Math. Prog. Comp. (2012) 4:275–306
DOI 10.1007/s12532-012-0040-5

FULL LENGTH PAPER

LP and SDP branch-and-cut algorithms for the minimum graph bisection problem: a computational comparison

Michael Armbruster · Marzena Fügenschuh · Christoph Helmberg · Alexander Martin

Received: 3 March 2011 / Accepted: 19 March 2012 / Published online: 15 April 2012

Abstract While semidefinite relaxations are known to deliver good approximations for combinatorial optimization problems like graph bisection, their practical scope is mostly associated with small dense instances. For large sparse instances, cutting plane techniques are considered the method of choice. These are also applicable for semidefinite relaxations via the spectral bundle method, which allows to exploit structural properties like sparsity. In order to evaluate the relative strengths of linear and semidefinite approaches for large sparse instances, we set up a common branch-and-cut framework for linear and semidefinite relaxations of the *minimum graph bisection problem*. It incorporates separation algorithms for valid inequalities of the bisection cut polytope described in a recent study by the authors. While the problem specific cuts help to strengthen the linear relaxation significantly, the semidefinite bound profits much more from separating the cycle inequalities of the cut polytope on a slightly enlarged support. Extensive numerical experiments show that this semidefinite branch-and-cut approach without problem specific cuts is a superior choice to the classical simplex approach exploiting bisection specific inequalities on a clear majority of our large sparse test instances from VLSI design and numerical optimization.

A conference version of this article appeared as [5].

M. Armbruster · C. Helmberg
Chemnitz University of Technology, Chemnitz, Germany

C. Helmberg
e-mail: helmberg@mathematik.tu-chemnitz.de

M. Fügenschuh (✉)
Beuth University of Applied Sciences, Berlin, Germany
e-mail: fuegenschuh@beuth-hochschule.de

A. Martin
University of Erlangen-Nuremberg, Erlangen-Nuremberg, Germany
e-mail: alexander.martin@math.uni-erlangen.de

Keywords Branch and cut algorithms · Cutting plane algorithms · Polyhedral combinatorics · Semidefinite programs · Graph bisection

Mathematics Subject Classification 90C05 · 90C06 · 90C10 · 90C22 · 90C27 · 90C57

1 Introduction

Let $G = (V, E)$ be an undirected graph with node set $V = \{1, \ldots, n\}$ and edge set $E \subseteq \{\{i, j\} : i, j \in V, i < j\}$. For an edge $\{i, j\}$ we will also write ij if there is no way of confusion. For given vertex weights $f_i \in \mathbb{N} \cup \{0\}$, $i \in V$, and edge costs $w_{ij} \in \mathbb{R}$, $ij \in E$, a partition of the vertex set V into two disjoint clusters S and $V \setminus S$ with sizes $f(S) := \sum_{i \in S} f_i \leq F$ and $f(V \setminus S) \leq F$, where $F \in \mathbb{N} \cap \left[\frac{1}{2} f(V), f(V)\right]$, is called a *bisection*. The *minimum bisection problem* (MB) asks for a bisection such that the total cost of edges in the *cut* $\delta(S) := \{ij \in E : i \in S \wedge j \in V \setminus S\}$ is minimal. The problem is known to be NP-hard [21]. The polytope associated with MB and studied in [6] is the *bisection cut polytope*

$$P_{\mathrm{B}} := \mathrm{conv}\{\, y \in \mathbb{R}^E : \; y = \chi^{\delta(S)},\; S \subseteq V,\; f(S) \leq F,\; f(V \setminus S) \leq F \,\},$$

where $\chi^{\delta(S)}$ is the incidence vector of the cut $\delta(S)$ with respect to the edge set E.

MB as well as P_{B} are related to other problems and polytopes already known in the literature. Obviously, the bisection cut polytope is contained in the *cut polytope* [7,12] $P_{\mathrm{C}} := \mathrm{conv}\left\{y \in \mathbb{R}^{|E|} : y = \chi^{\delta(S)}, S \subseteq V\right\}$. If $F = f(V)$ then MB is equivalent to the max-cut problem (using the negative cost function) and $P_{\mathrm{B}} = P_{\mathrm{C}}$. For $F = \lceil \frac{1}{2} |f(V)| \rceil$ MB is equivalent to the *equipartition problem* [11] and the bisection cut polytope equals the *equipartition polytope* [9,10,35]. Furthermore, MB is a special case of the minimum node capacitated graph partitioning problem (MNCGP) [17,18], where more than two clusters are available for the partition of the node set and each cluster has a common limited capacity. The objective in MNCGP is the same as in MB, *i. e.*, minimize the total cost of edges having endpoints in distinct clusters. Finally, we mention the *knapsack polytope* [41] $P_{\mathrm{K}} := \mathrm{conv}\{x \in \{0, 1\}^{|V|} : \sum_{v \in V} f_v x_v \leq F\}$, which plays a fundamental role in valid inequalities for P_{B}. Graph partitioning problems in general have numerous applications, for instance in numerics [22], VLSI-design [36], compiler-design [32] and frequency assignment [16]. A large variety of valid inequalities for the polytope associated with MNCGP is known [7,9,10,17,35] and, since MB is a special case of MNCGP, all those inequalities are also valid for P_{B}.

The emphasis of this work is on investigating the computational possibilities and scope offered by basic linear and semidefinite relaxations in combination with polyhedral results for large sparse graphs arising in real-world applications. In particular, the three main objectives are to develop a competitive sparse semidefinite branch-and-cut approach suitable for these requirements, to explore the usefulness of the valid inequalities for P_{B} given in [6] in combination with linear and semidefinite relaxations, and to compare the performance of semidefinite and linear branch-and-cut approaches within a uniform setting.

The basic linear relaxation is a slightly improved version of the best formulation within the study [17,18], which has the additional advantage to allow the use of the same separation routines for semidefinite and linear relaxations. Indeed, the basic linear relaxation can be designed to work directly with the edge formulation of P_{B} by assuming, without loss of generality (w.l.o.g.), that G contains a star $K_{1,n-1}$ with central node $1 \in V$ by adding edges $1j$ of cost zero, if necessary. The edges of this star directly serve as the usual binary node variables indicating to which subset of the partition each node belongs to. Because the cycle inequalities [7] together with integrality describe all cuts in G, our initial integer linear program is

$$\begin{aligned}
&\text{minimize } w^T y \\
&\text{subject to } \sum_{i=2}^{n} f_i y_{1i} \leq F, \\
&\qquad f_1 + \sum_{i=2}^{n} f_i (1 - y_{1i}) \leq F, \\
&\qquad \sum_{ij \in D} y_{ij} - \sum_{ij \in C \setminus D} y_{ij} \leq |D| - 1, \\
&\qquad D \subseteq C \subseteq E,\ |D| \text{ odd},\ C \text{ cycle in G}, \\
&\qquad y \in \{0,1\}^E.
\end{aligned} \tag{1}$$

The semidefinite relaxation is based on [39], which is also the basis of the famous approximation algorithms for max-cut of Goemans and Williamson [23] and for equipartition [19]. For its formulation, consider $x \in \{-1,1\}^V$ as indicator vector for the partition of the node set where $x_i = x_j$ if and only if the nodes i and j are on the same side of the cut. Then a feasible bisection vector x of MB satisfies $|f^T x| \leq 2F - f(V)$ and the cut vector induced by this bisection is $y_{ij} = \frac{1}{2}(1 - x_i x_j)$, *i. e.*, it has value 0 if $ij \in E$ is not in the cut and value 1 if ij is in the cut. Let $L \in \mathbb{R}^{V \times V}$, $L_{ii} = \sum_{ij \in E} w_{ij}$ for $i \in V$, $L_{ij} = -w_{ij}$ for $ij, ji \in E$ and $L_{ij} = 0$ otherwise, be the weighted Laplacian of G. The canonical semidefinite relaxation of MB is obtained by replacing xx^T by a symmetric, positive semidefinite matrix $X \succeq 0$ having its diagonal equal to the vector of all ones,

$$\begin{aligned}
&\text{minimize } \left\langle \tfrac{1}{4} L, X \right\rangle \\
&\text{subject to } \operatorname{diag}(X) = \mathbf{1}, \\
&\qquad \left\langle f f^T, X \right\rangle \leq [2F - f(V)]^2, \\
&\qquad X \succeq 0.
\end{aligned} \tag{2}$$

Here and in the following $\langle \cdot, \cdot \rangle$ denotes the canonical matrix inner product, *i. e.*, for matrices $A, B \in \mathbb{R}^{m \times n}$, $\langle A, B \rangle := \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} B_{ij}$. Interpreting X_{ij} as $x_i x_j$ the linear relation

$$y_{ij} = \frac{1 - X_{ij}}{2}$$

allows to go back and forth between $\{0, 1\}$ and $\{-1, 1\}$ representations of the cuts without loss of sparsity or structural properties like low rank representations (see, e.g., [24]). In particular, any valid inequality $a^T y \leq \beta$ for P_B is easily translated into an equally sparse matrix constraint $\langle A, X\rangle \leq \beta$. While the primal semidefinite variable X is not sparse, its dual matrix variable in the dual semidefinite program to (2) inherits all structural properties of the cost and constraint matrices. The spectral bundle method [30] was designed to exploit these structural properties in solving the dual. It does not need the primal X and is thus able to handle large sparse instances. Due to its good warm start possibilities, the spectral bundle method allows an efficient primal cutting plane approach [26]; the high quality of the bounds obtained there by improving relaxation (2) with cycle inequalities encouraged this work. It was also observed in [26] that these bounds improve sometimes considerably, if the cycle inequalities are not only separated with respect to the elements X_{ij} for $ij \in E$ but with respect to a slightly enlarged support as might be obtained by adding a few appropriate virtual edges of weight 0. As this *extended support* relaxation includes only one bisection specific inequality, namely $\langle ff^T, X\rangle \leq [2F - f(V)]^2$, it is one of the major surprise outcomes of this work that on our instances the new bisection specific cuts did *not* yield significant further improvement, quite in contrast to the strong improvements in the linear case, see Sect. 5.

Our instances for comparing linear and semidefinite branch-and-cut approaches are typically too large to be solved exactly, therefore we will mostly consider the size of the gap achieved after a given time limit. In the majority of the cases the semidefinite approach is the clear winner, not always because of better bounds but quite often also because of better solutions. While our implementation does not keep the full matrix X, the spectral bundle method allows to maintain a low rank approximation of X that is a good basis for efficient random hyperplane rounding methods of the Goemans'–Williamson's type, see Sect. 3. Almost as important as these are the possibilities offered by X to develop new criteria for branching decisions (Sect. 4).

A recent successful study of a combined semidefinite polyhedral branch-and-cut approach for max-cut is [40]. It considers triangle inequalities exclusively and uses a polyhedral bundle method with a semidefinite oracle that solves a dense semidefinite max-cut-relaxation by interior point methods. Their approach is suitable for rather dense graphs with up to 400 nodes, while ours is designed to be applicable for sparse graphs with up to 2000 nodes.

Other relevant computational approaches for graph partitioning include experiments with interior point based cutting plane approaches, see e.g. [14,38] for pure linear programming and [15,29] for semidefinite approaches. The main issue there is to develop efficient warm start procedures after the addition of cutting planes. This requires considerably different algorithmic techniques in comparison to the approaches employed here and is beyond the scope of this study.

The paper is structured as follows. In [6] we gave a detailed analysis of P_B including several classes of new and facet-defining inequalities. We briefly summarize these results and those from the literature in Sect. 2 and sketch our separation procedures. Section 3 provides the necessary background of the sparse semidefinite approach of [26] for improving (2) by cutting planes with support extension and describes the semidefinite rounding heuristics employed. Aspects concerning the incorporation of this

semidefinite cutting plane approach in a branching scheme are treated in Sect. 4. The LP-based and the semidefinite branch-and-cut algorithm were implemented within the same framework SCIP [1] and use the same separation machinery. Numerical results for the linear and semidefinite root node relaxations are given in Sect. 5. The branch-and-cut results are discussed in Sect. 6. Conclusions and some open problems will complete the paper in Sect. 7. This work is an outgrowth of the dissertations [4,20], which contain and investigate several further approaches and ideas for efficient implementation of linear and semidefinite branch-and-cut approaches in great depth. Codes and test instances are available from the authors on request.

2 Valid inequalities for P_B and their separation

A large variety of valid inequalities for the cut polytope, the equipartition polytope, and the polytope associated with MNCGP is known: cycle inequalities [7] of the cut polytope; tree, star, and cycle inequalities [9,10] as well as suspended tree and path block cycle inequalities [12,35] for the equipartition polytope; tree, star, cycle with ear, cycle with tails, and knapsack tree inequalities [17] valid for the polytope associated with MNCGP. Since MB is a special case of MNCGP and $P_B \subseteq P_C$ the bisection cut polytope inherits most of the valid inequalities mentioned above.

Exemplarily we cite the cycle inequalities which we will later use in both models for MB. Let the subgraph $C = (V_C, E_C)$ be a cycle in G. Let D be a subset of E_C such that $|D|$ is odd. Then the *cycle inequality*

$$\sum_{e \in D} y_e - \sum_{e \in E_C \setminus D} y_e \leq |D| - 1 \tag{3}$$

is a valid inequality for the cut polytope P_C. For separation we employ the exact polynomial approach of [7] that searches for shortest paths in an auxiliary graph constructed from two copies of the original graph with a total of $4|E|$ edges.

The cut structure implies that whenever there is a walk between two nodes of the graph with an even number of edges in the cut, the two end-nodes of the walk have to be in the same cluster. So, given a special root node r, a walk P_{rv} in G to some node v, an edge subset $H_v \subseteq P_{rv}$ of even cardinality, and an incidence vector y of a cut, if the term

$$1 - \sum_{e \in P_{rv} \setminus H_v} y_e - \sum_{e \in H_v} (1 - y_e) \tag{4}$$

evaluates to one then r and v are on the same side of the cut; for nodes in opposite clusters, it is at most zero. In [6] this is used to set up an inequality linking the cut structure and the capacity constraint on the node weights.

Proposition 1 (bisection knapsack walk inequality [6]) *Let* $\sum_{v \in V} a_v x_v \leq a_0$ *be a valid inequality for the knapsack polytope* P_K *with* $a_v \geq 0$ *for all* $v \in V$. *For a subset* $V' \subseteq V$, *a fixed root node* $r \in V'$, *walks* $P_{rv} \subseteq E$, *and sets* $H_v \subseteq P_{rv}$ *with* $|H_v|$ *even, the bisection knapsack walk inequality*

$$\sum_{v\in V'} a_v \left(1 - \sum_{e\in P_{rv}\setminus H_v} y_e - \sum_{e\in H_v} (1-y_e)\right) \le a_0. \tag{5}$$

is valid for the polytope P_{B}.

Given a root node r and a vector $y \in [0,1]^{|E|}$ the optimal walks P_{rv} and subsets H_v maximizing (4) can be found in polynomial time with an algorithm that follows the one for separating cycle inequalities, see [6].

The *knapsack tree inequalities* of [17] form a special case, where the walks P_{rv} are taken from a tree (T, E_T) of G rooted at r and $H_v = \emptyset$ for all $v \in V'$,

$$\sum_{v\in T} a_v \left(1 - \sum_{e\in P_{rv}} y_e\right) \le a_0. \tag{6}$$

Following [17], one may strengthen the coefficients of (6) to

$$\sum_{e\in E_T} \min\left\{\sum_{v:e\in P_{rv}} a_v,\ \sum_{v\in T} a_v - a_0\right\} y_e \ge \sum_{v\in T} a_v - a_0, \tag{7}$$

we call this a *truncated knapsack tree inequality*. A less obvious strengthening exploits the dependence of the coefficients in (7) on the choice of the root node, which we express by the notation

$$\alpha_0 := \sum_{v\in T} a_v - a_0, \qquad \alpha_e^r := \min\left\{\sum_{v:e\in P_{rv}} a_v,\ \alpha_0\right\},\ e \in E_T,$$

The strongest form is achieved if r enforces a sort of balance with respect to the cumulated node weights on the paths to r.

Theorem 2 [6] *Let* (T, E_T) *be a tree in* G. *The strongest truncated knapsack tree inequality, with respect to the knapsack inequality* $\sum_{v\in V} a_v x_v \le a_0$, *defined on* (T, E_T) *is obtained for a root* $r \in \mathcal{R} := \mathrm{Argmin}_{v\in T} \sum_{e\in E_T} \alpha_e^v$. *That is, if* $r \in \mathcal{R}$ *then* $\sum_{e\in E_T} \alpha_e^s y_e \ge \sum_{e\in E_T} \alpha_e^r y_e \ge \alpha_0$ *holds for all* $s \in T$ *and all* $y \in P_{\mathrm{B}}$. *In particular,* $\sum_{e\in E_T} \alpha_e^r y_e = \sum_{e\in E_T} \alpha_e^s y_e$ *holds for all* $r, s \in \mathcal{R}$ *and all* $y \in P_{\mathrm{B}}$.

The elements of the set $\mathcal{R}$ are called *minimal roots* of a given tree (T, E_T), and by Theorem 2 all minimal roots of (T, E_T) deliver the same strongest truncated knapsack tree inequality. Additional structural results allow to locate minimal roots algorithmically at almost no cost. This strengthening proved highly effective in our experiments. Note, if the inequality induces a facet then r is a minimal root by Theorem 2. In some cases the minimal root condition is also sufficient. In order to state this result, call a path in (T, E_T) *branch-less*, if its inner nodes are all of degree 2 in the tree.

Theorem 3 [6] *Assume that* $G = (T, E_T)$ *is a tree rooted at a node* $r \in T$, $f_v = 1$ *for all* $v \in T$ *and* $\frac{1}{2}|T| + 1 \le F < |T|$. *The truncated knapsack tree inequality* $\sum_{e\in E} \min\{|\{v : e \in P_{rv}\}|,\ |V| - F\} \ge |V| - F$ *is facet-defining for* P_{B} *if and only if one of the following conditions is satisfied:*

(a) *r is a minimal root and (T, E_T) satisfies the so-called branch-less path condition: each branch-less path with F nodes has one end-edge that is a leaf in (T, E_T),*
(b) $F = |T| - 1$.

To motivate a strengthening for general bisection knapsack walk inequalities consider the case of a disconnected graph with two components, one of them being a single edge $\{u, v\}$, the other connected one being $V' = V \setminus \{u, v\}$. If $y_{uv} = 1$ then u and v belong to different clusters and therefore the capacity remaining for the clusters in V' (e.g., the right-hand side of (5)) can be reduced to $F - \min\{f_u, f_v\} y_{uv}$. To generalize this idea we define for $\bar{G} \subseteq G$ with $\bar{V} \subseteq V$, $\bar{E} \subseteq E(\bar{V})$ and $a \in \mathbb{R}_+^{|\bar{V}|}$ a function $\beta_{\bar{G}} : \{0, 1\}^{|\bar{E}|} \to \mathbb{R} \cup \{\infty\}$ with

$$\beta_{\bar{G}}(y) = \inf \left\{ a(S), a(\bar{V} \setminus S) : S \subseteq \bar{V}, \max\left\{a(S), a(\bar{V} \setminus S)\right\} \leq a_0, y = \chi^{\delta_{\bar{G}}(S)} \right\}.$$

Now we look at the convex envelope $\breve{\beta}_{\bar{G}} : \mathbb{R}^{|\bar{E}|} \to \mathbb{R} \cup \infty$ of $\beta_{\bar{G}}(y)$, *i. e.*,

$$\breve{\beta}_{\bar{G}}(y) = \sup \left\{ \breve{\beta}(y) : \breve{\beta} : \mathbb{R}^{|\bar{E}|} \to \mathbb{R},\ \breve{\beta} \text{ convex},\ \breve{\beta}(z) \leq \beta_{\bar{G}}(z) \text{ for } z \in \{0, 1\}^{|\bar{E}|} \right\}.$$

Note that $\breve{\beta}_{\bar{G}}$ is a piecewise linear function on its domain.

Proposition 4 (capacity reduced bisection knapsack walk inequality [6]) *Let $\sum_{v \in V} a_v x_v \leq a_0$ with $a_v \geq 0$ for all $v \in V$ be a valid inequality for the knapsack polytope P_{K}. Let V_0 be a non-empty subset of V and $r \in V_0$. Select subgraphs $(V_l, E_l) = G_l \subset G$ with pairwise disjoint sets V_l, $V_l \cap V_0 = \emptyset$ and $E_l \subseteq E(V_l)$ for $l = 1, \ldots, L$. Find for each l an affine minorant for the convex envelope $\breve{\beta}_{G_l}$ such that*

$$c_0^l + \sum_{e \in E_l} c_e y_e \leq \breve{\beta}_{G_l}(y) \tag{8}$$

holds for all y in P_{B}. Then the capacity reduced bisection knapsack walk inequality

$$\sum_{v \in V_0} a_v \left(1 - \sum_{e \in P_{rv} \setminus H_v} y_e - \sum_{e \in P_{rv} \cap H_v} (1 - y_e) \right) \leq a_0 - \sum_{l=1}^{L} (c_0^l + \sum_{e \in E_l} c_e y_e) \tag{9}$$

is valid for P_{B}.

In certain cases it is possible to establish a full description of $\breve{\beta}_{\bar{G}}$ via a complete description of the cluster weight polytope defined as follows. Given a graph $G = (V, E)$ with non-negative node weights $a_v \in \mathbb{R}$ for all $v \in V$. For a set $S \subseteq V$ we define $h(S) := (a(S), (\chi^{\delta(S)})^T)^T \in \mathbb{R}^{|E|+1}$. With respect to a given $a_0 \in \mathbb{R}$ we call

$$P_{\mathrm{CW}} = \mathrm{conv}\{\, h(S) : S \subseteq V,\ a(S) \leq a_0,\ a(V \setminus S) \leq a_0 \,\}$$

the *cluster weight polytope*.

In [6], a full description of $P_{\mathrm{CW}}(\bar{G})$ is given for the special case that the subgraph $\bar{G} = (\bar{V}, \bar{E})$ is a star centered at some node $r \in \bar{V}$, $a \geq 0$, and a_0 satisfies $a(\bar{V}) \leq a_0$.

In order to state the nontrivial facets of this case, assume $a\big(\bar{V} \setminus \{r\}\big) > a_r$ and call a triple $\big(V_p, \bar{v}, V_n\big)$ feasible if it fulfills $\bar{V} = \{r, \bar{v}\} \dot{\cup} V_p \dot{\cup} V_n$ and $a\big(V_p\big) \leq \frac{1}{2} a\big(\bar{V}\big) < a\big(V_p\big) + a_{\bar{v}}$. For all feasible triples $\big(V_p, \bar{v}, V_n\big)$ the inequalities

$$y_0 - \sum_{v \in V_p} a_v y_{rv} - \big(a(\bar{V}) - 2a(V_p) - a_{\bar{v}}\big)\, y_{r\bar{v}} + \sum_{v \in V_n} a_v y_{rv} \geq 0 \tag{10}$$

are the facet-inducing inequalities for $P_{\mathrm{CW}}(\bar{G})$. Thus, inequalities (10) form the best linear minorants (8) to be used in (9) in this case.

In our experiments this rather involved strengthening technique proved far less effective than the simple root strengthening for knapsack tree inequalities, but this may be due to the fact that the problems are defined to have nonnegative weights, and the knapsack constraints exploit that.

3 A sparse semidefinite cutting plane approach

We employ the semidefinite cutting plane approach of [26] which is based on the spectral bundle method [28,30]. For a brief sketch of the essentials, let $\mathcal{S}^n$ and $\mathcal{S}^n_+$ denote the space of symmetric matrices and symmetric positive semidefinite matrices of order n, let $C = -\frac{1}{4} L$ denote the cost matrix of (2) and collect all constraint matrices of (2) together with additional cutting planes in a linear operator $\mathcal{A} : \mathcal{S}^n \to \mathbb{R}^m$ with $\mathcal{A}X = [\langle A_i, X\rangle]_{i \in \{1,\ldots,m\}}$. It is convenient and computationally useful to require $\|A_i\| := \sqrt{\langle A_i, A_i\rangle} = 1$ for $i \in \{1, \ldots, m\}$. In order to keep notation simple we assume that all constraints are given as inequalities with a right-hand side vector $b \in \mathbb{R}^m$, then the negative of (2) reads

$$\max\{\langle C, X\rangle : \mathcal{A}X + \eta = b, X \in \mathcal{S}^n_+, \eta \in \mathbb{R}^m_+\}. \tag{11}$$

The dual eigenvalue formulation

Because of the constraints $\mathrm{diag}(X) = \mathbf{1}$, the trace of any feasible X is equal to n, *i. e.*, defining

$$\mathcal{W} := \{X \in \mathcal{S}^n_+ : \langle I, X\rangle = 1\} = \mathrm{conv}\{vv^T : \|v\| = 1, v \in \mathbb{R}^n\}$$

we have $\{X \in \mathcal{S}^n_+ : \mathcal{A}X \leq b\} \subseteq n\mathcal{W}$. Introducing Lagrange multipliers $\mu \in \mathbb{R}^m$, a slater point or compactness argument shows that there is no duality gap for the Lagrangian dual,

$$\begin{aligned} &\sup_{X \in n\mathcal{W}, \eta \in \mathbb{R}^m_+} \inf_{\mu \in \mathbb{R}^n} [\langle C, X\rangle + (b - \mathcal{A}X - \eta)^T \mu] \\ &= \inf_{\mu \in \mathbb{R}^n} \sup_{X \in n\mathcal{W}, \eta \in \mathbb{R}^m_+} \Big[\Big\langle C - \mathcal{A}^T y, X\Big\rangle + (b - \eta)^T \mu\Big]. \end{aligned}$$

The Rayleigh–Ritz characterization of the maximum eigenvalue yields

$$\lambda_{\max}(C - \mathcal{A}^T\mu) = \max_{\|v\|=1} v^T(C - \mathcal{A}^T\mu)v = \max_{\|v\|=1} \left\langle C - \mathcal{A}^T\mu, vv^T \right\rangle$$
$$= \max_{W \in \mathcal{W}} \left\langle C - \mathcal{A}^T\mu, W \right\rangle. \qquad (12)$$

Therefore the dual is equivalent to the eigenvalue optimization problem

$$\min_{\mu \in \mathbb{R}^m} \varphi(\mu) := n\lambda_{\max}(C - \mathcal{A}^T\mu) + \sup_{\eta \in \mathbb{R}^m_+} (b - \eta)^T\mu.$$

A semidefinite model of $\lambda_{\max}$

The nonsmooth convex function φ can be evaluated at some given $\mu \geq 0$ by computing the maximum eigenvalue $\lambda_{\max}$ of the symmetric matrix $C - \sum_{i=1}^m A_i\mu_i$. If this is done by an iterative method (the ConicBundle package [27] uses a Lanczos method), structural properties like low rank representations and sparsity can be exploited conveniently. An eigenvector v to $\lambda_{\max}$ with $\|v\| = 1$ generates a subgradient via $b - \mathcal{A}(nvv^T)$, but any other normalized vector v also generates a linear minorant to the dual function via $W = vv^T$ as in (12). The key idea of the spectral bundle method is to form a manageable minorant of the objective by accumulating (approximate) eigenvector information in a subset of $\mathcal{W}$ of the form

$$\widehat{\mathcal{W}} := \{PUP^T + \alpha\overline{W} : U \in S^r_+, \alpha \geq 0, \langle I, U\rangle + \alpha = 1\} \subseteq \mathcal{W},$$

where $P \in \mathbb{R}^{n\times r}$ consists of r orthonormal columns ($P^TP = I_r$) and $\overline{W} \in \mathcal{W}$. The columns of P are meant to form a basis of the r most important directions spanned by previous eigenvectors while $\overline{W}$ collects hopefully less important residual contributions; both are updated at every iteration of the bundle method.

The bundle subproblem and the primal aggregate

In iteration k of the bundle method a next multiplier candidate μ^{k+1} is determined in the vicinity of a current *stability center* $\hat{\mu}^k$ by approximating

$$\operatorname*{argmin}_{\mu \in R^n} \varphi_{\widehat{\mathcal{W}}^k}(\mu) + \tfrac{1}{2}\|\mu - \hat{\mu}^k\|^2,$$
$$\text{where} \quad \varphi_{\widehat{\mathcal{W}}^k}(\mu) := \max_{W \in \widehat{\mathcal{W}}^k} n\left\langle C - \mathcal{A}^T\mu, W \right\rangle + \sup_{\eta \in \mathbb{R}^m_+} (b - \eta)^T\mu. \qquad (13)$$

If the actual progress $\varphi(\hat{\mu}^k) - \varphi(\mu^{k+1})$ is sufficiently large in comparison to the progress predicted by $\varphi(\hat{\mu}^k) - \varphi_{\widehat{\mathcal{W}}^k}(\mu^{k+1})$, a *descent step* is made by setting $\hat{\mu}^{k+1} := \mu^{k+1}$. Otherwise, in a *null step*, the stability center is left unchanged, $\hat{\mu}^{k+1} := \hat{\mu}^k$, but the subgradient information obtained by evaluating $\varphi(\mu^{k+1})$ is used to improve

the new model $\widehat{\mathcal{W}}^{k+1}$. For computing the candidate μ^{k+1} the spectral bundle method solves – for some slack vector estimate $\hat{\eta}^k$ – a (small) convex quadratic semidefinite program of the form

$$\begin{array}{ll} \text{maximize} & -\frac{1}{2}\|b-\hat{\eta}^k-\mathcal{A}(nW)\|^2+\langle C-\mathcal{A}^T\hat{\mu}^k, nW\rangle+(b-\hat{\eta}^k)^T\hat{\mu}^k \\ \text{subject to} & W=P_kUP_k^T+\alpha\overline{W}_k, \\ & \langle I,U\rangle+\alpha=1, \\ & U\in\mathcal{S}_+^r,\alpha\geq 0. \end{array} \tag{14}$$

Let $W^k = P^kU_k(P^k)^T + \alpha_k\overline{W}^k$ be the computed optimal solution and v^{k+1} a normalized eigenvector to $\lambda_{\max}(C-\mathcal{A}^T\mu^{k+1})$, then the next P^{k+1} is set to hold a basis of v^{k+1} and the eigenvectors of $P^kU^k(P^k)^T$ belonging to large eigenvalues of U^k (this only requires an eigenvector factorization of the $r\times r$ matrix $U^k = Q^k\Lambda_{U^k}(Q^k)^T$), while $\overline{W}^{k+1}$ collects that part of W^k that is not spanned by P^{k+1}. In order to guarantee convergence, the choice $P^{k+1}=v^{k+1}$ and $\overline{W}^{k+1}=W^k$ would be sufficient, but for purposes explained below we prefer a rather rich P and require $r\leq 20$ only for efficiency reasons. It is important to note that for the spectral bundle algorithm there is no need to store $\overline{W}^k$, only $\langle C,\overline{W}^k\rangle$ and $\mathcal{A}\overline{W}^k$ are needed. For other purposes, however, it will be useful to store at least part of it, e.g., its elements $\overline{W}_{ij}$ with $ij\in E$ or even some elements on an enlarged support outside E. The computed solution $\bar{X}^k := nW^k \succeq 0$ of (14) is of central importance to the cutting plane algorithm. We will refer to $\bar{X}^k$ as the (*primal*) *aggregate*. The iteration index k is of little relevance in the considerations to follow and will be dropped most of the time.

Separation

Cluster points of appropriate subsequences of the primal aggregates are optimal solutions of (11), see [25]. Therefore the vector $\bar{y}\in\mathbb{R}^E$ with $\bar{y}_{ij}=\frac{1}{2}(1-\bar{X}_{ij})$ for $ij\in E$ should eventually be landed in P_{B}, so our separation routines work with respect to the aggregate $\bar{X}$. Unfortunately, $\bar{X}$ is only a rough approximation that is typically infeasible and does not satisfy even the current set of constraints $\mathcal{A}X\leq b$. One consequence is that $\bar{y}$ may not even satisfy the bound constraints, *i. e.*, in general $\bar{y}\notin[0,1]^E$ (the bounds hold whenever $\mathrm{diag}(\bar{X})=\mathbf{1}$). Thus, $\bar{y}$ is truncated or computed with respect to some scaled $\bar{X}$ so as to lie in $[0,1]^E$ before separation routines are called. In particular, for a given, possibly enlarged support $\hat{E}$ with $E\subseteq\hat{E}\subseteq\binom{V}{2}$ so that the values $\bar{X}_{ij}$ are available for $ij\in\hat{E}$, we separate with respect to the three variants (each specified over all $ij\in\hat{E}$)

$$\bar{y}_{ij}^{(1)} := \frac{1}{2}-\frac{1}{2}\frac{\bar{X}_{ij}}{\max\{\bar{X}_{ii}:i\in V\}}, \tag{15}$$

$$\bar{y}_{ij}^{(2)} := \frac{1}{2}-\frac{1}{2}\frac{\bar{X}_{ij}}{\sqrt{\bar{X}_{ii}\bar{X}_{jj}}}, \tag{16}$$

$$\bar{y}_{ij}^{(3)} := \frac{1}{2}-\frac{1}{2}\max\{-1,\min\{\bar{X}_{ij},1\}\}. \tag{17}$$

A second consequence is that separation routines may find the same inequalities repeatedly and some care has to be taken that the algorithm does not stall due to this property. It was shown in [26] that convergence of appropriate subsequences to optimal solutions can be guaranteed for such a cutting plane approach within the bundle framework if a maximum violation oracle is used for separation, *i. e.*, for a given finite set of inequalities and a given input point y the oracle always returns an inequality whose violation with respect to y is maximal (see [8] for an improved analysis on weaker oracles for general bundle methods). The result relies on the fact that violation of each individual inequality will be reduced sufficiently over time so that eventually all violated inequalities will be detected. In theory and in practice repeated separation of the same inequalities is typical and requires some additional work in order to eliminate duplicates. After the addition of newly separated cutting planes there is no difficulty in continuing the bundle method from the "same" Lagrange multiplier solution $\hat{\mu}$ by setting the new multipliers to zero as in Algorithm 1. Indeed, by linearity the new zero coordinates do not influence the value in the center and the values $\bar{a}_i$ allow to extend the old subgradients to the new coordinates (*i. e.*, the bundle subproblem (14) can be set up without any loss of information because $\mathcal{A}(nW)$ is directly computable). The $\bar{a}_i$ are available whenever the support of the new inequalities is restricted to the support $\hat{E}$ of $\overline{W}$. This holds for our separation routines and so there is no need to rebuild the bundle model in spite of the changes in dimension. After every descent step that satisfies the bundle method's termination criterion for a relative precision of 0.2 we delete cuts with sufficiently large slack and call the separation routines, see Fig. 2 on page 292 for a schematic overview and [4] for a detailed description. Fortunately, no dramatic scaling problems seem to arise after the separation process, maybe because in our current scheme violation of the inequalities and changes in the multipliers seem to converge to zero at a common speed.

Algorithm 1 Separate and Add New Cuts

Input: current aggregate $\bar{X}_{ij}$ with $\overline{W}_{ij}$ for $ij \in \hat{E} \supseteq E$, current constraints A_i, b_i with multipliers $\hat{\mu}_i$ $(i = 1, \ldots, m)$ and an upper bound $\bar{k} \in \mathbb{N}$ on new cuts.
Output: $A_i, b_i, \bar{a}_i = \langle A_i, \overline{W} \rangle$ and $\hat{\mu}_i$ for $i = m+1, \ldots, m'$ with $m \leq m' \leq m + \bar{k}$.
1: Call separation routines for $\bar{y}^{(1)}$, $\bar{y}^{(2)}$, $\bar{y}^{(3)}$ of (15)–(17) on support $\hat{E}$ resulting in $k \leq \bar{k}$ inequalities $\langle A_i, X \rangle \leq b_i$ $(i = m+1, \ldots, m+k)$ violated for $\bar{X}$.
2: Find and delete duplicates $\rightarrow$ new cuts $\langle A_i, X \rangle \leq b_i$ for $i = m+1, \ldots, m+k'$, $k' \leq k$.
3: **return** $m' \leftarrow m + k'$, A_i, b_i, $\bar{a}_i \leftarrow \langle A_i, \overline{W} \rangle$ and $\hat{\mu}_i \leftarrow 0$ for $i = m+1, \ldots, m'$.

Support extension

Because the semidefiniteness constraint $X \succeq 0$ and the capacity constraint $\langle ff^T, X \rangle \leq (2F - f(V))^2$ couple all entries X_{ij}, the bound can sometimes be improved considerably by enforcing cycle inequalities on entries ij with $ij \notin E$. Within the max-cut setting, the approaches of [29,40] separate triangle inequalities on the complete graph, but this is computationally too expensive for graphs having more than a few hundred nodes. It was shown in [26] that adding just a few edges per node may already lead to strong improvements while maintaining sparsity. Good guidelines for choosing the

additional edges, however, are not yet known. For the unit weight case $f = \mathbf{1}$ it seems likely that there exists some favorable sparse graph structure that works reasonably well when added to any graph. Some experiments in this direction are reported in [4] but so far the dynamic support extension approach of [26] seems to work better. In order to keep the paper self contained we give a brief sketch of it, see Algorithm 2. Given the current support $\hat{E}$ and the exact values of the aggregate $\bar{X}$ on $\hat{E}$ (this is achieved by storing all values $\overline{W}_{ij}$ for $ij \in \hat{E}$), we first reduce this support to the joint support E' of all current sparse inequalities ($E \subseteq E' \subseteq \hat{E}$). Let $\bar{E} := \binom{V}{2 \setminus E'}$ denote the set of edges not in E'. The idea is to add for each node $\bar{\imath} \in V$ at most two edges from $\bar{E}$ that are hopefully part of a violated cycle inequality (3) in $\bar{X}$. For edges $\{i, j\} \in \bar{E}$ the correct value of $\bar{X}_{ij}$ is not available, but $\tilde{X}_{ij} := n[PUP^T]_{ij}$ may be used as a rough approximation. For each root node $\bar{\imath} \in V$ a shortest path tree is computed within (V, E') with respect to edge weights $1 - |\bar{X}_{ij}/\sqrt{\bar{X}_{ii}\bar{X}_{jj}}|$ for $ij \in E'$. With respect to this tree, each edge $\bar{\imath}\bar{\jmath} \in \bar{E}$ induces a cycle $\mathcal{C}_{\bar{\jmath}}$. For the cycle inequality (3) on this cycle we use the values $\tilde{y}_{ij} := \frac{1}{2}(1 - \bar{X}_{ij}/\sqrt{\bar{X}_{ii}\bar{X}_{jj}})$ for $ij \in \mathcal{C}_j \cap E^k$ and $\tilde{y}_{\bar{\imath}\bar{\jmath}} = \frac{1}{2}(1 - \tilde{X}_{ij})$ to determine a violation measure

$$v_{\bar{\imath}}(\bar{\jmath}) := \max \left\{ 1 + \sum_{e \in D} (\tilde{y}_e - 1) - \sum_{e \in \mathcal{C}_{\bar{\jmath}} \setminus D} \tilde{y}_e : D \subseteq \mathcal{C}_{\bar{\jmath}}, |D| \text{ odd} \right\}.$$

For each $\bar{\imath}$ we now add a "most violated" edge $\bar{\imath}\hat{\jmath}$ with $\hat{\jmath} \in \text{Argmax}\{v_{\bar{\imath}}(j) : \bar{\imath}j \in \bar{E}\}$ and a most undecided edge $\bar{\imath}\tilde{\jmath}$ with $\tilde{\jmath} \in \text{Argmin}\{|\tilde{X}_{\bar{\imath}j}| : \bar{\imath}j \in \bar{E}\}$. This support extension routine is called directly after the first separation step and after each tenth separation step. After every modification of $\hat{E}$ the matrix $\overline{W}$ has to be rebuilt from scratch.

Algorithm 2 Support Extension

Input: Current support $\hat{E} \supseteq E$ containing the support of all sparse A_i ($i \in \{1, \ldots, m\}$), aggregate values $\bar{X}_{ij}$ for $ij \in \hat{E}$, the bundle matrix P and the current U solving (14).

Output: new aggregate support $\hat{E}'$ comprising E and the support of all sparse A_i.

1: Set $E' \leftarrow E$.
2: **for all** $i \in \{1, \ldots, m\}$ with A_i having sparse support **do**
3: For the support $E'' \subset \binom{V}{2}$ of A_i set $E' \leftarrow E' \cup E''$ ($\subseteq \hat{E}$).
4: **end for**
5: Compute $\omega_{ij} \leftarrow 1 - |\bar{X}_{ij}/\sqrt{\bar{X}_{ii}\bar{X}_{jj}}|$ and $\tilde{y}_{ij} \leftarrow \frac{1}{2} - \frac{1}{2}\bar{X}_{ij}/\sqrt{\bar{X}_{ii}\bar{X}_{jj}}$ for $ij \in E'$.
6: Set $\hat{E}' \leftarrow E'$, $\bar{E} \leftarrow \binom{V}{2} \setminus E'$, and compute $\tilde{X}_{ij} \leftarrow n[PUP^T]_{ij}$ for $ij \in \bar{E}$.
7: **for all** $\bar{\imath} \in V$ **do**
8: Compute shortest path tree $T_{\bar{\imath}} = (V, E_{\bar{\imath}})$ on E' for weights ω and root $\bar{\imath}$.
9: **for all** $\bar{\imath}\bar{\jmath} \in \bar{E}$ **do**
10: Find the unique cycle $\mathcal{C}_{\bar{\jmath}} \subseteq E_{\bar{\imath}} \cup \{\bar{\imath}\bar{\jmath}\}$ and compute $\tilde{y}_{\bar{\imath}\bar{\jmath}} \leftarrow \frac{1}{2} - \frac{1}{2}\tilde{X}_{\bar{\imath}\bar{\jmath}}$.
11: Determine $v_{\bar{\imath}}(\bar{\jmath}) \leftarrow \max\{1 + \sum_{e \in D}(\tilde{y}_e - 1) - \sum_{e \in \mathcal{C}_{\bar{\jmath}} \setminus D} \tilde{y}_e : D \subseteq \mathcal{C}_{\bar{\jmath}}, |D| \text{ odd}\}$.
12: **end for**
13: Choose $\hat{\jmath} \in \text{Argmax}\{v_{\bar{\imath}}(j) : \bar{\imath}j \in \bar{E}\}$ and $\tilde{\jmath} \in \text{Argmin}\{|\tilde{X}_{\bar{\imath}j}| : \bar{\imath}j \in \bar{E}\}$.
14: Put $\hat{E}' \leftarrow \hat{E}' \cup \{\bar{\imath}\hat{\jmath}, \bar{\imath}\tilde{\jmath}\}$.
15: **end for**
16: **return** $\hat{E}'$.

Rounding heuristics

The primal aggregate $\bar{X} = n(PUP^T + \alpha\overline{W})$ offers several possibilities for designing rounding heuristics. In all approaches we first try to find a partition vector $\bar{x} \in \{-1, 1\}^n$ so that $\bar{x}\bar{x}^T$ is close to $\bar{X}$ and then apply local exchange heuristics in order to enforce feasibility of the bisection constraint $|f^T x| \leq 2F - f(V)$. One way to generate $\bar{x}$ is the random hyperplane rounding technique of Goemans and Williamson [23]. For this, we first compute an eigenvalue factorization of $U = Q\Lambda_U Q^T$ with $\lambda_{\max}(U) = [\Lambda_U]_{11} \geq \cdots \geq [\Lambda_U]_{rr} = \lambda_{\min}(U) \geq 0$ and approximate $\bar{X}$ by $\bar{P}\bar{P}^T$ with

$$\bar{P} = \sqrt{n}PQ\sqrt{\Lambda_U}. \tag{18}$$

Random hyperplane rounding is then applied to the n normalized row vectors of $\bar{P}$. In fact, for random $h \in \mathbb{R}^r$ we sort the values $\bar{P}_{i,\bullet}h$ $(i \in V)$ nonincreasingly and fill up one side of the partition in this order until the capacity restrictions are met or exceeded yielding a corresponding feasible or infeasible $\bar{x}$. A slightly simplified version of this seemed to work even better. Indeed, the hope is that $\bar{X}$ is actually close to some rank one matrix xx^T with $x \in \{-1, 1\}$ and the sign structure of an eigenvector to the maximum eigenvalue of $\bar{X}$ should already give a reasonable approximation. The first column of $\bar{P}$ should be a reasonable approximation of this eigenvector and we simply try $\bar{x}_i = \operatorname{sign} \bar{P}_{1i}$ as starting partition. If $[\Lambda_U]_{11}$ is not that much bigger than $[\Lambda_U]_{22}$ or if there are some values $\bar{P}_{1i}$ close to zero it might make sense to include some information from the second eigenvector as well. Therefore we generate two more starting vectors based on $p^+ = \bar{P}_{\bullet,1} + \bar{P}_{\bullet,2}$ and $p^- = \bar{P}_{\bullet,1} - \bar{P}_{\bullet,2}$ using again the sorted values of these vectors to generate initial partitions. For a given number of columns $\bar{k}$ and a corresponding sign pattern $a \in \{-1, 1\}^{\bar{k}-1}$ this procedure is described in Algorithm 3. This heuristic, called *SumPi* in [4], actually performed best in the experiments of [4]. Some more classical rounding approaches based on the values $\bar{y}_{ij} = \frac{1}{2}(1 - \bar{X}_{ij})$ for $ij \in E$ were also quite successful in [4], but on average SumPi was observed to be slightly superior. In our experiments reported in Sect. 5 we called SumPi after every descent step.

Algorithm 3 Initial Heuristic SumPi

Input: $\bar{P} \in \mathbb{R}^{V \times r}$ of (18), weights f_v $(v \in V)$, capacity F, constant $\bar{k} \leq r$, $a \in \{-1, 1\}^{\bar{k}-1}$, Laplace matrix L.
Output: Bisection vector $z \in \{-1, 1\}^n$ or zero vector on failure.

Compute $p_v \leftarrow P_{v1} + \sum_{l=2}^{\bar{k}} a_{l-1} P_{vl}$ and put $z_v \leftarrow 1$ for all $v \in V$
Sort $V = \{v_1, \ldots, v_n\}$ so that $p_{v_i} \leq p_{v_j}$ if $i < j$
for $v = v_1, \ldots, v_n$ **do**
  if $f_v + \sum_{i \in V : z_i = -1} f_i < f(V) - F$ **then**
    $z_v = -1$
  else if $f_v + \sum_{i \in V : z_i = -1} f_i \leq F$ and $\langle L(z - 2e_v), z - 2e_v \rangle < \langle Lz, z \rangle$ **then**
    $z_v = -1$
  end if
end for
Call local improvement heuristics on z in order to find feasible or better solutions.
return z.

4 Spectral bundle based branching

In this section we will concentrate on two topics, namely the possibilities offered for finding good branching decisions and the aspect of warmstart after branching. While semidefinite relaxations are quite powerful, they also require a lot of time to compute, so typically any semidefinite branch-and-cut code will only be able to visit very few nodes of the branch-and-cut tree within reasonable time. We use the standard approach of setting one edge variable X_{ij} to either -1 (ij is in the cut) or $+1$ (i and j belong to the same set in the bipartition), but explore new possibilities to exploit the information offered by the primal aggregate $\bar{X}$. The branching decision is then implemented by adding the constraint $X_{ij} = \pm 1$ and induces a big change on the values of relatively few Lagrange multipliers while many others need almost no corrections. This causes scaling problems and is a notoriously difficult situation for first order methods. Therefore we investigate some ideas to improve warm start properties by scaling heuristics.

Variable selection for branching

First note that there is no need to require $ij \in E$ when determining an X_{ij} to be set to ± 1 in semidefinite relaxations, because ij may always be added to E with a zero cost coefficient. In [29] the branching strategy resulting in the smallest branch-and-cut tree was rule $R3$, it selects an edge $ij \in \operatorname{Argmin}\{|\bar{X}_{ij}| : ij \in E\}$ that is "most undecided". In our new approaches we would like to improve this criterion by taking into account the global information offered in $\bar{P}^k$ of (18). Using the vector labeling interpretation of [23,37] each row $\bar{P}_{i,\bullet}$ ($i \in V$) represents node i by a vector in $\mathbb{R}^r$ and ideally they should all lie in a common one-dimensional subspace so that $\bar{X}$ is of rank one. We try to identify a common dominating direction of the vectors $\bar{P}_{i,\bullet}$ by computing a singular value decomposition of $\bar{P} = \bar{U}\Sigma\bar{V}$, with $\Sigma_{11} \geq \cdots \geq \Sigma_{rr} \geq 0$. As dominating direction we use $\bar{v} := (\bar{V}_{1,\bullet})^T \in \mathbb{R}^r$. A possible branching rule is to determine the nodes

$$\bar{\imath} \in \operatorname*{Argmax}_{i \in V} \left| \frac{\bar{P}_{i,\bullet}}{\|\bar{P}_{i,\bullet}\|} \bar{v} \right| \quad \text{and} \quad \bar{\jmath} \in \operatorname*{Argmin}_{i \in V} \left| \frac{\bar{P}_{i,\bullet}}{\|\bar{P}_{i,\bullet}\|} \bar{v} \right|$$

that are most and least articulate with respect to the dominating direction $\bar{v}$ with the intention to clarify the relative position of $\bar{\jmath}$ with respect to the majority of the nodes by setting $X_{\bar{\imath}\bar{\jmath}}$. This rule is called *Most Orthogonal* (*MO*) in [4]. In general, one would hope that nodes of almost parallel vectors induce a subgraph that is well decided with respect to a favorable partition of its nodes. Fixing the decision between two pronounced representatives of different one-dimensional subspaces should therefore encourage a decision between all nodes of the corresponding two subgraphs in one step. The branching strategy *Most Orthogonal Elaborate (MOEL)* tries to identify such favorable sets as follows. Given a parameter $\gamma \in (0, 1)$, say $\gamma = 0.2$, the set of nodes considered "orthogonal" to the dominating direction $\bar{v}$ is

$$O := \left\{ i \in V : \left| \frac{\bar{P}_{i,\bullet}}{\|\bar{P}_{i,\bullet}\|} \bar{v} \right| \leq \gamma \right\}.$$

If O is empty, use rule (MO). Otherwise, form set O_h for $h = 1, 2, \ldots, \bar{h}$ and suitable $\bar{h} \in \mathbb{N}$ by finding the next direction

$$\bar{v}_h = \left(\frac{\bar{P}_{i_h,\bullet}}{\|\bar{P}_{i_h,\bullet}\|} \right)^T \quad \text{for some} \quad i_h \in \operatorname{Argmin} \left\{ \left| \frac{\bar{P}_{i,\bullet}}{\|\bar{P}_{i,\bullet}\|} \bar{v} \right| : i \in O \setminus \bigcup_{h'=1}^{h-1} O_{h'} \right\}$$

and by collecting almost parallel vectors to v_h in

$$O_h := \left\{ i \in O \setminus \bigcup_{h'=1}^{h-1} O_{h'} : \left| \frac{\bar{P}_{i,\bullet}}{\|\bar{P}_{i,\bullet}\|} \bar{v}_h \right| \geq 1 - \gamma \right\}$$

so that $O \setminus \bigcup_{h=1}^{\bar{h}} O_h$ becomes empty. As an aside, attempts to identify the directions via singular value decomposition of the rows belonging to O failed so far, maybe due to the rather large diversity of the row vectors belonging to O. Node i_h is used as a representative of the set O_h and it remains to decide which of the edges $\bar{\imath} i_h$ should be branched on. To this end we define for each $h = 1, \ldots, \bar{h}$ two trial solutions

$$X_h^+ := P_h^+ (P_h^+)^T + \overline{W} \quad \text{and} \quad X_h^- := P_h^- (P_h^-)^T + \overline{W}$$

where

$$[P_h^+]_{i,\bullet} := \begin{cases} \bar{P}_{i,\bullet} & \text{for } i \in V \setminus O_h \\ \operatorname{sign}(\bar{P}_{i,\bullet} \bar{v}_h) \bar{P}_{\bar{\imath},\bullet} & \text{for } i \in O_h \end{cases}$$

and

$$[P_h^-]_{i,\bullet} := \begin{cases} \bar{P}_{i,\bullet} & \text{for } i \in V \setminus O_h \\ -\operatorname{sign}(\bar{P}_{i,\bullet} \bar{v}_h) \bar{P}_{\bar{\imath},\bullet} & \text{for } i \in O_h. \end{cases}$$

They correspond to setting all vectors of nodes in O_h to being positively or negatively aligned with the vector of the representative $\bar{\imath}$ of the dominating direction $\bar{v}$. Now we branch on an edge $\bar{\imath} i_{\hat{h}}$ with

$$\hat{h} \in \operatorname*{Argmin}_{h=1,\ldots,\bar{h}} \left(\langle C, X_h^+ \rangle + \langle C, X_h^- \rangle \right)$$

in the hope that this gives the best average improvement of the bound on both branches.

Scaling for warm start after branching

In principle there is no difficulty to start a bundle method from any given stability center $\hat{\mu}$, but subsequent progress of the method depends heavily on the scaling of the variables. If just a few multipliers have to change a lot while most others should roughly stay the same, first order methods typically exhibit very slow convergence. This is exactly the situation that is encountered whenever a branching constraint $X_{ij} = \pm 1$ is added to the converged Lagrangian relaxation of the preceding subproblem. The quadratic term $\frac{1}{2}\|\mu - \hat{\mu}^k\|^2$ in (13) penalizes movement in all directions equally while one would prefer wider steps in directions that are affected by the new constraint. One

can try to improve progress by adjusting the quadratic term via diagonal scaling to $\frac{1}{2}\|\mu-\hat{\mu}^k\|_H^2 = \frac{1}{2}(\mu-\hat{\mu}^k)^T H(\mu-\hat{\mu}^k)$ for some appropriate diagonal matrix $H \succ 0$. Several attempts to set up H are described in [4], here we only report on the approach by *spectral analysis at the current point* (*SACP*) that seemed to perform best among these. To fix ideas, suppose $X_{\bar{\imath}\bar{\jmath}} = -1$ is enforced by adding the constraint

$$\langle \bar{A}, X\rangle \le -1 \text{ with } \bar{A}_{ij} = \begin{cases} 1, & ij \in \{\bar{\imath}\bar{\jmath}, \bar{\jmath}\bar{\imath}\}, \\ 0, & \text{otherwise.} \end{cases}$$

Using Lagrange multiplier $\nu \ge 0$ for this constraint we would like to minimize

$$n\lambda_{\max}(C - \mathcal{A}^T\mu - \nu\bar{A}) + (b-\eta)^T\mu - \nu.$$

The current values are $\mu = \hat{\mu}$ and $\nu = 0$. We would like to increase ν to improve the bound. In order to study its influence on $\lambda_{\max}$ observe that

$$\bar{A} = \bar{v}\bar{v}^T - \bar{w}\bar{w}^T \text{ with } \bar{v}_i = \begin{cases} 1/\sqrt{2}, & i = \bar{\imath}, \\ 1/\sqrt{2}, & i = \bar{\jmath}, \\ 0, & \text{otherwise,} \end{cases} \quad \bar{w}_i = \begin{cases} 1/\sqrt{2}, & i = \bar{\imath}, \\ -1/\sqrt{2}, & i = \bar{\jmath}, \\ 0, & \text{otherwise.} \end{cases}$$

Thus increasing ν adds the positive semidefinite part $\nu\bar{w}\bar{w}^T$ to $C - \mathcal{A}^T\hat{\mu}$ while it subtracts a positive semidefinite part $\nu\bar{v}\bar{v}$. If $\bar{w}$ is almost an eigenvector to $\lambda_{\max}$, *i. e.*, $\bar{w}^T(C-\mathcal{A}^T\hat{\mu})\bar{w} \ge \lambda_{\max}(C-\mathcal{A}^T\hat{\mu}) - \epsilon$ for some small $\epsilon > 0$, then an increase in ν will increase $\lambda_{\max}$ by the same amount unless this can be compensated by changing some other variables of $\hat{\mu}$. Among these, only those indices i with rather large values $|\bar{w}^T A_i \bar{w}|$ can be expected to have a strong effect on the subspace spanned by $\bar{w}$. Because their interaction is hard to predict we allow larger movement for all of them. On the other hand, subtracting $\nu\bar{v}\bar{v}$ can do no harm but might open some space to move for variables μ_i with $|\bar{v}^T A_i \bar{v}|$ large. Motivated by this and putting $A_{m+1} := \bar{A}/\sqrt{2}$, $\hat{\mu}_{m+1} := \nu$, we define a diagonal scaling matrix H via $H_{ii} := 1/s_i$ for $i = 1, \ldots, m+1$ with

$$s_i := 1 + \sigma \begin{cases} \max\{(\bar{v}^T A_i \bar{v})^2, (\bar{w}^T A_i \bar{w})^2\}, & \text{if } \bar{w}^T(C-\mathcal{A}^T\hat{\mu})\bar{w} \ge \lambda_{\max}(C-\mathcal{A}^T\hat{\mu}) - \epsilon, \\ (\bar{v}^T A_i \bar{v})^2, & \text{otherwise.} \end{cases}$$

Based on the experiments in [4] we choose $\sigma = 20$ and $\epsilon = 0.1$ in our actual computations. The scaling is reset to $H = I$ after the next descent step.

5 Computational results at the root node

In this section we first introduce the common software framework and the data sets in Sect. 5.1. Next, in Sect. 5.2, we discuss the results of the cutting plane algorithms when separating the cycle, the knapsack tree and the bisection knapsack walk inequalities within the LP and the SDP model. Then, in Sect. 5.3, we analyze the performance of the heuristics METIS, SumPi and Goemans–Williamson when working with the semidefinite model. Section 5.4 illustrates the effect of separating cycle inequalities on a slightly extended support of the graph. Finally, in Sect. 5.5, we show the evolution

of the lower bound over time for the LP and the SDP model. All computations of this section are carried out only in the root node of the branch-and-bound tree.

5.1 Computational environment and instances

For solving the minimum graph bisection problem we use the state of the art branch-and-cut framework SCIP (Solving Constraint Integer Programs [1]). In combination with CPLEX as LP-solver SCIP is one of the fastest MIP-solvers currently available. Because SCIP can handle non-linear constraints, it is possible to replace the LP-solver with a semidefinite solver or to use both within one run. SCIP follows the common branch-and-cut strategy which we outline in Fig. 1. For full implementation details we refer to [1].

The semidefinite relaxation is solved by the ConicBundle code [27] which is an implementation of the spectral bundle method [30]. The integration of the SDP-solver into SCIP is explained in detail in [4]. The main solving loop of the SDP-solver is drafted in Fig. 2. For our computations we used SCIP version 1.1.0 [2] with ILOG CPLEX 12.10 [31] as LP-solver and Conic Bundle version 0.3.5 as SDP-solver. The computations were executed on Intel(R) Core(TM) i7 CPU 920 machines with 8 MB cache and 12 GB RAM under openSUSE Linux 11.1 (x86_64) in single thread mode.

The first set of graph instances that we used in our computational experiments is based on nets from VLSI design discussed in [33]. The authors consider the placement problem in the layout design of electronic circuits, which consists of finding a non-overlapping assignment of rectangular cells to positions on the chip so that wireability is guaranteed and certain technical constraints are met. Using heuristic algorithms for the optimization problem involved, a partial placement is obtained for the chips *alue, alut, diw, dmxa, gap* and *taq*. We use this as input for the minimum graph bisection

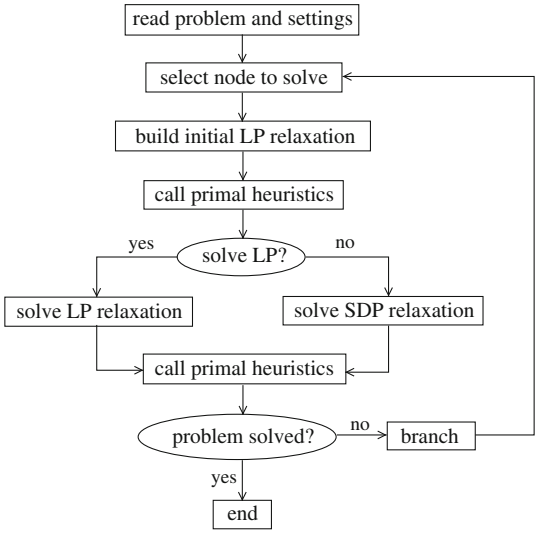

Fig. 1 Main solving loop in SCIP

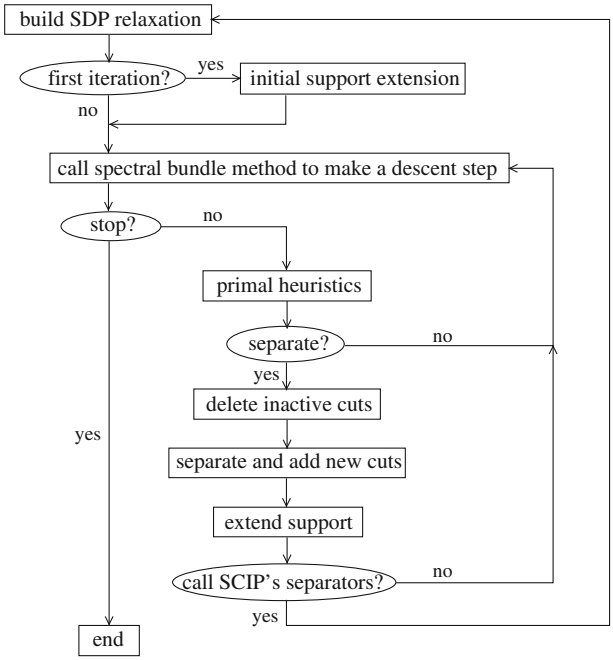

Fig. 2 Main solving loop of the SDP-solver

problem. The second set of graphs originates from nested dissection approaches for solving sparse symmetric linear KKT-systems and were communicated to us in [26] by Sharon Filipowski from Boeing. Their names start with kkt. These graphs have equally weighted nodes and edges. In Table 1 we give the characteristics of all graphs.

To illustrate the comparative quality of the bounds produced by the respective algorithmic choices on the 28 test instances we employ performance profiles [13]. In these, the benchmark value of each instance is set to the best (upper or lower) bound delivered over all algorithmic choices. The horizontal axis specifies a multiplicative deterioration factor and the vertical axis gives for each algorithm the number of instances for which the algorithm's bound lies within this factor times the best bound. The factor always starts with value one on the left-hand side so that the value on the left boundary gives the number of instances where each algorithm produced the best bound. As the factor gets worse (*i. e.*, it decreases for lower bounds and increases for upper bounds) the number of instances increases monotonically and we stop when for all algorithms each instance's bound is within the threshold value.

5.2 Separation at the root node

It is important to mention at this point that the SCIP-framework employs the same separation algorithms for both linear (1) and semidefinite (2) programs by transforming any $\bar{X}$ to $\bar{y}$ via $\bar{y}_{ij} = \frac{1}{2}(1 - \bar{X}_{ij})$ for all $ij \in E$. Separated inequalities $\sum_{ij \in E} k_{ij} y_{ij} \leq k_l$

Table 1 Characteristics of the test instances

Graph	No of nodes	Node weight		Mean node degree	No of edges	Edge weight			Density	Best lower bound
		Range	Mean			Range	Mean	Total		
taq	170	1–48	6.22	5.0	424	2–110	20.36	8,634	2.95	110[a]
taq	278	1–9	3.81	2.8	396	1–29	13.03	5,158	1.03	37[a]
taq	228	1–48	4.57	6.1	692	1–55	8.32	5,759	2.67	63[a]
taq	1,021	1–50	3.95	4.4	2,253	1–20	2.00	4,510	0.43	118[a]
taq	334	1–50	3.17	22.5	3,763	1–79	2.15	8,099	6.77	340.2
taq	1,021	1–50	3.95	10.7	5,480	1–79	5.26	28,800	1.05	1,598.7
taq	1,021	1–50	3.95	62.0	31,641	1–10	1.32	41,616	6.08	398.5
diw	166	1–50	3.98	6.1	507	1–18	4.82	2,446	3.70	28[a]
diw	681	1–50	3.78	4.4	1,494	1–12	2.06	3,081	0.65	142[a]
diw	681	1–50	3.78	9.1	3,104	1–36	6.03	18,706	1.34	999.9
diw	681	1–50	3.78	18.8	6,402	1–6	1.21	7,717	2.76	3,328.8
dmxa	1,755	1–48	2.76	4.2	3,686	1–10	2.03	7,501	0.24	94[a]
dmxa	1,755	1–48	2.76	12.4	10,867	1–5	1.24	13,502	0.71	146.1
gap	2,669	1–9	2.53	4.6	6,182	1–6	1.99	12,280	0.17	74[a]
gap	2,669	1–9	2.53	18.6	24,859	1–3	1.17	29,037	0.70	55[a]
alue	6,112	1–8	4.15	5.5	16,896	1–4	2.16	36,476	0.09	272[a]
alut	2,292	1–8	4.15	5.5	6,329	1–4	2.14	13,532	0.24	154[a]
alut	2,292	1–8	4.15	431.5	494,500	1–3	1.15	568,665	18.83	64,155.7
kkt	82	1–1	1.00	6.3	260	1–1	1.00	260	7.83	13[a]
kkt	115	1–1	1.00	7.5	433	1–1	1.00	433	6.61	28[a]
kkt	2,063	1–1	1.00	10.6	10,936	1–1	1.00	10,936	0.51	6[a]
kkt	2,117	1–1	1.00	13.2	14,001	1–1	1.00	14,001	0.63	561.2
kkt	5,150	1–1	1.00	7.7	19,906	1–1	1.00	19,906	0.15	146.5
kkt	2,817	1–1	1.00	17.7	24,999	1–1	1.00	24,999	0.63	6.2
kkt	2,186	1–1	1.00	34.6	37,871	1–1	1.00	37,871	1.59	2,067.4
kkt	17,990	1–1	1.00	5.1	45,883	1–1	1.00	45,883	0.03	6,564.7
kkt	17,148	1–1	1.00	13.1	112,633	1–1	1.00	112,633	0.08	8,095.2
kkt	20,006	1–1	1.00	24.2	241,947	1–1	1.00	241,947	0.12	9,438.9

[a] Optimum value

are translated into constraints $\langle K, X\rangle \leq k_s$ for the primal semidefinite relaxation by $K_{ij} = -\frac{1}{2}k_{ij}$ for all $ij \in E$ and $k_s = 2k_l - \sum_{ij\in E} k_{ij}$.

In Figs. 3 and 4 we present a comparison of the performance of the cutting plane algorithms based on the knapsack tree inequalities with minimal roots (Theorem 2), bisection knapsack walk inequalities (9) and cycle inequalities (3) in combination with the linear and the semidefinite relaxation. Note that the cycle inequalities are part of the integer linear model (1), they are separated in each setting of the LP relaxation. These tests consider the root node without branching and with a time limit of 2 h. We added inequalities of each class separately as long as violated cuts were found.

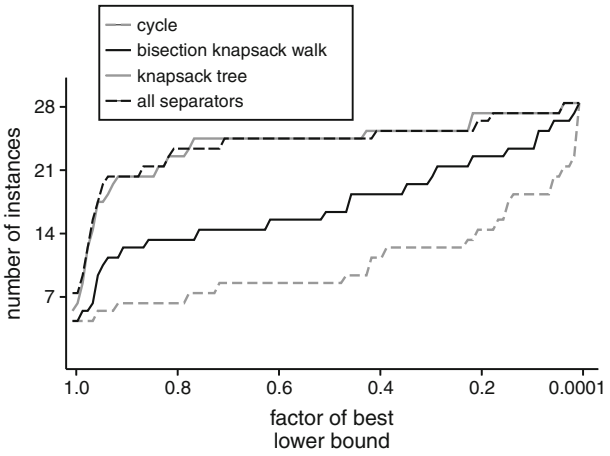

Fig. 3 Performance of separation routines with LP-relaxation

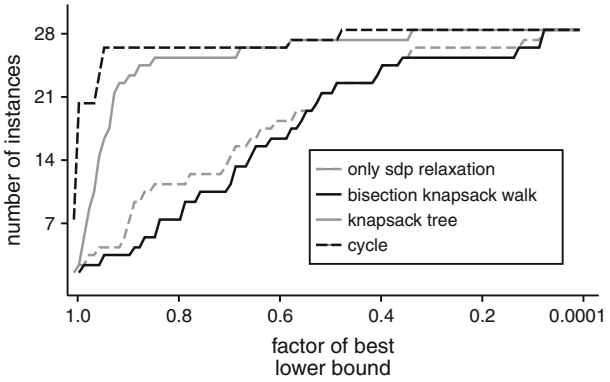

Fig. 4 Performance of separation routines with SDP-relaxation

Within the linear relaxation the knapsack tree inequalities have the biggest impact on the improvement of the lower bound. This may seem surprising in view of the fact that the knapsack walk inequalities subsume the knapsack tree inequalities; the reason is that, for speed, both separators start from a few seed nodes and then the minimal root strengthening of Theorem 2 boosts the performance of knapsack trees. In the LP model the best results are achieved when applying all separators, as one can see in Fig. 3. The cycle separator alone, however, achieves very poor lower bounds, thus studying the bisection cut polytope P_B pays off when considering the linear relaxation of MB. The situation is stunningly different in the semidefinite case. Here, the pure semidefinite relaxation yields already reasonable good lower bounds. For very large instances like *alut2292.494500* the separation routines only slow down the solution process and thus lead to worse bounds when computing time is a major limiting factor. The best results are obtained when the cycle inequalities are separated exclusively (on an extended support, see Sect. 5.4), additional knowledge about the bisection

knapsack polytope does not seem to help for these instances. From the bisection specific inequalities, *i. e.*, the knapsack tree and bisection knapsack walk inequalities, only the knapsack tree inequalities improve the basic SDP bound significantly, while the bisection knapsack walk inequalities seem not to help at all. In comparison to cycle inequalities, however, the bounds are worse and computation times are higher.

5.3 Primal heuristics

A major advantage of semidefinite relaxations over LP relaxations are the powerful possibilities they offer for rounding heuristics. Indeed, the random hyperplane rounding approach of Goemans and Williamson [23] produces very good solutions after the first few iterations of the semidefinite solver. In comparison, the slight variation SumPi suggested in Sect. 3 is sometimes slightly worse but seems to be a bit more stable than pure random hyperplane rounding. This is illustrated in Fig. 5. The bad performance of the LP rounding heuristics is due to the following reasons. On one side, for LP there is no efficient approximation algorithm even for max-cut. LP rounding typically only starts to work well, once the LP solution is reasonably close to optimal solutions, which may take a long time. Good solutions are often found only after quite a number of branching steps. On the other side, LP rounding methods do not bring reasonable results in our case as the bisection problem is not completely formulated in the running LP model. It is updated during the computation time as soon as a violated cycle inequality has been found. Standard rounding heuristics would greedily set most of the variables to zero. Therefore the LP model has to rely on heuristics like MeTiS [34] which base on the complete description of the bisection problem.

5.4 Support extension

The surprising strength of the cycle inequalities in combination with the SDP relaxation observed in Sect. 5.2 depends crucially on the fact that the inequalities are separated

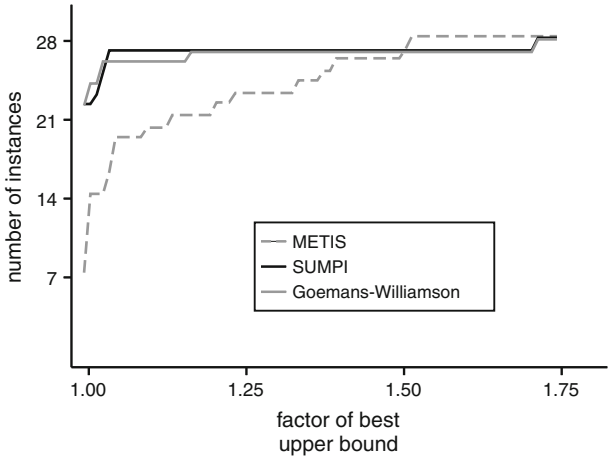

Fig. 5 The performance of primal heuristics applied to the semidefinite program

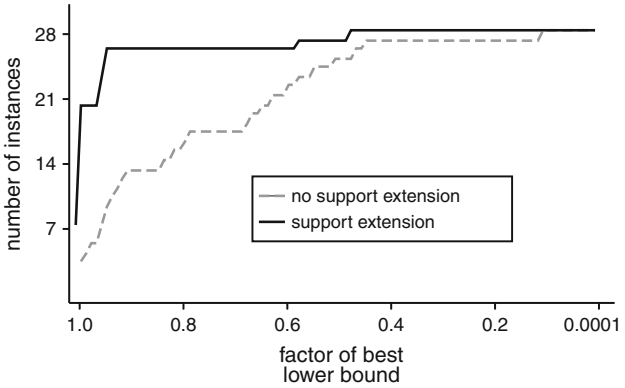

Fig. 6 Semidefinite relaxation in the root node with and without support extension

on an extended support. Figure 6 displays the effect of the dynamic support extension technique of [26] sketched in Sect. 3 in comparison to separating the cycle inequalities on the original support only. Indeed, for some instances without support extension almost no progress is achieved with respect to the pure SDP relaxation because none of the relevant edges are contained in cycles within the original graph.

5.5 Development of the lower bound over time

In Figs. 7, 8, 9 and 10 we compare the development of the lower bound of the LP versus the SDP relaxation over time for a few characteristic instances. The results are given for root node computations with the best respective setting for each relaxation. In particular, LP uses all separators and SDP employs only the cycle separator together with dynamic support extension. The time limit is 2 h. The plot of Fig. 7 gives the typical development of the objective function value over time. The semidefinite relaxation delivers a good bound rather quickly and then exhibits a strong tailing off effect often observed for bundle methods and cutting plane approaches. In contrast, the lower bound of the linear relaxation grows much slower, looking linear on a logarithmic time scale. Figures 8 and 9 give examples of some bigger instances where progress is very slow for SDP, LP performance being abysmal in these cases. An exception is the instance *kkt2817_24999* of Fig. 10, where the linear relaxation outperforms the semidefinite one, but the shape of both curves is again typical.

6 Computational results for branch-and-cut

The second part of our computational experiments focuses on our SCIP based semidefinite branch-and-cut approach sketched in Sect. 5.1. Motivated by the results of the previous section the parameters were set as follows. For the LP-relaxation the knapsack tree separator is given the highest priority and separation frequency, followed by the cycle and the bisection knapsack walk separators. The LP branching decision is

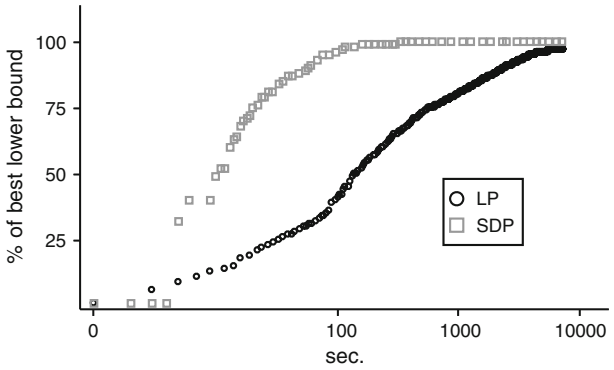

Fig. 7 dmxa1755.3686

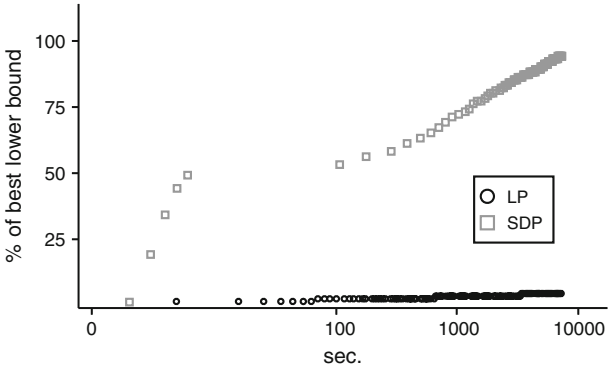

Fig. 8 alut2292.494500

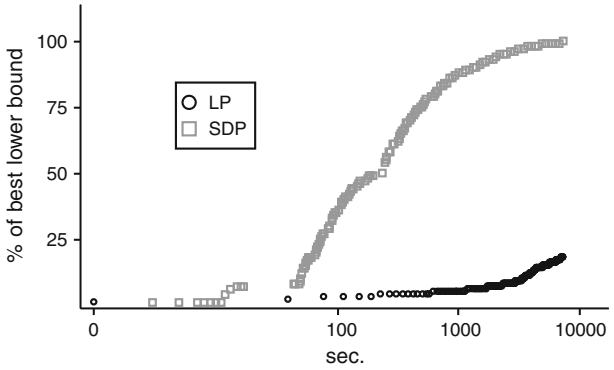

Fig. 9 kkt5150.19906

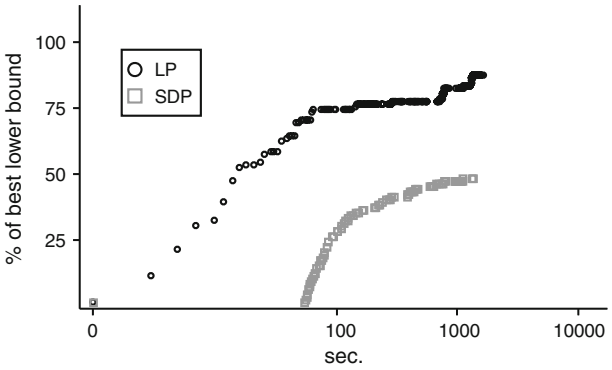

Fig. 10 kkt2817.24999

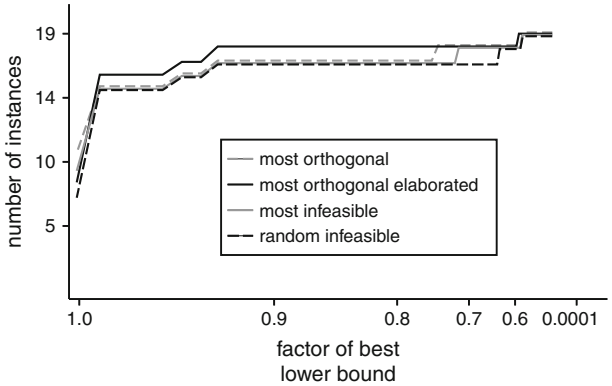

Fig. 11 The performance of the branching rules for SDP

determined by the standard setting provided by SCIP. For the semidefinite relaxation the cycle separator is the only separator. The SDP branching strategy is derived from the experiments presented in this section. We limited computation time to 4 h for each instance. In Sect. 6.1 we show the performance of the four branching rules implemented for SDP. The tests presented in Sect. 6.2 concern the decision when to start to branch in order to reduce tailing off effects. In Sect. 6.3 we show the impact of the scaling and warm start methods. Finally, in Sect. 6.4, we compare the performance of the LP and SDP based branch-and-cut algorithm.

Before embarking on the actual comparisons please note that all results of this section need to be interpreted with some care. Indeed, a general difficulty in judging the effectiveness of branching rules for SDP is the fact that due to rather long computation times in the root node the actual number of branching decisions observed within a computation time of 4 h is quite limited, in particular if the emphasis is on medium to large instances. An extensive study on smaller and in part random instances is given in [4] and results there point into the same directions observed here.

In Figs. 11, 12 and 13 of this section we only display results for those instances where branching occured in actual computations of at least one algorithmic choice.

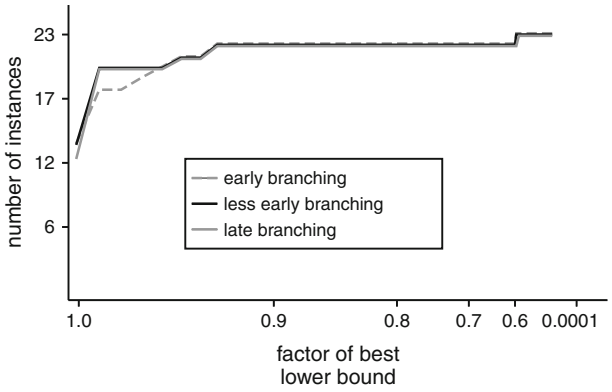

Fig. 12 Early versus late branching

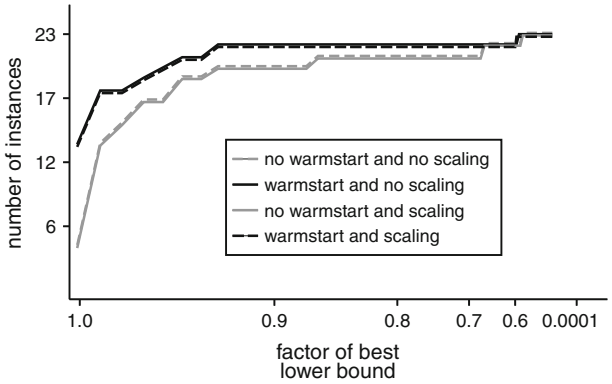

Fig. 13 The impact of warmstart and scaling

6.1 Branching rules for SDP

Figure 11 displays the performance of the branching strategies *most orthogonal* (MO) and *most orthogonal elaborate* (MOEL) suggested in Sect. 4 in comparison to the standard rules *most infeasible*, which picks the edge ij within the original support E that minimizes $|X_{ij}|$, and *random infeasible*, which picks a random edge $ij \in E$ among all edges with $|X_{ij}| \neq 1$. Results are only given for instances with at least three branch-and-bound nodes. While MOEL is not always the best choice, it produces good results quite reliably and so this is selected as the default rule in all further experiments.

Note that all newer branching rules (pseudocost branching, reliable branching, etc., see for instance [3]) for LP based branch-and-cut have also been taken into account with limited success. The reason is the same as mentioned above. These ideas are especially successful if the branch-and-bound trees are huge, because they gather information throughout the enumeration process about the variables to branch on. Since the SDP trees are much smaller these techniques tend to be much less effective.

6.2 Early versus late branching

As illustrated in Sect. 5.5, the joint tailing off effects of the spectral bundle method and the cutting plane approach result in very slow progress after strong initial improvements of the bound. Once this slow down is observed, the hope is to obtain faster progress by branching and computing bounds for the subproblems of each branch. In lack of a mathematically well founded indicator for identifying a good point for switching from the separation process to branching, we experimented with a decision rule based on the inner mechanics of the bundle method. For bundle methods, the number of functions evaluations that do not lead to descent steps is a good indicator for the progress of the method. In Table 2 and the corresponding Fig. 12 we consider three scenarios. In all of them we stop computation in a node for the usual combination of criteria regarding the progress of the bound, the violation of inequalities and the relative precision obtained (see [4] for details), giving the basic setting of *late branching*. In *early branching* we stop even before these criteria are met if the number of descent steps is at least ten and the total number of evaluations exceeds twenty times the number of descent steps. In *less early branching* the latter factor is set to 200. Even though the results are inconclusive, they might be interpreted as follows. For large scale instances the time limit prevents sufficient branching and spending more time in the root is better, but according to the results of Table 2 as well as experiments with random instances in [4], for small to medium size instances the *early branching* strategy seems to work best. Because branching is not really attractive for large scale instances we choose *early branching* as our default strategy in the sequel.

6.3 Warmstart and scaling

As discussed in Sect. 4, any nonnegative dual point may be chosen as starting point for the spectral bundle method, but because of the importance of scaling issues for first order methods it would be desirable to start at points, where all dual variables have to change by roughly the same amount. When considering the initialization of a new branch-and-cut node, it is therefore worth to investigate the effect of *warmstarts* the method from the best dual point computed in its parent node. In Sect. 4 we also introduced a scaling heuristic intended to mitigate the effects of scaling difficulties caused by the addition of a branching constraint $X_{ij} = \pm 1$. The impact of warmstarting and scaling options is displayed in Fig. 13. On these instances, warmstarting has a positive effect while the influence of the scaling heuristic is somewhat ambiguous. Additional experiences collected in [4] indicate some advantage for scaling on small to medium scale instances and so we decided in favor of warmstarting in combination with the scaling heuristic as our default strategy. It should be noted, however, that we do not consider the current approach satisfactory and more needs to be done. Yet, even in the case of significant progress in warmstarting it seems unlikely that the same efficiencies can be reached that are observed in restarting the simplex method within linear branch-and-cut methods.

Table 2 Early versus late branching with semidefinite relaxation; plotted in Fig. 12

Graph $n.m$	Early branching				Less early branching				Late branching			
	#b&b nodes	Upper bound	Lower bound	Gap	#b&b nodes	Upper bound	Lower bound	Gap	#b&b nodes	Upper bound	Lower bound	Gap
taq278.396	5	37	37	0	5	37	37	0	5	37	37	0
taq1021.2253	3	118	118	0	1	118	118	0	1	118	118	0
taq334.3763	272	342	337	1	89	342	336	1	87	342	336	1
taq1021.5480	53	1,650	1,579	4	7	1,650	1,588	3	7	1,650	1,590	3
taq1021.31641	56	404	389	3	3	404	399	1	2	404	399	1
diw166.507	7	28	28	0	7	28	28	0	7	28	28	0
diw681.1494	13	142	142	0	4	142	141	1	3	142	141	1
diw681.3104	25	1,011	1,001	0	5	1,011	997	1	2	1,011	996	1
diw681.6402	45	331	329	0	14	331	328	0	3	331	328	0
dmxa1755.3686	7	94	94	0	1	94	94	0	1	94	94	0
dmxa1755.10867	20	150	146	2	9	150	145	3	5	150	145	3
gap2669.6182	3	74	74	0	1	74	74	0	1	74	74	0
alue6112.16896	3	272	272	0	1	272	272	0	1	272	272	0
alut2292.6329	3	154	154	0	3	154	154	0	3	154	154	0
alut2292.494500	2	67,815	59,696	13	2	67,815	59,696	13	2	67,815	59,696	13
kkt2063.10936	51	6	6	0	11	6	6	0	11	6	6	0
kkt2117.14001	42	567	560	1	6	567	561	1	3	567	561	1
kkt5150.19906	9	150	140	6	2	150	144	3	2	150	144	3
kkt2817.24999	43	49	7	644	27	49	7	636	27	49	6	657
kkt2186.37871	24	2,077	2,049	1	6	2,077	2,065	0	2	2,077	2,063	0
kkt17990.45883	4	6,623	6,083	8	2	6,658	6,119	8	2	6,658	6,119	8
kkt17148.112633	3	8,173	8,078	1	3	8,194	8,082	1	2	8,194	8,082	1
kkt20006.241947	3	9,593	9,429	1	2	9,593	9,429	1	2	9,593	9,429	1

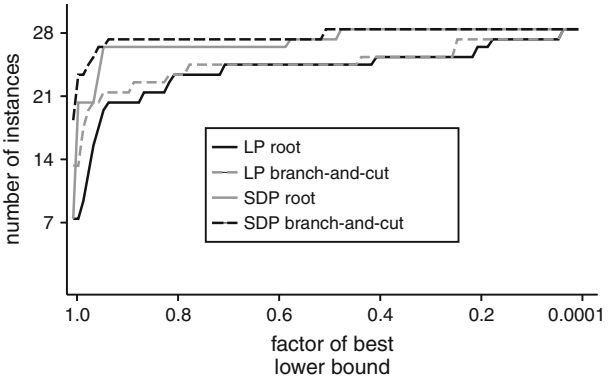

Fig. 14 Branch-and-cut based on the linear versus the semidefinite relaxation with early branching

6.4 LP versus SDP

Figure 14 displays the final results of the branch-and-cut codes for LP and SDP relaxations for the default settings obtained within 4 h of computation time for each instance and displays them along with the lower bounds achieved by theLP- and SDP-relaxation in the root also within 4 h of computation time as described in Sect. 5.5. In Table 3 we composed the figures for reference.

Although the LP based approach achieves better bounds in a couple of instances, it turns out that the advantage of the SDP over the LP approach matches the performance of the bounds as discussed in Sect. 5.5. At least from the LP side it is interesting to note that seven of the instances could be solved in the root, among them are even instances where the SDP approach had to branch. The rather ambiguous outcome of the comparison between SDP branch-and-cut bounds and the SDP root node bound may be seen as a disappointment at first, but is actually in line with the previous observations and folklore experience in branch-and-cut. For small to medium size instances branch-and-cut may actually help to speed up the bounding process, and the results indicate that our approach is a reliable choice for instances with up to 15000 edges and a few thousands of nodes. For larger problems, the time limit is best exploited by putting all efforts into the root node and branching is a waste of time.

7 Conclusion

While SDP relaxations are widely believed to be of use only for small dense instances, our work demonstrates by direct comparison within the same branch-and-cut framework that they are also an attractive and often superior choice in the classical linear programming domain of large and sparse graph partitioning instances. For bisection instances, the semidefinite cutting plane approach based on the spectral bundle method is certainly competitive to linear programming cutting plane methods. Problem specific cutting planes studied in [6], in particular knapsack tree inequalities centered at minimal roots, boost the quality of the LP bound but do not seem to improve the

Table 3 Branch-and-cut based on the linear versus the semidefinite relaxation; plotted in Fig. 14

Graph $n.m$	LP branch & cut				LP root	SDP branch & cut				SDP root
	#b&b nodes	Upper bound	Root lowbd	Lower bound	Lower bound	#b&b nodes	Upper bound	Root lowbd	Lower bound	Lower bound
taq170.424	217	110	102.5	110.0	102.5	1	110	110.0	110.0	103.5
taq278.396	1	37	37.0	37.0	37.0	5	37	35.0	37.0	35.3
taq228.692	1	63	63.0	63.0	62.9	1	63	63.0	63.0	62.8
taq1021.2253	1	130	114.0	114.0	114.0	3	118	116.9	118.0	117.3
taq334.3763	13,389	389	324.8	334.1	325.0	272	342	321.3	336.7	322.8
taq1021.5480	1	2,038	685.0	685.0	685.0	53	1,650	1,553.3	1,579.2	1,578.7
taq1021.31641	13	426	376.1	376.1	376.1	56	404	350.2	388.7	398.9
diw166.507	1	28	28.0	28.0	27.6	7	28	26.8	28.0	26.2
diw681.1494	18,871	144	136.1	139.3	136.1	13	142	138.1	142.0	140.6
diw681.3104	1	1,064	768.9	768.9	768.9	25	1,011	982.4	1,001.1	996.4
diw681.6402	11	360	315.0	316.9	315.5	45	331	317.6	328.8	326.4
dmxa1755.3686	17	126	91.3	91.3	91.3	7	94	91.3	94.0	93.3
dmxa1755.10867	192	191	142.8	143.6	142.8	20	150	141.6	146.0	144.5
gap2669.6182	1,1790	78	73.7	73.9	73.7	3	74	72.3	74.0	73.3
gap2669.24859	1	55	55.0	55.0	55.0	1	55	55.0	55.0	54.9
alue6112.16896	1	276	65.8	65.8	65.8	3	272	270.4	272.0	271.6
alut2292.6329	1	156	135.4	135.4	135.4	3	154	153.0	154.0	153.1
alut2292.494500	1	67,815	2,124.9	2,124.9	2,124.9	2	67,815	59,696.2	59,696.2	64,155.8
kkt82.260	1	13	13.0	13.0	12.6	1	13	13.0	13.0	12.9
kkt115.433	1	28	28.0	28.0	27.8	1	28	28.0	28.0	28.0
kkt2063.10936	1	6	6.0	6.0	5.6	51	6	3.4	6.0	3.4
kkt2117.14001	3,101	567	560.7	561.4	560.7	42	567	547.6	559.8	559.6

Table 3 continued

Graph $n.m$	LP branch & cut				LP root	SDP branch & cut				SDP root
	#b&b nodes	Upper bound	Root lowbd	Lower bound	Lower bound	#b&b nodes	Upper bound	Root lowbd	Lower bound	Lower bound
kkt5150.19906	1	156	35.7	35.7	35.7	9	150	126.2	140.3	146.5
kkt2817.24999	93,697	74	11.4	13.3	11.4	43	49	6.2	6.6	6.2
kkt2186.37871	16,144	2,090	1,674.0	1,674.1	1,674.0	24	2,077	2,017.2	2,049.1	2,067.4
kkt17990.45883	1	6,658	6,271.1	6,271.1	6,272.5	4	6,623	6,082.9	6,082.9	6,130.0
kkt17148.112633	3	8,258	8,147.0	8,147.0	8,147.0	3	8,173	8,078.4	8,078.4	8,095.2
kkt20006.241947	33	9,683	9,503.0	9,503.0	9,503.0	3	9,593	9,428.8	9,428.8	9,438.9

performance of the SDP bounds. Indeed, it is one of the major surprises that for the bisection instances considered here the best results were obtained for the basic semidefinite relaxation improved by exclusively separating cycle inequalities on a slightly extended support. This raises hopes for obtaining semidefinite bounds of similar quality for general constrained quadratic 0-1 programming problems without using problem specific cutting planes. The proposed extension of the semidefinite cutting plane method to a semidefinite branch-and-cut approach works well on instances with up to 15000 edges and a few thousand nodes. For larger instances more work is needed to increase the efficiency of warmstarting the spectral bundle method after the addition of branching constraints.

In order to improve the results it would be valuable to better understand structural properties that are required in the selection of a slightly extended support that is favorable for the separation of general valid inequalities of the cut polytope. This might also give further insight on why the current problem specific inequalities show so little effect on the SDP bound. On the algorithmic side, any progress on the scaling difficulties caused by new primal cuts within the spectral bundle method should be helpful for the semidefinite cutting plane method as well as for the semidefinite branch-and-cut approach.

Acknowledgments This work was supported by grants HE 3524/1-2 and MA1324/1-2 of Deutsche Forschungsgemeinschaft (DFG) and mostly carried out while Michael Armbruster was at Chemnitz University of Technology and Marzena Fügenschuh and Alexander Martin were at Darmstadt University of Technology.

References

1. Achterberg, T.: Constraint Integer Programming. PhD-thesis, Technische Universität Berlin, Verlag Dr. Hut, München (2008)
2. Achterberg, T.: SCIP 1.1.0. Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustr. 7, 14195 Berlin-Dahlem, Germany. http://scip.zib.de (2009)
3. Achterberg, T., Koch, T., Martin, A.: Branching rules revisited. Oper. Res. Lett. **33**, 42–54 (2005)
4. Armbruster, M.: Branch-and-Cut for a Semidefinite Relaxation of Large-scale Minimum Bisection Problems. PhD-thesis, Technische Universität Chemnitz, Chemnitz, Germany (2007)
5. Armbruster, M., Fügenschuh, M., Helmberg, C., Martin, A.: A comparative study of linear and semidefinite branch-and-cut methods for solving the minimum graph bisection problem. In: Lodi, A., Panconesi, A., Rinaldi, G., (eds.) Integer Programming and Combinatorial Optimization. Lecture Notes in Computer Science, vol. 5035, pp. 112–124. IPCO 2008. Springer, Berlin (2008)
6. Armbruster, M., Fügenschuh, M., Helmberg, C., Martin, A.: On the graph bisection cut polytope. SIAM J. Discrete Math. **22**(3), 1073–1098 (2008)
7. Barahona, F., Mahjoub, A.R.: On the cut polytope. Math. Progr. **36**, 157–173 (1986)
8. Belloni, A., Sagastizábal, C.: Dynamic bundle methods. Math. Progr. **120**, 289–311 (2009)
9. Conforti, M., Rao, M., Sassano, A.: The equipartition polytope I. Math. Progr. **49**, 49–70 (1990)
10. Conforti, M., Rao, M., Sassano, A.: The equipartition polytope II. Math. Progr. **49**, 71–90 (1990)
11. de Souza, C.C.: The graph equipartition problem: optimal solutions, extensions and applications. PhD-thesis, Université Catholique de Louvain, Louvain-la-Neuve, Belgium (1993)
12. Deza, M., Laurent, M.: Geometry of Cuts and Metrics. Algorithms and Combinatorics, vol. 15. Springer, Berlin (1997)
13. Dolan, E., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Progr. **91**(2), 201–213 (2002)
14. Engau, A., Anjos, M.F., Vannelli, A.: On interior-point warmstarts for linear and combinatorial optimization. SIAM J. Optim. **20**(4), 1828–1861 (2010)

15. Engau, A., Anjos, M.F., Vannelli, A.: On handling cutting planes in interior-point methods for solving semidefinite relaxations of binary quadratic optimization problems. Technical Report, École Polytechnique de Montréal, Canada. Optimization Methods and Software (to appear) (2010)
16. Eisenblätter, A.: Frequency assignment in GSM networks. PhD-thesis, Technische Universität Berlin, Berlin. ISBN 3-89873-213-4. ftp://ftp.zib.de/pub/zib-publications/books/PhD_eisenblaetter.ps.Z (2001)
17. Ferreira, C.E., Martin, A., de Souza, C.C., Weismantel, R., Wolsey, L.A.: Formulations and valid inequalities for the node capacitated graph partitioning problem. Math. Progr. **74**, 247–267 (1996)
18. Ferreira, C.E., Martin, A., de Souza, C.C., Weismantel, R., Wolsey, L.A.: The node capacitated graph partitioning problem: a computational study. Math. Progr. **81**(2), 229–256 (1998)
19. Frieze, A., Jerrum, M.: Improved approximation algorithms for MAX k-CUT and MAX BISECTION. Algorithmica **18**(1), 67–81 (1997)
20. Fügenschuh, M.: Relaxations and Solutions for the Minimum Graph Bisection Problem. Phd thesis, Darmstadt University of Technology, Darmstadt, Germany (2007)
21. Garey, M.R., Johnson, D.S.: Computers and Intractability. W. H. Freeman and Company, New York (1979)
22. Gilbert, J.R., Tarjan, R.E.: The analysis of a nested dissection algorithm. Numer. Math. **50**, 377–404 (1987)
23. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM **42**, 1115–1145 (1995)
24. Helmberg, C.: Fixing variables in semidefinite relaxations. SIAM J. Matrix Anal. Appl. **21**(3), 952–969 (2000)
25. Helmberg, C.: 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 (2000)
26. Helmberg, C.: A cutting plane algorithm for large scale semidefinite relaxations. In: Grötschel, M. (ed.) The Sharpest Cut, MPS-SIAM Series on Optimization, pp. 233–256. SIAM/MPS (2004)
27. Helmberg, C.: ConicBundle 0.3. Fakultät für Mathematik, Technische Universität Chemnitz. http://www.tuchemnitz.de/~helmberg/ConicBundle (2009)
28. Helmberg, C., Kiwiel, K.C.: A spectral bundle method with bounds. Math. Progr. **93**(2), 173–194 (2002)
29. Helmberg, C., Rendl, F.: Solving quadratic (0,1)-problems by semidefinite programs and cutting planes. Math. Progr. **82**(3), 291–315 (1998)
30. Helmberg, C., Rendl, F.: A spectral bundle method for semidefinite programming. SIAM J. Optim. **10**(3), 673–696 (2000)
31. ILOG, S.A.: 9 Rue de Verdun, 94253 Gentilly Cedex, France. ILOG AMPL CPLEX System, Version 12.1, User's Guide (2008)
32. Johnson, E., Mehrotra, A., Nemhauser, G.: Min-cut clustering. Math. Progr. **62**, 133–152 (1993)
33. Jünger, M., Martin, A., Reinelt, G., Weismantel, R.: Quadratic 0/1 optimization and a decomposition approach for the placement of electronic circuits. Math. Progr. **63**(3), 257–279 (1994)
34. Karypis, G., Kumar, V.: MeTiS: a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, version 4.0. Technical report, Department of Computer Science, University of Minnesota. http://glaros.dtc.umn.edu/gkhome/views/metis (1998)
35. Laurent, M., de Souza, C.C.: Some new classes of facets for the equicut polytope. Discrete Appl. Math. **62**, 167–191 (1995)
36. Lengauer, T.: Combinatorial Algorithms for Integrated Circuit Layout. Wiley, Chichester (1990)
37. Lovász, L.: On the Shannon capacity of a graph. IEEE Trans. Inf. Theory **IT-25**(1), 1–7 (1979)
38. Mitchell, J.E.: Computational experience with an interior point cutting plane algorithm. SIAM J. Optim. **10**(4), 1212–1227 (2000)
39. Poljak, S., Rendl, F.: Nonpolyhedral relaxations of graph-bisection problems. SIAM J. Optim. **5**(3), 467–487 (1995)
40. Rendl, F., Rinaldi, G., Wiegele, A.: Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. Math. Progr. **121**(2), 307–335 (2010)
41. Weismantel, R.: On the 0/1 knapsack polytope. Math. Progr. **77**, 49–68 (1997)