C. HELMBERG[1]

A Cutting Plane Algorithm for Large Scale Semidefinite Relaxations

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

ZIB-Report 01-26 (October 2001)

A Cutting Plane Algorithm for Large Scale Semidefinite Relaxations

Christoph Helmberg[*]

October 10, 2001

Abstract

The recent spectral bundle method allows to compute, within reasonable time, approximate dual solutions of large scale semidefinite quadratic 0-1 programming relaxations. We show that it also generates a sequence of primal approximations that converge to a primal optimal solution. Separating with respect to these approximations gives rise to a cutting plane algorithm that converges to the optimal solution under reasonable assumptions on the separation oracle and the feasible set. We have implemented a practical variant of the cutting plane algorithm for improving semidefinite relaxations of constrained quadratic 0-1 programming problems by odd-cycle inequalities. We also consider separating odd-cycle inequalities with respect to a larger support than given by the cost matrix and present a heuristic for selecting this support. Our preliminary computational results for max-cut instances on toroidal grid graphs and balanced bisection instances indicate that warm start is highly efficient and that enlarging the support may sometimes improve the quality of relaxations considerably.

MSC 2000: 90C22; 90C25, 90C27, 90C09, 90C20, 90C06
Keywords: bisection, equicut, max-cut, quadratic 0-1 programming, semidefinite programming, spectral bundle method, subgradient method

1 Introduction

Crowder, Johnson, and Padberg [7] initiated the rise of general mixed integer programming by solving several large scale, unstructured 0-1 linear programming problems via a unified cutting plane framework. Can we set up a similar framework for large scale quadratic 0-1 programming problems?

It seems likely, that this question motivated much of the work on the boolean quadric polytope and the max-cut polytope, see [29, 9] and references therein. In the late eighties basic techniques for lifting linear inequalities into quadratic space were developed [34, 27, 1]; Lovász and Schrijver [27] linked this to a semidefinite relaxation of quadratic 0-1 programming [35] and demonstrated by means of the stable set problem that much can be gained by doing so. Further evidence on the effectiveness of the semidefinite approach was provided by Goemans and Williamson [13] via their approximation algorithm for max-cut. These works provide a clear guideline on how one should set up and improve relaxations of constrained quadratic 0-1 programming problems. Unfortunately, the final ingredient, an efficient algorithm that

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

solves large scale semidefinite relaxations in acceptable time and allows for the addition of cutting planes on the fly, is still missing. In this work we would like to convince the reader that the spectral bundle method [19, 18] provides all that is needed — certainly not for all applications, but for many cases of relevance.

The dual of a semidefinite program with bounded feasible set can be transformed into a problem of minimizing the maximum eigenvalue of an affine matrix function (see, *e.g.*, [16]). The latter is a nonsmooth convex optimization problem that may be tackled by subgradient and bundle methods, see [21] and references therein. The spectral bundle method [19, 18] is tuned to eigenvalue optimization problems and their associated semidefinite programs in that it uses a quadratic semidefinite subproblem. It was already pointed out in [19], that the solutions of this quadratic semidefinite subproblems may be interpreted as (infeasible) approximate solutions to the primal semidefinite program. Here, we prove rigorously that these approximations converge to a primal optimal solution within the setting of the spectral bundle method with bounds [18]. Feltenmark and Kiwiel [11] proved a related result for a classical proximal bundle method; see also [4] and references therein for other approaches for generating primal solutions from subgradient methods.

The primal approximations will serve as input for a separation oracle. Since these approximations are infeasible in general, precautions have to be taken against the separation oracle returning the same inequality again and again. A realistic assumption, that is often fulfilled in practice, is that the oracle returns a maximally violated inequality from a finite representation $Ax \leq b$ of the polyhedron. The combination of the spectral bundle method with such an oracle yields a cutting plane algorithm that generates a sequence of iterates converging to primal and dual optimal solutions whenever a strictly feasible primal solution exists. To the best of our knowledge this is the first provably convergent cutting plane approach based on bundle methods. Since semidefinite programming includes linear and second order cone programming, the algorithm easily extends to cutting plane algorithms over products of these cones as long as the primal feasible set is bounded.

In our implementation we concentrate on the quadratic 0-1 programming setting and do not obey all requirements for theoretic convergence in favor of computational efficiency. For various reasons we prefer to work with the equivalent semidefinite relaxation of quadratic $\{-1, 1\}$ programming which is better known as the max-cut relaxation. We improve the basic relaxation by adding odd-cycle inequalities as cutting planes. In contrast to linear programming it may help to separate these inequalities also with respect to support not contained in the cost function. We present a simple heuristic for enlarging the support that turned out to considerably improve the quality of the bound for several bisection instances. In order to illustrate this and the behavior of the cutting plane approach in general, we present preliminary numerical results for max-cut instances on toroidal grid graphs and bisection instances from numerical linear algebra.

Here is an outline of the paper. In §2 we review the equivalence between 0-1 and $\{-1, 1\}$ formulations and lists some important properties of the odd-cycle inequalities. Next, in §3, we explain the basic steps of the spectral bundle method with bounds and prove primal convergence of the iterates. This part relies heavily on [18] which should be at hand. Based on these convergence properties we develop a conceptual cutting plane algorithm with convergence guarantee in §4. For efficiency reasons we employ a slightly different approach in practice. The implementational choices are described in §5. Finally, we present preliminary computational results §6.

Our notation is quite standard. The set of symmetric matrices of order n will be denoted by S_n. $A \succeq 0$, $A \in S_n^+$ refers to positive semidefinite matrices, $A \succ 0$ is used for positive definiteness. The trace $\operatorname{tr} A$ is the sum of the diagonal elements; $\operatorname{diag}(A)$ denotes the vector of diagonal elements. For $A, B \in S_n$ or $A, B \in \mathbb{R}^{m\times n}$, we employ the inner product $\langle A, B\rangle = \operatorname{tr} B^T A$. When minimizing some function $f(y)$, $\operatorname{argmin} f$ is refers to a unique minimizer of f and $\operatorname{Argmin} f$ to the set of minimizers. An (undirected) graph $G = (V, E)$ consists of a finite set of nodes $V \subset \mathbb{N}$ and a set of edges $E \subset \{\{i, j\} : i < j,\ i, j \in V\}$. We only consider graphs without loops or multiple edges. For an edge $\{i, j\}$ we will also write ij, because we typically associate edges with matrix elements a_{ij}. A set of edges $C \subset E$ is called a cycle (of length k), if $C = \{v_1v_2, v_2v_3, \ldots, v_kv_1\}$ for pairwise distinct $v_i \in V$, $i = 1, \ldots, k$. For a matrix $A \in S_n$, the support graph refers to $G = (V, E)$ with $V = \{1, \ldots, n\}$ and $E = \{ij : i < j, a_{ij} \neq 0, i, j \in V\}$.

2 SDP-Relaxations for quadratic 0-1 and $\{-1, 1\}$ programming

We first review the process by which the semidefinite Lovász-Schrijver relaxation for constrained quadratic 0-1 programming [27] can be transformed into an equivalent semidefinite relaxation for quadratic $\{-1, 1\}$ programming (we refer to the latter as the max-cut setting). For a cost matrix $\bar{C} \in S_n$, constraints matrices $\bar{A}_i \in S_n$ for $i \in M := \{1, \ldots, m\}$, and right hand side b a semidefinite program arising from a Lovász-Schrijver relaxation over $n-1$ 0-1 variables may read

$$\text{(SQP)}\qquad \begin{array}{rll} \max & \langle \bar{C}, \bar{Y}\rangle & \\ \text{s.t.} & \langle \bar{A}_i, \bar{Y}\rangle + s_i = b_i & \text{for } i \in M \\ & \bar{Y} = \begin{bmatrix} 1 & \operatorname{diag}(Y)^T \\ \operatorname{diag}(Y) & Y \end{bmatrix} \succeq 0, s \geq 0. & \end{array}$$

By employing the scaling

$$X = Q^{-1}\bar{Y}Q^{-T} \qquad \text{with} \quad Q = \begin{bmatrix} 1 & 0 \\ \frac{1}{2}e & \frac{1}{2}I_n \end{bmatrix} \quad \text{and} \quad Q^{-1} = \begin{bmatrix} 1 & 0 \\ -e & 2I_n \end{bmatrix}$$

and by transforming the coefficient matrices according to the identity $\langle \bar{A}, \bar{Y}\rangle = \langle Q\bar{A}Q^T, X\rangle$,

$$C := Q\bar{C}Q^T, \qquad A_i := Q\bar{A}_iQ^T \quad \text{for } i \in M, \tag{1}$$

we obtain an equivalent semidefinite relaxation within the max-cut setting [25, 15] (e denotes the vector of all ones)

$$\text{(SMC)}\qquad \begin{array}{rll} \max & \langle C, X\rangle & \\ \text{s.t.} & \langle A_i, X\rangle + s_i = b_i & \text{for } i \in M \\ & \operatorname{diag}(X) = e & \\ & X \succeq 0, s \geq 0. & \end{array}$$

Indeed, if the equality constraints describing the structure of $\bar{Y}$ are chosen appropriately, then (SQP) and (SMC) share the same slack and dual variables. Furthermore, the transformation (1) preserves sparsity and low rank structure of the constraints [15]. Thus, we may switch between both formulations without loss, in theory and in practice.

Likewise, the boolean quadric polytope and the max-cut polytope are isomorphic; this has already been proven in [8]. Both polytopes have been studied extensively (see [29] for the boolean quadric polytope and [9] for the max-cut polytope).

Our goal is to devise a cutting plane approach for generically improving semidefinite relaxations of type (SMC) or (SQP) by exploiting polyhedral knowledge about the underlying polytopes. Within our computational framework, the semidefinite relaxation of max-cut (SMC) offers significant advantages over (SQP) and therefore we will concentrate on the max-cut setting. In particular, we are interested in cutting planes that are suitable for large sparse cost matrices C.

Numerous classes of facet defining inequalities of the cut polytope appear in the literature, but for most of them no efficient separation algorithm or heuristic is available. In the case of large unstructured support graphs $G = (V, E)$, the only class that has proven to be of practical value is, so far, the class of odd-cycle inequalities. Formulated within the $\{-1, 1\}$ setting of (SMC) they read

$$\sum_{ij\in C\setminus F} x_{ij} - \sum_{ij\in F} x_{ij} \leq |C| - 2 \qquad \text{for } C \subset E \text{ a cycle, } F \subseteq C, |F| \text{ odd.} \tag{2}$$

Odd-cycle inequalities can be separated in polynomial time [5] by solving shortest path problems in an auxiliary graph with twice the number of nodes and four times the number of edges. They provide a complete description of the cut polytope for graphs not contractible to K_5 (the complete graph on 5 nodes) [33, 2].

In a pure polyhedral setting, enlarging the number of cycles in a graph (and therefore the number of separable odd-cycle inequalities) by adding edges with weight 0 does not improve the relaxation [3]. This is not true in combination with the semidefinite relaxation (SMC) as has been observed on many examples.

3 Primal convergence of the spectral bundle method

We first introduce some basic objects and concepts that we will need throughout the next two sections. For $a > 0$ let

$$\mathcal{W} = \{X \succeq 0 : \langle I, X\rangle = a\} \tag{3}$$

denote the set of positive semidefinite matrices of order n with constant trace a. Consider the semidefinite program

$$\text{(PSDP)} \qquad \begin{array}{rl} \max & \langle C, X\rangle \\ \text{s.t.} & \mathcal{A}X + s = b \\ & X \in \mathcal{W}, s \geq 0 \end{array}$$

with variables $X \in S_n$, $s \in \mathbb{R}^m_+$, cost matrix $C \in S_n$, right hand side vector $b \in \mathbb{R}^m$ and a constraint matrix (or linear map) $\mathcal{A} : S_n \to \mathbb{R}^m$. This covers all forms of linear programs over symmetric cones with bounded feasible sets. For theoretical purposes, equality constraints would do, but this formulation is closer to practical requirements.

Introducing Lagrange multipliers $y \in \mathbb{R}^m$ and dualizing we arrive at a dual problem

$$\min_{y\in\mathbb{R}^m} \left\{ f(y) := \sup_{(X,s)\in\mathcal{W}\times\mathbb{R}^m_+} \langle C, X\rangle + \langle b - s - \mathcal{A}X, y\rangle \right\}. \tag{4}$$

This problem is tightly related to the standard dual semidefinite program of (PSDP). By making use of (3), it can be verified that strong duality holds for (4) and (PSDP) (*v.* [19]).

The function f is the supremum over the family of linear functions

$$f_{W,\eta}(y) := \langle C, W\rangle + \langle b - \eta - \mathcal{A}W, y\rangle \qquad (W,\eta) \in \mathcal{W} \times \mathbb{R}^m_+ \tag{5}$$

and therefore convex. Denoting by $\mathcal{A}^T$ the adjoint of $\mathcal{A}$ (by definition it satisfies $\langle \mathcal{A}X, y\rangle = \langle X, \mathcal{A}^T y\rangle$ for all $(X, y) \in S_n \times \mathbb{R}^m$) we may express f as a sum of two well known supremums,

$$\begin{aligned} f(y) &= \langle b, y\rangle + \sup_{W \in \mathcal{W}} \langle C - \mathcal{A}^T y, W\rangle + \sup_{\eta \in \mathbb{R}^m_+} \langle -\eta, y\rangle \\ &= \langle b, y\rangle + \quad a\lambda_{\max}(C - \mathcal{A}^T y) \quad + \quad \imath_{\mathbb{R}^m_+}(y) \end{aligned} \tag{6}$$

The first supremum is a "max" and is related to the maximum eigenvalue function $\lambda_{\max}(\cdot)$ by the fact that $\lambda_{\max}(A) = \max\{\langle A, X\rangle : X \succeq 0, \langle I, X\rangle = 1\}$ (see, *e.g.*, [26]). The second supremum yields the indicator function $\imath_Y$ for $Y := \mathbb{R}^m_+$ ($\imath_Y(y) = 0$ for $y \in Y$ and ∞ otherwise). Thus, the effective domain of f is Y. For a feasible $\bar{y} \in Y$, the function value and a subgradient may be determined by computing $\lambda_{\max}(C - \mathcal{A}^T\bar{y})$ and a corresponding eigenvector v. With $W_S := avv^T$ a subgradient of f in $\bar{y}$ is, *e.g.*, $\nabla f_{W_S,0} = b - \mathcal{A}W_s$. By (4), the subdifferential of f at $\bar{y} \in Y$ is

$$\partial f(\bar{y}) = \{\nabla f_{W,\eta} : f_{W,\eta}(\bar{y}) = f(\bar{y}), (W,\eta) \in \mathcal{W} \times Y\}.$$

In order to solve (4) we employ the spectral bundle method with bounds [18] whose main steps we summarize next. Using subgradient information it forms a model $\hat{f}$ minorizing f and determines a new *candidate* y^+ as the minimizer of the augmented model $\hat{f} + \frac{u}{2}\|\cdot - \hat{y}\|^2$, where $\hat{y}$ is the *center of stability* and the *weight* $u > 0$ provides indirect control on the distance of y^+ to $\hat{y}$. At this candidate, f is evaluated and a subgradient is computed. If progress is good in comparison to the progress predicted by the model, the algorithm moves its center to the new candidate (a *descent step*, $\hat{y}$ is set to y^+). Otherwise the center is left unchanged (a *null step*) but the subgradient is used to improve the model.

The model $\hat{f}$ is formed as follows. For arbitrary subsets $\widehat{\mathcal{W}} \subseteq \mathcal{W}$ and $\widehat{Y} \subseteq Y$ define

$$f_{\widehat{\mathcal{W}},\widehat{Y}}(y) := \sup_{(W,\eta)\in\widehat{\mathcal{W}}\times\widehat{Y}} f_{W,\eta}(y) \qquad \le f(y) \qquad \text{for all } y \in \mathbb{R}^m. \tag{7}$$

For example, $f_{\mathcal{W},Y} = f$ and $f_{\mathcal{W},0}(y) = f(y)$ for all $y \in Y$. Instead of $f_{\{W\},\widehat{Y}}$ or $f_{\widehat{\mathcal{W}},\{\eta\}}$ we will also write $f_{W,\widehat{Y}}$ and $f_{\widehat{\mathcal{W}},\eta}$. In our algorithm we choose $\widehat{Y} = Y$ and

$$\widehat{\mathcal{W}} = \left\{PVP^T + \alpha\overline{W} : \langle I, V\rangle + \alpha a = a, V \in S^+_r, \alpha \ge 0\right\} \tag{8}$$

for a given *bundle* $P \in \mathbb{R}^{n\times r}$, $P^TP = I_r$, and *aggregate matrix* $\overline{W} \in \mathcal{W}$. The matrices P and $\overline{W}$ will be updated at the end of each iteration.

Given a center of stability $\hat{y}$ and a weight u, the candidate is now determined by computing $\operatorname{argmin} f_{\widehat{\mathcal{W}},Y} + \frac{u}{2}\|\cdot - \hat{y}\|^2$. Using standard saddle-point arguments from convex analysis [32] one can show that

$$\min_{y\in\mathbb{R}^m} f_{\widehat{\mathcal{W}},Y}(y) + \frac{u}{2}\|y - \hat{y}\|^2 = \max_{(W,\eta)\in\widehat{\mathcal{W}}\times Y} \min_{y\in\mathbb{R}^m} f_{W,\eta}(y) + \frac{u}{2}\|y - \hat{y}\|^2 \tag{9}$$

and that solving the right hand side yields the left hand side minimizer, as well. The minimizer of the right hand side inner minimization is (use (5))

$$y_{W,\eta} = \hat{y} - \frac{1}{u}\nabla f_{W,\eta} = \hat{y} - \frac{1}{u}(b - \eta - \mathcal{A}W). \tag{10}$$

Substituting this into the right hand side of (9) and using the definition (5) of $f_{W,\eta}$ we obtain the dual function to the augmented model,

$$\psi(W,\eta) := \langle C, W\rangle + \langle b-\eta-\mathcal{A}W, \hat{y}\rangle - \frac{1}{2u}\|b-\eta-\mathcal{A}W\|^2. \tag{11}$$

The exact maximizing pair of ψ would yield the exact candidate via (10). For efficiency reasons, however, we prefer to compute a rough approximation by a coordinate-wise approach in Gauss-Seidel fashion. In particular, we first fix an $\hat{\eta}$ and compute a

$$W^+ \in \operatorname{Argmax}\left\{\psi(W,\hat{\eta}) : W \in \widehat{\mathcal{W}}\right\} \tag{12}$$

by solving, by means of an interior point algorithm, the quadratic semidefinite subproblem

$$\begin{array}{ll} \max & \langle C, W\rangle + \langle b-\hat{\eta}-\mathcal{A}W, \hat{y}\rangle - \frac{1}{2u}\|b-\hat{\eta}-\mathcal{A}W\|^2 \\ \text{s.t.} & W = PVP^T + \alpha\overline{W} \\ & \operatorname{tr} V + a\alpha = a \\ & V \succeq 0, \alpha \geq 0. \end{array} \tag{13}$$

(Observe, that in solving (13) we need only $\mathcal{A}\overline{W}$ and $\langle C, \overline{W}\rangle$ and not $\overline{W}$ itself.) Then we determine the next η^+ as the maximizer for this fixed W^+,

$$\eta^+ := \operatorname{argmax}\left\{\psi(W^+,\eta) : \eta \in Y\right\} = \max\{0, -u\hat{y} + b - \mathcal{A}W^+\}. \tag{14}$$

The corresponding approximate candidate is feasible (*i.e.*, in the effective domain Y of f) and satisfies complementarity,

$$y^+ := y_{W^+,\eta^+} = \max\{\hat{y} - \frac{1}{u}(b-\mathcal{A}W^+), 0\}, \qquad \langle \eta^+, y^+\rangle = 0. \tag{15}$$

Even though we allow for several repetitions of these "coordinate-wise" steps in Algorithm 3.1 (we call those *inner iterations*), it is shown in [18] that one inner iteration suffices to ensure convergence.

As pointed out in the text following (6), evaluating $f(y^+)$ can be done by computing the maximum eigenvalue $\lambda_{\max}(C-\mathcal{A}^Ty^+)$. To exploit available structure in the matrix $C-\mathcal{A}^Ty^+$ it is advantageous to employ iterative methods like the Lanczos method (see, *e.g.*, [14]) that rely on matrix vector multiplications only. These methods produce successively better lower estimates and approximate eigenvectors of $\lambda_{\max}(C-\mathcal{A}^Ty^+)$. As soon as this lower estimate indicates a null step, the $W_S = avv^T$ corresponding to the approximate eigenvector v guarantees sufficient improvement of the model for convergence [18]. This is the rational for combining descent test and evaluation in step 2 of Algorithm 3.1 below.

In order to guarantee progress of the algorithm after a null step, the new model $\widehat{\mathcal{W}}^+$ has to contain W^+ and W_S. The minimal choice is $P^+ = v$ and $\overline{W}^+ = W^+$. A better strategy, that successively adapts the subspace spanned by the columns of P, is described in [18].

Algorithm 3.1 (spectral bundle method with bounds) *[18]*
Input: $y^0 \in \mathbb{R}^m_+$, $\varepsilon_{\text{opt}} \geq 0$, $\kappa_{\text{M}} \in (0,\infty]$, $\kappa \in (0,1)$, $\bar{\kappa} \in [\kappa, 1)$, a weight $u > 0$.

Step 0 (Initialization). Set $k=0$, $\hat{y}^0 = y^0$, $\eta^0 = 0$, $f(\hat{y}^0)$ and $\widehat{\mathcal{W}}^0$.

Step 1 (Trial point finding). Set $\hat{y} = \hat{y}^k$, $\widehat{\mathcal{W}} = \widehat{\mathcal{W}}^k$, $\hat{\eta} = \eta^k$.

(a) Find $W^+ \in \operatorname{Argmax}_{W \in \widehat{\mathcal{W}}} \psi(W, \hat{\eta})$ (*v.* (11)) and set $y^{+\frac{1}{2}} = y_{W^+,\hat{\eta}}$ (*v.* (10)).

(b) Set $\eta^+ = \operatorname{argmax}_{\eta \geq 0} \psi(W^+, \eta)$ (*v.* (14)) and $y^+ = y_{W^+,\eta^+}$ (feasible by (15)).

(c) (Stopping criterion) If $f(\hat{y}) - f_{W^+,\eta^+}(y^+) \leq \varepsilon_{\text{opt}}(|f(\hat{y})| + 1)$, then STOP.

(d) If $f_{\widehat{\mathcal{W}},0}(y^+) - f_{W^+,\eta^+}(y^+) > \kappa_{\text{M}}[f(\hat{y}) - f_{W^+,\eta^+}(y^+)]$, then set $\hat{\eta} = \eta^+$ and go to (a).

(e) Set $y^{k+1} = y^+$, $W^{k+1} = W^+$, and $\eta^{k+1} = \eta^+$.

Step 2 (Descent test). Find $W_S^{k+1} \in \mathcal{W}$ such that either

(a) $f(\hat{y}^k) - f_{W_S^{k+1},0}(y^{k+1}) \leq \bar{\kappa}[f(\hat{y}^k) - f_{W^{k+1},\eta^{k+1}}(y^{k+1})]$, or

(b) $f_{W_S^{k+1},0}(y^{k+1}) = f(y^{k+1})$ and $f(\hat{y}^k) - f(y^{k+1}) \geq \kappa[f(\hat{y}^k) - f_{W^{k+1},\eta^{k+1}}(y^{k+1})]$.

In case (a), set $\hat{y}^{k+1} = \hat{y}^k$ (null step), otherwise set $\hat{y}^{k+1} = y^{k+1}$ (descent step).

Step 3 (Model updating). Choose a closed convex $\widehat{\mathcal{W}}^{k+1} \supset \left\{W^{k+1}, W_S^{k+1}\right\}$, *e.g.*, as in (8).

Step 4. Increase k by 1 and go to Step 2.

The following theorem is proven in [18].

Theorem 3.2 *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* $f(\hat{y}^k) \downarrow \inf f$.

A close inspection of the proof yields an important observation.

Lemma 3.3 *If* $\operatorname{Argmin} f \neq \emptyset$ *and Algorithm 3.1 does not stop, it generates a subsequence* $K \subseteq \mathbb{N}$ *satisfying* $\nabla f_{W^k,\eta^k} \xrightarrow{K} 0$ *and* $f_{W^k,\eta^k}(y^k) \xrightarrow{K} f(\bar{y})$ *with* $\bar{y} \in \operatorname{Argmin} f$.

Proof. First consider the case of a finite number of descent steps. Then an infinite number of inner iterations or null steps occurs starting with some iteration $\bar{k}$. Let the final stability center be $\hat{y} = \hat{y}^{\bar{k}}$. Then it is shown in [18, Lemma 3.2c)][1] and [18, Lemma 3.4] that $f_{W^k,\eta^k}(y^k) \to f(\hat{y})$ and $y^k \to \hat{y} \in \operatorname{Argmin} f$. By $y^k = y_{W^k,\eta^k}$ (*v.* (15)) and (10) we conclude that $\nabla f_{W^k,\eta^k} = b - \eta^k - \mathcal{A}W^k = u(\hat{y} - y^k) \to 0$.

In the case of an infinite number of descent steps, assumption $\operatorname{Argmin} f \neq \emptyset$ ensures condition [18, (3.17)] to hold. Then the last paragraph of the proof of [18, Lemma 3.5] establishes that the subsequence $K := \{k : \hat{y}^k = y^k\}$ of candidates yielding descent steps satisfies the desired properties. ■

This motivates the following lemma.

Lemma 3.4 *Let* $K \subseteq \mathbb{N}$ *be a subsequence of iterates satisfying* $\nabla f_{W^k,\eta^k} \xrightarrow{K} 0$ *and* $f_{W^k,\eta^k}(y^k) \xrightarrow{K} f(\bar{y})$ *with* $\bar{y} \in \operatorname{Argmin} f$. *Then all cluster points of* $(W^k, \eta^k)_{k \in K}$ *are optimal solutions of (PSDP).*

[1]Strictly speaking, the fact that [18, Lemma 3.2c)] holds for inexact evaluation in Step 2 requires a slightly different proof, see [15]. Alternatively, one may avoid additional inner iterations by setting $\kappa_M = \infty$.

Proof. By construction, $W^k \in \widehat{\mathcal{W}}^{k-1} \subseteq \mathcal{W}$ for all $k \geq 1$ and $\mathcal{W}$ is compact. Theorem 3.2 and $\operatorname{Argmin} f \neq \emptyset$ imply that the $\hat{y}^k$ remain bounded. Therefore the vectors $-u\hat{y}^k + b - \mathcal{A}W^k$ remain bounded. By (14) and (15) the same is true for the η^k and the y^k for all k. Furthermore, the η^k are nonnegative by (14). Compactness ensures that there is at least one cluster point for $(W^k, \eta^k)_{k\in K}$. Now let $(\bar{W}, \bar{\eta})$ be such a cluster point and $\bar{K} \subseteq K$ a corresponding subsequence with $(W^k, \eta^k) \xrightarrow{\bar{K}} (\bar{W}, \bar{\eta})$. Then $\bar{W} \in \mathcal{W}$, $\bar{\eta} \geq 0$, and (*v.* (5)) $\nabla f_{W^k,\eta^k} = b - \eta^k - \mathcal{A}W^k \xrightarrow{\bar{K}} b - \bar{\eta} - \mathcal{A}\bar{W} = 0$. Thus, $(\bar{W}, \bar{\eta})$ is feasible for (PSDP). Furthermore, $f_{W^k,\eta^k}(y^k) = \langle C, W^k \rangle + \langle b - \eta^k - \mathcal{A}W^k, y^k \rangle \xrightarrow{\bar{K}} \langle C, \bar{W} \rangle = f(\bar{y})$. Since $f(\bar{y})$ is an upper bound on the objective value of $(PSDP)$ this implies optimality. ■

Before stating the main theorem of this section we need one more result.

Lemma 3.5 *If Algorithm 3.1 terminates for* $\varepsilon_{\text{opt}} = 0$ *then the terminating* (W^+, η^+) *is an optimal solution of (PSDP).*

Proof. If the algorithm terminates, then $f(\hat{y}) = f_{W^+,\eta^+}(y^+)$ and [18, (2.22)] yields $\hat{y} = y^+$. So by (15) and (10) we conclude $b - \eta^+ - \mathcal{A}W^+ = 0$. Together with $W^+ \in \widehat{\mathcal{W}} \subseteq \mathcal{W}$ (12), $\eta^+ \geq 0$ (14), and $f(\hat{y}) = f_{W^+,\eta^+}(\hat{y}) = \langle C, W^+ \rangle$ (*v.* (5)) we obtain feasibility and optimality of (W^+, η^+) for (PSDP). ■

Theorem 3.6 *Assume* $\operatorname{Argmin} f \neq \emptyset$ *and let* $\varepsilon_{\text{opt}} = 0$. *If Algorithm 3.1 terminates then the terminating* (W^+, η^+) *is an optimal solution of (PSDP). If the algorithm does not terminate there is a subsequence* $K \subseteq \mathbb{N}$ *so that all cluster points of* $(W^k, \eta^k)_{k\in K}$ *are optimal solutions of (PSDP).*

Proof. If the algorithm terminates, then Lemma 3.5 applies. Otherwise the result follows from the previous two lemmas 3.3 and 3.4. ■

4 Extension to a cutting plane algorithm

In this section we extend Algorithm 3.1 to a cutting plane algorithm for optimizing over the intersection of $\mathcal{W}$ with a polyhedron $\{X : \mathcal{A}X \leq b\}$ that is given by a special type of separation oracle. Note, that within our setting there is no hope for polynomiality; our aim is to establish convergence. Encouraged by Theorem 3.6 we would like to separate with respect to W^+. Unfortunately, W^+ is never feasible unless it is optimal. Thus, one is faced with the problem that a separation oracle may return the same cut over and over again without disclosing any further information. In order to avoid this, we require the oracle to return a maximally violated inequality out of a finite set of inequalities describing the polyhedron (many actual separation routines satisfy this requirement).

Definition 4.1 *A separation oracle for a polyhedron* $P = \{x : Ax \leq b\}$ *with* $A \in \mathbb{R}^{m\times n}$ *and* $b \in \mathbb{R}^m$ *is called a* maximum violation oracle with respect to $Ax \leq b$ *if, for a given point* $\bar{x} \in \mathbb{R}^n$, *it either asserts that* $\bar{x} \in P$ *or returns an inequality* $A_{\bar{\imath},\cdot}x \leq b_{\bar{\imath}}$ *with* $b_{\bar{\imath}} - A_{\bar{\imath},\cdot}\bar{x} \leq \min_{i\in\{1,\ldots,m\}} b_i - A_{i,\cdot}\bar{x} < 0$ *(*$A_{i,\cdot}$ *refers to the ith row of matrix A).*

In the following we assume that the polyhedron is given by a maximum violation oracle with respect to $\mathcal{A}X \leq b$. A call to this oracle for a given point W^+ will be denoted by

$\mathcal{O}(W^+)$. For convenience we will refer to the inequality returned by the oracle by its index in $M = \{1, \dots, m\}$.

At any point in time, the algorithm will work with a subset of the constraints of $\mathcal{A}X \le b$. We will call this the *active index set* $J \subseteq M$ and denote the corresponding subsystem by $\mathcal{A}_J X \le b_J$. In particular, the feasible set corresponding to J is

$$\mathcal{P}_J = \{X \in \mathcal{W} : \mathcal{A}_J X \le b_J\} \tag{16}$$

and the corresponding optimization problem reads

$$(\mathrm{PSDP}_J) \qquad \max\ \langle C, X\rangle \text{ s.t. } X \in \mathcal{P}_J. \tag{17}$$

In the analysis of the algorithm we need to investigate the convergence of the slack variables η and dual variables y; so in choosing our notation we must take care that modifications of J do not affect their dimension. Therefore we regard them as elements of $\mathbb{R}^m$ with $y_j = \eta_j = 0$ for $j \notin J$. In particular, we will use $\mathbb{R}^m_J = \{y \in \mathbb{R}^m : y_j = 0\ \forall j \in M \setminus J\}$ and $Y_J = \{y \in \mathbb{R}^m_J : y \ge 0\}$. We define $[\cdot]_J : \mathbb{R}^m \to \mathbb{R}^m_J$ to extract the support on J, *i.e.*, for $y \in \mathbb{R}^m$ the vector $\bar{y} = [y]_J$ is defined by $\bar{y}_j = y_j$ for $j \in J$ and $\bar{y}_j = 0$ for $j \in M \setminus J$. Modifications of J also affect all formulas and functions involving $\mathcal{A}$ and b of the previous section. We will indicate this by a superscript J,

$$\begin{aligned}
f^J_{W,\eta}(y) &:= \langle C, W\rangle + \langle [b - \eta - \mathcal{A}W]_J, y\rangle && (W,\eta) \in \mathcal{W} \times \mathbb{R}^m_+ && (18)\\
f^J_{\widehat{\mathcal{W}},\widehat{Y}}(y) &:= \sup_{(W,\eta)\in\widehat{\mathcal{W}}\times\widehat{Y}_J} f^J_{W,\eta}(y) && \widehat{\mathcal{W}} \subseteq \mathcal{W}, \widehat{Y}_J \subseteq Y_J && (19)\\
f^J(y) &:= f^J_{\mathcal{W},Y_J}(y) && && (20)\\
y^J_{W,\eta} &:= \hat{y} - \frac{1}{u}\nabla f^J_{W,\eta} = \hat{y} - \frac{1}{u}[b - \eta - \mathcal{A}W]_J && (\hat{y} \in Y_J) && (21)\\
\psi^J(W,\eta) &:= \langle C, W\rangle + \langle [b - \eta - \mathcal{A}W]_J, \hat{y}\rangle - \frac{1}{2u}\|[b - \eta - \mathcal{A}W]_J\|^2 && && (22)\\
\eta^+ &:= \operatorname{argmax}\left\{\psi^J(W^+,\eta) : \eta \in Y_J\right\} = \max\{0, [-u\hat{y} + b - \mathcal{A}W^+]_J\} && && (23)\\
y^+ &:= y^J_{W^+,\eta^+} = \max\{[\hat{y} - \frac{1}{u}(b - \mathcal{A}W^+)]_J, 0\}, && \langle \eta^+, y^+\rangle = 0. && (24)
\end{aligned}$$

Now consider the following modification of Algorithm 3.1.

Algorithm 4.2
Input: $J^0 \subseteq M$, $y^0 \in Y_{J^0}$, $\varepsilon_{\mathrm{opt}} \ge 0$, $\kappa_{\mathrm{M}} \in (0, \infty]$, $\kappa \in (0,1)$, $\bar{\kappa} \in [\kappa, 1)$, a weight $u > 0$.

Step 0 (Initialization). Set $k = 0$, $\hat{y}^0 = y^0$, $\eta^0 = 0$, $\hat{J}^0 = J^0$, $f^{\hat{J}^0}(\hat{y}^0)$ and $\widehat{\mathcal{W}}^0$.

Step 1 (Trial point finding). Set $\hat{y} = \hat{y}^k$, $\widehat{\mathcal{W}} = \widehat{\mathcal{W}}^k$, $\hat{\eta} = \eta^k$, $\hat{J} = \hat{J}^k$.

(a) Find $W^+ \in \operatorname{Argmax}_{W\in\widehat{\mathcal{W}}} \psi^{\hat{J}}(W, \hat{\eta})$ (*v.* (22)) and set $y^{+\frac{1}{2}} = y^{\hat{J}}_{W^+,\hat{\eta}}$ (*v.* (21)).

(b) Call $\mathcal{O}(W^+)$. If it returns inequality $j \notin \hat{J}$, set $J^+ = \hat{J} \cup \{j\}$, $\hat{J} = J^+$ and go to (a).

(c) Set $\eta^+ = \operatorname{argmax}_{\eta\ge 0} \psi^{\hat{J}}(W^+, \eta)$ (*v.* (23)) and $y^+ = y^{\hat{J}}_{W^+,\eta^+}$ (feasible by (24)).

(e) (Stopping criterion) If $f^{\hat{J}}(\hat{y}) - f^{\hat{J}}_{W^+,\eta^+}(y^+) \le \varepsilon_{\mathrm{opt}}(|f^{\hat{J}}(\hat{y})| + 1)$, then STOP.

(f) If $f^{\widehat{J}}_{\widehat{W},0}(y^+) - f^{\widehat{J}}_{W^+,\eta^+}(y^+) > \kappa_{\mathrm{M}}[f^{\widehat{J}}(\hat{y}) - f^{\widehat{J}}_{W^+,\eta^+}(y^+)]$, then set $\hat{\eta} = \eta^+$ and go to (a).

(g) Set $y^{k+1} = y^+$, $W^{k+1} = W^+$, $\eta^{k+1} = \eta^+$, and $J^{k+1} = \widehat{J}$.

Step 2 (Descent test). Find $W_S^{k+1} \in \mathcal{W}$ such that either

(a) $f^{J^{k+1}}(\hat{y}^k) - f^{J^{k+1}}_{W_S^{k+1},0}(y^{k+1}) \leq \bar{\kappa}[f^{J^{k+1}}(\hat{y}^k) - f^{J^{k+1}}_{W^{k+1},\eta^{k+1}}(y^{k+1})]$, or

(b) $f^{J^{k+1}}_{W_S^{k+1},0}(y^{k+1}) = f^{J^{k+1}}(y^{k+1})$ and
$f^{J^{k+1}}(\hat{y}^k) - f^{J^{k+1}}(y^{k+1}) \geq \kappa[f^{J^{k+1}}(\hat{y}^k) - f^{J^{k+1}}_{W^{k+1},\eta^{k+1}}(y^{k+1})]$.

In case (a), set $\hat{y}^{k+1} = \hat{y}^k$, $\widehat{J}^{k+1} = J^{k+1}$ (null step),
otherwise set $\hat{y}^{k+1} = y^{k+1}$, $\widehat{J}^{k+1} = \{j \in J^{k+1} : \eta_j = 0\}$ (descent step).

Step 3 (Model updating). Choose a closed convex $\widehat{\mathcal{W}}^{k+1} \supset \left\{W^{k+1}, W_S^{k+1}\right\}$.

Step 4. Increase k by 1 and go to Step 2.

Remarks 4.3 (i) In each execution of Step 1, the active index set $\widehat{J}$ is enlarged in substep (b) at most $|M|$ times, so there is no danger that an infinite loop is caused by Step 1b). In addition, if execution passes on to Step 1c) then the maximum violation oracle ensures

$$\min_{j \in M \setminus \widehat{J}} b_j - \mathcal{A}_j W^+ \geq \min_{j \in \widehat{J}} b_j - \mathcal{A}_j W^+. \tag{25}$$

(ii) Throughout the inner loop in Step 1 the active index set $\widehat{J}$ satisfies $\widehat{J}^k \subseteq \widehat{J} \subseteq J^{k+1}$. Since $\hat{y}^k \in Y_{\widehat{J}^k} \subseteq Y_{\widehat{J}} \subseteq Y_{J^{k+1}}$ with $\hat{y}^k_j = 0$ for all $j \in J^{k+1} \setminus \widehat{J}^k$ we obtain

$$f^{\widehat{J}^k}(\hat{y}^k) = f^{\widehat{J}}(\hat{y}) = f^{J^{k+1}}(\hat{y}). \tag{26}$$

and $y^+ \in Y_{\widehat{J}}$ (*v.* (24) with $J = \widehat{J}$).

(iii) The reduction of J^{k+1} in the descent step 2b) amounts to deleting all constraints $j \in J^{k+1}$ with $\eta_j^{k+1} > 0$ (the inactive constraints with positive slack). By complementarity (15), the corresponding coordinates of y^{k+1} are zero, $y_j^{k+1} = 0$ for $j \in \widehat{J}^{k+1} \setminus J^{k+1}$. Therefore, $\hat{y}^{k+1} = y^{k+1} \in \widehat{J}^{k+1}$ and by (20) and (18)

$$f^{\widehat{J}^{k+1}}(\hat{y}^{k+1}) = f^{J^{k+1}}(y^{k+1}) \quad \text{and} \quad f^{\widehat{J}^{k+1}}_{W^{k+1},\eta^{k+1}}(\hat{y}^{k+1}) = f^{J^{k+1}}_{W^{k+1},\eta^{k+1}}(y^{k+1}). \tag{27}$$

(iv) Updating and solving the model (*v.* (8) and (22)) after increasing $\widehat{J}$ in step 1b) is not an issue as long as the information stored about $\overline{W}$ allows to compute $\langle A_j, \overline{W} \rangle$ for all $j \in \widehat{J}$. If this is not possible, the information of the aggregate is lost for the model and $\overline{W}$ or $\mathcal{A}\overline{W}$ has to be rebuilt from scratch in the following iterations. This, however, is no obstacle for convergence.

For the proof of convergence we assume $\varepsilon_{\mathrm{opt}} = 0$ for the rest of this section. Some steps of the proof rely directly on [18].

Lemma 4.4 *If Algorithm 4.2 stops for some finite $\bar{k}$, then $\hat{y}^{\bar{k}} \in$* Argmin *f^M and W^+ is an optimal solution of (PSDP$_M$).*

Proof. If Algorithm 4.2 stops, then Lemma 3.5 and its proof apply for $f^{\widehat{J}}$, so W^+ is a (feasible) optimal solution of (PSDP$_{\widehat{J}}$) and

$$\langle C, W^+ \rangle = f^{\widehat{J}}(\hat{y}^{\bar{k}}) = \min_y f^{\widehat{J}}. \tag{28}$$

Because the last call to the oracle in Step 1b) for this W^+ did not yield new violated inequalities, we conclude from (25) that W^+ is feasible for $\mathcal{P}_M$ of (16). By (28), $\langle C, W^+ \rangle = \min_y f^{\widehat{J}} = \min_{y \in Y_{\widehat{J}}} f^M \geq \min_y f^M \geq \langle C, W^+ \rangle$ where the last inequality follows from the feasibility of W^+ for $\mathcal{P}_M$ and the strong duality theorem for semidefinite programming. ■

We may now concentrate on the case that the algorithm does not stop.

Lemma 4.5 *Suppose that at iteration $\bar{k}$ an infinite loop occurs in Step 1 of Algorithm 4.2. Then $\hat{y}^{\bar{k}} \in \operatorname{Argmin} f^M$ and all cluster points of the W^+ are optimal solutions of (PSDP$_M$).*

Proof. In an infinite loop in Step 1, the active index set $\widehat{J}$ must reach its maximal size in step 1c) after finitely many subiterations. From then on [18, Lemma 3.2 c)] applies for $f^{\widehat{J}}$ yielding $\hat{y}^{\bar{k}} \in \operatorname{Argmin} f^{\widehat{J}}$. By Theorem 3.6 all cluster points of W^+ converge to optimal solutions of (PSDP$_{\widehat{J}}$). Let $\bar{W}$ be such a cluster point (existence follows from the compactness of $\mathcal{W}$). Then, for arbitrary $\varepsilon > 0$ there is a W^+ satisfying $\|W^+ - \bar{W}\| < \varepsilon$ and $b_j - \mathcal{A}_j W^+ > -\varepsilon$ for all $j \in \widehat{J}$. Therefore, by (25) $b_j - \mathcal{A}_j W^+ > -\varepsilon$ for all $j \in M$; so $\bar{W}$ is feasible for $\mathcal{P}_M$ of (16). Optimality of $\bar{W}$ for (PSDP$_M$) may now be shown as in the proof of Lemma 4.4. ■

Lemma 4.6 *Suppose that after iteration $\bar{k}$ Algorithm 4.2 produces an infinite number of null steps. Then $\hat{y}^{\bar{k}} \in \operatorname{Argmin} f^M$ and all cluster points of the W^k are optimal solutions of (PSDP$_M$).*

Proof. Arguing as in the proof of Lemma 4.5, the maximal $\widehat{J}^k$ must be reached for some $\hat{k} \geq \bar{k}$. From then on [18, Lemma 3.4] applies for $f^{\widehat{J}^{\hat{k}}}$ and the proof is completed as for Lemma 4.5. ■

The proof for an infinite sequence of descent steps would be equally direct if we would not allow for the deletion of inequalities. The removal of inactive inequalities, however, is indispensable in practical applications. Unfortunately, the proof of [18, Lemma 3.5] breaks down in this setting, because the linear model $f^J_{W,\eta}$ is not necessarily a minorant of f^M for a proper subset $J \subset M$. To guarantee the boundedness of the $\hat{y}^k$ our proof needs the additional assumption that f^M is 0-coercive, *i.e.*, $f(y) \to \infty$ whenever $\|y\| \to \infty$. This is, *e.g.*, the case if there exists a strictly feasible X for (PSDP$_M$), *i.e.*, an $\bar{X} \succ 0$ satisfying $\bar{X} \in \mathcal{W}$ and $\mathcal{A}\bar{X} < b$.

Lemma 4.7 *If f^M is 0-coercive, then the $\hat{y}^k$ remain bounded and all cluster points are in $\operatorname{Argmin} f^M$. Furthermore, if the set $D := \{k : \hat{y}^{k+1} = y^{k+1}\}$ of descent iterations is infinite, all cluster points of the W^{k+1} for $k \in D$ are optimal solutions of (PSDP$_M$).*

Proof. By Lemma 4.6 we may concentrate on the case of an infinite number of descent iterations $D := \{k : \hat{y}^{k+1} = y^{k+1}\}$. Each iteration $k \in D$ satisfies the descent step criterion of Step 2a) and therefore, by (26) and (27),

$$0 \leq f^{\widehat{J}^k}(\hat{y}^k) - f^{\widehat{J}^{k+1}}_{W^{k+1},\eta^{k+1}}(\hat{y}^{k+1}) \leq \frac{1}{\kappa}[f^{\widehat{J}^k}(\hat{y}^k) - f^{\widehat{J}^{k+1}}(\hat{y}^{k+1})]. \tag{29}$$

Using the assumption that f^M is 0-coercive (as are then all f^J with $J \subseteq M$), there exists a minimizer $\tilde{y} \in \operatorname{Argmin} f^M$ and the $\hat{y}^k$ remain bounded, because the $f^{\hat{J}^k}(\hat{y}^k)$ are monotonically decreasing (*v.* (29)). Furthermore,

$$\begin{aligned}\sum_{k\in D}\left[f^{\hat{J}^k}(\hat{y}^k) - f^{\hat{J}^{k+1}}_{W^{k+1},\eta^{k+1}}(\hat{y}^{k+1})\right] &\le \tfrac{1}{\kappa}\sum_{k\in D}\left[f^{\hat{J}^k}(\hat{y}^k) - f^{\hat{J}^{k+1}}(\hat{y}^{k+1})\right] \\ &\le \tfrac{1}{\kappa}\left[f^{J^0}(\hat{y}^0) - f^M(\tilde{y})\right] < \infty\end{aligned} \tag{30}$$

and (29) imply

$$f^{\hat{J}^k}(\hat{y}^k) - f^{\hat{J}^{k+1}}_{W^{k+1},\eta^{k+1}}(\hat{y}^{k+1}) \xrightarrow{D} 0. \tag{31}$$

For $k \in D$, the gradient of $f^{J^{k+1}}_{W^{k+1},\eta^{k+1}}$ may be expressed by (21) (use $\hat{y}^{k+1} = y^{J^{k+1}}_{W^{k+1},\eta^{k+1}}$) as

$$\nabla f^{J^{k+1}}_{W^{k+1},\eta^{k+1}} = u(\hat{y}^k - \hat{y}^{k+1}). \tag{32}$$

This and $f^{J^{k+1}}_{W^{k+1},\eta^{k+1}}(\hat{y}^k) \le f^{J^{k+1}}(\hat{y}^k) = f^{\hat{J}^k}(\hat{y}^k)$ (use (20) and (26)) yields

$$f^{\hat{J}^k}(\hat{y}^k) \ge f^{J^{k+1}}_{W^{k+1},\eta^{k+1}}(\hat{y}^{k+1}) + \left\langle \nabla f^{J^{k+1}}_{W^{k+1},\eta^{k+1}}, \hat{y}^k - \hat{y}^{k+1}\right\rangle = f^{J^{k+1}}_{W^{k+1},\eta^{k+1}}(\hat{y}^{k+1}) + u\|\hat{y}^{k+1} - \hat{y}^k\|^2,$$

so by (27)

$$u\|\hat{y}^{k+1} - \hat{y}^k\|^2 \le f^{\hat{J}^k}(\hat{y}^k) - f^{\hat{J}^{k+1}}_{W^{k+1},\eta^{k+1}}(\hat{y}^{k+1}) \tag{33}$$

Combining (31)–(33) we obtain (*v.* (18))

$$\nabla f^{J^{k+1}}_{W^{k+1},\eta^{k+1}} = [b - \eta^{k+1} - \mathcal{A}W^{k+1}]_{J^{k+1}} \xrightarrow{D} 0. \tag{34}$$

Now let $\bar{W}$ be a cluster point of the W^{k+1} for $k \in D$ (existence follows from the compactness of $\mathcal{W}$). Then there is a subsequence $K \subseteq D$ with $W^{k+1} \xrightarrow{K} \bar{W}$ and (34) and (25) ensure the feasibility of $\bar{W}$ for $\mathcal{P}_M$ of (16). Furthermore the boundedness of the $\hat{y}^k$, (31), and $f^{J^{k+1}}_{W^{k+1},\eta^{k+1}}(\hat{y}^{k+1}) = \langle C, W^{k+1}\rangle + \left\langle \nabla f^{J^{k+1}}_{W^{k+1},\eta^{k+1}}, \hat{y}^{k+1}\right\rangle \xrightarrow{K} \langle C, \bar{W}\rangle = \inf_k f^{\hat{J}^k}(\hat{y}^k) \ge \inf f^M$ implies that $\bar{W}$ is an optimal solution of (PSDP$_M$) and that all cluster points $\bar{y}$ of the $\hat{y}^k$ satisfy $\bar{y} \in \operatorname{Argmin} f^M$. ∎

We may now state our main result.

Theorem 4.8 *Let $\mathcal{P}_M$ of* (16) *have a strictly feasible point. Then Algorithm 4.2 solves (PSDP$_M$).*

Proof. In the case of finitely many descent steps the proof follows from lemmas 4.4, 4.5, and 4.6. For infinitely many descent steps, the strict feasibility of $\mathcal{P}_M$ implies the 0-coercivity of f^M and so Lemma 4.7 completes the proof. ∎

Remark 4.9 *The strictly feasible point assumption could be dropped if the $\hat{y}^k$ remain bounded whenever there is a $\tilde{y} \in Y$ with $f^{\hat{J}^k}(\hat{y}^k) \ge f^M(\tilde{y})$ for all k (this assumption corresponds to [18, (3.16)]). We do not know whether this is true. Primal feasibility alone is not sufficient for boundedness, as can be seen from the well known example* max x_{12} s.t. $\begin{bmatrix} 1 & x_{12} \\ x_{12} & 0 \end{bmatrix} \succeq 0$.

5 Implementation

Algorithm 4.2 is convenient for theoretical investigations but has significant drawbacks in practice. The oracle has to be called for each subproblem solution, which is often computationally too expensive; the number of inequalities may grow enormously before the next descent step occurs; finally, the frequent changes in the model may slow down convergence.

Our implementation is tuned for semidefinite relaxations of quadratic $\{-1,1\}$ programming problems in the style of (SMC); we separate odd-cycle inequalities (2) exclusively. The code employs the C++ class library `SBmethod` [17] which implements Algorithm 3.1. We delete and add inequalities only after descent steps of Algorithm 3.1. In particular, the routines for deletion and separation are called, in this sequence, whenever the first ten descent steps have been completed and the condition $f(\hat{y}^k) - f_{W^{k+1},\eta^{k+1}}(y^{k+1}) \leq 5{\cdot}10^{-2}(|f(\hat{y}^k)| + 1)$ holds (here and in the following f refers to the relaxation currently in use). The code stops, when the stopping criterion of 1c) of Algorithm 3.1 is satisfied for the current relaxation with $\varepsilon_{\text{opt}} = 10^{-5}$ or when a given timelimit is reached.

The deletion routine runs as follows. Let $\bar{m}$ denote the number of constraints before deletion. We first determine the set $D \subset \{1,\dots,\bar{m}\}$ of cutting planes whose slack values η_j satisfy $\eta_j \geq 10^{-5}$. Then we delete the $|D| - \max\{|D|/4, \bar{m}/100\}$ constraints of D with largest slack. By this rather cautious strategy we hope to keep constraints that flip between being active and inactive; we do not bother about deleting inactive inequalities if their number is small in comparison to $\bar{m}$.

In the separation routine we will separate with respect to W^+, which in the following refers to the W^{k+1} that gave rise to the descent step preceding separation. Although W^+ has in theory the representation $W^+ = PV^+P^T + \alpha^+\overline{W}$ (*v.* (13)), `SBmethod` does not store $\overline{W}$ by default but only $\mathcal{A}\overline{W}$ and $\langle C, \overline{W}\rangle$. However, `SBmethod` supports updating $\overline{W}$ in dense form or on sparse support. Separating with respect to a given support of W^+ requires updating $\overline{W}$ on this support in each execution of Step 3 of Algorithm 3.1. For large support this may cause significant increase in computation time and memory consumption. In particular, for large n, say $n > 1000$, updating $\overline{W}$ in dense form is computationally too expensive. Therefore we concentrate on the sparse case.

For the separation routine we assume that $X = W^+$ is a sparse matrix being given by a weighted undirected graph $G = (V, E)$ with node set $V = \{1,\dots,n\}$, edge set $E \subset \{ij : i < j,\ i,j \in V\}$ and edge weights x_{ij} for $ij \in E$.

The separation routine for odd-cycle inequalities (2) employs the exact separation routine of [5] that uses shortest path computation in a graph having twice the number of nodes and four times the number of edges. In order to speed up this computation we make use of the well known trick to start the shortest path tree from both endpoints. In fact, due to symmetries in the graph both shortest path trees are identical and only one has to be computed. We first fix a random order of the nodes and then compute the shortest path for each node in this order. If a node is already covered by a newly separated violated odd-cycle inequality it is ignored in all subsequent shortest path computations of this call. For each starting node the shortest path computation is stopped if an odd-cycle inequality is found that is violated by at least 10^{-6} or if there is no such inequality containing this node.

The separation routine expects $-1 \leq x_{ij} \leq 1$. This would be guaranteed if W^+ satisfies $\operatorname{diag}(W^+) = e$ and $W^+ \succeq 0$. Unfortunately, only the latter is guaranteed by Algorithm 3.1; the deviation $\|e - \operatorname{diag}(W^+)\|$ may still be quite considerable. We enforce the box constraints by three different approaches,

$$\bar{x}_{ij} = w^+_{ij}/\max\{w^+_{ii} : i \in V\},\ \tilde{x}_{ij} = w^+_{ij}/\sqrt{w^+_{ii}w^+_{jj}},\ \hat{x}_{ij} = \max\{-1, \min\{w^+_{ij}, 1\}\} \quad \text{for } ij \in E.$$

and call the separation routine for all three, $\bar{X}$, $\tilde{X}$, and $\hat{X}$. We then normalize each new inequalities $\langle A, X\rangle \le b$ to $\|A\| = 1$ (A is a sparse symmetric matrix with Frobenius norm one). We add those that are not yet contained in the current description and are violated by at least 10^{-6} with respect to the original W^+.

It remains to specify the set E. `SBmethod` offers several types of coefficient matrices, one of them is `SYMMETRIC_SPARSE`. The cost matrix C is expected to be of this type. In the standard setting E is set to contain the union of the support of the cost matrix and all other constraint matrices of type `SYMMETRIC_SPARSE`. Since newly separated inequalities have their support within E, this choice makes sure that the cost of one matrix vector multiplication within the eigenvalue computation does not increase. Furthermore, because $\overline{W}$ is available on E, the inner product $\langle A, \overline{W}\rangle$ can be computed for all new inequalities $\langle A_i, \cdot\rangle\, X \le b_i$. Consequently $\mathcal{A}\widehat{W}$ is still available after the insertion of new constraints and there is no loss in the quality of the model. As starting values for the new y_i variables we choose $y_i = 0$ and, like in (26), the algorithm can continue without the need of recomputing any function values.

We will also consider a second setting where we modify E in the course of the algorithm. If $\overline{W}$ is updated on E then $W^+ = PV^+P^T + \alpha^+\overline{W}$ is not available in full; but if α^+ is small, then $W^+_{ij} \approx [PV^+P^T]_{ij}$ for $ij \notin E$. These approximations may be used in the search for edges that may be worth to add to E. Employing the separation procedure on the complete graph by including all approximations is computationally too expensive. As usual, one has to resort to heuristics. After experimenting with a few we have settled, for the time being, for the following routine which we call directly after the first separation step and then after each tenth separation step.

Start by setting E to the union of the support of all `SYMMETRIC_SPARSE` cost and constraint matrices currently contained in the description. For each node $\bar{\imath} \in V$ (in increasing order) we compute a shortest path tree on $G = (V, E)$ with respect to edge weights $x_{ij} = 1 - |w^+_{ij}/\sqrt{w^+_{ii}w^+_{jj}}|$. Each edge $\bar{\imath}j \notin E$ induces a cycle C_j with respect to this shortest path tree. For each such cycle C_j we find, with respect to the weight $x_{\bar{\imath}j} = [PV^+P^T]_{\bar{\imath}j}$, the "best" odd set $F \subseteq C_j$ (*v.* (2)) and add the edge $\bar{\imath}j$ to E that gives rise to the "most violated" odd cycle (even if the inequality is not violated). We also add the edge $\bar{\imath}j$ with $j \in \operatorname{argmin}\{[PV^+P^T]_{\bar{\imath}j} : \bar{\imath}j \notin E\}$ to E and continue with the next node. The idea is to add exactly two edges per node, one of them offering good possibilities for separation, the other providing support for difficult decisions. Edges that did not give rise to new violated inequalities in the following ten separation steps will be removed in reinitializing E in the next call to this routine. After adding edges to E, the old $\overline{W}$ must be reinitialized and rebuilt from scratch. This leads to a slight loss in the quality of the model but does not affect the warm start otherwise.

6 Computational Results

In order to illustrate the numerical behavior of the cutting plane approach we present preliminary results for a few max-cut instances on toroidal grid graphs and some bisection-instances of sparse KKT system matrices provided by Boeing. The experiments seem to indicate that the proposed cutting plane approach works well. Another independent issue of interest is the quality of the relaxations. To our astonishment, the results are rather discouraging for the pure max-cut examples, but appear to be quite promising for our bisection instances.

The numerical results were computed on a Linux-PC with two Intel Pentium III 800 MHz processors (256 KB Cache) and 1 GB of memory, but the code makes use of one processor only. CPU-time refers to the user time returned by the UNIX-routine `getrusage()`; measurement is started after completion of the input. We usually needed to run more than one process on the machine, which probably caused time measurement to be unreliable. Indeed, we observed significant deviations (up to 20% and more) for identical runs. Our computation times may therefore only serve as a rough guideline.

The performance of `SBmethod` strongly depends on numerous parameters (see the manual [17]). In order to make the runs more comparable for the different cutting plane settings, we have fixed these as far as possible to the same constant values for all instances. In particular we use a constant weight $u = 1$, parameters $\kappa = 0.1$, $\bar{\kappa} = 0.1$, $\kappa_M = 0.6$, $\varepsilon_{\text{opt}} = 10^{-5}$, we set the maximum number of columns to keep in the bundle P (*v.* (8)) to $n_K = 15$, the maximum number of columns to add to P to $n_A = 5$, and set a time limit of twenty hours.

The tables contain the following columns. *Problem* gives the name of the problem; n is the order of the matrix (the number of $\{-1, 1\}$ variables in the quadratic $\{-1, 1\}$ program); $\bar{m}$ gives the number of constraints at termination, *nz* displays the maximum number of elements in which $\overline{W}$ is updated during the run ($|E|$ of §5 plus diagonal elements); in *feas.* we report the best lower bound that we know of[2] or that we could generate by Goemans-Williamson rounding based on PV_+P^T combined with simple exchange heuristics; f_{term} is the value of the upper bound at termination; *time* lists the CPU-time in *hh:mm:ss*; k is the value of the iteration counter of Algorithm 3.1 at termination, *inner* gives the number of executions of Step 1a) of Algorithm 3.1 ($= k + 1+$ inner iterations); *desc.* displays the number of descent steps.

6.1 Max-cut on toroidal grid graphs

Large scale and sparse max-cut instances in pure form do not seem to appear frequently in practice. The only application we are aware of is in the computation of ground states for Ising spin glasses, see [23] and references therein. Experimentally it was observed that on instances over toroidal grid graphs the relaxation by odd-cycle inequalities yields bounds of very good quality. Linear programming approaches proved very successful on this class of problems [23, 28]. We present results for four instances of three dimensional toroidal grid graphs. Instances *toruspm3-8-50* and *toruspm3-8-50* have edge weights chosen uniformly form $\{-1, 1\}$, whereas the edge weights for *torusg3-8-50* and *torusg3-8-50* are taken according to the Gaussian normal distribution around zero. They are part of the the 7^{th} DIMACS challenge test set and turned out to be rather difficult to solve by the linear programming techniques.

Let A be the weighted adjacency matrix of the respective graph, then we set $C = \frac{1}{4}(\frac{e^TAe}{n}I - A)$ and use the elliptope (*v.* [9]) as initial semidefinite relaxation,

$$\max \langle C, X\rangle \text{ s.t. } \operatorname{diag}(X) = e,\ X \succeq 0. \qquad (35)$$

In Table 1 we first give the results for (35), then for separating odd-cycle inequalities with

[2]The values for *toruspm3-8-50*, *torusg3-8*, and *torusg3-15* were reported by Frauke Liers, personal communication, and 416.84814 was reported optimal for *torusg3-8*.

Table 1: Max-cut on toroidal grid graphs

Problem	n	$\bar{m}$	nz	feas.	f_{term}	time	k	inner	desc.
				elliptope without cuts					
toruspm3-8-50	512	512	0	456	527.8127	14	139	140	30
toruspm3-15-50	3375	3375	0	2944	3475.151	31:38	855	856	38
torusg3-8	512	512	0	416.84814	457.3618	19	201	202	39
torusg3-15	3375	3375	0	2841.96	3134.591	38:55	992	993	46
				elliptope with odd-cycles on support					
toruspm3-8-50	512	2790	2048	456	464.7413	20:00:01	73645	73675	67
toruspm3-15-50	3375	13822	13500	2944	3071.477	20:00:34	15932	15933	47
torusg3-8	512	2877	2048	416.84814	417.6862	20:00:01	50839	50870	64
torusg3-15	3375	13218	13500	2841.96	2879.676	20:00:36	15547	15549	46
				elliptope with odd-cycles on extended support					
toruspm3-8-50	512	3342	3910	456	464.8202	20:00:01	50303	51754	61
toruspm3-15-50	3375	15205	22323	2944	3073.534	20:01:00	16594	16866	44
torusg3-8	512	4113	4593	416.84814	417.7017	20:00:01	46787	48894	71
torusg3-15	3375	14496	22559	2841.96	2882.073	20:00:47	15057	15279	46

respect to the support of C as described in §5 and, finally, the results when including the heuristic for enlarging the support on every tenth call to the separation procedure (*v.* §5).

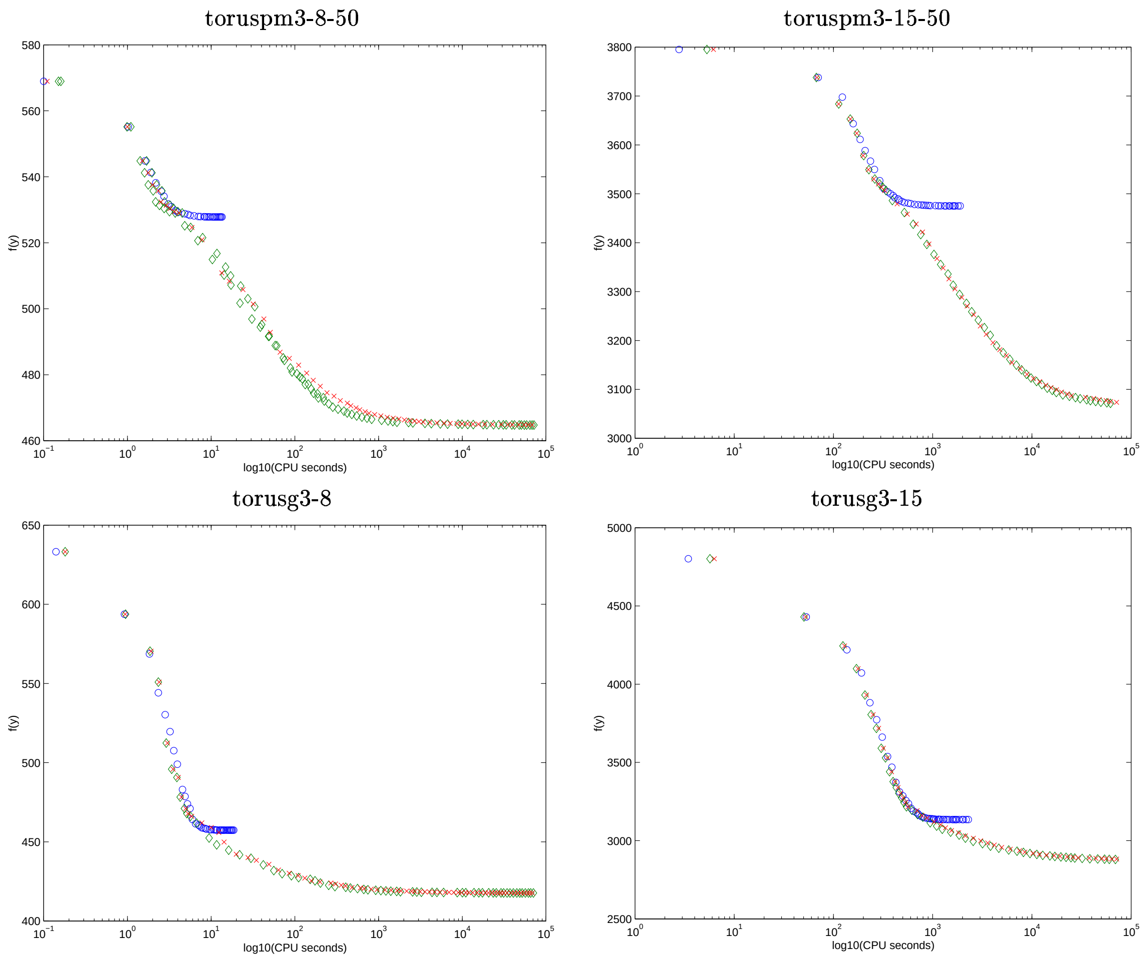
The improvement of the bound when including separation is considerable, but computation time reaches the limit of 20 hours in all cases. Listing the final values provides little insight into the development of the bound over time, therefore we also present plots in Table 2 that show this development with respect to a logarithmic time scale. It turns out, that the cutting plane approaches improve the bound considerably even before the bound on the elliptope converges. This shows that the warm start technique is very efficient. Enlarging the support does not seem to help at all for these instances, yet performance does not deteriorate much when employing it.

Even though the cutting plane approach is very successful in improving the basic semidefinite relaxation of these instances, the results are surprisingly poor in comparison to the linear programming bound that is based on separating only odd-cycles inequalities. Indeed, the values obtained by optimizing over the odd-cycle polytope alone are 464.7035 for *toruspm3-8-50*, <3063.757 for *toruspm3-15-50*, 417.6645 for *torusg3-8*, and <2873.773 for *torusg3-15*. What is more, the linear programming approach needs considerable less computation time. Theoretically, the bounds obtained by optimizing over the odd-cycle polytope cannot be better than the bounds obtained by intersecting the odd-cycle polytope with the elliptope. Therefore we have to blame this effect on the poor final convergence rate of the spectral bundle method. The experiments suggest that for toroidal grid graphs the improvement obtained by intersecting with the elliptope is only marginal (for smaller instances we have observed at least some improvement). This is astonishing in view of the strong theoretical results on the quality of the semidefinite bound [13, 10].

6.2 Graph bisection

The bisection instances were communicated to us by Sharon Filipowski from Boeing and stem from nested bisection approaches for solving sparse symmetric linear systems; standard bisection heuristics seem not to work well on these instances but no method was available to

Table 2: Max-cut on toroidal grid graphs; the horizontal axis gives CPU time in seconds in logarithmic scale, the vertical axis displays the upper bound; $\circ$ refers to the elliptope, $\diamond$ to separation on the support, $\times$ to separation on extended support.



judge the quality of the solutions produced. Each instance consists of a graph $G = (V, E)$ that represents the support structure of a sparse symmetric linear system. The task is to partition V into two sets $(S, V \setminus S)$ that differ in cardinality by at most $0.05 \cdot n$ so that the number of edges that have one endpoint in S and the other in $V \setminus S$ is minimized.

Let A denote the 0-1 adjacency matrix of G, then with $C = \frac{1}{4}(A - \frac{e^T Ae}{n} I)$ (note the change in sign to obtain a maximization problem) a canonical semidefinite relaxation (*v.* [30, 20, 24, 6]) reads

$$\max \langle C, X \rangle \text{ s.t. } \operatorname{diag}(X) = e,\ \langle ee^T, X \rangle \leq \lfloor 0.05 \cdot n \rfloor^2 .\ X \succeq 0. \tag{36}$$

In `SBmethod` the constraint matrix $A_{n+1} = ee^T$ can be represented in this structured form by the constraint class `GRAM_DENSE`. Therefore there is no need to work with a dense dual matrix and the matrix vector multiplication is still efficient. On these instances the aggregation mechanism often reduced the bundle size too much so that we forced the minimum number of columns to be kept in P to 10 by setting $n_{\min} = 10$ (see the manual [17]).

Table 3 lists our numerical results for the basic semidefinite relaxation, for the semidefinite relaxation combined with odd-cycle cutting planes on the support of C, and for separating odd-cycles inequalities on the dynamically enlarged support (*v.* §5). Plots of the progress of the algorithm with respect to a logarithmic time scale are displayed in tables 4 and 5. In order to obtain a better resolution of the relevant part we only show descent steps whose function value is below zero.

Quite contrary to the max-cut instances on toroidal grid graphs, the separation of odd-cycle inequalities on the support of C rarely yields significant improvement for our bisection instances, whereas enlarging the support seems to help a lot in most cases. We add some comments on the single instances. Using enlarged support allowed to prove optimality of the feasible $\{-1, 1\}$ solutions of *lowt01* and *putt01*. Extending the support also led to significant improvements of the bounds for *skwz02*, *orbe11_hc100*, and *heat02*. For *capt09* and, in particular, *plnt01* the gap between the best feasible solution known and the upper bound is still large. For the three large examples *trai27*, *lnts02*, and *traj33* the time limit was too short to reach definitive conclusions on the improvement; even the basic relaxation could not be solved to sufficient precision within this time span. Furthermore, for these three instances the computation time spent in the exact separation algorithm exceeds significantly the time spent in the spectral bundle code (*e.g.*, for *traj33* on extended support, 17 hours are due to separation!); the decrease in number of iterations and descent steps for the separation versions is mainly due to this effect and not to the increased work by updating a larger aggregate matrix $\overline{W}$. It should be noted, that the gap between feasible solution and bound of the basic relaxation is relatively small for *traj27*, *lnts02*, and *traj33*, so there is not much room for improvement.

The semidefinite relaxation for balanced bisection does not lend itself to comparison with a pure linear relaxation. Although there is a wealth of heuristics for graph bisection (*v.* [31] and references therein), we are not aware of any recent computational studies on bounds for this particular problem. In [12] a polyhedral approach is discussed for the more general node capacitated graph partitioning problem; computational results also include equipartition problems for graphs with up to 300 nodes and 500 edges. The original version of the code of [12] could not be run any longer, so we updated the code to work with CPLEX 7.1 [22] and tested it on the two small examples *lowt01* and *putt01* (the separation routines were obviously not designed for larger instances and far too slow for even *capt09*). We only computed the

Table 3: Bisection instances.

Problem	n	$\bar{m}$	nz	feas.	f_{term}	time	k	inner	desc.
relaxation (36) without cuts									
lowt01	82	83	0	-13	-4.541642	1	14	15	13
putt01	115	116	0	-28	-18.94562	1	17	18	15
capt09	2063	2064	0	-6	-0.6457718	34:37	360	361	199
skwz02	2117	2118	0	-567	-493.8798	1:50	73	74	37
orbe11_hc100	2186	2187	0	-2087	-1839.536	4:51	76	77	41
plnt01	2817	2818	0	-74	-4.334505	38:16	472	622	446
heat02	5150	5151	0	-150	-9.940345	16:43	379	388	63
traj27	17148	17149	0	-8174	-8140.877	20:01:13	1213	1214	96
lnts02	17990	17991	0	-6589	-6063.601	20:01:35	2181	2182	71
traj33	20006	20007	0	-9593	-9496.117	20:00:24	719	719	94
relaxation (36) with odd-cycles on support									
lowt01	82	179	342	-13	-4.542344	3	25	46	24
putt01	115	521	548	-28	-21.40906	16	106	188	84
capt09	2063	15613	12999	-6	-0.6586966	13:03:43	3999	11689	1369
skwz02	2117	46188	16118	-567	-506.2629	15:13:26	9430	9502	595
orbe11_hc100	2186	15860	40057	-2087	-1840.485	35:21	182	311	72
plnt01	2817	43907	27816	-74	-4.541399	20:00:56	3067	5765	793
heat02	5150	16796	25056	-150	-9.940555	1:22:02	552	990	64
traj27	17148	48005	129781	-8174	-8118.609	20:24:18	1190	1191	43
lnts02	17990	18900	63873	-6589	-6029.048	20:36:48	1384	1394	50
traj33	20006	64164	261953	-9593	-9460.441	20:40:23	529	530	40
relaxation (36) with odd-cycles on extended support									
lowt01	82	1230	1060	-13	-12.96362	25:17	6588	10505	101
putt01	115	1510	1482	-28	-27.99941	6:58	1887	3177	91
capt09	2063	14059	19527	-6	-4.142882	20:00:15	15332	23194	222
skwz02	2117	20347	28714	-567	-558.1742	20:00:31	19703	19718	191
orbe11_hc100	2186	20107	50048	-2087	-2033.389	20:00:55	10726	10753	168
plnt01	2817	30619	35477	-74	-8.023619	20:01:22	5011	25567	335
heat02	5150	18009	43739	-150	-144.1614	20:01:57	7964	8007	167
traj27	17148	26396	170827	-8174	-8105.315	20:33:10	1116	1116	32
lnts02	17990	21985	104148	-6589	-5961.739	20:55:52	831	833	36
traj33	20006	30269	306952	-9593	-9454.076	21:31:32	364	365	35

Table 4: Bisection instances; the horizontal axis gives CPU time in seconds in logarithmic scale, the vertical axis displays the upper bound; $\circ$ refers to relaxation (36), $\diamond$ to separation on the support, $\times$ to separation on extended support.

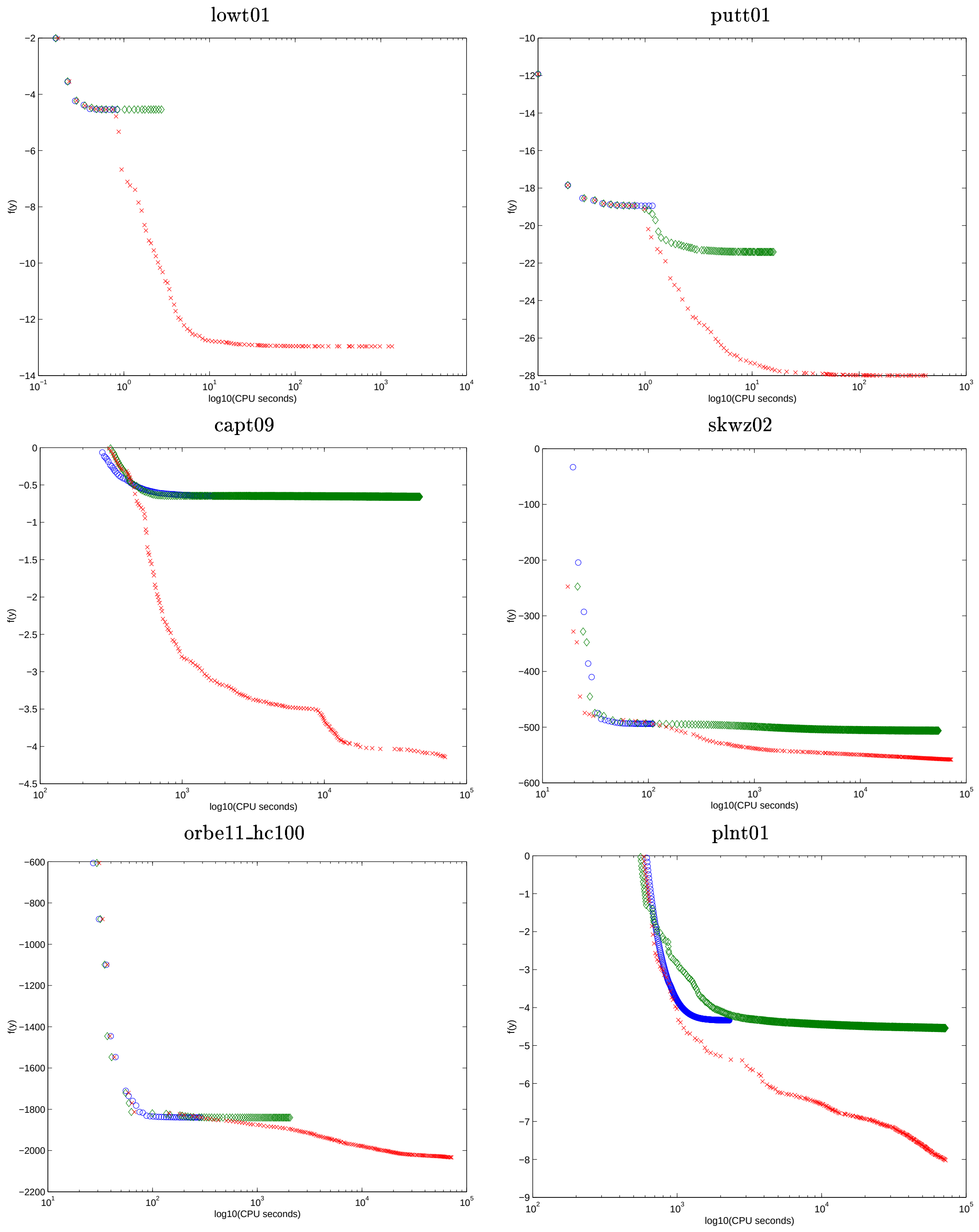
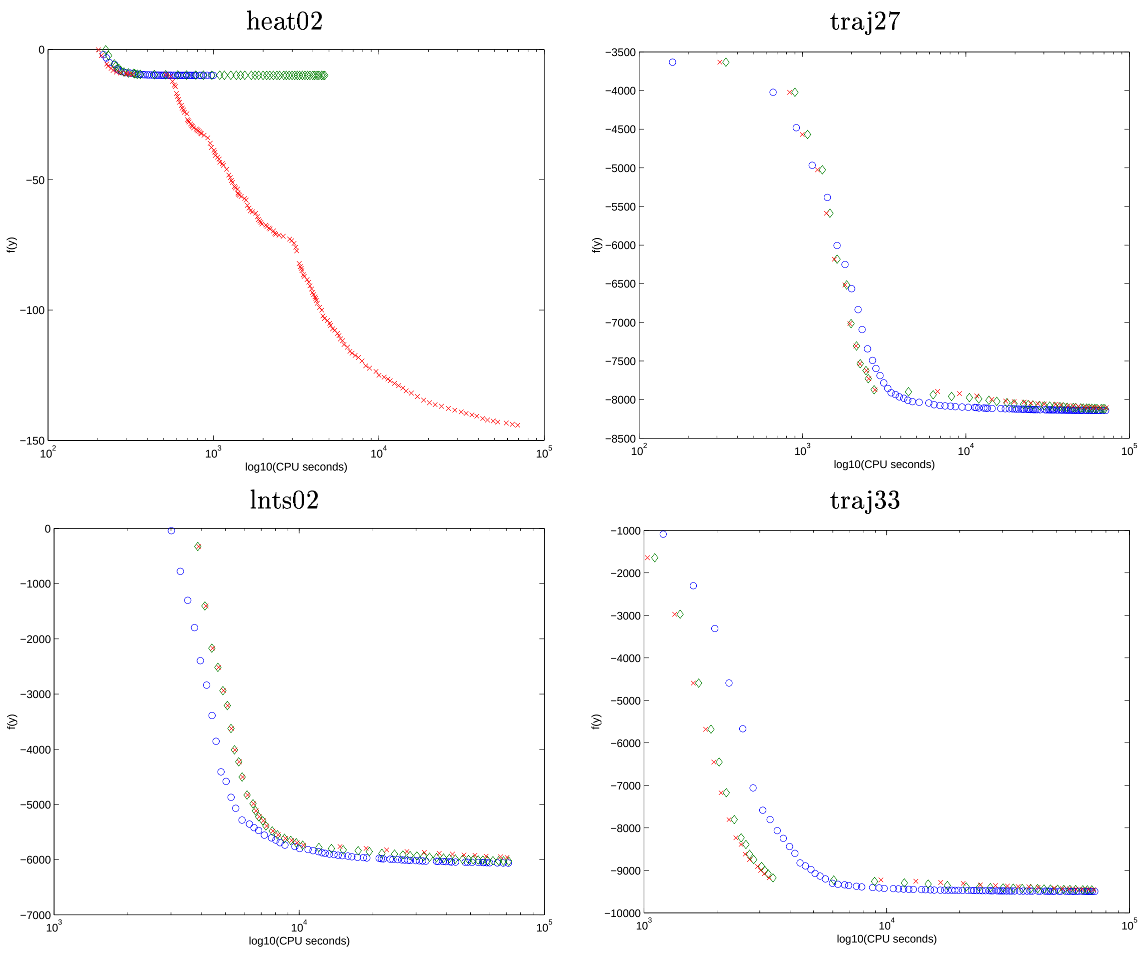


Table 5: Bisection instances; the horizontal axis gives CPU time in seconds in logarithmic scale, the vertical axis displays the upper bound; $\circ$ refers to relaxation (36), $\diamond$ to separation on the support, $\times$ to separation on extended support.



root node with all separation procedures switched on. The linear programming approach needed about twice the time to arrive at the same bound; about 95% of the total time were spent in the linear programming solver CPLEX 7.1. It is quite likely that the performance of the linear programming approach can be improved by finetuning it with respect to this class of instances. Yet we believe that this is sufficient evidence that the semidefinite cutting plane approach is competitive for these bisection problems.

Acknowledgment. I thank Mike Jünger and Frauke Liers for extensive discussions, comparative runs, and for making available their separation routines. I thank Sharon Filipowski for providing the bisection instances and for her constant interest in this work.

References

[1] E. Balas, S. Ceria, and G. Cornuejols. A lift-and-project cutting plane algorithm for mixed 0/1 programs. *Math. Programming*, 58:295–324, 1993.

[2] F. Barahona. The max–cut problem in graphs not contractible to K_5. *Oper. Res. Letters*, 2:107–111, 1983.

[3] F. Barahona. On cuts and matchings in planar graphs. *Math. Programming*, 60:53–68, 1993.

[4] F. Barahona and R. Anbil. The volume algorithm: Producing primal solutions with a subgradient method. *Math. Programming*, 87 A(3):385–399, 2000.

[5] F. Barahona and A. R. Mahjoub. On the cut polytope. *Math. Programming*, 36:157–173, 1986.

[6] S. Benson, Y. Ye, and X. Zhang. Mixed linear and semidefinite programming for combinatorial and quadratic optimization. *Optimization Methods and Software*, 11&12:515–544, 1999.

[7] H. Crowder, E. Johnson, and M. Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31:803–834, 1983.

[8] C. De Simone. The cut polytope and the boolean quadric polytope. *Discrete Mathematics*, 79:71–75, 1989.

[9] M. Deza and M. Laurent. *Geometry of Cuts and Metrics*, volume 15 of *Algorithms and Combinatorics*. Springer, 1997.

[10] U. Feige and G. Schechtman. On the optimality of the random hyperplane rounding technique for max cut. Manuscript, Department of Computer Science and Applied Mathematics, the Weizmann Institute, Rehovot 76100, Israel, Oct. 2000.

[11] S. Feltenmark and K. C. Kiwiel. Dual applications of proximal bundle methods, including Lagrangian relaxation of nonconvex problems. *SIAM J. Optim.*, 10(3):697–721, 2000.

[12] C. E. Ferreira, A. Martin, C. C. de Souza, R. Weismantel, and L. A. Wolsey. The node capacitated graph partitioning problem: a computational study. *Math. Programming*, 81(2):229–256, 1998.

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

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

[15] C. Helmberg. Fixing variables in semidefinite relaxations. *SIAM J. Matrix Anal. Appl.*, 21(3):952–969, 2000.

[16] C. Helmberg. 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, Oct. 2000.

[17] C. Helmberg. `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, Oct. 2000. URL: `http://www.zib.de/helmberg/SBmethod`.

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

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

[20] C. Helmberg, F. Rendl, R. J. Vanderbei, and H. Wolkowicz. An interior–point method for semidefinite programming. *SIAM J. Optim.*, 6(2):342–361, May 1996.

[21] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms II*, volume 306 of *Grundlehren der mathematischen Wissenschaften*. Springer, Berlin, Heidelberg, 1993.

[22] ILOG S.A., Gentilly, France. *ILOG CPLEX 7.1, User's Manual*, Mar. 2001. Information available at URL `http://www.ilog.com`.

[23] M. Jünger and G. Rinaldi. Relaxations of the max cut problem and computation of spin glass ground states. In P. Kischka, editor, *Proc. SOR '97*, pages 74–83, 1998.

[24] 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. Comput.

[25] M. Laurent, S. Poljak, and F. Rendl. Connections between semidefinite relaxations of the max-cut and stable set problems. *Math. Programming*, 77(2):225–246, 1997.

[26] A. S. Lewis and M. L. Overton. Eigenvalue optimization. *Acta Numerica*, pages 149–190, 1996.

[27] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM J. Optim.*, 1(2):166–190, May 1991.

[28] J. E. Mitchell. Computational experience with an interior point cutting plane algorithm. *SIAM J. Optim.*, 10(4):1212–1227, 2000.

[29] M. Padberg. The boolean quadric polytope: some characteristics, facets and relatives. *Math. Programming*, 45(1):139–172, Aug. 1989.

[30] S. Poljak and F. Rendl. Nonpolyhedral relaxations of graph-bisection problems. *SIAM J. Optim.*, 5(3):467–487, 1995.

[31] R. Preis. *Analyses and Design of Efficient Graph Partitioning Methods*. PhD thesis, Fachbereich Mathematik/Informatik, Universität Paderborn, Germany, July 2000.

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

[33] P. D. Seymour. Matroids and multicommodity flows,. *Europ. J. Combin.*, 2:257–290, 1981.

[34] H. Sherali and W. P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Disc. Math.*, 3(3):411–430, Aug. 1990.

[35] N. Z. Shor. Quadratic optimization problems. *Soviet Journal of Computer and Systems Sciences*, 25:1–11, 1987. Originally published in Tekhnicheskaya Kibernetika, No. 1, 1987, pp. 128-139.