An Interior Point Method for Semidefinite Programming and Max–Cut Bounds

Christoph Helmberg

Dissertation
zur Erlangung des Titels eines Doktors
der technischen Wissenschaften
eingereicht an der Technisch-
Naturwissenschaftlichen Fakultät
der Technischen Universität Graz

Graz, im Oktober 1994

Betreuer:
Doz. Dr. Franz Rendl

Adresse:
Institut für Mathematik B,
Kopernikusgasse 24
A-8010 Graz

To my parents.

Abstract

The presentation is divided into two parts. In Part I we introduce a primal–dual path–following interior point method for positive semidefinite programming based on the linearization of $ZX - \mu I$. This linearization leads to unsymmetric updates ΔX for X but by using the symmetric part of ΔX only, convergence can be guaranteed. The algorithm is well suited for the predictor–corrector approach proposed by Mehrotra [48] and is very efficient in practice.

In Part II we employ the code from Part I for computing upper bounds for the max–cut problem. In particular we combine the eigenvalue upper bound based on the maximal eigenvalue of the Laplacian of the graph with the polyhedral upper bound delivered by triangle inequalities. We also investigate the effect of larger clique inequalities. The bounds show considerable improvements to previous results on complete graphs. For the traditional approach the relaxation over the triangle inequalities was computationally too expensive for complete graphs on more than 30 nodes. We can now compute the intersection over both bounds for $n = 50$ within reasonable time and have also computed examples of size $n = 100$.

Ich danke Herrn Doz. Dr. Franz Rendl für die Betreuung der Dissertation und insbesondere für seine Bereitschaft, jederzeit meine Fragen ausführlich mit mir zu besprechen. Ich danke Professor Dr. Rainer E. Burkard, der mit großem persönlichen Einsatz hervorragende Arbeitsbedingungen geschaffen und gesichert hat.

Als Angestellter des Christian Doppler Laboratoriums für Diskrete Optimierung danke ich der Christian Doppler Gesellschaft, die diese Arbeit finanziell ermöglicht hat.

Contents

Part I

An Interior Point Method for Positive Semidefinite Programming

Chapter 1

Introduction to Part I

The interior point method is a general concept for transforming constrained into unconstrained optimization problems. The method is already known for a long time [25], the SUMT–code (Sequential Unconstrained Minimization Technique) developed by Fiacco and McCormick to solve problems by this method is still very well known. Yet the approach was only moderately successful in the beginning. Probably this is due to the fact that in general computers were not yet powerful enough to experiment with the method. The final breakthrough came with the paper of Karmarkar [35]. Karmarkar used the approach to solve linear programming problems and, what is probably most important, was able to prove polynomial convergence of his algorithm. His claim that in practice it was also faster than the Simplex Algorithm was and is not generally agreed to but helped to rise interest for the topic. In the next few years several different approaches for solving linear programming by interior point methods were developed, see e.g. [51, 38, 1, 52, 54, 68, 45], and indeed became more and more useful for real world applications.

At the same time several research groups discovered that nonlinear approaches might lead to interesting results for combinatorial optimization problems. In graph theory connections between the eigenvalues of the adjacency matrix or the Laplacian and certain properties of the underlying graph have already been examined for some time, see e.g. [15, 14, 50]. Not very surprisingly the extremal eigenvalues are instrumental. It was observed that by allowing some extra freedom for certain matrix elements it was possible to strengthen the information gained from the eigenvalues [43, 17]. This lead to minimax eigenvalue problems, a field of mathematical interest in its own right [34, 56, 5]. Another combinatorial field is $\{0,1\}$–programming. Here it was tried to formulate the $\{0,1\}$–constraints either by quadratic equations or to derive some additional constraints from other quadratic models, see e.g. [44]. Only recently it was observed that there is indeed a close connection between the eigenvalue and the $\{0,1\}$–approach. This connection is semidefinite programming [58, 2].

Semidefinite programming is linear programming over semidefinite variables. The success of interior point techniques for linear programming is very much based on the properties of the barrier function $\log x$ for non negative variables x. It is well known that $\log\det X$ displays the same properties for semidefinite matrices X and it is only natural to develop interior point methods for semidefinite programming. Several approaches can already be found in the literature [55, 34, 2, 70, 3, 64, 32], we will give a short survey in Section 1.2. The algorithm proposed here is based on the algorithm in [32] (very recently Kojima, Shindoh and Hara independently proposed the same type of algorithm for linear complementarity problems in semidefinite variables [39]). However, the setting is chosen such that linear programming is trivially included and the algorithm is explained in more detail.

In particular we will look at the primal–dual semidefinite programming pair,

$$\begin{array}{llll}
\text{(SDP)} & \text{Maximize} & \langle C, X\rangle + \langle b, s\rangle \\
& \text{subject to} & a - \mathcal{A}(X) - \mathcal{B}(s) = 0 \\
& & X \succeq 0, s \geq 0
\end{array}
\qquad
\begin{array}{llll}
\text{(DSDP)} & \text{Minimize} & \langle a, y\rangle \\
& \text{subject to} & Z + C - \mathcal{A}^T(y) = 0 \\
& & t + b - \mathcal{B}^T(y) = 0 \\
& & Z \succeq 0, t \geq 0.
\end{array}$$

The algorithm presented is a primal–dual path–following method based on the linearization of $ZX - \mu I$. The linearization is very successful in linear programming but leads to unsymmetric updates for X in semidefinite programming. However, it will be shown that by using the symmetric part of ΔX only, the step direction is a descent direction with respect to an appropriately defined merit function. Under the assumption of compactness of the primal problem and an "interior" starting point convergence to the optimal solution can be guaranteed. The algorithm is also well suited for Mehrotra's [48, 49] predictor–corrector approach and practical experiments were very successful.

For an outline of Part I this Chapter will deal with general concepts and short surveys on previous work. In Section 1.1 the duality aspects are investigated, Section 1.2 deals with interior point methods in general. In Chapter 2 the algorithm is developed whose convergence is shown in Chapter 3. The last Chapter of this part deals with a number of practical aspects and outlines some further research possibilities.

Notation

We denote the set of symmetric $n \times n$ matrices by $\mathcal{M}_n$. The inequality symbol $\succeq$ refers to the Löwner partial order induced by the cone of positive semidefinite matrices, i.e. for two matrices A and B in $\mathcal{M}_n$ $A \succeq B$ ($A \succ B$) whenever $A - B$ is positive semidefinite (positive definite). For (SDP) and (DSDP) $C \in \mathcal{M}_n$, $b \in \Re^m$, and $a \in \Re^k$ are the primal and dual cost coefficients, $X \in \mathcal{M}_n$ and $s \in \Re^m$ are the primal variables, $Z \in \mathcal{M}_n$, $y \in \Re^k$, and $t \in \Re^m$ are the dual variables, $\mathcal{A}(\cdot) : \mathcal{M}_n \to \Re^k$ and $\mathcal{B}(\cdot) : \Re^m \to \Re^k$ are linear operators representing the primal constraint coefficients, $\mathcal{A}^T(\cdot) : \Re^k \to \mathcal{M}_n$ and $\mathcal{B}^T(\cdot) : \Re^k \to \Re^m$ are the corresponding adjoint operators satisfying the adjoint identity

$$\langle \mathcal{A}(X), y\rangle = \left\langle X, \mathcal{A}^T(y)\right\rangle \qquad \text{and} \qquad \langle \mathcal{B}(s), y\rangle = \left\langle s, \mathcal{B}^T(y)\right\rangle$$

for all $X \in \mathcal{M}_n$, $s \in \Re^m$, and $y \in \Re^k$. As usual $\langle\cdot,\cdot\rangle$ denotes the inner product of the appropriate space, i.e. $\langle A, B\rangle = \mathrm{tr}(A^T B)$ for arbitrary $n \times m$ matrices A and B, and $\langle a, b\rangle = a^T b$ for arbitrary n–vectors a and b.

For any convex cone S in $\Re^n$, we let S^* denote the dual cone, i.e.,

$$S^* = \{y \in \Re^n |\; x^T y \geq 0 \quad \text{for all } x \in S\}$$

$A \circ B$ will denote the Hadamard product of two matrices (or vectors) and, in a slight abuse of notation, given a real valued function f defined on a real domain and a vector $v \in \Re^n$, we will occasionally write $f(v)$ for $(f(v_1), \ldots, f(v_n))^T$. Here, f will usually be the logarithm $\log(\cdot)$ or a power function. For $x \leq 0$ we set $\log(x) := -\infty$. I will denote the identity matrix and e the vector of all ones. If not stated explicitly the dimensions of I and e will be clear from context.

1.1 Duality

It is well known that duality is a key concept in convex programming. The basic questions of duality in convex programming were settled in the mid sixties, see e.g. [23]. In our case the object of interest is the cone of positive semidefinite matrices. Duality over cones is thoroughly studied in [72], a specialized treatment for the cone of semidefinite matrices can be found in [2]. For the general framework of [72] let S and T denote two closed convex cones in $\Re^n$ and $\Re^m$, respectively. Let $c \in \Re^n$, $a \in \Re^m$ and the $m \times n$ matrix A be given. The two optimization problems investigated are

$$\begin{array}{llll} \text{(P)} & \text{Maximize} & c^T x \\ & \text{subject to} & Ax - a \in S \\ & & x \in T, \end{array} \qquad \begin{array}{lll} \text{(D)} & \text{Minimize} & a^T y \\ & \text{subject to} & c - A^T y \in T^* \\ & & y \in S^*. \end{array}$$

It is shown in [72] that if both problems have feasible solutions and a so called constraint qualification holds then (P) and (D) form a primal–dual pair of problems and strong duality holds, i.e. the minimum attained in (P) coincides with the maximum attained in

(D). If these conditions are not satisfied the optimum may not be attained or a gap may remain between primal and dual optimal values. Examples can be found in the survey of Vandenberghe and Boyd on "Positive Definite Programming" [69].

For our purposes x will contain the primal variables of (SDP), the positive semidefinite matrix X and the vector $s \geq 0$. It is well known that the nonnegative orthant $\Re^m_+$ is self dual in $\Re^m$ and that the cone of positive semidefinite matrices is self dual in $\mathcal{M}_n$. The $\binom{n+1}{2}$ dimensional vector space $\mathcal{M}_n$ can be identified with $\Re^{\binom{n+1}{2}}$. Thus we let T denote the $\binom{n+1}{2} + m$ dimensional cone composed of the positive semidefinite $n \times n$ matrices and the nonnegative m–vectors. Obviously T is self dual as well ($T = T^*$), S is the trivial k–dimensional cone 0, and the dual cone to S is $S^* = \Re^k$. By denoting the first $\binom{n+1}{2}$ columns of A by $\mathcal{A}(\cdot)$, the last m columns by $\mathcal{B}(\cdot)$, and by changing the sign (P) is precisely the primal problem (SDP) and for (D) we have

$$\begin{array}{rl} \text{Minimize} & \langle a, y \rangle \\ \text{subject to} & \mathcal{A}^T(y) - C \succeq 0 \\ & \mathcal{B}^T(y) - b \geq 0 \\ & y \in \Re^k. \end{array} \tag{1.1}$$

By introducing slack variables Z and t we get (DSDP).

For a primal feasible point (s, X) and a dual feasible point (t, y, Z) weak duality is satisfied regardless of any constraint qualification. Although this result is trivial we give it here because we will need the formula for the gap between a primal and a dual feasible solution later on,

$$\langle a, y \rangle - \langle C, X \rangle - \langle b, s \rangle = \tag{1.2}$$
$$= \langle \mathcal{A}(X) + \mathcal{B}(s), y \rangle - \left\langle \mathcal{A}^T(y) - Z, X \right\rangle - \left\langle \mathcal{B}^T(y) - t, s \right\rangle \tag{1.3}$$
$$= \langle \mathcal{A}(X), y \rangle + \langle \mathcal{B}(s), y \rangle - \langle y, \mathcal{A}(X) \rangle + \langle Z, X \rangle - \langle y, \mathcal{B}(s) \rangle + \langle t, s \rangle \tag{1.4}$$
$$= \langle Z, X \rangle + \langle t, s \rangle \geq 0. \tag{1.5}$$

A more intuitive understanding of the nature of duality can be gained by looking at the Lagrangian of (SDP). We introduce Lagrange multipliers $y \in \Re^k$ for the equality constraints and see that

$$\max_{X \succeq 0, s \geq 0} \min_{y \in \Re^k} \langle C, X \rangle + \langle b, s \rangle + \langle y, a - \mathcal{A}(X) - \mathcal{B}(s) \rangle \leq$$
$$\leq \min_{y \in \Re^k} \max_{X \succeq 0, s \geq 0} \langle a, y \rangle + \left\langle C - \mathcal{A}^T(y), X \right\rangle + \left\langle b - \mathcal{B}^T(y), s \right\rangle.$$

Obviously X and s now play the role of Lagrange multipliers in the second problem. Since $X \succeq 0$ and $s \geq 0$ the inner maximization is bounded from above only if

$$C - \mathcal{A}^T(y) \preceq 0 \qquad \text{and} \qquad b - \mathcal{B}^T(y) \leq 0,$$

in which case the inner maximization must yield

$$\left\langle C - \mathcal{A}^T(y), X \right\rangle = 0 \qquad \text{and} \qquad \left\langle b - \mathcal{B}^T(y), s \right\rangle = 0.$$

Thus complementarity holds. Extracting the new problem from the second Lagrangian we get (1.1) once more. As before we have to assume some kind of constraint qualification to be sure that the duality gap from interchanging max and min is zero.

1.2 Interior Point Methods

Since the publication of Fiacco and McCormick's classical book on Sequential Unconstrained Minimization Techniques [25] interior point methods have been a standard method in nonlinear optimization for transforming constrained into unconstrained optimization problems. For the reader not acquainted with these techniques we give a short intuitive explanation of the main ideas.

For some standard minimization problem we replace each inequality constraint by a so called barrier function, penalizing points the more the closer they are to the specific boundary. For a sequence of points approaching the boundary the penalty will be steadily increasing and will tend to infinity on the boundary itself. This new unconstrained minimization problem is referred to as the barrier problem. Typically an interior point algorithm starts with a point somewhere in the middle, the interior, of the feasible region. It computes a descent direction with respect to the cost and penalty functions, and determines the next point by means of a line search. During this line search the barrier functions guarantee that the iterates will always stay within the interior. However, optimal solutions of the constrained problem may and will in general be located on the boundary of the feasible region. Although it is impossible for an interior point algorithm to hit the boundary itself we can get a sequence of points converging to the optimal solution by solving a sequence of barrier problems. Whenever the current iterate is "close" to the optimal solution of the current barrier problem, the influence of the barrier functions is diminished and the next barrier problem is solved with the current iterate as starting point. The algorithm is stopped whenever the current iterate is "close enough" to the optimal solution of the original problem. The parameter controlling the fading out of the barrier functions is usually called the barrier parameter.

To underline the close relationship between linear and positive semidefinite interior point methods we sketch a linear programming interior point algorithm. We start with the following primal dual pair of linear programs

$$\begin{array}{llll} \text{(P)} & \text{Maximize} & c^T x & \\ & \text{subject to} & Ax = b & \\ & & x \geq 0, & \end{array} \qquad \begin{array}{lll} \text{(D)} & \text{Minimize} & b^T y \\ & \text{subject to} & z + c - A^T y = 0 \\ & & z \geq 0. \end{array}$$

For deriving an interior point method we start by adding a barrier function for $z \geq 0$ to (D).

$$\begin{array}{ll} \text{Minimize} & b^T y - \mu e^T \log(z) \\ \text{subject to} & z + c - A^T y = 0. \end{array}$$

$\mu > 0$ is the barrier parameter. Now we form the Lagrangian of this barrier problem by introducing x as Lagrange multipliers for the equality constraints,

$$\min_{y,z} \max_{x} \; b^T y - \mu e^T \log(z) + (z + c - A^T y)^T x.$$

The optimal solution to the barrier problem is the saddle point of the Lagrangian, which is characterized by the Karush–Kuhn–Tucker (KKT) conditions.

$$\begin{aligned} Ax - b &= 0 \\ z + c - A^T y &= 0 \\ z \circ x - \mu e &= 0. \end{aligned}$$

This system is solved using Newton's method. The linearization of $z \circ x - \mu e$ yields currently the best computational results, for a theoretical discussion of the algorithm the reader is referred to [52, 53], for implementational experience to [45, 46]. In case of a finite optimal solution there is a unique solution (x_μ, y_μ, z_μ) of the KKT–conditions for each $\mu > 0$. The union of all these μ–optimal solutions forms the so called central path or central trajectory. An interior point algorithm is called a path–following algorithm if it tries to stay close to this central path. For iterates close to the central path the Newton direction is of very high quality. The central path is continuously differentiable and for $\mu \to 0$ the points (x_μ, y_μ, z_μ) converge to an optimal solution (x^*, y^*, z^*) of the original problem. Because $z_\mu \circ x_\mu = \mu e$ on the central path, the optimal solution at $\mu = 0$ satisfies strict coplementarity [47]. Although not all interior point algorithms trace the central path explicitly the step direction always contains a centering direction towards the central path implicitly.

Nesterov and Nemirovskii [55] observed that there is a class of functions, called strongly self concordant functions, for which Newton's method works especially well. For convex sets having a strongly self concordant barrier function which can be computed efficiently, Newton's method yields a polynomial time interior point algorithm. For non negative variables $x \geq 0$ the logarithm $\log(x)$ is a strongly self concordant barrier function. For $X \succeq 0$ a positive semidefinite variable, $\log\det X$ is a strongly self concordant barrier function and consequently there exist polynomial time interior point algorithms for solving semidefinite programs. They also give such algorithms explicitly in [55] but these seem to be mainly of academic value. At the same time Jarre developed a polynomial time algorithm already based on the notion of self concordance [34]. This algorithm is designed for solving the dual problem only. Also at the same time but independently Alizadeh noticed that several existing interior point algorithms for linear programming can be transformed to algorithms for semidefinite programming in a generic manner [2]. This is explicitly done for a projective method of Ye yielding an algorithm requiring the solution of an $n^2 \times n^2$ system in each iteration. In [64] Rendl, Vanderbei, and Wolkowicz study the linearizations of $Z - \mu X^{-1}$ and $X - \mu Z^{-1}$. The latter turns out to be equivalent to Jarre's algorithm in a primal–dual setting. Helmberg, Rendl, Vanderbei, and Wolkowicz [32] then use the linearization of $ZX - \mu I$ to derive a new algorithm, which is also the object of interest here. The algorithm is easy to implement and requires only the factorization of an $m \times m$ matrix per iteration. Obviously independently, the same algorithm is investigated by Kojima, Shindoh, and Hara in [39] for the more general complementarity problem. They prove polynomial running time of the algorithm if the initial point is close to the central trajectory. They also introduce two new potential reduction algorithms based on the same linearization. The paper has some important implications also for our work and we present a more detailed survey in Section 4.4. Another potential reduction algorithm was developed by Vandenberghe and Boyd [69]. Recently Alizadeh, Haberly, and Overton [3] studied the linearization of $ZX + XZ - \mu I$ which either leads to large systems because of Kronecker products or requires the solution of computationally expensive eigenvalue equations in each iteration. They provide some evidence that the additional cost might lead to a better numerical behavior of the algorithm.

1.3 Applications

Several applications of semidefinite programming can be found in the survey paper [69] and in [55]. They range widely over control theory, norm minization, geometrical optimization, $\{0,1\}$–programming, etc.. To provide some evidence of the importance of semidefinite programming we will list just a few selected problems.

1.3.1 Eigenvalue Optimization

Let there be given an affine matrix function $A(y) := A_0 + \sum_{i=1}^{m} y_i A_i$ with the $A_i \in \mathcal{M}_n$ being symmetric matrices. We want to minimize the maximal eigenvalue $\lambda_{max}(A(y))$ over $y \in \Re^m$. This is easily formulated as a (DSDP) by introducing an additional variable λ representing the maximal eigenvalue of $A(x)$,

$$\begin{array}{ll} \text{Minimize} & \lambda \\ \text{subject to} & \lambda I - A(y) \succeq 0 \end{array}$$

An inherent property of this problem is that the largest eigenvalue tends to cluster [57]. Therefore the maximal eigenvalue is a nondifferentiable objective function. Traditional optimization approaches required the knowledge of the multiplicity of the largest eigenvalue in the optimal solution [56]. For interior point methods eigenvalue clustering is not of importance.

Similarly we can minimize the spectral norm $||A(y)||$ for not necessarily symmetric matrices $A_i \in \Re^{p\times q}$. By making use of the Schur complement this is formulated as a (DSDP) as follows,

$$\begin{array}{ll} \text{Minimize} & \lambda \\ \text{subject to} & \begin{pmatrix} \lambda I & A(y) \\ A^T(y) & \lambda I \end{pmatrix} \succeq 0. \end{array}$$

Applications for eigenvalue optimization problems can be found in [56].

1.3.2 Logarithmic Chebychev Approximation

In many applications the logarithmic scale is preferred to the linear. In this case we want to find a good y such that $\log(Ay) \approx \log(b)$ instead of $Ay \approx b$. Logarithmic Chebychev approximation asks for the solution y of

$$\text{Minimize} \quad ||\log(Ay) - \log(b)||_\infty.$$

We assume $b > 0$ and for $x \leq 0$ we define $\log(x) := -\infty$. Let A be a $p \times k$ matrix and let a_i denote its rows. Assuming $a_i^T y > 0$ we have

$$|\log(a_i^T y) - \log(b_i)| = \log\max(a_i^T y/b_i, b_i/a_i^T y).$$

So the problem can be reformulated as

$$\begin{array}{lll} \text{Minimize} & t & \\ \text{subject to} & 1/t \leq a_i^T y/b_i \leq t & i = 1, \ldots, p, \end{array}$$

which is equivalent to the semidefinite program

$$\begin{array}{ll} \text{Minimize} & t \\ \text{subject to} & \begin{pmatrix} t - a_i^T y/b_i & 0 & 0 \\ 0 & a_i^T y/b_i & 1 \\ 0 & 1 & t \end{pmatrix} \succeq 0 \qquad i = 1, \ldots, p. \end{array}$$

As Vandenberghe and Boyd pointed out, this example nicely illustrates that semidefinite programming is much more general than linear programming and that it includes several problems which show no resemblance to (SDP) or (DSDP).

1.3.3 Quadratically Constrained Quadratic Programming

Once more the Schur complement can be used to transform a quadratic constraint

$$f_i(y) := (A_i y + b)^T (A_i y + b_i) - c_i^T y - d_i \leq 0$$

into a semidefinite constraint

$$\begin{pmatrix} I & A_i y + b_i \\ (A_i y + b_i)^T & c_i^T y + d_i \end{pmatrix} \succeq 0.$$

A quadratically constrained quadratic program of the form

$$\begin{array}{ll} \text{Minimize} & f_0(y) \\ \text{subject to} & f_i(y) \leq 0 \qquad i = 1, \ldots, q \end{array}$$

is therefore equivalent to the semidefinite program

$$\begin{array}{ll} \text{Minimize} & t \\ \text{subject to} & \begin{pmatrix} I & A_0 y + b_0 \\ (A_0 y + b_0)^T & c_0^T y + d_0 + t \end{pmatrix} \succeq 0 \\ & \begin{pmatrix} I & A_i y + b_i \\ (A_i y + b_i)^T & c_i^T y + d_i \end{pmatrix} \succeq 0 \qquad i = 1, \ldots, q. \end{array}$$

1.3.4 Combinatorial Optimization

Semidefinite programming can be used to compute bounds of high quality for $\{0,1\}$–programming problems. The historically first bound of this type is the Lovász ϑ–function [43]. For an undirected unweighted graph $G(V, E)$ with vertex set $V = \{1, \ldots, n\}$ and edge set E the ϑ–function is defined as follows,

$$\vartheta(G) = \min\{\lambda_{max}(A) | \; A \in \mathcal{M}_n, \operatorname{diag}(A) = e, \{i,j\} \notin E \Longrightarrow a_{ij} = 1\}.$$

Let $\overline{G}$ denote the complement of the graph G, $\omega(G)$ the size of the largest clique of G, and $\chi(G)$ the chromatic number of G. It is NP–complete to compute the latter two numbers. The importance of the ϑ–function is due to the fact that

$$\omega(G) \leq \vartheta(\overline{G}) \leq \chi(G).$$

A very recent survey on the ϑ–function can be found in [37].

The next important bound was the φ–function of Delorme and Poljak [17] giving an upper bound on the size of a maximum cut of a graph. In its original form is was formulated as an eigenvalue optimization problem. Today the corresponding dual program can also be used to find very good feasible solutions. This bound is discussed in detail in Part II.

Recently quadratic modeling approaches for general $\{0,1\}$–programs find increasing interest [44, 60]. These lead to relaxations based on semidefinite programming.

Chapter 2

Interior Point Algorithm

In this chapter we will develop a primal–dual path–following interior point method that solves (SDP) and (PSDP) simultaneously. In linear programming this approach was first proposed by [47] and implemented in [38].

For the algorithm we have to apply operator $\mathcal{A}$ not only to symmetric matrices, but to arbitrary $n \times n$ square matrices. We extend the definition of $\mathcal{A}$ by mapping the skew–symmetric part to zero. This implies for an arbitrary $n \times n$ matrix M

$$\mathcal{A}(M) = \mathcal{A}(M^T). \tag{2.1}$$

A well known technique for deriving interior point algorithms is to start with the barrier problem associated with (DSDP), it will be called the dual barrier problem:

$$\begin{array}{lll} \text{(DBP)} & \text{Minimize} & \langle a, y\rangle - \mu(\log\det Z + e^T \log t) \\ & \text{subject to} & Z + C - \mathcal{A}^T(y) = 0 \\ & & t + b - \mathcal{B}^T(y) = 0. \end{array} \tag{2.2}$$

The factor μ is a positive real number called the barrier parameter. For later use we will denote the objective function of (2.2) by

$$f_{D;\mu}(t, y, Z) := \langle a, y\rangle - \mu(\log\det Z + e^T \log t) \tag{2.3}$$

To switch from constrained to unconstrained optimization we look at the Lagrangian corresponding to (DBP) for some fixed μ.

$$L_\mu(s, t, X, y, Z) = \langle a, y\rangle - \mu(\log\det Z + e^T \log t) + \big\langle Z + C - \mathcal{A}^T(y), X\big\rangle + \big\langle t + b - \mathcal{B}^T(y), s\big\rangle \tag{2.4}$$

The saddle point of the Lagrangian is given by the solution of the KKT–conditions. They are obtained by using the adjoint identity for $\mathcal{A}$ and $\mathcal{B}$ and $\frac{d}{dZ}\log\det Z = Z^{-1}$ (for the latter result see e.g. [30]).

$$
\begin{aligned}
\nabla_X L_\mu &= Z + C - \mathcal{A}^T(y) = 0 && (2.5)\\
\nabla_s L_\mu &= t + b - \mathcal{B}^T(y) = 0 && (2.6)\\
\nabla_y L_\mu &= a - \mathcal{A}(X) - \mathcal{B}(s) = 0 && (2.7)\\
\nabla_t L_\mu &= s - \mu t^{-1} = 0 && (2.8)\\
\nabla_Z L_\mu &= X - \mu Z^{-1} = 0 && (2.9)
\end{aligned}
$$

The first two equations guarantee dual feasibility, the third primal feasibility. The last two nonlinear equations are the complementary slackness conditions. By the strict concavity of $\log\det Z$ and $e^t \log t$ there is a unique solution $(s_\mu, t_\mu, X_\mu, y_\mu, Z_\mu)$ to the KKT–conditions for any μ. The union of the solutions over $0 < \mu < \infty$ forms the central trajectory. For a point (s, t, X, y, Z) on the central trajectory we can determine the corresponding value of μ using (2.8) and/or (2.9),

$$\mu = \frac{\langle X, Z\rangle}{n} = \frac{\langle s, t\rangle}{m} = \frac{\langle X, Z\rangle + \langle s, t\rangle}{n+m}. \tag{2.10}$$

We repeat that $\langle X, Z\rangle + \langle s, t\rangle$ is the gap between the primal and dual objective value if the point is a feasible solution of the primal and the dual problem (1.2). (2.10) will also be used to associate μ values with points (s, t, X, y, Z) not lying on the central trajectory.

The interior point algorithm proceeds as follows. We start with a point (s, t, X, y, Z) for which $s > 0$, $t > 0$, $X \succ 0$, and $Z \succ 0$, but which is otherwise arbitrary. We estimate the current μ value of this point using (2.10) and divide it by two,

$$\mu = \frac{\langle X, Z\rangle + \langle s, t\rangle}{2(n+m)} \tag{2.11}$$

This choice performs very well in practical applications but does not guarantee monotonic decrease in μ.

Next we want to determine step directions $(\Delta s, \Delta t, \Delta X, \Delta y, \Delta Z)$ such that the new point $(s + \Delta s, t + \Delta t, X + \Delta X, y + \Delta y, Z + \Delta Z)$ lies on the central trajectory at this value of μ, i.e. the new point should satisfy equations (2.5)–(2.9) for this specific μ. Since (2.8) and (2.9) are nonlinear, we cannot solve the system directly. We can however choose from a variety of different linearizations stemming from several equivalent formulations of these equations. The choice of the linearization has a severe influence on the behavior of the algorithm. For sake of clarity we will postpone the detailed discussion (see Section 2.1) and choose the following representations,

$$
\begin{aligned}
t \circ s - \mu e &= 0 && (2.12)\\
ZX - \mu I &= 0. && (2.13)
\end{aligned}
$$

The linearization of the latter leads to non symmetric step directions, but we will get around this problem by using the symmetric part only. For simplicity of notation we rewrite (2.5)–(2.7) and (2.12)–(2.13) as the function

$$F_\mu(p) = F_\mu(s,t,X,y,Z) := \begin{pmatrix} Z + C - \mathcal{A}^T(y) \\ t + b - \mathcal{B}^T(y) \\ a - \mathcal{A}(X) - \mathcal{B}(s) \\ t \circ s - \mu e \\ ZX - \mu I \end{pmatrix} =: \begin{pmatrix} F_{d1} \\ F_{d2} \\ F_p \\ F_{ts} \\ F_{ZX} \end{pmatrix}. \tag{2.14}$$

The solution p^* to $F_\mu(p) = 0$ satisfies the KKT–conditions (2.5) to (2.9) and is the optimal solution to the barrier problem. To find a step direction $\Delta p = (\Delta s, \Delta t, \Delta X, \Delta y, \Delta Z)$ toward p^*, we use Newton's method, which says that Δp must satisfy

$$F_\mu(p) + \nabla F_\mu(\Delta p) = 0.$$

The step direction Δp is therefore the solution of the system

$$\begin{aligned} \Delta Z - \mathcal{A}^T(\Delta y) &= -F_{d1} && (2.15) \\ \Delta t - \mathcal{B}^T(\Delta y) &= -F_{d2} && (2.16) \\ -\mathcal{A}(\Delta X) - \mathcal{B}(\Delta s) &= -F_p && (2.17) \\ t \circ \Delta s + \Delta t \circ s &= -F_{ts} && (2.18) \\ Z\Delta X + \Delta Z X &= -F_{ZX}. && (2.19) \end{aligned}$$

We can solve this system efficiently by successively eliminating all variables but Δy. Indeed, we can express ΔZ and Δt in terms of Δy using (2.15) and (2.16),

$$\begin{aligned} \Delta Z &= \mathcal{A}^T(\Delta y) - F_{d1} && (2.20) \\ \Delta t &= \mathcal{B}^T(\Delta y) - F_{d2}. && (2.21) \end{aligned}$$

From (2.18) and (2.19) we get

$$\begin{aligned} \Delta s &= \mu t^{-1} - s - t^{-1}\Delta t s &&= \mu t^{-1} - s - t^{-1}\mathcal{B}^T(\Delta y)s + t^{-1}F_{d1}s && (2.22) \\ \Delta X &= \mu Z^{-1} - X - Z^{-1}\Delta Z X &&= \mu Z^{-1} - X - Z^{-1}\mathcal{A}^T(\Delta y)X + Z^{-1}F_{d2}X. && (2.23) \end{aligned}$$

Observe that ΔX will in general not be symmetric. We substitute the latter expressions into (2.17) for

$$\begin{aligned} &\mathcal{A}(Z^{-1}\mathcal{A}^T(\Delta y)X) + \mathcal{B}(t^{-1} \circ \mathcal{B}^T(\Delta y) \circ s) = \\ &= -F_p + \mathcal{A}(\mu Z^{-1} - X + Z^{-1}F_{d1}X) + \mathcal{B}(\mu t^{-1} - s + t^{-1} \circ F_{d2} \circ s) \qquad (2.24) \end{aligned}$$

The left hand side forms a new operator on Δy which we denote by

$$\mathcal{O}(\cdot) := \mathcal{A}(Z^{-1}\mathcal{A}^T(\cdot)X) + \mathcal{B}(t^{-1} \circ \mathcal{B}^T(\cdot) \circ s) \tag{2.25}$$

The right hand side of (2.24) can be simplified a little by using the defining equation for F_p. Now the final system for Δy reads

$$\mathcal{O}(\Delta y) = \mathcal{A}(\mu Z^{-1} + Z^{-1}F_{d1}X) + \mathcal{B}(\mu t^{-1} + t^{-1} \circ F_{d2} \circ s) - a. \tag{2.26}$$

Because of (2.1) we have

$$\mathcal{A}(Z^{-1}\mathcal{A}^T(\Delta y)X) = \mathcal{A}(X\mathcal{A}^T(\Delta y)Z^{-1}),$$

which implies that $\mathcal{O}$ is self adjoint, i.e. symmetric. It is not too surprising that $\mathcal{O}$ is indeed positive definite. For any vector $v \neq 0$

$$\begin{aligned}
&\langle \mathcal{O}(v), v\rangle = \\
&= \left\langle \mathcal{A}(Z^{-1}\mathcal{A}^T(v)X), v\right\rangle + \left\langle \mathcal{B}(t^{-1} \circ \mathcal{B}^T(v) \circ s), v\right\rangle \\
&= \left\langle Z^{-1}\mathcal{A}^T(v)X, \mathcal{A}^T(v)\right\rangle + \left\langle t^{-1} \circ \mathcal{B}^T(v) \circ s, \mathcal{B}^T(v)\right\rangle \\
&= \operatorname{tr}\left(Z^{-\frac{1}{2}}\mathcal{A}^T(v)X^{\frac{1}{2}}X^{\frac{1}{2}}\mathcal{A}^T(v)Z^{-\frac{1}{2}}\right) + \left\langle t^{-\frac{1}{2}} \circ \mathcal{B}^T(v) \circ s^{\frac{1}{2}}, t^{-\frac{1}{2}} \circ \mathcal{B}^T(v) \circ s^{\frac{1}{2}}\right\rangle \\
&= \left\langle Z^{-\frac{1}{2}}\mathcal{A}^T(v)X^{\frac{1}{2}}, Z^{-\frac{1}{2}}\mathcal{A}^T(v)X^{\frac{1}{2}}\right\rangle + \left\langle t^{-\frac{1}{2}} \circ \mathcal{B}^T(v) \circ s^{\frac{1}{2}}, t^{-\frac{1}{2}} \circ \mathcal{B}^T(v) \circ s^{\frac{1}{2}}\right\rangle \geq 0.
\end{aligned}$$

However, because of $s > 0$, $t > 0$, $X \succ 0$, and $Z \succ 0$, equality can only be attained if there is a vector v satisfying $\mathcal{A}^T(\Delta y) = 0$ and $\mathcal{B}^T(\Delta y) = 0$. This implies that the dual constraint matrix does not have full column rank, which is equivalent to the primal constraint matrix not having full row rank. Since we may always remove redundant equalities we may assume that the primal constraint matrix has indeed full row rank. In this case $\mathcal{O}$ is a positive definite operator.

By solving (2.26) and backsubstituting we get a quintuple $(\Delta s, \Delta t, \Delta X', \Delta y, \Delta Z)$ where $\Delta X'$ is not symmetric in general. We will see that it suffices to use the symmetric part of $\Delta X'$ only to get a good step direction, i.e. we use ΔX with

$$\Delta X = \frac{\Delta X'^T + \Delta X'}{2}.$$

To summarize, we solve for the step direction $(\Delta s, \Delta t, \Delta X, \Delta y, \Delta Z)$ by solving (2.26) for Δy. Substituting Δy into (2.20) and (2.21) we get ΔZ and Δt, respectively. Now Δs is best determined via (2.22) using Δt. Using (2.23) with ΔZ we get a $\Delta X'$ of which we take the symmetric part ΔX only.

Having determined the step direction $(\Delta s, \Delta t, \Delta X, \Delta y, \Delta Z)$ we would now like to step to the point $(s + \Delta s, t + \Delta t, X + \Delta X, y + \Delta y, Z + \Delta Z)$, but this new point may violate the "sign constraints", i.e. the new s and t have to remain strictly positive and the new X and Z have to remain positive definite. Therefore, by means of a line search, we determine a primal stepsize α_p and a dual stepsize α_d to arrive at a new point

$$\begin{aligned} s + \alpha_p \Delta s &\quad > 0 \\ t + \alpha_d \Delta t &\quad > 0 \\ X + \alpha_p \Delta X &\quad \succ 0 \\ y + \alpha_d \Delta y & \\ Z + \alpha_d \Delta Z &\quad \succ 0. \end{aligned}$$

and repeat. The algorithm continues until the current point (s, t, X, y, Z) satisfies primal feasibility, dual feasibility and the duality gap is sufficiently small. We complete the description of our interior point approach by giving a pseudo algorithm.

1. Input: a, b, C, $\mathcal{A}$, $\mathcal{B}$, tol.
2. Start with an appropriate $s > 0$, $t > 0$, $X \succ 0$, y, and $Z \succ 0$.
3. $\mu := (\langle Z, X\rangle + \langle t, s\rangle)/2n$
4. $F_{d1} := Z + C - \mathcal{A}^T(y)$, $F_{d2} := t + b - \mathcal{B}^T(y)$.
5. Compute the matrix representing $\mathcal{O}$: $M := \mathcal{O}(\cdot)$.
6. Solve $M\Delta y = \mathcal{A}(\mu Z^{-1} + Z^{-1}F_{d1}X) + \mathcal{B}(\mu t^{-1} + t^{-1} \circ F_{d2} \circ s) - a$ for Δy.
7. $\Delta Z := \mathcal{A}^T(\Delta y) - F_{d1}$, $\Delta t := \mathcal{B}^T(\Delta y) - F_{d2}$.
8. $\Delta X := \mu Z^{-1} - X - Z^{-1}\Delta Z X$, $\Delta s := \mu t^{-1} - s - t^{-1} \circ \Delta t \circ s$.
9. $\Delta X := (\Delta X + \Delta X^T)/2$.
10. Determine $0 < \alpha_p < 1$ and $0 < \alpha_d < 1$ close to one such that the sign constraints remain valid.
11. $s := s + \alpha_p \Delta s$, $X := X + \alpha_p \Delta X$,
    $t := t + \alpha_d \Delta t$, $y := y + \alpha_d \Delta y$, $Z := Z + \alpha_d \Delta Z$.
12. If $||Z + C - \mathcal{A}^T(y)|| + ||t + b - \mathcal{B}^T(y)|| + ||a - \mathcal{A}(X) - \mathcal{B}(s)|| > tol$ goto step 3.
13. if $\langle Z, X\rangle + \langle t, s\rangle > tol$ goto step 3 else stop.

2.1 Linearizations

In this section we will investigate the differences between linearizations stemming from different equivalent formulations of the nonlinear equations (2.8) and (2.9). We will mainly concentrate on (2.9) since it involves matrices rather than scalars. We assume that we always use the same formulation for both, vector and matrix equation, which seems to be the most natural choice and keeps the investigations within bounds. Essentially we have the following equivalent formulations of (2.9),

$$Z^{1/2}XZ^{1/2} - \mu I = 0, \tag{2.27}$$

$$X^{1/2}ZX^{1/2} - \mu I = 0. \tag{2.28}$$

$$X - \mu Z^{-1} = 0, \tag{2.29}$$

$$Z - \mu X^{-1} = 0, \tag{2.30}$$

$$ZX - \mu I = 0, \tag{2.31}$$

$$XZ - \mu I = 0, \tag{2.32}$$

$$ZX + XZ - 2\mu I = 0. \tag{2.33}$$

The first two forms (2.27) and (2.28) are very popular in linear programming. But in this case Z and X are diagonal matrices and the square root can be computed componentwise. For semidefinite programming full matrix square roots would have to be determined, this is computationally too expensive. Therefore the first two linearizations were not experimented with so far.

The third form was used in [64] and merits closer investigation. Using first order Taylor approximation the linearization of Z^{-1} is given by

$$(Z + \Delta Z)^{-1} = Z^{-1} - Z^{-1}\Delta Z Z^{-1}.$$

The equation corresponding to (2.19) reads

$$\Delta X + \mu Z^{-1}\Delta Z Z^{-1} = \mu Z^{-1} - X$$

and yields a symmetric step direction ΔX. Comparing this to (2.19) we see that X is replaced by μZ^{-1}. For points on the central trajectory to this value of μ this is actually equivalent. Using the same linearization for the vector equation the final equation for Δy corresponding to (2.26) reads

$$\begin{aligned}
&\mathcal{A}(Z^{-1}\mathcal{A}^T(\Delta y)Z^{-1}) + \mathcal{B}(t^{-1} \circ \mathcal{B}^T(\Delta y) \circ t^{-1}) = \\
&= \mathcal{A}(Z^{-1} + Z^{-1}F_{d1}Z^{-1}) + \mathcal{B}(t^{-1} - t^{-1} \circ F_{d2} \circ t^{-1}) - \frac{a}{\mu}.
\end{aligned}$$

The striking fact is that this equation contains no information about the primal variables any more. Indeed, it is exactly the same system Jarre derived in [34] for a purely dual approach. The advantage is that this is in fact a pure Newton method for the dual problem. By the theory of Nesterov and Nemirovskii [55] on self concordant functions it is well known that this yields a polynomial time algorithm for solving the dual problem. However, if we want to have a primal–dual algorithm this step direction performs poorly since it acts only on the information of the dual problem. This was also found to be true in practice. The algorithm displayed very fast convergence on the dual part but ran into serious trouble on the primal side.

The form (2.30) was examined in [64]. By using the linearization of X^{-1} we arrive at the following equation,

$$\Delta X + \frac{1}{\mu} X \Delta Z X = X - \frac{1}{\mu} XZX.$$

Again this yields a symmetric step direction. This time Z^{-1} is replaced by $\frac{1}{\mu}X$, and there is a little change on the right hand side of the equation for Δy,

$$\begin{aligned} &\mathcal{A}(X\mathcal{A}^T(\Delta y)X) + \mathcal{B}(s \circ \mathcal{B}^T(\Delta y) \circ s) = \\ &= -\mu F_p + \mathcal{A}(\mu X - XZX + XF_{d1}X) + \mathcal{B}(\mu s - s \circ t \circ s + s \circ F_{d2} \circ s) \\ &= -\mu F_p + \mathcal{A}(\mu X + XCX - X\mathcal{A}^T(y)X) + \mathcal{B}(\mu s - s \circ b \circ s + s \circ \mathcal{B}^T(y) \circ s) \end{aligned}$$

Again all the information about the current dual iterate is lost and the step direction is good only for the primal barrier problem. The same statements as above apply.

Form (2.31) is the choice for our algorithm. It is easy to see that form (2.32) results in exactly the same step direction. Indeed, all expressions $Z^{-1}AX$ are replaced by XAZ^{-1}. In all cases A is symmetric, therefore the second is the transpose of the first. Because of (2.1) the resulting system for Δy is exactly the same. $\Delta X_2'$ of linearization (2.32) is the transpose of $\Delta X_1'$ of linearization (2.31), therefore the symmetric part ΔX of both is the same. We do not have to consider the vector equation since the Hadamard product is commutative anyways.

In [3] form (2.33) is used because a symmetric update is guaranteed. The resulting step direction differs from the step direction of our algorithm. Alizadeh, Haberly, and Overton present some evidence for good numerical properties of this linearization. However, as the system is more difficult to solve, computation time per iteration can be considerably higher.

Summarizing we can say that by using the linearization of (2.31) we get a system of efficiently solvable equations, where the information about the primal and the dual current iterate is well exploited.

Chapter 3

Convergence

The main goal of this chapter is to prove convergence of the algorithm described in Chapter 2. First we will show that the step direction $(\Delta s, \Delta t, \Delta X, \Delta y, \Delta Z)$ indeed forms a descent direction with respect to an appropriately defined merit function. Then, by assuming the compactness of the set of primal feasible solutions and a strictly feasible starting point, we will be able to prove convergence of the method. It should be noted that we cannot make use of the theory of Nesterov and Nemirovskii [55] since the descent direction is not the Newton direction of the underlying problem. The line of argument follows a proof of Anstreicher and Vial [4, 71] for a convex programming problem without semidefinite variables.

3.1 Merit Function

In the course of the algorithm, after having computed the step direction, we have to perform a line search to guarantee feasibility of the variables with respect to the "sign constraints". For the prove of convergence we have to replace this somewhat inaccurately defined process by a more accurate concept. This is achieved by introducing a merit function. Essentially, the merit function gives an estimate of the distance of a point to the optimum and tends to infinity if a variable goes to zero or singularity. Since the step direction of the algorithm is a descent direction with respect to the merit function performing a line search with respect to the merit function will guarantee that the variables remain positive and that some progress to the optimum is made.

The merit function is defined as

$$\begin{aligned} f_\mu(s,t,X,y,Z) \;=\; & \langle Z, X\rangle - \mu \log\det(ZX) + \langle t, s\rangle - \mu e^T \log(t \circ s) + \\ & \frac{1}{2}||F_{d1}||^2 + \frac{1}{2}||F_{d2}||^2 + \frac{1}{2}||F_p||^2 \end{aligned}$$

The terms of the merit function have very straight forward interpretations. If (s, X) is a feasible solution of the primal and (t, y, Z) is a feasible solution of the dual then $\langle Z, X\rangle + \langle t, s\rangle$ is the gap between primal and dual objective value. $-\mu(\log\det(ZX) + \log(t\circ s))$ is a strictly convex barrier function keeping all variables positive (definite). The last three terms measure the amount of infeasibility.

We will give a short discussion of the most important properties of f_μ. It is continuously differentiable for $s > 0$, $t > 0$, $X \succ 0$, and $Z \succ 0$ and goes to infinity on the boundary. We have

$$\langle Z, X\rangle - \mu\log\det(ZX) = \operatorname{tr}(ZX) - \mu\log\prod_{i=1}^{n}\lambda_i(ZX) = \sum_{i=1}^{n}\lambda_i(ZX) - \mu\sum_{i=1}^{n}\log\lambda_i(ZX). \tag{3.1}$$

As the minimum of $(x - \mu\log x)$ for $x > 0$ is attained at $x = \mu$, function f_μ is bounded below by $(n+m)\mu(1-\log\mu)$.

$$f_\mu \geq (n+m)\mu(1-\log\mu) := c_\mu. \tag{3.2}$$

A saddle point $p_\mu = (s_\mu, t_\mu, X_\mu, y_\mu, Z_\mu)$ of the Lagrangian to DBP at parameter μ satisfies $Z_\mu X_\mu = \mu I$, $t\circ s = \mu e$, $F_{d1} = 0$, $F_{d2} = 0$, and $F_p = 0$. Evidently, a given point p is the saddle point if and only if the the minimum value c_μ is attained for p,

$$F_\mu(p) = 0 \Longleftrightarrow f_\mu(p_\mu) = (n+m)\mu(1-\log\mu).$$

Another interesting fact is the connection of the merit function to the primal and dual barrier problems. The DBP has already been introduced on Page 13, the primal barrier problem (PBP) is given by

$$\begin{array}{lll}(\text{PBP}) & \text{Minimize} & \langle C, X\rangle - \mu(\log\det X + e^T\log s) \\ & \text{subject to} & a - \mathcal{A}(X) - \mathcal{B}(s) = 0.\end{array} \tag{3.3}$$

For further use we will denote the objective function by

$$f_{P;\mu}(s, X) := \langle C, X\rangle - \mu(\log\det X + e^T\log s). \tag{3.4}$$

Let now (s, X) be a feasible point of (PBP) and (t, y, Z) a feasible point of (DBP), then

$$\begin{aligned} f_\mu(s, t, X, y, Z) &= f_{D;\mu}(t, y, Z) - f_{P;\mu}(s, X) && (3.5)\\ &= \langle Z, X\rangle - \mu\log\det(ZX) + \langle t, s\rangle - \mu e^T\log(t\circ s) && (3.6)\end{aligned}$$

For a feasible primal dual point the merit function is the difference between the objective values of the dual and primal barrier problems. Furthermore it follows that on the set of feasible points the merit function is strictly convex.

3.2 Descent Direction

Let $\Delta p = (\Delta s, \Delta t, \Delta X, \Delta y, \Delta Z)$ be the step direction computed by the algorithm as described on page 16. We will now prove that Δp is a descent direction of f_μ.

Lemma 1 *The directional derivative of f_μ in the direction Δp satisfies*

$$\left\langle \frac{\partial f_\mu}{\partial p}, \Delta p \right\rangle \leq 0 \tag{3.7}$$

with equality holding if and only if $F_\mu(p) = 0$.

Proof: Throughout the proof we will use the basic property that the inner product of a symmetric matrix with a skew–symmetric matrix is zero. We will need this property to replace ΔX with the not yet symmetrized $\Delta X'$, the solution to the system (2.15) to (2.19). In particular, for a symmetric matrix A

$$\langle A, \Delta X \rangle = \langle A, \Delta X' \rangle .$$

We have to prove that

$$\left\langle \frac{\partial f_\mu}{\partial s}, \Delta s \right\rangle + \left\langle \frac{\partial f_\mu}{\partial t}, \Delta t \right\rangle + \left\langle \frac{\partial f_\mu}{\partial X}, \Delta X \right\rangle + \left\langle \frac{\partial f_\mu}{\partial y}, \Delta y \right\rangle + \left\langle \frac{\partial f_\mu}{\partial Z}, \Delta Z \right\rangle \leq 0$$

The partial derivatives of f_μ are

$$\begin{aligned}
\frac{\partial f_\mu}{\partial s} &= t - \mu s^{-1} - \mathcal{B}^T(F_p) \\
\frac{\partial f_\mu}{\partial t} &= s - \mu t^{-1} + F_{d2} \\
\frac{\partial f_\mu}{\partial X} &= Z - \mu X^{-1} - \mathcal{A}^T(F_p) \\
\frac{\partial f_\mu}{\partial y} &= -\mathcal{A}(F_{d1}) - \mathcal{B}(F_{d2}) \\
\frac{\partial f_\mu}{\partial Z} &= X - \mu Z^{-1} + F_{d1}.
\end{aligned}$$

Collecting all partial derivatives of F_p we get

$$\begin{aligned}
\left\langle -\mathcal{B}^T(F_p), \Delta s \right\rangle + \left\langle -\mathcal{A}^T(F_p), \Delta X \right\rangle &= \left\langle -\mathcal{B}^T(F_p), \Delta s \right\rangle + \left\langle -\mathcal{A}^T(F_p), \Delta X' \right\rangle \\
&= \langle F_p, -\mathcal{B}(\Delta s) \rangle + \langle F_p, -\mathcal{A}(\Delta X') \rangle \\
&= \langle F_p, -\mathcal{B}(\Delta s) - \mathcal{A}(\Delta X') \rangle \\
&= \langle F_p, -F_p \rangle \\
&= -||F_p||^2,
\end{aligned}$$

where the third line follows from (2.17). Using (2.15) and (2.16) we find analogously for F_{d1} and F_{d2}

$$\begin{aligned} \langle F_{d1}, \Delta Z\rangle + \langle -\mathcal{A}(F_{d1}), \Delta y\rangle &= -||F_{d1}||^2 \\ \langle F_{d2}, \Delta t\rangle + \langle -\mathcal{A}(F_{d2}), \Delta y\rangle &= -||F_{d2}||^2. \end{aligned}$$

Next we examine the terms belonging to the partial derivative of $\langle Z, X\rangle - \log\det(ZX)$ and make use of (2.18).

$$\begin{aligned} &\left\langle Z - \mu X^{-1}, \Delta X\right\rangle + \left\langle X - \mu Z^{-1}, \Delta Z\right\rangle = \\ &= \left\langle Z - \mu X^{-1}, \Delta X'\right\rangle + \left\langle X - \mu Z^{-1}, \Delta Z\right\rangle \\ &= \operatorname{tr}\left((I - \mu X^{-1}Z^{-1})(Z\Delta X' + \Delta Z X)\right) \\ &= -\operatorname{tr}\left((I - \mu X^{-1}Z^{-1})(ZX - \mu I)\right) \\ &= -\operatorname{tr}(ZX - 2\mu I + \mu^2 (ZX)^{-1}) \\ &= -\sum_{i=1}^{n}\left(\lambda_i(ZX) - 2\mu + \mu^2\lambda_i^{-1}(ZX)\right) \\ &= -\sum_{i=1}^{n}\lambda_i(ZX)\left(1 - \mu\lambda_i^{-1}(ZX)\right)^2 \end{aligned}$$

As Z and X are positive definite, all eigenvalues of ZX are strictly positive (ZX is similar to $Z^{\frac{1}{2}}XZ^{\frac{1}{2}}$ which is clearly positive definite). The expression is less or equal to zero with equality holding if and only if $ZX = \mu I$.

Finally, using (2.19) we get

$$\left\langle t - \mu s^{-1}, \Delta s\right\rangle + \left\langle s - \mu t^{-1}, \Delta t\right\rangle = -(t \circ s)^T (e - \mu t^{-1} \circ s^{-1})^2$$

Again $t \circ s$ is strictly positive, so the expression is less or equal to zero with equality holding if and only if $t \circ s = \mu e$.

Summing up we have

$$\begin{aligned} \frac{\partial f_\mu}{\partial p}\Delta p &= -||F_p||^2 - ||F_{d1}||^2 - ||F_{d2}||^2 - \sum_{i=1}^{n}\lambda_i(ZX)\left(1 - \mu\lambda_i^{-1}(ZX)\right)^2 \\ &\quad -(t \circ s)^T(e - \mu t^{-1} \circ s^{-1})^2 \le 0 \end{aligned} \tag{3.8}$$

with equality holding if and only if the KKT–conditions are satisfied. □

3.3 Convergence for Fixed μ

For the proof of convergence for fixed μ it is necessary to confirm that the iterates of the algorithm stay within a compact set. Once this is established the proof is easily formulated using standard techniques from convergence analysis.

We make the assumption that the set

$$\mathcal{X} = \{X \mid a - \mathcal{A}(X) - \mathcal{B}(s) = 0, s \geq 0, X \succeq 0\} \tag{3.9}$$

is compact and that we start the algorithm with an interior point of the problem, i.e. a point $p^0 = (s^0, t^0, X^0, y^0, Z^0)$ such that

$$\begin{gathered} a - \mathcal{A}(X^0) - \mathcal{B}(s^0) = 0 \\ Z^0 + C - \mathcal{A}^T(y^0) = 0 \\ t^0 + b - \mathcal{B}^T(y^0) = 0 \\ s^0 > 0,\ t^0 > 0,\ X^0 \succ 0,\ Z^0 \succ 0. \end{gathered} \tag{3.10}$$

For computational convenience we work with inaccurate line search methods. Let the step direction, computed as described on page 16, be $\Delta p = (\Delta s, \Delta t, \Delta X, \Delta y, \Delta Z)$, then the stepsize α will have to satisfy the Goldstein Armijo conditions

$$\delta_1 \alpha \left| \left\langle \frac{\partial f_\mu}{\partial p}, \Delta p \right\rangle \right| \leq f_\mu(p) - f_\mu(p + \alpha \Delta p) \leq \delta_2 \alpha \left| \left\langle \frac{\partial f_\mu}{\partial p}, \Delta p \right\rangle \right| \tag{3.11}$$

with $0 < \delta_1 < \delta_2 < 1$. Because f_μ is bounded below and tends to infinity on the boundary such an $\alpha > 0$ always exists. As all equations of (3.10) are linear the new point $p + \alpha \Delta p$ is another point satisfying (3.10). Conditions (3.11) form a kind of sufficient decrease criterion and help to prove convergence.

We give a short survey of the compactness argument. Because the iterates remain feasible interior points of the primal and dual problem the merit function can be interpreted as the difference of the objective values of primal and dual barrier problem. Since this difference is decreasing and weak duality prohibits that both problems wander off into the same direction both objective values have to remain "close" to the optimum. Because of the barrier term $\log\det X$ this implies that the minimal eigenvalue of X is bounded away from zero. The term $\langle Z, X\rangle - \log\det ZX$ being bounded from above then implies the boundedness of Z. The analogous observation is true for s and t. Because $\mathcal{A}^T(\cdot)$ and $\mathcal{B}^T(\cdot)$ together have full column rank, the boundedness of t and Z implies the boundedness of y. We will now give the proof in detail.

As already pointed out in Section 3.1 conditions (3.10) imply that the merit function is the gap between primal and dual barrier problem evaluated at the current point, i.e. for $p = (s, t, X, y, Z)$ satisfying (3.10) we have

$$f_\mu(p) = f_{D;\mu}(t, y, Z) - f_{P;\mu}(s, X). \tag{3.12}$$

(3.10) is also satisfied for $(s^0, t^i, X^0, y^i, Z^i)$ for $i \geq 0$. Using (3.2) for this choice we get a lower bound for the dual barrier problem, independently of the iterates,

$$f_{D;\mu}(t^i, y^i, Z^i) \geq f_{P;\mu}(s^0, X^0) + c_\mu \tag{3.13}$$

By Lemma (1) the merit function is decreasing with respect to the iterates,

$$f_\mu(p^{i+1}) \leq f_\mu(p^i). \tag{3.14}$$

Together with (3.12) and (3.13) we obtain the following lower bound on $f_{P;\mu}$,

$$f_{P;\mu}(s^i, X^i) = f_{D;\mu}(t^i, y^i, Z^i) - f_\mu(p^i) \geq f_{P;\mu}(s^0, X^0) + c_\mu - f_\mu(p^0) \tag{3.15}$$

For an upper bound on $f_{P;\mu}$ we exploit the compactness of $\mathcal{X}$ and denote the maximal eigenvalue of all $X \in \mathcal{X}$ by $\hat{\lambda}_{max}$ and the largest component of all $s \in \mathcal{X}$ by $\hat{\sigma}_{max}$. If either of these values happens to be smaller than one, we set the corresponding value to 1. By weak duality $\max_{(s,X)\in\mathcal{X}}(\langle C, X\rangle + \langle b, s\rangle) \leq \langle a, y^0\rangle$, so we have for all iterates

$$\begin{aligned} f_{P;\mu}(s^i, X^i) &\leq \left\langle a, y^0 \right\rangle + \mu(\log\det X^i + e^T \log s^i) \leq \\ &\leq \left\langle a, y^0 \right\rangle + \mu(n \log \hat{\lambda}_{max} + m \log \hat{\sigma}_{max}) + \mu \log \lambda_{min}(X^i). \end{aligned}$$

Note, that by definition $\hat{\lambda}_{max} \geq 1$ and $\hat{\sigma}_{max} \geq 1$. Together with (3.15) this yields for all $i \geq 0$

$$\lambda_{min}(X^i) \geq \underbrace{\exp\left(-\frac{\langle Z^0, X^0\rangle + \langle t^0, s^0\rangle + f_\mu(p^0) - c_\mu}{\mu} + c\right)}_{=:\,\lambda_{min;\mu}} > 0 \tag{3.16}$$

where c is a negative constant (independent of μ). Analogously we get a lower bound $\sigma_{min;\mu}$ greater than zero for the components of the iterates s^i. Using these bounds we will now show that the iterates Z^i stay bounded. (3.14) implies that $\langle Z^i, X^i\rangle - \log\det Z^i X^i$ remains bounded. By (3.16) $-\log\det X^i$ remains bounded and this term can be omitted. Using the property that

$$\langle Z, X\rangle \geq \lambda_{min}(X) \sum_{j=1}^{n} \lambda_j(Z).$$

we get the following inequality,

$$\left\langle Z^i, X^i \right\rangle - \mu \log\det Z^i \geq \lambda_{min;\mu} \sum_{j=1}^{n} \lambda_j(Z^i) - \mu \sum_{j=1}^{n} \log \lambda_j(Z^i).$$

If the Z^i grow beyond bounds so will their trace $\mathrm{tr}(Z^i) = \sum_{j=1}^n \lambda_j(Z^i)$. For fixed i we will now investigate the optimal choice of the eigenvalues z_j summing up to the trace of Z^i such that the right hand side becomes minimal. Obviously,

$$\min_{z_j \geq 0; \sum z_j = \mathrm{tr}(Z^i)} \lambda_{min;\mu} \sum_{j=1}^{n} z_j - \mu \sum_{j=1}^{n} \log(z_j) \leq \left\langle Z^i, X^i \right\rangle - \mu \log\det Z^i, \tag{3.17}$$

where the right hand side must remain bounded. To find optimal z_j we investigate the Lagrangian

$$\mathcal{L} = \lambda_{min;\mu} \sum_{j=1}^{n} z_j - \mu \sum_{j=1}^{n} \log(z_j) + \gamma \left(\sum_{j=1}^{n} z_j - \operatorname{tr}(Z^i) \right).$$

The KKT–conditions read

$$\begin{aligned} \nabla_{z_j} \mathcal{L} &= \lambda_{min;\mu} - \frac{\mu}{z_j} + \gamma = 0 \qquad j = 1, \ldots, n \\ \nabla_{\gamma} \mathcal{L} &= \sum_{j=1}^{n} z_j - \operatorname{tr}(Z^i) = 0 \end{aligned}$$

which yield

$$z_j = \frac{\operatorname{tr}(Z^i)}{n} \qquad j = 1, \ldots, n.$$

Indeed $\operatorname{tr}(Z^i) \to \infty$ would imply that the left hand side of (3.17) grows beyond bounds. But as the right hand side is bounded the Z^i have to remain bounded. Analogously we get the boundedness of the t^i. The boundedness of the y^i now follows from the fact that the equality constraints on y have full column rank. This establishes that the iterates p^i remain within a compact set.

It is quite evident that the eigenvalues of the Z^i and the components of the t^i are also bounded away from zero since the eigenvalues of X and the components of s are bounded from above. We may therefore conclude that the interates remain within a compact set contained in the relative interior of the feasible region.

We are now ready to prove convergence for a fixed value of μ.

Theorem 2 *For a fixed value of μ let $\{p^i\}$ be a sequence generated by the algorithm with the stepsize satisfying conditions (3.11) at each iteration and with the starting point p^0 satisfying (3.10). Then $\{p^i\}$ converges to a point p^* in the relative interior of the feasible region and $F_\mu(p^*) = 0$.*

Proof: By the considerations above all points of the sequence lie within a compact set contained in the relative interior of the feasible region. Therefore sequence $\{p^i\}$ has a convergent subsequence $\{p^k\}_{k \in \mathcal{K}}$ converging to p^*, a point in the relative interior of the feasible region. By a standard result from global convergence theory (see Appendix A) conditions (3.11) imply that $\left\{ \left\langle \frac{\partial f_\mu}{\partial p}, \frac{\Delta p^k}{||\Delta p^k||} \right\rangle \right\}_{k \in \mathcal{K}} \to 0$. But because $\nabla F_\mu(p^*)$ is non singular $||\Delta p^k||$ is bounded for all $k \in \mathcal{K}$. Together with f_μ being continuously differentiable we must have $\left\{ \left\langle \frac{\partial f_\mu}{\partial p}, \Delta p^k \right\rangle \right\}_{k \in \mathcal{K}} \to 0$. From Lemma 1 we conclude that $F_\mu(p^*) = 0$. $\square$

3.4 Global Convergence

In the previous Section we have proved that for a fixed value of the barrier parameter μ the algorithm converges to the optimal solution of the primal and dual *barrier* problem. What we indeed want to show is that by phasing out μ slowly we obtain a sequence converging to the optimal solution of the original primal and dual problems (PSDP) and (DSDP), respectively. To this end we introduce a rule when to reduce the value of μ. This criterion turns out to give very precise bounds for the distance of the current iterate to the optimal solution of the barrier problem to parameter μ. Since the sequence of the optimal solutions converges to the global optimum this will establish global convergence.

By the proof of Theorem 2 we know that for fixed μ the directional derivative in the step direction tends to zero as the iterates approach the optimum. It seems natural to decrease μ whenever the derivative gets too flat, i.e. μ will be decreased by a constant factor $0 < \theta < 1$ whenever

$$\left|\left\langle \frac{\partial f}{\partial p}, \Delta p^i \right\rangle\right| < \beta\mu \tag{3.18}$$

for some $0 < \beta < \frac{1}{2}$. From the iterates satisfying this condition we form a new sequence $\{p^j\}_{j\in\mathcal{J}}$. For this sequence we will show convergence to a point satisfying the KKT-conditions of the original problems.

Theorem 3 *Let $\{\mu^k\}$ be a sequence of positive numbers converging to zero. Let $\{p^i\}$ be the sequence of points generated by the algorithm starting at a point satisfying (3.10) and using criterion (3.18) for switching to the next μ-value of the sequence $\{\mu^k\}$. Let $\{p^j\}_{j\in\mathcal{J}}$ be the subsequence of $\{p^i\}$ formed by the points at which μ-switching occurs. Then $\{p^j\}_{j\in\mathcal{J}}$ converges to a point p^* fulfilling the Karush–Kuhn–Tucker conditions.*

Proof: For a point p satisfying (3.18) for a certain μ we get from (3.8)

$$\begin{array}{rcll} \lambda_i(ZX)\left(1-\mu\lambda_i^{-1}(ZX)\right)^2 & < & \beta\mu & \quad i = 1\dots n \\ (t\circ s)^T(e-\mu t^{-1}\circ s^{-1})^2 & < & \beta\mu & \end{array}$$

For a variable $x > 0$ we have

$$x\left(1-\frac{\mu}{x}\right)^2 < \beta\mu \quad\Longrightarrow\quad \frac{\mu}{2} < x < 2\mu.$$

Therefore

$$\frac{\mu}{2} < \quad \lambda_i(ZX) \quad < 2\mu \tag{3.19}$$

$$\frac{\mu}{2}e < \quad t\circ s \quad < 2\mu e \tag{3.20}$$

and, for μ small enough, we get the following upper bound for f_μ,

$$\begin{aligned} f_\mu &= \langle Z, X\rangle + \langle t, s\rangle - \mu(\log\det ZX - e^T \log t \circ s) < \\ &= n \cdot 2\mu + m \cdot 2\mu - \mu(n \log \frac{\mu}{2} + m \log \frac{\mu}{2}) = \\ &= (n+m)\mu\,(2 - \log\mu - \log 2)\,. \end{aligned}$$

Furthermore the gap between primal and dual feasible solution is bounded by

$$\langle Z, X\rangle + \langle t, s\rangle < 2(n+m)\mu. \tag{3.21}$$

As in (3.16) this bounds the minimal eigenvalue of X

$$\begin{aligned} \lambda_{min}(X) &\geq \exp\left(-\frac{\langle Z, X\rangle + t^T(b - B(X)) + f_\mu - c_\mu}{\mu} + c_1\right) > & (3.22)\\ &> \exp\left(-2(n+m) - (n+m)(2-\log 2) + (n+m)\log\mu + c_1\right) = & (3.23)\\ &= \mu c_2 > 0 & (3.24) \end{aligned}$$

with a negative constant c_1 and a positive constant c_2 (independent of μ). It is now easy to bound $\lambda_{max}(Z)$ by

$$\mu c_2 \lambda_{max}(Z) < \langle Z, X\rangle < 2n\mu. \tag{3.25}$$

Analogously we get the boundedness of t and consequently of y. So for $\mu \to 0$ we have

$$\begin{aligned} \lambda_i(ZX) &\to 0 \qquad i = 1 \ldots n \quad \Longrightarrow \quad ZX \to 0 \\ t \circ s &\to 0 \\ f_\mu(s, t, X, y, Z) &\to 0 \end{aligned}$$

Because of the special choice of the p^j the same is true for the sequence $\{p^j\}_{j \in \mathcal{J}}$. Because of compactness there must be an accumulation point p^*. By the considerations above p^* is feasible and satisfies the KKT–conditions of the original problem. □

Note, that $f \to 0$ implies that the gap between the primal and dual solutions indicated by p^* is zero. So this also establishes strong duality.

Chapter 4

Some Practical Aspects

The algorithm suggested by the convergence analysis of Chapter 3 is mainly of academic interest. For practical applications other aspects have to be considered to get good performance. On the one hand this chapter will deal with some of these aspects, on the other we will outline some of the new important results reported in [39] and indicate further research possibilities.

4.1 Predictor–Corrector

For interior point methods for linear programming it was observed [19] that the step directions of most interior point algorithms are linear combinations of two search directions, the affine direction and the centering direction. The affine direction is responsible for the progress towards the desired optimum, the centering direction pulls the current iterate closer to the central trajectory. This concept was explicitly used in [48, 49] to derive a predictor–corrector approach for computing a new step direction. The method is very successful in practice [46, 13]. The concept can also be applied to semidefinite programming with the same success (see Section 7.3.2), a fact which was also observed in [3].

The affine direction is the solution to the linearization of the KKT–conditions to the original problem. With respect to system (2.15) to (2.19) the only change is that the barrier parameter μ is zero.

$$\begin{aligned}
\Delta Z - \mathcal{A}^T(\Delta y) &= -F_{d1} \\
\Delta t - \mathcal{B}^T(\Delta y) &= -F_{d2} \\
-\mathcal{A}(\Delta X) - \mathcal{B}(\Delta s) &= -F_p \\
t \circ ds + dt \circ s &= -t \circ s \\
Z\Delta X + \Delta Z X &= -ZX.
\end{aligned}$$

It can be solved in the same way and the final equation differs from (2.26) only in the right hand side,

$$\begin{aligned} &\mathcal{A}(Z^{-1}\mathcal{A}^T(\Delta y)X) + \mathcal{B}(t^{-1} \circ \mathcal{B}^T(\Delta y) \circ s) = \\ &= \ \mathcal{A}(Z^{-1}F_{d1}X) + \mathcal{B}(t^{-1} \circ F_{d2} \circ s) - a. \end{aligned}$$

The result is the affine step direction $\Delta p = (\Delta s, \Delta t, \Delta X, \Delta y, \Delta Z)$. Again ΔX is the symmetric part of $\Delta X'$. Now we want to find a corrector direction $\delta p = (\delta s, \delta t, \delta X, \delta y, \delta Z)$ for the point $p + \Delta p$, such that the corrected point $p + \Delta p + \delta p$ is close to the central trajectory at parameter μ. We have to solve (2.15) to (2.19) at the point $p + \Delta p$ for step direction δp.

$$\begin{aligned} \delta Z - \mathcal{A}^T(\delta y) &= -F_{d1} - \Delta Z + \mathcal{A}^T(\Delta y) = 0 \\ \delta t - \mathcal{B}^T(\delta y) &= -F_{d2} - \Delta t + \mathcal{B}^T(\Delta y) = 0 \\ -\mathcal{A}(\delta X) - \mathcal{B}(\delta s) &= -F_p + \mathcal{A}(\Delta X) + \mathcal{B}(\Delta s) = 0 \\ t \circ \delta s + \delta t \circ s &= \mu e - (t + \Delta t) \circ (s + \Delta s) = \mu I - \Delta t \circ \Delta s \\ Z\delta X + \delta Z X &= \mu I - (Z + \Delta Z)(X + \Delta X) = \mu I - \Delta Z \Delta X. \end{aligned}$$

Note that we have omitted the terms $\Delta t \circ \delta s + \delta t \circ \Delta s$ and $\Delta Z \delta X + \delta Z \Delta X$ in the last two equations. If the current iterate is already moderately close to the optimum, Δp will be small and so will the error by omitting the product. The gain however is that we can use the old factorization of the system matrix to solve for δy. Indeed the final equation of the system again changes on the right hand side only,

$$\begin{aligned} &\mathcal{A}(Z^{-1}\mathcal{A}^T(\delta y)X) + \mathcal{B}(t^{-1} \circ \mathcal{B}^T(\delta y) \circ s) = \\ &= \ \mathcal{A}(\mu Z^{-1} - Z^{-1}\Delta Z \Delta X) + \mathcal{B}(\mu t^{-1} - t^{-1}\Delta s \Delta t). \end{aligned}$$

The result is now the centering step direction δp, the search direction for the line search is $\Delta p + \delta p$. In practical experiments this step direction led to faster convergence and more stability of the algorithm.

Theoretically the process of correcting could be repeated several times. This approach was discussed in [13] for linear programming. It turned out that correcting once is obligatory whereas further correcting had little influence or might even worsen results. The same was observed in our case, in general a second correction led to very poor step directions, whereas a single correction brought along considerable improvements.

In linear programming the barrier parameter μ is mostly calculated with respect to the predictor $p + \Delta p$. In our case the predictor was in general not feasible and therefore the corresponding μ–value was quite useless. Even performing a line search and using the feasible predictor $p + \alpha \Delta p$ did not indicate serious advantages to simple a priori choices of μ.

4.2 Stepsize and Choice of μ

In Chapter 3 the stepsize is determined by a line search with respect to the merit function. In theory the stepsize should satisfy conditions (3.11) to guarantee convergence. In practice the evaluation of the merit function at a point p requires the point to be in the domain of the merit function. To guarantee this we have to find a steplength α such that $s+\alpha\Delta s > 0$, $t+\alpha\Delta t > 0$, $X+\alpha\Delta X \succ 0$, and $Z+\alpha\Delta Z \succ 0$. An upper bound on α is easily computed for the vectors s and t. For the matrices the signs of the eigenvalues have to be known. This can be achieved by factorizing $X+\alpha\Delta X$ and $Z+\alpha\Delta Z$. Since in many practical applications neither X nor Z are sparse this is a comparatively expensive process of $O(n^3)$ steps. There is a close connection between feasible stepsize and the choice of μ. Since these choices have decisive influence on the performance of the algorithm it seems to be worth to report some of our experience.

In our implementation we restrict the stepsize α to the interval $(0,1]$. Algorithms not going beyond the computed step are often referred to as short step algorithms. In linear programming it seems to be quite successful to use larger stepsizes as well. In our experiments, however, the step is too large most of the time. Only quite late in the optimization process stepsizes of size 1 appear more often. It is our feeling that it is better to influence stepsizes by appropriately choosing the barrier parameter than using larger stepsizes. In general it seems to be better to decrease μ too little than too much, it should be reduced cautiously in the beginning and less cautiously towards the end. We use (2.11) in the beginning and reduce μ by another factor $\frac{1}{2}$ if the last stepsize was close to 1. For our test problems this kept the stepsize changing slowly. As a consequence the previous steplength can be used as a good starting value for the line search.

In implementations the upper bound α_{vec} for the stepsize caused by the positivity restrictions of the vectors should always be computed first. The maximal step size is then $\min\{1,\alpha_{vec}\}$, if $\alpha_{vec} < 1$ this may reduce the number of matrix factorizations. Then the old stepsize (if it is in the feasible interval) is used to start the line search for the matrices. If the matrix is not positive definite, the stepsize is reduced by a factor 0.8, the search is continued by reduction steps till the matrix is positive definite. If, however, the matrix is positive definite and there is still room for a longer step, the search is continued by increasing steps (by a factor of 1.25) until the upper bound is reached or positive definiteness is violated. So in fact we try to make steps with α as close to 1 as possible. For security we also checked the value of the merit function, but only a few times a step was reduced because of this check (see Section 7.3.2). The evaluation of the merit function itself is computationally rather expensive since the computation of the determinant again requires the factorization of the matrices. In practical applications the line search with respect to the merit function should be switched off by default and should be used only in case of numerical difficulties.

If the concept of the merit function is not used at all it may be worth spending some thought on different stepsizes for primal and dual update. If the problem is very well behaved it may speed up convergence. On the other hand very different stepsizes may cause either of the problems to be far ahead of the other, which may result in a poor choice of the barrier parameter μ and a poor new step direction. For our examples the algorithm seemed to work better if a separate steplength was computed for each problem.

Another concept successfully applied in linear programming is the concept of weighted centers or the v-space [33]. Instead of centering towards μe one chooses some sequence of positive vectors v^i. Each v should be reasonably close to the current iterate product $t \circ s$ to guarantee a large stepsize. On the other hand v should pull the solution steadily towards μe to increase numerical stability. Analogously, for semidefinite programming one could try to center towards some appropriately chosen positive definite matrix A, instead of μI, which is then slowly transformed to the identity matrix. In contrast to linear programming, where there is a natural way to introduce the v–path by means of weighted barrier functions, there is no such derivation for the semidefinite case. Weighted barrier functions would suggest the use of several different barrier parameters, one for each independent log term, instead of only one parameter μ.

4.3 System Matrix and Sparsity

To get some more insight into the nature of the central equation (2.26) we look at the following equivalent representation of the linear operator $\mathcal{A}(\cdot)$,

$$\mathcal{A}(X) = \begin{pmatrix} \operatorname{tr}(A_1 X) \\ \vdots \\ \operatorname{tr}(A_k X) \end{pmatrix} \qquad \text{with } A_i \in \mathcal{M}_n \quad i = 1, \ldots, n. \tag{4.1}$$

This representation is also used in [2, 34]. Observe that for X not symmetric the skew–symmetric part of X is mapped to zero, since the trace of the product of a skew-symmetric matrix with a symmetric matrix is zero. The adjoint operator $\mathcal{A}^T(\cdot)$ takes the form

$$\mathcal{A}^T(y) = \sum_{i=1}^{k} y_i A_i \tag{4.2}$$

and can be interpreted as linear combination of symmetric matrices. Let us now take a look at

$$\mathcal{A}(Z^{-1}\mathcal{A}^T(\Delta y)X) = \begin{pmatrix} \operatorname{tr}(A_1 Z^{-1}\mathcal{A}^T(\Delta y)X) \\ \vdots \\ \operatorname{tr}(A_k Z^{-1}\mathcal{A}^T(\Delta y)X) \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^k \operatorname{tr}(XA_1Z^{-1}A_j)\Delta y_j \\ \vdots \\ \sum_{j=1}^k \operatorname{tr}(XA_nZ^{-1}A_j)\Delta y_j \end{pmatrix}. \quad (4.3)$$

The ij–th element of the linear operator on Δy on the right hand side is given by

$$\mathcal{A}_{ij} = \operatorname{tr}(A_i Z^{-1} A_j X). \quad (4.4)$$

It was already pointed out on Page 16 that the final matrix is positive definite. It can be factored efficiently in $O(k^3)$, for practical implementation we used the LDL^T–decomposition of [29]. It is now quite evident that the process of building the system matrix itself can be very expensive, to be more precise $O(k^2n^3)$ steps are needed if the A_i are dense. In comparison to $O(k^3)$ steps for factorizing the matrix this dominates the cost for computing the step direction whenever $k < n^3$. It is therefore extremely important to exploit sparsity or any other structure of the A_i if existent. There is literally no hope that X will be sparse and even if Z is sparse this does not indicate that Z^{-1} will be sparse as well. In practice this currently limits the use of this algorithm to comparatively modest sizes, i.e. k is bounded by at most a few thousands.

However, there may be some hope in another approach for solving interior point methods. In [67] a subspace path–following algorithm was introduced for linear programming. In this approach the previous step directions are accumulated and it is hoped that a good new step direction can be found in the subspace given by these directions. After a number of steps the whole system is solved once again for a centering step. This method seems to work quite well for linear programming, but so far it is not clear whether the concept can successfully be extended to semidefinite programming.

4.4 Polynomial Convergence

As mentioned in Section 1.2 basically the same algorithm was independently studied by Kojima, Shindoh and Hara. As they provide a proof for the polynomiality of the method it seems appropriate to present some of the main results of [39].

In analogy to linear programming Kojima et al. investigate the more general setting of a monotone linear complementarity problem (LCP) in semidefinite matrices. The task is to find a pair of semidefinite matrices (X, Z) such that they lie in a given $n(n+1)/2$ dimensional affine subspace $\mathcal{F}$ of $\mathcal{M}_n^2$ and satisfy the complementarity condition $\operatorname{tr}(ZX) = 0$, i.e.

find a pair $(Z, X) \in \mathcal{M}_n^2$ such that

$$\text{(LCP)} \qquad (X, Z) \in \mathcal{F},\ X \succeq 0,\ Z \succeq 0,\ \text{and } \operatorname{tr}(XZ) = 0.$$

An LCP is monotone if

$$\operatorname{tr}(X' - X)(Z' - Z) \geq 0 \quad \text{for every } (X', Z') \in \mathcal{F} \text{ and } (X, Y) \in \mathcal{F}.$$

To see that the monotone LCP contains semidefinite programming we look at

$$\begin{array}{llll} \text{(PP)} & \text{Maximize} & \langle C, X\rangle \\ & \text{subject to} & \mathcal{A}(X) = a \\ & & X \succeq 0, \end{array} \qquad \begin{array}{llll} \text{(DP)} & \text{Minimize} & \langle a, y\rangle \\ & \text{subject to} & Z + C - \mathcal{A}^T(y) = 0 \\ & & Z \succeq 0. \end{array}$$

In this case the affine subspace is

$$\mathcal{F} = \{X \mid \mathcal{A}(X) = a\} \times \left\{Z \mid Z = \mathcal{A}^T(y) + C,\, y \in \Re^k\right\},$$

and the monotonicity assumption is satisfied with equality,

$$\langle X' - X, Z' - Z\rangle = \left\langle X' - X, \mathcal{A}^T(y' - y)\right\rangle = \langle \mathcal{A}(X' - X), y' - y\rangle = 0.$$

The condition that $\mathcal{F}$ must be $n(n+1)/2$ dimensional corresponds to the Lagrangian interpretation that every restriction on one side must result in additional freedom on the other side.

One of the main results of the paper deals with the existence of the central trajectory for the monotone LCP. Let the set of interior feasible solutions ($(X, Z) \in \mathcal{F}$, $X \succ 0$, and $Z \succ 0$) be denoted by $\mathcal{F}_{++}$. If the set $\mathcal{F}_{++}$ is not empty then the set

$$\mathcal{C} = \{(X_\mu, Z_\mu) \mid XZ = \mu I,\, (X, Z) \in \mathcal{F}_{++},\, \mu > 0\}$$

forms a smooth trajectory (the central trajectory) which converges to a solution of the LCP for $\mu \to 0$. They also characterize (X_μ, Z_μ) as unique minimizer of the logarithmic barrier function

$$\Phi(\mu, X, Y) = \langle X, Z\rangle - \log\det XZ$$

over $\mathcal{F}_{++}$. On proving the existence of the central trajectory they also show the following. Suppose $(X^0, Y^0) \in \mathcal{F}_{++}$ and $\theta \geq \Phi(\mu, X^0, Z^0)$. Then the level set

$$\{(X, Z) \in \mathcal{F}_{++} \mid \Phi(\mu, X, Z) \leq \theta\}$$

is bounded and closed. This assertion has a strong impact on our global convergence proof since we can now drop the assumption of the compactness of $\mathcal{X}$ (see (3.9)).

They propose a general framework for interior point algorithms. This framework is based on the very same principles as our algorithm. Compute a Newton direction for the system using the linearization of $XZ = \mu I$ and symmetrize the step direction. Two parameters determine the specific type of algorithm within general framework. On the one hand this is the amount by which $\mu = \mathrm{tr}(XZ)/n$ should be reduced, they call this parameter the direction parameter $\beta > 0$. On the other hand the step size $\alpha \geq 0$ remains to be selected. The paper also gives an upper bound for the value of α. Let $X \succ 0$, $\Delta X \in \mathcal{M}_n$, then $X + \alpha\Delta X \succ 0$ if and only if

$$\alpha < sup\left\{\alpha \mid 1 + \alpha\lambda_{min}(X^{-1}\Delta X) \geq 0\right\}.$$

Kojima et al. present three different algorithms within this framework, a central trajectory following method, a potential reduction method and an infeasible–interior–point potential reduction method. The first method is very close to our algorithm and we will give some details. Let $\lambda_1, \ldots, \lambda_n$ denote the eigenvalues of XZ. By

$$\mathcal{N}(\gamma) = \left\{(X,Z) \in \mathcal{F}_{++} \;\middle|\; \left(\sum_{i=1}^{n}(\lambda_i - \mu)^2\right)^{\frac{1}{2}} \leq \gamma\mu \ \text{ with } \ \mu = \frac{\mathrm{tr}(ZX)}{n}\right\}$$

a horn neighborhood of the central trajectory is introduced. The width of this neighborhood is controlled by γ. For $\gamma \in (0, 0.1]$ this neighborhood has the convenient property (Theorem 4.1 in [39]) that for $\beta = 1 - \frac{\gamma}{\sqrt{n}}$ the "Newton step" $(X + \Delta X, Z + \Delta Z)$ is again in $\mathcal{N}(\gamma)$. Furthermore the new μ–value of the new point is smaller than the old value of μ by at least a factor of $(1 - \frac{\gamma}{2\sqrt{n}})$. So if $(X^0, Z^0) \in \mathcal{N}(\gamma)$ the algorithm yields a sequence $(X^j, Z^j) \in \mathcal{N}(\gamma)$ such that for every $j = 0, 1, \ldots$

$$\mathrm{tr}(X^j, Z^j) \leq \left(1 - \frac{\gamma}{2\sqrt{n}}\right)^j \mathrm{tr}(X^0 Z^0).$$

Thus if we require $\mathrm{tr}(X^j Z^j) < \varepsilon$ for a given $\varepsilon > 0$ then the algorithm will stop in $O(\sqrt{n}\log\frac{\mathrm{tr}(X^0Z^0)}{\varepsilon})$ iterations.

The algorithms given in [39] are mainly of theoretic interest. For practical applications the computation of eigenvalues is already too expensive and the direction parameter β is too large. However, the convergence proofs indicate that practical implementations are very likely to yield fast and stable algorithms, even if the strict theoretic rules are not closely observed.

4.5 Several Semidefinite Variables

For several applications it may be of interest to use more then only one matrix variable. In theory arbitrarily many matrices can be packed into one big matrix by appropriately defining the linear operators but for practical applications this approach is useless. Fortunately the algorithm extends in a very natural way to several matrix variables. For m matrix variables X^i, each of size $n_i \times n_i$, the primal problem can be formulated as follows,

$$\begin{array}{ll} \text{Maximize} & \sum_{i=1}^m \langle C_i, X_i \rangle \\ \text{subject to} & a - \sum_{i=1}^m \mathcal{A}_i(X) = 0 \\ & X_i \in \mathcal{M}_{n_i}, X_i \succeq 0 \qquad i = 1, \ldots, m. \end{array}$$

No additional complexity is introduced by variables of different dimensions. In fact the case treated so far uses one $n \times n$ matrix and m 1×1 matrices. The dual program is easily found to be

$$\begin{array}{ll} \text{Minimize} & \langle a, y \rangle \\ \text{subject to} & Z_i + C_i - \mathcal{A}_i^T(y) = 0 \qquad i = 1, \ldots, m \\ & Z_i \in \mathcal{M}_{n_i}, Z_i \succeq 0 \qquad i = 1, \ldots, m. \end{array}$$

For the algorithm the following system has to be solved

$$\begin{array}{rcll} \Delta Z_i - \mathcal{A}_i^T(\Delta y) & = & \mathcal{A}_i^T(y) - Z_i - C_i & =: -F_{di} \qquad i = 1, \ldots, m \\ -\sum_{i=1}^m \mathcal{A}_i(\Delta X_i) & = & \sum_{i=1}^m \mathcal{A}_i(X_i) - a & =: -F_p \\ Z_i \Delta X_i + \Delta Z_i X_i & = & \mu I - Z_i X_i & =: -F_{Z_i X_i} \qquad i = 1, \ldots, m. \end{array}$$

In the last row it may be worth thinking about a special barrier parameter μ_i for each i to quicken convergence. Again a defining equation for Δy alone can be extracted,

$$\sum_{i=1}^m \mathcal{A}_i(Z_i^{-1} \mathcal{A}_i^T(\Delta y) X_i) = \sum_{i=1}^m \mathcal{A}_i(\mu Z_i^{-1} - Z_i^{-1} F_{di} X_i) - a.$$

The convergence analysis extends in an obvious way and holds also in the case of several barrier parameters μ_i by treating each complementarity pair $Z_i X_i$ separately.

Part II

Max–Cut Relaxations Using Interior Point Methods

Chapter 5

Introduction to Part II

The max–cut problem asks for a bipartition of the node set of a graph such that the sum of the edges running between both sets is maximized. It is one of the earliest combinatorial problems known to be NP–complete [27]. Several other NP–complete problems can efficiently be transformed to it, in particular $\{0,1\}$–quadratic programming is easily seen to be equivalent to the max–cut problem. Therefore max–cut numbers among the central problems of combinatorial optimization. Practical applications of max–cut can be found in Physics and VLSI–design. An excellent survey on the max–cut problem is [62].

Relaxations of the max–cut problem yield upper bounds on the value of the max–cut and may offer hints for good feasible solutions. Strong upper bounds are indispensable in combinatorial optimization. Large numbers of heuristics for generating "good" solutions for all kinds of combinatorial problems have been developed, but how should one judge the quality of a solution unless the optimal solution or some upper bound on it is known? For some applications, e.g. for the spin glass model in Physics, only optimal solutions provide the necessary information. In this case enumeration by branch and bound cannot be avoided. Branch and bound methods heavily rely on good upper bounds to cut off large branches of the branch and bound tree. Better bounds can reduce the number of branches dramatically. Since in general the bound is evaluated for each subproblem computing time is clearly of importance and faster but less accurate bounds may be needed. The algorithms introduced here yield very good results in exhaustive computations as well as in the computation of comparatively fast high quality bounds for complete graphs.

Two successful approaches for computing upper bounds for max–cut can be found in the literature. The historically first is the polyhedral approach [6, 9, 11, 20, 21, 22, 63, 7]. The basic object of interest is the cut polytope, the convex hull of an $\binom{n}{2}$–dimensional representation of the feasible solutions. Several properties of the cut polytope were studied and many types of facet–defining inequalities were detected, yet complete descriptions of the cut polytope are only known for sizes up to $n = 7$ where n denotes the number of vertices. In combination with cutting plane algorithms facet–defining inequalities yield powerful relaxations of the max–cut polytope. A useful relaxation is formed by the triangle facets. Following [62] we will denote this upper bound by $\sigma(G)$. Several results indicate that $\sigma(G)$ is a very strong bound for certain classes of graphs. Although $\sigma(G)$ is in theory computable in polynomial time, the numerical evaluation of $\sigma(G)$ is itself a very hard problem in practice. For complete graphs $\sigma(G)$ was successfully computed only up to $n = 30$.

The second approach is based on the largest eigenvalue λ_{max} of the Laplacian L of the graph G, L is a matrix closely related to the adjacency matrix of the graph. In [18] Delorme and Poljak introduced the bound $\varphi(G)$ by optimizing λ_{max} over an additional correction vector on the main diagonal of L. The bound is known to be asymptotically optimal for a large class of random graphs and recently a very strong bound was proved for positively weighted graphs in [28]. Clearly the evaluation of the bound requires nonlinear optimization. Since the equivalence of $\varphi(G)$ to a positive semidefinite program was shown in [58] it is also possible to evaluate the bound exactly in practice by means of interior point methods [64, 32].

It was suggested in [58] to combine $\varphi(G)$ with facet defining inequalities, in particular with $\sigma(G)$. However, the traditional simplex approach proved inefficient. A new approach for combining semidefinite programming with cutting plane methods was needed. To this end the interior point method of Part I was developed. With this interior point algorithm it is possible to compute the new bound for complete graphs of sizes up to $n = 50$ within one hour on a DEC-station 5000/240 and even sizes of $n = 100$ are manageable. The approach offers new possibilities for comparatively fast computation of high quality bounds and can easily be extended to arbitrary (facet–defining) inequalities.

We give a short outline of the contents of Part II. In Section 5.1 some basic facts about the max–cut problem are listed. Section 5.2 of this Chapter contains some applications of max–cut. In Chapter 6 the polyhedral results concerning the cut polytope and the bound $\sigma(G)$ are summarized. The eigenvalue bound $\varphi(G)$ is introduced in Chapter 7. There we will also discuss the appropriate choice of the algorithm and give some numerical results on $\varphi(G)$. In Chapter 8 we will combine $\varphi(G)$ first with triangle inequalities, then with clique inequalities in general and discuss numerical results.

5.1 The Max–Cut Problem

For notation we keep close to the terminology used in [62], where also the references to the theorems of this section can be found. $G(V, E)$ denotes a finite, undirected, loopless graph on vertex set V and edge set E. The vertex set is always $V = \{1, \ldots, n\}$, therefore $|V| = n$. An edge $ij \in E$ connects vertices i and j. The graph is said to be weighted if in addition a weight function $c : E \to \Re$ is defined on the edges. The weight assigned to edge ij is denoted by c_{ij}. We will also use weights c_{ij} for unweighted graphs, in which case $c_{ij} = 1$ for $ij \in E$, and whenever an edged ij is not part of the edge set we set $c_{ij} = 0$. Sometimes c is used as a cost vector (c_{ij}) indexed by the edges.

For a subset $S \subset V$ of vertices the cut induced by S is the set of edges having one endpoint in S and the other in $V \setminus S$, it is denoted by

$$\delta(S) := \{ij \in E | i \in S, j \notin S\}.$$

The weight $c(F)$ of a subset $F \subset E$ of the edges is the sum of the weights of the edges in F. In particular, the weight $c(\delta(S))$ of a cut is given by

$$c(\delta(S)) = \sum_{ij \in \delta(S)} c_{ij}.$$

The max–cut problem asks for a set $S \subset V$ such that $c(\delta(S))$ is maximal. The weight of a maximal cut is denoted by

$$mc(G, c) := \max_{S \subset V} c(\delta(S)).$$

If it is clear which cost vector c is used we will write $mc(G)$. It is well known that the max–cut problem is NP–complete. In fact it was one of the six basic problems which appeared in the list of Karp [36]. The problem is very well studied in the literature and numerous results on lower bounds and expected size for different classes of graphs are known. We will list but a few basic properties to outline the complexity of the problem.

The following result, due to Yannakakis, illustrates that even under severe restrictions the problem remains untractable.

Theorem 4 *(Yannakakis)*
The max–cut problem is NP-complete for unweighted graphs with maximum degree three.

For planar graphs, however, the problem is "easy",

Theorem 5 *(Orlova,Dorfman;Hadlock)*
The max-cut problem is polynomially solvable on planar graphs.

A trivial lower bound on the size of the max–cut is given by

$$mc(G) \geq \frac{1}{2}c(E).$$

This is indeed easily verified by observing that any vertex which does not have at least half of its edgeweights in the cut can be "switched" to the other side thereby increasing the value of the cut. This bound can be improved to

Theorem 6 *(Poljak, Turzik)*
In every weighted graph $G(V, E)$,

$$mc(G) \geq \frac{1}{2}c(E) + \frac{1}{4}\min_{T \subset G} c(T),$$

where the minimum is taken over all spanning trees T of G. Moreover, the bound can be attained by an algorithm of running time O(mn).

Equality holds for unweighted complete graphs of odd order. An indication that it is difficult to develop algorithms yielding good bounds for the size of the max–cut is

Theorem 7 *(Arora, Lund, Motwani, Sudan, Szegedy)*
There is no polynomial–time approximation scheme for max–cut on unweighted graphs unless P=NP.

A polynomial time approximation scheme is a polynomial–time ε–approximation for a given $\varepsilon > 0$, i.e. a polynomial–time algorithm with output $alg(G)$ such that

$$\frac{|alg(G) - mc(G)|}{mc(G)} \leq \varepsilon.$$

5.2 Applications of Max–Cut

Although the max–cut problem looks a little academic at first sight a large number of applications can be found in the literature. We will present just a few to demonstrate the generality of the problem. For more applications the reader is referred to [62] and the references therein.

5.2.1 Ising Model of Spin Glasses

This application from Physics is studied in [8]. Spin glasses are experimental objects for investigating order and disorder phenomena, in particular the magnetic orientation of certain atoms in the spin glass at certain temperatures. At 0°K the orientation will be such that the whole system has minimum energy. Such a configuration is called groundstate. Physical experiments at 0°K are impossible and so mathematical models are the only means for determining feasible configurations. The Ising model reduces the possible magnetic orientations of atoms to either "+1" or "−1". A two dimensional model can be represented by a grid graph where the nodes correspond to the atoms and edges to the interactions between the atoms. The weight of the edges is used to model the strength and type of interaction. A popular choice is a rectangular grid (only neighbors influence each other) in the form of a torus. An exterior magnetic field is modeled by an additional vertex connected to all other vertices. A ground state corresponds to a maximum cut in this graph, the partition of the node set corresponds to atoms having orientation $+1$ and -1, respectively.

5.2.2 VLSI–Design

"Very Large Scale Integration"–design of computer chips is well known to offer a wide range of applications for all kinds of combinatorial problems [42]. The max–cut problem appears for instance in via minimization during the routing phase. In case of printed circuit boards this corresponds to minimizing the number of holes one has to drill to get from one side of the board to the other. In this particular case the maximum cut has to be found for planar graphs and is thus polynomially solvable [42]. Barahona et al. describe a slightly more general case where the underlying graph is not planar any more [8].

5.2.3 Radius of Nonsingularity

In [61] Poljak and Rohn give a very surprising application of the max–cut problem. For two given $n \times n$ matrices A and Δ they introduce the radius of nonsingularity

$$d(A, \Delta) := \min \{\varepsilon \geq 0 \mid A - \varepsilon\Delta \leq A' \leq A + \varepsilon\Delta \text{ for some singular } A'\},$$

where $\leq$ is a componentwise comparison. This radius may be infinite and it is zero if and only if A is singular. For $\Delta = J := ee^T$, the matrix of all ones, the radius of nonsingularity is denoted by $d(A)$ and can be interpreted as follows. Suppose the data of A is perturbed to $\tilde{A}$. If all elements are changed by less than $d(A)$ the new matrix $\tilde{A}$ cannot be singular. Thus $d(A)$ offers a good measurement for the robustness of A against becoming singular.

For matrices A with rational entries Poljak and Rohn show that computing $d(A)$ is equivalent to solving the max–cut problem. Furthermore the transformations are indeed computationally cheap so that computing $d(A)$ can be seen as an application of max–cut.

5.2.4 $\{0,1\}$–Quadratic Programming

$\{0,1\}$–quadratic programming is a well established discrete optimization problem, yet there seems to be no standard notation. We will use the formulation

$$qp(Q) := \max_{y\in\{0,1\}^n} y^t Q y$$

with Q being an arbitrary symmetric $n\times n$ matrix. Since y is a $\{0,1\}$–vector a linear term can always be added to the diagonal of Q. $\{0,1\}$–quadratic programming is well known to be equivalent to the max–cut problem. The standard trick [10] is to replace y by a $\{-1,1\}$–vector $x := 2y-e$ and to introduce a new $\{-1,1\}$–variable x_0 for the linear part.

$$\begin{aligned} y^TQy &= \frac{1}{4}(x+e)^TQ(x+e) \\ &= \frac{1}{4}x^TQx + \frac{1}{2}e^TQx + \frac{1}{4}e^TQe \\ &= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{q_{ij}}{2}x_ix_j + \sum_{i=1}^{n}\left(\frac{1}{2}\sum_{j=1}^{n}q_{ij}\right)x_0x_i + \frac{1}{4}e^TQe + \frac{1}{4}\mathrm{tr}(Q) \end{aligned}$$

We may introduce x_0 in this way since now x and $-x$ yield the same objective value and we can fix x_0 to 1. We associate each variable with a vertex of a new graph G and define the preliminary weights of the edges by

$$w_{ij} := \begin{cases} \dfrac{q_{ij}}{2} & \text{for } 1\le i<j\le n \\ \dfrac{1}{2}\displaystyle\sum_{k=1}^{n} q_{jk} & \text{for } i=0,\ 1\le j\le n. \end{cases}$$

Next we partition the vertex set according to the $1/-1$ pattern. W.l.o.g. let S denote the vertex set corresponding to value 1, then

$$\begin{aligned} y^tQy &= \sum_{i=0}^{n-1}\sum_{j=i+1}^{n} w_{ij} - 2\sum_{ij\in\delta(S)} w_{ij} + \frac{1}{4}e^TQe + \frac{1}{4}\mathrm{tr}(Q) \\ &= \sum_{ij\in\delta(S)} -2w_{ij} + e^TQe \end{aligned}$$

Thus if we define the following weights c_{ij} for the new graph

$$c_{ij} := -2w_{ij} = \begin{cases} -q_{ij} & \text{for } 1\le i<j\le n \\ -\displaystyle\sum_{k=1}^{n} q_{jk} & \text{for } i=0,\ 1\le j\le n, \end{cases}$$

then we have the very convenient equality

$$qp(Q) = mc(G,c) + e^TQe.$$

Chapter 6

Polyhedral Approach

For linear integer programming the field of polyhedral theory, initiated in the early seventies, offered new possibilities to get a handle on untractable problems. For some practical applications, such as the Traveling Salesman Problem or several other special instances of NP–complete problems, improvements were indeed astonishing. The polyhedral aspects of the max–cut problem are also well studied in the literature (see e.g. [6, 9, 11, 20, 21, 22, 63, 7]). In practical experiments the polyhedral approach was very successful on almost planar graphs but only moderately successful on general graphs.

In Section 6.1 the basic terminology and the cut polytope are introduced. Section 6.2 lists some important facets of the cut polytope. Section 6.3 deals with some properties of the metric polytope, the polytope induced by the "simplest" facets of the cut polytope. Again, the presentation is very much inspired by [62].

6.1 The Cut Polytope P_n^{cut}

In integer programming the set of feasible solutions can be seen as a family of points in a space of appropriate dimension. In the case of the max–cut problem a feasible solution for a given graph $G(V, E)$ is uniquely described by the characteristic vector $\mathcal{X}^S$ (also called cut vector) of the cut $\delta(S)$ induced by some $S \subset V$,

$$\mathcal{X}_{ij}^S = \begin{cases} 1 & ij \in \delta(S) \\ 0 & \text{otherwise.} \end{cases} \tag{6.1}$$

The basic idea of polyhedral theory is to relax the feasible set to the convex hull of all feasible points. In this case the optimal value of the relaxed linear program, if existent, is the optimal value attained on the original set. If in addition the algorithm stops in a vertex of the optimal face of the convex hull, as the simplex algorithm does, this is also an optimal solution of the original problem.

The convex hull of all cut vectors $\mathcal{X}^S$ is the so called cut polytope

$$P^{cut}(G) := \operatorname{conv}\left\{\mathcal{X}^S | S \subset V\right\}.$$

By the considerations above it is obvious that

$$mc(G) = \max\left\{c^T x | x \in P^{cut}(G)\right\},$$

with c being the cost vector of G. The somewhat less obvious fact that every cut vector $\mathcal{X}^S$ is indeed a vertex of the cut polytope can be seen by observing that $\mathcal{X}^S$ is the unique optimum for the cost vector

$$c_{ij} = \begin{cases} 1 & \mathcal{X}^S_{ij} = 1 \\ -1 & \text{otherwise.} \end{cases}$$

It is obvious that $P^{cut}(G)$ is contained in the unit cube $[0,1]^E$ and it was shown in [9] that it is full dimensional. In our setting the cut polytope of the complete graph K_n on n vertices is of special importance, it will be denoted by P^{cut}_n.

Lemma 8 *(Barahona, Mahjoub[11])*
Each pair of vertices of P^{cut}_n is adjacent.

Proof: Let $R, S \subset V$ be such that $R \neq S$, $R \neq V \setminus S$. Consider the cost vector c given by

$$c_{ij} = \begin{cases} 1 & ij \in \delta(R) \cap \delta(S) \\ 0 & ij \in \delta(R) \triangle \delta(S) \\ -1 & \text{otherwise.} \end{cases}$$

Then $\max c^T x$ is attained only for the vertices $\mathcal{X}^R$ and $\mathcal{X}^S$. □

This together with the fact that P^{cut}_n has 2^{n-1} vertices shows that P^{cut}_n is highly degenerate. It is also a highly symmetrical object, a fact which may also be of practical importance. By a result of Barahona and Mahjoub any vertex $\mathcal{X}^S$ can be mapped to any other vertex $\mathcal{X}^R$ by an automorphism. This operation is also called the switching of $\mathcal{X}^S$ to $\mathcal{X}^R$. It suffices to show this for switching $\mathcal{X}^\emptyset$ to $\mathcal{X}^S$, which is defined by $r_S : \Re^E \to \Re^E$ with $r_S(\xi) = \xi^S$ where

$$\xi^S_{ij} = \begin{cases} 1 - \xi_{ij} & ij \in \delta(S) \\ \xi_{ij} & \textit{otherwise.} \end{cases} \tag{6.2}$$

For all $S \subset V$ we have $r_S(P^{cut}_n) = P^{cut}_n$, $r_S(\mathcal{X}^\emptyset) = \mathcal{X}^S$, and $r_S(\mathcal{X}^S) = \mathcal{X}^\emptyset$. Since P^{cut}_n is invariant to this type of mapping it is sufficient to transform the cost vector c to c^S.

Lemma 9 *([11])*
For every $R, S \subset V$,

$$c^S(\delta(R \triangle S)) = c(\delta(R)) - c(\delta(S))$$

and hence

$$mc(G, c^S) = mc(G, c) - c(\delta(S)).$$

Note, that the transformed problem differs only by a constant from the original problem.

6.2 Facet–Defining Inequalities

Since the max–cut problem is NP–complete there is no hope that a "useful" polyhedral description can be found for the cut polytope, i.e. a complete description by linear inequalities where the number of inequalities is bounded by a polynomial in the size of the underlying graph. Therefore the approach generally taken is to start with an easy relaxation of the problem, for instance the unit cube, and to successively add further inequalities as needed. These inequalities should continuously tighten the relaxed set. Ideally this process ends with a vertex of the desired polytope. This could indeed be guaranteed if a complete description of the polytope was known. It is evident that a complete description of a polytope must include its facets, the facets also form the minimal description. Numerous classes of facets of the cut polytope can be found in the literature (see the survey [20, 21]). Nonetheless complete descriptions are known only for sizes $n \leq 7$. Since we will always work on the complete graph we will just list those facets of P_n^{cut} which are related to our work.

We give the obligatory definitions. A valid inequality for a polytope P is an inequality $a^T x \leq b$ which is valid for all $x \in P$. If for a certain valid inequality $a^T x \leq b$ the set $F = P \cap \left\{a^T x = b\right\}$ is neither the empty set nor the whole polytope, it is called a face F of P. Set–inclusion defines a partial order on the faces, the maximal faces are called facets. If the face induced by a valid inequality $a^T x \leq b$ is a facet, the inequality is called facet–defining or just a facet of P. To prove that an inequality $a^T x \leq b$ is facet–defining one has to show that the inequality is valid and that the dimension of the induced face is sufficiently large. The latter can be achieved for instance by showing that a sufficient number (in our case $\binom{n}{2}$) of vertices is contained in the face.

Probably the simplest facets are the triangle inequalities

Theorem 10 *(see e.g. [62]) For any triple $i, j, k \in V$ of distinct elements, the inequality*

$$x_{ij} + x_{ik} + x_{jk} \leq 2 \tag{6.3}$$

is a facet of P_n^{cut}.

The standard interpretation of this inequality is that a cut can contain at most two edges of a triangle ijk because at least two of the vertices must belong to the same set of the partition. The switching (6.2) of a facet $a^T x \leq b$ yields another facet $a^S x \leq b - a(\delta(S))$. Using this operation we get the corresponding homogeneous triangle inequalities

$$\begin{aligned} x_{ij} - x_{ik} - x_{jk} &\leq 0 \\ -x_{ij} + x_{ik} - x_{jk} &\leq 0 \\ -x_{ij} - x_{ik} + x_{jk} &\leq 0. \end{aligned}$$

A triangle inequality is defined over a clique on three vertices i, j, and k. The class of clique inequalities generalizes this concept. For a complete subgraph K_k of K_n

$$\sum_{ij \in K_k} x_{ij} \leq \left\lfloor \frac{k^2}{4} \right\rfloor$$

since $mc(K_k) = \lceil k/2 \rceil \lfloor k/2 \rfloor$. Obviously the use of the floor function has some effect only for k odd.

Theorem 11 *(Barahona,Grötschel,Mahjoub [9])*
The clique inequality is a facet when k is odd. For k even, however, it is not a facet (as it can be obtained as the sum of the $(k-1)$–clique inequalities).

The clique inequalities can be generalized even further by the so called hypermetric inequalities,

$$\sum_{1 \leq i < j \leq n} b_i b_j x_{ij} \leq \frac{1}{4} \left(\left(\sum_{i=1}^{n} b_i \right)^2 - 1 \right) \tag{6.4}$$

with $b_1, b_2, \ldots, b_n$ integers and $\sum_{i=1}^{n} b_i$ odd. The clique inequalities of size k, odd, are obtained by setting k variables b_i to either -1 or 1 and all the others to zero. All hypermetric inequalities are valid for P_n^{cut} but only some of them are facet–defining. Several other classes of facet–defining inequalities are known, see [20, 21] for a survey.

6.3 The Metric Polytope P_n^{met}

In practical applications it is not possible to consider all known classes of facet–defining inequalities, it may be quite difficult to find violated inequalities within even only one class. On the other hand adding a lot of violated inequalities in one rush is definitely not a good strategy since this will slow down the algorithm considerably and in the end several inequalities will turn out to be redundant for this specific instance anyways. Therefore one tries to select only a small set of powerful facets which should be easy to compute.

It is the general opinion that for the cut polytope the triangle inequalities

$$\begin{aligned} x_{ij} + x_{ik} + x_{jk} &\leq 2 \\ x_{ij} - x_{ik} - x_{jk} &\leq 0 \\ -x_{ij} + x_{ik} - x_{jk} &\leq 0 \\ -x_{ij} - x_{ik} + x_{jk} &\leq 0 \end{aligned}$$

for $1 \leq i < j < k \leq n$ form a very good starting set of facets. For complete graphs the polytope described by the triangle inequalities is called the metric polytope P_n^{met}. It is also very well studied and we list some of its major properties.

Obviously P_n^{met} is a relaxation of P_n^{cut}, since all triangle inequalities are facets, but in general not all facets are triangle inequalities,

$$\sigma(G, c) := \max_{x \in P_n^{met}} c^T x \geq \max_{x \in P_n^{cut}} c^T x = mc(G, c).$$

Again we will write $\sigma(G)$ if it is clear which c is meant. The relaxation makes some sense since the metric polytope is fully contained in the unit cube, i.e.

$$0 \leq x \leq 1 \qquad \forall x \in P_n^{met}.$$

This is easily obtained by summing the appropriate inequalities to get a bound for each component of x. On the other hand there are only $4\binom{n}{3}$ triangle inequalities, so $\sigma(G)$ can be computed in polynomial time. Trivial lower and upper bounds on $\sigma(G)$ are

$$\frac{2}{3} \sum_{ij \in E} c_{ij} \leq \sigma(G) \leq \sum_{ij \in E} c_{ij},$$

since the point $(\frac{2}{3}, \frac{2}{3}, \ldots, \frac{2}{3})$ is contained in P_n^{met} and $x_{ij} \leq 1$.

6.3.1 The Relation of P_n^{met} to P_n^{cut}

Clearly, all vertices of P_n^{cut} are contained in P_n^{met} and since they are vertices of the unit cube, they must also be vertices of P_n^{met}. All other vertices of the unit cube violate some triangle inequality.

Lemma 12 *(Barahona,Mahjoub [11])*
Every characteristic vector $\mathcal{X}^S$ of a cut $\delta(S)$ in K_n is a vertex of the metric polytope P_n^{met}, and there are no other integral vertices.

However, little is known about the fractional vertices of P_n^{met}. Another property linking P_n^{met} closely with P_n^{cut} is that low–dimensional faces of P_n^{cut} remain faces of P_n^{met}.

Theorem 13 *(Deza,Laurent,Poljak [22])*
For $d = 1, 2, 3$, any d distinct cut vectors determine a simplicial face of both P_n^{met} and P_n^{cut}.

A similar result has also been shown in [22] for some larger dimensional faces and they conjecture that all faces of dimension $d < \log_2 n$ of P_n^{cut} are also faces of P_n^{met}.

The barycenter of the cut polytope is the point $(\frac{1}{2}, \frac{1}{2}, \ldots \frac{1}{2})$. A possible indication on the effectiveness of an inequality might be its distance to the barycenter of the cut polytope, since an inequality close to the barycenter cuts off a large part of the unit cube. For the triangle inequalities the distance is $\frac{1}{2\sqrt{3}}$ and it is conjectured in [22] that this is the minimum value of all facets. The conjecture is based on the following result.

Theorem 14 *(Deza, Laurent, Poljak [22]) Let $a^T x \leq b$ be a facet of the cut polytope P_n^{cut} such that $a \in \{0, 1, -1\}^{\binom{n}{2}}$. Then the distance of the hyperplane $ax = b$ from the barycenter $(\frac{1}{2}, \frac{1}{2}, \ldots \frac{1}{2})$ is at least $\frac{1}{2\sqrt{3}}$.*

6.3.2 On the Quality of the Bound $\sigma(G)$

We will next discuss some of the properties of the upper bound $\sigma(G)$. We have already pointed out (see Theorem 5) that the max–cut problem is polynomially solvable on planar graphs. Of course it is desirable that an upper bound is also exact for the easy case. We will see that $\sigma(G)$ is exact for an even larger class of graphs.

The metric polytope $P^{met}(G)$ of a general graph $G(V, E)$ on n vertices is defined somewhat differently. However, by a result of Barahona [7], $P^{met}(K_n) = P_n^{met}$ and $P^{met}(G)$ is the projection of P_n^{met} to the subspace $\Re^E \subset \Re^{\binom{n}{2}}$. For graphs G not contractible to K_5 it was proved that $P^{cut}(G) = P^{met}(G)$ [66, 6]. Because $P^{met}(G)$ is the projection of P_n^{met} we have

$$mc(G, c) = \max_{x \in P^{cut}(G)} c^T x = \max_{x \in P^{met}(G)} c^T x = \max_{x \in P_n^{met}} c^T x = \sigma(G).$$

$\sigma(G)$ is polynomially solvable and so the class of polynomially solvable graphs can be extended by the following well known corollary.

Corollary 15 *The max–cut problem is solvable in polynomial time on graphs not contractible to K_5.*

Thus $\sigma(G)$ is tight for all cost vectors with support $\{ij | c_{ij} \neq 0\}$ not contractible to K_5.

Another aspect of interest is the asymptotic behavior of the bound on certain probabilistic graph models. A graph G of the class $G_{n,p}$ of random graphs is defined on n vertices and each edge ij is contained in E with probability p. The following theorems by Poljak and Tuza state the expected relative error for "dense" and "sparse" graphs.

Theorem 16 *(Poljak, Tuza [63])*
Let $p = p(n)$ be a function such that $0 < p < 1$ and $p(n) = \Omega(\sqrt{\frac{\log n}{n}})$. Then the expected relative error

$$\frac{\sigma(G_{n,p}) - mc(G_{n,p})}{mc(G_{n,p})} \tag{6.5}$$

tends to $\frac{1}{3}$ with probability $1 - o(1)$ as $n \to \infty$.

The result for sparse graphs reads

Theorem 17 *(Poljak, Tuza [63])*
Let $p = p(n)$ be a function such that $0 < p < 1$, $p(n) \cdot n \to \infty$, and $p(n) \cdot n^{1-\alpha} \to 0$ for every $\alpha > 0$. Then the relative error (6.5) tends to 1 with probability $1 - o(1)$ as $n \to \infty$.

Since $|E|/2$ is a trivial lower bound for $mc(G)$ the latter theorem implies that $\sigma(G)$ is asymptotically not better than the trivial upper bound $|E|$ for this class of sparse graphs.

6.3.3 Cutting Plane Algorithms

Several cutting plane algorithms for max–cut make use of triangle inequalities. As already outlined before the simplex algorithm is the basic optimization tool. Most cutting plane algorithms follow the same framework: The algorithm can be started with the trivial bounds $0 \leq x_{ij} \leq 1$, i.e. the feasible set is the unit cube. Then several violated inequalities, preferably facets are determined. Select an efficient set of these inequalities by some criterion, e.g. the amount of violation. Reoptimize, delete old inequalities now obsolete (having dual cost close to zero) and select new violated inequalities. At the same time one can try to construct good feasible solutions from the current iterate x. Repeat the process until either the gap between feasible solution and bound is small enough, no more violated inequalities can be found, or some iteration or time limit is exceeded.

Only few results have been published where triangle inequalities have been used exclusively, mostly they are combined with general hypermetric inequalities. The approach has been quite successful on some special graph models like toroidal grids with one additional vertex adjacent to all other vertices and on so called Carter and Williams type instances of $\{0, 1\}$–quadratic optimization [8, 10]. Even for large instances (about 5000 edges) the algorithm stopped with integral solution vectors x, which implies that the optimal solution of the max–cut problem has been found. For weighted complete graphs only [16] is known to us. They report that for graphs of size $n \leq 30$ with a single exception all solution vectors x were found to be integral. The small size of the experiments is due to the fact, that, because of the degeneracy of P_n^{met}, the simplex algorithm runs into serious difficulties. This was also observed in our experiments (see Section 8.5).

Chapter 7

Eigenvalue Bound

It is a well known fact that the eigenvalues of several matrix representations of graphs are closely related to certain combinatorial properties of the graphs (see e.g. [15, 14, 50]). In particular the extremal eigenvalues often provide upper or lower bounds for combinatorial optimization problems. In a more recent development these bounds are tightened considerably by eigenvalue optimization [43, 17]. Both approaches are based on the observation that a number of perturbations may be performed on the matrix representing the graph without affecting the validity of the upper bound based on the maximal eigenvalue of the matrix. Consequently the bound can be improved by minimizing the largest eigenvalue over all feasible perturbations. In [17] Delorme and Poljak introduce an upper bound $\varphi(G)$ of this type for the max–cut problem. It was observed in [58] that there is a very natural equivalent dual formulation of this bound, which immediately leads to positive semidefinite programming [64].

Section 7.1 introduces the upper bound $\varphi(G)$, Section 7.2 lists some important properties of this bound, and Section 7.3 gives some computational results using the interior point approach for semidefinite programming of Part I.

7.1 The Upper Bound $\varphi(G)$

Let $G(V,E)$ be a weighted, undirected graph on vertex set $\{1,2,\ldots,n\}$ and A its weighted adjacency matrix. We represent a partition of V into two sets $S \subset V$ and $V \setminus S$ by a partition vector $x \in \{-1,+1\}^n$ with

$$x_i = \begin{cases} -1 & i \in S \\ +1 & i \in V \setminus S. \end{cases}$$

Obviously $(x_i - x_j)^2$ is zero if i and j belong to the same side and it is equal to 4 if they belong to different sides of the partition. Therefore,

$$mc(G) = \max_{x \in \{-1,+1\}^n} \frac{1}{4} \sum_{1 \le i \le j \le n} a_{ij}(x_i - x_j)^2.$$

To get a concise representation we observe

$$\begin{aligned} \sum_{1 \le i \le j \le n} a_{ij}(x_i - x_j)^2 &= \sum_{1 \le i \le j \le n} a_{ij}(x_i^2 + x_j^2) - \sum_{1 \le i \le j \le n} 2a_{ij}x_i x_j \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}x_i^2 - \sum_{i=1}^{n} a_{ii}x_i^2 - \sum_{1 \le i < j \le n} 2a_{ij}x_i x_j \\ &= x^T \left(\mathrm{Diag}(Ae) - A\right) x. \end{aligned}$$

The symmetric matrix

$$L(G) := \mathrm{Diag}(Ae) - A \tag{7.1}$$

is called the Laplacian of G. If it is clear which graph G is meant we will write L instead of $L(G)$. The diagonal entry l_{ii} is the weighted degree of vertex i, and for positively weighted graphs L is positive semidefinite. For positively weighted graphs several connections between the extremal eigenvalues of L and properties of the graph have been established (see e.g. [26, 15, 14, 50, 31]). For the max–cut problem the following equality is now trivial but vital,

$$mc(G) = \frac{1}{4} \max_{x \in \{-1,+1\}^n} x^T L x. \tag{7.2}$$

Depending on how a relaxation of the right hand side is constructed we get several different upper bounds for $mc(G)$. We will introduce only two different variants which are useful for our purposes and show their equivalence via duality.

7.1.1 The Dual (Eigenvalue) Approach

For historical reasons we start with the eigenvalue approach. Looking at (7.2) the Rayleigh–Ritz characterization of the largest eigenvalue immediately comes into mind. Before relaxing the set, however, it is important to introduce some additional variability. Since for $x \in \{-1,+1\}^n$ and $u \in \Re^n$ with $e^T u = 0$

$$x^T \mathrm{Diag}(u) x = \sum_{i=1}^{n} u_i x_i^2 = e^T u = 0,$$

we have

$$\max_{x \in \{-1,+1\}^n} x^T L x = \min_{u \in \Re^n, e^T u = 0} \max_{x \in \{-1,+1\}^n} x^T (L + \mathrm{Diag}(u)) x.$$

The vector u is often referred to as correcting vector. Indeed, this at the first glance useless addition is the key to success. The term $\mathrm{Diag}(u)$ introduces the possibility of diagonal perturbations, which definitely have some influence if the set $\{-1,+1\}^n$ is relaxed. To get the eigenvalue formulation the relaxation to be used is obviously the unit sphere. Together with the necessary normalization this leads to the upper bound

$$\varphi(G) := \frac{n}{4} \min \left\{ \lambda_{max}(L(G) + \mathrm{Diag}(u)) \mid u \in \Re^n,\ e^T u = 0 \right\} \tag{7.3}$$

for the max–cut problem. Observe that $\lambda_{max}(L + \mathrm{Diag}(u))$ is an upper bound for every feasible u.

7.1.2 The Primal (Trace) Approach

This time we start by exploiting the fact that $\mathrm{tr}(AB) = \mathrm{tr}(BA)$ for arbitrary matrices A and B of dimensions $m \times k$ and $k \times m$, respectively. Clearly we have

$$x^T L x = \mathrm{tr}(x^T L x) = \mathrm{tr}(L x x^T).$$

For $x \in \{-1,+1\}^n$ the dyadic product xx^T is a positive semidefinite rank one matrix having all diagonal elements equal to one. On the other hand let X be a positive semidefinite rank one matrix having all diagonal entries equal to one, then it is easy to see that X must be the dyadic product of some vector $x \in \{-1,+1\}^n$. Therefore the optimization problem (7.2) is equivalent to the following positive semidefinite program,

$$\begin{array}{rl} \text{Maximize} & \frac{1}{4}\langle L, X\rangle \\ \text{subject to} & e - \mathrm{diag}(X) = 0 \\ & \mathrm{rank}(X) = 1 \\ & X \succeq 0. \end{array} \tag{7.4}$$

Since it is not known how to optimize over rank one matrices we simply drop this constraint and get the next upper bound

$$\begin{array}{rl} \varphi'(G) := \text{Maximize} & \frac{1}{4}\langle L, X\rangle \\ \text{subject to} & e - \mathrm{diag}(X) = 0 \\ & X \succeq 0. \end{array} \tag{7.5}$$

for the max-cut problem. Note, that the upper bound property is only guaranteed for the optimum and not for all feasible solutions.

7.1.3 Equivalence of Primal and Dual Approach

We will show the equivalence of (7.3) and (7.5) by transforming (7.3) into the dual semidefinite program of (7.5). Since $(A - \lambda_{max}(A)I)$ is negative semidefinite for a given symmetric matrix A, (7.3) is equivalent to

$$\begin{array}{rl} \text{Minimize} & \frac{n}{4}\lambda \\ \text{subject to} & \lambda I - L - \mathrm{Diag}(u) \succeq 0 \\ & e^T u = 0. \end{array} \tag{7.6}$$

We merge the diagonal matrices λI and $-\mathrm{Diag}(u)$ and get the diagonal matrix $\mathrm{Diag}(\lambda e - u)$. We give a new name to the argument,

$$y := \lambda e - u,$$

and observe that

$$e^T y = n\lambda.$$

By substituting the new variable into (7.6) we get

$$\begin{array}{rl} \text{Minimize} & \frac{1}{4}e^T y \\ \text{subject to} & \mathrm{Diag}(y) - L \succeq 0. \end{array}$$

Finally we introduce the positive semidefinite slack variable Z for a positive semidefinite program equivalent to (7.3), the dual program

$$(DP) \qquad \begin{array}{rl} \text{Minimize} & \left\langle \frac{1}{4}e, y \right\rangle \\ \text{subject to} & Z + L - \mathrm{Diag}(y) = 0 \\ & Z \succeq 0. \end{array} \tag{7.7}$$

We note that the adjoint operator to $\mathrm{Diag}(\cdot)$ is the operator $\mathrm{diag}(\cdot)$ since for an $n \times n$ matrix A and $y \in \Re^n$

$$\langle A, \mathrm{Diag}(y) \rangle = \sum_{i=1}^{n} a_{ii} y_i = \langle \mathrm{diag}(A), y \rangle .$$

By Part I the corresponding primal program is clearly

$$(PP) \qquad \begin{array}{rl} \text{Maximize} & \langle L, X \rangle \\ \text{subject to} & \frac{1}{4}e - \mathrm{diag}(X) = 0 \\ & X \succeq 0, \end{array} \tag{7.8}$$

which is just (7.5) with X scaled by $\frac{1}{4}$. Obviously, $X = \frac{1}{4}I$ is an interior point for (7.8) and the feasible set of (7.8) is compact. Therefore strong duality holds and the optimum is finite. This proves that

$$\varphi(G) = \varphi'(G),$$

and in the sequel only $\varphi(G)$ will be used.

7.2 Some Properties of $\varphi(G)$

In [40] Laurent and Poljak note that the (scaled) feasible set of the primal problem as stated in (7.8)

$$\tilde{\mathcal{L}}_n := \{X \mid X \succeq 0, \operatorname{diag}(X) = e\} \tag{7.9}$$

is also known as the set of correlation matrices and they investigate in detail the convex body formed by the vectors corresponding to the upper triangles (extracted by the operator $\tau : \mathcal{M}_n \rightarrow \Re^{\binom{n}{2}}$, $\tau(X) = (x_{ij})_{i<j}$)

$$\mathcal{L}_n := \left\{\tau(X) | X \in \tilde{\mathcal{L}}_n\right\}, \tag{7.10}$$

which they call an elliptope. Obviously, the elliptope is convex and contained in $[-1, 1]^{\binom{n}{2}}$ since the absolute value of an offdiagonal element of a positive semidefinite matrix may not be greater than maximum of the corresponding diagonal entries.

$\mathcal{L}_n$ is closely related to the cut polytope P_n^{cut}. For a given partition vector $p \in \{-1, +1\}^n$ corresponding to a partition $S \subset V$ and $V \setminus S$ consider the upper triangle $\tau_p = \tau(pp^T)$ of the matrix $pp^T \in \tilde{\mathcal{L}}_n$. Then

$$\mathcal{X}^S = \frac{1}{2}(e - \tau_p)$$

where $\mathcal{X}^S$ is the characteristic cut vector of S. Indeed, by translation of the barycenter to the origin, scaling and reflection on the origin the cut polytope is shifted into the $[-1, 1]^{\binom{n}{2}}$ cube. The vertices of the cut polytope can be identified with the corresponding vertices of the $[-1, +1]^{\binom{n}{2}}$ cube, these specific vertices are contained in $\mathcal{L}_n$. It follows that the transformed cut polytope is contained in $\mathcal{L}_n$, which gives the geometric interpretation of the relaxation.

$\mathcal{L}_n$ wraps the cut–polytope quite tightly, a picture of $\mathcal{L}_3$ is given in Figure 7.1. To classify the boundary points of $\mathcal{L}_n$ Laurent and Poljak use the dimension of the normal cone. For a convex body K and a boundary point x_0 of K the normal cone is defined by

$$N(K, x_0) = \left\{c \in \Re^n | c^T x \leq c^T x_0 \quad \forall x \in K\right\}.$$

A boundary point of $\mathcal{L}_n$ is called a vertex if its normal cone is full dimensional.

Theorem 18 *([40])*
$\mathcal{L}_n$ has 2^{n-1} vertices, namely the vertices corresponding to the vertices of P_n^{cut}.

For nonnegative edgeweights several results are known regarding the quality of the bound. For instance,

Figure 7.1: $\mathcal{L}_3$

Theorem 19 *(Delorme,Poljak [18])*
Let G be a bipartite graph with nonnegative edgeweights, then $mc(G) = \varphi(G)$.

The worst case of the bound found so far is the 5–cycle C_5, for which $\varphi(C_5) = 1.131 \cdot mc(C_5)$. It is conjectured in [62] that this is indeed the worst case for all graphs with nonnegative edgeweights. By the following theorem the claim is true for planar graphs.

Theorem 20 *([18])*
For every planar graph G with nonnegative edgeweights, $\varphi(G) \leq 1.131 \cdot mc(G)$.

Recently Goemans and Williamson succeeded in proving a slightly larger bound for general graphs with nonnegative edgeweights.

Theorem 21 *(Goemans, Williamson [28])*
For an arbitrary graph G with nonnegative edgeweights, $\varphi(G) \leq 1.1383 \cdot mc(G)$.

Another interesting result is that the bound is asymptotically optimal for the class of random graphs $G_{n,p}$ with fixed edge probability p.

Theorem 22 *([18])*
For every fixed p, $0 < 1 < p$,

$$\lim_{n \to \infty} \frac{\varphi(G_{n,p})}{mc(G_{n,p})} = 1.$$

Again, up to a constant, $\varphi(G)$ is invariant to the switching operation. Several other interesting properties of $\varphi(G)$ are known, we refer the reader to [18, 40].

7.3 Computing $\varphi(G)$ Using the Interior Point Approach

The formulation of $\varphi(G)$ as a primal–dual pair of positive semidefinite programs (7.8) and (7.7) directly fits into the framework of Part I. There is no need for the vector variables s and t. Operator $\mathcal{A}$ reads

$$\mathcal{A}(\cdot) = \operatorname{diag}(\cdot) \qquad \text{and} \qquad \mathcal{A}^T(\cdot) = \operatorname{Diag}(\cdot).$$

To derive a short representation of the operator $\mathcal{O}$ of (2.25), which is in this case

$$\mathcal{O}(\cdot) = \operatorname{diag}(Z^{-1}\operatorname{Diag}(\cdot)X),$$

we represent diag$(\cdot)$ in the form

$$\operatorname{diag}(X) = \begin{pmatrix} \operatorname{tr}(E_{ii}X) \\ \vdots \\ \operatorname{tr}(E_{nn}X) \end{pmatrix},$$

where E_{ii} is an $n \times n$ matrix having a 1 in position ii and is zero otherwise. E_{ii} is the dyadic product $e_i e_i^T$ of the i-th canonical basis vector e_i. By (4.4) we have

$$\mathcal{O}_{ij} = \operatorname{tr}(e_i e_i^T Z^{-1} e_j e_j^T X) = \operatorname{tr}(e_i Z^{-1} e_j \; e_j^T X e_i) = (Z^{-1})_{ij} \cdot X_{ij}.$$

The matrix representation M of $\mathcal{O}$ is just the Hadamard product of Z^{-1} and X,

$$M = Z^{-1} \circ X,$$

and can be computed in $O(n^2)$ time. The most expensive operations within one iteration of the algorithm are therefore the inversion of Z and the factorizations of the positive definite matrices M, X, and Z with time complexity $O(n^3)$.

It is easy to find starting points in the interior of the primal and dual feasible sets. For X the center of $\mathcal{L}_n$ is the natural choice since it is independent of the weights of the graph. For the dual variables choose y sufficiently large such that $\operatorname{Diag}(y) - L$ is strictly diagonally dominant. Using MATLAB notation the initialization sequence could read

```
X=eye(n)/4;
y=sum(abs(L))+ones(n,1);
Z=diag(y)-L;
```

In the course of the algorithm X ideally converges to the dyadic product $\frac{1}{4}xx^T$ of the optimal partition $x \in \{-1,1\}^n$. Therefore any vector corresponding to the signs of a row of X (0 may be assigned arbitrarily) will form a reasonable feasible solution. In our algorithm we applied a local improvement strategy to these initial cuts by considering swapping any single node from one side to the other. We always swapped the node yielding the greatest improvement and continued swapping till no further gain was possible.

7.3.1 Remarks to the Numerical Experiments

We did not give any numerical results in Part I since our numerical experience is primarily based on the max–cut problem. The results presented here will therefore not only be used to illustrate the behavior of the bounds but also to give some evidence for the claims of Part I. In particular we will investigate the effect of the predictor–corrector approach and the difference between common and separate primal and dual steplength. However, it does not seem to be possible to present useful results regarding the reduction of the barrier parameter μ since any heuristic may perform very well on one class of problems and poorly on others. Therefore we will employ a very simple but stable heuristic which works sufficiently well on all test sets, with and without adding additional cuts. The strategy is based on the steplength α of the previous iteration (for the first iteration we set $\alpha := 1$). Let

$$\mu_c := \frac{\langle X, Z\rangle + \langle s, t\rangle}{n+m}$$

be the μ value associated with the current point (s, t, X, y, Z) as in (2.10). If the same steplength is used for primal and dual problem the new μ is determined by the following rule,

$$\mu := \begin{cases} \mu_c & \alpha \leq 0.2 \\ \mu_c/2 & 0.2 < \alpha \leq 0.9 \\ \mu_c/4 & 0.9 \leq \alpha. \end{cases} \tag{7.11}$$

Furthermore we impose the additional constraint that the sequence of the μ^i is non increasing, i.e. if the μ-value computed as in (7.11) is greater than its predecessor μ^{i-1} we set $\mu := \mu^{i-1}$. If different steplengths are used for primal and dual update we set $\alpha = \min\{\alpha_p, \alpha_d\}$ and use (7.11) again. This choice is quite independent of the algorithmical variants and should help to underline the differences in the quality of the step directions.

There are two stopping criteria for the algorithm. The first stops if the bound is accurate in the first 6 digits

$$\frac{\langle a, y\rangle}{\langle C, X\rangle} - 1 < 10^{-6}. \tag{7.12}$$

The second stops if the upper bound is close enough to the value v_{cut} of the best cut found so far,

$$\frac{\langle a, y\rangle}{v_{cut}} - 1 < 10^{-6}. \tag{7.13}$$

Note, that to prove optimality of a cut it suffices to show $\langle a, y\rangle < v_{cut} + 1$ which in many cases results in considerably shorter computation times. However, we are interested in the quality of the bound rather than short computation times.

To our knowledge there are no standard test problems for max–cut on complete graphs, therefore we used several different random graph models for our numerical experiments.

$G_{n;-10}$: weighted complete graphs on n vertices with edgeweights uniformly chosen from $\{-10, \ldots, 10\}$.

$G_{n;10}$: weighted complete graphs on n vertices with edgeweights uniformly chosen from $\{0,\ldots,10\}$.

$G_{n;.5}$: graphs on n vertices with edge–probability one half, or equivalently, complete graphs with edgeweights uniformly chosen from $\{0,1\}$.

$G_{n;tr}$: "almost" planar graphs on n vertices. We start with a triangle over the vertices $\{1,2,3\}$ and successively insert the next vertex (randomly) into one of the triangles already constructed, thereby generating three new triangles. This is obviously a planar graph. Then we do the same in reverse order starting with a triangle $\{n,n-1,n-2\}$. Edges picked once or twice get weight 1, all others 0. This yields a very sparse graph which is not planar with high probability.

Most of the numerical results will be presented in a table (or a subset of its columns) of the following format.

$G_{?;?}$								
n	nr	%	hh:mm:ss.dd	iter	proved	exact	ineq	viol
30	10	0.02	00:02:20.34	91.9	10	9	218	0.0001

Table 7.1: Example for column explanation

The columns have the following meaning,

$G_{?;?}$: gives the name of the random class investigated in this table.

n: dimension of the problem.

nr: number of problem instances investigated.

%: average relative gap between best cut found and upper bound in percent, i.e. let v_{cut} denote the value of the best cut found so far and let y be the vector of dual variables in the optimal solution, then for one instance

$$\% := \frac{< a, y > - v_{cut}}{v_{cut}} \cdot 100.$$

hh:mm:ss.dd: average computation time for one instance. The program was written in C++ and compiled with the -O option. All times were computed on a DEC-station 5000/240 by the callable routine `times`, only the user time was considered. hh gives the number of hours, mm the number of minutes, ss the seconds, and dd gives $secs/100$. Sometimes we will omit hh if it is zero for the whole table.

iter: average number of iterations.

proved: number of instances where the upper bound was sufficient to prove optimality of the cut.

exact: number of instances where (7.13) was satisfied, i.e. the gap between upper bound and best cut value was closed.

ineq: average number of facets in the program on termination.

viol: average maximum violation of the triangle inequalities on termination.

7.3.2 Different Step Directions

By numerical experiments we will investigate the effect of dropping the merit function, choosing the predictor–corrector approach, or using different step sizes for primal and dual update. For all versions the same test problems (of the class $G_{n;-10}$) are used. The results are contained in Table 7.2. We would like to mention that the program was not designed for fast computation of $\varphi(G)$ but as a test environment for adding additional inequalities to the elliptope. Without additional inequalities much faster convergence can be achieved by a more aggressive μ–reduction. However, we are interested in a uniform treatment of all different variants and did therefore not bother with improving the results for this special case. We will give some short comments to the different algorithms used for Table 7.2.

1. Using the merit function: This is the setting closest to the convergence analysis of Chapter 3. However, conditions (3.11) are weakened in the sense that the merit function must be non increasing for each step. If this constraint is violated, the steplength is reduced by a factor 0.8 and tested again.

2. Without merit function: We drop the merit function and use the positive definiteness search only. This reduces computation times a little since the computation of the merit function involves two factorizations at least. The common steplength is $\alpha = \min\{\alpha_p, \alpha_d\}$. The results show that the effect of the merit function is negligible.

3. Separate primal and dual step size: If the merit function is not of importance there is no need for a common steplength. α_p and α_d are chosen such that they are as close to 1 as possible. The primal variables are updated using steplength α_p and the dual variables are updated with steplength α_d. It was already observed in linear programming that this may lead to faster convergence which is also true in this case.

4. Predictor–corrector and separate step size: The predictor–corrector approach is implemented as discussed in Section 4.1 and is well known to result in fewer iterations. In the current case the number of rows k is equal to the dimension n of the matrix variables. Since the additional backsolve involves several $n \times n$ matrix multiplications the gain in iterations is lost in computation time. However, for $k \gg n$ the advantage is clearly on the side of the predictor–corrector approach, since the computation and factorization of the central system matrix dominates the cost. The results demonstrate the very stable behavior of the predictor–corrector approach. The number of iterations is clearly less dependent on n. For large n the predictor–corrector approach is increasingly interesting.

5. Same step size and predictor–corrector: In combination with the predictor–corrector approach there is not much difference between using separate steplengths and the same step size, contrary to the case without predictor–corrector. There seems to be a slight tendency towards separate steplengths.

6. Merit function and predictor–corrector: For completeness we also include this case. Differences are marginal, but computation time is a little bit higher. The merit function is probably not worth the trouble.

Looking at the results it is quite obvious that $\varphi(G)$ is best computed using separate steplengths for primal and dual update but no predictor–corrector scheme. For larger systems ($k \gg n$) the predictor–corrector approach is obligatory. Our main interest is devoted to the combination of $\varphi(G)$ with facet–defining inequalities which definitely leads to the latter case. To make the data comparable in iterations and computation times we will therefore use the predictor–corrector approach with separate step sizes in general, also for the computation of $\varphi(G)$.

7.3.3 Performance on Some Classes of Random Graphs

We will now give some results regarding the quality of the bound $\varphi(G)$ when applied to the random graph classes introduced on Page 63. Since the bound was neither tight nor sufficient to prove optimality for any of the computed examples we drop the corresponding columns. The results are listed in Table 7.3

$G_{n;-10}$: For this class the bound $\varphi(G)$ gives the worst results. Not only is the gap still large but also computation times are longer than for the other classes. The latter might be due to the fact that the weights are distributed over a wider range. Although

n	nr	%	mm:ss.dd	iter
30	30	10.86	00:01.19	17.2
40	30	13.75	00:02.79	18.3
50	30	15.03	00:05.43	19.2
60	30	16.03	00:09.50	20.2
70	10	15.23	00:15.87	21.3
80	10	15.66	00:23.95	21.8
90	10	17.76	00:34.33	22.0
100	5	18.51	00:46.46	21.6
150	5	19.72	03:12.80	23.6
200	5	20.61	08:59.23	24.6

1. merit function, no pred. corr.

n	nr	%	mm:ss.dd	iter
30	30	10.86	00:01.02	17.2
40	30	13.75	00:02.43	18.3
50	30	15.03	00:04.75	19.2
60	30	16.03	00:08.25	20.2
70	10	15.23	00:14.09	21.3
80	10	15.66	00:21.07	21.8
90	10	17.76	00:30.33	22.0
100	5	18.51	00:41.24	21.6
150	5	19.72	02:50.33	23.6
200	5	20.61	08:05.69	24.6

2. same step size, no pred. corr.

n	nr	%	mm:ss.dd	iter
30	30	10.86	00:00.88	14.9
40	30	13.75	00:02.07	15.6
50	30	15.03	00:03.95	16.1
60	30	16.03	00:07.08	16.6
70	10	15.23	00:11.77	17.3
80	10	15.66	00:17.44	17.6
90	10	17.76	00:25.36	17.8
100	5	18.51	00:34.45	17.2
150	5	19.72	02:21.60	18.6
200	5	20.61	07:02.12	19.4

3. separate step size, no pred. corr.

n	nr	%	mm:ss.dd	iter
30	30	10.86	00:01.44	12.1
40	30	13.75	00:03.28	12.2
50	30	15.03	00:06.67	12.8
60	30	16.03	00:11.24	12.7
70	10	15.23	00:18.12	12.7
80	10	15.66	00:27.09	12.7
90	10	17.76	00:39.38	12.9
100	5	18.51	00:57.20	12.8
150	5	19.72	03:16.09	12.0
200	5	20.61	09:36.35	13.0

4. separate step size, pred. corr.

n	nr	%	mm:ss.dd	iter
30	30	10.86	00:01.43	12.1
40	30	13.75	00:03.43	12.8
50	30	15.03	00:06.70	12.9
60	30	16.03	00:11.29	12.7
70	10	15.23	00:17.59	12.4
80	10	15.66	00:27.01	12.8
90	10	17.76	00:39.48	13.0
100	5	18.51	00:54.10	12.8
150	5	19.72	03:14.93	12.0
200	5	20.61	09:41.00	13.0

5. same step size, pred. corr.

n	nr	%	mm:ss.dd	iter
30	30	10.86	00:01.54	12.0
40	30	13.75	00:03.66	12.7
50	30	15.03	00:07.15	12.9
60	30	16.03	00:11.98	12.7
70	10	15.23	00:18.66	12.4
80	10	15.66	00:28.57	12.7
90	10	17.76	00:42.43	13.0
100	5	18.51	00:56.75	12.8
150	5	19.72	03:46.05	13.0
200	5	20.61	10:02.21	13.0

6. merit function, pred. corr.

Table 7.2:

the bound itself is not so promising the quality of the feasible solutions is very good. Only in a few cases they could be improved by adding additional inequalities.

$G_{n;10}$: The results show that for this class of graphs the bound is better than 2% in average, so there is a big difference between positively weighted graphs and arbitrary weights. Indeed, the bound gets better and better as n increases. This is not astonishing in view of Theorem 22 since there is clearly a similarity between this graph class and the class $G_{n;p}$.

$G_{n;.5}$: The computational results correspond to the theoretical result of Theorem 22.

$G_{n;tr}$: The behavior of the bound is quite different for this class of almost planar graphs. For n increasing the bound gets worse, although we know from Theorem 21 that for graphs with nonnegative edgeweights it cannot get worse than 14% (with respect to the optimal solution).

7.3.4 Summary

Summarizing the numerical results we can say that $G_{n;-10}$ is clearly the most difficult class for $\varphi(G)$. For $G_{n;10}$ and $G_{n;.5}$ the bound is very good and gets better and better as n increases. For the examined range the gap is still increasing for $G_{n;tr}$ but is well known to remain bounded by 14% with respect to the optimal solution. Observe that for all four classes the maximal violation with respect to the triangle facets has a slight tendency towards getting smaller for n increasing. However, the violation is still very strong and much can be gained by combining $\varphi(G)$ with facet–defining inequalities. Astonishingly, the feasible solution constructed by the very simple heuristic explained on Page 62 can rarely be improved by adding new inequalities. For many of the small examples we were able to prove optimality of the cut.

For graphs with huge node sets (large n) the approach presented here is too slow. In [59] it was shown that good approximations of $\varphi(G)$ can be computed by applying subgradient optimization methods to the eigenvalue formulation (7.3). They first used sophisticated heuristics to construct good starting solutions and improved these by the Bundle–Trust code of [65]. They were able to compute very tight bounds for graphs with up to $50,000$ vertices. However, the subgradient approach runs into serious difficulties as soon as the maximal eigenvalue is not well separated any more. The strength of interior–point positive semidefinite programming is also based on the fact that eigenvalue clustering does not cause any difficulties.

$G_{n;-10}$					
n	nr	%	mm:ss.dd	iter	viol
30	30	10.86	00:01.44	12.1	0.2477
40	30	13.18	00:03.25	12.1	0.2471
50	30	15.03	00:06.56	12.7	0.2474
60	30	16.93	00:11.30	12.7	0.2477
70	10	15.23	00:18.08	12.7	0.2477
80	10	17.67	00:27.24	12.8	0.2476
90	10	17.58	00:39.89	13.1	0.2468
100	5	18.51	00:54.31	12.8	0.2455
150	5	18.51	03:15.76	12.0	0.2459
200	5	21.00	09:37.12	13.0	0.2458
250	5	20.67	21:06.64	13.4	0.2446
300	1	23.44	39:47.38	13.0	0.2351

$G_{n;10}$					
n	nr	%	mm:ss.dd	iter	viol
30	30	1.77	00:01.30	11.0	0.2480
40	30	1.62	00:02.98	11.1	0.2479
50	30	1.58	00:05.97	11.6	0.2475
60	30	1.57	00:10.64	11.9	0.2472
70	10	1.69	00:16.89	12.0	0.2468
80	10	1.52	00:25.36	12.0	0.2476
90	10	1.50	00:33.62	11.0	0.2466
100	5	1.58	00:46.09	11.0	0.2452
150	5	1.39	03:09.43	11.8	0.2447
200	5	1.27	08:46.19	12.0	0.2446
250	5	1.25	18:58.86	12.2	0.2447
300	1	1.15	38:05.10	13.0	0.2451

$G_{n;.5}$					
n	nr	%	mm:ss.dd	iter	viol
30	30	2.68	00:01.36	11.6	0.2476
40	30	2.70	00:02.94	11.1	0.2473
50	30	2.44	00:05.82	11.4	0.2474
60	30	2.45	00:10.35	11.7	0.2477
70	10	2.34	00:16.05	11.5	0.2481
80	10	2.45	00:24.75	11.8	0.2465
90	10	2.38	00:35.93	12.0	0.2467
100	5	2.18	00:49.75	12.0	0.2463
150	5	2.10	03:09.03	12.0	0.2465
200	5	1.89	08:46.35	12.2	0.2425
250	5	1.84	18:31.18	12.0	0.2430
300	1	1.85	35:22.92	12.0	0.2389

$G_{n;tr}$					
n	nr	%	mm:ss.dd	iter	viol
30	30	2.85	00:01.28	11.0	0.2471
40	30	2.91	00:02.94	11.1	0.2468
50	30	3.09	00:05.75	11.3	0.2468
60	30	3.30	00:10.19	11.7	0.2467
70	10	3.37	00:16.23	11.7	0.2464
80	10	3.44	00:25.08	12.0	0.2471
90	10	3.39	00:35.49	12.0	0.2465
100	5	3.55	00:48.97	12.0	0.2468
150	5	3.90	03:08.17	12.2	0.2449
200	5	3.86	08:38.73	12.4	0.2438
250	5	4.20	18:09.80	12.2	0.2419
300	1	4.67	33:32.77	12.0	0.2409

Table 7.3: $\varphi(G)$ for different classes of random graphs

Chapter 8

Combining $\varphi(G)$ With Facets

In Chapter 6 it was observed that the upper bound $\sigma(G)$ defined over the metric polytope P_n^{met} has several desirable properties but is difficult to compute on its own. The interior point method of Part I is very well suited for adding additional cuts such as the triangle inequalities to the positive semidefinite program (7.8) and it is only natural to improve the bound $\varphi(G)$ by optimizing over the "intersection" of $\mathcal{L}_n$ and P_n^{met}. Of course it is possible to add all kinds of different facets and inequalities but since this is increasingly time consuming we have restricted our investigations to clique–inequalities.

In Section 8.1 we will discuss an appropriate form for using facet–defining inequalities such as the hypermetric inequalities, which will lead to a proof for the validity of the very general class of gap inequalities. In Section 8.2 we will investigate implementational aspects of the process of adding new inequalities. The next Section will present numerical results on the combination of $\varphi(G)$ and $\sigma(G)$. In Section 8.4 larger cliques will be used as well. Finally, in Section 8.5, we will report about some less successful experiments with crossing over to the simplex method.

8.1 Gap Inequalities

In [41] Laurent and Poljak observed that there is an even larger class of valid inequalities for the cut polytope containing the hypermetric inequalities of (6.4). These so called gap inequalities are formed by n integers $b_1, \ldots, b_n$ such that for any partition $(S, V \setminus S)$ the two sums $\sum_{i \in S} b_i$ and $\sum_{i \in V \setminus S} b_i$ differ by at least γ and γ is the maximum integer with this property. γ is called the gap and the inequality takes the following form for vectors $y_{ij} \in [0,1]^{\binom{n}{2}}$,

$$\sum_{1 \le i < j \le n} b_i b_j y_{ij} \le \frac{1}{4}\Big(\Big(\sum_{i=1}^{n} b_i\Big)^2 - \gamma^2\Big). \tag{8.1}$$

They also proved the following

Lemma 23
Every gap inequality is valid for P_n^{cut}.

So far it is not known whether facet–defining inequalities with $\gamma > 1$ exist.

To get a better understanding of this type of inequalities we will take another look at partition vectors $\{-1,+1\}^n$ and at the primal model. In the description of the primaly feasible matrices

$$\{X \mid \operatorname{diag}(X) = e, X \succeq 0\}$$

the semidefiniteness constraint could be replaced by infinitely many linear constraints

$$b^t X b \geq 0 \qquad \forall b \in \Re^n.$$

To get a form similar to (8.1) we apply the transformation $X = J - 2Y$ which yields a matrix Y having $\tau(Y) \in [0,1]^{\binom{n}{2}}$ and $\operatorname{diag}(Y) = 0$ for a feasible X. We get

$$b^T e e^T b - 2b^t Y b = \left(\sum_{i=1}^{n} b_i\right)^2 - 4 \sum_{1 \leq i < j \leq n} b_i b_j y_{ij} \geq 0.$$

Obviously the feasible set of $\varphi(G)$ satisfies all inequalities without gap. However, we really want to restrict the feasible set to those matrices having the same gap as the matrices $X = pp^T$ with $p \in \{-1,+1\}^n$ a partition vector. If for a certain vector $\hat{b} \in \Re^n$

$$\min_{p \in \{-1,+1\}^n} |\hat{b}^T p| = \gamma$$

then we can restrict the feasible set further by the additional inequality

$$\hat{b}^t X \hat{b} \geq \min_{p \in \{-1,+1\}^n} \hat{b}^T p p^T \hat{b} = \gamma^2.$$

With the observation above this is also a proof of Lemma 23.

If all the b_i are integers and sum up to some odd number then clearly γ must be at least one. Thus the gap inequalities contain the hypermetric inequalities. For the clique–inequalities the b_i are restricted to $\{-1,0,+1\}$, therefore there must be an odd number of nonzero coefficients.

A clique inequality defined on X is introduced into our model by adding one line to $\mathcal{A}(\cdot)$ (the coefficients of the inequality) and introducing a new slack variable s_i, which forms an additional component of s. $\mathcal{B}(\cdot)$ is zero on the block corresponding to the equality constraints and the negative identity matrix $-I$ on the block of the inequality constraints. For each clique inequality the right hand side value is $1/4$. In the dual program we get one new $y-$ and one new t–component for each inequality. For the convenience of the reader we give the formulation of the primal program with h clique inequalities described by vectors $b^i \in \{-1,0,1\}^n$.

$$(PP) \qquad \begin{array}{rll} \text{Maximize} & \langle L, X\rangle & \\ \text{subject to} & \frac{1}{4}e - \text{diag}(X) = 0 & \\ & \frac{1}{4} - b^{iT}Xb^i - s_i = 0 & i = 1,\ldots,h \\ & X \succeq 0, s \geq 0, & \end{array}$$

In semidefinite programming it is computationally more efficient to work directly with the vectors b instead of the matrix formulation of (8.1). This helps to accelerate the construction of the matrix representing the $\mathcal{O}$ operator which in general is the most time consuming step within one iteration. The semidefinite programming formulation of an arbitrary inequality is given by $\text{tr}A_iX \geq \gamma$. In this specific case A_i has the convenient representatin bb^t which can be used for computing (4.4) efficiently,

$$\text{tr}(bb^tZ^{-1}A_jX) = \text{tr}((b^tZ^{-1})A_j(Xb)).$$

Independently of the structure of A_j one element can now be computed in $O(n^2)$ instead of $O(n^3)$. Moreover, the vectors b^tZ^{-1} and Xb have to be computed only once for several matrices A_j. If all other A_j are also of form vv^t the overall computation time for one line reduces to $O(n^2+kn)$, where k is the number of constraints and in our case $n \leq k \leq \binom{n+1}{2}$. The whole matrix can now be computed in $O(kn^2+k^2n)$ and the dominating cost of one iteration is the factorization of the matrix with $O(k^3)$ arithmetic operations, the best we can hope for.

8.2 The Algorithmical Framework

For interior point methods the introduction of violated inequalities forms a serious problem. This is immediately clear since the current solution is not in the interior of the new problem. Indeed, it may be difficult to find a new feasible interior point and it is almost impossible to make use of the current solution. Some approaches try to continue from the current solution by relaxing the new inequalities at first so that they are satisfied by the current solution. By giving a high weight to the barrier parameter corresponding to the relaxed inequalities the iterates are then driven away form the boundary, the relaxation can be tightened again. However, it was observed that continuing from a point close to the boundary is as expensive as restarting from the beginning. So far no efficient concept for the general case is known.

In our case the setting is definitely easier. The barycenter of the cut polytope is $(\frac{1}{2},\ldots,\frac{1}{2})$ and the polytope is full dimensional (see Chapter 6). In semidefinite terms this means that $\hat{X}=\frac{1}{4}I$ is always an interior point which should also lie quite symmetrically to the considered inequalities. This suggests that a good new starting point can be found on the straight line segment $X-\tau(X-\hat{X}),\ \tau\in[0,1]$. A violated clique inequality given by the vector b is satisfied if

$$b^T\left(X-\tau(X-\hat{X})\right)b>\frac{1}{4}.$$

Since the inequality is violated we have $b^T(X-\hat{X})b<0$ and therefore τ is bounded from below by

$$\tau>\frac{1-4b^TXb}{b^Tb-4b^TXb}.$$

For several violated inequalities the minimum value of τ yielding a feasible X is given by the maximum of all these expressions. To keep away from the boundary this τ is enlarged by a factor 1.01. The computation of the new slack variable s is straight forward. The quality of a new starting point also depends on the dual variables. The dual solution remains feasible if an additional primal constraint is introduced. In principle the y_i and t_i corresponding to the new inequality would have to be set to zero. But as the algorithm requires $t>0$ we simply accept infeasibility on the dual side and set $t_i=1$ and $y_i=0$. Of course infeasibility could be avoided by increasing the diagonal elements of Z by means of the corresponding dual variables y_j. Yet in practical experiments the first approach seemed more efficient since it kept closer to the current solution. $\langle a,y\rangle$ is an upper bound for max–cut if all y_i corresponding to inequalities are nonpositive which is easy to check. In practice full dual feasibility ($t_i=-y_i$) was achieved within three iterations. The new primal–dual pair is of good quality only if the new primal variables are quite similar to the old ones so that $ZX\approx\mu I$ remains valid. Indeed, this approach is of little use for restarting from an optimal solution (Z is almost singular, the new X is pushed back into the interior). But the approach works very well if after the first violation of an inequality at most two or three further steps are taken.

We will now discuss the process of adding violated triangle facets. In total there are $4\binom{n}{3}$ different inequalities. It makes sense to look for violated triangles by exhaustive search since this is of the same order of magnitude as the factorization of X, and clearly cheaper than the factorization of the big system matrix. In the optimal solution of $\varphi(G)$ literally thousands of them are violated. For computational efficiency of the algorithm it is extremely important to select a very small but strong subset of these inequalities.

The approach generally taken is to select a certain number of the most violated inequalities and iterate. However, to compute the new optimal solution and restart again is too expensive. Therefore we test all triangle inequalities for violation in each iteration. If a violation is detected we do not stop at once but perform another two or three iterations. If the primal–dual pair is allowed to proceed a little bit further into the desired direction a larger set of violated triangles is available and the violation criterion seems to be more representative. Now we add a small set of inequalities and compute a new starting point. In this case the construction of the new primal–dual pair, as discussed above, works very well in practice. Heuristically speaking this is due to the fact that the new primal and the old dual point are only about three iterations apart. The three additional iterations serve yet another purpose. Most of the time the next iteration based on the new primal–dual pair leads to the next violation. The three iterations enable the algorithm to compute a new useful primal dual pair before the next problem change occurs.

Adding constraints as soon as they are violated helps avoiding wrong directions quickly. Almost every three steps a new set of violated inequalities will be added. In contrast to the traditional simplex approach two succesive sets are very likely to be quite different because the barrier functions keep the iterates at some distance from the freshly added inequalities. In this manner a wide spectrum of constraints is generated within short time. Three iterations are comparatively cheap and so we may add only a few of the most violated inequalities each time thereby increasing the probability that we include the important ones only.

Of course not all inequalities selected will be tight in the optimal solution. To keep the constraint matrix small such constraints are removed as soon as possible. Whenever new inequalities are added the old ones are reconsidered. In the optimal solution "redundant" inequalities have positive slack and dual cost going to zero. In linear programming (see [24]) recently several indicators for zero variables have been studied. In our case, however, the problem description seems to change too fast for these criteria. Therefore we simply eliminate all constraints having dual cost smaller than 0.005 (sometimes we use 0.0005) times the maximal dual cost. The bad news is that this bound has some effect only if the solution is already close to optimal. In order to eliminate redundant inequalities we have to solve the problem to the end.

Solving the current relaxation exactly in regular intervals serves other purposes as well. On the one hand the bound may already be good enough to prove optimality, on the other the new optimal solution may yield a better feasible cut. After quite some practical experience we ended up with the following scheme. We first compute the optimal solution of $\varphi(G)$ and generate a feasible cut. If there are no violated triangle inequalities, stop. Otherwise we add the n most violated triangle facets and restart (a first "large add"). Then, for ten times we add the $n/3$ most violated inequalities as soon as encountered but with the delay of three iterations (we will call these "small adds"). After ten small adds we compute the optimal solution of the current setting. If there are still violated inequalities (the violation being at least 10^{-6}) add the n most violated inequalities, remove "redundant" constraints and proceed as before (a "large add"). For testing optimality we use (7.12), the algorithm stops whenever (7.13) is satisfied or no violated triangle inequalities are found or a prescribed number of "large adds" has been performed.

We also tested another approach for restarting with triangle inequalities. Since we check each iterate for violated triangle facets we simply stored the last iterate not violating any triangle inequalities. Whenever new facets were added we restarted from the last feasible iterate. The approach was very successful since it was never necessary to restart from the beginning. Sometimes, however, the iterates got to far into the wrong direction and had troubles getting back to the center. This method cannot be applied to general inequalities since checking feasibility in each iteration would be too expensive if not impossible.

For Simplex type algorithms the coming and going of the same constraints established a major problem. Furthermore the inequalities seemed to cluster on only part of the vertices. We did not observe a similar behavior of our algorithm. Probably because of the "small adds"–effect already explained before — the iterates are pushed away from newly added constraints — the constraints were always distributed nicely over all vertices. Only very few inequalities were eliminated and reintroduced, far too few to spend some thought on. However, this may also be due to our choice of test problems and the topic might be of interest if some special graphs with strong structural properties are tested.

8.3 The Upper Bound $\varphi(G) \cap \sigma(G)$

The feasible set obtained by adding all triangle inequalities can be thought of as the intersection of $\mathcal{L}_n$ and P_n^{met}. The upper bound induced by this set can be computed in polynomial time and clearly inherits all properties of $\varphi(G)$ and $\sigma(G)$, the most important ones being exactness for graphs not contractible to K_5 and asymptotical tightness for the class $G_{n;p}$ of random graphs. Our numerical results indicate that this bound is very powerful indeed and drastically improves on $\varphi(G)$ even if only a few triangle constraints are added. However, the bound may fail on very small graphs such as K_5. We denote the new bound by

$$\begin{aligned} \psi(G) := \qquad & \text{Maximize} && \langle L(G), X\rangle \\ & \text{subject to} && \tfrac{1}{4}e - \operatorname{diag}(X) = 0 \\ & && \tfrac{1}{4} - b^T X b \le 0 \qquad \forall\, b \in \{-1,0,1\}^n,\ b^T b = 3 \\ & && X \succeq 0. \end{aligned} \tag{8.2}$$

For $G = K_5$ we have

$$\frac{\psi(K_5)}{mc(K_5)} = \frac{6.25}{6} \approx 1.0417.$$

In Table 8.1 we give some numerical results for $\psi(G)$. The improvements in comparison to Table 7.3 are indeed enormous, but so are computation times. Nevertheless it is obvioulsy a lot easier to compute $\psi(G)$ than $\sigma(G)$. For $\sigma(G)$ cutting plane approaches using the Simplex method got to $n = 30$. The fact that in some cases the average violation is not less than 10^{-6} is due to stopping criterion (7.13). If the gap between upper bound and cut is closed sufficiently the algorithm is stopped, may the current iterate violate triangle inequalities or not. If the algorithm had been stopped by criterion (7.12) (i.e. the gap to the value of the cut could not be closed completely) the number of iterations needed was always around 250, quite independently on the dimension of the problem. The main cause for the drop in speed is the factorization of the big system matrix of size $k \times k$, where k is n plus the number of inequalities. Note that the number of triangle inequalities is far less than $\binom{n}{2}$, so the elliptope has some strong positive influence on the bound. Indeed the ratio of the number of inequalities in the optimal solution to $\binom{n}{2}$ seems to get smaller for n increasing. This is already suggested by the results of Table 7.3 where the maximal violation of the triangle inequalities was found to be decreasing with n increasing.

For branch & bound algorithms it is definitely to slow to compute $\psi(G)$ accurately. However it may be worth to add just a few triangles. In practice the bound is improved very fast in the beginning and then it takes a long, long time to get the maximal violation down. We give a typical example for a graph of the class $G_{100;-10}$ in Table 8.2. *iter* denotes the number of iterations, *bound* gives the dual objective value. As usual, % is the relative gap, for this specific example it is computed with respect to 2440 (this is the best feasible solution found, again it was constructed after the first 13 iterations and was not improved later on). *ineq* gives the current number of inequalities included in the program, and *viol* lists the maximal violation of the triangle facets. The first line of Table 8.2 gives the bound $\varphi(G)$, each further line corresponds to one "large add". In particular at most $100 + 10 \times \lfloor \frac{100}{3} \rfloor$ triangle inequalities are added from one line to the next. As a matter of fact, the zero violation of the last line states that there are *no* triangle inequalities violated with respect to the corresponding solution. If only one "large add" (including the successive ten "small adds") is carried out less than $5n$ triangle inequalities are added. At this point the bound is already very good and only a fraction of the computing time is needed (see Table 8.3).

n	nr	%	hh:mm:ss.dd	iter	proved	exact	ineq	viol
$G_{n;-10}$								
30	10	0.02	00:02:20.34	91.9	10	9	218	0.0001
40	5	0.11	00:07:43.17	93.4	4	4	267	0.0003
50	3	0.31	01:04:01.68	164.7	2	2	501	0.0002
60	1	2.11	03:55:58.08	221.0	0	0	689	0.0000
$G_{n;10}$								
30	10	0.04	00:05:12.42	110.2	8	7	209	0.0009
40	5	0.02	00:26:48.31	150.6	4	4	402	0.0011
50	3	0.11	01:02:31.67	173.7	1	1	516	0.0003
60	1	0.34	02:53:53.82	201.0	0	0	600	0.0000
70	1	0.31	06:06:52.22	214.0	0	0	754	0.0000
$G_{n;.5}$								
30	10	0.02	00:07:08.68	137.5	10	5	252	0.0005
40	5	0.09	00:50:27.04	214.8	5	1	428	0.0001
50	3	0.21	01:29:15.87	202.3	1	1	512	0.0001
60	1	0.57	04:50:26.62	300.0	0	0	601	0.0000
70	1	0.41	09:06:55.68	262.0	0	0	816	0.0000
$G_{n;tr}$								
30	10	0.23	00:08:12.05	143.8	10	4	237	0.0005
40	5	0.41	00:35:47.99	201.8	3	0	393	0.0000
50	3	0.13	00:32:42.26	133.0	3	2	417	0.0018
60	1	0.86	02:33:57.12	217.0	0	0	603	0.0000
70	1	0.64	05:57:02.72	227.0	0	0	750	0.0000

Table 8.1: $\psi(G)$ for different classes of random graphs. For $G_{n;tr}$ the factor for removing old triangles had to be set to $5 \cdot 10^{-4}$.

iter	bound	%	hh:mm:ss	ineq	viol
13	2797.54	14.65	00:00:58	0	0.2449
57	2580.17	5.745	00:21:09	430	0.1563
102	2550.13	4.513	02:07:10	820	0.08224
147	2541.24	4.149	06:58:05	1137	0.03165
188	2540.26	4.109	14:31:11	1287	0.002284
227	2540.25	4.109	21:57:14	1179	0

Table 8.2: Development of upper bound for a graph $G_{100;-10}$.

8.4 Adding Larger Cliques

For curiosity we also tried to add larger clique inequalities. Since we did not find any good separation procdure for discovering large violated cliques independently of smaller ones and exhaustive search is clearly to expensive, even for cliques of size 5, we applied the following scheme inspired by an heuristic proposed in [12]. Whenever new triangle inequalities were to be added we also tried to enlarge active clique inequalities by performing an exhaustive search for the two vertices (and corresponding signs) yielding the maximal violation. The new clique inequality was added to the list of candidates and was itself used again to construct one more larger clique.

To give an example, an active clique inequality of size 3 on vertices $(4, 11, 31)$ with sign structure $(-1, 1, 1)$ is enlarged to a clique of size 5 on vertices $(4, 11, 31, 13, 20)$ with sign structure $(-1, 1, 1, -1, 1)$. So vertices $(13, 20)$ combined with signs $(-1, 1)$ were found to yield the smallest product $b^T X b$. If this 5–clique constraint is violated it is added to the candidates. Independent of the violation the 5–clique is used to construct a 7–clique, say by adding vertices 8 and 12 with signs 1 and 1, which again is a canditate if it is violated. We do not try to enlarge the 7–clique any more.

In the current form all larger cliques are based on violated triangle inequalities. This means that K_5 can not be solved, although in principle the algorithm can handle 5–cliques, because the solution of $\varphi(K_5)$ does not violate any triangle inequality. For the graphs of our test classes this seems to be of no importance. Lots of large violated cliques were found up to sizes of 19. The dual cost pattern and the number of cliques remaining in the program very much correspond to the expected. Triangle inequalities are the most numerous and have the highest dual cost average, both values are constantly decreasing as the clique size increases.

The practical consequence of letting the clique inequalities grow from smaller cliques

n	nr	%	hh:mm:ss.dd	iter	proved	exact	ineq	viol
				$G_{n;-10}$				
30	30	0.31	00:00:22.99	52.0	23	15	129	0.0445
40	20	1.53	00:01:03.28	55.3	1	1	170	0.1235
50	20	2.70	00:01:57.87	55.4	3	1	210	0.1232
60	10	4.03	00:03:43.82	55.8	0	0	260	0.1584
70	10	5.43	00:05:59.29	54.7	0	0	300	0.1597
80	5	7.75	00:09:23.90	55.4	0	0	340	0.1512
90	5	6.30	00:14:42.19	57.2	0	0	390	0.1513
				$G_{n;10}$				
30	30	0.09	00:00:23.70	52.2	18	3	129	0.0808
40	20	0.19	00:01:00.22	53.2	5	2	170	0.1136
50	20	0.37	00:01:53.81	53.8	1	0	210	0.1377
60	10	0.50	00:03:32.09	53.9	0	0	260	0.1541
70	10	0.47	00:05:50.78	54.6	0	0	300	0.1548
80	5	0.53	00:08:52.84	54.2	0	0	340	0.1478
90	5	0.70	00:13:39.45	53.8	0	0	390	0.1629
				$G_{n;.5}$				
30	30	0.22	00:00:23.23	52.7	29	7	130	0.0779
40	20	0.38	00:00:59.94	53.2	12	0	170	0.1244
50	20	0.51	00:01:52.54	53.3	5	0	210	0.1325
60	10	0.65	00:03:31.86	54.0	0	0	260	0.1440
70	10	0.86	00:05:48.14	53.8	0	0	300	0.1496
80	5	0.97	00:09:10.33	55.6	0	0	340	0.1521
90	5	0.94	00:13:23.32	53.8	0	0	390	0.1518
				$G_{n;tr}$				
30	30	0.45	00:00:23.18	51.8	25	5	129	0.0894
40	20	0.70	00:01:00.96	53.7	13	0	169	0.1471
50	20	0.72	00:01:58.68	55.4	8	0	210	0.1552
60	10	1.00	00:03:50.21	57.0	0	0	257	0.1605
70	10	1.27	00:06:02.58	55.5	0	0	300	0.1681
80	5	1.61	00:09:11.84	54.8	0	0	340	0.1675
90	5	1.72	00:14:12.36	55.2	0	0	390	0.1688

Table 8.3: $\varphi(G)$ improved by one phase of adding triangles (less than $5n$ triangle facets)

is that the effect of larger cliques is delayed. But since smaller cliques are more effective anyways this is probably an advantage.

Again we will first look at some examples where there is no limit on the number of iterations (Table 8.4). The experiments are performed on the same problems used for Table 8.1. So far all problems except one were solved exactly. The single exception of the class $G_{n;.5}$ of size 60 is due to the fact that memory limits allowed for at most 1300 clique inequalities. If this upper bound is attained the algorithm continues to replace at most $n/3$ inequalities with least dual cost by violated inequalities. In the case of the example $G_{n;10}$ of size 60 this indeed helped to close the gap, but the cost in iterations and computation times is high. For $G_{n;.5}$ of size 60 the number of inequalities was obviously too small to close the gap, the algorithm stopped after completing the first large add above 1000 iterations. In general the problems may differ quite a lot with respect to the number of inequalities needed which can be seen in $G(n;tr)$. The problems of size 40 were hard to solve in contrast to the three examples of size 50. Again the number of inequalities is distinctively smaller than $\binom{n}{2}$.

By admitting only one large add and working on the same set of problems as used for Table (8.3) we get Table (8.5). Now we succeed in proving the optimality for several problems which could not (yet) be proved by triangles alone. Observe that triangle violation is considerably higher than in Table (8.3). So it is indeed the larger cliques which make the difference. There is also a small increase in computation time which is mainly due to the separation of the larger cliques. The additional expenses for building the system matrix and doing the line search with larger cliques are negligible.

8.5 Experience With Cross–Over

It seems to be generally believed that the Simplex method is better suited for adding inequalities. This is definitely not the case in this specific context. In the beginning of our experiments we also tried to implement a crossover from the interior point solution of $\varphi(G)$ to the simplex method (CPLEX). The success was moderate.

Since interior point methods in linear programming do not yield active constraints the same convenient way the simplex method does — due to the central path property all optimal vertices may even be far away from the optimal solution of the interior point code — the construction of a basis for the simplex method is a difficult problem (see e.g. [48]). For our semidefinite program the situation is indeed hopeless. Because the elliptope is not a polyhedron it is evident that in general there is no basis. One can however try to construct a system of constraints which keeps the solution close to the interior point

n	nr	%	hh:mm:ss.dd	iter	proved	exact	ineq	viol
				$G_{n;-10}$				
30	10	0.00	00:01:00.87	63.8	10	10	170	0.0003
40	5	0.00	00:05:33.42	80.0	5	5	278	0.0003
50	3	0.00	00:57:57.76	135.7	3	3	520	0.0005
60	1	0.00	15:50:47.60	318.0	1	1	1207	0.0004
				$G_{n;10}$				
30	10	0.00	00:02:16.47	73.0	10	10	196	0.0011
40	5	0.00	00:14:28.34	113.0	5	5	390	0.0017
50	3	0.00	03:40:53.50	208.3	3	3	681	0.0016
60	1	0.00	74:15:41.72	670.0	1	1	1300	0.0030
				$G_{n;.5}$				
30	10	0.00	00:02:45.25	88.4	10	10	228	0.0009
40	5	0.00	00:29:25.72	151.0	5	5	449	0.0015
50	3	0.00	05:05:05.44	249.7	3	3	744	0.0017
60	1	0.08	143:38:07.08	1041.0	1	0	1300	0.0274
				$G_{n;tr}$				
30	10	0.00	00:14:25.88	146.5	10	10	292	0.0006
40	5	0.00	04:13:10.51	307.0	5	5	563	0.0128
50	3	0.00	00:53:48.07	138.0	3	3	485	0.0011
60	1	0.00	16:23:25.00	377.0	1	1	1030	0.0345

Table 8.4: $\varphi(G)$ improved by adding cliques

n	nr	%	hh:mm:ss.dd	iter	proved	exact	ineq	viol
$G_{n;-10}$								
30	30	0.20	00:00:25.95	49.6	26	21	128	0.0336
40	20	1.31	00:01:09.25	54.8	3	1	170	0.1320
50	20	2.65	00:02:13.73	54.6	3	3	210	0.1441
60	10	3.93	00:04:12.24	55.3	0	0	260	0.1615
70	10	5.40	00:06:47.44	54.6	0	0	300	0.1721
80	5	7.68	00:10:37.87	55.6	0	0	340	0.1592
90	5	6.30	00:16:14.50	56.6	0	0	390	0.1679
$G_{n;10}$								
30	30	0.08	00:00:26.96	50.0	20	12	128	0.0776
40	20	0.17	00:01:04.20	52.0	6	3	169	0.1225
50	20	0.37	00:02:11.13	53.1	1	1	210	0.1634
60	10	0.50	00:04:10.38	54.2	0	0	260	0.1657
70	10	0.47	00:06:40.00	54.3	0	0	300	0.1736
80	5	0.52	00:09:59.92	54.0	0	0	340	0.1656
90	5	0.70	00:15:33.96	54.2	0	0	390	0.1703
$G_{n;.5}$								
30	30	0.16	00:00:27.41	51.5	29	9	128	0.0887
40	20	0.36	00:01:05.89	52.7	12	2	170	0.1465
50	20	0.49	00:02:11.14	53.0	5	1	210	0.1505
60	10	0.65	00:04:08.96	54.6	0	0	260	0.1690
70	10	0.87	00:06:43.06	53.8	0	0	300	0.1680
80	5	1.00	00:10:08.23	54.0	0	0	340	0.1699
90	5	0.97	00:15:20.97	54.0	0	0	390	0.1697
$G_{n;tr}$								
30	30	0.37	00:00:27.62	51.3	26	7	128	0.1075
40	20	0.64	00:01:07.07	53.2	14	0	170	0.1647
50	20	0.64	00:02:17.08	55.1	10	0	209	0.1687
60	10	0.95	00:04:18.71	56.9	0	0	259	0.1709
70	10	1.21	00:07:15.80	57.7	0	0	300	0.1749
80	5	1.52	00:10:52.87	57.0	0	0	340	0.1759
90	5	1.64	00:15:45.74	54.8	0	0	390	0.1781

Table 8.5: $\varphi(G)$ improved by one phase of adding cliques (less than $5n$ clique facets)

solution. From this initial set one can continue by adding triangle inequalities and vector constraints. Vector constraints are formed by some linear combinations of eigenvectors to negative eigenvalues of the matrix X corresponding to the current simplex solution.

Obviously the initial constraints must have some connection to the null space of the interior point solution X. All eigenvectors to the eigenvalue zero form tight constraints for the elliptope at X (see Section 8.1). The null space may and will in general be quite large, thus there are infinitely many possibilities for choosing such vectors. Each vector constraint forms a dense row for the simplex constraint matrix. The simplex model requires $\binom{n}{2}$ variables. Consequently the number of constraints of this type should be as small as possible to keep computation times within bounds.

We experimented with several different approaches for constructing good starting sets of inequalities but we never came close to the quality of the solution of $\varphi(G)$ provided by the interior point method. Only one strategy for constructing initial vector constraints seemed to be of some use. The idea is to choose a first eigenvector of the null space of X such that the projection on the cost function is maximal. The next zero eigenvector is selected by the same criterion but from the subspace orthogonal to the first and so on. In the algorithm we use the subspace spanned by the normalized eigenvectors v_i of the eigenvalues $\lambda_i(X) < 10^{-4}$. We collect these say k eigenvectors in a matrix $V = (v_1, \ldots, v_k)$. Then the first constraint vector c^1 is a linear combination of these eigenvectors,

$$c^1 := V \cdot a^1.$$

We use a normalized coefficient vector a^1. Clearly a^1 is the solution of

$$\max\left\{a^T V^T L(G) V a | a \in \Re^k, ||a|| = 1\right\}.$$

It is now obvious that the coefficients a^i are the eigenvectors of the matrix $V^T L(G) V$. Additionally to these k vector constraints we started with a certain number of triangle constraints violated in the interior point solution. Nonetheless it took several iterations of adding further triangle and vector constraints to approach the initial bound.

Chapter 9

Summary and Open Problems

In Part I we presented an interior point algorithm for semidefinite programming. The algorithm is very easy to implement, in particular it only requires the factorization of positive definite matrices. This algorithm was implemented in C++ and is, to our knowledge, the fastest semidefinite programming code currently available. In Part II we developed an algorithm for computing upper bounds for the max–cut problem using this semidefinite programming environment. The algorithm combines an eigenvalue upper bound with polyhedral cutting planes. For the max–cut problem on complete graphs the new approach is clearly superior to traditional cutting plane approaches employing the simplex algorithm. Recent computational results for other combinatorial problems indicate that semidefinite programming will lead to considerable improvements for several other applications as well.

However, positive semidefinite programming will in general be computationally too expensive for large sparse problems. Currently it is not known whether the sparsity of the cost matrix can be exploited at all. Therefore acceptable computation times (about one hour of CPU–time) can be expected only for $n \leq 500$ and $k \leq 2000$.

Finding an initial feasible interior point can in principle be achieved by a phase one approach or a big M method. The phase one approach requires a lot of extra time, the big M method is often accompanied by numerical difficulties. For cutting plane algorithms it is desirable to continue from the current solution without having to alter the problem description too much and without having to look for a new interior point. These requirements seem to condtradict both possibilities. We have already mentioned the approach to relax the violated inequalities at first. Can this be done efficiently for some practical applications?

What can be done if a problem is known to have feasible solutions but no interior points, i.e. if the feasible set is a subset of a face of the semidefinite cone? For some problems it is possible to project the feasible set into a lower dimensional space with interior points. For instance we can model the equipartition problem for n even by adding $\mathrm{tr}(JX) = e^T Xe = 0$ to the max–cut problem. e is an eigenvector to eigenvalue 0 for all feasible matrices. Therefore we can project the feasible set onto the cone spanned by all vectors orthogonal to e. Is there a generic way to detect and remove this kind of degeneracy? Can we exploit structural properties of the original problem in the projected space?

The traditional predictor–corrector approach to compute the new μ–value with respect to the predictor does not work very well for our algorithm. Is there an explanation for this behavior? What are good adaptive rules for computing μ? The quality of the search direction depends very much on the condition number of the final system matrix. What can be said about this matrix for points on the central trajectory? How much does the condition number depend on the linear operators $\mathcal{A}(\cdot)$ and $\mathcal{B}(\cdot)$?

For cutting plane approaches the traditional questions can be asked. What are important facets or inequalities? These may differ from those useful in simplex type algorithms since positive definiteness provides a lot of extra information. How can we separate them quickly? How can we judge whether an inequality will be tight in the optimal solution? Can we detect those not being tight before the optimal solution is computed? Whenever we find violated inequalities we need an interior point where we can return to. What should these interior points look like for different problems? What alternative strategies are there for finding good new starting points? Can we find starting points close to the central trajectory?

In particular for the max–cut problem we might want to ask whether the combination of the eigenvalue bound with all triangle facets leads to better worst case bounds for the max–cut problem. Or is it possible to give some estimates on the expected behavior?

Positive semidefinite programming certainly offers a lot of new possibilities. It will take some time and several answers before we know its exact place within combinatorial optimization but its effect will definitely be positive.

Acknowledgements

This work would not have been possible without the assistance and advice of several persons. In the first place I would like to thank Franz Rendl who was always willing to discuss all topics with me and supplied the most important ideas. I am also indebted to S. Poljak, G. Rinaldi, S. Karisch, and G. Sonnevend. I am grateful to the Christian Doppler Society and especially to Prof. R. Burkard for providing the necessary resources for this work. Finally, I thank my family for their support and patience.

Appendix A

Convergence Theorem

For the convenience of the reader we include the basic convergence result used in Theorem 2 as given in [4]. We are given a continuously differentiable function $f : P \to \Re$ which is defined on an open subset $P \subset R$ and which is bounded from below. The iterates are computed by

$$p^{i+1} = p^i + \alpha^i \Delta p^i.$$

The directional derivative is less than zero,

$$\left\langle \frac{\partial f}{\partial p}(p^i), \Delta p^i \right\rangle < 0,$$

and α_i is selected by the Golstein Armijo conditions $(0 < \delta_1 < \delta_2 < 1)$

$$\delta_1 \alpha \left| \left\langle \frac{\partial f}{\partial p}, \Delta p \right\rangle \right| \leq f(p) - f(p + \alpha \Delta p) \leq \delta_2 \alpha \left| \left\langle \frac{\partial f}{\partial p}, \Delta p \right\rangle \right|. \tag{A.1}$$

For this setting we have the following theorem.

Theorem 24 *If $\{p^i\}$ has a convergent subsequence in P then on this subsequence,*

$$\frac{\left\langle \frac{\partial f}{\partial p}(p^i), \Delta p^i \right\rangle}{||\Delta p^i||} \to 0.$$

Proof: For ease of notation we remove the denominator by scaling the directions Δp^i and steplengths α^i such that $||\Delta p^i|| = 1$. Assume, for contradiction, that $\left\langle \frac{\partial f}{\partial p}(p^i), \Delta p^i \right\rangle \not\to 0$ on the convergent subsequence. Then there must exist a further subsequence $\{p^k\}_{k \in \mathcal{K}}$ such that

$$\left| \left\langle \frac{\partial f}{\partial p}(p^k), \Delta p^k \right\rangle \right| \geq \gamma > 0$$

for all $k \in \mathcal{K}$. Using the fact that $f(\cdot)$ is bounded from below with the left–hand inequality of (A.1) we get $\left\{\alpha^k\right\}_{k\in\mathcal{K}} \to 0$. Since f is continuously differentiable, for each k there exists an $0 < \eta^k < 1$ such that

$$f(p^{k+1}) = f(p^k) + \alpha^k \left\langle \frac{\partial f}{\partial p}(p^k + \alpha^k \eta^k \Delta p^k), \Delta p^k \right\rangle .$$

We get

$$\frac{f(p^{k+1}) - f(p^k)}{\alpha^k} - \left\langle \frac{\partial f}{\partial p}(p^k), \Delta p^k \right\rangle = \left\langle \frac{\partial f}{\partial p}(p^k + \alpha^k \eta^k \Delta p^k) - \frac{\partial f}{\partial p}(p^k), \Delta p^k \right\rangle \to 0,$$

since $||\Delta p^k|| = 1$, $\alpha^k \to 0$, and ${p^k}_{k\in\mathcal{K}}$ is converging. Subtracting $\alpha^k \left|\left\langle \frac{\partial f}{\partial p}(p^k), \Delta p^k \right\rangle\right|$ from the left hand inequality of (A.1) yields

$$\frac{f(p^{k+1}) - f(p^k)}{\alpha^k} - \left|\left\langle \frac{\partial f}{\partial p}(p^k), \Delta p^k \right\rangle\right| \leq (\delta_2 - 1) \left|\left\langle \frac{\partial f}{\partial p}(p^k), \Delta p^k \right\rangle\right| .$$

The left hand side is converging to zero. Because of $\delta_2 < 1$ and $\left|\left\langle \frac{\partial f}{\partial p}(p^k), \Delta p^k \right\rangle\right| \geq \gamma$ for $k \in \mathcal{K}$ the right hand side is uniformly negative, which is a contradiction. □

Bibliography

[1] I. Adler, N. Karmarkar, G. C. Resende, and G. Veiga. An implementation of Karmarkar's algorithm for linear programming. *Mathematical Programming*, 44:297–335, 1989.

[2] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. Technical Report TR–93–050, International Computer Science Institute, Berkley, Sept. 1993.

[3] F. Alizadeh, J.-P. A. Haeberly, and M. L. Overton. A new primal–dual interior–point method for semidefinite programming. Technical report, Computer Science Department, Courant Institute of Mathematical Sciences, New York University New York, 1994.

[4] K. M. Anstreicher and J.-P. Vial. On the convergence of an infeasible primal–dual interior–point method for convex programming. Technical Report 16, Universite de Genève, Département d'économie commerciale et industrielle, Mar. 1993.

[5] V. Balakrishnan, E. Feron, S. Boyd, and L. El Ghaoui. Solving interpolation problems via generalized eigenvalue minimization. Technical report, Electrical Engineering Department, Stanford University, 1993. To appear in Proc. ACC, 1993.

[6] F. Barahona. The max–cut problem in graphs not contractible to K_5. *Oper. Res. Letters*, 2:107–111, 1983.

[7] F. Barahona. On cuts and matchings in planar graphs. *Mathematical Programming*, 60:53–68, 1993.

[8] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Oper. Res.*, 36:493–513, 1988.

[9] F. Barahona, M. Grötschel, and R. Mahjoub. Facets of the bipartite subgraph polytope. *Math. Oper. Res.*, 10(2):340–358, May 1985.

[10] F. Barahona, M. Jünger, and G. Reinelt. Experiments in quadratic 0–1 programming. *Mathematical Programming*, 44:127–137, 1989.

[11] F. Barahona and R. Mahjoub. On the cut polytope. *Mathematical Programming*, 36:157–173, 1986.

[12] L. Brunetta, M. Conforti, and G. Rinaldi. A branch–and–cut algorithm for the equicut problem. Technical report, Dipartimento die Matematica Pura e Applicata, Università degli Studi di Padova, and Istituto di Analisi dei Sistemi ed Informatica del CNR, 1994.

[13] T. J. Carpenter, I. J. Lustig, J. M. Mulvey, and D. F. Shanno. Higher–order predictor–corrector interior point methods with application to quadratic objectives. *SIAM J. Optimization*, 3(4):696–725, Nov. 1993.

[14] D. Cvetković, M. Doob, I. Gutman, and A. Torĝasev. *Recent Results in the Theory of Graph Spectra*, volume 36 of *Annals of discrete mathematics*. North-Holland, Amsterdam, 1988.

[15] D. Cvetković, M. Doob, and H. Sachs. *Spectra of Graphs*. VEB Deutscher Verlag der Wissenschaft, Berlin, 1979.

[16] C. De Simone and G. Rinaldi. A cutting plane algorithm for the max–cut problem. Technical Report 346, Istituto di Analisi dei Sistemi ed Informatica del CNR, Rome, 1992.

[17] C. Delorme and S. Poljak. Laplacian eigenvalues and the maximum cut problem. Rapport de Recherche 599, Universite de Paris–Sud, Centre d'Orsay, Laboratoire de Recherche en Informatique, 1990.

[18] C. Delorme and S. Poljak. Laplacian eigenvalues and the maximum cut problem. *Mathematical Programming*, 62(3):557–574, 1993.

[19] D. den Hertog and C. Roos. A survey of search directions in interior point methods for linear programming. *Mathematical Programming*, 52:481–509, 1991.

[20] M. Deza and M. Laurent. Facets for the cut cone I. *Mathematical Programming*, 56(2):121–160, 1992.

[21] M. Deza and M. Laurent. Facets for the cut cone II. *Mathematical Programming*, 56(2):161–188, 1992.

[22] M. Deza, M. Laurent, and S. Poljak. The cut cone III: On the triangle facets. *Graphs Combin.*, 8:125–142, 1992.

[23] U. Dieter. Dualität bei konvexen Optimierungs–(Programmierungs–)Aufgaben. *Unternehmensforschung*, 9(2):91–111, 1965.

[24] A. S. El-Bakry, R. A. Tapia, and Y. Zhang. A study of indicators for identifying zero variables in interior–point methods. *SIAM Review*, 36(1):45–72, Mar. 1994.

[25] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* Wiley, New York, 1968.

[26] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25, 1975.

[27] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman and Company, New York, 1979.

[28] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Technical report, M.I.T., 1994. Submitted to J. of the Association for Computing Machinery.

[29] G. H. Golub and C. F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore and London, 2 edition, 1990.

[30] A. Graham. *Kronecker Products and Matrix Calculus With Applications.* Mathematics and its Applications. Ellis Horwood Limited, Chichester, 1981.

[31] C. Helmberg, B. Mohar, S. Poljak, and F. Rendl. A spectral approach to bandwidth and separator problems in graphs. *Linear and Multilinear Algebra*, 1993. To appear.

[32] C. Helmberg, F. Rendl, R. J. Vanderbei, and H. Wolkowicz. An interior–point method for semidefinite programming. Technical Report 264, Department of Mathematics, Graz University of Technology, 1994. To appear in SIAM J. Optimization.

[33] B. Jansen, C. Roos, T. Terlaky, and J.-P. Vial. Primal–dual target–following algorithms for linear programming. Technical Report 93–107, Delft University of Technology, Faculty of Technical Mathematics and Informatics, Nov. 1993.

[34] F. Jarre. An interior–point method for minimizing the maximum eigenvalue of a linear combination of matrices. *Siam J. Control and Optimization*, 31(5):1360–1377, Sept. 1991.

[35] N. Karmakar. An new polynomial–time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[36] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thather, editors, *Complexity of Computer Computation*, pages 85–103. Plenum Press, New York, 1972.

[37] D. E. Knuth. The sandwich theorem. *The Electronic Journal of Combinatorics*, 1(# A1), 1994.

[38] M. Kojima, S. Mizuno, and A. Yoshise. A primal–dual interior point algorithm for linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming*, chapter 2, pages 29–47. Springer, New York, 1989.

[39] M. Kojima, S. Shindoh, and S. Hara. Interior–point methods for the monotone linear complementarity problem in symmetric matrices. Technical Report B–282, Department of Information Sciences, Tokyo Institute of Technology, Apr. 1994.

[40] M. Laurent and S. Poljak. On a positive semidefinite relaxation of the cut polytope. Rapport de Recherche 27, Laboratoire d'Informatique, URA 1327 du CNRS, Département de Mathématiques et d'Informatique, Ecole Normale Supérieure, Dec. 1993.

[41] M. Laurent and S. Poljak. The gap–inequality. Working paper, 1994.

[42] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. John Wiley and Sons Ltd, Chichester, 1990.

[43] L. Lovász. On the Shannon capacity of a graph. *IEEE Trans. Inform. Theory*, IT–25, 1979.

[44] L. Lovász and A. Schrijver. Cones of matrices and set–functions and 0–1 optimization. *SIAM J. Optimization*, 1(2):166–190, 1991.

[45] I. J. Lustig, R. E. Marsten, and D. F. Shanno. Computational experience with a primal–dual interior point method for linear programming. *Linear Algebra and its Applications*, 152:191–222, 1991.

[46] I. J. Lustig, R. E. Marsten, and D. F. Shanno. On implementing Mehrotra's predictor–corrector interior–point method for linear programming. *SIAM J. Optimization*, 2(3):435–449, Aug. 1992.

[47] N. Megiddo. Pathways to the optimal set in linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming*, chapter 8, pages 131–158. Springer, New York, 1989.

[48] S. Mehrotra. On finding a vertex solution using interior point methods. *Linear Algebra and its Applications*, 152:233–253, 1991.

[49] S. Mehrotra. On the implementation of a primal–dual interior point method. *SIAM J. Optimization*, 2:575–601, 1992.

[50] B. Mohar and S. Poljak. Eigenvalues in combinatorial optimization. In Brualdi, Friedland, and Klee, editors, *Combinatorial and Graph Theoretic Problems in Linear Algebra*. IMA Volume in Mathematics and Applications, Vol. 50, 1993.

[51] C. L. Monma and A. J. Morton. Computational experience with a dual affine variant of Karmakar's method for linear programming. *Operations Research Letters*, 6(6):261–267, Dec. 1987.

[52] R. D. C. Monteiro and I. Adler. Interior path following primal–dual algorithms. Part I: Linear programming. *Mathematical Programming*, 44:27–41, 1989.

[53] R. D. C. Monteiro and I. Adler. Interior path following primal–dual algorithms. Part II: Convex quadratic programming. *Mathematical Programming*, 44:27–41, 1989.

[54] R. D. C. Monteiro, I. Adler, and M. G. C. Resende. A polynomial–time primal–dual affine scaling algorithm for linear and convex quadratic programming and its power series extension. *Mathematics of Operations Research*, 15(2):191–214, May 1990.

[55] Y. Nesterov and A. Nemirovskii. *Interior–Point Polynomial Algorithms in Convex Programming*. SIAM Studies in Applied Mathematics, Philadelphia, 1993.

[56] M. L. Overton. Large–scale optimization of eigenvalues. *SIAM J. Optimization*, 2(1):88–120, Feb. 1992.

[57] G. Pataki. On the multiplicity of optimal eigenvalues. Technical Report #MSRR-604, Carnegie Mellon University, Aug. 1994.

[58] S. Poljak and F. Rendl. Nonpolyhedral relaxations of graph–bisection problems. Technical Report 55, DIMACS, Nov. 1992. To appear in SIAM J. Optimization.

[59] S. Poljak and F. Rendl. Node and edge relaxations of the max–cut problem. *Computing*, 52:123–137, 1994.

[60] S. Poljak, F. Rendl, and H. Wolkowicz. A recipe for semidefinite relaxation for $(0,1)$–quadratic programming. CORR Report 94-7, Department of Combinatorics and Optimization, University of Waterloo, Oct. 1994.

[61] S. Poljak and J. Rohn. Checking robust nonsingularity is NP–hard. *Math. Control Signals Systems*, 6:1–9, 1993.

[62] S. Poljak and Z. Tuza. The max–cut proplem — a survey. In L. Lovász and P. Seymour, editors, *Special Year on Combinatorial Optimization, DIMACS series in Discrete Mathematics and Theoretical Computer Science*. AMS, 1993. To be published.

[63] S. Poljak and Z. Tuza. On the expected relative error of the polyhedral approximation of the max–cut. Report no. 92757–or, Inst. für Diskrete MAthematik, Univ. Bonn, 1993.

[64] F. Rendl, R. J. Vanderbei, and H. Wolkowicz. Interior–point methods for max–min eigenvalue problems. Technical Report CORR 93-30, Department of Combinatorics and Optimization, Waterloo, Ontario, 1993. To appear in Proceedings of NATO conference on Nonlinear Programming, Il-Ciocco, Italy, 1993.

[65] H. Schramm and J. Zowe. A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. *SIAM J. Optimization*, 2:121–152, 1992.

[66] P. D. Seymour. Matroids and multicommodity flows,. *Europ. J. Combin.*, 2:257–290, 1981.

[67] G. Sonnevend, J. Stoer, and G. Zhao. Subspace methods for solving linear programming problems. *Annals of Operations Research, Spec. Vol. "Interior Point Methods"*, 1994. To appear.

[68] M. J. Todd and Y. Ye. A centered projective algorithm for linear programming. *Mathematics of Operations Research*, 15(3):508–529, Aug. 1990.

[69] L. Vandenberghe and S. Boyd. Positive definite programming. Technical report, Electrical Engineering Department, Stanford University, July 1994. Submitted to SIAM Review.

[70] L. Vandenberghe and S. Boyd. A primal–dual potential reduction method for problems involving matrix inequalities. Technical report, Electrical Engineering Department, Stanford University, May 1994. To be published in Mathematical Programming Series B.

[71] J.-P. Vial. Computational experience with a primal–dual interior–point method for smooth convex programming. Technical report, Département d'Economie Commercieale et Industrielle, Université de Genève, Apr. 1993. To appear in Optimization Methods and Software.

[72] H. Wolkowicz. Some applications of optimization in matrix theory. *Linear Algebra and its Applications*, 40:101–118, 1981.

Notation

$\mathcal{A}(\cdot)$	linear operator on symmetric matrices, $\mathcal{A}(\cdot) : \mathcal{M}_n \to \Re^k$
$\mathcal{B}(\cdot)$	linear operator on vectors, $\mathcal{B}(\cdot) : \Re^m \to \Re^k$
$c(E)$	sum of the edgeweights over the edges contained in E
$\delta(S)$	set of edges contained in the cut defined by $S \subset V$
$\mathrm{diag}(\cdot)$	extracts the diagonal of a square matrix
$\mathrm{Diag}(\cdot)$	converts a vector to a diagonal matrix
e	vector of all ones of appropriate size
E	set of edges
$\det(\cdot)$	determinant of a square matrix
$G(V, E)$	undirected (possibly weighted) graph
	on node set V and edge set E
I	identity matrix
J	$n \times n$ matrix of all ones
K_n	complete graph on n vertices
$L(G)$	Laplacian matrix of a Graph
$\mathcal{L}_n$	elliptope of dimension $\binom{n}{2}$,
	$\mathcal{L}_n = \{\tau(X) \| X \in \mathcal{M}_n, X \succeq 0, \mathrm{diag}(X) = e\}$
$\log(\cdot)$	logarithm with $\log(x \leq 0) = -\infty$, on vectors defined componentwise
$\lambda_{min}(\cdot)$ ($\lambda_{max}(\cdot)$)	smallest (largest) eigenvalue of a square matrix
$\mathcal{M}_n$	set of symmetric $n \times n$ matrices
$mc(G, c)$	value of maximum cut for a graph G with edgeweights c
$O(\cdot)$	$O(f(x)) = \{g : \Re \to \Re \mid \exists c_1 > 0, \exists c_2 > 0 : \forall x > c_2 : \|g(x)\| < c_1 \|f(x)\|\}$
$o(\cdot)$	$o(f(x)) = \left\{g : \Re \to \Re \mid \lim_{x \to \infty} \|\frac{g(x)}{f(x)}\| = 0\right\}$
$\Omega(\cdot)$	$\Omega(f(x)) = \{g : \Re \to \Re \mid \exists c_1 > 0, \exists c_2 > 0 : \forall x > c_2 : \|g(x)\| > c_1 \|f(x)\|\}$
P_n^{cut}	cut polytope of K_n
P_n^{met}	metric polytope of K_n

$\Re$	set of real numbers
$\sigma(G)$	max-cut bound by metric polytope
$\tau(\cdot)$	extracts upper triangle (without diagonal) from a square matrix
$\mathrm{tr}(\cdot)$	trace of square matrix, $\mathrm{tr}(A) = \sum_{i=1}^{n} a_{ii}$
V	set of vertices $V = \{1, \ldots n\}$
$\varphi(G)$	eigenvalue bound for max–cut
$\psi(G)$	max–cut bound by combining $\varphi(G)$ with triangle facets
χ^S	characteristic edgevector of the cut induced by $S \subset V$
$.^T$	transpose or adjoint operator
$\langle \cdot, \cdot \rangle$	inner product
$\|\| \cdot \|\|$	norm, $\|\|x\|\| = \sqrt{\langle x, x \rangle}$
$A \succeq B$ $(A \succ B)$	Löwner partial order on $\mathcal{M}_n$, in particular $A \succeq B$ $(A \succ B)$ if $A - B$ is positive semidefinite (positive definite).
$A \circ B$	Hadamard product for arbitrary $n \times m$ matrices, $(a_{ij}) \circ (b_{ij}) = (a_{ij} \cdot b_{ij})$
$R \triangle S$	$R \triangle S = (R \cup S) \setminus (R \cap S)$
(a, b)	open interval of real numbers from a to b
$[a, b]$	closed interval of real numbers from a to b