Solving SDP's in Non-commutative Algebras Part I

de Klerk, E.; Pasechnik, D.V.

Publication date:
2005

Document Version
Publisher's PDF, also known as Version of record

CentER

Discussion Paper

No. 2005–17

SOLVING SDP'S IN NON-COMMUTATIVE ALGEBRAS PART I: THE DUAL-SCALING ALGORITHM

By E. de Klerk, D.V. Pasechnik

January 2005

ISSN 0924-7815

Tilburg University

Solving SDP's in non-commutative algebras Part I: the dual-scaling algorithm

E. de Klerk* D.V. Pasechnik†

January 27, 2005

Abstract

Semidefinite programming (SDP) may be viewed as an extension of linear programming (LP), and most interior point methods (IPM's) for LP can be extended to solve SDP problems. However, it is far more difficult to exploit data structures (especially sparsity) in the SDP case. In this paper we will look at the data structure where the SDP data matrices lie in a low dimensional matrix algebra. This data structure occurs in several applications, including the lower bounding of the stability number in certain graphs and the crossing number in complete bipartite graphs. We will show that one can reduce the linear algebra involved in an iteration of an IPM to involve matrices of the size of the dimension of the matrix algebra only. In other words, the original sizes of the data matrices do not appear in the computational complexity bound. In particular, we will work out the details for the dual scaling algorithm, since a dual method is most suitable for the types of applications we have in mind.

JEL code: C61 - Optimization Techniques; Programming Models; Dynamic Analysis
Key words: semidefinite programming, matrix algebras, dual scaling algorithm, exploiting data structure

1 Introduction

We consider the semidefinite program in standard form:

$$(P): \quad p^* := \inf_X \left\{ \mathbf{Tr}\,(\tilde{A}_0 X) \ : \mathbf{Tr}\,(\tilde{A}_i X) = b_i \ (i = 1, \ldots, m),\ X \succeq 0 \right\},$$

*Tilburg University. E-mail: E.deKlerk@uvt.nl. Supported by the Netherlands Organisation for Scientific Research grant NWO 613.000.214 as well as the NSERC grant 283331 - 04. Part of this research was performed while on leave from the Department of Combinatorics and Optimization, University of Waterloo.
†Tilburg University. E-mail: D.V.Pasechnik@uvt.nl

which has the associated dual problem:

$$(D): \quad d^* := \sup_{y,S} \left\{ b^T y \ : \ \sum_{i=1}^{m} y_i \tilde{A}_i + S = \tilde{A}_0, \ S \succeq 0, \ y \in \mathbb{R}^m \right\}.$$

We assume throughout that both (P) and (D) satisfy the Slater condition.

Although semidefinite programming (SDP) is an extension of linear programming (LP), there has been much less progress on solving large scale SDP's using interior point methods than in the LP case; see e.g. [2].

In this paper we show how one can exploit problem structure if the matrices $\tilde{A}_i$ generate a (low dimensional) matrix algebra. Such SDP's arise, for example, from the estimation of the size of minimal distance binary codes [9], [4], and the lower bounding of the crossing numbers of complete bipartite graphs [5].

We will first describe the data structure we will consider, and then describe how it arises in applications, and subsequently how it can be exploited by interior point algorithms.

The algebraic data structure

We consider a subalgebra $\mathcal{A}$ of $M_n(\mathbb{C})$ spanned (as a vectorspace) by matrices $\{A_1, \ldots, A_d\}$, where $A_1 = I$. In addition,

1. the A_i's are pairwise orthogonal with respect to the coordinate-wise scalar product
$$A \circ B := \sum_{i,j} A_{ij} B_{ij}.$$
2. For any A_i there exists $A_{i'}$ so that $A_i = A_{i'}^T$.
3. the A_i's have non-negative entries;

Let $\Lambda = (\lambda_{ij}^k)$ denote the tensor of structure constants of $\tilde{A}$, that is

$$A_i A_j = \sum_k \lambda_{ij}^k A_k \qquad \text{for} \quad 1 \le i, j \le d. \tag{1}$$

If one defines matrices B_i $(i = 1, \ldots, d)$ via

$$[B_i]_{jk} = \lambda_{ji}^k$$

then the *regular representation* of $\mathcal{A}$ is an algebra isomorphism defined via

$$\phi : A_i \mapsto B_i \quad (i = 1, \ldots, d).$$

Given an $Z \in \mathcal{A}$, the spectra of Z and $\phi(Z)$ are the same (ignoring multiplicities of eigenvalues). This relation allows us to check if a (symmetric) matrix in $Z = Z^T \in \mathcal{A}$ is positive semidefinite by computing the smallest eigenvalue of the $d \times d$ matrix $\phi(Z)$.

How the data structure arises

Assume that the symmetric data matrices $\tilde{A}_i$ $(i = 0, \ldots, m)$ are invariant under the action of a transitive finite group G of permutation matrices, in the sense that

$$P^T \tilde{A}_i P = \tilde{A}_i \quad \forall P \in G,\ i = 0, \ldots, m.$$

The so-called *centralizer ring of* G:

$$\mathcal{A} := \left\{ Y \in \mathbb{R}^{n \times n} \mid Y = \frac{1}{|G|} \sum_{P \in G} P^T X P,\ X \in \mathbb{R}^{n \times n} \right\},$$

is a matrix algebra, i.e. a subspace of $\mathbb{R}^{n \times n}$ that is also closed under matrix multiplication.

This algebra has a basis $A_1, \ldots, A_d$ of (non-symmetric) $\{0, 1\}$ matrices that meet the conditions 1,2, and 3 on page 2. A matrix algebra with such a basis is also called a (non-commutative) association scheme.

The case where the A_i's commute reduces to a linear programming problem, and has important applications (see Schrijver [8], and Goemans and Rendl [3]). In this paper, we will only study the non-commutative case. Gaterman and Parillo [1] discuss the setting described here (and more general ones) in detail, giving proofs or references to the results only mentioned here.

Exploiting the data structure

If S is an optimal solution of problem (D), then so is $\frac{1}{|G|} \sum_{P \in G} P^T S P$. In other words, we may restrict the optimization to the intersection of $\mathcal{A}_G$ with the space of symmetric p.s.d. matrices, as opposed to the entire cone of p.s.d. matrices. Since we assume that the dimension of $\mathcal{A}$ is small, this leads to a reduction in problem size.

There are two obvious ways to reduce the problem size even further:

1. compute the irreducible block diagonal factorization (or any less fine block factorization) of the basis of $\mathcal{A}$;

2. work with the regular representation, or any other (low-dimensional) faithful representation, of $\mathcal{A}$ as opposed to $\mathcal{A}$ itself.

The first option is appealing since primal–dual interior point solvers like SeDuMi [10] can exploit block diagonal data structure. The drawback is that there is no simple relationship between the dimension of $\mathcal{A}$ and the sizes of the blocks. It is possible, in certain applications, that the block sizes are too large to work with, even though the dimension of $\mathcal{A}$ is modest. Moreover, it is not easy in general to compute the irreducible block factorization of $\mathcal{A}$.

In [9] good results were obtained for minimal distance codes using the block diagonalization approach by *finding* (as opposed to numeric or symbolic *computing*) of the factorization needed. The latter was also used later in [4]. In [5] a partial (i.e. not the finest) block factorization was used.

This paper is concerned with the second option. We will show that an interior point algorithm can be implemented in such a way that the linear algebra only involves matrices of the size d (the dimension of $\mathcal{A}$). To fix our ideas, we will discuss only the dual-scaling algorithm (see e.g. Ye [11]). The reason is three-fold:

- the computation for the dual-scaling method is simple compared to that of a primal–dual method;
- for many (combinatorial) applications the optimal values of (P) and (D) give a lower bound on a quantity of interest. A feasible dual solution will yield such a bound.
- for the applications we have in mind, a dual (strictly) feasible starting point is readily available.

2 The dual scaling method

In what follows we describe how the computations of the dual scaling method for an SDP with matrix data being symmetric linear combinations of the A_j's can be implemented.

In particular, we assume that the symmetric SDP data matrices are given as $\tilde{A}_i = \sum_{k=1}^{d} \beta_k^{(i)} A_k$ $(i = 0, \ldots, m)$, where the $\beta_k^{(i)}$ are given constants, a part of Λ that is also given.

Let $y \in \mathbb{R}^m$ define a strictly feasible solution of (D) via $S := \tilde{A}_0 - \sum_{i=1}^{m} y_i \tilde{A}_i \succ 0$.

The dual scaling method indeed only uses a dual feasible iterate to construct the search direction, but it can be interpreted as a method that also attempts to form a feasible primal solution at each iteration.

For a given $\mu > 0$, define

$$X(S,\mu) := \arg\min_{X} \left\{ \left\| \frac{S^{\frac{1}{2}} X S^{\frac{1}{2}}}{\mu} - I \right\| \;\middle|\; \mathbf{Tr}\, \tilde{A}_i X = b_i, \quad i = 1, \ldots, m \right\},$$

and

$$\delta_d(S,\mu) := \left\| \frac{S^{\frac{1}{2}} X(S,\mu) S^{\frac{1}{2}}}{\mu} - I \right\|.$$

Note that, if $X(S,\mu) \succeq 0$, then it is primal feasible. Moreover, if $\delta_d(S,\mu) = 0$ then $(X(S,\mu), S)$ are on the primal-dual central path with parameter μ.

It is easy to show that — if $\delta_d(S,\mu) < 1$, then $X(S,\mu) \succ 0$.

The first–order optimality conditions which yield $X(S,\mu)$ are

$$S\left[\frac{XS}{\mu} - I\right] - \sum_{i=1}^{m} \Delta y_i \tilde{A}_i = 0 \tag{2}$$

$$\mathbf{Tr}\,(\tilde{A}_i X) = b_i, \quad i = 1, \ldots, m. \tag{3}$$

Pre- and post-multiplying the first equation with S^{-1} and subsequently using the second equation yields:

$$\sum_{i=1}^{m} \Delta y_i \mathbf{Tr}\left(\tilde{A}_i S^{-1} \tilde{A}_j S^{-1}\right) = -\frac{1}{\mu} b_j + \mathbf{Tr}\left(\tilde{A}_j S^{-1}\right), \quad j = 1, \ldots, m. \tag{4}$$

The dual scaling method uses the search direction Δy that is obtained by solving (4), and ΔS follows from

$$\Delta S = -\sum_{i=1}^{m} \Delta y_i \tilde{A}_i.$$

In order to ensure that the update of S is positive definite, we choose a value of $\alpha > 0$ such that

$$S + \alpha \Delta S \succ 0.$$

A suitable choice is given by the Dikin ellipsoidal condition

$$\alpha < \frac{1}{\left\| S^{-\frac{1}{2}} \Delta S S^{-\frac{1}{2}} \right\|} = \frac{1}{\sqrt{\Delta y^T M \Delta y}}, \tag{5}$$

where M is the matrix defined by $M_{ij} := \mathbf{Tr}\left(\tilde{A}_i S^{-1} \tilde{A}_j S^{-1}\right)$.

An important observation is that it is *not* necessary to form $X(S, \mu)$ explicitly in order to decide whether or not it is positive definite, or to subsequently calculate the duality gap if it is indeed positive definite: by (2) we know that

$$X(S, \mu) \succeq 0 \Longleftrightarrow S + \sum_{i=1}^{m} \Delta y_i \tilde{A}_i \succeq 0.$$

Moreover, note that if $X(S, \mu) \succeq 0$, then the duality gap at $(X(S, \mu), S)$ is given by

$$\mathbf{Tr}\left(X(S, \mu) S\right) = \mu \mathbf{Tr}\, S^{-1}\left(S - \Delta S\right), \tag{6}$$

by (2). Another important observation is that — in our setting — we will not store the iterates S as matrices, we only store the vectors y.

The dual scaling method uses a dynamic updating strategy for μ, namely

$$\mu := \frac{\mathbf{Tr}\left(XS\right)}{n + \nu\sqrt{n}},$$

where S is the current dual iterate, X is the best-known primal solution, and $\nu \geq 1$ is a given parameter.

Dual scaling algorithm

Input
A strictly feasible primal-dual pair $(X^{(0)}, S^{(0)})$;

Parameters
An accuracy parameter $\epsilon > 0$;
A parameter $\nu \geq 1$;

begin
while $\mathbf{Tr}\left(X^{(k)}S^{(k)}\right) > \epsilon$ **do** for $k = 0, 1, \ldots$
$\mu_k := \frac{\mathbf{Tr}\,(X^{(k)}S^{(k)})}{n+\nu\sqrt{n}}$;
Obtain $\Delta y^{(k)}$ by solving (4);
if $X\left(S^{(k)}, \mu_k\right) \succ 0$ (i.e. **if** $S^{(k)} + \sum_{i=1}^m \Delta y_i^{(k)} \tilde{A}_i \succ 0$) **then**
$X^{(k+1)} := X\left(S^{(k)}, \mu_k\right)$, **else** $X^{(k+1)} := X^{(k)}$;
Choose $\alpha_k > 0$ to satisfy (5);
Let $y^{(k+1)} = y^{(k)} + \alpha_k \Delta y^{(k)}$;
$S^{(k+1)} := S^{(k)} - \alpha_k \sum_{i=1}^m \Delta y_i^{(k)} \tilde{A}_i$;

end
end

Again, the steps in the algorithm that involve $X(S^{(k)}, \mu_k)$ should be interpreted in light of our previous remarks: we do not have to form $X(S^{(k)}, \mu_k)$ explicitly in order to do the dual update, or to decide whether $X(S^{(k)}, \mu_k) \succ 0$. Moreover, the role of the matrix $X^{(k)}$ in the statement of the algorithm is also symbolic — we do not need to store this matrix in an implementation of the algorithm, since we only need the value of the duality gap $\mathbf{Tr}\left(X^{(k)}S^{(k)}\right)$. We can compute the duality gap from (6) if $X^{(k)} \neq X^{(k-1)}$, or from

$$\mathbf{Tr}\left(X^{(k)}S^{(k)}\right) = \mathbf{Tr}\,\tilde{A}_0 X^{(k)} - b^T y^{(k)},$$

if $X^{(k)} = X^{(k-1)}$.

The following complexity result is known for the dual scaling method.

Theorem 2.1 (Ye [11]). *The dual scaling method stops after at most*

$$O\left(\nu\sqrt{n}\log\left(\frac{\mathbf{Tr}\,\left(X^0S^0\right)}{\epsilon}\right)\right)$$

iterations. The output is a primal-dual feasible pair $(X(S,\mu), S)$ *such that* $\mathbf{Tr}\,(X(S,\mu)S) \leq \epsilon$.

We will now discuss the number of operations required per iteration of the dual scaling method, for the case where the data matrices $\tilde{A}_i$ are known to be elements of a (low dimensional) matrix algebra.

3 Summary of the computations per iteration

The computation per iteration can be performed using the following number of flops:

1. Compute S^{-1} in $O(d^3)$ flops;
2. Form the matrix $M_{ij} := \mathbf{Tr}\left(S^{-1}\tilde{A}_i S^{-1}\tilde{A}_j\right)$ $(i,j=1,\ldots,m)$ in $O(m^2d^2)$ flops;
3. Solve the linear system (4) in $O(m^3)$ flops.
4. Check if $X(S,\mu) \succeq 0$ in $O(d^3)$ flops.

Notice that the size n of the data matrices does not appear here. This is very important for the applications we have in mind, since we will typically have $m \ll d \ll n$.

3.1 Computing the inverse of S

Let $S = \sum_\ell \rho_\ell A_\ell$ be of full rank. Then $S^{-1} = \sum_q z_q A_q$ can be computed as follows.

$$A_1 = I = SS^{-1} = (\sum_\ell \rho_\ell A_\ell)(\sum_q z_q A_q) = \\ = \sum_{q,\ell} \rho_\ell z_q (\sum_k \lambda^k_{\ell q} A_k) = \sum_k A_k (\sum_{q,\ell} \rho_\ell z_q \lambda^k_{\ell q}). \quad (7)$$

Thus the equations defining z are

$$\sum_q z_q \sum_\ell \rho_\ell \lambda^k_{\ell q} = 0 \qquad \text{for} \quad k = 2,\ldots,d, \quad (8)$$

and

$$\sum_q z_q \sum_\ell \rho_\ell \lambda^1_{\ell q} = 1.$$

Due to the condition 1, $\lambda^1_{\ell q} = 0$ for all $\ell \neq q'$, and the latter equation simplifies to

$$\sum_q z_q \rho_{q'} \lambda^1_{q q'} = 1. \quad (9)$$

To summarize, S^{-1} can be found by solving the $d \times d$ linear system (8)-(9) in $O(d^3)$ flops.

3.2 Computing the search direction

Here we describe how to form and solve the linear system (4) to obtain the search direction.

First we describe computing, for $1 \leq i, j \leq \tilde{d}$,

$$M_{ij} = M_{ij}(S) := \mathbf{Tr}\, S^{-1}\tilde{A}_i S^{-1}\tilde{A}_j.$$

Keeping the notation $S^{-1} = \sum_q z_q A_q$, we get

$$S^{-1}A_iS^{-1}A_j = (\sum_k(\sum_q z_q\lambda_{qi}^k)A_k)(\sum_p(\sum_\ell z_\ell\lambda_{\ell j}^p)A_p) = \\ = \sum_{k,p}(\sum_q z_q\lambda_{qi}^k)(\sum_\ell z_\ell\lambda_{\ell j}^p)(\sum_r \lambda_{kp}^r A_r). \quad (10)$$

As $\mathbf{Tr}\, A_r = 0$ for all $r > 1$, only $r = 1$ in the formula above will contribute to the trace, i.e.

$$\mathbf{Tr}\, S^{-1}A_iS^{-1}A_j = \left(\sum_{l,k=1}^d \beta_j^{(l)}\beta_i^{(k)}\right)\mathbf{Tr}\sum_{k,p}(\sum_q z_q\lambda_{qi}^k)(\sum_\ell z_\ell\lambda_{\ell j}^p)\lambda_{kp}^1 I) = \\ = \left(\sum_{l,k=1}^d \beta_j^{(l)}\beta_i^{(k)}\right) n\sum_k(\sum_q z_q\lambda_{qi}^k)(\sum_\ell z_\ell\lambda_{\ell j}^{k'})\lambda_{kk'}^1 = z^T\Phi_{ij}z, \quad (11)$$

where the second line follows from the first by observing that $\lambda_{kp}^1 = 0$ for all $k \neq p'$, and

$$(\Phi_{ij})_{q\ell} := n\sum_k \lambda_{kk'}^1\lambda_{qi}^k\lambda_{\ell j}^{k'}.$$

Using $\tilde{A}_i = \sum_{k=1}^d \beta_k^{(i)}A_k$ $(i = 0, \ldots, d)$, we have that

$$S^{-1}\tilde{A}_iS^{-1}\tilde{A}_j = S^{-1}\left(\sum_{k=1}^d \beta_k^{(i)}A_k\right)S^{-1}\left(\sum_{l=1}^d \beta_l^{(j)}A_l\right) = \sum_{l,k=1}^d \beta_k^{(i)}\beta_l^{(j)}S^{-1}A_kS^{-1}A_l. \quad (12)$$

It follows that

$$M_{ij} := \mathbf{Tr}\left(S^{-1}\tilde{A}_iS^{-1}\tilde{A}_j\right) = \sum_{l,k=1}^d \beta_k^{(i)}\beta_l^{(j)}z^T\Phi_{kl}z = z^T\left(\sum_{l,k=1}^d \beta_k^{(i)}\beta_l^{(j)}\Phi_{kl}\right)z. \quad (13)$$

Note that the matrices $\sum_{l,k=1}^{d} \beta_k^{(i)} \beta_l^{(j)} \Phi_{kl}$ can be computed beforehand.

Thus M_{ij} can be computed in $O(d^2)$ flops, and the entire matrix M in $O(m^2 d^2)$.

Next we need to form the right hand side of the linear system involving M, namely (4). The components of the right hand side vector are given by

$$\begin{aligned} -\frac{1}{\mu} b_j + \mathbf{Tr}\left(\tilde{A}_j S^{-1}\right) &= -\frac{1}{\mu} b_j + \mathbf{Tr} \sum_{k,q=1}^{d} \beta_k^{(j)} z_q A_k A_q \\ &= -\frac{1}{\mu} b_j + \mathbf{Tr} \sum_{k,q=1}^{d} \beta_k^{(j)} z_q \sum_{l} \lambda_{kq}^{l} A_l \\ &= -\frac{1}{\mu} b_j + n \sum_{k,q=1}^{d} \beta_k^{(j)} z_q \lambda_{kq}^{1}, \quad (j = 1, \ldots, m) \end{aligned}$$

where we have again used that $\mathbf{Tr}(A_1) = n$ and $\mathbf{Tr}(A_i) = 0$ $(i \neq 1)$.

3.3 Computing the step length α

We use the step length

$$\alpha = \frac{\gamma}{\sqrt{\Delta y^T M \Delta y}}$$

where $\gamma < 1$ is a fixed positive constant (see (5)). Note that, by (4),

$$\begin{aligned} \Delta y^T (M \Delta y) &= \sum_j \Delta y_j \left(-\frac{1}{\mu} b_j + \mathbf{Tr}\left(\tilde{A}_j S^{-1}\right) \right) \\ &= \sum_j \Delta y_j \left(-\frac{1}{\mu} b_j + n \sum_{k,q=1}^{d} \beta_k^{(j)} z_q \lambda_{kq}^{1} \right). \end{aligned}$$

3.4 Checking if $X(S, \mu) \succeq 0$

Recall that

$$X(S, \mu) \succeq 0 \Longleftrightarrow S + \sum_{i=1}^{m} \Delta y_i \tilde{A}_i \succeq 0.$$

Using $S = \sum_\ell \rho_\ell A_\ell$ and $\tilde{A}_i = \sum_{k=1}^d \beta_k^{(i)} A_k$, we get

$$\begin{aligned} S + \sum_{i=1}^m \Delta y_i \tilde{A}_i &= \sum_\ell \rho_\ell A_\ell + \sum_{i=1}^m \left(\Delta y_i \sum_{k=1}^d \beta_k^{(i)} A_k \right) \\ &= \sum_{k=1}^d \left(\rho_k + \left(\sum_{i=1}^m \Delta y_i \beta_k^{(i)} \right) \right) A_k. \end{aligned}$$

We now use the regular representation of $\mathcal{A}$ as follows:

$$\phi \left(\sum_{k=1}^d \left(\rho_k + \left(\sum_{i=1}^m \Delta y_i \beta_k^{(i)} \right) \right) A_k \right) = \sum_{k=1}^d \left(\rho_k + \left(\sum_{i=1}^m \Delta y_i \beta_k^{(i)} \right) \right) B_k.$$

Since ϕ preserves the spectrum (up to multiplicities), we only need to check if the smallest eigenvalue of the $d \times d$ matrix $\sum_{k=1}^d \left(\rho_k + \left(\sum_{i=1}^m \Delta y_i \beta_k^{(i)} \right) B_k \right)$ is nonnegative.

3.5 Updating the duality gap

Case 1: $X(S, \mu) \succeq 0$

If $X(S, \mu) \succeq 0$, then $X(S, \mu)$ is a feasible solution of the primal problem and the duality gap at $(X(S, \mu), S)$ is given by

$$\begin{aligned} \mathbf{Tr}\left(X(S,\mu)S\right) &= \mu \mathbf{Tr}\left(S^{-1}\left(S - \Delta S\right)\right) = \mu n - \mu \mathbf{Tr}\left(S^{-1}\Delta S\right) \\ &= \mu n + \mu \mathbf{Tr}\left(\sum_i z_i A_i \sum_k \Delta y_k \tilde{A}_k \right) \\ &= \mu n + \mu \mathbf{Tr}\left(\sum_i z_i A_i \sum_k \Delta y_k \sum_{j=1}^d \beta_j^{(k)} A_j \right) \\ &= \mu n + \mu \mathbf{Tr}\left(\sum_{i,j,k} z_i \Delta y_k \beta_j^{(k)} A_i A_j \right) \\ &= \mu n + \mu \mathbf{Tr}\left(\sum_{i,j,k} z_i \Delta y_k \beta_j^{(k)} \sum_t \lambda_{ij}^t A_t \right) \\ &= \mu n + \mu n \sum_{i,j,k} z_i \Delta y_k \beta_j^{(k)} \lambda_{ij}^t, \end{aligned}$$

where we have again used that $\mathbf{Tr}\left(A_1\right) = n$ and $\mathbf{Tr}\left(A_i\right) = 0$ $(i \neq 1)$.

Case 2: $X(S,\mu) \not\succeq 0$

In this case the duality gap is computed by updating the dual objective value only. The change in dual objective value is $b^T \Delta y$.

4 Conclusion

We have shown that the dual scaling method for semidefinite programming can be implemented to exploit the particular data structure where the SDP data matrices come from a low dimensional matrix algebra. In this case the computational complexity per iteration only depends on the dimension of the algebra, and not on the sizes of the original data matrices.

In the second part of this paper, we will present computational results for SDP's that arise from the lower bounding of the crossing numbers of complete bipartite graphs, as described in [5].

References

[1] K. Gatermann and P.A. Parrilo, Symmetry groups, semidefinite programs, and sums of squares, *J. Pure Appl. Algebra*, 192, 95–128, 2004.

[2] K. Fujisawa, M. Kojima and K. Nakata. Exploiting Sparsity in Primal-Dual Interior-Point Methods for Semidefinite Programming. *Mathematical Programming, Series B*, 79, 235-253, 1997.

[3] M.X. Goemans and F. Rendl. Semidefinite Programs and Association Schemes, *Computing*, 63, 331–340, 1999.

[4] E. de Klerk, D.V. Pasechnik. Upper bounds on the stability number of an orthogonality graph via semidefinite programming. Preprint, 2004.

[5] E. de Klerk, J. Maharry, D.V. Pasechnik, B. Richter, and G. Salazar. Improved bounds for the crossing numbers of $K_{m,n}$ and K_n. E-print **math.CO/0404142** at `arXiv.org`, submitted for publication.

[6] L. Lovász. On the Shannon capacity of a graph. *IEEE Trans. on Information Theory*, 25:1–7, 1979.

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

[8] A. Schrijver, A comparison of the Delsarte and Lovász bounds. *IEEE Trans. Inform. Theory*, **25**(1979), 425–429.

[9] A. Schrijver. New code upper bounds from the Terwilliger algebra. Preprint, 2004. Available at `http://homepages.cwi.nl/~lex/`

[10] J.F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11-12:625–653, 1999.

[11] Y. Ye. *Interior point algorithms: theory and analysis*. John Wiley and Sons, New York, 1997.