Journal of Global Optimization **22:** 233–261, 2002.
© 2002 *Kluwer Academic Publishers. Printed in the Netherlands.*

Enhancing RLT relaxations via a new class of semidefinite cuts

HANIF D. SHERALI* and BARBARA M. P. FRATICELLI[⋆]
*Department of Industrial and Systems Engineering, 250 New Engineering Building, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA *Corresponding author (E-mail: hanifs@vt.edu)*

Abstract. In this paper, we propose a mechanism to tighten Reformulation-Linearization Technique (RLT) based relaxations for solving nonconvex programming problems by importing concepts from semidefinite programming (SDP), leading to a new class of *semidefinite cutting planes*. Given an RLT relaxation, the usual nonnegativity restrictions on the matrix of RLT product variables is replaced by a suitable positive semidefinite constraint. Instead of relying on specific SDP solvers, the positive semidefinite stipulation is re-written to develop a semi-infinite linear programming representation of the problem, and an approach is developed that can be implemented using traditional optimization software. Specifically, the infinite set of constraints is relaxed, and members of this set are generated as needed via a separation routine in polynomial time. In essence, this process yields an RLT relaxation that is augmented with valid inequalities, which are themselves classes of RLT constraints that we call *semidefinite cuts*. These semidefinite cuts comprise a relaxation of the underlying semidefinite constraint. We illustrate this strategy by applying it to the case of optimizing a nonconvex quadratic objective function over a simplex. The algorithm has been implemented in C++, using CPLEX callable routines, and two types of semidefinite restrictions are explored along with several implementation strategies. Several of the most promising lower bounding strategies have been implemented within a branch-and-bound framework. Computational results indicate that the cutting plane algorithm provides a significant tightening of the lower bound obtained by using RLT alone. Moreover, when used within a branch-and-bound framework, the proposed lower bound significantly reduces the effort required to obtain globally optimal solutions.

Key words: Reformulation-Linearization Technique (RLT), Semidefinite programming (SDP), Semidefinite cuts, Valid inequalities, Nonconvex quadratic programming.

1. Introduction

The Reformulation-Linearization Technique (RLT) is a unifying approach for solving discrete and continuous nonconvex optimization problems (see Sherali and Adams, 1999), for a comprehensive exposition). The RLT strategy is to suitably multiply appropriate constraints by nonnegative bound-factors, constraint-factors, or simply variables in a reformulation phase, and then to replace the products of original variables by new variables in order to derive a higher-dimensional lower bounding linear programming (LP) relaxation for the original problem. This RLT

[⋆] This material is based upon research supported by the *National Science Foundation* under Grant Number DMI-9812047.

process can actually generate a hierarchy of tighter relaxations, depending on the types of factor products employed in the reformulation phase. In practice, however, the lowest-level RLT relaxation (as dictated by the nature of the terms in the original problem) is most frequently implemented in order to control the size of the resulting relaxation, although higher-level relaxations have been successfully used in certain special applications. For many classes of problems, such a lowest-level RLT relaxation has proven effective in deriving tight lower bounds for the original problem. However, this observation is not a uniform experience, and even in the aforementioned cases, the overall process can greatly benefit by incorporating suitable general classes of additional RLT inequalities that serve to further tighten the relaxation, without having to resort to higher-level representations. With this motivation, we explore the generation of particular types of valid inequalities or cutting planes that are in fact generalized RLT constraints derived via semidefinite programming concepts. We call this class of valid inequalities *semidefinite cuts*. For some other classes of effective RLT cuts developed for the special case of quadratic polynomial programs, we refer the reader to Audet et al. (2000).

Semidefinite programming (SDP) offers a related relaxation strategy to RLT for solving certain types of nonconvex programming problems. Semidefinite programs are similar to LPs, except that the vector of variables is replaced by a matrix of appropriate variables, a special product operation is defined in lieu of the usual matrix–vector operations, and the matrix of variables is restricted to be positive semidefinite **(PSD)**, in contrast with the nonnegativity constraints on the variables in linear programming. SDP has been receiving increasing attention from the mathematical programming community since its inception over the past 5–10 years. Part of the reason for its popularity, as pointed out by Vandenbergh and Boyd (1996), is that SDP unifies several areas of mathematical programming (including linear and quadratic programming) from a theoretical point of view. Active set methods (similar to the simplex method in LP) were originally employed to solve SDP problems, but more recently, as shown by Alizadeh (1995), many interior point methods for solving linear programs can be directly modified and used to solve semidefinite programs in polynomial time. For a detailed overview of SDP, see Vandenbergh and Boyd (1996), or Alizadeh (1995). For articles that address theoretical results as well as various specific applications pertaining to SDP, see also: Wolkowicz et al. (2000), Todd (1998), Bertsimas and Ye (1998), Ramana and Pardalos (1996), Ramana and Goldman (1995), and Goemans and Williamson (1995).

More recently, there has been an impetus of research related to reformulating and solving SDPs as ordinary nonlinear programs. Vanderbei and Benson (2000) propose a smooth, convex, finite nonlinear programming representation of a given positive semidefinite constraint $X \succeq 0$, by noting that a symmetric matrix X is PSD if and only if it can be factored as $X = LDL^T$, where L is a unit lower triangular matrix, and D is a diagonal nonnegative matrix. Denoting $d_j(X)$, $j = 1, \ldots, n$, as the diagonal elements of D for a given $n \times n$ symmetric matrix X, Vanderbei and Benson show that each $d_j(X)$ is a concave function of the elements of X, and moreover, is twice continuously differentiable on the set of PSD matrices. Accordingly, they replace $X \succeq 0$ by the nonlinear, smooth constraints $d_j(X) \geq 0$ for $j = 1, ..., n$, and develop a specialized interior-point algorithm for solving the underlying semidefinite program. Burer and Monteiro (1998) consider linear semidefinite programs in the standard form to

$$\text{minimize } \{C \cdot X : A_i \cdot X = b_i \text{ for } i = 1, ..., m, \ X \succeq 0\},$$

where C and $A_i, i = 1, ..., m$ are symmetric $n \times n$ matrices, and where for any conformable square matrices $A = [A_{ij}]$ and $B = [B_{ij}]$, the dot product $A \cdot B$ is defined as the trace of $A^T B$, i.e., $A \cdot B = \sum_i \sum_j A_{ij} B_{ij}$. Also, here and throughout this paper, $X \succeq 0$ denotes that X is *symmetric and positive semidefinite*. Burer and Monteiro show that this problem can be solved as a nonlinear program in which X is replaced by a low-rank factorization RR^T, where R is an $n \times r$ matrix, with r taken as $\lceil \sqrt{2m} \rceil$. An augmented Lagrangian approach is then proposed to solve this resulting problem, using a limited-memory BFGS scheme for the inner-loop minimization process. However, the authors note that several local minima might exist, and offer no theoretical proof of convergence, although encouraging empirical results are presented.

Shor (1998) develops an alternative nondifferentiable optimization approach to semidefinite programming based on incorporating the nonsmooth convex constraint that restricts the smallest eigenvalue of X to be nonnegative. Given a symmetric $n \times n$ matrix X, if we denote the n real eigenvalues of X arranged in nondecreasing order by $\lambda_j(X), j = 1, ..., n$, then $X \succeq 0$ is equivalent to the condition that $\lambda_1(X) \geq 0$. Moreover, if we denote $\alpha^j \equiv \alpha^j(X), j = 1, ..., n$, as the set of linearly independent normalized eigenvectors corresponding to $\lambda_j(X), j = 1, ..., n$, then noting that $\lambda_j(X) = (\alpha^j)^T X \alpha^j \ \ \forall \ \ j = 1, ..., n$, we have that $X \succeq 0 \Longleftrightarrow \lambda_j(X) \geq 0$ for $j = 1, ..., n \Longleftrightarrow (\alpha^j)^T X \alpha^j \geq 0$ for $j = 1, ..., n$. It is interesting to note that as a function of symmetric matrices X, $\lambda_1(X)$ is a concave, but nondifferentiable, function (see Shor (1998), for example), although as demonstrated by Vanderbei and Benson (2000), the remaining eigenvalue functions $\lambda_j(X)$ for $j = 2, ..., n$, do not necessarily enjoy this concavity property. Furthermore, by the Raleigh-Ritz formula (which can be readily verified via the normalized eigen-basis diagonalization process), we have that

$$\lambda_1(X) = \min_{\|\alpha\|=1} (\alpha^T X \alpha).$$

Observe that as a function of X, λ_1 is hereby characterized as the minimum of a family of linear functions, and is therefore concave with a set of subgradients that can be characterized in terms of the normed eigenvectors α^* associated with $\lambda_1(X)$, where $\lambda_1(X) = \alpha^{*T} X \alpha^*$ for each such α^*. Accordingly, Shor (1998) incorporates the nonsmooth convex constraint $\lambda_1(X) \geq 0$ in the model formulation, in lieu of $X \succeq 0$, and proposes a nondifferentiable optimization strategy.

In this paper, we integrate the concepts of semidefinite programming and RLT to develop a class of semidefinite cuts that can be used to augment the RLT relaxation for any problem (discrete or continuous, linear or nonlinear) to which the latter technique is applicable. Given an RLT relaxation for any such problem, we show that we can further enhance this relaxation by incorporating an infinite class of particular RLT constraints that are based on semidefinite relationships. Rather than solve the resulting semi-infinite program, which in itself would require a specialized solution approach, we adopt the strategy of generating suitable members from the infinite constraint set as needed through a cutting plane or separation procedure. This separation routine is executed in polynomial time, thereby making the cut generation process efficient. The resulting set of cuts, which are in effect a form of special RLT constraints, are called *semidefinite cuts*. In essence, these cuts constitute a relaxation of the semidefinite constraint on the matrix of (second-order) variables. Moreover, each relaxation in this sequential process is a linear program whose solution can be updated using standard mathematical programming software. In addition, an upper bound can be computed by initializing a local search procedure with the solution obtained for the final relaxation. These bounds can be embedded within a branch-and-bound framework to determine a global optimum to the original problem.

Note that this concept of generating cutting planes based on semidefinite restrictions can be used to augment *any* RLT relaxation for discrete or continuous nonconvex programs, even *if* the overall relaxation contains sets of (nonlinear) convex constraints as in Sherali and Tuncbilek (1997). For example, Sherali and Wang (2001) have recently proposed a global optimization approach for solving general nonconvex factorable programs by integrating a polynomial approximation with an RLT scheme. In this context, our proposed approach can be applied identically by augmenting the simple nonnegativity and symmetry restrictions on the even-ordered RLT variables by a stronger positive semidefinite constraint, and then generating valid inequalities to tighten the relaxation in a manner similar to that exposed in the sequel.

As a point of *illustration* of this *general concept*, we will consider a specific example of the class of problems involving the minimization of a nonconvex quadratic objective function over a simplex (denoted **QP** below). This problem is interesting in its own right, and has been extensively studied by Nowak (1998a,b, 1999). It arises, for instance, in the context of finding a maximal weighted clique in an undirected graph.

$$\textbf{QP}: \qquad \text{Minimize} \quad \sum_i \sum_j C_{ij} x_i x_j \tag{1.1a}$$

$$\text{subject to } e^T x = 1 \tag{1.1b}$$

$$x \geq 0, \tag{1.1c}$$

where $x \in R^n$ and e is a vector of n ones. Although Problem QP is NP-Hard, it has a simple structure that makes it convenient to *illustrate the essence* of our approach, and extensions to more general problems are readily evident.

The first-level RLT relaxation RLT-1 (see Sherali and Tuncbilek, 1992) for Problem QP would multiply (1.1b) with each variable x_i, for $i = 1, ..., n$, and then substitute a nonnegative variable X_{ij} for each term $x_i x_j$ in the problem, where $X_{ij} \equiv X_{ji} \ \ \forall i, j = 1, ..., n$. To write this resulting problem in a specific manner that exposes connections with semidefinite programming and motivates our development, define $X \equiv [X_{ij}]$ to be an $n \times n$ (symmetric) matrix that represents the linearization of xx^T under the foregoing RLT substitution (i.e., $X \equiv [xx^T]_L$, where in general, $[\cdot]_L$ represents the standard linearization operation of RLT; in the present context, this involves the substitution of X_{ij} for the product term $x_i x_j$). Then, we can write the level-one RLT relaxation for QP in the form

$$\textbf{RLT}-\textbf{1}(\textbf{QP}): \qquad \text{minimize} \quad \sum_i \sum_j C_{ij} X_{ij} \tag{1.2a}$$

$$\text{subject to } e^T x = 1 \tag{1.2b}$$

$$Xe = x \tag{1.2c}$$

$$x \geq 0, X \geq 0 \ \text{ and symmetric.} \tag{1.2d}$$

Nowak (1998a,b, 1999) has proposed various SDP approaches for solving Problem QP. To derive a suitable semidefinite relaxation for QP, Nowak first employs the particular RLT constructs of multiplying the constraints $x_i \geq 0$ and $x_j \geq 0$ pairwise and squaring the constraint $e^T x = 1$, to derive the following quadratically constrained quadratic program (QQP).

$$\begin{aligned} \textbf{QQP}: \qquad & \text{Minimize} \quad \sum_i \sum_j C_{ij} x_i x_j \\ & \text{subject to } (e^T x)^2 = 1 \\ & x_i x_j \geq 0, \quad \forall 1 \leq i, j \leq n \\ & x \geq 0. \end{aligned}$$

By substituting $X = xx^T$, he then obtains a semidefinite relaxation for this representation as given by

$$\textbf{SDP}(\textbf{QQP}): \qquad \text{minimize} \quad C \cdot X \tag{1.3a}$$

$$\text{subject to } (ee^T) \cdot X = 1 \tag{1.3b}$$

$$X \geq 0 \tag{1.3c}$$

$$X \succeq 0 \tag{1.3d}$$

where $C = [C_{ij}]$. Nowak next constructs a convex quadratic function, $w(x) = x^T W x$, such that $W \leq C$ and $w(x)$ approximates $C \cdot X = x^T C x$. The matrix W is found by solving a separate semidefinite program. This produces an approximation for the convex envelope of the objective function, and the optimal solution to this convex program is used to provide an estimate for the global minimum of Problem

QP. Nowak has developed several lower bounding schemes for Problem QP, each based upon solving a different SDP problem to find W.

To illustrate our more general methodology, rather than focusing on such a specialized approach to Problem QP, we present in Section 2 an alternative semidefinite relaxation for Problem QP that is more closely associated with the usual RLT process, and which in fact yields a tighter relaxation than (1.3). This SDP relaxation is then shown to be equivalent to a suitable semi-infinite RLT relaxation. Based on this derivation, we develop a cutting plane strategy in Section 3. This strategy sequentially augments the first-level relaxation RLT-1(QP) with cutting planes that are automatically generated from the constraints in the semi-infinite representation using a special polynomial-time separation procedure. Several cut generation mechanisms are explored in this context. Thereafter, in Section 4, we demonstrate that potentially stronger classes of such cutting planes can be generated in a likewise fashion with comparable effort by simply replacing the semidefinite constraint $X \succeq 0$ by the restriction $X \succeq xx^T$, i.e.,

$$\begin{bmatrix} X & x \\ x^T & 1 \end{bmatrix} \succeq 0.$$

Computational results for employing cutting planes based on both these types of semidefinite constraints are provided in Section 5. Finally, Section 6 presents conclusions and suggestions for future research, including the extension of the proposed relaxation enhancement procedure to higher-level representations.

2. An alternative semidefinite relaxation of QP and its semi-infinite representation

There are several ways to construct a semidefinite relaxation for QP. One such formulation, suggested by Nowak, was presented above in (1.3). Alternatively, rather than squaring the simplex constraint, we can instead multiply it (on the right) by x^T as one would in an RLT approach, and use the substitution $X = xx^T$. Since X is symmetric, this yields the constraint $Xe = x$, which we append to the original problem. Relaxing $X = xx^T$ to $X \succeq 0$, we obtain the following semidefinite relaxation of QP. Note that from (2.4b,c), we get $(ee^T) \cdot X \equiv e^T Xe = e^T x = 1$, or that (1.3b) is implied. Hence, formulation (2.4) potentially yields a tighter relaxation of QP than that given by (1.3).

$$\begin{aligned} \textbf{SDP}(\textbf{QP}): \qquad \text{Minimize} \quad & C \cdot X && \text{(2.4a)} \\ \text{subject to} \quad & e^T x = 1 && \text{(2.4b)} \\ & Xe = x && \text{(2.4c)} \\ & x \geq 0, X \geq 0 && \text{(2.4d)} \\ & X \succeq 0. && \text{(2.4e)} \end{aligned}$$

We will now construct an equivalent semi-infinite linear programming restatement of Problem SDP(QP). This will facilitate the derivation of valid inequalities to augment the first-level RLT relaxation of Problem QP, given by (1.2). Consider the following result.

PROPOSITION 1 *The problem SDP(QP) given by* (2.4) *is equivalent to the semi-infinite linear program (SILP(QP)) stated in* (2.5) *below.*

$$\textbf{SILP}(\textbf{QP}): \quad \text{Minimize} \ \sum_i \sum_j C_{ij} X_{ij} \tag{2.5a}$$

$$\text{subject to } e^T x = 1 \tag{2.5b}$$

$$Xe = x \tag{2.5c}$$

$$[(\alpha^T x)^2]_L \geq 0, \ \forall \alpha \in R^n \ni \|\alpha\| = 1 \tag{2.5d}$$

$$x \geq 0, \ X \geq 0 \ \text{ and symmetric}. \tag{2.5e}$$

Proof. By definition, $X \succeq 0$ is equivalent to requiring that X is symmetric and that $\alpha^T X\alpha \geq 0$, $\forall \alpha \in R^n \ni \|\alpha\| = 1$, noting that any nonzero $\alpha \in R^n$ can be made of unit length. But $\alpha^T X\alpha = [\alpha^T(xx^T)\alpha]_L = [(\alpha^T x)(x^T\alpha)]_L = [(\alpha^T x)^2]_L$. Hence (2.4e) is equivalent to requiring X to be symmetric and such that (2.5d) holds true. This completes the proof. □

Proposition 1 reveals a connection between RLT and semidefinite relaxations. Observe that (2.5a, b, c, and e) are respectively identical to (1.2a, b, c, and d) that define the first-level RLT relaxation RLT-1(QP). The constraint set (2.5d) provides a potential strengthening of SDP(QP) or SILP(QP) over RLT-1(QP). The first-level RLT relaxation replaces the nonlinear substitution restriction $X = xx^T$ by simply requiring X to be nonnegative and symmetric. On the other hand, the semidefinite relaxation also requires X to satisfy the positive semidefiniteness condition associated with the identity $X = xx^T$. But note that as explored in Sherali and Tuncbilek (1997) and Audet et al. (2000), for example, aside from the minimal RLT representation constraints stated in (1.2) in the present context, the first-level RLT relaxation can optionally incorporate any other classes of linearized quadratic implied constraints. In particular, enhancing RLT-1(QP) with such implied restrictions of the type (2.5d) yields the semidefinite relaxation SDP(QP) as a special case. We therefore refer to the valid inequalities of the type (2.5d) as *semidefinite cuts* (or *SDP cuts*).

REMARK 1. As in Section 1, if we denote $\alpha^j \equiv \alpha^j(X)$, $j = 1, ..., n$, as the set of linearly independent normalized eigenvectors of X, then $X \succeq 0$ is equivalent to the condition that $(\alpha^j)^T X\alpha^j \geq 0$ for $j = 1, ..., n$. Hence, in the relationships embodied in (2.5d), we could focus on just the α-vectors corresponding to such eigenvectors of X, and generate violated members of these constraints in a relaxation framework based on detected negative eigenvalues. The Lanczos algorithm could be used for efficiently finding such extremal eigenvalues for this purpose (see

Paige (1972), for example). However, because of the complexity of this approach, given that X is a variable in the problem, we will find it more convenient to derive a (polynomial-time) separation mechanism for generating suitable members of (2.5d) in a sequential fashion, based on an LU factorization concept for X. □

3. Cutting plane generation scheme

Rather than solve the semi-infinite program SILP(QP) directly, we adopt the following relaxation approach which leads to a cutting plane generation strategy that can be applied in more general contexts. To begin with, we first solve SILP(QP) with the constraints (2.5d) omitted. Note that this relaxation corresponds precisely to the first-level RLT relaxation of QP as given by (1.2). Let us denote the resulting solution to this problem as $(\hat{x}, \hat{X})$. If $\hat{X} \succeq 0$, then $\hat{X}$ solves Problem SILP(QP) (or SDP(QP)) as well. Otherwise, the solution $\hat{X}$ violates at least one of the constraints (2.5d). The task now is to generate a suitable vector of unit length, $\alpha \in R^n$, for which the constraint $\alpha^T X \alpha \geq 0$ is not satisfied when $X = \hat{X}$, to yield a cutting plane of type (2.5d).

In essence, our solution recursively evaluates the entries of $\hat{X}$ to determine whether or not $\hat{X}$ is indeed positive semidefinite. Toward this end, consider the application of a superdiagonalization (or upper triangularization) process to the symmetric matrix $\hat{X}$ (see Bazaraa et al., 1993). In this process, proceeding in the order $i = 1, 2, ..., n$, we continue to zero out the elements in the i^{th} column under the current i^{th} diagonal element by performing elementary row operations using the i^{th} row, so long as the diagonal elements encountered remain positive. Starting with $G^1 \equiv \hat{X}$ for $i = 1$, at the i^{th} stage in this process, $i \in \{1, ..., n-1\}$, suppose that we have encountered all positive diagonal elements thus far, and that we are examining the reduced submatrix $G^i \in R^{(n-i+1)\times(n-i+1)}$ appearing in rows and columns $i, i+1, ..., n$. Let us view G^i in its partitioned form, where its first row and column are explicitly displayed as follows

$$G^i \equiv \begin{bmatrix} G^i_{11} & (g^i)^T \\ g^i & G \end{bmatrix}, \tag{3.6}$$

and consider the following result.

PROPOSITION 2 *Given G^i as in* (3.6), *suppose that $G^i_{11} > 0$ and define*

$$G^{i+1} = G - \frac{g^i (g^i)^T}{G^i_{11}}. \tag{3.7}$$

Then G^i is PSD if and only if G^{i+1} is PSD. Moreover, given any $\alpha^{i+1} \equiv (\alpha_{i+1}, ..., \alpha_n)^T$, by selecting

$$\alpha_i = \frac{-(\alpha^{i+1})^T g^i}{G^i_{11}},$$

we have that $(\alpha^i)^T G^i \alpha^i = (\alpha^{i+1})^T G^{i+1} \alpha^{i+1}$ *for*

$$\alpha^i \equiv \begin{pmatrix} \alpha_i \\ \alpha^{i+1} \end{pmatrix}.$$

Proof. By simplifying terms, we have from (3.6) and (3.7) that

$$(\alpha^i)^T G^i \alpha^i = G^i_{11}(\alpha_i + \frac{(\alpha^{i+1})^T g^i}{G^i_{11}})^2 + (\alpha^{i+1})^T G^{i+1} \alpha^{i+1}.$$

Clearly, if G^{i+1} is PSD, then so is G^i. Conversely, if G^i is PSD, then noting that by setting

$$\alpha_i \equiv \frac{-(\alpha^{i+1})^T g^i}{G^i_{11}} \text{ gives } (\alpha^i)^T G^i \alpha^i = (\alpha^{i+1})^T G^{i+1} \alpha^{i+1}, \tag{3.8}$$

we have that G^{i+1} is PSD. Moreover, in any case (3.8) holds true. This completes the proof. □

Note that the first part of Proposition 2 is based on the superdiagonalization procedure for checking the positive semidefiniteness of $\hat{X}$ (see Bazaraa et al., 1993). The related latter part of the result asserts that if G^i is not PSD, then since G^{i+1} must also not be PSD, we can seek an α^{i+1} such that $(\alpha^{i+1})^T G^{i+1} \alpha^{i+1} < 0$, and accordingly, we will have found an α^i, with its first component α_i given by (3.8), such that $(\alpha^i)^T G^i \alpha^i < 0$. We can repeat this process recursively until all components of α are determined. Upon normalizing this α, we will have generated a valid linear inequality of the form (2.5d) that is not satisfied for the current solution $\hat{X}$.

Next, consider the situation addressed by the following result in which for some stage $i \in \{1, ..., n-1\}$, we encounter a submatrix G^i of the type (3.6) for which $G^i_{11} = 0$ *and* $g^i \equiv 0$.

PROPOSITION 3 *Given* G^i *as in* (3.6), *suppose that* $G^i_{11} = 0$ *and that* $g^i \equiv 0$. *Then by letting*

$$G^{i+1} \equiv G \text{ and } \alpha_i = 0, \tag{3.9}$$

we have that G^i *is PSD if and only if* G^{i+1} *is PSD, and moreover,*

$$(\alpha^i)^T G^i \alpha^i = (\alpha^{i+1})^T G^{i+1} \alpha^{i+1} \text{ where } \alpha^i = \begin{pmatrix} \alpha_i \equiv 0 \\ \alpha^{i+1} \end{pmatrix}. \tag{3.10}$$

Proof. Similar to the proof of Proposition 2. □

This result asserts again that if G^{i+1} is not PSD and we find an α^{i+1} such that $(\alpha^{i+1})^T G^{i+1} \alpha^{i+1} < 0$, then we can recursively recover an α via (3.8) and (3.10) (according to whether the corresponding diagonal element is positive or zero, noting the condition of Proposition 3 in the latter case), such that (2.5d) is violated.

Now, let us consider two cases where for the *first time*, a situation other than the foregoing types arises.

Case (i): $G^i_{11} < 0$ in (3.6)**.**

Suppose that in the foregoing diagonalization process, we encounter for the first time a matrix G^i given by (3.6) having $G^i_{11} < 0$. In this case, we can take $\alpha^i = (\alpha_i, ..., \alpha_n)^T = (1, 0, ..., 0)$. Then $(\alpha^i)^T G^i \alpha^i = G^i_{11} < 0$, and we can subsequently compute the full vector α inductively using (3.8) and (3.10).

Case (ii): $G^i_{11} = 0$, but $G^i_{1j} = G^i_{j1} = \theta \neq 0$ for some $j \in \{2, ..., n-i+1\}$ in (3.6)**.**

In this case, we know that G^i is not PSD and we can find an α for which $\alpha^T \hat{X} \alpha \geq 0$ is violated as follows. Specifically, consider α^i to be of the form $\alpha^i = (\alpha_i, .., \alpha_n)^T = (\alpha_i, 0, , , .0, \alpha_{i+j-1}, 0, ..., 0)^T$. Let $G^i_{jj} = \phi, \xi = (\alpha_i, \alpha_{i+j-1})^T$, and

$$H = \begin{bmatrix} 0 & \theta \\ \theta & \phi \end{bmatrix}.$$

We then have that $(\alpha^i)^T G^i \alpha^i = \xi^T H \xi$, and if we can determine a ξ for which $\xi^T H \xi < 0$, we will have obtained an α^i for which $(\alpha^i)^T G^i \alpha^i < 0$. By using (3.8) and (3.10) recursively as before, we could thereby find an α for which $\alpha^T \hat{X} \alpha < 0$. This α could then be normalized to produce a valid inequality of the form (2.5d) that must be satisfied for all feasible solutions X. In order to determine such a vector α^i, consider the following result.

PROPOSITION 4 *Let* $\xi = (\alpha_i, \alpha_{i+j-1})^T$ *and let*

$$H = \begin{bmatrix} 0 & \theta \\ \theta & \phi \end{bmatrix},$$

where $\theta \neq 0$. *Then* $\xi^T H \xi$ *is minimized, subject to* $||\xi||^2 = 1$, *by selecting*

$$\alpha_i = \frac{1}{\sqrt{1 + \frac{\lambda^2}{\theta^2}}} \quad \text{and} \quad \alpha_{i+j-1} = \frac{\alpha_i \lambda}{\theta} \quad \text{where} \quad \lambda = \frac{\phi - \sqrt{\phi^2 + 4\theta^2}}{2}. \tag{3.11}$$

Moreover, at the solution (3.11), $\xi^T H \xi \equiv \lambda < 0$.

Proof. By the linear independence constraint qualification, the KKT necessary optimality conditions (see Bazaraa et al., 1993) for the problem of minimizing $\xi^T H \xi$ subject to $||\xi||^2 = 1$ yield for some λ,

$$H\xi = \xi\lambda, \ ||\xi||^2 = 1. \tag{3.12}$$

This implies that at any KKT solution, we have

$$\xi^T H \xi = \xi^T \xi \lambda = ||\xi||^2 \lambda = \lambda. \tag{3.13}$$

From (3.12) and (3.13), it follows that the optimal objective value sought equals the minimum eigenvalue λ of H, and the corresponding normalized eigenvector yields the optimal solution ξ. To find the minimum eigenvalue for H, consider the equation $det(H-\lambda I) = \lambda^2 - \phi\lambda - \theta^2 = 0$. Using the quadratic formula, we derive the minimum eigenvalue of H as

$$\lambda = \frac{\phi - \sqrt{\phi^2 + 4\theta^2}}{2}.$$

The corresponding eigenvector of H can be found via the system $(H - \lambda I)\xi = 0$, which gives $\alpha_{i+j-1} = \lambda\alpha_i/\theta$. Since, $||\xi||^2 = \alpha_i^2 + \alpha_{i+j-1}^2 = 1$, we have $\alpha_i = 1/\sqrt{1 + \frac{\lambda^2}{\theta^2}}$, where the positive square-root for computing α_i can be chosen without loss of generality. Furthermore, from (3.13), $\lambda = \xi^T H\xi < 0$ since X is not PSD. This completes the proof. □

REMARK 2. Note that in case of alternative choices of elements pertaining to Case (ii) for which Proposition 4 can be applied, we can select one that yields the most negative value of λ. □

Example 1. To illustrate, consider the following example. Suppose that the current solution $\hat{X}$ is given as follows:

$$\hat{X} = \begin{bmatrix} 0 & 0.15 & 0.15 \\ 0.15 & 0.2 & 0 \\ 0.15 & 0 & 0.2 \end{bmatrix}.$$

With $i = 1$, we have $G^i_{11} = 0$ and $G^i_{1j} = G^i_{j1} \neq 0$ for $j = 2$ and $j = 3$, indicating that there are two possible values of j that can generate a separating inequality. With $j = 2$, we have $G^i_{12} = G^i_{21} = 0.15$. This yields $\xi = (\alpha_1, \alpha_2)^T$ with $\theta = 0.15$ and $\phi = 0.2$ in the notation of Case (ii) above. From (3.11),

$$\lambda = \frac{0.2 - \sqrt{0.2^2 + 4(0.15)^2}}{2} = -0.08028,$$

$$\alpha_1 = \frac{1}{\sqrt{1 + \frac{(-0.08028)^2}{0.15^2}}} = 0.8817$$

and

$$\alpha_2 = \frac{(-0.08028)(0.8817)}{0.15} = -0.4719.$$

Hence, $\alpha = (0.8817, -0.4719, 0)^T$. Note that $||\alpha|| = 1$ and that

$$\alpha^T\hat{X}\alpha = \xi^T H\xi = (0.8817, -0.4719)\begin{pmatrix} 0 & 0.15 \\ 0.15 & 0.2 \end{pmatrix}\begin{pmatrix} 0.8817 \\ -0.4719 \end{pmatrix} = -0.08028,$$

which is the value of λ. In a similar fashion, we can calculate the corresponding α for $j = 3$ as $\alpha = (0.8817, 0, -0.4719)^T$, which also produces $\lambda = -0.08028$. Thus, the procedure has found two possible choices of α for which $\alpha^T X\alpha \equiv [(\alpha^T x)^2]_L \geq 0$ is not satisfied for the current solution $\hat{X}$. The corresponding linearized constraints are given as

$$0.7774X_{11} - 0.8321X_{12} + 0.2226X_{22} \geq 0$$

$$\text{and } 0.7774X_{11} - 0.8321X_{13} + 0.2226X_{33} \geq 0.$$

Since both of these cuts produced the same value of λ, we could arbitrarily choose either cut. □

The foregoing approach establishes an inductive polynomial-time process for generating valid inequalities for the first-level RLT relaxation. Since each recursive step of applying this process to G^i at iteration i is of complexity $O(n^2)$ and we perform at most n such steps, the complexity of the overall separation routine is $O(n^3)$. After obtaining an α for which $\alpha^T \hat{X}\alpha < 0, ||\alpha|| = 1$, and generating the corresponding inequality $[(\alpha^T x)^2]_L \geq 0$, we can append this to the current RLT relaxation. This problem could then be re-solved to obtain a new solution $(\hat{x}, \hat{X})$, and the procedure could be repeated until any of the following **termination criteria** is realized: the solution $\hat{X}$ for some relaxed problem turns out to be PSD, or some maximum limit K_1 on the number of LPs solved is attained, or the improvement in the lower bound from one iteration to the next is lesser than a prescribed $\delta > 0$ for some p consecutive iterations. Note that, as described in the sequel, we could generate multiple cuts at each iteration. Hence, we also impose a limit, K_2, on the number of inequalities of type (2.5d) that are generated for any particular solution $\hat{X}$. (In our computations, we used $K_1 = 100$, $K_2 = 100$, $\delta = 0.001$, and $p = 3$.) A flowchart for this approximate truncated scheme for solving SILP(QP) by way of augmenting RLT-1(QP) with the proposed SDP cuts is given in Figure 1.

Example 2. Suppose that the current solution $\hat{X}$ is given as follows.

$$\hat{X} = \begin{bmatrix} 0.04 & 0.08 & 0.2 \\ 0.08 & 0 & 0 \\ 0.2 & 0 & 0.4 \end{bmatrix}.$$

We can see that $\hat{X}$ is not PSD, since $\hat{X}_{22} = 0$ but $\hat{X}_{12} = \hat{X}_{21} = 0.08$. The procedure of Figure 1 starts with $i = 1$, $G^1 = \hat{X}$, and examines $G^1_{11} = \hat{X}_{11}$. Since $G^1_{11} > 0$, we store $G^1_{11} = 0.04$,

$$g^1 = \begin{pmatrix} 0.08 \\ 0.2 \end{pmatrix},$$

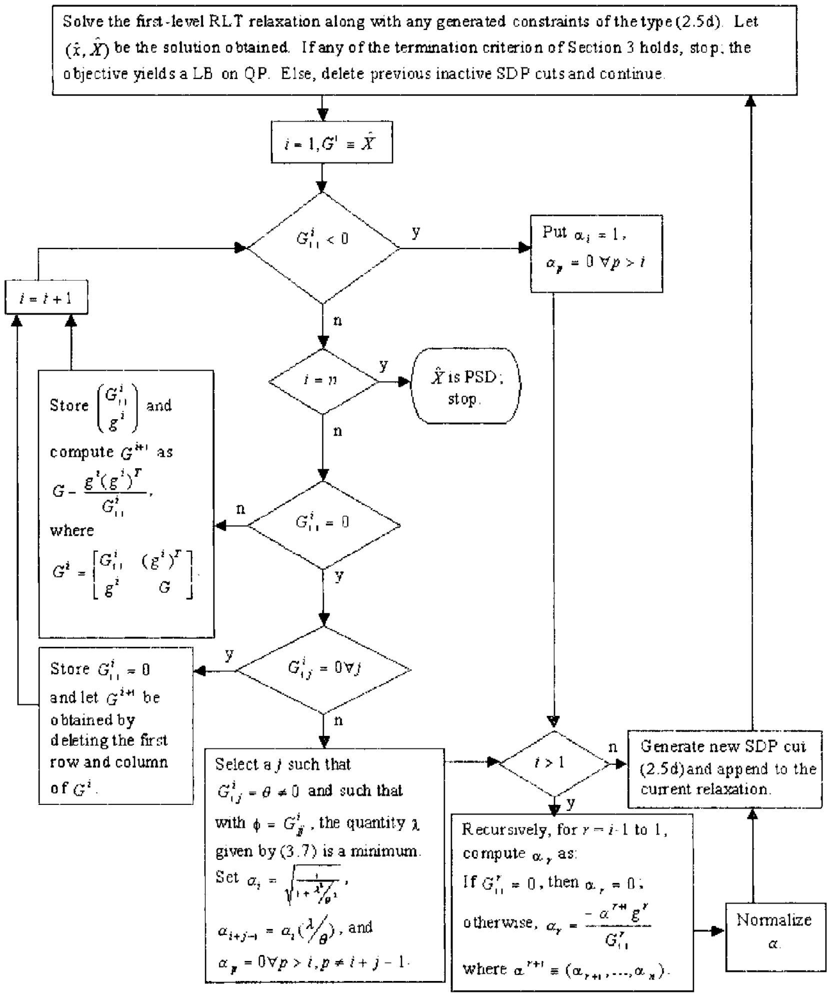
Solve the first-level RLT relaxation along with any generated constraints of the type (2.5d). Let $(\hat{x}, \hat{X})$ be the solution obtained. If any of the termination criterion of Section 3 holds, stop; the objective yields a LB on QP. Else, delete previous inactive SDP cuts and continue.
$i = 1, G^1 \equiv \hat{X}$
$G^i_{11} < 0$
y
Put $\alpha_i = 1$, $\alpha_p = 0 \,\forall p > i$
n
$i = n$
y
$\hat{X}$ is PSD; stop.
n
$i = i + 1$
Store $\begin{pmatrix} G^i_{11} \\ g^i \end{pmatrix}$ and compute G^{i+1} as $G - \frac{g^i(g^i)^T}{G^i_{11}}$, where $G^i = \begin{bmatrix} G^i_{11} & (g^i)^T \\ g^i & G \end{bmatrix}$.
n
$G^i_{11} = 0$
y
y
$G^i_{1j} = 0 \forall j$
Store $G^i_{11} = 0$ and let G^{i+1} be obtained by deleting the first row and column of G^i.
n
Select a j such that $G^i_{1j} = \theta \neq 0$ and such that with $\phi = G^i_{jj}$, the quantity λ given by (3.7) is a minimum. Set $\alpha_i = \sqrt{\frac{1}{1+\lambda^2/\theta^2}}$, $\alpha_{i+j-1} = \alpha_i(\lambda/\theta)$, and $\alpha_p = 0 \forall p > i, p \neq i+j-1$.
$i > 1$
n
Generate new SDP cut (2.5d) and append to the current relaxation.
y
Recursively, for $r = i\text{-}1$ to 1, compute α_r as: If $G^r_{11} = 0$, then $\alpha_r = 0$; otherwise, $\alpha_r = \frac{-\alpha^{r+1}g^r}{G^r_{11}}$ where $\alpha^{r+1} \equiv (\alpha_{r+1}, \dots, \alpha_n)$.
Normalize α.

Figure 1. Flow-chart for the Fundamental SDP Cut Generation Procedure.

and we derive the reduced matrix G^2 of Proposition 2 via (3.7) as

$$G^2 = \begin{bmatrix} 0 & 0 \\ 0 & 0.4 \end{bmatrix} - \frac{\begin{pmatrix} 0.08 \\ 0.2 \end{pmatrix}(0.08 \;\; 0.2)}{0.04} = \begin{bmatrix} -0.16 & -0.4 \\ -0.4 & -0.6 \end{bmatrix}.$$

At $i = 2$, $G^2_{11} = -0.16$ is negative. Hence, we take $\alpha^2 = (\alpha_2, \alpha_3)^T = (1, 0)^T$ which gives $(\alpha^2)^T G^2 \alpha^2 = -0.16$. At the final step in Figure 1, with $r = 1$, we compute $\alpha_1 = -(\alpha_2, \alpha_3) \cdot g^1/G^1_{11} = -2$ from Eq. (3.8). This yields $\alpha = (-2, 1, 0)^T$ with $\alpha^T \hat{X} \alpha = -0.16$. When we normalize α to $(\frac{-2}{\sqrt{5}}, \frac{1}{\sqrt{5}}, 0)^T$, we obtain $\alpha^T \hat{X} \alpha = -0.032$. The corresponding SDP cut is

$$[(\alpha^T x)^2]_L = 0.8X_{11} - 0.8X_{12} + 0.2X_{22} \geq 0,$$

which is violated for $X = \hat{X}$, since $[(\alpha^T x)^2]_L \equiv \alpha^T \hat{X} \alpha = 0.8(0.04) - 0.8(0.08) + 0.2(0) = -0.032$. □

REMARK 3. In the cut generation process described above, we have assumed that the matrix $\hat{X}$ is scanned with respect to its i^{th} diagonal element in the order $i = 1, ..., n$, and that a single SDP cut is generated once it is revealed that $\hat{X}$ is not PSD. There are several variations to this strategy that we could possibly adopt. One such variation is a *look-ahead feature* for the cut generation process. In this modification, when the matrix under consideration is G^i having dimension $n - i + 1$, we scan the entire diagonal (G^i_{qq} for $q = 1, ..., n - i + 1$) to see if any diagonal element is negative. If we find such a negative diagonal element, say $G^i_{QQ} > 0$, we take $\alpha_{i+Q-1} = 1$ and $\alpha_p = 0, \forall p \geq i, p \neq i + Q - 1$. As before, we use Eqs. (3.8) and (3.10) recursively to determine $\alpha_p \ \ \forall p \leq i - 1$ (if $i \geq 2$). In a similar manner, we can look ahead for cases where there is a diagonal element that equals zero, say, $G^i_{QQ} = 0$, but $G^i_{Qk} \neq 0$ for some $k \in \{1, ..., n - i + 1\}$, and generate a cut based on this revealed violation of positive semidefiniteness. Figures 2 and 3 provide detailed flow-charts of routines for implementing this look-ahead feature. Here, we use 'status = 0' to indicate that we should continue to increment i and look for additional cuts. Since it is only valid to increment i when either $G^i_{11} > 0$ or when $G^i_{1j} = G^i_{j1} = 0, \forall j = 1, ..., n - i + 1$, barring a further permutation of rows and columns of $\hat{X}$, we set 'status = 1' when either a Case (i) or Case (ii) violation is detected with respect to the leading element G^i_{11} of G^i. As a second variant of this strategy, whenever the leading element of the current reduced matrix G^i yields a Case (i) or Case (ii) violation, we generate the valid cuts as above, but instead of exiting from the cut generation routine, we examine if any of the other diagonal elements are positive. If so, we permute the rows and columns of G^i to make the most positive diagonal element as the leading element, and continue the cut generation process, taking care to record the appropriate order of the permuted indices for generating future cuts. Let us refer to this technique as the *full permutation strategy*. Since such a permutation strategy can consume significant computational effort, a third variant is developed in order to decrease computational effort while maintaining the benefits of permutation. In this variant, called the *diagonal sort strategy*, we perform an *nlog(n)* sort to arrange the diagonal elements in nonincreasing order, and we continue to generate cuts until we encounter a Case (i) or Case (ii) violation from the leading diagonal element. A fourth variant that applies to all of the foregoing strategies adds multiple cuts at each iteration, also using the look-ahead feature. Since there might be several distinct choices of α for composing SDP cuts as revealed during the sequential look-ahead process for the current solution $\hat{X}$, we attempt to generate a bundle of SDP cuts for each such $\hat{X}$ in order to possibly reduce the computational time

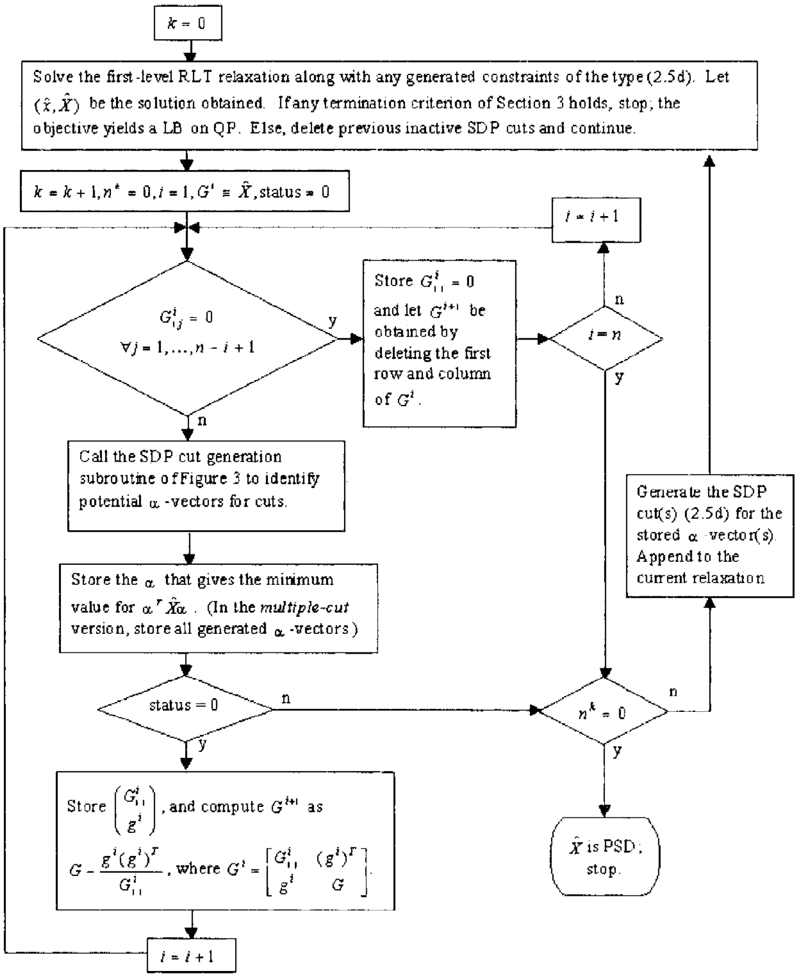
$k = 0$
Solve the first-level RLT relaxation along with any generated constraints of the type (2.5d). Let $(\hat{x}, \hat{X})$ be the solution obtained. If any termination criterion of Section 3 holds, stop; the objective yields a LB on QP. Else, delete previous inactive SDP cuts and continue.
$k = k+1, n^k = 0, i = 1, G^1 \equiv \hat{X}$, status = 0
$i = i+1$
$G^i_{1j} = 0$ $\forall j = 1,\ldots,n-i+1$
y
n
Store $G^i_{11} = 0$ and let G^{i+1} be obtained by deleting the first row and column of G^i.
$i = n$
n
y
Call the SDP cut generation subroutine of Figure 3 to identify potential α-vectors for cuts.
Generate the SDP cut(s) (2.5d) for the stored α vector(s). Append to the current relaxation
Store the α that gives the minimum value for $\alpha^T \hat{X} \alpha$. (In the *multiple-cut* version, store all generated α-vectors)
status = 0
n
$n^k = 0$
n
y
y
Store $\begin{pmatrix} G^i_{11} \\ g^i \end{pmatrix}$, and compute G^{i+1} as $G - \frac{g^i (g^i)^T}{G^i_{11}}$, where $G^i = \begin{bmatrix} G^i_{11} & (g^i)^T \\ g^i & G \end{bmatrix}$.
$\hat{X}$ is PSD; stop.
$i = i+1$

Figure 2. Flow-chart for the Look-Ahead SDP Cut Generation Procedure.

for the overall solution process. For all variants, we delete previously generated inactive cuts at each iteration. (We also implement an efficient check to avoid the generation of duplicated cuts.) In our experimental analysis, we will investigate both the single and multiple cut implementations, using both the original matrix $\hat{X}$ as well as an augmented matrix that will be considered in Section 4. □

Example 3. To illustrate the variants discussed in Remark 3, consider the matrix $\hat{X}$ from Example 2:

$$\hat{X} = \begin{bmatrix} 0.04 & 0.08 & 0.2 \\ 0.08 & 0 & 0 \\ 0.2 & 0 & 0.4 \end{bmatrix}.$$

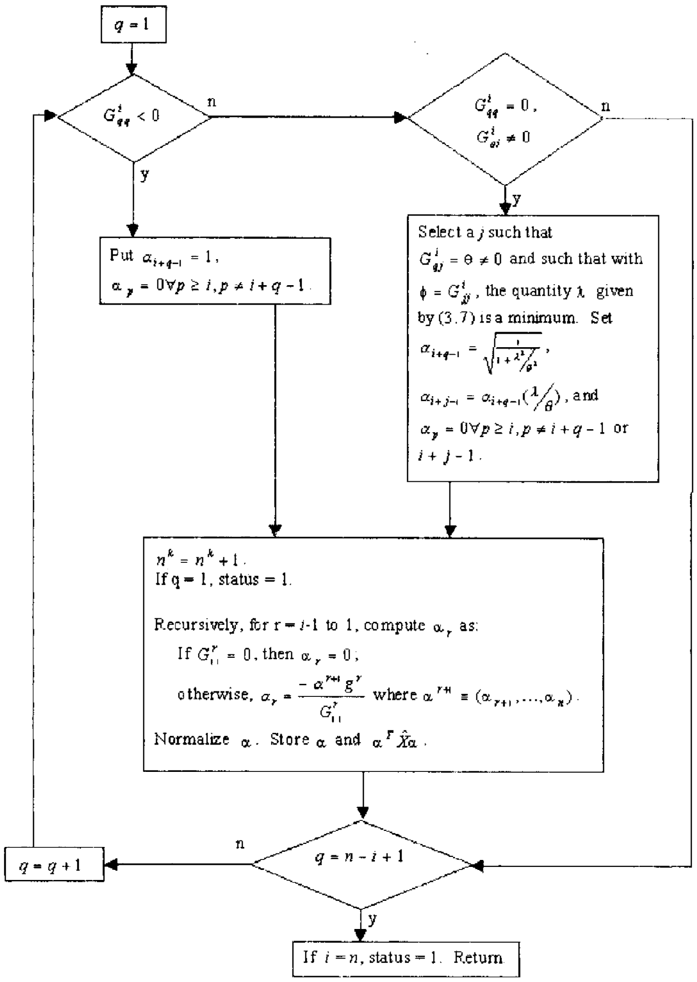
$q = 1$
$G^i_{qq} < 0$
n
y
$G^i_{qq} = 0$,
$G^i_{qj} \neq 0$
n
y
Put $\alpha_{i+q-1} = 1$,
$\alpha_p = 0 \forall p \geq i, p \neq i+q-1$.
Select a j such that
$G^i_{qj} = \theta \neq 0$ and such that with
$\phi = G^i_{jj}$, the quantity λ given
by (3.7) is a minimum. Set
$\alpha_{i+q-1} = \sqrt{\frac{1}{1+\lambda^2/\theta^2}}$,
$\alpha_{i+j-1} = \alpha_{i+q-1}(\lambda/\theta)$, and
$\alpha_p = 0 \forall p \geq i, p \neq i+q-1$ or
$i+j-1$.
$n^k = n^k + 1$.
If q = 1, status = 1.
Recursively, for r = i-1 to 1, compute α_r as:
If $G^r_{11} = 0$, then $\alpha_r = 0$;
otherwise, $\alpha_r = \frac{-\alpha^{r+1} g^r}{G^r_{11}}$ where $\alpha^{r+1} \equiv (\alpha_{r+1}, \ldots, \alpha_n)$.
Normalize α. Store α and $\alpha^T \hat{X}\alpha$.
n
$q = n - i + 1$
$q = q + 1$
y
If $i = n$, status = 1. Return

Figure 3. Flow-chart for the SDP Cut Generation Subroutine Invoked by the Look-Ahead Procedure of Figure 2.

With $i = 1$ and $G^1 = \hat{X}$, we can look-ahead and see that $\hat{X}_{22} = 0$ but $\theta = \hat{X}_{21} = \hat{X}_{12} = 0.08$. Accordingly, we can derive a violated constraint at this point itself, before incrementing i and examining G^2. If we take $\xi = (\alpha_2, \alpha_1)^T, \theta = \hat{X}_{12} = \hat{X}_{21} = 0.08$, and $\phi = \hat{X}_{11} = 0.04$, we obtain from Proposition 4 that $\alpha = (-0.6154, 0.7882, 0)^T$. The corresponding SDP cut is

$$0.3787X_{11} - 0.9701X_{12} + 0.6213X_{22} \geq 0,$$

which is currently violated since $\alpha^T \hat{X} \alpha = -0.0625$. Now, since $G^1_{11} > 0$, we could continue to increment i as before and generate the following SDP cut that was obtained in Example 2:

$$0.8X_{11} - 0.8X_{12} + 0.2X_{22} \geq 0,$$

with the corresponding $\alpha^T \hat{X} \alpha = -0.032$. Observe that the SDP cut generated by looking ahead had a violation nearly twice as large as the latter cut. In implementing the single-cut option of Figure 2, we would only add the first cut since we select the one cut that yields the largest violation. However, when using the multiple cut implementation of Figure 2, we would impose both of the above cuts before re-solving the current relaxation. □

REMARK 4. As a computational expedient, we have adopted the strategy to terminate the above cut generation process when either the resulting solution matrix $\hat{X}$ is PSD or when some practical stopping criterion is attained. In our computations, as indicated above, we set limits on the maximum number of cuts and iterations, as well as on the number of successive iterations performed while obtaining insufficient progress in tightening the lower bound. A question of interest that arises in this context is whether such a process can be induced to attain the ideal termination condition of $\hat{X}$ being PSD, even in an infinite convergence sense, if the other practical stopping criteria are omitted. One approach for attaining such a theoretically convergent process would be to impose a *spacer step*, whereby finitely often, a vector α is generated uniformly distributed on the surface of a unit sphere in R^n. Then, if $\hat{X}^*$ is the limiting matrix for some convergent subsequence of solutions $\hat{X}$ generated in an infinite process, we could not have the situation that there exists an $\overline{\alpha}$ for which $\overline{\alpha}^T \hat{X}^* \overline{\alpha} < 0$, because then there would exist an ϵ-neighborhood $N_\epsilon(\overline{\alpha})$ about $\overline{\alpha}$ for which $\alpha^T \hat{X}^* \alpha < 0 \ \forall \alpha \equiv N_\epsilon(\overline{\alpha}) \cap \{\alpha : \|\alpha\| = 1\}$. This would imply the absence of having generated any α in the latter region which has a nonzero measure on the surface of the unit sphere, a contradiction to the uniform distribution of the generated values of α on the surface of this sphere. □

4. SDP cuts using an augmented matrix

The SDP cuts derived in Section 3 for enhancing the RLT-1 relaxation have been developed by noting that the identity $X = xx^T$ implies the PSD restriction $X \succeq 0$. Another common tactic in semidefinite programming is to recognize that $X = xx^T$ also implies the stronger condition that $X \succeq xx^T$ (see Nowak (1998a,b, 1999), for example). Note that $X \succeq xx^T$ can be expressed as

$$\begin{bmatrix} X & x \\ x^T & 1 \end{bmatrix} \succeq 0.$$

From the viewpoint of RLT constraints (as per Proposition 1),

$$\begin{bmatrix} X & x \\ x^T & 1 \end{bmatrix} \succeq 0$$

translates to the class of SDP cuts

$$\left[\left(\alpha^T x + \alpha_{n+1}\right)^2\right]_L = \alpha^T X \alpha + 2\alpha_{n+1}\left(\alpha^T x\right) + \alpha_{n+1}^2 \geq 0$$

$$\forall \left(\alpha^T, \alpha_{n+1}\right) \ni \| \left(\alpha^T, \alpha_{n+1}\right) \| = 1.$$

In terms of the separation routine of the foregoing section, an identical procedure can be implemented on the matrix X^A, where

$$X^A = \begin{bmatrix} X & x \\ x^T & 1 \end{bmatrix}.$$

That is, given a solution $(\hat{x}, \hat{X})$, we can construct the matrix

$$\hat{X}^A = \begin{bmatrix} \hat{X} & \hat{x} \\ \hat{x}^T & 1 \end{bmatrix}$$

and apply the routine of Section 3 to the matrix $\hat{X}^A$ in lieu of $\hat{X}$.

Example 4. To illustrate the cut generation procedure using the foregoing augmented matrix, consider $\hat{X}$ as given in Example 1, and suppose that for all i, we have $\hat{x}_i = \sum_j \hat{X}_{ij}$ as required by (2.5c). This leads to the matrix $\hat{X}^A$ as follows:

$$\hat{X}^A = \begin{bmatrix} 0 & 0.15 & 0.15 & 0.3 \\ 0.15 & 0.2 & 0 & 0.35 \\ 0.15 & 0 & 0.2 & 0.35 \\ 0.3 & 0.35 & 0.35 & 1 \end{bmatrix}.$$

Since the upper left portion of the matrix contains $\hat{X}$, we can still derive the two SDP cuts that were obtained in Example 1. However, with the additional row and column of $\hat{X}^A$, we also have another possibility for generating an SDP cut inequality. With $i = 1$, we have $\hat{X}^A_{11} = 0$, but $\hat{X}^A_{14} = \hat{X}^A_{41} = 0.3 \neq 0$, and so we can apply Proposition 4 with $\xi = (\alpha_1, \alpha_4)^T, \theta = 0.3$, and $\phi = 1$. From (3.11), we get $\lambda = [1 - \sqrt{1^2 + 4(0.3)^2}]/2 = -0.0831, \alpha_1 = 0.9637$, and $\alpha_4 = -0.2669$. Hence,

$$\begin{pmatrix} \alpha \\ \alpha_{n+1} \end{pmatrix} = (0.9637, 0, 0, -0.2669)^T.$$

Note that

$$\| \begin{pmatrix} \alpha \\ \alpha_{n+1} \end{pmatrix} \| = 1$$

and that

$$\begin{pmatrix} \alpha \\ \alpha_{n+1} \end{pmatrix}^T \hat{X}^A \begin{pmatrix} \alpha \\ \alpha_{n+1} \end{pmatrix} = \xi^T H \xi = -0.0831,$$

which is the value of λ. The corresponding SDP cut is given by

$$-0.5145 x_1 + 0.9287 X_{11} \geq -0.07125,$$

which is currently violated, since we have $-0.5145\hat{x}_1 + 0.9287\hat{X}_{11} = -0.15435$. Thus, the procedure has found an

$$\begin{pmatrix} \alpha \\ \alpha_{n+1} \end{pmatrix}$$

for which

$$\begin{pmatrix} \alpha \\ \alpha_{n+1} \end{pmatrix}^T X^A \begin{pmatrix} \alpha \\ \alpha_{n+1} \end{pmatrix} \equiv [(\alpha^T x + \alpha_{n+1})^2]_L \geq 0$$

is not satisfied for the current solution $\hat{X}^A$. Recall that both of the cuts derived in Example 1 had $\alpha^T \hat{X} \alpha = -0.0803$; hence, examining the augmented matrix has produced a cut that is violated to a greater extent than the former cuts. In the single-cut option, we would therefore implement the cut that was generated by the present example, since it has a larger violation than either of the cuts generated in Example 1. In the multiple-cut implementation, we would append all of these generated cuts to the current RLT-1 relaxation before returning to re-solve the next relaxation. □

5. Computational results

To gauge the effectiveness of the proposed class of SDP cuts in solving Problem QP, we first conducted an experiment to evaluate the relative performance of the various cut generation strategies in enhancing the lower bound derived by RLT-1(QP) at the root node within a branch-and-bound framework. The first strategy, which serves as a baseline case, uses a single cut per iteration derived from the matrix $\hat{X}$ using no permutations. The remaining six strategies were composed by using each combination of the two matrix types (regular and augmented) with the three permutation types described in Remark 3 (no permutation, full permutation, and diagonal sort). These strategies are summarized in Table I. Since some preliminary computations indicated that the single cut approach was dominated by the multiple cut implementation, we consider the single cut strategy only in the baseline case. In addition to the stopping criteria mentioned in Section 3, we also limited each of the strategies to 60 s of CPU time per problem. (All computations were executed on a SUN Ultra-1 workstation, with CPLEX 6.5 being used to solve the generated LP relaxations.)

The sizes of the test problems range from 10 variables to 100, by increments of 10. For each problem, the objective coefficients were generated uniformly on the interval [0, 10]. The objective coefficients C_{ii} of the terms x_i^2 were always taken to be positive, while the coefficients C_{ij} of the terms $x_i x_j$ were permitted to be positive or negative. In order to vary the problem structure for a given size, the proportion of positive C_{ij} coefficients was varied through four values (0.1, 0.33, 0.66,

Table I. Summary of evaluated strategies.

Strategy	Number of cuts	Matrix type	Permutation strategy
1	Single	Regular	None
2	Multiple	Regular	None
3	Multiple	Regular	Full
4	Multiple	Regular	Diagonal sort
5	Multiple	Augmented	None
6	Multiple	Augmented	Full
7	Multiple	Augmented	Diagonal sort

0.9), and four problems were generated for each such value, creating a total of 16 problems for each problem size. We obtained a lower bound for each of these 160 problems using each of the seven proposed strategies. The data are summarized in Table II. For each problem, the SDP cut-enhanced bounds were all tighter than the RLT-1 bound, and the improvement was most pronounced with higher proportions of positive C_{ij} coefficients and smaller problem sizes. For instance, for the (four) 10-variable problems having 90% of the C_{ij} coefficients positive, the best SDP cut-enhanced bound improved the RLT-1 bound by an average of 65%; however, for the 100-variable problems having 10% of the C_{ij} coefficients positive, the SDP cut-enhanced bound only improved the RLT-1 bound by an average of 1.35%. In order to assess the relative performances of the different cut generation strategies, we ranked these methods for each problem size with respect to the bound obtained at the root node, as well as with respect to the CPU time required. For each problem, we computed the best (greatest) lower bound and the best (smallest) CPU time, and then calculated the percentage amount by which each method deviated from the best bound and time for the given problem. Since we have 16 problems of each size being solved using each of the seven strategies, this yields a total of 112 data points for each value of n. These data points pertaining to the bound and time deviations were ranked separately in increasing order for each value of n. In the case of ties, average ranks were assigned so that the sum of the ranks for each n equals $\sum_{i=1}^{112} i = 6328$. Tables III and IV present the rank-sums for each strategy for each value of n, as well as over the ten problem sizes, for the two respective criteria: lower bounds and CPU times.

The results indicate that the baseline strategy provides significantly worse bounds than its more sophisticated counterparts, but it has a slightly better than average performance with respect to computational time. When used with the regular matrix, the full permutation strategy provides a distinctly better bound than the non-permutation and diagonal-sort strategies, and this trend occurs across all problem sizes. Both permutation strategies (full or diagonal sort) provide a tighter lower bound than the non-permutation strategy when used in combination with the regular matrix, but the effect is less clear when used in combination with the

Table II. Average percentage improvement of the best SDP cut-enhanced bound over the RLT-1 bound.

Proportion	Number of Variables									
of $C_{ij} > 0$	10	20	30	40	50	60	70	80	90	100
0.1	45.63	34.76	25.01	15.94	11.50	7.40	5.23	3.41	1.28	1.35
0.33	56.72	43.86	29.46	21.48	14.13	10.25	5.69	4.18	2.38	1.43
0.66	59.30	55.44	46.18	35.40	24.20	19.94	14.37	10.25	7.34	5.85
0.9	65.18	64.28	59.69	58.10	52.38	44.76	39.61	33.02	28.11	19.14

Table III. Sum of lower bound rankings.

	Strategy (as defined in Table 1)						
n	1	2	3	4	5	6	7
10	1252	1011.5	798.5	803	1070.5	798	594.5
20	1290.5	1077	305	541.5	1125	945	1044
30	1153	895.5	293.5	661	936.5	1358	1030.5
40	1125	978	627	875	625	1410	688
50	1113.5	1086.5	735.5	992.5	673.5	1014	712.5
60	1165	1079	798.5	941.5	803.5	664	876.5
70	1124.5	1058	987.5	1013.5	729	654	761.5
80	1163	1100	1100	1100	633	673.5	558.5
90	1090.5	1071.5	1071.5	1071.5	730	502	791
100	1078.5	1099	1045	1068.5	747.5	487.5	802
Total	11555.5	10456	7762	9068	8073.5	8506	7859

Table IV. Sum of CPU time rankings.

	Strategy (as defined in Table 1)						
n	1	2	3	4	5	6	7
10	869	759	1108	930	415	1269.5	977.5
20	636	732	1225.5	1067.5	467.5	1478.5	721
30	869	853	1227	1172	716.5	799.5	691
40	1014	864	1099	1132	823	739	657
50	913	807	1128	1325	590	891.5	673.5
60	928.5	1026	992	1180	496	1149.5	556
70	1233	875	874	1175	559.5	936.5	675
80	579.5	818.5	1617	934	369	1215	795
90	752	934	944.5	1137.5	510	1254	796
100	985	1023.5	892.5	1101	405	1181.5	739.5
Total	8779	8692	11107.5	11154	5351.5	10914.5	7281.5

Table V. h-Statistic for the Kruskal–Wallis test.

	h-Statistic	
n	Lower bound	CPU time
10	17.11	26.23
20	43.83	46.55
30	42.49	16.08
40	30.37	11.76
50	12.76	23.34
60	10.65	26.5
70	12.13	21.07
80	25.38	60.25
90	19.18	21.90
100	19.19	24.47

augmented matrix strategy. There are several notable cases where the permutation strategy does not tighten the bounds obtained from the non-permuted method. In general, the augmented matrix strategy provides an improvement in bounds as compared to the regular matrix strategy, particularly as problem size increases. Overall the rankings indicate that Strategies 3 and 7 provide the best lower bounds, although Strategies 4, 5, and 6 are also competitive. Note that Strategy 3 performs better for smaller problems, while Strategies 5, 6, and 7 tend to perform better as the problem size increases. From Table IV, we see that, in general, the methods using the augmented matrix tend to require less computational time, with Strategies 5 and 7 emerging as clearly more time-efficient. Based upon the rankings shown in Tables III and IV, it appears that Strategy 7 provides desirable results in terms of both the quality of the lower bound obtained and the amount of CPU time consumed. In particular, it seems promising that Strategy 7 also performs well in both categories as problem size increases.

In order to determine whether or not the differences in the strategy rankings were significant, we performed a Kruskal–Wallis (rank-sum) test on the data for each n for the seven strategies. Table V indicates that the CPU times were significantly different at the 5% level $\left(h > \chi^2_{0.05,6} = 12.592\right)$ for each problem size other than for $n = 40$, and that the lower bounds were significantly different for all sizes except for $n = 60$ and $n = 70$. We performed an additional Kruskal–Wallis test by analyzing the combined data from all problem sizes. That is, we ranked each of the percentage deviations from 1 through 1120 $(= 160 \times 7)$, and performed the Kruskal–Wallis test using a sample size equal to 160 for each strategy. The test statistics for the lower bounds and CPU times were 66.77 and 119.9, respectively, which were much greater than $\chi^2_{0.05,6} = 12.592$, indicating that there were significant differences in the performance of the seven strategies.

Table VI. Number of problems for which the best lower bounds and CPU times were achieved for each strategy.

	Strategy (as defined in Table 1)						
	1	2	3	4	5	6	7
Lower bound	12	16	46	27	48	67	49
CPU time	18	9	1	3	83	26	22

As final comparative evidence, we directly display in Table VI the number of problems (out of 160) for which each strategy obtained the best lower bound and CPU time. The strategies that use the augmented matrix have the largest proportion of best lower bounds and best CPU times. Of the strategies based on the regular matrix, the ones that employed the full permutation and the diagonal sort techniques performed significantly better than the one that used no permutation.

Based upon this information, we narrowed our study to exploring the performance of using Strategies 3, 4, 5, 6, and 7 to generate SDP cuts wihtin a branch-and-bound framework. As a benchmark in this comparison, we also implemented the RLT-1 strategy without any cutting planes for computing lower bounds. In order to obtain the RLT representation for the current branch-and-bound node, we augment (1.2) with the constraints obtained by multiplying the bound-factors pairwise. Note that we only include these bound-factor product constraints of Sherali and Tuncbilek (1992) when the corresponding bounds are tighter than the implied bounds of 0 and 1. For each of the Strategies 3, 4, 5, 6, and 7, we also include the corresponding SDP cuts that were generated at the nodes on the chain connecting the current node to the root node in the enumeration tree. These cuts are likely to be most effective for the current node subproblem, although the cuts generated elsewhere in the tree are also valid. (In case this number exceeds the maximum allowable number of implemented cuts, we overwrite the cuts that were generated the earliest.) Since we have linear constraints, the LP solution for each node subproblem also provides an upper bound. At each stage of the branch-and-bound scheme, we select nodes based on the least lower bound rule. For the partitioning strategy, we select a branching variable, x_p, as given by

$$p \in \text{argmax}_{i=1,\ldots,n}\{\delta_i = |\sum_j C_{ij}(\hat{x}_i\hat{x}_j - \hat{X}_{ij})|\},$$

and we split the current interval $[\ell_p, u_p]$ at the value $\tilde{x}_p$ in order to derive two children nodes, where

$$\tilde{x}_p = \begin{cases} \hat{x}_p, & \text{if min } \{\hat{x}_p - \ell_p, u_p - \hat{x}_p\} \geq 0.1(u_p - \ell_p) \\ \frac{\ell_p+u_p}{2}, & \text{otherwise.} \end{cases}$$

This induces convergence to a global optimum (see Sherali and Tuncbilek, 1992). In our experimental analysis, we fathomed nodes when the lower bound exceeded

$(1-\epsilon)z_{\text{upper}}$, where z_{upper} is the best-known solution value. In our computations, we used $\epsilon = 0.0001$ for the 10- and 20-variable problems, and we used $\epsilon = 0.01$ for the 30-variable problems. Furthermore, we permitted a maximum of 10,000 nodes for the branch-and-bound routine when using RLT alone, and a maximum of 1000 nodes for the SDP cut-enhanced procedures. We also limited a maximum of 100 cuts to be generated per iteration, and we limited such sequential rounds of cuts per node to either one or five (as specified). The maximum number of stored cuts was taken as three times the maximum number of cuts that could be generated at any given node (i.e., 300 for the one-round-of-cuts limit and 1500 for the five-rounds-of-cuts case).

Tables VII–X display the results obtained for this branch-and-bound experimentation. Note that in all of these tables, the SDP cut strategies are numbered according to the order shown in Table I, and the baseline RLT strategy using no SDP cuts is referred to simply as RLT. Table VII presents the results obtained for the 10-variable problems, and it shows that for nearly every implementation strategy, the SDP cuts provide a *significant* improvement in the performance of the branch-and-bound algorithm over that using the RLT-1 relaxations alone. The SDP cuts greatly reduce the number of nodes generated as might be expected, but also substantially reduce the overall computational effort. Within the SDP cut-enhanced strategies, using five rounds of SDP cuts per node significantly reduces the number of nodes enumerated as compared with using a single round of SDP cuts; however, the computational time is not consistently reduced. In general, using five founds of cuts proves most valuable for the relatively more difficult problem instances (lower proportions of positive C_{ij} coefficients), and it does not appear to work well in conjunction with the no permutation strategy. Based upon the results from the 10-variable problems, it was evident that Strategy 5 (augmented matrix, no permutation) would not remain competitive for the more difficult problems, and Strategy 5 was dropped from consideration for the remaining analysis.

The results for the 20-variable problems are presented in Tables VIII and IX. Table VIII displays the average time and number of nodes for the various problem types and implementation strategies. Note that in contrast to the results for the 10-variable problems, several problems were not solved to optimality within the allowable number of nodes. In such cases when the gap between the best-known solution and least lower bound did not fall below 0.01%, we recorded the percentage gap at termination, and we summarize these results in Table IX. Note that although several SDP cut strategies do not significantly decrease the computational effort, they do significantly tighten the optimality gap. Similarly, the use of five rounds of cuts generally provides better results than one round of cuts across nearly all strategies, either by tightening the optimality gap or by decreasing computational effort. The striking result in Table IX is that one strategy, Strategy 4 (regular matrix, diagonal sort) used in combination with five rounds of cuts, obtained the optimal solution (within the allowable number of nodes) for every problem. Although Strategy 7 (augmented matrix, diagonal sort) with five rounds

Table VII. Average computation time (in seconds) and average number of nodes for problems of size (n = 10).

		Proportion of positive C_{ij} coefficients							
Strategy	Rounds	0.1		0.33		0.66		0.9	
	of cuts	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes
RLT	0	482.41	5657.5	65.84	1006.5	10.95	239.5	0.74	27
3	1	152.24	339	24.15	121.5	3.42	40	0.40	11
	5	135.90	92	32.79	44	4.55	17.5	0.79	7
4	1	88.16	195.5	22.77	118	4.00	40.5	0.45	11
	5	62.69	40.5	21.84	34.5	3.94	16.5	0.73	7.75
5	1	232.89	422	39.62	162.5	5.89	43.5	0.65	12
	5	419.18	205	109.76	96	9.21	23.5	0.70	3.5
6	1	187.61	360.5	31.03	127	7.84	51	0.65	12.5
	5	160.80	95.5	33.79	41.5	5.44	13.5	0.66	3.5
7	1	69.43	117	24.46	87	4.53	28.5	0.71	10.5
	5	52.76	32	29.80	26	4.21	9	0.81	3.5

Table VIII. Average computation time (in seconds) and average number of nodes for problems of size (n = 20).

		Proportion of positive C_{ij} coefficients							
Strategy	Rounds	0.1		0.33		0.66		0.9	
	of cuts	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes
RLT	0	6485.25	10001	3917.5	7133.5	86.02	371.5	5.91	51.5
3	1	5256	978.5	1304.75	426	25.31	61	3.44	22
	5	5695.75	534	887.75	130	25.04	22.5	3.83	9.5
4	1	6025.25	990.5	1162.25	367.5	24.82	60	4.11	27
	5	2775.25	323.5	638.75	96	28.34	24.5	3.37	9.5
6	1	5604	1001	2509.5	723.5	44.76	75	5.05	20
	5	10414.5	922	2625	311	95.64	26.5	4.93	7
7	1	6683.75	1001	1910	447	52.26	46	4.81	19.5
	5	6478.75	503.5	1450.25	157.5	79.77	21.5	7.78	8

of cuts also obtained the global optimum for all problems except one, it did not perform as well with respect to computational effort. Note that Strategy 4, with five rounds of cuts, dominated the other strategies in terms of both the average number of nodes enumerated and the average computational effort, particularly for the more difficult set of problems.

Based upon the results obtained for the 20-variable problems, we used only one SDP cut strategy, Strategy 4 with five rounds of cuts, to solve the 30-variable problems. The results comparing this strategy with the basic RLT scheme are shown in Table X. For the RLT bounding strategy, seven problems could not be solved to

Table IX. Average percentage optimality gap at termination for problems of size (n = 20).

		Proportion of positive C_{ij} coefficients			
Strategy	Rounds of cuts	0.1	0.33	0.66	0.9
RLT	0	7.07	0.78	0	0
3	1	2.02	0	0	0
	5	0.17	0	0	0
4	1	1.26	0	0	0
	5	0	0	0	0
6	1	5.71	0.02	0	0
	5	2.01	0	0	0
7	1	2.71	0	0	0
	5	0.06	0	0	0

Table X. Average results for problems of size (n = 30).

Proportion of	Time		Nodes		% Gap	
$C_{ij} > 0$	RLT	SDP Cuts	RLT	SDP Cuts	RLT	SDP Cuts
0.1	20574.5	12237	10001	499.5	9.50	0.95
0.33	17640.5	8554	9613	437.5	4.7	0
0.66	1559.75	380	1370.5	65.5	0	0
0.9	125.75	58	209.5	25	0	0

optimality (using a 1 tolerance) within the 10000 node limit, while the SDP cut-enhanced strategy failed to solve only one problem to global optimality within 1000 nodes. Furthermore, the SDP cuts drastically decrease the average computational effort as well as the number of nodes enumerated across all problem types. The overall results appear to indicate that the SDP cuts significantly decrease the computational effort and the number of nodes required to solve this class of problems to optimality. Moreover, this relative improvement becomes more pronounced as the degree of difficulty of the problem increases (larger n, smaller proportion of positive C_{ij} coefficient).

6. Conclusions and extensions

In this paper, we have explored connections between semidefinite programming (SDP) and the Reformulation-Linearization Technique (RLT), and have used this insight to develop a new class of semidefinite cuts to enhance RLT relaxations. This concept has been illustrated on a set of problems involving the minimization of a nonconvex quadratic function over a simplex. The process of closing the gap between a first-level RLT relaxation and a semidefinite relaxation for this problem was shown to yield an equivalent semi-infinite linear program in which the set of infinite constraints comprised a particular type of RLT constraints that we called semidefinite cuts (or SDP cuts). Based on this representation, a relaxation and row generation scheme was devised, leading to a polynomial-time SDP cut generation procedure. Although this cut generation process is quite efficient in practice, its worst-case complexity of $O(n^3)$ could be an obstacle for large-scale problems, particularly if the matrix of second-order variables turns out to be dense. Several cut generation strategies, based on using the original or augmented matrix of second-order variables, in natural or specially permuted form, were devised and tested. The SDP cut-enhanced relaxations not only provided significantly tighter lower bounds, but when embedded within a branch-and-bound framework to determine a global optimal solution, resulted in a substantial decrease in both the number of nodes enumerated and in the overall computational effort, particularly for more challenging problem instances. Of the proposed implementation strategies, the use of multiple cuts clearly dominated the single-cut approach, and the permutation and augmented matrix implementations also provided improved results for some problems. For the most challenging problems, the best combined strategy by far used five founds of SDP cuts at each node, generated via the regular matrix of second-order RLT variables, rearranged using the diagonal sort permutation strategy.

Observe that the proposed class of SDP cuts can be used in any context where RLT is applied. This includes problems having polynomial objective and constraint functions, factorable programming problems, or even linear mixed-integer programming problems. In all such cases, SDP cuts can be generated based on the (regular or augmented) matrix of second-order RLT variables. For example, consider an RLT relaxation that includes fourth-order RLT variables $X_{ijk\ell}$ representing the product term $x_i x_j x_k x_\ell, \forall 1 \leq i \leq j \leq k \leq \ell \leq n$. Let $X_{(2)}$ denote the *vector* comprising all distinct

$$\binom{n+1}{2}$$

second-order RLT variables, and let $X^{(4)}$ be a *matrix* comprised of the fourth-order RLT variables structured in the form $X^{(4)} \equiv [X_{(2)}X_{(2)}^T]_L$. Since $X^{(4)}$ must be PSD, we can impose a class of SDP cuts in the same spirit as (2.5d) in the form

$$\alpha^T X^{(4)}\alpha \equiv [(\alpha^T X_{(2)})^2]_L \geq 0 \forall \alpha \in R^{\binom{n+1}{2}} \ni \|\alpha\| = 1. \tag{6.14}$$

Then, given any $\hat{X}^{(4)}$ as part of a solution to the RLT relaxation, we can use the techniques of Section 3 identically to derive SDP cuts of the type (6.14) involving the higher dimensional variables.

We also observe that there is another viewpoint that can be adopted in this context by defining $A_{ijk\ell} = \alpha_i\ \alpha_j\ \alpha_k\ \alpha_\ell$, and imposing the semidefinite constraint $A \cdot X \geq 0$ for *any* vector $\alpha \in R^n$, where $A \cdot X = \sum_i \sum_j \sum_k \sum_\ell A_{ijk\ell} X_{ijk\ell}$, simply because

$$A \cdot X = \left[\left(\alpha^T x \right)^4 \right]_L .$$

This leads to the class of RLT constraints

$$[(\alpha^T x)^4]_L \geq 0, \forall \alpha \in R^n \ni \|\alpha\| = 1. \qquad (6.15)$$

However, in this case, we would need to devise a procedure for generating violated members of the constraints (6.15), if any exist, via a separation routine or a cutting plane-generation process. The extension of cuts of type (6.14) or (6.15) to RLT variables of general even order is evident. We propose the task of investigating the generation of such cuts and testing their implementation for enhancing higher order RLT relaxations for future research.

References

Alizadeh, F. (1995) Interior Point Methods in Semidefinite Programming with Applications to Combinatorial Optimization. *SIAM Journal of Optimization* 5(1), 13–51.

Audet, C., Hansen, P. Jaumard, B. and Savard, G. (2000) Branch and Cut Algorithm for Nonconvex Quadratically Constrained Quadratic Programming. *Mathematical Programming* 87(1), 131–152.

Bazaraa, M. S., Sherali, H. D. and Shetty, C. M., (1993) *Nonlinear Programming Theory and Applications*, Wiley, New York, NY, second edition.

Bertsimas, D. and Ye, Y. (1998) Semidefinite Relaxations, Multivariate Normal Distributions and Order Statistics. In: D.-Z. Du and P. M. Pardalos (eds.): *Handbook of Combinatorial Optimization*, Vol. 3. Kluwer Academic Publishers, pp. 1–19.

Burer, S. and Monteiro, R. (1998) A Nonlinear Programming Algorithm for Solving Semidefinite Programs via Low-rank Factorization. Presented at the ISMP Conference, Atlanta, GA.

Goemans, M. and Williamson, D. (1995), Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming *Journal of the Association for Computational Machinery* 42(6), 1115–1145.

Nowak, I. (1998) A Global Optimality Criterion for Non-Convex Quadratic Programming Over a Simplex . Pre-Print 98–17, Humboldt University, Berlin. Available online at http://www-iam.mathematik.hu-berlin.de/ivo/ivopages/work.html.

Paige, C.C. (1972) Computational Variants of the Lanczos Method for the Eigen-problems. *Journal of the Institute of Mathematics and Its Applications* 10, 373–381.

Ramana, M. and Goldman, A. J. (1995), Some Geometric Results in Semidefinite Programming. *Journal of Global Optimization* 7, 33–50.

Ramana, M. and Pardalos, P. M. (1996) Semidefinite Programming. In: T. Terlaky (ed.): *Interior Point Methods of Mathematical Programming*. Kluwer Academic Publishers, Dordrecht, pp. 369–398.

Sherali, H. D. and Adams, W. P. (1999), *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*. Boston, MA: Kluwer Academic Publishing.

Sherali, H. D. and Tuncbilek, C. H. (1992), A Global Optimization Algorithm for Polynomial Programming PRoblems Using a Reformulation-Linearization Technique. *Journal of Global Optimization* 2, 101–112.

Sherali, H. D. and Tuncbilek, C. H. (1997), Reformulation-Linearization/Convexification Relaxations for Univariate and Multivariate Polynomial Programming Problems. *Operations Research Letters* 21(1), 1–10.

Sherali, H. D. and Wang, H. (2001), Global Optimization of Nonconvex Factorable Programming Problems. *Mathematical Programming* 89(3), 459–478.

Shor, N. Z. (1998), *Nondifferentiable Optimization and Polynomial Problems*. Boston, MA: Kluwer Academic Publishing.

Todd, M. J. (1998), Semidefinite Programming Applications, Duality, and Interior-Point Methods. Presented at the Fall INFORMS meeting, Seattle, WA. Also available on the World Wide Web at http://www.orie.cornell.edu/mike-todd/todd.html.

Vandenbergh, L. and Boyd, S. (1996), Semidefinite Programming. SIAM Review 38(1), 49–95.

Vanderbei, R.J. and Benson, H. Y. (2000), On Formulating Semidefinite Programming Problems as Smooth Convex Nonlinear Optimization Problems. Working paper, Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ.

Wolkowicz, H., Saigal, R. and Vandenbergh, L. 2000, *Handbook of Semidefinite Progamming: Theory and Algorithms, and Applications*. Boston, MA: Kluwer Academic Publishers.