A SPECTRAL BUNDLE METHOD FOR SEMIDEFINITE PROGRAMMING

C. HELMBERG* AND F. RENDL†

Abstract. A central drawback of primal-dual interior point methods for semidefinite programs is their lack of ability to exploit problem structure in cost and coefficient matrices. This restricts applicability to problems of small dimension. Typically semidefinite relaxations arising in combinatorial applications have sparse and well structured cost and coefficient matrices of huge order. We present a method that allows to compute acceptable approximations to the optimal solution of large problems within reasonable time.

Semidefinite programming problems with constant trace on the primal feasible set are equivalent to eigenvalue optimization problems. These are convex nonsmooth programming problems and can be solved by bundle methods. We propose replacing the traditional polyhedral cutting plane model constructed from subgradient information by a semidefinite model that is tailored for eigenvalue problems. Convergence follows from the traditional approach but a proof is included for completeness. We present numerical examples demonstrating the efficiency of the approach on combinatorial examples.

Key words. eigenvalue optimization, convex optimization, semidefinite programming, proximal bundle method, large-scale problems.

AMS subject classifications. 65F15, 90C25; Secondary 52A41, 90C06.

1. Introduction. The development of interior point methods for semidefinite programming [19, 31, 1, 46] has increased interest in semidefinite modeling techniques in several fields such as control theory, eigenvalue optimization, and combinatorial optimization. In fact, interior point methods proved to be very useful and reliable solution methods for semidefinite programs of moderate size. However, if the problem is defined over large matrix variables or a huge number of constraints interior point methods grow terribly slow and consume huge amounts of memory. The most efficient methods of today [15, 23, 2, 32, 45, 29] are primal-dual methods that require, in each iteration of the interior point method, the factorization of a dense matrix of order equal to the number of constraints and one to three factorizations of the positive semidefinite matrix variables within the line search. For a typical workstation this restricts the number of constraints to 2000 and the size of the matrix variables to 500 if reasonable performance is required. For larger problems time and memory requirements are prohibitive. It is important to realize that either the primal or the dual matrix is generically dense even if cost and coefficient matrices are very sparse. Very recently, a pure dual approach was proposed in [4] which offers some possibilities to exploit sparsity. It is too early to judge the potential of this method.

In combinatorial optimization semidefinite relaxations where introduced in [27]. At that time they were mainly considered a theoretical tool for obtaining strong bounds [11, 28, 40]. With the development of interior point methods hopes soared high that these relaxations could be of practical value. Within short time several approximation algorithms relying on semidefinite programming were published, most of them based on the approach by Goemans and Williamson [8]. On the implementational side [14, 16, 20] cutting plane approaches for semidefinite relaxations of

*Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustraße 7, 14195 Berlin, Germany. helmberg@zib.de, http://www.zib.de/helmberg

† Universität Klagenfurt, Institut f. Mathematik, Universitätsstr. 65-67, A 9020 Klagenfurt, Austria. franz.rendl@uni-klu.ac.at. Financial support through the Austrian FWF Project P12660-MAT is greatfully acknowledged.

constrained quadratic 0-1 programming problems proved to yield solutions of high quality. However, as mentioned above, they were very expensive to compute even for problems of small size (a few hundred 0-1 variables). Problems arising in practical applications (starting with a few thousand 0-1 variables) were out of reach. We believe that the method proposed in this paper will open the door to problems of this size.

Although combinatorial applications are our primary concern we stress that the method is not restricted to this kind of problems. In fact it will be a useful alternative to interior point methods whenever the number of constraints or the order of the matrices is quite large.

We transform a standard dual semidefinite program into an eigenvalue optimization problem by reformulating the semidefinite constraint as a non-negativity constraint on the minimal eigenvalue of the slack matrix variable and lifting this constraint into the cost function by means of a Lagrange multiplier. The correct value of the Lagrange multiplier is known in advance if the primal feasible matrices have constant trace. (This is the case for the combinatorial applications we have in mind.)

In this paper we develop a bundle method for solving the problem of minimizing the maximal eigenvalue of an affine matrix function with an additional linear objective term. These functions are well known to be convex and non-smooth. A very general method for optimizing non-smooth convex functions is the bundle method, see e.g. [21, 42, 17, 18]. In each step the function value and a subgradient of the function is computed for some specific point. By means of the collected subgradients a cutting plane model of the function is formed. The minimizer of the cutting plane model augmented by a regularization term yields the new point. In the case of eigenvalue optimization the subgradient is formed by means of an eigenvector to the maximal eigenvalue. Extremal eigenvalues and associated eigenvectors of large symmetric matrices can be computed efficiently by Lanczos methods (see e.g. [9]). Lanczos methods need a subroutine that computes the product of the matrix with a vector. This allows to exploit any kind of structure present in the matrix.

The polyhedral cutting plane model used in traditional bundle algorithms is updated by new subgradient information such as to approximate well the subdifferential, and thus the function itself, in the vicinity of the current point. For eigenvalue optimization problems the subdifferential is generated by a semidefinite set, in particular by the intersection of a simple affine constraint and a face of the semidefinite cone. This suggests to use, instead of the traditional polyhedral cutting plane model, a semidefinite cutting plane model that works with an approximation of this face of the semidefinite cone. This specialization of the cutting plane model is the main contribution of the paper.

The semidefinite bundle approach allows for an intriguing interpretation in terms of the original semidefinite program. The cutting plane model requires that the dual slack matrix of the semidefinite program is positive semidefinite only with respect to a subspace of vectors, thus it may be interpreted as a relaxation of the dual semidefinite program. In general the optimal solution of this relaxed semidefinite problem will produce an indefinite dual slack matrix. One or more of the negative eigenvalues and corresponding eigenvectors of the slack matrix are used to update the subspace in order to improve the relaxation and the process is iterated.

This process trivially provides the optimal solution if the subspace grows to the full space. However, we show that during the algorithm generically the dimension of the subspace is bounded by (roughly) the square root of the number of constraints. If this is still considered too large the introduction of an aggregate subgradient guarantees convergence for restricted bundle sizes. In the extreme the bundle may consist of one new eigenvector to the maximal eigenvalue only.

In contrast, the 'classical' algorithms of Cullum, Donath, and Wolfe [6] and Polak and Wardi [38] require in each iteration the computation of all eigenvectors to eigenvalues within an ε-distance of the maximal eigenvalue, thus close to the optimal solution this number is at least as large as the multiplicity of the maximal eigenvalue in the optimal solution. In the quadratically convergent algorithm of Overton [35] each step is computed from a complete spectral decomposition of the matrix and a guess of the exact multiplicity of the maximal eigenvalue in the optimal solution. In recent work [33, 34] Oustry reinterprets the algorithm of Overton within the framework of the $\mathcal{U}$-Lagrangian introduced in [26] and embeds it in a first order method to ensure global convergence. Again, for global convergence the approach relies on the spectrum of all eigenvalues within ε-distance of the maximal eigenvalue and makes use of the entire spectral information to obtain local quadratic convergence.

Because of the restricted bundle size quadratic convergence is out of reach for our algorithm, it is a first order method only. In principle convergence follows from the traditional approach (see e.g. [21]) but we include a proof for completeness. We also present a primal-dual interior point code for solving the quadratic semidefinite programming problems associated with the semidefinite cutting plane models and discuss efficiency aspects. The properties of the algorithm are illustrated on several combinatorial examples.

In §2 some basic properties of semidefinite programs are stated. Then we transform semidefinite programs into eigenvalue optimization problems. Section 3 introduces the bundle method. The algorithm and the proof of convergence is given in §4. The quadratic semidefinite subproblems arising in the bundle method can be solved by interior point methods as explained in §5. Section 6 gives an outline of the implementation and briefly discusses the computation of the maximal eigenvalue and an associated eigenvector. Numerical examples on combinatorial problems are presented in §7. We conclude the paper with a summary and possible extensions and improvements in §8. For the convenience of the reader an appendix explaining the notation and the symmetric Kronecker product is included at the end of the paper.

2. Semidefinite programs and eigenvalue optimization. We denote the set of symmetric matrices of order n by S_n which we regard as a space isomorphic to $\mathbb{R}^{\binom{n+1}{2}}$. As scalar product of $A, B \in S_n$ (or more general, $A, B \in \mathbb{R}^{m\times n}$) we use $\langle A, B\rangle = \operatorname{tr}(B^T A)$ where the trace is the sum of the diagonal elements of a square matrix. We will often use the same symbol for the canonical scalar product of vectors $a, b \in \mathbb{R}^m$, $\langle a, b\rangle = b^T a$, the appropriate space will be clear from the context. The subset of positive semidefinite matrices S_n^+ is a full-dimensional, non-polyhedral convex cone in S_n and defines a partial order on the symmetric matrices by $A \succeq B \iff (A - B) \in S_n^+$. Positive definite matrices are denoted by S_n^{++} or $A \succ 0$.

Consider the standard primal-dual pair of semidefinite programs,

$$
\text{(P)}\quad \begin{array}{ll} \max & \langle C, X\rangle \\ \text{s.t.} & \mathcal{A}X = b \\ & X \succeq 0. \end{array}
\qquad\qquad
\text{(D)}\quad \begin{array}{ll} \min & b^T y \\ \text{s.t.} & Z = \mathcal{A}^T y - C \\ & Z \succeq 0. \end{array}
$$

$\mathcal{A} : S_n \to \mathbb{R}^m$ is a linear operator and $\mathcal{A}^T : \mathbb{R}^m \to S_n$ is its adjoint operator, defined by $\langle \mathcal{A}X, y\rangle = \langle X, \mathcal{A}^T y\rangle$ for all $X \in S_n$ and $y \in \mathbb{R}^m$. They are of the form

$$\mathcal{A}X = \begin{bmatrix} \langle A_1, X\rangle \\ \vdots \\ \langle A_m, X\rangle \end{bmatrix} \qquad \text{and} \qquad \mathcal{A}^T y = \sum_{i=1}^m y_i A_i$$

with $A_i \in S_n$, $i = 1, \ldots, m$. $C \in S_n$ is the cost matrix, $b \in \mathbb{R}^m$ the right-hand-side vector.

We assume some constraint qualification to hold, so that these problems satisfy strong duality in the sense that for any optimal solution X^* of (P) and any optimal solution (y^*, Z^*) of (D) we have

$$X^* Z^* = 0. \tag{2.1}$$

The following assumption allows a simple reformulation of the dual (D) as an eigenvalue optimization problem. We assume that

$$\mathcal{A}X = b \ \text{ implies } \ \operatorname{tr} X = a, \tag{2.2}$$

for some constant $a > 0$. In this case we can add $\operatorname{tr} X = a$ as a redundant constraint to the primal problem and obtain the following dual equivalent to (D)

$$\min\ a\lambda + b^T y \ \text{ s.t. } \ Z = \mathcal{A}^T y + \lambda I - C \succeq 0.$$

Now $a > 0$ implies $X \neq 0$ at the optimum, hence any optimal Z of this dual is singular. Therefore all dual optimal solutions Z satisfy $0 = \lambda_{\max}(-Z)$, leading to

$$\lambda = \lambda_{\max}(C - \mathcal{A}^T y).$$

Thus we have shown that (D) is equivalent to $\min_y a\lambda_{\max}(C - \mathcal{A}^T y) + b^T y$. For notational convenience we assume $a = 1$ and deal with the following problem.

$$\text{(E)} \qquad \min_{y \in R^m} \lambda_{\max}(C - \mathcal{A}^T y) + b^T y.$$

The eigenvalue problem (E) is a convex, non-smooth optimization problem. It is well studied in the literature. Here we only recall some basic facts. The function

$$\lambda_{\max}(X) = \max\{\langle W, X\rangle : \operatorname{tr} W = 1, W \succeq 0\}$$

is differentiable if and only if the maximal eigenvalue has multiplicity one. When optimizing eigenvalue functions, the optimum is generically attained at matrices whose maximal eigenvalue has multiplicity larger than one. In this case one has to consider the subdifferential of $\lambda_{\max}$ at X,

$$\partial\lambda_{\max}(X) = \{W \succeq 0 : \langle W, X\rangle = \lambda_{\max}(X), \operatorname{tr} W = 1\}$$

(see e.g. [35]). In particular, for any $v \in \mathbb{R}^n$ belonging to the eigenspace of the maximal eigenvalue of X, $W = vv^T$ is contained in the subdifferential of $\lambda_{\max}$ at X. For the function of interest,

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

the subdifferential of f at y can be derived by standard rules (see [17]),

$$\partial f(y) = \left\{ b - \mathcal{A}W : \left\langle W, C - \mathcal{A}^T y \right\rangle = \lambda_{\max}(C - \mathcal{A}^T y), \operatorname{tr} W = 1, W \succeq 0 \right\}.$$

Observe that the set of all subgradients is bounded.

REMARK 2.1. *Even though our assumption (2.2) might look artificial, it does hold for SDP arising from quadratic 0-1 optimization. It also holds for many other SDP derived as relaxations of combinatorial optimization problems, see for instance [1, 12, 24].*

3. The bundle method. In this section we develop a new method for minimizing f. We use two classical ingredients, the *proximal point idea*, and the *bundle concept*. The new contribution lies in the way that we derive the new iterate from the 'bundle' of subgradient information collected from previous iterates. Since our approach builds on several subtle ideas, we proceed in small steps and explain first, how we derive a minorant of f from local information.

3.1. Minorizing f by $\hat{f}$. Our first goal is to obtain a minorant $\hat{f}$ of f which approximates f in the neighborhood of the current iterates reasonably well, and which is easier to handle than f. Introducing the function

$$L(W, y) := \left\langle C - \mathcal{A}^T y, W \right\rangle + b^T y$$

we can express $f(y)$ as

$$f(y) = \max\{L(W, y) : W \succeq 0, \operatorname{tr} W = 1\}.$$

This formulation shows that lower approximations of f can be obtained by constraining W to a subset of all semidefinite matrices with $\operatorname{tr} W = 1$.

We propose the following choice for this subset. Let P be $n \times r$ with $P^T P = I_r$, and $\overline{W} \in S_n^+$ with $\operatorname{tr} \overline{W} = 1$ be two matrices. We restrict W to be contained in the set

$$\widehat{\mathcal{W}} = \left\{ \alpha \overline{W} + PVP^T : \alpha + \operatorname{tr} V = 1, \alpha \geq 0, V \succeq 0 \right\}. \tag{3.1}$$

The (convex) minorant $\hat{f}$, defined through P and $\overline{W}$, now reads

$$\hat{f}(y) := \max\{L(W, y) : W \in \widehat{\mathcal{W}}\}.$$

By definition, we have $\hat{f}(y) \leq f(y)\ \ \forall y$. If, for some $\hat{y} \in \mathbb{R}^n$, $vv^T \in \widehat{\mathcal{W}}$ for some eigenvector v to $\lambda_{\max}(C - \mathcal{A}^T \hat{y})$ then $\hat{f}(\hat{y}) = f(\hat{y})$. This is e.g. the case if v is a column of P or v is contained in the range space of P.

The intuitive idea behind our specific choice of $\widehat{\mathcal{W}}$ is as follows: the matrix P contains subgradient information from the current point $\hat{y}$, and perhaps from previous iterates. We explain below in detail, how we propose to select and update the matrix P. For computational efficiency, we would like to keep the number r of columns of P small, independent of the multiplicity of the largest eigenvalue. Therefore we collect indispensable subgradient information, that has to be removed from P, in an aggregate subgradient. This *aggregation* is the final ingredient of our local model of f. The matrix $\overline{W}$ plays the role of an aggregate subgradient. Again, we will discuss below, how $\overline{W}$ is updated during the algorithm. The main point here is that instead of optimizing over all semidefinite matrices W, we constrain ourselves to a small subset.

REMARK 3.1. *If we set $\overline{W} = 0$ and use for the matrix P a set of eigenvectors to the r largest eigenvalues at $\hat{y}$, we would end up with a model closely related to the approach from [6]. In this case it would be important to select r at least as large as the multiplicity of the largest eigenvalue. In our present approach this is not necessary.*

3.2. Proximal point idea. The next goal is to minimize $\hat{f}$ instead of f. Since $\hat{f}$ is built from local information from a few previous iterates, this model function is unlikely to be reliable for points far from the current iterate. Therefore we use the *proximal point idea* and add a penalty term for the displacement from the current point. Thus we determine a new candidate y from the current iterate $\hat{y}$ by solving the following convex problem, referred to as the augmented model. (Here $u > 0$ is some fixed real weight.)

$$\min_y \hat{f}(y) + \frac{u}{2} \|y - \hat{y}\|^2 .$$

We note that this minimization problem corresponds to the Lagrangian relaxation of $\min\{\hat{f}(y) : \|y - \hat{y}\|^2 \leq s^2\}$. Thus we replace the original function f by its minorant $\hat{f}$ and minimize locally around $\hat{y}$. The weight u controls (indirectly) the radius s of the sphere around $\hat{y}$, over which we minimize. Substituting the definition of $\hat{f}$, this problem is the same as

$$\min_y \max_{W \in \widehat{\mathcal{W}}} L(W, y) + \frac{u}{2} \|y - \hat{y}\|^2 . \tag{3.2}$$

This problem can be simplified, because y is unconstrained. Note that

$$L(W, y) = \langle C - \mathcal{A}^T \hat{y}, W \rangle + b^T \hat{y} + \langle b - \mathcal{A}W, y - \hat{y} \rangle .$$

Therefore we obtain

$$\begin{aligned} &\min_y \max_{W \in \widehat{\mathcal{W}}} L(W, y) + \frac{u}{2} \|y - \hat{y}\|^2 \\ &= \max_{W \in \widehat{\mathcal{W}},\ b - \mathcal{A}W + u(y - \hat{y}) = 0} L(W, y) + \frac{u}{2} \|y - \hat{y}\|^2 \\ &= \max_{W \in \widehat{\mathcal{W}}} \langle C - \mathcal{A}^T \hat{y}, W \rangle + b^T \hat{y} - \frac{1}{2u} \langle \mathcal{A}W - b, \mathcal{A}W - b \rangle . \end{aligned}$$

The first equality follows from interchanging min and max (see Corollary 37.3.2 of [41]) and using first order optimality for the inner minimization with respect to y,

$$y = \hat{y} + \frac{1}{u}(\mathcal{A}W - b). \tag{3.3}$$

The final problem is a semidefinite program with (concave) quadratic cost function. We will discuss in §5 how problems of this kind can be solved efficiently. Its optimal solution W^{k+1} gives the new trial point y by (3.3).

REMARK 3.2. *The choice of the weight u is somewhat of an art. There are several clever update strategies published in the literature, see for instance [21, 42].*

3.3. One iteration of the algorithm. The main ingredients of our approach have now been explained, so we can give a formal description of a general iteration k of the algorithm. To be consistent with the notation of the algorithm given in §4, let us denote by x^k what was called $\hat{y}$ in §3.2. The algorithm may have to compute several trial points $y^{k+1}, y^{k+2}, \ldots$ while keeping the same $x^k = x^{k+1} = \ldots$ if progress is not considered satisfactory (*null step*). For each y^{k+1} the function is evaluated and a subgradient (eigenvector) is computed. This information is added to $\widehat{\mathcal{W}}^k$ to form an improved model $\widehat{\mathcal{W}}^{k+1}$. Therefore, we assume that the current 'bundle' $P^k = P$

contains an eigenvector of $\lambda_{\max}(C - \mathcal{A}^T y^k)$ in its span (y^k may or may not be equal to x^k). Other than that, P need only satisfy $P^T P = I_r$. The minorant of f in iteration k is denoted by $\hat{f}^k$:

$$\hat{f}^k(y) := \max_{W \in \widehat{\mathcal{W}}^k} L(W, y).$$

Here $\widehat{\mathcal{W}}^k$ represents the current approximation to the set of all semidefinite matrices of trace one, see (3.1). It will be convenient to introduce also the regularized version of $\hat{f}^k$. We define

$$f^k(y) := \hat{f}^k(y) + \tfrac{u}{2} \left\| y - x^k \right\|^2.$$

The new trial point y^{k+1} is obtained by minimizing $f^k(y)$ with respect to y. As described above, this can be done as follows. First, solve by interior point methods (see §5)

$$\max_{W \in \widehat{\mathcal{W}}^k} \left\langle C - \mathcal{A}^T x^k, W \right\rangle + b^T x^k - \tfrac{1}{2u} \left\langle \mathcal{A}W - b, \mathcal{A}W - b \right\rangle, \tag{3.4}$$

yielding a (not necessarily unique) maximizer $W^{k+1} = \alpha^* \overline{W}^k + P^k V^* (P^k)^T$. Next, use (3.3) to compute

$$y^{k+1} = x^k + \tfrac{1}{u}(\mathcal{A}W^{k+1} - b). \tag{3.5}$$

To finish an iteration, we have to decide whether enough progress is made to perform a *serious step* or not, i.e. whether we are going to set $x^{k+1} = y^{k+1}$ or $x^{k+1} = x^k$, and how to update P^k and $\overline{W}^k$.

If P^k does not yet use the maximum number of columns allowed then the update process is simple: orthogonalize the new eigenvector with respect to P^k, add it as a new column to form P^{k+1} and continue. In general, however, P^k will already use the maximum number of columns and so we have to make room for the new subgradient information. Instead of simply eliminating some columns of P^k we can do better by exploiting the information available in α^* and V^*.

Let $Q \Lambda Q^T$ be an eigenvalue decomposition of V^*. Then the 'important' part of the spectrum of W^{k+1} (the important subspace within the space spanned by P^k) is spanned by the eigenvectors associated with the 'large' eigenvalues of V^*. Thus we split the eigenvectors of Q into two parts $Q = [Q_1 Q_2]$ (with corresponding spectra Λ_1 and Λ_2), Q_1 containing as columns the eigenvectors associated to 'large' eigenvalues of V^* and Q_2 containing the remaining columns,

$$W^{k+1} = P^k Q_1 \Lambda_1 (P^k Q_1)^T + \alpha^* \overline{W}^k + P^k Q_2 \Lambda_2 (P^k Q_2)^T. \tag{3.6}$$

Now the next P^{k+1} is computed to contain $P^k Q_1$ and at least one eigenvector v^{k+1} to the maximal eigenvalue of $C - \mathcal{A}^T y^{k+1}$,

$$P^{k+1} = \operatorname{orth}(\left[P^k Q_1 \ v^{k+1} \right]). \tag{3.7}$$

(The operator orth(.) indicates that we take an orthonormal basis of $[P^k Q_1 \ v^{k+1}]$.)

The next aggregate matrix is built in such a way that $W^{k+1} \in \widehat{\mathcal{W}}^{k+1}$. Since P^{k+1} contains only the important part of P^k, given by $P^k Q_1$, we include the remaining part of P^k, given by $P^k Q_2$ in $\overline{W}^{k+1}$:

$$\overline{W}^{k+1} = \frac{1}{\alpha^* + \operatorname{tr} \Lambda_2} (\alpha^* \overline{W}^k + P^k Q_2 \Lambda_2 (P^k Q_2)^T). \tag{3.8}$$

Note that $\overline{W}^{k+1}$ is scaled to have trace equal one.

PROPOSITION 3.3. *Update rules (3.7) and (3.8) ensure that* $W^{k+1} \in \widehat{\mathcal{W}}^{k+1}$.

Proof. Let W^{k+1} be of the form (3.6). By (3.7) there is an orthonormal matrix $\bar{Q}$ such that $P^{k+1}\bar{Q} = P^k Q_1$. Let $V = \bar{Q}\Lambda_1\bar{Q}^T$ and $\alpha = \alpha^* + \operatorname{tr}\Lambda_2$, then $V \succeq 0$, $\alpha \geq 0$, $\alpha + \operatorname{tr} V = 1$ and $W^{k+1} = P^{k+1}V(P^{k+1})^T + \alpha\overline{W}^{k+1} \in \widehat{\mathcal{W}}^{k+1}$. $\square$

We summarize some easy facts, which will be used in the convergence analysis of the algorithm.

$$f^k(y^{k+1}) \leq f^k(y) \quad \forall y,$$

since y^{k+1} is minimizer of f^k. Because $f^k(x^k) = \hat{f}^k(x^k)$ and $\hat{f}^k$ minorizes f we obtain

$$f^k(y^{k+1}) \leq f^k(x^k) \leq f(x^k). \tag{3.9}$$

Next let

$$f_*^k(y) := L(W^{k+1}, y) + \tfrac{u}{2}\left\|y - x^k\right\|^2.$$

Using the definition of y^{k+1} from (3.5) it follows easily that

$$f_*^k(y) = f_*^k(y^{k+1}) + \tfrac{u}{2}\left\|y - y^{k+1}\right\|^2. \tag{3.10}$$

Since $W^{k+1} \in \widehat{\mathcal{W}}^{k+1}$, the augmented model of the next iteration will satisfy

$$f^{k+1}(y) \geq f_*^k(y) \quad \forall y. \tag{3.11}$$

REMARK 3.4. *While the choice for the update of P^k is fairly natural, we could use other update formulas, such as $\overline{W}^{k+1} = W^{k+1}$ and $P^{k+1} = v^{k+1}$. The main properties guiding the update are that $W^{k+1} \in \widehat{\mathcal{W}}^{k+1}$ ensuring (3.11) and that in y^{k+1} the model is now supported by a subgradient of f pushing the model towards f in the vicinity of the last minimizer.*

4. Algorithm and convergence analysis. In the previous section we focused on the question of doing one iteration of the bundle method. Now we provide a formal description of the method and point out that except for the choice of the bundle, the nature of the subproblem, and some minor changes in parameters the algorithm and its proof are identical to the algorithm of Kiwiel as presented in [21]. To keep the paper self-contained we present and analyze a simplified variant for fixed u. We refer the reader to [21] for an algorithm with variable choice of u.

ALGORITHM 4.1.

Input: *An initial point $y^0 \in \mathbb{R}^m$, a (normalized) eigenvector v^0 to the maximal eigenvalue of $C - \mathcal{A}^T y^0$, an $\varepsilon > 0$ for termination, an improvement parameter $m_L \in (0, \frac{1}{2})$, a weight $u > 0$, an upper bound $R \geq 1$ on the number of columns of P.*

1. *(Initialization) $k = 0$, $x^0 = y^0$, $P^0 = v^0$, $\overline{W}^0 = v^0(v^0)^T$.*
2. *(Direction finding) Solve (3.4) to get y^{k+1} from (3.5). Decompose V^* into $V^* = Q_1\Lambda_1 Q_1^T + Q_2\Lambda_2 Q_2^T$ with $\operatorname{rank}(Q_1) \leq R - 1$. Compute $\overline{W}^{k+1}$ using (3.8).*
3. *(Evaluation) Compute $\lambda_{\max}(C - \mathcal{A}^T y^{k+1})$ and an eigenvector v^{k+1}. Compute P^{k+1} by (3.7).*
4. *(Termination) If $f(x^k) - \hat{f}^k(y^{k+1}) \leq \varepsilon$ then* **stop**.

5. *(Serious step) If*

$$f(y^{k+1}) \le f(x^k) - m_L(f(x^k) - \hat{f}^k(y^{k+1})) \tag{4.1}$$

then set $x^{k+1} = y^{k+1}$, continue with Step 7. Otherwise continue with Step 6.

6. *(Null step) Set $x^{k+1} = x^k$.*
7. *Increase k by 1 and go to Step 2.*

We prove convergence of the algorithm for $\varepsilon = 0$. If the algorithm stops after a finite number of iterations then by (3.9) $f^k(y^{k+1}) = f(x^k)$ which implies $y^{k+1} = x^k$ and thus by (3.5) $0 \in \partial f(x^k)$, so x^k is optimal. Assume in the following that the algorithm does not stop. First consider the case that only null steps occur after some iteration K.

LEMMA 4.2. *If there is a $K \ge 0$ such that (4.1) is violated for all $k \ge K$, then $\lim_{k\to\infty} \hat{f}^k(y^{k+1}) = f(x^K)$ and $0 \in \partial f(x^K)$.*

Proof. For convenience we set $x = x^K = x^{K+1} = \ldots$. Using the relations (3.10), (3.11) and (3.9), we obtain for all $k \ge K$

$$f_*^k(y^{k+1}) + \frac{u}{2}\left\|y^{k+2} - y^{k+1}\right\|^2 = f_*^k(y^{k+2}) \le f^{k+1}(y^{k+2}) \le f^{k+1}(x^{k+1}) \le f(x).$$

Therefore the $f^k(y^{k+1})$ converge to some $f^* \le f(x)$ and $\left\|y^{k+2} - y^{k+1}\right\| \to 0$. Let $g^{k+1} = b - \mathcal{A}(v^{k+1}(v^{k+1})^T)$ denote the computed gradient of f in y^{k+1} and observe that the linearization $\bar{f}$ of f in y^{k+1}

$$\bar{f}(y; y_{k+1}) = f(y^{k+1}) + \left\langle g^{k+1}, y - y^{k+1}\right\rangle$$

minorizes $\hat{f}^{k+1}$, because $(v^{k+1}(v^{k+1})^T) \in \widehat{\mathcal{W}}^{k+1}$. Thus

$$\begin{aligned}
0 &\le f(y^{k+1}) - \hat{f}^k(y^{k+1}) = \\
&= \bar{f}(y^{k+1}; y^{k+1}) - \hat{f}^k(y^{k+1}) \\
&= \bar{f}(y^{k+2}; y^{k+1}) - \hat{f}^k(y^{k+1}) - \left\langle g^{k+1}, y^{k+2} - y^{k+1}\right\rangle \\
&\le \hat{f}^{k+1}(y^{k+2}) - \hat{f}^k(y^{k+1}) + \left\|g^{k+1}\right\| \cdot \left\|y^{k+2} - y^{k+1}\right\| \\
&= f^{k+1}(y^{k+2}) - f^k(y^{k+1}) - u\left\|y^{k+2} - x\right\|^2 + u\left\|y^{k+1} - x\right\|^2 + \\
&\quad \left\|g^{k+1}\right\| \cdot \left\|y^{k+2} - y^{k+1}\right\|.
\end{aligned}$$

The convergence of the $f^k(y^{k+1})$, the boundedness of the gradients and the fact that $\left\|y^{k+2} - y^{k+1}\right\| \to 0$ imply that the last term goes to zero for $k \to \infty$. So for all $\delta > 0$ there is an $M \in \mathbb{N}$ such that for all $k > M$

$$f(y^{k+1}) - \delta \le \hat{f}^k(y^{k+1}) \le f(x) < \frac{f(y^{k+1}) - m_L \hat{f}^k(y^{k+1})}{1 - m_L} \le f(y^{k+1}) + \frac{m_L}{1 - m_L}\delta,$$

where '$<$' follows from (4.1) being violated for all $k > K$. Thus the sequences $f(y^{k+1})$ and $\hat{f}^k(y^{k+1})$ both converge to $f(x)$. y^{k+1} is the minimizer of the regularized function f^k. On the one hand this implies that $y^{k+1} \to x$. On the other hand 0 must be contained in the subgradient $\partial f^k(y^{k+1}) = \partial \hat{f}^k(y^{k+1}) + u(y^{k+1} - x)$, see (3.3). Therefore there is a sequence $h^k \in \partial \hat{f}^k(y^{k+1})$ of subgradients converging to zero. The $\hat{f}^k$ minorize f, the $\hat{f}^k(y^{k+1})$ converge to $f(x)$ and the y^{k+1} converge to x, hence zero must be contained in $\partial f(x)$. $\square$

We may concentrate on serious steps in the following. In order to simplify notation we will speak of x^k as the sequence generated by serious steps with all duplicates eliminated. By f^k (and the corresponding $\hat{f}^k$) we will refer to the function whose minimization gives rise to x^{k+1}.

The next lemma investigates the case that the $f(x^k)$ remain above some value $f(\tilde{x})$ for some fixed $\tilde{x}$.

LEMMA 4.3. *If*

$$f(x^k) > f(\tilde{x}) \quad \textit{for some fixed } \tilde{x} \in \mathbb{R}^m \textit{ and all } k \tag{4.2}$$

then the x^k converge to a minimizer of f.

Proof. First we prove the boundedness of the x^k. To this end denote by $g^{k+1} \in \partial \hat{f}^k(x^{k+1})$ the subgradient arising from the optimal solution of the minimization problem for f^k, $g^{k+1} = b - \mathcal{A}W^{k+1}$, and observe that by (3.3)

$$x^{k+1} - x^k = -\frac{g^{k+1}}{u}. \tag{4.3}$$

Since $\hat{f}^k$ minorizes f we obtain

$$f(x^k) \geq f(\tilde{x}) \geq \hat{f}^k(x^{k+1}) + \left\langle g^{k+1}, \tilde{x} - x^{k+1} \right\rangle.$$

Therefore the distance of x^{k+1} to $\tilde{x}$ can be bounded by

$$\begin{aligned}
&\left\| \tilde{x} - x^{k+1} \right\|^2 = \left\| \tilde{x} - x^k + x^k - x^{k+1} \right\|^2 = \\
&\leq \left\| \tilde{x} - x^k \right\|^2 + 2\left\langle \tilde{x} - x^k, x^k - x^{k+1} \right\rangle + 2\left\langle x^k - x^{k+1}, x^k - x^{k+1} \right\rangle \\
&= \left\| \tilde{x} - x^k \right\|^2 + 2\left\langle \tilde{x} - x^{k+1}, g^{k+1}/u \right\rangle \\
&\leq \left\| \tilde{x} - x^k \right\|^2 + \tfrac{2}{u}(f(x^k) - \hat{f}^k(x^{k+1})).
\end{aligned}$$

For any $k > K$, a recursive application of the bound above yields

$$\left\| \tilde{x} - x^k \right\|^2 \leq \left\| \tilde{x} - x^K \right\|^2 + \tfrac{2}{u} \sum_{i=K}^{\infty} (f(x^k) - \hat{f}^k(x^{k+1})). \tag{4.4}$$

By (4.1) the progress of the algorithm in each serious step is at least $m_L(f(x^k) - \hat{f}^k(x^{k+1}))$ and together with (4.2) we obtain

$$\sum_{i=0}^{\infty} (f(x^i) - \hat{f}^i(x^{i+1})) \leq \tfrac{1}{m_L}(f(x^0) - f(\tilde{x})).$$

Therefore the sequence of the x^k remains bounded and has an accumulation point $\bar{x}$. By replacing $\tilde{x}$ by $\bar{x}$ in (4.4) and choosing K sufficiently large, the remaining sum can be made smaller than an arbitrary small $\delta > 0$, thus proving the convergence of the x^k to $\bar{x}$. As the x^{k+1} converge to $\bar{x}$ the g^{k+1} converge to zero by (4.3), and since the sequence $(f(x^k) - \hat{f}^k(x^{k+1}))$ has to converge to zero as well, we conclude that $0 \in \partial f(\bar{x})$, i.e., $\bar{x}$ is a minimizer of f. $\square$

The lemma also implies that $f(x^k) \to \inf f$ if there are no minimizers. We summarize the discussion in the following theorem.

THEOREM 4.4. *[21] If the set of minimizers of f is not empty then the x^k converge to a minimizer of f. In any case $f(x^k) \to \inf f$.*

REMARK 4.5. *We have just seen that the bundle algorithm works correctly even if P contains only one column. In this case the use of the aggregate subgradient is crucial.*

To achieve correctness of the bundle algorithm without aggregate subgradients, it suffices to store in P only the subspace spanning the eigenvectors corresponding to non-zero eigenvalues of an optimal solution W^{k+1} of (3.2). Using the bound of [36] it is not too difficult to show that in this case the maximal number of columns one has to provide is the largest $\bar{r} \in \mathbb{N}$ satisfying $\binom{\bar{r}+1}{2} \leq m+1$ plus the number of eigenvectors to be added in each iteration (this is at least one). In our computational experiments we found that this upper bound is hardly ever reached. In fact, typical values for the maximal rank are around half this upper bound.

5. Solving the subproblem. In this section we concentrate on how the minimizer of f^k can be computed efficiently. We have already seen in §3 that this task is equivalent to solving the quadratic semidefinite program (3.4). Problems of this kind can be solved by interior point methods, see e.g. [7, 23]. Dropping the iteration index k and the constants in (3.4) we obtain for $y = x^k$

$$\begin{array}{ll} \min & \frac{1}{2u}\langle \mathcal{A}W, \mathcal{A}W\rangle - \frac{1}{u}\langle b, \mathcal{A}W\rangle - \left\langle C - \mathcal{A}^T(y), W\right\rangle \\ \text{s.t.} & W = \alpha\overline{W} + PVP^T \\ & \alpha + \operatorname{tr} V = 1 \\ & \alpha \geq 0, V \succeq 0. \end{array}$$

Expanding $W = \alpha\overline{W} + PVP^T$ into the cost function yields

$$\begin{array}{ll} \min & \frac{1}{2u}\left[\left\langle \mathcal{A}(PVP^T), \mathcal{A}(PVP^T)\right\rangle + 2\alpha\left\langle \mathcal{A}(PVP^T), \mathcal{A}\overline{W}\right\rangle + \alpha^2\left\langle \mathcal{A}\overline{W}, \mathcal{A}\overline{W}\right\rangle\right] \\ & -\left\langle \frac{1}{u}\mathcal{A}^T b + C - \mathcal{A}^T y, P^T V P\right\rangle - \alpha\left\langle \frac{1}{u}\mathcal{A}^T b + C - \mathcal{A}^T y, \overline{W}\right\rangle \\ \text{s.t.} & \alpha + \operatorname{tr} V = 1 \\ & \alpha \geq 0, V \succeq 0. \end{array}$$

Using the svec-operator (see the appendix for a definition and important properties of svec and the symmetric Kronecker product $\otimes_s$) to expand symmetric matrices from S_r into column vectors of length $\binom{r+1}{2}$ we obtain the quadratic program (recall that, for $A, B \in S_r$, $\langle A, B\rangle = \operatorname{svec}(A)^T \operatorname{svec}(B)$ and that $\operatorname{tr} V = \langle I, V\rangle$)

$$\begin{array}{ll} \min & \frac{1}{2}\operatorname{svec}(V)^T Q_{11}\operatorname{svec}(V) + \alpha q_{12}^T \operatorname{svec}(V) + \frac{1}{2}q_{22}\alpha^2 + c_1^T \operatorname{svec}(V) + c_2\alpha \\ & \alpha + s_I^T \operatorname{svec}(V) = 1 \\ & \alpha \geq 0, V \succeq 0, \end{array} \tag{5.1}$$

where (after some technical linear algebra)

$$Q_{11} = \frac{1}{u}\sum_{i=1}^{m} \operatorname{svec}(P^T A_i P)\operatorname{svec}(P^T A_i P)^T \tag{5.2}$$

$$q_{12} = \frac{1}{u}\operatorname{svec}(P^T \mathcal{A}^T(\mathcal{A}\overline{W})P) \tag{5.3}$$

$$q_{22} = \frac{1}{u}\left\langle \mathcal{A}\overline{W}, \mathcal{A}\overline{W}\right\rangle \tag{5.4}$$

$$c_1 = -\operatorname{svec}(P^T(\mathcal{A}^T(\tfrac{1}{u}b - y) + C)P) \tag{5.5}$$

$$c_2 = -(\left\langle \tfrac{1}{u}b - y, \mathcal{A}\overline{W}\right\rangle + \left\langle C, \overline{W}\right\rangle) \tag{5.6}$$

$$s_I = \operatorname{svec}(I).$$

At this point it is advisable to spend some thought on $\overline{W}$. The algorithm is designed for very large and sparse cost matrices C. $\overline{W}$ is of the same size as C. Initially it might be possible to exploit the low rank structure of $\overline{W}$ for efficient representations, but as the algorithm proceeds, the rank of $\overline{W}$ grows inevitably. Thus it is impossible to store all the information of $\overline{W}$. However, as we can see in (5.2) to (5.6), it suffices to have available the vector $\mathcal{A}\overline{W} \in \mathbb{R}^m$ and the scalar $\langle C, \overline{W}\rangle$ to construct the quadratic program. Furthermore, by the linearity of $\mathcal{A}(\cdot)$ and $\langle C, \cdot\rangle$, these values are easily updated whenever $\overline{W}$ is changed.

To solve (5.1) we employ a primal-dual interior point strategy. To formulate the defining equations for the central path we introduce a Lagrange multiplier t for the equality constraint, a dual slack matrix $U \succeq 0$ as complementary variable to V, a dual slack scalar $\beta \geq 0$ as complementary variable to α and a barrier parameter $\mu > 0$. The system reads

$$\begin{array}{rcrcl} F_U & = & Q_{11}\,\mathrm{svec}(V) + \alpha q_{12} + c_1 - t s_I - \mathrm{svec}(U) & = & 0 \\ F_\beta & = & \alpha q_{22} + q_{12}^T\,\mathrm{svec}(V) + c_2 - t - \beta & = & 0 \\ F_1 & = & 1 - \alpha - s_I^T\,\mathrm{svec}(V) & = & 0 \\ & & UV & = & \mu I \\ & & \alpha\beta & = & \mu. \end{array}$$

The step direction $(\Delta\alpha, \Delta\beta, \Delta U, \Delta V, \Delta t)$ is determined via the linearized system

$$\begin{aligned} Q_{11}\,\mathrm{svec}(\Delta V) + \Delta\alpha q_{12} - \Delta t s_I - \mathrm{svec}(\Delta U) &= -F_U \\ q_{12}^T\,\mathrm{svec}(\Delta V) + q_{22}\Delta\alpha - \Delta t - \Delta\beta &= -F_\beta \\ -\Delta\alpha - s_I^T\,\mathrm{svec}(\Delta V) &= -F_1 \\ (U \otimes_s V^{-1})\,\mathrm{svec}(\Delta V) + \mathrm{svec}(\Delta U) &= \mu\ \mathrm{svec}(V^{-1}) - \mathrm{svec}(U) \\ (\beta/\alpha)\Delta\alpha + \Delta\beta &= \mu\alpha^{-1} - \beta \end{aligned}$$

In the current context we prefer the linearization $(U \otimes_s V^{-1})\,\mathrm{svec}(\Delta V) + \mathrm{svec}(\Delta U)$ because it makes the system easy to solve for ΔV with relatively little computational work per iteration. The final system for ΔV reads

$$\begin{aligned} (5.7)\quad & (Q_{11} + U \otimes_s V^{-1} + (\frac{\beta}{\alpha} + q_{22}) s_I s_I^T - q_{12} s_I^T - s_I q_{12}^T)\,\mathrm{svec}(\Delta V) = \\ & = \mu\ \mathrm{svec}(V^{-1}) - \mathrm{svec}(U) - F_U - F_1 q_{12} - (\mu\alpha^{-1} - \beta - \frac{\beta}{\alpha} F_1 - q_{22} F_1) s_I. \end{aligned}$$

It is not too difficult to see that the system matrix is positive definite (because $U \otimes_s V^{-1} \succ 0$ it suffices to show that $Q_{11} + q_{22} s_I s_I^T - q_{12} s_I^T - s_I q_{12}^T \succeq 0$ using $\begin{bmatrix} Q_{11} & q_{12} \\ q_{12} & q_{22} \end{bmatrix} \succeq 0$). The main work per iteration is the factorization of this matrix (with $v \in S_r$ this is $\binom{r+1}{2}^3/3$) and it is not possible to do much better since Q_{11} has to be inverted at some point. Because of the strong dominance of the factorization it pays to employ a predictor corrector approach, but we will not delve into this here.

For $V \in S_r$ a strictly feasible primal starting point is

$$\begin{aligned} V^0 &= I/(r+1) \\ \alpha^0 &= 1/(r+1), \end{aligned}$$

a strictly feasible dual starting point can be constructed by choosing t^0 sufficiently negative such that

$$U^0 = \overset{-1}{\mathrm{svec}}(Q_{11}\,\mathrm{svec}(V^0) + \alpha^0 q_{12} + c_1) - t^0 I \succ 0$$

$$\beta^0 = \alpha q_{22} + q_{12}^T \operatorname{svec}(V) + c_2 - t^0 \ > 0.$$

Starting from this strictly feasible primal-dual pair we compute the first μ by $\mu = (\langle U, V\rangle + \alpha\beta)/(r+1)$, compute the step direction $(\Delta\alpha, \Delta\beta, \Delta U, \Delta V, \Delta t)$ as indicated above , perform a line search with line search parameter $0 < \delta \leq 1$ such that $(\alpha+\delta\Delta\alpha, \beta+\delta\Delta\beta, U+\delta\Delta U, V+\delta\Delta V, t+\delta\Delta t)$ is again strictly feasible, move to this new point, compute a new μ by

$$\mu = \min\left\{\mu_{\text{old}}, \gamma\frac{\langle U, V\rangle + \alpha\beta}{r+1}\right\} \text{ with } \gamma = \begin{cases} 1 & \text{if } \delta \leq \frac{1}{5} \\ \frac{5}{10} - \frac{4}{10}\delta^2 & \text{if } \delta > \frac{1}{5}, \end{cases}$$

and iterate. We stop if $(\langle U, V\rangle + \alpha\beta)/(r+1) < 10^{-7}$.

6. Implementation. In our implementation of the algorithm we largely follow the rules outlined in [21]. In particular u is adapted during the algorithm. The first guess for u is equal to the norm of the first subgradient determined by v^0. The scheme for adapting u is the same as in [21] except for a few changes in parameters. For example the parameter m_L for accepting a step as serious is set to $m_L = 0.2$ and the parameter m_R indicating that the model is so good (progress by the serious step is larger than $m_R[f(x^k) - \hat{f}^k(y^{k+1})]$) that u can be decreased is set to $m_R = 0.7$.

The stopping criterion is formulated in relative precision,

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

with $\varepsilon = 10^{-5}$ in the implementation.

The choice of the upper bound R on the number of columns r of P and the selection of the subspace merits some additional remarks. Observe that by Remark 4.5 it is highly unlikely the r violates the bound $\binom{r+1}{2} \leq m$ even if the number of columns of P is not restricted. $\binom{r+1}{2}$ is also the order of the system matrix in (5.7) and is usually considerably smaller than the size of the system matrix in traditional interior point codes for semidefinite programming which is of order m. Furthermore the order of the matrix variables is r as compared to n for traditional interior point codes. Thus if the number of constraints m is roughly of the same size as n and a matrix of order m is still considered factorizable then running the algorithm without bounding the number of columns of P may turn out to be considerably faster than running an interior point method. This can be observed in practice, see §7.

For huge n and m primal-dual interior point methods are not applicable any more, because X, Z^{-1}, and the system matrix are dense. In this case the proposed bundle approach allows to apply the powerful interior point approach at least on an important subspace of the problem. The correct identification of the relevant subspace in V^* is facilitated by the availability of the complementary variable U^*. U^* helps to discern between the small eigenvalues of V^* (because of the interior point approach we have $V^* \succ 0$!). Eigenvectors v of V^* that are of no importance for the optimal solution of the subproblem will have a large value $v^T U^* v$, whereas eigenvectors, that are ambiguous, will have both, a small eigenvalue $v^T V^* v$ and a small value $v^T U^* v$.

In practice we restrict the number of columns of P to 25 and provide room for at least five new vectors in each iteration (see below). Eigenvectors v that correspond to small but important eigenvalues λ of V^* ($\lambda < 10^{-3}\lambda_{\max}(V^*)$ and $\lambda > 10^{-2} v^T U^* v$) are added to $\overline{W}$; important eigenvectors ($\lambda > 10^{-3}\lambda_{\max}(V^*)$) are added to $\overline{W}$ only if more room is needed for new vectors.

For large m the computation of (5.2) to (5.6) is quite involved. A central object appearing in all constants is the projection of the constraint A_i on the space spanned by P, $P^T A_i P$. Since the A_i are of the same size as X which we assume to be huge, it is important to exploit whatever structure is present in A_i to compute this projection efficiently. In combinatorial applications the A_i are of the form vv^T with v sparse and the projection can be computed efficiently. In the projection step and in particular in forming Q_{11} the size of r is again of strong influence. If we neglect the computation of $\mathrm{svec}(P^T A_i P)$, the computation of Q_{11} still requires $2m\binom{\binom{r+1}{2}+1}{2}$ flops. Indeed, if m is large then for small r the construction of Q_{11} takes longer than solving the associated quadratic semidefinite program.

The large computational costs involved in the construction and solution of the semidefinite subproblems may lead to the conviction that this model may not be worth the trouble. However, the evaluation of the eigenvalue-function is in fact much more expensive. There has been considerable work on computing eigenvalues of huge, sparse matrices, see e.g. [9] and the references therein. For extremal eigenvalues of symmetric matrices there seems to be a general consensus, that Lanczos type methods work best. Iterative methods run into difficulties if the eigenvalues are not well separated. In our context it is to be expected that in the course of the algorithm the largest eigenvalues will get closer and closer till all of them are identical in the optimum. For reasonable convergence block Lanczos algorithms with blocksize corresponding to the largest multiplicity of the eigenvalues have to be employed. During the first ten iterations the largest eigenvalue is usually well separated and the algorithm is fast. But soon the eigenvalues start to cluster, larger and larger blocksizes have to be used, and the eigenvalue problem gets more and more difficult to solve. In order to reduce the number of evaluations it seems worth to employ powerful methods in the cutting plane model. The increase in computation time required to solve the subproblem goes hand in hand with the difficulty of the eigenvalue problem because of the correspondence of the rank of P and the number of clustered eigenvalues.

Iterative methods for computing maximal eigenvectors generically offer approximate eigenvectors to several other large eigenvalues, as well. The space spanned by these approximate eigenvectors is likely to be a good approximation of the true eigenspace. If the maximal number of columns for P is not yet attained it may be worth to include several of these approximate eigenvectors as well.

In our algorithm we use a block Lanczos code of our own that is based on a Fortran code of Hua (we guess that this is Hua Dai of [47]). It works with complete orthogonalization and employs Chebyshev iterations for acceleration. The choice of the blocksize is based on the approximate eigenvalues produced by previous evaluations but is at most 30. Four block Lanczos steps are followed by twenty Chebyshev iterations. This scheme is repeated till the maximal eigenvalue is found to the required relative precision. The relative precision depends on the distance of the maximal to the second largest eigenvalue but is bounded by 10^{-6}. As starting vectors we use the complete block of eigenvectors and Lanczos-vectors from the previous evaluation.

7. Combinatorial applications. The combinatorial problem we investigate is quadratic programming in $\{-1,1\}$ variables,

$$\text{(MC)} \qquad \max\ x^T C x \ \text{ s.t. } x \in \{-1,1\}^n .$$

In the case that C is the Laplace matrix of a (possible weighted) graph the problem is known to be equivalent to the max-cut problem.

The standard semidefinite relaxation is based on the identity $x^T C x = \langle C, xx^T \rangle$. For all $\{-1,1\}^n$ vectors, xx^T is a positive semidefinite matrix with all diagonal elements equal to one. We relax xx^T to $X \succeq 0$ and $\mathrm{diag}(X) = e$ and obtain the following primal-dual pair of semidefinite programs,

$$\text{(PMC)} \quad \begin{array}{rl} \max & \langle C, X\rangle \\ \text{s.t.} & \operatorname{diag}(X) = e \\ & X \succeq 0 \end{array} \qquad \text{(DMC)} \quad \begin{array}{rl} \min & e^T y \\ \text{s.t.} & C + Z - \operatorname{Diag}(y) = 0 \\ & Z \succeq 0. \end{array}$$

For non-negatively weighted graphs a celebrated result of Goemans and Williamson [8] says, that there is always a cut within .878 of the optimal value of the relaxation.

One of the first attempts to approximate (DMC) using eigenvalue optimization is contained in [39]. The authors use the Bundle code of Schramm and Zowe [42] with a limited number of bundle iterations, and so do not solve (DMC) exactly. So far the only practical algorithms for computing the optimal value were primal-dual interior point algorithms. However these are not able to exploit the sparsity of the cost function and have to cope with dense matrices X and Z^{-1}. An alternative approach based on a combination of the power method with a generic optimization scheme of Plotkin, Shmoys, and Tardos [37] was proposed in [22] but seems to be purely theoretical.

In Table 7.1 we compare the proposed bundle method to our semidefinite primal-dual interior point code of [14] (called PDIP in the sequel) for graphs on $n = m = 800$ nodes that were generated by `rudy`, a machine independent graph generator written by G. Rinaldi. Table 7.7 contains the command lines specifying the graphs. Graphs G_1 to G_5 are unweighted random graphs with a density of 6% (approx. 19000 edges). G_6 to G_{10} are the same graphs with random edge weights from $\{-1, 1\}$. G_{11} to G_{13} are toroidal grids with random edge weights from $\{-1, 1\}$ (1600 edges). G_{14} to G_{17} are unweighted 'almost' planar graphs having as edge set the union of two (almost maximal) planar graphs (approx. 4500 edges). G_{18} to G_{21} are the same almost planar graphs with random edge weights from $\{-1, 1\}$. In all cases the cost matrix C is the Laplace matrix of the graph divided by 4, i.e., let A denote the (weighted) adjacency matrix of G, then

$$C = \frac{1}{4}(\operatorname{Diag}(Ae) - A).$$

For a description of the code PDIP see [14], the termination criterion requires the gap between primal and dual optimal solution to be closed to a relative accuracy of $5 \cdot 10^{-6}$.

For the bundle algorithm, (DMC) is transformed into an eigenvalue optimization problem as described in §2. In addition the diagonal of C is removed so that, in fact, the algorithm works on the problem

$$\min_{y \in \mathbb{R}^n} n\lambda_{\max}(\bar{C} - \operatorname{Diag}(y)) + e^T y$$

with $\bar{C} = \frac{1}{4}(\operatorname{Diag}(\operatorname{diag}(A)) - A)$. This does not change problem (PMC) because the diagonal elements of X are fixed to one. The offset $\frac{1}{4}e^T(Ae - \operatorname{diag}(A))$ is added to the output only and has no influence on the algorithm whatsoever, in particular it has no influence on the stopping criterion. As starting vector y^0 we choose the zero vector. All other parameters are as described in §6.

All computation times, for the interior point code PDIP as well as for the bundle code, refer to the same machine, a Sun sparc Ultra 1 with a Model 140 UltraSPARC CPU and 64 MB RAM. The time measured is the user time and it is given in the format $hh : mm : ss$, hours:minutes:seconds, leading zeros are dropped.

The first column of Table 7.1 identifies the graphs. The second and third refer to PDIP and contain the optimal objective value produced (these can be regarded as highly accurate solutions) and the computation time. The fourth and fifth column give the same numbers for the bundle code.

On these examples the bundle code is superior to PDIP. Although the examples do belong to the favorable class of instances having small m and relatively large n, the difference in computation time is astonishing. Note that the termination criterion used in the bundle code is quite accurate, except for G_{11} which seems to be a difficult problem for the bundle method. This deviation in accuracy is *not* caused by cancellations in connection with the offset. The difficulty of an example does not seem to depend on the number of nonzeros but rather on the shape of the objective function. For toroidal grid graphs the maximum cut is likely to be not unique, thus the objective function will be rather flat. This flatness has its effect on the distribution of the eigenvalues in the optimal solution. Indeed, for G_{11} more eigenvalues cluster around the maximal eigenvalue than for the other problems. We illustrate this in Table 7.2, which gives the 30 largest eigenvalues of the solution at termination for problems G_1, G_6, G_{11}, G_{14}, and G_{18}.

TABLE 7.1

Comparison of the interior point (PDIP) and the bundle (B) approach. sol *gives the computed solution value and* time *gives the computation time.*

	PDIP-sol	PDIP-time	B-sol	B-time
G_1	12083.20	1:18:42	12083.22	4:11
G_2	12089.43	1:19:14	12089.45	5:19
G_3	12084.33	1:25:30	12084.34	4:38
G_4	12111.45	1:23:16	12111.46	3:37
G_5	12099.89	1:27:09	12099.91	4:38
G_6	2656.16	1:24:53	2656.18	3:57
G_7	2489.26	1:32:34	2489.29	7:54
G_8	2506.93	1:21:47	2506.95	3:38
G_9	2528.73	1:30:36	2528.75	3:51
G_{10}	2485.06	1:24:30	2485.08	3:56
G_{11}	629.16	1:28:41	629.21	45:26
G_{12}	623.87	1:34:55	623.89	31:14
G_{13}	647.13	1:37:52	647.14	18:44
G_{14}	3191.57	2:05:24	3191.58	14:30
G_{15}	3171.56	2:25:53	3171.56	23:20
G_{16}	3175.02	2:18:21	3175.04	16:59
G_{17}	3171.32	2:13:10	3171.35	16:32
G_{18}	1166.01	2:58:22	1166.02	18:32
G_{19}	1082.01	3:07:58	1082.04	12:27
G_{20}	1111.39	3:12:41	1111.40	11:59
G_{21}	1104.28	3:13:53	1104.29	13:35

Table 7.3 provides additional information on the performance of the bundle algorithm on the examples of Table 7.1. The second column gives the accumulated time spent in the eigenvalue computation, it accounts for roughly 90% of the computation time. *serious* displays the number of serious steps, *iter* gives the total number of iterations including both, serious and null steps. $\|g\|$ is the norm of the subgradient arising from the last optimal W^{k+1} before termination. For G_{11} the norm is considerably higher than for all other examples. Since the desired accuracy was not achieved for G_{11} by the standard stopping criterion it may be worth to consider an alternative stopping criterion taking into account the norm of the subgradient as well. Column *max-r* gives the maximal rank of P attained over all iterations. The rank of P would

TABLE 7.2
The 30 maximal eigenvalues after termination of examples G_1, G_6, G_{11}, G_{14}, and G_{18}.

	G_1	G_6	G_{11}	G_{14}	G_{18}
1	3.1190	3.2240	0.7653	1.0557	1.4175
2	·	3.2239	·	·	·
3	·	·	·	·	·
4	·	·	·	·	·
5	·	·	·	·	·
6	·	·	·	·	·
7	·	·	0.7653	·	·
8	·	·	0.7652	·	·
9	·	·	0.7652	·	·
10	·	·	0.7652	·	1.4175
11	·	·	0.7652	·	1.4155
12	·	·	0.7651	·	1.4090
13	3.1190	3.2239	0.7651	1.0557	1.4047
14	3.1135	3.2181	0.7650	1.0515	1.4007
15	3.1020	3.1968	0.7650	1.0500	1.3905
16	3.0928	3.1774	0.7649	1.0490	1.3834
17	3.0772	3.1556	0.7649	1.0450	1.3814
18	3.0594	2.7886	0.7648	1.0432	1.3798
19	2.7214	2.7716	0.7647	1.0398	1.3725
20	2.6964	2.7681	0.7646	1.0379	1.3709
21	2.6858	2.7269	0.7645	1.0358	1.3652
22	2.6834	2.6756	0.7644	1.0341	1.3583
23	2.6682	2.5851	0.7641	1.0331	1.3555
24	2.6649	2.5239	0.7639	1.0284	1.3510
25	2.6468	2.2357	0.7638	1.0266	1.3495
26	2.6274	1.8722	0.7636	1.0239	1.3480
27	2.6035	1.8473	0.7634	1.0231	1.3417
28	2.5137	1.7974	0.7633	1.0211	1.3397
29	2.4840	1.4859	0.7630	1.0180	1.3345
30	2.3281	1.4411	0.7629	1.0175	1.3291

have been bounded by 25, but this bound never came into effect for any of these examples. Aggregation was not necessary. Observe that the theoretic bound allows for r up to 39, yet the maximal rank is only half this number. The last column gives the time when the objective value was first within 10^{-3} of the optimum.

For combinatorial applications high accuracy of the optimal solution is of minor importance. An algorithm should deliver a reasonable bound fast and its solution should provide some hint on how a good feasible solution can be constructed. The bundle algorithm offers both. With respect to computation time the bundle algorithm displays the usual behavior of subgradient algorithms. Initially progress is very fast, but as the bound approaches the optimum there is a strong tailing off effect. We illustrate this by giving the objective values and computation times for the serious steps of example G_6 (the diagonal offset is $+77$ in this example) in Table 7.4. After one minute the bound is within 0.1% of the optimum. For the other examples see the last column of Table 7.3.

With respect to a primal feasible solution observe that $P^k V_*^k (P^k)^T$ is a successively better and better approximation to the primal optimal solution X^*. In case too much information is stored in the aggregate vector $\mathcal{A}\overline{W}^k$ (remember that it is not advisable to store $\overline{W}^k$ itself), P^k may be enriched with additional Lanczos-vectors from the eigenvalue computation. The solution of this enlarged quadratic semidefinite subproblem will be an acceptable approximation of X^*. It is not necessary to

TABLE 7.3

Additional information about the bundle algorithm for the examples of Table 7.1. λ-time *gives the total amount of time spent for computing the eigenvalues and eigenvectors,* serious *gives the number of serious steps,* iter *the total number of iterations including null steps.* $\|g\|$ *refers to the norm of the gradient resulting from the optimal solution of the last semidefinite subproblem.* max-r *is the maximum number of columns used in P (the limit would have been 25).* 0.1%-time *gives the time when the bound is within* 10^{-3} *of the optimum in relative precision.*

	λ-time	serious	iter	$\|g\|$	max-r	0.1%-time
G_1	3:12	22	33	0.1639	18	48
G_2	4:04	21	36	0.05035	18	1:02
G_3	3:20	21	33	0.04107	19	57
G_4	2:38	19	27	0.08235	19	54
G_5	3:41	23	39	0.04425	17	46
G_6	2:42	21	35	0.0646	18	1:09
G_7	5:46	24	65	0.0854	17	57
G_8	2:39	23	38	0.1549	17	59
G_9	2:59	24	34	0.0711	17	1:04
G_{10}	2:56	23	37	0.02997	17	1:15
G_{11}	42:10	97	172	0.4696	15	17:04
G_{12}	28:47	50	130	0.2579	15	8:19
G_{13}	17:24	43	78	0.218	15	6:17
G_{14}	12:43	41	59	0.1682	18	3:09
G_{15}	20:16	44	89	0.1059	18	3:08
G_{16}	14:35	31	69	0.2246	19	3:11
G_{17}	14:38	41	65	0.2079	18	3:22
G_{18}	16:46	38	98	0.08161	15	4:15
G_{19}	11:24	41	71	0.1571	15	3:34
G_{20}	11:05	42	71	0.08226	15	3:50
G_{21}	12:23	44	80	0.09432	15	3:32

construct the whole matrix X^*. In fact, the factorized form $(P^k\sqrt{V_*^k})(P^k\sqrt{V_*^k})^T$ is much more convenient to work with. For example the approximation algorithm of Goemans and Williamson [8] requires precisely this factorization. A particular x_{ij} element of X^* is easily computed by the inner product of row i and j of the $n \times r$ matrix $P^k\sqrt{V_*^k}$. In principle this opens the door for branch and cut approaches to improve the initial relaxation. This will be the subject of further work.

Table 7.5 gives a similar set of examples for $n = m = 2000$.

A last set of examples is devoted to the Lovász ϑ-function [27] which yields an upper bound on the cardinality of a maximal independent (or stable) set of a graph. For implementational convenience we use its formulation within the quadratic $\{-1, 1\}$ programming setting, see [24]. For a graph with k nodes and h edges we obtain a semidefinite program with matrix variables of order $n = k + 1$ and $m = k + 1 + h$ constraints. The examples we are going to consider have more than one thousand nodes and more than six thousand edges. For these examples interior point methods are not applicable any more because of memory requirements. It should be clear from the examples of Table 7.5 that there is also little hope for the bundle method to terminate within reasonable time. However, the most significant progress is achieved in the beginning and for the bundle method memory consumption is not a problem. We run these examples with a time limit of five hours. More precisely, the algorithm is terminated after the first serious step that occurs after five hours of computation time.

The graph instances are of the same type as above. The computational results are displayed in Table 7.6. The new columns n and m give the order of the matrix variable and the number of constraints, respectively. Observe that the toroidal grid graphs

TABLE 7.4
Detailed account of the serious steps of example G_6.

iter	value	time	$\|\|g\|\|$	max-r
0	2861.20			
1	2821.75	3	42.02	3
2	2798.60	4	28.83	7
3	2782.99	6	25.94	8
4	2736.11	9	18.58	8
5	2704.98	13	12.69	10
6	2685.84	17	10.05	12
7	2679.95	23	7.46	14
8	2670.10	28	5.992	15
9	2666.07	34	4.173	16
10	2660.31	48	3.268	16
11	2658.65	1:09	0.6295	17
12	2656.66	1:32	0.7753	17
13	2656.42	1:41	0.6636	18
14	2656.31	1:57	0.431	18
15	2656.27	2:14	0.2997	18
16	2656.22	2:45	0.1609	18
17	2656.20	3:01	0.13	18
18	2656.19	3:17	0.09695	18
19	2656.19	3:33	0.07243	18
20	2656.17	3:49	0.05042	18
21	2656.17	3:57	0.0646	18

TABLE 7.5
Examples for $n = m = 2000$.

	B-sol	B-time	λ-time	serious	iter	$\|\|g\|\|$	max-r
G_{22}	14135.98	38:11	28:00	26	52	0.0781	23
G_{23}	14145.58	1:19:29	1:01:06	32	107	0.3707	23
G_{24}	14140.88	28:04	21:11	25	40	0.1565	23
G_{25}	14144.30	43:44	32:26	27	59	0.3452	24
G_{26}	14132.93	34:45	26:37	31	48	0.3066	23
G_{27}	4141.68	24:56	18:54	23	37	0.1423	23
G_{28}	4100.81	29:41	21:08	23	39	0.1954	25
G_{29}	4208.94	48:16	37:53	27	65	0.1725	22
G_{30}	4215.42	1:02:39	48:16	27	83	0.1282	22
G_{31}	4116.70	26:11	19:02	24	38	0.1889	24
G_{32}	1567.80	6:20:54	6:09:52	144	312	0.892	15
G_{33}	1544.42	6:04:22	5:53:37	112	305	0.6887	15
G_{34}	1546.82	3:46:31	3:39:26	105	198	0.8369	15
G_{35}	8014.81	4:31:17	3:54:01	61	209	0.2397	22
G_{36}	8006.04	2:56:10	2:31:09	62	115	0.2634	24
G_{37}	8018.68	3:10:01	2:46:35	58	130	0.254	23
G_{38}	8015.01	4:03:53	3:39:24	58	155	0.1937	22
G_{39}	2877.71	1:20:24	1:12:23	50	85	0.2194	20
G_{40}	2864.96	2:42:02	2:30:21	59	158	0.2737	19
G_{41}	2865.29	1:41:33	1:32:50	51	108	0.1954	19
G_{42}	2946.29	1:32:45	1:24:12	59	93	0.1535	20

G_{48} and G_{49} are perfect with independence number 1500; the independence number of G_{50} is 1440 but G_{50} is not perfect. We do not know the independence number of the other graphs. Except for G_{48} and G_{49}, which have $\theta(G_{48}) = \theta(G_{49}) = 1500$ by perfectness, it is hard to judge the quality of the solutions. Tracing the development of the bounds the last serious steps of examples G_{43} to G_{47} and G_{51} to G_{54} still produced improvements of 0.5% to 1%. This and the rather large norm of the subgradient of

TABLE 7.6
Upper bound on the ϑ-function after five hours of computation time.

	n	m	B-sol	B-time	λ-time	serious	iter	$\|\|g\|\|$	max-r
G_{43}	1001	10991	308.47	6:02:20	3:40:57	38	104	0.5098	25
G_{44}	1001	10991	310.13	5:06:31	3:14:25	39	88	0.5493	25
G_{45}	1001	10991	309.00	5:10:38	3:19:25	40	87	0.5532	25
G_{46}	1001	10991	309.21	5:35:56	3:31:42	42	99	0.5535	25
G_{47}	1001	10991	310.84	5:24:52	3:10:35	44	100	0.5558	25
G_{48}	3001	9001	1526.53	5:11:31	4:59:57	54	94	0.4062	15
G_{49}	3001	9001	1521.24	5:06:21	4:53:45	53	102	0.3751	15
G_{50}	3001	9001	1536.12	5:17:51	5:01:25	50	124	0.4728	15
G_{51}	1001	6910	455.21	5:07:40	4:29:00	32	118	2.556	25
G_{52}	1001	6917	465.12	5:09:02	4:38:30	41	108	2.683	25
G_{53}	1001	6915	463.86	5:08:36	4:36:34	41	104	2.593	25
G_{54}	1001	6917	466.04	5:02:21	4:32:21	40	98	2.590	25

G_{51} and G_{54} indicate that the values cannot be expected to be 'good' approximations of the ϑ-function. Also note, that the size of the subspace required for G_{48} to G_{50} is still well below 25. In examples G_{51} to G_{54} the value of α is almost negligible, but for G_{43} to G_{47} the value of α is roughly $1/3$ at termination. Thus for these examples the restriction to 25 columns became relevant.

The computational results of Table 7.6 demonstrate that the algorithm has its limits. Nonetheless the bounds obtained are still useful and the primal approximation corresponding to the subgradient is a reasonable starting point for primal heuristics.

8. Conclusions and extensions. We have proposed a proximal bundle method for solving semidefinite programs with large sparse or strongly structured coefficient matrices. The semidefinite constraint is lifted into the objective function by means of a Lagrange multiplier a whose correct value is not known in general, except for problems with fixed primal trace. In the latter case a is precisely the value of the trace. The approach differs from previous bundle methods in that the subproblem is tailored for semidefinite programming. In fact the whole approach can be interpreted as semidefinite programming over subspaces where the subspace is successively corrected and improved till the optimal subspace is identified. The set of subgradients modeled by the semidefinite subproblem is a superset of the subgradients used in the traditional polyhedral cutting plane model. Therefore convergence of the new method is a direct consequence of previous proofs for traditional bundle methods. It is not yet clear whether the specialized model admits stronger convergence results. The choice of u is still very much an open problem of high practical importance.

For (constrained) quadratic $\{-1, 1\}$-programming the method offers a good bound within reasonable time and allows to construct an approximate primal optimal solution (of the relaxation) in compact representation. To improve the bound by a cutting plane approach the algorithm must be able to deal with sign constraints on the y-variables. In principle it is not difficult to model the sign constraints in the semidefinite subproblem. However, as a consequence the influence of the sign constrained y variables on the cost coefficients of the quadratic subproblem cannot be eliminated any longer, rendering the method impractical even for a moderate number of cutting planes. Alternatively one might consider active set methods but these entail the danger of destroying convergence. Together with K.C. Kiwiel we are currently working on alternative methods for incorporating sign constraints on y [13].

The backbone of the method is an efficient routine for computing the maximal eigenvalue of huge structured symmetric matrices. Although our own implementation

TABLE 7.7
Arguments for generating the graphs by the graph generator `rudy`.

G_1	`-rnd_graph 800 6 8001`
G_2	`-rnd_graph 800 6 8002`
G_3	`-rnd_graph 800 6 8003`
G_4	`-rnd_graph 800 6 8004`
G_5	`-rnd_graph 800 6 8005`
G_6	`-rnd_graph 800 6 8001 -random 0 1 8001 -times 2 -plus -1`
G_7	`-rnd_graph 800 6 8002 -random 0 1 8002 -times 2 -plus -1`
G_8	`-rnd_graph 800 6 8003 -random 0 1 8003 -times 2 -plus -1`
G_9	`-rnd_graph 800 6 8004 -random 0 1 8004 -times 2 -plus -1`
G_{10}	`-rnd_graph 800 6 8005 -random 0 1 8005 -times 2 -plus -1`
G_{11}	`-toroidal_grid_2D 100 8 -random 0 1 8001 -times 2 -plus -1`
G_{12}	`-toroidal_grid_2D 50 16 -random 0 1 8002 -times 2 -plus -1`
G_{13}	`-toroidal_grid_2D 25 32 -random 0 1 8003 -times 2 -plus -1`
G_{14}	`-planar 800 99 8001 -planar 800 99 8002 +`
G_{15}	`-planar 800 99 8003 -planar 800 99 8004 +`
G_{16}	`-planar 800 99 8005 -planar 800 99 8006 +`
G_{17}	`-planar 800 99 8007 -planar 800 99 8008 +`
G_{18}	`-planar 800 99 8001 -planar 800 99 8002 + -random 0 1 8001 -times 2 -plus -1`
G_{19}	`-planar 800 99 8003 -planar 800 99 8004 + -random 0 1 8002 -times 2 -plus -1`
G_{20}	`-planar 800 99 8005 -planar 800 99 8006 + -random 0 1 8003 -times 2 -plus -1`
G_{21}	`-planar 800 99 8007 -planar 800 99 8008 + -random 0 1 8004 -times 2 -plus -1`
G_{22}	`-rnd_graph 2000 1 20001`
G_{23}	`-rnd_graph 2000 1 20002`
G_{24}	`-rnd_graph 2000 1 20003`
G_{25}	`-rnd_graph 2000 1 20004`
G_{26}	`-rnd_graph 2000 1 20005`
G_{27}	`-rnd_graph 2000 1 20001 -random 0 1 20001 -times 2 -plus -1`
G_{28}	`-rnd_graph 2000 1 20002 -random 0 1 20002 -times 2 -plus -1`
G_{29}	`-rnd_graph 2000 1 20003 -random 0 1 20003 -times 2 -plus -1`
G_{30}	`-rnd_graph 2000 1 20004 -random 0 1 20004 -times 2 -plus -1`
G_{31}	`-rnd_graph 2000 1 20005 -random 0 1 20005 -times 2 -plus -1`
G_{32}	`-toroidal_grid_2D 100 20 -random 0 1 20003 -times 2 -plus -1`
G_{33}	`-toroidal_grid_2D 80 25 -random 0 1 20002 -times 2 -plus -1`
G_{34}	`-toroidal_grid_2D 50 40 -random 0 1 20001 -times 2 -plus -1`
G_{35}	`-planar 2000 99 20001 -planar 2000 99 20002 +`
G_{36}	`-planar 2000 99 20003 -planar 2000 99 20004 +`
G_{37}	`-planar 2000 99 20005 -planar 2000 99 20006 +`
G_{38}	`-planar 2000 99 20007 -planar 2000 99 20008 +`
G_{39}	`-planar 2000 99 20001 -planar 2000 99 20002 + -random 0 1 20001 -times 2 -plus -1`
G_{40}	`-planar 2000 99 20003 -planar 2000 99 20004 + -random 0 1 20002 -times 2 -plus -1`
G_{41}	`-planar 2000 99 20005 -planar 2000 99 20006 + -random 0 1 20003 -times 2 -plus -1`
G_{42}	`-planar 2000 99 20007 -planar 2000 99 20008 + -random 0 1 20004 -times 2 -plus -1`
G_{43}	`-rnd_graph 1000 2 10001`
G_{44}	`-rnd_graph 1000 2 10002`
G_{45}	`-rnd_graph 1000 2 10003`
G_{46}	`-rnd_graph 1000 2 10004`
G_{47}	`-rnd_graph 1000 2 10005`
G_{48}	`-toroidal_grid_2D 50 60`
G_{49}	`-toroidal_grid_2D 30 100`
G_{50}	`-toroidal_grid_2D 25 120`
G_{51}	`-planar 1000 100 10001 -planar 1000 100 10002 +`
G_{52}	`-planar 1000 100 10003 -planar 1000 100 10004 +`
G_{53}	`-planar 1000 100 10005 -planar 1000 100 10006 +`
G_{54}	`-planar 1000 100 10007 -planar 1000 100 10008 +`

(based on the code of Hua) seems to work sufficiently stable there is certainly much room for improvement. A straight forward approach to achieve serious speed-ups is to implement the algorithm on parallel machines, see for instance [43]. Rather recently interest in the Lanczos method has risen again, see [25, 3, 5, 10, 30] and references therein. Most of these papers are based on the concept of an implicit restart proposed in [44] which is a polynomial acceleration approach that does not require additional matrix vector multiplications. It will be interesting to test these new ideas within the bundle framework.

We thank K.C. Kiwiel for fruitful discussions and C. Lemaréchal and an anonymous referee for their constructive critisism that helped to improve the presentation.

Appendix. Notation.

$\mathbb{R}^n$	real column vector of dimension n
$M_{m,n}$	$m \times n$ real matrices
S_n	$n \times n$ symmetric real matrices
S_n^{++}	$n \times n$ symmetric positive definite matrices
S_n^{+}	$n \times n$ symmetric positive semidefinite matrices
$A \succ 0$	A is positive definite
$A \succeq 0$	A is positive semidefinite
I, I_n	identity of appropriate size or of size n
e	vector of all ones of appropriate dimension
$\lambda_{\max}(A)$	maximal eigenvalue of A
$\operatorname{tr} A$	trace of $A \in M_{n,n}$, $\operatorname{tr} A = \sum_{i=1}^n a_{ii} = \sum_{i=1}^n \lambda_i(A)$
$\langle A, B\rangle$	inner product in $M_{m,n}$, $\langle A, B\rangle = \operatorname{tr}(B^T A)$
$\operatorname{svec}(A)$	$\binom{n+1}{2}$ dimensional vector representation of $A \in S_n$
$A \otimes_s B$	symmetric Kronecker product of $A \in M_{m,n}, B \in M_{k,l}$
$\operatorname{diag}(A)$	the diagonal of $A \in M_n$ as a column vector
$\operatorname{Diag}(v)$	diagonal matrix with v on its main diagonal

S_n is isomorphic to $\mathbb{R}^{\binom{n+1}{2}}$ via the map $\operatorname{svec}(A)$ defined by stacking the columns of the lower triangle of A on top of each other and multiplying the offdiagonal elements with $\sqrt{2}$,

$$\operatorname{svec}(A) := \left[a_{11}, \sqrt{2}a_{21}, \ldots, \sqrt{2}a_{n1}, a_{22}, \sqrt{2}a_{32}, \ldots, a_{nn}\right]^T.$$

The factor $\sqrt{2}$ for offdiagonal elements ensures that, for $A, B \in S_n$,

$$\langle A, B\rangle = \operatorname{tr}(AB) = \operatorname{svec}(A)^T \operatorname{svec}(B).$$

The symmetric Kronecker product $\otimes_s$ is defined for arbitrary square matrices $A, B \in M_{n,n}$ by its action on a vector $\operatorname{svec}(C)$ for a symmetric matrix $C \in S_n$,

$$(A \otimes_s B) \operatorname{svec}(C) := \frac{1}{2} \operatorname{svec}(BCA^T + ACB^T).$$

Both concepts were first introduced in [2]. Here we use the notation introduced in [45]. From the latter paper we also cite some properties of the symmetric Kronecker product for the convenience of the reader.

1. $A \otimes_s B = B \otimes_s A$
2. $(A \otimes_s B)^T = B^T \otimes_s A^T$
3. $A \otimes_s I$ is symmetric if and only if A is.
4. $(A \otimes_s A)^{-1} = A^{-1} \otimes_s A^{-1}$
5. $(A \otimes_s B)(C \otimes_s D) = \frac{1}{2}(AC \otimes_s BD + AD \otimes_s BC)$
6. If $A \succ 0$ and $B \succ 0$ then $(A \otimes_s B) \succ 0$
7. $\operatorname{svec}(A)^T \operatorname{svec}(B) = \langle A, B\rangle = \operatorname{tr}(AB)$

REFERENCES

[1] F. Alizadeh, *Interior point methods in semidefinite programming with applications to combinatorial optimization*, SIAM J. Optim., 5 (1995), pp. 13–51.

[2] F. Alizadeh, J.-P. A. Haeberly, and M. L. Overton, *Primal-dual interior-point methods for semidefinite programming: Convergence rates, stability and numerical results*, SIAM J. Optim., 8 (1998), pp. 746–768.

[3] J. Baglama, D. Calvetti, and L. Reichel, *Iterative methods for the computation of a few eigenvalues of a large symmetric matrix*, BIT, 36 (1996), pp. 400–421.

[4] S. Benson, Y. Ye, and X. Zhang, *Solving large-scale sparse semidefinite programs for combinatorial optimization*, working paper, Department of Management Science, University of Iowa, IA, 52242, USA, Sept. 1997. To appear in SIAM J. Opt.

[5] D. Calvetti, L. Reichel, and D. Sorensen, *An implicitly restarted Lanczos method for large symmetric eigenvalue problems*, Electronic Transactions on Numerical Analysis, 2 (1994), pp. 1–21.

[6] J. Cullum, W. E. Donath, and P. Wolfe, *The minimization of certain nondifferentiable sums of eigenvalues of symmetric matrices*, Math. Programming Study, 3 (1975), pp. 35–55.

[7] L. Faybusovich, *Semidefinite programming: a path-following algorithm for a linear-quadratic functional*, SIAM J. Optim., (1996), pp. 1007–1024.

[8] M. X. Goemans and D. P. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, J. ACM, 42 (1995), pp. 1115–1145.

[9] G. H. Golub and C. F. van Loan, *Matrix Computations*, The Johns Hopkins University Press, 2^{nd} ed., 1989.

[10] R. G. Grimes, J. G. Lewis, and H. D. Simon, *A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 228–272.

[11] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, vol. 2 of Algorithms and Combinatorics, Springer, 2^{nd} ed., 1988.

[12] C. Helmberg, *Fixing variables in semidefinite relaxations*, in Algorithms – ESA'97, R. Burkard and G. Woeginger, eds., vol. 1284 of Lecture Notes in Computer Science, Springer, Sept. 1997, pp. 259–270.

[13] C. Helmberg, K. C. Kiwiel, and F. Rendl, *Incorporating inequality constraints in the spectral bundle method*, in Integer Programming and Combinatorial Optimization, R. E. Bixby, E. A. Boyd, and R. Z. Ríos-Mercado, eds., vol. 1412 of Lecture Notes in Computer Science, Springer, June 1998, pp. 423–435.

[14] C. Helmberg and F. Rendl, *Solving quadratic (0,1)-problems by semidefinite programs and cutting planes*, Math. Programming, 82 (1998), pp. 291–315.

[15] C. Helmberg, F. Rendl, R. J. Vanderbei, and H. Wolkowicz, *An interior–point method for semidefinite programming*, SIAM J. Optim., 6 (1996), pp. 342–361.

[16] C. Helmberg, F. Rendl, and R. Weismantel, *Quadratic knapsack relaxations using cutting planes and semidefinite programming*, in Integer Programming and Combinatorial Optimization, W. H. Cunningham, S. T. McCormick, and M. Queyranne, eds., vol. 1084 of Lecture Notes in Computer Science, Springer, June 1996, pp. 175–189.

[17] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms I*, vol. 305 of Grundlehren der mathematischen Wissenschaften, Springer, Berlin, Heidelberg, 1993.

[18] ———, *Convex Analysis and Minimization Algorithms II*, vol. 306 of Grundlehren der mathematischen Wissenschaften, Springer, Berlin, Heidelberg, 1993.

[19] F. Jarre, *An interior–point method for minimizing the maximum eigenvalue of a linear combination of matrices*, Siam J. Control and Optimization, 31 (1993), pp. 1360–1377.

[20] S. E. Karisch, F. Rendl, and J. Clausen, *Solving graph bisection problems with semidefinite programming*, Technical Report DIKU-TR-97/9, Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen, Denmark, July 1997. To appear in INFORMS J. Computing.

[21] K. C. Kiwiel, *Proximity control in bundle methods for convex nondifferentiable minimization*, Math. Programming, 46 (1990), pp. 105–122.

[22] P. Klein and H.-I. Lu, *Efficient approximation algorithms for semidefinite programs arising from MAXCUT and COLORING*, in STOC'96, 1996, pp. 338–347.

[23] M. Kojima, S. Shindoh, and S. Hara, *Interior–point methods for the monotone linear complementarity problem in symmetric matrices*, SIAM J. Optim., 7 (1997), pp. 86–125.

[24] M. Laurent, S. Poljak, and F. Rendl, *Connections between semidefinite relaxations of the max-cut and stable set problems*, Math. Programming, 77 (1997), pp. 225–246.

[25] R. B. Lehoucq and K. J. Maschhoff, *Implementation of an implicitly restarted block Arnoldi method*, Technical Report MCS-P649-0297, Argonne National Laboratory, 1997.

[26] C. Lemaréchal, F. Oustry, and C. Sagastizábal, *The $\mathcal{U}$-Lagrangian of a convex function*, technical report, INRIA, France, 1996. To appear in Transactions of the AMS.

[27] L. Lovász, *On the Shannon capacity of a graph*, IEEE Transactions on Information Theory, IT-25 (1979), pp. 1–7.

[28] L. Lovász and A. Schrijver, *Cones of matrices and set-functions and 0-1 optimization*, SIAM J. Optim., 1 (1991), pp. 166–190.

[29] R. D. C. Monteiro, *Polynomial convergence of primal-dual algorithms for semidefinite programming based on Monteiro and Zhang family of directions*, SIAM J. Optim., 8 (1998), pp. 797–812.

[30] R. B. Morgan and D. S. Scott, *Preconditioning the Lanczos algorithm for sparse symmetric eigenvalue problems*, SIAM J. Sci. Comput., 14 (1993), pp. 585–593.

[31] Y. Nesterov and A. Nemirovskii, *Interior–Point Polynomial Algorithms in Convex Programming*, SIAM Studies in Applied Mathematics, Philadelphia, 1994.

[32] Y. Nesterov and M. J. Todd, *Self-scaled barriers and interior-point methods for convex programming*, Math. Oper. Res., 22 (1997), pp. 1–42.

[33] F. Oustry, *The $\mathcal{U}$-Lagrangian of the maximum eigenvalue function*, technical report, INRIA-ENSTA, France, Nov. 1996. To appear in SIAM J. Optim.

[34] ———, *A second-order bundle method to minimize the maximum eigenvalue function*, technical report, INRIA-ENSTA, France, Nov. 1997. Submitted to Math. Programming.

[35] M. L. Overton, *Large–scale optimization of eigenvalues*, SIAM J. Optim., 2 (1992), pp. 88–120.

[36] G. Pataki, *On the rank of extreme matrices in semidefinite programming and the multiplicity of optimal eigenvalues*, Math. Oper. Res., 23 (1998), pp. 339–358.

[37] S. A. Plotkin, D. B. Shmoys, and E. Tardos, *Fast approximation algorithms for fractional packing and covering problems*, Math. Oper. Res., 20 (1995), pp. 257–301.

[38] E. Polak and Y. Wardi, *Nondifferentiable optimization algorithm for designing control systems having singular value inequalities*, Automatica, 18 (1982), pp. 267–283.

[39] S. Poljak and F. Rendl, *Node and edge relaxations of the max-cut problem*, Computing, 52 (1994), pp. 123–137.

[40] ———, *Nonpolyhedral relaxations of graph-bisection problems*, SIAM J. Optim., 5 (1995), pp. 467–487.

[41] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, New Jersey, 1970.

[42] H. Schramm and J. Zowe, *A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results.*, SIAM J. Optim., 2 (1992), pp. 121–152.

[43] D. S. Scott, *Implementing Lanczos-like algorithms on hypercube architectures*, Computer Physics Communications, 53 (1989), pp. 271–281.

[44] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.

[45] M. J. Todd, K. C. Toh, and R. H. Tütüncü, *On the Nesterov-Todd direction in semidefinite programming*, SIAM J. Optim., 8 (1998), pp. 769–796.

[46] L. Vandenberghe and S. Boyd, *A primal–dual potential reduction method for problems involving matrix inequalities*, Mathematical Programming, Series B, 69 (1995), pp. 205–236.

[47] S. Zhou and H. Dai, *The block Chebyshev-Lanczos method for solving large symmetric eigenvalue problems*, J. Nanjing Aeronaut. Inst., 21 (1989), pp. 22–28.