Comput Optim Appl (2009) 43: 181–195
DOI 10.1007/s10589-007-9137-6

Globally solving box-constrained nonconvex quadratic programs with semidefinite-based finite branch-and-bound

Samuel Burer · Dieter Vandenbussche

Received: 7 November 2006 / Revised: 17 July 2007 / Published online: 31 October 2007

Abstract We consider a recent branch-and-bound algorithm of the authors for nonconvex quadratic programming. The algorithm is characterized by its use of semidefinite relaxations within a finite branching scheme. In this paper, we specialize the algorithm to the box-constrained case and study its implementation, which is shown to be a state-of-the-art method for globally solving box-constrained nonconvex quadratic programs.

Keywords Nonconcave quadratic maximization · Nonconvex quadratic programming · Branch-and-bound · Lift-and-project relaxations · Semidefinite programming

1 Introduction

This paper studies the problem of maximizing a quadratic objective subject to box constraints:

$$\text{(QPB)} \quad \begin{array}{ll} \max & \frac{1}{2}x^T Qx + c^T x \\ \text{s.t.} & 0 \le x \le e, \end{array}$$

where $x \in \mathbb{R}^n$, e is the vector of all ones, and $(Q, c) \in \mathbb{R}^{n\times n} \times \mathbb{R}^n$ are the data. We will refer to the feasible set of (QPB) as P. Without loss of generality, we assume

S. Burer was supported in part by NSF Grants CCR-0203426 and CCF-0545514.

S. Burer (✉)
Department of Management Sciences, University of Iowa, Iowa City, IA 52242-1000, USA
e-mail: samuel-burer@uiowa.edu

D. Vandenbussche
Axioma, Inc., Marietta, GA 30068, USA

Q is symmetric. If Q is negative semidefinite, then (QPB) is solvable in polynomial time [12]. Here, however, we consider that Q is indefinite or positive semidefinite, in which case (QPB) is an $\mathcal{NP}$-hard problem [15]. Nevertheless, our goal is to obtain a global maximizer.

The difficulty in optimizing (QPB) is that it harbors many local maxima. There are numerous methods for solving (QPB) and more general nonconvex quadratic programs, including local methods [8] and global methods [14]. For a survey of methods to globally solve (QPB), see [6]. Existing global optimization techniques for (QPB) can be classified into two groups. Those in the first group work by recursively partitioning P, but due to the convex nature of P, it may theoretically be necessary to subdivide P infinitely (i.e., to generate an infinite branch-and-bound tree). In contrast, those in the second group use a finite branching scheme, based on ideas from [9], which allows for logical branching on the first-order KKT conditions of (QPB). We term this *KKT-branching*. Among finite methods, one of the most successful implementations has been the recent approach of [16, 17], which uses linear programming ("LP") relaxations along with cuts for the convex hull of first-order KKT points in a highly efficient branch-and-cut implementation.

Based on an extension of KKT-branching, the authors [5] have recently introduced the first finite branch-and-bound algorithm for globally solving general bounded nonconvex quadratic programming problems, of which (QPB) is a special case. In addition to KKT-branching, the method depends on the use of semidefinite relaxations of the first-order KKT conditions at each node of the tree. In particular, the natural LP relaxations are unbounded, and so a stronger tool (namely semidefinite programming, or "SDP") is employed.

The ideas explored in this paper are related to, but different from, other applications of SDP in branch-and-bound for NP-hard problems. For example, [10] studies a branch-and-cut approach for 0-1 QPs, which solves SDP relaxations, and [2] solves large instances of the quadratic assignment problem using convex QP relaxations, which have been derived from SDP considerations. More generally, our approach follows a long stream of papers which apply SDP to NP-hard problems. (We refer the reader to [5] for numerous related references.) In particular, it is worth mentioning that Braun and Mitchell [3] study SDP relaxations of complementarity constraints, just as we do in this paper via the first-order KKT constraints.

In this paper, we specialize the finite SDP-based branch-and-bound algorithm from [5] to the case of (QPB). In contrast to the general SDP relaxations suggested by the authors, we develop an SDP relaxation that better exploits branching information, while still maintaining a compact size. We then verify that the change in SDP relaxation does not affect the theoretical properties of the branch-and-bound algorithm and finally conduct a thorough comparison of our algorithm with that of [16], where we demonstrate our ability to solve significantly larger (QPB) problems in still modest amounts of time.

1.1 Terminology and notation

In this section, we introduce some of the notation that will be used throughout the paper. $\mathbb{R}^n$ refers to n-dimensional Euclidean space. The norm of a vector $x \in \mathbb{R}^n$

is denoted by $\|x\| := \sqrt{x^T x}$. We let $e_i \in \mathbb{R}^n$ represent the i-th unit vector. $\mathbb{R}^{n\times n}$ is the set of real, $n \times n$ matrices; $\mathcal{S}^n$ is the set of symmetric matrices in $\mathbb{R}^{n\times n}$; and $\mathcal{S}^n_+$ is the set of positive semidefinite symmetric matrices. The special notations $\mathbb{R}^{1+n}$ and $\mathcal{S}^{1+n}$ are used to denote the spaces $\mathbb{R}^n$ and $\mathcal{S}^n$ with an additional "0-th" entry prefixed or an additional 0-th row and 0-th column prefixed, respectively. Given a matrix $X \in \mathbb{R}^{n\times n}$, we denote by $X_{\cdot j}$ and $X_{i\cdot}$ the j-th column and i-th row of X, respectively. The inner product of two matrices $A, B \in \mathbb{R}^{n\times n}$ is defined as $A \bullet B := \text{trace}(A^T B)$, where trace$(\cdot)$ denotes the sum of the diagonal entries of a matrix. Given two vectors $x, z \in \mathbb{R}^n$, we denote their Hadamard product by $x \circ z \in \mathbb{R}^n$, where $[x \circ z]_j = x_j z_j$. Finally, diag(A) is defined as the vector with the diagonal of the matrix A as its entries.

2 The branch-and-bound algorithm

2.1 Finite KKT-branching

By introducing nonnegative multipliers y and z for the constraints $x \le e$ and $x \ge 0$ of P, respectively, any locally optimal solution x of (QPB) has the property that the sets

$$G_x := \{(y, z) \ge 0 : y - z = Qx + c\},$$
$$C_x := \{(y, z) \ge 0 : (e - x) \circ y = 0,\ x \circ z = 0\}$$

satisfy $G_x \cap C_x \neq \emptyset$. In words, G_x is the set of multipliers where the gradient of the Lagrangian vanishes, and C_x consists of those multipliers satisfying complementarity at x. The condition $G_x \cap C_x \neq \emptyset$ specifies that x is a first-order KKT point. Notice also that $y \circ z = 0$ for all $(y, z) \in C_x$.

One can easily show the following property of KKT points.

Proposition 2.1 ([7]) *Suppose $x \in P$ and $(y, z) \in G_x$. Then $x^T Qx + c^T x \le e^T y$, with equality if and only if $(y, z) \in C_x$.*

This shows that (QPB) may be reformulated as the following linear program with complementarity constraints:

$$\text{(KKT)} \quad \begin{array}{ll} \max & \frac{1}{2} e^T y + \frac{1}{2} c^T x \\ \text{s.t.} & x \in P \quad (y, z) \in G_x \cap C_x. \end{array}$$

The reformulation (KKT) suggests a finite branch-and-bound approach, where complementarity is recursively enforced using linear equations. A particular node of the tree is specified by four index sets $F^0, F^1, F^y, F^z \subseteq \{1, \ldots, n\}$, which satisfy $F^0 \cap F^z = \emptyset$ and $F^1 \cap F^y = \emptyset$. These index sets correspond to complementarities

that are enforced via linear equations in the following restriction of (KKT):

$$\begin{array}{lll} \max & \frac{1}{2}e^T y + \frac{1}{2}c^T x & \\ \text{s.t.} & x \in P \quad (y,z) \in G_x \cap C_x, & \\ & x_j = 0, \quad j \in F^0, & \\ & x_j = 1, \quad j \in F^1, & \\ & z_j = 0, \quad j \in F^z, & \\ & y_j = 0, \quad j \in F^y. & \end{array} \tag{1}$$

In fact, logical considerations imply that further conditions can be imposed:

$$F^0 \subseteq F^y,$$
$$F^1 \subseteq F^z.$$

(In addition, we could enforce $F^0 \cap F^1 = \emptyset$ since otherwise (1) is infeasible, but we do not explicitly do so since this type infeasibility is straightforward to check at run-time.)

At a given node, the branch-and-bound algorithm will solve a convex relaxation of (1). Due to the presence of the linear equations, one can relax the nonconvex quadratic constraint $(y,z) \in C_x$ and yet still maintain (partial) complementarity via the linear equations: $x_j z_j = 0$ will hold for all $j \in F^0 \cup F^z$, and $(1 - x_j) y_j = 0$ will hold for all $j \in F^1 \cup F^y$. Branching on a node amounts to choosing a complementarity to enforce in subsequent nodes and in particular involves one of the following actions:

(i) Selecting some $j \in \{1, \ldots, n\} \setminus (F^0 \cup F^z)$ and creating two children, one which adds j to F^0 and to F^y and one which adds j to F^z;
(ii) Selecting some $j \in \{1, \ldots, n\} \setminus (F^1 \cup F^y)$ and creating two children, one which adds j to F^1 and to F^z and one which adds j to F^y.

A natural branching strategy, which we will employ in the computational results of Sect. 4, is to select the complementarity constraint with the largest (normalized) violation. This is analogous to branching on the "most fractional" variable in integer programming. (More details on this branching strategy, particularly the concept of normalization, are given in Sect. 4.)

Finally, we remark that the root node in the tree has all four index sets empty, and a leaf node is specified by sets satisfying $F^0 \cup F^z = F^1 \cup F^y = \{1, \ldots, n\}$.

2.2 SDP relaxations

In [5], the authors have proposed the use of SDP relaxations to globally optimize general QPs. In this section, we briefly recapitulate their results as they apply to (1).

We first introduce a basic SDP relaxation of (QPB). Our motivation comes from the SDP relaxations of 0-1 integer programs introduced by [13]. Consider a matrix variable Y, which is related to $x \in P$ by the following quadratic equation:

$$Y = \begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}^T = \begin{pmatrix} 1 & x^T \\ x & xx^T \end{pmatrix}. \tag{2}$$

From (2), we observe the following:

- Y is symmetric and positive semidefinite, i.e., $Y \in \mathcal{S}_+^{1+n}$ (or simply $Y \succeq 0$);
- If we multiply the constraint $x \leq e$ of P by some x_i, we obtain the quadratic inequality $e\,x_i - x\,x_i \geq 0$, which is valid for P and can be written in terms of Y as

$$\big(e \big| -I\big) Y e_i \geq 0 \quad \forall i = 1, \ldots, n;$$

- The objective function of (QPB) can be modeled in terms of Y as

$$\frac{1}{2} x^T Q x + c^T x = \frac{1}{2} \begin{pmatrix} 0 & c^T \\ c & Q \end{pmatrix} \bullet \begin{pmatrix} 1 & x^T \\ x & xx^T \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 0 & c^T \\ c & Q \end{pmatrix} \bullet Y.$$

For convenience, we define

$$K := \{(x_0, x) \in \mathbb{R}^{1+n} \ : \ 0 \leq x \leq x_0\, e\}$$

and

$$\tilde{Q} := \frac{1}{2} \begin{pmatrix} 0 & c^T \\ c & Q \end{pmatrix},$$

which allow us to express the second and third properties above more simply as $Y e_i \in K$ and $x^T Q x/2 + c^T x = \tilde{Q} \bullet Y$. In addition, we let M_+ denote the set of all Y satisfying the first and second properties:

$$M_+ := \{Y \succeq 0 : Y e_i \in K \ \ \forall \ \ i = 1, \ldots, n\}.$$

Then we have the following equivalent formulation of (QPB):

$$\begin{array}{lll} \max & \tilde{Q} \bullet Y & \\ \text{s.t.} & Y = \begin{pmatrix} 1 & x^T \\ x & xx^T \end{pmatrix} \in M_+, & \\ & x \in P. & \end{array}$$

Finally, by dropping the last n columns of (2) (which is equivalent to relaxing the condition that the rank of Y is 1 due to (2)), we arrive at the following linear SDP relaxation of (QPB):

$$\begin{array}{llll} (\text{SDP}_0) & \max & \tilde{Q} \bullet Y & \\ & \text{s.t.} & Y \in M_+, \quad Y e_0 = (1; x), & \\ & & x \in P. & \end{array}$$

We next combine this basic SDP relaxation with the linear constraints of (KKT) introduced in Sect. 2.1 to obtain an SDP relaxation of (KKT). More specifically, we consider the following optimization over the combined variables (Y, x, y, z) in which the two objectives are equated using an additional constraint:

$$\begin{array}{llll} (\text{SDP}_1) & \max & \tilde{Q} \bullet Y & \\ & \text{s.t.} & Y \in M_+, \quad Y e_0 = (1; x), & \\ & & x \in P, \quad (y, z) \in G_x, & \\ & & \tilde{Q} \bullet Y = (e^T y + c^T x)/2. & \end{array} \tag{3}$$

This optimization problem is a valid relaxation of (KKT) if the constraint (3) is valid for all points feasible for (KKT), which indeed holds as follows: let (x, y, z) be such a point, and define Y according to (2); then (Y, x, y, z) satisfies the first four constraints of (SDP$_1$) by construction and the constraint (3) is satisfied due to Proposition 2.1 and (2).

2.3 The SDP relaxations within branch-and-bound

In Sect. 2.1, we have described the basic approach for constructing a branch-and-bound algorithm for (QPB) by recursively enforcing complementarities through branching. In conjunction with the SDP relaxations, the authors have established a finite branch-and-bound algorithm.

Theorem 2.2 ([5]) *The branch-and-bound approach for* (QPB), *which employs KKT-branching and SDP relaxations of the type* (SDP$_1$), *is both correct and finite*.

Regarding the theorem, it is important to discuss two points. First, although the SDP relaxation (SDP$_1$) has been constructed as a relaxation of (KKT), it is easy to extend the same ideas to yield SDP relaxations of the restricted KKT subproblem (1) associated with each node. The only modification is the inclusion of the linear equalities represented by F^0, F^1, F^y, and F^z into the definitions of P, K, and G_x in the natural way to yield corresponding sets $P(F^0, F^1)$, $K(F^0, F^1)$, and $G_x(F^y, F^z)$. Specifically, at any node, we define

$$P(F^0, F^1) := \left\{ x \in P: \begin{array}{ll} x_j = 1 & \forall j \in F^1 \\ x_j = 0 & \forall j \in F^0 \end{array} \right\}, \tag{4}$$

$$K(F^0, F^1) := \left\{ (x_0, x) \in K: \begin{array}{ll} x_j = x_0 & \forall j \in F^1 \\ x_j = 0 & \forall j \in F^0 \end{array} \right\}, \tag{5}$$

$$G_x(F^z, F^y) := \left\{ (y, z) \in G_x: \begin{array}{ll} y_j = 0 & \forall j \in F^y \\ z_j = 0 & \forall j \in F^z \end{array} \right\}.$$

In advance of the proofs of Propositions 3.1 and 3.2, we remark that the constraint $Ye_i \in K(F^0, F^1)$ will be enforced in the SDP relaxation at each node of the tree as part of the appropriate generalization of M_+. In particular, this means $[Ye_i]_j = [Ye_i]_0$ for all $j \in F^1$, which is equivalent to $[Ye_i]_j = x_i$, where x is a variable of the SDP relaxation (not to be confused with the notation (x_0, x) in the definition of $K(F^0, F^1)$ above).

Second, we have explained what the term *finite* means with respect to the branch-and-bound algorithm of Theorem 2.2, but what does *correct* mean? Overall, the branch-and-bound algorithm starts at the root node, and begins evaluating nodes, adding or removing nodes from the tree at each stage. Evaluating a node involves solving the SDP relaxation and then fathoming the node (if possible) or branching on the node (if fathoming is not possible and if the node is not a leaf). The algorithm finishes when all nodes generated by the algorithm have been fathomed. In order to prove that our algorithm does indeed finish, it is necessary to establish that all leaf

nodes will be fathomed. Otherwise, they cannot be eliminated from the tree since they cannot be branched on. This we term the *correctness* of the algorithm.

We explain fathoming in a bit more detail. Fathoming a node (i.e., eliminating it from further consideration) is only allowable if we can guarantee that its associated subproblem (1) contains no optimal solutions of (QPB), other than possibly the solution $(\bar{x}, \bar{y}, \bar{z})$ obtained from the relaxation. Such a guarantee can be obtained in two ways. First, if the relaxation is infeasible, then (1) is infeasible so that it contains no optimal solutions. Second, we can fathom if the relaxed objective value at $(\bar{x}, \bar{y}, \bar{z})$ is equal to or less than the (QPB) objective of some $x \in P$. In particular, we compare the relaxed value with the (QPB) value of $\bar{x}$ itself as well as that of other solutions encountered at other nodes. Because the (QPB) value of $\bar{x}$ cannot be greater than the relaxed value, fathoming occurs due to $\bar{x}$ only when the two values are equal, i.e., when the relaxation has no gap.

Accordingly, to have a correct branch-and-bound algorithm, it must be the case that (when feasible) the SDP relaxation has no gap at a leaf node. A key ingredient of proving Theorem 2.2 is establishing this fact.

A couple of remarks about fathoming are in order:

- A common viewpoint in branch-and-bound for integer programming is that fathoming can also occur by feasibility of the solution of the relaxed subproblem. In other words, when a relaxed solution is integer, that node can be fathomed because its integer subproblem has been provably solved to optimality. Indeed, the fact that the solution is integer is the certificate of optimality. In contrast, our situation is more subtle. If the relaxed solution $(\bar{x}, \bar{y}, \bar{z})$ happens to be a KKT point, then, depending on the precise form of the relaxation, it need not hold *a priori* that $(\bar{x}, \bar{y}, \bar{z})$ is an optimal solution of the complementarity subproblem (1). (More details are given in [5].) The difficulty arises from the fact that the relaxed objective function is not identical to the original objective of (QPB) due to linearization as in (SDP_1). (In integer programming, the original and subproblem objectives are identical because the objective is linear.) In cases where $(\bar{x}, \bar{y}, \bar{z})$ is in fact optimal for (1), the only available certificate is that the relaxed objective equals the (QPB) objective at $\bar{x}$. These subtleties have motivated our discussion on fathoming above and in particular are the reason we do not state "fathoming by feasibility" in analogy with integer programming.
- In practical computations, the fathoming just described must be modified to allow for floating point errors. For this, a fathoming tolerance is typically introduced. Details for our implementation are described in Sect. 4.

3 Further specialization to the box-constrained case

We now specialize our algorithm even further to (QPB). In Sect. 2.2, we defined the SDP relaxation (SDP_1) that explicitly contains the dual multipliers (y, z). We intend to show that we can actually handle (y, z) implicitly within an SDP relaxation similar to (SDP_0).

The key observation is that, for any x, one can easily obtain $(y, z) \in G_x$ by simply setting $y := \max(0, Qx + c)$ and $z := -\min(0, Qx + c)$, where *max* and *min* are taken

component-wise. Furthermore, by defining y and z in this way, fixing components of y and z to 0 (as is required during branch-and-bound) can be enforced by linear constraints that involve only x. In particular, $y_j = 0$ is equivalent to $[Qx]_j + c_j \leq 0$, and $z_j = 0$ is equivalent to $[Qx]_j + c_j \geq 0$.

Given F^0, F^1, F^y, and F^z, the corresponding restricted version of (QPB) that we consider simply substitutes the following $\widetilde{P}$ for P:

$$\widetilde{P} := P(F^0, F^1) \cap S(F^y, F^z), \tag{6}$$

where $P(F^0, F^1)$ is given by (4) and

$$S(F^y, F^z) := \left\{ x \in \mathbb{R}^n : \begin{array}{ll} [Qx]_j + c_j \leq 0 & \forall j \in F^y \\ [Qx]_j + c_j \geq 0 & \forall j \in F^z \end{array} \right\}.$$

Accordingly, the SDP relaxation that we consider is of the form (SDP_0), except we use $\widetilde{P}$ instead of P and the following $\widetilde{K}$ in place of K:

$$\widetilde{K} := K(F^0, F^1) \cap T(F^y, F^z), \tag{7}$$

where $K(F^0, F^1)$ is given by (5) and

$$T(F^y, F^z) := \left\{ (x_0, x) \in \mathbb{R}^{1+n} : \begin{array}{ll} [Qx]_j + x_0 c_j \leq 0 & \forall j \in F^y \\ [Qx]_j + x_0 c_j \geq 0 & \forall j \in F^z \end{array} \right\}.$$

3.1 Correctness of branch-and-bound

If we are to use the relaxation (SDP_0) tailored by (6) and (7) within branch-and-bound, then Theorem 2.2 no longer applies because it is based on (SDP_1). So, in this new context, we still have to guarantee correctness, i.e., we must ensure that a leaf node will always be fathomed. In other words, in the case of a feasible leaf node, we must verify that the upper bound obtained from the relaxation is equal to the value of the primal solution generated from solving this node. This is established by the following proposition.

Proposition 3.1 *Consider a feasible leaf node of the branch-and-bound tree, and suppose (Y, x) is a feasible solution to the relaxation (SDP_0) (appropriately tailored by $\widetilde{P}$ and $\widetilde{K}$ to incorporate the forced equalities at the node). Then $\tilde{Q} \bullet Y = \frac{1}{2} x^T Q x + c^T x$.*

Proof For convenience, we write

$$Y = \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix}$$

so that $\tilde{Q} \bullet Y = Q \bullet X/2 + c^T x$. Thus, it suffices to show

$$Q_{\cdot j}^T X_{\cdot j} = (Q_{\cdot j}^T x) x_j \quad \forall j \in \{1, \ldots, n\}. \tag{8}$$

Let j be any index. Recall that a leaf node satisfies $F^0 \cup F^z = \{1, \ldots, n\}$ with $F^0 \cap F^z = \emptyset$ and $F^1 \cup F^y = \{1, \ldots, n\}$ with $F^1 \cap F^y = \emptyset$. Moreover, since the leaf node is feasible, we certainly have $F^0 \cap F^1 = \emptyset$. So we have three possibilities: either $j \in F^0$, $j \in F^1$, or $j \in F^z \cap F^y$:

- Suppose $j \in F^0$. Since $x \in \widetilde{P}$ and $Ye_i \in \widetilde{K}$, we have that the j-th entry of each column of Y is 0, i.e., $Y_{j\cdot} = 0$. By symmetry, $Y_{\cdot j} = (x_j \,;\, X_{\cdot j}) = 0$. So (8) follows easily.
- Suppose now $j \in F^1$. Since $x \in \widetilde{P}$, we have $x_j = 1$. Furthermore, since $Ye_i \in \widetilde{K}$, we have $Y_{ji} = x_i$ for all $i = 1, \ldots, n$. So $Y_{j\cdot} = (1, x^T)$, and by symmetry, we have $X_{\cdot j} = x$. Since $x_j = 1$, we again see that (8) is satisfied.
- Lastly, suppose $j \in F^y \cap F^z$. Then $\widetilde{P}$ contains the implied constraint $[Qx]_j + c_j = 0$. In a similar fashion, $Ye_j \in \widetilde{K}$ implies $[QX_{\cdot j}]_j + x_j c_j = 0$. By multiplying the first equality by x_j and then combining with the second, we obtain (8). □

In fact, the following proposition shows that a correct branch-and-bound algorithm is obtained even if $\widetilde{K}$ is relaxed to

$$\overline{K} := K \cap T(F^y, F^z). \tag{9}$$

which differs from $\widetilde{K}$ in that the equalities coming from F^0 and F^1 are not enforced.

Proposition 3.2 *Suppose (Y, x) satisfies $Y \succeq 0$, $Ye_0 = (1; x)$, and $x \in \widetilde{P}$. If $Ye_i \in \overline{K}$ for all $i = 1, \ldots, n$, then $Ye_i \in \widetilde{K}$ for all $i = 1, \ldots, n$.*

Proof Let $i \in \{1, \ldots, n\}$. To show $Ye_i \in \widetilde{K}$, it suffices to show $X_{ji} = 0$ for all $j \in F^0$ and $X_{ji} = x_i$ for all $j \in F^1$, where we identify

$$Y = \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix}.$$

This is proved as follows:

- If $j \in F^0$, then $x \in \widetilde{P}$ and $Ye_j \in \overline{K}$ imply that $0 \le X_{\cdot j} \le x_j e = 0$. In particular, $X_{ij} = 0$, and hence, by symmetry, $X_{ji} = 0$.
- Suppose $j \in F^1$ and consider the following 2×2 principal submatrix of Y:

$$\begin{pmatrix} 1 & x_j \\ x_j & X_{jj} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & X_{jj} \end{pmatrix} \succeq 0.$$

  Since the determinant must be nonnegative, we have $X_{jj} \ge 1$. We also have $X_{jj} \le x_j = 1$, and so $X_{jj} = 1$. This gives rise to the following 3×3 principal submatrix of Y (after permuting rows and columns if $i < j$):

$$\begin{pmatrix} 1 & x_j & x_i \\ x_j & X_{jj} & X_{ji} \\ x_i & X_{ij} & X_{ii} \end{pmatrix} = \begin{pmatrix} 1 & 1 & x_i \\ 1 & 1 & X_{ji} \\ x_i & X_{ji} & X_{ii} \end{pmatrix} \succeq 0. \tag{10}$$

Note that if $X_{ii} = 0$, then $x_i = X_{ji} = X_{ii} = 0$, as desired. If $X_{ii} > 0$, then we can rescale the last row and column of the last matrix in (10) by $\sqrt{X_{ii}}$ to get

$$\begin{pmatrix} 1 & 1 & \bar{x}_i \\ 1 & 1 & \bar{X}_{ji} \\ \bar{x}_i & \bar{X}_{ji} & 1 \end{pmatrix}.$$

The determinant of this matrix is $-(\bar{x}_i - \bar{X}_{ji})^2$, which, by (10), must be nonnegative. This implies that $\bar{x}_i = \bar{X}_{ji} \Rightarrow x_i = X_{ji}$. □

In practice, it is likely that $\widetilde{K}$ will be a better choice than $\overline{K}$ because it allows for the easy elimination of variables (and hence smaller relaxations), while ultimately having the same number of constraints as $\widetilde{K}$ (i.e., those represented by $T(F^y, F^z)$). So in our computational results, we prefer $\widetilde{K}$ over $\overline{K}$. (More details are given in the next section. In particular, we will take advantages of CPLEX's preprocessing techniques to automate the elimination of variables.)

4 The branch-and-bound implementation

In this section, we describe our computational experience with the branch-and-bound algorithm for (QPB) using relaxations as discussed in the previous section.

4.1 Implementation details

One of the most fundamental decisions for any branch-and-bound algorithm is the method employed for solving the relaxations at the nodes of the tree. As the authors have done previously, we have chosen to use the method proposed by [4] for solving the SDP relaxations because of its applicability and scalability for SDPs of the type (SDP_0). For the sake of brevity, we only describe the features of the method that are relevant to our discussion, since a number of the method's features directly affect key implementation decisions.

The algorithm uses a Lagrangian constraint-relaxation approach, governed by an augmented Lagrangian scheme to ensure convergence, that focuses on obtaining subproblems that only require the solution of convex quadratic programs over the constraint set K or $\widetilde{K}$, which are solved using CPLEX [11]. In particular, many constraints (but not all) are relaxed with explicit dual variables. By the nature of the method, a valid upper bound on the optimal value of the relaxation is available at all times, which makes the method a reasonable choice within branch-and-bound even if the relaxations are not solved to high accuracy. For convenience, we will refer to this method as the *AL method*.

To further expedite the branch-and-bound process, we also attempted to tighten (SDP_0) by adding constraints of the form $Y(e_0 - e_i) \in K$, which arise by multiplying the original inequalities $0 \le x \le e$ by $1 - x_j$, which is nonnegative. Although these additional constraints do increase the computational cost of the AL method, the impact is small due to the decomposed nature of the AL method. Moreover, the benefit

to the overall branch-and-bound performance is dramatic due to strengthened bounds coming from the relaxations.

Some final details of the branch-and-bound implementation are:

- After solving a relaxation at a node to obtain $\bar{x}$, if branching is necessary, we compute $\bar{y} := \max(0, Q\bar{x} + c)$ and $\bar{z} := -\min(0, Q\bar{x} + c)$ as discussed above, and a branching index is selected by the choosing the complementarity constraint with the largest normalized violation.

  Normalization uses the following *a priori* upper bounds for $(y, z) \in G_x \cap C_x$. First note that all $(y, z) \in C_x$ satisfy $y \circ z = 0$. So if $y_i > 0$, then $z_i = 0$. From G_x, we have that $y_i - z_i = [Qx]_i + c_i$ and hence if $y_i > 0$ then

$$y_i = [Qx]_i + c_i = \sum_{j=1}^{n} Q_{ij}x_j + c_i \leq \sum_{j=1}^{n} \max(Q_{ij}, 0) + c_i.$$

  Similarly, if $z_i > 0$, then

$$z_i = -[Qx]_i - c_i = -\sum_{j=1}^{n} Q_{ij}x_j - c_i \leq -\sum_{j=1}^{n} \min(Q_{ij}, 0) - c_i.$$

  We then have, for example, that $\bar{x}_j\bar{z}_j/\hat{z}_j$ measures the normalized violation at index j, where $\hat{z}_j = -c_i - \sum_j \min(Q_{ij}, 0)$.
- It is not difficult to see that if $Q_{jj} \geq 0$ for some j, then there exists an optimal solution of (QPB) with $x_j \in \{0, 1\}$ (see [9, 17]). Hence, when branching on such an index, we bypass the standard rule for child creation and instead create two children, one with $x_j = 0$ and $y_j = 0$, and one with $x_j = 1$ and $z_j = 0$.
- After solving a relaxation, we use $\bar{x}$ as a starting point for a QP solver based on nonlinear programming techniques to obtain a locally optimal solution to (QPB). In this way, we obtain good lower bounds that can be used to fathom nodes in the tree.
- We use a best bound strategy for selecting the next node to solve in the branch-and-bound tree.
- Upon solution of each relaxation of the branch-and-bound tree, a set of dual variables is available for those constraints of the SDP relaxation that were relaxed in the AL method. These dual variables are then used as the initial dual variables for the solution of the next subproblem, in effect performing a crude warm-start, which proved extremely helpful in practice.
- We use a relative optimality tolerance for fathoming. In other words, for a given tolerance ε, a node with upper bound z_{ub} is fathomed if $(z_{ub} - z_{lb})/z_{lb} < \varepsilon$, where z_{lb} is the value of the best primal solution found thus far. In our computations, we experiment with $\varepsilon = 10^{-6}$ and $\varepsilon = 10^{-2}$, which we refer to as *default* and *1%* optimality tolerances, respectively.

4.2 Computational results

We compare our algorithm with the branch-and-cut algorithm of Vandenbussche and Nemhauser [16, 17]. For their algorithm, we use the parameter settings suggested in

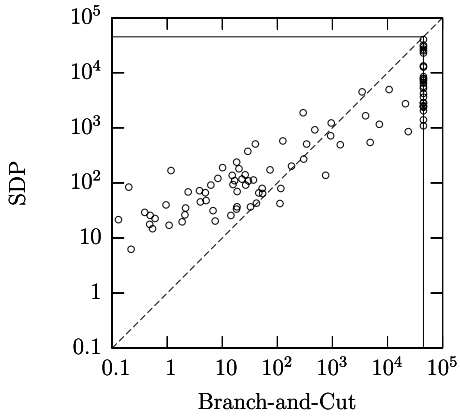

Fig. 1 CPU times (log-log scale) required for SDP-based branch-and-bound and LP-based branch-and-cut with default optimality tolerance on box-constrained QP instances. A maximum time limit of 45,000 seconds was enforced for all runs

their work and run their algorithm using the two branching selection schemes they employed: maximum normalized violation (identical to ours described above) and strong branching. For each instance we tested, we used the faster of these two to compare with our SDP approach. The creation of child nodes was also performed as described above.

The test problems include the 54 proposed by [17], and we have also generated larger instances to demonstrate the enhanced capabilities available using the SDP-based branch-and-bound. The additional instances vary in both size and density of the matrix Q. The different sizes are $n = 70$, 80, 90, and 100 and the different densities are 25%, 50%, and 75%. Nonzeros were uniformly generated integers over the interval $[-50, 50]$. For each size and density combination, we generated three different instances for a total of 36 new problems. Overall, we tested 90 instances.

For the first set of computational results we present, the default optimality tolerance was used to fathom the nodes. Figure 1 shows a log-log plot of the CPU times for our branch-and-bound algorithm against those of branch-and-cut. A maximum time limit of 45,000 seconds was enforced for all runs. From the figure, one can see that many instances are solved more quickly by branch-and-cut, while a large subset are solved faster by the SDP approach. In fact, branch-and-cut was unable to complete 25 of the 90 instances within the allotted time. Although the figure does not show size information for the instances, branch-and-cut is faster on the smaller instances, while our approach is faster on the larger instances. We conclude that the SDP-based approach scales better than branch-and-cut.

To illustrate how far some of the branch-and-cut instances were from finishing, we plot their optimality gaps at termination in Fig. 2. The instances are ordered first with respect to size and then with respect to density of the matrix Q, and the labels on the plot specify the instances of various dimensions. We show only the gaps for instances with $n \geq 70$, since all smaller instances terminated within the time limit. The figure shows that branch-and-cut was unable to solve most of the instances with $n \geq 80$, and the gaps clearly indicate that branch-and-cut still had much work to do on these instances. In contrast, the SDP approach completed all instances within the time limit.

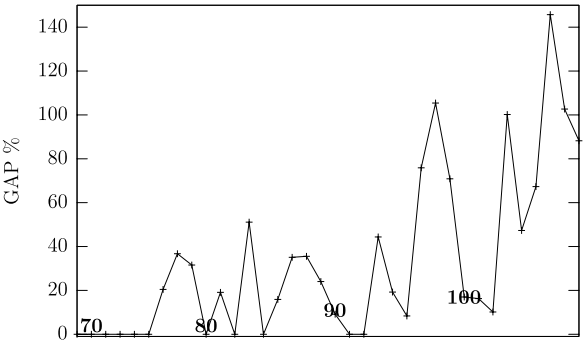

Fig. 2 Optimality gaps (%) upon termination for LP-based branch-and-cut when run with default optimality tolerance. Instances are ordered first with respect to size and then with respect to density of the matrix Q, and labels specify the instances of various dimensions

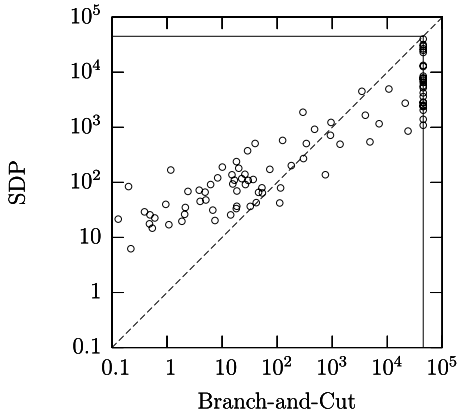

Fig. 3 CPU times (log-log scale) required for SDP-based branch-and-bound and LP-based branch-and-cut with 1% optimality tolerance on box-constrained QP instances. A maximum time limit of 20,000 seconds was enforced for all runs

The maximum number of nodes required for any SDP run was 305, while the same measure for branch-and-cut was 181,958. These numbers illustrate the tightness of the bounds provided by the SDP relaxation, but also point out the trade-off between the strength of a relaxation and the difficulty of solving it—since the solution of the LP relaxations was extremely quick compared to solving the SDP relaxations.

To further highlight the power of the SDP approach, we also solved the same instances using a 1% relative optimality tolerance. The results are presented in Figs. 3 and 4. We gave both algorithms a time limit of 20,000 CPU seconds. The SDP-based approach finished all instances, whereas branch-and-cut did not finish 26 problems. Note also that almost all instances that required more than 100 seconds with branch-and-cut could be solved more quickly with the SDP branch-and-bound.

4.3 Comparison with other methods

To provide some perspective on how our method compares with standard global optimization techniques for (QPB), we briefly discuss our method in relation to the algorithm of [1], which recursively subdivides the feasible region P into ever smaller (rectangular) boxes and bounds the objective on each box by approximating that box by an ellipsoid. Although the comparison is at a high level, we feel it highlights the typical differences between our method and standard global optimization methods.

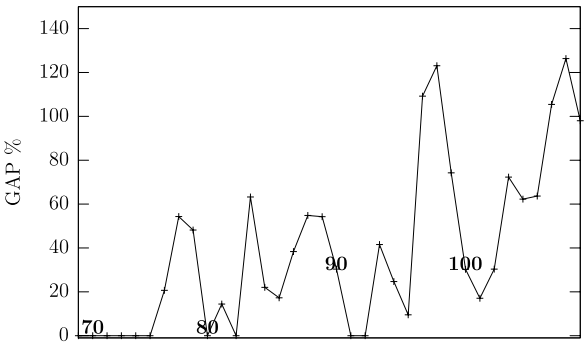

Fig. 4 Optimality gaps (%) upon termination for LP-based branch-and-cut when run with 1% optimality tolerance. Instances are ordered first with respect to size and then with respect to density of the matrix Q, and labels specify the instances of various dimensions

(Note also that we are unaware of any publicly available global optimization codes for (QPB).)

The first important difference is that An and Tao's method *theoretically* requires an infinite branch-and-bound tree, whereas our method is based on finite branching. Secondly, the bounding mechanism used by An and Tao appears to allow less overall tolerance than the semidefinite bounds we employ. For example, 1% is the tightest optimality tolerance An and Tao consider, whereas we have also solved to a tolerance of 0.0001%. Finally, An and Tao solve instances with n as high as 550, which exceeds the dimension solved in this paper. This attests to the speed and scalability of their bounding scheme.

References

1. An, L.T.H., Tao, P.D.: A branch and bound method via d.c. optimization algorithms and ellipsoidal technique for box constrained nonconvex quadratic problems. J. Glob. Optim. **13**(2), 171–206 (1998)
2. Anstreicher, K., Brixius, N., Goux, J.-P., Linderoth, J.: Solving large quadratic assignment problems on computational grids. Math. Program. **91**(3, Ser. B), 563–588 (2002). ISMP 2000, Part 1 (Atlanta, GA)
3. Braun, S., Mitchell, J.E.: A semidefinite programming heuristic for quadratic programming problems with complementarity constraints. Comput. Optim. Appl. **31**(1), 5–29 (2005)
4. Burer, S., Vandenbussche, D.: Solving lift-and-project relaxations of binary integer programs. SIAM J. Optim. **16**(3), 726–750 (2006)
5. Burer, S., Vandenbussche, D.: A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations. Manuscript, Department of Management Sciences, University of Iowa, Iowa City, IA, USA, June 2005. Revised April 2006 and June 2006. Math. Program. (to appear)
6. 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 (1997)
7. Giannessi, F., Tomasin, E.: Nonconvex quadratic programs, linear complementarity problems, and integer linear programs. In: Fifth Conference on Optimization Techniques, Rome, 1973, Part I. Lecture Notes in Comput. Sci., vol. 3, pp. 437–449. Springer, Berlin (1973)
8. 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 Acad., Dordrecht (2002)
9. Hansen, P., Jaumard, B., Ruiz, M., Xiong, J.: Global minimization of indefinite quadratic functions subject to box constraints. Naval Res. Logist. **40**(3), 373–392 (1993)
10. Helmberg, C.: Fixing variables in semidefinite relaxations. SIAM J. Matrix Anal. Appl. **21**(3), 952–969 (2000) (Electronic)

11. ILOG, Inc. ILOG CPLEX 9.0, User Manual (2003)
12. Kozlov, M.K., Tarasov, S.P., Khachiyan, L.G.: Polynomial solvability of convex quadratic programming. Dokl. Akad. Nauk SSSR **248**(5), 1049–1051 (1979)
13. Lovász, L., Schrijver, A.: Cones of matrices and set-functions and 0-1 optimization. SIAM J. Optim. **1**, 166–190 (1991)
14. Pardalos, P.: Global optimization algorithms for linearly constrained indefinite quadratic problems. Comput. Math. Appl. **21**, 87–97 (1991)
15. Pardalos, P.M., Vavasis, S.A.: Quadratic programming with one negative eigenvalue is NP-hard. J. Glob. Optim. **1**(1), 15–22 (1991)
16. Vandenbussche, D., Nemhauser, G.: A polyhedral study of nonconvex quadratic programs with box constraints. Math. Program. **102**(3), 531–557 (2005)
17. Vandenbussche, D., Nemhauser, G.: A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. Math. Program. **102**(3), 559–575 (2005)