Comput Optim Appl (2011) 48: 653–673
DOI 10.1007/s10589-009-9273-2

Relaxing the optimality conditions of box QP

Samuel Burer · Jieqiu Chen

Received: 24 October 2007 / Published online: 8 July 2009

Abstract We present semidefinite relaxations of nonconvex, box-constrained quadratic programming, which incorporate the first- and second-order necessary optimality conditions, and establish theoretical relationships between the new relaxations and a basic semidefinite relaxation due to Shor. We compare these relaxations in the context of branch-and-bound to determine a global optimal solution, where it is shown empirically that the new relaxations are significantly stronger than Shor's. An effective branching strategy is also developed.

1 Introduction

In this paper, we study semidefinite programming (SDP) relaxations for the fundamental problem of minimizing a nonconvex quadratic function over a box:

$$\min \left\{ \frac{1}{2} x^T Q x + c^T x : 0 \leq x \leq e \right\}, \tag{1}$$

where $x \in \Re^n$, $Q \in \Re^{n \times n}$, $c \in \Re^n$, and $e \in \Re^n$ is the all-ones vector. Without loss of generality, we assume Q is symmetric. If Q is not positive semidefinite (as we assume in this paper), then (1) is NP-hard [9].

There are numerous methods for solving (1) and more general nonconvex quadratic programs, including local methods [4] and global methods [8]. For a survey

Both authors supported in part by NSF Grant CCF-0545514.

S. Burer · J. Chen (✉)
Department of Management Sciences, University of Iowa, Iowa City, IA 52242-1994, USA
e-mail: jieqiu-chen@uiowa.edu

S. Burer
e-mail: samuel-burer@uiowa.edu

of methods to solve (1) globally, see De Angelis et al. [3] as well as Vandenbussche and Nemhauser [13, 14] and Burer and Vandenbussche [2].

Critical to any global optimization method for (1) is the ability to relax it into a convex problem, one which hopefully provides a tight lower bound on the optimal value with low computational cost. One standard approach is to linearize the quadratic term $x_i x_j$ via a single variable X_{ij} and then to enforce implied linear constraints, which link X_{ij} with x_i and x_j, e.g., $0 \leq X_{ij} \leq \min\{x_i, x_j\}$ [10]. The resulting relaxation is a linear program. A second approach also linearizes the terms $x_i x_j$—by introducing a symmetric matrix variable X to replace the aggregate xx^T—but then includes the valid semidefinite inequality $X \succeq xx^T$ to obtain an SDP relaxation.

In this paper, we focus on SDP relaxations of (1) rather than linear ones. In principle, it is always possible to combine linear and semidefinite approaches (yielding better bounds with added computational costs; see [1]), but the goal of this paper is to improve SDP relaxations.

Our approach is to consider semidefinite relaxations of (1), which incorporate the standard first- and second-order necessary optimality conditions for (1). Vandenbussche and Nemhauser [13, 14] and Burer and Vandenbussche [2] have previously considered linear and semidefinite relaxations, respectively, involving only the first-order conditions. The contributions of the current paper are to demonstrate how also to incorporate the second-order conditions and to illustrate the positive effects of doing so.

We point out that Nesterov [7] has considered incorporating the second-order conditions into SDP relaxations of quadratic optimization over p-norm boxes for $2 \leq p < \infty$, i.e., $\{x : \|x\|_p^p \leq 1\}$. However, Nesterov strongly uses the fact that the function $\|x\|_p^p$ is smooth for $p \in [2, \infty)$. Our case (p equal to ∞) is wholly different because of the lack of smoothness.

The paper is organized as follows. In Sect. 2, we review the first- and second-order optimality conditions of (1). In particular, we show how to express the second-order conditions without explicit knowledge of the inactive constraints. This will prove to be a critical ingredient in constructing semidefinite relaxations involving the second-order conditions. In Sect. 3, we review the basic semidefinite relaxation of (1) due to Shor [11] and then introduce a semidefinite relaxation, which incorporates the first- and second-order optimality conditions. We also construct a relaxation based only on the second-order conditions.

We will call the three relaxations just mentioned (SDP_0), (SDP_{12}), and (SDP_2), respectively. The subscript indicates the type of "order" information incorporated in the relaxation. By construction, it will hold that (SDP_{12}) is at least as strong as (SDP_2), which is at least as strong as (SDP_0). On the other hand, (SDP_{12}) requires the largest solution time, while (SDP_0) requires the smallest one.

Continuing in Sects. 4 and 5, we study the relationship of these three relaxations. In Sect. 4, we prove the surprising, somewhat negative result that all three achieve the same optimal value. (On the positive side, the proof establishes several interesting analytical properties of (SDP_0), which are of independent interest.) Despite this equivalence, Sect. 5 demonstrates positively that, in the context of branch-and-bound to globally solve (1), (SDP_{12}) and (SDP_2) are significantly stronger than (SDP_0), when each is appropriately tailored for use at any node of the tree. Our computational

experiments are described in detail in Sect. 5, including a new, effective branching strategy.

1.1 Notation and terminology

In this section, we introduce some of the notation that will be used throughout the paper. $\Re^n$ refers to n-dimensional Euclidean space; $\Re^{n\times n}$ is the set of real, $n \times n$ matrices. We let $e_i \in \Re^n$ represent the i-th unit vector. For a set $\mathcal{I}$ in a particular ground set, $\mathcal{I}^c$ is its complement in that ground set. The norm of a vector $v \in \Re^n$ is denoted by $\|v\| := \sqrt{v^T v}$. For a vector v and an index set $\mathcal{I}$, $v_\mathcal{I}$ is defined as the vector composed of entries of v that are indexed by $\mathcal{I}$. Also, given a matrix $A \in \Re^{n\times n}$, $A_{\mathcal{I}\mathcal{I}}$ is defined as the matrix composed of entries of A whose rows and columns are indexed by $\mathcal{I}$. We denote by A^j and A_i the j-th column and i-th row of A, respectively. The notation $\mathrm{diag}(A)$ is defined as the vector, which is the diagonal of A, while $\mathrm{Diag}(v)$ denotes the diagonal matrix with diagonal v. The inner product of two matrices $A, B \in \Re^{n\times n}$ is defined as $A \bullet B := \mathrm{trace}(A^T B)$. Given two vectors $x, v \in \Re^n$, we denote their Hadamard product by $x \circ v \in \Re^n$, where $[x \circ v]_j = x_j v_j$; an analogous definition applies to the Hadamard product of matrices. Finally, $A \succeq 0$ means matrix A is positive semidefinite, and $A \succ 0$ means A is positive definite.

2 Optimality conditions

In this section, we first state the standard first- and second-order necessary optimality conditions for (1) involving the set of inactive constraints. Then we derive an expression for the second-order conditions that does not explicitly require knowledge of the inactive constraint set.

For any x satisfying $0 \le x \le e$, define the following sets of inactive constraints:

$$\begin{aligned}\mathcal{I}_0(x) &:= \{i : x_i > 0\}\\ \mathcal{I}_1(x) &:= \{i : x_i < 1\}\\ \mathcal{I}(x) &:= \{i : 0 < x_i < 1\} = \mathcal{I}_0(x) \cap \mathcal{I}_1(x).\end{aligned}$$

Note that $\mathcal{I}(x)^c = \mathcal{I}_0(x)^c \cup \mathcal{I}_1(x)^c$ indexes the active constraints at x. Let $y, z \in \Re^n$ denote the Lagrange multipliers for the constraints $e - x \ge 0$ and $x \ge 0$, respectively. For fixed y, z, the Lagrangian of (1) is defined as

$$L(x; y, z) := \frac{1}{2} x^T Q x + c^T x - z^T x - y^T (e - x).$$

With these definitions, the necessary optimality conditions for (1) are

$$0 \le x \le e \tag{2a}$$

$$\nabla_x L(x; y, z) = Qx + c - z + y = 0 \tag{2b}$$

$$y \ge 0, \qquad z \ge 0 \tag{2c}$$

$$z_i = 0, \qquad y_j = 0 \quad \forall\, i \in \mathcal{I}_0(x), \forall j \in \mathcal{I}_1(x) \tag{2d}$$

$$v^T \nabla^2_{xx} L(x; y, z) v = v^T Q v \geq 0 \quad \forall v \in V(x) \tag{2e}$$

where

$$\begin{aligned} V(x) &:= \{v : e_i^T v = 0\ \forall i \in \mathcal{I}_0(x)^c,\ -e_j^T v = 0\ \forall j \in \mathcal{I}_1(x)^c\} \\ &= \{v : v_i = 0\ \forall i \in \mathcal{I}(x)^c\} \end{aligned}$$

is the null space of the Jacobian of the active constraints. By eliminating z and employing other straightforward simplifications, we can rewrite and label (2) as

$$0 \leq x \leq e \quad \text{(primal feasibility)} \tag{3a}$$

$$Qx + c + y \geq 0, \quad y \geq 0 \quad \text{(dual feasibility)} \tag{3b}$$

$$x \circ (Qx + c + y) = 0, \qquad y \circ (e - x) = 0 \quad \text{(complementary slackness)} \tag{3c}$$

$$Q_{\mathcal{I}(x)\mathcal{I}(x)} \succeq 0 \quad \text{(local convexity)}. \tag{3d}$$

Now we give an equivalent form of the local convexity condition (3d), which does not explicitly involve knowledge of $\mathcal{I}(x)$.

Proposition 2.1 *Given x, define*

$$w := x \circ (e - x). \tag{4}$$

Then the local convexity condition (3d) *at x is equivalent to*

$$Q \circ ww^T \succeq 0. \tag{5}$$

Proof For notational convenience, we write $\mathcal{I}$ for $\mathcal{I}(x)$ and D for $\mathrm{Diag}(x)$. We first show the equivalence of (3d) and the inequality

$$(I - D)DQD(I - D) \succeq 0.$$

Assume (3d) holds. By definition, $D(I - D)$ is a diagonal matrix such that, for all $i \in \mathcal{I}^c$, the i-th diagonal entry is 0. For any v, define $\tilde{v} := D(I - D)v$. Then $\tilde{v}_i = 0$ for all $i \in \mathcal{I}^c$ and

$$v^T (I - D)DQD(I - D)v = \tilde{v}^T Q \tilde{v} = \tilde{v}_{\mathcal{I}}^T Q_{\mathcal{I}\mathcal{I}} \tilde{v}_{\mathcal{I}} \geq 0.$$

So $(I - D)DQD(I - D)$ is positive semidefinite. Conversely, assume $(I - D) \times DQD(I - D) \succeq 0$. Since $D_{\mathcal{I}\mathcal{I}} > 0$ and $[I - D]_{\mathcal{I}\mathcal{I}} > 0$, for any partial vector $\tilde{v}_{\mathcal{I}}$, there exists some v such that the full vector $\tilde{v} := D(I - D)v$ extends $\tilde{v}_{\mathcal{I}}$ and also satisfies $\tilde{v}_{\mathcal{I}^c} = 0$. So

$$\tilde{v}_{\mathcal{I}}^T Q_{\mathcal{I}\mathcal{I}} \tilde{v}_{\mathcal{I}} = \tilde{v}^T Q \tilde{v} = v^T (I - D)DQD(I - D)v \geq 0,$$

which establishes (3d).

Now, the equivalence of (3d) and $Q \circ ww^T \succeq 0$ follows from

$$(I - D)DQD(I - D) = \mathrm{Diag}(w)Q\,\mathrm{Diag}(w) = Q \circ ww^T. \qquad \square$$

It follows from Proposition 2.1 that (1) can be reformulated as the following quadratic semidefinite program, which does not depend explicitly on knowledge of the inactive constraints:

$$\min\left\{\frac{1}{2}x^T Qx + c^T x : (3a)\text{–}(3c)\ \ (4)\ (5)\right\}. \tag{6}$$

3 Semidefinite relaxations

In this section, we first present the basic semidefinite relaxation of (1) due to Shor [11]. Then we introduce semidefinite relaxations of the new formulation (6).

3.1 Shor's bounded relaxation (SDP_0)

As is standard in the SDP literature (see for example [11]), we can use the non-convex equality $X = xx^T$ to represent (1) in the equivalent form

$$\min\left\{\frac{1}{2}Q \bullet X + c^T x : 0 \le x \le e, \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0, X = xx^T\right\}.$$

By dropping the constraint $X = xx^T$, we obtain the relaxation due to Shor:

$$\min\left\{\frac{1}{2}Q \bullet X + c^T x : 0 \le x \le e, \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0\right\}. \tag{7}$$

However, the following well known fact about (7) is easy to prove:

Proposition 3.1 *If $Q \not\succeq 0$, then the optimal value of (7) is $-\infty$.*

The reason why (7) is unbounded when $Q \not\succeq 0$ is that there is too much freedom for X. We can fix the problem of unboundedness by including some valid linear constraints implied by $X = xx^T$ and $0 \le x \le e$, e.g., $\mathrm{diag}(X) \le x$ [10]. Adding $\mathrm{diag}(X) \le x$ to (7), we get a bounded relaxation for (1):

$$\min\left\{\frac{1}{2}Q \bullet X + c^T x : 0 \le x \le e, \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0, \mathrm{diag}(X) \le x\right\}. \tag{SDP_0}$$

In particular, the optimal solution set of (SDP_0) is nonempty. We consider (SDP_0) to be the smallest, simplest semidefinite relaxation of (1).

We remark that Ye [15] has derived an approximation algorithm for quadratic programming over the box $\{x : -e \le x \le e\}$, which is simply a shifted and scaled version of (1). The main tool used by Ye is the equivalent version of (SDP_0) for the case $\{x : -e \le x \le e\}$.

3.2 Relaxations (SDP_{12}) and (SDP_2) of the optimality conditions

To relax (6), we consider the matrix

$$\begin{pmatrix} 1 \\ x \\ y \\ w \end{pmatrix} \begin{pmatrix} 1 \\ x \\ y \\ w \end{pmatrix}^T = \begin{pmatrix} 1 & x^T & y^T & w^T \\ x & xx^T & xy^T & xw^T \\ y & yx^T & yy^T & yw^T \\ w & wx^T & wy^T & ww^T \end{pmatrix} \succeq 0$$

and its linearized version

$$M = \begin{pmatrix} 1 & x^T & y^T & w^T \\ x & X & M_{xy}^T & M_{xw}^T \\ y & M_{xy} & Y & M_{yw}^T \\ w & M_{xw} & M_{yw} & W \end{pmatrix} \succeq 0.$$

We can relax the quadratic constraints (3c), (4) and (5) via M. For example, consider the j-th entry of $x \circ (Qx + c + y) = 0$ from (3c), which is

$$x_j(Q_j x + c_j + y_j) = 0.$$

Relaxing it via M, we have

$$Q_j X^j + c_j x_j + [M_{xy}]_{jj} = 0.$$

So, $x \circ (Qx + c + y) = 0$ is relaxed in total as

$$\text{diag}(QX) + c \circ x + \text{diag}(M_{xy}) = 0.$$

Constraints (4) and (5) can be relaxed in a similar way. Hence, we obtain the following SDP relaxation of (6), which we call (SDP_{12}):

$$\begin{aligned}
\min \quad & \frac{1}{2} Q \bullet X + c^T x && && \text{(8a)} \\
\text{s.t.} \quad & 0 \le x \le e, \qquad \text{diag}(X) \le x && && \text{(8b)} \\
& Qx + c + y \ge 0, \quad y \ge 0 && && \text{(8c)} \\
& \text{diag}(QX) + c \circ x + \text{diag}(M_{yx}) = 0, \qquad y - \text{diag}(M_{yx}) = 0 && && \text{(8d)} \\
& w = x - \text{diag}(X) && && \text{(8e)} \\
& Q \circ W \succeq 0 && && \text{(8f)} \\
& M \succeq 0. && && \text{(8g)}
\end{aligned}$$

We point out that $\text{diag}(X) \le x$ in (8b) is not a relaxation of a particular constraint in (6). Rather, it is added to prevent (SDP_{12}) from being unbounded as with (SDP_0).

We also study a relaxed version of (SDP_{12}), which we call (SDP_2):

$$\min \quad \frac{1}{2} Q \bullet X + c^T x \qquad \text{(9a)}$$

Table 1 Comparison of the sizes of (SDP_0), (SDP_2), and (SDP_{12})

	SDP_0	SDP_2	SDP_{12}
# variables	$(n^2+3n)/2$	n^2+3n	$9(n^2+n)/2$
# linear constraints	$3n$	$4n$	$8n$
# semidefinite constraints	1	3	2
Sizes of semidefinite constraints	$n+1$	$n, n+1, n+1$	$n, 3n+1$

$$\text{s.t.} \quad 0 \le x \le e, \qquad \operatorname{diag}(X) \le x \tag{9b}$$

$$w = x - \operatorname{diag}(X) \tag{9c}$$

$$Q \circ W \succeq 0 \tag{9d}$$

$$\begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0 \tag{9e}$$

$$\begin{pmatrix} 1 & w^T \\ w & W \end{pmatrix} \succeq 0. \tag{9f}$$

In essence, (SDP_2) maintains the minimal set of constraints from (SDP_{12}), which still explicitly relax the second-order optimality conditions. We are particularly interested in (SDP_2) because it captures the second-order optimality information (a main focus of this paper) and because its dimension is significantly lower than that of (SDP_{12}). Table 1 compares the sizes of the three SDPs.

4 Equivalence of the SDP relaxations

4.1 Equivalence of (SDP_0) and (SDP_2)

In this section, we establish the equivalence of (SDP_0) and (SDP_2). We will use the following generic result:

Lemma 4.1 *Consider two related optimization problems*:

$$\text{(A)} \quad \min\{f(x) : x \in P\}$$

$$\text{(B)} \quad \min\{f(x) : x \in P,\ y \in R(x)\},$$

where P represents the feasible set for (A) *and $R(x)$ defines the set of constraints* (*related to x*) *that y must satisfy. Let x^* be an optimal solution of* (A) *and suppose $R(x^*) \neq \emptyset$. Then any (x^*, y) with $y \in R(x^*)$ is optimal for* (B). *Therefore, the optimal value of* (B) *equals that of* (A).

Since the feasible set of (B) is more restrictive than that of (A), the optimal value of (B) is greater than or equal to that of (A). The inclusion $y \in R(x^*)$ and the fact that the objective value does not depend on y together imply that (A) and (B) achieve the same optimal value.

Our first step is to prove the following property of (SDP$_0$) at optimality.

Lemma 4.2 *Let* (x^*, X^*) *be an optimal solution of* (SDP$_0$)*, and define* $\mathcal{J} := \{i : X^*_{ii} < x^*_i\}$*. Then* $Q_{\mathcal{J}\mathcal{J}} \succeq 0$.

Proof The argument is based on examining an optimal solution for the dual of (SDP$_0$). It can be easily verified that the dual is

$$\begin{array}{ll} \max & \lambda - e^T y \\ \text{s.t.} & S = \frac{1}{2}\begin{pmatrix} -\lambda & (c+y-z-v)^T \\ c+y-z-v & Q + 2\,\mathrm{Diag}(v) \end{pmatrix} \succeq 0 \\ & y, z, v \geq 0, \end{array}$$

where y, z, v, S, and λ are, respectively, the multipliers for $e - x \geq 0$, $x \geq 0$, $x - \mathrm{diag}(X) \geq 0$, $(1, x^T; x, X) \succeq 0$, and the constraint associated with fixing the top-left entry of $(1, x^T; x, X)$ to 1.

Note that both (SDP$_0$) and its dual have nonempty interior. Specifically, the point

$$(x, X) = \left(\frac{1}{2}e, \frac{1}{4}ee^T + \varepsilon I\right)$$

is interior feasible for (SDP$_0$) for all $\varepsilon \in (0, 1/4)$. In addition, taking v sufficiently positive, λ sufficiently negative, and y, z positive such that $y - z - v$ has sufficiently small norm yields an interior solution of the dual with $S \succ 0$. Because both problems have interiors, strong duality holds. For the remainder of the proof, we let (x, X) and (λ, y, z, v, S) denote specific optimal solutions of the primal and dual.

Due to complementary slackness, $(x - \mathrm{diag}(X)) \circ v = 0$. So $v_{\mathcal{J}} = 0$, and it follows from $S \succeq 0$ that

$$\left[Q + 2\,\mathrm{Diag}(v)\right]_{\mathcal{J}\mathcal{J}} = Q_{\mathcal{J}\mathcal{J}} \succeq 0. \qquad \square$$

Theorem 4.3 *Let* (x^*, X^*) *be an optimal solution of* (SDP$_0$)*, and define* $w^* := x^* - \mathrm{diag}(X^*)$ *and* $W^* := w^*(w^*)^T$*. Then* (x^*, X^*, w^*, W^*) *is an optimal solution of* (SDP$_2$) *with the same optimal value.*

Proof For notational convenience, we drop the $*$ superscripts. By Lemma 4.1, we need only prove that (x, X, w, W) is feasible for (SDP$_2$), and to do so requires the verification of (9d) since all the other constraints of (SDP$_2$) are satisfied by construction.

Let $\mathcal{J}$ be defined as in Lemma 4.2. Then $w_{\mathcal{J}} > 0$, $w_{\mathcal{J}^c} = 0$, and $Q_{\mathcal{J}\mathcal{J}} \succeq 0$. Note that $[Q \circ W]_{ij} = 0$ if $i \in \mathcal{J}^c$ or $j \in \mathcal{J}^c$. So $Q \circ W = Q \circ ww^T \succeq 0$ is equivalent to $Q_{\mathcal{J}\mathcal{J}} \circ (w_{\mathcal{J}} w_{\mathcal{J}}^T) = \mathrm{Diag}(w_{\mathcal{J}})\, Q_{\mathcal{J}\mathcal{J}}\, \mathrm{Diag}(w_{\mathcal{J}}) \succeq 0$, which is true because $Q_{\mathcal{J}\mathcal{J}} \succeq 0$ and $w_{\mathcal{J}} > 0$. This proves (9d) and hence the theorem. $\square$

4.2 Equivalence of (SDP$_0$) and (SDP$_{12}$)

In the last subsection, we have proved that (SDP$_2$) is equivalent to (SDP$_0$). In this subsection, we show that even (SDP$_{12}$) is equivalent to (SDP$_0$). We start by proving

some properties of (SDP_0), which will facilitate the proof of equivalence later in Sect. 4.2.2 but are also of independent interest.

4.2.1 Additional properties of (SDP_0)

We will show that every optimal solution (x^*, X^*) of (SDP_0) satisfies the following two inequalities:

$$\operatorname{diag}(QX) + c \circ x \leq 0 \tag{10a}$$

$$Qx + c - (\operatorname{diag}(QX) + c \circ x) \geq 0. \tag{10b}$$

In other words, (10a) and (10b) are redundant for (SDP_0) in the sense that enforcing these inequalities does not change the optimal solution set. This knowledge will help us establish the equivalence between (SDP_{12}) and (SDP_0) in the next subsection.

To prove (10), we start by examining paths of solutions in the feasible set of (SDP_0). Given any feasible (x, X), consider two paths of emanating from (x, X) and depending on a specified index i. Each path is parameterized by $\alpha \geq 0$. We define $(x_1(\alpha), X_1(\alpha))$ and $(x_2(\alpha), X_2(\alpha))$ by

$$\begin{aligned}
x_1(\alpha) &:= x - \alpha x_i e_i \\
X_1(\alpha) &:= X - \alpha e_i (X^i)^T - \alpha X^i e_i^T + \alpha^2 X_{ii} e_i e_i^T \\
x_2(\alpha) &:= x + \alpha e_i \\
X_2(\alpha) &:= X + \alpha e_i x^T + \alpha x e_i^T + \alpha^2 e_i e_i^T.
\end{aligned}$$

Furthermore, for any $\beta \in [0, 1]$, we consider a third path $(x(\alpha), X(\alpha))$, which is a convex combination of $(x_1(\alpha), X_1(\alpha))$ and $(x_2(\alpha), X_2(\alpha))$:

$$\begin{aligned}
x(\alpha) &:= \beta x_1(\alpha) + (1 - \beta) x_2(\alpha) \\
X(\alpha) &:= \beta X_1(\alpha) + (1 - \beta) X_2(\alpha).
\end{aligned}$$

Our intent is to examine conditions on α and β such that $(x(\alpha), X(\alpha))$ is feasible for (SDP_0). We will also be interested in the objective value at $(x(\alpha), X(\alpha))$:

$$f(\alpha) := \frac{1}{2} Q \bullet X(\alpha) + c^T x(\alpha).$$

Proposition 4.4 *Let an index i be specified. Given (x, X) feasible for (SDP_0) and $\beta \in [0, 1]$, $(x(\alpha), X(\alpha))$ satisfies the following properties*:

(i) $x(\alpha)$ *differs from* x *only in the* i*-th entry, and* $x(\alpha)_i = x_i + \alpha(1 - \beta - \beta x_i)$;
(ii) $\operatorname{diag}(X(\alpha)) - x(\alpha)$ *differs from* $\operatorname{diag}(X) - x$ *only in the* i*-th entry, and*

$$\begin{aligned}
[\operatorname{diag}(X(\alpha)) - x(\alpha)]_i = (1 - \alpha\beta)(X_{ii} - x_i) + \alpha \big[&\alpha(\beta X_{ii} + 1 - \beta) \\
&+ 2(1 - \beta) x_i - (\beta X_{ii} + 1 - \beta) \big];
\end{aligned}$$

(iii) $\begin{pmatrix} 1 & x(\alpha)^T \\ x(\alpha) & X(\alpha) \end{pmatrix}$ *is positive semidefinite.*

Moreover,

$$f'(0) = \beta(-(Q^i)^T X^i - c_i x_i) + (1-\beta)((Q^i)^T x + c_i).$$

Proof It is straightforward to verify the formulas for $x(\alpha)_i$, $[\mathrm{diag}(X(\alpha)) - x(\alpha)]_i$, and $f'(0)$. Now we show that (iii) holds. From the definition of $(x_1(\alpha), X_1(\alpha))$, we see

$$\begin{pmatrix} 1 & x_1(\alpha)^T \\ x_1(\alpha) & X_1(\alpha) \end{pmatrix} = \begin{pmatrix} 1 & 0^T \\ 0 & I - \alpha e_i e_i^T \end{pmatrix} \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \begin{pmatrix} 1 & 0^T \\ 0 & I - \alpha e_i e_i^T \end{pmatrix} \succeq 0.$$

Furthermore, by using the Schur complement theorem twice, we have

$$\begin{aligned} \begin{pmatrix} 1 & x_2(\alpha)^T \\ x_2(\alpha) & X_2(\alpha) \end{pmatrix} \succeq 0 \quad &\Longleftrightarrow \quad \begin{pmatrix} 1 & (x+\alpha e_i)^T \\ x+\alpha e_i & X + \alpha e_i x^T + \alpha x e_i^T + \alpha^2 e_i e_i^T \end{pmatrix} \succeq 0 \\ &\Longleftrightarrow \quad (X + \alpha e_i x^T + \alpha x e_i^T + \alpha^2 e_i e_i^T) \\ &\qquad\quad - (x+\alpha e_i)(x+\alpha e_i)^T \succeq 0 \\ &\Longleftrightarrow \quad X - x x^T \succeq 0 \\ &\Longleftrightarrow \quad \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0, \end{aligned}$$

which is true due to the feasibility of (x, X). Therefore, (iii) follows because $(1, x(\alpha)^T; x(\alpha), X)$ is a convex combination of positive semidefinite matrices. □

The following corollaries are easy to establish.

Corollary 4.5 *Let (x, X) be feasible for (SDP$_0$), and let $\beta = 1$. For a specified index i, $(x(\alpha), X(\alpha))$ is feasible for all $\alpha \in [0, 1]$, and $f'(0) = -(Q^i)^T X^i - c_i x_i$.*

Corollary 4.6 *Let (x, X) be feasible for (SDP$_0$) with $x_i < 1$, which guarantees $1 + X_{ii} - 2x_i > 0$. Also let $\beta = 1/2$. For a specified index i, $(x(\alpha), X(\alpha))$ is feasible for all $\alpha \in [0, \frac{1+X_{ii}-2x_i}{1+X_{ii}}]$, and $f'(0) = \frac{1}{2}[Qx + c - (\mathrm{diag}(QX) + c \circ x)]_i$.*

We are now ready to prove that every optimal solution (x^*, X^*) of (SDP$_0$) satisfies (10). We need just one additional lemma, whose proof is a straightforward adaptation of the proof of Proposition 3.2 in [2]:

Lemma 4.7 *Let (x, X) be feasible for (SDP$_0$). Then $x_i = 1$ implies $X^i = x$.*

Theorem 4.8 *Let (x^*, X^*) be optimal for (SDP$_0$). Then (x^*, X^*) satisfies the inequalities (10).*

Proof We prove the following equivalent statement: suppose feasible (x, X) does not satisfy (10); then (x, X) is not optimal. We break the condition of not satisfying (10) into three subcases: (i) $[\text{diag}(QX) + c \circ x]_i > 0$ for some i; (ii) $[Qx + c - (\text{diag}(QX) + c \circ x)]_i < 0$ for some i and $x_i < 1$; and (iii) $[Qx + c - (\text{diag}(QX) + c \circ x)]_i < 0$ for some i and $x_i = 1$.

In case (i), Corollary 4.5 implies the existence of a feasible path emanating from (x, X) with decreasing objective. Hence, (x, X) is not optimal. Case (ii) follows similarly from Corollary 4.6.

Finally, we show that case (iii) actually cannot occur. Suppose $x_i = 1$. Then by Lemma 4.7,

$$\left[Qx + c - \text{diag}(QX) - c \circ x\right]_i = (Q^i)^T(x - X^i) + c_i(1 - x_i) = 0,$$

which is incompatible with (iii). □

4.2.2 *Proof of equivalence*

Note that (SDP_{12}) is more constrained than (SDP_0). By Lemma 4.1, it suffices to construct a feasible solution to (SDP_{12}) based on (x^*, X^*) to establish the equivalence of (SDP_0) and (SDP_{12}).

We construct the solution for (SDP_{12}) by defining

$$y := -(\text{diag}(QX^*) + c \circ x^*) \tag{11a}$$

$$Y := yy^T + \epsilon I, \tag{11b}$$

$$w := x^* - \text{diag}(X^*), \qquad W := ww^T \tag{11c}$$

$$M_{xw} := wx^{*T}, \qquad M_{yw} := wy^T \tag{11d}$$

$$M := \begin{pmatrix} 1 & x^{*T} & y^T & w^T \\ x^* & X^* & M_{xy}{}^T & M_{xw}{}^T \\ y & M_{xy} & Y & M_{yw}{}^T \\ w & M_{xw} & M_{yw} & W \end{pmatrix}, \tag{11e}$$

where $\epsilon > 0$ is a sufficiently large constant (more details below). Note that we have not specified M_{xy} yet; we will do so below.

We must check that the solution specified is indeed feasible for (SDP_{12}), which requires checking (8b)–(8g). Obviously, (8b) is satisfied by (x^*, X^*). It follows from (10a) and (10b) that (8c) is satisfied by (x^*, X^*, y). The constraint (8e) is satisfied by definition, and Theorem 4.3 illustrates that (8f) is satisfied. It remains to show that (8d) and (8g) hold. These will depend on the choice of ϵ and M_{xy}.

To prove (8d) and (8g), we exhibit an M_{xy} such that $\text{diag}(M_{xy}) = y$ and $M \succeq 0$. We first require the following lemma and proposition:

Lemma 4.9 *For an optimal solution* (x^*, X^*) *of* (SDP_0), *if* $X^{i*} = x_i^* x^*$, *then* $y_i(1 - x_i^*) = 0$, *where* y *is defined as in* (11a).

Proof We drop the superscripts $*$ to simplify notation. If $x_i = 1$, then $y_i(1 - x_i) = 0$, and if $x_i = 0$, then $y_i = -((Q^i)^T X^i + c_i x_i) = -x_i((Q^i)^T x + c_i) = 0$. If $0 < x_i < 1$, we show $y_i = 0$. Let $g_i := (Q^i)^T x + c_i$. We know $y_i = -x_i g_i \geq 0$, and so $g_i \leq 0$. On the other hand, by (10b), $g_i + y_i = (1 - x_i)g_i \geq 0$, and so $g_i \geq 0$. Hence, $g_i = 0$, which ensures $y_i = 0$. □

Proposition 4.10 *Let* (x^*, X^*) *be an optimal solution of* (SDP$_0$), *and define* y *as in* (11a). *Then there exists* $A \in \Re^{n \times n}$ *such that*

$$\operatorname{diag}(A(X^* - x^* x^{*T})) = y \circ (e - x^*).$$

Proof We drop the superscripts $*$ to simplify notation. We show equivalently that there exists a solution A to the system of equations

$$A_i(X - xx^T)^i = y_i(1 - x_i) \quad \forall i = 1, \ldots, n.$$

Note that the n equations just listed are separable; so we consider each i separately. If $(X - xx^T)^i \neq 0$, it is obvious that there exists a solution A_i; just take A_i equal to

$$\frac{y_i(1 - x_i)}{\|(X - xx^T)^i\|^2}\left[(X - xx^T)^i\right]^T.$$

On the other hand, if $(X - xx^T)^i = 0$, i.e., $X^i = x_i x$, then we know by Lemma 4.9 that $y_i(1 - x_i) = 0$ and thus A_i can be any vector. □

We define

$$M_{xy} := yx^{*T} + A\big(X^* - x^*(x^*)^T\big),$$

where A is any matrix as in Proposition 4.10. Then $\operatorname{diag}(M_{xy}) = y \circ x^* + y \circ (e - x^*) = y$, which ensures that (8d) is satisfied. Finally, it remains to show that (8g) holds, i.e., $M \succeq 0$, for this choice of M_{xy}.

In the following, we drop the superscripts $*$ to simplify notation. Note that

$$M = \begin{pmatrix} 1 \\ x \\ y \\ w \end{pmatrix}\begin{pmatrix} 1 \\ x \\ y \\ w \end{pmatrix}^T + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & X - xx^T & (M_{xy} - yx^T)^T & 0 \\ 0 & M_{xy} - yx^T & \epsilon I & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

and so it suffices to show

$$\begin{pmatrix} X - xx^T & (M_{xy} - yx^T)^T \\ M_{xy} - yx^T & \epsilon I \end{pmatrix} = \begin{pmatrix} X - xx^T & (A(X - xx^T))^T \\ A(X - xx^T) & \epsilon I \end{pmatrix} \succeq 0.$$

By the Schur complement theorem, this holds if and only if

$$(X - xx^T) - \epsilon^{-1}(X - xx^T)A^T A(X - xx^T) \succeq 0. \tag{12}$$

Consider the following straightforward lemma:

Lemma 4.11 *Suppose* $R, S \succeq 0$. *Then there exists* $\delta > 0$ *small enough such that* $R - \delta S \succeq 0$ *if and only if* $\text{Null}(R) \subseteq \text{Null}(S)$.

Because the null space of $X - xx^T$ is contained in the null space of $(X - xx^T)A^T A(X - xx^T)$, the lemma implies the existence of $\epsilon > 0$ large enough so that (12) holds. Taking such ϵ, we conclude that (8g) is satisfied.

Overall, we have shown that definition (11)—along with the definitions of M_{xy} and ϵ—is feasible for (SDP_{12}), which means (SDP_0) and (SDP_{12}) are equivalent by Lemma 4.1.

5 Comparison of SDP relaxations within branch-and-bound

In Sect. 4, we have shown that the three SDP relaxations (SDP_0), (SDP_2) and (SDP_{12}) are equivalent. In this section, we empirically compare these relaxations in the context of branch-and-bound for solving (1) globally, where the relaxations can have different effects on subdivided boxes. Our experiments on randomly generated problems illustrate the strength of the bounds produced by (SDP_2) and (SDP_{12}) over those produced by (SDP_0) in this context. Our approach will be to focus on the comparison of (SDP_0) and (SDP_2), while briefly commenting on (SDP_{12}) towards the end.

We would like to point out that our intention here is *not* to develop a branch-and-bound method for (1), which outperforms all other techniques. Rather, our primary goal is comparing (SDP_0), (SDP_{12}), and (SDP_2) in order to gauge the effect of incorporating optimality conditions (particularly the second-order conditions) into SDP relaxations for (1).

5.1 Branch-and-bound for box QP

The branch-and-bound algorithm we consider recursively subdivides the entire box $\{x \in \Re^n : 0 \leq x \leq e\}$ into smaller and smaller boxes and solves an appropriately tailored SDP relaxation—either (SDP_0) or (SDP_2)—on these smaller boxes. Lower bounds obtained from these relaxations are compared with a global upper bound to fathom as many small boxes as possible from consideration. When fathoming is not possible for a specific small box, that box is further subdivided. Moreover, the global upper bound is improved (whenever possible) throughout the course of the algorithm.

We measure the performance of the branch-and-bound algorithm in two ways: the total number of nodes in the branch-and-bound tree and the total time to complete the entire branch-and-bound process. The number of nodes is affected by the quality of the lower bounds. Our main comparison will be the lower bound calculations based on either (SDP_0) or (SDP_2). Since the branching strategy also affects the number of nodes, we will investigate two different branching strategies as well.

Before discussing our algorithm design choices below, we first present the SDP relaxations on the small boxes, which are modified appropriately from the corresponding versions on the entire box. Suppose the current node of the branch-and-bound tree corresponds to the box

$$\{x \in \Re^n : l \leq x \leq u\}.$$

Then the SDP relaxation that corresponds to (SDP_0) is

$$\begin{aligned} \min \quad & \frac{1}{2} Q \bullet X + c^T x && \text{(13a)} \\ \text{s.t.} \quad & l \le x \le u && \text{(13b)} \\ & \operatorname{diag}(X) - (l+u) \circ x + l \circ u \le 0 && \text{(13c)} \\ & \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0. && \text{(13d)} \end{aligned}$$

The constraint $\operatorname{diag}(X) - (l+u) \circ x + l \circ u \le 0$ is obtained by relaxing the valid inequality

$$x \circ x - (l+u) \circ x + l \circ u = (x-l) \circ (x-u) \le 0.$$

Note that, when $l = 0$ and $u = e$, this constraint is just $\operatorname{diag}(X) \le x$. So this inequality plays the role of bounding the diagonal of X on the smaller boxes.

To derive the relaxation corresponding to (SDP_2) on the smaller box, we first introduce the following notation depending on the bounds (l, u):

$$\begin{aligned} B_1 &:= \{i : \ell_i = 0 \text{ and } u_i = 1\} \\ B_2 &:= \{i : \ell_i = 0 \text{ and } u_i < 1\} \\ B_3 &:= \{i : \ell_i > 0 \text{ and } u_i = 1\} \\ B_4 &:= \{i : \ell_i > 0 \text{ and } u_i < 1\}. \end{aligned}$$

Note that $B_1 \cup B_2 \cup B_3 \cup B_4 = \{1, \ldots, n\}$. Now consider the following specialization of Proposition 2.1:

Proposition 5.1 *Given x satisfying $0 \le l \le x \le u \le e$, define $w \in \Re^n$ by*

$$\begin{aligned} w_{B_1} &:= x_{B_1} \circ (e_{B_1} - x_{B_1}) \\ w_{B_2} &:= x_{B_2} \\ w_{B_3} &:= e_{B_3} - x_{B_3} \\ w_{B_4} &:= e_{B_4}. \end{aligned} \qquad \text{(14)}$$

Then the local convexity condition (3d) at x is equivalent to

$$Q \circ ww^T \succeq 0.$$

Proof Recall that the proof of Proposition 2.1 established that the second-order condition for (1) at any $0 \le x \le e$ is equivalent to

$$(I - D)DQD(I - D) \succeq 0, \qquad \text{(15)}$$

where I is the identity matrix and $D = \operatorname{Diag}(x)$. For fixed i, if $l_i > 0$, then we know the i-th diagonal of D is strictly positive, and so we can replace $D_{ii} = x_i$ with $D_{ii} = 1$

in the inner two D's of (15) without affecting semidefiniteness. Similarly, if $u_i < 1$, then we know that the i-th diagonal entry of $I - D$ is positive and hence can be replaced by 1 without affecting semidefiniteness in (15). If $l_i > 0$ and $u_i < 1$, then both replacements can be made. Using arguments similar to the proof of Proposition 2.1, the resulting matrix $(I - D)DQD(I - D)$ after replacements is equal to $Q \circ ww^T$, where w is given by (14). □

With Proposition 5.1 in hand, the relaxation corresponding to (SDP$_2$) on the smaller box is

$$\begin{aligned}
\min \quad & \frac{1}{2} Q \bullet X + c^T x && \text{(16a)}\\
\text{s.t.} \quad & l \le x \le u && \text{(16b)}\\
& \operatorname{diag}(X) - (l+u) \circ x + l \circ u \le 0 && \text{(16c)}\\
& w_{B_1} = x_{B_1} - \operatorname{diag}(X_{B_1 B_1}) && \text{(16d)}\\
& w_{B_2} = x_{B_2} && \text{(16e)}\\
& w_{B_3} = e_{B_3} - x_{B_3} && \text{(16f)}\\
& w_{B_4} = e_{B_4} && \text{(16g)}\\
& W_{B_2 B_2} = X_{B_2 B_2} && \text{(16h)}\\
& W_{B_2 B_3} = x_{B_2} e_{B_3}^T - X_{B_2 B_3} && \text{(16i)}\\
& W_{B_3 B_3} = e_{B_3} e_{B_3}^T - x_{B_3} e_{B_3}^T - e_{B_3} x_{B_3}^T + X_{B_3 B_3} && \text{(16j)}\\
& W^{B_4} = w e_{B_4}^T && \text{(16k)}\\
& Q \circ W \succeq 0, \qquad \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0, \qquad \begin{pmatrix} 1 & w^T \\ w & W \end{pmatrix} \succeq 0. && \text{(16l)}
\end{aligned}$$

Note that the constraints (16d)–(16g) give rise to new constraints (16h)–(16k) between W and x, w, and X.

We now address the major design choices for the branch-and-bound algorithm:

- **Bounding**. We will compare strategies involving two types of lower bounds: those given by (13) and those given by (16). A single run of the branch-and-bound algorithm on a single instance will employ one or the other—or a combination of both, i.e., first (13) and then a switch to (16) (more details are given in the next subsection).

  For the global upper bound, we experimented with two ways to improve it at each node of the tree: (a) locally solve the small box QP at each node via MATLAB's `quadprog` function; (b) simply take the objective value $\frac{1}{2}(x^*)^T Q x^* + c^T x^*$ corresponding to the optimal x^* obtained from the lower bound calculation. Option (b) was a bit quicker, but at each node, the time for either (a) or (b) was dominated by the lower bound calculation. On the other hand, compared to (b), (a) generally resulted in fewer nodes in the tree and thus saved time overall. So we use (a) throughout the computations.

- **Branching**. We consider two branching strategies.

  The first branching strategy—which we will call *simple*—is the standard "bisection via longest edge" (see, for example, [5]). Consider the small box $\{x \in \Re^n : l \le x \le u\}$, which has been selected for branching. We select the longest edge of this box to branch on. More specifically, we choose the index i such that $u_i - l_i$ is the largest among all dimensions. If there is a tie, the smallest such index is chosen. By applying this strategy, we subdivide the box into two smaller boxes:

$$\left\{x \in \Re^n : l \le x \le u - \frac{1}{2}(u_i - l_i)e_i\right\}$$

$$\left\{x \in \Re^n : l + \frac{1}{2}(u_i - l_i)e_i \le x \le u\right\}.$$

  The second branching strategy—which we will call *advanced*—is more sophisticated and involves two ingredients:

  – It is well known that SDP relaxations such as (13) and (16) enforce a fairly weak approximation of $X = xx^T$ in the interior of $[l, u]$ and a fairly strong one near the boundary. Hence, if $\bar{x}$ is an optimal solution returned by (13) or (16) and if $\bar{x}_i \in (l_i, u_i)$ for some index i, then branching on i via the intervals $[l_i, \bar{x}_i]$ and $[\bar{x}_i, u_i]$ results in two relaxations with $\bar{x}$ on the boundary, thus strengthening the approximation of $X = xx^T$ precisely where needed and increasing the chances that $\bar{x}$ will be cut off in the relaxations. A similar logic guides the "most fractional" branching rule of integer programming.
  – For theoretical validity of the branch-and-bound algorithm, the branching strategy must subdivide all boxes in such a way that the longest edge of all unfathomed boxes tends to 0 in the limit. This is indeed the most basic property of "bisection via longest edge."

  So we design the second branching strategy as a combination of "most fractional" and "bisection via longest edge." For a given feasible solution $l \le \bar{x} \le u$, our strategy calculates, for each $i = 1, \ldots, n$, the values

$$\alpha_i = \frac{u_i - \bar{x}_i}{u_i - l_i} \cdot \frac{\bar{x}_i - l_i}{u_i - l_i} \in [0, 1/4]$$

$$\beta_i = u_i - l_i \in [0, 1]$$

  and then selects for branching the i such that $\alpha_i \beta_i$ is maximum. A large α_i favors "most fractional," while a large β_i favors "longest edge." In particular, if all $u_i - l_i$ are equal, then the strategy reduces to selecting the most fractional variable, whereas if one edge is significantly longer than the others, it will necessarily be selected for branching.

  By applying this strategy, the resulting two small boxes are as follows:

$$\left\{x \in \Re^n : l \le x \le u - (u_i - \bar{x}_i)e_i\right\}$$

$$\left\{x \in \Re^n : l + (\bar{x}_i - l_i)e_i \le x \le u\right\}.$$

- **Node selection**. We use a best-bound (breadth-first) strategy for selecting the next node to solve in the branch-and-bound tree.

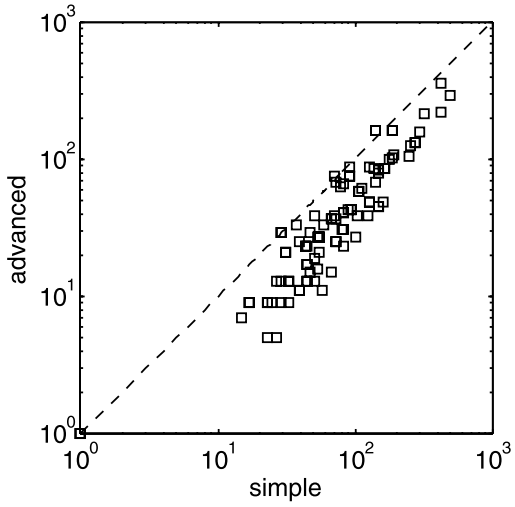

Fig. 1 Number of nodes required under test scenarios (i) and (ii). This demonstrates that the *advanced* branching strategy reduces the number of nodes significantly compared to the *simple* branching strategy

- **Fathoming tolerance**. A relative optimality tolerance is used for fathoming. For a given tolerance *tol*, a node with lower bound L is fathomed if $(U - L)/\max\{1, \frac{1}{2}(|U| + |L|)\} < tol$, where U is the current global upper bound. In our experiments, we set $tol = 10^{-3}$.

5.2 Implementation and results

For $n = 20$, we generated 100 instances of random data (Q, c) (entries uniform in $[-1, 1]$, which ensured $Q \not\succeq 0$ in all cases) and solved these instances using the branch-and-bound scheme outlined above. In particular, all instances were solved three times, each time with a different choice of lower bound calculation and branching strategy. The three choices were:

(i) lower bounds by (SDP_0) and simple branching strategy;
(ii) lower bounds by (SDP_0) and advanced branching strategy;
(iii) lower bounds by (SDP_2) and advanced branching strategy.

We will refer to these three choices as *test scenarios*. The goals of analyzing these three particular test scenarios are:

- to compare the two branching strategies via scenarios (i) and (ii) (see Fig. 1);
- to compare (SDP_0) and (SDP_2) via scenarios (ii) and (iii) (see Fig. 2).

The algorithm was coded in MATLAB (version 7.3, release 2006b) and all SDP relaxations were setup and solved using YALMIP [6] and SeDuMi (version 1.1) [12]. All computations were performed on an Intel Pentium D Processor running at 3.2 GHz with 2048 KB cache and 4 GB RAM under the Linux operating system.

Figure 1 contains a log–log plot depicting the number of nodes required by all instances in test scenarios (i) and (ii). For each of the 100 problem instances, a single point is plotted with its x-coordinate equal to the number of nodes under scenario (i) and its y-coordinate equal to the number of nodes under scenario (ii). Also depicted is

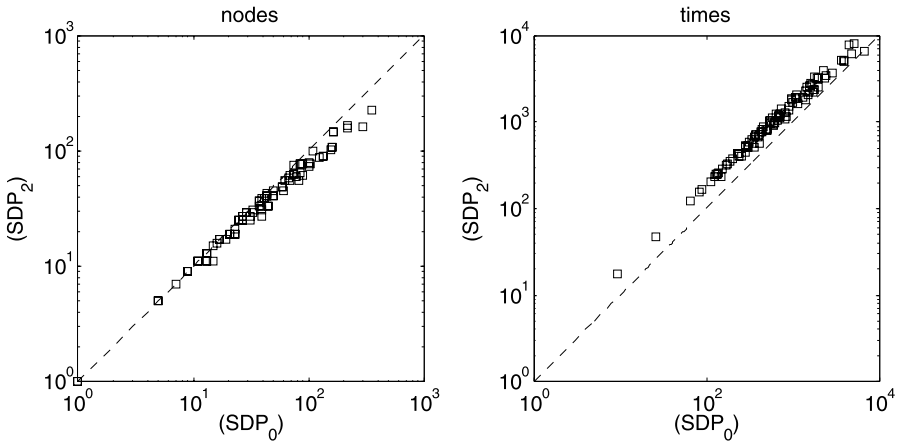

Fig. 2 Number of nodes and CPU times (seconds) required under test scenarios (ii) and (iii). This demonstrates that (SDP_2) results in fewer nodes compared to (SDP_0). However, the overall CPU time incurred by (SDP_2) is greater

the "$y = x$" dotted line, which divides the plot into two regions. In particular, a point plotted in the lower-right region indicates an instance that required fewer nodes under the advanced branching strategy of scenario (ii). Similar to Fig. 1, Fig. 2 compares the number of nodes required under test scenarios (ii) and (iii). Also depicted in a separate plot are the CPU times (in seconds). Both plots contain the $y = x$ dotted line for reference.

Our key interpretations of the figures are as follows:

- Figure 1: Since nearly all points are plotted in the lower-right region, the advanced branching strategy is clearly better than the simple branching strategy in terms of number of nodes. In particular, the number of nodes was reduced by more than 44% on average by using the advanced branching strategy. Furthermore, since (SDP_0) was used as the relaxation for lower bounds in both scenarios, the time per node for each instance was essentially the same, so that the reduction in nodes resulted in a real time reduction of about 44% as well.
- Figure 2: In all runs, the number of nodes required by (SDP_2) is no more than the number required by (SDP_0), which indicates that (SDP_2) provides stronger lower bounds than (SDP_0). In particular, on average the number of nodes required by (SDP_2) is 21% less than that of (SDP_0). However, the overall CPU times required for the entire branch-and-bound process are higher using (SDP_2). So (SDP_0) is the overall winner.

In light of Fig. 2, we wondered if some intelligent combination of (SDP_0) and (SDP_2) during the branch-and-bound procedure might perform better in terms of overall CPU time than using (SDP_0) only. Hopefully, we could reduce the number of nodes significantly—a benefit of (SDP_2)—while keeping the time small—a benefit of (SDP_0). We devised a strategy, which employs (SDP_0) early in the branch-and-bound tree, and then switches to (SDP_2) later in the tree when its stronger bounds may be useful for fathoming. Specifically, our strategy is the following:

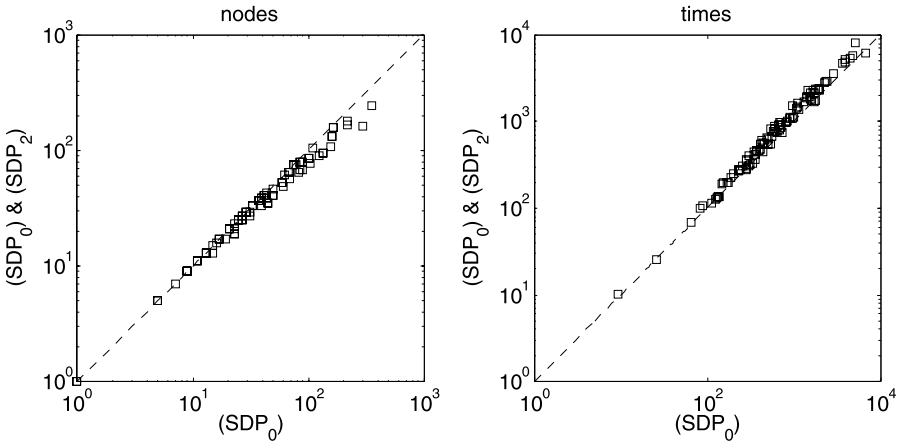

Fig. 3 Number of nodes and CPU times (seconds) required under test scenarios (ii) and (iv). Compared to scenario (iii) in Fig. 2, less time is required by scenario (iv), but still scenario (iv) requires more time than scenario (ii)

At the beginning of branch-and-bound, all lower bounds are calculated with (SDP_0) by default. At each node after the solution of (SDP_0), the optimal solution (x^*, X^*) is extracted and used to construct $W^*_{B_2B_2}$ and $W^*_{B_3B_3}$ according to (16h) and (16j), respectively. If either $Q_{B_2B_2} \circ W^*_{B_2B_2}$ or $Q_{B_3B_3} \circ W^*_{B_3B_3}$ are not positive semidefinite, then it follows that (x^*, X^*) cannot be part of a feasible solution for (SDP_2) at that node. In other words, solving (SDP_2) will cut off (x^*, X^*). In these cases, we flag all future descendants of the current node and calculate the lower bound via (SDP_2) for those descendants.

This combination of (SDP_0) was implemented as a new test scenario:

(iv) lower bounds by the above combination of (SDP_0) and (SDP_2) and advanced branching strategy (see Fig. 3).

For scenario (iv), on average, (SDP_0) is solved for 64% of the nodes, while (SDP_2) is solved for the remaining 36%. Compared to test scenario (ii), test scenario (iv) required 15% fewer nodes on average, which is again a testament to the strength of (SDP_2). In addition, the points in the time plot of Fig. 3 are shifted closer to the "$y = x$" line compared to the time plot of Fig. 2, which is an indication of less time used than in scenario (iii). (Keep in mind that both Figs. 2 and 3 share test scenario (ii) as the basis of comparison.) So our strategy of combining (SDP_0) and (SDP_2) was successful in reducing the number of nodes compared to (ii) and reducing the times compared to (iii). In fact, in the time plot, there were actually 6 instances below the "$y = x$" line, indicating that (iv) used less time than (ii) in these cases. However, on average the CPU times for scenario (iv) were still more than (ii), indicating that our strategy was not fully successful as hoped.

5.3 An additional test

For completeness, we compared (SDP_{12}) with (SDP_0) and (SDP_2) in the context of branch-and-bound. (SDP_{12}) on a smaller box $\{x \in \Re^n : l \leq x \leq u\}$ has all the constraints of (16) as well as the first order constraints (8c) and (8d). In addition, for a particular index i, if $x_i < 1$, then we fix $y_i = 1$; if $x_i > 0$, then we fix $[Qx + c + y]_i = 0$. Both of these rules are based on the complementary slackness condition (3c). We conducted the tests for the same 100 problems of size $n = 20$ above with the advanced branching strategy. The results were as follows: (SDP_{12}) required the fewest nodes among all three relaxations but required the longest times. In fact, by using (SDP_{12}) on average the number of nodes is reduced by 65% compared with (SDP_0) and 56% compared with (SDP_2).

6 Conclusion

In this paper, we have introduced new semidefinite relaxations of box-constrained quadratic programming: (SDP_{12}) and (SDP_2). (SDP_{12}) is based on relaxing both the first- and second-order necessary optimality conditions; (SDP_2) is similar except that it only incorporates second-order information. (SDP_2) has been our main focus since the first-order conditions have been studied in previous papers. We have compared these two relaxations with a basic semidefinite relaxation (SDP_0) and established the theoretical result that all three relaxations achieve the same optimal value.

Relaxing the standard second-order necessary optimality conditions is one of the main theoretical ideas of this paper. This task is non-trivial since it implicitly involves knowledge of the active/inactive constraint set at a general point. In the future, it may be possible to extend this technique to other problems, e.g., quadratic programming over the simplex, leading to stronger SDP relaxations in other contexts.

We have also empirically compared (SDP_0) and (SDP_2) in the context of branch-and-bound and demonstrated that (SDP_2) on subdivided boxes is significantly stronger, which indicates that the incorporation of second-order information in SDP relaxations can help globally solve Box QP. In particular, fewer branch-and-bound nodes are required when (SDP_2) is employed instead of (SDP_0), but overall, branch-and-bound with (SDP_2) requires a larger execution time. Future advances in SDP software may allow (SDP_2) to be solved faster, so that the benefits of its node reduction may also be reflected in overall CPU times.

Acknowledgements The authors would like to thank two anonymous referees for many helpful suggestions that have improved this paper immensely.

References

1. Anstreicher, K.M.: Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. J. Global Optim. **2**, 471–484 (2009)
2. Burer, S., Vandenbussche, D.: Globally solving box-constrained nonconvex quadratic programs with semidefinite-based finite branch-and-bound. Comput. Optim. Appl. **2**, 181–195 (2009)

3. De Angelis, P., Pardalos, P., Toraldo, G.: Quadratic programming with box constraints. In: Bomze, I.M., Csendes, T., Horst, R., Pardalos, P. (eds.) Developments in Global Optimization, pp. 73–94. Kluwer Academic, Norwell (1997)
4. Gould, N.I.M., Toint, P.L.: Numerical methods for large-scale non-convex quadratic programming. In: Trends in Industrial and Applied Mathematics, Amritsar, 2001. Appl. Optim., vol. 72, pp. 149–179. Kluwer Academic, Dordrecht (2002)
5. Horst, R., Tuy, H.: Global Optimization, second edition. Springer-Verlag, Berlin (1993). Deterministic Approaches
6. Löfberg Yalmip, J.: A toolbox for modeling and optimization in MATLAB. In: Proceedings of the CACSD Conference, Taipei, Taiwan (2004). URL http://control.ee.ethz.ch/~joloef/yalmip.php
7. Nesterov, Y.: Global quadratic optimization via conic relaxation. In: Saigal, R., Vandenberghe, L., Wolkowicz, H. (eds.) Handbook of Semidefinite Programming, pp. 363–386. Kluwer Academic, Dordrecht (2000)
8. Pardalos, P.: Global optimization algorithms for linearly constrained indefinite quadratic problems. Comput. Math. Appl. **21**, 87–97 (1991)
9. Pardalos, P.M., Vavasis, S.A.: Quadratic programming with one negative eigenvalue is NP-hard. J. Global Optim. **1**(1), 15–22 (1991)
10. Sherali, H.D., Adams, W.P.: A Reformulation-Linearization Technique (RLT) for Solving Discrete and Continuous Nonconvex Problems. Kluwer Academic, Dordrecht (1997)
11. Shor, N.: Quadratic optimization problems. Sov. J. Comput. Syst. Sci. **25**, 1–11 (1987). Originally published in Tekh. Kibern. **1**, 128–139 (1987)
12. Sturm, J.F.: Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optim. Methods Softw. **11/12**(1–4), 625–653 (1999). URL http://sedumi.mcmaster.ca/
13. Vandenbussche, D., Nemhauser, G.: A polyhedral study of nonconvex quadratic programs with box constraints. Math. Program. **102**(3), 531–557 (2005a)
14. Vandenbussche, D., Nemhauser, G.: A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. Math. Program. **102**(3), 559–575 (2005b)
15. Ye, Y.: Approximating quadratic programming with bound and quadratic constraints. Math. Program. **84**(2, Ser. A), 219–226 (1999)