A Boundary Point Method to solve Semidefinite Programs *

J. Povh
School of Business and Management
8000 Novo Mesto, Slovenia
janez.povh@guest.arnes.si

F. Rendl, A. Wiegele
Universität Klagenfurt
Institut für Mathematik
9020 Klagenfurt, Austria
franz.rendl@uni-klu.ac.at, angelika.wiegele@uni-klu.ac.at

August 30, 2006

Abstract

We investigate the Augmented Lagrangian Penalty function approach to solve Semidefinite Programs. It turns out that this method generates iterates which lie on the boundary of the cone of semidefinite matrices which are driven to the affine subspace described by the linear equations defining the semidefinite program. We provide some computational experience with this method and show in particular, that it allows to compute the theta number of a graph to reasonably high accuracy for instances which are beyond reach by other methods.

1 Introduction

We consider the following primal-dual pair of semidefinite problems (SDP) given by symmetric matrices C and $A_i, i = 1, \ldots, m$ of order n and a vector $b \in \mathbb{R}^m$.

$$(P) \quad \max\langle C, X\rangle \quad \text{such that } A(X) = b,\ X \succeq 0.$$

The dual to this problem is given by

$$(D) \quad \min b^T y \quad \text{such that } A^T(y) - C = Z \succeq 0.$$

The linear function $A(X)$ maps (symmetric) matrices to $\mathbb{R}^m$ with $(A(X))_i = \langle A_i, X\rangle$. Its adjoint is well known to be $A^T(y) = \sum_i y_i A_i$.

Under the assumption that both problems have strictly feasible points, it is well known that strong duality holds. In this case (X, y, Z) is optimal if and only if

$$X \succeq 0,\ A(X) = b, \quad Z \succeq 0,\ A^T(y) - Z = C, \quad ZX = 0. \tag{1}$$

It is widely accepted that the currently most efficient way to solve SDP consists in applying some variant of the primal-dual path-following technique to the system (1) with $ZX = 0$ replaced by $ZX - \mu I = 0$, or some variant of this condition. We refer to [13] for a recent survey of path-following methods for SDP.

*Supported in part by EU project Algorithmic Discrete Optimization (ADONET), MRTN-CT-2003-504438

There exist several open-source packages to solve SDP, see for instance Hans Mittelmann's website[1]. These algorithms show satisfactory performance for instances with n and m reasonably small such as $n \leq 1000$, $m \leq 10000$. This website reports computational results on SDP instances with $m \leq 7000$ and n in most cases much smaller than 1000. Since m could be $O(n^2)$, the limit $m \leq 10000$ restricts the use of interior point methods significantly.

In this short note we investigate alternatives to interior point methods. We have problems in mind where the size n of the primal matrix is not too large, say $n \leq 1000$, but the number of constraints m can be arbitrary.

Recently, Burer and Vandenbussche [3] have proposed to solve semidefinite programs arising from lift-and-project relaxations of binary integer problems by exploiting the Augmented Lagrangian technique. A similar approach has been investigated by J. Malick [8] to solve semidefinite least squares problems.

We follow this approach and briefly describe in section 2 the augmented Lagrangian technique applied to (D). In section 3 we study the optimality conditions for the Lagrangian subproblem. This leads to an interpretation of this approach as a boundary point method in the sense that the algorithm maintains semidefinite matrices X and Z with $ZX = 0$. Hence we follow the boundary of the cone of semidefinite matrices until we reach the affine subspace described by the linear equations in (1). This is elaborated in section 4. We conclude with computational experience of our method applied to the Lovász theta number.

Notation: We use standard notation from matrix theory and graph theory. The scalar product in the space of matrices is $\langle A, B\rangle := \operatorname{tr}(A^T B)$ with induced norm, often called Frobenius norm, given by $\|A\|_F^2 = \langle A, A\rangle$. The matrix of all ones is denoted by J.

2 An Augmented Lagrangian approach to solve (D)

We apply the augmented Lagrangian method to solve (D). Thus we introduce a Lagrange multiplier X for the dual equations $Z + C - A^T(y) = 0$ and consider for fixed $\sigma > 0$ the augmented Lagrangian L_σ:

$$L_\sigma(y, Z; X) := b^T y + \langle X, Z + C - A^T(y)\rangle + \frac{\sigma}{2}\|Z + C - A^T(y)\|^2.$$

Defining

$$W(y) := A^T(y) - C - \frac{1}{\sigma}X, \tag{2}$$

we rewrite the Lagrangian as $L_\sigma = b^T y + \frac{\sigma}{2}\|Z - W(y)\|^2 - \frac{1}{2\sigma}\|X\|^2$. Let

$$f(y, Z) := b^T y + \frac{\sigma}{2}\|Z - W(y)\|^2. \tag{3}$$

The augmented Lagrangian method to solve (D) consists in minimizing $f(y, Z)$ (approximately), to get y and $Z \succeq 0$. Then X is updated as $X \leftarrow X + \sigma(Z + C - A^T(y))$, see [1], and the whole process is iterated until convergence. An informal description is given in Table 1.

Clearly, the crucial step here is minimizing $f(y, Z)$, so we take a closer look at the optimality conditions of this problem.

3 Optimality conditions for the inner minimization

The problem

$$\min b^T y + \frac{\sigma}{2}\|Z - W(y)\|^2$$

[1]http://plato.asu.edu/ftp/sdplib.html

Select $\sigma > 0,\ X \succeq 0$.
while not done
For X fixed, solve min $L_\sigma(y, Z; X)$ approximately to get y and $Z \succeq 0$.
Update X: $X \leftarrow X + \sigma(Z + C - A^T(y))$
Check stopping condition
end while

Table 1: Augmented Lagrangian method to solve (D)

such that $y \in I\!R^m,\ Z \succeq 0$ is a convex quadratic SDP. After introducing a Lagrange multiplier $V \succeq 0$ for the constraint $Z \succeq 0$, we get its Lagrangian

$$L(y, Z; V) := f(y, Z) - \langle V, Z\rangle,$$

and the following KKT necessary conditions for optimality:

$$\nabla_y L = b - \sigma A(Z - A^T(y) + C + \frac{1}{\sigma}X) = 0,$$

$$\nabla_Z L = \sigma(Z - A^T(y) + C + \frac{1}{\sigma}X) - V = 0, \quad V \succeq 0,\ Z \succeq 0,\ VZ = 0.$$

Since the problem is convex with the Slater condition holding, these conditions are also sufficient for optimality. Expanding the gradient conditions, we note that y, Z is optimal if and only if there exists V such that

$$A(A^T(y)) = A(Z + C + \frac{1}{\sigma}X) - \frac{1}{\sigma}b \tag{4}$$

$$V = \sigma Z - \sigma W(y), \quad Z \succeq 0,\ V \succeq 0,\ VZ = 0. \tag{5}$$

For y fixed, the problem $\min_{Z\succeq 0}\ f(y, Z)$ is a projection onto the cone of semidefinite matrices. Therefore, Z must also satisfy the projection condition

$$Z = W(y)_+ := \operatorname{argmin}_{U\succeq 0}\|W(y) - U\|. \tag{6}$$

It is well known that W_+ can be computed from the eigenvalue decomposition of $W = P\Lambda P^T,\ (P^TP = I,\ \Lambda = \operatorname{diag}(\lambda_i))$ by partitioning Λ and P according to nonnegative and negative eigenvalues: $\Lambda = (\Lambda_+, \Lambda_-),\ P = (P_+, P_-)$ with $\operatorname{diag}(\lambda_+) \geq 0,\ \operatorname{diag}(\Lambda_-) < 0$, see for instance Higham [5]. In this case

$$W = P_+\Lambda_+P_+^T + P_-\Lambda_-P_-^T = W_+ + W_-.$$

Thus we can reformulate the necessary and sufficient conditions for optimality as follows. The triple (y, Z, V) satisfies (4) and (5) if and only if

$$A(A^T(y)) = A(Z + C + \frac{1}{\sigma}X) - \frac{1}{\sigma}b, \quad Z = W(y)_+, \quad V = -\sigma W(y)_-.$$

Keeping Z constant, we get y from the linear system (4), while keeping y constant, we get Z from (6). This suggests to carry out the inner minimization by alternating between solving (4) to get y and computing Z by projection using (6). This will be further elaborated in the next section.

Finally, the update on X is given by

$$X \leftarrow X + \sigma(Z + C - A^T(y)) = -\sigma W(y)_- = V \succeq 0, \tag{7}$$

using (2) and (6).

These updates for X and Z therefore motivate us to call this a boundary point method, as both Z and X are on the boundary of the cone of semidefinite matrices. Moreover $ZX = 0$ holds throughout. Hence, once feasibility with respect to the primal and dual linear equations is reached, we have an optimal solution.

Select $\sigma > 0, \{\varepsilon_k\} \to 0,\ \varepsilon > 0.$

$k = 0;\ X^k = 0;\ Z^k = 0;$

repeat until $\delta_{outer} \leq \varepsilon$ (Outer iteration for $k = 0, 1, \dots$)

  repeat until $\delta_{inner} \leq \sigma\varepsilon_k$ (Inner iteration: $(X^k,\ \sigma)$ held constant)

    Solve for y^k: $AA^T(y^k) = A(Z^k + C + \frac{1}{\sigma}X^k) - \frac{1}{\sigma}b;$

    $W = A^T(y^k) - C - \frac{1}{\sigma}X^k;$

    $Z^k = W_+;\ V^k = -\sigma W_-;$

    $\delta_{inner} = \|A(V^k) - b)\|;$

  end (repeat)

  $X^{k+1} = V^k;$

  $k \leftarrow k + 1;$

  $\delta_{outer} := \|Z^k - A^T(y^k) + C\|;$

end (outer repeat)

Table 2: Boundary point method to solve (D)

4 The Boundary Point Method

We now summarize the Boundary Point method formally. In view of the previous observations, the algorithm described in Table 2 is in fact a version of the Augmented Lagrangian method, but the steps of the algorithm do not make this obvious at first sight.

Let us take a closer look at the stopping conditions of this method. Let (y^k, Z^k) be the solution at the end of inner iteration k, corresponding to σ and X^k. Thus we have

$$Z^k = W(y^k)_+,\ V^k = X^k + \sigma(Z^k - A^T y^k + C) = -\sigma W(y^k)_-. \tag{8}$$

Since $A(V^k) - b = A(\sigma(Z^k + C - A^T(y^k)) + X^k) - b$, the inner stopping condition $\|A(V^k) - b\| \leq \sigma\varepsilon_k$ is equivalent to

$$\|AA^T(y^k) - A(Z^k + C) - \frac{1}{\sigma}(A(X^k) - b)\| \leq \varepsilon_k. \tag{9}$$

We also have $V^k \succeq 0,\ Z^k \succeq 0,\ V^k Z^k = 0$. Therefore the error in the inner subproblem, given by (9), can be interpreted as primal infeasibility of V^k. The inner iterations are stopped, once V^k is 'nearly' primal feasible, yielding the new multiplier $X^{k+1} \leftarrow V^k$. The overall stopping condition is dual feasibility of y^k, Z^k.

It is well known that the augmented Lagrangian method converges under some rather general conditions, if the inner subproblem is solved with sufficient accuracy, see for instance [1]. For the sake of completeness, we include the following elementary convergence result, showing that our method also converges to an optimal solution, if the inner subproblem is solved exactly, i.e. $\varepsilon_k = 0\ \forall k$. We also note that Burer and Vandenbussche [3] prove a similar result for their approach.

Theorem 1 *Suppose that (P) and (D) have optimal solutions satisfying (1). If $\varepsilon_k = 0\ \forall k$, then the iterates y^k, Z^k, generated by the boundary point method, converge to an optimal solution of (D), i.e.*

$$Z^k + C - A^T(y^k) \to 0 \text{ as } k \to \infty.$$

Proof: Let (X^*, y^*, Z^*) be an optimal solution of (P) and (D) satisfying (1). Furthermore, let y^k, Z^k be the (optimal) solutions of the inner subproblem in iteration k corresponding to X^k and σ. We denote the dual residue after iteration k by

$$d^k := Z^k - A^T(y^k) + C = (Z^k - Z^*) - A^T(y^k - y^*).$$

For the quantity $r^k := \langle V^k - X^*, d^k \rangle$ it therefore holds that

$$r^k = \langle V^k - X^*, Z^k - Z^* \rangle - \langle A(V^k - X^*), y^k - y^* \rangle.$$

The second term in this sum is zero because $\varepsilon_k = 0$ implies $A(V^k) = b = A(X^*)$. Moreover, $\langle V^k, Z^k \rangle = \langle X^*, Z^* \rangle = 0$, so we conclude that

$$r^k = -\langle X^*, Z^k \rangle - \langle V^k, Z^* \rangle \le 0$$

because all matrices involved are positive semidefinite. We also recall from (8) that $V^k = X^k + \sigma d^k$. Therefore we also have

$$r^k = \langle X^k - X^*, d^k \rangle + \sigma \|d^k\|^2.$$

As a consequence $\langle X^k - X^*, d^k \rangle \le -\sigma \|d^k\|^2$.

Let us now consider the update on X^{k+1} in the algorithm. We have from (8) that $X^{k+1} = X^k + \sigma d^k$. Therefore

$$\|X^{k+1} - X^*\|^2 = \|X^k - X^*\|^2 + \sigma^2 \|d^k\|^2 + 2\sigma \langle X^k - X^*, d^k \rangle \le \|X^k - X^*\|^2 - \sigma^2 \|d^k\|^2.$$

So we have for all k

$$\sigma^2 \|d^k\|^2 \le \|X^k - X^*\|^2 - \|X^{k+1} - X^*\|^2.$$

Summing the first N of these inequalities, we get

$$\sigma^2 \sum_{k=1}^{N} \|d^k\|^2 \le \|X^1 - X^*\|^2.$$

The right-hand side of this inequality is a constant, hence $d^k \to 0$ as $N \to \infty$. $\square$

Remark 1

1. *We emphasize that the above proof technique follows [3], where convergence is shown for an augmented Lagrangian approach applied to the primal problem (P).*

2. *It is known, see [1], that the augmented Lagrangian method also converges if the inner tolerances ε_k are chosen carefully, for instance $\sum_k \varepsilon_k < \infty$. These results also apply to the present algorithm.*

3. *The practical performance is also influenced by the choice of σ, and possible update strategies for σ. If σ is chosen too large, then the inner subproblem gets increasingly difficult, while dual feasibility (outer stopping condition) is reached more easily. In this case we generate (nearly) feasible dual iterates y, Z. Too small a value of σ has the opposite effect. We reach primal feasibility quickly, but overall convergence may be slow.*

The computationally expensive steps clearly are solving the linear system with coefficient matrix $A(A^T(.))$ of order m. In contrast to interior point methods, this coefficient matrix is constant throughout the algorithm. This can be exploited to speed up solving the system, for instance by computing the Cholesky decomposition of AA^T once at the beginning, and then performing backsolves to get y^k. In addition, we compute a full eigenvalue decomposition of the matrix $W(y)$ of order n in each inner iteration. This limits the method to instances where n is not too large, say $n \le 1000$.

To give some impression of the practical performance of this method, we show how this method can be used to compute the theta number of a graph.

n	m	time (seconds)	factorizations
200	10000	24	266
300	22500	72	278
400	40000	125	217
500	62500	225	203
600	90100	380	193
700	122000	732	207
800	157000	1270	216
900	202000	1950	226
1000	250000	2880	238

Table 3: Computing the theta number on random graphs with relative accuracy 10^{-8}. Computation times in seconds, averaged over 5 instances for each value of n.

5 Application: Computing the theta number

The theta number $\vartheta(G)$, associated to a graph G is the solution to the following SDP, see for instance [7]:

$$\vartheta(G) = \max\langle J, X\rangle \text{ such that } x_{ij} = 0 \ \forall [ij] \in E(G),\ \operatorname{tr}(X) = 1,\ X \succeq 0.$$

We recall that J denotes the matrix of all ones.

There are several reasons, why it is important to be able to compute $\vartheta(G)$. Lovasz [7] showed that the (polynomially computable) number $\vartheta(G)$ separates the clique number $\omega(G)$ from the chromatic number $\chi(G)$, $\omega(G) \leq \vartheta(G) \leq \chi(G)$. Both these numbers lead to NP-hard problems, and also in practice, these numbers are extremely difficult to compute. In case of perfect graphs, where $\omega = \chi$, the only tractable way to compute either of these numbers in polynomial time is currently based on using ϑ. We refer to the recent monograph by Schrijver [12] for a comprehensive collection of results related to this topic.

From a computational point of view, the SDP underlying ϑ constitutes a challenging playground for algorithms to solve SDP, because it is rather easy to generate SDP with varying complexity (by increasing the number of nodes of the graph, and (independently) the number of edges). The most reliable way to solve these problems is based on primal-dual interior-point methods. It turns out that once the number of constraints goes beyond $m = 10000$, this approach fails due to time and memory limitations, see for instance the recent study [4].

In contrast, we will see now that the present approach can handle much larger instances, with m up to 250,000.

We first consider random graphs with $|V(G)| = n$ and edge density 0.5, which are the hardest for standard methods, because $m \approx \frac{1}{4}n^2$, see for instance [4]. If e_i denotes column i of the identity matrix I, we can express the $m+1$ linear equations as $\langle E_{ij}, X\rangle = 0$, $\forall [ij] \in E(G)$, $\langle I, X\rangle = 1$ with $E_{ij} = e_i e_j^T + e_j e_i^T$. We also note that $\operatorname{tr}(E_{ij}) = 0$, $\langle E_{ij}, E_{kl}\rangle \neq 0$ only if $[ij] = [kl]$. Therefore $A(A^T(.))$ is in fact a diagonal matrix, so solving (4) is trivial.

Following Mittelmann [10] we measure the accuracy of primal and dual infeasibility as follows:

$$r_P := \frac{\|A(X) - b\|}{1 + \|b\|} = \frac{1}{2}\|A(X) - b\|,$$

$$r_D := \frac{\|A^T(y) - J - Z\|_F}{1 + \|J\|_F} = \frac{1}{n+1}\|A^T(y) - J - Z\|_F.$$

We do not monitor infeasibility with respect to $X \succeq 0$, $Z \succeq 0, ZX = 0$, as these conditions are satisfied up to machine accuracy throughout the algorithm. We set the stopping condition of our algorithm to

problem	n	m	ϑ	[6]	[2]	[14]	our method
theta62	300	13389	29.6412515	96	344	200	50
theta82	400	23871	34.3668929	457	695	635	87
theta83	400	39861	20.3018905	1820	852	900	70
theta102	500	37466	38.3905463	1299	1231	1300	143
theta103	500	62515	22.5285698	2317	1960	2000	110
theta104	500	87244	13.3361402	11953	2105	1440	124
theta123	600	90019	24.6686519	10538	2819	3000	205
theta162	800	127599	37.0097358	13197	6004	12000	570

Table 4: Comparison with other methods [2, 14, 6] on some instances of the TOH collection; computation times in seconds and accuracy level $\varepsilon = 10^{-5}$.

$\varepsilon = 10^{-8}$ which means we ask that both r_P and r_D are smaller than 10^{-8}. This gives roughly 8 significant digits of accuracy in our case.

We use a laptop (Pentium 4 with 2.1 Ghz, 2 GB Ram) and have implemented the boundary point method for this particular problem in Matlab running under Linux. The Matlab source file to compute $\vartheta(G)$ is available on our web-site. The computation times are essentially determined by the number of eigenvalue decompositions. In Table 3 we consider random graphs with edge density 0.5 and provide computation times (in seconds) and the number of eigenvalue decompositions needed to reach the required accuracy. To give an idea on the speed of this machine, we run the MATLAB benchmark command `bench`. Compared to an AMD Athlon 1.66 GHz, our machine is about twice as slow on LU factorizations and is slightly faster on sparse matrix operations. Our machine is about one third faster than a HP UX 875 Mhz dual workstation.

Finally, we also compare our method to the currently strongest codes available for this type of problem. Specifically, we consider the low-rank factorization method from Burer and Monteiro [2], the iterative augmented system approach from Toh [14] and the modified barrier method from Kocvara and Stingl [6].

In Table 4 we report in column 4 the solution values given also in [14] (Table 8). Our values are correct up to at least 8 significant digits. We also observe, that a relative accuracy of $\varepsilon = 10^{-8}$ is virtually out of reach or extremely time consuming for these methods. We close with a comparison of computation times for an accuracy requirement of $\varepsilon = 10^{-5}$ which is reachable by the other methods.

To have an easier comparison, we repeat the timings from [6]. According to this paper, the timings for the columns labeled [6] and [2] refer to seconds on a AMD Opteron (2.4 GHz), a machine which is significantly faster than ours. The timings from the column labeled [14] were obtained on a 700MHz HP workstation. Based on the Matlab benchmark, we estimate our machine to be at most twice as fast as this machine.

We see that on these instances our method often is more than 10 times faster as any of the other methods. On the other hand, we noted that our method performs rather poorly for instances where either X or Z has small rank in an optimal solution. Finally, our method also seems less efficient on extremely sparse or extremely dense graphs (density less than 5 % or at least 95 %).

Further details regarding convergence analysis, some algorithmic speed-ups, connections to proximal-point methods and applications to other problems will be reported in a forthcoming paper [9] and in the dissertation [11].

Acknowledgement: We thank Christoph Helmberg for many useful discussions regarding some practical aspects of our method.

References

[1] D. BERTSEKAS. *Constrained optimization and Lagrange multiplier methods.* Academic Press, 1982.

[2] S. BURER and R.D.C. MONTEIRO. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming (B)*, 95:329–357, 2003.

[3] S. BURER and D. VANDENBUSSCHE. Solving lift-and-project relaxations of binary integer programs. *SIAM Journal on Optimization*, 16:726 – 750, 2006.

[4] I. DUKANOVIC and F. RENDL. Semidefinite programming relaxations for graph coloring and maximal clique problems. *Mathematical Programming (forthcoming).*

[5] N. J. HIGHAM. Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra and its Applications*, 103:103–118, 1988.

[6] M. KOCVARA and M. STINGL. On the solution of large-scale sdp problems by the modified barrier method using iterative solvers. Technical report, 2005.

[7] L. LOVÁSZ. On the shannon capacity of a graph. *IEEE Trans. Inform. Theory*, 25:1–7, 1979.

[8] J. MALICK. A dual approach to semidefinite least-squares problems. *SIAM Journal on Matrix Analysis and Applications*, 26:272.284, 2004.

[9] J. MALICK, J. POVH, F. RENDL, and A. WIEGELE. Proximal methods for semidefinite programming. Technical report, 2006.

[10] H.D. MITTELMANN. An independent benchmarking of SDP and SOCP solvers. *Mathematical Programming (B)*, 95:407–430, 2003.

[11] J. POVH. *Applications of semidefinite and copositive programming in combinatorial optimization.* PhD thesis, Universtity of Ljubljana, Slovenia, 2006.

[12] A. SCHRIJVER. *Combinatorial optimization. Polyhedra and efficiency. Vol. B*, volume 24 of *Algorithms and Combinatorics.* Springer-Verlag, Berlin, 2003.

[13] M.J. TODD. A study of search directions in primal-dual interior-point methods for semidefinite programming. *Optimization Methods and Software*, 11:1–46, 1999.

[14] K.-C. TOH. Solving large scale semidefinite programs via an iterative solver on the augmented systems. *SIAM Journal on Optimization*, 14:670 – 698, 2003.