C. HELMBERG[1]

Numerical Evaluation of SBmethod

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

ZIB-Report 01-37 (December 2001)

Numerical Evaluation of SBmethod

Christoph Helmberg*

February 2001, revised December 2001

Abstract

We report numerical results for `SBmethod` — a publically available implementation of the spectral bundle method — applied to the 7th DIMACS challenge test sets that are semidefinite relaxations of combinatorial optimization problems. The performance of the code is heavily influenced by parameters that control bundle update and eigenvalue computation. Unfortunately, no mathematically sound guidelines for setting them are known. Based on our experience with `SBmethod`, we propose heuristics for dynamically updating the parameters as well as a heuristc for improving the starting point. These are now the default settings of `SBmethod` Version 1.1. We compare their performance on the DIMACS instances to our previous best choices for Version 1.0. `SBmethod` Version 1.1 is also part of the independent DIMACS benchmark by H. Mittelmann. Based on these results we try to analyze strengths and weaknesses of our approach in comparison to other codes for large scale semidefinite programming.

MSC 2000: 90C22; 90C06, 90-08
Keywords: semidefinite programming, large scale methods, computational, semidefinite relaxations

1 Introduction

The C++ program `SBmethod` is an implementation of the spectral bundle method of Helmberg and Rendl [2000]; Helmberg and Kiwiel [1999] (see Helmberg [2000a] for a self contained introduction) for large scale eigenvalue optimization problems of the form

$$\min_{y\in Y} \; a\,\lambda_{\max}(C-\mathcal{A}^T y)+b^T y. \tag{1}$$

The function $\lambda_{\max}(\cdot)$ denotes the maximum eigenvalue. With S_n denoting the set of symmetric matrices of order n, the given data is $C\in S_n$, $\mathcal{A}^T\colon \mathbb{R}^m\to S_n$, $b\in\mathbb{R}^m$, and $a\in\mathbb{R}$ with $a>0$. The matrix C is called the *cost matrix* and the linear operator $\mathcal{A}^T$ is defined by

$$\mathcal{A}^T y=\sum_{i=1}^m y_i A_i,$$

where the $A_i\in S_n$ for $i=1,\dots,m$ are given symmetric matrices of order n. The matrices C and A_i should be well structured. Sparsity and various low rank structures are supported by the code; the code is publically available, see Helmberg [2000b].

The set $Y\subset\mathbb{R}^m$, over which is optimized, is a Cartesian product of real numbers $\mathbb{R}$, nonnegative real numbers $\mathbb{R}_+=\{x\in\mathbb{R}: x\ge 0\}$, and nonpositive real numbers $\mathbb{R}_-=\{x\in\mathbb{R}: x\le 0\}$, *i.e.*, there is a partition $(J_=, J_\ge, J_\le)$ of the index set $\{1,\dots,m\}$ with

$$Y=\{y\in\mathbb{R}^m:\;\; y_i\ge 0 \text{ for } i\in J_\ge,\; y_i\le 0 \text{ for } i\in J_\le\}. \tag{2}$$

*Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustraße 7, D-14158 Berlin, `helmberg@zib.de`, `http://www.zib.de/helmberg`

The somewhat peculiar choice of Y was motivated by the following connection to semidefinite programs with constant trace $a > 0$. The eigenvalue optimization problem (1) is equivalent to the dual of the semidefinite program

$$\begin{array}{ll} \max & \langle C, X\rangle \\ \text{s.t.} & \langle I, X\rangle = a \\ & \langle A_i, X\rangle = b_i \quad i \in J_= \\ & \langle A_i, X\rangle \le b_i \quad i \in J_\ge \\ & \langle A_i, X\rangle \ge b_i \quad i \in J_\le \\ & X \succeq 0, \end{array} \tag{3}$$

where $\langle A, B\rangle = \sum_{i,j} a_{ij} b_{ij}$ denotes the canonical inner product for matrices $A, B \in \mathbb{R}^{m\times n}$, and $X \succeq 0$ is short for X being positive semidefinite. Alternatively, we write $X \in S_n^+$ with S_n^+ the set of symmetric positive semidefinite matrices of order n. If the equality constraints for $i \in J_=$ imply $\langle I, X\rangle = a$ then the equivalence of (1) to the dual of (3) is true without the constraint $\langle I, X\rangle = a$. Note, that in this setting the dual of (3) always has strictly feasible solutions and, thus, strong duality holds.

Semidefinite programs of type (3) arise frequently in semidefinite relaxations of combinatorial optimization problems (see, *e.g.*, the survey of Goemans [1997]). In this context, any feasible solution of (1) yields an upper bound on the combinatorial optimization problem. The desire to compute such bounds quickly via approximate solutions to (1) for well structured matrices A_i was the main motivation for writing this software.

We briefly review other algorithms designed for large scale semidefinite programming. Benson, Ye, and Zhang [2000] propose a dual potential reduction algorithm (DSDP in Mittelmann [2001]) that is able to exploit the sparsity of the slack matrix by means of a sparse factorization. Its advantages are polynomial convergence and a reliable stopping criterion. The system matrix for computing the Newton step, however, is in general a dense positive definite matrix order m. This restricts applicability to problems with rather small m that have dual slack matrices with sparse factors. Burer, Monteiro, and Zhang [2001]; Burer, Monteiro, and Zhang [1999] (BMZ in Mittelmann [2001]) reformulate the dual as a nonconvex quadratic program with m variables. The computation of the search direction requires a factorization of the dual slack matrix. It is applicable to large m and proves to be very efficient for small to medium sized matrix variables. For large matrices, fill-in in the factorization may cause high memory consumption and computation times. Burer and Monteiro [2001] (BMPR in Mittelmann [2001]) propose a primal heuristic for semidefinite relaxations of binary quadratic programs with equality constraints. It is based on a nonconvex quadratic program that corresponds to optimizing over matrix variables of bounded rank. So far, their results are stunning. In our context it is important that their objective value is — if the iterate is feasible — a primal bound on the primal optimal solution of the relaxation but not necessarily a bound for the underlying combinatorial optimization problem.

In comparison to these methods the characteristics of the spectral bundle method are: Its memory consumption is of the same order as the input data; there is no need for factorization; it is a dual subgradient type method that shows good initial progress but also a significant tailing off effect as the optimal value is approached. We believe that it is well suited for large scale problems where both, the order of the matrix variable and the number m of design variables may be large and where accuracy requirements are moderate.

The performance of `SBmethod` heavily depends on the choice of several parameters. At this point of development no mathematically sound guidelines are known for choosing these parameters automatically. In this text we report, for the 7^{th} DIMACS challenge instances that are positive semidefinite relaxations of combinatorial optimization problems, the best parameter settings that we found in several test runs. We do this for two reasons. First, we want to show the potential and explore the limits of our implementation, and second, we hope that these examples help to establish guidelines for choosing these parameter for other problem classes, as well. Therefore, we try to provide intuitive explanations on why these parameters might be reasonable. We have tried to implement heuristics that follow these guidelines in Version 1.1 of `SBmethod` and compare the results to our best settings. It is not surprising that for several instances the default choices need about twice the computation time of their tuned counterparts, but for a few instances the new default choices turn out to be better.

The paper consists of two main parts. In the first part we give a concise introduction to the basic steps of the algorithm and its parameters and describe the new heuristics. The second part reports computational results. In particular we will explain the exact formulation and parameter settings we used. In addition, the DIMACS benchmark of Mittelmann [2001] provides numerical results in a comparable setting for the codes BMPR, BMZ, DSDP, and BUNDLE (BUNDLE is `SBmethod` Version 1.1 with the same default settings). Based on this benchmark and our own experiments we try analyze the strengths and weaknesses of `SBmethod`. In Section 4 we offer some conclusions. For the convenience of the reader we have collected our notation in the appendix.

2 The Algorithm

We extend the objective function of (1) to $\mathbb{R}^m$,

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

where $\imath_Y$ denotes the indicator function whose value is 0 for $y \in Y$ and ∞ otherwise. Instead of minimizing f directly we construct sequences of simpler cutting surface models of f that are easier to minimize. These model functions are formed by minorizing $a\lambda_{\max}(C - \mathcal{A}^T\cdot)$ and $\imath_Y$ as explained in the following two steps.

In the first step, we form a minorant for $a\,\lambda_{\max}(C - \mathcal{A}^T\cdot)$. It is well known (see, *e.g.*, Lewis and Overton [1996]) that, for $a \geq 0$,

$$a\lambda_{\max}(C - \mathcal{A}^T y) = \max\{\langle C - \mathcal{A}^T y, W\rangle : \operatorname{tr} W = a, W \succeq 0\}.$$

Therefore, any $W \in \mathcal{W} := \{W \succeq 0 : \operatorname{tr} W = a\}$ gives rise to a linear function $\langle C - \mathcal{A}^T\cdot, W\rangle$ that minorizes the function $a\lambda_{\max}(C - \mathcal{A}^T\cdot)$. Maximizing over a subset $\widehat{\mathcal{W}} \subset \mathcal{W}$ yields, as the pointwise maximum of the linear functions, a convex function minorizing $a\lambda_{\max}(C - \mathcal{A}^T\cdot)$. `SBmethod` uses a subset of the form[1]

$$\widehat{\mathcal{W}} := \{PVP^T + \alpha\overline{W} : \operatorname{tr} V + \alpha = a, V \succeq 0, \alpha \geq 0\}, \tag{4}$$

where P is an orthonormal matrix of size $n \times r$ and $\overline{W} \in S_n^+$ is a positive semidefinite matrix of trace 1. We refer to P as the *bundle*, to the number of columns r of P as the *size* of the bundle, and to $\overline{W}$ as the *aggregate*; within a single iteration of the algorithm, P and $\overline{W}$ may be regarded as constant but both will be updated at the end of each iteration.

In the second step, we minorize $\imath_Y$ by a linear function $-\eta^T y$, where η is any element of the dual cone Y^* to Y, *i.e.*,

$$\begin{aligned} \eta \in Y^* &= \{\eta \in \mathbb{R}^m : \eta^T y \geq 0 \;\; \forall y \in Y\} \\ &= \{\eta \in \mathbb{R}^m : \eta_i = 0 \;\; \forall i \in J_=, \;\; \eta_i \geq 0 \;\; \forall i \in J_\geq, \;\; \eta_i \leq 0 \;\; \forall i \in J_\leq\}. \end{aligned}$$

Combining both steps, we obtain for any fixed $W \in \mathcal{W}$ and $\eta \in Y^*$ a linear minorant of f,

$$f_{W,\eta}(y) := \langle C - \mathcal{A}^T y, W\rangle + (b - \eta)^T y \;\; \leq f(y) \qquad \forall y \in \mathbb{R}^m. \tag{5}$$

A subset $\widehat{\mathcal{W}} \subset \mathcal{W}$ together with a fixed $\eta \in Y^*$ yields a convex minorant of f,

$$f_{\widehat{\mathcal{W}},\eta}(y) := \max_{W \in \widehat{\mathcal{W}}} \langle C - \mathcal{A}^T y, W\rangle + (b - \eta)^T y \;\; \leq f(y) \qquad \forall y \in \mathbb{R}^m.$$

[1]For promising numerical experiments with more general models, see, *e.g.*, Nayakkankuppam [1999].

Since we will form $\widehat{\mathcal{W}}$ from accumulated local information, the model $f_{\widehat{\mathcal{W}},\eta}$ can be expected to be only locally of reasonable quality. Therefore we determine the next candidate as the minimizer of

$$\min_{y\in\mathbb{R}^m}\max_{\eta\in Y^*} f_{\widehat{\mathcal{W}},\eta}(y)+\frac{u}{2}\|y-\hat{y}\|^2, \tag{6}$$

where $\hat{y}$ is the current *center of stability* (in the algorithm it is the starting point or the last successful iterate) and the *weight* u is a parameter that allows some indirect influence on the distance of the minimizer of (6) to $\hat{y}$.

Instead of solving (6) directly, we solve its dual,

$$\max_{W\in\widehat{\mathcal{W}},\eta\in Y^*}\ \min_{y\in\mathbb{R}^m}\langle C-\mathcal{A}^T y, W\rangle+(b-\eta)^T y+\frac{u}{2}\|y-\hat{y}\|^2. \tag{7}$$

The inner minimization over y is an unconstrained convex quadratic problem and can be solved explicitly for any choice of W and η,

$$y_{\min}(W,\eta):=\hat{y}+\frac{1}{u}(\mathcal{A}W-b+\eta). \tag{8}$$

Substituting this for y into (7) we obtain

$$\max_{W\in\widehat{\mathcal{W}},\eta\in Y^*}\ \langle C-\mathcal{A}^T\hat{y}, W\rangle+(b-\eta)^T\hat{y}-\frac{1}{2u}\|\mathcal{A}W-b+\eta\|^2. \tag{9}$$

In order to further reduce the complexity of solving this subproblem, we solve it by a sequence of coordinatewise optimization steps. In particular, we first fix $\hat{\eta}$ and solve (9) over $W\in\widehat{\mathcal{W}}$ yielding an optimal W^+, then we fix W^+ and solve (9) over $\eta\in Y^*$ and iterate this if necessary. Optimizing over $W\in\widehat{\mathcal{W}}$ for fixed $\hat{\eta}$ results in a small dimensional ("small" depending on the bundle size r) convex quadratic semidefinite programming problem in V and α (see the definition of $\widehat{\mathcal{W}}$ (4)) that can be solved efficiently by interior point methods. This yields

$$W^+\in\operatorname*{Argmax}_{W\in\widehat{\mathcal{W}}}\langle C-\mathcal{A}^T\hat{y}, W\rangle+(b-\hat{\eta})^T\hat{y}-\frac{1}{2u}\|\mathcal{A}W-b+\hat{\eta}\|^2. \tag{10}$$

See Helmberg and Rendl [2000]; Helmberg and Kiwiel [1999] for a description of the interior point code. Optimizing (9) over η for some fixed W is separable convex; the optimal argument $\eta_{\max}(W)$ is determined by

$$[\eta_{\max}(W)]_i=\begin{cases}\max\left\{0,-u\left[\hat{y}+\frac{1}{u}(\mathcal{A}W-b)\right]_i\right\} & i\in J_{\geq}\\ \min\left\{0,-u\left[\hat{y}+\frac{1}{u}(\mathcal{A}W-b)\right]_i\right\} & i\in J_{\leq}\\ 0 & i\in J_{=}.\end{cases} \tag{11}$$

Therfore, for W^+ of (10), we set

$$\eta^+:=\eta_{\max}(W^+). \tag{12}$$

Observe that η^+ also ensures feasibility and complementarity of $y_{\min}(W^+,\eta^+)$, see (8),

$$y^+:=y_{\min}(W^+,\eta^+)\in Y \qquad\text{and}\qquad (\eta^+)^T y^+=0. \tag{13}$$

The pair (W^+,η^+) is, in general, not an optimal solution of (9); it is only an approximation. We regard this approximation as *not* sufficiently accurate if

$$f_{\widehat{\mathcal{W}},\eta^+}(y^+)-f_{W^+,\eta^+}(y^+)>\kappa_M[f(\hat{y}^k)-f_{W^+,\eta^+}(y^+)] \tag{14}$$

for a *model-precision parameter* $\kappa_M\in(0,\infty)$, *i.e.*, if the gap at y^+ between the model value $f_{\widehat{\mathcal{W}},\eta^+}(y^+)$ and its linear minorant $f_{W^+,\eta^+}(y^+)$ is too big in comparison to the gap between the old function value $f(\hat{y}^k)$ and the linear minorant. In this case, the two coordinatewise steps are repeated, now fixing $\hat{\eta}$ to the new η^+.

If the solution of the model is considered sufficiently accurate, then y^+ is the new candidate at which the function is evaluated. We do this by a Lanczos method (see also Section 2.1) which generates a sequence of normalized vectors v whose *Ritz-values* $v^T(C - \mathcal{A}^T y^+)v$ yield successively better estimates of $\lambda_{\max}(C - \mathcal{A}^T y^+)$ until either the function value turns out to be too high for sufficient decrease in objective value or the vectors have converged to an eigenvector of $\lambda_{\max}$. More formally, the Lanczos process continues to generate better and better $W_S := avv^T \in \mathcal{W}$ until

(a) $f(\hat{y}) - f_{W_S,\eta^+}(y^+) \le \bar{\kappa}\left[f(\hat{y}) - f_{W^+,\eta^+}(y^+)\right]$, or

(b) $f_{W_S,\eta^+}(y^+) = f(y^+)$ and $f(\hat{y}) - f(y^+) \ge \kappa\left[f(\hat{y}) - f_{W^+,\eta^+}(y^+)\right]$.

In this test, the *descent parameter* $\kappa \in (0,1)$ controls the necessary progress relative to the gap $f(\hat{y}) - f_{W^+,\eta^+}(y^+)$ for accepting *descent steps* to y^+, and the *null step parameter* $\bar{\kappa} \in [\kappa, 1)$ controls the level for accepting a *null step*. At a descent step the algorithm moves the center of stability $\hat{y}$ to y^+. At a null step, $\hat{y}$ remains unchanged. In both cases the model $\widehat{\mathcal{W}}$ is updated so that W_S and W^+ are both contained in $\widehat{\mathcal{W}}^+$ and the algorithm iterates.

The algorithm stops as soon as

$$f(\hat{y}) - f_{W^+,\eta^+}(y^+) \le \varepsilon(|f(\hat{y})| + 1) \tag{15}$$

for a *termination precision parameter* $\varepsilon > 0$, *i.e.*, when the maximal progress of the next step ($f_{W^+,\eta^+}(y^+)$ may be regarded as a lower bound for this) is small in comparison to the absolute value of the function.

We now give the algorithm in detail, superscripts k are used to indicate iteration indices.

Algorithm 2.1 *(Spectral Bundle Method with Bounds)*
Input: $y^0 \in \mathbb{R}^n$, $\varepsilon > 0$, $\kappa \in (0,1)$, $\bar{\kappa} \in [\kappa, 1)$, $\kappa_M \in (0, \infty]$, $\kappa_\eta \in [0,1]$, *a weight* $u^0 > 0$.

```
1. Set $k = 0$, $\hat{y}^0 = y^0$, $\eta^0 = 0$, compute $f(y^0)$ and $\widehat{\mathcal{W}}^0$, choose $W^0 \in \widehat{\mathcal{W}}^0$.
2. (Trial point finding). Compute $\hat{\eta} = (1 - \kappa_\eta)\eta^k + \kappa_\eta \eta^k_{\max}(W^k)$, see (11).
    (a) Find $W^+$, $\eta^+$, $y^+$ by (10), (12), and (13) in this sequence.
    (b) (Stopping criterion). If $f(\hat{y}^k) - f_{W^+,\eta^+}(y^+) \le \varepsilon(|f(\hat{y}^k)| + 1)$ then **stop**.
    (c) If $f_{\widehat{\mathcal{W}}^k,\eta^+}(y^+) - f_{W^+,\eta^+}(y^+) > \kappa_M[f(\hat{y}^k) - f_{W^+,\eta^+}(y^+)]$ then set $\hat{\eta} = \eta^+$ and **goto** (a).
    (d) Set $y^{k+1} = y^+$, $W^{k+1} = W^+$, $\eta^{k+1} = \eta^+$.
3. (Descent test). Find $W_S^{k+1} \in \mathcal{W}$ such that either
    (a) $f(\hat{y}^k) - f_{W_S^{k+1},\eta^{k+1}}(y^{k+1}) \le \bar{\kappa}\left[f(\hat{y}^k) - f_{W^{k+1},\eta^{k+1}}(y^{k+1})\right]$, or
    (b) $f_{W_S^{k+1},\eta^{k+1}}(y^{k+1}) = f(y^{k+1})$ and $f(\hat{y}^k) - f(y^{k+1}) \ge \kappa\left[f(\hat{y}^k) - f_{W^{k+1},\eta^{k+1}}(y^{k+1})\right]$.
    In case (a), set $\hat{y}^{k+1} = \hat{y}^k$ (null step), otherwise set $\hat{y}^{k+1} = y^{k+1}$ (descent step).
4. (Weight updating). Choose a new weight $u^{k+1}$ (see Kiwiel [1990]; Helmberg and Kiwiel [1999]).
5. (Model updating). Choose a $\widehat{\mathcal{W}}^{k+1} \supset \{W^{k+1}, W_S^{k+1}\}$ of the form (4).
6. Increase $k$ by one and **goto** 2.
```

The following theorem holds for $\varepsilon = 0$.

Theorem 2.2 *Helmberg and Kiwiel [1999]*
Either $\hat{y}^k \to \bar{y} \in \operatorname{Argmin} f$, *or* $\operatorname{Argmin} f = \emptyset$ *and* $\|\hat{y}^k\| \to \infty$. *In both cases* $f(\hat{y}^k) \downarrow \inf f$.

For $\varepsilon > 0$ the stopping criterion (15) does not guarantee that at termination the desired relative precision is achieved, because $f_{W^+,\eta^+}(y^+)$ is not necessarily a lower bound on $\inf f$ (in contrast, interior point approaches like Benson, Ye, and Zhang [2000] provide a primal feasible objective value). However, the linear function $f_{W^+,\eta^+}(\cdot) = \langle C, W^+\rangle + \langle b - \eta^+ - \mathcal{A}W^+, \cdot\rangle$ (see (5)) is a global minorant of f. Thus, a small norm $\|b - \eta^+ - \mathcal{A}W^+\|$ (*i.e.*, W^+ is almost primal feasible) ensures that there is no significantly better solution within an appropriate ball around y^+. In the tables of §3 we list this norm in column $\|\nabla \bar{f}_+\|$ (in Mittelmann [2001] these numbers are given under the unfortunate caption "Error Measures (Bundle)").

The efficiency of the algorithm is governed by the parameters controlling the eigenvalue computation in the Lanczos method and the parameters determining the size and composition of the bundle in $\widehat{\mathcal{W}}$. Therefore we will describe these two aspects briefly in the following and propose first heuristics for determining them dynamically. Next, a heuristic is proposed for improving the starting point by a few steepest descent steps. The section is concluded by a summary of the computational cost and memory requirements of the steps of Algorithm 2.1.

2.1 Eigenvalue Computation

We implement step 3 of Algorithm 2.1 by computing, by means of the Lanczos method, an eigenvector v to the maximum eigenvalue of the matrix $C - \mathcal{A}^T y^k$ and by setting $W_S^k = avv^T$. The performance of the method is heavily influenced by parameters n_L and n_C, whose meaning we explain in the following.

The Lanczos process is an iterative method that generates, from a series of matrix vector multiplications, a partial tridiagonalization of the matrix. Each iteration, called a ***Lanczos step***, increases the order of the tridiagonal matrix by one. The eigenvectors and eigenvalues of the tridiagonal matrix are used to generate approximations to the eigenvectors and eigenvalues of the original matrix; slightly deviating from the usual terminology, we call these approximations *Lanczos vectors* and their *Ritz values*. See Golub and van Loan [1989] for an introduction and Parlett [1998]; Saad [1992] for a detailed description of the Lanczos method.

Our algorithm restarts the Lanczos process after n_L Lanczos steps using the Lanczos vector corresponding to the maximum eigenvalue of the tridiagonal matrix as new starting vector. Before each restart we check the null step criterion of step 3(a) of Algorithm 2.1. Within each restart-phase we use complete orthogonalization, *i.e.*, the new vector is orthogonalized with respect to all vectors generated since the last restart.

A large parameter n_L enhances the convergence of the Lanczos process but also increases computation time significantly (as well as memory consumption) because of the complete orthogonalization of the vectors. If the matrix vector multiplication is computationally expensive, then we prefer to choose a large parameter n_L, say 150.

The speed of convergence of the Lanczos process is also governed by the spectral separation, *i.e.*, the size of the gap between largest and second largest eigenvalue (multiplicities may be greater than one) relative to the spread of the entire spectrum. If the matrix vector multiplication is very cheap (extremely sparse matrices), then it may be worth to apply at each Lanczos step a spectral transformation to the matrix in order to increase the spectral separation. The code offers the possibility to transform the spectrum by applying a Chebychev polynomial to the matrix via a series of matrix vector multiplications. The parameter n_C allows to specify the degree of the Chebychev polynomial (one uses only odd degrees so as to preserve the ordering of the eigenvalues outside the Chebychev interval). In our experience this parameter should by either 0 (to be read as no spectral transformation) or at least 21. If $n_C > 0$, each restart of the Lanczos process requires roughly $n_L \cdot n_C$ matrix vector multiplications. Therefore, we recommend to choose n_L relatively small in this case, *e.g.*, between 20 and 50.

To the best of our knowledge, no satisfactory heuristics are known for choosing the parameters n_L and n_C in an automatic way.

In `SBmethod`, Version 1.0, the default heuristic assumes matrices to be very sparse and starts with computing a guess of the interval $[\lambda_{\max}, \lambda_{\min}]$ by ten Lanczos steps for block size two (the interval is needed to set up the Chebychev polynomial; starting vectors are generated randomly or from previous vectors). It then uses block size one and begins with $n_L = 15$ and $n_C = 20$ so as to quickly eliminate poor candidates y^{k+1} at the end of this restart cycle by criterion 3(a) of Algorithm 2.1. After this second restart it continues with $n_L = 20$ and $n_C = \min\{30+20\cdot\lfloor\#\text{restarts}/5\rfloor, 200\}$. The heuristic has the advantage that memory consumption is relatively small but it is a disaster if the matrix vector multiplication is expensive.

Therefore, in `SBmethod`, Version 1.1, we try to discern between expensive and cheap matrix vector operations. The matrix classes of `SBmethod` allow to determine a rough estimate of the flops of one matrix vector multiplication. With complete orthogonalization, Lanczos step h requires roughly $O(hn)$ flops. Without much conviction we fix the following rule: Assuming $n_C = 0$ we would expect $h = 50$ Lanczos steps to be reasonable for $n < 300$, $h = 100$ for $n < 1000$, and $h = 200$ for $n \geq 1000$. If the estimated number of flops of a matrix vector multiplication exceeds $h \cdot n$ then we consider the matrix vector multiplication sufficiently expensive, set $n_C = 0$ and $n_L = \min\{50\cdot\lfloor\#\text{restarts}/2+1\rfloor, 200\}$. The motivation for increasing n_L by 50 every second restart is again the hope to quickly eliminate poor candidates at the end of the first restarts by criterion 3(a) of Algorithm 2.1 while enhancing convergence for promising candidates. Otherwise, we decide in favor of $n_C > 0$ and compute a guess of the interval $[\lambda_{\max}, \lambda_{\min}]$ by ten Lanczos steps for block size two. Afterwards we use block size one and begin with $n_L = 15$ and $n_C = 20$, as in Version 1.0. After this second restart we continue with $n_L = 25$ and $n_C = \min\{20 + 10 \cdot \lfloor\#\text{restarts}/2\rfloor, 200\}$.

2.2 Model Updating

The model $\widehat{\mathcal{W}}$ of (4) is completely determined by the choice of the bundle P and the aggregate $\overline{W}$. In updating P and $\overline{W}$ to P^+ and $\overline{W}^+$ we have to ensure that the new model $\widehat{\mathcal{W}}^+$ contains $\{W^+, W_S = avv^T\}$ (v denotes the new Lanczos vector), see step 5 of Algorithm 2.1. In `SBmethod`, the construction of the new bundle P^+ from P and the new information obtained from Lanczos vectors is controlled by four parameters $n_K, n_{\min} \in \mathbb{N}_0$, $n_K \geq n_{\min}$ (maximum and minimum subspace dimension to keep), $n_A \in \mathbb{N}$ (maximum number of Lanczos vectors to add), and $t_a \in (0,1)$ (aggregation tolerance). These are used as follows.

Let $W^+ = PV^+P^T + \alpha^+\overline{W}$ denote the computed solution of (10) and let $V^+ = Q\Lambda Q^T$ be an eigenvalue decomposition of V^+ with $Q^TQ = I_r$ and $\Lambda = \text{Diag}(\lambda_1, \ldots, \lambda_r)$ a diagonal matrix with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r$. For

$$r_1 = \begin{cases} 0 & \text{if } n_K = 0 \\ \max\{i \in \{1, \ldots, n_K\} : \lambda_i \geq t_a\lambda_1\} \cup \{n_{\min}\} & \text{otherwise,} \end{cases}$$

let Q_1 contain the r_1 first columns of Q and Q_2 the last $r - r_1$ columns and let Λ_1 and Λ_2 denote the corresponding diagonal matrices. Furthermore, let n_L denote the number of Lanczos vectors returned by the eigenvalue computation routine and let L be the matrix formed by the $\min\{n_L, n_A\}$ Lanczos vectors with largest Ritz-values. Then the new model $\widehat{\mathcal{W}}^+$ is determined by

$$\begin{aligned} P^+ &= \text{orth}([PQ_1, L]) \\ \overline{W}^+ &= \frac{PQ_2\Lambda_2(PQ_2)^T + \alpha^+\overline{W}}{\text{tr}\,\Lambda_2 + \alpha^+}, \end{aligned} \tag{16}$$

where orth() constructs a matrix whose columns form an orthonormal basis of the space spanned by the columns of the argument. This construction ensures $\widehat{\mathcal{W}}^+ \supset \{W^+, W_S\}$ (see Helmberg and Rendl [2000]).

Remark 2.3 *As shown in Helmberg and Rendl [2000] it is not necessary to store and update $\overline{W}$ itself, but only $\mathcal{A}\overline{W}$ and $\langle C, \overline{W}\rangle$. Note, however, that the matrix W^+ may be interpreted as an approximate primal solution X (and the η variables as primal slack variables). If full knowledge about W^+ is desired, then $\overline{W}$ should be stored explicitly and `SBmethod` offers this possibility. In Helmberg [2001] we give an explicit proof for the convergence of W^+ to a primal optimal solution (under a standard regularity assumption) and exploit this property to set up a cutting plane approach for constrained quadratic 0-1 programming.*

Again, no reasonable guidelines are known for choosing these parameters.

In SBmethod, Version 1.0, the parameters n_K, n_A, $n_{\min}$, and t_a have to be specified by the user and are employed as described above.

In our experiments, an expensive model lead to fewer iterations and faster progress only if α — the contribution of the aggregate matrix $\overline{W}$ to W^+ — remained small. In particular, if α did not increase and the time for solving the model was small in comparison to the time spent in the Lanczos code, then increasing $n_{\min}$ to reduce the number of iterations and function evaluations seemed a good choice. If, however, the cost of solving the model exceeded the Lanczos computation significantly, then setting $n_K = 0$ usually lead to faster computation times. We have no idea how to explain these observations mathematically (an intuitive attempt might read, that a model that is of high quality along a narrow but promising subspace may produce candidates that avoid this promising subspace). In order to meet repeated requests for automatic parameter choices we have, with some reluctance, implemented a heuristic in SBmethod, Version 1.1, that mimics this behavior.

The heuristic of SBmethod, Version 1.1, adapts the parameters n_K, n_A, and $n_{\min}$ dynamically (in this case the corresponding input parameters are understood as upper bounds on the possible values). It starts with $n_{\min} = 0$. In each iteration it adds up to n_A new vectors by the following criterion. Let $\lambda_1 \geq \ldots \geq \lambda_{n_L}$ denote the Ritz values of the new Lanczos vectors. The vectors with largest Ritz values are added if $i \leq n_A$ and, for $i > 5$, if $\lambda_i > \underline{\lambda}$ for a lower bound $\underline{\lambda}$. This $\underline{\lambda}$ is determined as follows. Let $\mu = \max\{v^T(C - \mathcal{A}^T y^+)v : v = j\text{-th column of } PQ, j \in \{1, \ldots, r\}\}$ denote the maximum Ritz value of the columns of PQ (if these have not been computed explicitly we use the value $\mu = \langle C - \mathcal{A}^T y^+, W^+ \rangle / a$). Then $\underline{\lambda} = \lambda_1 - \max\{10^{-4}\lambda_1, 5 \cdot (\lambda_1 - \mu)\}$. P is then updated as explained above. As long as the accumulated time required for solving the quadratic semidefinite model is less than half the time spent in the eigenvalue routine, $n_{\min}$ is increased in each iteration by one; the latter rule is intended for situations where t_a is too restrictive. If, however, an iteration is encountered where $\alpha > \frac{3}{10}a$ and solving the augmented model takes ten times as long as computing the eigenvalues, then evidence is high that a sufficiently large bundle would be inefficient. In this case, n_K is set to zero and n_A to seven for all further iterations.

As many decisions in this heuristic lack mathematical justification, we do not expect it to deliver excellent choices for the bundle parameters; furthermore, its dependence on time accounting information has the undesirable effect that running the same algorithm with the same input twice may result in different output (even though convergence to an [existing] optimal solution is guaranteed). Yet, we hope that it performs reasonably well if no better setting from related problems is at hand.

2.3 Starting Point

For the feasible set Y (see (2)) it is easy to choose a feasible starting point y^0 (SBmethod uses $y^0 = 0$ by default), but this point may well be far away from the optimal solution to the effect, that the maximum eigenvalue is simple and well separated from the remaining eigenvalues. The corresponding unique eigenvector may then turn out to be the only relevant information for the model for several consecutive iterations. In this case, the method takes a sequence of steepest descent steps. Besides the fact, that the steepest descent direction is easily computed without solving the quadratic model, this behavior may also cause serious difficulties with the dynamic update rule for the weight u (see Step 4 of Algorithm 2.1). Indeed, in several applications the gradient changed only slightly along steepest descent directions. In consequence, the weight u was successively decreased to allow for larger and larger steps. When finally the eigenvector to the second eigenvalue became important, u was far too small and the update heuristics did not succeed well in increasing it again. The following *starting point heuristic*, now implemented in Version 1.1, tries to skip this initial phase by explicitly performing up to five steepest descent steps.

The heuristic starts by checking the gap $\Delta = \lambda_{\max}(C - \mathcal{A}^T y^0) - \lambda_2(C - \mathcal{A}^T y^0)$ between the maximum eigenvalue $\lambda_{\max}(C - \mathcal{A}^T y^0)$ and the second eigenvalue $\lambda_2(C - \mathcal{A}^T y^0)$ (here, λ_2 may be equal to $\lambda_{\max}$ if the eigenspace of the maximum eigenvalue is larger than one). If $a\Delta >$

$0.01 \cdot (|f(y^0)|+1)$ (the difference in objective values for the two choices of λ is large relative to the objective value), then a steepest descent step is made. In particular, let v denote the eigenvector to the maximum eigenvalue $\lambda_{\max}(C-\mathcal{A}^T y^0)$, then the steepest descent direction is $dy = \mathcal{A}(avv^T) - b$. We set all coordinates of dy to zero that do not point into directions of recession of Y; this yields a direction $\overline{dy}$. In order to obtain a safe guess for the step length $t \geq 0$ in direction $\overline{dy}$, we underestimate the decrease of $\lambda_{\max}$ and overestimate the increase in λ_2 along this direction by linear approximations and determine t as the point of intersection of both as follows.

For $\lambda_{\max}$ it is computationally more convenient to express this in terms of the objective value. The objective value decreases at most by $t\,\langle b-\mathcal{A}^T(avv^t), \overline{dy}\rangle$, because $b-\mathcal{A}^T(avv^t)$ is a subgradient of f at y^0,

$$f(y^0+t\,\overline{dy}) \geq a\lambda_{\max}(C-\mathcal{A}^T y^0) + \langle b, y^0\rangle + t\,\langle b-\mathcal{A}^T(avv^t), \overline{dy}\rangle \qquad \text{for } t \geq 0. \tag{17}$$

Since

$$\lambda_i(C-\mathcal{A}^T(y^0+t\,\overline{dy})) \leq \lambda_i(C-\mathcal{A}^T y^0) + t\lambda_{\max}(-\mathcal{A}^T\overline{dy}) \qquad \text{for } i=1,\ldots,n \text{ and } t \geq 0, \tag{18}$$

the increase of $\lambda_2(C-\mathcal{A}^T(y^0+t\overline{dy}))$ for $t \geq 0$ can be bounded using $\bar{\lambda} = \lambda_{\max}(-\mathcal{A}^T\overline{dy})$,

$$a\lambda_2(C-\mathcal{A}^T(y^0+t\,\overline{dy})) + \langle b, y^0+t\,\overline{dy}\rangle \leq a\lambda_2(C-\mathcal{A}^T y^0) + \langle b, y^0\rangle + t[a\bar{\lambda} + \langle b, \overline{dy}\rangle] \text{ for } t \geq 0. \tag{19}$$

By (18), eq. (19) is valid for λ_2 exchanged with $\lambda_{\max}$, as well. Thus, if $a\bar{\lambda} + \langle b, \overline{dy}\rangle < 0$ then the objective value decreases in this direction infinitely; the problem is unbounded from below and the algorithm terminates. Otherwise, the first step length t is computed by equating the right hand sides of (17) and (19), $t = a\Delta/[\langle b-\mathcal{A}^T(avv^t), \overline{dy}\rangle + a\bar{\lambda} + \langle b, \overline{dy}\rangle]$. If, for the new point $y^+ = y^0 + t\,\overline{dy}$, the decrease in objective value is less than 1% of the decrease predicted by the linear model then we stop the process and start with y^0. Otherwise we set $y^0 = y^+$, recompute Δ, check the size of the gap, and continue as before except that, instead of $\bar{\lambda}$, we use linear interpolation of the last two λ_2 values to model the increase in λ_2 (or zero if linear interpolation predicts a decrease). We repeat this process at most five times.

2.4 Computational Cost and Memory Requirements

We proceed along the steps of Algorithm 2.1.

Step 1. Evaluating $f(y^0)$ (or running the starting point heuristic) is dominated by the computation of $\lambda_{\max}(C-\mathcal{A}^T y)$. Each restart of the Lanczos method requires $n_L \cdot n_C$ matrix vector multiplications. Independent of the number of restarts, the Lanczos method needs $O(n_L^2 + n_L \cdot n)$ memory. Unfortunately, there is no a priori bound on the number of restarts. In multiplying the Lanczos vector v with the matrix $C-\mathcal{A}^T y$, `SBmethod` tries to exploit the structure of the coefficient matrices C and the A_i. In particular, before calling the eigenvalue routine, the linear combination of all sparse coefficient matrices is collected in a new sparse matrix; for all structured coefficient matrices of the form AA^T or $AB^T + BA^T$ the matrix vector product is computed via $A(A^T v)$ or $A(B^T v) + B(A^T v)$ using the original description. The memory requirement for $C - \mathcal{A}^T y$ is therefore of the same order as the input.

Step 2. Substep (a) involves determining the cost coefficients for the quadratic semidefinite subproblem (10) in variables V and α (see (4)) and solving it by an interior point method. The procedure is described in detail in Helmberg and Rendl [2000]; Helmberg [2000a]; we cite the result. The cost coefficients require the computation of $P^T CP$ and $P^T A_i P$, $i = 1,\ldots,m$ — one at a time — and the accumulation of the quadratic cost matrix in $O(mr^4)$ time and $O(r^4)$ memory ($r \leq n_K + n_A$ is the number of columns in P). Each iteration of the interior point method is dominated by the factorization of the quadratic cost matrix in $O(r^6)$ time; in `SBmethod` the number of interior point iterations is upper bounded by 50 and is about 15 on average. Memory requirements are $O(r^4)$. Note that the interior point method does no longer depend on the A_i, in particular not on m. Substep (b) is neglectable. In Substep (c) the evaluation of $f_{\widehat{W}^k,\eta^+}(y^+)$ requires computing $\lambda_{\max}(P^T(C-\mathcal{A}^T y)P)$ (see Helmberg and Kiwiel [1999]) and is thus dominated by Substep (a). It is shown in Helmberg and Kiwiel [1999] that in the case of an inner iteration the quadratic cost matrix of the quadratic semidefinite subproblem need not be recomputed. Substep (d) is executed in $O(m)$ time and memory since W^{k+1} is not computed explicitly.

Step 3. The only expensive operation is finding W_S^{k+1} by the Lanczos method; the same considerations as in Step 1 apply.

Step 4. Roughly, the update is based on information about the subgradient $b - \mathcal{A}W^{k+1}$, which is available by Step 2. It requires $O(m)$ time and memory.

Step 5. Computing P^+ by (16) using Householder QR needs $O((n_K+n_A)^2 n)$ time and $O((n_K+n_A)n)$ memory (see Golub and van Loan [1989]). `SBmethod` stores and updates $\mathcal{A}\overline{W}$ and $\langle C, \overline{W}\rangle$ in $O(m)$ time and space; the values are available from Step 2.

Step 6 is irrelevant.

Summary. In the default settings of `SBmethod` the parameters n_L, n_C, n_K, n_A are bounded above by some reasonably small constants. Therefore each restart of the Lanczos method, each substep of Step 2, and steps 4 and 5 all require linear time and space (linear with respect to the encoding length of the input). The number of restarts of the Lanczos method and the number of inner and overall iterations of the spectral bundle method, however, are hard to quantify and highly problem dependent.

3 Numerical Examples

In the following we discuss our experience with `SBmethod` on the combinatorial optimization based instances of the DIMACS challenge set from

`http://dimacs.rutgers.edu/Challenges/Seventh/Instances/`

Unless stated otherwise, we employ for Version 1.0 the rule of Helmberg and Kiwiel [1999] to update the weight u, the heuristic of Version 1.0 of §2.1 for setting the Lanczos parameters n_L and n_C, and use the following default parameter values: $\varepsilon = 10^{-5}$, $n_K = 20$, $n_{\min} = 5$, $n_A = 5$, $t_a = 0.01$, $\bar{\kappa} = \kappa = 0.1$, $\kappa_M = 0.6$, $\kappa_\eta = 1$, $y^0 = 0$. In our experiments with Version 1.0, we typically tuned the parameters n_L, n_C, n_K, $n_{\min}$, and n_A.

In contrast, the results for Version 1.1 will all be given for the same hardwired default parameters: $\varepsilon = 10^{-5}$, $n_K = 45$, $n_{\min} = 30$, $n_A = 10$, $t_a = 0.01$, $\bar{\kappa} = \kappa = 0.1$, $\kappa_M = 0.6$, $\kappa_\eta = 1$, $y^0 = 0$, weight updating by the Helmberg and Kiwiel [1999]-rule, and all three heuristics switched on (the heuristic for Version 1.1 of §2.1 for choosing Lanczos pramaters n_L and n_C; the bundle update heuristic of §2.2 with upper bounds n_K, $n_{\min}$, n_A; the starting point heuristic of §2.3 for improving y^0). This setting is the code named BUNDLE in Mittelmann [2001].

The numerical results were computed on a Linux-PC with an Intel Pentium III 800 MHz processor (256 KB Cache) and 960 MB.

Tables will be given in the following format (see, *e.g.*, Table 1). The first four columns give the name of the problem, the order of the matrix variable n, the number of constraints m, and the value of the optimal solution f^* as listed on

`http://dimacs.rutgers.edu/Challenges/Seventh/Instances/tablestat.html`

Column f_{term} displays the function value $f(\hat{y}^k)$ at termination and *rel_acc* measures the relative precision of this value with respect to the optimal solution,

$$\text{rel\_acc} = \frac{f_{\text{term}} - f^*}{|f^*| + 1}.$$

Columns *time* and *λ_%* show the total CPU-time in hours:minutes:seconds and its percentage spent in the Lanczos code for computing maximum eigenvalues and Lanczos vectors. Column k lists the final iteration numbers. The last but one column (*desc.*) displays the number of descent steps. The final column ($\|\nabla \bar{f}_+\|$) gives the norm of the gradient of the linear minorizing model $f_{W^+,\eta^+}(y^+)$ at termination.

We also provide plots to illustrate the progress of the algorithm over time (see, *e.g.*, Figure 1). Note that time is given in logarithmic scale. For each descent step the objective value $f(y^{k+1})$ is shown as circle and the model value $f_{W^{k+1},\eta^{k+1}}(y^{k+1})$ is shown as dot. The objective value in the starting point y^0 is also plotted; in this point, the model is set to the objective value. For Version 1.1 the starting point refers to the point obtained after applying the starting point heuristic of §2.3.

In order to put our results into perspective with respect to other methods for large scale semidefinite programming, we make use of the data provided in Mittelmann [2001]. In particular, we will end each section with a paragraph labeled *DIMACS benchmark* where we compare `SBmethod` (BUNDLE of the benchmark is `SBmethod` Version 1.1 with the same default setting) to BMZ (Burer, Monteiro, and Zhang [2001]) and DSDP (Benson, Ye, and Zhang [2000]). BMZ, DSDP, and `SBmethod` all compute dual feasible solutions and thus valid bounds for the underlying combinatorial optimization problem. BMPR (Burer and Monteiro [2001]) is uniformly faster than `SBmethod`(except for the atypical Hamming instances; it should be faster for more representative instances of the Lovász ϑ-function). Since it computes a primal bound for the relaxation (if the resulting matrix is indeed feasible), its objective value is not guaranteed to yield a bound on the combinatorial problem. Thus, BMPR does not serve the same purposes as BMZ, DSDP, and `SBmethod`; a detailed comparison is therefore of little relevance.

3.1 The Torus Set

We use the graph representation by the sparse weighted adjacency matrix A of order n (with $\operatorname{diag}(A) = 0$) to generate the problem description in `SBmethod` format. For $C = \frac{1}{4}(\frac{e^TAe}{n}I - A)$ (e denotes the vector of all ones) the primal semidefinite relaxation and its dual eigenvalue formulation read

$$\begin{array}{ll} \max & \langle C, X\rangle \\ \text{s.t.} & \operatorname{diag}(X) = e \\ & X \succeq 0 \end{array} \qquad \min_{y\in\mathbb{R}^n} n\lambda_{\max}(C - \operatorname{Diag}(y)) + e^Ty.$$

In `SBmethod` we use the sparse matrix format for C and the class `SINGLETON` to represent the matrices $A_i = E_{ii}$ for $i = 1, \ldots, n$, where $E_{ii} \in S_n$ has a one in element ii and is zero otherwise.

Table 1: Max-cut on toroidal grid graphs (Version 1.0)

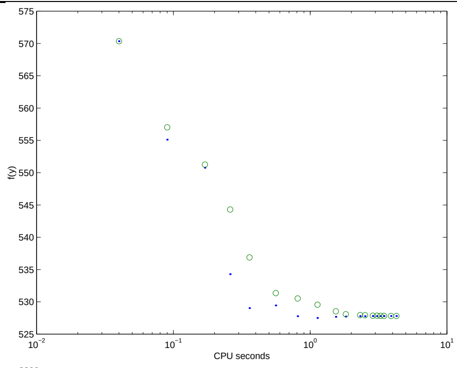
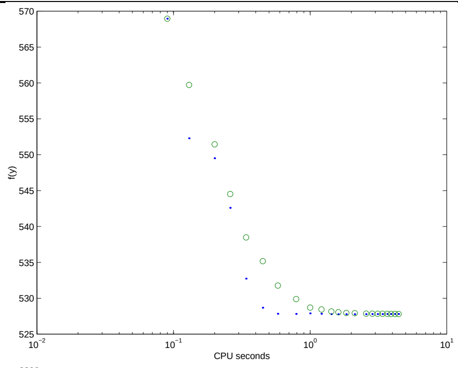
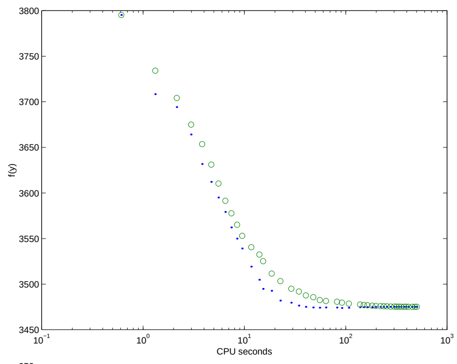
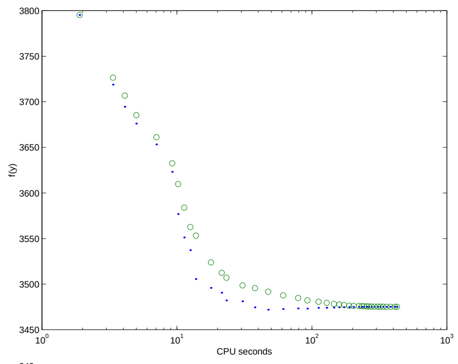
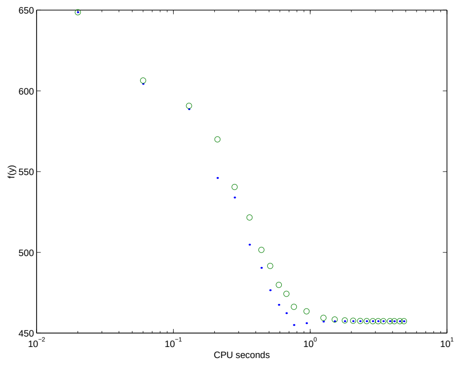
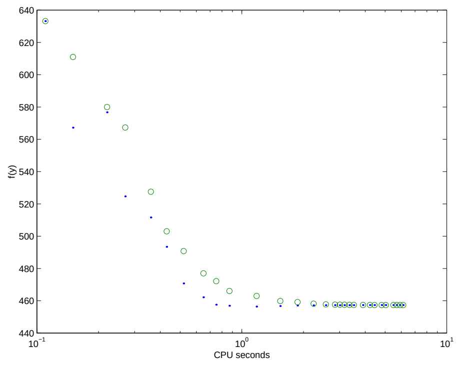
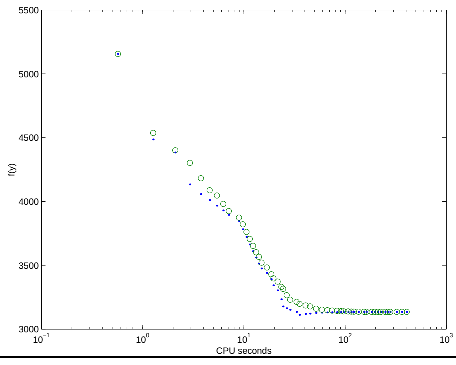
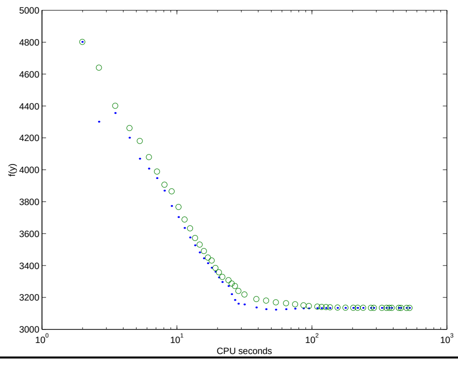
Problem	n	m	f^*	f_{term}	rel_acc	time	λ_%	k	desc.	$\|\nabla \bar{f}_+\|$
toruspm3-8-50	512	512	527.8087	527.8123	$6.9\cdot10^{-6}$	4	36	28	17	0.07
toruspm3-15-50	3375	3375	—	3475.156	—	8:27	30	102	43	0.17
torusg3-8	512	512	457.3582	457.3612	$6.6\cdot10^{-6}$	5	36	42	24	0.05
torusg3-15	3375	3375	—	3134.590	—	6:47	50	143	51	0.14

Table 2: Max-cut on toroidal grid graphs (Version 1.1)

Problem	n	m	f^*	f_{term}	rel_acc	time	λ_%	k	desc.	$\|\nabla \bar{f}_+\|$
toruspm-8-50	512	512	527.8087	527.8134	$9.0\cdot10^{-6}$	5	40	32	21	0.04
toruspm3-15-50	3375	3375	—	3475.162	—	7:10	40	100	39	0.18
torusg3-8	512	512	457.3582	457.3610	$6.1\cdot10^{-6}$	6	41	50	28	0.04
torusg3-15	3375	3375	—	3134.596	—	8:49	57	198	50	0.10

Computational results for Version 1.0 with $\varepsilon = 10^{-5}$ are listed in Table 1. The model is updated with parameters $n_K = 20$, $n_{\min} = 5$, $n_A = 8$, and $t_a = 0.01$. By Helmberg and Rendl [2000], Remark 4.5, the maximal bundle size needed to keep the contribution of the aggregate $\overline{W}$ small is roughly $\sqrt{2m}$ ($= \sqrt{2n}$ for max-cut). In practice, none of the four examples used more than 24 columns in P. This and the rather small number of iterations seems to indicate that the bundle

Figure 1: Max-cut on toroidal grid graphs

Problem	Version 1.0	Version 1.1
toruspm3-8-50		
toruspm3-15-50		
torusg3-8		
torusg3-15		

was large enough to accommodate all relevant information. Since the instances of the DIMACS challenge are grid graphs (average degree two or three) the cost matrices C are extremely sparse; therefore, the default heuristic for choosing n_L and n_C worked reasonably well.

In the case of Version 1.1, the results of Table 2 indicate roughly the same behavior, even though the percentage of time spent in the eigenvalue routine and the number of iterations has increased somewhat. The eigenvalue computation used Chebychev iterations as expected, but the new updating scheme for n_C may be worse. This and the different rule for adding new Lanczos vectors may both be responsible for the increase in iterations.

The plots of Figure 1 show that the starting point was improved by the heuristic, but the trade-off between improvement to time is not necessarily favorable.

DIMACS benchmark (Mittelmann [2001]). `SBmethod` (Version 1.1) is significantly faster on these instances than BMZ and DSDP. Moreover, the results are more accurate than those of BMZ. In the case of DSDP only the value for *toruspm3-15-50* is smaller; in fact the value of DSDP is even smaller than the lower bound produced by BMPR (a possible explanation might be, that the primal objective value is listed for DSDP). The good performance of `SBmethod` is probably due to the fact, that a rather small bundle size was sufficient to cover all relevant information (the contribution α^+ of the aggregate $\overline{W}$ remained small in the optimal solution of (10)). In consequence `SBmethod` converged in relatively few and inexpensive iterations.

3.2 The bisection set

The bisection instances are read and converted in the same way as the torus set (§3.1) except for one additional constraint (and a change of sign in the cost matrix to obtain a dual minimization problem),

$$\begin{array}{ll} \max & \langle C, X\rangle \\ \text{s.t.} & \operatorname{diag}(X) = e \\ & \langle ee^T, X\rangle = 0 \\ & X \succeq 0 \end{array} \qquad \min_{\binom{y}{y_{n+1}} \in \mathbb{R}^{n+1}} n\lambda_{\max}(C - \operatorname{Diag}(y) - y_{n+1}ee^T) + e^T y.$$

The additional constraint matrix $A_{n+1} = ee^T$ is represented in `SBmethod` in this form by means of the constraint class `GRAM_DENSE`. As described in §2.4, this allows `SBmethod`to compute the product of $(C - \mathcal{A}^T y)$ with a vector efficiently.

Table 3: Bisection problems (Version 1.0)

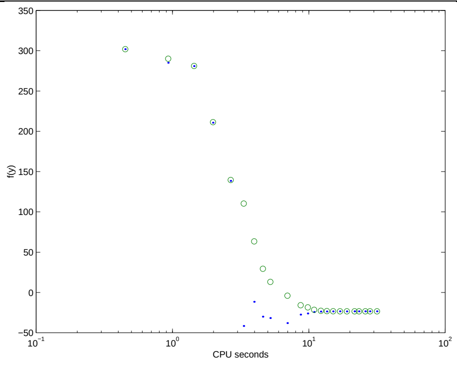
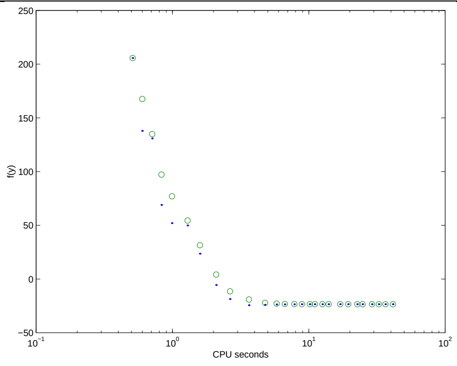
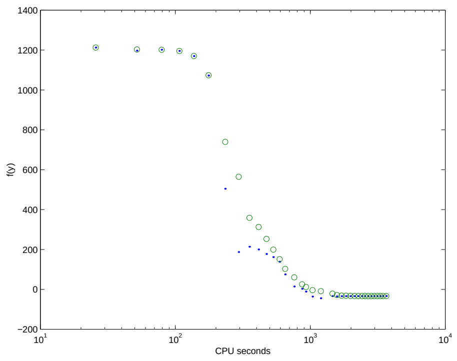
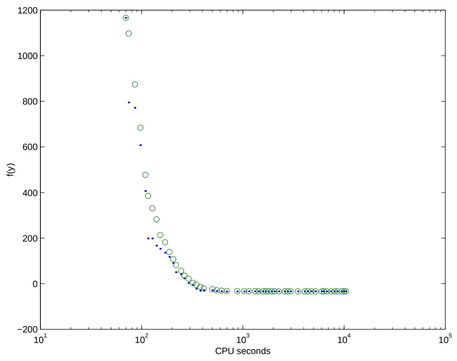
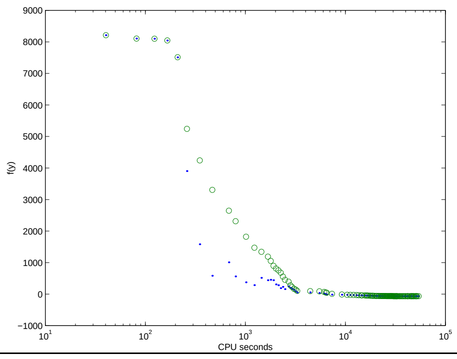
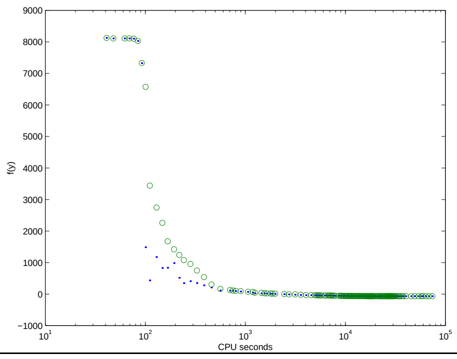
Problem	n	m	f^*	f_{term}	rel_acc	time	λ_%	k	desc.	$\|\nabla \bar{f}_+\|$
bm1	882	883	23.44340	23.43849	$-2.0\cdot10^{-4}$	32	78	38	22	0.06
biomedP	6514	6515	33.60140	33.59600	$-1.6\cdot10^{-4}$	1:01:42	47	58	34	0.11
industry	12637	12638	65.61310	65.60436	$-1.3\cdot10^{-4}$	15:01:11	41	418	110	0.04

Table 4: Bisection problems (Version 1.1)

Problem	n	m	f^*	f_{term}	rel_acc	time	λ_%	k	desc.	$\|\nabla \bar{f}_+\|$
bm1	882	883	23.44340	23.43855	$-2.0\cdot10^{-4}$	43	55	51	26	0.02
biomedP	6514	6515	33.60140	33.59922	$-6.3\cdot10^{-5}$	2:55:35	68	120	51	0.01
industry2	12637	12638	65.61310	65.60603	$-1.1\cdot10^{-4}$	20:53:51	58	542	141	0.01

Computational results for Version 1.0 with $\varepsilon = 10^{-4}$ are listed in Table 3. The model is updated with parameters $n_K = 45$, $n_{\min} = 25$, $n_A = 5$, and $t_a = 0.01$ except for the considerable smaller problem *bm1* where we use $n_{\min} = 12$. For eigenvalue computation we use $n_C = 0$ and $n_L = 200$ ($n_L = 100$ for *bm1*). In order to avoid premature termination for *industry2* we

Figure 2: Bisection problems

Problem	Version 1.0	Version 1.1
bm1		
biomedP		
industry2		

had to specify a different updating heuristic for weight u; we used the option `-u 1`. Otherwise these choices seemed to work reasonably well. This might be due to the following facts. For these problems the matrix vector multiplications are expensive (the cost matrix C of *biomedP* has approximately 630,000 nonzeros and for *industry2* roughly 800,000). This favors large n_L without Chebychev iterations. For expensive eigenvalue computations we would like to reduce the number of evaluations and therefore use a large bundle, $n_K = 45$ say. To our surprise, however, we observed that even for $t_a = 0.0001$ the bundle would not increase sufficiently to produce a model of good quality. Therefore, we forced the bundle size up by setting $n_{\min} = 25$. The rather small number of iterations seems to justify this approach.

The direct comparison with the results for Version 1.1 of Table 4 are somewhat misleading due to the higher precision requirements of Version 1.1. Indeed, a look at the plots of Figure 2 shows that Version 1.1 achieves the same objective values as Version 1.0 in comparable or even shorter time. It is surprising that for these instances the starting point heuristic of Version 1.1 seems to drastically speed up initial convergence in comparison to Version 1.0. For problems *bm1* and *biomedP* one reason is that the initial choice of the weight u, which is set to the norm of the subgradient in y^0, is significantly smaller in Version 1.1 due to the better y^0.

Observe, that in comparison to the torus set, the absolute size of the optimal objective value is small relative to the sum of all edge weights. Since the stopping criterion is formulated relative to the absolute size of the objective value it is not surprising that convergence is considerably slower for the bisection instances. Graphically, this is nicely illustrated in the plots of figures 1 and 2.

DIMACS benchmark (Mittelmann [2001]). `SBmethod` (Version 1.1) is significantly faster and more accurate than BMZ and DSDP on *bm1*. BMZ and DSDP (in fact, all other methods except BMPR) fail for *biomedP* and *industry2* due to lack of memory. The underlying graphs are quite dense and memory requirements for factorizing matrices with this support structure seem to be prohibitive. `SBmethod` and BMPR both do not require factorizations and are therefore able to work on these instances. Like in the case of the max-cut instances, the small number of iterations of `SBmethod` must be due to the small bundle size required to provide a model of high quality.

3.3 The Hamming set

For the Hamming set the task is to compute the Lovász ϑ-function for several instances of Hamming graphs. Various possibilities exist to formulate the ϑ-function as a semidefinite program (several are given in the seminal paper of Lovász [1979], many more have been developed since). In our experience the best approach for `SBmethod` is to employ the direct formulation as an eigenvalue optimization problem. Let $E \subset \{ij : i < j, i, j \in \{1, \ldots, n\}\}$ denote the set of edges and let E_{ij} be the matrix that has a one in entries ij and ji but is zero otherwise. Then the value of the ϑ-function for the graph with edge set E is the optimal value of the solution of the following programs,

$$\begin{array}{lll} \max & \langle ee^T, X\rangle & \\ \text{s.t.} & \langle I, X\rangle = 1 & \\ & \langle E_{ij}, X\rangle = 0 & ij \in E \\ & X \succeq 0 & \end{array} \qquad \min_{y \in \mathbb{R}^E} \lambda_{\max}(ee^T - \sum_{ij \in E} E_{ij} y_{ij}). \tag{20}$$

In `SBmethod` we describe the cost matrix by the class `GRAM_DENSE` and the constraints by the class `SINGLETON`.

Computational results for Version 1.0 with $\varepsilon = 10^{-4}$ are listed in Table 5. The model is updated with parameters $n_K = 0$ and $n_A = 5$ ($n_{\min}$ and t_a are irrelevant in this case). For eigenvalue computation we use $n_C = 0$ and $n_L = 60$. Again we try to argue why we believe these parameters to be reasonable. Since the number of edges in the graph is relatively high, the matrices are quite dense. This excludes Chebychev and asks for large n_L. The order of the matrices is rather small and so $n_L = 60$ covers already a significant portion of their spectrum. The instances have an exceedingly large number of constraints for their size. For this kind of problems it must be expected that the rank of the primal optimal solution is high and thus a large

Table 5: Lovász ϑ-function for Hamming graphs (Version 1.0)

Problem	n	m	f^*	f_{term}	rel_acc	time	λ_%	k	desc.	$\|\nabla \bar{f}_+\|$
hamming_9_8	512	2304	224.0000	224.0205	$9.1\cdot 10^{-5}$	10	87	86	54	0.00
hamming_10_2	1024	23040	102.4000	102.4093	$9.0\cdot 10^{-5}$	2:35	74	332	87	0.00
hamming_11_2	2048	56320	170.6666	170.6824	$9.2\cdot 10^{-5}$	8:09	72	453	205	0.00
hamming_7_5_6	128	1792	42.66666	42.66904	$5.5\cdot 10^{-5}$	1	74	42	25	0.00
hamming_8_3_4	256	16128	25.60000	25.60247	$9.3\cdot 10^{-5}$	26	51	159	54	0.00
hamming_9_5_6	512	53760	85.33333	85.33866	$6.2\cdot 10^{-5}$	43	61	62	58	0.00

Table 6: Lovász ϑ-function for Hamming graphs (Version 1.1)

Problem	n	m	f^*	f_{term}	rel_acc	time	λ_%	k	desc.	$\|\nabla \bar{f}_+\|$
hamming_9_8	512	2304	224.0000	224.0000	0.	0	45	2	0	0.01
hamming_10_2	1024	23040	102.4000	102.4000	0.	54	20	19	0	0.02
hamming_11_2	2048	56320	170.6667	170.6667	$1.9\cdot 10^{-8}$	4:28	13	22	0	0.02
hamming_7_5_6	128	1792	42.66667	42.66667	$7.6\cdot 10^{-9}$	0	67	1	0	0.00
hamming_8_3_4	256	16128	25.60000	25.60000	0.	5	14	8	0	0.01
hamming_9_5_6	512	53760	85.33333	85.33333	$-3.9\cdot 10^{-9}$	4	44	1	0	0.00

bundle would be needed to cover all relevant information. But large bundle size and large number of constraints entail high costs in the computation of the cost coefficients and the solution of the subproblem (10). In this case we experienced repeatedly that taking the smallest possible bundle, namely $n_K = 0$ and using new vectors only, helped to improve performance.

In solving these instances we encountered one particular obstacle. Looking at (20) it is obvious that the spectrum of the matrix in the starting point $y = 0$ is $\lambda_{\max}(ee^T) = n$ (with eigenvector $e/\sqrt{n}$) and $\lambda_i = 0$ otherwise. Thus, during computations the steepest descent direction $\mathcal{A}(ee^T/n)$ (see (8)) dominated for a long time. In consequence the dynamic updating strategy for the weight u repeatedly reduced u in order to increase the step size until the gap between largest and second largest eigenvalue was sufficiently small for a nontrivial model. At this point, however, u was already far too small to allow reasonable progress. In Version 1.0 the starting point heuristic was not available and so we decided in favor of an ugly but simple strategy: We increased the lower bound on u sufficiently. In particular, we used $u_{\min} = 0.005$ for the first three and $u_{\min} = 0.1$ for the last three instances.

The results of Version 1.1 of Table 6 are at first astonishing: The starting point heuristic already produces the exact optimal solution; the additional iterations are only needed to increase the model sufficiently until the stopping criterion holds.

The fact, that the steepest descent direction solves these problems effortlessly is obviously connected to the very specific choice of the instances, see Schiller [1999] for a detailed account of their properties. Since the plots are of little value here, we refrain from giving them.

DIMACS benchmark (Mittelmann [2001]). Due to our experience with other instances we do not consider the DIMACS set representative. In fact, it is to be expected that on less specialized problems BMZ performs better (if the factorization of the graph still fits into memory). Indeed, if the number of constraints is large (here, $m = |E|$), Remark 4.5 in Helmberg and Rendl [2000] indicates that a large bundle size may be needed to provide a model of high quality. The trade-off between quality and computation time is then often in favor of a small model at the cost of many iterations. In contrast, in BMZ the semidefiniteness constraint is transformed into optimizing a quadratic term over the Cholesky factor (the requirement of a positive diagonal is modeled by n sign constraints); the approach does not depend on the dimension of the active subspace of the semidefiniteness constraint. Therefore, BMZ is likely to be more effective if the active subspace is large. The Hamming instances are atypical in that a low dimensional subsapce is sufficient for `SBmethod` in spite of the large number of constraints. DSDP requires the factorization of a positive definite $m \times m$ matrix that is dense in general. Therefore memory requirements will be prohibitive for DSDP even for graphs with few nodes but large $m = |E|$.

3.4 The FAP set

We refer to Eisenblätter [2001] for a detailed description of the background of the problem and properties of the semidefinite relaxation. In the FAP instances we are given a weighted adjacency matrix A of order n (denoted by W in Pataki and Schmieta [1999]), a number of partitions p, and a set $E_0 \subset \mathcal{E} := \{ij : i < j, i, j \in \{1, \ldots, n\}\}$ of edges whose end nodes must lie in distinct partitions (we may assume $A_{ij} = 0$ for $ij \in E_0$). In this notation the semidefinite relaxation listed in Pataki and Schmieta [1999] reads

$$\begin{array}{ll} \min & \left\langle \frac{1}{2}\operatorname{Diag}(Ae) - \frac{p-1}{2p}(\operatorname{Diag}(Ae) - A), X \right\rangle \\ \text{s.t.} & \operatorname{diag}(X) = e \\ & X_{ij} = -\frac{1}{p-1} \qquad ij \in E_0 \\ & X_{ij} \geq -\frac{1}{p-1} \qquad ij \in \mathcal{E} \\ & X \succeq 0 \end{array} \tag{21}$$

Observe, that the dual of this relaxation yields a completely dense slack matrix; `SBmethod` is not designed for dense problems. In practice, we observed only little loss in quality of the bound by switching to a weaker relaxation, that requires $X_{ij} \geq 0$ only for ij in the support of A, *i.e.*, $ij \in E := \{ij \in \mathcal{E}, A_{ij} \neq 0\}$ (restricting the attention to the support of the cost function is a natural approach in combinatorial optimization). Indeed, the optimal value of

$$\begin{array}{ll} \min & \left\langle \frac{1}{2}\operatorname{Diag}(Ae) - \frac{p-1}{2p}(\operatorname{Diag}(Ae) - A), X \right\rangle \\ \text{s.t.} & \operatorname{diag}(X) = e \\ & X_{ij} = -\frac{1}{p-1} \qquad ij \in E_0 \\ & X_{ij} \geq -\frac{1}{p-1} \qquad ij \in E \\ & X \succeq 0. \end{array} \tag{22}$$

cannot be larger than that of (21), because a feasible X of (21) is also feasible for (22). Within the DIMACS test set, instances of (22) are referred to as `fap..-sup` problems and we will concentrate on these.

Since `SBmethod` requires a primal maximization problem we set $C = \frac{p-1}{2p}(\operatorname{Diag}(Ae) - A) - \frac{1}{2}\operatorname{Diag}(Ae)$. With $2X_{ij} = \langle E_{ij}, X\rangle$ (the matrix E_{ij} has a one in entries ij and ji and is zero otherwise) the negative of relaxation (22) reads

$$\begin{array}{ll} \max & \langle C, X\rangle \\ \text{s.t.} & \operatorname{diag}(X) = e \\ & \langle E_{ij}, X\rangle = -\frac{2}{p-1} \qquad ij \in E_0 \\ & \langle E_{ij}, X\rangle \geq -\frac{2}{p-1} \qquad ij \in E \\ & X \succeq 0 \end{array} \tag{23}$$

$$\min_{\substack{y \in \mathbb{R}^{n+|E_0|+|E|} \\ y_{ij} \leq 0 \ \forall ij \in E}} n\lambda_{\max}(C - \sum_{ij \in \{ii: i=1,\ldots,n\} \cup E_0 \cup E} E_{ij} y_{ij}) + \sum_{i=1}^{n} y_{ii} - \sum_{ij \in E_0 \cup E} \frac{2}{p-1} y_{ij}.$$

Again we used constraint class `SINGLETON` to implement E_{ij} in `SBmethod` and the sparse format for C.

For each dual feasible y the negative of the objective value of the eigenvalue problem (23) yields a lower bound on (21) and therefore on the original frequency assignment problem. In comparison to the sum of all edge weights (2862.862 for *fap09-sup*, 27213.05 for *fap25-sup*, and 96383.72 for *fap36-sup*) the (absolute) value of the optimal solution is small, so we can expect that it is difficult to achieve reasonable precision. Also note that the number of constraints is large. These two facts suggest that the spectral bundle method is not well suited for these instances.

Table 7: Frequency assignment problems (Version 1.1)

Problem	n	m	f_{term}	time	λ_%	k	inner	desc.	$\|\nabla \bar{f}_+\|$
fap09-sup	174	15225	10.79693	247:17:21	28	4657007	4657115	83	$4.1 \cdot 10^{-3}$
fap25-sup	2118	322924	*12.10477	315:36:46	56.13	187907	187907	96	0.15
fap36-sup	4110	1154467	*62.54669	682:14:24	57	112121	112122	95	0.87

A "*" indicates that computation was terminated externally before the stopping criterion was satisfied.

In our experiments with Version 1.0 we used the cost matrix $\overline{C} = \frac{p-1}{2p}[\operatorname{Diag}(Ae) - A] = C + \frac{1}{2}\operatorname{Diag}(Ae)$. Later it was agreed that only the cost matrix C should be used for experiments. A constant offset of the dual objective value may influence decisions based on relative sizes (*e.g.*, the stopping criterion) and so the old results are not directly comparable. In view of the high computation times and limited resources we decided to refrain from recomputing and presenting results for Version 1.0.

For Version 1.1, the results are displayed in Table 7. For instances *fap25-sup* and *fap36-sup* it was hopeless to wait for termination by the stopping criterion and we had to kill the processes at some point; the numbers given represent the values at the time when the processes were stopped. It is surprising that, in spite of the enormous number of sign constrained variables, the number of additional inner iterations to improve model precision is neglectable (108 for *fap09-sup*, none for *fap25-sup* and just one for *fap36-sup*). In all three cases the algorithm switched within a few iterations from a big bundle to a small one with $n_K = 0$ and $n_A = 7$ (see §2.2). In the plots of Figure 3 the positive effect of this switch is clearly visible. Yet, for *fap25-sup* and *fap36-sup* the final percentage of computation time spent in eigenvalue computation is rather high and it might be worth to use a slightly larger n_A.

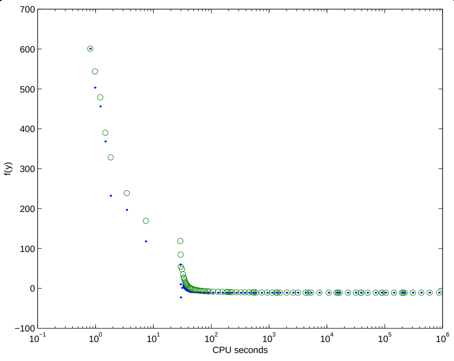
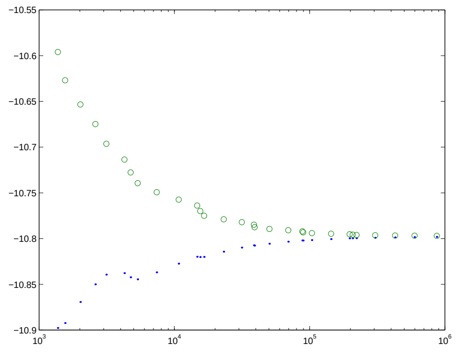
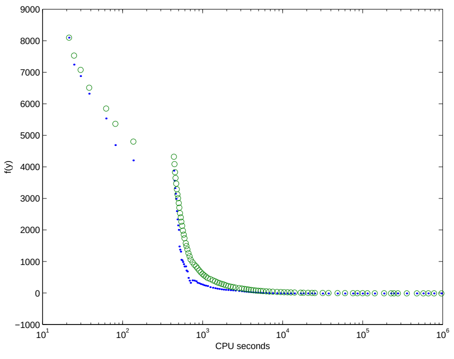
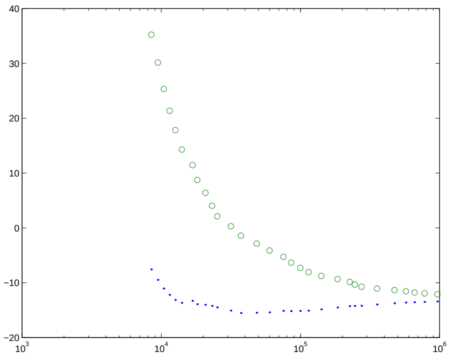
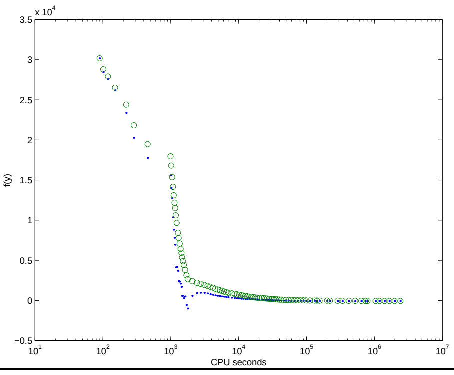
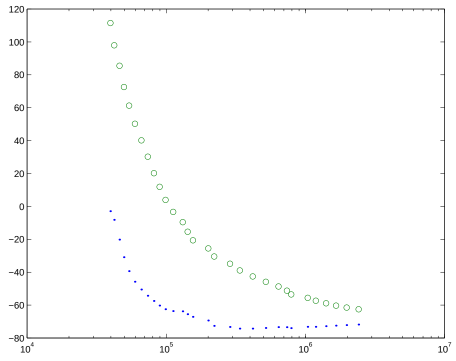
Again, the plots of Figure 3 nicely illustrate the unfavorable consequences of relative precision and the strong tailing off effect. The close-up of the 30 last iterations shows that the descent steps are evenly spaced on a *logarithmic* time scale.

DIMACS benchmark(Mittelmann [2001]). According to H. Mittelmann, personal communication, SBmethod (and probably BMZ as well) has been stopped prematurely for instances *fap25-sup* and *fap36-sup* because of the approaching submission deadline (unfortunately, this is not yet mentioned in report Mittelmann [2001]). For the *fap*-instances the same arguments apply as in the discussion of the ϑ-function. The large number of constraints entails a large active subspace in the semidefiniteness constraint. Therefore the heuristic of SBmethod Version 1.1 switches to a small bundle size and convergence is slow. Clearly, BMZ is faster. Note, however, that the results of SBmethod are significantly more accurate than those of BMZ (except for *fap36-sup*, that was given less computation time than BMZ). If computation times for achieving the same level of accuracy are compared, then BMZ is still significantly faster but the difference is less dramatic. Again, DSDP is unable to cope with the huge number of constraints.

4 Conclusion

In contrast to interior point methods, the code SBmethod is capable of dealing with large scale problems but requires tuning of a number of relevant parameters. The spectral bundle method is a very recent approach and we still lack good heuristics, not to speak of mathematically sound rules, for adapting these parameters automatically. A good choice of the parameters, however, yields very competitive results and there is hope that the connection between problem instance and good

Figure 3: Frequency assignment problems (Version 1.1)

Problem	Version 1.1	close-up: the last 30 descent steps
fap09-sup		
fap25-sup		
fap36-sup		

parameter choices will get more evident over time. The new starting point heuristic seems to work reasonably well. We are not fully satisfied with the current dynamic rules for updating parameters and we do not expect them to produce acceptable results for all new problem classes. Yet, the proposed heuristics may at least help to alleviate the problem of finding an initial choice for a new class of instances.

Based on the DIMACS benchmark of Mittelmann [2001] we believe it is fair to conclude that BMPR, BMZ, and `SBmethod` are currently the only methods suitable for large scale semidefinite programming. BMPR is a primal (possibly infeasible) method and is clearly fastest. In connection with semidefinite relaxations of combinatorial optimization problems it is a good source for primal heuristics but it cannot safely be used as a bounding procedure. BMZ and `SBmethod` are both dual feasible methods and yield valid bounds on the underlying combinatorial optimization problems. If the Cholesky factor of the dual slack matrix $Z = \mathcal{A}^T y - C$ does not suffer from excessive fill-in and the active subspace of the semidefiniteness constraint on Z is large (typically connected to a large number m of constraints) then BMZ seems to be the method of choice. If factorizing $C - \mathcal{A}^T y$ is out of scope or the active subspace is small (as is the case for a small number m of constraints) then `SBmethod` is likely to yield better results. It is worth noting, that in spite of the weak nature of the classical subgradient stopping criterion (15), the solutions produced by `SBmethod` are as acurrate as those of competing interior point codes. The dual interior point code DSDP exhibits the same behavior as primal dual interior point methods: if the number of constraints m is big then forming and factorizing the dense positive definite system matrix of order m is computationally prohibitive. Attempts to replace this step by a conjugate gradient approach have so far not produced convincing results.

A frequently heard concern about the dual approach of the spectral bundle method is that no primal information is available. In fact, the solution W^+ of the quadratic semidefinite subproblem (10) may be interpreted as an approximate primal solution as we have pointed out already in Helmberg and Rendl [2000]. A formal proof that W^+ indeed converges to a primal optimal solution (assuming a standard regularity condition) is given in Helmberg [2001]. There it is also shown how this information can be exploited to set up a primal cutting plane algorithm.

Acknowledgment. I thank K. C. Kiwiel who not only contributed many ideas to the code but also improved its reliability significantly. He pointed out many mistakes in an earlier version of the code by miraculously spotting tiny inconsistencies in tons of log-files that I sent him by email. I also thank G. Pataki and an anonymous referee for their valuable criticism that helped to improve the presentation.

A Notation

General.

$\mathbb{R}$	real numbers
$\mathbb{R}^n$	real column vector of dimension n
S_n	$n \times n$ symmetric real matrices
S_n^+, $A \succeq 0$	$n \times n$ symmetric positive semidefinite matrices
I, I_n	identity of appropriate size or of size n
e	vector of all ones of appropriate dimension
E_{ij}	symmetric matrix with ij-entry equal to one and zero otherwise
$\lambda_i(A)$	i-th eigenvalue of $A \in S_n$, usually $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$
$\lambda_{\min}(A)$, $\lambda_{\max}(A)$	minimum and maximum eigenvalue of A
Λ_A	diagonal matrix with $(\Lambda_A)_{ii} = \lambda_i(A)$
A^T	transpose of A
$\operatorname{tr}(A)$	trace of $A \in \mathbb{R}^{n\times n}$, $\operatorname{tr}(A) = \sum_{i=1}^n a_{ii} = \sum_{i=1}^n \lambda_i(A)$
$\langle A, B\rangle$	inner product in $\mathbb{R}^{m\times n}$, $\langle A, B\rangle = \operatorname{tr}(B^T A)$
$\|A\|_{\mathrm{F}}$	Frobenius norm of A, $\|A\|_{\mathrm{F}} = \sqrt{\langle A, A\rangle}$
$\operatorname{Diag}(v)$	diagonal matrix with v on its main diagonal
$\operatorname{diag}(A)$	the diagonal of $A \in \mathbb{R}^{n\times n}$ as a column vector
argmin	minimizing argument
Argmin	set of minimizing arguments

Spectral bundle method.

Y	dual feasible set
y	design variables in $\mathbb{R}^m$, feasible for $y \in Y$
η	Lagrange multipliers for sign constraints on y
$\mathcal{W}$, $W \in \mathcal{W}$	positive semidefinite matrices of trace 1, $\mathcal{W} = \{\, W \succeq 0 : \operatorname{tr} W = 1\,\}$
$\widehat{\mathcal{W}} \subset \mathcal{W}$,	semidefinite model, $\widehat{\mathcal{W}} := \{PVP^T + \alpha\overline{W} : \operatorname{tr} V + \alpha = a, V \succeq 0, \alpha \geq 0\}$
$\mathcal{A}, \mathcal{A}^T$	constraint matrix and adjoint, $[\mathcal{A}X]_i = \langle A_i, X\rangle$, $\mathcal{A}^T y = \sum_{i=1}^m y_i A_i$
$C - \mathcal{A}^T y$	affine matrix function, C is a given "cost matrix"
b	given "right hand side" vector
f	objective function, $f(y) = a\lambda_{\max}(C - \mathcal{A}^T y) + b^T y + \imath_Y(y)$
$f_{W,\eta}$	affine minorant of f, $f_{W,\eta} = \langle C, W\rangle + \langle b - \eta - \mathcal{A}W, y\rangle$
$f_{\widehat{\mathcal{W}},\widehat{Y}}$	model of f for $\widehat{\mathcal{W}} \subseteq \mathcal{W}$, $\widehat{Y} \subseteq Y$, $f_{\widehat{\mathcal{W}},\widehat{Y}}(y) = \sup_{(W,\eta)\in\widehat{\mathcal{W}}\times\widehat{Y}} f_{W,\eta}(y)$
u	weight of quadratic term

Parameters of `SBmethod` [default values are given for Version 1.1].

ε	termination precision (15) [$\varepsilon = 10^{-5}$]
κ_η	initial convex combination of old and new η (Step 2) [$\kappa_\eta = 1$]
κ_M	relative model precision (Step 2c) [$\kappa_M = 0.6$]
κ	relative descent step precision (Step 3) [$\kappa = 0.1$]
$\bar{\kappa}$	relative null step precision (Step 3) [$\bar{\kappa} = 0.1$]
n_L	number of Lanczos iterations per restart [Lanczos heuristic §2.1]
n_C	degree of Chebychev polynomial for spectral transformation [Lanczos heuristic §2.1]
n_K	maximum number of columns to keep [$n_K = 45$, bundle update heuristic §2.2]
$n_{\min}$	minimum number of columns to keep [$n_{\min} = 30$, bundle update heuristic §2.2]
n_A	maximum number of columns to add [$n_A = 10$, bundle update heuristic §2.2]
y_0	starting point [set $y_0 = 0$ and apply starting point heuristic §2.3]
u	weight of quadratic term in (6), update rule of Helmberg and Kiwiel [1999]

References

Benson, S., Ye, Y., and Zhang, X. (2000). Solving large-scale sparse semidefinite programs for combinatorial optimization. *SIAM J. Optim.*, 10(2):443–461.

Burer, S. and Monteiro, R. D. (2001). A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. Technical report, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332. To appear in Math. Programming, Series B.

Burer, S., Monteiro, R. D., and Zhang, Y. (1999). Interior-point algorithms for semidefinite programming based on a nonlinear programming formulation. Technical report, Department of Computational and Applied Mathematics, Rice University, Houston, Texas 77005.

Burer, S., Monteiro, R. D. C., and Zhang, Y. (2001). Solving a class of semidefinite programs via nonlinear programming. Technical report, Dept. of Computational and Applied Mathematics, Rice University, Houston, Texas 77005, USA. Combined and revised version of technical reports TR99-17 and TR99-23. Tentatively accepted for Mathematical Programming A.

Eisenblätter, A. (2001). *Frequency Assignment in GSM Networks.* PhD-Thesis, Technische Universität Berlin, Berlin. ISBN 3-89873-213-4, URL: `ftp://ftp.zib.de/pub/zib-publications/books/PhD_eisenblaetter.ps.Z`.

Goemans, M. X. (1997). Semidefinite programming in combinatorial optimization. *Math. Programming*, 79:143–161.

Golub, G. H. and van Loan, C. F. (1989). *Matrix Computations.* The Johns Hopkins University Press, 2^{nd} edition.

Helmberg, C. (2000a). Semidefinite programming for combinatorial optimization. Habilitationsschrift TU Berlin, Jan. 2000; ZIB-Report ZR 00-34, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustraße 7, 14195 Berlin, Germany.

Helmberg, C. (2000b). `SBmethod` — a C++ implementation of the spectral bundle method. Manual to Version 1.1, ZIB-Report ZR 00-35, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustraße 7, 14195 Berlin, Germany. URL: `http://www.zib.de/helmberg/SBmethod`.

Helmberg, C. (2001). A cutting plane algorithm for large scale semidefinite relaxations. ZIB-Report ZR 01-26, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustraße 7, 14195 Berlin, Germany.

Helmberg, C. and Kiwiel, K. C. (1999). A spectral bundle method with bounds. ZIB Preprint SC-99-37, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustraße 7, 14195 Berlin, Germany. Revised Sep. 2001, to appear in Math. Programming.

Helmberg, C. and Rendl, F. (2000). A spectral bundle method for semidefinite programming. *SIAM J. Optim.*, 10(3):673–696.

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

Lewis, A. S. and Overton, M. L. (1996). Eigenvalue optimization. *Acta Numerica*, pages 149–190.

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

Mittelmann, H. D. (2001). An independent benchmarking of SDP and SOCP solvers. Technical report, Dept. Mathematics, Arizona State University, Tempe, AZ 85287-1804, USA.

Nayakkankuppam, M. V. (1999). *Optimization Over Symmetric Cones.* PhD-Thesis, Department of Computer Science, Graduate School of Arts and Science, New York University.

Parlett, B. N. (1998). *The Symmetric Eigenvalue Problem*, volume 20 of *Classics in Applied Mathematics.* SIAM, Philadelphia.

Pataki, G. and Schmieta, S. (1999). The DIMACS library of mixed semidefinite-quadratic-linear programs. Draft manuscript, Computational Optimization Research Center, Columbia University. Available at `http://dimacs.rutgers.edu/Challenges/Seventh/Instances/`.

Saad, Y. (1992). *Numerical Methods for Large Eigenvalue Problems.* Halsted Press, New York.

Schiller, F. (1999). Zur Berechnung und Abschätzung von Färbungszahlen und der ϑ-Funktion von Graphen. Diplomarbeit, Technische Universität Berlin, Fachbereich Mathematik, Sekretariat MA 4-1, Straße des 17. Juni 136, 10623 Berlin, Germany.