LINEAR PROGRAMMING (LP) APPROACHES TO SEMIDEFINITE PROGRAMMING (SDP) PROBLEMS

By

Kartik Krishnan

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Mathematics

Approved by the
Examining Committee:

John E. Mitchell, Thesis Adviser

Kristin Bennett, Member

Mark Goldberg, Member

Thomas Yu, Member

Rensselaer Polytechnic Institute
Troy, New York

April 2001
(For Graduation May 2002)

LINEAR PROGRAMMING (LP) APPROACHES TO SEMIDEFINITE PROGRAMMING (SDP) PROBLEMS

By

Kartik Krishnan

An Abstract of a Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Mathematics

The original of the complete thesis is on file
in the Rensselaer Polytechnic Institute Library

Examining Committee:

John E. Mitchell, Thesis Adviser
Kristin Bennett, Member
Mark Goldberg, Member
Thomas Yu, Member

Rensselaer Polytechnic Institute
Troy, New York

April 2001
(For Graduation May 2002)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Until recently, the study of interior point methods has dominated algorithmic research in semidefinite programming (SDP). From a theoretical point of view, these interior point methods offer everything one can hope for; they apply to all SDP's, exploit second order information and offer polynomial time complexity. Still for practical applications with many constraints k, the number of arithmetic operations, per iteration is often too high. This motivates the search for other approaches, that are suitable for large k and exploit problem structure.

Recently Helmberg and Rendl developed a scheme that casts SDP's with a constant trace on the primal feasible set as eigenvalue optimisation problems. These are convex nonsmooth programming problems and can be solved by bundle methods. In this talk we propose a linear programming framework to solving SDP's with this structure. Although SDP's are *semi infinite* linear programs, we show that only a small number of constraints, namely those in the bundle maintained by the spectral bundle approach, bounded by the square root of the number of constraints in the SDP, and others polynomial in the problem size are typically required. The resulting LP's can be solved rather quickly and provide reasonably accurate solutions. We also describe a cutting plane approach, where an SDP is solved as a sequence of LP's. However to make the resulting method competitive with interior point methods for the SDP, several refinements are necessary. In particular the cutting plane algorithm uses an interior point algorithm to solve the LP relaxations approximately, since this results in the generation of better constraints than a simplex cutting plane algorithm. We present numerical examples demonstrating the efficiency of the two approaches on combinatorial examples.

This is joint work with my advisor John Mitchell.

CHAPTER 1
The Semidefinite Programming (SDP) problem

1.1 Introduction

Semidefinite Programming (SDP) has been one of the most exciting and active research areas in optimisation recently. This tremendous interest was spurred by the discovery of important applications in combinatorial optimisation and control theory, the development of efficient interior point algorithms for solving SDP problems, and the depth and elegance of the underlying optimisation theory.

Semidefinite Programming (SDP) is concerned with choosing a symmetric matrix to optimise a linear function subject to linear constraints and a further crucial constraint that the matrix be positive semidefinite. It thus arises from the linear programming problem (LP) by replacing the vector of variables with a symmetric matrix and replacing the nonegativity restriction on the vector of variables, with the requirement that this symmetric matrix be positive semidefinite. SDP has other similarities with LP too. It is convex, has a rich duality theory (although not as strong as LP's), and also most interior points methods for LP, can be transformed into interior points methods for SDP. There are some difficulties though and we shall examine these later.

Semidefinite programming has its origins in the theory of linear matrix inequalities (LMI's) developed in the Soviet Union in the '40s,'50s,'60s. In the early 1970's Donath and Hoffman [13], and then Cullum, Donath and Wolfe [12] showed that NP hard graph partitioning algorithms could be solved as eigenvalue optimisation problems - as we shall see, these are closely connected with SDP. In 1979 Lovasz [41] formulated an SDP problem, that provided an upper bound on the Shannon capacity of the graph and thereby found the capacity of the pentagon, solving a long open conjecture. We shall return to the Lovasz theta function in Chapter 3.

However the key contributons to SDP came from Nesterov and Nemirovskii [46], who introduced the theory of self concordant barrier functions, which provided a general framework for solving various nonlinear optimisation including the SDP

in polynomial time, and Alizadeh [1] who showed that interior point methods pioneered by Karmarkar [29] for LP could be extended to SDP. Finally Goemans and Williamson [15] showed that the SDP relaxation could provide a provably good approximation to the max cut problem in combinatorial optimisation.

Excellent references for the SDP include the survey paper by Vandenberghe and Boyd [56] which discusses a number of applications, we describe three of these in section (1.6), the book by Boyd et al [9] which describes SDP with regard to control theory, the book by Nesterov and Nemirovskii [46], the paper by Alizadeh [1], the survey paper by Lewis and Overton [38] on eigenvalue optimisation, the SDP handbook edited by Wolkowicz et al [58], Helmberg's habilitation thesis [19] and finally Todd's [52] survey paper on semidefinite optimisation. We should also mention the excellent websites by Helmberg [20] on semidefinite programming, and Wright [59] on interior point methods, that allow one to keep abreast of recent developments in this exciting area of optimisation. The research in semidefinite programming has exploded in the late '80s and '90s. The recent handbook on semidefinite programming edited by Wolkowicz et al [58] lists 877 references, almost all since 1990.

This chapter is organised as follows. Section 2 presents some linear algebra preliminaries, especially on positive semidefinite matrices, section 3 introduces the SDP and indicates its many similarities to LP, section 4 describes the duality theory in SDP, section 5 describes the geometry of SDP, especially a bound on the rank of extreme matrices in SDP. section 5 describes three applications of SDP, section 6 describes interior point methods for semidefinite programming, section 7 describes large scale methods, applicable to SDP's with a large number of constraints m. Most of these large scale methods, require some special assumptions and formulate SDP's as nonlinear optimisation problems. Finally in section 8 we present a short overview of the main constribution of this thesis, which is a set of linear programming (LP) approaches to the semidefinite programming problem.

1.2 Linear Algebra Preliminaries

In this section we present some preliminaries regarding positive semidefinite matrices. An excellent reference is the book by Horn and Johnson [27].

1.2.1 Positive semidefinite matrices

In the following section we work with the set of symmetric matrices $\mathcal{S}^n$ which is a vector space in $R^{\frac{n(n+1)}{2}}$. A matrix $A \in \mathcal{S}^n$ is symmetric positive semidefinite if $A = A^T$ and

$$d^T A d \;\geq\; 0 \quad \forall d \in R^n \tag{1.1}$$

We denote this by $A \succeq 0$ and the set of symmetric positive semidefinite matrices by S^n_+. A matrix $A \in \mathcal{S}^n$ is symmetric positive definite if

$$d^T A d \;>\; 0 \quad \forall d \in R^n \backslash \{0\} \tag{1.2}$$

We denote this by $A \succ 0$ and the set of symmetric positive definite matrices by S^n_{++}. We also introduce some other terminology that will be useful later. diag(X) is a vector whose components are the diagonal elements of X, and Diag(d) is a diagonal matrix, with the components of d. In the succeeding sections we use Trace(X) and tr(X) interchangeably, to denote the trace of the symmetric matrix X.

The following are various ways of characterising positive semidefinite (definite) matrices.

1. Any principal submatrix of a postive semidefinite (definite) matrix is positive semidefinite (definite). In particular, all diagonal elements of a positive semidefinite matrix ≥ 0 (positive).

2. Each eigenvalue of a positive semidefinite (definite) matrix ≥ 0 (positive real number).

Lemma 1 *The positive semidefiniteness of a matrix can be checked in $\frac{n^3}{3}$ arithmetic operations, using a Cholesky factorisation.*

1.2.2 The cone of semidefinite matrices

A set $C \subseteq R^n$ is a cone if it is closed under *nonnegative* multiplication and addition i.e. $x, y \in C \Rightarrow \lambda(x+y) \in C, \forall \lambda \geq 0$.

Lemma 2 *S_+^n is a full dimensional, closed convex cone in $R^{\binom{n+1}{2}}$.*

The set of positive definite matrices S_{++}^n is not a cone because $0 \notin S_{++}^n$. In fact S_{++}^n is the interior of the cone S_+^n and the boundary of S_+^n consists of the positive semidefinite matrices having at least one zero eigenvalue.

1.3 Semidefinite Programs

Consider the following pair of semidefinite programming problems

$$\begin{array}{lrcll} \min & C \bullet X & & & \\ \text{s.t.} & A_i \bullet X & = & b_i & i = 1, \ldots, m \\ & X & \succeq & 0 & \end{array} \tag{1.3}$$

with dual

$$\begin{array}{lrcrcl} \max & b^T y & & & & \\ \text{s.t.} & \sum_{i=1}^m y_i A_i & + & S & = & C \\ & & & S & \succeq & 0 \end{array} \tag{1.4}$$

where the variables $X, S \in \mathcal{S}^n$, the space of real symmetric $n \times n$ matrices, $b \in R^m$, the matrices $A_i, i = 1, \ldots, m \in \mathcal{S}^n$ and $C \in \mathcal{S}^n$ are the given problem parameters. The inequality $X \succeq 0$ means that X is positive semidefinite namely $d^t X d \geq 0, \forall d \in R^n$. Finally $C \bullet X$ stands for the inner product of the symmetric matrices C and X and is defined as

$$C \bullet X \;=\; \text{Trace}(CX) \;=\; \sum_{i=1}^n \sum_{j=1}^n C_{ij} X_{ij}$$

Here $\text{Trace}(X) = \sum_{i=1}^n X_{ii}$.

We can think of the SDP as a generalisation of the standard linear programming problem

$$\begin{array}{lrcll} \min & c^T x & & & \\ \text{s.t.} & a_i^T x & = & b_i & \forall i = 1, \ldots, m \\ & x & \geq & 0 & \end{array} \tag{1.5}$$

with dual

$$\begin{array}{lrcrcl} \max & b^T y & & & & \\ \text{s.t.} & \sum_{i=1}^m y_i a_i & + & s & = & c \\ & & & s & \geq & 0 \end{array} \tag{1.6}$$

Here a_i denote the rows of a $m \times n$ matrix A. If we compare (1.3) with (1.5), we find that the SDP is indeed an LP, where the vectors c, x, s, a_i's are replaced by matrices C, X, S, A_i's respectively and the elementwise nonegativity constraints $x \geq 0, s \geq 0$ are replaced by the requirement that $X \succeq 0, S \succeq 0$. Thus semidefinite programming is essentially linear programming over the cone of semidefinite matrices.

1.4 Duality in SDP

Consider the semidefinite programming problems introduced in the previous section.

$$\begin{array}{lrcll} \min & C \bullet X & & & \\ \text{subject to} & \mathcal{A}(X) & = & b & (SDP) \\ & X & \succeq & 0, & \end{array}$$

with dual

$$\begin{array}{lrcrcll} \max & b^T y & & & & & \\ \text{subject to} & \mathcal{A}^T y & + & S & = & C & (SDD) \\ & & & S & \succeq & 0 & \end{array}$$

where $\mathcal{A} : \mathcal{S}^n \to R^k$ and $\mathcal{A}^T : R^k \to \mathcal{S}^n$ are of the form

$$\mathcal{A}(X) \;=\; \begin{bmatrix} A_1 \bullet X \\ \vdots \\ A_k \bullet X \end{bmatrix} \quad \text{and} \quad \mathcal{A}^T y \;=\; \sum_{i=1}^k y_i A_i$$

with $A_i \in \mathcal{S}^n, i = 1, \ldots, k$. $C \in \mathcal{S}^n$ is the cost matrix, $b \in R^k$ the RHS vector.

Lemma 3 *(Weak Duality) If X is feasible in (SDP) and (y,S) in (SDD), then*

$$C \bullet X - b^T y \;=\; X \bullet S \;\geq\; 0 \tag{1.7}$$

Proof:

$$C \bullet X - b^T y \;=\; (\textstyle\sum_{i=1}^m y_i A_i + S) \bullet X - b^T y \;=\; \sum_{i=1}^m (A_i \bullet X) y_i + S \bullet X - b^T y \;=\; S \bullet X$$

Since X and S are semidefinite matrices we have $S \bullet X = X \bullet S >= 0$. □

We call the difference between the optimal value of (P) and (D), which is always nonnegative by the result above, the *duality gap*. Strong duality is the assertion that the duality gap is zero and that both problems attain their optima whenever both problems are feasible. This is true for the LP, but it does not always hold for SDP problems.

We now turn to conditions ensuring that strong duality holds. It turns out that a strict feasibility (Slater) condition suffices. We will first need a few definitions

$$\begin{aligned} F(P) &:= \{X : \mathcal{A}(X) = b, X \succeq 0\}, \\ F^0(P) &:= \{X \in F(P) : X \succ 0\}, \\ F(D) &:= \{(y, S) : \mathcal{A}^T y + S = C, S \succeq 0\} \\ F^0(D) &:= \{(y, S) \in F(D) : S \succ 0\} \end{aligned}$$

Lemma 4 *(Strong Duality) Suppose that $F(P)$ and $F^0(D)$ are nonempty. Then (SDP) has a nonempty compact set of optimal solutions, and the optimal values of (SDP) and (SDD) are equal.*

Corollary 1 *Suppose $F^0(P)$ and $F(D)$ are nonempty. Then (SDD) has a nonempty compact set of optimal solutions, and there is no duality gap*

Corollary 2 *Suppose (SDP) and (SDD) have strictly feasible solutions. Then each has a nonempty compact set of optimal solutions, and there is no duality gap.*

1.5 Geometry of SDP

In this section we present a lemma from Pataki [48], regarding the rank of extremal matrices in semidefinite programming. This lemma will be useful, when we consider the minimum number of vectors r in the bundle maintained by the spectral bundle approach discussed in section 1.8.1 and Chapter 2.

Lemma 5 *(i) Let F be a face of dimension k of the feasible set of (SDP). For $X \in F$ the rank r of X is bounded by*

$$\binom{r+1}{2} \leq m + k$$

(ii) Let F be a face of dimension k of the feasible set of (SDD). For $Z \in F$ the rank r of Z is bounded by

$$\binom{r+1}{2} \leq \binom{n+1}{2} - m + k$$

As a corollary, we obtain a bound on the number of nonzeros in extreme solutions of linear programs. Consider the LP with feasible set $\phi = \{x \in \mathcal{R}^n : Ax = b, x \geq 0\}$. Here $A \in \mathcal{R}^{m \times n}$. Note that since $x \geq 0 \Leftrightarrow \text{diag}(x) \succeq 0$, the rank of a vector x, as we may call it, is equal to the number of nonzeros in the vector x. Thus (5) gives $r \leq m + k$. If we are at the extreme point of ϕ, i.e. a face of dimension of zero 0, we get $r \leq m$. This corresponds to the number of nonzeros in a basic feasible solution x.

1.6 Examples

In this section, we present three applications of semidefinite programming. These applications are taken from Vandenberghe and Boyd [56]. The first example appears in control theory, while the SDP relaxation appearing in example 2, was used by Goemans and Williamson [15] in their celebrated 0.878 approximation algorithm for the max cut problem. The third example is on the minimum eigenvalue of a symmetric matrix.

1.6.1 Example 1 : Minimising the maximum eigenvalue

Problems of this type arise in control theory, structural optimisation, combinatorial optimisation etc. For example in control theory it might appear as stabilising a differential equation.

The problem is as follows :

Suppose we have a symmetric matrix, say $A(x)$ depending affinely (linearly) on a vector $x \in \mathcal{R}^k$: $A(x) = A_0 + \sum_{i=1}^{k} x_i A_i$, where $A_i \in \mathcal{S}^n, i = 1, \ldots, k$. Now we wish to choose x to minimise the maximum eigenvalue of $A(x)$ denoted by $\lambda_{max}(A(x))$.

Note that $\lambda_{max}(A(x)) \leq t$ iff $\lambda_{max}(A(x) - tI) \leq 0$. Since $\lambda_{max}(.)$ is a convex function we have $\lambda_{max}(A(x) - tI) = -\lambda_{min}(tI - A(x))$. Thus we have $\lambda_{min}(tI - A(x)) \geq 0$. This holds iff $tI - A(x) \succeq 0$. The SDP in dual form (SDD) is

$$\begin{array}{lll} \max & -t & \\ \text{s.t.} & tI - A(x) & \succeq \quad 0 \end{array} \tag{1.8}$$

The dual variable is $y = (t, x)$.

1.6.2 Example 2 : The Max Cut Problem

Given a graph $G = (V, E)$, and a nonnegative weight vector $w = w_{ij} \in \mathcal{R}_+^E$. For $S \subseteq V$, $\delta(S)$ denotes $\delta(S) = \{(i, j) \in E : i \in S, j \notin S\}$, the *cut* determined by S, with *weight* equal to $w(\delta(S)) = \sum_{i,j \in \delta(S)} w_{ij}$. We want to find the cut of maximum weight.

We can assume that the graph is complete by setting $w_{ij} = 0$ for all non edges (i, j); we also set $w_{ii} = 0$, $\forall i$.

Let us define a vector $x \in \mathcal{R}^n$, with $x_i = 1$, if $i \in S$ and -1 otherwise. Then $x_i x_j = -1$ if $(i, j) \in \delta(S)$ and 1 otherwise. Now define the Laplacian matrix $L \in \mathcal{S}^n$, by setting $L_{ij} = -\frac{w_{ij}}{4}$, $i \neq j$, and $L_{ii} = \sum_j \frac{w_{ij}}{4}$, $\forall i$.

We then have

$$w(\delta(S)) \;=\; \tfrac{1}{2} \textstyle\sum_{i<j} w_{ij}(1 - x_i x_j) \;=\; \tfrac{1}{4} \sum_i \sum_j w_{ij}(1 - x_i x_j) \;=\; x^T L x$$

Since every $(1, -1)$ vector corresponds to a cut, the max cut problem is equiv-

alent to the following integer quadratic programming problem

$$\max \quad x^T L x, \quad x_i \in \{1, -1\}, i \in V$$

This is also equivalent to the following nonconvex quadratically constrained quadratic problem

$$\max \quad x^T L x, \quad x_i^2 = 1, i \in V \tag{1.9}$$

Let us define the rank one matrix $X = xx^T$. We have $X_{ii} = x_i^2 = 1, \forall i$ and $X_{ij} = x_i x_j$. Also $X \succeq 0$, since $d^T X d = (d^T x)^2 \geq 0$. Now we rewrite the objective function in (1.9) as $x^T L x = \text{tr}(x^T L x) = L \bullet xx^T = L \bullet X$. Here the first equality follows from the fact that $x^T L x$ is a scalar, and the second follows from $\text{tr}(AB) = \text{tr}(BA)$.

Thus (1.9) is equivalent to

$$\max \quad L \bullet X, \quad X_{ii} = 1, \forall i \in V \quad X \succeq 0 \quad X \text{rank one}$$

If we relax the last constraint, we get the SDP relaxation (1.10) of the max cut problem

$$\max \quad L \bullet X, \quad X_{ii} = 1, \forall i \in V \quad X \succeq 0 \tag{1.10}$$

1.6.3 Example 3 : The minimum eigenvalue of a symmetric matrix

As a final example, we indicate how we could in practice express the minimum eigenvalue of a symmetric matrix as an SDP. This characterisation of the minimum eigenvalue will also be useful, when we motivate the spectral bundle method in section 1.8.1 and Chapter 2.

From the variational characterisation of the minimum eigenvalue of a symmetric matrix A, we have

$$\begin{array}{rcll} \lambda_{min}(A) & = & \min & x^T A x \\ & & x^T x = 1 & \end{array} \tag{1.11}$$

Now define a matrix $X = xx^T$. The objective function in (1.11) can be expressed as $x^T A x = \mathrm{tr}(Axx^T) = A \bullet X$.

Also $x^T x = \mathrm{tr}(xx^T) = \mathrm{tr}(X) = 1$.

Finally $X = xx^T$ implies that X is rank one and positive semidefinite.

A SDP relaxation of the minimum eigenvalue is

$$\begin{array}{rcll} \lambda_{min}(A) & = & \min & A \bullet X \\ & & \{X : \mathrm{tr}(X) = 1, X \succeq 0\} & \end{array} \tag{1.12}$$

Note that we have dropped the requirement that X need to be rank one. However this is not necessary for the following reason.

Every positive semidefinite matrix X has a spectral decomposition $X = \sum_{i=1}^{n} \lambda_i v_i v_i^T$, where $\lambda_i, i = 1, \ldots, n$ are the eigenvalues and $q_i, i = 1, \ldots, n$ the corresponding eigenvectors.

Note that the constraint $\mathrm{tr}(X) = 1$ implies that $\sum_{i=1}^{n} \lambda_i = 1$. Also all the $\lambda_i \geq 0, \forall i$, since X is psd. Thus $\{X : \mathrm{tr}(X) = 1, X \succeq 0\}$ is precisely the convex hull of $\{qq^T : q^T q = 1\}$. Now since $C \bullet X$ is a linear function in the matrix variable X, and minimising a linear function over a set gives the same result as minimising it over its convex hull. Thus (1.11) and (1.12) are equivalent.

Thus we can express the minimum eigenvalue of a symmetric matrix A as an SDP (1.12).

1.7 Interior Point Algorithms for SDP

Interior Point methods for the solution of SDP problems were first proposed by [1] and [46]. Alizadeh showed that most, if not all primal or dual algorithms for linear programming could be extended to SDP in a mechanical way. On the other hand Nesterov and Nemirovskii presented a deep and unified theory of interior point

methods for solving general cone programming problems using the notion of the self concordant barrier. In particular, since the cone of positive semidefinite matrices admits a self concordant barrier function, they showed that semidefinite programs can be *solved in polynomial time*.

The basic idea is to transform a constrained optimisation problem (in this case a SDP) into an unconstrained problem. One way to handle inequality constraints consists in adding a barrier term to the cost function. In case of minimisation problems, the barrier term is small in the interior of the feasible region, but grows to infinity as we approach the boundary of the feasible region. Thus if we start in the interior of the feasible region, the barrier term prevents us from leaving the feasible region. Furthermore, a descent direction in a point close to the boundary will automatically point away from the boundary. However the optimal solution is usually located on the boundary. In order to produce a sequence of iterates that converge to the optimum, a mechanism has to be provided that reduces the influence of the barrier function as the optimisation process continues. This is achieved by weighting the barrier term and diminishing the weight successively. This is a sequential unconstrained minimisation technique, and under certain conditions the minima of the sequence of barrier problems can be shown to converge to an optimal solution of the original problem. Typically the minima of the subproblems is not computed exactly but approximated by a few Newton steps. Since Newton's method exploits second order information, the algorithms converge very fast. An approximate optimal solution (to any degree of precision) is obtained within a polynomial number of iterations. On the other hand the computation of a single step is computationally rather expensive and this limits these methods to problems of moderate size, say around 3000 constraints.

The important references on interior point methods for SDP include Alizadeh et al [2], Helmberg et al [23], Monteiro [44], Todd et al [55]. These are primal dual algorithms. Benson et al [7] propose a dual scaling algorithm. Finally Zhang [63] develops a framework to extend some primal dual interior point algorithms from linear programming to semidefinite programming.

In this section we discuss interior point algorithms for the SDP. We present a

generic interior point algorithm for the SDP and consider the computational issues involved.

Consider the semidefinite programming problem

$$\begin{array}{lrcll} \min & C \bullet X & & & \\ \text{subject to} & \mathcal{A}(X) & = & b & (SDP) \\ & X & \succeq & 0, & \end{array}$$

with dual

$$\begin{array}{lrcrcll} \max & b^T y & & & & & \\ \text{subject to} & \mathcal{A}^T y & + & S & = & C & (SDD) \\ & & & S & \succeq & 0 & \end{array}$$

where $X, S \in \mathcal{S}^n$, the space of real symmetric $n \times n$ matrices.

We will assume that strong duality holds for this pair, i.e. both the primal as well as the dual are strictly feasible.

Interior point algorithms start within the cone of positive semidefinite matrices. In order to avoid leaving the cone during the optimisation process the task of minimising the original SDP is replaced by approximately minimising a sequence of auxiliary barrier problems. The auxiliary barrier problem is of the form

$$\begin{array}{lrcl} \min & C \bullet X - \mu \log\det(X) & & \\ \text{s.t.} & \mathcal{A}(X) & = & b \\ & X & \succ & 0 \end{array}$$

Here $\mu > 0$ is the barrier parameter and $-\log\det(X)$ is the *self concordant* barrier function. This barrier function grows to infinity if an eigenvalue of X tends to zero i.e. as we approach the boundary of the semidefinite cone.

We transform the barrier problem into an unconstrained problem, by introducing a Lagrange multiplier y for the equality constraints.

$$L_\mu(X, y) \;=\; C \bullet X - \mu \log\det(X) + y^T(b - \mathcal{A}(X))$$

Note that $L_\mu(X, y)$ is a strictly convex function on $\mathcal{S}^n_+$. Writing down the *KKT*

conditions for optimality we have.

$$\begin{aligned} \bigtriangledown_X L_\mu &= C - \mu X^{-1} - \mathcal{A}^T y &= 0 \\ \bigtriangledown_y L_\mu &= b - \mathcal{A}(X) &= 0 \end{aligned}$$

Setting $S = \mu X^{-1}$ we can rewrite the *KKT* conditions as

$$\begin{aligned} \mathcal{A}(X) &= b, \quad X \succ 0 \\ \mathcal{A}^T y + S &= C, \quad S \succ 0 \\ XS &= \mu I \end{aligned} \tag{1.13}$$

The first two conditions correspond to primal and dual feasibility. For $\mu = 0$, the third condition corresponds to the complementary slackness condition $XS = 0$.

We denote the solution to 1.13 for some fixed $\mu > 0$ by (X_μ, y_μ, S_μ). This forms the *central path* which is a smooth curve, and which converges to the optimal solution (X^*, y^*, S^*), as $\mu \to 0$.

If we solve (1.13) by Newton's method, we get the following linearised system

$$\begin{aligned} \mathcal{A} \triangle X &= -(\mathcal{A}X - b) &= F_p \\ \mathcal{A}^T \triangle y + \triangle S &= -(\mathcal{A}^T y + S - C) &= F_d \\ \triangle XS + X \triangle S &= \mu I - XS \end{aligned} \tag{1.14}$$

Since X and S are matrices, they do not always commute i.e. $XS \neq SX$. In general we cannot expect that there exist symmetric $\triangle X$ and $\triangle Z$ that solve 1.14. However we can introduce a symmetrisation operator

$$H_P(M) = \tfrac{1}{2}(PMP^{-1} + (PMP^{-1})^T)$$

In the simplest case when $P = I$ we get $H_I(M) = \frac{1}{2}(M + M^T)$. Different P correspond to different search directions. A good survey of the various search directions appears in Todd [53].

We are now ready to present the algorithmic framework for a typical interior point scheme for the SDP.

Generic Interior Point Algorithm for the SDP

Input : $\mathcal{A}, b, C$ and some starting point (X^0, y^0, S^0) with $X^0 \succ 0$ and $S^0 \succ 0$.

1. Choose the barrier parameter μ.
2. Compute $(\triangle X, \triangle y, \triangle S)$ by solving (1.14).
3. Choose some $\alpha \in (0, 1]$ so that $X + \alpha \triangle X$ and $S + \alpha \triangle S$ are positive definite.
4. Set $(X, y, S) = (X + \alpha \triangle X, y + \alpha \triangle y, S + \alpha \triangle S)$.
5. If $||\mathcal{A}X - b||$ and $||\mathcal{A}^T y + S - C||$ and $X \bullet S$ are small enough, then **stop**, else **goto 1**.

We need strong restrictions on the initial starting point, and the values of μ and α to prove polynomial iteration complexity.

We now examine some computational aspects of interior point methods in semidefinite programming. Consider (1.14).

We first express $\triangle X$ in terms of $\triangle S$, and then write $\triangle S$ in terms of $\triangle y$. This yields the folowing system depending on $\triangle y$ only.

$$\begin{aligned} \mathcal{A}(X\mathcal{A}^T(\triangle y)S^{-1}) &= b - \mathcal{A}(\mu S^{-1} + XF_d S^{-1}) \\ \triangle S &= F_d - \mathcal{A}^T \triangle y \\ \triangle X &= \mu S^{-1} - X - X \triangle S S^{-1} \end{aligned} \tag{1.15}$$

After solving this system, we symmetrise $\triangle X$ by setting $\triangle X = \frac{1}{2}(\triangle X + \triangle X^T)$.

Consider $My = \mathcal{A}(X\mathcal{A}^T(y)S^{-1})$. The ith row of My is given by

$$A_i \bullet X\mathcal{A}^T(y)S^{-1} = \sum_{j=1}^m y_j \text{Trace}(XA_i S^{-1} S_j A_j)$$

Therefore we have $M_{ij} = \text{Trace}(XA_i S^{-1} A_j)$. This matrix is symmetric and positive definite, if we assume $\mathcal{A}(.)$ to have full row rank.

Solving for $\triangle y$ requires $\frac{m^3}{3}$ arithmetic operations, using a Cholesky decomposition. Morever M has be to recomputed in each iteration. An efficient way to build one row of M is the following

1. Compute XA_iS^{-1} once in $O(n^3)$ time
2. Determine the m single elements via $XA_iS^{-1} \bullet A_j$ in $O(mn^2)$.

In total the construction of M requires $O(mn^3 + m^2n^2)$ arithmetic operations and so the construction of M is the most expensive operation in each iteration.

In many applications the constraint matrices A_i have a special structure. For most of the combinatorial optimisation problems such as max cut etc, these matrices have a rank one structure. This reduces the computation time to $O(mn^2 + m^2n)$ operations. Benson et al [7] have proposed a dual scaling algorithm that exploits this rank one structure, and the sparsity in the dual slack matrix. However in their approach the matrix M is dense, and the necessity to store and factorise this dense matrix M limits the applicability of these methods to problems with around 3000 constraints on a well equipped work station.

1.8 Large Scale Approaches

We now turn to methods that are based on nonsmooth and smooth optimisation techniques for nonlinear programming formulations of (SDP) or (SDD). Some of these place restrictions on the SDP problems that can be handled.

1.8.1 The Spectral bundle method

We only give a brief overview of the spectral bundle method in this section. Since the vectors in the bundle P maintained by this scheme, will be used as cutting planes in our LP approach described in Chapter 3, we provide a detailed description of this method in Chapter 2.

The spectral bundle scheme is due to Helmberg and Rendl [22]. The bundle is suitable for large m. However it is only a first order method, with no polynomial bound on the number of iterations, unlike the interior point methods, discussed in the previous section. Nevertheless excellent computational results have been

obtained for problems that are inaccessible to the latter methods due to their size. A second order method which converges globally and which enjoys asymptotically a quadratic rate of convergence was recently devloped by Oustry [47]. Helmberg and Kiwiel [21] also extended the method to problems with bounds.

The essential idea of the method is the following

Consider the eigenvalue optimisation problem

$$\max_y \quad a\lambda_{min}(C - \mathcal{A}^T \mathrm{y}) \quad + \quad b^T \mathrm{y} \tag{1.16}$$

Problems of this form are equivalent to the dual of semidefinite programs (SDP), whose primal feasible set has a constant trace, i.e. Trace$(X) = a$ for all $X \in \{X \succeq 0 : \mathcal{A}(X) = b\}$. This can be easily verified as follows. From section 1.6.3 on the minimum eigenvalue function, we have $\lambda_{min}(C - \mathcal{A}^T y) = \min_{X:\mathrm{tr}X=1, X\succeq 0}(C - \mathcal{A}^T y) \bullet X$. If this X also satisfies $\mathcal{A}X = b$, then we essentially the solving the primal problem (SDP), with the additional constraint that $\mathrm{tr}(X) = a$. Also we require that either the primal or the dual SDP contain a strictly feasible point, so that strong duality $XS = 0$ holds for the primal dual pair.

In the spectral bundle scheme the maximum eigenvalue is approximated by means of vectors in the subspace spanned by the bundle P. This amounts to solving the following problem (1.17) in lieu of (1.16).

$$\max_y \quad a\lambda_{min}(P^t(C - \mathcal{A}^T y)P) + b^T y \tag{1.17}$$

This in turn is equivalent to solving the following (SDP)

$$\min_{X:\mathrm{tr}(X)=a, \mathcal{A}(PXP^T)=b, X\succeq 0} \quad C \bullet (PXP^T) \tag{1.18}$$

(1.18) implies that we are approximately solving (SDP), by considering only a subset of the feasible X matrices. By keeping the number of columns r in P small, the resulting SDP can be solved quickly. Helmberg and Rendl [22] using results from

Pataki [48] were able to show that r is bounded by $\binom{r+1}{2} \leq k+1$, i.e. the dimension of the subspace P is roughly bounded by the square root of number of constraints. The optimum solution of (1.17) typically produces an indefinite dual slack matrix. The negative eigenvalues and corresponding eigenvectors are used to the update the subspace, i.e. P and the process is iterated.

To summarise the essential idea of the bundle method is the following Todd [52]; it can be thought as providing an approximation by considering only a subset of the feasible X matrices, using this to improve the dual solution y, and using this in turn to improve the subset of feasible solutions in the primal.

1.8.2 A nonlinear programming algorithm employing low rank factorisation

We now turn to methods that generate nonconvex nonlinear programming problems in a lower dimension, and apply interior point methods for their solution. The method is due to Burer, Monteiro and Zhang [10, 11].

Consider the following SDP and its dual

$$\min \quad C \bullet X, \quad \mathrm{diag}(X) = d, \quad \mathcal{A}(X) = b, \quad X \succeq 0 \tag{1.19}$$

with dual

$$\max \quad d^T z + b^T y, \quad \mathrm{Diag}(z) + \mathcal{A}^T y + S = C, \quad S \succeq 0 \tag{1.20}$$

Burer et al [11] suggest solving (1.20) by means of an equivalent nonlinear programming problem, obtained by eliminating variables. They consider only strictly feasible solutions of (1.20). Their procedure is based on a theorem stating that for each $(w, y) \in \mathcal{R}^n_{++} \times \mathcal{R}^m$, there is a unique strictly lower triangular matrix $\bar{L} = \bar{L}(w, y)$ and a unique $z = z(w, y) \in \mathcal{R}^n$ satisfying

$$C - \mathrm{Diag}(z) - \mathcal{A}^T y \;\; = \;\; (\mathrm{Diag}(w) + \bar{L})(\mathrm{Diag}(w) + \bar{L})^T \tag{1.21}$$

and that $\bar{L}(w,y)$ and $z(w,y)$ are infinitely differentiable. By requiring S to have a nonsingular Cholesky factorisation, we are implicitly requiring that S be positive definite. Thus (1.20) can be replaced by the smooth but nonconvex problem

$$\max_{w,y} \quad d^T z(w,y) + b^T y, \quad w > 0 \tag{1.22}$$

The authors then suggest a log barrier method and a potential reduction method to solve (1.22). Although the objective function in (1.22) is nonconvex, Burer et al prove global convergence of their methods, and have obtained excellent computational results on large scale problems.

1.8.3 The method due to Vanderbei and Benson

We conclude this chapter with the method due to Vanderbei and Benson [57].

The method essentially amounts to factorising the primal positive semnidefinite matrix X as $X = LDL^T$. Here L is a unit lower triangular matrix, and D is a diagonal matrix. Note that D is unique on the domain of positive semidefinite matrices, L however is unique only when X is positive definite.

Let $D = \mathrm{Diag}(d(X))$. Here $d(X) \in \mathcal{R}^n$. Vanderbei and Benson [57] establish that d is a concave function. The constraint that $X \succeq 0$ is replaced with the requirement that $d(X) \geq 0$ i.e. be nonnegative vector.

1.9 A linear programming (LP) approach to SDP

The main objective of this thesis is to present a linear programming (LP) approach to semidefinite programming. A semidefinite programming problem can be regarded as a convex nonsmooth optimization problem, so it can be represented as a semi-infinite linear programming problem. However in Chapter 3, we show that only a small number of constraints, namely those in the bundle maintained by the spectral bundle approach, bounded by the square root of the number of constraints in the SDP, in conjunction with families of problem specific constraints labelled as *box* constraints are typically required. In Chapter 4 we describe a cutting plane LP approach to solve an SDP, as a sequence of linear programs. To make

the resulting method competitive with interior point SDP approaches several refinements are necessary. The cutting plane method uses an interior point algorithm to solve the linear programming relaxations approximately, because this resulted in the generation of better constraints than a simplex cutting plane method. We present numerical examples demonstrating the efficiency of the two approaches on combinatorial examples.

CHAPTER 2
The spectral bundle method

2.1 Introduction

In the previous chapter we introduced the semidefinite programming problem (SDP), and also interior point methods to solving the SDP. From a theoretical point of view, these interior point methods offer everything one can hope for. They are applicable to all SDP's, exploit second order information, and offer polynomial time complexity. The main computational task here, is the factorisation of the *Schur complement* matrix of size m, in computing the search direction. Typically this is a dense matrix and a Cholesky factorisation would require $\frac{m^3}{3}$ arithmetic operations, limiting the applicability of these methods to problems with about 3000 constraints on a well equipped work station. This motivates the search for other approaches, that are suitable for large m and exploit problem structure. The spectral bundle method that we describe in this chapter offers these features. However the method is only applicable to SDP's with a constant trace on the primal feasible set. The method recasts such SDP's as eigenvalue optimisation problems. Morever the spectral method is only a first order method, without any polynomial bound on the number of iterations. The method is due to Helmberg and Rendl [22]. Good references include [22], [21], [58] and [19].

This chapter is organised as follows. Section 2 explains how a SDP with a constant trace on the primal feasible set can in practice, be formulated as an eigenvalue optimisation problem. The maximum eigenvalue is a nonsmooth convex function and we study some of its properties in section 3. A general technique for minimising nonsmooth functions is the proximal bundle approach and we review this in section 4. We indicate how the proximal bundle idea can be extended to the spectral bundle method in section 5. We explain one iteration of the spectral bundle method in detail in section 6. In the bundle approach the maximum eigenvalue is approximated by means of vectors in the bundle P. This amounts to solving a fairly small quadratic semidefinite problem with a concave objective function and

we examine this in section 7. In section 8 we present the entire spectral bundle algorithm. We conclude with the salient features of the bundle approach in section 9.

2.2 The SDP as an eigenvalue optimisation problem

Consider the following primal dual pair of semidefinite programs

$$\begin{array}{llllll} & \text{Maximise} & C \bullet X & & \text{Minimise} & b^t y \\ (P) & \text{s.t.} & \mathcal{A}(X) = b & (D) & \text{s.t.} & S = \mathcal{A}^t y - C \\ & & X \succeq 0 & & & S \succeq 0 \end{array}$$

We assume that the following condition holds

$$\exists \quad \text{a} \quad \bar{y} \in R^m \quad \text{with} \quad \mathcal{A}^T \bar{y} = I \tag{2.1}$$

This assumption allows us to formulate (D) as an eigenvalue optimisation problem.

$$\text{Minimise}_y \quad a\lambda_{max}(C - \mathcal{A}^T y) \quad + \quad b^t y \tag{2.2}$$

We have the following proposition due to Helmberg [19]

Theorem 1 *[Helmberg 00] If $\mathcal{A}$ satisfies $\mathcal{A}^T \bar{y} = I$, for some $\bar{y} \in R^k$ then (SDD) is equivalent to (2.2) for $a = \max\{0, b^T \bar{y}\}$. Furthermore, if (SDP) is feasible then all its feasible solutions X satisfy $trX = a$, the primal optimum is attained and is equal to the infimum of (SDD)*

We can now establish the equivalence between (D) and (2.2) as follows. We know that the dual (D) is strictly feasible from (2.1). From the above theorem (1), we know that the primal feasible set has a constant trace a i.e. $\text{Trace}(X) = a$ for all $X \in \{X \succeq 0, \mathcal{A}X = b\}$. Adding this redundant constraint to the primal SDP (P), we get the following

$$
\begin{array}{llrcl@{\qquad}llrcl}
 & \text{Maximise} & C \bullet X & & & & \text{Minimise} & ay_0 + b^T y & & \\
(P') & \text{s.t.} & \mathcal{A}X & = & b & (D') & \text{s.t.} & S & = & y_0 I + \mathcal{A}^T y - C \\
 & & I \bullet X & = & a & & & S & \succeq & 0 \\
 & & X & \succeq & 0 & & & & &
\end{array}
$$

Let $\bar{X}$ and $\bar{S}$ with an appropriate $(\bar{y_0}, \bar{y})$ be a pair of primal and dual optimal solutions. Since the dual has a strictly feasible starting point, strong duality holds for the primal dual pairs (P),(D) and (P'),(D') respectively. Then at optimality we must have $\bar{X}\bar{S} = 0$. Thus $\bar{X}$ and $\bar{S}$ are simultaneously diagonalisable, i.e. there is an orthonormal matrix $P \in \mathcal{R}^{n \times n}$ such that $\bar{X} = P\Lambda_{\bar{X}}P^T$ and $\bar{S} = P\Lambda_{\bar{S}}P^T$, where $\Lambda_{\bar{X}}$ and $\Lambda_{\bar{S}}$ are diagonal matrices with nonegative entries. Thus we have $\Lambda_{\bar{X}}\Lambda_{\bar{S}} = 0$. Since $\mathrm{tr}(\bar{X}) = a$, we note that $\bar{X} \neq 0$. This implies that at least one entry of $\Lambda_{\bar{S}}$ must be zero. This in turn implies that $\bar{S}$ is singular. Now since $S \succeq 0$, all dual optimal solutions S satisfy $\lambda_{max}(-S) = -\lambda_{min}(S) = 0$. This leads to

$$y_0 \quad = \quad \lambda_{max}(C - \mathcal{A}^T y)$$

If we plug in this expression for y_0 in (D') we get the eigenvalue optimisation problem (2.2). Another way to look at this, is that we are rewriting the constraint $S \succeq 0$ as $-\lambda_{min}(S) \leq 0$, and lifting this into the objective function by means of a Lagrange multiplier $a > 0$. Thus we have shown that (D) is equivalent to (2.2). Without any loss of generality, we will assume henceforth that $a = 1$.

A large class of semidefinite programs and, in particular, several relaxations of combinatorial optimisation problems such as max cut, Lovasz theta, box constrained QP's satisfy this requirement, and can be formulated in this way.

2.3 The maximum eigenvalue function

In this section we review some of the properties of the maximum eigenvalue optimisation. An excellent reference is Lewis and Overton's paper on eigenvalue optimisation [38]. The maximum eigenvalue function can be cast as the following

semidefinite program. This follows from the Rayleigh Ritz variational characterisation of the maximum eigenvalue of a symmetric matrix.

$$\lambda_{max}(X) \quad = \quad \max_{W:W\succeq 0,\mathrm{trace}(W)=1} X \bullet W \tag{2.3}$$

This characterisation of $\lambda_{max}(.)$ as the maximum over a family of linear functions implies that $\lambda_{max}(.)$ is convex. The subgradients of $\lambda_{max}(.)$ at X, are the matrices W satisfying the subgradient inequality (2.4).

$$\lambda_{max}(Y) \geq \lambda_{max}(X) + W \bullet (Y - X) \ , \forall Y \in \mathcal{S}^n \tag{2.4}$$

$$\begin{array}{rcl} W & = & \{W : W \succeq 0, \mathrm{trace}(W) = 1, X \bullet W = \lambda_{max}(X)\} \\ & = & \{PVP^T : \mathrm{trace}(V) = 1, V \succeq 0\} \end{array} \tag{2.5}$$

Here P is an orthonormal matrix, whose columns are eigenvectors of the maximum eigenvalue of X. Thus any eigenvector to the maximum eigenvalue of X gives rise to a subgradient at X. The subdifferential of λ_{max} at X i.e. the set of subgradients at X is precisely the set (2.5). The second expression for the subdifferential follows from the fact $\{PVP^T : \mathrm{trace}(V) = 1, V \succeq 0\}$ is the convex hull of all rank one matrices pp^T where p is a normalised eigenvector to $\lambda_{max}(C - \mathcal{A}^T y)$.

This variational characterisation of the maximum eigenvalue enables us to rewrite (2.2) as the following SDP

$$\begin{array}{rcl} f(y) & = & \max_{W\in\mathcal{W}}(C - \mathcal{A}^T y) \bullet W + b^T y \\ & = & \max_{W\in\mathcal{W}} C \bullet W + (b - \mathcal{A}W)^T y \end{array} \tag{2.6}$$

The subdifferential of f at y is the set

$$\partial f(y) \;=\; \{b - \mathcal{A}(W) : W \in \mathcal{W}, (C - \mathcal{A}^T y) \bullet W = \lambda_{max}(C - \mathcal{A}^T y)\} \qquad (2.7)$$

Thus a point y_* is a minimiser of the convex function f if and only if $0 \in \partial f(y_*)$ or equivalently

$$\begin{array}{rcll} y_* \in \text{Argmin} f & \Leftrightarrow & \exists W_* \in \mathcal{W} : \mathcal{A}(W) = b & \text{and} \\ & & (C - \mathcal{A}^T y_*) \bullet W_* = \lambda_{max}(C - \mathcal{A}^T y_*) & \end{array} \qquad (2.8)$$

2.4 The Proximal Bundle Method

The spectral method is a specialisation of the proximal bundle method introduced by Kiwiel [33] to the eigenvalue optimisation problems. The maximum eigenvalue function $\lambda_{max}(.)$ is a nonsmooth convex function. A general scheme to minimise such a function is the bundle method; see Lemarechal's article in [45], [33] and [51], and the books [25], [26]. The spectral bundle method specialises the proximal bundle method to eigenvalue optimisation problems, and employs a semidefinite nonpolyhedral cutting plane model; it is particularly well suited for large scale problems because of its aggregration possibilities. We start with an intuitive explanation of the general principle of this method

Assume we want to minimise a convex nondifferentiable function f. We assume that we have an oracle i.e. a subroutine that given an input point $\bar{y}$ computes the objective value $f(\bar{y})$ and a subgradient $\bar{g} \in \partial f(\bar{y})$. By definition subgradients satisfy the subgradient inequality

$$f(y) \;\geq\; f(\bar{y}) + \bar{g}^T(y - \bar{y}) \quad \forall y$$

We construct a cutting plane model for $\hat{f}$ for f minorising f as follows

Let $y^1, \ldots, y^k$ denote a set of points, for which we have used the oracle to compute $f(y^i)$ and g^i , $i = 1, \ldots, k$. A possible cutting plane model is then

$$\begin{aligned} \hat{f}^k &= \max_{i=1,\ldots,k} f(y^i) + (g^i)^T(y - y_i) \qquad \text{or} \\ \hat{f}^k &\geq f(y^i) + (g^i)^T(y - y_i) \qquad \forall i = 1, \ldots, k \end{aligned}$$

Note that the linear approximations $f(y^i) + (g^i)^T(y - y^i)$ of f will be of reasonable quality only in the vicinity of y^i. Therefore to compute the new iterate y^{k+1}, we concentrate on a neighbourhood around the last successful iterate $\hat{y}^k$ and determine the new trial point y^{k+1} as the minimiser of

$$f^k(y) = \hat{f}^k(y) + \tfrac{u}{2}||y - y^k||^2$$

The quadratic term ensures that the minimum of the *augmented model* is both finite and unique. The weight $u > 0$.

If the function value $f(y^{k+1})$ at the new iterate y^{k+1} shows reasonable progress in comparison to the decrease predicted by the model value $\hat{f}^k(y^{k+1})$ i.e. for some parameter $\kappa \in (0, 1)$ we have

$$f(y^k) - f(y^{k+1}) \geq \kappa[f(\hat{y}^k) - \hat{f}^k(y^{k+1})]$$

We then take a *serious* or *descent* step by setting the new iterate $\hat{y}^{k+1} = y^{k+1}$. Else we retain the old iterate i.e. $\hat{y}^{k+1} = \hat{y}^k$. We call such a step a *null step*. Note that although we have not changed y, we have updated the model with valuable subgradient information, since we have now have $\hat{f}^k(y) \geq f(y^{k+1}) + (g^{k+1})^T(y - y^{k+1})$ in our cutting plane model.

These steps are repeated until the decrease predicted by the cutting plane model is small relative to the function value.

2.5 The proximal bundle idea applied to the spectral bundle method

We are now ready to describe the basic ingredients of the spectral bundle method, based on the proximal bundle method introduced in (2.4).

1. The function f we wish to minimise is

$$f(y) \;=\; \lambda_{max}(C - \mathcal{A}^T y) + b^T y \;=\; \max_{W \in \mathcal{W}}(C - \mathcal{A}^T y) \bullet W + b^t y$$

Here $\mathcal{W} = \{W : W \succeq 0, \mathrm{tr}(W) = 1\}$.

2. The subgradient g^k at iteration k is an eigenvector correponding to $\lambda_{max}(C - \mathcal{A}^T y^k)$ duly orthonormalised as explained later.

3. Our cutting plane model at iteration k has the following form

$$\begin{aligned} \hat{\mathcal{W}}^k &= \{P_k V P_k^T + \alpha \bar{W}_k : \mathrm{tr}(V) + \alpha = 1, V \in \mathcal{S}_+^{r_k}, \alpha \geq 0\} \\ f_{\hat{\mathcal{W}}^k}(y) &= max_{W \in \hat{\mathcal{W}}^k}(C - \mathcal{A}^T y) \bullet W + b^T y \end{aligned}$$

This is similar to (2.4) since we have $f(y) = f_{\mathcal{W}}(y) >= f_{\hat{\mathcal{W}}^k}(y)$ with equality at $y^i, i = 1, \ldots, k$. Note that we are maximising over a subset $\hat{\mathcal{W}}^k$ of $\mathcal{W}$. The following lemma gives us an expression for $f_{\hat{\mathcal{W}}^k}(y)$.

Theorem 2

$$\begin{aligned} f_{\hat{\mathcal{W}}^k}(y) &= \max\{\lambda_{max}((P^k)^T(C - \mathcal{A}^T y)P^k, (C - \mathcal{A}^T y) \bullet \bar{W}_k\} + b^T y \\ &\leq \qquad\qquad f(y) \end{aligned} \tag{2.9}$$

Proof: The right hand inequality is easily obtained since we are now maximising over a subset $\hat{\mathcal{W}}^k$ of $\mathcal{W}$. To prove the left hand side equality, we observe that $\hat{\mathcal{W}}^k$ is the convex hull of $\bar{W}^k$ and the set $T = \{P^k V (P^k)^T : \mathrm{tr}V = 1, V \succeq 0\}$. Now since we are maximising a linear function, over this convex set, we can consider these two sets separately and take the maximum of the results. Now $f_{\bar{W}^k}(y) = (C - \mathcal{A}^T y) \bullet \bar{W}^k$ and

$$\begin{aligned} \max_{W \in T} f_W(y) &= \max_{V \succeq 0, \mathrm{tr}V = 1}(C - \mathcal{A}^T y) \bullet P^k V (P^k)^T + b^t y \\ &= \max_{V \succeq 0, \mathrm{tr}V = 1}(P^k)^T (C - \mathcal{A}^T y) P^k \bullet V + b^T y \\ &= \qquad \lambda_{max}((P^k)^T (C - \mathcal{A}^t y) P^k) + b^T y \end{aligned} \tag{2.10}$$

The 2nd equation follows from the fact that $\mathrm{tr}(AB) = \mathrm{tr}(BA)$ and the last equation follows from the variational characterisation of the maximum eigenvalue function. □

This theorem is extremely important since it implies that we are solving (2.9) in every iteration in lieu of (2.2). Here P^k constitutes the bundle and $P^k \in R^{n\times r}$. Note that $(P^k)^T(C - \mathcal{A}^T y)P^k \in \mathcal{S}^r$. By keeping the number of columns r in P small, the resulting SDP can be solved quickly. Helmberg and Rendl [22] were able to show that $\binom{r+1}{2} \leq m$ i.e. the dimension of subspace P is roughly bounded by the square root of the number of constraints m. This is the *main idea* behind the spectral bundle approach.

4. The augmented model is expressed in terms of the following function

$$L^k(y, W) \;=\; f_{\mathcal{W}}(y) + \tfrac{u}{2}||y - \hat{y}^k||^2 \;=\; (C - \mathcal{A}^T y) \bullet W + b^T y + \tfrac{u}{2}||y - \hat{y}^k||^2$$

and

$$f^k(y) \;=\; \max_{W\in\hat{\mathcal{W}}^k} L^k(y, W)$$

Since our aim is to minimise $f^k(y)$, we need to solve

$$\min_y f^k(y) \;=\; \min_y \max_{W\in\hat{\mathcal{W}}^k} L^k(y, W) \tag{2.11}$$

The dual problem to this is

$$\max_{W\in\hat{\mathcal{W}}^k} \min_y L^k(y, W) \tag{2.12}$$

In the dual (2.12), the inner minimisation over y is unconstrained. Thus for a fixed $W \in \hat{\mathcal{W}}^k$, the corresponding optimal y can be determined explicitly.

$$y^k_{min}(W) \;=\; \hat{y}^k + \tfrac{1}{u}(\mathcal{A}(W) - b) \tag{2.13}$$

We note that strong duality holds for this primal dual pair.

Theorem 3 *We have*

$$\min_y \max_{W \in \hat{\mathcal{W}}^k} L^k(y, W) \;=\; L^k(y^{k+1}, W^{k+1}) \;=\; \max_{W \in \hat{\mathcal{W}}^k} \min_y L^k(y, W)$$

with y^{k+1} unique and W^{k+1} an optimal solution of

$$\begin{array}{lccc} \min & \frac{1}{2u}||b - \mathcal{A}W||^2 - (C - \mathcal{A}^T\hat{y}^k) - b^T\hat{y}^k & & \\ s.t. & W & = & P_k V P_k^T + \alpha \bar{W}_k \quad (QSP) \\ & trV + \alpha & = & 1 \\ & V & \succeq & 0 \\ & \alpha & \geq & 0 \end{array}$$

Proof: The first part follows from the *saddle point* strong duality for convex functions. The primal problem results from substituting the expression for $y^K_{min}(W)$ in (2.13) in (2.12). We then switch the sign of the cost function in the primal to obtain (QSP). The uniqueness of y^{k+1} follows from the strict convexity of

$$f^k(y) \;=\; f_{\hat{\mathcal{W}}^k}(y) + \tfrac{u}{2}||y - \bar{y}^k||^2$$

□

We will have more to say about the quadratic subproblem in section 2.7.

The proof that the spectral bundle method converges is quite involved. We refrain from presenting this proof, but only list the two decisive conditions that have to be met in each iteration

Consider iteration k. We require the following

$$W^{k+1} \in \hat{\mathcal{W}}^{k+1} \quad \text{and} v^{k+1}(v^{k+1})^T \in \hat{\mathcal{W}}^{k+1} \tag{2.14}$$

Here $W^{k+1} = \alpha_* \bar{W}^k + P^k V^* (P^k)^T$, $\hat{\mathcal{W}}^{k+1} = \{\alpha \bar{W}^{k+1} + P^{k+1} V (P^{k+1})^T : \alpha + \mathrm{tr} V = 1, \alpha \geq 0, V \succeq 0\}$ is our current cutting plane model and v^{k+1} is a normalised eigenvector corresponding to $\lambda_{max}(C - \mathcal{A}^T y^{k+1})$.

2.6 One iteration of the algorithm

We now give a formal description of a general iteration k of the bundle algorithm.

Our current iterate is $\hat{y}^k$. The minorant of f in iteration k is denoted by $\hat{f}^k$.

$$\hat{f}^k(y) \quad = \quad \max_{W \in \hat{\mathcal{W}}^k} L^k(y, W)$$

Here $\hat{\mathcal{W}}^k = \{\alpha \bar{W}^k + P^k V (P^k)^T : \alpha + \mathrm{tr} V = 1, \alpha \geq 0, V \succeq 0\}$, is the current approximation to the set of all semidefinite matrices of trace one. In the spirit of the bundle method we also introduce the regularised version of $\hat{f}^k$ denoted as f^k.

$$f^k(y) \quad = \quad \hat{f}^k(y) + \tfrac{u}{2} ||y - \hat{y}^k||^2$$

Firstly we solve (QSDP), yielding a maximiser $W^{k+1} = \alpha_* \bar{W}^k + P^k V^* (P^k)^T$. Here α_* and V^* are the solutions, obtained by solving (QSDP). We then compute

$$y^{k+1} \quad = \quad \hat{y}^k + \tfrac{1}{u}(\mathcal{A} W^{k+1} - b)$$

We then update P and $\bar{W}$ as follows

If P^k does not yet use the maximum number of columns, simply orthogonalise the eigenvector v^{k+1} corresponding to $\lambda_{max}(C - \mathcal{A}^T y^{k+1})$ with respect to the columns in P^k and add it as a new column to get P^{k+1}. However if P^k already uses the maximum number of columns, the update process is a little involved and is as follows

Consider the eigenvalue decomposition $Q \Lambda Q^T$ of V^*. Then the important part of the spectrum of W^{k+1} is spanned by the eigenvectors associated with the *large* eigenvalues of V^*. Thus we partition the eigenvectors Q of V^* into two parts $Q = [Q_1 Q_2]$. Here Q_1 contains the eigenvectors associated with the $r - 1$ largest

eigenvalues of V^*, and Q_2 contains the remaining columns.

The new bundle P^{k+1} is computed to contain P^kQ_1 and at least one eigenvector v^{k+1} corresponding to the maximal eigenvalue of $C - \mathcal{A}^T y^{k+1}$.

$$P^{k+1} \quad = \quad \mathrm{orth}([P^kQ_1, v^{k+1}]) \tag{2.15}$$

Here orth indicates that we take an orthonormal basis of $[P^kQ_1, v^{k+1}]$.

We construct the new aggregrate matrix $\bar{W}^{k+1}$ such that $W^{k+1} \in \hat{\mathcal{W}}^{k+1}$. Since the important part of P^k given by P^kQ_1, we include the remaining P^kQ_2 in $\bar{W}^{k+1}$.

$$\bar{W}^{k+1} \quad = \quad \tfrac{1}{\alpha_* + \mathrm{tr}\Lambda_2}(\alpha^* \bar{W}^k + P^kQ_2\Lambda_2(P^kQ_2)^T) \tag{2.16}$$

Theorem 4 *The update formulas (2.15) and (2.16) ensure that (2.14) are satisfied.*

Proof: Since P^{k+1} is given by (2.15), we find that the columns of P^{k+1} form an orthonormal basis for the space spanned by P^kQ_1 and v^{k+1}. Therefore we write v^{k+1} as $v^{k+1} = P^{k+1}q$, where q is any normalised vector. Then $v^{k+1}(v^{k+1})^T = P^{k+1}qq^T(P^{k+1})^T$. Thus we have shown that $v^{k+1}(v^{k+1})^T \in \hat{\mathcal{W}}^k$.

To show that $W^{k+1} \in \hat{\mathcal{W}}^{k+1}$, we proceed as follows

There is an orthonormal matrix $\bar{Q}$ with $P^kQ_1 = P^{k+1}\bar{Q}$. Let α_* and V^* be optimal solutions to the quadratic semidefinite subproblem (QSP) (3).

Also define $\bar{\alpha} = \mathrm{tr}\Lambda_2 + \alpha_*$ and $\bar{V} = \bar{Q}\Lambda_1\bar{Q}^T$.

$$\begin{array}{rcc} W^{k+1} & = & \alpha_*\bar{W}^k + P^kV_*(P_k)^T \\ & = & \alpha_*\bar{W}^k + (P^kQ_1)\Lambda_1(P^kQ_1)^T + (P^kQ_2)\Lambda_2(P^kQ_2)^T \\ & = & \bar{\alpha}\bar{W}^{k+1} + P^{k+1}\bar{V}(P^{k+1})^T \\ & \in & \hat{\mathcal{W}}^{k+1} \end{array}$$

Also $\bar{\alpha} + \mathrm{tr}\bar{V} = \alpha_* + \mathrm{tr}\Lambda_1 + \mathrm{tr}\Lambda_2 = 1$, by the feasibility of (V_*, α_*) for (QSP).

□

2.7 The quadratic semidefinite subproblem (QSP)

The quadratic semidefinite subproblem (3) can also be rewritten as (2.7). This amounts to dropping all the subscripts, and essentially substituting $W = PVP^T + \alpha\bar{W}$ in (3).

$$\begin{array}{lrcl} \min & \frac{1}{2}\begin{bmatrix} \mathrm{svec}V \\ \alpha \end{bmatrix}^T \begin{bmatrix} Q_{11} & q_{12} \\ q_{12}^T & q_{22} \end{bmatrix} \begin{bmatrix} \mathrm{svec}V \\ \alpha \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}^T \begin{bmatrix} \mathrm{svec}V \\ \alpha \end{bmatrix} + d & & \\ \text{s.t.} & \alpha + s_I^T \mathrm{svec}V & = & 1 \\ & V & \succeq & 0 \\ & \alpha & \geq & 0 \end{array}$$

where

$$\begin{array}{rcc} Q_{11} & = & \frac{1}{u}\sum_{i=1}^m \mathrm{svec}(P^T A_i P)\mathrm{svec}(P^T A_i P)^T \\ q_{12} & = & \frac{1}{u}\mathrm{svec}(P^T \mathcal{A}^T(\mathcal{A}\bar{W})P) \\ q_{22} & = & \frac{1}{u}(\mathcal{A}\bar{W})^T \mathcal{A}\bar{W} \\ c_1 & = & -\mathrm{svec}(P^T[\mathcal{A}^T(\frac{1}{u}b - \hat{y}) + C]P) \\ c_2 & = & -(C \bullet \bar{W} + (\frac{1}{u}b - \hat{y})^T \mathcal{A}\bar{W}) \\ d & = & b^T(\frac{1}{2u}b - \hat{y}) \\ s_I & = & \mathrm{svec}I_r \end{array} \tag{2.17}$$

Several remarks are now in order

1. This is a problem in $\binom{r+1}{2} + 1$ variables. Here r is the bundle size i.e. the number of columns in the P matrix. Note that $r \ll n$, where n is the size of the original SDP. The quadratic subproblem can be solved efficiently using interior point methods, provided r is not too large, say, around 30.

2. We are always dealing with the vector $\mathcal{A}\bar{W}$ and the scalar $C \bullet \bar{W}$. We do not need to explicitly form the matrix $\bar{W}$ and this saves us a lot of memory and time. Note that $W \in \mathcal{S}^n$.

3. We are dealing with the projected constraint matrices $P^T A_i P \in \mathcal{S}^r$. These

are much smaller in size than the actual constraint matrices $A_i \in \mathcal{S}^n$.

4. The most expensive operation in computing the cost coefficients is forming Q_{11}. This requires $O(mr^4)$ arithmetic operations in all.

2.8 The Spectral Bundle algorithm

Input

1. An initial point $y^0 \in R^m$.
2. A normalised eigenvector v^0 for the maximal eigenvalue of $C - \mathcal{A}^T y^0$.
3. A termination parameter $\epsilon > 0$.
4. An improvement parameter $m_L \in (0, \frac{1}{2})$.
5. A weight $u > 0$.
6. An upper bound r on the number of columns of P.
7. The maximum number of iterations *MAXITER*.

Algorithm

1. **Initialisation** Set $k = 0$, $\hat{y}^0 = y^0$, $P^0 = v^0$, $\bar{W}^0 = v^0(v^0)^T$.
2. **for** $k = 1 :$ MAXITER **do**
   begin
3. **Direction Finding** Solve the QSP (3) to get V^* and α^*.
4. **Evaluation** Compute the following
   (a) $W^{k+1} = \alpha \bar{W}^k + P^k V_* (P^k)^T$.
   (b) $y^{k+1} = \hat{y}^k + \frac{1}{u}(\mathcal{A}W^{k+1} - b)$.
   (c) Decompose $V^* = Q_1 \Lambda_1 Q_1^T + Q_2 \Lambda_2 Q_2^T$ with $\text{rank}(Q_1) \leq r - 1$.

(d) $\bar{W}^{k+1} = \frac{1}{\alpha^* + \mathrm{tr}\Lambda_2}(\alpha^* \bar{W}^k + P^k Q_2 \Lambda_2 (P^k Q_2)^T)$.

(e) $\lambda_{max}(C - \mathcal{A}^t y^{k+1})$ and an eigenvector v^{k+1}.

(f) $P^{k+1} = \mathrm{orth}([P^k Q_1, v^{k+1}])$.

5. **Termination** If $f(\hat{y}^k) - \hat{f}^k(y^{k+1}) \leq \epsilon$ then **stop**.

6. **Serious Step** If

$$f(y^{k+1}) \quad \leq \quad f(\hat{y}^k) - m_L(f(\hat{y}^k) - \hat{f}^k(y^{k+1})$$

then set $\hat{y}^{k+1} = y^{k+1}$. Go to step 7. Else goto step 6.

7. **Null Step** Set $\hat{y}^{k+1} = \hat{y}^k$.

end

2.9 Salient features of the bundle method

1. The bundle method recasts the semidefinite programming problem (D) as an eigenvalue optimisation problem (2.2). To do this, we need to ensure the dual has a strictly feasible point i.e. satisfies (2.1). This is also equivalent to assuming that the primal feasible solutions X to (P) have a constant trace a i.e. $\mathrm{tr}X = a$, $\forall X \in \{X \succeq 0 : \mathcal{A}X = b\}$. A large number of semidefinite programs satisfy this requirement, in particular several important relaxations of combinatorial optimisation problems.

2. The maximum eigenvalue function $\lambda_{max}(C - \mathcal{A}^T y)$ is a nonsmooth convex function. It can be solved using the traditional bundle approach, with the subgradient information provided by the eigenvectors corresponding to the maximum eigenvalue. A cutting plane model is formed using the collected subgradients, augmented by a regularisation term, which imposes a penalty for the displacement from the current iterate.

3. The maximum eigenvalue is approximated by means of the vectors in the subspace spanned by the bundle P. This amounts to solving the following

problem in lieu of (2.2).

$$\min_y \quad a\lambda_{max}(P^T(C - \mathcal{A}^T y)P) \quad + \quad b^T y$$

Note that (3) requires that the dual slack matrix $S = C - \mathcal{A}^t y$ be positive semidefinite only with respect to the subspace spanned by the bundle vectors i.e. the columns of P. Therefore the resulting solution to (3) is typically an indefinite matrix S. The negative eigenvalues of this S and their correspinding eigenvectors are used to update the cutting plane model, and the process is iterated. By keeping the number of columns r in P small, the resulting SDP (3), with a concave quadratic cost function can be solved quickly. The number of columns r in P is roughly bounded by the square root of the number of constraints.

4. Helmberg and Rendl [22] also introduce the idea of a *aggregrate* subgradient, whereby the spectral bundle approach converges even for restricted bundle sizes i.e. for small r. The idea here is to keep all the important cutting planes in the bundle, and aggregrate the least important ones. In the extreme case the bundle P consists of a new eigenvector corresponding to the maximal eigenvalue alone.

5. The extremal eigenvalues and their corresponding eigenvectors are computed using an iterative scheme such as the *Lanczos* method.

6. Since we deal with restricted bundle sizes, so that the problem (3) can be solved quickly, the spectral bundle method is a *first order method* only. A second order method which converges globally, and which enjoys asympotically a quadratic rate convergence was recently developed by Oustry [47].

7. Helmberg and Kiwiel [21] were able to extend to the spectral bundle method to problems with bounds.

CHAPTER 3
A linear programming (LP) approach to SDP

3.1 Introduction

In chapter 1 we discussed the semidefinite programming (SDP), and various method to solve it. The interior point emthods offer polynomial time complexity, apply to all SDP's, and exploit second order information. However for practical applications with many constraints k, the number of arithmetic operations, per iteration is often too high. This motivates the search for other approaches, that are suitable for large k.

One large scheme is the spectral bundle method due to Helmberg and Rendl [22]. We discussed this method in detail in chapter 2. The bundle scheme recasts SDP's with a constant trace on the primal feasible set as eigenvalue optimisation problems. These are convex nonsmooth programming problems and can be solved by bundle methods. In this chapter we propose a linear programming (LP) framework to solving SDP's with this structure. Although SDP's are *semi infinite* LP's, we show that only a small number of constraints, namely those in the bundle maintained by the spectral bundle, bounded by the square root of the number of constraints in the SDP, in conjunction with families of problem specific constraints, are typically required. The resulting LP's can be solved rather quickly and provide reasonably accurate solutions. We present numerical examples demonstrating the efficiency of the approach on combinatorial examples.

This chapter presents my work with my advisor John Mitchell. Extracts of this chapter are taken from our paper Krishnan and Mitchell [30], which we have submitted to the *SIAM Journal on Optimisation*. This paper can be downloaded from *http://www.rpi.edu/∼ kartis/publications.html*.

This chapter is organised as follows. Section 2 explains why an SDP is an *semi-infinite* LP, section 3 describes a linear programming formulation for the SDP, section 4 explains how we can get a lower bound from our LP relaxation, when SDP is a minimisation problem, section 5 describes the set of linear constraints, to

be used in our LP formulation. We present our computational results in section 6, and conclude with some observations and acknowledgements in sections 7 and 8 respectively.

3.2 A semi infinite LP formulation for the SDP

Consider the semidefinite programming problem

$$\begin{array}{lrcll} \min & C \bullet X & & & \\ \text{subject to} & \mathcal{A}(X) & = & b & (SDP) \\ & X & \succeq & 0, & \end{array}$$

with dual

$$\begin{array}{lrcrcll} \max & b^T y & & & & & \\ \text{subject to} & \mathcal{A}^T y & + & S & = & C & (SDD) \\ & & & S & \succeq & 0 & \end{array}$$

where $\mathcal{A} : \mathcal{S}^n \to R^k$ and $\mathcal{A}^T : R^k \to \mathcal{S}^n$ are of the form

$$\mathcal{A}(X) \;=\; \begin{bmatrix} A_1 \bullet X \\ \vdots \\ A_k \bullet X \end{bmatrix} \quad \text{and} \quad \mathcal{A}^T y \;=\; \sum_{i=1}^k y_i A_i$$

with $A_i \in \mathcal{S}^n, i = 1, \ldots, k$. $C \in \mathcal{S}^n$ is the cost matrix, $b \in R^k$ the RHS vector.

In addition we assume that

Assumption 1

$$\mathcal{A}(X) \;=\; b \quad \textit{implies} \quad trX \;=\; a \tag{3.1}$$

for some constant $a \geq 0$.

A large class of semidefinite programs, in particular several important relaxations of combinatorial optimisation problems, can be formulated to satisfy (3.1) such as max cut, Lovasz theta, semidefinite relaxations of box constrained quadratic pr ograms etc.

The set of symmetric matrices $\mathcal{S}^n$ is isomorphic to $\mathcal{R}^{\binom{n+1}{2}}$ via the map svec(.). svec(A) of a symmetric matrix $A \in \mathcal{S}^n$ is a vector in $\mathcal{R}^{\binom{n+1}{2}}$ obtained by stacking the columns of the lower triangle of A on top of each other and multiplying the off diagonal elements with $\sqrt{2}$, i.e.

$$\text{svec}(A) \;:=\; [a_{11}, \sqrt{2}a_{21}, \ldots, \sqrt{2}a_{n1}, a_{22}, \sqrt{2}a_{32}, \ldots, a_{nn}]^T$$

The factor $\sqrt{2}$ for the off diagonal elements ensures that for $A, B \in \mathcal{S}^n$, we have

$$A \bullet B \;=\; \text{svec}(A)^T\text{svec}(B)$$

Thus we can rewrite (SDP) as the following problem

$$\begin{array}{lrcl} \min & \text{svec}(C)^T\text{svec}(X) & & \\ \text{s.t.} & A\text{svec}(X) & = & b \\ & X & \succeq & 0 \end{array} \tag{3.2}$$

Here $A \in \mathcal{R}^{m\times\binom{n+1}{2}}$ is the constraint matrix, svec(X) and svec(C) are vectors in $\mathcal{R}^{\binom{n+1}{2}}$, and $b \in \mathcal{R}^m$.

Thus SDP is a linear programming problem in $\frac{n(n+1)}{2}$ variables, except for the convex constraint $X \succeq 0$. This constraint is equivalent to

$$d^T X d \;\geq\; 0 \quad \forall d \in R^n \tag{3.3}$$

These constraints are linear inequalities $\text{tr}(dd^T X) \geq 0$ in the matrix variable, and can also be written as $\text{svec}(dd^T)^T\text{svec}(X) \geq 0$, but there are an *infinite* number of them. Thus SDP is equivalent to the semi-infinite linear programming problem

$$\begin{array}{lrcl} \min & \text{svec}(C)^T\text{svec}(X) & & \\ \text{s.t.} & A\text{svec}(X) & = & b \\ & \text{svec}(dd^T)^T\text{svec}(X) & \geq & 0 \quad \forall d \end{array} \tag{3.4}$$

In the subsequent sections, we will show that we can take a finite number of

constraints in (3.4),and yet obtain a reasonable polyhedral approximation of semidefinite cone.

3.3 A linear programming formulation

Since $d^T X d \geq 0$ can be rewritten as $\mathrm{tr}(dd^T X) \geq 0$, we need to consider only rank one matrices $M = dd^T$. We then have the following corollary.

Corollary 3 *The symmetric $n \times n$ matrix S is positive semidefinite if and only if $S \bullet M \geq 0$ for all symmetric rank one matrices M.*

It follows from Corollary 3 that (SDD) is equivalent to the following semi-infinite linear programming problem:

$$\begin{array}{lll} \max & b^T y & \\ \text{subject to} & dd^T \bullet (C - \mathcal{A}^T y) \;\geq\; 0 & \text{for all vectors } d. \qquad (LDD) \end{array}$$

We propose looking at linear programming relaxations of (LDD). Given a finite set of vectors $\{d_i, i = 1, \ldots, m\}$, we obtain the relaxation

$$\begin{array}{lll} \max & b^T y & \\ \text{subject to} & d_i d_i^T \bullet \mathcal{A}^T y \;\leq\; d_i d_i^T \bullet C & \text{for } i = 1, \ldots, m. \qquad (LDR) \end{array}$$

We are dealing with (SDD) here, since we have k variables and so we need at least k constraints for (LDR) to have an extreme point. Since (SDP) has $\frac{n(n+1)}{2}$ variables, a typical finite linear programming relaxation would need at least $O(n^2)$ inequalities to get a basic feasible solution. Since we have far fewer variables in the dual formulation, we typically get smaller linear programs. Another reason for working with (SDD) is that the spectral bundle method, discussed in Chapter 2, employs (SDD), recasting it as an eigenvalue optimisation problem (2.2).

We now derive the linear programming dual to (LDR). We have

$$d_i d_i^T \bullet \mathcal{A}^T y \;=\; d_i d_i^T \bullet (\sum_{j=1}^{k} y_j A_j)$$

$$= \sum_{j=1}^{k} y_j d_i^T A_j d_i.$$

Thus, the constraints of (LDR) can be written as

$$\sum_{j=1}^{k} y_j d_i^T A_j d_i \leq d_i^T C d_i \quad \text{for } i = 1, \ldots, m.$$

It follows that the dual problem is

$$\begin{array}{lrcl} \min & \sum_{i=1}^m d_i^T C d_i x_i & & \\ \text{subject to} & \sum_{i=1}^m d_i^T A_j d_i x_i & = & b_j \quad \text{for } j = 1, \ldots, k \\ & x & \geq & 0. \end{array}$$

This can be rewritten as

$$\begin{array}{lrcll} \min & C \bullet (\sum_{i=1}^m x_i d_i d_i^T) & & & \\ \text{subject to} & \mathcal{A}(\sum_{i=1}^m x_i d_i d_i^T) & = & b & (LPR) \\ & x & \geq & 0. & \end{array}$$

Note that any feasible solution x to (LPR) will give a feasible solution to (SDP), namely $X = \sum_{i=1}^m x_i d_i d_i^T$, since $x \geq 0$.

A good way of generating cutting planes $\sum_{j=1}^k y_j d^T A_j d \leq d^T C d$ requires searching for d satisfying $dd^T \bullet (C - \mathcal{A}^T y) = d^T C d - \sum_{j=1}^k y_j d^T A_j d < 0$. One such choice for d would be to look at the eigenvectors d corresponding to the negative eigenvalues of $C - \mathcal{A}^T y$, since these satisfy $d^T(C - \mathcal{A}^T y)d = \lambda(C - \mathcal{A}^T y) < 0$, when the d's are normalised.

In Krishnan and Mitchell [31] we develop a cutting plane framework, where we add a number of d corresponding to the most negative eigenvalues of $C - \mathcal{A}^T y$ to our finite collection of linear inequalities and solve the new LP relaxation using an interior point approach, with the initial LP's being solved rather cheaply. Continuing in this way, one can produce a sequence of LP's whose solutions converge to the optimal solution of the SDP. We present more details in chapter 4.

3.4 Getting a lower bound from our LP relaxation

When (SDP) is a minimisation problem, our LP approach gives an upper bound on the SDP objective value. In this section we discuss how we can generate a lower bound from our LP relaxation. The SDP value lies somewhere in between these two LP objective values. The approach makes use of Lemma 1.

We want a feasible solution to

$$\begin{array}{lrcll} \max & b^T y & & & \\ \text{s.t.} & S & = & C - \mathcal{A}^T y & (SDD) \\ & S & \succeq & 0 & \end{array}$$

Our linear programming approach gives a candidate S and y, satisfying the linear constraint but with S not psd.

A procedure to make S psd, generating $b^T y - \bar{\lambda} b^T \bar{y}$, which serves as the lower bound is as follows

1. Solve the LP
$$\begin{array}{lrcl} \min & b^T y & & \\ \text{s.t.} & \mathcal{A}^T y & = & I \end{array}$$
Let $\bar{y}$ be the optimal solution.

2. Find the most negative eigenvalue of S. Let $\bar{\lambda}$ denote the absolute value of this eigenvalue.

3. Change y to $y - \bar{\lambda}\bar{y}$, which changes S to $S + \bar{\lambda} I$, a psd matrix. The objective function value is now $b^T y - \bar{\lambda} b^T \bar{y}$, a lower bound on the optimal value of (SDP).

3.5 A set of linear constraints

A set of linear constraints for (LDR) can be derived from the bundle information used by the spectral bundle method.

At iteration i of the spectral bundle method, we work with the following approximation $\phi_i(y)$ of the convex function $f(y) = a\lambda_{max}(\mathcal{A}^T y - C) + b^T y$

$$\phi_i(y) \;=\; \max_{W \in \bar{\mathcal{W}}_i} (\mathcal{A}^T y - C) \bullet W$$

where $\bar{\mathcal{W}}_i$ is

$$\bar{\mathcal{W}}_i \quad = \quad \{P_i V P_i^T \quad + \quad \alpha \bar{W}_i \quad : \quad \mathrm{tr} V + \alpha = a, V \in \mathcal{S}_+^{r_i}, \alpha \geq 0\}$$

We refer to P_i as the *bundle*, the number of columns r_i of P_i as the *size* of the bundle and to $\bar{W}_i$ as the *aggregrate*.

Here P_i is of size $n \times r_i$ for some value r_i bounded above by $\sqrt{k}$, where k is the number of constraints in the SDP. We denote the columns of P_i as p_j , $j = 1, \ldots, r_i$. These columns represent a good subspace to approximate the maximum eigenvalue of $\mathcal{A}^T y_i - C$. $\bar{W}_i$ serves as the aggregrate and contains the less important subgradient information. This helps to keep the number of columns in P_i small, so that the SDP 3.5 can be solved quickly.

When the stopping criteria is met $W^+ := PV^+P^T + \alpha^+ \bar{W}$ serves as the approximation to the primal matrix X, where V^+ and $\alpha+$ are the final values of V and α.

We propose using the columns of P as the vectors $\{d_j\}$, $\forall j = 1, \ldots, r$ in (LDR). Since the number of vectors r in the bundle P, is $O(\sqrt{k})$, and we need k constraints to guarantee a basic feasible solution with k variables in the problem, we need to look for other constraints as well. These other constraints, labelled as *box* constraints are problem specific as we shall see in section 5. They are polynomial in the problem size n. The rationale for using the columns of the P matrix is as follows.

At iteration k we add vector p_k, a normalised eigenvector corresponding to $\lambda_{max}(\mathcal{A}^T y_k - C)$, where y_k is the current estimate of y. p_k is duly orthogonalised with the remaining vectors in the bundle, so as to keep P an orthonormal matrix. These eigenvectors correspond to subgradients of $\lambda_{max}(\mathcal{A}^T y - C)$. Now $\lambda_{max}(\mathcal{A}^T y - C) = -\lambda_{min}(C - \mathcal{A}^t y)$, since the maximum eigenvalue is a convex function. Therefore the vectors in the bundle p are orthogonalised versions of the eigenvectors corresponding to the most negative eigenvalues of $(C - \mathcal{A}^T y)$, usually satisfying $pp^T \bullet (C - \mathcal{A}^T y) \leq 0$. They are excellent candidates for the cutting planes d described in section 2.

Another chain of reasoning is as follows.

When the bundle method terminates, $W = PVP^T + \alpha \bar{W}$ provides an approx-

imation for the primal matrix X. If we disregard the aggregate $\bar{W}$, since its sole purpose is to keep the number of columns r in P small, we have $X = W = PVP^T$. Note that V here is a diagonal matrix because of the way in which the new bundle P and new aggregrate $\bar{W}$ are constructed. Therefore PVP^T is a spectral decomposition of the primal matrix X, with V containing the eigenvalues of X, and P containing the corresponding eigenvectors. Since $V \in \mathcal{S}^r_+$, this amounts to saying that X is of rank $\leq r$, and the remaining eigenvalues of X are zero. Now from strong duality we have $XS = 0$ at optimality. Since X and $S \in \mathcal{S}^n$, the columns P lie in the null space of S. These should be especially useful, in eventually ensuring that S is psd.

We offer empirical evidence for this choice of d in section 6.

The drawback of using using the columns of columns of P as vectors d in (LDR), is these d are dense, leading to a dense linear programming problem. We try to compensate for these dense constraints, by choosing d for our box constraints, that are sparse. For more details refer to section 5.

We illustrate the procedure on the max cut problem (section 3.5.1), min bisection (section 3.5.2), **k** equipartition problem (section 3.5.3), Lovasz theta (section 3.5.4), and box constrained QP's (section 3.5.5).

3.5.1 The Max Cut problem

The semidefinite programming relaxation of the max cut problem was proposed by Goemans and Williamson [15] and is

$$\begin{array}{lrcl} \max & \frac{L}{4} \bullet X & & \\ \text{subject to} & \operatorname{diag}(X) & = & e \\ & X & \succeq & 0, \end{array} \tag{3.5}$$

with dual

$$\begin{array}{lrcrcl} \min & e^T y & & & & \\ \text{subject to} & -\operatorname{Diag}(y) & + & S & = & -\frac{L}{4} \\ & & & S & \succeq & 0 \end{array} \tag{3.6}$$

Here $L = \text{Diag}(Ae) - A$ is the Laplacian matrix of the graph, where A is the weighted adjacency matrix. Note that the a in $\text{tr}X = a$ is trivially n, the number of nodes in the graph.

Since S is psd, we have $d^T S d = d^T(\text{Diag}(y) - \frac{L}{4})d \geq 0$, $\forall d$. In particular we propose to use the following d for the max cut problem.

MC1 Setting $d = e_i$, $\forall i = 1 \ldots n$, where e_i is the ith standard basis vector for R^n. In particular e_i has a one in the ith position and zeros elsewhere. This generates the constraint $y_i \geq \frac{L_{ii}}{4}$, $i = 1 \ldots n$.

MC2 Setting $d = (e_i + e_j)$ and $(e_i - e_j)$, $\forall \{i, j\} \in E$, gives rise to the constraints $y_i + y_j \geq \frac{Lii}{4} + \frac{Ljj}{4} + \frac{L_{ij}}{2}$ and $y_i + y_j \geq \frac{Lii}{4} + \frac{Ljj}{4} - \frac{Lij}{2}$ respectively. Together these give $y_i + y_j \geq \frac{Lii}{4} + \frac{L_{jj}}{4} + |\frac{Lij}{2}|$.

MC3 The constraints in the bundle namely the columns p_i , $i = 1, \ldots, r$ of the matrix P.

We consider two LP relaxations, $LP1$ containing MC1 and MC3 and $LP2$ containing MC2 in addition to the constraints in $LP1$. Both these relaxations are discussed below.

$LP1$ is

$$\begin{array}{lrcll} \min & e^T y & & & \\ \text{subject to} & y_i & \geq & \frac{L_{ii}}{4} & \forall i = 1, \ldots, n \\ & \sum_{i=1}^n p_{ji}^2 y_i & \geq & p_j^T \frac{L}{4} p_j & \forall j = 1, \ldots, r \end{array} \tag{3.7}$$

with dual

$$\begin{array}{lll} \max & \sum_{i=1}^{n} \frac{L_{ii}}{4} x_i + \sum_{j=1}^{r} w_j \frac{p_j^T L p_j}{4} & \\ \text{subject to} & \begin{bmatrix} 1 & & \uparrow & & \uparrow \\ & \ddots & p_1^2 & \dots & p_r^2 \\ & & 1 \quad \downarrow & & \downarrow \end{bmatrix} \begin{bmatrix} x \\ w \end{bmatrix} & = \quad e \\ & \begin{bmatrix} x \\ w \end{bmatrix} & \geq \quad 0 \end{array} \tag{3.8}$$

Here p_{ji} refers to the jth component of vector p_i and p_i^2, $i = 1, \ldots, r$ are vectors obtained by squaring all the components of p_i , $i = 1, \ldots, r$. (3.7) has $n + r$ constraints in all. Note that $x \in R^n$ and $w \in R^r$ are the dual variables corresponding to $y \geq \text{diag}\frac{L}{4}$ and the bundle constraints respectively. To get a solution X to SDP, set $X = \text{Diag}(x) + \sum_{j=1}^{r} w_j p_j p_j^T$. This matrix X is positive semidefinite since $x \geq 0$ and $w \geq 0$. Morever

$$\frac{L}{4} \bullet X \quad = \quad \frac{L}{4} \bullet (\text{Diag}(x) + \sum_{j=1}^{r} w_j p_j p_j^T) \quad = \quad \sum_{i=1}^{n} \frac{L_{ii}}{4} x_i + \sum_{j=1}^{r} w_j \frac{p_j^T L p_j}{4}$$

This is precisely the objective value in (3.8). We have thus generated the X which could be used in the Goemans and Williamson rounding procedure [15] to generate an approximate solution to the max cut problem.

$LP2$ is

$$\begin{array}{lrcll} \min & e^T y & & & \\ \text{subject to} & y_i & \geq & \frac{L_{ii}}{4} & \forall i = 1, \ldots, n \\ & y_i + y_j & \geq & \frac{Lii}{4} + \frac{L_{jj}}{4} + |\frac{Lij}{2}| & \forall \{i, j\} \in E \\ & \sum_{i=1}^{n} p_{ji}^2 y_i & \geq & p_j^T \frac{L}{4} p_j & \forall j = 1, \ldots, r \end{array} \tag{3.9}$$

$LP2$ (3.9) has $m + n + r$ constraints in all. To generate a solution X to SDP, set $X = \text{Diag}(x) + \sum_{j=1}^{m} w_j d_j d_j^T + \sum_{j=1}^{r} w_{m+j} p_j p_j^T$. Here d_i , $i = 1, \ldots, m$,

corresponding to the edge constraints, are either $d = e_i + e_j$ or $e_i - e_j$ depending on the sign of L_{ij}. It is easy to verify that this X is psd and achieves the dual objective value.

3.5.2 The Min Bisection problem

The semidefinite programming relaxation for the min bisection problem was proposed by Frieze and Jerrum [14] and Ye [60] and is

$$\begin{array}{lrcl} \min & \frac{L}{4} \bullet X & & \\ \text{subject to} & \text{diag}(X) & = & e \\ & ee^T \bullet X & = & 0 \\ & X & \succeq & 0, \end{array} \tag{3.10}$$

with dual

$$\begin{array}{lccccccc} \min & e^T y & & & & & & \\ \text{subject to} & -y_0(ee^T) & - & \text{Diag}(y) & + & S & = & \frac{L}{4} \\ & & & & & S & \succeq & 0 \end{array} \tag{3.11}$$

Note that Frieze and Jerrum [14] had the equipartition constraint as $ee^T \bullet X \leq 0$. But since the optimal X is psd, we must have $e^T X e = ee^T \bullet X \geq 0$ at optimality, which is equivalent to $ee^T \bullet X = 0$.

Here L refers to the Laplacian matrix of the graph. y_0 is the dual variable corresponding to the constraint $ee^T \bullet X = 0$. To get the signs right, we need to take the negative of the objective value of (SDD) to get the optimal solution to the min bisection problem. Again $a = n$.

Since y_0 does not appear in the dual objective function, we rewrite the equipartition constraint in (3.10) as $(ee^T + I) \bullet X = n$. This is true since Trace$X = I \bullet X = n$.

This gives the SDP

$$\begin{array}{lrcl} \min & \frac{L}{4} \bullet X & & \\ \text{subject to} & \text{diag}(X) & = & e \\ & (ee^T + I) \bullet X & = & n \\ & X & \succeq & 0, \end{array} \tag{3.12}$$

with dual

$$\begin{array}{lrcl} \min & ny_0 + e^T y & & \\ \text{subject to} & S = \text{Diag}(y) + y_0(ee^T + I) & + & \frac{L}{4} \\ & S & \succeq & 0 \end{array} \tag{3.13}$$

Since $S = y_0(ee^T + I) + \text{Diag}(y) + \frac{L}{4}$ is p.s.d. we require $d^T S d = d^T(y_0(ee^T + I) + \text{Diag}(y) + \frac{L}{4})d \geq 0$, $\forall d$.

In particular we propose to use the following d for the min bisection problem.

MB1 Setting $d = e_i$, $\forall i = 1 \ldots n$, gives $2y_0 + y_i \geq -\frac{Lii}{4}$, $\forall i = 1 \ldots n$.

MB2 Setting $d = e_i - e_j$, $\forall \{i, j\} \in E$, we obtain $y_i + y_j + 2y_0 \geq \frac{Lij}{2} - \frac{Lii}{4} - \frac{Ljj}{4}$, $\forall \{i, j\} \in E$.

MB3 Setting $d = e_i + e_j$, $\forall \{i, j\} \in E$, we obtain $6y_0 + y_i + y_j \geq -\frac{L_{ij}}{2} - \frac{L_{ii}}{4} - \frac{L_{jj}}{4}$, $\forall \{i, j\} \in E$.

MB4 Setting $d = e$, where e is the all ones vector gives $(n^2 + n)y_0 + \sum_{i=1}^{n} y_i \geq 0$, since $Le = 0$.

MB5 The constraints in the bundle namely the columns p_i , $i = 1, \ldots, r$ of the matrix P.

The constraints in MB2 provide a lower bound on the LP objective value. To see this sum up all these constraints along a cycle say $\{1, 2, \ldots, n, 1\}$. This gives $ny_0 + \sum_{i=1}^{n} y_i \geq B \bullet \frac{L}{4}$, where B is a matrix with negative ones along the diagonal and ones elsewhere.

The resulting LP is

$$
\begin{array}{lrcll}
\min & ny_0 + e^T y & & & \\
\text{subject to} & 2y_0 + y_i & \geq & -\frac{L_{ii}}{4} & \forall i = 1 \ldots n \\
& 2y_0 + y_i + y_j & \geq & \frac{L_{ij}}{2} - \frac{L_{ii}}{4} - \frac{L_{jj}}{4} & \forall \{i,j\} \in E \\
& 6y_0 + y_i + y_j & \geq & -\frac{L_{ij}}{2} - \frac{L_{ii}}{4} - \frac{L_{jj}}{4} & \forall \{i,j\} \in E \\
& (n^2+n)y_0 + \sum_{i=1}^n y_i & \geq & 0 & \\
& ((p_j^T e)^2 + 1)y_0 + \sum_{i=1}^n p_{ji}^2 y_i & \geq & -p_j^T \frac{L}{4} p_j & \forall j = 1, \ldots, r
\end{array}
\tag{3.14}
$$

Note that the optimal face is unbounded, since there is a ray in the direction $y_0 = 1, y_i = -1, \forall i = 1, \ldots, n$. Therefore we impose a upper bound for y_0 say u. This is not surprising since the dual SDP (3.13) has an unbounded optimal face as well. To observe this set $y_0 \to \infty$ in (3.13). Doing so keeps S psd, and since y_0 does not appear in the objective function, this value remains unchanged. This gives the LP

$$
\begin{array}{lrcll}
\min & ny_0 + e^T y & & & \\
\text{subject to} & 2y_0 + y_i & \geq & -\frac{L_{ii}}{4} & \forall i = 1 \ldots n \\
& 2y_0 + y_i + y_j & \geq & \frac{L_{ij}}{2} - \frac{L_{ii}}{4} - \frac{L_{jj}}{4} & \forall \{i,j\} \in E \\
& 6y_0 + y_i + y_j & \geq & -\frac{L_{ij}}{2} - \frac{L_{ii}}{4} - \frac{L_{jj}}{4} & \forall \{i,j\} \in E \\
& (n^2+n)y_0 + \sum_{i=1}^n y_i & \geq & 0 & \\
& ((p_j^T e)^2 + 1)y_0 + \sum_{i=1}^n p_{ji}^2 y_i & \geq & -p_j^T \frac{L}{4} p_j & \forall j = 1, \ldots, r \\
& y_0 & \leq & u &
\end{array}
\tag{3.15}
$$

Here $p_{ji}, \forall j = 1, \ldots, n$ refers to the jth component of vector $p_i, \forall i = 1, \ldots, r$. (3.15) has $2m + n + r + 1$ constraints in all.

If we set the upper bound constraint, we are in essence solving the following pair of SDP's

$$\begin{array}{lrcl} \min & \begin{bmatrix} \frac{L}{4} & 0 \\ 0 & u \end{bmatrix} & \bullet & \begin{bmatrix} X & 0 \\ 0 & x_s \end{bmatrix} \\ \text{subject to} & \operatorname{diag}(X) & = & e \\ & (ee^T + I) \bullet X + x_s & = & n \\ & \begin{bmatrix} X & 0 \\ 0 & x_s \end{bmatrix} & \succeq & 0 \end{array} \tag{3.16}$$

with dual

$$\begin{array}{lrcl} \max & ny_0 + e^T y & & \\ \text{subject to} & \operatorname{Diag}(y) + y_0(ee^T + I) + S & = & \frac{L}{4} \\ & S & \succeq & 0 \\ & y_0 & \leq & u \end{array} \tag{3.17}$$

Here x_s is the dual variable corresponding to the upper bound constraint $y_0 \leq u$. Since $I \bullet X = n$, we have $ee^T \bullet X = -x_s$ in (3.16). Similarly the dual variable corresponding to this upper bound constraint in the LP (3.15) should provide an estimate for $ee^T \bullet X$.

3.5.3 The k equipartition problem

The **k** *equipartition* problem was introduced by Donath and Hoffman [13] and Rendl and Wolkowicz [50]. Our SDP formulation is taken from Lisser and Rendl [39]. It is interesting to note that that for $k = 2$ we have the min bisection problem. The **k** equipartition partition correponds to a $\{0, 1\}$ formulation, and the equivalence of (3.18) for $k = 2$ with the min bisection SDP (3.10) can be easily established.

The Lisser and Rendl [39] formulation is based on the following quadratic program in binary variables.

$$\begin{array}{lrcl} \min & \frac{1}{2}\text{Trace}Z^T LZ & & \\ \text{subject to} & Z^T e & = & ae \\ & Ze & = & e \\ & Z_{ij} & = & \{0,1\} \quad \forall i = 1,\ldots,n \quad j = 1,\ldots q, \end{array}$$

L is the Laplacian matrix of the graph, $a = \frac{n}{q}$, where n is the number of nodes and q is the number of partitions desired . Also Z is a $n \times q$, $(0,1)$ matrix whose rows correspond to the vertices V of the graph and whose columns represent the a equal components in which V is to be partitioned. An entry Z_{ij} of this matrix is 1 if vertex i belongs to component j in the partition.

A SDP relaxation is the following [39]

$$\begin{array}{lrcl} \min & \frac{1}{2}\text{Trace}LX & & \\ \text{subject to} & \text{diag}(X) & = & e \\ & Xe & = & ae \\ & X & \succeq & 0 \end{array} \tag{3.18}$$

This can also be written as

$$\begin{array}{lrcll} \min & \frac{L}{2} \bullet X & & & \\ \text{subject to} & e_i e_i^T \bullet X & = & 1 & \forall i = 1,\ldots,n \\ & \frac{1}{2}(ee_i^T + e_i e^T) \bullet X & = & a & \forall i = 1,\ldots,n \\ & X & \succeq & 0 & \end{array} \tag{3.19}$$

with dual

$$\begin{array}{lrcl} \min & \sum_{i=1}^n y_i & + & a\sum_{i=1}^n y_{n+i} \\ \text{subject to} & S & = & \sum_{i=1}^n e_i e_i^T y_i + \sum_{i=1}^n \frac{1}{2}(ee_i^T + e_i e^T) y_{n+i} + \frac{L}{2} \\ & S & \succeq & 0 \end{array} \tag{3.20}$$

We propose the following d

kEQ1 Setting $d = e_i$, $\forall i = 1, \ldots, n$ gives $y_i + y_{n+i} \geq -\frac{L_{ii}}{2}$, $\forall i$.

kEQ2 Setting $d = e$, gives $\sum_{i=1}^{n} y_i + n \sum_{i=1}^{n} y_{n+i} \geq 0$, since $Le = 0$.

kEQ3 Setting $d = (e_i + e_j)$, $\forall \{i, j\} \in E$ gives $y_i + y_j + 2y_{n+i} + 2y_{n+j} \geq -\frac{L_{ii}}{2} - \frac{L_{jj}}{2} - L_{ij}$.

kEQ4 Setting $d = (e_i - e_j)$, $\forall \{i, j\} \in E$ gives $y_i + y_j \geq -\frac{L_{ii}}{2} - \frac{L_{jj}}{2} + L_{ij}$.

kEQ5 The vectors p_i , $i = 1, \ldots, r$, in the bundle P.

The constraints in kEQ2 ensure that the y_{n+i} , $i = 1, \ldots, n$ don't take arbitarily large negative values, making the LP unbounded.

The resulting LP is

$$
\begin{array}{lrcll}
\min & \sum_{i=1}^{n} y_i + a \sum_{i=1}^{n} y_{n+i} & & & \\
\text{subject to} & y_i + y_{n+i} & \geq & -\frac{L_{ii}}{2} & \forall i = 1, \ldots, n \\
& \sum_{i=1}^{n} p_{ji}^2 y_i + \sum_{j=1}^{n} p_{ji}(p_j^T e) y_{n+i} & \geq & -p_j^T \frac{L}{2} p_j & \forall j = 1, \ldots, r \\
& \sum_{i=1}^{n} y_i + n \sum_{i=1}^{n} y_{n+i} & \geq & 0 & \\
& y_i + y_j + 2y_{n+i} + 2y_{n+j} & \geq & -\frac{L_{ii}}{2} - \frac{L_{jj}}{2} - L_{ij} & \forall \{i, j\} = 1, \ldots, m \\
& y_i + y_j & \geq & -\frac{L_{ii}}{2} - \frac{L_{jj}}{2} + L_{ij} & \forall \{i, j\} = 1, \ldots, m
\end{array}
\tag{3.21}
$$

(3.21) has $n + 2m + r + 1$ constraints in all.

3.5.4 The Lovasz theta function

The Lovasz theta function was introduced by Lovasz [41] in connection with the Shannon capacity of the graph. There are various formulations of the Lovasz theta function, and their equivalence is established in Grotschel, Lovasz and Schrijver [17]. There are also a number of SDP's whose objective value gives the Lovasz theta function. We consider two such formulations in this section. The first formulation is taken from Grotschel, Lovasz and Schrijver [17] and also appears in Gruber and Rendl [18], and the *SDPT3* manual [54].

The Lovasz theta function is an upper bound on the independent set number of a graph, and a lower bound on the chromatic number of the complementary

graph. The importance of the Lovasz theta function lies in the fact that computing the independent set number and the chromatic number are *NP complete* problems, whereas the Lovasz theta function, which is sandwiched in between these two numbers, can be computed in polynomial time by solving an SDP. This SDP can be written as

$$\begin{array}{lrcll} \min & C \bullet X & & & \\ \text{subject to} & I \bullet X & = & 1, & \\ & A_k \bullet X & = & 0, & k = 1, \ldots, n \\ & X & \succeq & 0, & \end{array} \tag{3.22}$$

with dual

$$\begin{array}{llllllll} \min & y_0 & & & & & & \\ \text{subject to} & -y_0 I & + & -\sum_{k=1}^{m} y_k A_k & + & S & = & C \\ & & & & & S & \succeq & 0 \end{array} \tag{3.23}$$

Here C is the matrix of all minus ones, $A_k = e_i e_j^T + e_j e_i^T$, where the kth edge of the graph (with m edges) is from vertex i to vertex j. Here e_i denotes the ith unit vector. The a in $\text{tr}(X) = a$ is 1, and appears as a constraint in the original SDP.

Since $S = y_0 I + \sum_{k=1}^{m} y_k A_k + C$ is p.s.d., we require that $d^T S d = d^T (y_0 I + \sum_{k=1}^{m} y_k A_k + C) d \geq 0, \forall d$.

In particular we propose to use the following d for the Lovasz theta problem.

LT1 Setting $d = e_i$, $i = 1 \ldots n$, gives $y_0 \geq 1$. This constraint implies that a lower bound on the Lovasz theta number is one. (The Lovasz theta number is an upper bound on the maximum clique number of the graph, and each graph trivially has a clique of size 1.)

LT2 Setting $d = e_i + e_j$, $\forall \{i, j\} \in E$ gives $y_0 + y_k \geq 2$, $k = 1, \ldots, m$.

LT3 Setting $d = e_i - e_j$, $\forall \{i, j\} \in E$ gives $y_0 - y_k \geq 0$, $k = 1, \ldots, m$.

LT4 Setting $d = e$ gives $ny_0 + 2\sum_{i=1}^{n} y_i \geq n^2$.

LT5 The bundle constraints namely the columns p_i , $i = 1, \ldots, r$ of the matrix P.

The constraints in LT2 and LT3 are required to force y_0 to move as y_k move. Otherwise the LP is trivial to solve. The optimal solution is $y_0 = 1$ for an objective value of 1, adjusting the y_k , $k = 1, \ldots, m$ so as to satisfy the remaining constraints.

The resulting LP is

$$\begin{array}{lrcll} \min & y_0 & & & \\ & y_0 + y_k & \geq & 2 & \forall k = 1, \ldots, m \\ & y_0 - y_k & \geq & 0 & \forall k = 1, \ldots, m \\ & ny_0 + 2\sum_{i=1}^{n} y_i & \geq & n^2 & \\ & p_i^T(y_0 I + \sum_{k=1}^{m} y_k A_k + C)p_i & \geq & 0 & \forall i = 1, \ldots, r \\ & y_0 & \geq & 1 & \end{array} \tag{3.24}$$

Here $A_k = e_i e_j^T + e_j e_i^T$, $\forall k = \{i, j\}$. (3.24) has $2m + 2 + r$ constraints in all.

There is yet another SDP relaxation called the maximum stable set relaxation that is equivalent to the Lovasz theta problem. We briefly mention this relaxation below. This equivalence was established in Kleinberg and Goemans [34]. Also refer to Benson and Ye [6] for more details.

The maximum stable set problem is

$$\begin{array}{lrcll} \max & \begin{bmatrix} 0.5 & \ldots & 0 & -0.25 \\ & \ddots & & \vdots \\ 0 & \ldots & 0.5 & -0.25 \\ -0.25 & \ldots & -0.25 & 0 \end{bmatrix} \bullet X & & & \\ \text{subject to} & \text{diag}(X) & = & e & \\ & \frac{1}{2}(e_i e_n^T + e_n e_i^T + e_j e_n^T + e_n e_j^T - e_i e_j^T - e_j e_i^T) \bullet X & = & 1 & \forall \{i, j\} \in E \\ & X & \succeq & 0 & \end{array}$$

with dual

$$\begin{array}{lll} \min & \sum_{i=1}^{n} y_i \; + \; \sum_{\{i,j\}\in E} y_{ij} & \\ \text{subject to} & S \; = & \\ & \sum_{i=1}^{n} e_i e_i^T y_i + \sum_{\{i,j\}\in E} \frac{1}{2}(e_i e_n^T + e_n e_i^T + e_j e_n^T + e_n e_j^T - e_i e_j^T - e_j e_i^T) y_{ij} - C & \\ & S \; \succeq \; 0 & \end{array}$$

Here $C = 0.5I - 0.25(ee_n^T + e_n e^T)$. Here the underlying graph G has $n-1$ vertices, the nth vertex is artificial, with no edges connecting it to the other edges. This vertex is definitely part of the maximal stable set, and is used to identify the stable set in the graph. Morever the a in $\text{Trace}(X) = a$ is n.

We consider the following d.

MSS1 Setting $d = e_k$, $\forall k = 1, \ldots, n-1$ gives $y_k \geq 0.5$

MSS2 Setting $d = e_n$ gives $y_n \geq 0$

MSS3 Setting $d = e$ gives $\sum_{i=1}^{n} y_i + \sum_{\{i,j\}\in E} y_{ij} \geq 0$. This gives a trivial lower bound on the LP objective value.

MSS4 Setting $d = (e_k - e_l)$, $\forall \{k,l\} \in E$ gives $y_k + y_l + y_{kl} \geq 1$.

MSS5 Setting $d = (e_k + e_l)$, $\forall \{k,l\} \in E$ gives $y_k + y_l - y_{kl} \geq 1$.

MSS6 The bundle constraints, i.e. the columns p_k , $k = 1, \ldots, r$ of the matrix P. These give $\sum_{i=1}^{n} p_{ki}^2 y_i + \sum_{\{i,j\}\in E} (p_{ki}p_{kn} + p_{kj}p_{kn} - p_{ki}p_{kj}) y_{ij} \geq 0.5 - 0.5(p_k^T e)(p_{kn})$.

The resulting LP is

$$
\begin{array}{lrcll}
\min & \sum_{i=1}^{n} y_i + \sum_{\{i,j\}\in E} y_{ij} & & & \\
\text{subject to} & \sum_{i=1}^{n} y_i + \sum_{\{i,j\}\in E} y_{ij} & \geq & 0 & \\
& y_i + y_j - y_{ij} & \geq & 1 & \forall \{i,j\} \in E \\
& y_i + y_j + y_{ij} & \geq & 1 & \forall \{i,j\} \in E \\
& \sum_{i=1}^{n} p_{ki}^2 y_i + \sum_{\{i,j\}\in E} (p_{ki}p_{kn} + p_{kj}p_{kn} - p_{ki}p_{kj}) y_{ij} & & & \\
& & \geq & 0.5 - 0.5(p_k^T e)p_{kn} & k = 1, \ldots, r \\
& y_i & \geq & 0.5 & i = 1, \ldots, n-1 \\
& y_n & \geq & 0 &
\end{array}
\tag{3.25}
$$

(3.25) has $2m + r + 1$ constraints in all.

This problem has a trivial objective value of 0. This is not surprising since our objective was to minimise $\sum_{i=1}^{n} y_i + \sum_{\{i,j\}\in E} y_{ij}$, whereas constraint MSS3 indicates that this should be greater than 0. We are unable to find any cutting planes that cut off this trivial solution. Note that constraint MSS3 provides a trivial lower bound on the objective function, in the absence of which the problem is unbounded.

3.5.5 Box constrained QP's

Consider the the following box constrained QP

$$
\begin{array}{lrcl}
\max & x^T Q x & & \\
\text{subject to} & -e & \leq x \leq & e
\end{array}
$$

A SDP relaxation of this problem, proposed by Ye [61] is

$$
\begin{array}{lrcl}
\max & Q \bullet X & & \\
\text{subject to} & \text{diag}(X) & \leq & e, \\
& X & \succeq & 0,
\end{array}
\tag{3.26}
$$

with dual

$$\begin{array}{lrcl} \min & e^t y & & \\ \text{subject to} & \text{Diag}(y) & \succeq & Q \\ & y & \geq & 0 \end{array} \tag{3.27}$$

Note that we can rewrite the primal SDP in the standard form as

$$\text{max} \quad \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix} \bullet \begin{bmatrix} X & 0 \\ 0 & \text{Diag}(x_s) \end{bmatrix}$$

$$\text{subject to} \quad A_i \bullet \begin{bmatrix} X & 0 \\ 0 & \text{Diag}(x_s) \end{bmatrix} = 1 \quad \forall i = 1, \ldots, n$$

$$\begin{bmatrix} X & 0 \\ 0 & \text{Diag}(x_s) \end{bmatrix} \succeq 0$$

Here $A_i \in \mathcal{S}^{2n}$ and is given by $e_i e_i^T + e_{n+i} e_{n+i}^T$.

The dual SDP is given by

$$\begin{array}{lrcl} \min & e^T y & & \\ \text{subject to} & S & = & \sum_{i=1}^m y_i A_i + \begin{bmatrix} -Q & 0 \\ 0 & 0 \end{bmatrix} \succeq 0 \end{array}$$

This problem is similar to the max cut problem, except that all the matrix sizes have been doubled, and the A_i matrices are now the sum of two rank one matrices. The value of $a = n$.

We use the following d.

QPB1 Setting $d = e_i$, $\forall i = 1, \ldots, n$ gives $y_i \geq Q_{ii}$ and $y_i \geq 0$, $\forall i = 1, \ldots, n$. Therefore set $y_i \geq \max\{Q_{ii}, 0\}$. Since Q is a typically a p.s.d. matrix, this constraint is $y_i \geq Q_{ii}$.

QPB2 The columns p_i , $i = 1, \ldots, r$ in the bundle P.

The resulting LP is

$$\begin{array}{lrcll} \min & & e^T y & & \\ \text{subject to} & p_j^T(\sum_{i=1}^m y_i A_i - \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix})p_j & \geq & 0 & \forall j = 1, \ldots, r \\ & y_i & \geq & Q_{ii} & \end{array} \tag{3.28}$$

(3.28) has $n + r$ constraints in all.

3.6 Computational results

In this section we test the linear programming approach on a number of combinatorial optimisation problems, taken from the 7th DIMACS Implementation Challenge [49] and Borchers' SDPLIB [8]. For the **k** equipartition problem we take a random instance from Rendl [39] and a 32 node NFL problem from Mitchell [43]. The bundle constraints are computed using Helmberg's spectral bundle code *SBmethod, Version 1.1* [24] available at *http://www.zib.de/helmberg/index.html*. *CPLEX 6.5* [28] is employed in solving the LP relaxations, within the *MATLAB* framework. All tests are executed on a *Sun Ultra 5.6, 440MHz* machine with *128* MB of memory.

The spectral bundle approach requires a number of parameters. We briefly mention some of the important parameters toegther with their default values. For a detailed description of these parameters, we refer the reader to Helmberg's user's manual for *SBmethod* [24].

1. The relative tolerance (*-te*). The default value is $1e - 5$.
2. The size of the bundle i.e. the number of columns in P. This in turn is controlled by
   (a) The maximum number of vectors kept n_k (*-mk*). The default value is 20.
   (b) The maximum number of vectors added (*-ma*). The default value is 5.
   (c) The minimum number of vectors added n_{min} (*-mik*). The default value is 5.
3. The time limit in seconds (*-tl*).

It must be emphasised that the main computational task in the bundle approach is computing the vectors in the bundle P. Solving the resulting LP relaxations is relatively trivial.

For some of the problems the bundle approach failed to converge to the desired tolerance ($1e-5$). For these problems, we provide run times in parentheses next to the bundle size r , and also the objective value attained by the bundle approach next to the optimal SDP objective value.

The columns in the tables represent

n Problem size i.e. the number of nodes in the graph.

k Number of SDP constraints.

m Number of edges in the graph.

r Bundle size, the number of columns in P.

% Error $|\frac{SDP-LP}{SDP} \times 100|$.

Table 3.1 SDP LP1 LP2 The objective value of the various relaxations.

% Error $\frac{SDP-LP2}{SDP} \times 100$.

Table 3.3 m1 m2 Number of constraints in LP's (3.7) and (3.9) respectively.

Table 3.2 r Bundle size, i.e. the number of columns in matrix P.

(No of Hrs) The number of hours the spectral code was run.

Bundle Objective value attained by the bundle

Table 3.4 m1 Number of constraints in LP (3.15).

Table 3.5 q Number of equipartitions required.

m1 Number of constraints in LP (3.21).

Table 3.6 m1 Number of constraints in LP (3.24).

Table 3.7 m1 Number of constraints in LP (3.28).

Name	n	m	r	SDP	LP1	LP2	% Error
toruspm-8-50 [1]	512	1536	16	527.81	525.91	526.36	0.27
toruspm3-15-50 [1]	3375	10125	21	3.47e+03	3.43e+03	3.45e+03	0.81
torusg3-8 [1]	512	1536	13	4.57e+07	4.54e+07	4.55e+07	0.43
torusg3-15 [1]	3375	10125	18	3.13e+08	3.10e+08	3.11e+08	0.69
mcp100 [2]	100	269	10	226.16	225.75	225.86	0.13
mcp124-1 [2]	124	149	20	141.99	81.02	141.95	0.03
mcp124-2 [2]	124	318	11	269.88	268.97	269.21	0.25
mcp124-3 [2]	124	620	11	467.75	467.37	467.44	0.06
mcp124-4 [2]	124	1271	10	864.42	863.72	863.81	0.07
mcp250-1 [2]	250	331	10	317.26	172.04	317.21	0.02
mcp250-2 [2]	250	612	14	531.93	519.03	531.38	0.1
mcp250-3 [2]	250	1283	13	981.17	980.32	980.48	0.07
mcp250-4 [2]	250	2421	13	1681.96	1679.70	1680.00	0.12
mcp500-1 [2]	500	625	10	598.15	321.39	596.67	0.25
mcp500-2 [2]	500	1223	13	1070.06	1069.90	1069.95	0.01
mcp500-3 [2]	500	2355	14	1847.97	1843.20	1844.13	0.21
mcp500-4 [2]	500	5120	15	3566.74	3559.80	3560.64	0.17
maxG11 [2]	800	1600	11	629.16	27.56	626.99	0.35
maxG32 [2]	2000	4000	14	1567.64	25.11	1560.94	0.43
maxG51 [2]	1000	5909	19	4003.81	3988.30	3992.60	0.3
maxG55 [2]	5000	14997	25	12870	11523.66	12855.03	0.12
maxG60 [2]	7000	17148	21	15222.27 [6]	8606.60	13837.29	9.1
maxG60 [2]	7000	17148	24	15222.27 [6]	8615.31	14286.71	6.15
maxG60 [2]	7000	17148	25	15222.27 [6]	5334.17	14796.33	2.79

Table 3.1: Max Cut Test Results

In table 3.1 we compare the SDP objective value with the LP relaxations $LP1$ (3.7) and $LP2$ (3.9). It is seen that in most cases $LP1$ alone provides a fairly good approximation to the SDP objective value. However for problems *mcp124-1*, *mcp500-1*, *maxG11*, *maxG32* we need to work with the relaxation $LP2$. These examples underline the importance of the m rank 2 constraints in $LP2$. The LP relaxation $LP2$ provides an excellent approximation to the SDP with the $\%error$

[1] DIMACS [49]
[2] SDPLIB [8]
[3] Rendl [39]
[4] Mitchell [43]
[5] Memory exceeded
[6] Spectral bundle failed to terminate

Name	r (No of hrs)	Bundle
maxG60 [2]	21(1)	15318.40
maxG60 [2]	24(2)	15298.20
maxG60 [2]	25(6)	15270.28

Table 3.2: Bundle solutions for maxG60

well below 1% of the SDP objective value.

The bundle approach fails to converge for problem *maxG60*. For the *maxG55* problem we run the bundle code for 10 hours. The bundle approach attains an objective value of 12870 (reported SDP objective in SDPLIB is 9999.210). Morever since the LP relaxations $LP1$ and $LP2$ are larger than 9999.210 it appears that the reported solution in SDPLIB is incorrect. For the *maxG60* problem we consider three runs with the bundle code, of durations 1 hour, 2 hours and 6 hours respectively. The objective value attained by the bundle scheme, and the number of vectors in the bundle P are listed in table 3.2. It is seen that longer the run, the closer the bundle objective to the SDP objective value, and also tighter is the $LP2$ relaxation.

We list the sizes of the three max cut relaxations in table 3.3. The SDP has n constraints, whereas $LP1$ and $LP2$ are LP relaxations with $O(n+\sqrt{k})$ and $O(n+m+\sqrt{k})$ constraints respectively. Here $k=n$.

In table 3.4 we compare the SDP objective value with the LP relaxation (3.15). Here $u=1$, the upper bound on the variable y_0. It is seen that the LP relaxation provides an excellent approximation to the SDP objective value. The value of the dual variable, corresponding to the upper bound constraint $y_0 \leq 1$ provides an estimate for $|ee^t \bullet X|$. This value is below 0.1 for all, but one of the reported instances. A typical LP relaxation has $O(m+n+\sqrt{k})$ constraints, with $k=n+1$.

In table 3.5 we compare the SDP objective value of the **k** equipartition problem with the value of LP relaxation (3.21). It is interesting to note that the LP relaxation of the bisection problem, i.e. **k** equipartition with $q=2$, does not have an unbounded optimal face as the LP relaxation $LP1$ (3.14) of min bisection. However the SDP relaxation (3.18) which represents a $\{0,1\}$ formulation of min bisection, as opposed to the $\{-1,1\}$ SDP formulation (3.10) is relatively harder to solve, using the bundle approach, since we are dealing with more constraints. Among **k** equipar-

Name	SDP		LP1	LP2
	k	n	m1	m2
toruspm-8-50	512	512	528	2064
toruspm3-15-50	3375	3375	3396	13521
torusg3-8	512	512	525	2061
torusg3-15	3375	3375	3393	13518
mcp100	100	100	110	379
mcp124-1	124	124	144	293
mcp124-2	124	124	135	453
mcp124-3	124	124	135	755
mcp124-4	124	124	134	1405
mcp250-1	250	250	260	591
mcp250-2	250	250	264	876
mcp250-3	250	250	263	1546
mcp250-4	250	250	263	2684
mcp500-1	500	500	510	1135
mcp500-2	500	500	513	1736
mcp500-3	500	500	514	2869
mcp500-4	500	500	515	5635
maxG11	800	800	811	2411
maxG32	2000	2000	2014	6014
maxG51	1000	1000	1019	6928
maxG55	5000	5000	5025	20022
maxG60	7000	7000	7025	24173

Table 3.3: Sizes of the max cut relaxations

tition problems, we need a larger bundle P for the bipartition problems *gpp124-1*, *gpp250-1*. For these problems we choose the maximum number of vectors kept $n_k = 50$ and $n_{min} = 25$. With these enhanced bundle sizes we are able to get tighter LP approximations to *gpp1241*. For instance with the default bundle size we are only able to get an LP approximation of 43.0370 (SDP objective value is 7.3431), but with the larger bundle size our LP relaxation is 7.3443 for a $\%error$ of 0.016. However this results in bundle sizes r much larger than $\sqrt{k}$. A typical LP relaxation has $O(m + n + \sqrt{k})$ constraints, where $k = 2n$.

In table 3.6 we compare the SDP objective value of the Lovasz theta problem with the value of the LP relaxation (3.24). Since the number of constraints in the Lovasz theta problem is $O(m)$ a larger bundle is typically required. Thus the

Name	m	n	r	SDP	LP	m1	% Error	$\lvert ee^T \bullet X \rvert$
bm1 [1]	4711	882	10	23.44	24.99	10315	6.59	0.062
gpp100 [2]	264	100	10	44.94	45.14	639	0.44	0.019
gpp124-1 [2]	149	124	10	7.34	7.34	433	0.01	0.001
gpp124-2 [2]	318	124	10	46.86	47.27	771	0.55	0.029
gpp124-3 [3]	620	124	11	153.01	153.40	1376	0.16	0.011
gpp124-4 [2]	1271	124	11	418.99	419.98	2678	0.14	0.039
gpp250-1 [2]	331	250	10	15.45	15.45	923	0.88	0.002
gpp250-2 [2]	612	250	12	81.87	82.09	1487	0.26	0.019
gpp250-3 [2]	1283	250	13	303.50	305.21	2830	0.56	0.018
gpp250-4 [2]	2421	250	13	747.30	750.58	5106	0.43	0.029
gpp500-1 [2]	625	500	11	25.30	26.25	1762	3.62	0.182
gpp500-2 [2]	1223	500	13	156.06	157.73	2960	1.06	0.055
gpp500-3 [2]	2355	500	15	513.02	517.91	5226	0.95	0.015
gpp500-4 [2]	5120	500	15	1567.02	1575.95	10756	0.57	0.008
biomedP [1]	629839	6514	-	33.60	MM [5]	-	-	-
industry2 [1]	798219	12637	-	65.61	MM [5]	-	-	-

Table 3.4: Min Bisection Test Results

Name	m	n	q	r (No of Hrs)	SDP	LP	m1	% Error
nfl1 [4]	496	32	8	9	7.34e+05	7.32e+05	1034	0.23
nfl2 [4]	496	32	4	9	6.29e+05	6.29e+05	1034	0
A100 [3]	4070	100	2	10	8.62e+05	8.62e+05	8251	4.6e-04
gpp100 [2]	364	100	2	10	44.94	44.95	839	0.0031
gpp124-1 [2]	149	124	2	10	7.34	43.03	433	486.09
gpp124-1 [2]	149	124	2	31	7.34	7.34	454	0.016
gpp124-2 [2]	318	124	2	8	46.86	46.86	769	0.0036
gpp250-1 [2]	331	250	2	40 (3)	15.45 [6] (15.41)	18.76	953	21.4438
gpp250-2 [2]	612	250	2	11	81.87	81.87	1486	0.0037

Table 3.5: k Equipartition Test Results

bundle method is fairly time consuming on these problems. For instance *theta3* the bundle approach requires about 4 hours to converge to the desired tolerance, whereas it fails to converge for instances *theta4*, *theta5*, *theta6*. For these problems we incorporate the bundles after a 6 hour run. The objective value attained by the bundle approach during this period, is reported in parantheses next to the SDP objective value. We retain the default bundle parameters. However for the DIMACS

Name	m	n	r (No of Hrs)	SDP	LP	m1	% Error
hamming-9-8 [1]	2304	512	23	224	221.68	4633	1.03
hamming-7-5-6 [1]	1792	128	12	42.67	42.53	3598	0.33
hamming-10-2 [1]	23040	1024	-	102.40	MM [5]	-	-
hamming-11-2 [1]	56320	2048	-	170.67	MM [5]	-	-
hamming-8-3-4 [1]	16128	256	43	25.60	25.32	32301	1.094
hamming-9-5-6 [1]	53760	512	-	85.33	MM [5]	-	-
theta1 [2]	103	50	10	23	22.95	218	0.2161
theta2 [2]	497	100	20	32.87	32.84	1016	0.10
theta3 [2]	1105	150	25	42.17	40.14	2237	4.81
theta3 [2]	1105	150	47	42.17	42.11	2259	0.14
theta4 [2]	1948	200	25(6)	50.32 [6] (50.35)	38.12	3923	24.25
theta4 [2]	1948	200	51(6)	50.32 [6] (50.33)	49.63	3949	1.38
theta5 [2]	3027	250	25(6)	57.23 [6] (57.28)	33.30	6081	41.81
theta5 [2]	3027	250	51(6)	57.23 [6] (57.58)	51.87	6107	9.38
theta5 [2]	3027	250	51(10)	57.23 [6] (57.39)	54.64	6107	4.79
theta6 [2]	4374	300	25(6)	63.48 [6] (63.56)	30.06	8775	52.64
theta6 [2]	4374	300	59(6)	63.48 [6] (65.05)	50.54	8809	20.39
theta6 [2]	4374	300	58(10)	63.48 [6] (64.32)	54.57	8808	14.03
theta6 [2]	4374	300	58(15)	63.48 [6] (63.91)	57.27	8808	9.78

Table 3.6: Lovasz theta Test Results

Hamming instances, since the optimal solution can be computed analytically, the bundle approach is able to find the optimal bundle in a few iterations. We are unable to solve to solve *hamming-10-2*, *hamming-11-2* and *hamming9-5-6* where we run out of memory. It is seen that in the instances where the bundle approach converges, our LP solution is an excellent approximation to the SDP objective value with the $\%error$ well under 1%. To improve on the LP relaxations for the SDPLIB problems *theta4*, *theta5* and *theta6*, we choose larger bundle sizes $n_k = 50$ and $n_{min} = 25$ and $n_{add} = 25$. However this slows down the bundle code. We compute the LP relaxations, by choosing the bundle after 6 and 10 hour runs respectively. We are able to considerably strength en our LP relaxation in all cases. A typical LP relaxation has $O(m + \sqrt{k})$ constraints, where $k = O(m)$. It must be emphasised here that all the SDPLIB Lovasz theta problems can be solved in under an hour using any interior point package such as [54].

Table 3.7 compares the SDP objective of the box constrained QP with the

Name	k	n	r	SDP	LP	m1	% Error
qpG11 [2]	800	1600	11	1.181e+04	1.1764e+04	811	0.39
qpG51 [2]	1000	2000	10	2448.659	2434.502	1010	0.57

Table 3.7: Box QP Test Results

value of the LP relaxation (3.28). This LP provides an excellent approximation to the SDP objective value and has $O(n + \sqrt{k})$ constraints. Here $k = n$.

3.7 Conclusions

We have presented a LP approach to solving semidefinite programming problems. This approach requires the bundle constraints generated by the *spectral bundle* approach due to Helmberg and Rendl. The number of these constraints is bounded by the square root of the number of constraints k in the SDP. Typically fewer than these are required, due to the aggregration employed in the bundle approach. In addition to the bundle constraints, a few others *polynomial* in the problem size n are generally required. The main computational task in the LP approach is in computing the optimal bundle P. Solving the resulting LP relaxations is relatively trivial. We have also presented an interior cutting plane framework, which could be used to solve SDP as a sequence of LP's. We use this cutting plane approach to demonstrate the importance of the bundle constraints in the LP formulation.

It appears from the results presented in this chapter that

- The LP approach is very successful in solving the max cut and the box constrained QP problems. A typical maxcut LP requires $O(n + m + \sqrt{k})$ constraints, where $k = n$. For the box constrained QP instances reported, we find that $O(n + \sqrt{k})$ constraints suffice. These LP's can be solved easily using any of the commercial packages available.

- The original LP relaxation of the min bisection problem (3.14), like the SDP (3.10) has an unbounded optimal face. Hence we need to introduce an additional constraint $y_0 \leq u$ in our LP relaxation. Here y_0 is the dual variable corresponding to the equipartition constraint $ee^T \bullet X = 0$. Thus we are in

practice, solving a variant of the min bisection problem However on all the reported problems the value $|ee^T \bullet X|$ is very small of the order of 0.1. A typical LP has $O(m + n + \sqrt{k})$ constraints, where $k = n + 1$.

- The spectral bundle approach is fairly time consuming on Lovasz theta problems. This is because the number of constraints in the SDP is $O(m)$. Moreover we need larger bundle sizes than those provided by the bundle approach to tighten our LP relaxations. It appears that the traditional interior point methods outperform the spectral bundle method, especially on some of the smaller Lovasz theta problems. A typical LP has $O(m + \sqrt{k})$ constraints, where $k = O(m)$.

- The **k** equipartition SDP (3.18) does not have an unbounded optimal face, unlike the min bisection SDP (3.10). However (3.18) has more constraints k and hence computing the optimal bundle is time consuming. A typical LP has $O(m+n+\sqrt{k})$ constraints where $k = 2n$. In certain instances, we need larger bundle sizes $r > \sqrt{k}$ to tighten our LP relaxations.

- For the maximum stable set formulation of the Lovasz theta number, we are unable to find cutting planes, that cut off the point with a trivial objective value zero. We could investigate, incorporating the bundle LP within the cutting plane framework of section 6, and try to cut off this point.

- The bundle LP, with the additional box constraints introduced in section 5, is superior to the naive interior point cutting plane approach discussed in section 6. Not only are we are able to get tighter relaxations in the former case, but the resulting LP's are smaller as well. Thus the bundle constraints seem to be optimal in a certain sense, in that they identify an *important subspace* on which the matrix of dual slacks S should be positive semidefinite, and directions not in subspace do not seem all that important.

- We could in practice strengthen our bundle LP relaxation, by incorporating it within the cutting plane framework. We investigate this in greater detail in Krishnan and Mitchell [31].

- The spectral bundle approach depends on a number of parameters, especially the maximum and minimum bundle sizes, the number of Lanczos vectors added in each iteration. In this paper we have experimented with various bundle sizes r, especially with regard to the Lovasz theta and $\mathbf{k}$ equipartition problems. Further investigation of the choice of these parameters, especially the role they play with regard to the strength of our LP relaxations is necessary.

To conclude it is felt that a beginning is made to solve an SDP, with a constant trace on the primal feasible set, as an LP. Although SDP's are *semi infinite* LP's, we provide empirical evidence that only a few constraints, polynomial in the problem size n, are typically required. Furthermore one could incorporate the above framework in a cutting plane LP approach, to solving semidefinite programs. We pursue this in [31].

3.8 Acknowledgements

The author wishes to thank Christoph Helmberg for making his spectral bundle code available and for useful discussions.

CHAPTER 4
Cutting plane approaches

4.1 Introduction

In the previous chapter, we presented a linear programming (LP) approach to solving semidefinite programming problems. The linear constraints employed in the LP approach, were taken from the bundle maintained by the spectral bundle approach, and a set of *box* constraints, which were problem specific, but polynomial in the problem size. In this chapter we present a cutting plane approach to solving an SDP. This amounts to solving an SDP as a sequence of LP's. However to make the resulting method competitive with interior point methods for SDP, several refinements are required. We describe such a method in this chapter. The LP's are solved approximately, using an interior point approach, since this results in better cutting planes, than an interior point approach.

Most of the work in this chapter is still in progress. Extracts from this chapter are taken from our forthcoming paper Krishnan and Mitchell [31].

This chapter is organised as follows. Section 2 presents our cutting plane LP algorithm, section 3 describes how we compute the eigenvectors corresponding to the most negative eigenvalues, which are needed as cutting planes, section 4 describes how we generate a feasible starting point for each LP relaxation, section 5 discusses some criteria employed in dropping constraints, section 6 presents some computational results on the max cut problem, and we conclude with some observations in section 7.

4.2 A cutting plane LP approach to solving SDP

In this section we describe a scheme to solve SDP, as a sequence of LP's incorporated in a cutting plane framework. For more details, we refer the reader to Krishnan and Mitchell [31].

For convenience we assume we are solving the LP relaxation (LDR), where the objective is to minimise $b^T y$. Our constraints are now of the form $dd^T \bullet (\mathcal{A}^T y + C) \geq$

0. The y that solves the original LP relaxation (LDR) is obtained by reversing the sign.

Cutting plane Algorithm

Input :

$\mathcal{A}(.), b, C$, the optimal solution y_1 to the current relaxation LDR_1.

Parameters :

TOL The starting tolerance to which we solve the LP relaxations. Typically $1e-1$.

s The number of eigenvectors i.e. cutting planes added in each iteration. Typically $s = (20, 30)$.

μ The factor $\in (0, 1)$ by which the tolerance is reduced in each iteration. Typically $\mu = 0.95$.

MAXITER The maximum number of iterations carried out.

for $i = 1 :$ MAXITER **do**
begin

1. Compute the current dual slack matrix $S = \mathcal{A}^t y_i + C$.
2. Compute the s most negative eigenvalues $\lambda_j, j = 1, \ldots, s$ of S, together with the corresponding normalised eigenvectors $d_j, j = 1, \ldots, s$ using the *Lanczos* scheme. Note that $d_j^T(\mathcal{A}^T y + C)d_j = \lambda_j < 0, j = 1, \ldots, s$.
3. **if** $|\lambda_1| < \epsilon$ or $TOL < 1e-8$ **then** exit. Here λ_1 is the most negative eigenvalue.
4. Add these s cutting planes $d_j d_j^T \bullet (C + \mathcal{A}^T y) \geq 0, j = 1, \ldots, s$, to the current LP relaxation.
5. Solve the current LP relaxation, using an interior point LP code with a strictly feasible starting point, to the desired tolerance TOL. We illustrate how we generate a strictly feasible starting point for the max cut problem in section 4.4. Let the optimal solution to current LP relaxation be y_{i+1}.

6. TOL $= \mu \times$ TOL.

7. We drop constraints if necessary. Our criteria for dropping constraints is similar to Atkinson and Vaidya [3]. For more details on dropping constraints refer to section 4.5.

end

We find that we are able to get tighter relaxations using an interior point code, as compared to the simplex scheme. Morever since we solve our LP relaxations initially to reasonably high tolerances TOL, we find that it is better to solve the current LP relaxation with a strictly feasible starting point, so as to guarantee primal and dual feasibility, at these tolerances.

4.3 Computing the extremal eigenvalues and eigenvectors of S

We need a scheme that enables us to compute the k most negative eigenvalues of the dual slack matrix in step (2). The *Lanczos* scheme is ideally suited for this purpose. We give a brief overview of the Lanczos scheme below. An excellent reference for the Lanczos scheme is the book by Golub and Van Loan [16].

In order to ensure that the Lanczos scheme computes the k most negative eigenvalues of the dual slack matrix $S = C + \mathcal{A}^T y$, we perform the following

1. Run the power series iteration on the matrix S to compute the maximal eigenvalue $\lambda_{max}(S)$.

2. If $\lambda_{max}(S) < 0$, then run the Lanczos scheme with the matrix S.

3. If $\lambda_{max}(S) > 0$, then run the Lanczos scheme with the matrix $A = S - \lambda_{max} I$. Note that the maximal eigenvalue of A will be negative. The eigenvalues of S are related to those of A by $\lambda_S = \lambda_A + \lambda_{max}(S)$.

4. The Lanczos scheme accesses the matrix S only through matrix vector multiplications. Thus S can be stored in a sparse format and we only need a function that performs a sparse matrix vector multiplication. The Lanczos

scheme converts the matrix S into a symmetric tridiagonal matrix T of size $k < n$, such that the k eigenvalues of T are the k extremal eigenvalues of S. The tridiagonal matrix is stored as two vectors, a k vector α containing the diagonal elements of T and a $k-1$ vector β containing its subdiagonal elements.

5. The eigenvalues of T can be found by using some of the specialised routines available for tridiagonal matrices such as bisection etc. Excellent routines to compute the eigenvalues and eigenvectors of symmetric tridiagonal matrices are available in *LAPACK*. See the *LAPACK* users guide for more details [37].

For the sake of completeness, we present the power iteration and the Lanczos schemes below. More details can be found in [16].

Power Iteration

Input : A matrix $A \in \mathcal{S}^n$ and a vector $q^0 \in \mathcal{R}^n$. **Ouput :** $\lambda_{max}(A) = \lambda^k$ and its eigenvector q^k.

for $k = 1, \ldots$

$z^k = Aq^{k-1}$

$q^k = \frac{z^k}{||z^k||_2}$

$\lambda^k = (q^k)^T A q^k$

end

Lanczos Algorithm

Input : A symmetric matrix A and $w \in \mathcal{R}^n$ having unit norm. Note that we do need the matrix A explicitly, but only a function $A.mult(w)$, that given A and w computes the matrix vector product Aw.

Output: A symmetric tridiagonal matrix T_k with the property that the eigenvalues of T_k are eigenvalues of A. The diagonal and the subdiagonal elements of T_k are stored in $\alpha(1:k)$ and $\beta(1:k-1)$ respectively.

$v(1:n) = 0; \beta_0 = 1; k = 0.$

while $\beta_k \neq 0$

if $k \neq 0$

for $i = 1 : n$

$t = w_i$; $w_i = \frac{v_i}{\beta_k}$; $v_i = -\beta_k t$

end

end

$v = v + A.mult(w)$

$k = k + 1$; $\alpha_k = w^T v$; $v = v - \alpha_k w$; $\beta_k = ||v||_2$

end

4.4 Generating a feasible starting point for LP relaxations

We now illustrate how we generate a strictly feasible starting point for the max cut problem. The technique is similar to the one used in [42]. For convenience we will consider the max cut LP relaxation $LP1$. The primal and dual relaxations are (3.7) and (3.8) respectively.

We generate a strictly feasible starting point for (3.7) by choosing $\mathcal{A}^T y + C$ to be positive definite. This can be done by appropriately increasing y, and checking the positive definiteness of $S = \mathcal{A}^T y + C$ via a Cholesky factorisation. This y is strictly feasible for (3.7).

To generate a strictly feasible starting point for (3.8), we observe that we can rewrite (3.8) as

$$\begin{array}{lll} \max & \sum_{i=1}^{n} \frac{L_{ii}}{4} x_i + c^T w & \\ \text{s.t.} & \left[\begin{array}{cc} I & A \end{array} \right] \left[\begin{array}{c} x \\ w \end{array} \right] & = \quad e \\ & \left[\begin{array}{c} x \\ w \end{array} \right] & \geq \quad 0 \end{array}$$

Here $x \in R^n$, $w \in R^q$ and I,A are the $n \times n$ identity and $n \times q$ matrices respectively.

Set $a_0 = Ae$ i.e. sum all the columns corresponding to w in (3.7).

Thus we have replaced the vector w with a single variable x_0. Now generate a strictly feasible starting point by solving the following LP

$$\begin{array}{lll} \max & & x_0 \\ \text{subject to} & \left[\begin{array}{cc} I & a_0 \end{array}\right] \left[\begin{array}{c} x \\ x_0 \end{array}\right] & = \quad e \\ & \left[\begin{array}{c} x \\ x_0 \end{array}\right] & \geq \quad 0 \end{array}$$

The optimal solution to this LP is obtained, via a minimum ratio test, by setting $x_0 = \min_{i=1,\ldots,n}(\frac{1}{a_0(i)})$. Hence to generate a strictly feasible starting point we set $x_0 = \mu \min_{i=1,\ldots,n}(\frac{1}{a_0(i)})$ where $\mu \in (0,1)$ and then assign this value x_0 for all the variables in w and then compute $x_i = 1 - a_0 x_0, i = 1, \ldots, n$. This gives a strictly feasible starting point to (3.8).

4.5 Dropping constraints

We are yet to develop an effective criteria for dropping constraints in our cutting plane scheme. However the following criterion due to Atkinson and Vaidya [3] is particularly useful. We explain this below.

If we have a bounded full dimensional polytope $P := \{x \in \mathcal{R}^n : Ax \geq b\}$. Here $A \in \mathcal{R}^{m \times n}, b \in \mathcal{R}^m$. The convex logarithmic barrier function over P is the function

$$F(x) \quad = \quad -\sum_{i=1}^{m} \log(a_i^T x - b_i) \tag{4.1}$$

Here a_i is the ith row of matrix A. We represent each constraint $a_i^T x - b_i \geq 0$ by means of the pair (a_i, b_i).

Consider the following merit function

$$\sigma_i(x) \quad = \quad \frac{a_i^T (\nabla^2 F(x))^{-1} a_i}{(a_i^T x - b_i)^2}, \quad 1 \leq i \leq m \tag{4.2}$$

We set $\kappa(a_i, b_i) = a_i^T x - b_i$ i.e. the slack, when the hyperplane corresponding to the pair (a_i, b_i) is first introduced.

We also define $\mu_i(x) = \frac{(a_i^T x - b_i)}{\kappa(a_i, b_i)}, i = 1, \ldots, m.$

We are now ready to describe the criterion in [3] for dropping constraints. Consider a typical iteration k.

1. If there are no indices j such that $\mu_j(x) > 2$, then no constraints are dropped in iteration k.

2. If there is an index j such that $\mu_j(x) > 2$ and $\sigma_j(x) < 0.04$, then the jth hyperplane is unimportant. Drop (a_j, b_j) from the set of hyperplanes defining P. Else choose all hyperplanes (a_j, b_j) with $\sigma_j(x) >= 0.04$. These are still important. However reset their $\kappa(a_j, b_j)$ values as $\kappa(a_j, b_j) = a_j^T x - b_j$ i.e. set κ to the current slack.

Note that estimating the $\sigma_i(.)$'s requires computing

$$\nabla^2 F(x) \;=\; \sum_{i=1}^{m} \frac{a_i a_i^T}{s_i^2} \;=\; A^T S^{-2} A \tag{4.3}$$

We need $(\nabla^2 F(x))^{-1}$ which is fairly expensive to compute in each iteration. However note that we are computing a factorisation of a matrix $(A^T D^{-2} A)$ where $D = \text{Diag}(x./s)$, while solving each LP relaxation using an interior point scheme. We could use this factorisation, instead of explicitly computing $\nabla^2 F(x)$ using (4.3). This should save us time.

We find the criterion in [3] to be fairly restrictive, with the result that few, if any hyperplanes are dropped in each iteration. Thus we consider the only the value of $\mu_j(x)$ in dropping constraints. If this value is greater than 2 then constraint j is dropped, else it is retained.

4.6 Computational results

We present some preliminary results with the cutting plane scheme on the max cut problem (1.10).

Table 4.1 compares the LP relaxation with the bundle and box constraints discussed in section 5 called the bundle LP, with the cutting plane LP relaxation introduced in section 6, called the interior LP For the cutting plane approach the

Name	SDP	Bundle LP	Interior LDR	Interior LPR	Eig	Iter
mcp100 [2]	226.16	225.85	226.13	226.11	-1.1e-3	200
mcp124-1 [2]	141.99	141.95	141.94	141.88	-1.6e-3	185
mcp124-2 [2]	269.88	269.21	269.76	269.72	-2.8e-3	200
mcp124-3 [2]	467.75	467.44	467.40	467.35	-1.2e-2	200
mcp124-4 [2]	864.41	863.81	862.69	862.67	-3.7e-2	200
mcp250-1 [2]	317.26	317.21	316.95	316.91	-1.1e-2	200
mcp250-2 [2]	531.93	531.38	529.77	529.73	-2.7e-2	200
mcp250-3 [2]	981.17	980.48	976.98	976.92	-5.3e-2	200
mcp250-4 [2]	1681.96	1680.00	1675.23	1675.14	-8.5e-2	200
mcp500-1 [2]	598.15	596.67	594.28	586.68	-3.5e-2	100
mcp500-2 [2]	1070.06	1069.95	1059.51	1049.32	-5.3e-2	100
mcp500-3 [2]	1847.97	1844.13	1830.11	1823.00	-1.2e-1	100
mcp500-4 [2]	3566.74	3560.64	3534.11	3532.74	-1.8e-1	140
maxG11 [1]	629.16	626.99	627.31	619.98	-1.8e-2	100
toruspm-8-50 [1]	527.81	525.91	516.90	508.87	-6e-2	100

Table 4.1: Strengths of various LP relaxations

Name	m_{sb}	m_{icp}
mcp100 [2]	379	869
mcp124-1 [2]	293	718
mcp124-2 [2]	453	1026
mcp124-3 [2]	755	1221
mcp124-4 [2]	1405	1332
mcp250-1 [2]	591	1224
mcp250-2 [2]	876	1911
mcp250-3 [2]	1546	2312
mcp250-4 [2]	2684	2465
mcp500-1 [2]	1135	1626
mcp500-2 [2]	1736	1929
mcp500-3 [2]	2869	2253
mcp500-4 [2]	5635	3195
maxG11 [1]	2411	2610
toruspm-8-50 [1]	2064	2197

Table 4.2: Sizes of the various LP relaxations

initial LP relaxation LDR_1 is obtained by setting $y_i = \frac{L_{ii}}{4}$, $\forall i = 1, \ldots, n$. The other cutting plane parameters discussed in section 6 are $TOL = 1$, $s = 20$, $\mu = 0.95$ and $MAXITER = 100$. We used Zhang's *LIPSOL* [62], with a strictly feasible starting point to solve the sequence of interior LP's. Since we end up with a tolerance TOL of $(0.95)^{100} = 5e-3$ at the end of 100 iterations and the primal and dual objective values need not necessarily agree, we provide the objective value of the dual (LPR), which is always a lower bound on the SDP objective value. It is seen that the bundle LP provides a tighter relaxation than the interior LP for most instances, and is also the smaller LP relaxation. However in some of the smaller test cases such as *mcp124-1*, we are able to do better with the interior point code. The fact that the interior point LP is giving good SDP relaxations is reflected in the very small negative eigenvalues of the dual slack matrix S, that we encounter. For most cases this value is well below -0.1.

We also tried incorporating the cutting plane framework, where the sequence of LP's are solved using the *CPLEX 6.5* [28] simplex solver. The interior LP relaxation clearly outperforms the simplex LP relaxation, and we are able to generate better cutting planes using the former approach. Morever since we are solving the LP's initially to high tolerances TOL in the interior approach, we seem to get fairly quickly to the periphery of the positive semidefinite cone, reflected in the small negative eigenvalues we encounter. As a result we add not only stronger, but fewer cutting planes in the interior point cutting plane scheme as compared to the simplex cutting plane approach, with the result that the intermediate LP relaxations can be solved quickly.

4.7 Conclusions

In this chapter, we present a cutting plane LP approach to solving semidefinite programming problems. We compare it with the LP approach with the bundle and box constraints discussed in the previous chapter. It appears from the results in this chapter that

1. It is better to solve the LP relaxations approximately with an interior point approach, as compared to a simplex scheme since this results in better cutting

planes.

2. The LP approach with the box and bundle constraints generally outperforms the cutting plane approach. Besides the former is also smaller than the cutting plane LP. To make the cutting plane approach competitive with the former scheme, further refinements are required, and we are currently working on these.

3. We could in practice improve the LP relaxation with the box and bundle constraints, by incorporating it in the cutting plane scheme.

4. A major drawback in the cutting plane approach, is that the eigenvectors employed as d, result in dense constraints. This results in factorising a dense matrix $A^T D^{-2} A$ in the interior point scheme. To keep this computation time down, we have tried to drop constraints. We are also looking at other approaches such as the volume algorithm due to Barahona et al [4], as an alternative to the interior point LP approach.

5. Although the LP approach in chapter 3 and the cutting plane scheme discussed in chapter 4 provide excellent relaxations to the SDP problem, our aim is eventually to solve IP's (integer programs). These LP relaxations could help in fixing variables in a branch and bound approach to solving IP's.

CHAPTER 5
Future Directions

This thesis proposes two LP approaches to solving the semidefinite programming problem (SDP). The first LP approach employs the vectors in the bundle maintained by the spectral bundle approach, together with a set of problem specific box constraints. The second LP approach is a cutting plane LP approach, which solves SDP as a sequence of LP's, with each LP solved approximately with an interior point code.

1. The spectral bundle code is extremely time consuming on SDP's with a large number of constraints such as the Lovasz theta problem, where traditional interior point approaches tend to outperform the spectral bundle scheme. Our LP relaxations discussed in chapter 3, on the larger Lovasz theta problems do not compare favorably with the SDP relaxation. One way to improve the LP relaxation, is to incorporate it in the cutting plane approach discussed in chapter 4. We could also look at the second order bundle approach proposed by Oustry [47] and see if we can get tighter LP relaxations.

2. SDP's have become very popular lately, since they provide excellent approximation algorithms for various combinatorial optimisation problems. The main computational task in the approximation algorithm is to solve the SDP relaxation. Our LP approach enables us to solve SDP's that are inaccessible to the interior point methods due to their size. Nevertheless it would be interesting to compute the primal X matrix using our LP approach and employ it in a rounding scheme to observe the quality of the solutions obtained.

3. The cutting plane LP approach is applicable to all SDP's. Since we do not employ the bundle scheme, we do not need to make any assumptions, such as the constant trace on the primal feasible set here. However to make the cutting plane scheme competitive with interior point methods for SDP, several refinements other than those already described in chapter 4 are required.

4. The LP approaches discussed in chapters 3 and 4 can be utilised in fixing variables in a branch and bound technique, and the constraints used in our LP approaches could be used as cutting planes in branch and cut and cutting plane approaches to solving integer programming (IP) problems. This is important since we are eventually interested in solving problems such as max cut which are IP's, rather than just computing their SDP relaxations.

5. Recently Kojima and Tuncel [35, 36] have proposed two interesting methods, one based on successive semidefinite programming (SDP) relaxations and another based on successive LP relaxations, to solving optimisation problems on nonconvex sets F. Many optimisation problems, with nonlinear objective functions can be posed in this way. Since our scheme amounts to solving an SDP as a LP with constraints polynomial in the problem size, we can provide a way of combining their two approaches.

6. Recently Kim and Kojima [32] have proposed a second order cone programming [40] relaxation of nonconvex quadratic optimisation problems. The SOCP technique is a reasonable compromise between the effectiveness of the SDP relaxation and the low computational cost of the lift and project LP relaxation. We intend to investigate second order cone programming (SOCP) relaxations of semidefinite programming problems. These should be tighter relaxations than LP approaches for solving SDP's.

References

[1] Farid Alizadeh, *Interior Point Methods in Semidefinite Programming with applications to Combinatorial Optimisation*, SIAM Journal on Optimisation, 5 (1995), pp. 13-51.

[2] F. Alizadeh, J.P.A. Haeberly and M.L. Overton, *Primal-dual interior point methods for semidefinite programming : convergence rates, stability and numerical results*, SIAM Journal on Optimisation, 8(1998), pp. 746-768.

[3] D.S. Atkinson and P. Vaidya, *A cutting plane algorithm for convex programming that uses analytic centers*, Mathematical Programming, 69(1995), pp. 1-43.

[4] F. Barahona and R. Anbil, *The Volume Algorithm : producing primal solutions with a subgradient method*, IBM Research Report, Yorktown Heights, RC 21103(94395), February 1998.

[5] F. Barahona and R. Anbil, *On some difficult linear programs coming from Set Partitioning*, IBM Research Report, Yorktown Heights, RC 21410(96674), February 1999.

[6] S. Benson and Yinyu Ye, *Approximating Maximum Stable Set and Minimum Graph Coloring Problems with the Positive Semidefinite Relaxation*, Working Paper, University of Iowa, Iowa, 1999.

[7] S.J. Benson, Yinyu Ye and Xiong Zhang, *Solving Large-Scale Semidefinite Programs for Combinatorial Optimisation*, SIAM Journal on Optimisation, 10 (2000), pp. 443-461.

[8] Brian Borchers, *SDPLIB 1.2, A Library for Semidefinite Programming Test Problems*, June 1999.

[9] S. Boyd, L. El Ghaoui, E. Feron and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, SIAM Studies in Applied Mathematics, SIAM, Philadelphia, USA, 1994.

[10] S. Burer, R.D.C. Monteiro and Y. Zhang, *Solving Semidefinite Programs via Nonlinear Programming. Part I : Transformations and Derivatives*, Working paper, School of ISyE, Georgia Tech., Atlanta, GA, September 1999.

[11] S. Burer, R.D.C. Monteiro and Y. Zhang, *Solving Semidefinite Programs via Nonlinear Programming. Part II : Interior point Methods for a Subclass of SDP's*, Working paper, School of ISyE, Georgia Tech., Atlanta, GA, October 1999.

[12] J. Cullum, W.E. Donath and P. Wolfe, *The minimisation of certain nondifferentiable sums of eigenvalues of symmetric matrices*, Mathematical Programming Study, 3(1975), pp. 35-55.

[13] W.E. Donath and A.J. Hoffman, *Lower bounds for the partitioning of graphs*, IBM Jl. of Research and Development 17(1973), pp. 120-125.

[14] Alan Frieze and M.R. Jerrum, *Improved approximation algorithms for Max k-Cut and Max Bisection*, Algorithmica 18, pp. 61-77.

[15] M. Goemans and D.P. Williamson, *Improved approximation algorithms for max cut and satisfiability problems using semidefinite programming*, J. A.C.M 42 (1995), pp. 1115-1145.

[16] G.H. Golub and C.F. Van Loan, *Matrix Computations*, Third edition, The John Hopkins University Press, Baltimore, 1996.

[17] M. Grotschel, L. Lovasz and A. Schrijver, *Geometric Algorithms and Combinatorial Optimisation*, Springer-Verlag, New York, 1988.

[18] G. Gruber and F. Rendl, *Computational experience with Stable Set Relaxations*, Department of Mathematics, University of Klagenfurt, Austria, December 2000.

[19] Christoph Helmberg, *Semidefinite Programming*, ZIB-Report ZR-00-34, Konrad-Zuse-Zentrum Berlin, October 2000.

[20] Christoph Helmberg, *Semidefinite Programming Homepage*, *http://www.zib.de/helmberg/semidef.html*

[21] C. Helmberg and K.C. Kiwiel, *A Spectral Bundle Method with bounds*, ZIB Preprint SC-99-37, Konrad-Zuse-Zentrum Berlin, December 1999.

[22] C. Helmberg and F. Rendl, *A Spectral Bundle Method for Semidefinite Programming*, SIAM Journal on Optimisation, 10 (2000), pp. 673-696.

[23] C. Helmberg, F. Rendl, R. Vanderbei and H. Wolkowicz, *An interior point method for semidefinite programming*, SIAM Journal on Optimisation, 6(1996), pp. 673-696.

[24] Christoph Helmberg, *SBmethod : A C++ Implementation of the Spectral Bundle Method*, ZIB-Report ZR-00-35, Konrad -Zuse-Zentrum Berlin, October 2000.

[25] J.B. Hiriart Urruty and C. Lemarechal, *Convex Analysis and Minimisation Algorithms I*, Springer Verlag, Berlin, Heidelberg, 1993.

[26] J.B. Hiriart Urruty and C. Lemarechal, *Convex Analysis and Minimisation Algorithms II*, Springer Verlag, Berlin Heidelberg, 1993.

[27] Roger Horn and Charles Johnson, *Matrix Analysis*, Cambridge University Press, 1990.

[28] ILOG CPLEX 6.5, *ILOG Inc.*, 1080 Linda Vista Avenue, Mountain View, CA 94043.

[29] N.K. Karmarkar, *A new polynomial time algorithm for linear programming*, Combinatorica, 4(1984), pp. 373-395.

[30] Kartik Krishnan and John E. Mitchell, *A Linear Programming Approach to Semidefinite Programming Problems*, Dept. of Mathematical Sciences, RPI, Troy, NY, April 2001 (submitted to SIAM Journal on Optimisation).

[31] Kartik Krishnan and John E. Mitchell, *A Cutting plane LP approach to solving semidefinite programming problems*, Dept. of Mathematical Sciences, RPI, Troy, NY, April 2001.

[32] S. Kim and Masakazu Kojima, *Second Order Cone Programming Relaxation of Nonconvex Quadratic Optimisation Problems*, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, July 2000.

[33] K.C. Kiwiel, *Proximity Control in bundle methods for convex nondifferentiable minimisation*, Mathematical Programming 46 (1990), pp. 105-122.

[34] Jon Kleinberg and Michel Goemans, *The Lovasz Theta Function and a Semidefinite Programming Relaxation of Vertex Cover*, SIAM Journal on Discrete Mathematics, 11 (1998), pp. 196-204.

[35] Masakazu Kojima and Levent Tuncel, *Cones of Matrices and Successive Convex Relaxations of Nonconvex Sets*, SIAM Journal on Optimisation, 10(3), pp. 750-778.

[36] Masakazu Kojima and Levent Tuncel, *Discretisation and localisation in successive convex relaxation methods for nonconvex quadratic optimisation*, Mathematical Programming, 89(2000), pp. 79-111.

[37] *LAPACK User's Guide*, 3rd Edition, SIAM, Philadelphia (1999).

[38] A.S. Lewis and M.L. Overton, *Eigenvalue Optimisation*, Acta Numerica 5(1996), pp. 149-190.

[39] A. Lisser and F. Rendl, *Telecommunication Clustering using Linear and Semidefinite Programming*, Dept. of Mathematics, University of Klagenfurt, Austria, December 2000.

[40] M. Lobo, L. Vandenberghe, S. Boyd and H. Lebret, *Applications of Second Order Cone Programming*, Linear Algebra and its Applications, 284(1998), pp. 193-228.

[41] L. Lovasz, *On the Shannon capacity of a graph*, IEEE Transactions on Information Theory, 25(1979), pp. 1-7.

[42] J.E. Mitchell and M.J. Todd, *Solving combinatorial optimisation problems using Karmarkar's algorithm*, Mathematical Programming, 56(1992), pp. 245-282.

[43] John E. Mitchell, *Realignment in the NFL*, Dept. of Mathematical Sciences, RPI, Troy, NY, November 2000.

[44] R.D.C Monteiro, *Primal Dual path following algorithms for semidefinite programming*, SIAM Journal on Optimisation, 7(1997), pp. 663-678.

[45] G.L. Nemhauser, A.H.G. Rinnooy Kan and M.J. Todd, *Optimisation*, Elsevier Publishers B.V., Amsterdam, Holland, 1989.

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

[47] Francois Oustry, *A second order bundle method to minimise the maximum eigenvalue function*, Mathematical Programming 89 (2000), pp. 1-33.

[48] Gabor Pataki, *On the Rank of Extreme Matrices in Semidefinite Programs and the Multiplicity of Optimal Eigenvalues*, Mathematics of Operations Research 23(2), pp. 339-358.

[49] Gabor Pataki and Stefan Schmieta, *The DIMACS library of semidefinite -quadratic-linear programs*, Computational Optimisation Research, Columbia University, NY, November 1999.

[50] F. Rendl and H. Wolkowicz, *A Projection technique for partitioning the nodes of a graph*, Annals of Operations Research 58(1995), pp. 155-179.

[51] H. Schramm and J. Zowe, *A version of the bundle idea for minimising a nonsmooth function : Conceptual idea, convergence analysis, numerical results*, SIAM Journal on Optimisation 2 (1992), pp. 121-152.

[52] M.J. Todd, *Semidefinite Optimisation*, Survey paper, Department of Operations Research, Cornell University, NY, February 2001.

[53] M.J. Todd, *A study of search directions in interior point methods for semidefinite programming*, Optimisation Methods and Software, 12(1999) pp. 1-46.

[54] K.C. Toh, M.J. Todd and R.H. Tutuncu, *SDPT3 - A Matlab Software Package for semidefinite programming, version 2.1*, Optimisation Methods and Software, 11 (1999), pp. 1-49.

[55] M.J. Todd, K.C. Toh and R.H. Tutuncu, *On the Nesterov-Todd direction in semidefinite programming*, SIAM Journal on Optimisation, 8(1998), pp. 769-796.

[56] L. Vandenberghe and S. Boyd, *Semidefinite Programming*, SIAM Review, 38(1996), pp. 49-95.

[57] R. Vanderbei and H. Benson, *On Formulating Semidefinite Programming Problems as Smooth Convex Nonlinear Optimisation Problems*, ORFE-99-01, Princeton University, NJ, January 2000.

[58] Henry Wolkowicz, Romesh Saigal and Lieven Vandenberghe, *Handbook on Semidefinite Programming*, Kluwer Academic Publishers, 2000.

[59] Stephen Wright, *Interior Point Methods Online*, *http://www-unix.mcs.anl.gov/otc/InteriorPoint/*

[60] Yinyu Ye, *A .699 Approximation Algorithm for Max Bisection*, Mathematical Programming, ISSN : 0025-5610, 2000.

[61] Yinyu Ye, *Approximating quadratic programming with bound and quadratic constraints*, Mathematical Programming, 84 (1999), pp. 219-226.

[62] Yin Zhang, *Solving Large-Scale Linear Programs by Interior Point Methods Under the MATLAB Environment*, Technical Report, Department of

Computational and Applied Mathematics, Rice University, Houston, Texas, 1997, pp. 1-26.

[63] Y. Zhang, *On extending some primal-dual interior point algorithms from linear programming to semidefinite programming*, SIAM Journal on Optimisation, 8(1998), pp. 365-386.