley250`	250	0.1	-100	100	-100	100

- `bqpbe`

	n	d	c^-	c^+	q^-	q^+
bqpbe100.1	100	1.0	−100	100	−50	50
bqpbe120.3	120	0.3	−100	100	−50	50
bqpbe120.8	120	0.8	−100	100	−50	50
bqpbe150.3	150	0.3	−100	100	−50	50
bqpbe150.8	150	0.8	−100	100	−50	50
bqpbe200.3	200	0.3	−100	100	−50	50
bqpbe200.8	200	0.8	−100	100	−50	50
bqpbe250.1	250	0.1	−100	100	−50	50

6.5 Numerical Results

In this section we want to compare the Biq Mac solver with other exact solution methods. Some tables to briefly summarize the results are given within this section, tables with the detailed results can be found in Appendix C. If not stated otherwise, all computations were done on a Pentium IV, 3.6 GHz and 2 GB RAM with operating system Linux.

Comparison with other SDP based algorithms. First, we want to exemplify the comparison between Biq Mac and the B&B algorithm that uses the basic SDP relaxation, as developed by Poljak and Rendl [106] (Section 3.3.2). Figure 6.5 shows the decrease of the bound over time of Biq Mac and of the algorithm using the method of [106]. This small example, an unweighted random graph of size $n = 60$ and edge-probability 0.5, has the optimal (**MC**) value 536 and since the edge-weights are integers we can stop the algorithm as soon as the upper bound is strictly smaller than 537. Biq Mac used 20 seconds to solve this problem, whereas using only the basic SDP relaxation in a B&B framework (as done in [106]) results in 453.16 seconds computation time (8483 nodes in the B&B tree). Although the initial gap of the [106]-B&B algorithm is rather small, there is no further significant progress in all the subsequent nodes of the B&B tree. Therefore, albeit the bounding procedure is very cheap, the overall time of the algorithm is significantly larger than the time that Biq Mac uses. This nicely shows the great impact of including triangle-inequalities in the SDP relaxation.

Like Biq Mac, the algorithm of Helmberg and Rendl [54] tries to use (**SDPMET**) as bounding routine (see Section 4.2). Still, this method performs worse than Biq Mac. To illustrate this we plot in Figure 6.6 the decrease of the bound over time within the root-node of the B&B tree for both approaches. We took a graph from the Beasley collection (bqp250-6) with $n = 250$, see Section 6.4.2. The optimal value $z_{MC} = 43931$ (bottom line in the table). We also computed z_{SDPMET} and found that $z_{SDPMET} \leq 44095$ (dashed horizontal line). The topmost curve in the figure shows the decrease of the upper bound during the progress of adding more and more constraints and solving the resulting SDP by an Interior-Point Method.

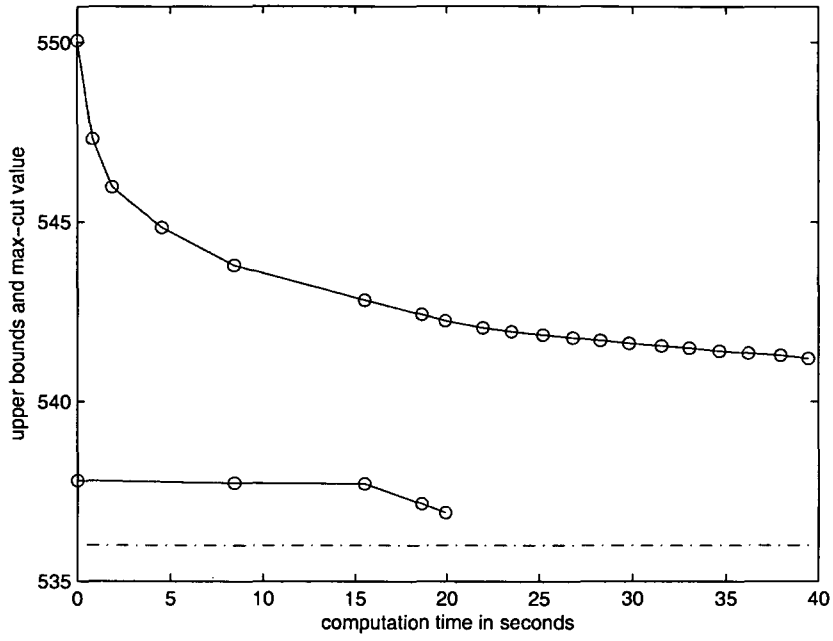

Figure 6.5: Solving an instance of $G_{0.5}$, $n = 60$, with the B&B algorithm introduced by Poljak and Rendl [106] (line above) and Biq Mac. Optimal solution of Biq Mac after 20 seconds, [106] takes 453.16 seconds.

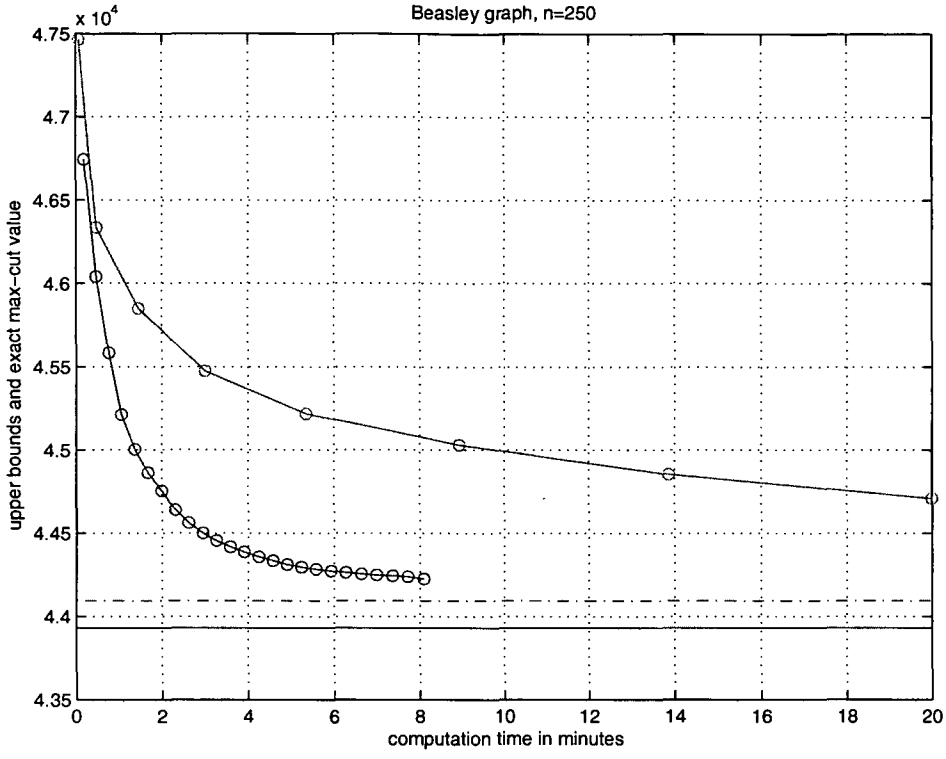

Figure 6.6: Beasley graph with $n = 250$. The optimal value of Max-Cut is 43931, $z_{SDPMET} \approx 44095$.

The curve below inf Figure 6.6 shows the progress of the upper bound of Biq Mac. This picture should be convincing enough, that the bounding routine used in Biq Mac, i.e. the Bundle Method, is substantially more efficient in reaching z_{SDPMET} and therefore, the overall B&B algorithm is superior to the method described in [54].

These two examples illustrate the performance of Biq Mac compared to the SDP based methods [106] and [54]. Since this is a generic behavior, these characteristic pictures apply generally to any instance to be solved. Therefore we can claim that our method is superior to the other SDP based B&B methods [106] and [54] and we will skip comparing with these two methods in the subsequent sections.

Comparison with solution methods other than SDP. Other methods for which numerical results are available are LP based algorithms, the convex quadratic approach of Billionnet and Elloumi [23], the second-order cone programming algorithm of Muramatsu and Suzuki [97] and the method of Pardalos and Rodgers [102]. Since the latter two perform already worse than [54], which

n	d	solved	min-time h:min	avg-time h:min	max-time h:min	min nodes	avg nodes	max nodes
$G_{0.5}$								
100	0.5	10	5	50	3:44	65	610	2925
$G_{-1/0/1}$								
100	0.99	10	7	56	2:31	79	651	1811
$G_{[-10,10]}$								
100	0.5	10	9	38	1:13	97	435	815
100	0.9	10	5	57	3:12	51	679	2427
$G_{[1,10]}$								
100	0.5	10	7	48	2:02	111	576	1465
100	0.9	10	12	40	1:26	155	464	1007

Table 6.1: Average Branch & Bound results for Max-Cut problems. Run times on a Pentium IV, 3.6 GHz, 2GB RAM.

can be beaten by our code, we may omit comparisons of these two methods with Biq Mac. (See also Sections 3.3.4 and 3.3.5.)

LP base methods (Section 3.3.1) perform very well for sparse (MC) instances. Therefore, in the subsequent section we will use these instances for comparison of these LP based methods with Biq Mac.

Comparison with the work of Billionnet and Elloumi [23] (Section 3.3.3) will be done in detail in the subsequent sections.

To the best of our knowledge, no other codes or papers presenting competitive numerical results are available.

6.5.1 Numerical Results of (MC) Instances

rudy-generated instances

Table 6.1 lists the computation times (minimum, average and maximum) and the number of nodes (minimum, average, maximum) of the resulting Branch & Bound tree of the $n = 100$-instances as described in Section 6.4.1. The branching rule used for these kind of instances is R2.

The average computation times for all kind of instances are approximately one hour. Nevertheless, instances may also be solved within some minutes, and contrary, it could take more than three hours for some graphs to obtain a solution.

The results show that we outperform on these classes of instances all other solution approaches known so far. The currently strongest results on these graphs are due to Billionnet and Elloumi [23]. They are not able to solve instances $G_{-1/0/1}$ of size $n = 100$ at all. Also, they could solve only two instances out of ten of $G_{0.5}$, $n = 100$. The detailed results can be found in Tables C.1–C.4.

Problem number	n	[87] time (sec)	Biq Mac time (sec)	Problem number	n	[87] time (sec)	Biq Mac time (sec)
2 dimensional				3 dimensional			
g10_5555	100	0.15	10.12	g5_5555	125	2.68	18.01
g10_6666	100	0.14	15.94	g5_6666	125	3.29	24.52
g10_7777	100	0.18	14.89	g5_7777	125	3.07	26.00
g15_5555	225	0.44	304.03	g6_5555	216	20.56	280.85
g15_6666	225	0.78	359.87	g6_6666	216	37.74	2025.74
g15_7777	225	0.67	346.89	g6_7777	216	27.30	277.95
g20_5555	400	1.70	6690.99	g7_5555	343	95.25	432.71
g20_6666	400	3.50	35205.95	g7_6666	343	131.34	550.12
g20_7777	400	2.61	8092.80	g7_7777	343	460.01	117782.75

Table 6.2: Test runs on torus graphs with Gaussian distribution. Branch & Cut algorithm computed on an 1.8 GHz machine, Branch & Bound done on a Pentium IV, 3.6 GHz. Time in seconds.

Applications in Physics: Ising instances

As explained in Section 6.4.1, there are two kind of Ising instances: toroidal grid graphs and complete graphs.

Instances of the first kind can be solved efficiently by an LP-based Branch & Cut algorithm (see Liers et al. [87]). The computation times of their and our algorithm are reported in Table 6.2 (for further details see Table C.5). As can be seen, on these sparse instances the LP-based method outperforms our algorithm. However, we find a solution within a gap of 1% in reasonable time for all these samples.

The run-time of the Branch-Cut & Price algorithm (Liers [86]) developed for the second kind of problems depends strongly on the parameter σ. For σ close to zero, we have a complete graph with Gaussian-distributed weights. But for σ chosen suitably large, some of the edges become 'unimportant' and the pricing works very well for these graphs. In Table 6.3 the computation times of [86] and our algorithm is given. (The detailed Biq Mac results can be found in Table C.6.) For $\sigma = 3.0$, roughly speaking we have the same computation times. But for $\sigma = 2.5$, the Branch-Cut & Price algorithm already takes more than 20 hours for instances of size $n = 150$, whereas our algorithm needs almost similar computation times as in the $\sigma = 3.0$ case.

For both kind of instances we used branching rule R3.

Problem number	n	[86] h:min:sec	Biq Mac h:min:sec	Problem number	n	[86] h:min:sec	Biq Mac h:min:sec
$\sigma = 3.0$				$\sigma = 2.5$			
100_5555	100	4:52	1:36	100_5555	100	18:22	1:32
100_6666	100	0:24	0:34	100_6666	100	6:27	1:06
100_7777	100	7:31	0:48	100_7777	100	10:08	0:47
150_5555	150	2:36:46	4:38	150_5555	150	21:28:39	4:25
150_6666	150	4:49:05	3:55	150_6666	150	23:35:11	5:39
150_7777	150	3:48:41	6:06	150_7777	150	31:40:07	9:19
200_5555	200	9:22:03	10:07	200_5555	200	–	10:05
200_6666	200	32:48:03	18:53	200_6666	200	–	17:55
200_7777	200	8:53:26	22:42	200_7777	200	–	21:38
250_5555	250	21:17:07	1:46:29	250_5555	250	–	3:00:28
250_6666	250	7:42:25	15:49	250_6666	250	–	1:17:04
250_7777	250	17:30:13	57:24	250_7777	250	–	1:10:50
300_5555	300	17:20:54	2:20:14	300_5555	300	–	6:43:47
300_6666	300	10:21:40	1:32:22	300_6666	300	–	9:04:38
300_7777	300	18:33:49	3:12:13	300_7777	300	–	13:00:10

Table 6.3: Test runs on Ising instances (complete graphs). Branch-Cut & Price computed on a 1.8 GHz Machine, Branch & Bound on a 3.6 GHz PC. Times in hours:minutes:seconds.

6.5.2 Numerical Results of (QP) Instances

In this section we report the results for the instances derived from (QP). Best known lower and upper bounds for `bqpgka` and `beasley` data are reported at the pseudo-Boolean website maintained by Boros, Hammer, and Tavares [25]. Our results can be found in Tables C.7–C.12, summarized we obtain the following:

- `bqpgka`.
  - **Set a.** All problems are solved in the root node of the Branch & Bound tree within seconds.
  - **Set b.** These instances could all be solved, but were extremely challenging for our algorithm. The reason is, that the objective value in the Max-Cut formulation is of magnitude 10^6, and therefore even a relative gap of 0.1% does not allow to fathom the node. However, allowing a relative gap of 0.1%, we can solve all problems in the root node of the Branch & Bound tree.
  - **Set c.** Similar to a, also these instances were solved within some seconds in the root node of the Branch & Bound tree.
  - **Set d.** The problems of set d could be solved within at most 7 minutes.
  - **Set e.** The instances with 10, 20, 30 and 40% density could all be solved within 2 hours of computation time. The 50% instance has been solved after 35 hours. According to [25], these problems have not been solved before.

- `beasley`.

  Solving the 10 problems of size $n = 100$ can be done in the root node within one minute. Regarding the $n = 250$ instances, two out of the ten problems have been solved before (see [25]), for the other eight problems we could prove optimality for the first time. Six out of these eight were solved within 5 hours, the other two needed 15 and 80 hours, respectively.

- `bqpbe`.

  We report the results of Billionnet and Elloumi [23] and our results in Table 6.4. As is shown in this table, [23] could not solve all out of the ten problems from the $n = 120$ variables and 80% density instances on, whereas our method still succeeded to solve them all. From the instances $n = 150, d = 80\%$ on, the convex-quadratic approach failed to solve any instance within their time limit of 3 hours. We still managed to obtain solutions to all of these instances (although for one graph it took 54 hours).

		[23]				Biq Mac			
			CPU time (sec)				CPU time (sec)		
n	d	solved	min	avg.	max	solved	min	avg.	max
100	1.0	10	27	372	1671	10	86	178	436
120	0.3	10	168	1263	4667	10	29	162	424
120	0.8	6	322	3909	9898	10	239	1320	3642
150	0.3	1		6789		10	1425	2263	2761
150	0.8	0		–		10	1654	1848	2133
200	0.3	0		–		10	6282	35294	174819
200	0.8	0		–		10	5541	47740	148515
250	0.1	0		–		10	12211	13295	16663

Table 6.4: Comparison between the algorithm of Billionnet and Elloumi [23] and our Branch & Bound approach. Computation times of the convex-quadratic algorithm were obtained on a laptop Pentium IV, 1.6 GHz (time limit 3 hours), our results were computed on a Pentium IV of 3.6 GHz.

Deciding which branching rule is advisable for these instances is not so obvious anymore. Tentatively, for sparse problems R3 is superior, but the denser the instances are, the better is the performance of R2. A general recipe or an intelligent way of deciding at the top levels of the B&B tree which rule to follow would be worthwhile.

6.6 Extensions

Apart from (**MC**) and (**QP**) other problems from Combinatorial Optimization can be modelled in such a way, that we can use Biq Mac to solve them.

6.6.1 The Bisection Problem

In Section 2.3 we introduced the graph partitioning problem, and as a special case of it the bisection problem, which is given as

$$\begin{array}{ll} \min & \frac{1}{2}\sum_{[ij]\in E} c_{ij}(1 - x_i x_j) \\ \text{s.t.} & \sum_{i=1}^{n} x_i = 0 \\ & x \in \{\pm 1\}^n, \end{array}$$

for a graph $G = (V(G), E(G))$ and edge weights c_{ij}. By setting $X := xx^T$, the following natural semidefinite relaxation arises:

$$\begin{array}{ll} \min & \frac{1}{4}\mathrm{tr}LX \\ \text{s.t.} & \mathrm{tr}JX = 0 \\ & \mathrm{diag}(X) = e \\ & X \succeq 0, \end{array}$$

where L is the Laplace matrix of graph G and J is the matrix of all ones.

Let A be the adjacency matrix of an unweighted graph. Solving the Max-Cut problem of a graph with cost matrix

$$B = -A + J$$

gives a cut $\delta(S)$ with value z. If $|S| = \frac{n}{2}$ then

$$\frac{n^2}{4} - z$$

is the optimal value of the bisection problem. (The "$-$" in $B = -A + J$ arises, because we do a maximization instead of minimizing, and the J comes from the constraint $\mathrm{tr}JX = 0$, that is lifted into the objective function. The Lagrange multiplier for this constraint is guessed to be one.)

We consider the instances introduced by Johnson, Aragon, McGeoch, and Schevon [63] of size $n = 124$ and $n = 250$ and summarize in Table 6.5 the best results for these instances known so far (see Karisch and Rendl [65]). With our algorithm we could prove optimality of the known lower bounds of all instances of size $n = 124$, and one of the instances of size $n = 250$. To the best of our knowledge, these exact solutions were obtained for the first time. The improved gap for the instances of size $n = 250$ and densities 0.02, 0.04 and 0.08 were obtained after a time limit of 32 hours cpu-time.

6.6.2 Max-2Sat

In Section 2.4 the Max-Sat problem has been introduced, and as a special case of it Max-kSat. We consider here Max-2Sat, i.e. we have Boolean variables $\{x_1, \ldots, x_n\}$ and clauses that are the disjunction of at most two literals. Let us introduce variables $\{y_1, \ldots, y_n\}$, with the correspondence $y_i = 0$ if x_i is false, and $y_i = 1$ if x_i is true. For each clause we can then form a cost coefficient in the (0-1) model by the following transformation

$$\begin{array}{lll} x_i \vee x_j & \rightarrow & y_i + y_j - y_i y_j, \\ \overline{x}_i \vee x_j & \rightarrow & 1 - y_i + y_i y_j, \\ \overline{x}_i \vee \overline{x}_j & \rightarrow & 1 - y_i y_j. \end{array}$$

With this correspondence every problem arising from Max-2Sat can be transformed to an instance of (**QP**)and thus solved by Biq Mac.

		best known			Biq Mac
n	d	bound	$\lvert E_{cut}\rvert$	gap	gap
124	0.02	12.01	13	0	0
124	0.04	61.22	63	1	0
124	0.08	170.93	178	7	0
124	0.16	440.08	449	8	0
250	0.01	26.06	29	2	0
250	0.02	103.61	114	10	8
250	0.04	327.88	357	29	22
250	0.08	779.55	828	48	35

Table 6.5: Best known results of the bisection problem for the Johnson Graphs and the new gap obtained by Biq Mac.

6.6.3 The Stable Set Problem

The stable set problem of a graph $G = (V(G), E(G))$, introduced in Section 2.2, can also be transformed into a problem of the form (**QP**), such that the stable set problem can be solved using Biq Mac.

To do this transformation, let $\bar{A}$ be the adjacency matrix of the complement of graph G. Define

$$B := \begin{cases} \bar{a}_{ij} & \text{if } \bar{a}_{ij} \neq 0 \\ -M & \text{if } \bar{a}_{ij} = 0. \end{cases},$$

with M being a sufficiently large number and solve the problem

$$\begin{array}{ll} \max & x^T B x \\ \text{s.t.} & x \in \{0,1\}^n. \end{array} \tag{6.3}$$

This is a quadratic (0-1) problem, that can be transformed to Max-Cut. Problem (6.3) asks to assign 0 or 1 to each variable x_i. Thus, b_{ij} contributes to the objective value if and only if both, x_i and x_j, are assigned value 1, meaning that for $b_{ij} = -M$, it is very likely that not both, x_i and x_j will have value 1.

Thus, all nodes i with $x_i = 1$ form a stable set since an edge of the original graph (corresponding entry in B has value $-M$) is (hopefully) not in the cut. Of course, the final result has to be checked, since there is no guarantee that the solution will indeed form a stable set.

6.6.4 The Quadratic Knapsack Problem

Another standard Combinatorial Optimization problem is the quadratic knapsack problem:

$$\begin{array}{lll} (\mathbf{QK}) & \max & x^T C x \\ & \text{s.t.} & a^T x \leq b \\ & & x \in \{0,1\}^n. \end{array}$$

A detailed study of this problem can be found in the book of Kellerer, Pferschy, and Pisinger [67]. Clearly, by simply providing the possibility of handling one linear constraint, we can also solve problems of this type by Biq Mac. Incorporating the possibility of having one (or even more) linear constraint(s) is part of ongoing research with C. Buchheim [personal communication, 2006].

6.7 Concluding Remarks on Biq Mac

The solver Biq Mac, available via the web interface `http://BiqMac.uni-klu.ac.at`, solves (**MC**) and (**QP**) problems of any density up to size $n = 100$. To the best of our knowledge, no other algorithm can manage these instances in a routine way. Many kind of instances of sizes up to $n = 300$ can be solved as well. For the first time optimality could be proved for several problems of the OR-library, for instance all problems that are reported at the pseudo-Boolean website [25] with dimensions up to $n = 250$ are now solved. Also for the bisection problem optimality for some of the Johnson-Graphs has been proved for the first time, and for those where we could not close the gap, we reduced it significantly.

Using Biq Mac for sparse graphs is not advisable. Since linear programming based methods are capable of exploiting sparsity, solutions might be obtained much faster when applying these methods to sparse data.

We are currently working on an extended version of Biq Mac that is capable of handling some linear constraints explicitly, which will broaden the field of applications. Also, having a recipe for the proper branching rule or a way of dynamically deciding which rule to use, is work in progress.

Summary and Outlook

This thesis covers algorithms for large-scale Semidefinite Programs, as well as solution methods for the Max-Cut problem, an NP-complete problem arising from Combinatorial Optimization.

We explain basic theoretical issues of Semidefinite Programs and the most well-known solution methods (Chapter 1). The field of Combinatorial Optimization problems is reviewed (Chapter 2) and one of the problems, namely the Max-Cut problem, is discussed in more detail in Chapter 3.

Chapter 4 addresses SDP based relaxations for Max-Cut. We introduce a new relaxation for sparse problems where the support of the graph plays a major role to develop the SDP-relaxation and we implemented a Bundle Method to solve this Semidefinite Program. A topic of further investigations is to look at the practical behavior of other sparse models that are independent of the structure of the graph.

The increasing demand of algorithms for solving large-scale SDPs motivated the work in Chapter 5. Apart from recalling the concept of Bundle Methods and applying it in Semidefinite Programming, we developed a Spectral Bundle method that uses second order information. An implementation of this algorithm is part of ongoing work. This code should then also be capable of solving the sparse relaxation from Chapter 4 for higher dimensions. Another new algorithm introduced in Chapter 5 is the Boundary Point Method. The augmented Lagrangian algorithm is the background of this method, that works astonishing well for the problem of calculating the ϑ-number of a graph. For solving relaxations arising from other sources, the algorithm does not behave as nicely. Finding a way to make the Boundary Point Method applicable to more problems is part of future research.

In the final chapter we introduced the Biq Mac solver – a tool for solving Max-Cut (or, equivalently unconstrained binary quadratic programs) and the Biq Mac library – various instances arising from Max-Cut or unconstrained quadratic (0-1) programming. We made this solver publicly available via the web-site `http://BiqMac.uni-klu.ac.at`. Also, the collection of test problems is accessible. An ongoing maintenance of this library in order to develop benchmarks for Max-Cut and unconstrained quadratic (0-1) problems is intended.

Appendix A

Background Material

A.1 Positive Semidefinite Matrices

In this section we state some of the definitions and theorems concerning positive semidefinite matrices. All the proofs can be found in Horn and Johnson [61].

Theorem A.1 (Characterization of positive semidefinite matrices) *Let $X \in \mathcal{S}_n$. The following statements are equivalent.*

- $X \succeq 0$.
- $y^t X y \geq 0$ *for all* $y \in \mathbb{R}^n$.
- $\lambda(X) \geq 0$ *for all Eigenvalues* λ *of* X.
- *All principal minors of* X *are non-negative.*
- $\exists L \in \mathbb{R}^{n \times n} : \; X = LL^T$.

Lemma A.2 *Any principal submatrix of a positive definite matrix is positive definite.*

Theorem A.3 (Fejer) $A \in \mathcal{S}_n^+$ *if and only if* $\mathrm{tr}AB \geq 0 \; \forall B \in \mathcal{S}_n^+$

Lemma A.4 *If* $A, B \in \mathcal{S}_n^+$, *then* $\langle A, B \rangle \geq 0$ *and* $\langle A, B \rangle = 0$ *if and only if* $AB = 0$.

Observation A.5 *If* $A \in \mathcal{S}_n^+$ *and* $a_{ii} = 0$ *for some* $i \in \{1, \ldots, n\}$, *then* $a_{ij} = a_{ji} = 0$ *for all* $j \in \{1, \ldots, n\}$.

Definition A.6 (Schur complement) *If*

$$M = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix}$$

where A is nonsingular and C is symmetric. Then

$$C - B^T A^{-1} B$$

is called the Schur complement *of A in M.*

Theorem A.7 *Let*

$$M = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix}$$

where A is positive definite and C is symmetric. Then

$$M \succeq 0 \Leftrightarrow C - B^T A^{-1} B \succeq 0$$

A.2 Convexity, Minimax Inequality

Definition A.8 (convex set) *A set $C \subseteq \mathbb{R}^n$ is* convex *if $c_1, c_2 \in C$ implies $\alpha c_1 + (1-\alpha)c_2 \in C$ for all $\alpha \in [0,1]$.*

Definition A.9 (convex hull) *The* convex hull $\mathrm{conv}(S)$ *of some set $S \subseteq \mathbb{R}^n$ is the smallest convex set that contains S.*

Lemma A.10 $\mathrm{conv}(\{vv^T, \|v\| = 1\}) = \{W \succeq 0,\ \mathrm{tr}W = 1\}$.

Lemma A.11 (minimax inequality) *[117, Lemma 36.1] Let f be a function from the non-empty product set $C \times D$ to $[-\infty, \infty]$, then*

$$\sup_{u \in C} \inf_{v \in D} f(u,v) \leq \inf_{v \in D} \sup_{u \in C} f(u,v).$$

Corollary A.12 *[117, Theorem 37.3] Let f be a function from the non-empty product set $C \times D$, C and D being closed convex sets in $\mathbb{R}^m$ and $\mathbb{R}^n$, respectively, to $[-\infty, \infty]$. If either C or D is bounded, then*

$$\sup_{u \in C} \inf_{v \in D} f(u,v) = \inf_{v \in D} \sup_{u \in C} f(u,v).$$

Appendix B

Problem Generation

B.1 rudy Calls

All Max-Cut instances, apart from those arising from physical applications, are generated using rudy, a machine independent graph generator written by G. Rinaldi [116]. The commands are given in this section.

```
rudy -rnd_graph 80 50 8000 > g05_80.0
rudy -rnd_graph 80 50 8001 > g05_80.1
rudy -rnd_graph 80 50 8002 > g05_80.2
rudy -rnd_graph 80 50 8003 > g05_80.3
rudy -rnd_graph 80 50 8004 > g05_80.4
rudy -rnd_graph 80 50 8005 > g05_80.5
rudy -rnd_graph 80 50 8006 > g05_80.6
rudy -rnd_graph 80 50 8007 > g05_80.7
rudy -rnd_graph 80 50 8008 > g05_80.8
rudy -rnd_graph 80 50 8009 > g05_80.9

rudy -rnd_graph 100 50 10000 > g05_100.0
rudy -rnd_graph 100 50 10001 > g05_100.1
rudy -rnd_graph 100 50 10002 > g05_100.2
rudy -rnd_graph 100 50 10003 > g05_100.3
rudy -rnd_graph 100 50 10004 > g05_100.4
rudy -rnd_graph 100 50 10005 > g05_100.5
rudy -rnd_graph 100 50 10006 > g05_100.6
rudy -rnd_graph 100 50 10007 > g05_100.7
rudy -rnd_graph 100 50 10008 > g05_100.8
rudy -rnd_graph 100 50 10009 > g05_100.9

rudy -rnd_graph 100 10 1000 -random 0 1 1000 -times 2 -plus -1 > pm1s_100.0
rudy -rnd_graph 100 10 1001 -random 0 1 1001 -times 2 -plus -1 > pm1s_100.1
rudy -rnd_graph 100 10 1002 -random 0 1 1002 -times 2 -plus -1 > pm1s_100.2
rudy -rnd_graph 100 10 1003 -random 0 1 1003 -times 2 -plus -1 > pm1s_100.3
rudy -rnd_graph 100 10 1004 -random 0 1 1004 -times 2 -plus -1 > pm1s_100.4
rudy -rnd_graph 100 10 1005 -random 0 1 1005 -times 2 -plus -1 > pm1s_100.5
rudy -rnd_graph 100 10 1006 -random 0 1 1006 -times 2 -plus -1 > pm1s_100.6
rudy -rnd_graph 100 10 1007 -random 0 1 1007 -times 2 -plus -1 > pm1s_100.7
rudy -rnd_graph 100 10 1008 -random 0 1 1008 -times 2 -plus -1 > pm1s_100.8
rudy -rnd_graph 100 10 1009 -random 0 1 1009 -times 2 -plus -1 > pm1s_100.9

rudy -rnd_graph 100 99 1000 -random 0 1 1000 -times 2 -plus -1 > pm1d_100.0
rudy -rnd_graph 100 99 1001 -random 0 1 1001 -times 2 -plus -1 > pm1d_100.1
rudy -rnd_graph 100 99 1002 -random 0 1 1002 -times 2 -plus -1 > pm1d_100.2
rudy -rnd_graph 100 99 1003 -random 0 1 1003 -times 2 -plus -1 > pm1d_100.3
rudy -rnd_graph 100 99 1004 -random 0 1 1004 -times 2 -plus -1 > pm1d_100.4
rudy -rnd_graph 100 99 1005 -random 0 1 1005 -times 2 -plus -1 > pm1d_100.5
rudy -rnd_graph 100 99 1006 -random 0 1 1006 -times 2 -plus -1 > pm1d_100.6
rudy -rnd_graph 100 99 1007 -random 0 1 1007 -times 2 -plus -1 > pm1d_100.7
rudy -rnd_graph 100 99 1008 -random 0 1 1008 -times 2 -plus -1 > pm1d_100.8
rudy -rnd_graph 100 99 1009 -random 0 1 1009 -times 2 -plus -1 > pm1d_100.9

rudy -rnd_graph 100 50 1000 -random -10 10 1000 > w05_100.0
rudy -rnd_graph 100 50 1001 -random -10 10 1001 > w05_100.1
rudy -rnd_graph 100 50 1002 -random -10 10 1002 > w05_100.2
rudy -rnd_graph 100 50 1003 -random -10 10 1003 > w05_100.3
rudy -rnd_graph 100 50 1004 -random -10 10 1004 > w05_100.4
```

```
rudy -rnd_graph 100 50 1005 -random -10 10 1005 > w05_100.5
rudy -rnd_graph 100 50 1006 -random -10 10 1006 > w05_100.6
rudy -rnd_graph 100 50 1007 -random -10 10 1007 > w05_100.7
rudy -rnd_graph 100 50 1008 -random -10 10 1008 > w05_100.8
rudy -rnd_graph 100 50 1009 -random -10 10 1009 > w05_100.9

rudy -rnd_graph 100 90 1000 -random -10 10 1000 > w09_100.0
rudy -rnd_graph 100 90 1001 -random -10 10 1001 > w09_100.1
rudy -rnd_graph 100 90 1002 -random -10 10 1002 > w09_100.2
rudy -rnd_graph 100 90 1003 -random -10 10 1003 > w09_100.3
rudy -rnd_graph 100 90 1004 -random -10 10 1004 > w09_100.4
rudy -rnd_graph 100 90 1005 -random -10 10 1005 > w09_100.5
rudy -rnd_graph 100 90 1006 -random -10 10 1006 > w09_100.6
rudy -rnd_graph 100 90 1007 -random -10 10 1007 > w09_100.7
rudy -rnd_graph 100 90 1008 -random -10 10 1008 > w09_100.8
rudy -rnd_graph 100 90 1009 -random -10 10 1009 > w09_100.9

rudy -rnd_graph 100 50 1000 -random 1 10 1000 > pw05_100.0
rudy -rnd_graph 100 50 1001 -random 1 10 1001 > pw05_100.1
rudy -rnd_graph 100 50 1002 -random 1 10 1002 > pw05_100.2
rudy -rnd_graph 100 50 1003 -random 1 10 1003 > pw05_100.3
rudy -rnd_graph 100 50 1004 -random 1 10 1004 > pw05_100.4
rudy -rnd_graph 100 50 1005 -random 1 10 1005 > pw05_100.5
rudy -rnd_graph 100 50 1006 -random 1 10 1006 > pw05_100.6
rudy -rnd_graph 100 50 1007 -random 1 10 1007 > pw05_100.7
rudy -rnd_graph 100 50 1008 -random 1 10 1008 > pw05_100.8
rudy -rnd_graph 100 50 1009 -random 1 10 1009 > pw05_100.9

rudy -rnd_graph 100 90 1000 -random 1 10 1000 > pw09_100.0
rudy -rnd_graph 100 90 1001 -random 1 10 1001 > pw09_100.1
rudy -rnd_graph 100 90 1002 -random 1 10 1002 > pw09_100.2
rudy -rnd_graph 100 90 1003 -random 1 10 1003 > pw09_100.3
rudy -rnd_graph 100 90 1004 -random 1 10 1004 > pw09_100.4
rudy -rnd_graph 100 90 1005 -random 1 10 1005 > pw09_100.5
rudy -rnd_graph 100 90 1006 -random 1 10 1006 > pw09_100.6
rudy -rnd_graph 100 90 1007 -random 1 10 1007 > pw09_100.7
rudy -rnd_graph 100 90 1008 -random 1 10 1008 > pw09_100.8
rudy -rnd_graph 100 90 1009 -random 1 10 1009 > pw09_100.9
```

B.2 Pardalos-Rodgers Generator Parameters

In this section the parameters for calling the generator of Pardalos and Rodgers [102] are given. Unfortunately, for the `beasley` data we do not know the seed, but the data sets can be downloaded from the OR-Library [17].

The parameters are specified in the following order:

```
n density seed OffDiagLower OffDiagUpper DiagLower DiagUpper
```

B.2.1 Parameters for the bqpgka Instances

Set a.

```
50 0.1 10 -100 100 -100 100
60 0.1 10 -100 100 -100 100
70 0.1 10 -100 100 -100 100
80 0.1 10 -100 100 -100 100
50 0.2 10 -100 100 -100 100
30 0.4 10 -100 100 -100 100
30 0.5 10 -100 100 -100 100
100 0.0625 10 -100 100 -100 100
```

Set b.

```
 20 1.0 10 0 63 -100 0
 30 1.0 10 0 63 -100 0
 40 1.0 10 0 63 -100 0
 50 1.0 10 0 63 -100 0
 60 1.0 10 0 63 -100 0
 70 1.0 10 0 63 -100 0
 80 1.0 10 0 63 -100 0
 90 1.0 10 0 63 -100 0
100 1.0 10 0 63 -100 0
125 1.0 10 0 63 -100 0
```

Set c.

```
 40 0.8  10 -100 100 -50 50
 50 0.6  70 -100 100 -50 50
 60 0.4  31 -100 100 -50 50
 70 0.3  34 -100 100 -50 50
 80 0.2   8 -100 100 -50 50
 90 0.1  80 -100 100 -50 50
100 0.1 142 -100 100 -50 50
```

Set d.

```
100 0.1   31 -50 50 -75 75
100 0.2   37 -50 50 -75 75
100 0.3  143 -50 50 -75 75
100 0.4   47 -50 50 -75 75
100 0.5   31 -50 50 -75 75
100 0.6   47 -50 50 -75 75
100 0.7   97 -50 50 -75 75
100 0.8  133 -50 50 -75 75
100 0.9  307 -50 50 -75 75
100 1.0 1311 -50 50 -75 75
```

Set e.

```
200 0.1 51 -50 50 -100 100
200 0.2 43 -50 50 -100 100
200 0.3 34 -50 50 -100 100
200 0.4 73 -50 50 -100 100
200 0.5 89 -50 50 -100 100
```

B.2.2 Parameters for the bqpbe Instances

```
100 1.0 1. -50 50 -100 100
100 1.0 2. -50 50 -100 100
100 1.0 3. -50 50 -100 100
100 1.0 4. -50 50 -100 100
100 1.0 5. -50 50 -100 100
100 1.0 6. -50 50 -100 100
100 1.0 7. -50 50 -100 100
100 1.0 8. -50 50 -100 100
100 1.0 9. -50 50 -100 100
100 1.0 10. -50 50 -100 100

120 0.3 1. -50 50 -100 100
120 0.3 2. -50 50 -100 100
120 0.3 3. -50 50 -100 100
120 0.3 4. -50 50 -100 100
120 0.3 5. -50 50 -100 100
120 0.3 6. -50 50 -100 100
120 0.3 7. -50 50 -100 100
120 0.3 8. -50 50 -100 100
120 0.3 9. -50 50 -100 100
120 0.3 10. -50 50 -100 100

120 0.8 1. -50 50 -100 100
120 0.8 2. -50 50 -100 100
120 0.8 3. -50 50 -100 100
120 0.8 4. -50 50 -100 100
120 0.8 5. -50 50 -100 100
120 0.8 6. -50 50 -100 100
120 0.8 7. -50 50 -100 100
120 0.8 8. -50 50 -100 100
120 0.8 9. -50 50 -100 100
120 0.8 10. -50 50 -100 100

150 0.3 1. -50 50 -100 100
150 0.3 2. -50 50 -100 100
150 0.3 3. -50 50 -100 100
150 0.3 4. -50 50 -100 100
150 0.3 5. -50 50 -100 100
150 0.3 6. -50 50 -100 100
150 0.3 7. -50 50 -100 100
150 0.3 8. -50 50 -100 100
150 0.3 9. -50 50 -100 100
150 0.3 10. -50 50 -100 100

150 0.8 1. -50 50 -100 100
150 0.8 2. -50 50 -100 100
150 0.8 3. -50 50 -100 100
150 0.8 4. -50 50 -100 100
150 0.8 5. -50 50 -100 100
150 0.8 6. -50 50 -100 100
150 0.8 7. -50 50 -100 100
150 0.8 8. -50 50 -100 100
150 0.8 9. -50 50 -100 100
150 0.8 10. -50 50 -100 100

200 0.3 1. -50 50 -100 100
200 0.3 2. -50 50 -100 100
200 0.3 3. -50 50 -100 100
200 0.3 4. -50 50 -100 100
200 0.3 5. -50 50 -100 100
200 0.3 6. -50 50 -100 100
200 0.3 7. -50 50 -100 100
200 0.3 8. -50 50 -100 100
200 0.3 9. -50 50 -100 100
200 0.3 10. -50 50 -100 100

200 0.8 1. -50 50 -100 100
200 0.8 2. -50 50 -100 100
200 0.8 3. -50 50 -100 100
200 0.8 4. -50 50 -100 100
200 0.8 5. -50 50 -100 100
200 0.8 6. -50 50 -100 100
200 0.8 7. -50 50 -100 100
200 0.8 8. -50 50 -100 100
200 0.8 9. -50 50 -100 100
200 0.8 10. -50 50 -100 100

250 0.1 1. -50 50 -100 100
250 0.1 2. -50 50 -100 100
250 0.1 3. -50 50 -100 100
250 0.1 4. -50 50 -100 100
250 0.1 5. -50 50 -100 100
250 0.1 6. -50 50 -100 100
250 0.1 7. -50 50 -100 100
250 0.1 8. -50 50 -100 100
250 0.1 9. -50 50 -100 100
250 0.1 10. -50 50 -100 100
```

Appendix C

Tables with Detailed Numerical Results

This chapter presents the detailed numerical results. For each instance collected in the Biq Mac library, we tried solving the problem using both branching rules and list in the subsequent tables the solution times in seconds and the number of Branch & Bound nodes. All calculations were done on a Pentium IV with 3.6 GHz and 2GB RAM, operating system Linux.

If no run-time is given, the problem could not be solved within a time limit of 50 hours CPU-time. Instance bqp250-8 was the only one of the bqp250 instances that could not be solved within 50 hours. Therefore, we suspended the time limit for this instance and could solve it within 80 hours CPU-time.

Problem number	solution	R2 time	R2 nodes	R3 time	R3 nodes
$n = 80$					
0	929	182.92	71	238.73	55
1	941	76.84	15	18.98	3
2	934	112.21	33	84.50	19
3	923	976.91	351	2203.60	583
4	932	133.78	69	212.61	45
5	926	215.35	75	231.77	55
6	929	123.67	57	148.46	31
7	929	82.37	25	91.59	21
8	925	162.59	63	264.24	63
9	923	330.96	123	589.17	151
avg		239.76		408.37	
$n = 100$					
0	1430	2776.93	553	9675.74	1365
1	1425	13464.55	2925	71701.51	10793
2	1432	624.29	131	1721.97	215
3	1424	6702.87	1267	22413.26	3343
4	1440	317.40	65	551.99	65
5	1436	582.25	131	1358.17	161
6	1434	677.97	157	1912.96	243
7	1431	1431.73	289	4889.02	637
8	1432	1294.62	253	3413.08	447
9	1430	1852.16	329	4141.01	569
avg		2972.48		12177.87	

Table C.1: Test runs on $G_{0.5}$ – unweighted graphs with edge probability 1/2, generated by `rudy` [116].

Problem number	solution	R2 time	R2 nodes	R3 time	R3 nodes
$n = 100, d = 10\%$					
0	127	115.42	15	108.55	9
1	126	462.06	73	852.32	87
2	125	83.46	9	71.83	5
3	111	445.69	65	915.51	87
4	128	431.47	61	412.21	41
5	128	78.28	9	102.49	7
6	122	280.45	49	358.55	33
7	112	6.07 (1 node)			
8	120	93.75	9	27.48	3
9	127	83.48	9	53.12	3
avg		208.01		315.44	
$n = 100, d = 99\%$					
0	340	4872.18	937	16426.31	2259
1	324	9050.11	1811	34012.08	4663
2	389	7096.11	1347	23822.08	3307
3	400	1397.60	271	5076.16	633
4	363	5725.13	1123	16390.85	2185
5	441	1427.39	265	3459.10	421
6	367	1244.64	219	2581.31	299
7	361	398.20	111	1206.08	121
8	385	435.97	79	551.26	65
9	405	2003.08	351	4769.97	607
avg		3365.04		10829.52	

Table C.2: Test runs on $G_{-1/0/1}$, generated by `rudy` [116].

Problem number	solution	R2 time	R2 nodes	R3 time	R3 nodes
$n = 100, d = 0.5$					
0	1646	3834.44	741	10947.47	1429
1	1606	864.83	175	2032.56	235
2	1902	725.80	145	1283.40	147
3	1627	2698.21	511	7391.22	929
4	1546	4026.38	795	12924.05	1637
5	1581	4421.14	815	12852.52	1631
6	1479	1017.11	187	2001.12	229
7	1987	518.67	97	935.22	103
8	1311	4424.00	781	14528.44	1897
9	1752	513.75	99	620.35	63
avg		2304.43		6551.64	
$n = 100, d = 0.9$					
0	2121	2643.33	479	5635.04	715
1	2096	11547.53	2427	61864.89	8551
2	2738	5360.98	1021	16919.78	2249
3	1990	6359.25	1231	29773.96	4085
4	2033	2798.67	543	8506.84	1045
5	2433	303.18	51	206.95	15
6	2220	396.64	105	863.69	93
7	2252	1620.30	277	3213.58	387
8	1843	774.00	187	1930.54	215
9	2043	2513.62	469	7228.53	941
avg		3413.75		13614.38	

Table C.3: Test runs on $G_{[-10,10]}$, generated by **rudy** [116].

Problem number	solution	R2 time	R2 nodes	R3 time	R3 nodes
$n = 100, d = 0.5$					
0	8190	7330.78	1465	31722.94	4557
1	8045	2401.30	459	8225.73	1121
2	8039	3576.21	717	9948.73	1339
3	8139	588.33	111	1497.10	183
4	8125	3735.51	765	15701.08	2119
5	8169	982.42	181	1946.29	251
6	8217	6042.72	1281	22252.43	3041
7	8249	1865.00	337	4107.17	499
8	8099	434.10	121	757.90	79
9	8099	1655.69	325	4971.55	629
avg		2861.21		10113.09	
$n = 100, d = 0.9$					
0	13585	3310.94	619	10167.78	1331
1	13417	3932.08	753	14050.58	1843
2	13461	1391.55	257	3502.67	433
3	13656	741.64	155	2068.90	245
4	13514	2089.08	397	4966.71	611
5	13574	2628.79	545	7200.92	891
6	13640	1980.68	355	5417.56	677
7	13501	5154.51	1007	16854.64	2231
8	13593	1506.58	299	4360.23	515
9	13658	1263.11	255	3323.02	391
avg		2399.90		7191.30	

Table C.4: Test runs on $G_{[1,10]}$, generated by `rudy` [116].

Problem number	solution	R2 time	R2 nodes	R3 time	R3 nodes
2 dimensional					
t2g10_5555	6049461	10.12 (1 node)			
t2g10_6666	5757868	15.94 (1 node)			
t2g10_7777	6509837	14.89 (1 node)			
t2g15_5555	15051133	344.47	3	304.03	3
t2g15_6666	15763716	369.64	3	359.87	3
t2g15_7777	15269399	1556.21	13	346.89	3
t2g20_5555	24838942	84289.71	195	6690.99	9
t2g20_6666	29290570			35205.95	45
t2g20_7777	28349398	53363.11	101	8092.80	11
3 dimensional					
t3g5_5555	10933215	18.01 (1 node)			
t3g5_6666	11582216	24.52 (1 node)			
t3g5_7777	11552046	26.00 (1 node)			
t3g6_5555	17434469	292.41	3	280.85	3
t3g6_6666	20217380	8849.19	185	2025.74	19
t3g6_7777	19475011	290.09	3	277.95	3
t3g7_5555	28302918	432.71 (1 node)			
t3g7_6666	33611981	550.12 (1 node)			
t3g7_7777	29118445	116550.36	567	117782.75	243

Table C.5: Test runs on torus graphs with gaussian distribution from F. Liers [personal communication, Dec. 2005].

Problem number	solution	R2 time	R2 nodes	R3 time	R3 nodes
$n = 100$					
2.5-100_5555	2460049	542.28	85	92.43	7
2.5-100_6666	2031217	502.50	55	65.65	5
2.5-100_7777	3363230	654.60	81	47.49	3
3.0-100_5555	2448189	276.21	91	96.45	7
3.0-100_6666	1984099	306.41	137	33.82	3
3.0-100_7777	3335814	80.70	5	48.16	3
$n = 150$					
2.5-150_5555	4363532	1342.15	49	265.14	7
2.5-150_6666	4057153	2940.15	167	338.94	7
2.5-150_7777	4243269	2752.18	157	559.14	11
3.0-150_5555	4279261	1932.23	157	278.02	7
3.0-150_6666	3949317	1229.10	127	235.00	7
3.0-150_7777	4211158	1852.36	147	366.80	9
$n = 200$					
2.5-200_5555	6294701	8139.05	277	604.76	7
2.5-200_6666	6795365	4518.01	73	1074.65	11
2.5-200_7777	5568272	10258.25	599	1298.48	13
3.0-200_5555	6215531	7048.29	263	607.20	9
3.0-200_6666	6756263	4806.96	115	1133.12	13
3.0-200_7777	5560824	8357.32	557	1362.60	15
$n = 250$					
2.5-250_5555	7919449	14740.00	409	10828.49	59
2.5-250_6666	6925717	7581.61	87	4623.67	27
2.5-250_7777	6596797	15868.20	739	4249.66	23
3.0-250_5555	7823791*	16351.90	739	6388.59	39
3.0-250_6666	6903351*	16529.27	526	949.19	5
3.0-250_7777	6418276*	11979.72	251	3443.85	19
$n = 300$					
2.5-300_5555	8579363	77388.27	1207	24226.96	83
2.5-300_6666	9102033	79469.93	1041	32678.08	105
2.5-300_7777	8323804	102622.88	1523	46810.17	139
3.0-300_5555	8493173*	26310.19	499	8414.36	29
3.0-300_6666	8915110*	26437.34	169	5541.51	19
3.0-300_7777	8242904*	29937.24	475	11533.05	39

Table C.6: Test runs on Ising instances from F. Liers [personal communication, Dec. 2005].

Problem				MC	R2		R3	
number	n	density	solution	offset	time	nodes	time	nodes
1a	50	10%	3414	196	10.80	3	8.19	3
2a	60	10%	6063	2368	0.93 (1 node)			
3a	70	10%	6037	1389	6.57 (1 node)			
4a	80	10%	8598	3159	4.50 (1 node)			
5a	50	20%	5737	1915	1.17 (1 node)			
6a	30	40%	3980	1318	0.27 (1 node)			
7a	30	50%	4541	1597	0.25 (1 node)			
8a	100	6.25%	11190	2329	3.85 (1 node)			
1b	20	100%	133	-17568	7.28	17	3.35	5
2b	30	100%	121	-41031	22.02	39	24.30	21
3b	40	100%	118	-74655	47.79	47	116.64	61
4b	50	100%	129	-121089	146.69	85	499.73	187
5b	60	100%	150	-175052	278.80	159	949.13	281
6b	70	100%	146	-240964	746.59	265	3553.72	757
7b	80	100%	160	-312422	1899.89	521	7995.91	1359
8b	90	100%	145	-397423	5853.37	1573	37551.34	5259
9b	100	100%	137	-493637	12887.51	3687	108811.64	13335
10b	125	100%	154	-770674	44382.63	6195		
1c	40	80%	5058	1627	0.84 (1 node)			
2c	50	60%	6213	2343	0.80 (1 node)			
3c	60	40%	6665	2517	1.90 (1 node)			
4c	70	30%	7398	2144	4.04 (1 node)			
5c	80	20%	7362	3672	3.56 (1 node)			
6c	90	10%	5824	1942	5.61 (1 node)			
7c	100	10%	7225	1824	4.81 (1 node)			

Table C.7: Test runs on (**QP**) instances given by Glover et al. [37].

Problem number	density	solution	MC offset	R2 time	R2 nodes	R3 time	R3 nodes
$n = 100$							
1d	10%	6333	1869	6.90 (1 node)			
2d	20%	6579	-2342	73.56	5	58.51	3
3d	30%	9261	-1839	163.80	13	58.83	3
4d	40%	10727	-1439	512.96	73	127.74	7
5d	50%	11626	-110	396.16	133	408.82	27
6d	60%	14207	1245	246.92	21	55.95	3
7d	70%	14476	3063	441.55	139	287.35	19
8d	80%	16352	501	218.57	17	50.72	3
9d	90%	15656	-3992	542.82	137	227.36	13
10d	100%	19102	5309	523.06	115	236.75	13
$n = 200$							
1e	10%	16464	-1217	6402.55	111	1064.70	9
2e	20%	23395	3196	6985.19	347	106660.29	877
3e	30%	25243	-4410	6274.33	239	30790.67	291
4e	40%	35594	5813	5865.77	315	21308.70	173
5e	50%	35154	6924	124428.82	4653		

Table C.8: Test runs on (**QP**) instances given by Glover et al. [37].

Problem number	solution	MC offset	R2 time	R2 nodes	R3 time	R3 nodes
$n = 100, d = 0.1$						
1	7970	-3834	10.86 (1 node)			
2	11036	1455	13.80 (1 node)			
3	12723	6069	6.43 (1 node)			
4	10368	-221	5.96 (1 node)			
5	9083	-1812	7.56 (1 node)			
6	10210	771	850.62	193	114.61	7
7	10125	707	9.68 (1 node)			
8	11435	-508	9.24 (1 node)			
9	11455	910	5.79 (1 node)			
10	12565	3846	9.99 (1 node)			
avg			92.99		19.39	
$n = 250, d = 0.1$						
1	45607	-1214	18110.09	273	9271.20	37
2	44810	5797	15864.07	187	4822.60	19
3	49037	16642	20832.16	275	4873.42	19
4	41274	-7978	18379.23	251	4522.65	17
5	47961	4665	18486.30	319	5363.81	21
6	41014	-2917	18182.37	289	52501.54	223
7	46757	8740	20056.34	369	9071.76	37
8	35726	-13323				
9	48916	12071	17179.34	297	11576.67	47
10	40442	-1657	17594.49	225	16479.80	63
avg			18298.27		13164.83	

Table C.9: Test runs on (**QP**) instances of Beasley [18].

Problem number	MC solution	R2 time	R2 nodes	R3 time	R3 nodes
$n = 100, d = 1.0$					
1	19412	664.57	129	93.43	5
2	17290	706.48	95	95.36	5
3	17565	544.56	71	85.61	5
4	19125	600.70	165	267.77	15
5	15868	535.24	101	148.81	9
6	17368	502.52	63	77.87	5
7	18629	523.12	103	122.92	9
8	18649	404.90	131	435.56	31
9	13294	456.88	149	319.53	23
10	15352	407.03	53	132.23	9
avg		524.70		177.91	
$n = 120, d = 0.3$					
1	13067	1083.60	175	423.66	15
2	13046	401.61	19	89.22	3
3	12418	696.53	43	103.24	3
4	13867	1129.83	119	195.63	7
5	11403	886.36	61	150.99	5
6	12915	259.34	13	75.39	3
7	14068	36.57 (1 node)			
8	14701	28.69 (1 node)			
9	10458	762.37	69	319.28	11
10	12201	737.44	53	194.88	7
avg		602.23		161.76	
$n = 120, d = 0.8$					
1	18691	688.17	171	2933.12	131
2	18827	818.38	153	1054.84	47
3	19302	985.86	145	521.65	21
4	20765	953.82	99	239.18	9
5	20417	926.51	81	259.37	9
6	18482	1030.43	129	505.80	21
7	22194	669.08	173	2268.45	89
8	19534	773.28	173	3642.45	167
9	18195	1111.93	179	1010.81	41
10	19049	1014.86	187	763.31	29
avg		897.23		1319.90	

Table C.10: Test runs on (**QP**) instances introduced by Billionnet and Elloumi [23].

Problem number	MC solution	R2 time	R2 nodes	R3 time	R3 nodes
$n = 150, d = 0.3$					
1	18889	2466.37	157	688.63	13
2	17816	2105.60	111	755.74	15
3	17314	1425.05	43	191.84	3
4	19884	2717.84	163	504.27	9
5	16817	2409.87	213	1836.79	37
6	16780	1857.17	219	10930.41	239
7	18001	2349.17	145	1371.85	27
8	18303	2760.79	317	16273.79	353
9	12838	2533.31	215	28383.01	681
10	17963	2003.51	237	13841.23	309
avg		2262.87		7477.76	
$n = 150, d = 0.8$					
1	27089	1925.36	221	5506.98	123
2	26779	1908.74	245	27225.34	615
3	29438	1653.92	253	17992.29	363
4	26911	2030.32	161	2456.64	51
5	28017	1729.93	249	7568.35	143
6	29221	1763.91	229	7927.27	177
7	31209	2132.61	231	24500.87	515
8	29730	1690.69	211	8194.58	167
9	25388	1759.55	255	20271.17	457
10	28374	1886.02	209	7425.42	147
avg		1848.11		12906.89	

Table C.11: Test runs on (**QP**) instances introduced by Billionnet and Elloumi [23].

Problem number	MC solution	R2 time	R2 nodes	R3 time	R3 nodes
$n = 200, d = 0.3$					
1	25453	174819.36	9609		
2	25027	19846.61	659		
3	28023	6300.07	343	44787.86	425
4	27434	7780.47	405		
5	26355	6282.42	329	171819.67	1705
6	26146	9836.04	381	89217.41	869
7	30483	7627.37	191	2695.88	23
8	27355	15607.40	553		
9	24683	37224.26	1237		
10	23842	67613.29	2273		
avg		35293.73			
$n = 200, d = 0.8$					
1	48534	5540.85	299	101078.75	911
2	40821	148515.07	5859		
3	43207	120890.63	6083		
4	43757	9838.67	423		
5	41482	41981.84	1486		
6	49492	6884.75	371	9716.35	81
7	46828	6119.02	337		
8	44502	121630.18	4947		
9	43241	6479.26	257	21990.27	191
10	42832	9519.16	435		
avg		47739.94			
$n = 250, d = 0.1$					
1	24076	12958.59	449	10466.01	45
2	22540	12833.21	319	6664.61	29
3	22923	13984.10	303	3841.13	15
4	24649	13181.81	443	7062.74	29
5	21057	16663.19	367	38850.59	181
6	22735	12531.74	271	12507.86	57
7	24095	12211.04	181	1348.28	5
8	23801	13128.19	427	31042.86	135
9	20051	12806.56	331	30602.57	139
10	23159	12648.59	461	28913.40	123
avg		13294.70		17130.01	

Table C.12: Test runs on (**QP**) instances of the type of those introduced by Billionnet and Elloumi [23], but with larger dimensions.

Bibliography

[1] Farid Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM J. Optim.*, 5(1):13–51, 1995.

[2] Noga Alon and Nabil Kahale. Approximating the independence number via the θ-function. *Math. Programming*, 80(3, Ser. A):253–264, 1998.

[3] Miguel F. Anjos and Henry Wolkowicz. Strengthened semidefinite relaxations via a second lifting for the Max-Cut problem. *Discrete Appl. Math.*, 119(1-2):79–106, 2002.

[4] Michael Armbruster, Michael L. Overton, Franz Rendl, and Angelika Wiegele. The spectral bundle method with second order information. Technical report, Alpen-Adria-Universität Klagenfurt, Universitätsstr. 65-67, 9020 Klagenfurt, Austria, 2005.

[5] W. E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9:17–29, 1951.

[6] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[7] Laura Bahiense, Nelson F. Maculan, and Claudia A. Sagastizábal. The volume algorithm revisited: relation with bundle methods. *Math. Program.*, 94(1, Ser. A):41–69, 2002.

[8] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Math. Programming*, 58 (3, Ser. A):295–324, 1993.

[9] Francisco Barahona. The max-cut problem on graphs not contractible to K_5. *Oper. Res. Lett.*, 2(3):107–111, 1983.

[10] Francisco Barahona. On cuts and matchings in planar graphs. *Math. Programming*, 60(1, Ser. A):53–68, 1993.

[11] Francisco Barahona and Ranga Anbil. The volume algorithm: producing primal solutions with a subgradient method. *Math. Program.*, 87(3, Ser. A):385–399, 2000.

[12] Francisco Barahona and László Ladányi. Branch and cut based on the volume algorithm: Steiner trees in graphs and max-cut. *RAIRO Oper. Res.*, 40(1):53–73, 2006.

[13] Francisco Barahona and Ali Ridha Mahjoub. On the cut polytope. *Math. Programming*, 36(2):157–173, 1986.

[14] Francisco Barahona, Michael Jünger, and Gerhard Reinelt. Experiments in quadratic 0-1 programming. *Math. Programming*, 44(2, (Ser. A)):127–137, 1989.

[15] Earl R. Barnes and Alan J. Hoffman. Partitioning, spectra and linear programming. In *Progress in combinatorial optimization (Waterloo, Ont., 1982)*, pages 13–25. Academic Press, Toronto, ON, 1984.

[16] John E. Beasley. Or-library: distributing test problems by electronic mail. *J. Oper. Res. Soc.*, 41(11):1069–1072, 1990.

[17] John E. Beasley. Or-library. `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`.

[18] John E. Beasley. Heuristic algorithms for the unconstrained binary quadratic programming problem. Technical report, The Management School, Imperial College, London SW7 2AZ, England, 1998.

[19] Richard Bellman and Ky Fan. On systems of linear inequalities in Hermitian matrix variables. In *Proc. Sympos. Pure Math., Vol. VII*, pages 1–11. Amer. Math. Soc., Providence, R.I., 1963.

[20] Claude Berge. Perfect graphs. In *Six papers on Graph Theory*, pages 1–21. Academic Press, Indian Statistical Institute, Calcutta, 1963.

[21] Claude Berge. Un application de la theorie des graphes à un probleme de codage. In E. R. Caianiello, editor, *Automata Theory*, pages 25–34. Academic Press, 1966.

[22] Dimitri P. Bertsekas. *Constrained optimization and Lagrange multiplier methods.* Computer Science and Applied Mathematics. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], New York, 1982.

[23] Alain Billionnet and Sourour Elloumi. Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. *Math. Programming*, 2006. to appear.

[24] Brian Borchers. CSDP, a C library for semidefinite programming. *Optim. Methods Softw.*, 11/12(1-4):613–623, 1999.

[25] Endre Boros, Peter L. Hammer, and Gabriel Tavares. The pseudo-boolean optimization website, 2005. `http://rutcor.rutgers.edu/~pbo/`.

[26] Samuel Burer and Renato D. C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Math. Program.*, 95(2, Ser. B):329–357, 2003.

[27] Samuel Burer and Dieter Vandenbussche. Solving lift-and-project relaxations of binary integer programs. *SIAM J. Optim.*, 16(3):726–750 (electronic), 2006.

[28] Maria Chudnovsky, Neil Robertson, P. D. Seymour, and Robin Thomas. Progress on perfect graphs. *Math. Program.*, 97(1-2, Ser. B):405–422, 2003.

[29] Etienne de Klerk. *Aspects of semidefinite programming*, volume 65 of *Applied Optimization*. Kluwer Academic Publishers, Dordrecht, 2002.

[30] Charles Delorme and Svatopluk Poljak. Laplacian eigenvalues and the maximum cut problem. *Math. Programming*, 62(3, Ser. A):557–574, 1993.

[31] Michel Deza and Monique Laurent. Applications of cut polyhedra. I, II. *J. Comput. Appl. Math.*, 55(2):191–216, 217–247, 1994.

[32] Michel Marie Deza and Monique Laurent. *Geometry of cuts and metrics*, volume 15 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 1997.

[33] Wilm E. Donath and Alan J. Hoffman. Lower bounds for the partitioning of graphs. *IBM J. Res. Develop.*, 17:420–425, 1973.

[34] Richard J. Duffin. Infinite programs. In *Linear inequalities and related systems*, Annals of Mathematics Studies, no. 38, pages 157–170. Princeton University Press, Princeton, N. J., 1956.

[35] 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. *Math. Program.*, 105(2-3, Ser. B):451–469, 2006.

[36] Michael R. Garey and David S. Johnson. *Computers and intractability*. W. H. Freeman and Co., San Francisco, Calif., 1979.

[37] Fred Glover, Gary Kochenberger, and Bahram Alidaee. Adaptative memory tabu search for binary quadratic programs. *Management Sci.*, 44(3):336–345, 1998.

[38] Michel X. Goemans and David P. Williamson. New $\frac{3}{4}$-approximation algorithms for the maximum satisfiability problem. *SIAM J. Discrete Math.*, 7 (4):656–666, 1994.

[39] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comput. Mach.*, 42(6):1115–1145, 1995.

[40] Édouard Goursat. *Functions of a complex variable, 2, Pt. 1.* Ginn & Co., 1916.

[41] Martin Grötschel and George L. Nemhauser. A polynomial algorithm for the max-cut problem on graphs without long odd cycles. *Math. Programming*, 29(1):28–40, 1984.

[42] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[43] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2 of *Algorithms and Combinatorics: Study and Research Texts.* Springer-Verlag, Berlin, 1988.

[44] Nebojša Gvozdenović and Monique Laurent. Approximating the chromatic number of a graph by semidefinite programming. Technical report, Centrum voor Wiskunde en Informatica, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands, 2006.

[45] Frank O. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM J. Comput.*, 4(3):221–225, 1975.

[46] Peter L. Hammer. Some network flow problems solved with pseudo-Boolean programming. *Operations Res.*, 13:388–399, 1965.

[47] Pierre Hansen. Methods of nonlinear $0-1$ programming. *Ann. Discrete Math.*, 5:53–70, 1979.

[48] Johan Håstad. Some optimal inapproximability results. In *STOC '97 (El Paso, TX)*, pages 1–10 (electronic). ACM, New York, 1999.

[49] Christoph Helmberg. Semidefinite programming website. `http://www-user.tu-chemnitz.de/~helmberg/semidef.html`.

[50] Christoph Helmberg. Fixing variables in semidefinite relaxations. *SIAM J. Matrix Anal. Appl.*, 21(3):952–969 (electronic), 2000.

[51] Christoph Helmberg. Semidefinite programming for combinatorial optimization, 2000. Habilitation thesis.

[52] Christoph Helmberg. Semidefinite programming. *European J. Oper. Res.*, 137(3):461–482, 2002.

[53] Christoph Helmberg and Francois Oustry. Bundle methods to minimize the maximum eigenvalue function. In *Handbook of semidefinite programming*, volume 27 of *Internat. Ser. Oper. Res. Management Sci.*, pages 307–337. Kluwer Acad. Publ., Boston, MA, 2000.

[54] Christoph Helmberg and Franz Rendl. Solving quadratic $(0,1)$-problems by semidefinite programs and cutting planes. *Math. Programming*, 82(3, Ser. A):291–315, 1998.

[55] Christoph Helmberg and Franz Rendl. A spectral bundle method for semidefinite programming. *SIAM J. Optim.*, 10(3):673–696 (electronic), 2000.

[56] Christoph Helmberg, Franz Rendl, Robert J. Vanderbei, and Henry Wolkowicz. An interior-point method for semidefinite programming. *SIAM J. Optim.*, 6(2):342–361, 1996.

[57] Christoph Helmberg, Krzysztof C. Kiwiel, and Franz Rendl. Incorporating inequality constraints in the spectral bundle method. In *Integer programming and combinatorial optimization (Houston, TX, 1998)*, volume 1412 of *Lecture Notes in Comput. Sci.*, pages 423–435. Springer, Berlin, 1998.

[58] Nicholas J. Higham. Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra Appl.*, 103:103–118, 1988.

[59] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex analysis and minimization algorithms. I*, volume 305 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1993.

[60] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex analysis and minimization algorithms. II*, volume 306 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1993.

[61] Roger A. Horn and Charles R. Johnson. *Matrix analysis*. Cambridge University Press, Cambridge, 1985.

[62] David S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.*, 9:256–278, 1974.

[63] David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, and Catherine Schevon. Optimization by simulated annealing: an experimental evaluation. part i, graph partitioning. *Oper. Res.*, 37(6):865–892, 1989.

[64] David Karger, Rajeev Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. *J. ACM*, 45(2):246–265, 1998.

[65] Stefan E. Karisch and Franz Rendl. Semidefinite programming and graph equipartition. In *Topics in semidefinite and interior-point methods (Toronto, ON, 1996)*, volume 18 of *Fields Inst. Commun.*, pages 77–95. Amer. Math. Soc., Providence, RI, 1998.

[66] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*, pages 85–103. Plenum, New York, 1972.

[67] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems.* Springer-Verlag, Berlin, 2004.

[68] Sunyoung Kim and Masakazu Kojima. Second order cone programming relaxation of nonconvex quadratic optimization problems. *Optim. Methods Softw.*, 15(3-4):201–224, 2001.

[69] Krzysztof C. Kiwiel. *Methods of descent for nondifferentiable optimization*, volume 1133 of *Lecture Notes in Mathematics.* Springer-Verlag, Berlin, 1985.

[70] Krzysztof C. Kiwiel. A survey of bundle methods for nondifferentiable optimization. In *Mathematical programming (Tokyo, 1988)*, volume 6 of *Math. Appl. (Japanese Ser.)*, pages 263–282. SCIPRESS, Tokyo, 1989.

[71] Krzysztof C. Kiwiel. Proximity control in bundle methods for convex nondifferentiable minimization. *Math. Programming*, 46(1, (Ser. A)):105–122, 1990.

[72] Donald E. Knuth. The sandwich theorem. *Electron. J. Combin.*, 1:Article 1, approx. 48 pp. (electronic), 1994.

[73] Michal Kocvara and Michael Stingl. On the solution of large-scale SDP problems by the modified barrier method using iterative solvers. *Math. Programming.* Published online: September 2006.

[74] Masakazu Kojima, Susumu Shindoh, and Shinji Hara. Interior-point methods for the monotone semidefinite linear complementarity problem in symmetric matrices. *SIAM J. Optim.*, 7(1):86–125, 1997.

[75] Peter Lancaster. On eigenvalues of matrices dependent on a parameter. *Numer. Math.*, 6:377–387, 1964.

[76] Jean B. Lasserre. An explicit equivalent positive semidefinite program for nonlinear 0-1 programs. *SIAM J. Optim.*, 12(3):756–769 (electronic), 2002.

[77] Monique Laurent. A comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0-1 programming. *Math. Oper. Res.*, 28(3): 470–496, 2003.

[78] Monique Laurent and Svatopluk Poljak. On a positive semidefinite relaxation of the cut polytope. *Linear Algebra Appl.*, 223/224:439–461, 1995.

[79] Monique Laurent and Svatopluk Poljak. On the facial structure of the set of correlation matrices. *SIAM J. Matrix Anal. Appl.*, 17(3):530–547, 1996.

[80] Monique Laurent and Franz Rendl. Semidefinite programming and integer programming. In K. Aardal, G.L. Nemhauser, and R. Weismantel, editors, *Discrete Optimization*, volume 12 of *Handbooks in Operations Research and Management Science*, chapter 8, pages 393–514. Elsevier, Amsterdam, The Netherlands, 2005.

[81] Richard B. Lehoucq, Danny C. Sorensen, and Chao Yang. *ARPACK users' guide*, volume 6 of *Software, Environments, and Tools*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998.

[82] Claude Lemarechal. Bundle methods in nonsmooth optimization. In *Nonsmooth optimization (Proc. IIASA Workshop, Laxenburg, 1977)*, volume 3 of *IIASA Proc. Ser.*, pages 79–102. Pergamon, Oxford, 1978.

[83] Claude Lemaréchal, Arkadii Nemirovskii, and Yurii Nesterov. New variants of bundle methods. *Math. Programming*, 69(1, Ser. B):111–147, 1995.

[84] Thomas Lengauer. *Combinatorial algorithms for integrated circuit layout.* Applicable Theory in Computer Science. John Wiley & Sons Ltd., Chichester, 1990.

[85] Adrian S. Lewis and Michael L. Overton. Eigenvalue optimization. In *Acta numerica, 1996*, volume 5 of *Acta Numer.*, pages 149–190. Cambridge Univ. Press, Cambridge, 1996.

[86] Frauke Liers. *Contributions to Determining Exact Ground-States of Ising Spin-Glasses and to their Physics.* PhD thesis, Universität zu Köln, 2004.

[87] Frauke Liers, Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi. *Computing Exact Ground States of Hard Ising Spin Glass Problems by Branch-and-Cut*, pages 47–68. Wiley, 2004.

[88] László Lovász. On the Shannon capacity of a graph. *IEEE Trans. Inform. Theory*, 25(1):1–7, 1979.

[89] László Lovász and Alexander Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM J. Optim.*, 1(2):166–190, 1991.

[90] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *J. Assoc. Comput. Mach.*, 41(5):960–981, 1994.

[91] Marko M. Mäkelä. Survey of bundle methods for nonsmooth optimization. *Optim. Methods Softw.*, 17(1):1–29, 2002.

[92] Jérôme Malick. A dual approach to semidefinite least-squares problems. *SIAM J. Matrix Anal. Appl.*, 26(1):272–284 (electronic), 2004.

[93] Jérôme Malick, Janez Povh, Franz Rendl, and Angelika Wiegele. Proximal methods for semidefinite programming. Technical report, Alpen-Adria-Universität Klagenfurt, Universitätsstr. 65-67, 9020 Klagenfurt, Austria, 2006.

[94] Hans Mittelmann. Benchmarks for optimization software. `http://plato.asu.edu/bench.html`.

[95] Renato D. C. Monteiro. Primal-dual path-following algorithms for semidefinite programming. *SIAM J. Optim.*, 7(3):663–678, 1997.

[96] Renato D. C. Monteiro and Michael J. Todd. Path-following methods. In *Handbook of semidefinite programming*, volume 27 of *Internat. Ser. Oper. Res. Management Sci.*, pages 267–306. Kluwer Acad. Publ., Boston, MA, 2000.

[97] Masakazu Muramatsu and Tsunehiro Suzuki. A new second-order cone programming relaxation for MAX-CUT problems. *J. Oper. Res. Soc. Japan*, 46(2):164–177, 2003.

[98] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13 of *SIAM Studies in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1994.

[99] G. I. Orlova and Ya. G. Dorfman. Finding the maximum cut in a graph. *Izv. Akad. Nauk SSSR Tehn. Kibernet.*, (3):155–159, 1972.

[100] Michael L. Overton. Large-scale optimization of eigenvalues. *SIAM J. Optim.*, 2(1):88–120, 1992.

[101] Michael L. Overton and Robert S. Womersley. Second derivatives for optimizing eigenvalues of symmetric matrices. *SIAM J. Matrix Anal. Appl.*, 16(3):697–718, 1995.

[102] Panos M. Pardalos and Gregory P. Rodgers. Computational aspects of a branch and bound algorithm for quadratic zero-one programming. *Computing*, 45(2):131–144, 1990.

[103] Panos M. Pardalos and Gregory P. Rodgers. Parallel branch and bound algorithms for quadratic zero-one programs on the hypercube architecture. *Ann. Oper. Res.*, 22(1-4):271–292, 1990.

[104] Beresford N. Parlett. *The symmetric eigenvalue problem*, volume 20 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998.

[105] Svatopluk Poljak. Polyhedral and eigenvalue approximations of the max-cut problem. In *Sets, graphs and numbers (Budapest, 1991)*, volume 60 of *Colloq. Math. Soc. János Bolyai*, pages 569–581. North-Holland, Amsterdam, 1992.

[106] Svatopluk Poljak and Franz Rendl. Solving the max-cut problem using eigenvalues. *Discrete Appl. Math.*, 62(1-3):249–278, 1995.

[107] Svatopluk Poljak and Franz Rendl. Nonpolyhedral relaxations of graph-bisection problems. *SIAM J. Optim.*, 5(3):467–487, 1995.

[108] Svatopluk Poljak and Zsolt Tuza. The expected relative error of the polyhedral approximation of the max-cut problem. *Oper. Res. Lett.*, 16(4): 191–198, 1994.

[109] Svatopluk Poljak and Zsolt Tuza. Maximum cuts and large bipartite subgraphs. In *Combinatorial optimization (New Brunswick, NJ, 1992–1993)*, volume 20 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 181–244. Amer. Math. Soc., Providence, RI, 1995.

[110] Janez Povh, Franz Rendl, and Angelika Wiegele. A Boundary Point Method to solve Semidefinite Programs. *Computing*, to appear.

[111] Franz Rendl. Semidefinite programming and combinatorial optimization. *Appl. Numer. Math.*, 29(3):255–281, 1999.

[112] Franz Rendl and Renata Sotirov. Bounds for the quadratic assignment problem using the bundle method. *Math. Programming*. Published online: September 2006.

[113] Franz Rendl and Angelika Wiegele. Semidefinite relaxations for sparse max-cut problems. Technical report, Alpen-Adria-Universität Klagenfurt, Universitätsstr. 65-67, 9020 Klagenfurt, Austria, 2005.

[114] Franz Rendl and Henry Wolkowicz. A projection technique for partitioning the nodes of a graph. *Ann. Oper. Res.*, 58:155–179, 1995.

[115] Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele. SDP based Branch & Bound for Max-Cut. Technical report, Alpen-Adria-Universität Klagenfurt, Universitätsstr. 65-67, 9020 Klagenfurt, Austria, 2006.

[116] Giovanni Rinaldi. Rudy. `http://www-user.tu-chemnitz.de/~helmberg/rudy.tar.gz`.

[117] R. Tyrrell Rockafellar. *Convex analysis.* Princeton Mathematical Series, No. 28. Princeton University Press, Princeton, N.J., 1970.

[118] Helga Schramm and Jochem Zowe. A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results. *SIAM J. Optim.*, 2(1):121–152, 1992.

[119] Alexander Schrijver. *Combinatorial optimization. Polyhedra and efficiency. Vol. A,B,C*, volume 24 of *Algorithms and Combinatorics.* Springer-Verlag, Berlin, 2003.

[120] Alexander Schrijver. On the history of combinatorial optimization (till 1960). In K. Aardal, G.L. Nemhauser, and R. Weismantel, editors, *Discrete Optimization*, volume 12 of *Handbooks in Operations Research and Management Science*, chapter 1, pages 1–68. Elsevier, Amsterdam, The Netherlands, 2005.

[121] Hanif D. Sherali and Warren P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Discrete Math.*, 3(3):411–430, 1990.

[122] Jos F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optim. Methods Softw.OPTMS*, 11-12:625–653, 1999.

[123] Michael J. Todd. A study of search directions in primal-dual interior-point methods for semidefinite programming. *Optim. Methods Softw.*, 11/12(1-4): 1–46, 1999.

[124] Kim-Chuan Toh. Solving large scale semidefinite programs via an iterative solver on the augmented systems. *SIAM J. Optim.*, 14(3):670–698 (electronic), 2003.

[125] Kim-Chuan Toh, Michael J. Todd, and Reha H. Tütüncü. SDPT3—a MATLAB software package for semidefinite programming, version 1.3. *Optim. Methods Softw.*, 11/12(1-4):545–581, 1999.

[126] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM Rev.*, 38(1):49–95, 1996.

[127] Henry Wolkowicz and Qing Zhao. Semidefinite programming relaxations for the graph partitioning problem. *Discrete Appl. Math.*, 96/97:461–479, 1999.

Index