Math. Program., Ser. A (2014) 143:61–86
DOI 10.1007/s10107-012-0594-z

FULL LENGTH PAPER

Improved semidefinite bounding procedure for solving Max-Cut problems to optimality

Nathan Krislock · Jérôme Malick · Frédéric Roupin

Received: 26 January 2012 / Accepted: 13 September 2012 / Published online: 13 October 2012

Abstract We present an improved algorithm for finding exact solutions to Max-Cut and the related binary quadratic programming problem, both classic problems of combinatorial optimization. The algorithm uses a branch-(and-cut-)and-bound paradigm, using standard valid inequalities and nonstandard semidefinite bounds. More specifically, we add a quadratic regularization term to the strengthened semidefinite relaxation in order to use a quasi-Newton method to compute the bounds. The ratio of the tightness of the bounds to the time required to compute them depends on two real parameters; we show how adjusting these parameters and the set of strengthening inequalities gives us a very efficient bounding procedure. Embedding our bounding procedure in a generic branch-and-bound platform, we get a competitive algorithm: extensive experiments show that our algorithm dominates the best existing method.

N. Krislock (✉)
INRIA Grenoble Rhône-Alpes, Grenoble, France

N. Krislock
PIMS Postdoctoral Fellow in the Department of Computer Science,
University of British Columbia, Vancouver, BC, Canada

N. Krislock
Department of Mathematical Sciences, Northern Illinois University, DeKalb, IL, USA
e-mail: krislock@math.niu.edu

J. Malick
CNRS, Laboratory J. Kunztmann, Grenoble, France
e-mail: jerome.malick@inria.fr

F. Roupin
Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS (UMR 7030), F-93430, Villetaneuse, France
e-mail: frederic.roupin@lipn.univ-paris13.fr

Keywords Combinatorial optimization · Semidefinite programming · Quasi-Newton algorithm · Triangle inequalities

Mathematics Subject Classification 90C22 · 90C27 · 90C57

1 Introduction

Maximizing a quadratic function over the vertices of an hypercube is an important problem of discrete optimization. As far as formulation is concerned, it is the simplest problem of nonlinear (mixed-)integer programming. However, this problem is NP-hard [27] and it is considered a computational challenge to be solved to optimality, even for instances of moderate size.

The Max-Cut problem is one of the most famous NP-hard problems of this type, as is evidenced by its consideration in such seminal works as [16] and [12], and in the survey chapter [25] of the recent handbook [2]. Given a graph $G = (V, E)$ with edge weights w_{ij} for $ij \in E$ and $w_{ij} = 0$ for $ij \notin E$, Max-Cut is the problem of finding a bipartition of the nodes V such that the sum of the weights of the edges across the bipartition is maximized. Let $n = |V|$ be the cardinality of V; we can state Max-Cut as

$$\begin{array}{ll} \text{maximize} & \displaystyle\sum_{i \le j} w_{ij} \left(\frac{1 - x_i x_j}{2} \right) \\ \text{subject to} & x \in \{-1, 1\}^n . \end{array} \tag{1}$$

We can rewrite the problem of Max-Cut as

$$\text{(MC)} \qquad \begin{array}{ll} \text{maximize} & x^T Q x \\ \text{subject to} & x \in \{-1, 1\}^n \end{array} \tag{2}$$

where the matrix Q is defined as $Q := \frac{1}{4} L$, and L is the Laplacian matrix of the weighted graph G; see, e.g., [3].

There have been many methods for finding exact solutions of Max-Cut using semidefinite programming (SDP), including early efforts of [14], the QCR method [4], and most recently the state-of-the-art Biq Mac method [29]. This paper builds on this line of research: we present an improved branch-and-bound algorithm inspired by the above mentioned approaches and introducing new techniques. As shown by the extensive numerical experiments of this paper, our new algorithm dominates the best existing methods.

The algorithm uses a branch-and-(cut-and)-bound paradigm, using the standard valid triangle inequalities together with nonstandard semidefinite bounds. Bounds of the same type have been used recently in several papers (for general binary quadratic problems [20,21] and specifically for the k-cluster problem [22]). More precisely, we apply here the bounds to the Max-Cut problem that have been introduced in [21] for general quadratic optimization problems with quadratic and linear constraints; however, we provide a different motivation for these bounds, looking at them from a

quadratic penalty point of view (see Sect. 3). This reveals the key role of two positive parameters, α and ε, controlling the tightness of the bound.

Our main contribution is an improved bounding procedure obtained by reducing the α and ε parameters to zero and iteratively adding triangle inequality cuts (see Sect. 4). We show that our bounding procedure converges to the classic reinforced semidefinite bound for the Max-Cut problem, in an efficient and robust way.

We embed our bounding procedure within a generic branch-and-bound platform for solving Max-Cut to optimality (see Sect. 5). We compare the overall algorithm with the best existing exact resolution method, the Biq Mac solver [29] on the 328 instances in the Biq Mac Library [31] with size $n < 500$. To prove unambiguously that the new bounding procedure is of high practical value, we make a fair and complete computational study between the two codes: we have compiled and run both our code and the Biq Mac code (kindly provided by the authors of Biq Mac) on the same machine, and have used the same libraries and compilation flags for both codes.

We finish this section with a remark about our choice to focus on Max-Cut for presenting our approach. With the help of classic reformulation techniques, more general binary quadratic optimization problems can be recast as Max-Cut. We could, for example, add a linear term to the objective in problem (2), or use variables in $\{0, 1\}$. For instance, the binary homogeneous quadratic optimization problem

$$\begin{array}{ll} \text{minimize} & x^T Q x \\ \text{subject to} & x \in \{0, 1\}^n \end{array} \tag{3}$$

can be reformulated as Max-Cut by considering a change of variable and by increasing the size of the problem by one; see, e.g., [14, Section 2]. Therefore, to keep the presentation as simple as possible, we only present our developments for the Max-Cut problem (1). On the other hand, we consider both Max-Cut problems and binary quadratic optimization problems in our numerical experiments.

2 Semidefinite relaxations and Biq Mac

In this section, we introduce notation, briefly recall the classic semidefinite relaxation of Max-Cut (see, e.g., [19,28]), and provide a sketch of the Biq Mac method [29], the method to which we will compare.

Semidefinite formulation and relaxation. The inner product of two matrices X and Y is defined and denoted by $\langle X, Y\rangle = \text{trace}\,(X^T Y)$. The notation $X \succeq 0$ means that X is symmetric positive semidefinite. Introducing the symmetric positive semidefinite rank-one matrix $X = xx^T$, we observe that we can write the Max-Cut problem as

$$\text{(MC)} \qquad \begin{array}{ll} \text{maximize} & \langle Q, X\rangle \\ \text{subject to} & \text{diag}(X) = e,\ X \succeq 0, \\ & X = xx^T, \end{array} \tag{4}$$

where $\text{diag}(X)$ is the vector of the diagonal entries of the matrix X, and the vector of all ones is represented by e—for simplicity, we allow the size of the vector e to

be determined from the context. The classic semidefinite relaxation is obtained by dropping the nonconvex rank-one constraint in problem (4):

$$\text{(SDP)} \qquad \begin{array}{ll} \text{maximize} & \langle Q, X\rangle \\ \text{subject to} & \operatorname{diag}(X) = e,\ X \succeq 0. \end{array} \tag{5}$$

Since the feasible set of problem (5) is strictly larger than that of problem (4), we can easily see that the optimal value of problem (5) provides an upper bound on the weight of the optimal cut:

$$(\text{MC}) \leq (\text{SDP}).$$

There exist some theoretical results quantifying the tightness of the above inequality (see, in particular, the seminal work [12]). However, in practice, the ratio of the tightness of the bound to the time needed to compute the bound is too low to allow this bound be used efficiently in branch-and-bound schemes to solve Max-Cut problems to optimality (see, e.g., [14,29,30]).

Strengthening inequalities. In order to improve the performance of the bound, it is necessary to tighten the bounds by adding valid inequalities to the SDP relaxation (5). The most popular class of inequalities are the triangle inequalities, defined for $1 \leq i < j < k \leq n$ by

$$\begin{aligned} X_{ij} + X_{ik} + X_{jk} &\geq -1, \\ X_{ij} - X_{ik} - X_{jk} &\geq -1, \\ -X_{ij} + X_{ik} - X_{jk} &\geq -1, \\ -X_{ij} - X_{ik} + X_{jk} &\geq -1, \end{aligned}$$

(see [14, Section 4] and [29, Section 3.2]). They represent the fact that for any 3-cycle of vertices in the graph, we can only have an even number of edges cut (i.e., either no edges are cut, or exactly two edges are cut). There are

$$4\binom{n}{3} = 4\left(\frac{n(n-1)(n-2)}{6}\right)$$

triangle inequalities in total. Adding any subset of those valid inequalities can strengthen the SDP bound (5). In particular, let I be a subset of the triangle inequalities, and $A_I : \mathbb{S}^n \to \mathbb{R}^{|I|}$ be the corresponding linear operator describing the inequalities in this subset. We define the (SDP_I) problem as

$$(\text{SDP}_I) \qquad \begin{array}{ll} \text{maximize} & \langle Q, X\rangle \\ \text{subject to} & \operatorname{diag}(X) = e,\ X \succeq 0, \\ & A_I(X) + e \geq 0. \end{array} \tag{6}$$

Adding such valid constraints gives an improved bound on the value of the maximum cut:

$$(\text{MC}) \leq (\text{SDP}_I) \leq (\text{SDP}). \tag{7}$$

Note that the dual of problem (6) is given by

$$\begin{array}{ll} \text{minimize} & e^T y + e^T z \\ \text{subject to} & Q - \text{Diag}(y) + A_I^*(z) \preceq 0, \\ & z \geq 0, \end{array} \tag{8}$$

where $\text{Diag}(y)$ is the diagonal matrix with the vector y along its diagonal, and A_I^* is the adjoint linear operator of A_I.

Let us call $(\text{SDP}_{I_{\text{all}}})$ the problem with all the triangle inequalities added, which gives the best possible (SDP_I) bound. Unfortunately, since there is a very large number of triangle inequalities, the $(\text{SDP}_{I_{\text{all}}})$ problem is intractable even for moderate values of n. However, we can get this bound by using only the inequalities that are active at an optimal solution of $(\text{SDP}_{I_{\text{all}}})$. Such a set of active inequalities is obviously unknown, but we will see in Sect. 4 how we approximate it (see Algorithm 1 and Theorem 1). For the moment, we just note that in practice we have to select a (small) number of promising inequalities.

In addition to the triangle inequalities, other cutting planes could be used to tighten the relaxation: hypermetric inequalities [14], general gap inequalities [18], or even a sophisticated choice based on the objective function [9]. The nature of the valid inequalities has no impact on the theoretical developments of the next section; we will deal with a general set of inequalities I. However, in the computational experiments, as is done in [29], we only use triangle inequalities as it greatly simplifies the selection of inequalities and leads to satisfactory results.

Best existing method. Biq Mac [29,30] is currently the best solver for solving Max-Cut problems to optimality. This method uses a branch-and-bound approach and solves the strengthened SDP relaxation (SDP_I) in Eq. (6) with a nonsmooth optimization algorithm. By dualizing the triangle inequality constraints in problem (6), the authors of the Biq Mac solver obtain the nonsmooth convex dual function

$$\theta_I(z) = e^T z \; + \; \max \left\{ \left\langle Q + A_I^*(z), X \right\rangle : \; \text{diag}(X) = e, \; X \succeq 0 \right\} \tag{9}$$

where $z \in \mathbb{R}_+^{|I|}$ is the dual variable. The value $\theta_I(z)$ provides an upper bound on the value of the maximum cut for each $z \in \mathbb{R}_+^{|I|}$; the goal is to minimize $\theta_I(z)$ to obtain the tightest such upper bound. The two main ingredients of the bounding procedure of Biq Mac are as follows:

1. the function value $\theta_I(z)$ and a subgradient $g \in \partial\theta_I(z)$ are evaluated by solving the semidefinite programming problem in Eq. (9) by a customized primal-dual interior-point method;
2. the nonsmooth convex function θ_I is minimized by a bundle method [15].

The extensive numerical experiments of [29,30] show that Biq Mac dominates all other approaches (see the six tested approaches in [29, Section 7]).

3 Our semidefinite bounds

We present the family of semidefinite bounds that we will use in our branch-and-bound method for solving Max-Cut to optimality. By construction, these bounds are less tight than corresponding usual SDP bounds (see Sect. 3.1). On the other hand, Sect. 3.2 presents their useful properties that will lead us to the bounding procedure of the next section (Algorithm 1) whose bounds converge to the usual SDP bounds in the limit.

First we need some more notation. For any matrix A, we denote by $\|A\|_F$ the Frobenius norm of A, which is defined by $\|A\|_F = \sqrt{\langle A, A\rangle}$. For a real number a, we denote its nonnegative part by $a_+ = \max\{a, 0\}$. We extend this definition to vectors as follows: if $x \in \mathbb{R}^n$, then $(x_+)_i = (x_i)_+$, for $i = 1, \ldots, n$. For a given symmetric matrix A, we denote its positive semidefinite part by A_+; more specifically, if the eigendecomposition of A is given by $A = U\,\mathrm{Diag}(\lambda)U^T$, with the eigenvalues $\lambda \in \mathbb{R}^n$, and orthogonal matrix $U \in \mathbb{R}^{n\times n}$, then we have

$$A_+ = U\,\mathrm{Diag}(\lambda_+)U^T.$$

We denote similarly a_-, x_-, and A_-.

3.1 The family of semidefinite bounds

We begin with the following simple fact.

Lemma 1 *Let $0 \le \varepsilon \le 1$ and $X \succeq 0$ such that all the diagonal entries X_{ii} lie in $[1-\varepsilon, 1+\varepsilon]$. Then $(1-\varepsilon)^2 n \le \|X\|_F^2 \le (1+\varepsilon)^2 n^2$.*

Proof Let $X \succeq 0$ as defined above. We have

$$\|X\|_F^2 = \sum_{ij} X_{ij}^2 = \sum_i X_{ii}^2 + \sum_{i\neq j} X_{ij}^2 \ge \sum_i X_{ii}^2 \ge n(1-\varepsilon)^2,$$

Now take (i, j) and extract the submatrix whose determinant is nonnegative

$$\det \begin{pmatrix} X_{ii} & X_{ij} \\ X_{ij} & X_{jj} \end{pmatrix} = X_{ii}X_{jj} - X_{ij}^2 \ge 0.$$

Thus $X_{ij}^2 \le X_{ii}X_{jj} \le (1+\varepsilon)^2$. Also, for each i, we have $X_{ii}^2 \le (1+\varepsilon)^2$, so that $\|X\|_F^2 = \sum_{ij} X_{ij}^2 \le (1+\varepsilon)^2 n^2$. □

A particular case of the above result is when X is feasible in (SDP_I): we have $\mathrm{diag}(X) = e$, so we take $\varepsilon = 0$ in Lemma 1 to get

$$n^2 - \|X\|_F^2 \ge 0.$$

Adding a multiple of this nonnegative term to the objective function, we obtain the regularized problem

$$(\mathrm{SDP}_I^\alpha) \quad \begin{array}{ll} \text{maximize} & \langle Q, X\rangle + \frac{\alpha}{2}\left(n^2 - \|X\|_F^2\right) \\ \text{subject to} & \mathrm{diag}(X) = e,\ X \succeq 0, \\ & A_I(X) \geq -e. \end{array} \tag{10}$$

Proposition 1 *The following holds: for all I and for all $\alpha \geq 0$,*

$$(\mathrm{MC}) \leq (\mathrm{SDP}_I) \leq \left(\mathrm{SDP}_I^\alpha\right). \tag{11}$$

If $\alpha = 0$, the second inequality is an equality. If $\alpha > 0$, the two inequalities are equalities if and only if there exists a rank-one optimal solution of (SDP_I) or (SDP_I^α). Moreover, we have

$$\alpha' \leq \alpha \Rightarrow \left(\mathrm{SDP}_I^{\alpha'}\right) \leq \left(\mathrm{SDP}_I^\alpha\right), \tag{12}$$

and

$$I' \subseteq I \Rightarrow \left(\mathrm{SDP}_{I'}^\alpha\right) \geq \left(\mathrm{SDP}_I^\alpha\right). \tag{13}$$

Proof Inequality (13) clearly holds since the feasible set of (SDP_I^α) is included in that of $(\mathrm{SDP}_{I'}^\alpha)$. We now prove (12), from which we can deduce the remaining easily. For any feasible matrix X in problem (10), we have $n^2 - \|X\|_F^2 \geq 0$. Therefore $\alpha' \leq \alpha$ yields

$$\langle Q, X\rangle + \alpha'\left(n^2 - \|X\|_F^2\right) \leq \langle Q, X\rangle + \alpha\left(n^2 - \|X\|_F^2\right)$$

which gives $(\mathrm{SDP}_I^{\alpha'}) \leq (\mathrm{SDP}_I^\alpha)$. Taking now $\alpha' = 0$ gives the second inequality in (11); the first inequality comes from (7). For the case of equality, note first from problem (4) that $(\mathrm{MC}) = (\mathrm{SDP}_I)$ if and only if there is a rank-one optimal solution of (SDP_I). Finally, we use a corollary of [20, Theorem 1] which states that if $X \succeq 0$ and $\mathrm{diag}(X) = e$, then $\|X\|_F = n$ if and only if $\mathrm{rank}\,(X) = 1$, completing the proof. □

This proposition says, first, that the bound (SDP_I^α) is strictly weaker than (SDP_I) (except in the very special situation that (SDP_I) has a rank-one optimal solution), and, second, that making α smaller reduces the difference. We will come back to this second point when describing our algorithm where we gradually decrease α.

Let us now fix α and I, and let us dualize the affine constraints of (SDP_I^α). We define the Lagrangian function with respect to the primal variable $X \succeq 0$ and the dual variables $y \in \mathbb{R}^n$ and $z \in \mathbb{R}_+^{|I|}$ as

$$\begin{aligned} L(X; y, z) &:= \langle Q, X\rangle + \frac{\alpha}{2}\left(n^2 - \|X\|_F^2\right) + \langle y, e - \mathrm{diag}(X)\rangle + \langle z, e + A_I(X)\rangle \\ &= \left\langle Q - \mathrm{Diag}(y) + A_I^*(z), X\right\rangle - \frac{\alpha}{2}\|X\|_F^2 + e^T y + e^T z + \frac{\alpha}{2}n^2. \end{aligned}$$

The associated dual function is then defined by

$$F_I^\alpha(y,z) := \max_{X\succeq 0} L(X;y,z). \tag{14}$$

By weak duality, we have an upper bound on (MC) for given (I,α,y,z) with $\alpha \geq 0$ and $z \geq 0$. More precisely, we have

$$(\text{MC}) \leq (\text{SDP}_I) \leq \left(\text{SDP}_I^\alpha\right) \leq F_I^\alpha(y,z). \tag{15}$$

These $F_I^\alpha(y,z)$ for given (I,α,y,z) are the bounds that we are going to use to solve Max-Cut to optimality; we study them in the next section.

3.2 Mathematical study of the bounds $F_I^\alpha(y,z)$

This first result gives us a useful explicit expression of the bound $F_I^\alpha(y,z)$ for $\alpha > 0$. Let us consider the positive semidefinite matrix

$$X_I(y,z) := \left[Q - \text{Diag}(y) + A_I^*(z)\right]_+. \tag{16}$$

Proposition 2 *Let $\alpha > 0$ and let I be a set of inequalities. Then, for F_I^α and X_I defined in Eqs. (14) and (16), respectively, we have*

$$F_I^\alpha(y,z) = \frac{1}{2\alpha}\|X_I(y,z)\|_F^2 + e^T y + e^T z + \frac{\alpha}{2}n^2, \qquad \text{for all } y,z. \tag{17}$$

Proof Let y and z be given, and let $M := Q - \text{Diag}(y) + A_I^*(z)$ and $f(X) := \langle M, X\rangle - \frac{\alpha}{2}\|X\|_F^2$. Then we have by Eq. (14) that

$$F_I^\alpha(y,z) = \max_{X\succeq 0} f(X) + e^T y + e^T z + \frac{\alpha}{2}n^2. \tag{18}$$

Since $\alpha > 0$, we have that $\max_{X\succeq 0} f(X)$ is a convex optimization problem. The dual problem of $\max_{X\succeq 0} f(X)$ is given by

$$\min_{S\succeq 0}\max_X f(X) + \langle S, X\rangle.$$

Since $\max_{X\succeq 0} f(X)$ is strictly feasible, strong duality holds (see, e.g., [7]); therefore,

$$\max_{X\succeq 0} f(X) = \min_{S\succeq 0}\max_X f(X) + \langle S, X\rangle.$$

Let $S \succeq 0$ be fixed. Since $\nabla f(X) = M - \alpha X$, the maximum of $f(X) + \langle S, X\rangle$ over all X is attained when $X = \frac{1}{\alpha}(M+S)$. Therefore,

$$\max_X f(X) + \langle S, X\rangle = \left\langle M, \frac{1}{\alpha}(M+S)\right\rangle - \frac{\alpha}{2}\left\|\frac{1}{\alpha}(M+S)\right\|_F^2 + \left\langle S, \frac{1}{\alpha}(M+S)\right\rangle.$$

Simplifying the expression on the right-hand-side, we have

$$\max_X f(X) + \langle S, X\rangle = \frac{1}{2\alpha}\,\|M+S\|_F^2\,.$$

Therefore,

$$\max_{X\succeq 0} f(X) = \min_{S\succeq 0}\max_X f(X) + \langle S, X\rangle = \min_{S\succeq 0}\frac{1}{2\alpha}\,\|M+S\|_F^2\,.$$

This last problem asks what is the nearest positive semidefinite matrix S to the matrix $-M$. The answer to this question is $S = (-M)_+$. Furthermore, it is not hard to show that $M + (-M)_+ = M_+$, so we have that

$$\max_{X\succeq 0} f(X) = \frac{1}{2\alpha}\,\|M+(-M)_+\|_F^2 = \frac{1}{2\alpha}\,\|M_+\|_F^2 = \frac{1}{2\alpha}\,\|X_I(y,z)\|_F^2\,.$$

Substituting this back into Eq. (18), we have the result in Eq. (17). □

This proposition makes it clear that the bounds that we consider here correspond (up to a change of sign and notation) to the ones introduced in [21] for general 0-1 quadratic optimization problems (see $\Theta(\lambda,\mu,\alpha)$ in Theorem 2 of [21]). The introduction of the bounds we give here is different though: it is more direct, as it does not rely on the reformulation of the initial problem with the so-called matrix spherical constraint (see [20]).

Fixing α and I, we obtain the best bound $F_I^\alpha(y,z)$ when (y,z) is an optimal solution of the problem

$$\left(\mathrm{DSDP}_I^\alpha\right)\qquad \begin{array}{ll}\text{minimize} & \frac{1}{2\alpha}\,\|X_I(y,z)\|_F^2 + e^T y + e^T z + \frac{\alpha}{2}n^2\\ \text{subject to} & y \text{ free},\ z\geq 0,\end{array} \tag{19}$$

which is the dual problem of (SDP_I^α).

Observe that when $\alpha = 0$ the dual function has the usual expression

$$F_I^0(y,z) = \begin{cases} e^T y + e^T z, & \text{if } Q - \mathrm{Diag}(y) + A_I^*(z) \preceq 0,\\ +\infty, & \text{otherwise,}\end{cases}$$

which leads us back to the dual problem (8). We see that the condition

$$Q - \mathrm{Diag}(y) + A_I^*(z) \preceq 0$$

is exactly $X_I(y,z) = 0$, which opens another view on the approach we consider. Indeed, from Eq. (17) the function $F_I^\alpha(y,z)$ can be viewed as a penalization of the standard dual function $F_I^0(y,z)$. The parameter α is interpreted as a penalization parameter controlling the loss in the tightness of the bound.

In view of solving problem (19), the next proposition is crucial.

Proposition 3 *Let $\alpha > 0$. Then the dual function F_I^α is convex and differentiable, with partial gradients given by*

$$\nabla_y F_I^\alpha(y,z) = e - \frac{1}{\alpha}\, diag\,(X_I(y,z)),$$
$$\nabla_z F_I^\alpha(y,z) = e + \frac{1}{\alpha} A_I\,(X_I(y,z))\,.$$

Proof Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be defined by $f(x) = \frac{1}{2}\|x_+\|_2^2$. Then f is a convex and differentiable function invariant under permutations of the entries of x. The gradient of f is $\nabla f(x) = x_+$. From [6, Sect. 5.2], we have that the function $g\colon \mathbb{S}^n \to \mathbb{R}$ defined by $g(X) = f(\lambda(X))$ is convex and differentiable with gradient

$$\nabla g(X) = U\,\mathrm{Diag}(\nabla f(\lambda(X)))U^T,$$

where U is any $n \times n$ orthogonal matrix satisfying $X = U\,\mathrm{Diag}(\lambda(X))U^T$; thus $\nabla g(X) = X_+$. Now note that

$$F_I^\alpha(y,z) = \frac{1}{\alpha} g\left(Q - \mathrm{Diag}(y) + A_I^*(z)\right) + e^T y + e^T z + \frac{\alpha}{2} n^2,$$

so we immediately have that F_I^α is convex and differentiable. Now we simply apply the chain rule, $\nabla_x\,[g(Mx)] = M^*\nabla g(Mx)$, where $M\colon \mathbb{R}^p \to \mathbb{S}^n$ is any linear operator, to get the desired result. □

As expected, this proposition is in the same vein as [21, Theorem 2] establishing differentiability results for the corresponding bounds. The proof given here is original though, using the differentiability of spectral functions as a direct argument.

The final proposition considers what could happen to the value of the dual function F_I^α when reducing the value of α with fixed variables y and z.

Proposition 4 *Let $0 \le \varepsilon \le 1$ and $0 < \alpha' < \alpha$. Let (y,z) such that $\left\|\nabla_y F_I^\alpha(y,z)\right\|_\infty \le \varepsilon$. Then*

$$F_I^{\alpha'}(y,z) \;\le\; F_I^\alpha(y,z) + \frac{n^2}{2}\frac{(\alpha-\alpha')}{\alpha'}\left(\alpha(1+\varepsilon)^2 - \alpha'\right)$$

and

$$F_I^{\alpha'}(y,z) \;\ge\; F_I^\alpha(y,z) + \frac{n^2}{2}\frac{(\alpha-\alpha')}{\alpha'}\left(\frac{\alpha}{n}(1-\varepsilon)^2 - \alpha'\right).$$

Proof We have

$$F_I^{\alpha'}(y,z) - F_I^\alpha(y,z) = \frac{1}{2}\left(\frac{1}{\alpha'} - \frac{1}{\alpha}\right)\|X_I(y,z)\|_F^2 - \frac{n^2}{2}(\alpha - \alpha').$$

We apply Lemma 1 to $X = \frac{1}{\alpha} X_I(y,z)$, which satisfies $\max_i |X_{ii} - 1| \le \varepsilon$ by Proposition 3 and the assumption $\left\|\nabla_y F_I^\alpha(y,z)\right\|_\infty \le \varepsilon$. This gives

$$\alpha^2(1-\varepsilon)^2 n \leq \|X_I(y,z)\|_F^2 \leq \alpha^2(1+\varepsilon)^2 n^2.$$

We deduce the first desired bound, as

$$\begin{aligned} F_I^{\alpha'}(y,z) - F_I^{\alpha}(y,z) &\leq \frac{1}{2}\left(\frac{1}{\alpha'} - \frac{1}{\alpha}\right)\alpha^2(1+\varepsilon)^2 n^2 - \frac{n^2}{2}(\alpha - \alpha') \\ &= \frac{n^2}{2}\frac{(\alpha-\alpha')}{\alpha'}\left(\alpha(1+\varepsilon)^2 - \alpha'\right). \end{aligned}$$

Using the other inequality, we get in a similar way the other bound. □

This result tells us that $F_I^{\alpha'}(y,z)$ may increase over $F_I^{\alpha}(y,z)$, but not much if $\alpha - \alpha'$ is small enough and α' is not too small. This gives the intuition that we should not decrease α too much or too quickly. In our numerical experiments, we take $\alpha' = \frac{\alpha}{2}$ and never take α smaller than 10^{-5} (see Sect. 5.2). Later we will present Lemma 3 which will give us a more precise picture.

3.3 Computing the bounds

For a given set of inequalities I and a tightness level $\alpha > 0$, the computation of the value of the bounding function F_I^{α} and its gradient essentially corresponds to the computation of $X_I(y,z)$, which, in turn, reduces to computing the positive eigenvalues and corresponding eigenvectors of the symmetric matrix $Q - \mathrm{Diag}(y) + A_I^*(z)$ (see Eq. 16). To compute this partial eigenvalue decomposition, we use the LAPACK [1] routine `DSYEVR`—for improved performance we use the Intel Math Kernel Library (MKL) rather than the Automatically Tuned Linear Algebra Software (ATLAS) package. When requesting a partial eigenvalue decomposition of a matrix, `DSYEVR` first tridiagonalizes the matrix, then finds the positive eigenvalues using bisection, and finally uses the inverse iteration method to find the corresponding eigenvectors; see [13] for more information.

In view of the differentiability result (Proposition 3), we can solve problem (19) with any classic nonlinear optimization algorithm that can handle nonnegativity constraints. Among all possible algorithms and software, quasi-Newton methods are known to be efficient [5,24]. In our experiments, we use the FORTRAN code L-BFGS-B [8,32] which has been recently upgraded to version 3.0 [23]. We use the default parameters of the code. Note, though, that we do an initial scaling of the problem by normalizing the constraints we dualize.

In summary, the two main ingredients needed to use our bounds in practice are the following:

1. the function value $F_I^{\alpha}(y,z)$ and gradient $\nabla F_I^{\alpha}(y,z)$ are evaluated by computing a single partial eigenvalue decomposition;
2. the smooth convex function F_I^{α} is minimized by a quasi-Newton method.

Thus, our method is built using reliable and efficient numerical codes: the eigensolver `DSYEVR` and the quasi-Newton solver L-BFGS-B.

4 Improved semidefinite bounding procedure

This section presents the bounding procedure using the bounds $F_I^\alpha(y,z)$ introduced in the previous section, provides a theoretical analysis, and finally a numerical illustration.

4.1 Description of the bounding procedure

We have three ways to improve the tightness of the bound $F_I^\alpha(y,z)$: adding triangle inequalities, reducing the parameter α, or reducing the tolerance ε in the stopping criteria of the quasi-Newton method:

1. *Adding inequalities.* In view of (13), we can improve the bound by adding violated (triangle) inequalities to I; that is,

$$\text{find } i \text{ such that } A_i(X) + 1 < 0$$

   where $X = \frac{1}{\alpha} X_I(y,z)$. Recall that we only need to add the triangle inequalities that are active at the optimal solution that satisfies all triangle inequalities. We do this by adding a predefined number of the most violated inequalities each time to improve the bound as quickly as possible. We add just a few inequalities at the beginning when we are far from the optimal solution, then we add more inequalities when we are closer to this optimal solution. Moreover, we borrow an idea from the description of Biq Mac in [30] where, instead of using the current iterate X_k to find violated triangle inequalities, a convex combination of the current iterate with the previous iterate is used: $X^{\text{test}} = \lambda X_{k-1} + (1-\lambda)X_k$. However, while Biq Mac uses $\lambda = 0.9$, we take the opposite strategy and use $\lambda = 0.2$ since we found this works well in our code.
2. *Reducing* α. We can reduce $\alpha > 0$ and solve problem (19) again, warm-starting with the current (y,z). In practice, we reduce α if the number of violated triangle inequalities is small for the current value of α. In other words, our strategy consists of changing the target when we can no longer make good progress by adding inequalities.
3. *Reducing* ε. We can request more accuracy from the quasi-Newton method by reducing the tolerance $\varepsilon > 0$ in the stopping test:

$$\max\left\{ \left\| \nabla_y F_I^\alpha(y,z) \right\|_\infty, \ \left\| [\nabla_z F_I^\alpha(y,z)]_- \right\|_\infty \right\} < \varepsilon.$$

   Note that the expressions of the gradients of F_I^α (Proposition 3) gives

$$\max\left\{ \left\| \operatorname{diag}\left(\frac{1}{\alpha} X_I(y,z)\right) - e \right\|_\infty, \ \left\| \left[A_I\left(\frac{1}{\alpha} X_I(y,z)\right) + e \right]_- \right\|_\infty \right\} < \varepsilon.$$

How we control the decrease of ε and α, and how we add inequalities, is important for the overall efficiency. Our algorithm, described formally in Algorithm 1, can be viewed as a succession of cutting plane methods (alternating adding cuts and reducing

Algorithm 1 Improved semidefinite bounding algorithm

Input: Scalars $\alpha_1 > 0$, $\varepsilon_1 > 0$, and parameters $0 < \texttt{ScaleAlpha}, \texttt{ScaleEps} < 1$.

$y_0 \leftarrow 0 \in \mathbb{R}^n$ {initial y variables} and $z_0 \leftarrow 0 \in \mathbb{R}^0$ {initial z variables}

$I_0 \leftarrow \emptyset$ {initial set of triangle inequalities I}

for $k = 1, 2, \ldots$ **do**

Starting from (y_{k-1}, z_{k-1}), use a descent method (e.g., a quasi-Newton method) to compute $(y_k, \hat{z}_k)$ such that

$$\max\left\{ \left\| \operatorname{diag}(X_k) - e \right\|_\infty,\ \left\| \left[A_{I_{k-1}}(X_k) + e \right]_- \right\|_\infty \right\} < \varepsilon_k,$$

where $X_k \leftarrow \frac{1}{\alpha_k} X_{I_{k-1}}(y_k, \hat{z}_k)$.

Update the bound: $F_k \leftarrow F^{\alpha_k}_{I_{k-1}}(y_k, \hat{z}_k)$.

Remove triangle inequalities that are not active:

$$I^-_{k-1} \leftarrow \left\{ i \in I_{k-1} : (\hat{z}_k)_i = 0 \text{ and } A_i(X_k) + 1 > \varepsilon_k \right\}.$$

Add triangle inequalities that are violated:

Let $X^{\text{test}}_k \leftarrow \lambda X_{k-1} + (1-\lambda) X_k$ (taking $X_0 = X_1$) and let $i_1, \ldots, i_\ell$ be the indices $i \notin I_{k-1}$ such that $A_i(X^{\text{test}}_k) + 1 \le \texttt{ViolTol}_k$, where $\texttt{ViolTol}_k \le 0$, ordered such that $A_{i_1}(X^{\text{test}}_k) \le \cdots \le A_{i_\ell}(X^{\text{test}}_k)$. Let

$$I^+_{k-1} \leftarrow \{i_1, \ldots, i_K\}, \quad \text{where } K = \min\{\ell, \texttt{MaxIneqAdded}_k\}.$$

Update set of inequalities: $I_k \leftarrow \left(I_{k-1} \setminus I^-_{k-1} \right) \cup I^+_{k-1}$.

Initialize multipliers for added inequalities to zero:

$$\text{for each } i \in I_k, \qquad (z_k)_i \leftarrow \begin{cases} (\hat{z}_k)_i, & \text{if } i \in I_{k-1}, \\ 0, & \text{if } i \notin I_{k-1}. \end{cases}$$

if the number of inequalities added, K, is less than $\texttt{MinIneqViol}$ **then**

$\alpha_{k+1} \leftarrow \texttt{ScaleAlpha} \cdot \alpha_k$, $\varepsilon_{k+1} \leftarrow \texttt{ScaleEps} \cdot \varepsilon_k$

else

$\alpha_{k+1} \leftarrow \alpha_k$, $\varepsilon_{k+1} \leftarrow \varepsilon_k$

end if

end for

α and ε). We give details in Sect. 5.1 on the parameters we chose to efficiently interlace the decrease of α and ε with the management of the set of enforced inequalities.

4.2 Convergence of the bounding procedure

We now analyze Algorithm 1, starting with the case when the set of triangle inequalities I is fixed.

Lemma 2 *Let I be a (fixed) set of valid triangle inequalities. Consider five sequences indexed by k: $\alpha_k > 0$, $\varepsilon_k > 0$, $y_k \in \mathbb{R}^n$, $z_k \in \mathbb{R}^{|I|}_+$, $X_k \succeq 0$ such that*

$$\max\left\{ \left\| \nabla_y F^{\alpha_k}_I(y_k, z_k) \right\|_\infty,\ \left\| [\nabla_z F^{\alpha_k}_I(y_k, z_k)]_- \right\|_\infty \right\} < \varepsilon_k, \tag{20}$$

and $X_k := X_I(y_k, z_k)/\alpha_k$. *If both* $(\alpha_k)_k$ *and* $(\varepsilon_k)_k$ *tend to* 0, *and if the sequence* $(X_k, y_k, z_k)_k$ *converges, then its limit is a primal-dual solution of* (SDP$_I$), *and*

$$\lim_{k\to+\infty} F_I^{\alpha_k}(y_k, z_k) = (\text{SDP}_I). \tag{21}$$

Proof First note that the feasible set of problem (6) is closed and bounded (by n), so there exists a primal optimal solution. Moreover, problem (6) satisfies Slater's constraint qualification: the identity matrix satisfies the affine constraints (equalities and inequalities) and strictly satisfies the positive semidefinite constraint. Therefore (see, e.g., [7]), there is no duality gap between between primal-dual problems (6) and (8), and there exists a dual optimal solution. Thus the following conditions are necessary and sufficient for optimality of (X, S, y, z):

$$Q - \text{Diag}(y) + A_I^*(z) = S \tag{22}$$
$$\langle X, S\rangle = 0, \quad X \succeq 0, \quad S \preceq 0 \tag{23}$$
$$\text{diag}(X) = e, \quad A_I(X) + e \geq 0 \tag{24}$$
$$z^T(A_I(X) + e) = 0, \quad z \geq 0 \tag{25}$$

Assume that $(X_k, y_k, z_k)_k$ converges to $(\bar{X}, \bar{y}, \bar{z})$. Let us show that the triple satisfies the above optimality conditions. By construction of X_k,

$$Q - \text{Diag}(y_k) + A_I^*(z_k) = \alpha_k X_k + S_k, \qquad \text{for all } k,$$

where $S_k := \left[Q - \text{Diag}(y_k) + A_I^*(z_k)\right]_-$. The sequence $(S_k)_k$ converges to

$$\bar{S} := \left[Q - \text{Diag}(\bar{y}) + A_I^*(\bar{z})\right]_-,$$

by continuity of the operator $[\cdot]_-$. Passing to the limit shows that $(\bar{X}, \bar{S}, \bar{y}, \bar{z})$ satisfies (22) since α_k vanishes. By construction we also have $\langle S_k, X_k\rangle = 0$, $X_k \succeq 0$, and $S_k \preceq 0$. Since the cones of positive semidefinite and negative semidefinite matrices are closed, passing to the limit, we get that $\bar{X}$ and $\bar{S}$ satisfy (23).

By Proposition 3, condition (20) can be written as

$$\max\left\{\|\text{diag}(X_k) - e\|_\infty,\ \|[A_I(X_k) + e]_-\|_\infty\right\} < \varepsilon_k.$$

Passing to the limit we have that $\bar{X}$ satisfies (24). By assumption, we have that $z_k \geq 0$ for all k, so we also have that $\bar{z} \geq 0$. Finally, we write

$$\left|z_k^T(A_I(X_k) + e)\right| \leq \|z_k\|_1 \|A_I(X_k) + e\|_\infty < \|z_k\|_1 \varepsilon_k.$$

Since ε_k vanishes, at the limit, we have that $\bar{z}$ and $\bar{X}$ satisfy (25). Altogether, this shows that $(\bar{X}, \bar{y}, \bar{z})$ is indeed a primal-dual solution.

Finally (17) yields

$$F_I^{\alpha_k}(y_k, z_k) = e^T y_k + e^T z_k + \alpha_k \left(\|X_k\|^2 + n^2 \right) \Big/ 2.$$

Therefore we have $\lim_{k \to +\infty} F_I^{\alpha_k}(y_k, z_k) = e^T \bar{y} + e^T \bar{z}$. Since there is no duality gap for the primal-dual problems (6) and (8), we conclude that (21) holds. □

The parameters α and ε control the level of tightness of the bound in that smaller values give tighter upper bounds. Furthermore, we can reach the usual SDP bound, (SDP_I), in the limit as α and ε both approach zero. In practice, we interlace the decrease of these control parameters with the process of adding violated inequalities; the overall convergence follows from the next lemma.

Lemma 3 *Let the sequence* $(\alpha_k, \varepsilon_k, X_k, y_k, \hat{z}_k, z_k, I_k, F_k)_k$ *be generated by Algorithm 1. Then the following holds:*

(i) *For all* $k = 1, 2, \ldots$, *we have:*

$$X_k = \frac{1}{\alpha_k} X_{I_k}(y_k, z_k) \quad \text{and} \quad F_k = F_{I_k}^{\alpha_k}(y_k, z_k).$$

(ii) *There exists* $\gamma > 0$ *such that:*

$$F_{k+\ell} \leq F_k + \gamma \alpha_k, \quad \text{for all } k, \ell.$$

(iii) *If* $\alpha_k \to 0$, *then* $(F_k)_k$ *is a convergent sequence.*

Proof The first item follows from the observation that, based on the definition of I_k and z_k in Algorithm 1, we have $A^*_{I_k}(z_k) = A^*_{I_{k-1}}(\hat{z}_k)$ and $e^T z_k = e^T \hat{z}_k$; this follows from the fact that inequalities are removed only if they have a zero multiplier $(\hat{z}_k)_i$, and multipliers for added inequalities are initialized with $(z_k)_i = 0$.

For the second item, first let $r := \texttt{ScaleAlpha}^{-1} > 1$. Since we have the bound $\|\nabla_y F_{I_{k-1}}^{\alpha_k}(y_k, \hat{z}_k)\|_\infty < \varepsilon_k$, Proposition 4 gives:

$$F_{I_{k-1}}^{\alpha_{k+1}}\left(y_k, \hat{z}_k\right) \leq F_{I_{k-1}}^{\alpha_k}\left(y_k, \hat{z}_k\right) + \frac{n^2}{2} \frac{(\alpha_k - \alpha_{k+1})}{\alpha_{k+1}} \left(\alpha_k (1 + \varepsilon_k)^2 - \alpha_{k+1} \right). \tag{26}$$

From the previous item, we have

$$F_{k+1} = F_{I_k}^{\alpha_{k+1}}\left(y_{k+1}, \hat{z}_{k+1}\right) \leq F_{I_k}^{\alpha_{k+1}}(y_k, z_k) = F_{I_{k-1}}^{\alpha_{k+1}}\left(y_k, \hat{z}_k\right),$$

where the inequality follows from the fact that $(y_{k+1}, \hat{z}_{k+1})$ is generated using a descent method starting from (y_k, z_k). Therefore, if $\alpha_{k+1} = \texttt{ScaleAlpha} \cdot \alpha_k = r^{-1} \alpha_k$, by inequality (26) we have

$$F_{k+1} \leq F_k + \frac{n^2}{2}(r - 1) \left((1 + \varepsilon_k)^2 - r^{-1} \right) \alpha_k.$$

Since $0 < \texttt{ScaleEps} < 1$, we have $\varepsilon_k \leq \varepsilon_1$, for all k. Letting

$$\delta := n^2(r-1)\left((1+\varepsilon_1)^2 - r^{-1}\right)\Big/ 2 > 0,$$

we have that

$$\begin{cases} F_{k+1} \leq F_k + \delta\alpha_k, & \text{if } \alpha_{k+1} = \texttt{ScaleAlpha} \cdot \alpha_k, \\ F_{k+1} \leq F_k, & \text{if } \alpha_{k+1} = \alpha_k, \end{cases} \tag{27}$$

for all k. Let ℓ and k be given, and let $k_1, \ldots, k_p$ be the p indices $k \leq k_i < k + \ell$ such that $\alpha_{k_i+1} = \texttt{ScaleAlpha} \cdot \alpha_{k_i}$. Then, from repeated application of the inequalities (27), we obtain

$$\begin{aligned} F_{k+\ell} &\leq F_k + \delta\left(\alpha_{k_1} + \alpha_{k_2} + \cdots + \alpha_{k_p}\right) \\ &= F_k + \delta\left(\alpha_k + r^{-1}\alpha_k + \cdots + r^{-(p-1)}\alpha_k\right) \\ &= F_k + \delta\left(1 + \frac{1}{r} + \cdots + \frac{1}{r^{p-1}}\right)\alpha_k \\ &\leq F_k + \delta\left(\frac{r}{r-1}\right)\alpha_k. \end{aligned}$$

Letting $\gamma := \frac{\delta r}{r-1} > 0$, we obtain the desired result.

For the third item, suppose $\alpha_k \to 0$. Then, by the previous item, we can conclude

$$\limsup_{k\to+\infty} F_k \leq \liminf_{k\to+\infty} F_k,$$

so the sequence $(F_k)_k$ converges. □

Theorem 1 *Let the sequence $(\alpha_k, \varepsilon_k, X_k, y_k, \hat{z}_k, z_k, I_k, F_k)_k$ be generated by Algorithm 1. If $(\alpha_k)_k$ and $(\varepsilon_k)_k$ both converge to zero, and $(\bar{X}, \bar{y}, \bar{z}, \bar{I})$ is an accumulation point of the sequence $(X_k, y_k, z_k, I_k)_k$, then $(\bar{X}, \bar{y}, \bar{z})$ is a primal-dual solution of $(\mathrm{SDP}_{\bar{I}})$, and the sequence F_k converges to the classic semidefinite bound:*

$$\lim_{k\to+\infty} F_{I_k}^{\alpha_k}(y_k, z_k) = (\mathrm{SDP}_{\bar{I}}). \tag{28}$$

Proof Consider a subsequence of $(X_k, y_k, z_k, I_k)_k$ that converges to $(\bar{X}, \bar{y}, \bar{z}, \bar{I})$. Since the I_k are finite sets, the convergence of a subsequence to $\bar{I}$ means that there are infinitely many indexes k_i of this subsequence such that $I_{k_i} = \bar{I}$. Observe now that the sequence $(X_{k_i}, y_{k_i}, z_{k_i})_i$ satisfies the assumptions of Lemma 2 with $I = \bar{I}$. Therefore,

$$\lim_{i\to+\infty} F_{k_i} = \lim_{i\to+\infty} F_{I_{k_i}}^{\alpha_{k_i}}(y_{k_i}, z_{k_i}) = \lim_{i\to+\infty} F_{\bar{I}}^{\alpha_{k_i}}(y_{k_i}, z_{k_i}) = (\mathrm{SDP}_{\bar{I}}).$$

Since $\alpha_k \to 0$, from Lemma 3 we also have that the entire sequence $(F_k)_k$ converges, so we conclude that $\lim_{k\to+\infty} F_k = (\mathrm{SDP}_{\bar{I}})$. □

This result gives an important observation about our bounding procedure Algorithm 1. Although, by Proposition 1, the bounds $F_I^\alpha(y, z)$, for $\alpha > 0$, are weaker than the usual SDP bound (SDP_I), we have, by Theorem 1, that we can get as close as we like using Algorithm 1.

Theorem 1 thus shows that, in the limit, the tightness of the bounds is only governed by the selection of promising inequalities I_k. In the next corollary, we look closely at a special case to show that Algorithm 1 can approximate an "optimal" set of inequalities.

Corollary 1 *Let the assumptions of Theorem 1 hold. If the sequence* $(X_k, y_k, z_k, I_k)_k$ *converges to* $(\bar{X}, \bar{y}, \bar{z}, \bar{I})$ *and if* $\mathtt{ViolTol}_k \to 0$, *then* $(\mathrm{SDP}_{\bar{I}})$ *is equal to the semidefinite bound* $(\mathrm{SDP}_{I_{\mathrm{all}}})$ *with all the inequalities, and* $(\bar{X}, \bar{y}, \tilde{z})$ *is a primal-dual solution of* $(\mathrm{SDP}_{I_{\mathrm{all}}})$, *where* $\tilde{z} \in \mathbb{R}^{4\binom{n}{3}}$ *is obtained from* $\bar{z}$ *by expanding with zeros the entries that are not indexed by* $\bar{I}$.

Proof We use the assumption of the convergence of the whole sequence. For k large enough, the set I_k is equal to $\bar{I}$. Thus no more inequalities are added at iteration k, which means

$$A_i\left(X_k^{\mathrm{test}}\right) + 1 \geq \mathtt{ViolTol}_k, \quad \text{for all } i \notin \bar{I}.$$

Since $(X_k^{\mathrm{test}})_k$ also converges to $\bar{X}$, we pass to the limit to get $A_i(\bar{X}) + 1 \geq 0$ for all $i \notin \bar{I}$. We also know by Theorem 1 that $\bar{X}$ is optimal, hence feasible, for $(\mathrm{SDP}_{\bar{I}})$. Therefore we have that

$$A_i(\bar{X}) + 1 \geq 0, \quad \text{for any inequality } i, \tag{29}$$

and that $\bar{X}$ is in fact feasible for $(\mathrm{SDP}_{I_{\mathrm{all}}})$. We observe now $(\bar{X}, \bar{y}, \tilde{z})$ satisfies the optimality conditions of $(\mathrm{SDP}_{I_{\mathrm{all}}})$ [i.e., conditions (22–25) from the proof of Lemma 2]. Indeed (29) implies that (24) is satisfied for $X = \bar{X}$ and $I = I_{\mathrm{all}}$, and so are (22) and (25) by construction of $\tilde{z}$. Thus we have $(\mathrm{SDP}_I) = (\mathrm{SDP}_{I_{\mathrm{all}}})$ and the rest of the proof follows from Theorem 1. □

4.3 Numerical illustration of the bounding procedure

This section presents a numerical comparison of our bounding procedure Algorithm 1 with the one of Biq Mac [29]. The code for each solver was compiled on the same machine using the same numerical libraries. We take one problem of the Biq Mac Library [31]—specifically Beasley bqp250.6, the problem highlighted in [29, Figure 1]—and we plot the convergence curve for the two bounding procedures in Fig. 1. For this particular problem, note that we have $(\mathrm{SDP}_{I_{\mathrm{all}}}) \approx 41178$ (see [29]) and (MC) = 41014. In our test, our bounding procedure computes a bound of 41210 after 206 s while the Biq Mac bounding procedure is only able to attain a bound of 41251 after 604 s of computing time.

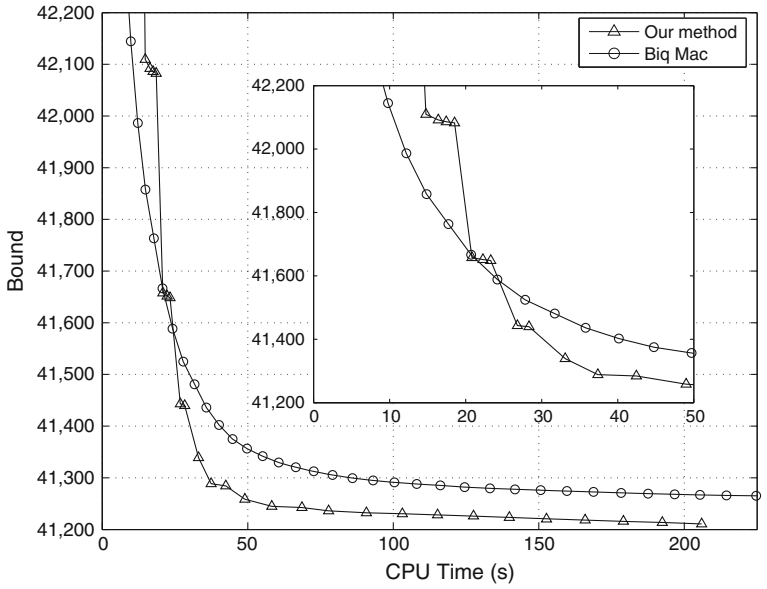

Fig. 1 Comparison of the bounding procedure of Biq Mac [29] and our bounding procedure Algorithm 1 on the Beasley bqp250.6 problem—the behaviour shown here is typical

The plot of Fig. 1 is typical of the behaviour of the two solvers: our bounding procedure has a slower start since we take a large value of α at the beginning, then the convergence curves intersect, after which we see that our bounding procedure is able to compute good bounds in less time than that of Biq Mac. Therefore, the computing time needed for our solver to get to a zone of "useful bounds" is smaller than the one used by Biq Mac—this is crucial for the relative efficiency of our branch-and-bound method and is the reason for the good numerical results of the next section.

Figure 1 also shows large decreases in the value of the dual function on the curve for our bounding procedure: this corresponds to the iterations when α is decreased. We also note that our bounding procedure is able to attain a tighter bound than the one of Biq Mac. This may be due to a combination of: a different selection of inequalities, and the slow convergence of the bundle method near the end of the solving process.

5 Solving Max-Cut to optimality

In this section, we describe our implementation of a branch-and-bound method for solving Max-Cut to optimality. The scheme of our algorithm is simple and follows the usual branch-and(-cut-and)-bound paradigm—the novelty of our approach is essentially our bounding procedure that we described in Algorithm 1. Finally, we provide a complete numerical comparison to the leading Biq Mac solver [29].

5.1 Branch-and-bound implementation

While Biq Mac uses a dedicated code, we embed our bounding procedure in a generic code: we use the BOB Branch & Bound platform [10], which provides an easy and flexible way to implement a branch-and-bound algorithm. The BOB platform only requires the user to implement the following: (1) a bounding procedure, (2) a method for generating a feasible solution, and (3) a method for generating subproblems. Here we give some details about these points.

We have already described our bounding procedure in Sect. 4, but we would like to mention here how we decide when to stop our bounding procedure within the branch-and-bound method. Suppose the value of the best-known solution for the current subproblem is given by β. If we are able to compute a bound that is below β, we can stop our bounding procedure and prune this subproblem from the branch-and-bound tree. Indeed, in our method, stopping the run of the quasi-Newton solver at any time gives the valid bound $F_{I_k}^{\alpha_k}(y_k, z_k)$. Similarly, Biq Mac can stop their feasible interior-point method at any time and get a valid bound; this is because Biq Mac uses the following form of their dual function

$$\theta_I(z) = e^T z \; + \; \min \left\{ e^T y : \operatorname{Diag}(y) \succeq Q + A_I^*(z) \right\}$$

implying that $(MC) \leq \theta_I(z) \leq e^T z + e^T y$, for any feasible vector y.

We can also stop the bounding procedure when we detect that we will not be able to reach the value of β in a reasonable time. Biq Mac estimates the time to reach β using a linear approximation to the time-bound curve [30]. In our method, we fit a hyperbolic function to the time-bound curve, given its hyperbolic nature that can be observed in Fig. 1. If this hyperbola has a horizontal asymptote that is far above the value of β, we terminate the bounding procedure and add the current subproblem, together with its computed bound and fractional solution, to the list of subproblems on which we will branch. We also terminate the bounding procedure if the difference between consecutive bounds, $F_{k-1} - F_k$, is less than one, we are still at least two away from the lower bound β, and α_k is very small (on the order of 10^{-4}).

In all subproblems in the branch-and-bound tree, except the root problem, we wait until the termination of the bounding procedure to compute a feasible solution (upper-bound) using the Goemans-Williamson (GW) random hyperplane algorithm [12] then local swapping, as is done in Biq Mac. However, we do not use the technique used in Biq Mac of repeatedly running the GW algorithm (while progress is still made) on a matrix obtained by shifting the current matrix X in the direction $\tilde{x}\tilde{x}^T$, where $\tilde{x} \in \{-1, 1\}^n$ is the best cut found so far. In the root problem, to get an upper-bound β, we run the GW procedure up to two times during the bounding procedure—this value of β is then used to inform us when we should terminate the bounding procedure in the root problem.

For generating subproblems, Biq Mac considers two branching strategies: "easy first" (R2), and "difficult first" (R3); for details, see [29]. We use two different but similar "easy first" and "difficult first" branching strategies in the $\{0, 1\}$-model: we branch on the variable in the first row/column of X with fractional value that is furthest

from, or closest to, $\frac{1}{2}$, respectively. For the sake of simplicity, we also refer to our "easy first" and "difficult first" branching strategies as (R2) and (R3), respectively. In our numerical results, we compare our method using (R2) branching and our method using (R3) branching to Biq Mac using (R2) branching and Biq Mac using (R3) branching—in total, we compare four methods.

Finally, we note that we have used a best-first exploration strategy in that we always take the subproblem from the current list of subproblems that has the greatest computed bound.

5.2 Description of the numerical tests

Comparison. There exist several efficient methods for solving Max-Cut and binary quadratic programming problems, such as [4] and [26]. However, the numerical tests of [29] show that the results of the other methods are dominated by Biq Mac, so we only compare our solution method to Biq Mac.

Test problems. We use the test problems in the Biq Mac Library [31], which consists of both Max-Cut problems and binary quadratic optimization problems. Some are randomly generated instances, others come from a statistical physics application. We refer to [29, Section 6] for a description of the data set. Since solving the instances in the Biq Mac Library with $n = 500$ is beyond the reach of current solvers (including Biq Mac and our method) we restrict our tests to the 328 instances in the Biq Mac Library with $n < 500$.

Machine. In our tests we used a Dell T-7500 (using a single core) with 4 GB of memory running the Linux operating system. We implemented our algorithm in C / FORTRAN and have used the Intel Math Kernel Library (MKL) for the eigenvalue computations. We have compiled and run both our code and the Biq Mac code (kindly provided by the authors) on the same machine, and have used the same libraries (i.e., MKL) and compilation flags for both codes.

Parameters. We used Biq Mac with its default parameters, except for using both the non-default (R2) and the default (R3) branching strategies; additionally, we changed the time limit to 100 h. We used the following values for Algorithm 1 in our tests:

1. *Reducing* α. We use $\texttt{ScaleAlpha} = 0.5$ and start with $\alpha_1 = 10$; however, if no inequalities are added after the first iteration in the root problem (i.e., $|I_1| = 0$ or $|I_2| = 0$), we start with a smaller α_1. We continue to reduce α_k until the minimum value of $\alpha_k = 10^{-5}$.
2. *Reducing* ε. With (R3) branching, we use $\varepsilon_1 = 0.08$ and $\texttt{ScaleEps} = 0.93$, with a minimum value of $\varepsilon_k = 0.05$. When using (R2) branching, we found that it is better to be less aggressive in decreasing ε, so we take $\varepsilon_1 = 0.2$ and $\texttt{ScaleEps} = 0.95$, with a minimum value of $\varepsilon_k = 0.1$.
3. *Inequalities.* We typically set $\texttt{ViolTol}_k = -5 \times 10^{-2}$. We set $\texttt{MaxIneqAdded}_k = 20k$ (resp. $30k$) and $\texttt{MinIneqViol} = 30$ (resp. 60) for $n < 150$ (resp. for $n \geq 150$). This means that we usually add less inequalities than Biq Mac.

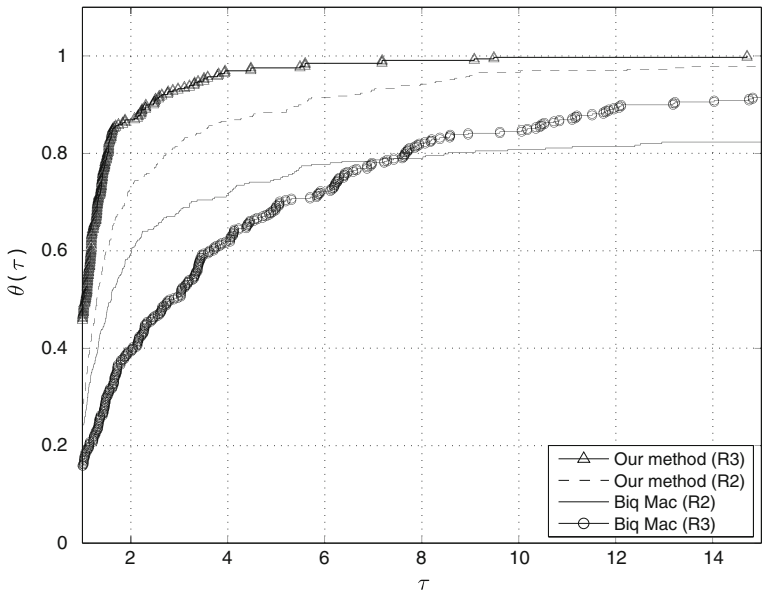

Fig. 2 Performance profile curves (30) for the four solvers on the 328 problems

Performance profiles. We plot the results using performance profiles [11], that we briefly describe here. Let P be the set of problems used to benchmark the solvers. For each problem $p \in P$, we define $t_p^{\min}$ as the minimum time required to solve p over all the solvers. Then, for each solver, we consider the performance profile function θ, which is defined as

$$\theta(\tau) = \frac{1}{|P|} \left| \left\{ p \in P : t_p \leq \tau t_p^{\min} \right\} \right|, \quad \text{for } \tau \geq 1, \tag{30}$$

where t_p is the time required for the solver to solve problem p. The function θ is therefore a cumulative distribution function, and $\theta(\tau)$ represents the probability of the solver to solve a problem from P within a multiple τ of the minimum time required by all solvers considered.

5.3 Computational results

We report aggregated results from the 328 test-problems and almost 1,600 h of computing time: Fig. 2 gives the performance profile, Table 1 counts the number of times a solver attains the minimum solution time, and Table 2 summarizes the computing times. For the full tables of numerical results, see the detailed results available online at the link lipn.univ-paris13.fr/BiqCrunch/repository/papers/KMR2012-results.pdf.

Compared to Biq Mac, our algorithm performs well for a large part of the problems: 241 out of the 328 problems are solved strictly faster by using our solver, which is around 75 % of the test-problems. When considering only the "hard problems" (those

Table 1 The number of times a solver attains the minimum solution time for each set of problems from the Biq Mac Library [31] having $n < 500$

Problem sets	Biq Mac		Our method	
	(R2)	(R3)	(R2)	(R3)
bqp50	**6/10**	**6/10**	0/10	4/10
bqp100	**5/10**	**5/10**	1/10	4/10
bqp250	0/10	0/10	1/10	**9/10**
gkaa	**5/8**	**5/8**	2/8	2/8
gkab	0/10	0/10	4/10	**6/10**
gkac	**5/7**	**5/7**	2/7	0/7
gkad	1/10	1/10	4/10	**5/10**
gkae	0/5	0/5	**3/5**	2/5
be100.d100	0/10	2/10	3/10	**5/10**
be120.d30	2/10	2/10	2/10	**6/10**
be120.d80	2/10	0/10	1/10	**7/10**
be150.d30	1/10	0/10	3/10	**6/10**
be150.d80	**6/10**	0/10	3/10	1/10
be200.d30	0/10	0/10	**9/10**	1/10
be200.d80	1/10	0/10	**8/10**	1/10
be250.d10	0/10	0/10	2/10	**8/10**
g05.n60	3/10	**5/10**	1/10	3/10
g05.n80	3/10	0/10	2/10	**5/10**
g05.n100	**5/10**	0/10	0/10	**5/10**
pm1s80.d090	**7/10**	**7/10**	1/10	2/10
pm1s100.d010	2/10	1/10	1/10	**7/10**
pm1d80.d090	2/10	0/10	**4/10**	**4/10**
pm1d100.d090	1/10	0/10	1/10	**8/10**
w100.d010	2/10	3/10	**5/10**	1/10
w100.d050	3/10	0/10	3/10	**4/10**
w100.d090	**4/10**	0/10	2/10	**4/10**
pw100.d010	1/10	1/10	2/10	**7/10**
pw100.d050	4/10	0/10	1/10	**5/10**
pw100.d090	0/10	0/10	3/10	**7/10**
ising100	0/6	0/6	**3/6**	**3/6**
ising150	0/6	0/6	1/6	**5/6**
ising200	0/6	0/6	2/6	**4/6**
ising250	0/6	0/6	**4/6**	2/6
ising300	0/6	0/6	2/6	**4/6**
t2g10	**3/3**	**3/3**	0/3	0/3
t2g15	0/3	0/3	**3/3**	0/3
t2g20	0/3	0/3	1/3	**2/3**
t3g5	**3/3**	**3/3**	0/3	0/3
t3g6	0/3	1/3	**2/3**	0/3
t3g7	**2/3**	**2/3**	0/3	1/3
Total for each method	79/328	52/328	92/328	**150/328**
Grand total	87/328 (27 %)		**242/328 (74 %)**	

The best results are indicated in bold

Table 2 Minimum / mean / maximum CPU time (s) for Biq Mac and our method to solve problems in the Biq Mac Library [31]

Problem sets	Biq Mac			Our method		
	Min	Mean	Max	Min	Mean	Max
bqp50	**0.11**	0.70	5.03	0.17	**0.22**	**0.26**
bqp100	**0.92**	3.77	22.29	1.40	**2.58**	**11.12**
bqp250	526.18	7,052.28	54,277.92	**47.13**	**1,736.11**	**14,714.12**
gkaa	**0.06**	**0.55**	**1.37**	**0.06**	0.65	2.11
gkab	1.11	1,650.88	10,719.44	**0.10**	**787.00**	**5,104.21**
gkac	0.16	**0.62**	**1.02**	**0.13**	0.85	1.95
gkad	**1.53**	32.98	79.33	1.63	**18.06**	**55.72**
gkae	126.98	6,116.31	25,721.41	**21.77**	**3,897.82**	**16,654.15**
be100.d100	16.70	42.88	**108.69**	**3.79**	**33.51**	114.88
be120.d30	**4.52**	32.64	113.27	5.01	**17.68**	**76.30**
be120.d80	39.12	146.09	**211.55**	**32.60**	**142.54**	268.07
be150.d30	39.21	367.92	844.33	**7.74**	**285.44**	**767.08**
be150.d80	441.62	506.15	**561.98**	**363.55**	**500.04**	608.20
be200.d30	1,079.71	8,223.51	43,850.81	**102.61**	**4,852.01**	**32,612.82**
be200.d80	1,210.52	10,692.92	35,247.12	**596.22**	**6,759.69**	**25,518.98**
be250.d10	328.64	2,437.95	3,754.20	**27.98**	**409.71**	**905.98**
g05.n60	**0.26**	**5.74**	**13.27**	0.67	7.42	24.84
g05.n80	6.39	**63.09**	**274.50**	**2.80**	63.10	296.10
g05.n100	**93.45**	803.88	4,197.29	96.05	**721.25**	**3,382.27**
pm1s80.d090	**0.58**	4.76	23.96	0.96	**3.92**	**13.40**
pm1s100.d010	**1.15**	49.87	130.28	1.85	**34.70**	**88.48**
pm1d80.d090	24.56	71.90	212.47	**13.74**	**56.31**	**138.19**
pm1d100.d090	106.24	945.40	2868.06	**56.94**	**620.76**	**1,800.63**
w100.d010	**1.37**	**23.15**	**131.45**	1.71	23.47	152.09
w100.d050	109.99	563.27	1152.49	**81.02**	**475.32**	**1,061.46**
w100.d090	38.67	**836.25**	**2,945.89**	**26.50**	997.06	4675.78
pw100.d010	**1.64**	65.34	228.70	2.06	**34.88**	**113.86**
pw100.d050	122.67	**715.95**	**1,798.61**	**81.34**	**732.48**	2,506.69
pw100.d090	209.02	606.82	1,297.98	**201.61**	**509.48**	**1,061.18**
ising100	7.74	12.76	19.86	**2.62**	**3.04**	**3.87**
ising150	42.55	60.31	85.74	**7.74**	**10.70**	**12.89**
ising200	150.39	233.99	403.19	**19.72**	**27.26**	**37.59**
ising250	165.79	981.56	2,161.49	**47.88**	**138.27**	**406.84**
ising300	991.28	4,268.83	11,858.56	**97.43**	**455.77**	**1,628.66**
t2g10	**1.63**	**2.22**	**3.04**	3.50	4.14	5.30
t2g15	56.55	66.36	73.34	**38.87**	**42.45**	**46.03**
t2g20	1,014.38	4,676.45	11,132.18	**351.37**	**1,159.24**	**2,748.55**
t3g5	**3.22**	**4.42**	**5.49**	7.18	7.39	7.71
t3g6	52.34	**159.36**	**367.12**	**41.85**	218.22	566.66
t3g7	**81.37**	5,931.61	17,601.56	123.46	**3,838.06**	**11,234.02**

The best result are indicated in bold

that Biq Mac does not solve at the root node), this percentage increases to 85 % (226 out of 269).

Table 2 shows that, on some problems (e.g., g05 or w100) our method is roughly equivalent to Biq Mac, and on some others it is two times quicker (e.g., gk) or even 6 times quicker (e.g., be250.d10). In particular, we obtain substantial improvements, in both time and number of nodes, for large-sized problems, as can be seen in the detailed results available online at the web address mentioned above. Additionally, we note that both our methods are able to solve all test problems within the 100 h time limit (Biq Mac (R3) fails to solve 6 instances within this time limit).

The performance profile in Fig. 2 shows that both of our methods dominate both of the Biq Mac methods in terms of speed, and also robustness, since the curves for our methods are constantly above the ones for Biq Mac.

6 Conclusions

In this paper, we presented an improved semidefinite bounding procedure to efficiently solve Max-Cut and binary quadratic programming problems to optimality. We precisely analyzed its theoretical convergence properties, and we conducted a complete numerical comparison with the leading Biq Mac method. Our algorithm is shown to be often faster and more robust than Biq Mac; in particular, our solver was able to solve around 75 % of the problems of the Biq Mac Library (with $n < 500$) in less time than Biq Mac.

A key point is that the two main ingredients of our method are eigenvalue decomposition to evaluate the function and its gradient, and a quasi-Newton method to minimize this smooth function; on the other hand, the two main ingredients of the Biq Mac method are an interior-point method to evaluate a function and its subgradient, and a bundle method to minimize this nonsmooth function.

There is still room for improvement in our current implementation. In particular, we would like to investigate how we can make the eigenvalue decomposition more efficient by somehow using prior information. We could also try decoupling the optimization of the y and z dual variables, as is done in Biq Mac, to possibly make the bound computation more efficient.

By focusing on Max-Cut and binary quadratic programming problems, we have shown unambiguously the interest of our semidefinite-based method to solve these classes of problems to optimality. Because our approach is built on the general bounds of [21], it can also be applied to any binary quadratic problems with additional linear or quadratic, equality or inequality, constraints. In our future work, such as [17], we will consider this generalization, which is nontrivial due the work required to make a generic code and to the complication of having semidefinite relaxations that may not be strictly feasible. Due to its inherent flexibility, we believe that our method has a strong potential to handle other classes of problems, even those for which semidefinite-based methods are not yet competitive.

Acknowledgments We are grateful to Angelika Wiegele for the discussions we have had and for providing us with a copy of the Biq Mac solver for our numerical comparison.

References

1. Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.: LAPACK Users' Guide, 3rd edn. Society for Industrial and Applied Mathematics, Philadelphia (1999)
2. Anjos, M.F., Lasserre, J.B. (eds.): Handbook on Semidefinite, Conic and Polynomial Optimization, International Series in Operations Research & Management Science, vol.166. Springer, USA (2012)
3. Anjos, M.F., Lasserre, J.B.: Introduction to semidefinite, conic and polynomial optimization. In: Anjos, M.F., Lasserre, J.B (eds.) Handbook on Semidefinite, Conic and Polynomial Optimization, International Series in Operations Research & Management Science, vol.166, pp. 1–22. Springer, USA (2012)
4. Billionnet, A., Elloumi, S.: Using a mixed integer quadratic programming solver for the unconstrained quadratic 0–1 problem. Math. Program. **109**, 55–68 (2007)
5. Bonnans, J., Gilbert, J., Lemaréchal, C., Sagastizábal, C.: Numerical Optimization. Springer, Berlin (2003)
6. Borwein, J.M., Lewis, A.S.: Convex Analysis and Nonlinear Optimization: Theory and Examples. Springer, Berlin (2000)
7. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2008)
8. Byrd, R.H., Lu, P., Nocedal, J., Zhu, C.: A limited memory algorithm for bound constrained optimization. SIAM J. Sci. Comput. **16**(5), 1190–1208 (1995)
9. Cadoux, F., Lemaréchal, C.: Reflections on generating (disjunctive) cuts. EURO J. Comput. Optim. (Submitted, 2012)
10. Cun, B.L., Roucairol, C., The PNN Team: BOB: A Unified Platform for Implementing Branch-and-Bound Like Algorithms. Technical Report 95/16, University of Versailles Saint-Quentin-en-Yvelines (1995).
11. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)
12. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM **42**(6), 1115–1145 (1995)
13. Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd edn. Johns Hopkins University Press, Baltimore (1996)
14. Helmberg, C., Rendl, F.: Solving quadratic (0,1)-problems by semidefinite programs and cutting planes. Math. Program. **82**, 291–315 (1998)
15. Hiriart-Urruty, J.B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms. Springer, Heidelberg (Two volumes) (1993)
16. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of Computer Computations (Proceedings of Symposium, IBM Thomas J. Watson Res. Center, Yorktown Heights, NY, 1972), pp. 85–103. Plenum, New York (1972)
17. Krislock, N., Malick, J., Roupin, F.: Improved semidefinite branch-and-bound algorithm for k-cluster (Submitted, 2012). Available online as preprint hal-00717212
18. Laurent, M., Poljak, S.: Gap inequalities for the cut polytope. Eur. J. Comb. **17**(2–3), 233–254 (1996)
19. Lemaréchal, C., Oustry, F.: Semidefinite Relaxations and Lagrangian Duality with Application to Combinatorial Optimization. Rapport de Recherche 3710, INRIA (1999)
20. Malick, J.: Spherical constraint in Boolean quadratic programming. J. Glob. Optim. **39**(4) (2007)
21. Malick, J., Roupin, F.: On the bridge between combinatorial optimization and nonlinear optimization: a family of semidefinite bounds leading to Newton-like methods. Math. Program. B (to appear, 2012). Available online as preprint hal-00662367
22. Malick, J., Roupin, F.: Solving k-cluster problems to optimality using adjustable semidefinite programming bounds. Math. Program. B Special Issue Mixed Integer Nonlinear Program (to appear, 2012)
23. Morales, J.L., Nocedal, J.: Remark on "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization". ACM Trans. Math. Softw. **38**(1), 1–4 (2011)
24. Nocedal, J., Wright, S.J.: Numerical Optimization, 2nd edn. Springer, Berlin (2006)
25. Palagi, L., Piccialli, V., Rendl, F., Rinaldi, G., Wiegele, A.: Computational approaches to Max-Cut. In: Anjos, M.F., Lasserre, J.B. (eds.) Handbook on Semidefinite, Conic and Polynomial Optimization, International Series in Operations Research & Management Science, vol.166, pp. 821–847. Springer, USA (2012)

26. Pardalos, P., Rodgers, G.: Computational aspects of a branch and bound algorithm for quadratic zero-one programming. Computing **45**, 134–144 (1990)
27. Pardalos, P.M., Vavasis, S.A.: Quadratic programming with one negative eigenvalue is NP-hard. J. Glob. Optim. **1**, 15–22 (1991)
28. Poljak, S., Rendl, F., Wolkowicz, H.: A recipe for semidefinite relaxation for (0,1)-quadratic programming. J. Glob. Optim. **7**, 51–73 (1995)
29. Rendl, F., Rinaldi, G., Wiegele, A.: Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. Math. Program. **121**, 307–335 (2010)
30. Wiegele, A.: Nonlinear Optimization Techniques Applied to Combinatorial Optimization Problems. Ph.D. thesis, Alpen-Adria-Universität Klagenfurt (2006)
31. Wiegele, A.: Biq Mac Library–A collection of Max-Cut and quadratic 0–1 programming instances of medium size. Technical report, Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria (2007)
32. Zhu, C., Byrd, R.H., Lu, P., Nocedal, J.: Algorithm 778: L-BFGS-B: fortran subroutines for large-scale bound-constrained optimization. ACM Trans. Math. Softw. **23**(4), 550–560 (1997)