SF2827: Topics in Optimization

An Overview of Cutting Plane Methods for Semidefinite Programming

Supervisor:
Anders Forsgren
Department of Mathematics
Royal Institute of Technology
(KTH)
andersf@kth.se

Author:
Luca Furrer
luca.furrer@epfl.ch

May 2009

Abstract

Cutting plane methods are an alternative to interior methods to solve semidefinite programming (SDP) problems. In this paper, we are going to present five different cutting plane algorithms for semidefinite programming. The methods use relaxations and the semi-infinite representation of SDP's. The presented methods are the polyhedral cutting plane method, the polyhedral bundle method, the block diagonal cutting plane method and the primal active set approach. We present also a generic cutting plane methods which tries to unify all the other methods.

In the second part the implementation of the polyhedral cutting plane method into MATLAB is presented followed by some test examples.

Contents

1 Introduction

This paper was written in the context of the course ”Topics in Optimization” by Anders Forsgren at the Royal Institute of Technology of Stockholm, KTH, during the authors exchange year at this school during 2008/2009. The course treated semidefinite programming, an interesting domain in optimization. There existes literature concerning this subject e.g. the surveys of Laurent and Rendl [9] and Todd [14]. There exist efficient interior point methods to solve such program but they have problem size limitation and they are not so good to warm start. However there are also cutting plane methods available. In Krishnan and Mitchell [8] several are presented.

This paper presents different cutting plane methods for semidefinite programming as well as a generic algorithm for the different cutting plane methods. There exists methods using a relaxation of the transformation to a semi-infinite program, methods using the idea of bundle methods as well as one which mimics the simplex method for linear programming. The discussed methods are the polyhedral methods polyhedral cutting plane method, the polyhedral bundle method and the non-polyhedral methods spectral bundle method, block diagonal cutting plane method and the primal active set method. In the conclusion we compare the different methods.

There finds also the description of the implementation of the polyhedral cutting plane method in MATLAB. This implementation was tested with some problems in particular an ellipsoid optimization problem.

2 Semidefinite Programming

Let us introduce the notations first. If A and B are two symmetric matrices of size $n \times n$, i.e. $A, B \in \mathcal{S}^n$, then $\cdot \cdot \cdot$ denotes the Frobenius inner product. So

$$A \cdot B = tr(A^T B) = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} B_{ij}.$$

We use also the corresponding norm

$$||A||_F = \sqrt{tr(A^T A)}.$$

To denote that A is definite or positive semidefinite we use $A \succ 0$ respectively $A \succeq 0$. So let us consider the following SDP problem

$$\begin{array}{ll} \text{minimize} & C \cdot X \\ \text{subject to} & \mathcal{A}(X) = b, \quad (SDP) \\ & X \succeq 0. \end{array}$$

This problem has the dual

$$\begin{array}{ll} \text{maximize} & b^T y \\ \text{minimize} & \mathcal{A}^T y + S = C, \quad (SDD) \\ & S \succeq 0. \end{array}$$

with $X, S, C \in \mathcal{S}^n$ and $y, b \in \mathbb{R}^m$, where m is the number of constraints. Here C and b are given by the problem and X, S and y are the unknown variables. The notation $\mathcal{A}$ stands for a linear operator

$$\mathcal{A} : \mathcal{S}^n \to \mathbb{R}^m$$

with an adjoint

$$\mathcal{A}^T : \mathbb{R}^m \to \mathcal{S}^n.$$

Those two operators have this form:

$$\mathcal{A}(X) = \begin{bmatrix} A_1 \cdot X \\ \vdots \\ A_m \cdot X \end{bmatrix}$$

and

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

where A_i, $i = 1, \ldots, m$, are given problem parameters.

To apply primal-dual based algorithms to solve an semidefinite programming problem, it is often required that strong duality holds, i.e. that the optimal objective value of (SDP) is equal to the one of (SDD).

Theorem 1. *If the set of solutions to (SDP) and the set of solutions to (SDD) such that $S \succ 0$ are non-empty then the duality gap is zero, i.e. the optimal values of (SDP) and (SDD) are equal.*

A proof of this theorem can be found in Todd [14, Chapter 4]. The following two theorems are direct consequences of the one before.

Theorem 2. *If the set of solutions to (SDP) such that $X \succ 0$ and the set of solutions to (SDD) are non-empty then the duality gap is zero, i.e. the optimal values of (SDP) and (SDD) are equal.*

Theorem 3. *If (SDP) and (SDD) have both strictly feasible solutions then strong duality holds*

The last two theorems are direct consequences of Theorem 1.

To read more about semidefinite programming in general there existe some literature among others Laurent and Rendl [9] and Todd [14]. For more information for interior methods there finds among others the papers of Helmberg, Rendl, Vanderbrei and Wolkowicz [3], Monteiro [11] and Zhang [16].

3 Semi-infinite Programming Formulation

Sometimes it would be interesting to avoid the notation of positive semidefinite in a (SDP) problem. Since we have that $X \succeq 0$ is equivalent to

$$d^T X d = dd^T \cdot X \geq 0 \quad \forall d \in \mathbb{R}^n$$

Therefore it is possible to formulate (SDP) and (SDD) as a semi-infinite programs.

$$\begin{array}{lll} \text{minimize} & C \cdot X & \\ \text{subject to} & \mathcal{A}(X) = b & (SIP) \\ & d^T X d \geq 0 \quad \forall ||d||_2 = 1 & \end{array}$$

$$\begin{array}{lll} \text{maxmimize} & b^T y & \\ \text{subject to} & \mathcal{A}^T y + S = C & (SID) \\ & d^T S d \geq 0 \quad \forall ||d||_2 = 1 & \end{array}$$

with $d \in \mathbb{R}^n$. This subject and the solutions of such problems is discussed in Krishnan and Mitchell [7]. To solve such problems we present to theorems which are both found in [7], where one find also their proofs which are omitted in this document.

Theorem 4. *If the solution of (SIP) is finite and the objective value of (SIP) and (SID) are the same. Then (SID) has a finite discretization (SIR) with the same optimal value.*

Hence it is possible to find a subset of vectors $d_i, i = 1, \ldots, m$, such that we obtain linear program to solve.

$$\begin{array}{lll} \text{maxmimize} & b^T y & \\ \text{subject to} & \mathcal{A}^T y + S = C & (SIR) \\ & d_i^T S d_i \geq 0 \quad i = 1, \ldots, m & \end{array}$$

The choice of vectors is not obvious but there exists another theorem which reduces the number of vectors needed.

Theorem 5. *If there exists an discretization of (SID) which has the same objective value as (SID) then there exists a discretization (SIR) with $m \leq k$ where m is the size of the vector set and k is the number of constraint matrices A_i.*

So it is possible to find a (SIR) which is a linear problem and has limited number of constraints but it is not obvious how to find the set. Some approaches are presented in the next chapters which discuss several algorithms to solve semidefinite programming problems with a cutting plane approach.

4 Polyhedral Cutting Plane Algorithm

As first algorithm we consider an algorithm called the polyhedral cutting plane algorithm. The polyhedral cutting plane method was introduced to semidefinite programming by Krishnan and Mitchell [5] as well as by Goldfarb [1]. Actually Goldfarb considers conic programming, where semidefinite programming is a special case.

In order to use this method we add two assumptions to basic semidefinite programming problem.

Assumption 1

$$\mathcal{A}(X) = b \Rightarrow tr(X) = a \text{ for some constant } a.$$

Assumption 2 (SDP) and (SDD) both have strictly feasible solutions. This means that $\{X \in \mathcal{S}^n : \mathcal{A}(X) = b, X \succ 0\}$ and $\{(y, S) \in \mathbb{R}^k \times \mathcal{S}^n : \mathcal{A}y + S = C, S \succ 0\}$ are non-empty.

The first assumption allows us to transform (SDD) to an eigenvalue optimization problem. This assumption is required for the bundle method to create linear constraints. The second assumption assures that (SDP) and (SDD) have an optimal solution and that the optimal values are equal. Therefore we have also a duality gap equal to zero.

As already discussed in section it is possible to reformulate (SDD) to a semi-infinite program (LDD).

$$\begin{array}{ll} \text{maximize} & b^T y \\ \text{subject to} & dd^T \cdot (C - \mathcal{A}^T y) \geq 0 \quad \forall d \quad (LDD) \end{array}$$

We want to consider a relaxation of this problem. We consider a finte set of variables $\{d_i, i = 1, \ldots, m\}$ at the place of d. Hence we get the following problem:

$$\begin{array}{ll} \text{maximize} & b^T y \\ \text{subject to} & d_i d_i^T \cdot (C - \mathcal{A}^T y) \geq 0 \text{ for } i = 1, \ldots, m \quad (LDR) \end{array}$$

A reason why we work with with the relaxation of (SDD) instead of (SDP) is that this problem has m variables but (SDP) leads to $n(n-1)/2$ variables and most of the time we have that $m < n(n-1)/2$.

When we consider the constraints of (LDR), we find

$$d_i d_i^T \cdot \mathcal{A}^T y = d_i d_i^T \cdot \left(\sum_{j=1}^{k} y_j A_j \right) = \sum_{j=1}^{k} y_j d_i^T A_j d_i$$

and we see that the constraints are equivalent to

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

And therefore we find the the dual of (LDR)

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{m} d_i^T C d_i x_i \\ \text{subject to} & \sum_{i=1}^{m} d_i^T A_j d_i x_i = b_j \quad \forall j = 1, \ldots, m \\ & x \geq 0 \end{array}$$

which can be rewritten as the following problem

$$\begin{array}{ll} \text{minimize} & C \cdot \sum_{i=1}^{m} x_i d_i d_i^T \\ \text{subject to} & \mathcal{A} \left(\sum_{i=1}^{m} x_i d_i d_i^T \right) = b \quad (LPR) \\ & x \geq 0 \end{array}$$

The following lemma helps us to compare the solution of (LPR) to the solution of (SDP).

Lemma 1. *A feasible solution x to (LPR) gives a feasible solution X to (SDP)*

Proof. Let $X = \sum_{i=1}^{m} x_i d_i d_i^T$. From (LPR) we have that X satisfies the condition $\mathcal{A}X = b$. So we have also to prove that X is semi-definite positive.

$$d^T X d = d^T \left(\sum_{i=1}^{m} x_i d_i d_i^T \right) d = \sum_{i=1}^{m} x_i (d_i^T d)^2 \geq 0$$

for all d since $x \geq 0$. □

With help of this lemma, we can deduce an upper bound for the solution of (SDP) with help of the optimal value of (LDR). So we get the following algorithm:

Algorithm 1. *1. (Initialization) Choose an initial set of constraints for (LDR) and the termination parameters $\epsilon_1, \epsilon_2 > 0$. Set the current lower and upper bounds to $LB = -\infty$ and $UB = \infty$. Choose also TOL as the tolerance to solve (LDR) and (LPR) as well as μ to update the tolerance if wanted.*

2. In the k'th iteration, obtain a solution y^k to the problem (LDR) by using an interior method and tolerance TOL. Update the upper bound $UB = \min\{UB, b^T y^k\}$.

3. Compute $\lambda = |\lambda_{min}(C - \mathcal{A}^T y^k)|$ and a corresponding eigenvector d. Update $LB = \max\{LB, b^T y^k + \lambda a\}$ where a comes from assumption 1. If $|LB - UB| \leq \epsilon_1$ or $\lambda \leq \epsilon_2$, got to step 5.

4. Add the constraint

$$dd^T \cdot \mathcal{A}^T y \leq dd^T \cdot C$$

to (LDR). Set $k = k + 1$ and $TOL = \mu TOL$

5. The current solution (x^k, y^k) for (LDR) and (LPR) gives an optimal solution X for (SDP) and y for (SDD) (see lemma 1).

Note It is also possible to add s constraints to (LDR). In this case use the s most negative eigenvalues and the corresponding eigenvector. But in this case it is sometimes necessary to drop some constraints. See Krishnan [6].

5 Block Diagonal Cutting Plane Scheme

The non-polyhedral block diagonal cutting plane algorithm is similar to the algorithm discussed in section 4. Actually this algorithm can nearly be considered as an generalization of the polyhedral cutting plane algorithm. This is not exactly the case but it is possible to find the connection of those two algorithms. It is discussed in Oskoorouchi and Goffin [12]. There is also a short overview in Krishnan and Mitchell [8].

Let us denote the multiplicity of $\lambda_{min}(C - \mathcal{A}^T y)$ by r. The main idea is now to add the following semidefinite constraints

$$\sum_{i=1}^{m} y_i (D^T A_i D) \succeq (D^T C D),$$

where $D \in \mathbb{R}^{n \times r}$, with $D^T D = I_r$, whose columns form an eigenbasis for the eigenspace of $C - \mathcal{A}^T y$ with eigenvalue $\lambda_{min}(C - \mathcal{A}^T y)$, instead of

$$\sum_{i=1}^{m} y_i (d_j^T A_i d_j) \succeq (d_j^T C d_j) \quad j = 1, \ldots, r$$

to the considered problem. So actually d_j is included as a row in the matrix D. So we can form our problem.

$$\begin{array}{lll} \text{maximize} & b^T y & (LDR') \\ \text{subject to} & \displaystyle\sum_{i=1}^{m} y_i (D_j^T A_i D_j) \succeq (D_j^T C D_j), \quad j = 1, \ldots, k & \end{array}$$

The dual of the changed relaxation dual, and therefor the primal of the changed problem is:

$$\begin{array}{ll} \text{minimize} & C \cdot \left(\displaystyle\sum_{i=1}^{k} D_i V_i D_i^T \right) \\ \text{subject to} & \mathcal{A} \left(\displaystyle\sum_{i=1}^{k} D_i V_i D_i^T \right) = b \quad (LPR') \\ & V_i \succeq 0, \quad i = 1, \ldots, k. \end{array}$$

With the knowledge of (LPR') and (LDR') it is possible to formulate the algorithm.

Algorithm 2. *1. Choose the initial set of constraints for (LDR'), the tolerance TOL to solve (LDR') and (LPR') and the termination parameters ϵ_1 and $\epsilon_2 > 0$. Set the lower and upper bound to $-\infty$ respectively ∞. If desired choose an update parameter μ for the tolerance.*

2. Find an approximate solution with tolerance TOL to (X^k, y^k) to (LDR') and (LPR').

3. Compute $\lambda = \lambda_m in(C - \mathcal{A}^T y^k)$ and an orthonormal matrix $D^k \in \mathbb{R}^{n \times r^k}$. r^k is the multiplicity of the eigenvalue. Update $LB = \max\{LB, b^T y^k + \lambda a\}$ and $UB = \min\{UB, b^T y^k\}$.

4. If $|UB - LB| \leq \epsilon_1$ or $|\lambda| \leq \epsilon_2$ then we have found an optimal solution. Stop. Otherwise add the following constraint to (LDR')

$$\sum_{i=1}^{m} y_i (D_k^T A_i D_k) \succeq (D_k^T C D_k).$$

5. Set $k = k + 1$, $TOL = \mu TOL$ and go to step 2.

One may recognise that this algorithm is the same as polyhedral cutting plane algorithm of section 4 as long the multiplicity of λ is one. So one may see the connection of those two algorithms. It is also possible to get (X^k, y^k) with help of the analytic center. This is well discussed in [12].

6 Polyhedral Bundle Scheme

Another way to improve the polyhedral cutting plane algorithm is using proximal bundle as for example presented in Kiwiel [4]. We assume again that Assumptions 1 and 2 hold.

The idea is to maximize $b^T y - u/2||y - y^k||$ instead of maximizing $b^T y$. With this method we penalize big steps away from the current iterate y^k. With a small u we penalize big steps less than with a big u. If u is too big, the algorithm is forced to perform steps which are close to null steps, which is not that useful. There are many proposed choices for u. But all of them keep the solution bounded.

By changing the objective function as describen get a changed version of (LPR) from section 4.

$$\begin{array}{ll} \text{maximize} & b^T y - \frac{u}{2}||y - \hat{y}||^2 \\ \text{subject to} & d_i d_i^T \cdot (C - \mathcal{A}^T y) \geq 0 \text{ for } i = 1, \ldots, m \quad (BD) \end{array}$$

There is only a change in the objective function compared to (LPR). The constraints are staying the sames.

We get also the Lagrangian dual

$$\begin{array}{lll} \text{minimize} & \frac{1}{2u}||b - \mathcal{A}(X)||^2 - b^T \hat{y} - (C - \mathcal{A}^t \hat{y}) \cdot X & \\ \text{subject to} & X = \sum_{i=1}^{k} x_i d_i d_i^T & (BP) \\ & \sum_{i=1}^{k} x_i = 1 & \\ & x_i \geq 0, \quad i = 1, \ldots, k. & \end{array}$$

Due to strong dualtiy (BD) and (BP) have the same objective value. Therefore we have the relation

$$y = \hat{y} + \frac{1}{u}(b - \mathcal{A}(X)). \tag{1}$$

Algorithm 3. *1. Let $y^1 \in \mathbb{R}^m$, let p be the normalized eigenvector corresponding to $\lambda_{min}(C - \mathcal{A}^T y^1)$. Also choose the weight $u > 0$, an improvement parameter $\nu \in (0, 1)$ and finally a termination parameter $\epsilon > 0$*

2. *At the k'th iteration, compute* X^{k+1} *from (BP) and* y *with help of equation* (1) *where* $\hat{y} = y^k$. *Also let* $f_{X^{k+1}}(y^{k+1}) = b^T y^{k+1} + (C - \mathcal{A}^t y^k) \cdot X^{k+1}$

3. *If*
$$f_{X^{k+1}}(y^{k+1}) \leq \epsilon$$
stop.

4. *Compute* $f(y^{k+1})$ *and the eigenvector* p^{k+1} *corresponding to* $\lambda_{min}(C - \mathcal{A}^T y^{k+1})$.

5. *If*
$$f(y^{k+1}) \geq \nu(f_{X^{k+1}}(y^{k+1})$$
then perform a serious step, i.e. $\hat{y}^{k+1} = y^{k+1}$. *Else perform a null step, i.e.* $y^{k+1} = y^k$

6. *Let* $k = k + 1$ *and return to step 2.*

7 Spectral Bundle Method

In this section we discuss the spectral bundle method due to Helmberg and Rendl [2]. Krishnan and Mitchell [8] also discuss this method briefly. It also uses the bundle method. But the big difference is that we transform the problem to a eigenvalue optimization problem and solve it. It is considered to be a good solver.

The considered problem is (SDP) with dual (SDD). We assume also that strong duality holds and therefore for any optimal solution X^* to (SDP) and any optimal solution Z^* to (SDD) it holds that

$$X^* Z^* = 0.$$

Further we assume that assumption 2

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

holds.

Actually we can add $tr(X) = a$ as a redundant constraint to (SDP). From this new problem with the added constraint we get a new dual (SDD') to (SDP).

$$\begin{array}{ll} \text{maximize} & a\lambda + b^T y \\ \text{subject to} & Z = \mathcal{A}^T y + \lambda I - C \succeq 0 \quad (SDD') \end{array}$$

Since we have $a > 0$ X cannot be zero at the optimum and therefore any optimal Z must be singular. Hence all dual optimal solutions Z satisfy $0 = \lambda_{min}(-Z)$ which leads to $\lambda = \lambda_{min}(C - \mathcal{A}^T y)$. Thus it follows that (SDD) and (SDD') are equivalent to the problem

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

To avoid complicated and inconvenient notations we set $a = 1$. And we get a eigenvalue problem

$$\min \lambda_{min}(C - \mathcal{A}^T y) + b^T y \quad (E)$$

which is a well know und studied problem. We are mainly interested in the function

$$f(y) = \lambda_{min}(C - \mathcal{A}^T y) + b^T y$$

Now we want to majorize f by $\hat{f}$. We set

$$\hat{f}(y) = \min\{(C - \mathcal{A}^T y) \cdot W : W \in \widehat{W}\}$$

where

$$\widehat{W} = \{\alpha \overline{W} + PVP^T : \alpha + tr(V) = 1, \alpha \geq 0, V \succeq 0\}$$

with $P \in \mathbb{R}^{n \times r}$ and $P^T P = I$ and $\overline{W} \succeq 0$ and $tr(W) = 1$.

Using the idea of proximal bundle we get the algorithm. The variable u is at free choice.

Algorithm 4. *1. (Initialization) Start with an initial point $y^0 \in \mathbb{R}^m$, a normalized eigenvector v^0 for $\lambda_{min}(C - \mathcal{A}^T y)$, an $\epsilon > 0$ for termination, an improvement parameter $\nu \in (0, \frac{1}{2})$, a weight $u > 0$. We set as well $k = 0$, $x^0 = y^0$, $P^0 = v^0$ and $\overline{W}^0 = v^0 {v^0}^T$*

2. In the k'th iteration solve

$$\max_{W \in \widehat{W}^k} (C - \mathcal{A}x^k) \cdot W^k + b^T x^k - \frac{1}{2u}(\mathcal{A}(W^k) - b) \cdot (\mathcal{A}(W^k) - b)$$

to get $y^{k+1} = x^k + \frac{1}{u}(b - \mathcal{A}(W^{k+1}))$ Decompose V^* into $V^* = Q_1 \Lambda_1 Q_1^T + Q_2 \Lambda_2 Q_2^T$. Compute

$$\overline{W}^{k+1} = \frac{1}{\alpha^* + tr(\Lambda_2)} \left(\alpha^* \overline{W}^k + P^k Q_2 \Lambda_2 (P^k Q_2)^T \right).$$

3. Compute $\lambda_{min}(C - \mathcal{A}^T y^{k+1})$ and the corresponding eigenvector v^{k+1}. Compute P^{k+1} by searching for the orthonormal basis to $P^k Q_1 v^{k+1}$

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

5. If

$$f(y^{k+1}) \leq f(x^k) - \nu(f(x^k) - \hat{f}^k(y^{k+1}))$$

then set $x^{k+1} = y^{k+1}$. Otherwise set $x^{k+1} = x^k$

6. $k = k + 1$ and go to step 2.

One may see that the similarities between this algorithm and the polyhedral bundle algorithm. But it is also obvious that the problems to solve in the steps are different.

8 Primal Active Set Approach

There exists also a primal active set approach to solve semidefinite programming problems as a sequence of smaller SDP's. As the simplex method for linear programming it uses the notation of extrem points. Unfortunately the number of documentations on this algorithm is rather small and unavailable, therefore we present this algorithm as it has been done by Mitchell and Krishnan in [8]. There are some similarities to the other algorithms but the simplex approach let it move a little bit away from the other algorithms.

Algorithm 5. *1. Consider an extreme point solution $X^1 = P^1 V^1 {P^1}^T$. Set the lower and upper bound to $-\infty$ respectively ∞ and choose an $\epsilon > 0$ for precision.*

2. In the k'th iteration, choose a subset of m independent equations from

$$\begin{aligned}\bar{S}_{11} = P_1^T(C - \mathcal{A}^T y)P_1 = 0\\ \bar{S}_{12} = P_1^T(C - \mathcal{A}^T y)P_2 = 0\end{aligned}$$

Solve the resulting system $\bar{S}_B = 0$ for y^{k+1}

3. If $S^{k+1} = (C - \mathcal{A}^T y^{k+1}) \succeq 0$, stop. Otherwise update $\bar{P}^{k+1}$ to be the columns of $\bar{S}_B = 0$. Compute the normalized eigenvector of $\lambda_{min}(S^{k+1})$ p^{k+1}. Set $\bar{P}^{k+1} = orth|\bar{P}^{k+1}, p^{k+1}|$. Update $LB = \max\{LB, b^T y^k + \lambda a\}$ and $UB = \min\{UB, b^T y^k\}$. If $|UB - LB| < \epsilon$ stop.

4. Solve

$$\begin{array}{ll} \textit{minimize} & ({\bar{P}^{k+1}}^T C \bar{P}^{k+1}) \cdot V \\ \textit{subject to} & ({\bar{P}^{k+1}}^T A_i \bar{P}^{k+1}) \cdot V = b_i \quad i = 1, \ldots, m \\ & V \succeq 0 \end{array}$$

Let $V^{k+1} = R_1^{k+1} M^{k+1} {R_1^{k+1}}^T$ with $M^{k+1} \succ 0$. Set $P_1^{k+1} = \bar{P}^{k+1} R_1^{k+1}$ and $X^{k+1} = P_1^{k+1} M^{k+1} {P_1^{k+1}}^T$.

5. If X^{k+1} is not an extrem point, use the Algorithm 1 in chapter 4 of Pataki [13] and return to step 1.

9 Generic Cutting Plane Method

There have been several cutting plane methods presented before in this paper. Now we want to describe a generic cutting plane method as done in Krishnan and Mitchell [8]. We assume that Assumptions 1 and 2 hold. For simplicity we consider the special case of Assumption 1 that

$$\mathcal{A}(X) = b \Rightarrow tr(X) = 1$$

There is on one side the semi-infinte approach as discussed in Section 4 where we transform (SDD) to this semi-infinite problem

$$\begin{array}{ll} \max & b^T y \\ \text{s.t.} & dd^T \cdot \mathcal{A}y \leq dd^T \cdot C \quad \forall ||d||_2 = 1 \quad (LDM). \end{array}$$

The advantage of the (LDM) formulation is that we can reduce the number of involved variables.

We construct a relaxation of (LDM) by considering a discretization of $||d||_2 = 1$ by a finite set of vectors $\{d_1, i = 1, \ldots, k\}$. We obtain the following discretization of (LDM):

$$\begin{array}{ll} \max & b^T y \\ \text{s.t.} & d_i d_i^T \cdot \mathcal{A}y \leq d_i d_i^T \cdot C \quad \text{for } i = 1, \ldots, k \quad (LDR) \end{array}$$

with the dual

$$\begin{array}{ll} \min & C \cdot \left(\sum_{i=1}^{k} x_i d_i d_i^T \right) \\ \text{s.t.} & \mathcal{A}\left(\sum_{i=1}^{k} x_i d_i d_i^T \right) = b \quad (LPR) \end{array}$$

We used as well the bundle methods. So finally we get a generic cutting plane algorithm.

Algorithm 6. *1. Choose an inital point $\hat{y}$, an initial finite set $\mathcal{D} = \{D_i\}$ and a scalar $u \geq 0$.*

2. Solve the subproblem

$$\begin{array}{ll} \textit{maximize} & \lambda + b^T y - \frac{u}{2}||y - \hat{y}||^2 \\ \textit{subject to} & D_i^T (C - \mathcal{A}^T y) D_i \succeq \lambda I, \quad i \in \mathcal{D} \end{array}$$

to get (y^*, λ^*).

3. *If $S^* = (C - \mathcal{A}^T y^*) \succeq 0$, we are optimal. Stop. Otherwise find $D \in \mathbb{R}^{n \times r}$, with $r \leq \sqrt{2m}$ such that $D^T(C - \mathcal{A}^T y^*)D \not\succeq 0$.*

4. *Add D to $\mathcal{D}$ or aggregate D into $\mathcal{D}$*

5. *Set $\hat{y} = y^*$ and return to step 2*

The vector λ in the subproblem refers to $tr(X) = 1$. One recognizes that for $u = 0$ we get to the algorithms which do not use the proximal bundle methods.

10 Conclusion

We presented several cutting plane methods which are the polyhedral cutting plane method, the polyhedral bundle method, the spectral bundle method, the block diangonal cutting plane method and the primal active set method. This methods are all working with relaxations of (SDP) and (SDD). We saw that they have all an algorithm which is similaire and presented it in Section 2.

But we did not compare the different algorithms. The algorithm in Section 4 can be implemented in polynomial time. There are computational results for this algorithm available in [5] and [6]. This algorithm can be compared with interior point methods even if his convergence is rather poor.

Algorithm 2 can also be implemented in polynomial time. After [8] the spectral bundle algorithm described in Section 7 showed some good results for large problems which could hardly been solved by interior methods. Hence it is a interesting algorithm.

The method in Section 1 tries to apply the simplex method to semidefinite programming but it does not find always directly an extrem point and therefore it is necessary to use another algorithm to find them. The computational performance of this algorithm is apparently still under discussion.

There are some cutting plane algorithms which are interesting in view of computational performance. It is possible to warm start them and at least one is very efficient for large semidefinite programs.

11 Implementation

In order to solve semidefinite programming problems with help of a computer, we decided to implement a cutting plane method. The chosen algorithm was the polyhedral cutting plane algorithm which is presented in Chapter 4. The algorithm was implemented in MATLAB with help of the modelling language YALMIP [10], which simplifies the handling of constraints and objective function and uses then a external solver such as the implemented solver of MATLAB or of another solver such as CDD. YALMIP is free of charge and openly distributed and therefore easy to access.

To implement the polyhedral cutting plane algorithm we followed Algorithm 1 with some slight changes which appeared due to problems during the implementation. The corresponding function in MATLAB is called `function [X,y]=cuttingplane(C, A, b, epsilon1, epsilon2)`. Where the returned values are corresponding to the primal and dual solution of (SDP) and (SDD). The input variables `C,A` and `b` are as in (SDP). `epsilon1` and `epsilon2` are the values for the stopping conditions denoted as ϵ_1 and ϵ_2 in the algorithm.

The function uses beyond the normal MATLAB functions

- YALMIP
- `function Ay=opAt(A,y)`.

Hence it is necessary to have YALMIP installed on the computer to use `cuttingplane`. The file `opAt.m`, which is part of this code for the algorithm, should also be saved in the same directory as `cuttingplane.m`. YALMIP is used to solve (LDR) which is a main step of the algorithm and was chosen because it allows to handle the constraints for a linear programming problem in a simple way which corresponds to the notation in the algorithm. The disadvantage is that it takes YALMIP some time to translate the constraints into the form required by the solver. But this is compensated that we do not have to this by ourself. `opAt` is a small function that calculates $\mathcal{A}^T y$.

Due to a unknown reason the stopping condition based on the lower and upper bounds as described in this document, where not working and the $|UB - LB|$ did not converge to zero. Therefore this stopping condition was changed to the condition $|y^k - y^{k-1}| \leq \epsilon_1$. The stopping condition on λ was not concerned. The algorithm seem to work well with this changed condition.

Another small change is that instead of let the user define the tolerance with which (LDR) is solved, the standard value of YALMIP was used and therefore it is also not possible to reduce the tolerance for solving (LDR).

One may remember that we made an assumption in order to use the polyhedral cutting plane algorithm which was

$$\mathcal{A}(X) = b \Rightarrow tr(X) = a \text{ for some constant } a.$$

In the implementation it was avoided to pass this a with the input variables to avoid confusion. Since a was mainly used in the stopping condition based on the lower and upper bound which was removed, this did not influence the implementation of the algorithm. It is also possible to solve problems which do not have a constant trace and therefore do not fulfill the assumption. With the problems the algorithm was tested there appeared no problems with the fact, that the trace is not supposed to be constant. However it can not be totally assured that this is the case for all possible SDP-problems. But hence the trace of X gets approximated during the repetitions of the algorithm and it converges, it should work for a big part of the problems.

It can also happen that YALMIP returns `NaN` as a value of the optimal solution by solving (LPR). Since it is not possible to compute for example eigenvalues of matrices containing `NaN`, those are replaced by 0. This avoids problems in the first iterations where the problem has not yet many constraints.

Another problem which raised have been problems that leaded to a infinite solution for (LPR). The attempt to solve this problem by creating new constraints to avoid this, did not worked out well. The idea was to add the corresponding eigenvector to the maximal eigenvalue

$$\lambda_{\max}\left(\sum_{j=1}^{m} r_j A_j\right),$$

where r is the descent direction for (LDR), to the set of constraints. But as already said this did not sove the problem. Therefore this procedure was not added to the algorithm. Instead the algorithm stops and returns the problem of indicated by the solver, e.g. unconstrained problem.

The source code of the implementation of the polyhedral cutting plane algorithm can be found on `https://documents.epfl.ch/users/f/fu/furrer/www/cuttingplane/`.

12 Ellipsoid Problem

In this section a semidefinite programming problem which is used to test the implementation of the polyhedral cutting plane algorithm is presented. This problem comes from Vandenberghe and Boyd [15, p. 58-59]. It is a geometrical problem.

Suppose that we have k ellipsoids $\varepsilon_1, \ldots, \varepsilon_k$ given. We want find the ball with minimal radius containing all ellipsoids.

The ellipsoids can be described with help of quadratic functions

$$f_i = x^T A_i x + 2b_i^T x + c_i, \quad i = 1, \ldots, k,$$

where $\varepsilon_i = \{x | f_i \leq 0\}$. It is possible the express the fact that one ellipsoid contains an in a matrix form. If we have the ellipsoids $\varepsilon = \{x | f \leq 0\}$ and $\tilde{\varepsilon} = \{x | \tilde{f} \leq 0\}$ with the corresponding functions

$$f = x^T A x + 2b^T x + c \text{ and } \tilde{f} = x^T \tilde{A} x + 2\tilde{b}^T x + \tilde{c},$$

then ε contains $\tilde{\varepsilon}$ if and only if there exists a $\tau \geq 0$ such that

$$\begin{bmatrix} A & b \\ b^T & c \end{bmatrix} \preceq \tau \begin{bmatrix} \tilde{A} & \tilde{b} \\ \tilde{b}^T & \tilde{c} \end{bmatrix}$$

Now since a ball is also an ellipsoid it can be described as well in the form used above. The ball $\mathcal{S}$ can be represented by

$$f = x^T x - 2x_c^T x + \gamma.$$

Therefore for a ball $\mathcal{S}$ which contains all ellipsoids $\varepsilon_1, \ldots, \varepsilon_k$, there must exist $\tau_1, \ldots, \tau_k \geq 0$ such that

$$\begin{bmatrix} I & -x_c \\ -x_c & \gamma \end{bmatrix} \preceq \tau_i \begin{bmatrix} \mathcal{A}_i & b_i \\ b_i^T & c_i \end{bmatrix} \quad \text{for } i = 1, \ldots, k. \tag{2}$$

Since we want to minimize the ball's radius $r = \sqrt{x_c^T x_c - \gamma}$, we use $r^2 \leq t$, which can be presented in a matrix inequality

$$\begin{bmatrix} I & x_c \\ x_c^T & \gamma \end{bmatrix} \succeq 0, \tag{3}$$

and we minimize t.

Gathering all this information together we get the following semidefinite program

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \begin{bmatrix} I & -x_c \\ -x_c & \gamma \end{bmatrix} \preceq \tau_i \begin{bmatrix} \mathcal{A}_i & b_i \\ b_i^T & c_i \end{bmatrix} \quad \text{for } i = 1, \dots, k \\ & \tau_i \geq 0, \quad i = 1, \dots, k, \\ & \begin{bmatrix} I & x_c \\ x_c^T & \gamma \end{bmatrix} \succeq 0. \end{array}$$

In this program the variables are $x_c, \tau_1, \dots, \tau_k, \gamma$ and t.

But this problem in not in standard form which must be known to solve it with help of our implemented cutting plane solver. Therefore we are going to transform this problem into standard form.

First one may recognize that it is possible to rewrite (2) in one block diagonal matrix inequation. In the same manner we can add the other constraints to this matrix to get one huge blockdiagonal matrix inequation.

Then we need to separate the variables and constants in (2) and (3). Since our semidefinite program is closer to the dual form then the primal form we want to transform it into the form

$$\begin{array}{ll} \text{maximize} & b^T y \\ \text{minimize} & \displaystyle\sum_{i=1}^{m} y_i A_i \preceq C. \end{array}$$

Therefore we separate the the variables and the corresponding matrices. For example (2) becomes

$$-x_{c1} \begin{bmatrix} 0 & e_1 \\ e_1^T & 0 \end{bmatrix} - \cdots - x_{cn} \begin{bmatrix} 0 & e_n \\ e_n^T & 0 \end{bmatrix} + \gamma \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} +$$
$$+t \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} - \tau_1 \begin{bmatrix} A_1 & b_1 \\ b_1^T & c_1 \end{bmatrix} - \cdots - \tau_k \begin{bmatrix} A_k & b_k \\ b_k^T & c_k \end{bmatrix} \preceq \begin{bmatrix} -I & 0 \\ 0 & 0 \end{bmatrix}.$$

Since the notation for ellipsoids in a n-dimensional space is not convenient to write down we are going to consider now the 2-dimensional space. It is possible to do the same thing without any problems for higher dimensions.

Let us set

$$y = \begin{bmatrix} x_{c1} & x_{c2} & \gamma & t & \tau_1 & \cdots & \tau_k \end{bmatrix}^T$$

Now it is necessary to find the corresponding matrices denoted with B_i since A_i is already used. The matrix corresponding to x_{c1} is a block diagonal matrix with

$$\begin{bmatrix} 0 & -e_1 \\ -e_1^T & 0 \end{bmatrix}$$

for the first k blocks at the beging corresponding to (2), then a block of $k \times k$ zero-matrix and then the matrix of the beging again. The matrix for x_{c2} is the same except e_1 is switched to e_2.

For γ there are first k matrices of the form

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix},$$

followed also $k \times k$ zero-matrix and at the end, there is

$$\begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}.$$

This block diagonal matrix is multiplied with γ. The matrix which is multiplied with t has the same size as the other and consists out of zeros except for the last block which is

$$\begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}.$$

The matrices corresponding to τ_i, $i = 1, \dots, k$ are for the first k block zero except the ith block which is

$$-\begin{bmatrix} A_i & b_i \\ b_i^T & c_i \end{bmatrix}.$$

The next block is zero except for the ith diagonal element which is -1. The last block is 0.

The C matrix is

$$\begin{bmatrix} -I & 0 \\ 0 & 0 \end{bmatrix}$$

for the first k blocks, then follows a $k \times k$ zero-matrix and at the last block is

$$\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}.$$

The only component missing is now b, which is a zero-vector except for the 4 component which is corresponding to t. This component is -1 since t is minimized but we are in dual form which must be maximized.

Then we have a problem in the form

$$\begin{array}{ll} \text{maximize} & b^T y \\ \text{minimize} & \displaystyle\sum_{i=1}^{m} y_i B_i \preceq C, \end{array}$$

where B_i are the matrices introduced before. Hence this problem can also be written in the primal form

$$\begin{array}{ll} \text{minimize} & C \cdot X \\ \text{subject to} & B_i \cdot X = b_i \quad \text{for } i = 1, \ldots, k, \\ & X \succeq 0. \end{array}$$

So we have transformed the ellipsoid problem into standard form. It is now possible to test the implemented algorithm on it.

13 Numerical Results and Test Problems

In this chapter two different tests of the algorithm are presented: one small example and ellipsoid problem example. The algorithm was tested with YALMIP using the standard solver of MATLAB for linear programming `linprog`. First we consider a small semidefinite programming problem with constant trace

$$\begin{array}{ll} \text{minimize} & \begin{bmatrix} 2 & 1 \\ 1 & 5 \end{bmatrix} \cdot X \\ \text{subject to} & I \cdot X = 2, \\ & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot X = 1. \end{array}$$

We used $\epsilon_1 = \epsilon_2 = 10^{-5}$ as precision parameter. With this setting a result was found in 8 iterations. The result is

$$X = \begin{bmatrix} 1.7071 & -0.7071 \\ -0.7071 & 0.2929 \end{bmatrix}.$$

This result result consists with the optimal value obtained on a other way. This was just a small problem to test if the algorithm is working. To test it on a bigger problem we are going to consider the an ellipsoid problem as presented in Chapter 12. Since this is a geometrical problem it is rather easy to check the obtained result on a graphical way.

We consider the following ellipses in the the 2-dimensional space:

$$\begin{array}{lll} A_1 = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} & b_1 = \begin{bmatrix} -2 \\ 8 \end{bmatrix} & c_1 = 20 \\ A_2 = \begin{bmatrix} 2 & -0.1 \\ -0.1 & 1.5 \end{bmatrix} & b_2 = \begin{bmatrix} 1 \\ -5 \end{bmatrix} & c_2 = -10 \\ A_3 = \begin{bmatrix} 1 & 0.1 \\ 0.1 & 1 \end{bmatrix} & b_3 = \begin{bmatrix} 5 \\ 4 \end{bmatrix} & c_3 = 30 \\ A_4 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & b_4 = \begin{bmatrix} 6 \\ -1 \end{bmatrix} & c_4 = 36 \\ A_5 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & b_5 = \begin{bmatrix} -1 \\ 7 \end{bmatrix} & c_5 = 48 \end{array} \tag{4}$$

The goal is to find the circle with minimal radius which contains all the ellipses. This problem can be expressed as in a standard semidefinite programming form. When we solve the problem with `cuttingplane` we get a result. The optimal solution for the primal problem is not particularly interesting since the basic problem is formulated in the dual form. The result for the dual problem is

$$y = \left[\begin{array}{ccccccccc} -0.15 & -0.48 & -64.29 & 64.54 & 1.15 & 1.27 & 2.83 & 2.41 & 5.69 \end{array} \right].$$

The solution can be presented graphically as done in Figure 1. The green ellipses are corresponding to (4). The red circle is the optimal solution. One

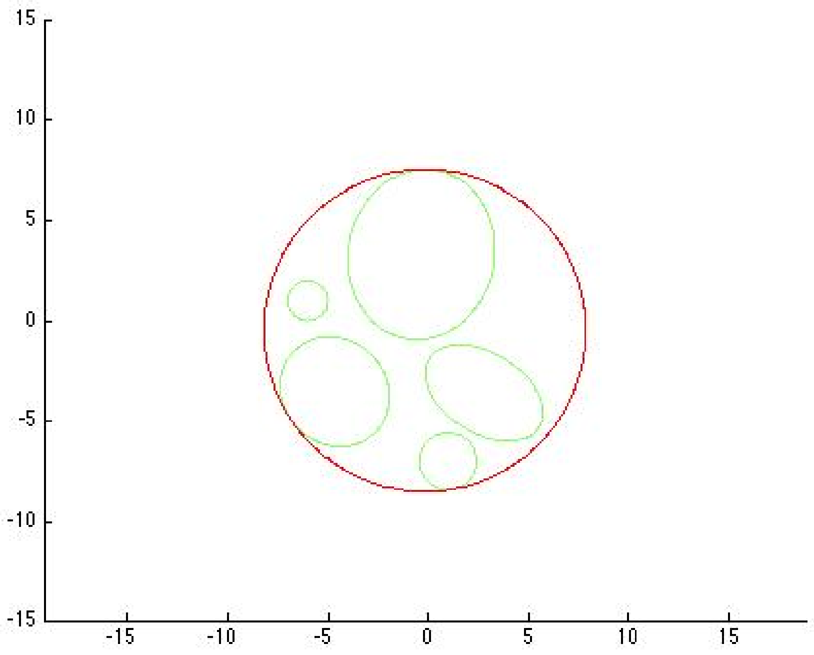

Figure 1: Solution of the ellipsoid problem

can check graphically that the red circle is the optimal solution and the algorithm works also for problems when the assumption for a constant trace is dropped. This solution was found performing 45 iterations. The stopping parameters have been $\epsilon_1 = \epsilon_2 = 10^{-5}$

By analyzing the time spend on during the process one recognizes that a lot of the time is used by functions of YALMIP especially by `solvesdp`. Actually the time for solving the problem was rather high. This is mainly due to the reformulation of the problem such that it can be solved by `linprog`. Therefore the algorithm would probably become a lot faster formulating the problem in another way such that it takes less time to transform it. It must also be mentioned that `linprog` is probably not the best sover available and there could be some gains by using another one.

Other improvements of the algorithm have already discussed in Chapter 4. For example one may consider to add more than one constraint to the set of all constraints during an iteration.

After all, the algorithm is working good but it could be optimized. There are some possible improvements. One should also not forget that this algorithm is not considered to be the best of the presented algorithms but it contains the main idea of a cutting plane approach to semidefinite programming.

References

[1] D Goldfarb. The simplex method for conic programming. Technical report, CORC, Columbia University, 2002.

[2] C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM J. Optim.*, 10(3):673–696 (electronic), 2000.

[3] Christoph Helmberg, Franz Rendl, Robert J. Vanderbei, and Henry Wolkowicz. An interior-point method for semidefinite programming. *SIAM J. Optim.*, 6(2):342–361, 1996.

[4] Krzysztof C. Kiwiel. Proximity control in bundle methods for convex nondifferentiable minimization. *Math. Programming*, 46(1, (Ser. A)):105–122, 1990.

[5] K Krishnan and J E Mitchell. A linear programming approach to semidefinite programming problems. Technical report, Department of Mathematical Science, Rensselaer Polytechnic Instituite, 2001.

[6] Kartik Krishnan. *Linear programming (LP) approaches to semidefinite programming (SDP) problems.* PhD thesis, Rensselaer Polytechnic Institute, 2001.

[7] Kartik Krishnan and John E. Mitchell. Semi-infinite linear programming approaches to semidefinite programming problems. In *Novel approaches to hard discrete optimization (Waterloo, ON, 2001)*, volume 37 of *Fields Inst. Commun.*, pages 123–142. Amer. Math. Soc., Providence, RI, 2003.

[8] Kartik Krishnan and John E. Mitchell. A unifying framework for several cutting plane methods for semidefinite programming. *Optim. Methods Softw.*, 21(1):57–74, 2006.

[9] Monique Laurent and Franz Rendl. Semidefinite programming and integer programming. In K. Aardal, G. Nemhauser, and R. Weismantel, editors, *Discrete Optimization*, volume 12 of *Handbooks in Operations Research and Management Science*, pages 393–514. Elsevier B.V., 2005.

[10] J. Löfberg. Yalmip : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[11] Renato D. C. Monteiro. Primal-dual path-following algorithms for semidefinite programming. *SIAM J. Optim.*, 7(3):663–678, 1997.

[12] Mohammad R. Oskoorouchi and Jean-Louis Goffin. The analytic center cutting plane method with semidefinite cuts. *SIAM J. Optim.*, 13(4):1029–1053 (electronic), 2003.

[13] Gábor Pataki. Cone-LP's and semidefinite programs: geometry and a simplex-type method. In *Integer programming and combinatorial optimization (Vancouver, BC, 1996)*, volume 1084 of *Lecture Notes in Comput. Sci.*, pages 162–174. Springer, Berlin, 1996.

[14] M. J. Todd. Semidefinite optimization. *Acta Numer.*, 10:515–560, 2001.

[15] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM Rev.*, 38(1):49–95, 1996.

[16] Yin Zhang. On extending some primal-dual interior-point algorithms from linear programming to semidefinite programming. *SIAM J. Optim.*, 8(2):365–386 (electronic), 1998.