Research Article

A New Spatial Branch and Bound Algorithm for Quadratic Program with One Quadratic Constraint and Linear Constraints

Jing Zhou

College of Science, Zhejiang University of Technology, Hangzhou, Zhejiang 310023, China

Correspondence should be addressed to Jing Zhou; zhoujing@zjut.edu.cn

Received 17 September 2019; Accepted 2 January 2020; Published 25 February 2020

Academic Editor: Francesco Riganti-Fulginei

Copyright © 2020 Jing Zhou. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper proposes a novel second-order cone programming (SOCP) relaxation for a quadratic program with one quadratic constraint and several linear constraints (QCQP) that arises in various real-life fields. This new SOCP relaxation fully exploits the simultaneous matrix diagonalization technique which has become an attractive tool in the area of quadratic programming in the literature. We first demonstrate that the new SOCP relaxation is as tight as the semidefinite programming (SDP) relaxation for the QCQP when the objective matrix and constraint matrix are simultaneously diagonalizable. We further derive a spatial branch-and-bound algorithm based on the new SOCP relaxation in order to obtain the global optimal solution. Extensive numerical experiments are conducted between the new SOCP relaxation-based branch-and-bound algorithm and the SDP relaxation-based branch-and-bound algorithm. The computational results illustrate that the new SOCP relaxation achieves a good balance between the bound quality and computational efficiency and thus leads to a high-efficiency global algorithm.

1. Introduction

In this paper, we consider a quadratic program with one quadratic constraint and several linear constraints in the following form:

$$\begin{align*}
\text{min} & \quad f(x) = x^T Q_0 x + q_0^T x, \\
\text{s.t.} & \quad x^T Q_1 x + q_1^T x \leq c, \\
& \quad D x \leq d,
\end{align*}$$

(1)

where $x \in \mathbb{R}^n$ is the decision variable, $Q_i \in \mathbb{R}^{n \times n}$ is a symmetric matrix, and $q_i \in \mathbb{R}^n$ for $i = 0, 1$, $D \in \mathbb{R}^{m \times n}$, $d \in \mathbb{R}^m$ and $c \in \mathbb{R}$. (1) covers many important combinatorial optimization problems and engineering problems, such as the binary quadratic programming problem [1], the max-cut problem [2], the quadratic knapsack problem [3,4], the binary least-square problem [5], the image processing problem [6], the multiuser detection problem [7], the project selection and resource distribution problems [8], the multisensor beamforming problem [9], and the system equilibrium problem [10]. It is known that if $Q_0$ and $Q_1$ are all positive semidefinite, then the problem becomes convex and can be solved efficiently by using SOCP methods [11]. In this paper, we assume that $Q_0$ is positive semidefinite while $Q_1$ is not, and the feasible region of (1) is bounded with nonempty relative interior points.

As a quadratically constrained quadratic programming problem, a typical approach to solve (1) is to exploit the branch-and-bound methods [12,13]. Note that the tightness and the computing efficiency of solving the convex relaxation problem of (1) at each node are critical factors that can affect the performance of the branch-and-bound algorithm. Thus, designing an effective convex relaxation of (1) is a hot topic in the literature. To the best of my knowledge, the SDP relaxation has become an attractive approach for obtaining good convex relaxations [14,15]. Though the SDP relaxation is tighter, it would also tremendously enlarge the dimension of the problem by lifting the original $n$-dimensional variable vector $x$ to an $(n+1) \times (n+1)$ variable matrix. Consequently, solving the SDP relaxation of a huge-size problem
should take a very long computational time. Hence, designing a convex relaxation which can be efficiently solved even for huge-size problem while maintaining the strength of the convex relaxation are investigated in the literatures [16, 17].

In this paper, we will develop a new SOCP relaxation for (1) employing the simultaneous matrix diagonalization technique. We first give the following definition.

**Definition 1.** \( M \in \mathbb{R}^{n \times n} \) and \( N \in \mathbb{R}^{m \times m} \) are called simultaneously diagonalizable (SD) if there exists a nonsingular matrix \( F \in \mathbb{R}^{n \times n} \) such that \( F^T MF \) and \( F^T NF \) are both diagonal matrices.

In the past few years, many researches have shown that for some quadratically constrained quadratic programming problems, the simultaneous matrix diagonalization technique that is embedded into a convex relaxation may result in a tighter convex relaxation than that is not. Ben-Tal and Den Hertog [18] showed that a quadratic program with one or two quadratic constraints has a hidden conic quadratic representation if the matrices in the quadratic forms are simultaneously diagonalizable. Jiang and Li [19] also applied the simultaneous diagonalization techniques to solve a quadratic programming problem. Zhou and Xu [20] proposed a simultaneous diagonalization-based SOCP relaxation for convex quadratic program with linear complementarity constraints. They also designed a new SOCP relaxation for the generalized trust-region problem and provided a sufficient condition under which the proposed SOCP relaxation is exact [21].

The main contributions of this paper are twofold.

1. We first decompose the matrix \( Q_1 \) according to the signs of its eigenvalues such that \( Q_1 = A - B \), where \( A \) and \( B \) are both positive semidefinite. Then, we propose a new SOCP relaxation via simultaneous diagonalization of the two positive semidefinite matrices \( Q_0 \) and \( B \). We further show that the new SOCP relaxation is as tight as the SDP relaxation when \( Q_0 \) is positive definite or \( Q_1 \) is negative semidefinite.

2. We derive a spatial branch-and-bound algorithm based on the new SOCP relaxation, and the following extensive numerical results show that the proposed algorithm outperforms the branch-and-bound algorithm as derived from the SDP relaxation. It implies that the new SOCP relaxation balances the bound tightness and computing efficiency better than that of the SDP relaxation.

The rest of the paper is organized as follows. In Section 2, a new SOCP relaxation for (1) is introduced. We also show that the new SOCP relaxation is as tight as SDP relaxation in certain cases. In Section 3, we propose a spatial branch-and-bound algorithm based on the new SOCP relaxation. Some numerical experiments are conducted to illustrate the effectiveness of the proposed algorithm in Section 4. Finally, some concluding remarks are provided in Section 5.

**Notations.** For two \( n \times n \) real matrices \( A = (A_{ij}) \) and \( B = (B_{ij}) \), \( A^{-1} \) represents the inverse of \( A \) and \( A \cdot B = \text{trace}(A^T B) = \sum_{i,j=1}^n A_{ij} B_{ij} \). For a real symmetric matrix \( X, X \succ 0 \) and \( X \succeq 0 \) mean that \( X \) is positive semidefinite and positive definite, respectively. \( I_n \) denotes the \( n \times n \) identity matrix, and \( e \) is a unit vector with all elements being 1. Given a vector \( a \in \mathbb{R}^n \), \( \text{diag}(a) \) corresponds to an \( n \times n \) diagonal matrix with its diagonal elements equal to \( a \).

### 2. A Simultaneous Diagonalization-Based SOCP Relaxation

In this section, we derive a new SOCP relaxation exploiting the techniques of difference of convex (DC) decomposition, and two positive semidefinite matrices are simultaneously diagonalizable. First of all, we present two lemmas concerning simultaneous matrix diagonalization that appear to play a central role in this paper.

**Lemma 1** (see [22, 23]). If \( M \in \mathbb{R}^{b \times b} \) and \( N \in \mathbb{R}^{b \times b} \) are two symmetric matrices and \( M \succ 0 \), then there exists a nonsingular matrix \( F \in \mathbb{R}^{b \times b} \) such that \( F^T MF \) and \( F^T NF \) are both diagonal matrices.

**Lemma 2** (see [22, 23]). If \( M \in \mathbb{R}^{b \times b} \) and \( N \in \mathbb{R}^{b \times b} \) are two positive semidefinite matrices, then there exists a nonsingular matrix \( F \in \mathbb{R}^{b \times b} \) such that \( F^T MF \) and \( F^T NF \) are both diagonal matrices.

Let \( r \) be the number of negative eigenvalues of \( Q_1 \), and without loss of generality, the first \( r \) eigenvalues of \( Q_1 \) are supposed to be negative. Thus, \( Q_1 = A - B \) where \( A = \sum_{i=1}^n \sigma_i t_i t_i^T \) and \( B = \sum_{i=1}^n \sigma_i \eta_i \eta_i^T \) in which \( \sigma_i \geq 0, \sigma_1, \ldots, \sigma_r, \sigma_{r+1}, \ldots, \sigma_n \) are eigenvalues and \( \eta_i \), \( i = 1, \ldots, n \) are corresponding eigenvectors of \( Q_1 \).

Since \( Q_0 \succeq 0, B \succeq 0 \) and \( \text{rank}(B) = r \), the proof of Lemma 2 [20] provides a method to find a nonsingular matrix \( F \in \mathbb{R}^{b \times b} \) such that \( F^T Q_0 F = \text{diag}(\mu_1, \ldots, \mu_n) \) and \( F^T B F = \text{diag}(\lambda_1, \ldots, \lambda_j, 0, \ldots, 0) \) with \( \lambda_i > 0, i = 1, \ldots, r \) and \( \mu_j \geq 0, j = 1, \ldots, n \).

Let \( x = Fy \), \( p_0 = F^T q_0 \), and \( p_i = F^T q_i \); then, (1) can be reformulated as

\[
\begin{align*}
\min & \quad \sum_{j=1}^n H_j y_j^2 + p_0^T y, \\
\text{s.t.} & \quad y^T F^T A F y - \sum_{j=1}^n \lambda_j y_j^2 + p_1^T y \leq c, \\
& \quad D F y \leq d.
\end{align*}
\] (2)

Equation (2) still cannot be solved in polynomial time as the quadratic constraint remains nonconvex. Then, we derive a new SOCP relaxation for (1) by introducing auxiliary variables \( t_i \) for \( i = 1, \ldots, r \).
\[
\begin{aligned}
\min & \quad \sum_{j=1}^{r} \mu_j t_j + \sum_{j=r+1}^{n} \mu_j y_j^2 + p_0^T y, \\
\text{s.t.} & \quad y^T F^T A F y - \sum_{j=1}^{r} \lambda_j t_j + p_1^T y \leq c, \\
& \quad D F y \leq d, \\
& \quad y_j^2 \leq t_j, \quad j = 1, \ldots, r.
\end{aligned}
\]

(3)

In general, the new SOCP relaxation provides a weaker lower bound than the SDP relaxation. Compared with the fact that the SDP relaxation lifts the original \(n\)-dimensional variable vector to an \((n + 1) \times (n + 1)\) variable matrix, the new SOCP relaxation only lifts the original \(n\)-dimensional variable vector to an \((n + r)\)-dimensional variable vector, which implies that (3) can be solved more quickly than (4). Therefore, the SOCP relaxation presents greater potential in some real-life applications.

Next, we will show that (3) is as tight as the SDP relaxation under certain circumstances. The SDP relaxation of (1) is [14]:

\[
\begin{aligned}
\min & \quad Q_0 \cdot X + q_0^T x, \\
\text{s.t.} & \quad Q_1 \cdot X + q_1^T x \leq c, \\
& \quad D x \leq d, \\
& \quad X = x x^T.
\end{aligned}
\]

(4)

Suppose that the nonsingular matrix \(F\) from (3) also diagonalizes \(A\). In fact, it implies that \(Q_0\) and \(Q_1\) are simultaneously diagonalizable. In such a case, we have \(F^T Q_0 F = \text{diag}(\mu_1, \ldots, \mu_n)\) and \(F^T Q_1 F = \text{diag}(\lambda_1, \ldots, -\lambda_1, \ldots, \lambda_n, \ldots, -\lambda_n)\) with \(\mu_i > 0\) for \(i = 1, \ldots, r\), \(\lambda_i \geq 0\) for \(i = r + 1, \ldots, n\), and \(\mu_i \geq 0\) for \(i = 1, \ldots, n\) according to Lemmas 1 and 2. Consequently, (3) can be reformulated as

\[
\begin{aligned}
\min & \quad \sum_{j=1}^{r} \mu_j t_j + \sum_{j=r+1}^{n} \mu_j y_j^2 + p_0^T y, \\
\text{s.t.} & \quad \sum_{j=r+1}^{n} \lambda_j y_j^2 - \sum_{j=1}^{r} \lambda_j t_j + p_1^T y \leq c, \\
& \quad D F y \leq d, \\
& \quad y_j^2 \leq t_j, \quad j = 1, \ldots, r.
\end{aligned}
\]

(5)

Theorem 1. When \(Q_0\) and \(Q_1\) are simultaneously diagonalizable, (5) is as tight as (4).

Proof. On one hand, if \((y, t)\) is a feasible solution of (5), then let \(x = F y\) and \(X = F(y y^T + \text{diag}(t_1, \ldots, t_r, y_1^2, \ldots, y_r^2, 0, \ldots, 0))F^T\). It is easy to see that \(X \succeq x x^T\) and

(1) \(Q_1 \cdot X + q_1^T x = Q_1 \cdot (F(y y^T + \text{diag}(t_1, \ldots, t_r, y_1^2, \ldots, y_r^2, 0, \ldots, 0))F^T + q_1^T x, = F^T Q_1 F \cdot (y y^T + \text{diag}(t_1, \ldots, t_r, y_1^2, \ldots, y_r^2, 0, \ldots, 0)) + q_1^T x, = \text{diag}(\lambda_1, \ldots, -\lambda_1, \lambda_{r+1}, \ldots, \lambda_n) \cdot (y y^T + \text{diag}(t_1, \ldots, t_r, y_1^2, \ldots, y_r^2, 0, \ldots, 0)) + q_1^T x, = \sum_{j=r+1}^{n} \lambda_j y_j^2 - \sum_{j=1}^{r} \lambda_j t_j + p_1^T y \leq c, (6)

(2) \(D x = D F y \leq d, \\
(3) \quad Q_0 \cdot X + q_0^T x = \text{diag}(\mu_1, \ldots, \mu_n) \cdot (y y^T + \text{diag}(t_1, \ldots, t_r, y_1^2, \ldots, y_r^2, 0, \ldots, 0)) + q_0^T x, = \sum_{j=1}^{r} \mu_j t_j + \sum_{j=r+1}^{n} \mu_j y_j^2 + p_0^T y.
\]

(4)

It implies that \((x, X)\) is a feasible solution of (4), and the optimal objective function value of (4) is no more than the one of (5).

On the other hand, if \((x, X)\) is a feasible solution of (4), then we define \(y = F^{-1} x, G = F^{-1} X (F^T)^{-1}\) and \(t_i = G_{ii}\) for \(i = 1, \ldots, r\). \(G \succeq y y^T\) implies \(t_i \geq y_i^2\) for \(i = 1, \ldots, r\). Also,

(5) \(D F y = D x \leq d, \\
(6) \quad \sum_{j=1}^{r} \mu_j t_j + \sum_{j=r+1}^{n} \mu_j y_j^2 + p_0^T y = \text{diag}(\mu_1, \ldots, \mu_n) \cdot \text{diag}(t_1, \ldots, t_r, y_1^2, \ldots, y_r^2, \ldots, y_n^2) + p_0^T y, \leq \text{diag}(\lambda_1, \ldots, -\lambda_1, \lambda_{r+1}, \ldots, \lambda_n) \cdot G + q_0^T x = (F^T)^{-1} \text{diag}(\lambda_1, \ldots, -\lambda_1, \lambda_{r+1}, \ldots, \lambda_n) F^{-1} \cdot F G F^T + q_1^T x, = Q_1 \cdot X + q_1^T x \leq c, (7)

(5) \(D F y = D x \leq d, \\
(6) \quad \sum_{j=1}^{r} \mu_j t_j + \sum_{j=r+1}^{n} \mu_j y_j^2 + p_0^T y = \text{diag}(\mu_1, \ldots, \mu_n) \cdot \text{diag}(t_1, \ldots, t_r, y_1^2, \ldots, y_r^2, \ldots, y_n^2) + p_0^T y, \leq \text{diag}(\lambda_1, \ldots, -\lambda_1, \lambda_{r+1}, \ldots, \lambda_n) \cdot G + q_0^T x = Q_0 \cdot X + q_0^T x.
\]
It implies that \((y, t)\) is a feasible solution of (5) and the optimal objective function value of (5) is no more than that of (4).

Therefore, (5) is as tight as (4).

\[ \square \]

**Proposition 1.** When \(Q_0\) is positive definite or \(Q_1\) is negative semidefinite, (5) is as tight as (4).

**Proof.** If \(Q_0\) is positive definite or \(Q_1\) is negative semidefinite, then \(Q_0\) and \(Q_1\) are simultaneously diagonalizable according to Lemmas 1 and 2. Hence, (5) is as tight as (4) according to Theorem 1.

In what follows, we will use two simple examples to show that (5) is as tight as (4) under the conditions of Proposition 1, but (5) can be solved faster than (4).

**Example 1.** Consider the problem with \(n = 3\) and \(m = 6\):

\[
\begin{align*}
\min & \quad x^T \begin{bmatrix} 32 & 18 & 44 \\ 18 & 93 & 14 \\ 44 & 14 & 82 \end{bmatrix} x + \begin{bmatrix} 6 \\ 14 \\ -6 \end{bmatrix} x, \\
\text{s.t.} & \quad x^T \begin{bmatrix} -50 & -27 & -14 \\ -78 & -49 & 3 \\ -4 & 3 & -37 \end{bmatrix} x + \begin{bmatrix} 18^T \\ 8 \\ 3 \end{bmatrix} x \leq -49, \\
& \quad 0 \leq x_i \leq 1, \quad i = 1, 2, 3.
\end{align*}
\]

The optimal objective function value of (4) is 7.9022, and the CPU time is 0.0448 seconds. Since \(Q_0\) is positive definite, we can find \(F = \begin{bmatrix} 0.3509 & -0.0183 & -0.1260 \\ -0.0056 & -0.0063 & 0.1133 \\ -0.1954 & -0.0990 & 0.0418 \end{bmatrix}\), \(F^T Q_1 F = \begin{bmatrix} -6.1233 & -0.4227 & 0.1347 \end{bmatrix}\), and \(F^T Q_0 F = \begin{bmatrix} 1, 1, 1 \end{bmatrix}\). The optimal value of (5) is 7.9022, and the CPU time is 0.0196 seconds.

**Example 2.** Consider the problem with \(n = 3\) and \(m = 6\):

\[
\begin{align*}
\min & \quad x^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & -1 \\ 0 & 1 & 4 \end{bmatrix} x + \begin{bmatrix} -1 \end{bmatrix} x, \\
\text{s.t.} & \quad -x^T x \leq -1, \\
& \quad 0 \leq x_i \leq 1, \quad i = 1, 2, 3.
\end{align*}
\]

The optimal objective function value of (4) is -1, and the CPU time is 0.1364 seconds. Since \(Q_1\) is negative definite, we can find \(F = \begin{bmatrix} 1.0000 & 0 & 0 \\ 0 & -0.5257 & -0.8507 \\ 0 & -0.8507 & 0.5257 \end{bmatrix}\), \(F^T Q_1 F = \begin{bmatrix} -1, -1, -1 \end{bmatrix}\), and \(F^T Q_0 F = \begin{bmatrix} 0, 0.3820, 2.6180 \end{bmatrix}\). The optimal value of (5) is -1, and the CPU time is 0.0196 seconds.

### 3. A Spatial Branch-and-Bound Algorithm

In this section, we develop a spatial branch-and-bound algorithm based on (3) for (1). Kim and Kojima [24] pointed out that it is necessary to add some appropriate constraints to the auxiliary variables to improve the lower bound. In order to enhance the relaxation effect and design a branch-and-bound algorithm, we introduce reformulation-linearization technique (RLT) constraints to (3). (1) is bounded, and thus there must exist a lower bound \(l_j\) and an upper bound \(u_j\) for each \(y_j, j = 1, \ldots, r\). Therefore, we can add RLT constraints \(t_j \leq (l_j + u_j)y_j - l_ju_j\) for \(j = 1, \ldots, r\) into (3).

The new SOCP relaxation with RLT constraints is in the form

\[
\begin{align*}
\min & \quad \sum_{j=1}^{r} \mu_j t_j + \sum_{j=1}^{n} \mu_j y_j^2 + p_0^T y, \\
\text{s.t.} & \quad y^T A^T F y - \sum_{j=1}^{r} \lambda_j t_j + p_1^T y \leq c, \\
& \quad DF y \leq d, \\
& \quad y_j^2 \leq t_j, \\
& \quad t_j \leq (l_j + u_j)y_j - l_ju_j, \quad j = 1, \ldots, r.
\end{align*}
\]

The initial lower bound \(l^0_j\) and upper bound \(u^0_j\) of \(y_j, j = 1, \ldots, r\) are solved by the following 2r linear programming problems:

\[
\begin{align*}
l^0_j/u^0_j = \min/\max & \quad y_j, \\
\text{s.t.} & \quad DF y \leq d.
\end{align*}
\]

Taking Example 1 as an instance, the optimal value of (10) is 15.3976, which is better than the optimal value 7.9022 of (3).

**Lemma 3.** Let \((y, t)\) be an optimal solution of (10) over the initial box \([l^0, u^0]\). If \(y_j^* = t_j\) for \(j = 1, \ldots, r\), then \(x = F y\) is an optimal solution of (3).

**Proof.** If \(y_j^* = t_j\) for \(j = 1, \ldots, r\), then \(y\) is a feasible solution of (2). Consequently, \(x = F y\) is an optimal solution of (1).

According to Lemma 3, we can see that if \((y, t)\) is not an optimal solution, then there must exist some \(j \in \{1, \ldots, r\}\) satisfying \(t_j > y_j^*\). Thus, we can select the index \(j^*\) and split the initial box \([l^0, u^0]\) into two new boxes \([l^0, u^0]\) and \([l^0, u^0]\) with \(l^0 = l^0 + u^0, u^0 = (l^0_j + u^0_j)/2, l^0 = l^0, u^0 = u^0,\) and \(l^0_j = (l^0_j + u^0_j)/2\). Consequently, we generate two new
nodes over \([l^i, u^i]\) and \([\bar{l}^i, \bar{u}^i]\), respectively, in the branch-and-bound tree.

Before describing the spatial branch-and-bound algorithm, we give the following definition.

**Definition 2.** For a given \(\varepsilon > 0\) and a vector \(y \in \mathbb{R}^n\), let \(x = Fy\). If \(x^TQ_x + \frac{1}{2}x^2 + \lambda_j \leq \varepsilon\), then \(x\) is called an \(\varepsilon\)-feasible solution of (1). Let \(V_{\text{OCQP}}\) be the optimal objective function value of (1), if the \(\varepsilon\)-feasible solution \(x\) also satisfies \(x^TQ_x + \frac{1}{2}x^2 + \varepsilon \leq V_{\text{OCQP}}\), then \(x\) is called an \(\varepsilon\)-optimal solution of (1). Define the function

\[ f_* (x) = \begin{cases} x^TQ_x + \frac{1}{2}x^2, & \text{if } x \text{ is } \varepsilon \text{-feasible}, \\ +\infty, & \text{otherwise}. \end{cases} \tag{12} \]

The proposed algorithm is presented in Algorithm 1.

Although the branch-and-bound algorithm based on (5) is not detailed here, it can be easily obtained by changing (3) with (5) in the proposed algorithm.

Next, we will prove that Algorithm 1 converges after exploring finite nodes and returns an \(\varepsilon\)-optimal solution.

**Lemma 4.** Suppose that the node \([y^k, t^k, l^k, u^k]\) is chosen from \(D\) in Line 11 of Algorithm 1 such that \(l^k = \min \{l^j | l^j \in D\}\) and \(j^* = \arg\max_{y^k, t^k, l^k, u^k} \lambda_j \left( t^k_j - (y^k)_j \right)\). For any \(\varepsilon > 0\), there exists a \(\delta > 0\) such that if \((u^k_j - l^k_j) \leq \delta\), then Algorithm 1 terminates in Line 13.

**Proof.** Let \(x^k = Fy^k\); then,
\[(x^k)^T Q_1 x^k + q_1^T x^k = (y^k)^T F^T A F y^k - \sum_{j=1}^r \lambda_j \cdot (y_j^k)^2 + p_1^T y^k,
= (y^k)^T F^T A F y^k - \sum_{j=1}^r \lambda_j t_j^k + p_1^T y^k + \sum_{j=1}^r \lambda_j \left( t_j^k - (y_j^k)^2 \right) \leq c + \sum_{j=1}^r \lambda_j \left( t_j^k - (y_j^k)^2 \right) \leq c + \frac{r}{4} \sum_{j=1}^r \lambda_j \left( u_j^k - \frac{p_j^k}{4} \right)^2 \leq c + \frac{r}{4} \max_{j} \left( \lambda_j \right) \] (13)

Theorem 2. Algorithm 1 returns an \( \varepsilon \)-optimal solution after line 13.

4. Numerical Experiments

In this section, we compare the proposed algorithm with Algorithm 1 in [25] which is a SDP relaxation-based branch-and-bound algorithm (SDP_BB). These algorithms are implemented in MATLAB R2013b on a PC with Window 7, 2.50 GHz Inte Dual Core CPU processors and 8 GB RAM. The SDP relaxations are solved by Sedum 1.02 [26] and the SOCP relaxations are solved by Cplex solver 12.6.3. The error tolerance is set to be \( \varepsilon = 5e - 4 \).

The instances for the experiments are generated as follows. The entries of \( q_i \) are integers uniformly drawn at random from intervals \([-8, 18]\) for each \( j = 0, 1 \). The elements of matrices \( Q_1 \) are integers uniformly drawn at random from intervals \([-100, 10]\) (reference [17]). Then, \( Q_0 \) and \( Q_1 \) are replaced by their real symmetric parts. \( Q_0 \) is decomposed as \( Q_0 = V D V^T \) where \( D \) is the diagonal matrix of eigenvalues, and \( V \) is the orthogonal matrix whose columns are the corresponding eigenvectors. Let \( Q_j = V |D| V^T \). We generate a vector \( a \in \mathbb{R}^n \) with the entries uniformly drawn at random from intervals \([0,1]\). Then, we let \( c = \text{cell}(a^T q_0 + a^T Q_1 a), Q_0 = Q_0/\sqrt{a^T q_0 + a^T Q_0 a}, Q_1 = \sqrt{a^T q_1 + a^T Q_1 a}, D = \begin{bmatrix} 1 & -I \\ -I & 0 \end{bmatrix} \) and \( d = \frac{\varepsilon}{4} \). Five instances are generated for each given problem size in Table 1. The explored nodes (exp nodes) and CPU time in seconds (CPU time) are displayed for each algorithm in Table 1. The symbol “—” means the instance cannot be solved within the given maximum computing time of 10000 seconds.

Table 1 shows that

(1) It is easy to observe that the average time cost for each explored node, i.e., CPU time/exp nodes of the proposed algorithm SOCP_BB, is much lower than that of the algorithm SDP_BB for each instance. Also, this evidence is deepened when \( n \) becomes larger. The observation further illustrates that the computing efficiency of solving the SOCP relaxation is higher than that of solving the SDP relaxation.

(2) For most of the instances, both the number of explored nodes and the CPU time of the proposed algorithm SOCP_BB are smaller than those of the algorithm SDP_BB. It implies that the proposed SOCP relaxation offers a good trade-off between the computing efficiency and bound quality for (1).
5. Conclusion

By exploiting the simultaneous diagonalization technique, we derive a new SOCP relaxation for (1). Then, we prove that the SOCP relaxation is as tight as the well-known SDP relaxation in certain cases. Furthermore, we design a spatial branch-and-bound algorithm based on the new SOCP relaxation. Finally, numerical experiments are conducted to demonstrate the efficiency of the proposed method by comparing with the SDP relaxation-based branch-and-bound algorithm, and the promising results illustrates that the simultaneous diagonalization-based SOCP relaxation indeed well balance the bound quality and computing time.

Data Availability

The author declares that all data and material in the paper are available and veritable.

Conflicts of Interest

The author declares no conflicts of interest.

Acknowledgments

This research has been supported by the National Natural Science Foundation of China (Grant no. 11701512).

References


