

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

Nathan Krislock, Jérôme Malick, Frédéric Roupin

▶ **To cite this version:**

Nathan Krislock, Jérôme Malick, Frédéric Roupin. Improved semidefinite bounding procedure for solving Max-Cut problems to optimality. Mathematical Programming, 2014, 143 (1-2), pp.61-86. 10.1007/s10107-012-0594-z . hal-00665968

HAL Id: hal-00665968
https://hal.science/hal-00665968

Submitted on 3 Feb 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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

Nathan Krislock * Jérôme Malick † Frédéric Roupin ‡

February 3, 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 can be controlled by 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.

Keywords combinatorial optimization, semidefinite programming, quasi-Newton algorithm, triangle inequalities

*INRIA Grenoble Rhône-Alpes, nathan.krislock@inria.fr

†CNRS, Lab. J. Kunztmann, Grenoble, jerome.malick@inria.fr

‡LIPN - CNRS UMR7030 - Université Paris-Nord, roupin@lipn.univ-paris13.fr

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 [25] 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 [15] and [11], and in the survey chapter [23] 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

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

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 [13], the QCR method [4], and most recently the state-of-the-art Biq Mac method [27]. 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 [18, 19] and specifically for the k-cluster problem [20]). More precisely, we apply here the bounds to the Max-Cut problem that have been introduced in [19] 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 Section 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 Section 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 Section 5). We compare the overall algorithm with the best existing exact resolution method, the Biq Mac solver [27] on the 328 instances in the Biq Mac Library [29] 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., [13, 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., [17, 26]), and provide a sketch of the Biq Mac method [27], 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

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

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):

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

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 [11]). 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 to solve the Max-Cut problem to optimality (see, e.g., [13, 27, 28]).

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 [13, Section 4] and [27, 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 $\mathcal{I}$ be a subset of the triangle inequalities, and $\mathcal{A}_{\mathcal{I}}\colon \mathcal{S}^n \to \mathbb{R}^{|\mathcal{I}|}$ be the corresponding linear operator describing the inequalities in this subset. We define the ($\text{SDP}_{\mathcal{I}}$) problem as

$$\begin{array}{lll} & \text{maximize} & \langle Q, X\rangle \\ (\text{SDP}_{\mathcal{I}}) & \text{subject to} & \text{diag}(X) = e,\ X \succeq 0, \\ & & \mathcal{A}_{\mathcal{I}}(X) + e \geq 0. \end{array} \tag{6}$$

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

$$(\text{MC}) \leq (\text{SDP}_{\mathcal{I}}) \leq (\text{SDP}). \tag{7}$$

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

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

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

There is a very large number of triangle inequalities: adding all of them would make the ($\text{SDP}_{\mathcal{I}}$) problem intractable even for moderate values of n. However, to get the best possible ($\text{SDP}_{\mathcal{I}}$) bound, we only need to add the inequalities that are active at the optimal solution of the problem having all triangle inequalities, which we call ($\text{SDP}_{\mathcal{I}_{\text{all}}}$). Such a set of active inequalities is obviously unknown, but we will see in Section 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 [13], general gap inequalities [16], or even a sophisticated choice based on the objective function. 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 $\mathcal{I}$.

However, in the computational experiments, as is done in [27], we only use triangle inequalities as it greatly simplifies the selection of inequalities and leads to satisfactory results.

Best existing method. Biq Mac [27, 28] 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 ($\text{SDP}_{\mathcal{I}}$) in equation (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

$$\theta_{\mathcal{I}}(z) = e^T z \quad + \quad \begin{array}{ll} \text{maximize} & \langle Q + \mathcal{A}_{\mathcal{I}}^*(z), X \rangle \\ \text{subject to} & \text{diag}(X) = e,\ X \succeq 0, \end{array} \tag{9}$$

where $z \in \mathbb{R}_+^{|\mathcal{I}|}$ is the dual variable. The value $\theta_{\mathcal{I}}(z)$ provides an upper bound on the value of the maximum cut for each $z \in \mathbb{R}_+^{|\mathcal{I}|}$; the goal is to minimize $\theta_{\mathcal{I}}(z)$ 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 $\theta_{\mathcal{I}}(z)$ and a subgradient $g \in \partial\theta_{\mathcal{I}}(z)$ are evaluated by solving the semidefinite programming problem in equation (9) by a customized primal-dual interior-point method;

2. the nonsmooth convex function $\theta_{\mathcal{I}}$ is minimized by a bundle method [14].

The extensive numerical experiments of [27, 28] show that Biq Mac dominates all other approaches (see the six tested approaches in [27, Section 7]).

3 New family of 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 Section 3.1). On the other hand, Section 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 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 $A = U\,\text{Diag}(\lambda)U^T$,

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 new 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 $(SDP_{\mathcal{I}})$: 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

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

Proposition 1. *The following holds: for all $\mathcal{I}$ and for all $\alpha \ge 0$,*

$$(\mathrm{MC}) \le (\mathrm{SDP}_{\mathcal{I}}) \le (\mathrm{SDP}_{\mathcal{I}}^{\alpha}). \tag{11}$$

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 $(\mathrm{SDP}_{\mathcal{I}})$ *or* $(\mathrm{SDP}^{\alpha}_{\mathcal{I}})$. *Moreover, we have*

$$\alpha' \leq \alpha \quad \Rightarrow \quad (\mathrm{SDP}^{\alpha'}_{\mathcal{I}}) \leq (\mathrm{SDP}^{\alpha}_{\mathcal{I}}), \tag{12}$$

and

$$\mathcal{I}' \subseteq \mathcal{I} \quad \Rightarrow \quad (\mathrm{SDP}^{\alpha}_{\mathcal{I}'}) \geq (\mathrm{SDP}^{\alpha}_{\mathcal{I}}). \tag{13}$$

Proof. Inequality (13) clearly holds since the feasible set of $(\mathrm{SDP}^{\alpha}_{\mathcal{I}})$ is included in that of $(\mathrm{SDP}^{\alpha}_{\mathcal{I}'})$. 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

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

which gives $(\mathrm{SDP}^{\alpha'}_{\mathcal{I}}) \leq (\mathrm{SDP}^{\alpha}_{\mathcal{I}})$. 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 $(\mathrm{MC}) = (\mathrm{SDP}_{\mathcal{I}})$ if and only if there is a rank-one optimal solution of $(\mathrm{SDP}_{\mathcal{I}})$. Finally, we use a corollary of [18, 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 $(\mathrm{SDP}^{\alpha}_{\mathcal{I}})$ is strictly weaker than $(\mathrm{SDP}_{\mathcal{I}})$ (except in the very special situation that $(\mathrm{SDP}_{\mathcal{I}})$ 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 $\mathcal{I}$, and let us dualize the affine constraints of $(\mathrm{SDP}^{\alpha}_{\mathcal{I}})$. We define the Lagrangian function with respect to the primal variable $X \succeq 0$ and the dual variables $y \in \mathbb{R}^n$ and $z \in \mathbb{R}^{|\mathcal{I}|}_+$ as

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

The associated dual function is then defined by

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

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

$$(\mathrm{MC}) \leq (\mathrm{SDP}_{\mathcal{I}}) \leq (\mathrm{SDP}^{\alpha}_{\mathcal{I}}) \leq F^{\alpha}_{\mathcal{I}}(y, z). \tag{15}$$

These $F_{\mathcal{I}}^{\alpha}(y,z)$ for given $(\mathcal{I},\alpha,y,z)$ 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 $F_{\mathcal{I}}^{\alpha}(y,z)$

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

$$X_{\mathcal{I}}(y,z) := \left[Q - \mathrm{Diag}(y) + \mathcal{A}_{\mathcal{I}}^{*}(z)\right]_{+}. \tag{16}$$

Proposition 2. *Let $\alpha > 0$ and let $\mathcal{I}$ be a set of inequalities. Then, for $F_{\mathcal{I}}^{\alpha}$ and $X_{\mathcal{I}}$ defined in equations* (14) *and* (16)*, respectively, we have*

$$F_{\mathcal{I}}^{\alpha}(y,z) = \frac{1}{2\alpha}\left\|X_{\mathcal{I}}(y,z)\right\|_F^2 + e^T y + e^T z + \frac{\alpha}{2}n^2, \qquad \textit{for all } y,\ z. \tag{17}$$

Proof. Let $M = Q - \mathrm{Diag}(y) + \mathcal{A}_{\mathcal{I}}^{*}(z)$ and $f(X) = \langle M, X\rangle - \frac{\alpha}{2}\|X\|_F^2$. Since $\alpha > 0$, we have that f is a concave function. Moreover, $\max_{X\succeq 0} f(X)$ is a strictly feasible convex optimization problem, so strong duality holds (see, e.g., [7]); that is,

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

Then, from the observation that

$$\min_{S\succeq 0}\max_{X} f(X) + \langle S, X\rangle = \min_{S\succeq 0}\frac{1}{2\alpha}\|M+S\|_F^2 = \frac{1}{2\alpha}\left\|[M]_{+}\right\|_F^2,$$

the remainder of the proof follows easily. □

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 [19] for general 0-1 quadratic optimization problems (see $\Theta(\lambda,\mu,\alpha)$ in Theorem 2 of [19]). 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 spherical constraint (see [18]).

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

$$(\mathrm{DSDP}_{\mathcal{I}}^{\alpha}) \qquad \begin{array}{ll} \text{minimize} & \frac{1}{2\alpha}\left\|X_{\mathcal{I}}(y,z)\right\|_F^2 + e^T y + e^T z + \frac{\alpha}{2}n^2 \\ \text{subject to} & y \text{ free},\ z \geq 0, \end{array} \tag{18}$$

which is the dual problem of $(\mathrm{SDP}_{\mathcal{I}}^{\alpha})$.

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

$$F_{\mathcal{I}}^{0}(y,z) = \begin{cases} e^T y + e^T z, & \text{if } Q - \operatorname{Diag}(y) + \mathcal{A}_{\mathcal{I}}^{*}(z) \preceq 0, \\ +\infty, & \text{otherwise,} \end{cases}$$

which leads us back to the dual problem (8). We see that the condition $Q - \operatorname{Diag}(y) + \mathcal{A}_{\mathcal{I}}^{*}(z) \preceq 0$ is exactly $X_{\mathcal{I}}(y,z) = 0$, which opens another view on the approach we consider. Indeed, from equation (17) the function $F_{\mathcal{I}}^{\alpha}(y,z)$ can be viewed as a penalization of the standard dual function $F_{\mathcal{I}}^{0}(y,z)$. The parameter α is interpreted as a penalization parameter controlling the loss in the tightness of the bound.

In view of solving problem (18), the next proposition is crucial.

Proposition 3. *Let $\alpha > 0$. Then the dual function $F_{\mathcal{I}}^{\alpha}$ is convex and differentiable, with partial gradients given by*

$$\nabla_y F_{\mathcal{I}}^{\alpha}(y,z) = e - \frac{1}{\alpha} \operatorname{diag}\left(X_{\mathcal{I}}(y,z)\right),$$
$$\nabla_z F_{\mathcal{I}}^{\alpha}(y,z) = e + \frac{1}{\alpha} \mathcal{A}_{\mathcal{I}}\left(X_{\mathcal{I}}(y,z)\right).$$

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 $g\colon \mathcal{S}^n \to \mathbb{R}$ defined by $g(X) = f(\lambda(X))$ is convex and differentiable with gradient $\nabla g(X) = U \operatorname{Diag}(\nabla f(\lambda(X)))U^T$, where U is any $n \times n$ orthogonal matrix satisfying $X = U \operatorname{Diag}(\lambda(X))U^T$; thus $\nabla g(X) = X_+$. Now note that

$$F_{\mathcal{I}}^{\alpha}(y,z) = \frac{1}{\alpha} g(Q - \operatorname{Diag}(y) + \mathcal{A}_{\mathcal{I}}^{*}(z)) + e^T y + e^T z + \frac{\alpha}{2} n^2,$$

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

As expected, this proposition is in the same vein as [19, 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 $F_{\mathcal{I}}^{\alpha}$ when reducing the value of α with fixed variables y and z.

Proposition 4. *Let $0 \leq \varepsilon \leq 1$ and $0 < \alpha' < \alpha$. Consider (y, z) such that $\|\nabla_y F_{\mathcal{I}}^{\alpha}(y, z)\|_{\infty} \leq \varepsilon$. Then*

$$F_{\mathcal{I}}^{\alpha'}(y, z) \;\leq\; F_{\mathcal{I}}^{\alpha}(y, z) + \frac{n^2}{2}\frac{(\alpha - \alpha')}{\alpha'}\left(\alpha(1+\varepsilon)^2 - \alpha'\right)$$

and

$$F_{\mathcal{I}}^{\alpha'}(y, z) \;\geq\; F_{\mathcal{I}}^{\alpha}(y, z) + \frac{n^2}{2}\frac{(\alpha - \alpha')}{\alpha'}\left(\frac{\alpha}{n}(1-\varepsilon)^2 - \alpha'\right).$$

Proof. We have

$$F_{\mathcal{I}}^{\alpha'}(y, z) - F_{\mathcal{I}}^{\alpha}(y, z) = \frac{1}{2}\left(\frac{1}{\alpha'} - \frac{1}{\alpha}\right)\|X_{\mathcal{I}}(y, z)\|_F^2 - \frac{n^2}{2}(\alpha - \alpha').$$

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

$$\alpha^2(1-\varepsilon)^2 n \leq \|X_{\mathcal{I}}(y, z)\|_F^2 \leq \alpha^2(1+\varepsilon)^2 n^2.$$

We deduce the first desired bound, as

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

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

This result tells us that $F_{\mathcal{I}}^{\alpha'}(y, z)$ may increase over $F_{\mathcal{I}}^{\alpha}(y, z)$, 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 Section 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 $\mathcal{I}$ and a tightness level $\alpha > 0$, the computation of the value of the bounding function $F_{\mathcal{I}}^{\alpha}$ and its gradient essentially corresponds to the computation of $X_{\mathcal{I}}(y, z)$, which, in turn, reduces to computing the positive eigenvalues and corresponding eigenvectors of the symmetric matrix $Q - \mathrm{Diag}(y) + \mathcal{A}_{\mathcal{I}}^*(z)$ (see equation (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 [12] for more information.

In view of the differentiability result (Proposition 3), we can solve problem (18) 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, 22]. In our numerical experiments, we use the FORTRAN code L-BFGS-B [8, 30] which has been recently upgraded to version 3.0 [21]. 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 $F_{\mathcal{I}}^{\alpha}(y,z)$ and gradient $\nabla F_{\mathcal{I}}^{\alpha}(y,z)$ are evaluated by computing a single partial eigenvalue decomposition;

2. the smooth convex function $F_{\mathcal{I}}^{\alpha}$ 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 $F_{\mathcal{I}}^{\alpha}(y,z)$ 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 $F_{\mathcal{I}}^{\alpha}(y,z)$: 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 $\mathcal{I}$; that is,

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

where $X = \frac{1}{\alpha} X_{\mathcal{I}}(y, z)$. 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 [28] 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 (18) 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:

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

Note that the expressions of the gradients of $F_{\mathcal{I}}^{\alpha}$ (Proposition 3) gives

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

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 α and ε). We give details in Section 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.

Algorithm 1 Improved semidefinite bounding algorithm

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

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

$\mathcal{I}_0 \leftarrow \emptyset$ {initial set of triangle inequalities $\mathcal{I}$}

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

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

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

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

Remove triangle inequalities that are not active:

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

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 $i \notin \mathcal{I}_{k-1}$ such that $\mathcal{A}_i(X^{\text{test}}_k) + 1 \le \texttt{ViolTol}_k$, where $\texttt{ViolTol}_k \le 0$, ordered such that $\mathcal{A}_{i_1}(X^{\text{test}}_k) \le \cdots \le \mathcal{A}_{i_\ell}(X^{\text{test}}_k)$. Let

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

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

Initialize multipliers for added inequalities to zero:

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

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

4.2 Convergence of the bounding procedure

We now analyze Algorithm 1, starting with the case when the set of triangle inequalities $\mathcal{I}$ is fixed.

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

$$\max\left\{\left\|\nabla_y F_{\mathcal{I}}^{\alpha_k}(y_k, z_k)\right\|_\infty,\ \left\|[\nabla_z F_{\mathcal{I}}^{\alpha_k}(y_k, z_k)]_-\right\|_\infty\right\} < \varepsilon_k, \tag{19}$$

and $X_k := X_{\mathcal{I}}(y_k, z_k)/\alpha_k$. 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 $(\mathrm{SDP}_{\mathcal{I}})$, and

$$\lim_{k\to+\infty} F_{\mathcal{I}}^{\alpha_k}(y_k, z_k) = (\mathrm{SDP}_{\mathcal{I}}). \tag{20}$$

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):

$$Q - \mathrm{Diag}(y) + \mathcal{A}_{\mathcal{I}}^*(z) = S \tag{21}$$

$$\langle X, S\rangle = 0, \quad X \succeq 0, \quad S \preceq 0 \tag{22}$$

$$\mathrm{diag}(X) = e, \quad \mathcal{A}_{\mathcal{I}}(X) + e \geq 0 \tag{23}$$

$$z^T(\mathcal{A}_{\mathcal{I}}(X) + e) = 0, \quad z \geq 0 \tag{24}$$

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,

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

where $S_k := [Q - \mathrm{Diag}(y_k) + \mathcal{A}_{\mathcal{I}}^*(z_k)]_-$. The sequence $(S_k)_k$ converges to

$$\bar{S} := [Q - \mathrm{Diag}(\bar{y}) + \mathcal{A}_{\mathcal{I}}^*(\bar{z})]_-\,,$$

by continuity of the operator $[\cdot]_-$. Passing to the limit shows that $(\bar{X}, \bar{S}, \bar{y}, \bar{z})$ satisfies (21) 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 (22).

By Proposition 3, condition (19) can be written as

$$\max\left\{\|\mathrm{diag}(X_k) - e\|_\infty\,,\ \|[\mathcal{A}_\mathcal{I}(X_k) + e]_-\|_\infty\right\} < \varepsilon_k.$$

Passing to the limit we have that $\bar{X}$ satisfies (23). By assumption, we have that $z_k \geq 0$ for all k, so we also have that $\bar{z} \geq 0$. Finally, we write

$$|z_k^T(\mathcal{A}_\mathcal{I}(X_k) + e)| \leq \|z_k\|_1 \, \|\mathcal{A}_\mathcal{I}(X_k) + e\|_\infty < \|z_k\|_1 \, \varepsilon_k.$$

Since ε_k vanishes, at the limit, we have that $\bar{z}$ and $\bar{X}$ satisfy (24). Altogether, this shows that $(\bar{X}, \bar{y}, \bar{z})$ is indeed a primal-dual solution.

Finally (17) yields

$$F_\mathcal{I}^{\alpha_k}(y_k, z_k) = e^T y_k + e^T z_k + \alpha_k(\|X_k\|^2 + n^2)/2.$$

Therefore we have $\lim_{k\to+\infty} F_\mathcal{I}^{\alpha_k}(y_k, z_k) = e^T\bar{y} + e^T\bar{z}$. Since there is no duality gap for the primal-dual problems (6) and (8), we conclude that (20) 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, ($\mathrm{SDP}_\mathcal{I}$), 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 $(\alpha_k, \varepsilon_k, X_k, y_k, \hat{z}_k, z_k, \mathcal{I}_k, F_k)_k$ be generated by Algorithm 1. Then the following holds:*

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

$$X_k = \frac{1}{\alpha_k} X_{\mathcal{I}_k}(y_k, z_k) \qquad \text{and} \qquad F_k = F_{\mathcal{I}_k}^{\alpha_k}(y_k, z_k).$$

(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 $\mathcal{I}_k$ and z_k in Algorithm 1, we have $\mathcal{A}^*_{\mathcal{I}_k}(z_k) = \mathcal{A}^*_{\mathcal{I}_{k-1}}(\hat{z}_k)$ 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

$$\|\nabla_y F^{\alpha_k}_{\mathcal{I}_{k-1}}(y_k, \hat{z}_k)\|_\infty < \varepsilon_k,$$

by Proposition 4 we have

$$F^{\alpha_{k+1}}_{\mathcal{I}_{k-1}}(y_k, \hat{z}_k) \leq F^{\alpha_k}_{\mathcal{I}_{k-1}}(y_k, \hat{z}_k) + \frac{n^2}{2} \frac{(\alpha_k - \alpha_{k+1})}{\alpha_{k+1}} \left(\alpha_k (1+\varepsilon_k)^2 - \alpha_{k+1}\right). \quad (25)$$

From the previous item, we have

$$F_{k+1} = F^{\alpha_{k+1}}_{\mathcal{I}_k}(y_{k+1}, \hat{z}_{k+1}) \leq F^{\alpha_{k+1}}_{\mathcal{I}_k}(y_k, z_k) = F^{\alpha_{k+1}}_{\mathcal{I}_{k-1}}(y_k, \hat{z}_k),$$

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 (25) 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

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

we have that

$$\begin{cases} F_{k+1} \leq F_k + \delta\alpha_k, & \text{if } \alpha_{k+1} = \texttt{ScaleAlpha} \cdot \alpha_k, \\ F_{k+1} \leq F_k, & \text{if } \alpha_{k+1} = \alpha_k, \end{cases} \quad (26)$$

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 (26), 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 that

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

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

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

$$\lim_{k\to+\infty} F_{\mathcal{I}_k}^{\alpha_k}(y_k, z_k) = (\mathrm{SDP}_{\bar{\mathcal{I}}}). \tag{27}$$

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

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

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

We now have a key observation about our bounding procedure Algorithm 1. Although, by Proposition 1, the bounds $F_{\mathcal{I}}^{\alpha}(y, z)$, for $\alpha > 0$, are weaker than the usual SDP bound $(\mathrm{SDP}_{\mathcal{I}})$, we have, by Theorem 1, that we can get as close as we like using Algorithm 1.

At the limit, the quality of the bounds is only governed by the selection of promising inequalities in $\mathcal{I}_k$. Looking at a special case, we give some intuition to the fact that the selection of Algorithm 1 approximates an "optimal" set of inequalities $\mathcal{I}$: enforcing the inequalities in $\mathcal{I}$ would in fact enforce all the inequalities.

Corollary 1. *Let the assumptions of Theorem 1 hold. If the sequence $(X_k, y_k, z_k, \mathcal{I}_k)_k$ converges to $(\bar{X}, \bar{y}, \bar{z}, \bar{\mathcal{I}})$ and if $\texttt{ViolTol}_k \to 0$, then $(\mathrm{SDP}_{\bar{\mathcal{I}}})$ is equal to the semidefinite bound $(\mathrm{SDP}_{\mathcal{I}_{\mathrm{all}}})$ with all the inequalities, and $(\bar{X}, \bar{y}, \tilde{z})$ is a primal-dual solution of $(\mathrm{SDP}_{\mathcal{I}_{\mathrm{all}}})$, 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 $\bar{\mathcal{I}}$.*

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

$$\mathcal{A}_i(X_k^{\text{test}}) + 1 \geq \texttt{ViolTol}_k, \quad \text{for all } i \notin \bar{\mathcal{I}}.$$

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

$$\mathcal{A}_i(\bar{X}) + 1 \geq 0, \quad \text{for any inequality } i, \tag{28}$$

and that $\bar{X}$ is in fact feasible for $(\text{SDP}_{\mathcal{I}_{\text{all}}})$. We observe now $(\bar{X}, \bar{y}, \tilde{z})$ satisfies the optimality conditions of $(\text{SDP}_{\mathcal{I}_{\text{all}}})$ (i.e., conditions (21-24) from the proof of Lemma 2). Indeed (28) implies that (23) is satisfied for $X = \bar{X}$ and $\mathcal{I} = \mathcal{I}_{\text{all}}$, and so are (21) and (24) by construction of $\tilde{z}$. Thus we have $(\text{SDP}_{\mathcal{I}}) = (\text{SDP}_{\mathcal{I}_{\text{all}}})$ 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 [27]. We take one problem of the Biq Mac Library [29]—specifically Beasley bqp250.6, the problem highlighted in [27, Figure 1]—and we plot the convergence curve for the two bounding procedures in Figure 1. The code for each solver was compiled on the same machine using the same numerical libraries.

This plot 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.

In Figure 1, one can see 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.

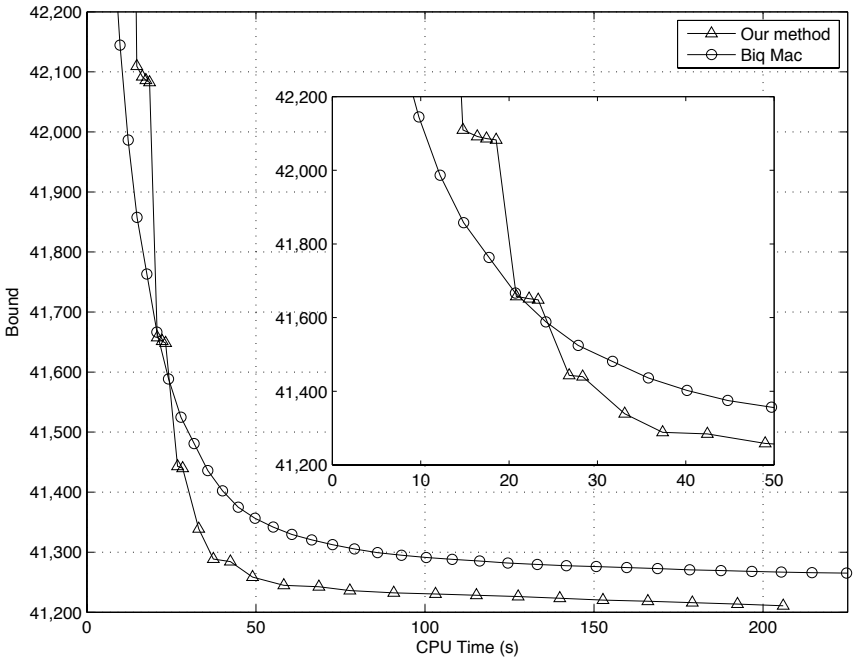


Figure 1: Comparison of the bounding procedure of Biq Mac [27] and our bounding procedure Algorithm 1 on the Beasley bqp250.6 problem. The behaviour shown here is typical.

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 [27].

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 [9], 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 Section 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 $F_{\mathcal{I}_k}^{\alpha_k}(y_k, z_k)$. 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

$$\theta_{\mathcal{I}}(z) = e^T z \quad + \quad \begin{array}{ll} \text{minimize} & e^T y \\ \text{subject to} & \text{Diag}(y) \succeq Q + \mathcal{A}_{\mathcal{I}}^*(z), \end{array}$$

implying that $(MC) \leq \theta_{\mathcal{I}}(z) \leq e^T z + e^T y$, 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 [28]. In our method, we fit a hyperbolic function to the time-bound curve, given its hyperbolic nature that can be observed in Figure 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 [11] 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 [27]. 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 [24] and [4]. However, the numerical tests of [27] 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 [29], 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 [27, 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 ran 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 hours. We used the following values for Algorithm 1 in our tests:

1. *Reducing α.* We use `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., $|\mathcal{I}_1| = 0$ or $|\mathcal{I}_2| = 0$), 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 `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 `ScaleEps` $= 0.95$, with a minimum value of $\varepsilon_k = 0.1$.

3. *Inequalities.* We have used $\texttt{MaxIneqAdded}_k = 20k$ (resp. $30k$) and `MinIneqViol` $= 30$ (resp. 60) for $n < 150$ (resp. for $n \geq 150$). This means that we usually add less inequalities than Biq Mac.

Performance profiles. We plot the results using performance profiles [10], that we briefly describe here. Let $\mathcal{P}$ be the set of problems used to benchmark the solvers. For each problem $p \in \mathcal{P}$, 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

$$\theta(\tau) = \frac{1}{|\mathcal{P}|} \left| \left\{ p \in \mathcal{P} : t_p \leq \tau t_p^{\min} \right\} \right|, \quad \text{for } \tau \geq 1, \tag{29}$$

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 $\mathcal{P}$ 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 1600 hours of computing time: Figure 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 Appendix A.

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

Table 1: The number of times a solver attains the minimum solution time for each set of problems from the Biq Mac Library [29] having $n < 500$.

Problem sets	Biq Mac		Our method	
	(R2)	(R3)	(R2)	(R3)
bqp50 (Table 3)	**6/10**	**6/10**	0/10	4/10
bqp100 (Table 4)	**5/10**	**5/10**	1/10	4/10
bqp250 (Table 5)	0/10	0/10	1/10	**9/10**
gkaa (Table 6)	**5/8**	**5/8**	2/8	2/8
gkab (Table 7)	0/10	0/10	4/10	**6/10**
gkac (Table 8)	**5/7**	**5/7**	2/7	0/7
gkad (Table 9)	1/10	1/10	4/10	**5/10**
gkae (Table 10)	0/5	0/5	**3/5**	2/5
be100.d100 (Table 11)	0/10	2/10	3/10	**5/10**
be120.d30 (Table 12)	2/10	2/10	2/10	**6/10**
be120.d80 (Table 13)	2/10	0/10	1/10	**7/10**
be150.d30 (Table 14)	1/10	0/10	3/10	**6/10**
be150.d80 (Table 15)	**6/10**	0/10	3/10	1/10
be200.d30 (Table 16)	0/10	0/10	**9/10**	1/10
be200.d80 (Table 17)	1/10	0/10	**8/10**	1/10
be250.d10 (Table 18)	0/10	0/10	2/10	**8/10**
g05.n60 (Table 19)	3/10	**5/10**	1/10	3/10
g05.n80 (Table 20)	3/10	0/10	2/10	**5/10**
g05.n100 (Table 21)	**5/10**	0/10	0/10	**5/10**
pm1s80.d090 (Table 22)	**7/10**	**7/10**	1/10	2/10
pm1s100.d010 (Table 23)	2/10	1/10	1/10	**7/10**
pm1d80.d090 (Table 24)	2/10	0/10	**4/10**	**4/10**
pm1d100.d090 (Table 25)	1/10	0/10	1/10	**8/10**
w100.d010 (Table 26)	2/10	3/10	**5/10**	1/10
w100.d050 (Table 27)	3/10	0/10	3/10	**4/10**
w100.d090 (Table 28)	**4/10**	0/10	2/10	**4/10**
pw100.d010 (Table 29)	1/10	1/10	2/10	**7/10**
pw100.d050 (Table 30)	4/10	0/10	1/10	**5/10**
pw100.d090 (Table 31)	0/10	0/10	3/10	**7/10**
ising100 (Table 32)	0/6	0/6	**3/6**	**3/6**
ising150 (Table 33)	0/6	0/6	1/6	**5/6**
ising200 (Table 34)	0/6	0/6	2/6	**4/6**
ising250 (Table 35)	0/6	0/6	**4/6**	2/6
ising300 (Table 36)	0/6	0/6	2/6	**4/6**
t2g10 (Table 37)	**3/3**	**3/3**	0/3	0/3
t2g15 (Table 38)	0/3	0/3	**3/3**	0/3
t2g20 (Table 39)	0/3	0/3	1/3	**2/3**
t3g5 (Table 40)	**3/3**	**3/3**	0/3	0/3
t3g6 (Table 41)	0/3	1/3	**2/3**	0/3
t3g7 (Table 42)	**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%)**	

Table 2: Minimum / mean / maximum CPU time (s) for Biq Mac and our method to solve problems in the Biq Mac Library [29].

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	7052.28	54277.92	**47.13**	**1736.11**	**14714.12**
gkaa	**0.06**	**0.55**	**1.37**	**0.06**	0.65	2.11
gkab	1.11	1650.88	10719.44	**0.10**	**787.00**	**5104.21**
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	6116.31	25721.41	**21.77**	**3897.82**	**16654.15**
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	1079.71	8223.51	43850.81	**102.61**	**4852.01**	**32612.82**
be200.d80	1210.52	10692.92	35247.12	**596.22**	**6759.69**	**25518.98**
be250.d10	328.64	2437.95	3754.20	**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	4197.29	96.05	**721.25**	**3382.27**
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**	**1800.63**
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**	**1061.46**
w100.d090	38.67	**836.25**	**2945.89**	**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**	**1798.61**	**81.34**	732.48	2506.69
pw100.d090	209.02	606.82	1297.98	**201.61**	**509.48**	**1061.18**
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	2161.49	**47.88**	**138.27**	**406.84**
ising300	991.28	4268.83	11858.56	**97.43**	**455.77**	**1628.66**
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	1014.38	4676.45	11132.18	**351.37**	**1159.24**	**2748.55**
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**	5931.61	17601.56	123.46	**3838.06**	**11234.02**

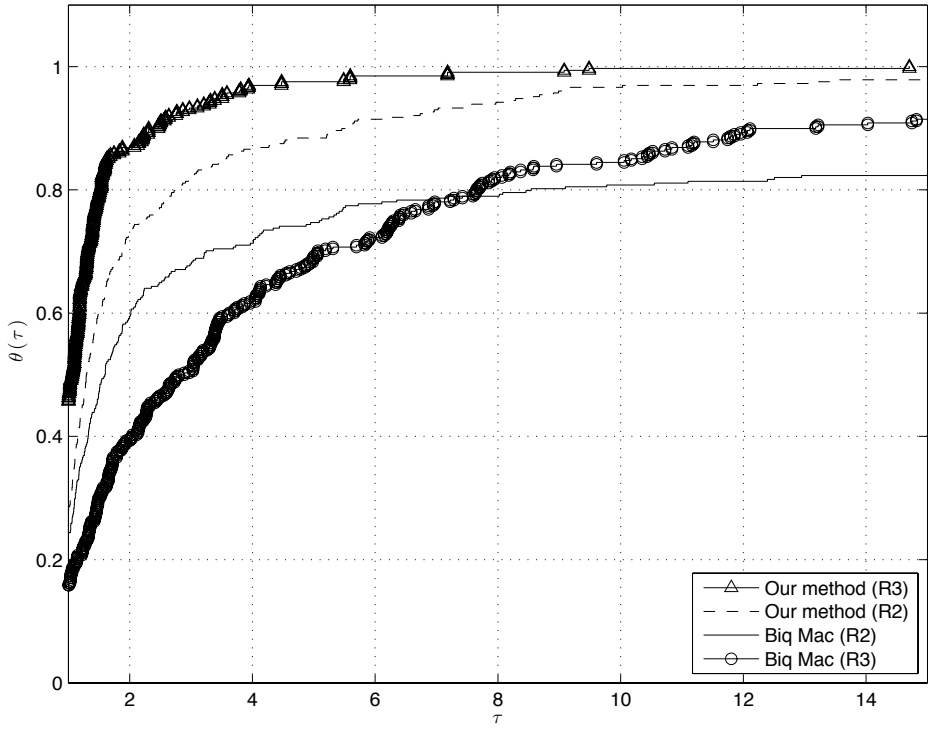


Figure 2: Performance profile curves (29) for the four solvers on the 328 problems.

only the "hard problems" (those 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 in Appendix A; see, e.g., bqp250 (Table 5), be250.d10 (Table 18), and ising300 (Table 36). Additionally, we note that both our methods are able to solve all test problems within the 100 hour time limit (Biq Mac (R3) fails to solve 6 instances within this time limit).

The performance profile in Figure 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.

Moreover, 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 based on the general bounds of [19], it can also be applied to binary quadratic problems with additional linear or quadratic constraints. In our future work, we will consider this generalization. Due to the inherent flexibility of our method, 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, third edn. Society for Industrial and Applied Mathematics, Philadelphia, PA (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 US (2012)

[3] Anjos, M.F., Lasserre, J.B.: Introduction to semidefinite, conic and polynomial optimization. In: M.F. Anjos, J.B. Lasserre (eds.) Handbook on Semidefinite, Conic and Polynomial Optimization, *International Series in Operations Research & Management Science*, vol. 166, pp. 1–22. Springer US (2012)

[4] Billionnet, A., Elloumi, S.: Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. Mathematical Programming **109**, 55–68 (2007)

[5] Bonnans, J., Gilbert, J., Lemaréchal, C., Sagastizábal, C.: Numerical Optimization. Springer Verlag (2003)

[6] Borwein, J.M., Lewis, A.S.: Convex Analysis and Nonlinear Optimization : Theory and Examples. Springer-Verlag (2000)

[7] Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2008)

[8] Byrd, R.H., Lu, P., Nocedal, J., Zhu, C.: A limited memory algorithm for bound constrained optimization. SIAM Journal on Scientific Computing **16**(5), 1190–1208 (1995)

[9] Cun, B.L., Roucairol, C., The PNN Team: BOB: A unified platform for implementing branch-and-bound like algorithms. Tech. Rep. 95/16, University of Versailles Saint-Quentin-en-Yvelines (1995)

[10] Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Mathematical Programming **91**, 201–213 (2002)

[11] 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)

[12] Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd edn. Johns Hopkins University Press, Baltimore, MD (1996)

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

[14] Hiriart-Urruty, J.B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms. Springer Verlag, Heidelberg (1993). Two volumes

[15] Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972), pp. 85–103. Plenum, New York (1972)

[16] Laurent, M., Poljak, S.: Gap inequalities for the cut polytope. European Journal of Combinatorics **17**(2-3), 233 – 254 (1996)

[17] Lemaréchal, C., Oustry, F.: Semidefinite relaxations and Lagrangian duality with application to combinatorial optimization. Rapport de Recherche 3710, INRIA (1999)

[18] Malick, J.: Spherical constraint in Boolean quadratic programming. Journal of Global Optimization **39**(4) (2007)

[19] Malick, J., Roupin, F.: On the bridge between combinatorial optimization and nonlinear optimization: a family of semidefinite bounds leading to Newton-like methods. To appear in Mathematical Programming (2011). Available online as preprint hal-00662367

[20] Malick, J., Roupin, F.: Solving k-cluster problems to optimality using adjustable semidefinite programming bounds. To appear in Math. Programming B, special issue on Mixed-Integer Nonlinear Programming (2011)

[21] 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)

[22] Nocedal, J., Wright, S.J.: Numerical Optimization, 2nd edn. Springer (2006)

[23] Palagi, L., Piccialli, V., Rendl, F., Rinaldi, G., Wiegele, A.: Computational approaches to Max-Cut. In: M.F. Anjos, J.B. Lasserre (eds.) Handbook on Semidefinite, Conic and Polynomial Optimization, *International Series in Operations Research & Management Science*, vol. 166, pp. 821–847. Springer US (2012)

[24] Pardalos, P., Rodgers, G.: Computational aspects of a branch and bound algorithm for quadratic zero-one programming. Computing **45**, 134–144 (1990)

[25] Pardalos, P.M., Vavasis, S.A.: Quadratic programming with one negative eigenvalue is NP-hard. Journal of Global Optimization **1**, 15–22 (1991)

[26] Poljak, S., Rendl, F., Wolkowicz, H.: A recipe for semidefinite relaxation for (0,1)-quadratic programming. Journal of Global Optimization **7**, 51–73 (1995)

[27] Rendl, F., Rinaldi, G., Wiegele, A.: Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. Mathematical Programming **121**, 307–335 (2010)

[28] Wiegele, A.: Nonlinear optimization techniques applied to combinatorial optimization problems. Ph.D. thesis, Alpen-Adria-Universität Klagenfurt (2006)

[29] Wiegele, A.: Biq Mac Library—A collection of Max-Cut and quadratic 0-1 programming instances of medium size. Tech. rep., Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria (2007)

[30] 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)

A Tables of numerical results

Table 3: CPU time (s) and nodes to solve the bqp50 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
1	11.19	5.03	23	7	0.35	**0.25**	1	1
2	0.31	0.31	1	1	0.58	**0.24**	1	1
3	**0.11**	**0.11**	1	1	0.98	0.17	1	1
4	0.25	0.25	1	1	0.63	**0.22**	1	1
5	**0.20**	**0.20**	1	1	0.31	0.21	1	1
6	**0.13**	**0.13**	1	1	0.25	0.21	1	1
7	**0.22**	**0.22**	1	1	0.27	0.26	1	1
8	**0.16**	**0.16**	1	1	0.26	0.19	1	1
9	**0.18**	**0.18**	1	1	0.29	0.25	1	1
10	0.45	0.45	1	1	0.29	**0.24**	1	1
Total:	**6/10**	**6/10**			0/10	4/10		

Table 4: CPU time (s) and nodes to solve the bqp100 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
1	2.40	2.40	1	1	**1.40**	1.72	1	1
2	2.80	2.80	1	1	9.54	**2.06**	1	1
3	**1.14**	**1.14**	1	1	7.39	1.60	3	1
4	**0.92**	**0.92**	1	1	3.08	1.42	1	1
5	**1.43**	**1.43**	1	1	3.99	1.69	1	1
6	139.09	22.29	169	7	40.10	**11.12**	13	3
7	1.87	1.87	1	1	5.60	**1.80**	1	1
8	**1.62**	**1.62**	1	1	1.83	1.63	1	1
9	**1.11**	**1.11**	1	1	3.10	1.46	1	1
10	2.12	2.12	1	1	5.84	**1.67**	1	1
Total:	**5/10**	**5/10**			1/10	4/10		

Table 5: CPU time (s) and nodes to solve the bqp250 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
1	3709.00	1790.82	291	41	660.04	**171.08**	37	3
2	3094.08	1063.38	173	25	502.86	**166.32**	31	3
3	4220.53	526.18	309	11	65.49	**47.13**	1	1
4	3425.14	1615.88	243	37	**53.04**	185.95	1	3
5	3590.11	727.07	293	17	2122.92	**55.17**	101	1
6	3937.78	13014.86	309	323	2953.55	**1213.87**	331	19
7	3872.44	1119.27	297	25	917.86	**173.05**	55	3
8	54277.92	>360000.00	4341	>15387	40039.83	**14714.12**	5571	307
9	3398.68	3011.68	341	73	3241.21	**373.52**	263	7
10	3363.66	2452.87	233	59	2651.67	**393.81**	189	7
Total:	0/10	0/10			1/10	**9/10**		

Table 6: CPU time (s) and nodes to solve the gkaa problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
1	1.03	1.03	1	1	**0.25**	0.33	1	1
2	**0.19**	**0.19**	1	1	0.58	0.42	1	1
3	1.37	1.37	1	1	1.74	**0.93**	1	1
4	**0.82**	**0.82**	1	1	1.45	1.07	1	1
5	**0.18**	**0.18**	1	1	0.29	0.29	1	1
6	0.08	0.08	1	1	0.07	**0.06**	1	1
7	**0.06**	**0.06**	1	1	**0.06**	0.07	1	1
8	**0.68**	**0.68**	1	1	3.54	2.11	1	1
Total:	**5/8**	**5/8**			2/8	2/8		

Table 7: CPU time (s) and nodes to solve the gkab problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
1	1.11	1.11	1	1	1.83	**0.10**	13	1
2	5.97	9.54	43	33	4.76	**2.18**	29	3
3	12.69	33.03	47	83	11.95	**9.44**	53	13
4	29.59	108.85	89	173	24.36	**18.42**	79	27
5	66.27	300.14	161	335	**39.20**	42.15	85	43
6	131.01	1044.74	175	853	**86.56**	94.96	163	67
7	545.12	2509.01	551	1459	**212.73**	266.60	331	113
8	1630.28	11068.75	1623	6009	851.14	**789.10**	1379	331
9	3367.32	30173.75	3609	14173	2008.11	**1608.04**	3083	703
10	10719.44	178991.20	5881	43609	**5104.21**	7968.24	6309	2107
Total:	0/10	0/10			4/10	**6/10**		

Table 8: CPU time (s) and nodes to solve the gkac problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
1	0.16	0.16	1	1	**0.13**	0.15	1	1
2	**0.17**	**0.17**	1	1	0.27	0.25	1	1
3	**0.38**	**0.38**	1	1	0.44	0.43	1	1
4	0.97	0.97	1	1	**0.73**	0.86	1	1
5	**0.72**	**0.72**	1	1	1.36	0.93	1	1
6	**1.02**	**1.02**	1	1	1.97	1.54	1	1
7	**0.94**	**0.94**	1	1	2.99	1.95	1	1
Total:	**5/7**	**5/7**			2/7	0/7		

Table 9: CPU time (s) and nodes to solve the gkad problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
1	**1.53**	**1.53**	1	1	1.96	2.10	1	1
2	15.93	9.59	5	3	**1.63**	2.65	1	1
3	34.73	11.38	13	3	**1.85**	2.79	1	1
4	124.40	35.21	77	9	**3.25**	4.23	1	1
5	112.72	74.91	129	21	86.70	**55.72**	125	11
6	73.86	10.09	31	3	**2.14**	3.03	1	1
7	118.68	79.33	123	25	84.20	**46.04**	111	9
8	57.30	11.26	21	3	3.08	**2.94**	1	1
9	127.02	50.54	139	15	100.76	**34.08**	103	7
10	124.43	45.92	117	13	114.69	**30.97**	93	5
Total:	1/10	1/10			4/10	**5/10**		

Table 10: CPU time (s) and nodes to solve the gkae problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
1	1391.44	126.98	129	5	285.89	**21.77**	37	1
2	1902.51	13839.63	413	665	**1047.08**	2581.48	373	87
3	1436.94	25243.00	241	289	**896.16**	1159.87	237	43
4	1393.70	3876.28	311	177	928.96	**869.92**	293	31
5	25721.41	>360000.00	4717	>19270	**16654.15**	41480.79	4977	1705
Total:	0/5	0/5			**3/5**	2/5		

Table 11: CPU time (s) and nodes to solve the be100.d100 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
1	145.63	19.84	107	5	**8.07**	11.06	1	1
2	133.79	18.53	83	5	**3.79**	6.27	1	1
3	113.25	16.70	61	5	**4.83**	5.73	1	1
4	134.86	66.40	153	17	214.01	**40.08**	101	5
5	108.61	37.96	89	11	206.99	**25.98**	55	3
6	123.99	17.93	69	5	8.13	**6.82**	1	1
7	122.58	30.69	109	9	203.55	**22.37**	79	3
8	115.35	**108.69**	141	31	157.96	114.88	119	15
9	120.99	89.84	153	27	127.67	**84.72**	107	11
10	90.89	**22.22**	53	7	154.37	23.57	57	3
Total:	0/10	2/10			3/10	**5/10**		

Table 12: CPU time (s) and nodes to solve the be120.d30 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
1	223.63	113.27	161	19	432.05	**76.30**	125	5
2	92.97	15.33	23	3	8.99	**7.18**	1	1
3	120.97	16.76	31	3	11.50	**7.76**	1	1
4	248.71	40.07	137	7	14.49	**12.88**	1	1
5	147.27	25.69	43	5	**6.25**	8.12	1	1
6	40.13	14.98	9	3	7.25	**6.57**	1	1
7	**6.10**	**6.10**	1	1	9.42	6.77	1	1
8	**4.52**	**4.52**	1	1	5.01	6.62	1	1
9	168.79	64.48	67	11	293.80	**38.91**	61	3
10	164.08	25.17	55	5	**9.13**	12.15	1	1
Total:	2/10	2/10			2/10	**6/10**		

Table 13: CPU time (s) and nodes to solve the be120.d80 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
1	194.02	605.39	169	121	**176.44**	387.06	155	35
2	211.55	211.99	151	43	210.19	**155.44**	127	17
3	221.43	117.90	135	21	265.80	**114.50**	131	9
4	246.84	39.12	163	7	284.60	**34.34**	37	3
5	207.09	43.70	91	7	120.71	**32.60**	19	3
6	218.58	120.64	137	23	284.45	**97.73**	109	7
7	**185.74**	402.85	171	75	239.83	290.84	163	27
8	**194.87**	896.84	173	191	268.07	500.34	159	47
9	234.79	182.72	171	35	278.10	**177.83**	137	15
10	241.88	170.65	183	31	323.77	**128.65**	157	9
Total:	2/10	0/10			1/10	**7/10**		

Table 14: CPU time (s) and nodes to solve the be150.d30 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
1	552.72	170.68	145	15	784.90	**111.72**	101	7
2	487.97	162.74	125	15	750.87	**90.88**	125	7
3	229.30	39.21	29	3	11.00	**7.74**	1	1
4	656.06	111.98	181	9	**27.47**	32.53	1	3
5	560.44	344.08	193	31	769.36	**101.60**	181	7
6	529.45	2214.69	215	219	511.70	**487.36**	219	35
7	572.02	340.56	163	29	703.18	**128.91**	121	9
8	**607.62**	3461.01	243	359	665.17	909.82	243	63
9	844.33	5863.95	231	671	**767.08**	1072.17	227	87
10	528.52	4004.32	253	427	**466.50**	1004.20	225	65
Total:	1/10	0/10			3/10	**6/10**		

Table 15: CPU time (s) and nodes to solve the be150.d80 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
1	499.99	1207.49	225	129	**483.87**	593.74	195	49
2	536.48	7004.88	243	715	**475.02**	1872.56	233	155
3	**441.62**	3404.46	257	329	443.02	1019.10	239	71
4	526.56	591.11	167	59	468.46	**363.55**	151	27
5	**482.92**	1604.64	255	137	542.07	572.02	237	39
6	**501.99**	2033.05	223	201	536.32	802.06	205	59
7	**525.18**	5880.48	273	565	608.20	1211.89	217	95
8	**507.49**	1963.59	209	189	522.82	760.43	205	53
9	**477.27**	3903.36	253	413	544.90	1615.86	235	121
10	561.98	1634.39	217	157	**480.67**	568.80	207	43
Total:	**6/10**	0/10			3/10	1/10		

Table 16: CPU time (s) and nodes to solve the be200.d30 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
1	43850.81	>360000.00	7927	>25405	**32612.82**	56902.36	8985	2399
2	2553.97	87629.96	461	2627	**1479.82**	10624.76	503	355
3	1570.50	9464.23	305	351	**798.46**	1345.49	285	43
4	2597.34	90241.20	527	2251	**844.91**	7671.43	369	253
5	1630.94	52812.92	339	1121	**808.48**	4523.10	343	155
6	2054.03	34443.12	395	801	**778.60**	3057.63	293	119
7	1571.54	1079.71	195	25	642.16	**102.61**	69	3
8	4503.36	83747.75	685	2097	**1135.38**	5076.93	373	209
9	7737.93	74078.57	1217	2639	**2551.76**	6329.74	693	251
10	14656.52	319573.82	2359	11431	**7407.25**	22016.54	1935	861
Total:	0/10	0/10			**9/10**	1/10		

Table 17: CPU time (s) and nodes to solve the be200.d80 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
1	**1210.52**	25458.48	307	951	1572.38	4581.21	321	137
2	35247.12	>360000.00	5907	>34105	**25518.98**	81821.55	6581	3597
3	26318.86	>360000.00	6069	>23691	**14474.43**	64771.02	6067	2789
4	1866.05	77647.69	403	1785	**1062.46**	7627.42	389	305
5	9189.31	241577.21	1671	6189	**4949.74**	17366.65	1417	743
6	1580.43	3643.09	355	87	1051.23	**596.22**	239	21
7	1344.09	77552.67	345	1963	**712.30**	6756.64	333	265
8	25950.40	>360000.00	4941	>14429	**16204.96**	44663.86	5053	1895
9	1499.53	11130.84	259	211	**930.82**	935.36	253	37
10	2722.89	111291.76	509	3035	**1574.58**	8805.80	441	363
Total:	1/10	0/10			**8/10**	1/10		

Table 18: CPU time (s) and nodes to solve the be250.d10 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
1	3703.05	2098.62	319	49	2473.27	**182.31**	255	3
2	3611.26	1516.95	277	35	3618.13	**176.91**	367	3
3	3996.90	896.11	187	19	**34.85**	51.04	1	1
4	4157.19	1522.45	353	35	1518.58	**151.37**	153	3
5	3754.20	4516.16	305	113	2731.70	**903.37**	361	15
6	3471.50	3285.36	225	83	2440.30	**432.71**	215	7
7	2394.79	328.64	81	7	**27.98**	39.76	1	1
8	3675.89	3826.01	295	87	2980.22	**529.83**	307	9
9	3636.68	3992.81	311	97	3243.97	**905.98**	411	13
10	3664.60	3708.73	451	85	3920.43	**751.78**	443	13
Total:	0/10	0/10			2/10	**8/10**		

Table 19: CPU time (s) and nodes to solve the g05.n60 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
0	6.91	3.34	15	5	8.74	**2.15**	19	3
1	1.98	2.02	3	3	**0.89**	1.00	1	1
2	9.08	**6.87**	17	11	7.08	7.83	23	11
3	**0.26**	**0.26**	1	1	0.67	0.99	1	1
4	**13.27**	21.45	43	41	24.84	28.31	51	31
5	**0.34**	**0.34**	1	1	5.89	5.00	5	3
6	11.04	**8.42**	29	15	12.08	9.31	37	11
7	13.36	**4.48**	33	7	7.52	12.83	15	7
8	7.95	5.23	17	9	6.98	**4.95**	19	7
9	13.26	14.39	43	23	15.35	**11.80**	39	25
Total:	3/10	**5/10**			1/10	3/10		

Table 20: CPU time (s) and nodes to solve the g05.n80 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
0	49.16	66.56	77	61	55.22	**43.71**	89	47
1	14.25	6.39	13	5	**2.80**	3.66	1	1
2	34.09	20.54	43	17	**18.88**	22.24	45	15
3	**274.50**	512.58	385	533	296.10	386.78	571	355
4	40.71	52.76	69	45	43.36	**36.54**	79	29
5	**52.10**	53.65	71	49	67.16	75.93	109	77
6	**36.44**	40.81	59	33	42.73	36.54	71	31
7	24.45	18.48	29	15	28.41	**18.26**	43	17
8	51.92	64.20	71	61	73.96	**37.41**	131	47
9	80.64	141.93	113	141	119.53	**73.64**	199	87
Total:	3/10	0/10			2/10	**5/10**		

Table 21: CPU time (s) and nodes to solve the g05.n100 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
0	665.41	2194.05	547	1301	1060.48	**663.41**	1105	417
1	4197.29	14856.75	3533	9565	4527.49	**3382.27**	5345	2293
2	**151.21**	448.26	125	237	291.61	209.21	399	113
3	1540.31	4409.16	1157	2803	2143.00	**1304.29**	2533	859
4	**93.45**	133.74	75	67	104.66	96.05	105	51
5	**126.56**	339.26	117	171	325.38	182.40	313	97
6	**197.72**	422.34	181	227	346.61	237.33	355	131
7	**351.58**	1176.15	297	633	479.94	540.45	567	283
8	344.37	904.24	261	509	643.44	**331.47**	719	257
9	370.94	852.80	279	503	599.54	**326.16**	671	275
Total:	**5/10**	0/10			0/10	**5/10**		

Table 22: CPU time (s) and nodes to solve the pm1s80.d090 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
0	**0.65**	**0.65**	1	1	1.64	1.19	1	1
1	**0.58**	**0.58**	1	1	1.63	1.50	1	1
2	26.71	23.96	27	17	26.74	**13.40**	41	13
3	**1.19**	**1.19**	1	1	1.73	1.90	1	1
4	**0.65**	**0.65**	1	1	0.96	2.33	1	1
5	**0.90**	**0.90**	1	1	1.05	1.22	1	1
6	14.74	12.53	13	9	7.88	**6.61**	13	3
7	**1.33**	**1.33**	1	1	7.41	4.41	11	3
8	17.59	4.64	15	3	**3.22**	8.08	3	7
9	**1.13**	**1.13**	1	1	5.10	6.20	1	1
Total:	**7/10**	**7/10**			1/10	2/10		

Table 23: CPU time (s) and nodes to solve the pm1s100.d010 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
0	26.44	27.62	15	15	55.87	**16.78**	49	13
1	130.28	143.84	93	93	190.01	**88.48**	193	47
2	18.80	20.78	9	9	24.77	**12.27**	29	9
3	100.00	110.81	59	59	286.00	**67.01**	305	45
4	80.33	88.18	51	51	63.50	**39.82**	75	25
5	**27.45**	30.03	15	15	73.63	33.63	89	23
6	80.07	88.12	51	51	137.16	**71.86**	171	47
7	**1.15**	**1.15**	1	1	1.85	2.55	1	1
8	10.88	12.16	5	5	34.03	**5.95**	29	3
9	23.26	25.69	11	11	**9.39**	13.08	5	5
Total:	2/10	1/10			1/10	**7/10**		

Table 24: CPU time (s) and nodes to solve the pm1d80.d090 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
0	33.17	27.54	51	23	37.43	**15.07**	59	17
1	**111.02**	293.45	173	275	114.29	171.07	243	131
2	64.88	131.77	99	113	**43.10**	55.73	113	57
3	30.36	31.32	39	27	30.85	**18.44**	45	19
4	37.74	46.06	65	37	**31.43**	44.47	75	45
5	212.47	419.49	307	405	**138.19**	177.21	365	185
6	105.25	114.51	147	109	67.88	**58.97**	149	65
7	**64.04**	211.43	111	183	109.78	98.84	253	111
8	41.18	44.06	59	37	**30.98**	33.19	65	37
9	29.88	24.56	35	21	28.31	**13.74**	47	15
Total:	2/10	0/10			**4/10**	**4/10**		

Table 25: CPU time (s) and nodes to solve the pm1d100.d090 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
0	1229.18	8419.09	911	2281	1414.77	**1026.52**	1507	667
1	2868.06	17301.82	2157	4717	3236.25	**1800.63**	3575	1207
2	1863.13	11310.46	1375	3523	2047.23	**1092.72**	2443	823
3	**376.93**	1955.87	281	553	379.67	386.88	587	275
4	1651.17	4804.95	1203	2241	1138.05	**948.92**	1771	729
5	381.03	1900.61	271	475	335.83	**276.72**	485	189
6	350.15	1289.96	243	305	246.31	**207.73**	389	139
7	106.24	499.16	109	109	**80.74**	114.22	141	65
8	134.02	358.58	99	81	116.56	**56.94**	133	39
9	494.13	2500.91	355	637	430.41	**336.99**	529	241
Total:	1/10	0/10			1/10	**8/10**		

Table 26: CPU time (s) and nodes to solve the w100.d010 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
0	27.70	7.09	13	3	**5.28**	12.13	1	1
1	57.66	**17.95**	35	3	43.34	21.59	23	3
2	**131.45**	296.92	77	137	595.77	152.09	491	47
3	43.14	**13.46**	23	5	35.67	13.81	33	5
4	2.12	2.12	1	1	**1.85**	2.47	1	1
5	18.31	16.32	9	7	24.89	**9.76**	17	3
6	2.35	2.35	1	1	**2.22**	2.64	1	1
7	50.86	31.53	27	13	**24.06**	26.43	23	11
8	**1.37**	**1.37**	1	1	1.71	2.08	1	1
9	12.45	7.85	5	3	**2.30**	2.69	1	1
Total:	2/10	3/10			**5/10**	1/10		

Table 27: CPU time (s) and nodes to solve the w100.d050 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
0	**945.25**	2164.97	737	1217	1335.05	1061.46	969	459
1	**211.42**	515.41	191	263	290.43	311.70	229	109
2	**175.46**	357.04	143	177	186.81	202.08	175	63
3	612.54	2087.57	489	1159	825.00	**563.16**	949	319
4	1040.41	2704.91	835	1499	964.91	**720.19**	1125	357
5	1027.07	2436.14	789	1331	1137.77	**802.39**	1349	413
6	220.65	587.59	171	305	**188.11**	189.44	223	93
7	137.39	196.16	105	93	**84.50**	86.61	93	39
8	1152.49	2879.08	875	1637	1069.53	**775.14**	1255	465
9	109.99	135.18	91	65	**81.02**	83.06	87	33
Total:	3/10	0/10			3/10	**4/10**		

Table 28: CPU time (s) and nodes to solve the w100.d090 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
0	**539.51**	1163.23	411	629	1038.44	770.67	653	271
1	**2945.89**	14097.89	2525	8485	5897.16	4675.78	4413	1821
2	1364.49	4619.19	1087	2711	**1167.26**	1519.29	1179	765
3	**1487.39**	6598.30	1199	3939	1637.23	1630.03	1789	897
4	717.48	1943.66	589	1065	800.85	**608.70**	845	293
5	79.69	38.67	57	17	28.48	**26.50**	23	9
6	**89.86**	190.11	103	87	131.40	138.67	131	57
7	377.18	949.78	271	509	374.73	**299.03**	391	157
8	205.76	439.04	213	209	**187.90**	225.81	209	105
9	596.26	1637.01	459	903	571.00	**473.28**	637	267
Total:	**4/10**	0/10			2/10	**4/10**		

Table 29: CPU time (s) and nodes to solve the pw100.d010 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
0	32.32	28.47	15	11	94.80	**27.52**	63	13
1	144.42	112.86	107	47	82.52	**49.65**	83	15
2	53.49	57.39	33	25	174.22	**49.84**	123	21
3	95.28	120.84	61	55	109.82	**35.61**	99	15
4	15.11	8.02	7	3	**2.64**	3.10	1	1
5	20.20	8.33	9	3	**2.50**	3.20	1	1
6	56.65	51.35	29	23	37.24	**23.02**	35	11
7	89.73	65.23	57	27	60.77	**42.09**	57	15
8	**1.64**	**1.64**	1	1	2.06	2.15	1	1
9	228.70	244.33	157	111	172.00	**113.86**	191	43
Total:	1/10	1/10			2/10	**7/10**		

Table 30: CPU time (s) and nodes to solve the pw100.d050 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
0	**1798.61**	7433.26	1447	4679	3732.88	2506.69	2949	1161
1	**529.82**	1787.33	413	1033	692.52	628.58	745	357
2	831.04	2315.65	697	1381	1136.32	**710.04**	1095	373
3	**145.66**	371.98	119	197	164.25	157.69	185	85
4	**911.92**	2763.95	765	1629	1188.62	970.65	1305	549
5	232.10	374.56	169	201	232.79	**218.71**	259	117
6	1726.73	5507.96	1401	3327	1751.65	**1359.81**	2031	737
7	454.48	913.14	333	485	384.06	**300.39**	429	153
8	122.67	196.11	121	91	103.26	**81.34**	113	37
9	406.52	1093.26	315	611	**390.92**	397.39	445	219
Total:	4/10	0/10			1/10	**5/10**		

Table 31: CPU time (s) and nodes to solve the pw100.d090 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
0	753.88	4508.30	559	1175	1122.32	**677.50**	979	321
1	971.02	6800.50	723	1827	1113.34	**883.57**	1237	485
2	362.03	1957.67	267	501	354.55	**334.49**	405	199
3	209.02	1261.49	155	301	**201.61**	255.36	215	131
4	544.85	2324.07	401	575	396.38	**363.92**	467	195
5	647.55	3797.79	533	919	**605.58**	658.84	733	337
6	577.00	2826.12	391	717	448.18	**405.48**	541	261
7	1297.98	8382.66	981	2253	1112.62	**1061.18**	1375	621
8	381.88	2154.22	303	515	347.05	**313.59**	393	175
9	323.01	1536.21	247	369	**247.90**	288.67	289	145
Total:	0/10	0/10			3/10	**7/10**		

Table 32: CPU time (s) and nodes to solve the ising100 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
251005555	165.65	19.86	113	7	**2.86**	3.15	1	1
251006666	141.74	9.41	65	3	**2.62**	2.96	1	1
251007777	145.33	10.01	75	3	2.96	**2.76**	1	1
301005555	142.05	18.78	99	7	**3.87**	4.51	1	1
301006666	69.03	7.74	115	3	4.36	**2.98**	1	1
301007777	33.13	10.76	11	3	3.51	**3.14**	1	1
Total:	0/6	0/6			**3/6**	**3/6**		

Table 33: CPU time (s) and nodes to solve the ising150 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
251505555	505.60	54.44	99	7	14.49	**12.89**	1	1
251506666	1060.72	70.97	151	7	9.04	**8.48**	1	1
251507777	630.15	85.74	139	9	**10.79**	12.19	1	1
301505555	538.02	48.18	199	5	15.86	**11.72**	1	1
301506666	651.94	59.97	141	7	7.94	**7.74**	1	1
301507777	460.91	42.55	191	5	15.38	**12.57**	1	1
Total:	0/6	0/6			1/6	**5/6**		

Table 34: CPU time (s) and nodes to solve the ising200 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
252005555	1880.36	150.39	231	9	23.37	**19.72**	1	1
252006666	2923.63	194.08	317	9	**24.72**	27.52	1	1
252007777	2139.40	403.19	475	19	**27.20**	30.21	1	1
302005555	1329.85	212.69	273	11	27.77	**23.75**	1	1
302006666	2529.97	247.57	271	13	39.85	**37.59**	1	1
302007777	1054.57	195.99	255	9	31.61	**30.58**	1	1
Total:	0/6	0/6			2/6	**4/6**		

Table 35: CPU time (s) and nodes to solve the ising250 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
252505555	3440.53	2161.49	393	59	3490.20	**406.84**	125	7
252506666	4499.77	885.76	145	25	**53.01**	174.96	1	3
252507777	3600.15	1066.09	641	27	3548.82	**213.82**	201	3
302505555	8293.36	1029.29	859	29	**47.88**	61.41	1	1
302506666	4366.78	165.79	501	5	**52.41**	53.64	1	1
302507777	3013.34	580.93	191	17	**55.69**	73.98	1	1
Total:	0/6	0/6			**4/6**	2/6		

Table 36: CPU time (s) and nodes to solve the ising300 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
253005555	21859.40	3420.25	931	55	**102.98**	143.62	1	1
253006666	23203.24	4624.07	1839	71	**110.44**	121.36	1	1
253007777	27282.55	11858.56	1327	175	7108.80	**1628.66**	141	15
303005555	7605.89	1201.86	545	19	118.33	**99.26**	1	1
303006666	4235.17	991.28	131	15	1549.33	**97.43**	23	1
303007777	8623.52	3516.96	541	57	6342.78	**695.85**	123	5
Total:	0/6	0/6			2/6	**4/6**		

Table 37: CPU time (s) and nodes to solve the t2g10 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
5555	**1.98**	**1.98**	1	1	5.30	5.48	1	1
6666	**1.63**	**1.63**	1	1	3.63	4.30	1	1
7777	**3.04**	**3.04**	1	1	3.50	4.40	1	1
Total:	**3/3**	**3/3**			0/3	0/3		

Table 38: CPU time (s) and nodes to solve the t2g15 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
5555	157.29	56.55	3	3	**38.87**	43.13	1	1
6666	79.80	73.34	3	3	**42.44**	49.30	1	1
7777	245.17	69.18	5	3	**46.03**	57.02	1	1
Total:	0/3	0/3			**3/3**	0/3		

Table 39: CPU time (s) and nodes to solve the t2g20 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
5555	24877.86	1014.38	225	7	**351.37**	386.62	1	1
6666	72586.37	11132.18	715	75	33617.12	**2748.55**	447	7
7777	23906.54	1882.78	221	13	8668.10	**377.79**	27	1
Total:	0/3	0/3			1/3	**2/3**		

Table 40: CPU time (s) and nodes to solve the t3g5 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
5555	**3.22**	**3.22**	1	1	7.35	7.18	1	1
6666	**5.49**	**5.49**	1	1	7.27	7.57	1	1
7777	**4.55**	**4.55**	1	1	7.71	8.55	1	1
Total:	**3/3**	**3/3**			0/3	0/3		

Table 41: CPU time (s) and nodes to solve the t3g6 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
5555	62.40	52.34	3	3	**41.85**	48.24	1	1
6666	1882.42	**367.12**	165	17	3300.55	566.66	83	9
7777	116.28	58.61	5	3	**46.16**	54.61	1	1
Total:	0/3	1/3			**2/3**	0/3		

Table 42: CPU time (s) and nodes to solve the t3g7 problems.

Prob.	Biq Mac time		Biq Mac nodes		Our time		Our nodes	
	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)	(R2)	(R3)
5555	**81.37**	**81.37**	1	1	123.46	152.74	1	1
6666	**111.89**	**111.89**	1	1	156.69	178.42	1	1
7777	24999.25	17601.56	549	191	78224.93	**11234.02**	575	81
Total:	**2/3**	**2/3**			0/3	1/3		