Semidefinite-Based Branch-and-Bound for Nonconvex Quadratic Programming

Samuel Burer* Dieter Vandenbussche†

June 27, 2005

Abstract

This paper presents a branch-and-bound algorithm for nonconvex quadratic programming, which is based on solving semidefinite relaxations at each node of the enumeration tree. The method is motivated by a recent branch-and-cut approach for the box-constrained case that employs linear relaxations of the first-order KKT conditions. We discuss certain limitations of linear relaxations when handling general constraints and instead propose semidefinite relaxations of the KKT conditions, which do not suffer from the same drawbacks. Computational results demonstrate the effectiveness of the method, with a particular highlight being that only a small number of branch-and-bound nodes are required. Furthermore, specialization to the box-constrained case yields a state-of-the-art method for globally solving this class of problems.

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 linear constraints:

$$\begin{aligned} \max \ & \frac{1}{2} x^T Q x + c^T x \\ & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{QP}$$

where $x \in \mathbb{R}^n$ is the variable and $(Q, A, b, c) \in \mathbb{R}^{n \times n} \times \mathbb{R}^{m \times n} \times \mathbb{R}^m \times \mathbb{R}^n$ are the data. Clearly, any quadratic program (or "QP") over linear constraints can be put in the above standard form. Without loss of generality, we assume Q is symmetric. If Q is negative semidefinite,

*Department of Management Sciences, University of Iowa, Iowa City, IA 52242-1000, USA. (Email: samuel-burer@uiowa.edu). This author was supported in part by NSF Grant CCR-0203426.

†Department of Mechanical and Industrial Engineering, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801, USA. (Email: dieterv@uiuc.edu).

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

We will refer to the feasible set of (QP) as

$$P := \{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\}$$

and make the following assumptions:

Assumption 1.1. *P is nonempty and bounded. In particular, P is contained in the unit box $\{x \in \mathbb{R}^n : 0 \leq x \leq e\}$.*

Assumption 1.2. *P contains an interior point, i.e., there exists $x \in P$ such that $Ax < b$ and $x > 0$.*

Note that, if P is bounded but not inside the unit box, then Assumption 1.1 can be enforced by a simple variable scaling. Likewise, if P does not satisfy Assumption 1.2, then dimension-reduction techniques can be used to transform P into an equivalent polyhedron, which has an interior.

The difficulty in optimizing (QP) is that it harbors many local maxima. Much of the literature on nonconvex quadratic programming has focused on obtaining first-order or (weak) second-order Karush-Kuhn-Tucker (KKT) critical points, which have a good objective value, using nonlinear programming techniques such as active-set or interior-point methods. The recent survey paper by Gould and Toint (2002) provides information on such methods. On the other hand, a variety of general-purpose global optimization techniques can be tailored to solve (QP) exactly. For example, Sherali and Tuncbilek (1995) developed a Reformulation-Linearization-Technique (RLT) for solving nonconvex QPs. For a review of several global optimization methods for nonconvex QP, see the work by Pardalos (1991). Floudas and Visweswaran (1995) also present a general survey, including local and global optimization methods, various applications, and special cases. Problems of the form (QP) may also be solved using general purpose global optimization solvers, such as BARON (Sahinidis, 1996).

Semidefinite programming (or "SDP") relaxations of quadratically constrained quadratic programs (QCQPs), of which (QP) is a special case, have been studied by a number of researchers. One of the first efforts in this direction was made by Lovász and Schrijver (1991), who developed a hierarchy of SDP relaxations for 0-1 linear integer programs, where one can think of the constraint $x_i \in \{0, 1\}$ as the quadratic equality $x_i^2 = x_i$. Later, approximation guarantees using SDP relaxations were obtained for specific problems, the premiere example being the Goemans and Williamson (1995) analysis of the maximum cut problem on graphs; see also Nesterov (1998); Ye (1999). Recent work by Bomze and de Klerk (2002) has studied strong SDP relaxations of nonconvex QP over the simplex. Sherali and Fraticelli (2002) discuss the use of semidefinite cuts to improve RLT relaxations of QPs, and Anstreicher gives insight into how including semidefiniteness improves RLT relaxations. Probably one of the most general approaches for relaxing QCQPs has been proposed by Kojima and Tunçel (2000), who provide an extension of the Lovász-Schrijver approach to any optimization problem that can be expressed by a quadratic objective and (a possibly infinite number

of) quadratic constraints. To our knowledge, no one has studied in detail the specific SDP relaxations that we consider in this paper.

For the specific case, called *BoxQP*, when $(A, b) = (I, e)$ and e is the all-ones vector, there has been a considerable amount of effort towards solving (QP) exactly. One of the most successful methods is the recent work of Vandenbussche and Nemhauser (2005a,b). The key idea in their approach is to incorporate linear programming (or "LP") relaxations of the first-order KKT conditions—as well as cuts for the convex hull of first-order KKT points—in a finite branch-and-cut algorithm. The algorithm implicitly enumerates and compares all first-order KKT points to obtain a global maximum. It is important to keep in mind that the approach of Vandenbussche and Nemhauser is based heavily on the structure of BoxQP, and so it is not immediately clear how one can generalize their approach.

The goal of this paper is to expand our capabilities for solving (QP) exactly. We begin in Section 2 by considering in detail the approach of Vandenbussche and Nemhauser, where, in particular, we outline the main obstacles for extending their approach to general QPs. Then, in Section 3, we show how SDP can be used to answer one of the key questions raised in Section 2.

Motivated by this use of SDP, we turn our attention in Section 4 to developing new, tight SDP relaxations for (QP). Together with the results of Section 3, we present two different SDP relaxations of the first-order KKT conditions of (QP). The relaxations are based heavily on ideas from the work of Lovász and Schrijver (1991) mentioned above, but we incorporate important elements that reflect and exploit the specific structure of the problem at hand. Then, in Section 5, we show how these SDP relaxations can be incorporated into a finite branch-and-bound algorithm for solving (QP) exactly. An important aspect is establishing the correctness of the branch-and-bound algorithm, which turns out to be a non-trivial task. We then present detailed computational results, which demonstrate the effectiveness of our branch-and-bound algorithm. One highlight is that the algorithm uniformly requires only a small number of nodes in the branch-and-bound tree.

In Section 6, we specialize our branch-and-bound approach to BoxQP, allowing us to develop an SDP relaxation that better exploits branching information, while still maintaining a compact size. We then conduct a thorough comparison of our BoxQP algorithm with that of Vandenbussche and Nemhauser, where we demonstrate our ability to solve significantly larger BoxQP problems in still modest amounts of time.

We should remark that the current study has been motivated to a large extent by a recent work of the authors (Burer and Vandenbussche, 2004), which provides efficient computational techniques for solving the Lovász-Schrijver SDP relaxations of 0-1 integer programs. Such SDPs—including the related ones introduced in this paper—are too large to be solved with conventional SDP algorithms, and so the ability to solve these types of SDPs is an indispensable component of this paper.

1.1 Terminology and Notation

In this section, we introduce some of the notation that will be used throughout the paper. $\mathbb{R}^n$ will refer 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, and e is the vector of all ones.

$\mathbb{R}^{n\times n}$ is the set of real, $n \times n$ matrices, $\mathcal{S}^n$ is the set of symmetric matrices in $\mathbb{R}^{n\times n}$, while $\mathcal{S}^n_+$ is the set of positive semidefinite symmetric matrices. The special notation $\mathbb{R}^{1+n}$ and $\mathcal{S}^{1+n}$ is 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 $X_{\cdot j}$ and $X_{i\cdot}$ as 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 $[x \circ z]_j = x_j z_j$. diag(A) is defined as the vector with the diagonal of A as its entries.

2 Issues in Extending the Vandenbussche-Nemhauser Approach

In this section, we review a general method for reformulating (QP) as an LP with complementarity constraints (see Giannessi and Tomasin, 1973). This method was employed by Vandenbussche and Nemhauser as the basis for their branch-and-cut algorithm to solve BoxQP.

By introducing nonnegative multipliers y and z for the constraints $Ax \le b$ and $x \ge 0$ in P, respectively, any locally optimal solution x of (QP) must have the property that the sets

$$G_x := \{(y, z) \ge 0 : A^T y - z = Qx + c\}$$
$$C_x := \{(y, z) \ge 0 : (b - Ax) \circ y = 0, \quad x \circ z = 0\}$$

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 with x. The condition $G_x \cap C_x \neq \emptyset$ specifies that x is a first-order KKT point.

One can show the following property of KKT points.

Proposition 2.1. *(Giannessi and Tomasin, 1973) Suppose $x \in P$ and $(y, z) \in G_x \cap C_x$. Then $x^T Q x + c^T x = b^T y$.*

Proof. We first note that $(y, z) \in C_x$ implies $(Ax)^T y = b^T y$ and $x^T z = 0$. Next, premultiplying the equality $A^T y - z = Qx + c$ by x^T yields $x^T Q x + c^T x = (Ax)^T y - x^T z = b^T y$, as desired. □

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

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

By dropping $(y, z) \in C_x$, we obtain a natural LP relaxation:

$$\begin{aligned} \max \quad & \frac{1}{2} b^T y + \frac{1}{2} c^T x \\ \text{s.t.} \quad & x \in P \quad (y, z) \in G_x. \end{aligned} \tag{RKKT}$$

This relaxation immediately suggests a LP-based finite branch-and-bound approach, where the complementarity constraints, $(y, z) \in C_x$, are recursively enforced through branching. However, there is a fundamental problem with such an approach, namely that (RKKT) has an unbounded objective, as we detail in the next proposition and corollary.

Proposition 2.2. *If P is bounded, then the set*

$$R := \{(\Delta y, \Delta z) \geq 0 : A^T \Delta y - \Delta z = 0\}$$

contains nontrivial points. Moreover:

- *$b^T \Delta y \geq 0$ for all $(\Delta y, \Delta z) \in R$; and*
- *if P has an interior, then $b^T \Delta y > 0$ for all nonzero $(\Delta y, \Delta z) \in R$.*

Proof. To prove both parts of the proposition, we consider the primal LP $\max \{d^T x : x \in P\}$ and its dual $\min \{b^T y : A^T y - z = d, (y, z) \geq 0\}$ for a specific choice of d.

Let $d = e$. Because P is bounded, the dual has a feasible point (y, z). It follows that $(\Delta y, \Delta z) := (y, z + e)$ is a nontrivial element of R.

Next let $d = 0$, and note that the dual feasible set equals R, which immediately implies the second statement of the proposition.

To prove the third statement, keep $d = 0$ and let x^0 be an interior-point of P. Note that x^0 is optimal for the primal. Complementary slackness thus implies that $(0, 0)$ is the unique optimal solution of the dual, which proves the result. □

Corollary 2.3. *(RKKT) has an unbounded objective value.*

Proof. Recall that P is bounded and has a nonempty interior by Assumptions 1.1 and 1.2. We first note that R defined in Proposition 2.2 is the recession cone of G_x (for arbitrary x). Hence, (RKKT) contains a nontrivial direction of recession $(\Delta y, \Delta z)$ in the variables (y, z) such that $b^T \Delta y > 0$, which proves that it has an unbounded objective value. □

From this corollary, it would seem that branch-and-bound for (QP) based on LP relaxations like (RKKT) is not even a well-defined approach. However, Vandenbussche and Nemhauser used precisely this approach for solving BoxQP, providing the best known computational results for this special case of QP thus far. There are two main reasons for the success of their algorithm:

- They were able to address the unboundedness of (RKKT) by exploiting the structure of BoxQP to reveal valid upper bounds on $(y, z) \in G_x \cap C_x$, which could then be incorporated in G_x, effectively bounding (RKKT). In the specific case of BoxQP, we have

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

Hence, 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

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

- They developed several classes of valid inequalities for $G_x \cap C_x$ in the BoxQP case, i.e., linear constraints satisfied by all $(y,z) \in G_x \cap C_x$, that tightened the LP relaxation considerably and could also be easily separated. The aggressive use of these inequalities within a branch-and-cut approach yielded significant computational gains.

The above discussion and the successful methodology of Vandenbussche and Nemhauser strongly suggest that any LP-based approach for solving (QP) based on (RKKT) must somehow address the issue of unboundedness. Moreover, it is our opinion that solving the boundedness problem is not sufficient—one must also incorporate strong cuts. Vandenbussche and Nemhauser were successful on both counts in the case of BoxQP, but it is not clear how their techniques can be generalized to (QP).

2.1 Further results concerning unboundedness

We continue now to examine the unboundedness issue for general QP in hopes of shedding some additional light on the subject. (We will also study the problem from the perspective of SDP in the next section.)

Even though (RKKT) has an unbounded objective value, it is interesting to observe that movement in the direction of an extreme ray, which causes the unbounded objective, also causes a violation of the complementarity constraints C_x. We formalize this by showing that the convex hull of complementary solutions (x,y,z) consists of the convex hull of extreme points of (RKKT) satisfying complementarity and the cone of recession directions $(\Delta x, \Delta y, \Delta z)$ of (RKKT) satisfying $b^T \Delta y = 0$. More precisely, define

$$S := \{(x,y,z) \ : \ x \in P, \ (y,z) \in G_x \cap C_x\}$$

to be the feasible set of (KKT) and

$$\mathcal{C} := \{(0, \Delta y, \Delta z) \geq 0 \ : \ (\Delta y, \Delta z) \in R\}$$

to be the recession cone of (RKKT). (Here, we have used the Assumption 1.1 that P is bounded, and R is the recession cone of G_x as defined in Proposition 2.2). Also denote by V_c^{LP} the set of extreme points (x,y,z) of (RKKT) satisfying $(y,z) \in C_x$; we remark that $V_c^{LP} \subseteq S$.

Proposition 2.4. *The following equality holds:*

$$\operatorname{conv}(S) = \operatorname{conv}(V_c^{LP}) + \left\{(0, \Delta y, \Delta z) \in \mathcal{C} \ : \ b^T \Delta y = 0\right\}$$

Proof. ($\supseteq$) We show

$$S \supseteq V_c^{LP} + \{(0, \Delta y, \Delta z) \in \mathcal{C} : b^T \Delta y = 0\}$$

from which the required inclusion immediately follows. Let $(x, y, z) \in V_c^{LP}$ and $(0, \Delta y, \Delta z) \in \mathcal{C}$ such that $b^T \Delta y = 0$. Recall that $(x, y, z) \in V_c^{LP}$ implies $(y, z) \in C_x$. We wish to show $(\bar{x}, \bar{y}, \bar{z}) := (x, y + \Delta y, z + \Delta z) \in S$. Clearly, $\bar{x} \in P$ and $(\bar{y}, \bar{z}) \in G_{\bar{x}}$, and so it remains to show $(\bar{y}, \bar{z}) \in C_{\bar{x}}$, which is equivalent to $\bar{z}^T \bar{x} + \bar{y}^T (b - A\bar{x}) = 0$ since all the components of $\bar{x}$, $\bar{z}$, $\bar{y}$, and $b - A\bar{x}$ are nonnegative. We have that

$$\begin{aligned}
\bar{z}^T \bar{x} + \bar{y}^T (b - A\bar{x}) &= (z + \Delta z)^T x + (y + \Delta y)^T (b - Ax) \\
&= z^T x + y^T (b - Ax) + x^T (\Delta z - A^T \Delta y) + b^T \Delta y \\
&= x^T (\Delta z - A^T \Delta y) + b^T \Delta y && [\text{since } (y, z) \in C_x] \\
&= b^T \Delta y && [\text{since } (\Delta y, \Delta z) \in R].
\end{aligned}$$

The result now follows from the assumption $b^T \Delta y = 0$.

($\subseteq$) We prove

$$S \subseteq \operatorname{conv}(V_c^{LP}) + \{(0, \Delta y, \Delta z) \in \mathcal{C} : b^T \Delta y = 0\}$$

from which the desired inclusion follows by taking the convex hull of both sides. Let V^{LP} denote the complete set of extreme points of (RKKT), i.e., not just those satisfying complementarity as with V_c^{LP}. Because S is contained in the feasible set of (RKKT), it follows that

$$S \subseteq \operatorname{conv}(V^{LP}) + \mathcal{C},$$

i.e., any point $(x, y, z) \in S$ can be expressed as

$$(x, y, z) = \sum_j \lambda_j (x^j, y^j, z^j) + (0, \Delta y, \Delta z)$$

for some finite subset $\{(x^j, y^j, z^j)\} \subseteq V^{LP}$, some $(0, \Delta y, \Delta z) \in \mathcal{C}$, and some convex combination λ.

Recall that Proposition 2.2 implies $b^T \Delta y \geq 0$. We claim that in fact $b^T \Delta y = 0$. Because $(y, z) \in C_x$ and $(\Delta y, \Delta z) \in R$, we have

$$\begin{aligned}
0 &= (b - Ax)^T y + x^T z \\
&= (b - Ax)^T \left(\sum_j \lambda_j y^j + \Delta y \right) + x^T \left(\sum_j \lambda_j z^j + \Delta z \right) \\
&= b^T \Delta y + x^T \left(\Delta z - A^T \Delta y \right) + (b - Ax)^T \left(\sum_j \lambda_j y^j \right) + x^T \left(\sum_j \lambda_j z^j \right) \\
&= b^T \Delta y + (b - Ax)^T \left(\sum_j \lambda_j y^j \right) + x^T \left(\sum_j \lambda_j z^j \right).
\end{aligned}$$

Since all three terms in the final expression are nonnegative, it follows that $b^T \Delta y = 0$. We next claim further that each (x^j, y^j, z^j) is in V_c^{LP}; we must show $(b - Ax^j)^T y^j + (x^j)^T z^j = 0$. The proof continues the above equations:

$$\begin{aligned} 0 &= (b - Ax)^T \left(\sum_j \lambda_j y^j \right) + x^T \left(\sum_j \lambda_j z^j \right) \\ &= \left(\sum_j \lambda_j (b - Ax^j) \right)^T \left(\sum_j \lambda_j y^j \right) + \left(\sum_j \lambda_j x^j \right)^T \left(\sum_j \lambda_j z^j \right) \\ &= \sum_j \lambda_j^2 \left[(b - Ax^j)^T y^j + (x^j)^T z^j \right] + 2 \sum_{j<i} \lambda_j \lambda_i \left[(b - Ax^j)^T y^i + (x^j)^T z^i \right]. \end{aligned}$$

Again, since all terms are nonnegative, we have $(b - Ax^j)^T y^j + (x^j)^T z^j = 0$, as desired. □

In particular, Proposition 2.4 shows that $\text{conv}(S)$ is a polyhedron, which is not immediately obvious since S can be described as the union of a finite number of possibly unbounded polyhedra. The proposition also shows that, while (RKKT) has an unbounded objective value, the LP obtained from maximizing $(c^T x + b^T y)/2$ over $\text{conv}(S)$ has a bounded objective value. In fact, it guarantees that we may equivalently optimize over just $\text{conv}(V_c^{LP})$ without changing the optimal objective value. Unfortunately, this still does not lead to a tractable relaxation.

Still, one might be inclined to try to optimize over just the vertices of (RKKT), that is, over the set V^{LP} defined in the above proof. Unfortunately, this is also difficult in general.

Proposition 2.5. *Given a linear program as a polynomial-time separation oracle, optimizing an arbitrary linear function over its vertices is $\mathcal{NP}$-hard.*

Proof. Consider the dominant of the convex hull of incidence vectors of $s-t$ paths of a graph $G = (N, E)$ (see Nemhauser and Wolsey, 1988):

$$\left\{ x \in \mathbb{R}_+^{|E|} : a_H^T x \geq 1 \quad \forall \quad H \in \mathcal{H} \right\},$$

where $\mathcal{H}$ is the set of $s - t$ cuts and a_H is the incidence vector of a cut $H \in \mathcal{H}$. Optimizing an arbitrary linear objective over the vertices of this polyhedron corresponds to solving the longest path problem, which is $\mathcal{NP}$-hard (see Garey and Johnson, 1979). □

Finally, it is indeed possible to obtain a tractable and bounded tightening of (RKKT), without cutting off any of its vertices, by making use of a result that relates the size of the inequalities that define a polyhedron to the size of its vertices and extreme rays. Define φ as the maximum size of any of the inequalities defining a polyhedron Π and denote ν as the maximum size of any vertex or extreme ray of Π.

Theorem (Theorem 10.2 in Schrijver (1986))**.** *Suppose Π is a rational polyhedron in $\mathbb{R}^n$. Then $\varphi \leq 4n^2\nu$ and $\nu \leq 4n^2\varphi$.*

Hence, if $w \in \mathbb{R}^n$ is a vertex of Π, then $\text{size}(w) = \sum_i \text{size}(w_i) \leq 4n^2\varphi$. Using the usual definition of size, we have that $\text{size}(w_i) = \lceil \log_2(|p_i| + 1) \rceil + \lceil \log_2(q_i + 1) \rceil + 1 \leq 4n^2\varphi$, where $w_i = p_i/q_i$ and $p_i \in \mathbb{Z}$ and $q_i \in \mathbb{N}$ are relatively prime. We may now conclude that $|w_i| \leq |p_i| \leq \mathcal{O}(2^{4n^2\varphi})$. Hence, given an arbitrary polyhedron in inequality form, we may obtain bounds on the components w_i of any vertex w of that polyhedron. While this does provide a way to bound the variables in (RKKT), this bound is clearly very weak and would most likely perform very poorly in practice.

In the next section, we show how one can obtain valid upper bounds on all the dual multipliers through the use of SDP relaxations of (QP).

3 Bounding the Lagrange Multipliers Using SDP

We have discussed in Section 2 how bounding the Lagrange multipliers (y, z) in (RKKT) becomes an issue when attempting to extend the work of Vandenbussche and Nemhauser to problems more general than BoxQP. We now show how this obstacle can be overcome by an application of semidefinite programming.

We first introduce a basic SDP relaxation of (QP). Our motivation comes from the SDP relaxations of 0-1 integer programs introduced by Lovász and Schrijver (1991). Consider a matrix variable Y, which is related to $x \in P$ by the following quadratic equation:

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

From (1), 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 constraints $Ax \leq b$ of P by some x_i, we obtain the quadratic inequalities $b\, x_i - Ax\, x_i \geq 0$, which are valid for P; these inequalities can be written in terms of Y as
$$\big(b\big| - A\big)\, Y e_i \geq 0 \quad \forall \quad i = 1, \ldots, n.$$
- the objective function of (QP) 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

$$K := \big\{(x_0, x) \in \mathbb{R}^{1+n} \;:\; Ax \leq x_0\, b,\; (x_0, x) \geq 0\big\}$$

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 $Ye_i \in K$ and $x^TQx/2 + c^Tx = \tilde{Q} \bullet Y$. In addition, we let M_+ denote the set of all Y satisfying the first and second properties:

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

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

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

Finally, by dropping the last n columns of the equation (1), we arrive at the following linear SDP relaxation of (QP):

$$\begin{aligned} \max \quad & \tilde{Q} \bullet Y \\ \text{s.t.} \quad & Y \in M_+ \quad Ye_0 = (1; x) \\ & x \in P. \end{aligned} \tag{SDP$_0$}$$

We next combine this basic SDP relaxation with the LP relaxation (RKKT) introduced in Section 2 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:

$$\begin{aligned} \max \quad & \tilde{Q} \bullet Y \\ \text{s.t.} \quad & Y \in M_+ \quad Ye_0 = (1; x) \\ & x \in P \quad (y, z) \in G_x \\ & \tilde{Q} \bullet Y = (b^Ty + c^Tx)/2. \end{aligned} \tag{SDP$_1$}$$

$$\tilde{Q} \bullet Y = (b^Ty + c^Tx)/2. \tag{2}$$

This optimization problem is a valid relaxation of (KKT) if the constraint (2) 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 (1); then (Y, x, y, z) satisfies the first four constraints of (SDP$_1$) by construction and the constraint (2) is satisfied due to Proposition 2.1 and (1).

We will study the relaxation (SDP$_1$) in more detail in the following sections, but we would like to point out an interesting property of the feasible set of (SDP$_1$), namely that it is bounded even though the feasible set of (RKKT) is not.

Proposition 3.1. *The feasible set of (SDP$_1$) is bounded.*

Proof. We first argue that the feasible set of (SDP$_0$) is bounded. By Assumption 1.1, the variable x is bounded, and hence Ye_0 is as well. Because Y is symmetric, this in turn implies that the 0-th row of Y is bounded. Now consider $Ye_i \in K$ for $i \geq 1$. Since we know that the 0-th component of Ye_i is bounded, the definition of K and Assumption 1.1 imply that Ye_i is itself bounded. In total, we have that Y is bounded.

We now show that the recession cone of the feasible set of (SDP_1) is trivial. Using the definition of R in Proposition 2.2 as well as the result of the previous paragraph, the recession cone is

$$\left\{(\Delta Y, \Delta x, \Delta y, \Delta z) : (\Delta Y, \Delta x) = (0,0),\ (\Delta y, \Delta z) \in R,\ b^T \Delta y = 0\right\}.$$

However, Assumption 1.2 and Proposition 2.2 imply $b^T \Delta y > 0$ for all nontrivial $(\Delta y, \Delta z) \in R$. So the recession cone is trivial. □

With Proposition 3.1 in hand, we now can compute bounds on each y_i by simply maximizing y_i over the feasible set of (SDP_1), and similarly for z_i. This could be used, for example, as a preprocessing technique for bounding the Lagrange multipliers when extending the method of Vandenbussche and Nemhauser to general QP. We stress, however, that their method was successful not only because it was able to bound the multipliers (y, z), but also because it incorporated many strong cuts. Nevertheless, we find it interesting that one can address the key question of bounds on (y, z) be appealing to SDP relaxations of (KKT).

4 SDP Relaxations for Quadratic Programs

Inspired by the use of (SDP_1) to bound the Lagrange multipliers (y, z), we now focus solely on the use of SDP to solve (QP). In particular, we will no longer consider LP relaxations, e.g., (RKKT) with bounded (y, z).

Comparing (SDP_0) with (SDP_1), one immediate advantage of (SDP_1) is that the explicit appearance of (y, z) suggests a finite branch-and-bound approach for (KKT), in which the complementarity constraints are enforced through branching. Even though (SDP_0) is a more compact relaxation, there is no clear way to incorporate complementarity information. As a result, a branch-and-bound algorithm based on (SDP_0) would have to employ other techniques, such as those found in general global optimization approaches.

Of course, even with the advantage of finite branch-and-bound, a natural question arises: how much tighter is (SDP_1) than (SDP_0)? Although we have been unable to obtain a precise answer to this question, some simple comparisons of the two relaxations reveal interesting relationships, as we describe next.

For each $x \in P$ and any H_x such that $G_x \cap C_x \subseteq H_x \subseteq G_x$, define

$$\delta(x, H_x) := \frac{1}{2} \min\left\{b^T y : (y, z) \in H_x\right\} + \frac{1}{2} c^T x.$$

Based on $\delta(x, H_x)$ with $H_x := G_x$, we can characterize those (Y, x), which are feasible for (SDP_0) but not for (SDP_1).

Proposition 4.1. *Let (Y, x) be feasible for (SDP_0). Then there exists $(y, z) \in G_x$ such that (Y, x, y, z) is feasible for (SDP_1) if and only if $\tilde{Q} \bullet Y \geq \delta(x, G_x)$.*

Proof. First consider the case when $\tilde{Q} \bullet Y < \delta(x, G_x)$, which implies $\tilde{Q} \bullet Y < (b^T y + c^T x)/2$ for all $(y, z) \in G_x$. Hence, (Y, x) cannot be feasible for (SDP_1).

When $\tilde{Q} \bullet Y \geq \delta(x, G_x)$, we know that $\tilde{Q} \bullet Y \geq (b^T y + c^T x)/2$ for at least one $(y, z) \in G_x$, namely the optimal solution defining $\delta(x, G_x)$. Using Proposition 2.2 along with Assumptions

1.1 and 1.2, we can adjust (y, z) so that $b^T y$ increases to satisfy (2), while maintaining $(y, z) \in G_x$. □

This proposition essentially demonstrates that the added variables and constraints, which give (SDP$_1$) from (SDP$_0$), are effective in reducing the feasible set when $\delta(x, G_x)$ is greater than $\tilde{Q} \bullet Y$ for a large portion of (Y, x).

From a practical standpoint, we have actually never observed a situation in our test problems for which the optimal value of (SDP$_1$) is strictly lower than that of (SDP$_0$). We believe this is evidence that the specific function $\delta(x, G_x)$ is rather weak, i.e., it is often less than or equal to $\tilde{Q} \bullet Y$. Nevertheless, the perspective provided by Proposition 4.1 gives insight into how one can tighten the SDP relaxations. Roughly speaking, the proposition tells us there are two basic ways to tighten:

(i) lower $\tilde{Q} \bullet Y$, that is, tighten the basic SDP relaxation (SDP$_0$) so that (Y, x) better approximates (1);

(ii) raise $\delta(x, G_x)$, that is, replace G_x in the KKT relaxation (SDP$_1$) with an H_x that better approximates $G_x \cap C_x$; note that $H_1 \subseteq H_2$ implies $\delta(x, H_1) \geq \delta(x, H_2)$ for all $x \in P$.

We will see in the next section that enforcing complementarities in branch-and-bound serves to accomplish both (i) and (ii), and this branching significantly tightens the SDP relaxations as evidenced by a small number of nodes in the branch-and-bound tree. In addition, we will introduce below a second SDP relaxation of (KKT) that is constructed in the spirit of (i) and (ii).

Before we describe the next relaxation, however, we would like to make a comment about the specific constraint (2) in (SDP$_1$), which equates the objectives of (SDP$_0$) and (RKKT). Though this constraint is key in bounding the feasible set of (SDP$_1$) and in developing the perspective given by Proposition 4.1, we have found in practice that it does not tighten the SDP relaxation unless many complementarity constraints have been fixed through branching. Moreover, SDPs that do not include (2) can often be solved much more quickly than ones that do. So, while (2) is an important constraint theoretically (we will discuss additional properties below and in Section 5), we have found that it can be sometimes practically advantageous to avoid. We should remark, however, that, to maintain boundedness, dropping (2) does require one to include valid bounds for (y, z)—bounds that should be computed in a preprocessing phase that explicitly incorporates (2) (see the discussion in Section 2). We will discuss this in more detail in the next section.

The SDP that we introduce next relaxes (KKT) by handling the quadratic constraints $(y, z) \in C_x$ directly. We follow the discussion at the beginning of Section 2, except this time we focus on the variables (x, y, z) instead of just x. Let $\hat{Y}$ be related to (x, y, z), which is a feasible solution of (KKT), by the following quadratic equation:

$$\hat{Y} = \begin{pmatrix} 1 \\ x \\ y \\ z \end{pmatrix} \begin{pmatrix} 1 \\ x \\ y \\ z \end{pmatrix}^T = \begin{pmatrix} 1 & x^T & y^T & z^T \\ x & xx^T & xy^T & xz^T \\ y & yx^T & yy^T & yz^T \\ z & zx^T & zy^T & zz^T \end{pmatrix}. \tag{3}$$

Defining $\hat{n} := n + m + n$,

$$\hat{K} := \left\{(x_0, x, y, z) \in \mathbb{R}^{1+\hat{n}} \ : \ Ax \leq x_0\, b,\ A^T y - z = Qx + x_0\, c,\ (x_0, x, y, z) \geq 0\right\},$$

and

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

and letting $\hat{Y}_{xy}$ and $\hat{Y}_{xz}$ denote the xy- and xz-blocks of $\hat{Y}$, respectively, we observe the following:

- $\hat{Y} \in \mathcal{S}_+^{1+\hat{n}}$ (or simply $Y \succeq 0$);
- $\hat{Y} e_i \in \hat{K}$ for all $i = 1, \ldots, \hat{n}$;
- using Proposition 2.1, the objective of (KKT) can be expressed as $\hat{Q} \bullet \hat{Y}$;
- similar to (SDP_1), the equation $\hat{Q} \bullet \hat{Y} = (b^T y + c^T x)/2$ is satisfied;
- the condition $(y, z) \in C_x$ is equivalent to $\operatorname{diag}(A\hat{Y}_{xy}) = b \circ y$ and $\operatorname{diag}(\hat{Y}_{xz}) = 0$.

We let $\hat{M}_+$ denote the set of all $\hat{Y}$ satisfying the first and second properties:

$$\hat{M}_+ := \left\{\hat{Y} \succeq 0 : \hat{Y} e_i \in \hat{K} \ \ \forall \ \ i = 1, \ldots, \hat{n}\right\}.$$

By dropping the last $\hat{n}$ columns of the equation (3), we arrive at the following relaxation:

$$\begin{array}{lll} \max & \hat{Q} \bullet \hat{Y} & \qquad (\text{SDP}_2) \\ \text{s.t.} & \hat{Y} \in M_+ \quad \hat{Y} e_0 = (1; x; y; z) & \\ & x \in P \quad (y, z) \in G_x & \\ & \hat{Q} \bullet \hat{Y} = (b^T y + c^T x)/2 & \\ & \operatorname{diag}(A\hat{Y}_{xy}) = b \circ y \quad \operatorname{diag}(\hat{Y}_{xz}) = 0. & \end{array}$$

Clearly this relaxation is at least as strong as (SDP_1), though it is much bigger. We can see its potential for strengthening (SDP_1) through the perspective of (i) and (ii) above. By lifting and projecting with respect to (x, y, z) and the linear constraints $x \in P$, $(y, z) \in G_x$, we accomplish (i), namely we tighten the relationship between the original variables and the positive semidefinite variable. By incorporating the "diag" constraints, which represent complementarity, we are indirectly tightening the constraint $(y, z) \in G_x$ by reducing it to a smaller one that better approximates $G_x \cap C_x$. The computational results will confirm the strength of (SDP_2) over (SDP_1).

We end this section with an interesting property of the two relaxations (SDP_1) and (SDP_2), which will become especially relevant in the next section, when we discuss branch-and-bound.

Proposition 4.2. *Suppose (x, y, z) is part of an optimal solution for (SDP$_1$) or (SDP$_2$). If $(y, z) \in C_x$, then x is an optimal solution of (QP).*

Proof. We prove the result for (SDP$_1$) as the proof for (SDP$_2$) is similar. Let (Y, x, y, z) be an optimal solution containing (x, y, z); in particular, (2) holds, i.e.,

$$\tilde{Q} \bullet Y = \frac{1}{2} b^T y + \frac{1}{2} c^T x.$$

Because $(y, z) \in G_x \cap C_x$, Proposition 2.1 implies $(b^T y + c^T x)/2 = x^T Q x/2 + c^T x$. So $\tilde{Q} \bullet Y$ equals the function value at x. Since $\tilde{Q} \bullet Y$ is an upper bound on the optimal value of (QP), it follows that x is an optimal solution. □

5 The SDP-Based Branch-and-Bound Algorithm

In this section, we discuss the generic branch-and-bound framework that we use to solve (QP) based on (SDP$_1$) and (SDP$_2$) and then provide a detailed description of the implementation as well as computational results on a set of test problems.

5.1 Branch-and-bound framework

In previous sections, we have alluded to how one can construct a branch-and-bound algorithm for (QP) by recursively enforcing complementarities through branching. We now describe this algorithm explicitly and discuss how to incorporate (SDP$_1$) and (SDP$_2$).

We first describe a typical node in the branch-and-bound tree. A node is specified by four index sets $F^x, F^z \subseteq \{1, \ldots, n\}$ and $F^{b-Ax}, F^y \subseteq \{1, \ldots, m\}$ such that $F^x \cap F^z = \emptyset$ and $F^{b-Ax} \cap F^y = \emptyset$. F^x gives those indices j for which the equality $x_j = 0$ is enforced at the node; the other sets are defined similarly, e.g., $A_{i\cdot} x = b_i$ is enforced for all $i \in F^{b-Ax}$. By enforcing these linear equalities, we guarantee complementarity: $x_j z_j = 0$ for all $j \in F^x \cup F^z$, and $(b_i - A_{i\cdot} x) y_i = 0$ for all $i \in F^{b-Ax} \cup F^y$. The root node in the tree has all four sets empty, and a leaf node is specified by sets satisfying $F^x \cup F^z = \{1, \ldots, n\}$ and $F^{b-Ax} \cup F^y = \{1, \ldots, m\}$.

At a particular node of the tree, the branch-and-bound algorithm solves a relaxation of the following optimization, which is (KKT) with the additional linear constraints specified by F^x, F^z, F^{b-Ax}, F^y:

$$\begin{array}{lll} \max & \frac{1}{2} b^T y + \frac{1}{2} c^T x & \\ \text{s.t.} & x \in P & (y, z) \in G_x \cap C_x \\ & x_j = 0 & j \in F^x \\ & z_j = 0 & j \in F^z \\ & A_{i\cdot} x = b_i & i \in F^{b-Ax} \\ & y_i = 0 & i \in F^y. \end{array} \tag{4}$$

In our case, the types of relaxations that we will deal with have the following useful property: the relaxation is infeasible if and only if (4) is infeasible, and if the relaxation is feasible, each

solution corresponds naturally to a solution of (4). When feasible, let (x^*, y^*, z^*) correspond to the optimal solution obtained from the relaxation.

If possible, we would like to fathom the node (i.e., eliminate it from further consideration), which is only allowable if we can guarantee that (4) contains no optimal solutions of (QP), other than possibly (x^*, y^*, z^*). Such a guarantee can be obtained in three ways. First, if the relaxation is infeasible, then (4) is infeasible so that it contains no optimal solutions. Second, we can fathom if (x^*, y^*, z^*) is the optimal solution of (4), which is verified only if the relaxed objective equals the objective of (4) at (x^*, y^*, z^*), i.e., the relaxation has no gap. In this case, we fathom but keep a record of (x^*, y^*, z^*) as a possible candidate for the global optimal solution. Third, we can fathom if the relaxed objective value at (x^*, y^*, z^*) is below the true objective of (QP) at some previously encountered candidate.

If we cannot fathom, then we must branch. Branching on a node involves selecting some $j \in \{1, \ldots, n\} \setminus (F^x \cup F^z)$ or some $i \in \{1, \ldots, m\} \setminus (F^{b-Ax} \cup F^y)$ and then creating two child nodes, one with $j \in F^x$ and one with $j \in F^z$ (if a j was selected), or one with $i \in F^{b-Ax}$ and one with $i \in F^y$ (if an i was selected). If we are at a leaf node, then we are unable to branch.

Accordingly, to have a correct branch-and-bound algorithm, we must be able to fathom all leaf nodes (since we cannot branch on them). In other words, it must be the case that (when feasible) the relaxation has no gap at a leaf node. For many types of relaxations, the condition that the relaxation has no gap is equivalent to $(y^*, z^*) \in C_{x^*}$, which is enforced at a leaf (by definition). When such relaxations are employed in branch-and-bound, they automatically guarantee the correctness of the algorithm.

In total, the algorithm starts at the root node, and continues evaluating nodes, fathoming nodes, and branching on nodes until no more branching occurs. Assuming correctness, the algorithm will terminate after a finite number of nodes, and upon termination, the candidate (x^*, y^*, z^*) with the highest objective value is a global optimal solution of (QP).

We propose to use relaxations of the type (SDP_1) and (SDP_2) in branch-and-bound. More specifically, we use these as the basis for relaxation at each node, where the only modification is the inclusion of the linear equalities represented by F^x, F^z, F^{b-Ax}, and F^y into the definitions of P, G_x, K, and $\hat{K}$ in the natural way. For example, at any node, we define

$$\hat{K} := \left\{ (x_0, x, y, z) \in \mathbb{R}^{1+\hat{n}} \;:\; \begin{array}{cccc} Ax \leq x_0\, b, & A^T y - z = Qx + x_0\, c, & (x_0, x, y, z) \geq 0 \\ A_{i\cdot} x = x_0\, b_i & \forall\, i \in F^{b-Ax} & y_i = 0 & \forall\, i \in F^y \\ x_j = 0 & \forall\, j \in F^x & z_j = 0 & \forall\, j \in F^z \end{array} \right\}.$$

But do the relaxations (SDP_1) and (SDP_2) guarantee the correctness of the branch-and-bound algorithm as discussed above? The following simple modification of Proposition 4.2 provides a positive answer.

Proposition 5.1. *Consider a node of the branch-and-bound tree, and suppose (x, y, z) is part of an optimal solution for (SDP_1) and (SDP_2) (appropriately tailored to incorporate the forced equalities at the node). If $(y, z) \in C_x$, then (x, y, z) is an optimal solution of (4).*

It is important to note that the above proposition strongly uses the objective constraint (2). For purely practical advantage, we will drop this constraint in the computational experi-

ments. Fortunately, there is a simple strategy to guarantee correctness, as we explain in the next subsection.

5.2 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 mentioned in the introduction, we have chosen to use the method proposed by Burer and Vandenbussche (2004) for solving the SDP relaxations because of its applicability and scalability for SDPs of the type (SDP_1) and (SDP_2). 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 $\hat{K}$. In particular, constraints such as (2) 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*.

Recall that the boundedness of (y, z) in (SDP_1) and (SDP_2) relies on the equality constraint (2). So, in principle, relaxing this constraint with an unrestricted multiplier λ can cause problems for the AL method because its subproblems may have unbounded objective value corresponding to $b^T y \to \infty$. (This behavior was actually observed in practice.) Hence to ensure that the subproblems in the AL method remain bounded, one must restrict $\lambda \leq 0$ so that the term $\lambda\, b^T y$ appears in the subproblem objective. In fact, it is not difficult to see that this is a valid restriction on λ.

Early computational experience demonstrated that (2) often slows down the solution of the SDP subproblems, mainly due to the fact that it induces certain dense linear algebra computations in the AL subproblems. On the other hand, removing this constraint completely from (SDP_1) and (SDP_2) had little negative effect on the quality of the relaxations, especially at the top levels of the branch-and-bound tree. As pointed out in Section 4, (2) is unlikely to have any effect until many complementarities have been fixed, that is, until H_x gets close to $G_x \cap C_x$. Of course, it is theoretically unwise to ignore (2) because both the boundedness of (y, z) and the correctness of branch-and-bound depend on it. (We were able to generate an example of a leaf node that would not fathomed if (2) was left out.)

It is nevertheless possible to drop (2) in the branch-and-bound relaxations and still manage the boundedness and correctness issues:

- In a preprocessing phase, we can include (2) and compute bounds on (y, z) using the results from Section 3. However, rather than solving $n + m$ different SDPs to obtain individual bounds on the components of y and z, we solve just two, with objectives $e^T y$ and $e^T z$, which yield valid upper bounds for y_i and z_j, respectively, since $y, z \geq 0$. We also solve these two SDPs to a loose accuracy.

- At the leaf nodes, instead of (SDP_1) or (SDP_2), we substitute a straightforward LP relaxation of (4), which correctly fathoms leaf nodes. (In actuality, in our computations branch-and-bound completed before reaching the leaf nodes, and so we actually never had to invoke this LP relaxation.)

So we chose to remove (2) from all calculations except the preprocessing for upper bounds on (y, z). Excluding the constraint usually did not result in an increase in the number of nodes in the branch-and-bound tree, and so we were able to realize a signficant reduction in overall computation time.

To further expedite the branch-and-bound process, we also attempted to tighten (SDP_1) by adding constraints of the form $Y(e_0 - e_i) \in K$, which are valid since we have assumed that $P \subseteq [0, 1]^n$. Note that while the constraint $x \leq e$ may be redundant for P, the constraints $Y(e_0 - e_i) \in K$, which are based on $x \leq e$, are generally not redundant for the SDP. 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.

In the case of (SDP_2), we can add similar constraints. Assuming that bounds for (y, z) have already been computed as described above, we can then rescale these variables so that they, like x, are also contained in the unit box. We now may add $\hat{Y}(e_0 - e_i) \in \hat{K}$ to tighten (SDP_2).

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

- Complementarities to branch on were chosen using a maximum normalized violation approach. Given a primal solution (x^*, y^*, z^*) and slack variables $s^* := b - Ax^*$ obtained from a relaxation, we computed $\text{argmax}_j\{x_j^* z_j^*\}$ and $\text{argmax}_i\{s_i^* y_i^* / \bar{s}_i\}$, where $\bar{s}_i$ is an upper bound on the slack variable that has been computed *a priori* in a preprocessing phase (in the same way as bounds for (y, z) are computed). Recall that x, y, and z are already scaled to be in the unit interval and hence the violations computed are appropriately normalized. We branch on the resulting index (i or j) corresponding to the highest normalized violation.

- After solving a relaxation, we use x^* as a starting point for a QP solver based on nonlinear programming techniques to obtain a locally optimal solution to (QP). In this way, we obtain good lower bounds that can be used to fathom nodes in the tree.

- We used a best bound strategy for selecting the next node to solve in the branch-and-bound tree.

- At the termination 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 warmstart, which we found extremely helpful in practice.

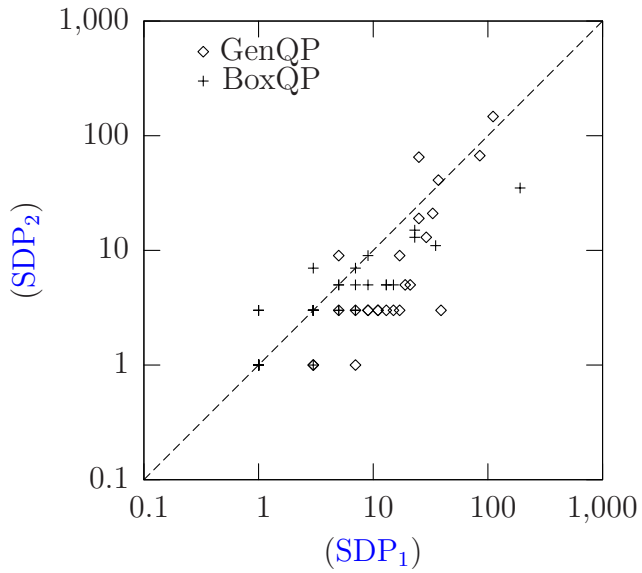

Figure 1: Number of nodes (log-log scale) required for (SDP_1) and (SDP_2) without optimality tolerance on 54 box-constrained and 24 general QP instances.

5.3 Computational results

We tested the two SDP relaxations within branch-and-bound on two types of instances. First, we used it to solve the 54 box-constrained instances of (QP) that were introduced by Vandenbussche and Nemhauser (2005b). These instances range in size from 20 variables to 60 variables and have varying densities for the matrix Q. We also randomly generated 24 interior-feasible QPs with general constraints $Ax \leq b$ and the constraints $0 \leq x \leq e$ to ensure boundedness. These instances range in size from 10 to 20 variables and 1 to 20 general constraints and have fully dense Q and A matrices.

All these instances were solved using both (SDP_1) and (SDP_2) as the SDP relaxations on a Pentium 4 2.4 GHz under the Linux operating system. (The same testing environment is also used for the computations in Section 6.) In Figure 1, we compare the number of nodes required for both relaxation types when no optimality tolerance was used for fathoming nodes. Not surprisingly, (SDP_2) generally requires fewer nodes as it should be a tighter relaxation.

In Figure 2, we compare CPU times for the same runs. The (SDP_2) subproblems took significantly longer to solve, which is to be expected as they are larger and more complex than those encountered with (SDP_1). We should note here that it is possible to tweak the implementation so that the subproblems are not solved to a high level of accuracy, especially at nodes with small depth in the branch-and-bound tree. This way you may enumerate a few more nodes but may save in total CPU time.

In practice, it is common to use a positive optimality tolerance for fathoming, unlike the runs just discussed, which use no optimality tolerance. In Figures 3 and 4, we show plots comparing results on the same instances when using a 1% relative optimality tolerance in

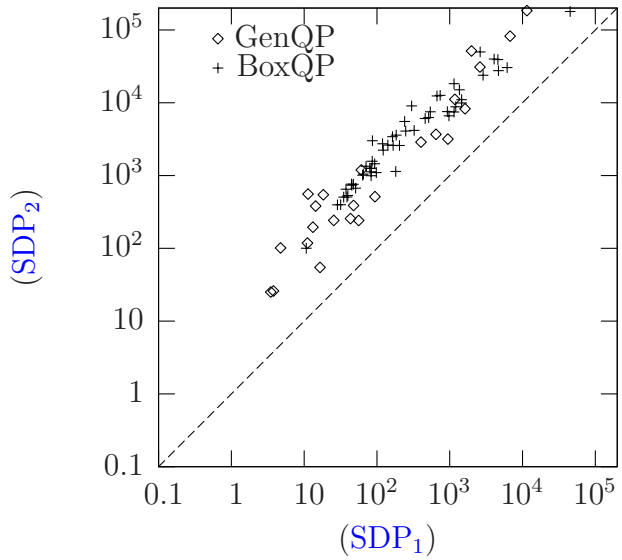

Figure 2: CPU times (log-log scale) required for (SDP$_1$) and (SDP$_2$) without optimality tolerance on 54 box-constrained and 24 general QP instances.

the algorithm. In short, when compared with (SDP$_1$), (SDP$_2$) continues to use fewer nodes but takes more time.

It is interesting to note that, in Figures 3 and 4, the vast majority of problems (47 of the 54 box-constrained instances and 23 of the 24 general QP instances) were solved in one node by both variants of the branch-and-bound algorithm. This provides clear evidence of the strength of these relaxations on these QPs. In addition, the times in Figure 4 are about one order of magnitude less than those in Figure 2, which perhaps gives a more realistic picture of the performance of these algorithms in practice, when a positive optimality tolerance is used.

We remark that, while we believe the results presented here for the box-constrained case are competitive with other exact algorithms (see Section 6), it may certainly be the case that other methods outperform ours on the general QP instances, which are relatively small. Nevertheless, the paper's goal was to develop finite branch-and-bound algorithms for general QP based on solving semidefinite relaxations, and we were eager to provide computation on general QPs. It is clear that the bottleneck in our approach is the solution of the SDPs by the AL method, which, although faster than more common SDP algorithms, can still use improvement. The small number of nodes indicates that, if we can speed-up the SDP solves significantly, then the overall branch-and-bound algorithm will be quite efficient.

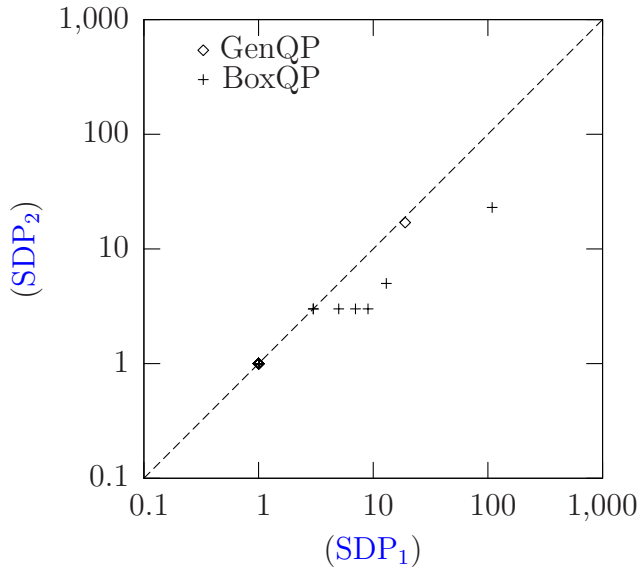

Figure 3: Number of nodes (log-log scale) required for (SDP_1) and (SDP_2) with 1% optimality tolerance on 54 box-constrained and 24 general QP instances.

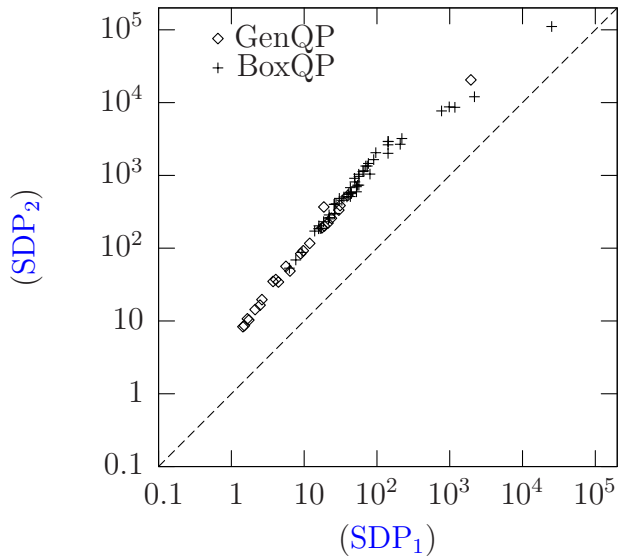

Figure 4: CPU times (log-log scale) required for (SDP_1) and (SDP_2) with 1% optimality tolerance on 54 box-constrained and 24 general QP instances.

6 Specialization to the Box-Constrained Case

With hopes of streamlining our branch-and-bound approach in certain situations, we now specialize our algorithm to BoxQP,

$$\begin{aligned} \max \ & \frac{1}{2}x^T Q x + c^T x \\ & 0 \le x \le e, \end{aligned}$$

where the only constraints of the QP are simple upper and lower bounds. Written explicitly, the basic SDP relaxation (SDP_0) for BoxQP is

$$\begin{aligned} \max \ & \tilde{Q} \bullet Y \\ \text{s.t.} \ & Y \succeq 0 \quad Ye_0 = (1; x) \\ & Ye_i \in K \quad \forall \ \ i = 1, \dots, n \\ & 0 \le x \le e, \end{aligned} \tag{BoxSDP_0}$$

where $K = \{(x_0, x) \in \mathbb{R}^{1+n} : 0 \le x \le x_0\, e\}$ is the homogenization of the hypercube in $\mathbb{R}^n$. Recall that, in the case of BoxQP,

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

As before, our objective is to combine the strength of SDP relaxation with the finite branch-and-bound algorithm provided by the KKT reformulation. In Sections 3 and 4, we defined two SDP relaxations, namely (SDP_1) and (SDP_2), that explicitly contain the dual multipliers (y, z). We intend to show that, in the case of BoxQP, we can handle (y, z) implicitly in the basic SDP relaxation.

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 componentwise. 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 $[Qx]_j + c_j \le 0$, and $z_j = 0$ is equivalent to $[Qx]_j + c_j \ge 0$.

When branching on the complementarity constraints $(e - x) \circ y = 0$ in C_x, we obtain variable fixings $x_j = 1$ or $y_j = 0$ for some index j. Similarly, $x \circ z$ yields fixings of the form $x_j = 0$ or $z_j = 0$. To describe an arbitrary node in our branch-and-bound tree, we denote F^0 (resp., F^1) as the set of indices of components of x that have been fixed to 0 (resp., 1). Lastly, we denote F^y (resp., F^z) as the indices of the components of y (resp., z) that have been fixed to 0. (In the notation of Section 5.1, F^0 corresponds to F^x, and F^1 corresponds to F^{e-x}.) Given F^1, F^0, F^y, and F^z, the corresponding restricted version of (QP) that we consider simply substitutes the following $\tilde{P}$ for P:

$$\tilde{P} = \left\{ x \in \mathbb{R}^n : \begin{array}{c} 0 \le x \le e \\ x_j = 0 \ \ \forall\, j \in F^0 \quad x_j = 1 \ \ \forall\, j \in F^1 \\ [Qx]_j + c_j \le 0 \ \ \forall\, j \in F^y \\ [Qx]_j + c_j \ge 0 \ \ \forall\, j \in F^z \end{array} \right\}. \tag{5}$$

Accordingly, the SDP relaxation that we consider, which implicitly handles (y, z), is of the form (BoxSDP$_0$), except we use $\tilde{P}$ instead of P and the following $\tilde{K}$ in place of K:

$$\tilde{K} = \left\{ (x_0, x) \in \mathbb{R}^{1+n} : \begin{array}{c} 0 \le x \le x_0\, e \\ x_j = 0 \;\; \forall\, j \in F^0 \qquad x_j = x_0 \;\; \forall\, j \in F^1 \\ [Qx]_j + x_0\, c_j \le 0 \;\; \forall\, j \in F^y \\ [Qx]_j + x_0\, c_j \ge 0 \;\; \forall\, j \in F^z \end{array} \right\} \tag{6}$$

6.1 Correctness of branch-and-bound

If we are to use the relaxation (BoxSDP$_0$) tailored by (5) and (6) within branch-and-bound, then we must ensure that a leaf node will always be fathomed. (The proof of correctness discussed in Section 5 is no longer applicable.) 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 6.1. *Consider a feasible leaf node of the branch-and-bound tree, and suppose (Y, x) is a feasible solution to the relaxation (BoxSDP$_0$) (appropriately tailored by $\tilde{P}$ and $\tilde{K}$ 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

$$Q_{\cdot j}^T X_{\cdot j} = (Q_{\cdot j}^T x) x_j \quad \forall j \in \{1, \ldots, n\}. \tag{7}$$

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 $x \in \tilde{P}$ and $Ye_i \in \tilde{K}$, 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 (7) follows easily.

- Suppose now $j \in F^1$. Since $x \in \tilde{P}$, we have $x_j = 1$. Furthermore, since $Ye_i \in \tilde{K}$, 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 (7) is satisfied.

- Lastly, suppose $j \in F^y \cap F^z$. Then $\tilde{P}$ contains the implied constraint $[Qx]_j + c_j = 0$. In a similar fashion, $Ye_j \in \tilde{K}$ 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 (7).

□

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

$$\bar{K} = \left\{ (x_0, x) \in \mathbb{R}^{1+n} : \begin{array}{c} 0 \le x \le x_0\, e \\ [Qx]_j + x_0\, c_j \le 0 \;\; \forall\, j \in F^y \\ [Qx]_j + x_0\, c_j \ge 0 \;\; \forall\, j \in F^z \end{array} \right\}. \tag{8}$$

Clearly, $\bar{K}$ differs from $\tilde{K}$ in that the equalities coming from F^0 and F^1 are not enforced.

Proposition 6.2. *Suppose (Y, x) satisfies $Y \succeq 0$, $Ye_0 = (1; x)$, and $x \in \tilde{P}$. If $Ye_i \in \bar{K}$ for all $i = 1, \ldots, n$, then $Ye_i \in \tilde{K}$ for all $i = 1, \ldots, n$.*

Proof. Let $i \in \{1, \ldots, n\}$. To show $Ye_i \in \tilde{K}$, 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 proven as follows:

- If $j \in F^0$, then $x \in \tilde{P}$ and $Ye_j \in \bar{K}$ imply that $0 \le X_{\cdot j} \le ex_j = 0$. 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 $i < j$):

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

  Note that if $X_{ii} = 0$, then $x_i = X_{ji} = X_{ii} = 0$, as desired. If $X_{ii} > 0$, then we can use the Schur complement theorem to show that

  $$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} - X_{ii}^{-1} \begin{pmatrix} x_i \\ X_{ji} \end{pmatrix} \begin{pmatrix} x_i \\ X_{ji} \end{pmatrix}^T = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}^T - X_{ii}^{-1} \begin{pmatrix} x_i \\ X_{ji} \end{pmatrix} \begin{pmatrix} x_i \\ X_{ji} \end{pmatrix}^T \succeq 0$$

  Now suppose that $X_{ji} \ne x_i$. This implies that (x_i, X_{ji}) can be written as a linear combination

  $$(x_i, X_{ji}) = \alpha(1, 1) + \beta(v_1, v_2)$$

  for some α, β, and (v_1, v_2) such that $\beta \ne 0$, $\|(v_1, v_2)\| = 1$, and $(v_1, v_2)^T(1, 1) = 0$, that is (v_1, v_2) is perpendicular to $(1, 1)$. This implies

  $$0 \le \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}^T \left[\begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}^T - X_{ii}^{-1} \begin{pmatrix} x_i \\ X_{ji} \end{pmatrix} \begin{pmatrix} x_i \\ X_{ji} \end{pmatrix}^T \right] \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = 0 - X_{ii}^{-1} \beta^2,$$

  a contradiction. Hence, if $X_{ii} > 0$, we also conclude that $X_{ji} = x_i$.

□

The benefit of using $\bar{K}$ over $\tilde{K}$ is that it includes fewer explicit constraints, which makes the SDP relaxation smaller and hence may improve computational performance for many SDP solvers. In our case, however, for the AL method, which solves convex quadratic subproblems over the constraints $\tilde{K}$ (or $\bar{K}$), it is actually advantageous to explicitly eliminate variables as can be done with $\tilde{K}$. So in our computational results, we prefer $\tilde{K}$ over $\bar{K}$.

6.2 Implementation details and computational results

Our branch-and-bound algorithm for BoxQP follows the framework laid out in Section 5.1 and incorporates many of the compuational details discussed in Section 5.2 (as applicable). For example, constraints of the form $Y(e_0 - e_i) \in \tilde{K}$ are included to further tighten the relaxations.

The main differences between the BoxQP implementation and the general one of Section 5 are found in branching:

- After solving a relaxation at a node to obtain (x^*, X^*), if branching is necessary, we compute $y^* := \max(0, Qx^* + c)$ and $z^* := -\min(0, Qx^* + c)$ as discussed above, and a branching index is selected by the choosing the complementarity constraint with the largest normalized violation. We use the upper bounds for y and z discussed in Section 2 to normalize. For example, $x_j^* z_j^* / \bar{z}_j$ measures the normalized violation at index j, where $\bar{z}_j = -c_i - \sum_j \min(Q_{ij}, 0)$.

- The constraints of BoxQP logically imply that, if $x_j = 0$, then $1 - x_j = 1$, and so $y_j = 0$ must hold if $(1 - x_j)y_j = 0$. Hence, any branch that sets $x_j = 0$ may also set $y_j = 0$. Said differently, if j is added to F^0, then j may be simultaneously added to F^y. Likewise, if $j \in F^1$, we may put $j \in F^z$. This sort of compound branching is not implied by our framework but is used in our implementation to further streamline the branching process.

- Furthermore, it is not difficult to see that if $Q_{jj} \geq 0$ for some j, then there exists an optimal solution to BoxQP with $x_j \in \{0, 1\}$ (see Vandenbussche and Nemhauser (2005a)). 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 $z_j = 0$.

We compare our algorithm with the branch-and-cut algorithm of Vandenbussche and Nemhauser. For their algorithm, we use the parameter settings suggested in 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 to our SDP approach. The creation of child nodes was also performed as described above.

The test problems include the 54 proposed by Vandenbussche and Nemhauser (2005b) (also used in Section 5), and we have also generated larger instances to demonstrate the new capabilities available using the SDP-based branch-and-bound. The additional instances vary

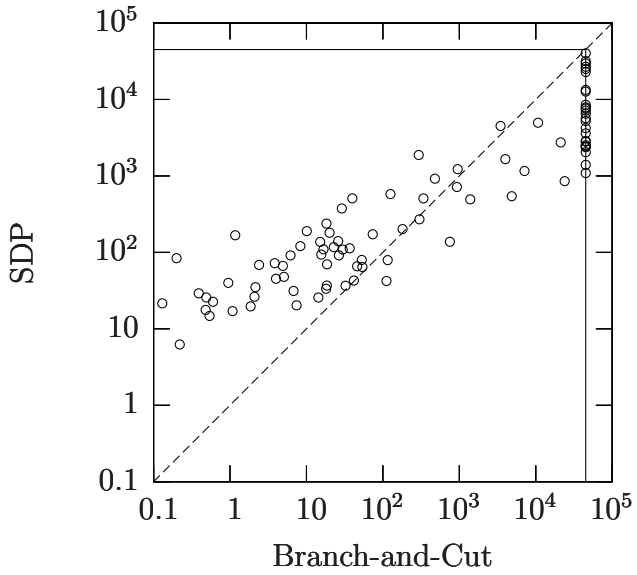

Figure 5: CPU times (log-log scale) required for SDP-based branch-and-bound and LP-based branch-and-cut without optimality tolerance on 90 box-constrained QP instances. A maximum time limit of 45,000 seconds was enforced for all runs.

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%. 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, no optimality tolerance (relative or absolute) was used to fathom the nodes. Figure 5 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 Figure 6. 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.

The maximum number of nodes required for any SDP run was 306, while the same measure for branch-and-cut was 181,958. These numbers illustrate the tightness of the

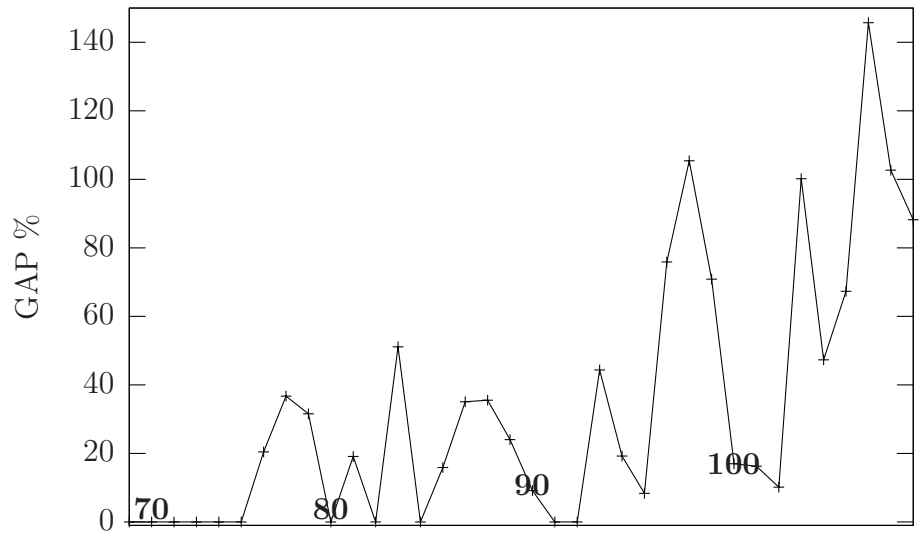

Figure 6: Optimality gaps (%) upon termination for LP-based branch-and-cut when run without optimality tolerance. Instances are ordered first with respect to size and then with respect to density of the matrix Q, and labels specify the instances of various dimensions.

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 Figures 7 and 8. 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.

7 Final Remarks

In this paper, we discuss the use of various SDP relaxations to find global maximizers to nonconcave quadratic programs. We begin by reviewing a classical result which shows that QPs may be reformulated as linear programs with an additional set of complementarity constraints, involving both the original variables and dual multipliers for the original constraints. Vandenbussche and Nemhauser (2005b) used this to develop a finite branch-and-cut algorithm for box-constrained QPs. We show that the main difficulty with extending their approach to the case with general constraints is that the natural LP relaxation is always unbounded. Furthermore, it is not clear how to bound this LP relaxation in an easy way, as was done for the box-constrained case. To overcome this obstacle, we turn to SDP. Empirical results have shown that an SDP relaxation for QP based on the lift-and-project concept of

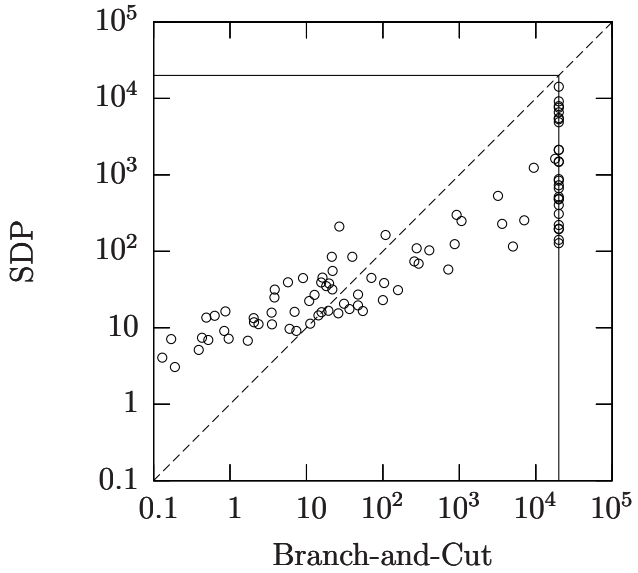

Figure 7: CPU times (log-log scale) required for SDP-based branch-and-bound and LP-based branch-and-cut with 1% optimality tolerance on 90 box-constrained QP instances. A maximum time limit of 20,000 seconds was enforced for all runs.

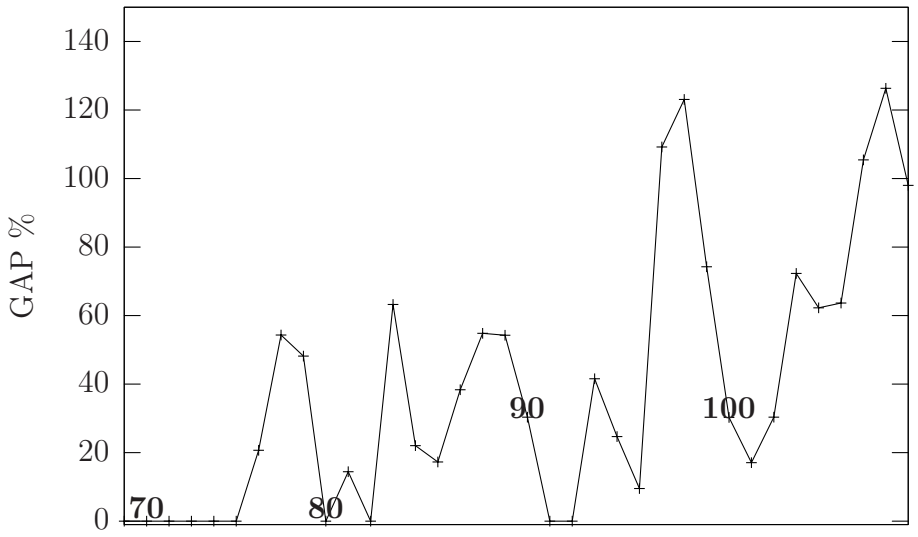

Figure 8: Optimality gaps (%) upon termination for LP-based branch-and-cut when run with 1% optimality tolerance. Instances are ordered first with respect to size and then with respect to density of the matrix Q, and labels specify the instances of various dimensions.

Lovász and Schrijver (1991) can yield very tight bounds on the optimal value of the QP. By developing an SDP relaxation that combines this lift-and-project SDP with the complementarity reformulation, we show that bounds on the variables of this complementarity formulation can be computed via SDP algorithms.

This leads us to examine how one can use these SDPs as relaxations directly within a branch-and-bound scheme. We describe two types of SDP relaxations for general QP and show that they yield correct, finite branch-and-bound algorithms. We compare their performance on a set of randomly generated problems. The strength of the relaxations is clearly demonstrated by the very small branch-and-bound trees required to solve the problems. Significant improvements in the solution of large-scale SDPs will enhance our ability to take advantage of the tight bounds that these relaxations provide.

In the case of BoxQP, one can design a branch-and-bound algorithm that implicitly accounts for the complementarity reformulation of the QP while still employing the tight bounds from the SDP relaxation. This significantly simplifies the SDP relaxations and allows for much faster solution of the subproblems in the branch-and-bound tree. Our computational tests demonstrate the superiority of this approach over the best known algorithms for globally optimizing BoxQP problems.

References

K. M. Anstreicher. Combining RLT and SDP for nonconvex QCQP. Talk given at the Workshop on Integer Programming and Continuous Optimization, Chemnitz University of Technology, November 7-9, 2004.

I. M. Bomze and E. de Klerk. Solving standard quadratic optimization problems via linear, semidefinite and copositive programming. *J. Global Optim.*, 24(2):163–185, 2002. Dedicated to Professor Naum Z. Shor on his 65th birthday.

S. Burer and D. Vandenbussche. Solving lift-and-project relaxations of binary integer programs. Manuscript, Department of Mechanical and Industrial Engineering, University of Illinois Urbana-Champaign, Urbana, IL, USA, June 2004. Revised March 2005. To appear in *SIAM Journal on Optimization*.

C. Floudas and V. Visweswaran. Quadratic optimization. In R. Horst and P. Pardalos, editors, *Handbook of global optimization*, pages 217–269. Kluwer Academic Publishers, 1995.

M. R. Garey and D. S. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-completeness*. W. H. Freeman and Company, San Fransisco, 1979.

F. Giannessi and E. Tomasin. Nonconvex quadratic programs, linear complementarity problems, and integer linear programs. In *Fifth Conference on Optimization Techniques (Rome, 1973), Part I*, pages 437–449. Lecture Notes in Comput. Sci., Vol. 3. Springer, Berlin, 1973.

M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of ACM*, 42: 1115–1145, 1995.

N. I. M. Gould and P. L. Toint. Numerical methods for large-scale non-convex quadratic programming. In *Trends in industrial and applied mathematics (Amritsar, 2001)*, volume 72 of *Appl. Optim.*, pages 149–179. Kluwer Acad. Publ., Dordrecht, 2002.

M. Kojima and L. Tunçel. Cones of matrices and successive convex relaxations of nonconvex sets. *SIAM J. Optim.*, 10(3):750–778, 2000.

L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, 1:166–190, 1991.

G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1988.

Y. Nesterov. Semidenite relaxation and nonconvex quadratic optimization. *Optimization Methods and Software*, 9:141–160, 1998.

P. Pardalos. Global optimization algorithms for linearly constrained indefinite quadratic problems. *Computers and Mathematics with Applications*, 21:87–97, 1991.

N. V. Sahinidis. BARON: a general purpose global optimization software package. *J. Glob. Optim.*, 8:201–205, 1996.

A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, New York, 1986.

H. D. Sherali and B. M. P. Fraticelli. Enhancing RLT relaxations via a new class of semidefinite cuts. *J. Global Optim.*, 22:233–261, 2002.

H. D. Sherali and C. H. Tuncbilek. A reformulation-convexification approach for solving nonconvex quadratic programming problems. *J. Global Optim.*, 7:1–31, 1995.

D. Vandenbussche and G. Nemhauser. A polyhedral study of nonconvex quadratic programs with box constraints. *Mathematical Programming*, 102(3):531–557, 2005a.

D. Vandenbussche and G. Nemhauser. A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. *Mathematical Programming*, 102(3):559–575, 2005b.

Y. Ye. Approximating quadratic programming with bound and quadratic constraints. *Math. Program.*, 84(2, Ser. A):219–226, 1999.