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:

$$\begin{aligned} \min \quad & f(x) = x^T Q_0 x + q_0^T x, \\ \text{s.t.} \quad & x^T Q_1 x + q_1^T x \leq c, \\ & Dx \leq d, \end{aligned} \quad (1)$$

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 multi-sensor 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 $N \in \mathbb{R}^{n \times n}$ are called simultaneously diagonalizable (SD) if there exists a nonsingular matrix $F \in \mathbb{R}^{n \times n}$ such that $F^T M F$ and $F^T N F$ 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 by 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 , $X \succeq 0$ and $X \succ 0$ mean that X is positive semidefinite and positive definite, respectively. I_n denotes the n -dimensional 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 $M \in \mathbb{R}^{h \times h}$ and $N \in \mathbb{R}^{h \times h}$ are two symmetric matrices and $M \succ 0$, then there exists a nonsingular matrix $F \in \mathbb{R}^{h \times h}$ such that $F^T M F$ and $F^T N F$ are both diagonal matrices.*

Lemma 2 (see [22, 23]). *If $M \in \mathbb{R}^{h \times h}$ and $N \in \mathbb{R}^{h \times h}$ are two positive semidefinite matrices, then there exists a nonsingular matrix $F \in \mathbb{R}^{h \times h}$ such that $F^T M F$ and $F^T N F$ 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 $A = \sum_{i=r+1}^n \sigma_i \eta_i \eta_i^T$ and $B = \sum_{i=1}^r \sigma_i \eta_i \eta_i^T$ in which $\sigma_i \geq 0$, $-\sigma_1, \dots, -\sigma_r, \sigma_{r+1}, \dots, \sigma_n$ are eigenvalues and η_i , $i = 1, \dots, n$, 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 $F \in \mathbb{R}^{n \times n}$ such that $F^T Q_0 F = \text{diag}(\mu_1, \dots, \mu_n)$ and $F^T B F = \text{diag}(\lambda_1, \dots, \lambda_r, 0, \dots, 0)$ with $\lambda_i > 0$, $i = 1, \dots, r$ and $\mu_j \geq 0$, $j = 1, \dots, n$.

Let $x = Fy$, $p_0 = F^T q_0$, and $p_1 = F^T q_1$; then, (1) can be reformulated as

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

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 $i = 1, \dots, r$:

$$\begin{aligned}
\min \quad & \sum_{j=1}^r \mu_j t_j + \sum_{j=r+1}^n \mu_j y_j^2 + p_0^T y, \\
\text{s.t.} \quad & y^T F^T A F y - \sum_{j=1}^r \lambda_j t_j + p_1^T y \leq c, \\
& DF y \leq d, \\
& y_j^2 \leq t_j, \quad j = 1, \dots, r.
\end{aligned} \quad (3)$$

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]:

$$\begin{aligned}
\min \quad & Q_0 \cdot X + q_0^T x, \\
\text{s.t.} \quad & Q_1 \cdot X + q_1^T x \leq c, \\
& Dx \leq d, \\
& X \succeq xx^T.
\end{aligned} \quad (4)$$

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 $F^T Q_0 F = \text{diag}(\mu_1, \dots, \mu_n)$ and $F^T Q_1 F = \text{diag}(-\lambda_1, \dots, -\lambda_r, \lambda_{r+1}, \dots, \lambda_n)$ with $\lambda_i > 0$ for $i = 1, \dots, r$, $\lambda_i \geq 0$ for $i = r+1, \dots, n$, and $\mu_i \geq 0$ for $i = 1, \dots, n$ according to Lemmas 1 and 2. Consequently, (3) can be reformulated as

$$\begin{aligned}
\min \quad & \sum_{j=1}^r \mu_j t_j + \sum_{j=r+1}^n \mu_j y_j^2 + p_0^T y, \\
\text{s.t.} \quad & \sum_{j=r+1}^n \lambda_j y_j^2 - \sum_{j=1}^r \lambda_j t_j + p_1^T y \leq c, \\
& DF y \leq d, \\
& y_j^2 \leq t_j, \quad j = 1, \dots, r.
\end{aligned} \quad (5)$$

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 $x = Fy$ and $X = F(yy^T + \text{diag}(t_1 - y_1^2, \dots, t_r - y_r^2, 0, \dots, 0))F^T$. It is easy to see that $X \succeq xx^T$ and

$$\begin{aligned}
(1) \quad & Q_1 \cdot X + q_1^T x \\
&= Q_1 \cdot (F(yy^T + \text{diag}(t_1 - y_1^2, \dots, t_r - y_r^2, 0, \dots, 0))F^T) + q_1^T x, \\
&= F^T Q_1 F \cdot (yy^T + \text{diag}(t_1 - y_1^2, \dots, t_r - y_r^2, 0, \dots, 0)) + q_1^T x, \\
&= \text{diag}(-\lambda_1, \dots, -\lambda_r, \lambda_{r+1}, \dots, \lambda_n) \\
&\quad \cdot (yy^T + \text{diag}(t_1 - y_1^2, \dots, t_r - y_r^2, 0, \dots, 0)) + q_1^T x, \\
&= \sum_{j=r+1}^n \lambda_j y_j^2 - \sum_{j=1}^r \lambda_j t_j + p_1^T y \leq c, \\
(2) \quad & Dx = DF y \leq d, \\
(3) \quad & Q_0 \cdot X + q_0^T x \\
&= \text{diag}(\mu_1, \dots, \mu_n) \cdot (yy^T + \text{diag}(t_1 - y_1^2, \dots, t_r - y_r^2, 0, \dots, 0)) + q_0^T x, \\
&= \sum_{j=1}^r \mu_j t_j + \sum_{j=r+1}^n \mu_j y_j^2 + p_0^T y.
\end{aligned} \quad (6)$$

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 $i = 1, \dots, r$. $G \succeq yy^T$ implies $t_i \geq y_i^2$ for $i = 1, \dots, r$. Also,

$$\begin{aligned}
(4) \quad & \sum_{j=r+1}^n \lambda_j y_j^2 - \sum_{j=1}^r \lambda_j t_j + p_1^T y \\
&= \text{diag}(-\lambda_1, \dots, -\lambda_r, \lambda_{r+1}, \dots, \lambda_n) \cdot \text{diag}(t_1, \dots, t_r, y_{r+1}^2, \dots, y_n^2) + p_1^T y \\
&\leq \text{diag}(-\lambda_1, \dots, -\lambda_r, \lambda_{r+1}, \dots, \lambda_n) \cdot G + q_1^T x \\
&= (F^T)^{-1} \text{diag}(-\lambda_1, \dots, -\lambda_r, \lambda_{r+1}, \dots, \lambda_n) F^{-1} \\
&\quad \cdot F G F^T + q_1^T x, \\
&= Q_1 \cdot X + q_1^T x \leq c, \\
(5) \quad & DF y = Dx \leq d, \\
(6) \quad & \sum_{j=1}^r \mu_j t_j + \sum_{j=r+1}^n \mu_j y_j^2 + p_0^T y \\
&= \text{diag}(\mu_1, \dots, \mu_n) \cdot \text{diag}(t_1, \dots, t_r, y_{r+1}^2, \dots, y_n^2) + p_0^T y \\
&\leq \text{diag}(\mu_1, \dots, \mu_n) \cdot G + q_0^T x \\
&= Q_0 \cdot X + q_0^T x.
\end{aligned} \quad (7)$$

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. $\square$

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 $m = 6$:

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

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 $F = \begin{bmatrix} 0.3509 & -0.0183 & -0.1260 \\ -0.0056 & -0.0063 & 0.1133 \\ -0.1954 & -0.0990 & 0.0418 \end{bmatrix}$, $F^T Q_1 F = \text{diag}(-6.1233, -0.4227, 0.1347)$, and $F^T Q_0 F = \text{diag}(1, 1, 1)$. 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 $m = 6$:

$$\begin{aligned} \min \quad & x^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} x + \begin{bmatrix} -1 \\ 1 \\ 4 \end{bmatrix}^T x, \\ \text{s.t.} \quad & -x^T x \leq -1, \\ & 0 \leq x_i \leq 1, \quad i = 1, 2, 3. \end{aligned} \quad (9)$$

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 $F = \begin{bmatrix} 1.0000 & 0 & 0 \\ 0 & -0.5257 & -0.8507 \\ 0 & -0.8507 & 0.5257 \end{bmatrix}$, $F^T Q_1 F = \text{diag}(-1, -1, -1)$, and $F^T Q_0 F = \text{diag}(0, 0.3820,$

2.6180). 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 , $j = 1, \dots, r$. Therefore, we can add r RLT constraints $t_j \leq (l_j + u_j)y_j - l_j u_j$ for $j = 1, \dots, r$ into (3).

The new SOCP relaxation with RLT constraints is in the form

$$\begin{aligned} \min \quad & \sum_{j=1}^r \mu_j t_j + \sum_{j=r+1}^n \mu_j y_j^2 + p_0^T y, \\ \text{s.t.} \quad & y^T F^T A F y - \sum_{j=1}^r \lambda_j t_j + p_1^T y \leq c, \\ & D F y \leq d, \\ & y_j^2 \leq t_j, \\ & t_j \leq (l_j + u_j)y_j - l_j u_j, \quad j = 1, \dots, r. \end{aligned} \quad (10)$$

The initial lower bound l_j^0 and upper bound u_j^0 of y_j , $j = 1, \dots, r$ are solved by the following $2r$ linear programming problems:

$$\begin{aligned} l_j^0/u_j^0 = \min/\max \quad & y_j, \\ \text{s.t.} \quad & D F y \leq d. \end{aligned} \quad (11)$$

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 $y_j^2 = t_j$ for $j = 1, \dots, r$, then $x = Fy$ is an optimal solution of (3).

Proof. If $y_j^2 = t_j$ for $j = 1, \dots, r$, then y is a feasible solution of (2). Consequently, $x = Fy$ is an optimal solution of (1). $\square$

According to Lemma 3, we can see that if (y, t) is not an optimal solution, then there must exist some $j \in \{1, \dots, r\}$ satisfying $t_j > y_j^2$. Thus, we can select the index j^* and split the initial box $[l^0, u^0]$ into two new boxes $[l^a, u^a]$ and $[l^b, u^b]$ with $l^a = l^0$, $u^a = u^0$, $u_{j^*}^a = (l_{j^*}^0 + u_{j^*}^0)/2$, $l^b = l^0$, $u^b = u^0$, and $l_{j^*}^b = (l_{j^*}^0 + u_{j^*}^0)/2$. Consequently, we generate two new

Require: An instance of (1) and a given error tolerance $\varepsilon > 0$. Set iteration step $k = 1$, the upper bound $V^* = +\infty$ and $x^* = \emptyset$.

- (1) Solve (11) for l^0 and u^0 .
- (2) **If** (11) is infeasible, **then**
- (3) (1) is infeasible and terminate.
- (4) **end if**
- (5) Solve (10) over $[l^0, u^0]$ for its optimal objective function value lb^0 and optimal solution (y^0, t^0) . Let $x^* = Uy^0$ and $V^* = f_\varepsilon(y^0)$.
- (6) Construct a set D and insert $\{y^0, t^0, lb^0, l^0, u^0\}$ into it.
- (7) **loop**
- (8) **if** $D = \emptyset$ **then**
- (9) return (x^*, V^*) and terminate.
- (10) **end if**
- (11) Choose a node from D , denoted as $\{y^k, t^k, lb^k, l^k, u^k\}$ such that $lb^k = \min\{lb^i \mid lb^i \in D\}$ and remove it from D .
- (12) **if** $V^* - lb^k \leq \varepsilon$, **then**
- (13) return (x^*, V^*) and terminate.
- (14) **end if**
- (15) Set $k \leftarrow k + 1$.
- (16) Choose $j^* = \arg \max_{j \in \{1, \dots, r\}} |\lambda_j| (t_j^k - (y_j^k)^2)$.
- (17) Construct the box $[l^a, u^a]$ by setting $l^a = l^k$, $u^a = u^k$, $u_{j^*}^a = (l_{j^*}^k + u_{j^*}^k)/2$ and construct the box $[l^b, u^b]$ by setting $l^b = l^k$, $u^b = u^k$, $l_{j^*}^b = (l_{j^*}^k + u_{j^*}^k)/2$.
- (18) **if** (10) over $[l^a, u^a]$ is feasible, **then**
- (19) Solve (10) over $[l^a, u^a]$ for its optimal objective function value lb^a and optimal solution (y^a, t^a) . Denote $ub^a = f_\varepsilon(Fy^a)$.
- (20) **if** $ub^a < V^*$, **then**
- (21) $V^* = ub^a$ and $x^* = Fy^a$.
- (22) **end if**
- (23) **if** $V^* - lb^a > \varepsilon$, **then**
- (24) insert $\{y^a, t^a, lb^a, l^a, u^a\}$ into D .
- (25) **end if**
- (26) **end if**
- (27) **if** (10) over $[l^b, u^b]$ is feasible, **then**
- (28) Solve (10) over $[l^b, u^b]$ for its optimal objective function value lb^b and optimal solution (y^b, t^b) . Denote $ub^b = f_\varepsilon(y^b)$.
- (29) **if** $ub^b < V^*$, **then**
- (30) $V^* = ub^b$ and $x^* = Fy^b$.
- (31) **end if**
- (32) **if** $V^* - lb^b > \varepsilon$, **then**
- (33) insert $\{y^b, t^b, lb^b, l^b, u^b\}$ into D .
- (34) **end if**
- (35) **end if**
- (36) **end loop**

ALGORITHM 1: A spatial branch-and-bound algorithm (SOCB_BB) for solving (1).

nodes over $[l^a, u^a]$ and $[l^b, u^b]$, 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 $x^T Q_1 x + q_1^T x \leq c + \varepsilon$, then x is called an ε -feasible solution of (1). Let V_{QCQP} be the optimal objective function value of (1), if the ε -feasible solution x also satisfies $x^T Q_0 x + q_0^T x \leq V_{\text{QCQP}} + \varepsilon$; then, x is called an ε -optimal solution of (1). Define the function

$$f_\varepsilon(x) = \begin{cases} x^T Q_0 x + q_0^T x, & \text{if } x \text{ is } \varepsilon\text{-feasible,} \\ +\infty, & \text{otherwise.} \end{cases} \quad (12)$$

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 ε -optimal solution.

Lemma 4. Suppose that the node $\{y^k, t^k, lb^k, l^k, u^k\}$ is chosen from D in Line 11 of Algorithm 1 such that $lb^k = \min\{lb^i \mid lb^i \in D\}$ and $j^* = \arg \max_{j \in \{1, \dots, r\}} \lambda_j (t_j^k - (y_j^k)^2)$. For any $\varepsilon > 0$, there exists a $\delta > 0$ such that if $(u_{j^*}^k - l_{j^*}^k) \leq \delta$, then Algorithm 1 terminates in Line 13.

Proof. Let $x^k = Fy^k$; then,

$$\begin{aligned}
(x^k)^T Q_1 x^k + q_1^T x^k &= (y^k)^T F^T A F y^k - \sum_{j=1}^r \lambda_j \\
&\quad \cdot (y_j^k)^2 + p_1^T y^k, \\
&= (y^k)^T F^T A F y^k - \sum_{j=1}^r \lambda_j t_j^k \\
&\quad + p_1^T y^k + \sum_{j=1}^r \lambda_j (t_j^k - (y_j^k)^2) \\
&\leq c + \sum_{j=1}^r \lambda_j (t_j^k - (y_j^k)^2) \leq c \\
&\quad + \sum_{j=1}^r \frac{\lambda_j (u_j^k - l_j^k)^2}{4} \leq c \\
&\quad + \frac{r\delta^2}{4} \max\{\lambda_j \mid j = 1, \dots, r\}.
\end{aligned} \tag{13}$$

For any $\varepsilon > 0$, set $\delta = 2\sqrt{\varepsilon}/\sqrt{r \max\{\lambda_j \mid j = 1, \dots, r\}}$, and then x^k is an ε -feasible solution. Besides,

$$\begin{aligned}
(x^k)^T Q_0 x^k + q_0^T x^k - V_{\text{QCQP}} &= \sum_{j=1}^n \mu_j (y_j^k)^2 + p_0^T y^k - V_{\text{QCQP}} \\
&\leq \sum_{j=1}^n \mu_j (y_j^k)^2 + p_0^T y^k - lb^k = \sum_{j=1}^r \mu_j ((y_j^k)^2 - t_j^k) \leq 0.
\end{aligned} \tag{14}$$

Therefore, x^k is an ε -optimal solution from Definition 2 and Algorithm 1 terminates in Line 13. $\square$

The proof of Lemma 4 implies that if $\sum_{j=1}^r \lambda_j (t_j^k - (y_j^k)^2) \leq \varepsilon$, then $x^k = F y^k$ is an ε -feasible solution. Thus, we aim to reduce $\sum_{j=1}^r \lambda_j (t_j^k - (y_j^k)^2)$ in the branch-and-bound algorithm. Hence, we choose $\arg \max_{j \in \{1, \dots, r\}} \lambda_j (t_j^k - (y_j^k)^2)$ as the variable selection strategy.

Theorem 2. Algorithm 1 returns an ε -optimal solution after exploring at most nodes.

$$N_\varepsilon = \prod_{j=1}^r \frac{(u_j^0 - l_j^0) \sqrt{r \max\{\lambda_j \mid j = 1, \dots, r\}}}{\sqrt{\varepsilon}}. \tag{15}$$

Proof. Algorithm 1 shows that if the algorithm does not terminate in line 13, then a chosen box is split in half and two new boxes are generated. After exploring k nodes, the initial box $[l^0, u^0]$ would be split into $k + 1$ boxes. If Algorithm 1 does not obtain an ε -optimal solution in line 13, it is easy to check that for each box $[l^k, u^k]$ among those $k + 1$ boxes, $u_j^k - l_j^k \geq \min\{(\delta/2), u_j^0 - l_j^0\}$ with $\delta = 2\sqrt{\varepsilon}/\sqrt{r \max\{\lambda_j \mid j = 1, \dots, r\}}$ for $j = 1, \dots, r$. Otherwise, if there exists a j^* such that

$u_{j^*}^k - l_{j^*}^k \leq \delta$, and the j^* -th edge has been selected as a branching direction, then following from Lemma 4, we conclude that $x^k = F y^k$ is an ε -optimal solution. It contradicts with the assumption that Algorithm 1 does not obtain an ε -optimal solution in line 13 at node k . Hence, the volume of each box is no smaller than $\prod_{j=1}^r \min\{(\delta/2), u_j^0 - l_j^0\}$. Due to the fact that the total volume of all the $k + 1$ boxes is no more than the one of the initial box $[l^0, u^0]$, it is easy to check that, after exploring $N_\varepsilon = \prod_{j=1}^r (2(u_j^0 - l_j^0)/\delta) = \prod_{j=1}^r ((u_j^0 - l_j^0)/\sqrt{r \max\{\lambda_j \mid j = 1, \dots, r\}})/\sqrt{\varepsilon}$ nodes, Algorithm 1 returns an ε -optimal solution. $\square$

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 Inter Dual Core CPU processors and 8 GB RAM. The SDP relaxations are solved by SeDuMi 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_j 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 $Q_0 = V |D| V^T$. We generate a vector $a \in \mathbb{R}^n$ with the entries of a uniformly drawn at random from intervals $[0, 1]$. Then, we let $c = \text{ceil}(a^T q_1 + a^T Q_1 a)$, $Q_0 = Q_0 / |a^T q_0 + a^T Q_0 a|$, $q_0 = q_0 / \sqrt{|a^T q_0 + a^T Q_0 a|}$, $D = \begin{bmatrix} I \\ -I \end{bmatrix}$, and $d = \begin{bmatrix} e \\ 0 \end{bmatrix}$. 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).

TABLE 1: Comparisons between SOCP_BB and SDP_BB.

n	SOCP_BB		SDP_BB	
	Exp nodes	CPU time	Exp nodes	CPU time
10	109	4.4539	208	50.0983
10	1	0.0240	1	0.0698
10	128	5.1209	230	41.2431
10	92	3.4678	75	11.5937
10	1	0.0228	2	0.1881
20	125	8.3562	176	69.9982
20	1664	115.1088	3114	1067.7725
20	566	38.1457	991	330.2524
20	30	1.9629	54	13.2143
20	55	3.5615	134	31.8341
30	1	0.08702	2	0.5089
30	114	9.6467	272	179.0759
30	4389	419.0829	3910	1431.4212
30	70	5.9402	92	34.3371
30	2983	260.3418	7523	4909.2678
40	21	2.8233	52	29.3877
40	54	6.5244	214	121.6062
40	74	8.3827	185	110.2765
40	136	15.4152	252	156.2269
40	31	3.3112	200	119.5072
50	152	24.5512	483	456.7316
50	139	21.9021	445	411.3182
50	1	0.3228	1	0.4154
50	150	23.5961	210	187.4114
50	19	3.9621	107	104.7606
60	13	2.3786	59	82.2647
60	5	1.1534	35	54.2444
60	103	18.7057	305	426.8943
60	18	4.1163	63	93.2252
60	19	4.2133	79	113.4004
70	5434	1332.6043	—	—
70	32	7.8642	203	432.7264
70	3677	891.9838	—	—
70	60	14.3761	111	250.3169
70	258	57.3409	724	1556.7962
80	11	4.1058	27	78.7600
80	69	17.6592	306	948.1594
80	136	45.5027	607	1896.2123
80	12634	4315.1516	—	—
80	81	24.2367	626	1829.0213
90	1979	820.2060	758	3708.4941
90	13148	4698.9012	—	—
90	3563	1322.3103	—	—
90	13469	4953.5129	—	—
90	151	54.3170	1814	8381.2107
100	23	12.6613	161	964.7115
100	337	153.4485	—	—
100	9	6.2885	47	264.1438
100	5454	2783.0369	—	—
100	102	48.8754	313	1859.7320

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

- [1] J. Zhou, D. Chen, Z. Wang, and W. Xing, "A conic approximation method for the 0-1 quadratic knapsack problem," *Journal of Industrial and Management Optimization*, vol. 9, no. 3, pp. 531–547, 2013.
- [2] B. G. Alex, D. S. Nicholas, S. Shahram, B. Mats, and O. Bjorn, "Convex optimization-based beamforming: from receive to transmit and network designs," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 62–75, 2010.
- [3] J. Zhou and Z. Deng, "A low-dimensional SDP relaxation based spatial branch and bound method for nonconvex quadratic programs," *Journal of Industrial and Management Optimization*, 2019.
- [4] G. Kochenberger, J.-K. Hao, F. Glover et al., "The unconstrained binary quadratic programming problem: a survey," *Journal of Combinatorial Optimization*, 2019.
- [5] C. Lu and X. Guo, "Convex reformulation for binary quadratic programming problems via average objective value maximization," *Optimization Letters*, vol. 9, no. 3, pp. 523–535, 2015.
- [6] J. Keuchel, C. Schnorr, C. Schellewald, and D. Cremers, "Binary partitioning, perceptual grouping, and restoration with semidefinite programming," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 11, pp. 1364–1379, 2003.
- [7] X. Mu and Y. Zhang, "A new rank-two semidefinite programming relaxation method for multiuser detection problem," *Wireless Personal Communications*, vol. 65, no. 1, pp. 223–233, 2012.
- [8] G. Gallo, P. L. Hammer, and B. Simeone, "Quadratic knapsack problems," *Mathematical Programming Studies*, vol. 12, pp. 132–149, 1980.
- [9] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM*, vol. 42, no. 6, pp. 1115–1145, 1995.
- [10] M. C. Ferris and J. S. Pang, "Engineering and economic applications of complementarity problems," *Siam Review*, vol. 39, no. 4, pp. 669–713, 1997.
- [11] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, "Applications of second-order cone programming," *Linear*

- Algebra and Its Applications*, vol. 284, no. 1–3, pp. 193–228, 1998.
- [12] Y. Gao, F. Li, and S. Jin, “A branch and bound reduced algorithm for quadratic programming problems with quadratic constraints,” *Mathematical Problems in Engineering*, vol. 2013, 2013.
 - [13] H. Peng, Z. Wu, P. Shao, and C. Deng, “Dichotomous binary differential evolution for knapsack problems,” *Mathematical Problems in Engineering*, vol. 2016, Article ID 5732489, 12 pages, 2016.
 - [14] X. Bao, N. V. Sahinidis, and M. Tawarmalani, “Semidefinite relaxations for quadratically constrained quadratic programming: a review and comparisons,” *Mathematical Programming*, vol. 129, no. 1, pp. 1–29, 2011.
 - [15] Y. Wang and R. Shi, “Tightness of semidefinite programming relaxation to robust transmit beamforming with SINR constraints,” *Mathematical Problems in Engineering*, vol. 2013, Article ID 508014, 10 pages, 2013.
 - [16] S. Burer, S. Kim, and M. Kojima, “Faster, but weaker, relaxations for quadratically constrained quadratic programs,” *Computational Optimization and Applications*, vol. 59, no. 1–2, pp. 27–45, 2014.
 - [17] X. J. Zheng, X. L. Sun, and D. Li, “Nonconvex quadratically constrained quadratic programming: best D.C. decompositions and their SDP representations,” *Journal of Global Optimization*, vol. 50, no. 4, pp. 695–712, 2011.
 - [18] A. Ben-Tal and D. Den Hertog, “Hidden conic quadratic representation of some nonconvex quadratic optimization problems,” *Mathematical Programming*, vol. 143, no. 1–2, pp. 1–29, 2014.
 - [19] R. Jiang and D. Li, “Simultaneous diagonalization of matrices and its applications in quadratically constrained quadratic programming,” *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1649–1668, 2016.
 - [20] J. Zhou and Z. Xu, “A simultaneous diagonalization based SOCP relaxation for convex quadratic programs with linear complementarity constraints,” *Optimization Letters*, vol. 13, no. 7, pp. 1615–1630, 2019.
 - [21] J. Zhou, C. Lu, Y. Tian, and X. Tang, “A SOCP relaxation based branch-and-bound method for generalized trust-region subproblem,” *Journal of Industrial and Management Optimization*, 2019.
 - [22] R. W. Newcomb, “On the simultaneous diagonalization of two semi-definite matrices,” *Quarterly of Applied Mathematics*, vol. 19, no. 2, pp. 144–146, 1961.
 - [23] J. Wang, J. Lu, and Y. Feng, “Congruence diagonalization of two hermite matrices simultaneously,” *International Journal of Algebra*, vol. 4, no. 23, pp. 1119–1125, 2010.
 - [24] S. Kim and M. Kojima, “Second order cone programming relaxation of nonconvex quadratic optimization problems,” *Optimization Methods and Software*, vol. 15, no. 3–4, pp. 201–224, 2001.
 - [25] C. Lu, Z. Deng, J. Zhou, and X. Guo, “A sensitive-eigenvector based global algorithm for quadratically constrained quadratic programming,” *Journal of Global Optimization*, vol. 73, no. 2, pp. 371–388, 2019.
 - [26] J. F. Sturm, “Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones,” *Optimization Methods and Software*, vol. 11, no. 1–4, pp. 625–653, 1999.