A Subspace Semidefinite Programming for Spectral Graph Partitioning

Suely Oliveira[1], David Stewart[2], and Takako Soma[1]

[1] The Department of Computer Science
The University of Iowa, Iowa City, IA 52242, USA
{oliveira, tsoma}@cs.uiowa.edu
[2] The Department of Mathematics
The University of Iowa, Iowa City, IA 52242, USA
dstewart@math.uiowa.edu

Abstract. A semidefinite program (SDP) is an optimization problem over $n \times n$ symmetric matrices where a linear function of the entries is to be minimized subject to linear equality constraints, and the condition that the unknown matrix is positive semidefinite. Standard techniques for solving SDP's require $O(n^3)$ operations per iteration. We introduce subspace algorithms that greatly reduce the cost os solving large-scale SDP's. We apply these algorithms to SDP approximations of graph partitioning problems. We numerically compare our new algorithm with a standard semidefinite programming algorithm and show that our subspace algorithm performs better.
Keywords: semidefinite programming, subspace methods

1 Introduction

A semidefinite program (SDP) [1, 18, 22] is the problem of minimizing a linear function over symmetric matrices with linear constraints and the constraint that the matrix is positive semi-definite. That is,

$$\min_X C \bullet X \quad \text{subject to} \tag{1}$$

$$A_i \bullet X = a_i, \quad i = 1, 2, \ldots, m, \tag{2}$$

$$X \succeq 0 \tag{3}$$

where $A \bullet B = \text{trace}(A^T B) = \sum_{i,j} a_{ij} b_{ij}$ and $A \succeq B$ means that $A - B$ is a positive semi-definite matrix. Note that A and B are symmetric, then $A \bullet B = \text{trace}(AB)$. This is essentially an ordinary linear program where the non-negativity constraint is replaced by a semidefinite constraint on matrix variables. Semidefinite programming reduces to linear program when all the matrices are diagonal. Semidefinite programs and linear programs are special instances of a more general problem class called conic linear programs, where one seeks to minimize a linear objective function subject to linear constraints and a constraint that the unknown lies in a given closed convex cone. Both the cone of semidefinite

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2329, pp. 1058–1067, 2002.

matrices for semidefinite programming, and the non-negative orthant for linear program, are homogeneous, self-dual cones.

One of the main aspects in which semidefinite programming differs from linear program is that the non-negative orthant is a polyhedral cone, whereas the semidefinite cone is not. Thus, developing simplex type algorithms for semidefinite programming is a difficult task. However, it is fairly straightforward to design polynomial time primal-dual interior-point algorithms for these problems. Currently various software packages are available for solving semidefinite programs using interior-point algorithms, such as CSDP [4], SDPA [7], SDPpack [2], SDPT3 [21], SP [23], and a Matlab toolbox by Rendl [17].

Subspace methods have been developed to solve system of linear equations [11] and to compute eigenvalues [3]. In this paper we have developed a method for solving large semidefinite programming problems using subspaces. Subspace methods have been use to compute a few of the extreme eigenvalues and the corresponding eigenvectors of a large, sparse, symmetric matrices, such as the Lanczos method, which is based on Krylov subspaces [14], and the Davidson method which uses Rayleigh matrices [6]. Generalized Davidson [19] and Jacobi–Davidson type algorithms [20] have been introduced, and theoretical studies have been done in [5, 12]. Davidson type subspace methods have been applied to solve graph partitioning problem [9, 10] and extended eigenproblem [13]. Here we use subspace methods to solve semidefinite programs.

Semidefinite programming has been an active research area in mathematics and engineering. In particular, many hard optimization problems with integer constraints can be relaxed to a problem with convex quadratic constraints which can be formulated as a semidefinite program (see, for example, [15]). These semidefinite programs provide approximations to the original, hard problem which can usually be solved in polynomial time. Usually, approximations from semidefinite programming relaxations are better than those from linear programming. In particular, we will apply semidefinite programming to the graph partitioning problem.

Graph partitioning is universally employed in the parallelization of calculations on unstructured grids including finite element and finite difference techniques. In many calculations, the underlying computational structure can be conveniently modeled as a graph in which vertices correspond to computational tasks and edges reflect data dependencies. Once a graph model of a computation is constructed, graph partitioning can be used to determine how to divide the work and data for an efficient parallel computation. The goal of the graph partitioning problem is to divide the graph into equally weighted sets in such a way that the weight of the edges crossing between sets is minimized. In other words, the graph partitioning is used to evenly distribute the computations among the processors while minimizing interprocessor communication, so that the corresponding assignment of tasks to processors leads to efficient execution. In general, computing the optimal partitioning is an NP-hard problem. Therefore, heuristics need to be used to get approximate solutions for these problems. The graph partitioning problem for high performance scientific computing has

been studied extensively over the past decade. While graph partitioning problems are NP-hard, in practice they are also large-scale, which can make even the application of heuristics difficult.

This paper is organized as follows. Section 2 describes semidefinite programming. The new method of solving the semidefinite programming using subspace algorithms is introduced in Section 3, and Section 4 discusses the algorithm for the new method. In Section 5, numerical results are presented.

2 Semidefinite Programming

A semidefinite program (SDP) is an optimization problem of the following form: Let C be a given symmetric $n \times n$ matrix, $a \in \Re^m$ and $\mathcal{A}$ be a linear operator that maps symmetric matrices of size n into vectors in $\Re^m$ given by $\mathcal{A}(X)_i = A_i \bullet X$. Note that the adjoint of this operator is $\mathcal{A}^T(y) = \sum_{i=1}^m y_i A_i$.

$$\textbf{Primal Problem}: \min_X \operatorname{tr}(CX) \qquad \text{subject to (a) } \mathcal{A}(X) - a = 0, \quad \text{(b) } X \succeq 0. \tag{4}$$

The general duality theory for semidefinite programs has been studied [24, 16]. We derive the Lagrangian dual to semidefinite program directly. Introducing the Lagrange multiplier $y \in \Re^m$ for the equality constraints, we see that

$$\begin{aligned} \mathcal{L}(X, y) &= C \bullet X - y^T(\mathcal{A}(X) - a) - M \bullet X \\ &= (C - \mathcal{A}^T(y) - M) \bullet X + y^T a, \end{aligned}$$

and the analogues of the Kuhn–Tucker conditions for optimality are

$$\begin{gathered} \nabla_X \mathcal{L}(X, y, M) = C - \mathcal{A}^T(y) - M = 0, \\ M,\ X \succeq 0, \quad M \bullet X = 0, \qquad \mathcal{A}(X) = a. \end{gathered}$$

Note that $\nabla_X f(X)$ is the matrix whose (i, j) entry is $\partial f / \partial x_{ij}(X)$. Thus, we get the dual semidefinite program

$$\textbf{Dual Problem}: \max_y a^T y \qquad \text{subject to (a) } C - \mathcal{A}^T(y) \succeq 0. \tag{5}$$

It is assumed that there exits (X, y) such that X is positive semidefinite, $\mathcal{A}(X) = a$, and $C - \mathcal{A}^T(y)$ is positive semidefinite. These points are called feasible points. In this case it is easy to show weak duality: that is, the minimum value of primal problem (4) is greater or equal to the maximum value of the dual problem (5). If X is also positive definite, then it is a feasible interior point. In this case, Slater's theorem applies (see, e.g., [16]) and strong duality holds; that is, the minimum value of the primal problem (4) is equal to the maximum value of the Lagrange dual problem (5).

The theoretical basis of interior-point algorithms is presented in [8].

3 Subspace Semidefinite Programming

We present a subspace algorithm for solving the semidefinite program (4). The optimization problem being (4) and (5), we introduce Lagrange multipliers y and M for (4) and look for stationary points of the function

$$\mathcal{L}(X, y, M) = C \bullet X - y^T(\mathcal{A}(X) - a) - M \bullet X. \tag{6}$$

Taking the partial derivative of $\mathcal{L}$ with respect to the components of X, we obtain the residual R:

$$\nabla_X \mathcal{L}(X, y, M) = C - \mathcal{A}^T(y) - M = R.$$

At the optimal (X, y), we note that $R = 0$. Also note that the modified residual matrix $\widetilde{R} = C - \mathcal{A}^T(y) = M$, is positive semidefinite at the optimal pair (X, y). Taking the partial derivative of $\mathcal{L}$ with respect to the components of y, we obtain $\nabla_y \mathcal{L}(X, y, M) = \mathcal{A}(X) - a$.

Suppose that V is an $n \times k$ matrix of orthonormal columns and W is an $m \times l$ matrix of orthonormal columns. These represent subspaces of $\Re^n$ of dimensions k and l respectively. We can represent $\hat{X}$ in subspace V with $\hat{X} = V^T X V$ and $\hat{y}$ in subspace W with $\hat{y} = W^T y$, we can rewrite (4) as

$$\begin{aligned} \min_{\hat{X}} C \bullet (V\hat{X}V^T), \qquad & \text{subject to (a) } \mathcal{A}(V\hat{X}V^T) - a = 0, \\ & \text{(b) } V\hat{X}V^T \succeq 0. \end{aligned} \tag{7}$$

Note that $X = V\hat{X}V^T \succeq 0$ is and only if $\hat{X} \succeq 0$. This subspace SDP (7) effectively reduces the number of variables. In the dual problem (5) we can set $y = W\hat{y}$ to restrict y to a subspace. This gives the following dual subspace problem:

$$\max_{\hat{y}} a^T(W\hat{y}) \qquad \text{subject to (a) } C - \mathcal{A}^T(W\hat{y}) \succeq 0. \tag{8}$$

This dual subspace problem (8) effectively reduces the number of constraints.

Combining the subspace problems to reduce both the number of variables (in the primal problem), and the number of constraints, pre-multiply the constraint (a) in (7) by W. We introduce new operators: Let $\hat{\mathcal{A}}$ be the linear operator $\hat{\mathcal{A}}(\hat{X}) = \mathcal{A}(V\hat{X}V^T)$, that maps symmetric matrices of size $n \times n$ into vectors in $\Re^m$, and $\hat{\hat{\mathcal{A}}}$ be the linear operator $\hat{\hat{\mathcal{A}}}(\hat{X}) = W^T \hat{\mathcal{A}}(\hat{X})$ that maps symmetric matrices of size $n \times n$ into vectors in $\Re^l$.

$$\mathcal{A}(V\hat{X}V^T) = \begin{pmatrix} \operatorname{tr}(A_1 V\hat{X}V^T) \\ \vdots \\ \operatorname{tr}(A_m V\hat{X}V^T) \end{pmatrix} = \begin{pmatrix} \operatorname{tr}(V^T A_1 V\hat{X}) \\ \vdots \\ \operatorname{tr}(V^T A_m V\hat{X}) \end{pmatrix} = \begin{pmatrix} \hat{A}_1 \bullet \hat{X} \\ \vdots \\ \hat{A}_m \bullet \hat{X} \end{pmatrix} = \hat{\mathcal{A}}(\hat{X})$$

where $\hat{A}_i = V^T A_i V$ for $i = 1, 2, \ldots, m$. Also,

$$W^T(\hat{\mathcal{A}}(\hat{X})) = \begin{pmatrix} \sum_j (W_{j1}\hat{A}_j) \bullet \hat{X} \\ \vdots \\ \sum_j (W_{jl}\hat{A}_j) \bullet \hat{X} \end{pmatrix} = \begin{pmatrix} \hat{\hat{A}}_1 \bullet \hat{X} \\ \vdots \\ \hat{\hat{A}}_l \bullet \hat{X} \end{pmatrix} = \hat{\hat{\mathcal{A}}}(\hat{X}).$$

Since $W^T(\mathcal{A}(V\hat{X}V^T) - a) = W^T(\hat{\mathcal{A}}(\hat{X}) - a) = W^T(\hat{\mathcal{A}}(\hat{X})) - W^T a = \hat{\hat{\mathcal{A}}}(\hat{X}) - W^T a$, (7) becomes the problem with both reduced number of primal variables and reduced number of constraints:

$$\min_{\hat{X}} V^T C V \bullet \hat{X} \qquad \text{subject to (a) } \hat{\hat{\mathcal{A}}}(\hat{X}) - W^T a = 0 \quad \text{(b) } \hat{X} \succeq 0. \tag{9}$$

We introduce Lagrange multipliers $\hat{y}$ and N and look for stationary points of the function

$$\hat{\mathcal{L}}(\hat{X}, \hat{y}, N) = V^T C V \bullet \hat{X} - \hat{y}^T(\hat{\hat{\mathcal{A}}}(\hat{X}) - W^T a) - N \bullet \hat{X}$$

Taking the partial derivative of $\hat{\mathcal{L}}$ with respect to the components of $\hat{X}$, we obtain

$$\begin{aligned} \nabla_{\hat{X}} \hat{\mathcal{L}}(\hat{X}, \hat{y}, N) &= V^T(C - \mathcal{A}^T(y) - \hat{M})V \\ &= V^T(\nabla_X \mathcal{L}(X, y, \hat{M}))V = V^T R V \end{aligned}$$

where $\hat{M} = VNV^T$ is an approximation of M. Taking the partial derivative of $\hat{\mathcal{L}}$ with respect to the components of $\hat{y}$, we obtain

$$\begin{aligned} \nabla_{\hat{y}} \hat{\mathcal{L}}(\hat{X}, \hat{y}, N) &= \hat{\hat{\mathcal{A}}}(\hat{X}) - W^T a = W^T(W\hat{\hat{\mathcal{A}}}(\hat{X}) - a) \\ &= W^T(\hat{\mathcal{A}}(\hat{X}) - a) = W^T(\mathcal{A}(V\hat{X}V^T) - a) \\ &= W^T(\mathcal{A}(X) - a) = W^T \nabla_y \mathcal{L}(X, y, M). \end{aligned}$$

Projecting the constraint (a) in (8) on the subspace V, and since $V^T(\mathcal{A}^T(W\hat{y}) - C)V = V^T((W^T\mathcal{A})^T(\hat{y}) - C)V = V^T((W^T\mathcal{A})^T(\hat{y}))V - V^T C V = (W^T\hat{\mathcal{A}})^T(\hat{y}) - V^T C V = \hat{\hat{\mathcal{A}}}^T(\hat{y}) - V^T C V$, we have the Lagrangian dual of (9)

$$\max_{\hat{y}} (W^T a)^T \hat{y} \qquad \text{subject to } (\hat{\hat{\mathcal{A}}})^T(\hat{y}) - V^T C V \succeq 0. \tag{10}$$

Below we present the main steps of our subspace algorithm for semidefinite programming.

Algorithm 1 – Subspace Algorithm for Semidefinite Programming

1. Define the data for the problem: C an $n \times n$ matrix, a and m-dimensional vector, and an operator $\mathcal{A}$ where $\mathcal{A}(X)_i = A_i \bullet X$.
2. Define $n \times 1$ vector V_1 and $m \times 1$ vector W_1.
3. Compute an interior feasible starting point, $\hat{X}$ and $\hat{y}$, in subspace. $\hat{X}$ is found by solving feasibility problem. $\hat{y} = W_j^T y$, where W_j is the current orthogonal basis.
4. Compute the semidefinite programming solution $(\hat{X}, \hat{y})$ on the subspace. The projected matrix, operator and vector on the subspace are $\hat{C} = V_j^T C V_j$, $\hat{\hat{\mathcal{A}}} = W_j^T(\hat{\mathcal{A}}) = W_j^T \begin{pmatrix} V_j^T A_1 V_j \\ \vdots \\ V_j^T A_m V_j \end{pmatrix}$ and $\hat{a} = W_j^T a$, where V_j and W_j are the current orthogonal basis. smallest value.

5. Compute $\tilde{R} = C - \mathcal{A}^T(y)$ (or some representation of it), and compute the eigenvector r of $\tilde{R}$ with the minimal eigenvalue.
6. Orthonormalize r against the current orthogonal basis V_j. Append the orthonormalized vector to V_j to give V_{j+1}.
7. Estimate residual $p = \mathcal{A}(X) - a = \mathcal{A}(V_j \hat{X} V_j^T) - a$.
8. Orthonormalize p against the current orthogonal basis W_j. Append the orthonormalized vector to W_j to give W_{j+1}.

4 Implementation

The main motivation to study this kind of problem comes from applications in discrete optimization. In particular we will investigate powerful and tractable relaxation of the min-cut problem.

The min-cut problem is the problem of partitioning the node set of an edge-weighted undirected graph into two parts so as to minimize the total weight of edges cut by the partition. We assume that the graph in question is complete (if not, non existing edges can be given weight 0 to complete the graph). Mathematically, the problem can be formulated as follows. Let the graph be given by its weighted adjacency matrix A. Define the matrix $L = \mathrm{diag}(Ae) - A$, where e is the vector of all ones. The matrix L is called the Laplacian matrix associated with the graph. If a cut S is represented by a vector x where $x_i \in \{-1, 1\}$ depending on whether or not $i \in S$, we get the following formulation for the min-cut problem, where $C = -L$.

$$\min_x \frac{1}{4} x^T C x \qquad \begin{array}{l} \text{subject to (a) } x \in \{-1,1\}^n, \\ \qquad\qquad \text{(b) } e^T x = 0. \end{array} \tag{11}$$

Using $X = \frac{1}{4} x x^T$, this is equivalent to

$$\min_X C \bullet X \qquad \begin{array}{l} \text{subject to (a) } \mathrm{diag}(X) = \frac{1}{4} e, \\ \qquad\qquad \text{(b) } \mathrm{rank}(X) = 1, \\ \qquad\qquad \text{(c) } X \bullet (ee^T) = 0, \\ \qquad\qquad \text{(d) } X \succeq 0. \end{array} \tag{12}$$

Dropping the rank condition we obtain a problem of the form SDP with $a = \frac{1}{4} e$ and $A(X) = \mathrm{diag}(X)$.

Algorithm 1 has been implemented using CSDP developed by Borchers [4] to solve min-cut problem.

5 Numerical Results

We compared our new subspace algorithm with the original semidefinite programming. The new algorithms have been implemented and run on a HP VISUALIZE Model C240 workstation, with a 236MHz PA-8200 processor and 512MB

RAM. The existing semidefinite software, CSDP [4], was used for solving the problem in a subspace, and for our comparison study. Also note that the matrices in this implementation are all explicitly represented as dense matrices.

The following is the graph partitioning problem that we would like to solve:

$$\begin{aligned} \min_X C \bullet X \qquad & \text{subject to (a) } \mathcal{A}(X) = \tfrac{1}{4} \\ & \text{(b) } X \bullet (ee^T) = 0, \\ & \text{(c) } X \succeq 0 \end{aligned} \tag{13}$$

There are n constraints from (a) in (13). We rewrite the problem so that we can reduce the number of constraints in original problem, m. The operator $\mathcal{A}$ is defined in terms of m diagonal matrices A_i where $(A_i)_{jj}$ is one if $j \equiv i \pmod m$, and zero otherwise, $i = 1, \cdots, m$, and $m \leq n$:

$$\begin{aligned} \min_X C \bullet X \qquad & \text{subject to (a) } \mathcal{A}(X) = (n/4m)e \\ & \text{(b) } X \bullet (ee^T) = 0, \\ & \text{(c) } X \succeq 0 \end{aligned} \tag{14}$$

where e is the vector of ones of the appropriate size. For example, if $m = 1$ is chosen for the number of constraints in original problem, the minimization problem can be written as:

$$\begin{aligned} \min_X C \bullet X \qquad & \text{subject to (a) } \mathcal{A}(X) = \tfrac{1}{4}n \\ & \text{(b) } X \bullet (ee^T) = 0, \\ & \text{(c) } X \succeq 0 \end{aligned} \tag{15}$$

where $\mathcal{A}(X) = I \bullet X$. Note that this is equivalent to spectral partitioning. Therefore, (a) in (15) is same as $\text{trace}(X) = \frac{1}{4}n$. If the unknown is a 9×9 matrix and $m = 3$, the first constraint in (14) would be $\mathcal{A}(X) = \frac{3}{4}e \in \Re^3$ and $\mathcal{A}$ is given in terms of 3 symmetric matrices: $A_1 = \text{diag}(1,0,0,1,0,0,1,0,0)$, $A_2 = \text{diag}(0,1,0,0,1,0,0,1,0)$, and $A_3 = \text{diag}(0,0,1,0,0,1,0,0,1)$.

Both algorithms were run with square matrices of various sizes. Figure 1 compares the observed running timings. The vertical-axis shows the timings in seconds. The horizontal-axis is the size of the matrices, 9, 25, 100, 400, and 900. The subspace was expanded until the stopping criteria have been met, which were a constraint for the primal problem, $\text{norm}(\mathcal{A}(X) - a) = \text{norm}(p) < 10^{-3}$, and the difference between primal and dual problems (the duality gap), is less than 10^{-7}. Table 1 summarizes the size of subspace when it reached the stopping criteria and the number of constraints. From this graph we can see that the subspace algorithm takes less than the original algorithm for all the test cases.

Figure 2 shows $\text{norm}(p)$ when the subspace were expanded in the subspace method. We used a fixed 400 by 400 matrix, and the number of constraints used was 10. The horizontal-axis is number of iterations, which describes the size of the subspace. The figure shows that after three iterations $\text{norm}(p)$ is reasonably small. This means that in order to find the solution of the original problem using the subspace method, only 10 constraints were needed and the problem was solved the problem until the size of the subspace reaches 3.

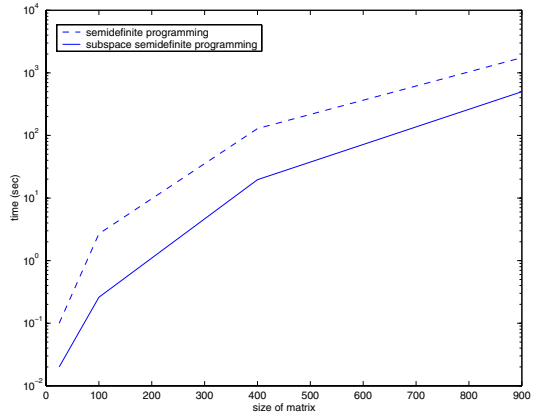

Fig. 1. Timings comparing subspace semidefinite programing against semidefinite programming.

Table 1. Timings comparing subspace semidefinite programing.

size of matrix	25	100	400	900
size of subspace	2	2	3	3
number of constraints	5	10	10	10

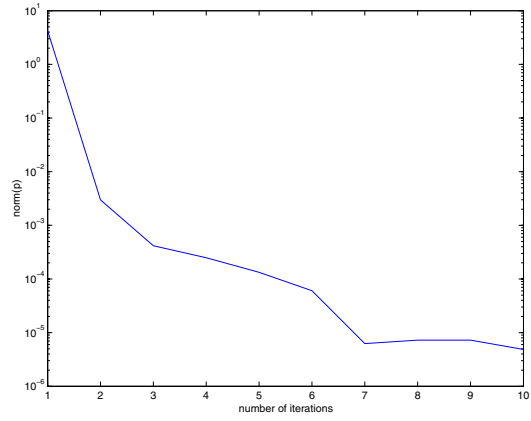

Fig. 2. norm(p) of subspace semidefinite programming.

Figure 3 shows the timings when the number of constrains were changed. We used the fixed size of the matrix, 400 by 400, and the program was run until the stopping criteria described above have been satisfied. In this example all the criteria were met when the size of the subspace had reached to 3 by 3. The number of constraints used was 10, 20, 40, 80, 200, and 400 which describes horizontal-axis. The figure shows the timings increases as the number of constraints increases.

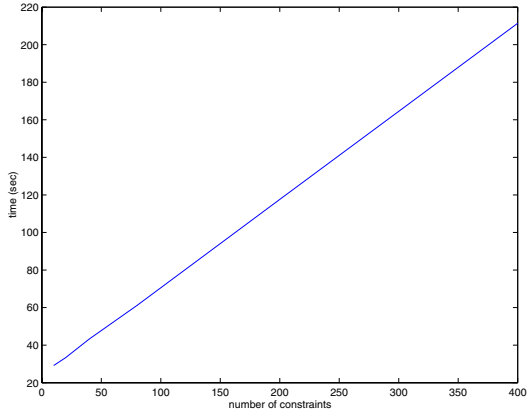

Fig. 3. Timings of subspace semidefinite programming using different number of constraints.

References

1. F. Alizadeh. Interior-point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5(1):13–51, 1995.
2. F. Alizadeh, J.-P. A. Haeberly, M. V. Nayakkankuppam, M. L. Overton, and S. Schmieta. SDPpack user's guide — version 0.9 beta for Matlab 5.0. Technical Report TR1997-737, Computer Science Department, New York University, New York, NY, June 1997.
3. W. E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenproblem. *Quarterly Applied Mathematics*, 9:17–29, 1951.
4. B. Borchers. CSDP, 2.3 User's Guide. *Optimization Methods and Software*, 11(1):597–611, 1999.
5. M. Crouzeix, B. Philippe, and M. Sadkane. The Davidson method. *SIAM J. Sci. Comput.*, 15(1):62–76, 1994.
6. E. R. Davidson. The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices. *J. Comp. Phys.*, 17:87–94, 1975.
7. K. Fujisawa, M. Kojima, and K. Nakata. SDPA User's Manual — Version 4.50. Technical Report B, Department of Mathematical and Computing Science, Tokyo Institute of Technology, Tokyo, Japan, July 1999.

8. C. Helmberg, F. Rendl, R. J. Vanderbei, and H. Wolkowicz. An interior-point method for semidefinite programming. *SIAM J. Optim.*, 6(2):342–361, 1996.
9. M. Holzrichter and S. Oliveira. A graph based Davidson algorithm for the graph partitioning problem. *International Journal of Foundations of Computer Science*, 10:225–246, 1999.
10. M. Holzrichter and S. Oliveira. A graph based method for generating the Fiedler vector of irregular problems. In *Lecture Notes in Computer Science*, volume 1586, pages 978–985. Springer, 1999. Proceedings of the 11th IPPS/SPDP'99 workshops.
11. C. Lanczos. Solution of systems of linear equations by minimized iterations. *J. Research Nat'l Bureau of Standards*, 49:33–53, 1952.
12. S. Oliveira. On the convergence rate of a preconditioned subspace eigensolver. *Parallel Algorithms and Applications*, 63:219–231, 1999.
13. S. Oliveira and T. Soma. A multilevel algorithm for spectral partitioning with extended eigen-models. In *Lecture Notes in Computer Science*, volume 1800, pages 477–484. Springer, 2000. Proceedings of the 15th IPDPS 2000 workshops.
14. B. N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice-Hall, 1980.
15. S. Poljak, F.Rendl, and H. Wolkowicz. A recipe for semidefinite relaxation for (0,1)-quadratic programming. *J. Global Optim.*, 7(1):51–73, 1995.
16. M. V. Ramana. An exact duality theory for semidefinite programming and its complexity implications. *Mathematical Programming, Ser. B*, 77(2):129–162, 1997.
17. F. Rendl. A Matlab toolbox for semidefinite programming. Technical report, Technische Universität Graz, Institut für Mathematik, Kopernikusgasse 24, A-8010 Graz, Austria, 1994.
18. F. Rendl, R. J. Vanderbei, and H. Wolkowicz. Max-min eigenvalue problems, primal-dual interior point algorithms, and trust region subproblems. *Optimization Methods and Software*, 5:1–16, 1995.
19. Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Manchester University Press, Oxford Road, Manchester M13 9PL, UK, 1992.
20. G. L. G. Sleijpen and H. A. Van der Vorst. A Jacobi-Davidson iteration method for linear eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 17(2):401–425, 1996.
21. K. C. Toh, M. J. Todd, and P. H. Tütüncü. SDPT3 — a Matlab software package for semidefinite programming, version 2.1. Technical report, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, September 1999.
22. L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, 1996.
23. L. Vandenberghe and S. Boyd. SP Software for semidefinite programming User's guide, version 1.0. Technical report, Information System Laboratory, Stanford University, Stanford, CA, November 1998.
24. H. Wolkowicz. Some applications of optimization in matrix theory. *Linear Algebra and its Applications*, 40:101–118, 1981.