A SOCP RELAXATION BASED BRANCH-AND-BOUND METHOD FOR GENERALIZED TRUST-REGION SUBPROBLEM

JING ZHOU
Department of Applied Mathematics, College of Science
Zhejiang University of Technology, Hangzhou, Zhejiang, 310032, China

CHENG LU
School of Economics and Management
North China Electric Power University, Beijing, 102206, China

YE TIAN*
School of Business Administration
and Collaborative Innovation Center of Financial Security
Southwestern University of Finance and Economics, Chengdu, 611130, China

XIAOYING TANG
Department of Information Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong

(Communicated by Oleg Prokopyev)

Abstract. This paper proposes a second-order cone programming (SOCP) relaxation for the generalized trust-region problem by exploiting the property that any symmetric matrix and identity matrix can be simultaneously diagonalizable. We show that our proposed SOCP relaxation can provide a lower bound as tight as that of the standard semidefinite programming (SDP) relaxation. Moreover, we provide a sufficient condition under which the proposed SOCP relaxation is exact. Since the standard SDP relaxation suffers from a much heavier computing burden, the proposed SOCP relaxation has a much higher efficiency in solving process. Then we design a branch-and-bound algorithm based on this SOCP relaxation to obtain the global optimal solution for a general problem. Three types of numerical experiments are carried out to demonstrate the effectiveness and efficiency of our proposed SOCP relaxation.

1. Introduction. In this paper, we consider a generalized trust-region problem in the following form:

\[
\begin{align*}
\min & \quad x^T Q x + q^T x, \\
\text{s.t.} & \quad \|x - w_i\| \leq d_i, \quad i = 1, \cdots, m, \\
& \quad \|x - w_i\| \geq d_i, \quad i = m+1, \cdots, m+p, \\
& \quad Ax \leq b,
\end{align*}
\]

\((GTRS)\)
where $x \in \mathbb{R}^n$ is the decision variable, $Q \in \mathbb{R}^{n \times n}$ is a symmetric matrix, $q \in \mathbb{R}^n$, $w_i \in \mathbb{R}^n$, $d_i > 0$, $i = 1, \cdots, m + p$, $A \in \mathbb{R}^{s \times n}$, $b \in \mathbb{R}^s$, $m$, $p$ and $s$ are nonnegative integers. We assume the feasible region of (GTRS) is bounded with nonempty relative interior points.

Note that, many well-known quadratic optimization problems fall into the category of (GTRS). When $m = 1$ and $s = 0$, (GTRS) becomes a classical trust-region subproblem (TRS) which is an essential ingredient of the famous trust-region methods. It is worth pointing out that (TRS) is commonly used in the continuous non-convex optimization problems [21, 22]. Sturm et al. [24] proved that quadratic program with one quadratic constraint is equivalent to its SDP relaxation and its optimal solution can be retrieved from the optimal solution of SDP relaxation by a special matrix decomposition algorithm. Ben-Tal et al. [4] showed that, when the quadratic forms are simultaneously diagonalizable, (TRS) has a hidden convexity which leads to an exact SOCP relaxation. Ho-Nguyen et al. [12] presented an exact SOCP-based convex reformulation of (TRS) based on the observation that any optimal solution of (TRS) must be on the boundary of its feasible region. When $m = 2$ and $s = 0$, (GTRS) is a Celis-Dennis-Tapia (CDT) problem, which was first proposed by Celis, Dennis and Tapia [10]. Ai et al. [1] presented a sufficient and necessary condition to characterize the situation when (CDT) and its Lagrangian dual have no duality gap and proved that if the strong duality holds, then an optimal solution of (CDT) can be retrieved from an optimal solution of its SDP relaxation. Then Bomze et al. [7] proved some new sufficient and necessary conditions for both local and global optimality and gave a complete characterization in the degenerate case. Besides, they provided some verifiable conditions under which both the usual Lagrangian relaxation and the copositive relaxation are equivalent for an extended (CDT) problem [8]. Bienstock [6] showed how to adapt a construction of Barvinok’s method [2] so as to obtain a polynomial-time algorithm for the quadratic program with a fixed number of quadratic constraints.

When $m = 1$ and $s \geq 1$, Burer et al. [9] showed that when two parallel cuts are added to (TRS), the corresponding nonconvex problem has an exact representation as a semidefinite program with additional linear and second-order-cone constraints. Jeyakumar et al. [14] showed that the nonconvex quadratic program with one quadratic constraint and $s$ linear constraints has an exact SDP-relaxation under a certain dimension condition and Ho-Nguyen et al. [12] extended their results to some more general cases. When $m = 0$ and $p > 0$, a special case occurs in the reformulation of semi-supervised support vector machine model [25]. There, the corresponding optimization problem has a convex quadratic objective function, some linear constraints and several nonconvex quadratic constraints in the form $\|x_i\| \geq 1$. For the general cases of (GTRS), Bienstock et al. [5] provided an algorithm by enumerating candidates for the optimal solution on each face of the feasible domain. Beck et al. [3] suggested a branch-and-bound algorithm employing the polynomial solvability of (TRS) to compute all local minimizers of the problem.

1.1. Solving (GTRS) by exploiting the simultaneous diagonalization technique. In this paper, we aim to design a new approach for (GTRS) using simultaneous diagonalization technique.

**Definition 1.1.** For $B_i \in \mathbb{R}^{n \times n}$, $i = 1, \ldots, n$, if there exists a nonsingular matrix $F \in \mathbb{R}^{n \times n}$ such that $F^T B_i F$ is a diagonal matrix for all $i = 1, \ldots, n$, then we call these $B_i$s are simultaneously diagonalizable.
It is worth pointing out that many papers have shown the merit of employing the technique of simultaneous diagonalization when designing a SOCP relaxation for the nonconvex quadratically constrained quadratic programming (QCQP) problem in the past few years. For example, Ben-Tal et al. [4] proved that for the problem of minimizing a quadratic objective function subject to one or two quadratic constraints, it is possible to derive an equivalent SOCP relaxation when the matrices in the quadratic forms are simultaneously diagonalizable. Jiang et al. [13] offered a necessary and sufficient simultaneously diagonalizable condition for any finite collection of real symmetric matrices under the existence assumption of a semidefinite matrix pencil. They also applied the simultaneous diagonalization techniques to quadratic program with one or two quadratic constraints. Zhou et al. [27] proposed a simultaneous diagonalization based SOCP relaxation for convex quadratic program with linear complementarity constraints and proved that it is equivalent to the SDP relaxation when the objective matrix is positive definite.

Note that, the quadratic constraints in this paper can be reformulated as $x^T I_n x - 2w_i^T x + w_i^T w_i \leq (\geq) d_i^2$, hence $Q$ and the identity matrix $I_n$ can be simultaneously diagonalizable by eigenvalue decomposition of $Q$. Based on this observation, we can apply the state-of-the-art technique of simultaneous diagonalization into the SOCP relaxation. Since the proposed SOCP relaxation cannot guarantee a global optimal solution for a general problem, a branch-and-bound scheme is employed in our algorithm.

1.2. Contributions and the outline of the paper. The main contributions of this paper can be concluded as the following five aspects. First, we apply the simultaneous diagonalization technique to derive a novel SOCP relaxation. Second, we show that the proposed SOCP relaxation is as tight as the standard SDP relaxation [18]. However, the SOCP relaxation has a much smaller dimension than the standard SDP relaxation, thus, it has a much bigger advantage in computing. Hence it has the ability to solve some large-sized problems. Third, the SOCP relaxation can be further simplified by reducing the number of the second-order cone constraints for a special case. Fourth, we provide a sufficient condition under which the proposed SOCP relaxation is exact and an optimal solution of (GTRS) can be retrieved from the optimal solution of the proposed SOCP relaxation (see Section 2 for the four contributions listed above). Fifth, we design a SOCP relaxation based branch-and-bound algorithm which can well balance the computing time and bound quality (see Section 3). And extensive numerical experiments are carried out to show the effectiveness of the proposed algorithm (see Section 4).

Notation. For a real symmetric matrix $X$, $X \succeq 0$ means that $X$ is positive semidefinite. For $n$ by $n$ real matrices $A = (A_{ij})$ and $B = (B_{ij})$, $A \bullet B = \text{tr}(A^T B) = \sum_{i,j=1}^{n} A_{ij} B_{ij}$. Given a vector $a \in \mathbb{R}^n$, $\text{Diag}(a)$ corresponds to an $n \times n$ diagonal matrix with its diagonal equal to $a$. $I_n$ denotes the $n$-dimensional identity matrix.

2. A simultaneous diagonalization based SOCP relaxation. In this section, we first introduce the classical SOCP relaxation proposed by Kim et al. [15] and then derive a new SOCP relaxation based on the property that $Q$ and $I_n$ can be simultaneously diagonalizable.

Let $Q = \sum_{j=1}^{n} \lambda_j \eta_j \eta_j^T$ where $\lambda_1 \leq \ldots \leq \lambda_r < 0 \leq \lambda_{r+1} \leq \ldots \leq \lambda_n$ are eigenvalues with $r \geq 0$ being the number of negative eigenvalues and $\eta_j$, $j = 1, \ldots, n$, are corresponding eigenvectors [20]. In this paper, we assume $r \geq 1$, i.e., the objective function is nonconvex. In fact, (TRS) and (CDT) [1, 4, 12] all satisfy the
assumption. Thus the classical SOCP relaxation is (ref. [15]):

\[
\begin{align*}
\min & \quad \sum_{j=1}^{r} \lambda_j \tau_j + \sum_{j=r+1}^{n} \lambda_j (\eta_j^T x)^2 + q^T x, \\
\text{s.t.} & \quad \sum_{j=1}^{n} \alpha_j - 2w_i^T x + w_i^T w_i \leq d_i^2, \quad i = 1, \ldots, m, \\
& \quad \sum_{j=1}^{n} \alpha_j - 2w_i^T x + w_i^T w_i \geq d_i^2, \quad i = m + 1, \ldots, m + p, \\
& \quad Ax \leq b, \\
& \quad (\eta_j^T x)^2 \leq \tau_j, \quad j = 1, \ldots, r, \\
& \quad x_j^2 \leq \alpha_j, \quad j = 1, \ldots, n.
\end{align*}
\]

It is obvious that those auxiliary variables \( \tau_j \) can be infinity since they are not bounded above in (SOCP).

Recently, many researches focus on how to improve the SOCP relaxation by employing the technique of simultaneous diagonalization. In this paper, we also adopt this technique to design a new SOCP relaxation based on the observation that \( U^TQU = \text{Diag}(\lambda_1, \ldots, \lambda_n) \) and \( U^T I_n U = I_n \), where \( U = (\eta_1, \ldots, \eta_n) \). Let \( x = Uy \), then (GTRS) can be equivalently reformulated as:

\[
\begin{align*}
\min & \quad \sum_{j=1}^{n} \lambda_j y_j^2 + q^T U y, \quad \text{(RGTRS)} \\
\text{s.t.} & \quad \sum_{j=1}^{n} y_j^2 - 2w_i^T U y + w_i^T w_i \leq d_i^2, \quad i = 1, \ldots, m, \\
& \quad \sum_{j=1}^{n} y_j^2 - 2w_i^T U y + w_i^T w_i \geq d_i^2, \quad i = m + 1, \ldots, m + p, \\
& \quad A U y \leq b.
\end{align*}
\]

Obviously, if \( y^* \) is an optimal solution of (RGTRS), then \( x^* = U y^* \) is an optimal solution of (GTRS).

Note that, both (GTRS) and (RGTRS) are nonconvex problems. To obtain a convex relaxation of (RGTRS), we first introduce variables \( t_j \) and let \( y_j^2 = t_j \), then relax these equalities into inequalities \( y_j^2 \leq t_j \) for \( j = 1, \ldots, n \). Thus a new SOCP relaxation can be written as:

\[
\begin{align*}
\min & \quad \sum_{j=1}^{n} \lambda_j t_j + q^T U y, \quad \text{(NSOCP)} \\
\text{s.t.} & \quad \sum_{j=1}^{n} t_j - 2w_i^T U y + w_i^T w_i \leq d_i^2, \quad i = 1, \ldots, m, \\
& \quad \sum_{j=1}^{n} t_j - 2w_i^T U y + w_i^T w_i \geq d_i^2, \quad i = m + 1, \ldots, m + p, \\
& \quad A U y \leq b, \\
& \quad y_j^2 \leq t_j, \quad j = 1, \ldots, n.
\end{align*}
\]
In (NSOCP), the auxiliary variables \( t_j \) cannot be infinite since they are bounded above by \( m \) linear constraints
\[
\sum_{j=1}^{n} t_j - 2 \omega_i^T U y + \omega_i^T w_i \leq d_i^2.
\]
Hence, (NSOCP) is better than (SOCP) in general. In order to show the effectiveness of the proposed SOCP relaxation, we first recall the standard SDP relaxation of (GTRS) and then compare it with our relaxation.

The standard SDP relaxation of (GTRS) is as follows (ref. [18]):

\[
\begin{align*}
\min & \quad Q \cdot X + q^T x, \\
\text{s.t.} & \quad \text{tr}(X) - 2 \omega_i^T x + \omega_i^T w_i \leq d_i^2, \quad i = 1, \cdots, m, \\
& \quad \text{tr}(X) - 2 \omega_i^T x + \omega_i^T w_i \geq d_i^2, \quad i = m + 1, \cdots, m + p, \\
& \quad A x \leq b, \\
& \quad X \succeq x x^T.
\end{align*}
\]

And we have the following important conclusion.

**Theorem 2.1.** (NSOCP) and (SDP) share the same optimal objective function value.

Theorem 2.1 implies that the proposed SOCP relaxation provides a lower bound as tight as that of the standard SDP relaxation. However, the SDP relaxation would tremendously enlarge the dimension of the problem by lifting the original \( n \)-dimensional variable vector to an \( n \times n \) or \( (n+1) \times (n+1) \) variable matrix. Since the speeds of existing SDP solvers such as CVX and SeDuMi [11, 23] are quite slow, the SDP relaxation should take a very long computational time or even be unable to process those large-sized problems. Comparatively, our proposed relaxation falls into the SOCP which is much more efficient than the SDP. Therefore, the total efficiency of the model can be largely improved and it should have a much bigger potential to be used in some real-life applications.

Furthermore, there are \( n \) convex quadratic constraints and \( m + p + s \) linear constraints in (NSOCP). When \( p = 0 \), i.e., all constraints of (GTRS) are convex, then (NSOCP) can be reformulated as:

\[
\begin{align*}
\min & \quad \sum_{j=1}^{r} \lambda_j t_j + \sum_{j=r+1}^{n} \lambda_j y_j^2 + q^T U y, \\
\text{s.t.} & \quad \sum_{j=1}^{r} t_j + \sum_{j=r+1}^{n} y_j^2 - 2 \omega_i^T U y + \omega_i^T w_i \leq d_i^2, \quad i = 1, \cdots, m, \\
& \quad A U y \leq b, \\
& \quad y_j^2 \leq t_j, \quad j = 1, \cdots, r.
\end{align*}
\]

In what follows, we will show that the number of second-order cone constraints can be further reduced when there exists a \( \hat{r} \geq 2 \) such that \( \lambda_1 = \cdots = \lambda_{\hat{r}} \). In such a case, we can get the following relaxation:

\[
\begin{align*}
\min & \quad \lambda_1 t_1 + \sum_{j=\hat{r}+1}^{n} \lambda_j t_j + q^T U y, \\
\text{s.t.} & \quad t_1 + \sum_{j=\hat{r}+1}^{n} t_j - 2 \omega_i^T U y + \omega_i^T w_i \leq d_i^2, \quad i = 1, \cdots, m,
\end{align*}
\]
\[ t_1 + \sum_{j=r+1}^{n} t_j - 2w_i^T U y + w_i^T w_1 \geq d_i^2, \quad i = m + 1, \cdots, m + p, \]
\[ A U y \leq b, \]
\[ \sum_{j=1}^{\hat{r}} y_j^2 \leq t_1, \]
\[ y_j^2 \leq t_j, \quad j = \hat{r} + 1, \cdots, n. \]

And we have the following theorem.

**Theorem 2.2.** When there exists a \( \hat{r} \geq 2 \) such that \( \lambda_1 = \cdots = \lambda_{\hat{r}} \), (NSOCP) and (NSOCP1) share the same optimal objective function value.

It is worth pointing out that, although (NSOCP1) has fewer constraints than (NSOCP), the new constraint \( \sum_{j=1}^{\hat{r}} y_j^2 \leq t_1 \) is more difficult than those simple constraints \( y_j^2 \leq t_j, \quad j = 1, \ldots, \hat{r} \). Hence, (NSOCP1) does not guarantee a superior efficiency compared with (NSOCP). In fact, they win or lose each other in different cases. And the numerical experiment shows this result in Section 4.3.

Generally speaking, (NSOCP) provides a lower bound of (RGTRS). Next, we will prove that (NSOCP) is exact for a special case of (RGTRS).

**Lemma 2.3.** If \((\bar{y}, \bar{t})\) is an optimal solution of (NSOCP), let \( y^* = \bar{y}, \quad t_j^* = (y_j^*)^2 \) for \( j = 2, \ldots, n \), and \( \tilde{t}_j = \sum_{j=1}^{n} \tilde{t}_j - \sum_{j=2}^{n} t_j^* \), then \((y^*, t^*)\) is also an optimal solution of (NSOCP).

In order to state the following theorem and proof concisely, we first define some notations for substitutions. Define \( v_0 = U^T q \) and \( v_i = -2U^T w_i \) for \( i = 1, \ldots, m + p \). Then let \( v_{ij} \) denote the \( j \)-th element of vector \( v_i \) for \( i = 0, \ldots, m + p \).

**Theorem 2.4.** When \( s = 0 \) and the optimal solution of (NSOCP) is obtained, (NSOCP) and (RGTRS) share the same optimal objective function value if one of the following two conditions is satisfied.

1. \( \max \{ v_{11}, \quad i = 1, \ldots, m \} \leq \frac{v_{01}}{\lambda_1} \leq \min \{ v_{11}, \quad i = m + 1, \ldots, m + p \} \),
2. \( \min \{ v_{11}, \quad i = 1, \ldots, m \} \geq \frac{v_{01}}{\lambda_1} \geq \max \{ v_{11}, \quad i = m + 1, \ldots, m + p \} \).

Moreover, the optimal solution of (RGTRS) can be retrieved from the one of (NSOCP).

It is worth noting that, for a quadratic programming problem with one or two quadratic constraints, Ben et al. [4] proved that their SOCP relaxation based on the simultaneous diagonalization is exact under a certain condition. Since (RGTRS) has \( m + p \) quadratic constraints, it is a generalization of their problem. Therefore, the above theorem indeed extends the conclusion of [4]. Next, a simple example is given to illustrate the solving process using Theorem 2.4.

**Example 1.** Consider the following nonconvex quadratic program with three quadratic constraints:

\[
\min \quad -2y_1^2 - y_2^2 - 4y_1 + y_2, \\
\text{s.t.} \quad \| y - \begin{bmatrix} 2 \\ -1 \end{bmatrix} \| \leq 4, \\
\quad \| y - \begin{bmatrix} -1 \\ 1 \end{bmatrix} \| \leq 1,
\]
The optimal solution and optimal objective function value of (NSOCP) are $y^* = (-0.4470, 1.5000)^T$, $t^* = (0.6440, 2.2501)^T$ and $lb = -0.25$, respectively. It is straightforward to verify that $\max_v v_i, i = 1, 2 \leq \frac{\max_i v_i}{\max_i v_{i+1}} \leq v_{31}$. Hence, using Theorem 2.4, we can obtain $\hat{y} = (-0.1340, 1.5000)^T$ which is an optimal solution of (RGTRS).

3. A SOCP relaxation based branch-and-bound algorithm. In this section, we develop a branch-and-bound algorithm for (GTRS) based on the proposed SOCP relaxation. Note that, the branch-and-bound algorithm is a common and effective scheme to achieve the global optimal solution of the nonconvex quadratically constrained quadratic program [19].

Kim et al. [15] pointed out that it is necessary to add some appropriate constraints on the auxiliary variables to improve the lower bound. Since both (GTRS) and (RGTRS) are bounded, there exists a lower bound $l_j$ and an upper bound $u_j$ for each $y_j, j = 1, \cdots, n$. Thus we can add $n$ RLT constraints into (NSOCP). Note that, the RLT constraints are in the following forms:

$$t_j \leq (l_j + u_j)y_j - l_j u_j. \quad \text{(RLT)}$$

Hence, the proposed SOCP relaxation with RLT constraints is in the form:

$$\begin{align*}
\min & \quad \sum_{j=1}^n \lambda_j t_j + q^T U y, \\
\text{s.t.} & \quad \sum_{j=1}^n t_j - 2w_i^T U y + w_i^T w_i \leq d_i^2, \ i = 1, \cdots, m, \\
& \quad \sum_{j=1}^n t_j - 2w_i^T U y + w_i^T w_i \geq d_i^2, \ i = m + 1, \cdots, m + p, \\
& \quad AUy \leq b, \\
& \quad y_j^2 \leq t_j, \ j = 1, \cdots, n, \\
& \quad t_j \leq (l_j + u_j)y_j - l_j u_j, \ j = 1, \cdots, n.
\end{align*}$$

Proposition 1. If $(y^*, t^*)$ is an optimal solution of (NSOCP+RLT) and $t_j^* = (y_j^*)^2$ for $j = 1, \cdots, n$, then $y^*$ is an optimal solution of (RGTRS).

Proof. If $t_j^* = (y_j^*)^2$, then $\sum_{j=1}^n \lambda_j t_j^* + q^T U y^* = \sum_{j=1}^n \lambda_j (y_j^*)^2 + q^T U y^*$. It is obvious that $y^*$ is an optimal solution of (RGTRS). \qed

Proposition 1 shows that if (NSOCP+RLT) is not tight, then there must exist some $j \in \{1, \cdots, n\}$ such that $t_j^* > (y_j^*)^2$. Based on this observation, we can design a branching strategy by selecting the index with the maximal $t_j^* - (y_j^*)^2$. Before describing the branch-and-bound algorithm, we give the following definition.

Definition 3.1. For a given $\epsilon > 0$ and a vector $y \in \mathbb{R}^n$, if $y^T y - 2w_i^T U y + w_i^T w_i \leq d_i^2 + \epsilon$ for all $i = 1, \ldots, m$ and $y^T y - 2w_i^T U y + w_i^T w_i \geq d_i^2 - \epsilon$ for all $i = m + 1, \ldots, m + p$, then $y$ is called an $\epsilon$-feasible solution of (RGTRS). Let $V_{\text{RGTRS}}$ be the optimal objective function value of (RGTRS), if the $\epsilon$-feasible solution $y$ also satisfies $\sum_{j=1}^n \lambda_j y_j^2 + q^T U y \leq V_{\text{RGTRS}} + \epsilon$, then $y$ is called an $\epsilon$-optimal solution.
of (RGTRS) and $x = Uy$ is called an $\epsilon$-optimal solution of (GTRS). Define the function

$$f_\epsilon(y) = \begin{cases} \sum_{j=1}^{n} \lambda_j y_j^2 + q^T Uy & \text{if } y \text{ is } \epsilon\text{-feasible}, \\ +\infty, & \text{otherwise} \end{cases}$$

Our proposed algorithm is presented in Algorithm 1. Its framework includes the following four steps:

1. **Initialization.** The initial lower bound $l_j^0$ and upper bound $u_j^0$ of $y_j$, $j = 1, \cdots, n$, are obtained by solving the following $2n$ convex quadratically constrained linear programming problems:

$$l_j^0 / u_j^0 = \min / \max y_j,$$

$s.t. \quad \|Uy - w_i\| \leq d_i, \quad i = 1, \cdots, m,$

$$AUy \leq b.$$

2. **The node selection strategy.** We use the classical “best-first” selection strategy, i.e., select the one with the lowest bound among the live subproblems.

3. **The variable selection strategy and branching rules.** Let $(y^k, t^k)$ be the solution of the (NSOCP+RLT) at the current node $k$ over $[l^k, u^k]$. Set $j^* = \arg \max_{j \in \{1, \cdots, n\}} (t^k_j - (y^k_j)^2)$, where $j^*$ is the index of the selected variables $(y^k, t^k)$.

   Then set $l^a = l^k$, $u^a = u^k$, $u^a_j = l^k_j + u^k_j$, $l^b = l^k$, $u^b = u^k$, $l^b_j = l^k_j + u^k_j$. Thus, one branching node is over $[l^a, u^a]$ and the other branching node is over $[l^b, u^b]$.

4. **Lower Bound and Upper Bound.** As pointed out by the proposed branching rule, each branching node is over an interval $[l, u]$. Then we add different RLT constraints based on different intervals into (NSOCP+RLT). We compute the lower bound $lb$ for each branching node via solving (NSOCP+RLT). Also, we get an upper bound $ub = f_\epsilon(y^*)$, where $(y^*, t^*)$ is an optimal solution of (NSOCP+RLT).

Finally, the following Lemma and Theorem show that Algorithm 1 converges in finite iterations and returns an $\epsilon$-optimal solution.

**Lemma 3.2.** Assume the node $\{y^k, t^k, l^b_k, l^a_k\}$ is chosen from $D$ in Line 11 of Algorithm 1 and $j^* = \arg \max_{j \in \{1, \cdots, n\}} (t^k_j - (y^k_j)^2)$. For any $\epsilon > 0$, there exists a $\delta > 0$ such that if $(u^k_j - l^k_j) \leq \delta$, then Algorithm 1 terminates in Line 13.

**Theorem 3.3.** Algorithm 1 returns an $\epsilon$-optimal solution after taking at most

$$N_\epsilon = \prod_{j=1}^{n} \left[ \frac{\max\{\sum_{j=1}^{r} |\lambda_j|, \sqrt{n}\}}{\sqrt{\epsilon}} (u_j^0 - l_j^0) \right]$$

iterations.
Algorithm 1 A Branch-and-Bound Algorithm (NSOCP_BB) for Solving (GTRS)

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

1: *Initialization:* Solve (y-Bound) for $l^0$ and $u^0$.
2: if (y-Bound) is infeasible, then
3: (GTRS) is infeasible and terminate.
4: end if
5: *Lower bound and upper bound:* Solve (NSOCP+RLT) over $[l^0, u^0]$ for its optimal objective function value $lb^0$ (a lower bound of (GTRS)) and optimal solution $(y^0, t^0)$. Let $x^* = Uy^0$ and $V^* = f_\epsilon(y^0)$ (an upper bound of (GTRS)).
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: *Node selection strategy:* Choose a node from $D$ by the node selection strategy, denoted as $\{y^k, t^k, lb^k, l^k, u^k\}$ such that $lb^k = \min\{lb^i | lb^i \in D\}$. Delete it from $D$.
12: if $V^* - lb^k \leq \epsilon$, then
13: return $(x^*, V^*)$ and terminate.
14: end if
15: Set $k \leftarrow k + 1$.
16: *Variable selection strategy:* Choose $j^*$ by the variable selection strategy.
17: *Branching:* Set $l^a = l^k$, $u^a = u^k$, $u^a_{j^*} = l^k_{j^*} + u^k_{j^*}$, $l^b = l^k$, $u^b = u^k$, $t^a_{j^*} = \frac{lb^k + ub^k}{2}$ by the branching rules.
18: if (NSOCP+RLT) over $[l^a, u^a]$ is feasible, then
19: *Lower bound and upper bound:* Solve (NSOCP+RLT) over $[l^a, u^a]$ for its optimal objective function value $lb^a$ and optimal solution $(y^a, t^a)$. Denote $ub^a = f_\epsilon(y^a)$.
20: if $ub^a < V^*$, then
21: $V^* = ub^a$ and $x^* = Uy^a$.
22: end if
23: if $V^* - lb^a > \epsilon$, then
24: insert $\{y^a, t^a, lb^a, l^a, u^a\}$ into $D$.
25: end if
26: end if
27: if (NSOCP+RLT) over $[l^b, u^b]$ is feasible, then
28: *Lower bound and upper bound:* Solve (NSOCP+RLT) over $[l^b, u^b]$ for its optimal objective function value $lb^b$ and optimal solution $(y^b, t^b)$. Denote $ub^b = f_\epsilon(y^b)$.
29: if $ub^b < V^*$, then
30: $V^* = ub^b$ and $x^* = Uy^b$.
31: end if
32: if $V^* - lb^b > \epsilon$, then
33: insert $\{y^b, t^b, lb^b, l^b, u^b\}$ into $D$.
34: end if
35: end if
36: end loop
4. Numerical experiments. In this section, three types of (GTRS) problems including trust-region problem with linear inequality constraints [3], generalized Celis-Dennis-Tapia problem [7] and the subproblem of Sparse Source localization [3] are used to test the effectiveness of the proposed algorithm. When \( p = 0 \), we compare the proposed algorithm with the benchmark ED algorithm [16] which is a SDP relaxation based branch-and-bound algorithm for nonconvex quadratic program with convex quadratic constraints. When \( p > 0 \), we compare the proposed algorithm with the benchmark ACS algorithm [17] which is a sensitive-eigenvector based branch-and-bound algorithm for nonconvex quadratically constrained quadratic programming problem. Lu et al. [17] showed that ACS algorithm is efficient for solving nonconvex quadratic program with dense parameters and nonconvex quadratic constraints and is competitive with the solvers BARON, COUENNE and SCIP. These algorithms are implemented in MATLAB R2013b on a PC with Windows 7, 2.50 GHZ Inter Dual Core CPU processors and 8 GB RAM. The SOCP relaxations are solved by Cplex solver 12.6.3 and the SDP relaxations are solved by SeDuMi 1.02 [23]. In all algorithms, the error tolerance is set to be \( \epsilon = 10^{-3} \). Ten instances are generated for each given problem size in Tables 2 and 3. The (average) evaluated nodes ((ave) nodes) and (average) CPU time in seconds ((ave) CPU(sec)) are displayed for each algorithm in the following three tables.

### 4.1. The trust-region problem with linear inequality constraints.

In the case of \( m = 1 \) and \( s \geq 1 \), (GTRS) turns into the trust-region problem with linear inequality constraints. The detailed information of the instances is shown in Table 1. Note that, they are chosen from the open source data sets [3] available at [https://drive.google.com/file/d/0B9PeyTrETyApSDY2bllLNUY2cUE](https://drive.google.com/file/d/0B9PeyTrETyApSDY2bllLNUY2cUE). We use (NSOCP+RLT) to obtain the lower bound and compare the proposed algorithm with the ED algorithm.

| Case name       | \( n \) | \( s \) | NSOCP_BB nodes | NSOCP_BB CPU(sec) | ED nodes | ED CPU(sec) | BFS [3] nodes | BFS [3] CPU(sec) |
|-----------------|--------|--------|----------------|-------------------|----------|-------------|----------------|-----------------|
| Data_lin_8_20   | 8      | 20     | 19             | 0.5483            | 19       | 2.0817      | 3471           | 3.56            |
| Data_lin_10_11  | 10     | 11     | 5              | 0.3004            | 1        | 0.4391      | 155            | 0.238           |
| Data_300_90     | 90     | 90     | 1              | 0.6515            | 1        | 0.7701      | 3              | 0.0341          |
| Data_300_10     | 100    | 10     | 5              | 6.5704            | 15       | 113.0250    | 93             | 0.29            |
| Data_300_15     | 100    | 15     | 11             | 7.3087            | 33       | 281.2684    | 6439           | 21.11           |
| Data_200_10     | 200    | 10     | 2              | 9.6405            | 1        | 11.7755     | 255            | 0.96            |
| Data_200_15     | 200    | 15     | 10             | 27.6235           | 8        | 81.9576     | 2047           | 7.93            |
| Data_300_10     | 300    | 10     | 22             | 31.8774           | 21       | 304.4157    | 287            | 3.75            |
| Data_300_15     | 300    | 15     | 29             | 40.7551           | 22       | 426.1334    | 8959           | 128.03          |

It can be observed from Tables 1 that the proposed algorithm clearly outperforms the ED algorithm in CPU time and this superiority will be largely magnified when \( n \) becomes larger. Moreover, we list the computational results of BFS algorithm appearing in [3]. They suggested a branch-and-bound algorithm employing the polynomial solvability of (TRS) to compute all local minimizers of the problem. Though their algorithm seems more effective for most of the instances, we discover that for those problems which are harder to be solved by using the BFS algorithm [3], Algorithm 1 shows its advantage in computational time. For examples, the computational times of the instance Data_3_100_15 are 7.31 and 21.11 seconds for
the proposed algorithm and BFS algorithm, respectively; the computational times of the instance Data3_300_15 are 40.76 and 128.03 seconds for the proposed algorithm and BFS algorithm, respectively.

4.2. The generalized Celis-Dennis-Tapia problem. When \( m \geq 2 \) and \( s = 0 \), (GTRS) turns into the generalized Celis-Dennis-Tapia problem. The instances for the experiment are generated as follows (ref. [7]): The first constraint is set to be \( x^T x \leq 1 \). The entries of \( Q \in \mathbb{R}^{n \times n} \), \( q \in \mathbb{R}^n \), \( P_i \in \mathbb{R}^{n \times n} \), \( \xi_i \in \mathbb{R}^n \), \( i = 1, \cdots, m \) are independently generated from the standard normal distribution and \( Q \) is replaced by its real symmetric parts. Scale all the \( P_i \)'s and \( \xi_i \)'s by \( \|\xi_i\| \), respectively. Then set \( w_i = P_i^{-1}\xi_i \) and \( d_i = \|P_i^{-1}\| \). Obviously, the original point is a feasible solution of the generated instance. To drop the easy instances, we only select the instances which cannot be solved within 1 iteration. Table 2 lists the comparison results with various \( n \) and \( m \).

Table 2. The generalized Celis-Dennis-Tapia problem

| \( n \) | \( m \) | NSOCP_BB | ED |
|---|---|---|---|
| | | ave nodes | ave CPU(sec) | ave nodes | ave CPU(sec) |
| 100 | 4 | 4.0 | 3.4839 | 2.1 | 12.0705 |
| 150 | 4 | 3.8 | 5.6403 | 2.1 | 35.2577 |
| 200 | 4 | 4.4 | 9.9031 | 2.7 | 110.9290 |
| 250 | 4 | 3.5 | 15.3188 | 2.7 | 244.2893 |
| 300 | 4 | 4.1 | 23.9567 | 3.3 | 594.1906 |
| 350 | 4 | 6.8 | 44.4904 | 2.7 | 818.5657 |
| 400 | 4 | 4.7 | 56.4051 | 3.0 | 1670.5532 |
| 100 | 7 | 3.8 | 5.5790 | 2.9 | 19.0970 |
| 150 | 7 | 3.7 | 10.1234 | 2.7 | 49.3179 |
| 200 | 7 | 5.1 | 19.0962 | 3.0 | 141.8477 |
| 250 | 7 | 4.3 | 29.4947 | 2.6 | 254.5980 |
| 300 | 7 | 4.8 | 45.4383 | 2.7 | 554.2895 |
| 350 | 7 | 3.3 | 64.0829 | 2.9 | 1336.9454 |
| 400 | 7 | 4.0 | 91.8627 | 2.9 | 2551.6678 |
| 100 | 10 | 4.9 | 9.3095 | 3.1 | 22.9624 |
| 150 | 10 | 5.6 | 21.4694 | 2.3 | 53.6741 |
| 200 | 10 | 5.3 | 32.4885 | 3.3 | 169.8376 |
| 250 | 10 | 3.6 | 52.2134 | 3.0 | 317.0527 |
| 300 | 10 | 5.3 | 80.7183 | 2.7 | 554.8825 |
| 350 | 10 | 6.3 | 123.4788 | 2.4 | 884.3760 |
| 400 | 10 | 8.3 | 181.5320 | 2.8 | 1584.7842 |
| 100 | 20 | 7.9 | 93.9798 | 3.2 | 99.6843 |
| 150 | 20 | 6.9 | 86.4431 | 2.8 | 120.1224 |
| 200 | 20 | 5.6 | 140.9858 | 3.4 | 286.7472 |
| 250 | 20 | 6.3 | 218.4065 | 2.5 | 430.6950 |
| 300 | 20 | 6.2 | 326.3330 | 3.3 | 913.2317 |
| 350 | 20 | 9.5 | 497.2161 | 3.8 | 1747.1609 |
| 400 | 20 | 7.8 | 672.3612 | 3.0 | 2350.9039 |

It is observed from Table 2 that, for different \( m \), although the average number of the evaluated nodes of the proposed algorithm is slightly larger than that of the algorithm ED, the CPU time is significantly shorter in all the instances. The above conclusion is strengthened when \( n \) becomes larger. This implies that the
The proposed SOCP relaxation provides an effective lower bound much more quickly than the SDP relaxation. Hence, the proposed SOCP relaxation well balances the computing time and bound quality.

4.3. The subproblem of Sparse Source localization. Beck et al. [3] showed that the sparse source localization can be divided into $2^{m+p}$ generalized trust-region subproblems. The instances for the experiment are generated as follows (ref. [3]):

$$\sigma \in \{-1,1\}^{m+p}$$

are randomly generated, $Q = -\sum_{i=1}^{m+p} \sigma_i I$, $w_i \sim U([-49,51])$ for $i = 1, \cdots, m + p$ where $U(\cdot)$ means the uniform distribution. $q = 2 \sum_{i=1}^{m} \sigma_i w_i$, $\alpha = -50e \in \mathbb{R}^n$, $\gamma_i \sim N(0, \zeta)$ where $\zeta$ is a given noise value and $N(\cdot)$ denotes the normal distribution. First set $d_i = \|\alpha - w_i\| + \gamma_i$ and then let $d_i = \max(d_i, 1)$, for $i = 1, \cdots, m + p$. Generate an independent noise component $\xi \sim N(0, 0.300)$ and $d_1 = \max\{d_1 + \xi, 1\}$. In Table 3, we only test the instances which are feasible and satisfying $m > p$. Although a branch-and-bound algorithm based on (NSOCP1) (NSOCP1_BB) is not detailed here, it can be easily obtained by adding corresponding RLT constraints to (NSOCP1) and changing ($\text{NSOCP}^\text{+RLT}$) with ($\text{NSOCP1}^\text{+RLT}$) in the proposed algorithm.

### Table 3. The subproblem of Sparse Source localization

| n | $m + p$ | $\delta$ | NSOCP1_BB | NSOCP_BB | ACS |
|---|---|---|---|---|---|
|   |   |   | ave nodes | ave CPU(sec) | ave nodes | ave CPU(sec) | ave nodes | ave CPU(sec) |
| 3 | 6 | 0.1 | 6.7 | 0.268 | 10.1 | 0.3792 | 12.2 | 2.4176 |
| 3 | 6 | 1 | 10.7 | 0.3737 | 13.0 | 0.4631 | 15.3 | 3.4611 |
| 3 | 10 | 0.1 | 9.1 | 0.3544 | 8.3 | 0.3251 | 15.2 | 2.6044 |
| 3 | 10 | 1 | 13.2 | 0.4549 | 13.5 | 0.4847 | 17.3 | 2.9758 |
| 13 | 10 | 0.1 | 7.5 | 0.2794 | 7.6 | 0.3015 | 15.5 | 3.1541 |
| 13 | 1 | 22 | 0.6727 | 28.3 | 0.9795 | 18.1 | 3.2683 |
| 6 | 0.1 | 13.6 | 0.466 | 16.4 | 0.5690 | 12.6 | 2.7129 |
| 6 | 1 | 22.9 | 0.6634 | 23.5 | 0.7513 | 13.9 | 2.8062 |
| 10 | 0.1 | 15.5 | 0.5797 | 17.4 | 0.6106 | 24.1 | 6.3445 |
| 10 | 1 | 21.1 | 0.6255 | 27.3 | 0.8490 | 25.3 | 5.1821 |
| 14 | 0.1 | 13.4 | 0.4803 | 12.7 | 0.4958 | 17.1 | 4.1855 |
| 14 | 1 | 25.1 | 0.8989 | 32.8 | 1.1869 | 25.3 | 7.1386 |
| 6 | 0.1 | 37.1 | 1.0944 | 44.4 | 1.4771 | 22.6 | 5.2888 |
| 6 | 1 | 33.7 | 0.8846 | 31.8 | 0.9472 | 42.2 | 10.4813 |
| 10 | 0.1 | 20.5 | 0.6051 | 19.3 | 0.6300 | 26.3 | 7.5633 |
| 10 | 1 | 25.5 | 0.7560 | 23.9 | 0.7938 | 20.5 | 5.6457 |
| 13 | 0.1 | 16.1 | 0.6268 | 15.8 | 0.6625 | 24.2 | 6.8016 |
| 13 | 1 | 36.0 | 1.1874 | 35.2 | 1.2666 | 26.0 | 8.3991 |
| 6 | 0.1 | 49.6 | 1.6442 | 48.7 | 1.8016 | 24.6 | 4.5381 |
| 6 | 1 | 9.6 | 0.4971 | 50.3 | 1.9330 | 236.8 | 23.6932 |
| 10 | 0.1 | 25.4 | 0.9164 | 45.9 | 1.6842 | 57.0 | 13.6420 |
| 10 | 1 | 48.3 | 1.5458 | 48.3 | 1.7473 | 63.8 | 16.7940 |
| 7 | 0.1 | 99.0 | 2.8795 | 166.6 | 5.7123 | 189.2 | 46.6260 |
| 7 | 1 | 53.6 | 1.6242 | 93.6 | 3.1935 | 174.2 | 53.2129 |
| 10 | 0.1 | 76.1 | 2.3453 | 79.3 | 2.8379 | 78.7 | 20.8993 |
| 10 | 1 | 342.6 | 9.6811 | 527.2 | 17.8453 | 58.8 | 14.7023 |
| 8 | 0.1 | 186.2 | 5.2471 | 203.3 | 7.3216 | 602.2 | 72.8404 |
| 8 | 1 | 60.9 | 1.8836 | 62.4 | 2.3540 | 202.9 | 47.6604 |
| 10 | 0.1 | 51.5 | 1.7470 | 48.7 | 1.9087 | 17.4 | 3.5242 |
| 10 | 1 | 117.0 | 3.4418 | 100.1 | 3.6475 | 62.0 | 16.2960 |

Table 3 shows that when all the eigenvalues of $Q$ are equivalent, in most of the instances, the proposed branch-and-bound algorithm based on (NSOCP1) is more effective than the algorithm based on (NSOCP). The above observation implies that the proposed SOCP relaxation is indeed further improved in that case. Besides, for the same $n$ and $m + p$, solving the problem with noise value 1 costs more CPU time than that with noise value 0.1 for most of the instances. Hence, the subproblem of Sparse Source localization with a big noise value is harder to solve. Furthermore, the CPU time clearly outperforms that of algorithm ACS [17] in all instances. This again illustrates that the proposed SOCP relaxation well balances the computing time and bound quality.
5. **Conclusion.** By employing the simultaneous diagonalization technique, we design a new SOCP relaxation for (GTRS). Then we prove that this relaxation is equivalent to the well-known standard SDP relaxation. After that, we show that the proposed relaxation is exact under certain cases. Finally, three different types of problems are tested to demonstrate the efficiency of the proposed method. By comparing with benchmark algorithms ED and ACS, the promising results indeed verify its advantage in reducing the CPU time. This is due to its low complexity in obtaining a good lower bound. Moreover, it is worth pointing out that this advantage becomes more and more obvious as the dimension of variables increases. Therefore, the proposed method is more practical for some real-applications in this big data era. For the future research, we may focus on how to improve the upper bound for the case of $p > 0$.

**Acknowledgments.** Zhou’s research has been supported by the National Natural Science Foundation of China Grant # 11701512, Zhejiang Provincial Natural Science Foundation of China Grant # LQ16A010010 and the Swiss Government Excellence Scholarship. Tian’s research has been supported by the National Natural Science Foundation of China Grants #11401485 and # 71331004 and the Fundamental Research Funds for the Central Universities # JBK1805005. Lu’s research has been supported by the National Natural Science Foundation of China Grant # 11701177.

**Appendix**

**Proof of Theorem 2.1:**

**Proof.** On one hand, if $(y, t)$ is a feasible solution of (NSOCP), then define $x = Uy$ and $X = U(yy^T + \text{Diag}(t_1 - y_1^2, \ldots, t_n - y_n^2))U^T$. Since $t_j \geq y_j^2$ for $j = 1, \ldots, n$, it is easy to see that $X \succeq xx^T$. Moreover, it can be proved that

$$Q \bullet X + q^T x = U \text{Diag}(\lambda_1, \ldots, \lambda_n) U^T \bullet X = \text{Diag}(\lambda_1, \ldots, \lambda_n) \bullet U^T X U$$

$$= \sum_{j=1}^{n} \lambda_j t_j + q^T U y,$$  

$$\text{tr}(X) - 2w_i^T x = \sum_{j=1}^{n} t_j - 2w_i^T U y,$$  

$$Ax = AUy.$$

Since $(y, t)$ is a feasible solution of (NSOCP), we have $\sum_{j=1}^{n} t_j - 2w_i^T U y + w_i^T U y \leq d_i^2$, $i = 1, \ldots, m$, $\sum_{j=1}^{n} t_j - 2w_i^T U y + w_i^T U y \geq d_i^2$, $i = m+1, \ldots, m+p$ and $AUy \leq b$. Consequently, $(x, X)$ is also a feasible solution of (SDP) and the optimal objective function value of (SDP) is no more than that of (NSOCP).

On the other hand, if $(x, X)$ is a feasible solution of (SDP), then define $y = U^T x$ and $Y = U^T X U$. Let $t_j$ be the $j_{th}$ diagonal element of $Y$ for $j = 1, \ldots, n$, i.e., $t_j = Y_{jj}$ for $j = 1, \ldots, n$. It is obvious that $Y \succeq yy^T$ and

$$\sum_{j=1}^{n} \lambda_j t_j + q^T U y = \text{Diag}(\lambda_1, \ldots, \lambda_n) \bullet Y + q^T U y$$

$$= \text{Diag}(\lambda_1, \ldots, \lambda_n) \bullet U^T X U + q^T x$$

$$= U \text{Diag}(\lambda_1, \ldots, \lambda_n) U^T \bullet X + q^T x = Q \bullet X + q^T x,$$
\[
\sum_{j=1}^{n} t_j - 2w_i^T U y = \text{tr}(Y) - 2w_i^T U y = \text{tr}(X) - 2w_i^T x, \quad i = 1, \cdots, m + p, \quad (5)
\]

\[
AU y = Ax.
\]

Since \((x, X)\) is a feasible solution of \((\text{SDP})\), we have \(\text{tr}(X) - 2w_i^T x + w_i^T w_i \leq d_i^2, i = 1, \cdots, m, \text{tr}(X) - 2w_i^T x + w_i^T w_i \geq d_i^2, i = m + 1, \cdots, m + p\) and \(Ax \leq b\). Consequently, \((y, t)\) is also a feasible solution of \((\text{NSOCP})\) and the optimal objective function value of \((\text{NSOCP})\) is no more than that of \((\text{SDP})\).

Above all, \((\text{NSOCP})\) and \((\text{SDP})\) share the same optimal objective function value. \(
\square
\)

**Proof of Theorem 2.2:**

**Proof.** On one hand, if \((y, t)\) is a feasible solution of \((\text{NSOCP})\), then let \(\bar{t}_1 = \sum_{j=1}^{r} t_j, \bar{t}_j = t_j\) for \(j = r + 1, \cdots, n\) and \(\bar{y} = y\), it is straightforward to prove that \((\bar{y}, \bar{t})\) is a feasible solution of \((\text{NSOCP1})\) and \(\lambda_1 \bar{t}_1 + \sum_{j=r+1}^{n} \lambda_j \bar{t}_j + q^T U y = \sum_{j=1}^{n} \lambda_j t_j + q^T U y\) when \(\lambda_1 = \cdots = \lambda_r\). Hence, the optimal objective function value of \((\text{NSOCP1})\) is no more than that of \((\text{NSOCP})\).

On the other hand, if \((\bar{y}, \bar{t})\) is a feasible solution of \((\text{NSOCP1})\), then let \(y = \bar{y}, t_j = \bar{t}_j\) for \(j = r + 1, \cdots, n, t_j = y_j^2\) for \(j = 2, \cdots, r\) and \(t_1 = t_1 - \sum_{j=2}^{r} t_j\), then it is easy to prove that \((y, t)\) is a feasible solution of \((\text{NSOCP})\) and \(\lambda_1 \bar{t}_1 + \sum_{j=r+1}^{n} \lambda_j \bar{t}_j + q^T U \bar{y} = \lambda_1 \bar{t}_1 + \sum_{j=r+1}^{n} \lambda_j \bar{t}_j + q^T U \bar{y}\) when \(\lambda_1 = \cdots = \lambda_r\). Hence, the optimal objective function value of \((\text{NSOCP})\) is no more than that of \((\text{NSOCP1})\).

In sum, \((\text{NSOCP})\) and \((\text{NSOCP1})\) share the same optimal objective function value when \(\lambda_1 = \cdots = \lambda_r\). \(
\square
\)

**Proof of Lemma 2.3:**

**Proof.** First, it is easy to prove that \((y^*, t^*)\) is also a feasible solution of \((\text{NSOCP})\) due to that \(\sum_{j=1}^{n} t_j^* = \sum_{j=1}^{n} \bar{t}_j\) and \(y^* = \bar{y}\). Second,

\[
\sum_{j=1}^{n} \lambda_j t_j^* + q^T U y^* = \sum_{j=1}^{n} \lambda_j \bar{t}_j + q^T U \bar{y}
\]

\[
= \sum_{j=1}^{n} \lambda_j (t_j^* - \bar{t}_j) = \lambda_1 (t_1^* - \bar{t}_1) + \sum_{j=2}^{n} \lambda_j (t_j^* - \bar{t}_j)
\]

\[
= \lambda_1 \sum_{j=1}^{n} \bar{t}_j - \sum_{j=2}^{n} t_j^* - \bar{t}_1 + \sum_{j=2}^{n} \lambda_j (\bar{y}_j^2 - \bar{t}_j)
\]

\[
= \lambda_1 \sum_{j=1}^{n} \bar{t}_j - \sum_{j=2}^{n} (y_j^*)^2 + \sum_{j=2}^{n} \lambda_j (\bar{y}_j^2 - \bar{t}_j)
\]

\[
= \sum_{j=2}^{n} (\lambda_1 - \lambda_j) (\bar{t}_j - \bar{y}_j^2).
\]

Since \(\lambda_1 \leq \lambda_j\) and \(\bar{t}_j \geq \bar{y}_j^2\) for \(j = 2, \cdots, n\), we have \(\sum_{j=2}^{n} (\lambda_1 - \lambda_j) (\bar{t}_j - \bar{y}_j^2) \leq 0\). Moreover, since \((\text{NSOCP})\) is a minimization problem, \((\bar{y}^*, \bar{t}^*)\) is also an optimal solution of \((\text{NSOCP})\). \(
\square
\)
Proof of Theorem 2.4:

Proof. Suppose an optimal solution of (NSOCP) is \((y^*, t^*)\). If \(t_j^* = (y_j^*)^2\) for \(j = 1, \ldots, n\), then it is obvious that \(y^*\) is an optimal solution of (RGTRS). Otherwise, from Lemma 2.3, we can obtain an optimal solution \((y^*, t^*)\) which satisfies \(t_j^* = (y_j^*)^2\) for \(j = 2, \ldots, n\) and \(t_1^* > (y_1^*)^2\). Moreover, we construct a new vector \(\tilde{y}\) where \(\tilde{y}_j = y_j^*\) for \(j = 2, \ldots, n\) and

\[
\tilde{y}_1 = \begin{cases} 
-\frac{\nu}{\lambda_1} - \frac{-\nu - \sqrt{\nu^2 + 4\lambda_1 (\nu t_1^* + \nu_0 y_1^*)}}{2\lambda_1}, & \text{if } \max \{v_{i1}, i \leq m\} \leq \frac{\nu}{\lambda_1} \leq \min \{v_{i1}, i > m\}, \\
-\frac{\nu}{\lambda_1} + \frac{-\nu + \sqrt{\nu^2 + 4\lambda_1 (\nu t_1^* + \nu_0 y_1^*)}}{2\lambda_1}, & \text{if } \min \{v_{i1}, i \leq m\} \geq \frac{\nu}{\lambda_1} \geq \max \{v_{i1}, i > m\}.
\end{cases}
\]

Then we will prove that \(\tilde{y}\) is an optimal solution of (RGTRS).

When \(\max \{v_{i1}, i = 1, \ldots, m\} \leq \frac{\nu_0}{\lambda_1} \leq \min \{v_{i1}, i = m + 1, \ldots, m + p\}\), for the equation \(\lambda_1 z^2 + v_{01} z - (\lambda_1 t_1^* + v_{01} y_1^*) = 0\), it is easy to check that \(\tilde{y}_1\) is one of its solutions and \(\tilde{y}_i\) is the bigger one. Since \(\lambda_1 (y_1^*)^2 + v_{01} y_1^* - (\lambda_1 t_1^* + v_{01} y_1^*) > 0\), then we have \(\tilde{y}_1 > y_1^*\) due to that \(\lambda_1 < 0\). Hence, for \(i = 1, \ldots, m\),

\[
\tilde{y}_1^2 + v_{i1} \tilde{y}_1 = \frac{\nu}{\lambda_1} \tilde{y}_1 + \frac{\nu}{\lambda_1} v_{i1} \tilde{y}_1 = \tilde{t}_1^* + v_{i1} y_1^* + (v_{i1} - \frac{\nu_0}{\lambda_1})(\tilde{y}_1 - y_1^*)
\]

and for \(i = m + 1, \ldots, m + p\),

\[
\tilde{y}_1^2 + v_{i1} \tilde{y}_1 = \tilde{t}_1^* + v_{i1} y_1^* + (v_{i1} - \frac{\nu_0}{\lambda_1})(\tilde{y}_1 - y_1^*)
\]

When \(\min \{v_{i1}, i = 1, \ldots, m\} \geq \frac{\nu_0}{\lambda_1} \geq \max \{v_{i1}, i = m + 1, \ldots, m + p\}\), for the equation \(\lambda_1 z^2 + v_{01} z - (\lambda_1 t_1^* + v_{01} y_1^*) = 0\), it is straightforward that \(\tilde{y}_1\) is one of its solutions and \(\tilde{y}_i\) is a smaller one. Since \(\lambda_1 (y_1^*)^2 + v_{01} y_1^* - (\lambda_1 t_1^* + v_{01} y_1^*) > 0\), we have \(\tilde{y}_1 < y_1^*\). Following the similar steps, we can get the same conclusion that

\[
\tilde{y}_1^2 + v_{i1} \tilde{y}_1 \leq \tilde{t}_1^* + v_{i1} y_1^*, i = 1, \ldots, m
\]

\[
\tilde{y}_1^2 + v_{i1} \tilde{y}_1 \geq \tilde{t}_1^* + v_{i1} y_1^*, i = m + 1, \ldots, m + p.
\]

For \(j = 2, \ldots, n\) and \(i = 1, \ldots, m + p\), it is obvious that \(\tilde{y}_j^2 + v_{ij} \tilde{y}_j = t_j^* + v_{ij} y_j^*\). Then it follows that \(\tilde{y}\) is a feasible solution of (RGTRS) because \(\sum_{j=1}^{n} (\tilde{y}_j^2 + v_{ij} \tilde{y}_j) \leq \sum_{j=1}^{n} (t_j^* + v_{ij} y_j^*) = \sum_{j=1}^{n} t_j^* - 2w_i^T U y^*\) for \(i = 1, \ldots, m\) and \(\sum_{j=1}^{n} (\tilde{y}_j^2 - 2w_i^T U \tilde{y}) = \sum_{j=1}^{n} (\tilde{y}_j^2 + v_{ij} \tilde{y}_j) \geq \sum_{j=1}^{n} (t_j^* + v_{ij} y_j^*) = \sum_{j=1}^{n} t_j^* - 2w_i^T U y^*\) for \(i = m + 1, \ldots, m + p\). Besides,

\[
\sum_{j=1}^{n} \lambda_j \tilde{y}_j^2 + \sum_{j=1}^{n} v_{0j} \tilde{y}_j
\]
that.

Besides, half along the direction perpendicular to the selected edge to generate two new sub-boxes. If Algorithm 1 does not terminate in line 13, we claim that, for each sub-box \([\tilde{u}, \tilde{v}]\),

\[
\sum_{j=1}^{n} \lambda_j \tilde{y}_j^2 + q^T U \tilde{y}^* = \sum_{j=1}^{n} \lambda_j (\tilde{y}_j^*)^2 + q^T U \tilde{y}^*
\]

and for \(i = m+1, \ldots, m+p,\)

\[
(y^k)^T y^k - 2w_i^T U y^k + w_i^T w_i - d_i^2 - (\sum_{j=1}^{n} t_j^k - 2w_i^T U y^k + w_i^T w_i - d_i^2)
\]

\[
= -\sum_{j=1}^{n} t_j^k - (y_j^k)^2 \geq -n(t_j^k - (y_j^k)^2) \geq -n\frac{(u_j^k - l_j^k)^2}{4} \geq -n\delta^2.
\]

Besides,

\[
\sum_{j=1}^{n} \lambda_j (y_j^k)^2 + q^T U y^k - V_{RGTRS}
\]

\[
\leq \sum_{j=1}^{n} \lambda_j (y_j^k)^2 + q^T U y^k - lb_k = \sum_{j=1}^{n} \lambda_j ((y_j^k)^2 - t_j^k)
\]

\[
\leq \sum_{j=1}^{r} |\lambda_j| (t_j^k - (y_j^k)^2) \leq (t_j^k - (y_j^k)^2) \sum_{j=1}^{r} |\lambda_j|
\]

\[
\leq \sum_{j=1}^{r} |\lambda_j| \frac{(u_j^k - l_j^k)^2}{4} \leq \sum_{j=1}^{r} |\lambda_j| \delta^2.
\]

For any \(\epsilon > 0,\) we set \(\delta = \min\{\frac{2\sqrt{\epsilon}}{\sum_{j=1}^{n} |\lambda_j|}, \frac{2\sqrt{\epsilon}}{\max\{\sum_{j=1}^{n} |\lambda_j|, \sqrt{n}\}}\},\) then \(y^k\) is an \(\epsilon\)-optimal solution from our definition. Consequently, Algorithm 1 terminates in Line 13.

**Proof of Theorem 3.3:**

Proof. Under the condition that problem (GTRS) is bounded with nonempty relative interior points, the initial box \([l_0, u_0]\) is bounded and nonempty. In every iteration of Algorithm 1, if it does not terminate in line 13, then a chosen box is split in half along the direction perpendicular to the selected edge to generate two new sub-boxes. After \(k\) iterations, the initial box \([l_0, u_0]\) would be split into \(k+1\) sub-boxes. If Algorithm 1 does not terminate in line 13, we claim that, for each sub-box \([l_j^*, u_j^*]\) among those \(k+1\) sub-boxes, \(u_j^* - l_j^* \geq \frac{4}{\sqrt{n}}u_j^0 - l_j^0\) with \(\delta = \max\{\sum_{j=1}^{n} |\lambda_j|, \sqrt{n}\}\) for \(j = 1, \ldots, n\). If there exists a \(j^*\) such that \(u_{j^*}^* - l_{j^*}^* \leq \delta\), then the \(j^*\)-th edge
has never been selected as a branching direction. Otherwise, if the node \( z \) is chosen by the node selection strategy in line 11, Algorithm 1 does not terminate in line 13 and \( j^* \) is chosen by the variable selection strategy in line 16. Then, following from Lemma 3.2, we know \( y^* \) is an \( \epsilon \)-optimal solution as we expected. This contradicts with the assumption that Algorithm 1 does not get an \( \epsilon \)-optimal solution in line 13 at node \( z \). Hence, the volume of each sub-box is no smaller than \( \prod_{j=1}^{n} \min\{\frac{\delta}{2}, u_j^0 - l_j^0\} \).

Let \( N_\epsilon = \prod_{j=1}^{n} \left[ \frac{2(u_j^0 - l_j^0)}{\delta} \right] = \prod_{j=1}^{n} \left[ \frac{\max\{\sum_{i=1}^{\lambda_i} |\sqrt{\pi}| \lambda_j \}}{\sqrt{\epsilon}}(u_j^0 - l_j^0) \right] \). Since the total volume of all the \( k + 1 \) sub-boxes is no more than that of the initial box \([l^0, u^0]\), it is easy to check that, at most \( N_\epsilon \) iterations, Algorithm 1 must satisfy the termination condition and return an \( \epsilon \)-optimal solution. \( \square \)

REFERENCES

[1] W. Ai and S. Zhang, Strong duality for the CDT subproblem: A necessary and sufficient condition, SIAM J. Optim., 19 (2009), 1735–1756.
[2] A. I. Barvinok, Feasibility testing for systems of real quadratic equations, Discrete Comput. Geom., 10 (1993), 1–13.
[3] A. Beck and D. Pan, A branch and bound algorithm for nonconvex quadratic optimization with ball and linear constraints, J. Global Optim., 69 (2017), 309–342.
[4] A. Ben-Tal and D. Den Hertog, Hidden conic quadratic representation of some nonconvex quadratic optimization problems, Math. Program., 143 (2014), 1–29.
[5] D. Bienstock and A. Michalka, Polynomial solvability of variants of the trust-region subproblem, Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms Society for Industrial and Applied Mathematics, New York, 2014, 380–390.
[6] D. Bienstock, A note on polynomial solvability of the CDT problem, SIAM J. Optim., 26 (2016), 488–498.
[7] I. M. Bomze and M. L. Overton, Narrowing the difficulty gap for the Celis-Dennis-Tapia problem, Math. Program., 151 (2015), 459–476.
[8] I. M. Bomze, V. Jeyakumar and G. Li, Extended trust-region problems with one or two balls: Exact copositive and Lagrangian relaxations, J. Global Optim., 71 (2018), 551–569.
[9] S. Burer and K. M. Anstreicher, Second-order-cone constraints for extended trust-region subproblems, SIAM J. Optim., 23 (2013), 432–451.
[10] M. R. Celis, J. E. Dennis and R. A. Tapia, A trust region strategy for nonlinear equality constrained optimization, Numerical Optimization, 1984 (1985), 71–82.
[11] M. Grant and S. Boyd, CVX: Matlab Software for Disciplined Convex Programming, version 2.1, (2014). Available at: http://cvxr.com/cvx.
[12] N. Ho-Nguyen and F. Kilinc-Karzan, A second-order cone based approach for solving the trust-region subproblem and its variants, SIAM J. Optim., 27 (2017), 1485–1512.
[13] R. Jiang and D. Li, Simultaneous diagonalization of matrices and its applications in quadratically constrained quadratic programming, SIAM J. Optim., 26 (2016), 1649–1668.
[14] V. Jeyakumar and G. Y. Li, Trust-region problems with linear inequality constraints: Exact SDP relaxation, global optimality and robust optimization, Math. Program., 147 (2014), 171–206.
[15] S. Kim and M. Kojima, Second order cone programming relaxation of nonconvex quadratic optimization problems, Optim. Methods Softw., 15 (2001), 201–224.
[16] C. Lu, Z. Deng and Q. Jin, An eigenvalue decomposition based branch-and-bound algorithm for nonconvex quadratic programming problems with convex quadratic constraints, J. Global Optim., 67 (2017), 475–493.
[17] C. Lu, Z. Deng, J. Zhou and X. Guo, A sensitive-eigenvector based global algorithm for quadratically constrained quadratic programming, J. Global Optim., 73 (2019), 371–388.
[18] Z. Q. Luo, W. K. Ma, A. M. C. So, Y. Ye and S. Zhang, Semidefinite relaxation of quadratic optimization problems, IEEE Signal Processing Magazine, 27 (2010), 20–34.
[19] D. R. Morrison, S. H. Jacobson, J. J. Sauppe and E. C. Sewell, Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning, Discrete Optim., 19 (2016), 79–102.
[20] K. B. Petersen and M. S. Pedersen, The matrix cookbook, *Technical University of Denmark*, 7 (2008), 510pp.

[21] N. Sagara and M. Fukushima, A trust region method for nonsmooth convex optimization, *J. Ind. Manag. Optim.*, 1 (2005), 171–180.

[22] Z. Sheng, G. Yuan, Z. Cui, X. Duan and X. Wang, An adaptive trust region algorithm for large-residual nonsmooth least squares problems, *J. Ind. Manag. Optim.*, 14 (2018), 707–718.

[23] J. F. Sturm, Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones, *Optim. Methods Softw.*, 11 (1999), 625–653.

[24] J. F. Sturm and S. Zhang, On cones of nonnegative quadratic functions, *Math. Oper. Res.*, 28 (2003), 246–267.

[25] Y. Tian, Z. Deng, J. Luo and Y. Li, An intuitionistic fuzzy set based S3VM model for binary classification with mislabeled information, *Fuzzy Optim. Decis. Mak.*, 17 (2018), 475–494.

[26] J. Zhou, S.-C. Fang and W. Xing, Conic approximation to quadratic optimization with linear complementarity constraints, *Comput. Optim. Appl.*, 66 (2017), 97–122.

[27] J. Zhou and Z. Xu, A simultaneous diagonalization based SOCP relaxation for convex quadratic programs with linear complementarity constraints, *Optim. Letters*, 13 (2019), 1615–1630.

Received July 2018; 1st revision February 2019; final revision May 2019.

E-mail address: zhoujing@zjut.edu.cn
E-mail address: lucheng1983@163.com
E-mail address: yetian@swufe.edu.cn
E-mail address: xytang@ie.cuhk.edu.hk