Convex Quantifier Elimination for Semidefinite Programming

Hirokazu Anai[1] and Pablo A. Parrilo[2]

[1] Computer System Laboratories,
Fujitsu Laboratories Ltd
Kamikodanaka 4-1-1, Nakahara-ku Kawasaki, 211-8588, Japan
anai@jp.fujitsu.com

[2] Automatic Control Laboratory
Swiss Federal Institute of Technology (ETH)
CH-8092 Zürich, Switzerland
parrilo@control.ee.ethz.ch

Abstract. Semidefinite Programming (SDP) is a class of convex optimization problems with a linear objective function and linear matrix inequality (LMI) constraints. SDP problems have many applications in engineering and applied mathematics. We propose a reasonably fast algorithm to prove and solve SDP exactly by exploiting the convexity of the SDP feasibility domain. This is achieved by combining a symbolic algorithm of cylindrical algebraic decomposition (CAD) and a lifting strategy that takes into account the convexity properties of SDP. The effectiveness of our method is examined by applying it to some examples on QEPCAD and maple.

1 Introduction

Semidefinite Programming (SDP) is one of the recent main developments in mathematical programming, with many applications in applied mathematics and engineering problems. SDPs are convex optimization problems with a linear objective function and linear matrix inequality (LMI) constraints (see [3],[9]). In particular, a wide variety of questions in areas such as systems and control theory can be cast and solved as SDP problems, hence their great practical and theoretical interest.

Usually, SDP problems are solved numerically using interior point methods, hence obtaining approximate solutions with finite precision. In certain cases (for instance, applications in algebraic geometry [13]) an exact algebraic representation of the optimal solution, and not just numerical values, is desired. Additionally, in critical situations such as ill-posed problems, there is a real danger of arriving at an incorrect answer; we may obtain a "numerically" feasible solution for an infeasible problem, or vice versa. Hence it is important to develop exact methods for deciding their feasibility, and also characterizing and computing the exact solution of SDP problems. This can be accomplished by symbolic optimization methods based on quantifier elimination (QE) [1]. The downside of this approach are the bad computational complexity properties of generic QE algorithms.

For these reasons, in this paper we propose an improved algorithm to prove and solve SDP problems *exactly* based on a symbolic method of *cylindrical algebraic decomposition (CAD)* [5], and the careful exploitation of the *convexity* of the SDP domain through a bisection-based lifting strategy. For simplicity, we will also assume the availability of a feasible interior point, a reasonable requirement common with interior point methods. This contributes to reduce the computation time by restricting the parameter space to an a priori known feasible region. This assumption can be removed at a slightly higher computational cost. We examine the performance of our method by solving some examples using QEPCAD[1] and maple.

Our method can be regarded as a specialized CAD algorithm for SDP problems exploiting convexity and an initial interior point. The ideas used here could be generalized to other classes of convex programming. Thus, we refer to the approach as *"convex quantifier elimination."*

Local quantifier elimination proposed in [8] aims at decreasing the size of output formula and the computation time by restricting the parameter space to an interesting area around the given point. They proposed local quantifier elimination procedure based on the quantifier elimination by virtual substitution [14, 11]. Convex quantifier elimination differs from local quantifier elimination in the following point: Given a first-order formula $\varphi(v_1, \ldots, v_n)$ and a point $a = (a_1, \ldots, a_n) \in \mathbb{R}^n$, local quantifier elimination computes from φ a quantifier-free formula φ^* and a semi-algebraic set $S \in \mathbb{R}^n$ containing the point a, such that for all $s \in S$ we have that $\varphi(s)$ and $\varphi^*(s)$ are equivalent. Convex quantifier elimination ensures that the semi-algebraic set S including the given point a is a unique feasible region by virtue of the convexity.

The organization of the rest of the paper is as follows: a brief introduction of SDP problems and a reduction procedure to QE problems is explained in §2. A review of the basic ideas and notation of cylindrical algebraic decomposition in presented in §3. In §4 we present our efficient method to solve SDP problems exactly based on a highly specialized CAD algorithm. Computational results for some concrete examples are presented to demonstrate the validity of our approach in §5. Finally, in §6 we present our concluding remarks.

2 Semidefinite programming

2.1 SDP problems

We present next the definition of semidefinite programming problems. A symmetric matrix $A \in \mathbb{R}^{n \times n}$ is *positive (semi) definite* if and only if the quadratic form $x^T Ax > 0$ (≥ 0) for all $x = (x_1, \cdots, x_n) \in \mathbb{R}^n$ s.t. $x \neq 0$, where x^T stands for the transpose of x. In the sequel, when A is positive (semi) definite, we denote it by $A \succ 0$ $(\succeq 0)$. For a real symmetric matrix A, $A \succ 0$ $(\succeq 0)$ if and only if all

[1] See `http://www.cs.usna.edu/~qepcad/B/QEPCAD.html`

eigenvalues of A are positive (non negative). A *linear matrix inequality (LMI)* is a matrix inequality of the form

$$M(x) = M_0 + \sum_{i=1}^{m} x_i M_i \succ 0 \ (\succeq 0) \tag{1}$$

where $x \in \mathbb{R}^m$ is the variable vector and $M_i = M_i^T \in \mathbb{R}^{n \times n}$, $i = 0, \ldots, m$, are given symmetric matrices.

In general, there are three types of (quasi) convex optimization problems with LMI constraints; *Feasibility*, *Linear objective minimization under LMI constraints* and *Generalized eigenvalue minimization* [9]. Among them we focus on the second one, the minimization of a linear objective function of a vector variable $x \in \mathbb{R}^m$ subject to a linear matrix inequality, i.e.,

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{s.t.} & M(x) \succeq 0, \end{array} \tag{2}$$

where $c \in \mathbb{R}^m$. This problem is called *Semidefinite Programming (SDP).* For a vector x_0, if $M(x_0) \succeq 0$, x_0 is called *feasible.* If there is no feasible solution, we say that the problem (2) is *infeasible.* Notice that set of feasible points in a *convex set*, and that the optimal solution, if achieved, lies on the boundary of the feasible set. SDP includes many important convex optimization problems such as linear programming as special cases.

SDP problems are usually solved by a numerical techniques based on interior point methods, hence obtaining approximate solutions of finite precision. As explained, in certain applications or critical situations, e.g. ill-posed problems, exact representations of the solutions of its feasibility are desired. Hence we propose a method of computing the exact feasible solution of SDP problems using a symbolic optimization method based on quantifier elimination.

2.2 Reducing a SDP problem to a QE problem

In general, optimization problems of minimizing an objective function $h(x)$ subject to a constraint that is a first-order formula $\phi(x)$ are solved by using QE as follows: First introduce a new indeterminate z assigned to the objective function h and consider the new first-order formula

$$\phi'(x, z) = \ \phi(x) \wedge (z - h(x) \geq 0).$$

We call the polynomial $z - h$ an *objective polynomial.* Then the problem of minimizing h subject to ϕ is formulated as a QE problem

$$\varPhi \equiv \ ^{\exists}x_1 \cdots \ ^{\exists}x_m(\phi').$$

Next eliminate all quantified variables $x_1, \ldots, x_m$ to have the resulting quantifier-free formula $\varPhi'$ in z. Then $\varPhi'$ gives a finite union M of intervals for z, which shows the possible range of z. If M is empty, ψ is unsolvable (*i.e.* infeasible); if

M is unbounded from below, h has no minimum w.r.t. ϕ; if $\mu \in M$ is a lowest endpoint of M, then μ is the minimum value of z w.r.t. ϕ (see [15] for details).

We show next how a linear matrix inequality constraint in a SDP problem can be reduced to a QE problem: determining (semi)definiteness for a real symmetric matrix can be achieved without computing eigenvalues by using the following well-known Sylvester's criterion:

Theorem 1 (Sylvester's criterion). *Let $M = (m_{ij}) \in \mathbb{C}^{n\times n}$ be a Hermitian matrix. Then M is positive semidefinite if and only if all principal minors of M are non negative* i.e.

$$\det\ M \begin{pmatrix} i_1 & i_2 & \cdots & i_r \\ i_1 & i_2 & \cdots & i_r \end{pmatrix} \geq 0, \tag{3}$$

for $1 \leq i_1 < i_2 < \cdots < i_r \leq n,\ r = 1, 2, \cdots, n,$ where

$$M \begin{pmatrix} i_1 & i_2 & \cdots & i_r \\ j_1 & j_2 & \cdots & j_r \end{pmatrix}$$

denotes the $r \times r$ submatrix of M which consists of (i_k, j_l)-entries of M, where $1 \leq i_1 < i_2 < \cdots < i_r \leq n$ and $1 \leq j_1 < j_2 < \cdots < j_r \leq n$.

By this criterion, a linear matrix inequality $M(x) \succeq 0$ can be exactly reduced to an equivalent formula that is the conjunction of $2^n - 1 (\equiv \sum_{r=1}^{n} \binom{n}{r})$ inequalities, say δ_k $(k = 1, \ldots, 2^n - 1)$, of the form (3). Consequently, the SDP problem (2) can be recast as the following QE problem:

$$\Psi \equiv\ ^{\exists}x_1 \cdots\ ^{\exists}x_m(\rho). \tag{4}$$

where

$$\rho \equiv \left(z - c^T x \geq 0\ \wedge \bigwedge_{k=1}^{2^n-1} \delta_k \geq 0 \right).$$

In general, to solve (4) exactly we need to use a quantifier elimination based on CAD. In practice we can solve the QE problem (4) only for matrices $M(x)$ of very small size because of the many constraints δ_k and the doubly exponential worst-case complexity of the CAD algorithm (see [1] for example). Therefore, it is clearly desirable a more efficient way of solving SDP problems exactly based on symbolic methods.

In the subsequent sections we propose a different, more efficient CAD algorithm adapted to SDP problems by exploiting the special structure and convexity of SDP domains.

3 CAD algorithm

We briefly sketch the basic ideas of cylindrical algebraic decomposition, see [5] for details. Assume that we are given an input formula

$$\varphi(u_1, \ldots, u_m) \equiv \mathsf{Q}_1 x_1 \ldots \mathsf{Q}_n x_n\ \psi(u_1, \ldots, u_m, x_1 \ldots x_n), \quad \mathsf{Q}_i \in \{\exists, \forall\}.$$

Let F be the set of polynomials appearing in ψ as left hand sides of atomic formulas. We say that $C \subseteq \mathbb{R}^{m+n}$ is *sign-invariant* for F if every polynomial in F has a constant sign on all points in C. Then $\psi(c)$ is either "true" or "false" for all $c \in C$.

Suppose we have a finite sequence $\mathcal{D}_1, \dots, \mathcal{D}_{m+n}$ for F which has the following properties:

1. Each $\mathcal{D}_i$ is a finite partition of $\mathbb{R}^i$ into connected semi-algebraic cells. For $1 \leq j \leq n$ each $\mathcal{D}_{m+j}$ is labeled with Q_j
2. $\mathcal{D}_{i-1}$ for $1 < i \leq m+n$ consists exactly of the projections of all cells in $\mathcal{D}_i$ along the coordinate of the i-th variable in $(u_1, \dots, u_m, x_1 \dots x_n)$.
   For each cell $C \in \mathcal{D}_{i-1}$ we can determine the preimage $S(C) \subseteq \mathcal{D}_i$ under the projection.
3. For each cell $C \in \mathcal{D}_m$ we know a quantifier-free formula $\delta_C(u_1, \dots, u_m)$ describing this cell.
4. Each cell $C \in \mathcal{D}_{m+n}$ is sign-invariant for F. Moreover for each cell $C \in \mathcal{D}_{m+n}$ we are given a *test point* t_C in such a form that we can determine the sign of $f(t_C)$ for each $f \in F$ and thus evaluate $\varphi(t_C)$.

Then a finite partition $\mathcal{D}_{m+n}$ of $\mathbb{R}^{m+n}$ for F is called an *F-invariant cylindrical algebraic decomposition* of $\mathbb{R}^{m+n}$. A quantifier-free equivalent formula φ is obtained as the disjunction of all δ_C for which $C \in \mathcal{D}_m$ is *valid* in the following sense:

1. For $m \leq i < m+n$, we have $\mathcal{D}_{i+1}$ that is labeled:
   (a) If $\mathcal{D}_{i+1}$ is labeled "$\exists$", then $C \in \mathcal{D}_i$ is valid if at least one $C' \in S(C)$ is valid.
   (b) If $\mathcal{D}_{i+1}$ is labeled "$\forall$", then $C \in \mathcal{D}_i$ is valid if all $C' \in S(C)$ are valid.
2. A cell $C \in \mathcal{D}_{m+n}$ is valid if $\varphi(t_C)$ is "true".

The algorithm to obtain such a sequence $\mathcal{D}_1, \dots, \mathcal{D}_{m+n}$, the quantifier-free formula δ_C, and the test point t_C consists of two phases, the *projection phase* and *construction (lifting) phase.*

Projection phase: In the projection phase, one constructs from $F \subseteq \mathbb{R}[u_1, \dots, u_m, x_1, \dots, x_n]$ a new finite set $F' \subseteq \mathbb{R}[u_1, \dots, u_m, x_1, \dots, x_{n-1}]$ that satisfies a particular condition, detailed next. Consider $a, b \in \mathbb{R}^{m+n-1}$ such that for all $f' \in F'$ the signs of both $f'(a), f'(b) \in \mathbb{R}$ are equal. Then for all $f \in F$ the corresponding univariate polynomials $f(a, x_n), f(b, x_n) \in \mathbb{R}[x_n]$ both have the same number of different real and complex roots. This guarantees the following property called "*delineability*": Let C be a connected subset of $\mathbb{R}^{m+n-1}$ that is sign-invariant for F'. For each $f \in F$ consider the functions $\rho_k : C \to \mathbb{R}$ assigning to $a \in C$ the k-th real root of $f(a, x_n) \in \mathbb{R}[x_n]$. Then all these ρ_k are continuous. Moreover, the graph of the various ρ_k do not intersect. In other words, the order of the real roots does not change as they continuously change their position in the real line.

The step from F to F' is called a *projection* and denoted by $F' := PROJ(F)$. We call polynomials in F' *projection polynomials* and the irreducible factors of projection polynomials of F' *projection factors*. Iterative application of $PROJ$-operator leads to a finite sequence

$$F_{m+n}, \ldots, F_1, \quad where \;\; F_{m+n} := F, \; F_i := PROJ(F_{i+1}) \;\; for \;\; 1 \leq i < m+n.$$

The $PROJ$-operator computes certain coefficients, discriminants, resultants, and subresultant coefficients obtained from the polynomials in F_{i+1} and their higher derivatives, regarded as univariate polynomials in their last variable, which is the $(i+1)$-st one in $(u_1, \cdots, u_m, x_1, \ldots, x_n)$. The final set F_1 contains only univariate polynomials in u_1.

Construction phase: In the construction phase we first construct a partition $\mathcal{D}_1$ of the real line $\mathbb{R}^1$ into finitely many intervals that are sign-invariant for F_1. The real zeros of the univariate polynomials in F_1 define a sign invariant decomposition of $\mathbb{R}^1$. The partition $\mathcal{D}_1$ consists of cells given by these zeros and the intermediate open intervals. Thus we isolate the above zeros and find test points in each interval. This procedure is called the *base phase.* For an open interval we may choose a rational test point but for a zero in general we need its exact representation of an algebraic number.

For $1 \leq i < m+n$ the partitions $\mathcal{D}_i \subseteq \mathbb{R}^i$ are computed recursively: The roots of all polynomials in F_i as univariate polynomials in their last variable are delineated above each connected cell C in $\mathcal{D}_{i-1}$. Thus we can cut the *cylinder* above C into finitely many connected semi-algebraic cells. Then $\mathcal{D}_i$ is a collection of all these cells arising from all cylinders above the cells of $\mathcal{D}_{i-1}$

Consider the lifting from the partition $\mathcal{D}_1$ of $\mathbb{R}^1$ to a partition $\mathcal{D}_2$ of $\mathbb{R}^2$ since remaining lifting procedures until $\mathbb{R}^{m+n}$ are achieved by repeating the same procedure as the lifting from $\mathbb{R}^1$ to $\mathbb{R}^2$. We show the construction of the test points of each cell of $\mathbb{R}^2$ belonging to the cylinder over a cell $C \in \mathcal{D}_1$ with a test point α. First specialize the polynomials in $F_2 := PROJ^{m+n-2}(F)$ by the test point α of C. We then get a set of univariate polynomials in u_2 and deal with these polynomials in u_2 in the same way as the base phase, i.e., root isolation and choice of test points. The lifting from $\mathbb{R}^1$ to $\mathbb{R}^2$ is regarded as the construction of the second component of the test points of $\mathcal{D}_{m+n}$.

The Construction phase produces a list of (indexed) cells and their test points. We know which cells $S(C)$ in $\mathcal{D}_i$ origin from which cell C in $\mathcal{D}_{i-1}$. This implies that a finite sequence $\mathcal{D}_1, \ldots, \mathcal{D}_{m+n}$ for F has a structure of a tree representation. The first level of nodes under the root of the tree corresponds to the cells in $\mathcal{D}_1$. The second level of nodes stands for the cells in $\mathcal{D}_2$, i.e., the cylinders over the cells of $\mathbb{R}^1$. The leaves represent the cells of $\mathcal{D}_{m+n}$ (i.e., CAD of $\mathbb{R}^{m+n}$). A test point of the corresponding cell is stored in each node or leaf. To each level of the tree there are a number of projection polynomials F_i whose signs define a cell when evaluated over a test point.

4 Convex quantifier elimination: a specialized CAD for SDP problems

4.1 Improving CAD algorithm

Projection phase It is crucially important for the efficiency of CAD construction that the *PROJ* operator produces as small a set of polynomials as possible, while still ensuring the cylindrical arrangement of cells in the resulting decomposition. Improved projection operators have been proposed by several authors [12, 10, 4].

The complexity of the projection phase is given by the following: given r irreducible polynomials of degree less than or equal to d in N variables, then after $N-1$ projection steps we have $(r \cdot d)^{2^{O(N)}}$ polynomials of degree at most $d^{2^{O(N)}}$. It is often the case that we cannot decrease the number of variables when we reduce the target problem considered to an equivalent first-order formula. Thus, ideally we should produce an equivalent input formula with fewer polynomials, and smaller degree.

Construction phase There are two devices to improve the efficiency of the construction phase:

(a) Avoiding algebraic computation during lifting processes: Revisit the extension of $\mathcal{D}_1$ to $\mathcal{D}_2$. Let C be a cell of $\mathcal{D}_1$ with a test point α. Consider all polynomials $f(u_2) := f(\alpha, u_2) \in F_2$ that are not identically zero. The real roots of $f(u_2)$'s determine the test points of each cell of $\mathbb{R}^2$ in the cylinder over a cell C. Let β be a root of $f(u_2)$. If the test point α is an algebraic number, we need computations over the algebraic extension field $\mathbb{Q}(\alpha)$ for root isolation. Moreover it is required that the test point of each cell be a vector of algebraic numbers over a simple algebraic extension of $\mathbb{Q}$. Hence we need to compute a primitive element γ for $\mathbb{Q}(\alpha, \beta)$ and represent the test point (α, β) as pairs of elements of $\mathbb{Q}(\gamma)$. Computations over an algebraic extension field are typically more expensive than those over $\mathbb{Q}$. For the efficiency of the construction phase, we should consider the possibility of avoiding the use of algebraic test points.

(b) Pruning unnecessary branches of a CAD tree: In general not every cell in the construction is actually necessary for eliminating quantifiers in a given input formula φ. This observation was first made and generalized to *partial CAD* by H. Hong [6]. Partial CAD systematically exploits the logical structure of the input formula. This greatly reduces the number of cells to be considered. Furthermore we can expect to exploit the special structure of the input formula (e.g., convexity) in order to prune unnecessary branches of a CAD tree.

4.2 Exploiting convexity of SDP

The feasibility domain of an SDP is a convex set, and this implies several important properties: (i) the feasible region is a unique connected region, (ii) the determinant polynomial vanishes on the boundary of the feasible region of the SDP. Moreover, we also assume that a feasible interior point is available (we can remove this assumption, at a slightly higher computational cost). We will use these properties to improve the CAD algorithms in the sequel. The next theorem guarantees (ii):

Theorem 2. *The determinant vanishes on the boundary of the domain of positive semidefiniteness of a real symmetric matrix.*

Sketch of the proof: All eigenvalues of a symmetric matrix are real and the matrix is positive semidefinite if and only if all eigenvalues are greater than or equal to zero. Moreover the eigenvalues, as zeros of the characteristic polynomial, depend continuously on the entries of the matrix. Suppose now the values of the entries of the symmetric matrix are such that the matrix is on the boundary of its positive semidefinite domain. Then in every neighborhood of this point there is a point, where the matrix is not positive semidefinite, and hence has a negative eigenvalue. So by the continuity of eigenvalues, the matrix must have a zero as an eigenvalue at this point, and therefore the determinant vanishes. $\square$

The two properties described above allow us to formulate specialized methods for both the projection and lifting stages. For simplicity, we assume that the SDP to be solved have a strictly feasible solution, i.e., there exists an x such that $M(x) \succ 0$.

(1) improved projection phase: As shown in §2, $M(x) \succeq 0$ can be reduced to an equivalent formula that is the conjunction of $2^n - 1 (\equiv \sum_{r=1}^{n} \binom{n}{r})$ inequalities. Since the boundary of the feasible region of SDP is a part of the determinant polynomial $\det M(x)$, it is sufficient that we consider the set $\{z - c^T x, \det M(x)\}$ as an input set of CAD to obtain the test point which provides a minimum of the objective function. This greatly improves the efficiency of the projection phase in contrast to the reduction according to Sylvester's criterion.

Remark 1. Notice that this property does *not* imply that we can replace the LMI constraint in the original problem with just the constraint $\det M(x) = 0$, as the other principal minors do play an important role in selecting the optimal solution. Nevertheless, it is always the case that the determinant vanishes at the optimal solution.

(2) improved construction phase: We construct a CAD for an input set

$$\{z - c^T x, \det M(x)\}$$

in order to solve an SDP given by (2). After m recursive projections we have a set of univariate polynomials in z. We denote the set of test points which are real roots of the univariate polynomials in z by T_R, the set of test points taken from the intervals between the roots by T_I.

Since the SDP feasibility domain is convex, the feasible region of z is a unique isolated interval. The endpoints of the feasible interval of z correspond to the maximum and minimum of z, i.e., the objective function. We call the left

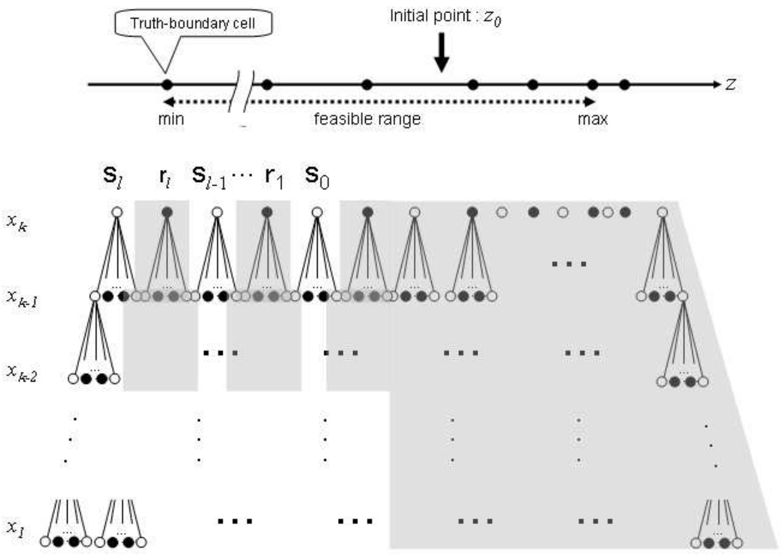

Fig. 1. A typical CAD tree by the convex QE for SDP.

endpoint giving the minimum of z a *truth-boundary cell*, which is contained in T_R.

Suppose that we have an interior point $x_0 = (x_1^0, \dots, x_m^0)$ of the SDP feasible set. This is the same setting as in numerical interior point methods. Then since we have a feasible value $z_0 = c^T x_0$ of z, the feasible interval of z consists of the cell containing z_0 and the connected cells. Here we consider only the case $z_0 \notin T_R$ because if $z_0 \in T_R$ then we can regard the test point of the next left interval of z_0 as z_0 and then we can proceed in the same way as shown below.

The test points larger than z_0 are not needed in finding the minimum of z. We can denote the test points smaller than z_0 as follows:

$$-\infty < \cdots < s_\ell < r_\ell < s_{\ell-1} < r_{\ell-1} < \cdots < r_2 < s_1 < r_1 < s_0 < r_0 < z_0,$$

where $r_i \in T_R$, $s_i \in T_I$, and let r_ℓ be the the truth-boundary cell, i.e., the minimum of z.

In order to find the truth-boundary cell we start with construction of a CAD from over the cell with a test point s_0. If the cell with a test point s_0 has a “true” leaf then we construct a CAD over the next cell s_1. Repeat this similarly until a certain s_i has no “true” leaf. In other words, we make *depth-first search* of “true” leaf of the CAD tree over s_i's from the right to the left until a certain s_i has no “true” leaf. When testing the feasibility of the lifted point in $\mathbb{R}^m$, the full positive semidefiniteness condition on $M(x)$ should be checked, and not just the determinant. For efficiency reasons, it is desirable to choose the order of s_i's to be considered according to a standard bisection scheme.

We have then the following proposition:

Proposition 1. *There exists an integer ℓ such that $s_{\ell-1}$ has a "true" leaf but s_ℓ has no "true" leaf. Then r_ℓ is the truth-boundary cell.*

The coordinate x_{min} which gives the minimum value of z can be obtained by lifting over the truth boundary cell. Note that we do not have to use test points in T_R to identify which test point is the truth-boundary cell. As for the lifting at the level of the tree corresponding to x_i, we can also prune unnecessary branches since the feasible region consists of the connected cells to the cell with a test point x_i^0. Thus we can ignore outer both sides of the feasible region.

A typical CAD tree by the convex QE proposed here for a SDP problem is demonstrated in Fig. 1. The white node "○" stands for a test point in T_I and the black one "●" a test point in T_R . The construction procedure is skipped in the shaded area of the CAD tree.

5 Examples

We have examined our method presented in precedings sections for the following SDP problems by using QEPCAD and maple for projection phase and for construction phase, respectively. [2]

Example 1. A feasible interior point is: $(a, b, c) = (0, 3, 0)$.

$$\text{objective: } a+b+c, \qquad \text{s.t.} \begin{bmatrix} 1 & a & 3-b \\ a & 5 & c \\ 3-b & c & 9 \end{bmatrix} \geq 0.$$

Example 2. A feasible interior point is: $(a, b, c) = (1, -1, 1)$.

$$\text{objective: } a+2b+3c, \qquad \text{s.t.} \begin{bmatrix} a & b \\ b & c+1 \end{bmatrix} \geq 0, \quad \begin{bmatrix} 1 & b+c \\ b+c & 2-a \end{bmatrix} \geq 0.$$

Example 3. This example arises from the minimization of a symmetric quartic polynomial. A feasible interior point is: $(\gamma, a, b, c, d) = (60, -1, 12, 2, 4)$.

$$\text{objective: } \gamma, \qquad \text{s.t.} \begin{bmatrix} \gamma & a & b \\ a & 1 & c \\ b & c & 2d \end{bmatrix} \geq 0, \quad 1-2a \geq 0, \quad \begin{bmatrix} 2b & d \\ d & 1 \end{bmatrix} \geq 0, \quad 2c-3 \geq 0.$$

[2] All the computations are executed on a PC with a CPU Pentium III 1GHz and 756 MB memory.

5.1 Projection phase

Table 1 shows the number of projection factors appearing at each level of the CAD tree, total number of projection factors, and total time to accomplish the projection phase for the above three examples. Here, "–" implies that we could not finish the computation. "$*_1$" means the total number of projection factors occurring until level 2 and "$*_2$" stands for the total time until QEPCAD halted on computing level 1. "Sylv" uses Sylvester's criterion to reduce an SDP to a first-order formula and "A&P" uses our approach shown in the previous section for the reduction . These results show that projection phase of our approach performs much faster than that of the approach using Sylvester's criterion.

Ex.1	level	1	2	3	4	total	time
	Sylv	152	22	7	3	184	250
	A&P	7	5	3	2	17	10

Ex.2	level	1	2	3	4	total	time
	Sylv	160	18	7	4	189	200
	A&P	37	9	4	3	53	70

Ex.3	level	1	2	3	4	5	6	total	time
	Sylv	–	2879	88	14	8	5	2995^{*_1}	169250^{*_2}
	A&P	57	58	12	5	3	2	137	1240

Table 1. Computational results for projection phase

5.2 Construction phase

We tried to compute a CAD for the input set according to A&P's reduction by using QEPCAD to get the minimum of objective functions. We used the option `+N50000000`, this means a size for SACLIB's garbage collected array. The default value is `+N2000000`. However, QEPCAD halted due to lack of memory with a message "`Too few cells reclaimed`" for all examples.

We have not yet finished the implementation of our proposed method. We manually applied our strategy for choosing test points and searching for true leaves for the above examples on `maple`. Then we were able to solve the construction phases for all examples in a few minutes.

6 Conclusions

We have introduced a convexity-based quantifier elimination algorithm as an efficient method to compute exact (algebraic) representations of the solution of SDP problems. Our approach can be regarded as a specialized CAD algorithm for partially exploiting the convexity of the feasible set of SDP. This is also a successful attempt to combine symbolic and numeric approaches to achieve efficiency. We hope this work will lead to further generalizations of symbolic approaches to exploit convexity to other related problems.

A maple-implementation of the method proposed here on top of the SyNRAC-package [2] is planned.

Acknowledgments The authors would like to thank C.W. Brown for his invaluable help. The authors would like to thank K. Yokoyama, S. Hara, and V. Weispfenning for fruitful discussions.

References

1. H. Anai. On solving semidefinite programming by quantifier elimination. *In Proc. of American Control Conference, Philadelphia*, pages 2814–2818, 1998.
2. H. Anai and H. Yanami. SyNRAC: A maple-package for solving real algebraic constraints. In *Proceedings of the International Workshop on Computer Algebra Systems and Their Applications: CASA'2003.* To appear in the series LNCS, Springer-Verlag.
3. S. Boyd, L. Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory.* SIAM Studies in Applied Mathematics, vol 15. SIAM, 1994.
4. C. W. Brown. Improved projection for cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 32(5):447–465, 2001.
5. G. Collins. *Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition, LNCS 32.* Springer Verlag, 1975.
6. G. Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12(3):299–328, Sept. 1991.
7. A. Dolzmann, T. Sturm and V. Weispfenning. Real Quantifier Elimination in Practice In Matzat, B. H. and Greuel, G.-M. and Hiss, G., editor, Algorithmic Algebra and Number Theory, pages 221-247, Springer, 1998.
8. A. Dolzmann and V. Weispfenning. Local quantifier elimination. In C. Traverso, editor, *ISSAC 2000: 7–9 August 2000, University of St. Andrews, Scotland: proceedings of the 2000 International Symposium on Symbolic and Algebraic Computation*, pages 86–94, New York, NY 10036, USA, 2000. ACM Press.
9. P. Gahinet, A.Nemirovski, A.J.Laub, and M.Chilai. *LMI Control Toolbox User's Guide, For Use with MATLAB.* The MATH WORKS INC, 1995.
10. H. Hong. An improvement of the projection operator in cylindrical algebraic decomposition. In *ISSAC: Proceedings of the ACM SIGSAM International Symposium on Symbolic and Algebraic Computation*, pages 261–264, 1990.
11. R. Loos and V. Weispfenning. Applying linear quantifier elimination. *The Computer Journal*, 36(5):450–462, 1993. Special issue on computational quantifier elimination.
12. S. McCallum. An improved projection operation for cylindrical algebraic decomposition of three-dimensional space. *Journal of Symbolic Computation*, 5(1-2):141–161, Feb.–Apr. 1988.
13. P. A. Parrilo, Semidefinite programming relaxations for semialgebraic problems, *Math. Prog. Ser. B*, to appear, 2003.

14. V. Weispfenning. The complexity of linear problems in fields. *Journal of Symbolic Computation*, 5(1–2):3–27, Feb.–Apr. 1988.
15. V. Weispfenning. Simulation and optimization by quantifier elimination. *J. Symbolic Computation*, 24(2):189–208, 1997.