Math. Prog. Comp. (2010) 2:1–19
DOI 10.1007/s12532-010-0010-8

FULL LENGHT PAPER

Optimizing a polyhedral-semidefinite relaxation of completely positive programs

Samuel Burer

Received: 25 December 2008 / Accepted: 9 January 2010 / Published online: 3 February 2010

Abstract It has recently been shown (Burer, Math Program 120:479–495, 2009) that a large class of NP-hard nonconvex quadratic programs (NQPs) can be modeled as so-called *completely positive programs*, i.e., the minimization of a linear function over the convex cone of completely positive matrices subject to linear constraints. Such convex programs are NP-hard in general. A basic tractable relaxation is gotten by approximating the completely positive matrices with *doubly nonnegative matrices*, i.e., matrices which are both nonnegative and positive semidefinite, resulting in a *doubly nonnegative program* (DNP). Optimizing a DNP, while polynomial, is expensive in practice for interior-point methods. In this paper, we propose a practically efficient decomposition technique, which approximately solves the DNPs while simultaneously producing lower bounds on the original NQP. We illustrate the effectiveness of our approach for solving the basic relaxation of box-constrained NQPs (BoxQPs) and the quadratic assignment problem. For one quadratic assignment instance, a best-known lower bound is obtained. We also incorporate the lower bounds within a branch-and-bound scheme for solving BoxQPs and the quadratic multiple knapsack problem. In particular, to the best of our knowledge, the resulting algorithm for globally solving BoxQPs is the most efficient to date.

Keywords Nonconvex quadratic program · Relaxation · Linear programming · Semidefinite programming · Decomposition

Mathematics Subject Classification (2000) 90C26 · 90C05 · 90C22 · 90C99

This research was partially supported by NSF Grant CCF-0545514.

S. Burer (✉)
Department of Management Sciences, University of Iowa, Iowa City, IA 52242-1994, USA
e-mail: samuel-burer@uiowa.edu

1 Introduction

We study the following NP-hard nonconvex quadratic program having nonnegative and binary variables, linear equality constraints, and complementarity conditions:

$$\begin{aligned} \min_x \quad & x^T Q x + 2c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \\ & x_B \in \{0,1\}^{|B|} \\ & [xx^T]_E = 0, \end{aligned} \tag{NQP}$$

where $x \in \Re^n$ is the decision vector and $Q \in \Re^{n\times n}$, $c \in \Re^n$, $A \in \Re^{m\times n}$, $b \in \Re^m$, $B \subseteq [n]$, and $E \subseteq [n]^2$ are the data. In particular, the equation $[xx^T]_E = 0$ encodes $x_i x_j = 0$ for all $(i, j) \in E$. Q is assumed symmetric but not positive semidefinite. While representing linear constraints as equations over nonnegative variables is quite general (e.g., by splitting free variables and adding slacks to inequalities), this form will be critical for our paper.

(NQP) models many interesting, difficult problems. For example, the box-constrained nonconvex quadratic program (see [8] and references therein)

$$\min_x \left\{ x^T Q x + 2c^T x : x \in [0,1]^p \right\} \tag{BoxQP}$$

can be modeled by adding nonnegative slacks to the upper bounds $x \leq e$ (where e is the all-ones vector), padding Q and c with zeros, and taking $A = (I, I)$, $b = e$, $B = \emptyset$, and $E = \emptyset$. In addition, the quadratic assignment problem (see [2] and references therein)

$$\min_X \left\{ \text{trace}(Q_1 X Q_2 X^T) : X \in \Pi_p \right\}, \tag{QAP}$$

where Π_p is the set of $p \times p$ permutation matrices, can be modeled by taking $x := \text{vec}(X)$, $Q := Q_1 \otimes Q_2$, $B = \{1, \ldots, p^2\}$, and then constructing $Ax = b$ to model the doubly stochastic nature of X. In addition, (QAP) can be modeled with complementarities arising from the combinatorial nature of permutation matrices, e.g., $X_{11}X_{12} = 0$ and $X_{11}X_{21} = 0$.

Extending the work of [4,5,11,21], Burer [6] has recently shown that (NQP) is equivalent to the linear conic program

$$\min_{x,X,Y} \quad C \bullet Y \tag{CPP}$$

$$\text{s.t.} \quad Ax = b, \quad \text{diag}(AXA^T) = b^2 \tag{1}$$

$$x_B = \text{diag}(X_{BB}) \tag{2}$$

$$X_E = 0 \qquad (3)$$
$$Y = \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \in \mathcal{K},$$

where $C := (0, c^T; c, Q)$ and $\mathcal{K} := \left\{NN^T : 0 \leq N \in \Re^{(1+n)\times q} \text{ for some } q\right\}$ is the closed, convex cone of $(1 + n) \times (1 + n)$ completely positive matrices. Formally, equivalence of (NQP) and (CPP) relies on the equation Feas(CPP) $=$ Conv $\left\{(1, x^T; x, xx^T) : x \in \text{Feas(NQP)}\right\}$, where Feas$(\cdot)$ and Conv$(\cdot)$ indicate the feasible set and convex hull, respectively. The result ensures, in particular, that both (NQP) and (CPP) have the same optimal value. It should be noted that, although (CPP) is convex, it is still NP-hard. Specifically, the separation problem for $\mathcal{K}$ is widely thought to have exponential or worse complexity [3,17].

If $Y \in \mathcal{K}$, then it necessarily holds that Y is *doubly nonnegative*, i.e., nonnegative ($Y \geq 0$) and positive semidefinite ($Y \succeq 0$). Hence, replacing $\mathcal{K}$ in (CPP) by $\mathcal{D} := \{Y : Y \geq 0, Y \succeq 0\}$ results in the relaxation

$$\min_{x,X,Y} \left\{ C \bullet Y : (1), (2), (3), Y = \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \in \mathcal{D} \right\}. \qquad \text{(DNP)}$$

In fact, an entire hierarchy of polyhedral-semidefinite relaxations is available; see [18] and [4]. From the theory of interior-point methods (IPMs), (DNP) can be solved in polynomial time to any fixed precision. However, in practice, IPMs are limited to fairly small problem sizes as we demonstrate by example in the following paragraph.

Consider the simple case of *standard quadratic programming* [5], i.e., when $m = 1$, $A = e^T$, $b = 1$, $B = \emptyset$, and $E = \emptyset$. We generated a random C for each of $n = 25, 50, 75, 100$ and solved the four instances with SeDuMi [26]; the resulting CPU times (in seconds) were 1, 30, 385, and 1787, respectively. Even for the basic case of standard quadratic programming, the CPU times increase dramatically as n gets larger. The reason for this performance is the constraint $Y \in \mathcal{D}$, which causes the Newton system in every iteration to be size $O(n^2 \times n^2)$.

As an alternative to IPMs, one may consider any number of large-scale SDP methods developed in recent years. We highlight here those large-scale methods that use decomposition coupled with the augmented Lagrangian method. This trend began with the work of [7] for solving lift-and-project polyhedral-semidefinite relaxations and has continued in the works [22,28,30] for more general SDPs.

In this paper, we introduce a specialized augmented-Lagrangian decomposition method whose basic idea is to separate the constraints of (DNP) into two sets, namely $Ax = b, \text{diag}(AXA^T) = b^2, Y \succeq 0$ and $x_B = \text{diag}(X_{BB}), X_E = 0, Y \geq 0$. We illustrate the performance of our algorithm on instances of (BoxQP) and (QAP) and compare with the methods of Burer–Vandenbussche [7] and Zhao–Sun–Toh [30].

An important feature of our method is that, at any stage of the computation, a quickly computable, valid lower bound on the NP-hard problem is available. Valid lower bounds are critical if one wishes to incorporate the relaxations in a branch-and-bound

scheme for globally solving (NQP).[1] In this paper, we incorporate these bounds into branch-and-bound algorithms for solving (BoxQP) and a class of binary NQPs having more general constraints (specifically, quadratic multiple knapsack problems). We compare with Burer–Vandenbussche, who also incorporated their method within branch-and-bound for globally optimizing several classes of continuous NQPs, including (BoxQP).

This paper is organized as follows. In Sect. 2, we introduce our decomposition method and describe it in detail. We also briefly compare our method to that of Burer–Vandenbussche [7]. Sections 3 and 4 are then devoted to the performance of our algorithm for solving (DNP) and solving (NQP) via branch-and-bound, respectively, with particular emphasis on how our results compare to those of Burer–Vandenbussche and Zhao–Sun–Toh. The main conclusions of the paper are:

(i) for (BoxQP), our method is about an order of magnitude faster than Burer–Vandenbussche for solving the relaxation (DNP) and also for globally solving (NQP) via branch-and-bound; thus, to the best of our knowledge, we obtain is a state-of-the-art method for the global optimzation of (BoxQP);
(ii) for (QAP), our method solves the relaxation (DNP) about three times faster on average than both Burer–Vandenbussche and Zhao–Sun–Toh; it also achieves a best known global lower bound on a particular instance from QAPLIB;
(iii) for the quadratic multiple knapsack problem, our method with no particular enhancements can globally solve reasonably sized instances of (NQP) in moderate amounts of time;
(iv) overall, our method shows promise as a general purpose solver for the relaxation (DNP), which in turn can be used effectively within branch-and-bound for globally solving (NQP).

In Sect. 5, we close the paper with a few comments on a possible extension of our method.

1.1 Assumptions and definitions

We assume throughout the paper that the feasible set of (NQP) implies known (possibly infinite) upper bounds $x \le u$. We also define $U := (1, u^T; u, uu^T)$ so that the bounds $Y \le U$ are valid for (DNP). The method of Sect. 2 makes sense even if all $u_j = \infty$, but finite bounds, if known, can be used easily and effectively as will be done in Sects. 3 and 4. We will also find it helpful to define the following matrices:

$$M := \begin{pmatrix} b & -A \end{pmatrix}, \tag{4}$$

$$N := \text{matrix whose columns form an orthonormal basis of Null}(M). \tag{5}$$

In particular, it holds that $N^T N = I$. We do not specify the number of columns of N, i.e., the dimension of Null(M), but all matrix calculations carefully match dimensions.

[1] We remark that the method of [15] can be used to recover valid lower bounds from any SDP method.

Finally, recall that the size of Y is $(1+n)\times(1+n)$; we index the rows and columns of Y by $\{0, 1, \ldots, n\}$. Also, projection onto the positive semidefinite cone is denoted as $\text{proj}_{\succeq 0}(\cdot)$; similarly, projection onto an arbitrary convex set $\mathcal{J}$ is denoted $\text{proj}_{\mathcal{J}}(\cdot)$.

2 The decomposition method

In the following subsections, we examine the structure of the doubly nonnegative relaxation (DNP) in detail and design our decomposition method to exploit this structure.

2.1 Additional properties of the doubly nonnegative program

We first establish some additional properties of (DNP).

Proposition 1 *Suppose $Y = (1, x^T; x, X) \succeq 0$, and let M be given by (4). Then the following are equivalent: (i) $Ax = b$, $\text{diag}(AXA^T) = b^2$; (ii) $MYM^T = 0$; (iii) $MY = 0$.*

Proof (i) $\Rightarrow$ (ii): Let $a^T x = \beta$ be any row of $Ax = b$. We have

$$\begin{pmatrix} \beta & -a^T \end{pmatrix} Y \begin{pmatrix} \beta \\ -a \end{pmatrix} = \beta^2 - 2\beta a^T x + a^T X a = \beta^2 - 2\beta^2 + \beta^2 = 0.$$

Considering all rows of $Ax = b$, this shows that $\text{diag}(MYM^T) = 0$. Since $MYM^T \succeq 0$ because $Y \succeq 0$, it follows that $MYM^T = 0$.

(ii) $\Rightarrow$ (iii): Let $Y = VV^T$ be a Gram representation of Y, which exists because $Y \succeq 0$. We have $0 = \text{trace}(MYM^T) = \text{trace}(MVV^TM^T) = \|MV\|_F^2$, and so $MV = 0$, which implies $MY = (MV)V^T = 0$.

(iii) $\Rightarrow$ (i): $Ax = b$ follows because

$$0 = MY_{\bullet 0} = \begin{pmatrix} b & -A \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix} = b - Ax.$$

Also, letting $a^T x = \beta$ be any row of $Ax = b$, $MY = 0$ implies $MYM^T = 0$, which in turn implies

$$0 = \begin{pmatrix} \beta & -a^T \end{pmatrix} Y \begin{pmatrix} \beta \\ -a \end{pmatrix} = \beta^2 - 2\beta a^T x + a^T X a = a^T X a - \beta^2.$$

Taking all rows of $Ax = b$ together, this is precisely $\text{diag}(AXA^T) = b^2$. □

For the solution of (DNP), the above proposition allows us to work instead with the equivalent

$$\min_{x,X,Y} \left\{ C \bullet Y : MYM^T = 0, (2), (3), Y = \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \in \mathcal{D} \right\}. \tag{6}$$

This formulation will indeed be the basis of our decomposition in Sect. 2.3. At first sight, this choice may seem counterintuitive because $MYM^T = 0$ contains $O(m^2)$ constraints instead of the original $O(m)$ constraints, but its specific form will have advantages for the decomposition.

2.2 Specialized cones and projections

Related to (6), we now introduce some technical definitions and procedures that we use in the next subsection.

Given M from (4), define the following closed, convex cone:

$$\mathcal{J} := \{Z \succeq 0 : MZM^T = 0\}. \tag{7}$$

$\mathcal{J}^*$ denotes the dual cone of $\mathcal{J}$ with respect to the trace inner product.

Lemma 1 $\mathcal{J} = \left\{NPN^T : P \succeq 0\right\}$.

Proof ($\supseteq$): Let $P \succeq 0$, and define $Z := NPN^T$. Then $Z \succeq 0$ and $MZM^T = (MN)P(MN)^T = 0$. ($\subseteq$): Given $Z \in \mathcal{J}$, let $Z = VV^T$ be a Gram representation of Z. Because $MZM^T = 0$, it holds that $MV = 0$, i.e., each column of V is in Null(M). Hence, given V, there exists a (unique) W such that $V = NW$. Defining $P := WW^T \succeq 0$, we see that $Z = NPN^T$, as desired. □

Proposition 2 *For any symmetric* R, $\text{proj}_{\mathcal{J}}(R) = N \,\text{proj}_{\succeq 0}\left(N^T RN\right) N^T$ *and* $\text{proj}_{\mathcal{J}^*}(R) = R + \text{proj}_{\mathcal{J}}(-R)$.

Proof By definition, $\text{proj}_{\mathcal{J}}(R)$ is the unique optimal solution of $\min_Z\{\|R - Z\|_F^2 : Z \in \mathcal{J}\}$. By the preceding lemma, this optimization problem can be rewritten as

$$\min_P \left\{ \left\| R - NPN^T \right\|_F^2 : P \succeq 0 \right\}$$

for which the objective function equals

$$\begin{aligned} \left\| R - NPN^T \right\|_F^2 &= R \bullet R - 2R \bullet NPN^T + NPN^T \bullet NPN^T \\ &= R \bullet R - 2N^T RN \bullet P + P \bullet P. \end{aligned}$$

Swapping in the constant $N^T RN \bullet N^T RN$ for $R \bullet R$, this objective is in turn equivalent to $\left\| N^T RN - P \right\|_F^2$. In other words, the above optimization is equivalent to

$$\min_P \left\{ \left\| N^T RN - P \right\|_F^2 : P \succeq 0 \right\},$$

which has optimal solution $\text{proj}_{\succeq 0}(N^T RN)$. In total, $\text{proj}_{\mathcal{J}}(R) = N \,\text{proj}_{\succeq 0}(N^T RN)N^T$.

The second statement of the proposition follows from standard convex analysis [16]. □

Let k be the number of columns of N. It is then easy to see that $\text{proj}_{\mathcal{J}}(R)$ requires $O(n^2k + k^3) = O(n^2k)$ time. In particular, if M has full-row rank, then $k = n - m$ and the time is $O(n^2(n-m))$. The basic mathematical operations are matrix multiplications and a projection onto the $k \times k$ positive semidefinite cone, which is dominated by the cost of a single spectral decomposition. In practice, both operations can be peformed using LAPACK [1].

2.3 The decomposition algorithm

For notational simplicity, we assume from this point that Y, x, and X are always related by the equation $Y = (Y_{00}, x^T; x, X)$.

Starting from its equivalent formulation (6), we rewrite (DNP) as

$$\min_{Y} \left\{ C \bullet Y : \begin{array}{c} x_B = \text{diag}(X_{BB}),\ X_E = 0,\ Y_{00} = 1,\ Y = Y^T,\ 0 \le Y \le U \\ Y \in \mathcal{J} \end{array} \right\},$$

where $\mathcal{J}$ is defined in (7). We next introduce an auxiliary variable Z and consider the auxiliary problem

$$\min_{Y,Z} \left\{ C \bullet Y : \begin{array}{c} x_B = \text{diag}(X_{BB}),\ X_E = 0,\ Y_{00} = 1,\ Y = Y^T,\ 0 \le Y \le U \\ Z \in \mathcal{J} \\ Y = Z \end{array} \right\}, \tag{8}$$

which will be the basis of our decomposition.

The idea of the decomposition is to relax the constraint $Y = Z$ using a multiplier S and then to apply the standard augmented Lagrangian method. Using duality theory, it can be shown fairly easily that $S \in \mathcal{J}^*$ holds without loss of generality. We also introduce the penalty parameter $\sigma > 0$ to yield the augmented Lagrangian function $L_{(S,\sigma)}(Y, Z) := C \bullet Y - S \bullet (Y - Z) + \frac{\sigma}{2}\|Y - Z\|_F^2$ and the associated subproblem

$$\min_{Y,Z} \left\{ L_{(S,\sigma)}(Y, Z) : \begin{array}{c} x_B = \text{diag}(X_{BB}),\ X_E = 0,\ Y_{00} = 1,\ Y = Y^T,\ 0 \le Y \le U \\ Z \in \mathcal{J} \end{array} \right\}. \tag{9}$$

After the solution of (9) to calculate (Y, Z), S is updated in the standard way; in this case, the formulas reads

$$S \leftarrow \text{proj}_{\mathcal{J}^*}\left(S - \sigma\,(Y - Z)\right), \tag{10}$$

where $\text{proj}_{\mathcal{J}^*}(\cdot)$ is given by Proposition 2.

We now state the augmented Lagrangian framework as Algorithm 1. Note that a number of important details have not yet been specified. In particular, in the following subsection, we discuss our approach for solving the subproblem (9) and for calculating the lower bound v. In Sect. 3, we discuss how accurately to solve (9), how the initial penalty parameter σ_0 is chosen, the precise strategy for updating σ and v, and the algorithm's overall termination criteria.

Algorithm 1 Augmented Lagrangian method for optimizing (DNP) relaxation of (NQP) via auxiliary problem (8)

Input: Data (Q, c, A, b, B, E). Derived data (M, N) given by (4) and (5). Initial penalty parameter $\sigma_0 > 0$.
Output: Approximate solution Y and valid lower bound v for (DNP).
1: Set $(S, \sigma) = (0, \sigma_0)$ and $v = -\infty$.
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Optimize (9) to obtain (Y, Z).
4: Update S according to (10).
5: Update σ.
6: Update v.
7: If termination criterion met, then STOP.
8: **end for**

2.4 The subproblem and lower bound

We now discuss in detail how to solve the subproblem (9) within Algorithm 1. We recommend block coordinate descent over Y and Z. The idea of using block coordinate descent is also employed in [7,22,28,30].

Let us first examine the subproblem over Y when Z is held constant. Neglecting constant terms, the optimization is

$$\begin{array}{ll} \min\limits_{Y} & (C - S - \sigma Z) \bullet Y + \frac{\sigma}{2}\|Y\|_F^2 \\ \text{s.t.} & x_B = \operatorname{diag}(X_{BB}),\ X_E = 0,\ Y_{00} = 1,\ Y = Y^T \\ & 0 \le Y \le U, \end{array} \tag{11}$$

which immediately decomposes into $n(n+1)/2 - |B| - |E| - 1$ one-dimensional strictly convex subproblems of the form $\min_y\{\alpha y + \beta y^2 : 0 \le y \le \mu\}$, each of which can be solved easily in constant time.

Neglecting constant terms, the subproblem over Z with Y held constant is

$$\min_{Z} \left\{ (S - \sigma Y) \bullet Z + \frac{\sigma}{2}\|Z\|_F^2 : Z \in \mathcal{J} \right\}.$$

Completing the square and dividing by σ, the objective is $\left\|\left(Y - \sigma^{-1}S\right) - Z\right\|_F^2$, and so the problem is just the projection of $Y - \sigma^{-1}S$ onto $\mathcal{J}$, i.e., $Z = \operatorname{proj}_{\mathcal{J}}\left(Y - \sigma^{-1}S\right)$ which can be calculated by Proposition 2.

To calculate a valid lower bound v on the optimal value of (DNP) given S, we consider the Lagrangian relaxation of (8), which is just (9) with σ set to 0. This problem

is separable over Y and Z. Moreover, by standard convex analysis, the subproblem over Z has optimal value 0 because $S \in \mathcal{J}^*$. Hence, the problem becomes (11) with σ set to 0, which can be solved in $O(n^2)$ time. In principle, it is possible that the obtained bound is trivial, i.e., equal to $-\infty$, but when u and U are finite, the bound is guaranteed to be finite.

2.5 Comparison with the method of Burer–Vandenbussche

In this subsection, we briefly summarize the decomposition method of Burer–Vandenbussche [7] for solving lift-and-project semidefinite relaxations of problems such as (NQP). Our intent is to draw a contrast with our method, which will in turn provide insight into the computational results of Sects. 3 and 4. However, we caution the reader that many of the details of [7] are overly simplified for the sake of brevity.

In essence, Burer–Vandenbussche propose to solve (DNP) via the auxiliary formulation

$$\min_{Y,Z} \left\{ C \bullet Y : \begin{array}{c} MY = 0,\ x_B = \operatorname{diag}(X_{BB}),\ X_E = 0,\ Y_{00} = 1,\ 0 \le Y \le U \\ Z \succeq 0 \\ Y = Z \end{array} \right\}, \tag{12}$$

which is similar to (8) except that the condition $MZM^T = 0$ implied by $Z \in \mathcal{J}$ in (8) is shifted to the equivalent condition $MY = 0$ in (12); see Proposition 1. Note also that, in (12), symmetry on Y is not explicitly enforced as it is in (8). Instead, it is only enforced implicitly through the equation $Y = Z$ because $Z \succeq 0$ is symmetric by definition. Burer–Vandenbussche then apply decomposition by relaxing the "difficult" constraint $Y = Z$ with a multiplier $S \succeq 0$ and then invoking the augmented Lagrangian approach.

After relaxing $Y = Z$, the constraints of the Burer–Vandenbussche augmented-Lagrangian subproblem are separable with respect to $Y_{\bullet 0}, Y_{\bullet 1}, \ldots, Y_{\bullet n}$, and Z since symmetry is not enforced on Y. This leads Burer–Vandenbussche to solve the augmented Lagrangian subproblem using block coordinate descent over $Y_{\bullet 0}, Y_{\bullet 1}, \ldots, Y_{\bullet n}$, and Z. In this case, each subproblem over $Y_{\bullet j}$ is a convex QP over essentially the polyhedron $\{x \ge 0 : Ax = b\}$, which is solved in practice using CPLEX's pivoting algorithm for convex QP [14]. Note that, if symmetry on Y were enforced, then the smaller-sized subproblems over the columns of Y would not be possible, thus increasing the complexity of Burer–Vandenbussche's method. The subproblem over Z is equivalent to the projection of a matrix onto the positive semidefinite cone, which can be done at the cost of a single spectral decomposition, an $O(n^3)$ operation.

To summarize, we observe that the Y-subproblems of Burer–Vandenbussche are much more expensive than our Y-subproblem, and their Z-subproblem is slightly more expensive than ours.

We also remark that Burer–Vandenbussche focused primarily on linear constraints of the from $Ax \le b$ as opposed to $Ax = b, x \ge 0$ as we do in this paper. For them, it was in some sense just a question of the preferred standard form, and so we are

able to tweak the presentation of their method to $Ax = b, x \geq 0$ above. However, our method in Sect. 2 specifically exploits the structure of $Ax = b, x \geq 0$. This makes it appropriate for solving the doubly nonnegative relaxation (DNP) of the completely positive formulation (CPP) of (NQP), which itself requires the form $Ax = b, x \geq 0$.

3 Computational results: doubly nonnegative relaxation

In this section, we illustrate the performance of Algorithm 1 on instances of (DNP) for the nonconvex quadratic programs (BoxQP) and (QAP). We also compare with the methods of [7] and [30]. We start by specifying many of the implementation details of Algorithm 1.

3.1 Implementation details

We have implemented Algorithm 1 for solving (DNP) in Matlab (version 7.6.0.324, R2008a) on a Linux PC having four 2.4 GHz Intel Core2 Quad CPU processors (with 4,096 KB cache) and 4 GB RAM. However, all calculations use only one processor. In particular, we have used the Matlab command `maxNumCompThreads(1)` to ensure that Matlab makes use of only one processor.

As is clear from the statement of Algorithm 1 and from Sect. 2.4, the solution of (11) and projection onto $\mathcal{J}$ are the two key computational subroutines. Note also that the lower bound calculation and projection onto $\mathcal{J}^*$ rely on these subroutines. Hence, for efficiency, we have implemented these in the C programming language and connected them to Matlab using Matlab's MEX library.

A fundamental implementation choice for Algorithm 1 is how accurately to solve the subproblem (11). Consistent with the experience of Burer–Vandenbussche [7] and other augmented-Lagrangian methods for SDP, we find that it pays only to solve these problems very loosely. In particular, as in Burer–Vandenbussche, we do one loop of block coordinate over Y and Z.

To save a bit of time, we also do not update the lower bound v every iteration. Instead, we update it every 25 iterations, which is the same as Burer–Vandenbussche.

The practical performance of Algorithm 1—indeed, *any* augmented Lagrangian algorithm—depends immensely on how σ is initialized and updated throughout the algorithm. We first discuss our update rule, which we have found to work well in practice after much experimentation. The basic idea is to be more agressive with σ (i.e., increase it more) when the bound v is converging more quickly (usually near the beginning of the algorithm) and conversely to be more conservative with σ when v is converging slowly (usually near the end). If v decreases, we even allow σ to decrease. The exact rule is as follows. Let v be the current lower bound associated with the multiplier S, and let $\bar{v}$ be the maximum lower bound achieved so far during the course of Algorithm 1. Then update σ as

$$\sigma \leftarrow \left(1 + \frac{v - \bar{v}}{1 + |\bar{v}|}\right)\sigma.$$

The ratio $(v - \bar{v})/(1 + |\bar{v}|)$ gives the percentage change in the bound. Adding this ratio to 1 gives the muliplicative factor for σ (usually above 1, but sometimes below). Very occasionally, it happens that the bound v deteriorates dramatically relative to $\bar{v}$, so much so that the ratio is nonpositive, making the update rule nonsensical. As a safeguard in these cases, we do not update σ and instead solve subsequent subproblems more accurately, i.e., more than one loop of block coordinate descent, until a sensible update to σ is realized. Note that σ is updated as often as v is updated, i.e., every 25 iterations in our implementation.

In the best case, an update rule for σ could identify the optimal σ during the course of the algorithm even with a poor choice of σ_0. Unfortunately, our update rule is not perfect: it is sensitive to the initial penalty parameter σ_0. Despite many attempts, we have been unable to establish a single rule or strategy for setting σ_0 that works well for all problem instances. (Such a strategy is an important area for further investigation.) Instead, we calibrate and choose σ_0 for each problem class. Specifically, in Sect. 3.2 below, we set $\sigma_0 := \max_{ij} |C_{ij}|$ for (BoxQP), while in Sect. 3.3 we set $\sigma_0 := 1$ for (QAP).

Finally, we terminate the augmented Lagrangian algorithm if either: (i) the average relative change of v over the last 5 updates goes below 10^{-5}; or (ii) the number of iterations exceeds 6,000. Burer–Vandenbussche use similar stopping criteria.

3.2 Box-constrained nonconvex quadratic programs

We solve the doubly nonnegative relaxation (DNP) for 99 random instances of (BoxQP) with sizes ranging from $p = 20$ to $p = 125$. Note that, in this case, to put (BoxQP) in the form of (NQP), one must add slack variables. In particular, the number of variables in (NQP) is $n = 2p$. We also have $m = p$ and $B = E = \emptyset$. Moreover, we have implied bounds $u = e$, which gives $U = ee^T$.

The 99 instances come from the following sources: (i) 54 instances with sizes $20 \le p \le 60$ generated in [27]; (ii) 36 instances with sizes $70 \le p \le 100$ generated in [8]; and (iii) 9 instances with $p = 125$ generated for this paper. The instances with $p \ge 70$ have been generated in the same way as the smallest 54 instances, i.e., for varying densities of Q, nonzeros of (Q, c) are uniformly generated integers in $[-50, 50]$.

We compare Algorithm 1 with the method of Burer–Vandenbussche. Although we have not summarized all the details of their algorithm, we remark that the two methods are directly comparable in that they both solve and produce lower bounds on (DNP). The basic question we ask is: which method produces better bounds more quickly?

To make a fair comparison, we control for the lower bounds obtained and adopt the following definition of algorithm run time on a particular instance:

> Let v_{A1} be the best (maximum) lower bound obtained by Algorithm 1 on a particular instance, and let v_{BV} be the best lower bound obtained by Burer–Vandenbussche. Define $\hat{v} := \min\{v_{A1}, v_{BV}\}$. In particular, both methods have achieved the lower bound $\hat{v}$ on that instance. Then, by definition, the run time of Algorithm 1 is the time when it first reaches the bound $\hat{v}$, and similarly the run time for Burer–Vandenbussche is the time when it first reaches $\hat{v}$.

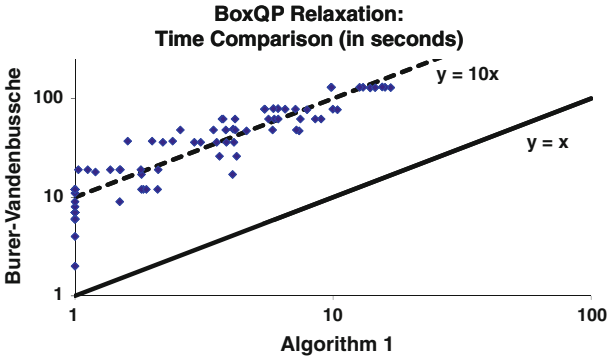

Fig. 1 Comparison of Algorithm 1 and Burer–Vandenbussche for solving the (DNP) relaxation of 99 instances of (BoxQP). Each *axis* is algorithm run time (in seconds) plotted on a logarithmic scale. The *lines* $y = x$ and $y = 10x$ are plotted for reference

With this definition of run time, we summarize our tests in Fig. 1, which is a log-log scatter plot of the run times of Burer–Vandenbussche versus the run times of Algorithm 1 on the 99 instances. All times are in seconds. For reference, the lines $y = x$ and $y = 10x$ are plotted as well.

Figure 1 clearly shows that Algorithm 1 outperforms Burer–Vandenbussche, achieving the same lower bound in much less time (often more than 10 times faster). On average, Algorithm 1 is 10.21 times faster.

We mention one additional statistic. Over all 99 instances, the average time per augmented-Lagrangian iteration for Algorithm 1 is 0.0058 s, while the same measurement for Burer–Vandenbussche is 0.0393 s. In other words, Algorithm 1 is about 7 times faster than Burer–Vandenbussche per iteration. Since the two algorithms both follow the augmented-Lagrangian framework with similar implementation choices (see Sect. 3.1) and hence seem to converge at roughly the same rate, this difference in time-per-iteration explains Fig. 1 to a large degree.

3.3 The quadratic assignment problem

We solve the relaxation (DNP) for 92 instances of (QAP), which are all of the instances from QAPLIB [9] having $p \leq 36$ (see the website [23]). When (QAP) is put into the form (NQP), we have $n = p^2$, $m = 2p$, and $B = [n]$. Moreover, we have the implied bounds $u = e$ and $U = ee^T$. We also enforce the so-called *gangster operator* [29], which are complementarities in a set E of size $O(n^3)$.

Different polyhedral-semidefinite relaxations for the QAP have been investigated in the literature, and several of them have been shown to be equivalent. In particular, the lift-and-project relaxation solved by Burer–Vandenbussche [7] is equivalent to the copositive one solved in [30]; see [21]. In its derivation—lifting from the original space of $\mathrm{vec}(X)$ to the quadratic space of $\mathrm{vec}(X)\mathrm{vec}(X)^T$—our relaxation is quite similar to Burer–Vandenbussche, and we believe, but have not rigorously proved, that

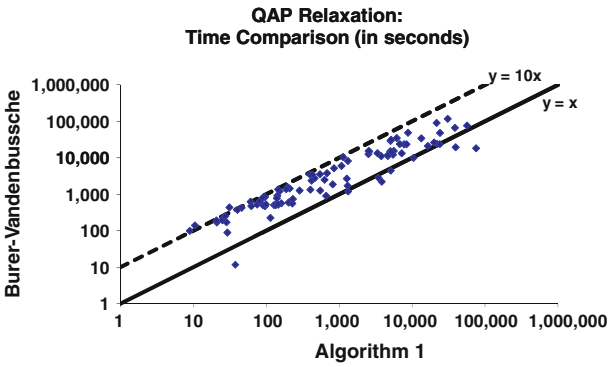

Fig. 2 Comparison of Algorithm 1 and Burer–Vandenbussche for solving the (DNP) relaxation of 92 instances of (QAP). Each *axis* is algorithm run time (in seconds) plotted on a logarithmic scale. The *lines* $y = x$ and $y = 10x$ are plotted for reference

the two are equivalent. If this is indeed the case, then all three algorithms produce bounds for the QAP that are directly comparable.

We also mention here that, although not employed in this paper, [12] has devised a pre-processing technique to reduce the size of the SDP relaxation of the QAP. Their method works particularly well when an instance possesses inherent symmetry. Such is the case with relaxations of the *esc* instances of QAPLIB, which can be solved in a couple of minutes after pre-processing. The size reduction also benefits the numerical accuracy of the algorithms, and as a result some best known lower bounds can be obtained by their method, e.g., for instance *esc32h*.

Figure 2 reports the results of Algorithm 1 and Burer–Vandenbussche; the figure is setup exactly as Fig. 1 of the previous subsection. On the vast majority of instances, Algorithm 1 computes the same bound more quickly than Burer–Vandenbussche; overall, Algorithm 1 is 4.83 times faster on average.

We also plot in Fig. 3 the following time-per-iteration ratio for each of the 92 instances versus the problem size p:

$$\begin{aligned}\text{time-per-iteration ratio} &:= \frac{\text{Algorithm 1 average time per iteration}}{\text{Burer-Vandenbussche average time per iteration}} \\ &:= \frac{\text{Algorithm 1} \ \frac{\text{time}}{\text{number of iterations}}}{\text{Burer-Vandenbussche} \frac{\text{time}}{\text{number of iterations}}},\end{aligned}$$

where both the time and the number of iterations are defined with respect to the bound $\hat{v}$ obtained by both methods (see previous subsection for the definition of $\hat{v}$).

Figure 3 shows that Algorithm 1 spends significantly less per iteration than Burer–Vandenbussche, which accounts to a large degree for the overall results in Fig. 2. However, Fig. 3 shows an upward trend as p increases. This indicates that, as p gets larger, the relative difference in performance between the two algorithms will grow smaller. This trend is due to the fact that, for larger p, each iteration in either

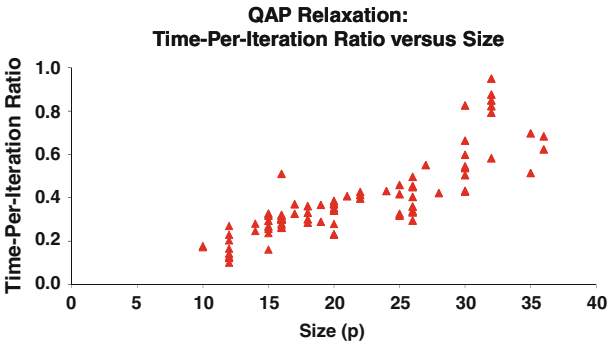

Fig. 3 Comparison of the time-per-iteration ratio between Algorithm 1 and Burer–Vandenbussche versus the size p when solving the (DNP) relaxation of 92 instances of (QAP). A ratio less than 1 indicates that Algorithm 1 requires less time per iteration than Burer–Vandenbussche on that instance

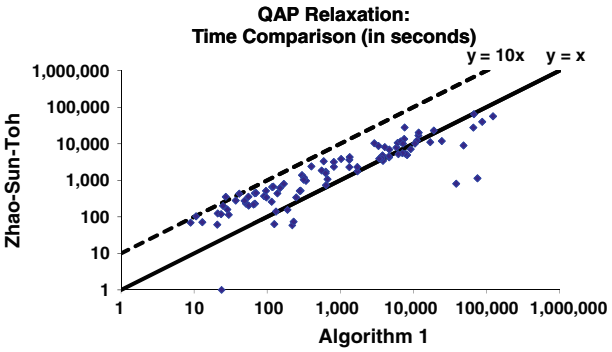

Fig. 4 Comparison of Algorithm 1 and Zhao–Sun–Toh for solving the (DNP) relaxation of 92 instances of (QAP). Each *axis* is algorithm run time (in seconds) plotted on a logarithmic scale. The *lines* $y = x$ and $y = 10x$ are plotted for reference

algorithm is dominated by the $O(p^6)$ projection of a $p^2 \times p^2$ matrix onto a positive semidefinite cone. This is a bottleneck for both methods.

Finally, Fig. 4 is the same as Fig. 2 except that Algorithm 1 is compared with the method of Zhao–Sun–Toh [30]. The method of Zhao–Sun–Toh [15] was incorporated within the execution of Zhao–Sun–Toh to compute valid lower bounds; this modification did not significantly increase the run times for Zhao–Sun–Toh. Overall, the figure shows that Algorithm 1 performs better than Zhao–Sun–Toh on most problems; on average, Algorithm 1 is 2.94 times faster.

Since (QAP) is a particularly well studied problem, in the online Appendix[2] we catalog in detail the performance of the three algorithms on the 92 instances. In particular, we point out that Algorithm 1 achieves the best known lower bound for the instance *tai35b* according to the [23] website (visited on November 2, 2009). Also, Zhao–Sun–Toh obtains the best known lower bound on *tai30b*.

[2] Available at the website http://dollar.biz.uiowa.edu/~sburer.

4 Computational results: branch-and-bound

Consider a branch-and-bound method where each node has a relaxation of the type (DNP) so that Algorithm 1 may be applied. By and large, incorporating Algorithm 1 into branch-and-bound is straightforward. The main modifications we make relate to warm-starting the augmented Lagrangian algorithm at the current node using information from the parent node.

In particular, we save the last multiplier S from the parent node and modify it in a problem-dependent way to initialize the S at the current node. By "problem-dependent," we mean, for example, padding S by an extra row and column of zeros when a new row and column are introduced into the Y of the parent's relaxation to form the Y of the child's relaxation.

We also save the final penalty parameter σ of the parent and initialize $\sigma_0 := \sqrt{\sigma}$ for the child. The reason for the square root is experimental. We tried setting σ_0 equal to σ, but found that the penalty parameters became too large deep in the tree when the multipliers S were becoming closer and closer to optimal. Hence, we devised the update $\sqrt{\sigma}$ to gradually lower the penalty parameter to more reasonable levels deeper in the tree.

Finally, we also impose a 1000-iteration limit on Algorithm 1 at each node of the branch-and-bound tree. In the next subsection, the branch-and-bound algorithm of Burer–Vandenbussche is run with the same 1000-iteration limit.

4.1 Box-constrained nonconvex quadratic programs

As discussed in the Introduction, Burer and Vandenbussche [8] have used their augmented Lagrangian method along with the lower bounds it produces to solve instances of (BoxQP) globally. We replicate the exact same branch-and-bound algorithm, e.g., same branching rule, node-selection rule, fathoming tolerance, primal heuristic, etc. The only change we make is to use Algorithm 1 in place of their augmented Lagrangian method for solving the relaxation at each node. While it is certainly possible that numerical differences in the relaxations can result in different branch-and-bound trees, we anticipate that any overall differences in run times will be due mainly to differences between Algorithm 1 and that of Burer–Vandenbussche.

In the interest of space, we do not give all details of the branch-and-bound algorithm; please see [8] for the full description. However, we indicate the structure of (DNP) at any node in the tree. A node is based on the original formulation (BoxQP) with a finite list of additional "optimality cuts" (say, $a^T x \leq \beta$) and variable fixings ($x_j = 0$ or $x_j = 1$). To form (DNP), we add slacks to $a^T x \leq \beta$ and $x \leq e$ to bring the problem into the form of (NQP). In fact, associated with the slack on $a^T x \leq \beta$, we calculate its implied upper bound μ, and then add the scaled-slack equation $a^T x + \mu\, s = \beta$ with $0 \leq s \leq 1$. We found this resulted in more numerically stable relaxations. We handle the variable fixings by setting their lower and upper bounds equal to one another and then carry the bounds and variables throughout the calculations. Another alternative would have been just to eliminate the variables, but our approach involves only a small

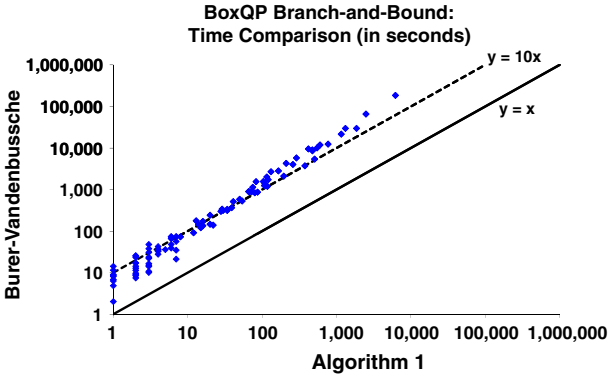

Fig. 5 Comparison of Algorithm 1 and Burer–Vandenbussche for globally solving 99 instances of (BoxQP). Each *axis* is algorithm run time (in seconds) plotted on a logarithmic scale. The *lines* $y = x$ and $y = 10x$ are plotted for reference

overhead and yet keeps our data structures consistent and more transparent from node to node.

We globally solve the same 99 (BoxQP) instances from Sect. 3.2 and compare overall timings in Fig. 5. This figure is setup just as Figs. 1 and 2 in Sect. 3 and clearly shows that the advantage of Algorithm 1 seen in Fig. 1 transfers well to the context of branch-and-bound. For example, for all problems that took Algorithm 1 1,000 s or more, the same problems took Burer–Vandenbussche at least ten times as long. Overall, Algorithm 1 is 11.28 times faster on average. For the 9 largest problems of size $p = 125$, Algorithm 1 is 20.50 times faster.

To date the method of Burer–Vandenbussche has been the most efficient for globally solving (BoxQP). In particular, off-the-shelf solvers such as BARON [24] have been outperformed by a specialized LP-based branch-and-cut algorithm [27], and Burer–Vandenbussche [8] present their method, which outperforms the branch-and-cut method. Our experiments here show convincingly that Algorithm 1 outperforms Burer–Vandenbussche and hence is a state-of-the-art algorithm for globally solving nonconvex box-constrained quadratic programs.

4.2 The quadratic multiple knapsack problem

The binary *quadratic (single) knapsack problem* $\min_x\{x^T Qx + 2c^T x : a^T x \leq \beta,\ x \in \{0, 1\}^p\}$ is a generalization of the standard 0-1 knapsack problem and has been well studied in the last few years; see [19] for a recent survey. Specialized methods for solving the quadratic knapsack problem have been presented in [10] and [20]. These methods can solve random instances of size $p \approx 1,500$ in a few thousand seconds on a modern PC. A semidefinite cutting plane approach is presented in [13] for $p \approx 50$.

Pisinger et al. [20] discuss a scheme for generating random (Q, c, a, β), which has become standard in the literature. In particular, $a > 0$ and $\beta > 0$ so that the problem is feasible (with $x = 0$). Also, one typically takes $c = 0$.

We generated several random instances and put them in the following "extended" form matching (NQP):

$$\begin{array}{ll} \min_x & x^T Q x + 2\, c^T x \\ \text{s.t.} & a^T x + \beta s = \beta,\ x + y = e \\ & (x, y) \in \{0, 1\}^{2p},\ s \geq 0 \\ & x_j y_j = 0\ \forall\ j = 1, \ldots, p. \end{array}$$

Note that all variables, including the slack s, have an implied upper bound of 1. The extended form is used in order to strengthen the resulting (DNP) relaxation as much as possible. We then implemented a straightforward branch-and-bound algorithm based on calculating bounds from (DNP) via Algorithm 1 and branching on the most fractional variables. To generate primal solutions, we take the x from (DNP), round it to a binary vector, and then greedily remove items from the knapsack until x becomes feasible. This is the simplest primal heuristic presented in [20]. Though our method was successful for $p \approx 50$, it was not competitive with the specialized method of [20].

On the other hand, our method does not exploit the structure of the quadratic knapsack problem to the extent that the specialized methods do. For example, the lower bounds calculated in [20] rely on a linear programming relaxation, which decomposes into n continuous linear knapsack problems that are extremely quick to solve. It is not clear how additional constraints would affect these methods. In addition, these methods can reduce the overall size of a problem using clever reduction techniques.

With these observations, we hypothesize that our method generalizes well in the presence of additional linear inequality constraints, which is a natural concern beyond a single constraint or knapsack. So we consider the binary *quadratic multiple knapsack problem* $\min_x\{x^T Q x + 2\, c^T x : Ax \leq b,\ x \in \{0, 1\}^p\}$. Let q be the number of knapsacks, i.e., the number of rows of A. In particular, in order to ensure that the problem is feasible, we generate instances with each row of $Ax \leq b$ a single knapsack $a^T x \leq \beta$ with $a > 0$ and $\beta > 0$. There does not seem to be much work on globally optimizing the quadratic multiple knapsack problem; a heuristic is presented in [25].

We implemented a simple branch-and-bound algorithm just as we did for the single-knapsack case and ran the following experiments: for each of the values $p = 10, 20, 30, 40, 50$, solve 50 random instances with 5 knapsacks (i.e., $q = 5$) and report the average and standard deviation of the solution times; repeat with $q = \lceil p/2 \rceil$ for each p. For completeness, we also ran $q = 1$ for each p (i.e., the single knapsack case). The results of these experiments are shown in Table 1.

Overall, we believe that Table 1 shows a reasonable pattern for increasing q: irrespective of q, one can solve instances of the quadratic multiple knapsack problem having a few tens of variables in a reasonable amount of time (say, in about 20 min on average). This is a reflection of the strength of (DNP) and the speed of Algorithm 1 to solve it. We anticipate that these results could be improved, for example, with more intelligent branching strategies and/or primal heuristics.

Table 1 Average time (in seconds) and standard deviation in parentheses for globally solving 50 instances of the quadratic multiple knapsack problem for various combinations of p (number of variables) and q (number of knapsack constraints)

p	$q = 1$	$q = 5$	$q = \lceil \frac{p}{2} \rceil$
10	3.2 (2.1)	3.3 (2.9)	3.4 (4.1)
20	18.3 (22.4)	33.2 (32.3)	29.9 (26.2)
30	104.5 (212.3)	206.8 (678.0)	78.6 (61.0)
40	930.1 (3204.4)	329.3 (394.3)	303.7 (344.2)
50	1048.8 (5180.6)	476.1 (867.5)	812.8 (1273.4)

5 Future extension

It would be interesting to generalize Algorithm 1 to arbitrary linear conic programs, where the convex cone of interest is $\mathcal{D}$, the set of doubly nonnegative matrices. In principle, this paper's decomposition approach involving the auxiliary variable Z and the equation $Y = Z$ could become the basis of just such an algorithm, but the following key issue would need to be addressed: should the arbitrary linear constraints be grouped into the Y subproblem or the Z subproblem, or divided between the subproblems in some intelligent manner? Whatever the choice, one would require that the Y and Z subproblems remain quick and easy to solve (as in this paper) in order to ensure overall efficiency of the decomposition method.

Acknowledgments The author would like to thank the three anonymous referees and editor for their helpful comments and corrections on the first draft of this paper. The author is also in debt to the technical editor for testing the code and providing many ease-of-use suggestions. In addition, the author thanks Zhao, Sun, and Toh for sharing their code.

References

1. Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.: LAPACK Users' Guide. Society for Industrial and Applied Mathematics, Philadelphia (1999)
2. Anstreicher, K.M.: Recent advances in the solution of quadratic assignment problems. Math. Program. (Ser. B) **97**(1–2), 27–42 (2003)
3. Berman, A., Rothblum, U.G.: A note on the computation of the CP-rank. Linear Algebra Appl. **419**, 1–7 (2006)
4. Bomze, I.M., de Klerk, E.: Solving standard quadratic optimization problems via linear, semidefinite and copositive programming. J. Glob. Optim. **24**(2), 163–185 Dedicated to Professor Naum Z. Shor on his 65th birthday (2002)
5. Bomze, I.M., Dür, M., de Klerk, E., Roos, C., Quist, A.J., Terlaky, T.: On copositive programming and standard quadratic optimization problems. J. Glob. Optim. **18**(4), 301–320 GO'99 Firenze (2000)
6. Burer, S.: On the copositive representation of binary and continuous nonconvex quadratic programs. Math. Program. **120**, 479–495 (2009)
7. Burer, S., Vandenbussche, D.: Solving lift-and-project relaxations of binary integer programs. SIAM J. Optim. **16**(3), 493–512 (2006)
8. Burer, S., Vandenbussche, D.: Globally solving box-constrained nonconvex quadratic programs with semidefinite-based finite branch-and-bound. Comput. Optim. Appl. **43**(2), 181–195 (2009)

9. Burkard, R.E., Karisch, S., Rendl, F.: QAPLIB—a quadratic assignment problem library. Eur. J. Oper. Res. **55**, 115–119 (1991)
10. Caprara, A., Pisinger, D., Toth, P.: Exact solution of the quadratic knapsack problem. Inf. J. Comput. **11**(2), 125–137 (1999). Combinatorial optimization and network flows
11. de Klerk, E., Pasechnik, D.V.: Approximation of the stability number of a graph via copositive programming. SIAM J. Optim. **12**(4), 875–892 (2002)
12. de Klerk, E., Sotirov, R.: Exploiting group symmetry in semidefinite programming relaxations of the quadratic assignment problem. Math. Program. **122**(2 Ser. A), 225–246 (2010)
13. Helmberg, C., Rendl, F., Weismantel, R.: A semidefinite programming approach to the quadratic knapsack problem. J. Comb. Optim. **4**(2), 197–215 (2000)
14. ILOG, Inc.: ILOG CPLEX 9.0, User Manual (2003)
15. Jansson, C., Chaykin, D., Keil, C.: Rigorous error bounds for the optimal value in semidefinite programming. SIAM J. Numer. Anal. **46**(1), 180–200 (2007)
16. Moreau, J.-J.: Décomposition orthogonale d'un espace hilbertien selon deux cônes mutuellement polaires. C. R. Acad. Sci. Paris **255**, 238–240 (1962)
17. Murty, K.G., Kabadi, S.N.: Some NP-complete problems in quadratic and nonlinear programming. Math. Program. **39**(2), 117–129 (1987)
18. Parrilo, P.: Structured semidefinite programs and semi-algebraic geometry methods in robustness and optimization. PhD thesis, California Institute of Technology (2000)
19. Pisinger, D.: The quadratic knapsack problem—a survey. Discret. Appl. Math. **155**(5), 623–648 (2007)
20. Pisinger, D., Rasmussen, A., Sandvik, R.: Solution of large-sized quadratic knapsack problems through aggressive reduction. Inf. J. Comput. **19**(2), 280–290 (2007)
21. Povh, J., Rendl, F.: Copositive and semidefinite relaxations of the quadratic assignment problem. Discret. Optim. **6**(3), 231–241 (2009)
22. Povh, J., Rendl, F., Wiegele, A.: A boundary point method to solve semidefinite programs. Computing **78**(3), 277–286 (2006)
23. QAPLIB. http://www.seas.upenn.edu/qaplib/
24. Sahinidis, N.V.: BARON: a general purpose global optimization software package. J. Glob. Optim. **8**, 201–205 (1996)
25. Sarac, T., Sipahioglu, A.: A genetic algorithm for the quadratic multiple knapsack problem. In: Advances in Brain, Vision, and Artificial Intelligence, vol. 4729 of Lecture Notes in Computer Science, pp. 490–498. Springer, Heidelberg (2007)
26. Sturm, J.F.: Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optim. Methods Softw. **11/12**(1–4), 625–653 (1999)
27. Vandenbussche, D., Nemhauser, G.: A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. Math. Program. **102**(3), 559–575 (2005)
28. Wen, Z., Goldfarb, D., Yin, W.: Alternating Direction Augmented Lagrangian Methods for Semidefinite Programming. Manuscript, Department of Industrial Engineering and Operations Research, Columbia University, New York (2009)
29. Zhao, Q., Karisch, S., Rendl, F., Wolkowicz, H.: Semidefinite programming relaxations for the quadratic assignment problem. J. Comb. Optim. **2**, 71–109 (1998)
30. Zhao, X., Sun, D., Toh, K.: A Newton-CG augmented Lagrangian method for semidefinite programming. Preprint, National University of Singapore, Singapore, Mar (2008). To appear in SIAM Journal on Optimization