Mathematical Programming Computation (2019) 11:119–171
https://doi.org/10.1007/s12532-018-0142-9

FULL LENGTH PAPER

New global algorithms for quadratic programming with a few negative eigenvalues based on alternative direction method and convex relaxation

Hezhi Luo[1] · Xiaodi Bai[2] · Gino Lim[3] · Jiming Peng[3]

Received: 16 February 2017 / Accepted: 2 August 2018 / Published online: 22 August 2018

Abstract
We consider a quadratic program with a few negative eigenvalues (QP-r-NE) subject to linear and convex quadratic constraints that covers many applications and is known to be NP-hard even with one negative eigenvalue (QP1NE). In this paper, we first introduce a new global algorithm (ADMBB), which integrates several simple optimization techniques such as alternative direction method, and branch-and-bound, to find a globally optimal solution to the underlying QP within a pre-specified ϵ-tolerance. We establish the convergence of the ADMBB algorithm and estimate its complexity. Second, we develop a global search algorithm (GSA) for QP1NE that can locate an optimal solution to QP1NE within ϵ-tolerance and estimate the worst-case complexity bound of the GSA. Preliminary numerical results demonstrate that the ADMBB algorithm can effectively find a global optimal solution to large-scale QP-r-NE instances when $r \leq 10$, and the GSA outperforms the ADMBB for most of the tested QP1NE instances. The software reviewed as part of this submission was given the DOI (digital object identifier) https://doi.org/10.5281/zenodo.1344739.

Keywords Quadratic programming · Alternative direction method · Convex relaxation · Branch-and-bound · Line search · Computational complexity

Mathematics Subject Classification 90C20 · 90C22 · 90C26

The research of Hezhi Luo is jointly supported by NSFC Grants 11871433 and 11371324 and the Zhejiang Provincial NSFC Grants LY17A010023 and LY18A010011.
The research of Xiaodi Bai is supported by NSFC Grants 11371103 and 11701511.
The research of Jiming Peng is supported by NSF Grants CMMI-1131690 and CMMI-1537712.

✉ Jiming Peng
jopeng@uh.edu

Extended author information available on the last page of the article

1 Introduction

In this paper, we consider the following class of quadratically constrained quadratic programming (QCQP) problems defined by

$$\begin{aligned} &\min\ f(x) = x^T Q x + q^T x \\ &s.t.\ x \in \mathcal{F} \end{aligned} \tag{1}$$

where

$$\mathcal{F} = \left\{ x \in \mathbb{R}^n \,\middle|\, \begin{array}{ll} x^T Q_i x + q_i^T x \le d_i, & i = 1, \ldots, m, \\ Ax \le b, & x \ge 0 \end{array} \right\},$$

Q is an $n \times n$ symmetric indefinite or negative definite matrix with r negative eigenvalues, Q_i $(i = 1, \ldots, m)$ are positive semidefinite $n \times n$ matrices, $q \in \mathbb{R}^n$, $q_i \in \mathbb{R}^n$, $d_i \in \mathbb{R}$ $(i = 1, \ldots, m)$, and $A \in \mathbb{R}^{l \times n}$, $b \in \mathbb{R}^l$. The QCQP model has been widely applied in different fields. For a general introduction to QCQP, we refer to the survey papers [10,12] and the references therein. It is well-known that problem (1) is NP-hard and even finding its local minimum is NP-hard [32], while several special subclasses of QCQP have been identified to be polynomially solvable [16,47]. We assume in this paper that $\mathcal{F}$ is bounded and the Slater condition holds for problem (1), i.e., there is an interior point in $\mathcal{F}$.

The broad application range of the QCQP model and the computational challenge it poses have attracted researchers from various disciplines [10] and a host of algorithms and theories have been reported in the literature. The survey papers [10,33] summarized various algorithms and theoretical results on QP up to that time. Many early works on non-convex QP have focused on obtaining first-order or (weak) second-order Karush–Kuhn–Tucker (KKT) critical points with a good objective value. For example, Konno [19] proposed a simple alternative updating scheme for a special subclass of QP (the bilinear program) that converges to a KKT point. He further discussed how to use a local search procedure based on the linear optimization technique to find a local optimum. Palacios-Gomez et al. [29] proposed a successive linear optimization approach that can find a KKT point of QP. Ye [46] presented an interior-point method to find a KKT point of linearly constrained quadratic programming (LCQP). Le Thi and Pham Dinh [22] developed a D.C. (difference of convex functions) optimization algorithm (called DCA) to find a KKT point of LCQP. Gould and Toint [12] gave a survey on various optimization techniques for finding KKT points of LCQP. Several general-purpose global optimization solvers can also be used for nonconvex QPs. For example, Sherali et al. [40] developed a reformulation-linearization-technique (RLT) for solving non-convex QPs. Vavasis [43] developed the so-called ϵ-approximation algorithm for LCQP with a complexity $\mathcal{O}(\lceil \frac{n(n+1)}{\sqrt{\epsilon}} \rceil^r N)$, where r is the number of negative eigenvalues in the associated LCQP, and N is the complexity to solve a convex QP. Visweswaran and Floudas [44] developed a branch-and-bound (B&B) method for the subclass of bi-convex quadratic optimization. For a review of early global optimization methods for non-convex QP, see the survey papers by Pardalos [31] and Floudas and Visweswaran [10]. In two important papers [41,42], Vandenbussche and

Nemhauser used the first-order KKT points to develop a finite B&B algorithm for a box constrained QP. Le Thi and Pham Dinh [22] proposed a decomposition B&B method for LCQP. A combined DCA–B&B algorithm via ellipsoidal technique was proposed in [23] for a box constrained QP, and was further extended in [24] to non-convex QP with convex quadratic constraints. In a series of papers [6–8], Cambini and his colleagues developed specific algorithms for QPs with only a few negative eigenvalues. Off-the-shelf software packages, such as BARON [36] built upon the B&B approach and various convexification techniques, are available for the solution of LCQP.

In the past two decades, the area of semidefinite programming (SDP) has undergone a fast development and many exciting results have been reported [1,45]. Based on the SDP framework, several global approaches for QP have been proposed. For example, Sherali and Adams [39] proposed lifting-and-projection methods for constructing successive linear relaxations of a 0–1 polytope. Lovász and Schrijver [26] considered successive SDRs of a binary polytope and established the finite convergence of the procedure. Lasserre [20] combined results about representations of positive polynomials as sums of squares and the moments theory to construct successive SDRs. It has been shown that Larsserre's hierarchy also enjoys a finite convergence under certain conditions [21,28]. Kojima and Tunçel [17,18] discussed successive SDRs for generic QP. As noted in [17,34,38], the number of variables and constraints grows very fast as the level of the hierarchy increases in the above-mentioned hierarchical procedures. This makes those approaches computationally not scalable. As an alternative, Burer and Vandenbussche [4,5] developed a B&B approach for QP based on SDRs. Saxena et al. [37,38] discussed how to derive disjunctive cuts to improve the SDR for generic QPs. Chen and Burer [9] further adopted the so-called completely positive program to improve the B&B approach for LCQP. As reported in [9], these B&B-based global solvers can solve many small-scale LCQP instances, but may run out of time/memory and return only a suboptimal solution when the problem size is between 50 and 100.

Motivated by such observations, we consider in this paper the issue of developing an effective global solver for large-scale problem (1) with a few negative eigenvalues.[1] For this purpose, we first rewrite the quadratic term in the objective function of problem (1) via the so-called D.C. program [14], i.e.,

$$x^T Qx = x^T Q_+ x - x^T Q_- x = x^T Q_+ x - \|Cx\|^2, \tag{2}$$

where both Q_+ and Q_- are symmetric positive semi-definite matrices derived from the singular value decomposition of Q, and $C \in \mathbb{R}^{r\times n}$ is a matrix whose rows are constructed from the negative eigenvalues of Q and their associated eigenvectors, r is the number of negative eigenvalues of Q. One way to tackle problem (1) is the D.C. algorithm (DCA) [22,35] which iteratively solves a convex optimization problem derived by replacing the quadratic term $-\|Cx\|^2$ by its linear approximation $-2(Cx^k)^T Cx$ at the point x^k. In [22,35], Le Thi and Pham Dinh showed that the

[1] It is worth mentioning that in [8], the authors developed a global algorithm for QPs with a small number of negative eigenvalues. However, the algorithm was tested only on small scale instances.

DCA for QCQPs converges to a stationary point of the underlying problem. They also reported some empirical results showing that the DCA can find very good solutions to many test problems.

In this paper, we propose to develop an effective global solver for problem (1) based on Lagrangian method. To start, we observe that a global optimal solution to problem (1) can be located at some point $t^* = Cx^*$ where x^* is the optimal solution of the problem. Consequently, we can transfer problem (1) into the following lifted optimization problem

$$\min_{(x,t)\in\mathcal{F}_t} x^T Q_+ x + q^T x - \|t\|^2, \tag{3}$$

where

$$\mathcal{F}_t = \left\{(x,t) \in \mathbb{R}^{n+r} | Cx - t = 0,\ x \in \mathcal{F}\right\}.$$

Let $t_l, t_u \in \mathbb{R}^r$ denote the lower and upper bounds of $t = Cx$ over $\mathcal{F}$ obtained via solving the following linear optimization problems

$$t_l^i = \min_{x\in\mathcal{F}} c_i^T x, \quad t_u^i = \max_{x\in\mathcal{F}} c_i^T x, \quad i = 1, \ldots, r, \tag{4}$$

where c_i^T denotes the ith row vector of matrix C. Note that problems (1) and (3) are equivalent. Now let us consider the Lagrangian method for the lifted problem (3) with respect to the constraint $Cx - t = 0$. By selecting the Lagrangian multipliers associated with the negative quadratic items according to the KKT conditions in the Lagrangian method, we reformulate the original problem (1) as the following equivalent bi-convex QP problem

$$\min_{x\in\mathcal{F}, t\in\mathbb{R}^r} x^T Q_+ x + q^T x - 2t^T Cx + \|t\|^2. \tag{5}$$

Based on the above reformulation (5), we propose a new alternative direction method (NADM) for problem (1) that updates x and t alternatively.

It should be pointed out that if we consider only the x-part of the sequence generated by the proposed NADM, it is identical to the sequence provided by the DCA [22,35] when both the NADM and DCA have the same starting point. However, as we shall see in our later analysis, the t-sequence (the Lagrangian multipliers or $Cx(t)$) generated from the NADM enjoy several appealing properties that has not been well explored in the study of DCA. For example, the t-sequence generated by the NADM converges in a monotonic manner to a so-called quasi-ϵ-local optimal solution while the x-sequence converges to a KKT point of problem (1). Moreover, the relatively simple structure of the lifted problem allows us to estimate the gap between the lifted problem and its convex relaxation. As we shall see later, these new properties of the NADM approach and the lifted problem facilitate the development of new effective global solvers for special sub-classes of QCQPs. It should be pointed out that the classical ADM for bi-convex optimization problem converges only to a stationary point or a partial optimal

solution of the underlying problem [2,11], and the DCA can only provide a sequence converging to a stationary point of the underlying problem [22,35].

The primal goal of this work is to develop a new global algorithm (called ADMBB) that can find a globally optimal solution to problem (1) within a pre-specified ϵ-tolerance by combining the NADM algorithm with the B&B framework, convex relaxation and initialization technique. We establish the convergence of the ADMBB and estimate its complexity. Specifically, we show that the ADMBB algorithm has a complexity bound $\mathcal{O}(N\prod_{i=1}^{r}\lceil\frac{\sqrt{r}(t_u^i-t_l^i)}{2\sqrt{\epsilon}}\rceil)$, where N is the complexity for solving the relaxed subproblem, a convex QP. This improves the complexity result of $\mathcal{O}(\lceil\frac{n(n+1)}{\sqrt{\epsilon}}\rceil^r N)$ in [43] for LCQP when $t_u^i - t_l^i \ll n(n+1)$. It should be pointed out that the complexity bound in [43] is also the best-case estimate due to the usage of the discretization scheme, while the complexity bound in our work is the worst-case estimate. Numerical experiments illustrate that the ADMBB algorithm can effectively find a global optimal solution to randomly generated large-scale instances of problem (1) in which the number of negative eigenvalues is fewer than or equals 10.

We also consider the issue of developing new global algorithms for non-convex QP without resorting to the B&B framework. For such a purpose, we focus only on the special case of problem (1) where $r = 1$ (denoted by QP1NE). The QP1NE model covers the classical k-clique problem in discrete optimization that is well-known to be NP-hard [33]. The problem of minimizing the product of two non-negative linear functions, which appears in numerous discrete optimization problems such as the constrained minimal spanning tree problem [13], can also be cast as a special case of QP1NE. We propose a new global algorithm for QP1NE that integrates NADM, convex relaxation and line search technique. To establish the global convergence of the new algorithm, we first show that the NADM for QP1NE enjoys a complexity bound $\mathcal{O}(\frac{t_u^1-t_l^1}{\sqrt{\epsilon}})$, where ϵ is the stopping criteria used in the NADM. We also show that the t-sequence generated from the NADM for QP1NE converges to a so-called semi-local minimum, a notion to be introduced later. Such a property allows us to run the NADM algorithm from t_l^1 and t_u^1 separately. Let t_l^* and t_u^* denote the corresponding accumulation points of the two sequences from these two runs. We show that there exists at least one global optimal solution t^* to the lifted problem of QP1NE in the interval $[t_l^*, t_u^*]$. This indicates that we can use these two runs of the NADM algorithm to substantially reduce the search space of t^*. We then introduce a new line search procedure based on convex relaxation for QP1NE that runs in $\mathcal{O}(\log\frac{t_u^1-t_l^1}{\sqrt{\epsilon}})$ time. We show that the new line search procedure can cut the interval $[t_l^*, t_u^*]$ without missing a potential global optimal solution. By combining the NADM and the new line search technique, we develop a new global algorithm (called GSA) for QP1NE which enjoys a complexity bound $\mathcal{O}(N\frac{t_u^1-t_l^1}{\sqrt{\epsilon}}\log\frac{t_u^1-t_l^1}{\sqrt{\epsilon}})$. This improves the best-known complexity result of $\mathcal{O}(\frac{N}{\epsilon})$ in [13] for a special case of QP1NE in the literature. Numerical results illustrate that the GSA algorithm can more effectively find a global optimal solution to randomly generated large-scale QP1NE instances than the ADMBB with $r = 1$.

The remaining part of the paper is organized as follows. In Sect. 2, we describe the NADM for problem (3) and investigate its properties. Particularly, by applying

the NADM with different starting points, we can obtain a feasible solution to the underlying problem, which can be used as an upper bound in the new B&B approach. In Sect. 3, based on the NADM and convex relaxation of problem (1), we introduce a B&B procedure to find the global optimal solution to problem (1). Global convergence of the proposed algorithm will be established and its complexity will be estimated. In Sect. 4, based on the convex relaxation of QP1NE, we introduce a line search procedure to further narrow down the interval that potentially contains the global optimal solution to QP1NE. Global convergence of the proposed algorithm will be established and its complexity will be estimated. We test the performance of the proposed algorithm and report numerical results in Sect. 5. Finally, we conclude the paper in Sect. 6 by discussing some future research directions.

2 A new alternative direction method and its properties

In this section, we present a new alternative direction method (NADM) for problem (3) and explore its properties to both a KKT point and a quasi-ϵ-local solution of problem (1).

We start by considering a special Lagrangian method for problem (3) as follows

$$\mathcal{L}(x, t, \lambda) = x^T Q_+ x + q^T x - \|t\|^2 + \lambda^T (Cx - t), \tag{6}$$

where we apply the Lagrangian multipliers to the constraint $Cx - t = 0$. Now let us examine the KKT conditions of the above Lagrangian function with respect to t, we have

$$\frac{\partial \mathcal{L}(x, t, \lambda)}{\partial t} = -2t - \lambda = 0 \iff \lambda = -2t. \tag{7}$$

Recall that the optimal solution of problem (3) must be a KKT point of the above Lagrangian function (see page 244 of [3]). Using the relation (7), we can rewrite the Lagrangian function as the following

$$\mathcal{L}^*(x, t) = x^T Q_+ x + q^T x - 2t^T Cx + \|t\|^2. \tag{8}$$

The function $\mathcal{L}^*(x, t)$ is clearly bi-convex. For convenience, we call $\mathcal{L}^*(x, t)$ the optimal Lagrangian function. Note that because

$$x^T Q_+ x + q^T x - 2t^T Cx + \|t\|^2 = x^T Q_+ x + q^T x - \|Cx\|^2 + \|t - Cx\|^2,$$

we can easily prove the following result:

Theorem 2.1 *Problem* (1) *is equivalent to the following bi-convex QP problem*

$$\min_{x \in \mathcal{F}, t \in \mathbb{R}^r} \mathcal{L}^*(x, t) = x^T Q_+ x + q^T x - 2t^T Cx + \|t\|^2. \tag{9}$$

The above theorem establishes the equivalence between non-convex QP and bi-convex QP [11]. It should mention that the idea of recasting non-convex QP as bi-convex optimization has been widely used in the literatures [10,11,44]. The novelty of model (9) lies in the fact that there are no constraints in t while other existing bi-convex models for non-convex QP usually involve certain constraints on t. For a comparison, we mention that Cambini et al. [8] considered a restricted variant of problem (9) with an additional constraint $Cx = t$ for fixed t. One can see that the solution of problem (9) can improve the objective function value more effectively compared with its restricted variant.

Note that in problem (9), t is free and the constraint set $\mathcal{F}$ is convex. If we fix x temporarily, then the optimal solution to problem (9) is $t = Cx$. For temporarily fixed t, problem (9) reduces to a convex QP:

$$\min_{x\in\mathcal{F}} x^T Q_+ x + q^T x - 2t^T Cx + \|t\|^2, \tag{10}$$

which can be solved by many optimization solvers. It is easy to see that for any given $t \in [t_l, t_u]$, problem (10) is feasible and well-defined. Based on the Slater condition, the KKT conditions for problem (10) are necessary for a local minimum. Therefore, we immediately have

Proposition 2.1 *If $\bar{x}$ is an optimal solution of problem (10) with $t = C\bar{x}$, then $\bar{x}$ is a KKT point of problem (1).*

Let $\mathcal{S}(t)$ denote the optimal solution set of problem (10) and $\mu(t)$ the objective value at the optimal solution. The following proposition summarizes some properties at the optimal solution of problem (10) with respect to the parameter t.

Proposition 2.2 *Let $x(t) \in \mathcal{S}(t)$ and $\mu(t)$ be the objective value at the optimal solution of problem (10). We have*

(i) *If $x \in \mathcal{F}$ satisfies the following inequality*

$$||Cx - t|| \le ||Cx(t) - t||,$$

then it must hold

$$f(x) \ge f(x(t)).$$

(ii) *Let x^* be the global optimal solution of the problem (1). Then*

$$\|Cx(t) - t\| \le \left\|Cx^* - t\right\|. \tag{11}$$

(iii) *The map $Cx(t)$ is monotone with respect to t in the sense that*

$$(t_1 - t_2)^T \left[Cx\,(t_1) - Cx\,(t_2)\right] \ge 0, \tag{12}$$

where

$$t_1 \ne t_2 \in [t_l, t_u], \quad x(t_1) \in \mathcal{S}(t_1), \quad x(t_2) \in \mathcal{S}(t_2).$$

(iv) $\mu(t)$ *is a continuous function of* t *for every* $t \in [t_l, t_u]$.

Proof Statement (i) follows easily from the fact that $x(t)$ is the optimal solution of problem (10) and that $\|Cx - t\| \leq \|Cx(t) - t\|$.

Now, we turn to Statement (ii). Since x^* is the global solution of problem (1) and $x(t) \in \mathcal{F}$, we have

$$f(x(t)) \geq f\left(x^*\right). \tag{13}$$

Recall that x^* is a feasible solution of problem (10). Since $x(t) \in \mathcal{S}(t)$, we have

$$f(x(t)) + \|Cx(t) - t\|^2 \leq f(x^*) + \left\|Cx^* - t\right\|^2. \tag{14}$$

It follows from (13) and (14) that

$$\|Cx(t) - t\| \ \leq \ \left\|Cx^* - t\right\|.$$

This proves Statement (ii).

Next we consider Statement (iii). Let $t_1 \neq t_2 \in [t_l, t_u]$, $x(t_1) \in \mathcal{S}(t_1)$, and $x(t_2) \in \mathcal{S}(t_2)$. From the optimality conditions of problem (10), we obtain

$$\begin{aligned} f(x(t_1)) + \|Cx(t_1) - t_1\|^2 &\leq f(x(t_2)) + \|Cx(t_2) - t_1\|^2, \\ f(x(t_2)) + \|Cx(t_2) - t_2\|^2 &\leq f(x(t_1)) + \|Cx(t_1) - t_2\|^2. \end{aligned}$$

Adding the above two inequalities together, we derive

$$t_2^T Cx(t_2) + t_1^T Cx(t_1) \geq t_2^T Cx(t_1) + t_1^T Cx(t_2) \Longleftrightarrow (t_1 - t_2)^T C\left(x(t_1) - x(t_2)\right) \geq 0,$$

which yields relation (12).

Finally, we prove Statement (iv). For a given $t \in [t_l, t_u]$, we consider the following problem

$$\hat{\mu}(t) = \min_{x \in \mathcal{F}} \ x^T Q_+ x + q^T x - 2t^T Cx. \tag{15}$$

Clearly, problems (15) and (10) have the same optimal solution set and $\mu(t) = \hat{\mu}(t) + \|t\|^2$. It thus suffices to show the continuity of $\hat{\mu}(t)$ for $t \in [t_l, t_u]$. Let us consider another point $\bar{t} \in [t_l, t_u]$. Let $x(t) \in \mathcal{S}(t)$ and $x(\bar{t}) \in \mathcal{S}(\bar{t})$. Note that $Cx(\bar{t}) \in [t_l, t_u]$, it is easy to check $\|Cx(\bar{t})\| \leq L$, where $L = \max\{\|t_l\|, \|t_u\|, \|t_u - t_l\|\}$. We thus obtain

$$\begin{aligned} \hat{\mu}(t) &= x(t)^T Q_+ x(t) + q^T x(t) - 2t^T Cx(t) \\ &\leq x(\bar{t})^T Q_+ x(\bar{t}) + q^T x(\bar{t}) - 2t^T Cx(\bar{t}) \\ &= x(\bar{t})^T Q_+ x(\bar{t}) + q^T x(\bar{t}) - 2\bar{t}^T Cx(\bar{t}) + 2\left(\bar{t} - t\right)^T Cx(\bar{t}) \\ &= \hat{\mu}(\bar{t}) + 2\left(\bar{t} - t\right)^T Cx(\bar{t}) \\ &\leq \hat{\mu}(\bar{t}) + 2\left\|\bar{t} - t\right\| \left\|Cx(\bar{t})\right\| \end{aligned}$$

$$\leq \hat{\mu}(\bar{t}) + 2L \left\| \bar{t} - t \right\|,$$

where the second inequality follows from the Schwarz inequality. Similarly, we have

$$\begin{aligned}
\hat{\mu}(t) &= x(t)^T Q_+ x(t) + q^T x(t) - 2t^T C x(t) \\
&= x(t)^T Q_+ x(t) + q^T x(t) - 2\bar{t}^T C x(t) + 2\left(\bar{t} - t\right)^T C x(t) \\
&\geq x(\bar{t})^T Q_+ x(\bar{t}) + q^T x(\bar{t}) - 2\bar{t}^T C x(\bar{t}) + 2\left(\bar{t} - t\right)^T C x(t) \\
&= \hat{\mu}(\bar{t}) + 2\left(\bar{t} - t\right)^T C x(\bar{t}) \\
&\geq \hat{\mu}(\bar{t}) - 2\left\|\bar{t} - t\right\| \left\| C x(\bar{t}) \right\| \\
&\geq \hat{\mu}(\bar{t}) - 2L \left\| \bar{t} - t \right\|.
\end{aligned}$$

Combining the above two inequalities, we obtain

$$\left| \hat{\mu}(\bar{t}) - \hat{\mu}(t) \right| \leq 2L \left\| \bar{t} - t \right\|.$$

This establishes the continuity of $\hat{\mu}(t)$ for $t \in [t_l, t_u]$. This completes the proof. □

We now describe the NADM for problem (1) based on the bi-convex reformulation (9).

Algorithm 2.1 [*New alternative direction method NADM*(t^0, ϵ)]

Input: Initial parameter $t^0 \in [t_l, t_u]$ and stopping criterion $\epsilon > 0$;
Step 0 Set $k = 0$;
Step 1 Solve problem (10) with $t = t^k$ for optimal solution $x(t^k)$. Set $x^{k+1} = x(t^k)$, and $t^{k+1} = Cx^{k+1}$;
Step 2 If $\|t^{k+1} - t^k\| > \sqrt{\epsilon}$, then set $k = k + 1$ and go back to step 1; Otherwise stop, and output (x^{k+1}, t^{k+1}) as the final solution.

Note that since $\mathcal{F}$ is bounded, there exists at least one accumulation point for the sequence $\{x^k\}$ generated by the NADM. Our next result characterizes the monotonicity of the sequence $\{t^k\}$ generated by the NADM.

Lemma 2.1 *Let the sequence* $\{t^k\}$ *be generated by Algorithm* 2.1. *Then the sequence* $\{t^k\}$ *is monotone in the sense that*

$$\left(t^k - t^{k-1}\right)^T \left(t^{k+1} - t^k\right) \geq 0.$$

Proof To show the monotonicity of the sequence $\{t^k\}$, recall Step 1 in the algorithm, we have

$$t^{k+1} = Cx^{k+1} = Cx\left(t^k\right) \quad \Rightarrow \quad t^{k+1} - t^k = C\left(x\left(t^k\right) - x\left(t^{k-1}\right)\right).$$

It follows from Conclusion (iii) of Proposition 2.2 that

$$\left(t^{k+1} - t^k\right)^T \left(t^k - t^{k-1}\right) \geq 0,$$

which implies the monotonicity of the sequence $\{t^k\}$. This completes the proof. □

We observe that at the kth iteration of the NADM, from Step 2, we obtain x^{k+1} by solving problem (10) with $t = Cx^k$, i.e.,

$$\begin{aligned} x^{k+1} &= \arg\min_{x\in\mathcal{F}} \left\{ x^T Q_+ x + q^T x - 2\left(Cx^k\right)^T Cx + \left\|Cx^k\right\|^2 \right\} \\ &= \arg\min_{x\in\mathcal{F}} \left\{ x^T Q_+ x + q^T x - \left\|Cx^k\right\|^2 - 2\left(C^T Cx^k\right)^T \left(x - x^k\right) \right\}. \end{aligned}$$

It should be pointed out that the same procedure has been used in the DCA for solving problem

$$\min_{x\in\mathcal{F}} x^T Q_+ x + q^T x - \|Cx\|^2 . \tag{16}$$

In other words, the NADM can be viewed as a specific DCA for solving problem (16). Therefore, the following convergence result of the NADM is an immediate consequence of the DCA's convergence theorem for the general D.C. programming (see [22,35]).

Proposition 2.3 *Let $\{x^k\}$ be an infinite sequence generated by Algorithm 2.1 with $\epsilon = 0$. We have*

(i) $f(x^{k-1}) - f(x^k) \geq \|t^{k-1} - t^k\|^2$ *for all* k.
(ii) *The sequence* $\left\{f(x^k)\right\}$ *converges and* $\lim_{k\to\infty} \|t^{k-1} - t^k\| = 0$.
(iii) *Any accumulation point of* $\{x^k\}$ *is a KKT point of problem* (1).

We next explore some new properties at the accumulation point of a sequence generated from NADM. We first introduce the so-called quasi-ϵ-local minimizer as follows.

Definition 2.1 $x^* \in \mathcal{F}$ is called a quasi-ϵ-local solution of problem (1) with respect to a neighborhood

$$\mathcal{N}\left(x^*, \epsilon\right) = \left\{x \in \mathbb{R}^n : \left\|x - x^*\right\| \leq \sqrt{\epsilon}\right\} \tag{17}$$

if for every $x \in \mathcal{N}(x^*, \epsilon) \cap \mathcal{F}$, the following relation holds:

$$f\left(x^*\right) \leq f(x) + \mathcal{O}(\epsilon), \tag{18}$$

where $\lim_{\epsilon\to 0} \frac{\mathcal{O}(\epsilon)}{\epsilon} = \beta$ for some $\beta > 0$.

We remark that the above definition can be viewed as an extension of the so-called ϵ-quasi-solution introduced in [25]. We have

Theorem 2.2 *Suppose that $x(t)$ is the optimal solution to problem (10) for some $t \in [t_l, t_u]$. If $t = Cx(t)$, then $x(t)$ is a quasi-ϵ-local solution of problem (1) with respect to $\mathcal{N}(x(t), \epsilon)$.*

Proof Suppose that $t = Cx(t)$. Since $x(t)$ is the optimal solution to problem (10), we have

$$f(x) + \|Cx(t) - Cx\|^2 \geq f(x(t)) + \|Cx(t) - t\|^2 = f(x(t)), \quad \forall x \in \mathcal{F}. \tag{19}$$

For a given $\epsilon > 0$, let us choose a neighborhood $\mathcal{N}(x(t), \epsilon)$ defined by (17). It follows that

$$\|Cx - Cx(t)\|^2 \leq \|C\|^2 \|x - x(t)\|^2 \leq \epsilon \|C\|^2, \quad \forall x \in \mathcal{N}(x(t), \epsilon),$$

which, together with (19), yields

$$f(x(t)) \leq f(x) + \mathcal{O}(\epsilon), \quad \forall x \in \mathcal{F} \cap \mathcal{N}(x(t), \epsilon),$$

where $\lim_{\epsilon \to 0} \frac{\mathcal{O}(\epsilon)}{\epsilon} = \|C\|^2$. This prove that $x(t)$ is a quasi-ϵ-local solution of problem (1) with respect to the neighborhood $\mathcal{N}(x(t), \epsilon)$. □

Based on Theorem 2.2, we can obtain the following convergence result of Algorithm 2.1 to a quasi-ϵ-local minimizer of problem (1).

Theorem 2.3 *Let $\epsilon = 0$ in Algorithm 2.1. We have*

(i) *If the algorithm stops at the kth iteration, then x^{k+1} is a quasi-ϵ-local solution of problem (1).*
(ii) *If the algorithm generates an infinite sequence $\{x^k\}$, then any accumulation point of $\{x^k\}$ is a quasi-ϵ-local solution of problem (1).*

Proof (i) If the algorithm stops at the kth iteration in Step 2, then $t^{k+1} = t^k$. We see that x^{k+1} solves problem (10) with $t = t^k = Cx^{k+1}$ and $t^k = Cx(t^k)$. By Theorem 2.2, x^{k+1} is a quasi-ϵ-local solution of problem (1).

(ii) If the algorithm generates an infinite sequence $\{(x^k, t^k)\}$, then by Proposition 2.3 (ii), we get

$$\lim_{k \to \infty} \left\| t^{k-1} - t^k \right\| = 0. \tag{20}$$

Now let $(\bar{x}, \bar{t})$ be an accumulation point of $\{(x^k, t^k)\}$. Then there exists a subsequence $\{(x^{k_j}, t^{k_j})\}$ such that $x^{k_j} \to \bar{x}$ and $t^{k_j} \to \bar{t}$ as $j \to \infty$. By (20), we obtain $t^{k_j - 1} \to \bar{t}$ as $j \to \infty$. Since $t^k = Cx^k$ for all k, we have $\bar{t} = C\bar{x}$. The closedness of $\mathcal{F}$ implies $\bar{x} \in \mathcal{F}$. Since x^{k_j} is an optimal solution of problem (10) with $t = t^{k_j - 1}$, we have

$$\begin{aligned} &x^T Q_+ x + q^T x - 2\left(t^{k_j-1}\right)^T Cx + \left\| t^{k_j-1} \right\|^2 \\ &\geq \left(x^{k_j}\right)^T Q_+ x^{k_j} + q^T x^{k_j} - 2\left(t^{k_j-1}\right)^T Cx^{k_j} + \left\| t^{k_j-1} \right\|^2, \quad \forall x \in \mathcal{F}. \end{aligned}$$

Then, by taking the limit on both sides of the inequality as $j \to \infty$, we obtain

$$x^T Q_+ x + q^T x - 2\bar{t}^T Cx + \|\bar{t}\|^2 \geq \bar{x}^T Q_+ \bar{x} + q^T \bar{x} - 2\bar{t}^T C\bar{x} + \|\bar{t}\|^2, \quad \forall x \in \mathcal{F}.$$

This implies that $\bar{x}$ is an optimal solution of problem (10) with $t = \bar{t} = C\bar{x}$. Therefore, by Theorem 2.2, we infer that $\bar{x}$ is a quasi-ϵ-local solution of problem (1). This completes the proof. □

Theorem 2.3 and Proposition 2.3 show that NADM converges to both a so-called quasi-ϵ-local minimizer and a KKT point of problem (1). Lemma 2.1 shows that the t-sequence generated from NADM converges in a monotonic manner. In Sect. 4.1, we further show that the t-sequence provided by NADM for QP1NE converges to a semi-local minimizer. As we shall see later, these new properties of the NADM facilitate the design of new global solvers for special subclasses of QCQPs. For example, the solution provided by NADM (or DCA) can be used as an upper bound in the new B&B approach for problem (1). By combining NADM with initialization technique and line search technique, we develop a new global search algorithm for QP1NE.

3 A new global optimization algorithm

In this section, we develop a new global algorithm that can find a globally optimal solution to problem (1) within a pre-specified ϵ-tolerance by combining the NADM algorithm, branch-and-bound (B&B) framework and convex relaxation techniques. We establish the convergence of the algorithm and estimate its complexity.

3.1 The quadratic convex relaxation

We start by considering a restricted version of the lifted problem (3) where the variable t is in a sub-rectangle $[l, u]$ with $l, u \in \mathbb{R}^r$. That is,

$$\begin{aligned} \min \quad & f(x,t) = x^T Q_+ x + q^T x - \|t\|^2, \\ \text{s. t.} \quad & Cx - t = 0, \\ & x \in \mathcal{F}, \quad t \in [l,u], \end{aligned} \tag{21}$$

where $l, u \in \mathbb{R}^r$ with $[l, u] \subseteq [t_l, t_u]$. Let $\lambda_i, i = 1, \ldots, r$ be the absolute values of the negative eigenvalues of Q, and $p_i, i = 1, \ldots, r$ be the corresponding orthogonal unit eigenvectors. From the construction of matrix C in (2), we see that the row vectors of C are $c_i^T = \sqrt{\lambda_i}\, p_i^T$, $i = 1, \ldots, r$. Then $t_i = c_i^T x = \sqrt{\lambda_i}\, p_i^T x$, $i = 1, \ldots, r$. Let p_i, $i = r+1, \ldots, n$ be the orthogonal unit eigenvectors corresponding to the non-negative eigenvalues of Q. Let $s_i = t_i^2$ for $i = 1, \ldots, r$. Since $\sum_{i=1}^n p_i p_i^T = I$, we have

$$\sum_{i=1}^{r} \frac{s_i}{\lambda_i} = \sum_{i=1}^{r} x^T \left(p_i p_i^T \right) x \le x^T \left(\sum_{i=1}^{n} p_i p_i^T \right) x = x^T x \le \bar{u}^T x, \quad \forall x \in \mathcal{F},$$

where $\bar{u} \in \mathbb{R}^n$ is the upper bound of variable x over $\mathcal{F}$. On the other hand, when $t \in [l, u]$, we have

$$s_i = t_i^2 \le (l_i + u_i)\, t_i - l_i u_i, \quad i = 1, \ldots, r.$$

It is worth mentioning that $(l_i + u_i)t_i - l_i u_i$ is the concave envelope of t_i^2 on $[l_i, u_i]$, and the above inequality is the so-called secant cut introduced in [38]. Therefore, we can derive the following convex relaxation for problem (21):

$$\min x^T Q_+ x + q^T x - \sum_{i=1}^{r} s_i, \tag{22}$$

$$\begin{aligned} \text{s. t.} \quad & Cx - t = 0, \quad x \in \mathcal{F}, \quad t \in [l, u], \\ & t_i^2 \le s_i, \quad i = 1, \ldots, r, \\ & s_i \le (l_i + u_i) t_i - l_i u_i, \quad i = 1, \ldots, r, \\ & \sum_{i=1}^{r} \frac{s_i}{\lambda_i} \le \bar{u}^T x. \end{aligned} \tag{23}$$

We remark that it is easy to show that if we remove the last constraint (23) in problem (22), then we obtain the following simple relaxation:

$$\begin{aligned} & \min\ x^T Q_+ x + \left[q - C^T (l + u)\right]^T x + l^T u, \\ & \text{s. t. } x \in \mathcal{F}, \quad l \le Cx \le u. \end{aligned} \tag{24}$$

The following example indicates that problem (22) is strictly tighter than (24).

Example 3.1 Consider a nonconvex quadratic programming problem:

$$\begin{aligned} \min \quad & 0.5(25x_1 - 7x_2 + 8x_3)^2 + 23x_1 + 37x_2 + 12x_3 \\ & - (2x_1 + 6x_2 - x_3)^2 - (x_1 - x_2 - 4x_3)^2, \\ s.t. \quad & 28x_1^2 + 28x_2^2 + 10x_3^2 + 2x_1x_3 + x_1 + 5x_2 \le 16, \\ & -5x_1 + 3x_2 + 4x_3 \le 5, \\ & 0 \le x_1, x_2, x_3 \le 1. \end{aligned} \tag{25}$$

For the above example,the global optimal solution is $x^* = (0, 0, 0)^T$ with an objective function value $f(x^*) = 0$. Note that we have $C = (c_1, c_2)^T$ with $c_1 = (2, 6, -1)^T$ and $c_2 = (1, -1, -4)^T$, and $\lambda_1 = \lambda_2 = 1$. By solving problems in (4), we obtain $t_l = (-1, -4.6)^T$ and $t_u = (8, 1)^T$. By solving the relaxation model (22) with $[l = t_l, u = t_u]$, we obtain a lower bound -8.3437, which is tighter than the bound -16.0277 provided by the simple relaxation (24).

As pointed out in the introduction, there exist many other strong relaxation models for QCQP that usually involve intensive computation. In this work, we combine the relaxation model (22) with other simple optimization techniques to develop a global

algorithm for QCQP. Our choice is based on the relative simplicity of the relaxation model (22) and its good approximation behavior as shown in our next theorem, which compares the objective values at the optimal solutions to problem (21) and its relaxation (22). We have

Theorem 3.1 *Let $f^*_{[l,u]}$ and $v^*_{[l,u]}$ be the optimal value of problem (21) and its relaxation (22), respectively. Let $(\bar{x}, \bar{s}, \bar{t})$ be the optimal solution to problem (22). Then we have*

$$0 \le f^*_{[l,u]} - v^*_{[l,u]} \le f(\bar{x}, \bar{t}) - v^*_{[l,u]} = \sum_{i=1}^{r} \left(\bar{s}_i - \bar{t}_i^2\right) \le \frac{1}{4}\|u - l\|^2.$$

Proof Clearly, $(\bar{x}, \bar{t})$ is a feasible solution to problem (21). Since problem (22) is a convex relaxation of problem (21), it follows from the choice of $f^*_{[l,u]}$ that

$$\begin{aligned} 0 \le f^*_{[l,u]} - v^*_{[l,u]} &\le f(\bar{x}, \bar{t}) - v^*_{[l,u]} = \sum_{i=1}^{r} \left(\bar{s}_i - \bar{t}_i^2\right) \\ &\le \sum_{i=1}^{r} \left[-\bar{t}_i^2 + (l_i + u_i)\bar{t}_i - l_i u_i\right] \\ &\le \frac{1}{4}\sum_{i=1}^{r}(u_i - l_i)^2 = \frac{1}{4}\|u - l\|^2, \end{aligned}$$

where the third inequality follows from the constraints on s_i in (22). This completes the proof. □

From Theorem 3.1, we have

Proposition 3.1 *Let $(\bar{x}, \bar{s}, \bar{t})$ be the optimal solution to problem (22) and $\epsilon > 0$. If $\sum_{i=1}^{r}[\bar{s}_i - \bar{t}_i^2] \le \epsilon$, then $(\bar{x}, \bar{t})$ is an ϵ-optimal solution of problem (21).*

Proof Note that $(\bar{x}, \bar{t})$ is a feasible solution to problem (21). By Theorem 3.1, we obtain

$$0 \le f(\bar{x}, \bar{t}) - f^*_{[l,u]} \le f(\bar{x}, \bar{t}) - v^*_{[l,u]} = \sum_{i=1}^{r} \left(\bar{s}_i - \bar{t}_i^2\right) \le \epsilon.$$

Thus, $f(\bar{x}, \bar{t}) \le f^*_{[l,u]} + \epsilon$ and so $(\bar{x}, \bar{t})$ is an ϵ-optimal solution of problem (21). This completes the proof. □

Theorem 3.1 indicates that when the diameter of the rectangle $[l, u]$ is very small, the relaxed model (22) can provide a very good approximation to the original problem (21). Moreover, from Theorem 3.1 and Proposition 3.1, we see that if

$$u_i - l_i \le 2\sqrt{\epsilon/r}, \quad i = 1, \ldots, r, \tag{26}$$

then $(\bar{x}, \bar{t})$ can be viewed as an ϵ-approximation solution to problem (21). Based on this observation, we can divide each interval $[t_l^i, t_u^i]$ into $\lceil \frac{\sqrt{r}(t_u^i - t_l^i)}{2\sqrt{\epsilon}} \rceil$ subintervals $[l_i, u_i]$, where t_l^i, t_u^i are defined by (4). Correspondingly, we can divide the feasible region $\mathcal{F}$ into many subregions where each subregion is bounded with box constraints like $t_i \in [l_i, u_i]$, $u_i - l_i \leq 2\sqrt{\epsilon/r}$. We can solve the relaxed problem (22) in all the subregions to obtain an ϵ-approximate solution in every subregion. Among all these ϵ-approximate solutions, we choose the one with the minimal objective function value as the final solution, which is clearly a global ϵ-approximate solution to the original problem (1). From the above discussion, we can see that such partitioning procedure can provide an ϵ-approximation to problem (1). For convenience, we call such a procedure the brutal force algorithm. The following result is an immediate consequence of our above discussion.

Theorem 3.2 *The brutal force partitioning algorithm can find a global ϵ-approximate solution to problem* (1) *in $\mathcal{O}(\prod_{i=1}^{r} \lceil \frac{\sqrt{r}(t_u^i - t_l^i)}{2\sqrt{\epsilon}} \rceil N)$ time where N is the complexity for solving problem* (22).

It is worth comparing the complexity result in the above theorem with the complexity result $\mathcal{O}(\lceil \frac{n(n+1)}{\sqrt{\epsilon}} \rceil^r N)$ in [43], where N denotes the complexity for solving a convex QP of the same size as (1). One can easily see that the key difference is that we have replaced the quantity $n(n+1)$ by $t_u^i - t_l^i$. Such an improvement is significant when $t_u^i - t_l^i \ll n(n+1)$. On the other hand, we also would like to mention that in [43], the so-called ϵ-approximate solution $\bar{x}$ is defined as follows: $f(\bar{x}) - f(x^*) \leq \epsilon[f(\tilde{x}) - f(x^*)]$ for some $\tilde{x} \in \mathcal{F}$, where x^* is the optimal solution of the problem, while we use the standard definition $f(\bar{x}) \leq f(x^*) + \epsilon$.

We remark that the brutal force algorithm is very conservative and not effective since it works with very small neighborhoods. In next subsection, we shall discuss how to integrate the NADM algorithm with other techniques to develop an effective global algorithm for problem (1).

3.2 The ADMBB algorithm

In this subsection, we introduce a new global algorithm that integrates the NADM with branch-and-bound (B&B) techniques based on the above convex relaxation. For convenience, we call it ADMBB. It is worthwhile mentioning that, different from other existing global algorithms for QCQP, the ADMBB uses NADM to compute upper bounds for the underlying QP, and combine the computed bounds and convex relaxation to determine whether the obtained solution is globally optimal.

Now, we describe the new branching technique based on partitioning the feasible rectangle (denoted by $\mathcal{B}$) of t in problem (22). To obtain the strong lower bound from relaxation (22) over sub-rectangles, we present an adaptive branch-and-cut rule by making use of the special structure of problem (22). Consider the sub-rectangle $\mathcal{B} = [l, u]$, and let (x^*, s^*, t^*) be the optimal solution to problem (22) over $\mathcal{B}$. If $\sum_{i=1}^{r} [s_i^* - (t_i^*)^2] \leq \epsilon$, then by Proposition 3.1, x^* is an ϵ-optimal solution to problem (21) over $\mathcal{B}$ and so this sub-rectangle will not be further partitioned. Otherwise, there

exists at least an $i \in \{1, \ldots, r\}$ such that $s_i^* - (t_i^*)^2 > \epsilon/r$. Choose i^* that maximizes $s_i^* - (t_i^*)^2$. Partition $\mathcal{B}$ into two sub-rectangles $\mathcal{B}_1$ and $\mathcal{B}_2$ along the edge $[l_{i^*}, u_{i^*}]$ at point w, where $w \in (l_{i^*}, u_{i^*})$ is called the branching point. We consider the constraints over intervals $[l_{i^*}, w]$ and $[w, u_{i^*}]$, respectively,

$$s_{i^*} \leq (l_{i^*} + w)\, t_{i^*} - l_{i^*} w, \quad s_{i^*} \leq (w + u_{i^*})\, t_{i^*} - w u_{i^*}. \tag{27}$$

From Fig. 1, it is easy to see that (x^*, s^*, t^*) is cut off by the two constraints in (27) when $w = t_{i^*}^*$. A natural question to ask is how to choose an appropriate branching point w such that the relaxation (22) over $\mathcal{B}_i$ $(i = 1, 2)$ can provide a tighter lower bound. Let l_1 and l_2 denote the straight lines $s_{i^*} = (l_{i^*} + w)t_{i^*} - l_{i^*} w$ and $s_{i^*} = (w + u_{i^*})t_{i^*} - w u_{i^*}$, respectively. The area between the parabola $s_{i^*} = t_{i^*}^2$ and straight lines l_1 and l_2 over $[l_{i^*}, u_{i^*}]$ is then given by:

$$\begin{aligned} S(w) &= \int_{l_{i^*}}^{w} \left[(l_{i^*} + w)\, t - l_{i^*} w - t^2 \right] dt + \int_{w}^{u_{i^*}} \left[(w + u_{i^*})\, t - w u_{i^*} - t^2 \right] dt \\ &= \frac{1}{2} (u_{i^*} - l_{i^*})\, w^2 - \frac{1}{2} \left(u_{i^*}^2 - l_{i^*}^2 \right) w + \frac{1}{6} \left(u_{i^*}^3 - l_{i^*}^3 \right) \\ &= \frac{1}{2} (u_{i^*} - l_{i^*}) \left(w - w^* \right)^2 + \frac{1}{24} (u_{i^*} - l_{i^*})^3, \end{aligned}$$

where $w^* = \frac{l_{i^*} + u_{i^*}}{2}$. Since $S(w)$ attains its minimum at w^*, a popular choice is to use w^* as a branching point to partition $\mathcal{B}$ along the edge $[l_{i^*}, u_{i^*}]$. However, it is possible that the constraints (27) with $w = w^*$ can not cut off the solution (x^*, s^*, t^*) to problem (22) over $\mathcal{B}$. In this case, the lower bound from problem (22) over either $\mathcal{B}_1$ or $\mathcal{B}_2$ will not be improved substantially. As an alternative, we may choose $w = t_{i^*}^*$ as the branching point. In other words, if the solution (x^*, s^*, t^*) is cut off by constraints (27) with $w = w^*$, then partition the sub-rectangle $\mathcal{B}$ along the edge $[l_{i^*}, u_{i^*}]$ at the middle point $w^* = \frac{l_{i^*} + u_{i^*}}{2}$. Otherwise, partition the sub-rectangle $\mathcal{B}$ along the edge $[l_{i^*}, u_{i^*}]$ at point $w^* = t_{i^*}^*$.

We are now ready to describe the new global algorithm for problem (1).

Algorithm 3.1 (*The ADMBB algorithm*)

Input: $Q, q, Q_i, q_i, d_i (i = 1, \ldots, m)$, A, b, initial bound t_l,t_u and stop criteria $\epsilon > 0$.

Output: A global ϵ-solution x^*.

Step 0 (**Initialization**) Let r be the number of negative eigenvalues of Q. If $r \leq 5$, set $\rho = 2^r$, $\mu_j \in \{-1, 1\}^r$, $j \in \{1, \ldots, \rho\}$; Else set $\rho = 2$, $\mu_1 = e$, $\mu_2 = -e$, where $e \in \mathbb{R}^r$ is the vector of ones. For $j \in \{1, \ldots, \rho\}$, solve problem $\min\limits_{(x,t) \in \mathcal{F}_t} \mu_j^T t$ to get the optimal solution t_j, respectively.

Step 1 Find quasi-ϵ-local solutions x_j^k of problem (1) by running NADM(t^0, ϵ) with $t^0 = t_j$ for $j = 1, \ldots, \rho$. Set $x^* := \arg\min\{f(x_j^k),\ j = 1, \ldots, \rho\}$ and $v^* = f(x^*)$.

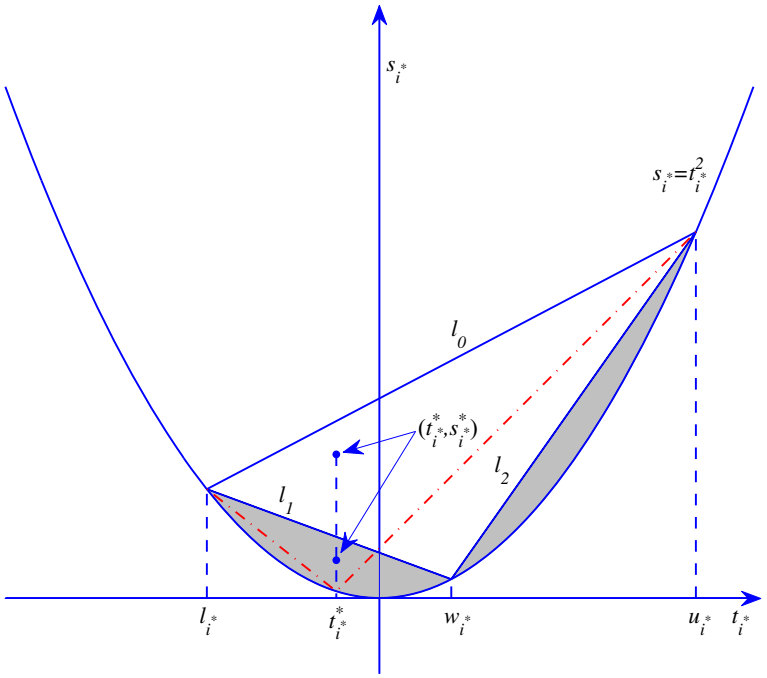

Fig. 1 Partition on the variable t_{i^*}. The straight lines $l_0 : s_{i^*} = (l_{i^*} + u_{i^*})t_{i^*} - l_{i^*}u_{i^*}$, $l_1 : s_{i^*} = (l_{i^*} + w_{i^*})t_{i^*} - l_{i^*}w_{i^*}$, and $l_2 : s_{i^*} = (w_{i^*} + u_{i^*})t_{i^*} - w_{i^*}u_{i^*}$, where $w_{i^*} = \frac{l_{i^*}+u_{i^*}}{2}$. The shaded region is made up of the parabola $s_{i^*} = t_{i^*}^2$ and two straight lines l_1 and l_2. If point $(t_{i^*}^*, s_{i^*}^*)$ is in the triangle region, then it is cut off by constraints (27) with $w = w_{i^*}$. If point $(t_{i^*}^*, s_{i^*}^*)$ is in the shaded region, then it is cut off by constraints (27) with $w = t_{i^*}^*$

Step 2 Solve problem (22) with $[l = t_l, u = t_u]$ to obtain the optimal solution (x^0, s^0, t^0) and lower bound v^0. If $f(x^0) < v^*$, then update upper bound $v^* = f(x^0)$ and solution $x^* = x^0$.
Set k:=0, $l^k = t_l$, $u^k = t_u$, $\mathcal{B}^k$:=$[l^k, u^k]$, Ω:= $\left\{[\mathcal{B}^k, v^k, (x^k, s^k, t^k)]\right\}$.

Step 3 **While** $\Omega \neq \emptyset$ **Do** (the main loop)

(S3.1) (**Node Selection**) Choose the node $[\mathcal{B}^k, v^k, (x^k, s^k, t^k)]$ from Ω with the smallest v^k and remove it from Ω.

(S3.2) (**Termination**) If either $v^k \geq v^* - \epsilon$ or $f(x^k) - v^k \leq \epsilon$, then both x^* and x^k are an ϵ-optimal solution to problem (1), stop.

(S3.3) (**Partition**) Choose i^* maximizing $s_i^k - (t_i^k)^2$ for $i = 1, \ldots, r$. Set $w_{i^*} = \frac{l_{i^*}^k + u_{i^*}^k}{2}$ and

$$\Gamma_k(w_{i^*}) = \left\{ (s_{i^*}, t_{i^*}) \,\middle|\, \begin{array}{l} s_{i^*} > \left(l_{i^*}^k + w_{i^*}\right) t_{i^*} - l_{i^*}^k w_{i^*} \\ s_{i^*} > \left(w_{i^*} + u_{i^*}^k\right) t_{i^*} - w_{i^*} u_{i^*}^k \end{array} \right\}.$$

If $(s_{i^*}^k, t_{i^*}^k) \in \Gamma_k(w_{i^*})$, then set the branching point $\beta_{i^*} = w_{i^*}$; else set $\beta_{i^*} = t_{i^*}^k$. Partition $\mathcal{B}^k$ into two sub-rectangles $\mathcal{B}^{k1}$ and $\mathcal{B}^{k2}$ along the edge $[l_{i^*}^k, u_{i^*}^k]$ at point β_{i^*}.

(S3.4) Solve problem (22) over $\mathcal{B}^{k_j}$ to obtain a lower bound v^{k_j} and an optimal solution $(x^{k_j}, s^{k_j}, t^{k_j})$ for $j = 1, 2$. Set $\Omega = \Omega \cup \{[\mathcal{B}^{k_1}, v^{k_1}, (x^{k_1}, s^{k_1}, t^{k_1})], [\mathcal{B}^{k_2}, v^{k_2}, (x^{k_2}, s^{k_2}, t^{k_2})]\}$.

1. **(S3.5)** (**Restart NADM**) Set $\hat{x} = \arg\min\{f(x^{k_1}), f(x^{k_2})\}$. If $f(\hat{x}) < v^*$, then find a quasi-ϵ-local solution $\bar{x}^k$ of problem (1) by running NADM(t^0, ϵ) with $t^0 = C\hat{x}$, update solution $x^* = \arg\min\{f(\hat{x}), f(\bar{x}^k)\}$ and upper bound $v^* = f(x^*)$.

(S3.6) (**Node deletion**) Delete from Ω all the nodes $[\mathcal{B}^j, v^j, (x^j, s^j, t^j)]$ with $v^j \geq v^* - \epsilon$. Set $k = k + 1$.

2. **End while**

We list a few key differences between the ADMBB and other existing global algorithms for QCQP in the literatures [4,5].

(i) In Step 1, we run NADM from different starting point to find a quasi-ϵ-local solution of problem (1).
(ii) In Step (S3.1), we restart NADM to find a better quasi-ϵ-local solution if the optimal objective function value at the relaxation problem is less than the current upper bound.
(iii) In Step (S3.3), the optimal solution of the relaxation problem corresponding to the node with the smallest lower bound is cut off after each iteration so that the lower bound will be improved.

We next present several technical results regarding the sequences $\{s^k\}$ and $\{t^k\}$ generated by Algorithm 3.1. Recall that (x^k, s^k, t^k) is the optimal solution of problem (22) with $[l = l^k, u = u^k]$, from the proof of Theorem 3.1, we immediately have

Lemma 3.1 $s_i^k - (t_i^k)^2 \leq \frac{1}{4}(u_i^k - l_i^k)^2$, $i = 1, \ldots, r$ *for all* k.

Lemma 3.2 *At the* k*th iteration, if* $\max\limits_{i=1,\ldots,r} \{s_i^k - (t_i^k)^2\} \leq \epsilon/r$, *then Algorithm 3.1 stops and both* x^* *and* x^k *are a global* ϵ*-approximate solution to problem* (1).

Proof Note that since (x^k, s^k, t^k) is the optimal solution of problem (22) with $[l = l^k, u = u^k]$ and v^k is its optimal value, by Theorem 3.1, we follow that $f(x^k) - v^k \leq \epsilon$. From Steps 2 and (S3.5) of the algorithm, we see that $f(x^k) \geq v^* = f(x^*)$ for all k. Thus $v^k - v^* \geq v^k - f(x^k) \geq -\epsilon$. This means that two stopping criteria are met, so the algorithm stops. By Step (S3.1), v^k is the smallest lower bound. Thus $f^* \geq v^k$, where f^* denotes the optimal value of problem (1). Note that both x^* and x^k are feasible solutions to problem (1). Therefore, we follow that both $f(x^k) \geq f^* \geq v^k \geq f(x^k) - \epsilon$ and

$$f\left(x^*\right) \geq f^* \geq v^k \geq v^* - \epsilon = f\left(x^*\right) - \epsilon,$$

which imply that both x^k and x^* are a global ϵ-solution to problem (1). This finishes the proof. □

We next establish the convergence of Algorithm 3.1. From Lemmas 3.2 and 3.1, we see that if, at the kth iteration, the chosen node $[\mathcal{B}^k, v^k, (x^k, s^k, t^k)]$ with the smallest

lower bound v^k satisfies either $u_{i^*}^k - l_{i^*}^k \leq 2\sqrt{\epsilon/r}$ for the chosen i^* in partition or $u_i^k - l_i^k \leq 2\sqrt{\epsilon/r}$ for $i = 1, \ldots, r$, then the algorithm stops and yields a global ϵ-approximate solution x^* to problem (1). At the kth iteration, if the algorithm does not stop, then from Lemma 3.2 we have $s_{i^*}^k - (t_{i^*}^k)^2 > \epsilon/r$ for the selected index i^* in Step (S3.3). It follows from Lemma 3.1 that $u_{i^*}^k - l_{i^*}^k > 2\sqrt{\epsilon/r}$, i.e., the i^*th edge of the sub-rectangle $\mathcal{B}^k$ must be longer than $2\sqrt{\epsilon/r}$. According to Step (S3.3), it will be divided at either point $t_{i^*}^k$ or the midpoint $(u_{i^*}^k + l_{i^*}^k)/2$. Note that if $u_i^k - l_i^k \leq 2\epsilon/\sqrt{r}$, then the ith direction will never be chosen in Step (S3.3) as a branching direction at the k-iteration. This implies that all the edges of the sub-rectangle corresponding to a node selected by the least lower bound rule at every iteration will never be shorter than $2\sqrt{\epsilon/r}$. Therefore, every edge of the initial rectangle will be divided into at most $\left\lceil \frac{\sqrt{r}(t_u^i - t_l^i)}{2\sqrt{\epsilon}} \right\rceil$ sub-intervals. In other words, to obtain an ϵ-optimal solution to problem (1), the total number of the relaxed subproblem (22) required to be solved in all the runs of Algorithm 3.1 is bounded above by

$$\prod_{i=1}^{r} \left\lceil \frac{\sqrt{r}\left(t_u^i - t_l^i\right)}{2\sqrt{\epsilon}} \right\rceil.$$

From the above discussion, we immediately obtain the following result.

Theorem 3.3 *The ADMBB Algorithm can find a global ϵ-approximate solution to problem (1) by solving at most $\prod_{i=1}^{r} \lceil \frac{\sqrt{r}(t_u^i - t_l^i)}{2\sqrt{\epsilon}} \rceil$ relaxed subproblems (22).*

4 A new global search algorithm for QP1NE

In this section, we develop a new global search algorithm for QP1NE that can find a globally optimal solution to QP1NE within a pre-specified ϵ-tolerance by integrating the NADM, convex relaxation and line search technique. The section consists of four subsections. In Sect. 4.1, we estimate the complexity of the NADM for QP1NE and show that the generated t-sequence converges to a semi-local minimizer, a notion to be introduced later. In Sect. 4.2, we present two restricted variants of the NADM approach that will be used in the proposed global algorithm. In Sect. 4.3, we describe a line search procedure based on convex relaxation to reduce the subinterval that may contain a global minimizer of QP1NE. In Sect. 4.4, we first describe a global algorithm for QP1NE that combines the NADM algorithm with a line search procedure. Then we establish the convergence of the proposed algorithm and estimate its complexity.

4.1 Complexity and convergence of NADM for QP1NE

In this subsection, we present the new convergence result of the NADM algorithm for QP1NE and estimate its complexity. Particularly, we show that the t-sequence generated from the NADM algorithm converges in a monotonic manner to a semi-local minimizer of the following problem

$$\min_{t\in[t_l,t_u]} g(t) = f(x(t)), \tag{28}$$

where $x(t)$ is an optimal solution to problem (29) below. It is easy to verify that problem (3) with $r = 1$ and problem (28) are equivalent.

To start, we recall that for QP1NE, it holds $r = 1$. In such a case, problem (10) can be rewritten as

$$\min_{x\in\mathcal{F}} x^T Q_+ x + q^T x - 2tc_1^T x + t^2, \tag{29}$$

where $t \in [t_l, t_u]$ is a given parameter with $t_l = \min_{x\in\mathcal{F}} c_1^T x$ and $t_u = \max_{x\in\mathcal{F}} c_1^T x$. By Proposition 2.2, we immediately have

Corollary 4.1 *Let $x(t)$ be the optimal solution of problem (29) and $\mu(t)$ denote its optimal value. We have*

(i) *If $x \in \mathcal{F}$ satisfies the following inequality*

$$\left|c_1^T x - t\right| \le \left|c_1^T x(t) - t\right|,$$

then it must hold

$$f(x) \ge f(x(t)).$$

(ii) *Let x^* be the global optimal solution of QP1NE. Then*

$$\left|c_1^T x(t) - t\right| \le \left|c_1^T x^* - t\right|.$$

(iii) *The function $c_1^T x(t)$ is increasing with respect to t.*
(iv) *The function $\mu(t)$ is continuous at $t \in [t_l, t_u]$.*

Now let us consider the sequences $\{x^k\}$ and $\{t^k\}$ generated by NADM for QP1NE. We have

Corollary 4.2 *Let the two sequences $\{x^k\}$ and $\{t^k\}$ be generated by NADM(t^0, ϵ). Let x^* be the global solution of QP1NE and $t^* = c_1^T x^*$. We have*

(i) *If $t^0 = t_l^1$, then the sequence $\{t^k\}$ is increasing and bounded above by t^*;*
If $t^0 = t_u^1$, then the sequence $\{t^k\}$ is decreasing and bounded below by t^*.
(ii) *$f(x^{k-1}) - f(x^k) \ge |t^k - t^{k-1}|^2$ for all k.*

Proof Statement (ii) follows directly from Statement (i) of Proposition 2.3. Now we prove Statement (i). We first consider the case when $t^0 = t_l$. By Lemma 2.1, the sequence $\{t^k\}$ is monotone, i.e.,

$$\left(t^k - t^{k-1}\right)\left(t^{k+1} - t^k\right) \ge 0, \quad \forall k \ge 1. \tag{30}$$

Recall from Step 1 that $t^1 = c_1^T x^1 \geq t_l = t^0$, which, by (30), implies $t^1 \leq t^2$. Let us assume that $t^{i-1} \leq t^i$ for some $i \geq 2$. Thus from (30) we can infer that $t^i \leq t^{i+1}$. This proves that $\{t^k\}$ is an increasing sequence. On the other hand, we see from (4) that $t^0 \leq t^* = c_1^T x^*$. From Statement (ii) of Corollary 4.1, we derive $c_1^T x^1 - t^0 \leq c_1^T x^* - t^0$ and thus $t^1 = c_1^T x^1 \leq t^* = c_1^T x^*$. Using Statement (ii) of Corollary 4.1 again, we obtain $c_1^T x^2 - t_1 \leq c_1^T x^* - t_1$, i.e., $t^2 \leq c_1^T x^* = t^*$. Let us assume that $t^i = c_1^T x^i \leq c_1^T x^* = t^*$ for some $i \geq 2$. Thus from Statement (ii) of Corollary 4.1 we can conclude $t^{i+1} \leq c_1^T x^* = t^*$. This proves that $\{t^k\}$ is bounded above by t^*. Thus Statement (i) holds when $t^0 = t_l$. The case $t^0 = t_u$ follows similarly. The proof is finished. □

We next estimate the complexity of the NADM algorithm for QP1NE. Denote $\{t^k : k = 0, 1, \ldots, K\}$ be the sequence generated by NADM. The monotonicity of the t-sequence implies that $|t^K - t^0| \leq t_u - t_l$. Recall Step 2 in NADM with $r = 1$, we have

$$\left| t^{k+1} - t^k \right| \geq \sqrt{\epsilon}, \quad k = 0, 1, \ldots, K-1.$$

It follows immediately

$$K \leq \left\lceil \frac{t_u - t_l}{\sqrt{\epsilon}} \right\rceil,$$

which provides a complexity bound for NADM.

Theorem 4.1 *The NADM algorithm for QP1NE terminates in at most* $\lceil \frac{t_u - t_l}{\sqrt{\epsilon}} \rceil$ *iterations.*

We now introduce a new definition that will be used in our analysis later.

Definition 4.1 For a univariate function $h(t)$, we say t^* is a lower semi-local minimizer if there exists a positive number $\gamma > 0$ such that

$$h\left(t^*\right) \leq h(t), \quad \forall t \in \left[t^* - \gamma, t^*\right],$$

and an upper semi-local minimizer if there exists a positive number $\gamma > 0$ such that

$$h\left(t^*\right) \leq h(t), \quad \forall t \in \left[t^*, t^* + \gamma\right].$$

Our next theorem shows that the sequence $\{t^k\}$ generated by NADM converges monotonically to a semi-local minimizer of problem (28). We have

Theorem 4.2 *Let* $\epsilon = 0$ *in the NADM algorithm. If* $\{t^k\}$ *is an infinite sequence obtained by running the NADM algorithm from a starting point* $t^0 = t_l$ *(or* $t^0 = t_u$*), then* $\{t^k\}$ *is an increasing (or decreasing) sequence converging to* t^+ *and* t^+ *is a lower (or upper) semi-local minimizer of problem* (28).

Proof We first consider the case when $t^0 = t_l$. By Statement (i) of Corollary 4.2, $\{t^k\}$ is an increasing sequence converging to t^+ and consequently

$$\lim_{k\to\infty} \left| t^{k+1} - t^k \right| = 0. \tag{31}$$

Note from (28) and $x^{k+1} = x(t^k)$ that $g(t^k) = f(x^{k+1})$. From Statement (ii) of Corollary 4.2, we can conclude that $\{g(t^k)\}$ is a decreasing sequence.

For every $t \in [t^{k-1}, t^k]$, let $x(t)$ be the optimal solution to problem (29). From Statement (iii) of Corollary 4.1, we have

$$t^k = c_1^T x\left(t^{k-1}\right) \le c_1^T x(t) \le c_1^T x\left(t^k\right) = t^{k+1},$$

which yields

$$\left| c_1^T x(t) - t^k \right| \le \left| c_1^T x(t^k) - t^k \right|.$$

Since $x(t^k)$ is the optimal solution to problem (29) with $t = t^k$, we follow from Statement (i) of Corollary 4.1 that

$$g(t) = f(x(t)) \ge f\left(x\left(t^k\right)\right) = g\left(t^k\right), \quad \forall t \in [t^{k-1}, t^k]. \tag{32}$$

Let $\mu(t)$ denote the optimal value to problem (29). Since $x(t^k) \in \mathcal{S}(t^k)$ and $t^{k+1} = c_1^T x(t^k)$, we can follow

$$\mu\left(t^k\right) = g\left(t^k\right) + \left(t^{k+1} - t^k\right)^2. \tag{33}$$

By Conclusion (iv) of Corollary 4.1, the function $\mu(t)$ is continuous at $t \in [t_l, t_u]$. Now let us recall the fact that $t^k \to t^+ \in [t_l, t_u]$. Therefore, it follows from (33) and (31) that

$$\lim_{k\to\infty} g\left(t^k\right) = \lim_{k\to\infty} \left[\mu\left(t^k\right) - \left(t^{k+1} - t^k\right)^2\right] = \mu\left(t^+\right) \ge g\left(t^+\right). \tag{34}$$

Since $\{g(t^k)\}$ is a decreasing sequence and by (34), we infer

$$g\left(t^k\right) \ge g\left(t^+\right), \quad \forall k. \tag{35}$$

For every $t \in [t^0, t^+)$, since $\{t^k\}$ is an increasing sequence converging to t^+, there exists an integer $k > 0$ such that $t \in [t^{k-1}, t^k]$. From (32) and (35), we obtain

$$g(t) \ge g\left(t^+\right), \quad \forall t \in [t^0, t^+).$$

This shows that t^+ is a lower semi-local minimizer of $g(t)$. The case when $t^0 = t_u$ follows similarly. The proof of the theorem is completed. □

4.2 Two restricted variants of NADM

Based on the results in the previous subsection, we can apply the NADM algorithm with two different starting points $t^0 = t_l$ and $t^0 = t_u$. Let us denote the two sequences by $\{t_l^k\}$ and $\{t_u^k\}$. Since both sequences are monotone and bounded, these two sequences either have a finite number of elements or have a sequence converging to its accumulation point. Let us denote the accumulation points of these two sequences by $\bar{t}_l$ and $\bar{t}_u$, respectively. Then, from Statement (i) of Corollary 4.2, we have

$$\bar{t}_l \le t^* \le \bar{t}_u.$$

This yields immediately the following result.

Corollary 4.3 *Suppose that we set $\epsilon = 0$ in the NADM algorithm and run it twice from two different starting points $t^0 = t_l$ and $t^0 = t_u$, and obtain the sequences denoted by $\{t_l^k\}$ and $\{t_u^k\}$, respectively. If these two sequences converge to the same point, i.e., $\bar{t}_l = \bar{t}_u$, then the global optimal solution to QP1NE can be attained at a point $c^T x^* = \bar{t}_l$.*

In general, it may happen that $\bar{t}_l < \bar{t}_u$. In such a case, we can conclude that the optimal solution to problem (28) can be found in the interval $[\bar{t}_l, \bar{t}_u]$. In such a case, instead of problem (29), in Step 1 of the NADM algorithm we may need to solve the following restricted convex optimization problem

$$\begin{aligned} &\min x^T Q_+ x + q^T x - 2t c_1^T x + t^2 \\ &\text{s. t.} \quad c_1^T x \ge t, \ x \in \mathcal{F} \end{aligned} \tag{36}$$

or

$$\begin{aligned} &\min x^T Q_+ x + q^T x - 2t c_1^T x + t^2 \\ &\text{s. t.} \quad c_1^T x \le t, \ x \in \mathcal{F}. \end{aligned} \tag{37}$$

For convenience, we call the NADM algorithm with problem (36) as NADM1, and the NADM algorithm with problem (37) as NADM2. It is easy to see that if $t = t_l$, then problem (36) reduces to problem (29). If $t = t_u$, then problem (37) reduces to problem (29). Note that when $t^* = c_1^T x^*$, from the optimality of x^* we can conclude that $x(t^*) = x^*$. It follows from Statement (iii) of Proposition 2.2 that if $t \le t^* = c_1^T x^*$, then problem (29) reduces to problem (36); if $t \ge t^*$, then problem (29) reduces to problem (37).

Another concern is the multiplicity in the optimal solution set to problem (36) or problem (37), as illustrated by the following example:

Example 4.1 Consider the following constrained concave optimization:

$$\begin{aligned} \min \quad & 5x_1 + 2x_2 - (x_1 + 2x_2)^2 \\ \text{s. t.} \quad & 2x_1 + 5x_2 \le 6, \quad 0 \le x_1, x_2 \le 1. \end{aligned} \tag{38}$$

It is easy to see that this problem has a unique global optimal solution: $x^* = (0, 1)^T$, with the optimal value $f^* = -2$ and $c_1^T x^* = 2$ ($c_1 = (1, 2)^T$ in the above problem). By solving problems in (4) with $c_1 = (1, 2)^T$, we obtain initial bounds $t_l = 0$ and $t_u = 2.6$. Note that when $t_0 = 2.6$, problem (37) with $t = t_0$ reduces to the following

$$\begin{aligned} \min \quad & -0.2x_1 - 8.4x_2 \\ \text{s. t.} \quad & 2x_1 + 5x_2 \le 6, \quad 0 \le x_1, x_2 \le 1, \end{aligned}$$

which has a unique optimal solution $x^1 = (0.5, 1)^T$ with $c^T x^1 = 2.5$. Let $t_1 = 2.5$. Then problem (37) with $t = t_1$ becomes

$$\begin{aligned} \min \quad & -8x_2 \\ \text{s. t.} \quad & 2x_1 + 5x_2 \le 6, \quad x_1 + 2x_2 \le 2.5, \\ & 0 \le x_1, x_2 \le 1. \end{aligned}$$

The optimal solution set of the above problem is defined by

$$\bar{\mathcal{S}}(t_1) = \left\{ x \in \mathbb{R}^2 : 0 \le x_1 \le 0.5, \ x_2 = 1 \right\}.$$

To address the multiplicity issue in the solution set of problem (37), we propose to solve the following problem

$$\min \ c_1^T x \quad \text{s. t.} \ \ x \in \bar{\mathcal{S}}(t_1),$$

yielding the optimal solution $x^2 = (0, 1)^T$ with the optimal value $c^T x^2 = 2$, which is also the global optimal solution to problem (38).

From the above example, we see that if the optimal solution of problem (36) or problem (37) is not unique, then we need to solve the following problem

$$\max \ c_1^T x \quad \text{s. t.} \ \ x \in \mathcal{S}_1(t), \tag{39}$$

or

$$\min \ c_1^T x \quad \text{s. t.} \ \ x \in \mathcal{S}_2(t), \tag{40}$$

where $\mathcal{S}_1(t)$ and $\mathcal{S}_2(t)$ denote the optimal solution set of problems (36) and (37), respectively. For convenience, we still call the resulting NADM algorithms NADM1 and NADM2, respectively. It is easy to see that both NADM1 and NADM2 enjoy the same iteration bound as that of the original NADM specified by Theorem 4.1.

Similar to Corollary 4.1, we also have the following result for problem (36) or problem (37).

Corollary 4.4 *Let $x(t)$ be the optimal solution to problem (36) [or problem (37)] and $\mu(t)$ be its optimal value. We have*

(i) *If $x \in \mathcal{F}$ satisfies the following inequality*

$$\left|c_1^T x - t\right| \leq \left|c_1^T x(t) - t\right|,$$

then it must hold

$$f(x) \geq f(x(t)).$$

(ii) *Let x^* be the global optimal solution of QP1NE. Then*

$$\left|c_1^T x(t) - t\right| \leq \left|c_1^T x^* - t\right|. \tag{41}$$

(iii) *The function $c_1^T x(t)$ is increasing with respect to t.*
(iv) *The function $\mu(t)$ is lower semi-continuous at $t \in [t_l, t_u]$.*

Proof We first prove Statement (iii). Suppose that there exist $t_1 < t_2 \in [t_l, t_u]$, $x(t_1) \in \mathcal{S}(t_1)$, and $x(t_2) \in \mathcal{S}(t_2)$ satisfying

$$c_1^T x(t_1) > c_1^T x(t_2) \geq t_2. \tag{42}$$

It follows that $x(t_1)$ is a feasible solution for problem (36) with $t = t_2$. From the optimality conditions of problem (36) we obtain

$$\begin{aligned} x(t_2)^T Q_+ x(t_2) + q^T x(t_2) - 2t_2 c_1^T x(t_2) &\leq x(t_1)^T Q_+ x(t_1) + q^T x(t_1) - 2t_2 c_1^T x(t_1), \\ x(t_1)^T Q_+ x(t_1) + q^T x(t_1) - 2t_1 c_1^T x(t_1) &\leq x(t_2)^T Q_+ x(t_2) + q^T x(t_2) - 2t_1 c_1^T x(t_2), \end{aligned}$$

where the second relation is implied by the fact that $x(t_2)$ is a feasible solution for problem (36) with $t = t_1$ (because $c_1^T x(t_2) \geq t_2 > t_1$). Adding the above two inequalities together, we derive

$$\begin{aligned} & t_2 c_1^T x(t_2) + t_1 c_1^T x(t_1) \geq t_2 c_1^T x(t_1) + t_1 c_1^T x(t_2) \\ \Longleftrightarrow\ & (t_2 - t_1)\left(c_1^T x(t_2) - c_1^T x(t_1)\right) \geq 0, \end{aligned}$$

which contradicts to relation (42). This further implies that Statement (iii) holds.

We next prove Statement (iv). Given any $\beta \in \mathbb{R}$, let us denote $\mathcal{L}(\beta) = \left\{t \in [t_l^1, t_u^1] : \mu(t) \leq \beta\right\}$. Let $t_k \in \mathcal{L}(\beta)$ and $t_k \to \bar{t}$ as $k \to \infty$. Then we have $\mu(t_k) \leq \beta$ and hence, there exists $x^k \in \mathcal{F}$ such that

$$\mu(t_k) = \left(x^k\right)^T Q_+ x^k + q^T x^k - 2t_k c_1^T x^k + t_k^2 \leq \beta, \quad c_1^T x^k \geq t_k.$$

Since the set $\mathcal{F}$ is bounded, $\{x^k\}$ admits at least a limit point. Let $\mathcal{K}$ be an infinite subset of $\mathbb{N} = \{1, 2, \ldots\}$ such that $\lim_{k\to\infty, k\in\mathcal{K}} x^k = \bar{x}$. Since $\mathcal{F}$ is closed, we have $\bar{x} \in \mathcal{F}$. Taking the limit in the above inequalities, we obtain

$$\bar{x}^T Q_+ \bar{x} + q^T \bar{x} - 2\bar{t} c_1^T \bar{x} + \bar{t}^2 \le \beta, \quad c_1^T \bar{x} \ge \bar{t}.$$

Thus, $\bar{x}$ is a feasible solution of problem (36) with $t = \bar{t}$. So,

$$\mu(\bar{t}) \le \bar{x}^T Q_+ \bar{x} + q^T \bar{x} - 2\bar{t} c_1^T \bar{x} + \bar{t}^2 \le \beta,$$

which implies $\bar{t} \in \mathcal{L}(\beta)$. This proves that $\mathcal{L}(\beta)$ is closed for any $\beta \in \mathbb{R}$. Thus $\mu(t)$ is lower semi-continuous at $t \in [t_l, t_u]$. The proof is finished. □

Similar to Corollary 4.2, we have the following result.

Corollary 4.5 *Let the two sequence $\{x^k\}$ and $\{t^k\}$ be generated by Algorithm NADM1 (or NADM2) and x^* be the global solution of QP1NE. We have*

(i) *The sequence $\{t^k\}$ generated by NADM1 is non-decreasing. Particularly, if $t^0 \le t^* = c_1^T x^*$, then the sequence $\{t^k\}$ is bounded above by t^*.*
(ii) *The sequence $\{t^k\}$ generated by NADM2 is non-increasing. Particularly, if $t^0 \ge t^* = c_1^T x^*$, then the sequence $\{t^k\}$ is bounded below by t^*;*
(iii) *$f(x^{k-1}) - f(x^k) \ge |t^{k-1} - t^k|^2$ for all k.*

Corollary 4.6 *Let the sequence $\{t^k\}$ be generated by Algorithm NADM1 (or NADM2) and $\mu(t)$ be the optimal value of problem (36) [or problem (37)]. If $\lim_{k\to\infty} t^k = t^+$, then $\lim_{k\to\infty} \mu(t^k) = \mu(t^+)$.*

Proof We consider the case where $\{t^k\}$ is generated by Algorithm NADM1. By Corollary 4.5 (i), $\{t^k\}$ is increasing. If $\lim_{k\to\infty} t^k = t^+$, then $t^k \le t^+$ for all k. Let $x(t^k) \in \mathcal{S}_1(t^k)$ and $x(t^+) \in \mathcal{S}_1(t^+)$. It is easy to see that $x(t^+)$ is a feasible solution to problem (36) with $t = t^k$ (because $c_1^T x(t^+) \ge t^+ \ge t^k$). We then obtain

$$\begin{aligned}
\mu\left(t^k\right) &= x\left(t^k\right)^T Q_+ x\left(t^k\right) + q^T x\left(t^k\right) - 2t^k c_1^T x\left(t^k\right) + \left(t^k\right)^2 \\
&\le x\left(t^+\right)^T Q_+ x\left(t^+\right) + q^T x\left(t^+\right) - 2t^k c_1^T x\left(t^+\right) + \left(t^k\right)^2 \\
&= x\left(t^+\right)^T Q_+ x\left(t^+\right) + q^T x\left(t^+\right) - 2t^+ c_1^T x\left(t^+\right) + \left(t^+\right)^2 \\
&\quad + 2\left(t^+ - t^k\right) c_1^T x\left(t^+\right) + \left(t^k\right)^2 - \left(t^+\right)^2 \\
&= \mu\left(t^+\right) + 2\left(t^+ - t^k\right) c_1^T x\left(t^+\right) + \left(t^k\right)^2 - \left(t^+\right)^2.
\end{aligned}$$

By Corollary 4.4 (iv), $\mu(t)$ is lower semi-continuous at t^+. It follows from the above inequality that

$$\mu\left(t^+\right) \le \liminf_{k\to\infty} \mu\left(t^k\right) \le \limsup_{k\to\infty} \mu\left(t^k\right) \le \mu\left(t^+\right).$$

This proves $\lim_{k\to\infty} \mu(t^k) = \mu(t^+)$. The proof is finished. □

Similar to Theorem 4.2, we can obtain the following result for NADM1 or NADM2 by using Corollaries 4.4–4.6.

Theorem 4.3 *Let $\epsilon = 0$ in the NADM1 (or NADM2) algorithm, x^* be the global solution of QP1NE and $t^* = c_1^T x^*$. If $\{t^k\}$ is an infinite sequence obtained by running NADM1 (or NADM2) from a starting point $t^0 \le t^*$ (or $t^0 \ge t^*$), then $\{t^k\}$ is an increasing (or decreasing) sequence converging to t^+ and t^+ is a lower (or upper) semi-local minimizer of problem* (28).

4.3 A new line search procedure

As pointed out in Sect. 4.2, we can run the NADM algorithm with two different starting points $t^0 = t_l$ and $t^0 = t_u$, and generate two t-sequences with two accumulation points $\bar{t}_l$ and $\bar{t}_u$ satisfying $\bar{t}_l \le \bar{t}_u$. If $\bar{t}_l = \bar{t}_u$, then $\bar{t}_l$ is a globally optimal solution of QP1NE. In what follows we focus on the case $\bar{t}_l < \bar{t}_u$.

Note that since both $\bar{t}_l$ and $\bar{t}_u$ are accumulation points from the sequences generated by the NADM algorithm, by Theorem 4.2, they are also semi-local minimizers of $g(t)$, where $g(t)$ is given in (28). As a consequence, the NADM algorithm with starting point $\bar{t}_l$ and $\bar{t}_u$ can no longer generate a sequence to improve the objective function $g(t)$. On the other hand, if we define

$$t_b = \arg\min \left(g\left(\bar{t}_l\right), g\left(\bar{t}_u\right) \right),$$

and x_b be the solution of problem (29) with $t = t_b$. Then t_b can be viewed as a potential global optimal solution of $g(t)$. An interesting question arises here is how to further cut the subinterval $[\bar{t}_l, \bar{t}_u]$ without missing a solution better than t_b? We next introduce a new line search procedure to achieve such a goal. For convenience, we rewrite restricted problem (21) with $r = 1$

$$\begin{aligned} \min \quad & f(x,t) = x^T Q_+ x + q^T x - t^2, \\ \text{s. t.} \quad & c_1^T x - t = 0, \\ & x \in \mathcal{F}, \quad t \in [l_t, u_t], \end{aligned} \tag{43}$$

and consider its convex relaxation

$$\begin{aligned} \min \quad & x^T Q_+ x + q^T x - s, \\ \text{s. t.} \quad & c_1^T x - t = 0, \\ & t^2 \le s, \quad s \le (l_t + u_t)\, t - l_t u_t, \\ & x \in \mathcal{F}, \quad t \in [l_t, u_t]. \end{aligned} \tag{44}$$

From Theorem 3.1, we have immediately

Theorem 4.4 *Let $f^*_{[l_t,u_t]}$ and $b_{[l_t,u_t]}$ denote the optimal value of problem (43) and its relaxation (44), respectively. Let $(\bar{x}, \bar{s}, \bar{t})$ be the optimal solution to problem (44). Then it holds*

$$f^*_{[l_t,u_t]} - b_{[l_t,u_t]} \leq f\left(\bar{x}, \bar{t}\right) - b_{[l_t,u_t]} \leq \frac{1}{4}(u_t - l_t)^2 . \tag{45}$$

We next introduce a new line search procedure based on Theorem 4.4.

Algorithm 4.1 [*Line search algorithm $LSA(t_0, \delta, g_b, t_b, \epsilon)$*]

Input: Boundary point $t = t_0$, δ, parameters t_b and $g_b = g(t_b) \leq g(t_0)$, stopping criterion ϵ;
Output: Final value δ;
Step 1 Let $l_t = \min(t_0, t_0 + \delta)$, $u_t = \max(t_0, t_0 + \delta)$;
Step 2 Solve the relaxation problem (44) to obtain a lower bound $b_{[l_t,u_t]}$ and an optimal solution $(\bar{x}, \bar{s}, \bar{t})$;
Step 3 If $f(\bar{x}, \bar{t}) < g_b$, then update $(x_b, t_b) = (\bar{x}, \bar{t})$, $g_b = f(x_b, t_b)$;
If either $b_{[l_t,u_t]} \geq g_b - \epsilon$ or $f(\bar{x}, \bar{t}) - b_{[l_t,u_t]} \leq \epsilon$, then stop and output δ;
Otherwise update $\delta = \frac{\delta}{2}$ and go to Step 1;

As shown in the proof of Theorem 4.2, the function $g(t)$ defined in (28) is lower semi-continuous at $t = t_0 \in [t_l, t_u]$. Due to the fact that $g(t_0) \geq g_b$, it follows from Theorems 4.4 that the line search procedure $LSA(t_0, \delta, g_b, t_b, \epsilon)$ is well-defined. Moreover, if $\delta \leq 2\sqrt{\epsilon}$, then from relation (45) we can see that the line search procedure will stop. Since in the above line search procedure, we cut off half of the interval length in all the iterations before the final output, it follows directly that

Theorem 4.5 *Given $t_0 = c_1^T x(t_0)$ and $g_b \leq g(t_0)$, the line search procedure $LSA(t_0, \delta, g_b, t_b, \epsilon)$ will stop in at most $\mathcal{O}(\log \frac{\delta}{2\sqrt{\epsilon}})$ steps.*

We next characterize the final subinterval $[l_t, u_t]$ provided by the line search procedure $LSA(t_0, \delta, g_b, t_b, \epsilon)$. We have

Theorem 4.6 *Given $t_0 = c_1^T x(t_0)$ and $g_b \leq g(t_0)$, the line search procedure $LSA(t_0, \delta, g_b, t_b, \epsilon)$ will identify an subinterval $[l_t, u_t]$ such that*

$$g(t) \geq g_b - \epsilon, \quad \forall t \in [l_t, u_t].$$

Proof We first note that in the above line search procedure, two stopping criteria, $b_{[l_t,u_t]} \geq g_b - \epsilon$ and $f(\bar{x}, \bar{t}) - b_{[l_t,u_t]} \leq \epsilon$, are used. The conclusion of the theorem holds whenever the criterion $b_{[l_t,u_t]} \geq g_b - \epsilon$ is used. If the criterion $f(\bar{x}, \bar{t}) - b_{[l_t,u_t]} \leq \epsilon$ is used in the line search procedure, there are two possibilities $b_{[l_t,u_t]} \geq g_b - \epsilon$ or $b_{[l_t,u_t]} < g_b - \epsilon$. In the first case we obtain the desirable inequality in the theorem. In the latter case, we have

$$f\left(\bar{x}, \bar{t}\right) \leq b_{[l_t,u_t]} + \epsilon < g_b.$$

Under such a circumstance, we should have updated $g_b = f(\bar{x}, \bar{t})$ as described in Step 3 of the line search procedure. It follows from the stopping criterion that

$$g_b - g(t) \leq g_b - b_{[l_t, u_t]} = f\left(\bar{x}, \bar{X}, \bar{t}\right) - b_{[l_t, u_t]} \leq \epsilon,$$

which equals to the inequality in the theorem. □

4.4 The global search method

In this subsection, we introduce a new global search algorithm for QP1NE, establish its convergence and estimate its complexity.

We first discuss how to integrate the NADM algorithm with the new line search procedure to develop a global algorithm for QP1NE. As pointed out in Sect. 4.2, we can first run the NADM algorithm with starting points $t^0 = t_l$ and $t^0 = t_u$, respectively, and obtain two accumulation points $(x_l, \bar{t}_l)$ and $(x_u, \bar{t}_u)$. It is easy to see that $t^* = c_1^T x^* \in [\bar{t}_l, \bar{t}_u]$, where x^* is the global optimal solution of QP1NE.

For simplicity of discussion, let us define $x_b = \arg\min(f(x_l), f(x_u))$ and $f_b = f(x_b)$ with $t_b = c_1^T x_b$. We select $\delta = \frac{\bar{t}_u - \bar{t}_l}{2}$ and run $LSA(\bar{t}_l, \delta, f_b, t_b, \epsilon)$. Because $\delta > 0$, the final subinterval identified in the line search procedure becomes $[\bar{t}_t, \bar{t}_l + \delta]$. Moreover, from Theorem 4.6, we have

$$g(t) \geq f_b - \epsilon, \quad \forall t \in [\bar{t}_l, \bar{t}_l + \delta].$$

The above relation implies that we have either

$$t^* \notin [\bar{t}_l, \bar{t}_l + \delta],$$

or the updated point x_b is an ϵ-approximate solution to QP1NE. This implies that we can cut off the subinterval $[\bar{t}_l, \bar{t}_l + \delta]$ without missing a potentially global ϵ-solution to QP1NE, and then update $\bar{t}_l = \bar{t}_l + \delta$. Similarly, we can also run $LSA(\bar{t}_u, -\delta, f_b, t_b, \epsilon)$ and then use the output from it to cut off another subinterval $[\bar{t}_u - \delta, \bar{t}_u]$, and update $\bar{t}_u = \bar{t}_u - \delta$. Note that after the updates on $\bar{t}_l$ and $\bar{t}_u$, we can reuse NADM1 and NADM2 to find two new semi-local optimal solutions, and reuse LSA to identify two new subintervals that can be cut off. We can repeat such a process until a stopping criteria is met.

The new global search algorithm for QP1NE is described below.

Algorithm 4.2 [*Global search algorithm (GSA)*]

Input: $Q, q, Q_i, q_i, d_i (i = 1, \ldots, m)$, A, b, initial bound t_l, t_u and stopping criteria $\epsilon > 0$;

Output: A global ϵ-solution (x^*, t^*);

Step 0 Find two semi-local minimal solutions (x_l^0, t_l^0) and (x_u^0, t_u^0) of QP1NE by running $ADM(t_l, \epsilon)$ and $ADM(t_u, \epsilon)$, respectively. Set $x_b^0 = \arg\min\{f(x_l^0), f(x_u^0)\}$, $f_b^0 = f(x_b^0)$, $t_b^0 = c_1^T x_b^0$. Set $k = 0$.

Step 1 **While** $t_u^k - t_l^k > 2\sqrt{\epsilon}$ **Do (the main loop)**

(S1.1) Set $\delta^k = (t_u^k - t_l^k)/2$. Solve problem (44) with $[l_t = t_l^k, u_t = t_l^k + \delta^k]$ and $[l_t = t_u^k - \delta^k, u_t = t_u^k]$ to obtain the optimal values v_l^k and v_u^k, respectively.

(S1.2) If $v_l^k \geq f_b^k - \epsilon$, then find a cut-off subinterval $[t_l^k, t_l^k + \delta^k]$ and update $t_l^k = t_l^k + \delta^k$; Run $NADM1(t_l^k, \epsilon)$ to find a semi-local minimal solution (x_l^k, t_l^k); Update $k = k + 1$, $t_l^k = c_1^T x_l^{k-1}$, $t_u^k = t_u^{k-1}$, $x_b^k = \arg\min\{f_b^{k-1}, f(x_l^k)\}$, $f_b^k = f(x_b^k)$, $t_b^k = c_1^T x_b^k$, go to step (S1.5).

(S1.3) If $v_u^k \geq f_b^k - \epsilon$, then find a cut-off subinterval $[t_u^k - \delta^k, t_u^k]$, update $t_u^k = t_u^k - \delta^k$; Run $NADM2(t_u^k, \epsilon)$ to find a semi-local minimal solution (x_u^k, t_u^k); Update $k = k+1, t_l^k = t_l^{k-1}, t_u^k = c_1^T x_u^{k-1}, x_b^k = \arg\min\{f_b^{k-1}, f(x_u^k)\}$, $f_b^k = f(x_b^k)$, $t_b^k = c_1^T x_b^k$, go to step (S1.5).

(S1.4) If $v_l^k < f_b^k - \epsilon$ and $v_u^k < f_b^k - \epsilon$, then set $\delta_l^k = \delta_u^k = \delta^k/2$; Run $LSA(t_l^k, \delta_l^k, f_b^k, t_b^k, \epsilon)$ and $LSA(t_u^k, -\delta_u^k, f_b^k, t_b^k, \epsilon)$ to find two cut-off subintervals $[t_l^k, t_l^k + \delta_l^k]$ and $[t_u^k - \delta_u^k, t_u^k]$; Update $t_l^k = t_l^k + \delta_l^k$, $t_u^k = t_u^k - \delta_u^k$; Run $NADM1(t_l^k, \epsilon)$ and $NADM2(t_u^k, \epsilon)$ to find two semi-local minimal solutions (x_l^k, t_l^k) and (x_u^k, t_u^k), respectively; Update $k = k + 1$, $t_l^k = c_1^T x_l^{k-1}$, $t_u^k = c_1^T x_u^{k-1}$, $x_b^k = \arg\min\{f_b^{k-1}, f(x_l^k), f(x_u^k)\}$, $f_b^k = f(x_b^k)$, $t_b^k = c_1^T x_b^k$.

(S1.5) If $|t_u^k - t_l^k| \leq 2\sqrt{\epsilon}$, then go to Step 2.
End While

Step 2 If $t_u^k \geq t_l^k$, then output (x_b^k, t_b^k) as the final solution. Otherwise, solve problem (44) with $[l_t = t_l^k, u_t = t_u^k]$ to obtain an ϵ-approximate solution (x^*, s^*, t^*); If $f(x_b^k) \leq f(x^*)$, then set $(x^*, t^*) = (x_b^k, t_b^k)$; Output (x^*, t^*) as the final solution.

We next discuss the convergence of the GSA algorithm and estimate its complexity. We consider the worst-case scenario where the Step (S1.4) is always executed at each iteration in the main loop of the algorithm. There are three sequences generated by the GSA algorithm, $\{f_b^k\}$, $\{t_l^k\}$ and $\{t_u^k\}$. It is easy to verify that both t_u^k and f_b^k are decreasing while t_l^k is increasing in terms of k. Note that the two sequences t_l^k and t_u^k are updated after one run of either the LSA algorithm or the NADM algorithm. Therefore, the total number of updates in t_l^k and t_u^k can be expressed as

$$T = T_{LSA} + \sum_k T_{ADM}^k,$$

where T_{LSA} is the total number of the LSA runs, and T_{ADM}^k is the number of updates in the parameters t_l^k and t_u^k in the kth run of both NADM1 and NADM2.

Now let us examine the case when both t_l^k and t_u^k are updated after one run of the LSA algorithm. Since a necessary condition to cut half of $|\delta|$ is when $|\delta| > 2\sqrt{\epsilon}$, it follows immediately that the width of the final subinterval (the subinterval to be cut-off) in the line search process is at least $\sqrt{\epsilon}$. This implies

$$t_u^k - t_l^k \leq t_u^{k-1} - t_l^{k-1} - 2\sqrt{\epsilon}$$

if both t_l^k and t_u^k are updated after one run of the LSA algorithm. If t_l^k and t_u^k are updated in the ADM process (by NADM1 and NADM2 respectively), then the increment in every update of t_l^k (or decrement in the update of t_u^k) is also at least $\sqrt{\epsilon}$. Recall the fact there is no overlaps in the subintervals scanned by the NADM algorithm and the cut-off subintervals identified in the LSA process. It follows

$$T_{LSA} + \sum_k T_{ADM}^k \leq \left\lceil \frac{t_u - t_l}{\sqrt{\epsilon}} \right\rceil.$$

As stated in Theorem 4.5, the total number of steps (which equals to the total number of subproblems (44) solved) in every run of the LSA algorithm is bounded above by

$$\mathcal{O}\left(\log \frac{t_u - t_l}{\sqrt{\epsilon}}\right).$$

From the above discussion, we immediately obtain the following result.

Theorem 4.7 *The GSA algorithm can find an ϵ-approximate solution to QP1NE by solving at most $\mathcal{O}(\frac{t_u - t_l}{\sqrt{\epsilon}} \log \frac{t_u - t_l}{2\sqrt{\epsilon}})$ quadratic convex optimization problems*[2].

5 Numerical results

In this section, we present computational results of the ADMBB and GSA algorithms for the QCQP and QP1NE problems, respectively. The algorithm is coded in Matlab R2013b and run on a PC (3.4GHz, 16GB RAM). All the linear and convex quadratic subproblems in ADMBB and GSA are solved by the QP solver in CPLEX 12.6 with Matlab interface [15].

In our computational experiments, the stopping parameter ϵ is set as $\epsilon = 10^{-6}$. We use the notations described in Table 1 in the discussion of computational results.

5.1 Numerical results of the ADMBB algorithm

In this subsection, we test the ADMBB algorithm on four classes of randomly generated quadratic optimization problems: the concave 0–1 quadratic program, the box constrained quadratic program, the linearly constrained quadratic program and the convex quadratically constrained quadratic program problem. We also test the ADMBB algorithm on the optimal spectrum sharing problem in MIMO cognition radio networks studied in [48]. Since the complexity of ADMBB grows exponentially in terms of the negative eigenvalues of the Hessian matrix in the objective function, our experiments are restricted only to instances with a few negative eigenvalues (say, $r \leq 10$).

[2] All the quadratic convex optimization problems solved in the GSA algorithm are in form of (36), (37) or (44).

Table 1 Notations

n	The number of variables
n_{qc}	The number of quadratic inequality constraints
n_{lc}	The number of linear constraints
r	The number of negative eigenvalues of the matrix Q
Opt. val	The average optimal value obtained by ADMBB or GSA for five test instances
CPU	The average CPU time of ADMBB or GSA in seconds for five test instances
Iter	The average number of iterations in the main loop of ADMBB or GSA for five test problems
T_{ADM}	The average number of restarting ADM in the main loop of ADMBB for five test problems
Val_{ADM}	The average objective value at the solution computed by ADM in ADMBB for five test problems
T_{LSA}	The average number of running LSA in the main loop of GSA for five test problems
It_{ADM}	The average number of iterations in the NADM algorithms used in GSA for five test problems
t_{ADM}	The average CPU time of ADM used in GSA in seconds for five test problems

5.1.1 The concave 0–1 quadratic program

We first test the ADMBB algorithm on the following concave 0–1 quadratic programming problems:

$$\begin{aligned} &\min\ x^T Qx + q^T x \\ &s.t.\ x \in \{0, 1\}^n, \end{aligned} \tag{46}$$

where Q is an $n \times n$ symmetric negative semidefinite matrix with the r negative eigenvalues, $r \geq 2$, $q \in \mathbb{R}^n$. Since the objective function is concave, problem (46) is equivalent to the following

$$\begin{aligned} &\min\ x^T Qx + q^T x \\ &s.t.\ x \in [0, 1]^n. \end{aligned}$$

Note that $Q = -C^T C$ for some $C \in \mathbb{R}^{r\times n}$, the above problem can be further rewritten as the following lifted problem

$$\begin{aligned} &\min\ q^T x - \|t\|_2^2 \\ &s.t.\ x \in [0, 1]^n,\ \ t = Cx, \end{aligned}$$

which can be solved by the ADMBB algorithm. For numerical comparison, we also use the global optimization package CPLEX [15] for 0–1 quadratic programming to find the global optimal solution for problem (46) in medium and large scale.

The parameters in the test problems are randomly generated in the same fashion as in [22]. For self-completion, we next describe how the test problems in this work are generated. Let $\xi \in U[\underline{a}, \overline{a}]$ be a random number uniformly distributed in the interval $[\underline{a}, \overline{a}]$:

- Q takes the form: $Q = PTP^T, T = \text{diag}(T_1, \ldots, T_n), P = W_1W_2W_3, W_j = I - 2\frac{w_j w_j^T}{\|w_j\|^2}$, $j = 1, 2, 3$, $w_j = (w_{j1}, \ldots, w_{jn})^T$, $w_{jk} \in U[-1, 1]$ for $j = 1, 2, 3$ and $k = 1, \ldots, n$; $T_k \in U[-1, 0]$ for $k = 1, \ldots, r$, and $T_k = 0$ for $k = r + 1, \ldots, n$. Also, $q \in U[-1, 1]$.

Table 2 compares the average performance of ADMBB for five instances of problem (46) with that of CPLEX. As one can see from the table that, for medium and large size instances, both ADMBB and CPLEX are able to find the global optimal solution effectively. However, CPLEX usually takes much longer time than ADMBB when $r \leq 5$, while ADMBB takes much longer time than CPLEX when $r > 5$. We also observe that the CPU time of ADMBB grows very fast as the number of the negative eigenvalue increases.

5.1.2 The box constrained quadratic program

In this subsection we test ADMBB on the box constrained quadratic programming problems (BCQP) as follows

$$\begin{aligned} &\min\ x^T Qx + q^T x \\ &s.t.\ x \in [0, 1]^n, \end{aligned} \tag{47}$$

where Q is an $n \times n$ symmetric indefinite matrix with the r negative eigenvalues, $q \in \mathbb{R}^n$.

We compare the ADMBB with the BBKKT-SDP algorithm developed by Burer and Vandenbussche [5] that combines the B&B framework with SDP relaxation for BCQP. The code for BBKKT-SDP can be found in http://sburer.github.io/projects.html, with the default setting. For numerical comparison, we also use BBKKT-SDP to find the global optimal solution for test problems in small scale.

We first compare the performance of ADMBB and BBKKT-SDP for problem (47) with a few negative eigenvalues ($r \leq 15$). The parameters in the test problems are randomly generated in the same way as described in Sect. 5.1.1, where $T_k \in U[-1, 0]$ for $k = 1, \ldots, r$, and $T_k \in U[0, 1]$ for $k = r + 1, \ldots, n$. The average numerical results for five test problems of the same size are summarized in Table 3. One can see from the table that, for all test problems, both ADMBB and BBKKT-SDP are able to find the global optimal solution. As one can see from Table 3, ADMBB is more effective than BBKKT-SDP when $r \leq 10$, while BBKKT-SDP is more effective than ADMBB when $r = 15$. We also observe that the CPU time of BBKKT-SDP grows very fast as the size of the test problem grows, while the CPU time of ADMBB grows very fast as the number of the negative eigenvalue increases.

5.1.3 The linearly constrained quadratic program

In this subsection we test ADMBB on the following linearly constrained quadratic program problems (denoted by LCQP-r-NE):

$$\min \quad x^T Qx + q^T x$$

Table 2 Average performance of ADMBB versus CPLEX for five instances of problem (46)

n	r	ADMBB			CPLEX	
		Opt.val	CPU	Iter	Opt.val	CPU
500	3	− 121.446	0.36	32	− 121.4465	0.13
1000	3	− 247.1086	0.51	39	− 247.1086	0.46
1500	3	− 376.6812	0.61	37	− 376.6812	1.59
2000	3	− 504.7223	0.79	35	− 504.7223	3.76
2500	3	− 618.3592	0.97	37	− 529.7206	7.24
3000	3	− 762.1462	1.29	41	− 762.1462	12.25
3500	3	− 885.5978	1.26	29	− 885.5978	19.21
4000	3	− 995.1895	1.72	34	− 995.1895	28.33
4500	3	− 1119.9666	1.81	34	− 1119.9666	40.38
5000	3	− 1248.6297	1.99	33	− 1248.6297	54.85
500	5	− 124.1126	1.86	142	− 124.1126	0.15
1000	5	− 249.2471	2.00	115	− 249.2471	0.51
1500	5	− 377.1099	3.40	138	− 377.1099	1.74
2000	5	− 511.4220	3.55	113	− 511.4220	4.02
2500	5	− 624.5077	4.41	107	− 624.5077	7.75
3000	5	− 752.0400	7.93	152	− 752.0400	12.47
3500	5	− 877.9040	6.90	119	− 877.9040	19.18
4000	5	− 989.9204	9.51	138	− 989.9204	28.31
4500	5	− 1125.3339	12.53	168	− 1125.3339	39.85
5000	5	− 1248.1847	8.52	89	− 1248.1847	54.24
500	8	− 129.7859	10.72	972	− 129.7859	0.13
1000	8	− 253.5912	12.99	719	− 253.5912	0.46
1500	8	− 373.0101	42.11	1310	− 373.0101	1.55
2000	8	− 497.8023	59.38	1188	− 497.8023	3.72
2500	8	− 638.0319	39.26	732	− 638.0319	7.19
3000	8	− 747.6093	50.50	709	− 747.6093	12.22
3500	8	− 868.5273	81.47	898	− 868.5273	19.14
4000	8	− 1006.2500	88.32	782	− 1006.2500	28.27
4500	8	− 1127.5053	102.14	769	− 1127.5053	39.83
5000	8	− 1267.4917	126.08	861	− 1267.4917	54.23
500	10	− 125.9935	47.39	3424	− 125.9935	0.13
1000	10	− 254.8184	74.02	3092	− 254.8184	0.45
1500	10	− 378.7925	80.82	2272	− 378.7925	1.56
2000	10	− 512.7058	79.93	1634	− 512.7058	3.70
2500	10	− 618.8881	64.54	1027	− 618.8881	7.86
3000	10	− 753.2915	167.37	2029	− 753.2915	12.35
3500	10	− 901.0884	187.09	1728	− 901.0884	19.86
4000	10	− 1014.0447	309.57	2394	− 1014.0447	28.70

Table 2 continued

n	r	ADMBB			CPLEX	
		Opt.val	CPU	Iter	Opt.val	CPU
4500	10	− 1118.4502	367.41	2186	− 1118.4502	39.98
5000	10	− 1237.8992	416.51	2317	− 1237.8992	54.22

Table 3 Average performance of ADMBB versus BBKKT-SDP for five instances of problem (47)

n	r	ADMBB			BBKKT-SDP	
		Opt.val	CPU	Iter	Opt.val	CPU
50	5	− 8.6920	5.24	518	− 8.6920	27.51
100	5	− 11.1617	4.21	276	− 11.1617	167.65
150	5	− 18.3942	5.93	189	− 18.3942	926.54
200	5	− 23.3290	12.26	176	− 23.3290	2906.16
50	8	− 10.1957	13.42	1661	− 10.1957	15.12
100	8	− 15.6582	13.36	1344	− 15.6582	261.59
150	8	− 20.6107	21.17	1439	− 20.6107	1223.20
200	8	− 22.7770	15.52	739	− 22.7770	3606.50
50	10	− 8.7769	127.75	7685	− 8.7769	24.78
100	10	− 16.7020	54.74	4396	− 16.7020	297.64
150	10	− 17.7594	66.18	4163	− 17.7594	1403.02
200	10	− 26.8220	97.90	4141	− 26.8220	3558.76
50	15	− 14.0598	3364.47	46309	− 14.0598	21.67
100	15	− 17.3562	3119.90	51414	− 17.3562	487.79
150	15	− 22.8929	2577.85	38748	− 22.8929	1064.74
200	15	− 29.4655	3306.36	48358	− 29.4655	3606.05

$$s.t. \quad Ax \le b, \ x \in [0,1]^n, \tag{48}$$

where Q is an $n \times n$ symmetric indefinite matrix with the r negative eigenvalues, $r \ge 2$, $q \in \mathbb{R}^n$, and $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$. To test the performance of ADMBB, we randomly generate the parameters in the test problems in the following fashion:

- The parameters Q and q are randomly generated in the same fashion as in Sect. 5.1.1 with $T_k \in U[-1,0]$ for $k = 1,\ldots,r$, and $T_k \in U[0,1]$ for $k = r+1,\ldots,n$.
- Let $A = (a_{ij}) \in \mathbb{R}^{l\times n}$ and $b = (b_1,\ldots,b_l)^T \in \mathbb{R}^l$. For each $i = 1,\ldots,l$, $j = 1,\ldots,n$, $a_{ij} \in U[-5,5]$ and $b_i \in U[v_i, v_i+1]$ with $v_i = 0.5\sum_{j=1}^n \max(0, a_{ij})$.

For numerical comparison, we also use the BBKKT-SDP proposed in [4] to find the global solution to the underlying LCQP. The BBKKT-SDP code for LCQP can be downloaded from http://sburer.github.io/projects.html.

In Table 4, we compare the average performance of ADMBB and BBKKT-SDP for five random instances of problem (48). From Table 4, one can see that, for all test

Table 4 Average performance of ADMBB versus BBKKT-SDP for five instances of problem (48)

n_{lc}	n	r	ADMBB			BBKKT-SDP	
			Opt.val	CPU	Iter	Opt.val	CPU
10	50	5	− 9.2099	3.00	250	− 9.2099	101.30
16	80	5	− 10.5547	5.49	310	− 10.5547	434.85
20	100	5	− 13.7867	6.54	342	− 13.7867	961.48
30	150	5	− 18.3922	7.99	172	− 18.3922	3267.64
40	200	5	− 24.2800	24.43	165	–	–
10	50	8	− 11.2068	15.07	1766	− 11.2068	68.13
16	80	8	− 12.3439	20.76	1719	− 12.3439	399.35
20	100	8	− 14.3047	15.80	1422	− 14.3047	1262.06
30	150	8	− 19.3166	28.67	1710	− 19.3166	3364.07
40	200	8	− 26.4148	46.49	1658	–	–
10	50	10	− 12.2747	73.00	6715	− 12.2747	88.81
16	80	10	− 15.5889	117.60	7318	− 15.5889	402.91
20	100	10	− 13.9999	99.88	6347	− 13.9999	995.14
30	150	10	− 20.5975	79.84	4355	− 20.5975	2959.18
40	200	10	− 25.7982	113.44	4179	–	–

"–" stands for the method fails to find the global solution within 3600 s at all cases

problems, ADMBB is able to find the global optimal solution, while BBKKT-SDP fails to find the global optimal solutions for several large instances of LCQP-r-NE with $n = 200$. Moreover, BBKKT-SDP usually takes longer time than ADMBB. We also observe that the CPU time of ADMBB grows very fast as the number of the negative eigenvalue increases, while the CPU time of BBKKT-SDP grows very fast as the size of the test problem grows.

In Table 5, we report the testing result of ADMBB for medium and large scale LCQP-r-NE instances. As one can see from Table 5, ADMBB is able to effectively find the global optimal solution for all the test problems.

5.1.4 The quadratically constrained QP-r-NE problem

In this subsection we test ADMBB on generic convex quadratically constrained QP-r-NE problems as follows

$$\begin{aligned} \min \quad & x^T Q_+ x + q^T x - \|Cx\|_2^2 \\ s.t. \quad & x^T Q_i x + q_i^T x \le d_i, \quad i = 1, \ldots, m, \\ & Ax \le b, \ x \in [0,1]^n, \end{aligned} \tag{49}$$

where Q_+ is an $n \times n$ symmetric positive semi-definite matrix and $C \in \mathbb{R}^{r \times n}$ derived from the singular value decomposition of the $n \times n$ symmetric indefinite matrix Q with the r negative eigenvalues, Q_i is an $n \times n$ symmetric positive semidefinite matrix, $q, q_i \in \mathbb{R}^n, d_i \in \mathbb{R}, i = 1, \ldots, m$, $A \in \mathbb{R}^{l \times n}, b \in \mathbb{R}^l$.

Table 5 Average numerical results of ADMBB for five instances of problem (48)

n_{lc}	n	r	ADMBB				
			Opt.val	CPU	Iter	Val_{ADM}	T_{ADM}
100	500	2	− 54.9735	7.79	24	− 54.9735	5
120	600	2	− 66.1446	13.40	24	− 66.1446	5
140	700	2	− 77.2127	25.83	24	− 77.2127	8
160	800	2	− 88.6795	41.02	31	− 88.6795	7
180	900	2	− 96.1075	45.18	25	− 96.1075	4
200	1000	2	− 109.1305	69.41	24	− 109.1305	6
300	1500	2	− 154.4003	238.04	23	− 154.4003	8
400	2000	2	− 207.7687	499.72	27	− 207.7687	8
500	2500	2	− 263.6916	994.20	24	− 263.6916	10
600	3000	2	− 316.3382	1738.51	24	− 316.3382	10
100	500	3	− 52.3923	18.91	56	− 52.3923	8
120	600	3	− 63.1775	28.15	49	− 63.1775	8
140	700	3	− 75.2076	51.07	59	− 75.2076	8
160	800	3	− 84.7700	68.22	48	− 84.7700	6
180	900	3	− 92.9606	96.39	49	− 92.9606	9
200	1000	3	− 101.4180	135.09	52	− 101.4180	8
300	1500	3	− 158.6386	423.59	55	− 158.6386	9
400	2000	3	− 214.3383	1015.25	62	− 214.3383	12
100	500	4	− 53.2018	34.16	100	− 53.2018	8
120	600	4	− 61.5990	53.31	92	− 61.5990	7
140	700	4	− 79.8624	89.83	93	− 79.8624	9
160	800	4	− 83.3692	138.01	95	− 83.3692	8
180	900	4	− 97.0748	173.27	77	− 97.0748	7
200	1000	4	− 105.0501	251.01	97	− 105.0501	10
300	1500	4	− 155.8478	840.80	96	− 155.8478	17
400	2000	4	− 215.4962	1683.98	85	− 215.4962	13
100	500	5	− 57.4846	59.45	154	− 57.4846	6
120	600	5	− 64.8063	101.50	172	− 64.8063	11
140	700	5	− 71.8350	176.95	147	− 71.8350	10
160	800	5	− 90.5077	250.84	160	− 90.5077	11
180	900	5	− 99.1977	364.84	169	− 99.1977	7
200	1000	5	− 109.1660	534.45	179	− 109.1595	15
60	300	8	− 34.3331	80.11	1494	− 34.1465	21
80	400	8	− 47.7195	121.90	1272	− 47.4732	13
100	500	8	− 56.5444	151.87	986	− 56.4740	25
120	600	8	− 66.8500	285.42	1090	− 66.6807	11
140	700	8	− 75.3080	314.48	816	− 74.9937	20

Table 5 continued

n_{lc}	n	r	ADMBB				
			Opt.val	CPU	Iter	Val_{ADM}	T_{ADM}
160	800	8	− 89.1028	813.22	1366	− 88.7718	14
60	300	10	− 37.5442	222.53	3440	− 37.4433	32
80	400	10	− 48.0852	267.71	2522	− 47.7347	12
100	500	10	− 55.7344	400.00	2273	− 55.5589	53
120	600	10	− 70.3193	527.06	1937	− 70.2200	15

The parameters in the test problems are randomly generated as follows:

- The parameters Q, q, A, b are randomly generated in the same fashion as in Sect. 5.1.3. Set $Q_+ = \sum_{j=r+1}^{n} T_j p_j p_j^T$, and $C = \left(\sqrt{-T_1}\, p_1, \ldots, \sqrt{-T_r}\, p_r\right)^T$, where p_j is the jth column vector of P, $j = 1, \ldots, n$.
- For each $i = 1, \ldots, m$, Q_i is randomly generated in the same fashion as in Sect. 5.1.1 with $T_j \in U[0, 5]$ for $j = 1, \ldots, n$. Also, $q_i \in U[0, 10]$ for $i = 1, \ldots, m$. For feasibility consideration, we set $d_i \in U[v_i, v_i + 1]$ with $v_i = x_0^T Q_i x_0 + q_i^T x_0$ for $i = 1, \ldots, m$, where x_0 is randomly drawn from [0,1].

Tables 6 and 7 summarize the numerical results of ADMBB for medium and large scale QCQP-r-NE instances. As one can see from Tables 6 and 7, ADMBB is able to effectively find the global optimal solution for all the test problems.

5.1.5 An application in MIMO cognitive radio networks

In this subsection, we consider an application example of QCQP, which is the optimal spectrum sharing problem in cognitive radio networks with multiple-input multiple-output (MIMO) links studied in [48].

We first give a short description of optimal spectrum sharing problem (more details can be found in [48]). Consider a cognitive radio network in which a secondary user intends to share the spectrum with a primary system consisting of K primary links. We use the subscript S to denote the secondary link and the subscript k to denote the kth primary link. Let M_S (or N_S) and M_k (or N_k) denote the number of transmit (or receive) antennae of the secondary and primary links, respectively. We denote by $\mathbf{H}_{S,S} \in \mathbb{C}^{N_S \times M_S}$ the channel matrix from the secondary transmitter to the secondary receiver, and $\mathbf{H}_{k,S} \in \mathbb{C}^{N_k \times M_S}$, $\mathbf{H}_{S,k} \in \mathbb{C}^{N_S \times M_k}$ and $\mathbf{H}_{k,j} \in \mathbb{C}^{N_k \times M_j}$ the channel matrices from the secondary transmitter to the kth primary receiver, from the kth primary transmitter to the secondary receiver, and from the jth primary transmitter to the kth primary receiver, respectively. Let t_S and r_S denote the beforming vectors at the secondary transmitter and receiver, respectively. Let t_k and r_k denote the beamforming vectors at the kth primary receiver and transmitter, respectively. In

Table 6 Average numerical results of ADMBB for five instances of problem (49)

n_{qc}	n_{lc}	n	r	ADMBB				
				Opt.val	CPU	Iter	Val_{ADM}	T_{ADM}
1	80	400	2	− 42.6771	108.19	19	− 42.6764	2
1	100	500	2	− 53.9867	187.04	16	− 53.9857	2
1	120	600	2	− 62.5769	341.67	17	− 62.5762	1
1	140	700	2	− 74.9288	555.47	14	− 74.9280	2
1	160	800	2	− 82.2338	759.29	14	− 82.2329	1
1	180	900	2	− 93.6923	955.33	13	− 93.6906	1
1	200	1000	2	− 101.7063	1524.22	14	− 101.7052	2
3	80	400	2	− 44.2671	416.61	21	− 44.2668	3
3	100	500	2	− 48.6407	685.65	17	− 48.6399	2
3	120	600	2	− 63.1333	1072.51	15	− 63.1326	2
3	140	700	2	− 75.5520	1719.26	14	− 75.5516	2
3	160	800	2	− 87.2709	3036.00	16	− 87.2697	2
3	180	900	2	− 99.6242	2862.67	12	− 99.6230	1
3	200	1000	2	− 101.9297	3698.20	8	− 101.8748	2
5	80	400	2	− 45.1700	737.35	17	− 45.1696	2
5	100	500	2	− 56.1476	1712.66	20	− 56.1473	1
5	120	600	2	− 64.2648	2694.26	18	− 64.2638	1
5	140	700	2	− 74.4309	3418.63	12	− 74.4295	2
1	80	400	3	− 44.1024	232.12	38	− 44.1016	3
1	100	500	3	− 56.2606	417.84	37	− 56.2603	1
1	120	600	3	− 69.4026	750.07	38	− 69.4022	2
1	140	700	3	− 75.3231	1083.91	29	− 75.3196	1
1	160	800	3	− 86.4293	2043.15	35	− 86.4279	2
1	180	900	3	− 95.3763	2168.96	29	− 95.3741	2
1	200	1000	3	− 104.1409	2390.73	23	− 104.1260	2
3	80	400	3	− 43.7858	816.71	36	− 43.7853	2
3	100	500	3	− 56.9297	1335.62	38	− 56.9293	2
3	120	600	3	− 64.1430	2126.43	35	− 64.1422	4
3	140	700	3	− 75.4758	2918.58	23	− 75.4750	3
5	60	300	3	− 33.9268	651.53	37	− 33.9265	2
5	80	400	3	− 44.3435	1462.74	38	− 44.3431	3
5	100	500	3	− 49.2281	2158.46	23	− 49.2266	3

[48], the signal to interference and noise ratio (SINR) on the secondary link is given by

$$\gamma_S = \frac{\alpha_{S,S} \left| r_S^H H_{S,S} t_S \right|^2}{\sum_{k=1}^{K} \alpha_{S,k} \left| r_S^H H_{S,k} t_k \right|^2 + N_0},$$

Table 7 Average numerical results of ADMBB for five instances of problem (49)

n_{qc}	n_{lc}	n	r	ADMBB				
				Opt.val	CPU	Iter	Val_{ADM}	T_{ADM}
1	40	200	4	− 22.0760	134.82	103	− 22.0758	4
1	60	300	4	− 34.0213	216.35	78	− 34.0210	3
1	80	400	4	− 45.7184	424.27	77	− 45.7183	1
1	100	500	4	− 57.8119	779.78	65	− 57.8114	3
1	120	600	4	− 62.5722	1105.07	55	− 62.5711	4
1	140	700	4	− 76.6197	2187.67	65	− 76.6194	1
1	160	800	4	− 79.3598	2476.59	46	− 79.3564	3
1	180	900	4	− 94.3289	3516.91	47	− 94.3276	1
3	40	200	4	− 24.2392	218.06	76	− 24.2390	3
3	60	300	4	− 31.2141	739.74	82	− 31.2139	2
3	80	400	4	− 43.3003	1588.86	73	− 43.2999	4
3	100	500	4	− 52.0223	1896.64	46	− 52.0204	2
3	120	600	4	− 64.0476	3163.99	45	− 64.0466	3
5	40	200	4	− 24.3226	451.07	78	− 24.3226	1
5	60	300	4	− 33.8154	1545.14	82	− 33.8153	4
5	80	400	4	− 44.2731	2396.94	52	− 44.2718	4
1	40	200	5	− 23.9571	154.88	152	− 23.9569	2
1	60	300	5	− 34.2778	406.40	144	− 34.2777	4
1	80	400	5	− 43.3732	1194.10	191	− 43.3727	4
1	100	500	5	− 55.6727	1615.99	134	− 55.6721	4
1	120	600	5	− 64.6875	2444.75	112	− 64.6866	3
1	140	700	5	− 77.1353	2858.63	76	− 77.1305	3
3	40	200	5	− 22.4403	500.32	162	− 22.4402	3
3	60	300	5	− 33.2965	1397.35	150	− 33.2961	4
3	80	400	5	− 43.4613	2456.16	118	− 43.4603	4
3	100	500	5	− 57.8293	3524.89	79	− 57.8282	1
5	20	100	5	− 12.7221	163.44	188	− 12.7221	3
5	40	200	5	− 24.9583	1053.58	171	− 24.9583	3
5	60	300	5	− 30.4720	1926.13	99	− 30.4716	4

and the interference to primary link k is given by $\alpha_{k,S}|r_k^H H_{k,S} t_S|^2$, where $\alpha_{S,S}, \alpha_{S,k}$ and $\alpha_{k,S}$ respectively denote the path losses from the secondary transmitter to the secondary receiver, from the kth primary transmitter to the secondary receiver and from the secondary transmitter to the kth primary receiver, and N_0 is a given constant and r_S^H is the conjugate transpose of r_S. The problem is to find the optimal beamforming vectors t_S and r_S that maximize the SINR on the secondary link subject to the interference to primary link k less than a tolerable threshold ϵ_k. This results in the following optimization problem

$$\begin{aligned} \max_{t_S \in \mathbb{C}^{N_S}, r_S \in \mathbb{C}^{M_S}} \quad & \gamma_S \\ \text{s. t.} \quad & \alpha_{k,S} \left| r_k^H \mathbf{H}_{k,S} t_S \right|^2 \le \epsilon_k, \quad k = 1, \ldots, K, \\ & t_S^H t_S \le P_{S,\max}, \end{aligned} \tag{50}$$

where $P_{S,\max}$ is the maximum transmission power of the secondary link. As pointed out in [48], however, the secondary transmitter may not know the the channel state information on the links between primary receivers in practice. If the secondary transmitter knows the vector $\mathbf{H}_{k,S}^H r_k$ (called Scenario 1 in [48]), then the interference constraint can be rewritten as constraint $t_S^H Q_k^1 t_S \le 1$, where $Q_k^1 = \frac{\alpha_{k,S}}{\epsilon_k} \mathbf{H}_{k,S}^H r_k r_k^H \mathbf{H}_{k,S}$. If the secondary transmitter knows $\mathbf{H}_{k,S}$ but not r_k (called Scenario 2 in [48]), then the interference constraint can be replaced by

$$\Pr\left(\alpha_{k,S} \left| r_k^H \mathbf{H}_{k,S} t_S \right|^2 \le \epsilon_k \right) \le 1 - \delta_k, \quad k = 1, \ldots, K$$

for a given $\delta_k \in (0, 1)$. It was shown in [48] that if r_k has a normalized complex Gaussian distribution, then the above probabilistic constraint is equivalent to the constraints $t_S^H Q_k^2 t_S \le 1, k = 1, \ldots, K$, where $Q_k^2 = \frac{1-\delta_k^{1/(N_k-1)}}{\epsilon_k} \mathbf{H}_{k,S}^H \mathbf{H}_{k,S}$. On the other hand, as pointed out in [48], the objective function of problem (50) can be simplified as $\gamma_S = t_S^H A t_S$, where $A = \alpha_{S,S} \mathbf{H}_{S,S}^H \Phi^{-1} \mathbf{H}_{S,S}$ and $\Phi = \sum_{k=1}^K \alpha_{S,k} \mathbf{H}_{S,k} t_k t_k^H \mathbf{H}_{S,k}^H + N_0 I$. Based on these observations, problem (50) can be reformulated as the following M_S-dimensional complex QCQP problem

$$\begin{aligned} \max_{t_S \in \mathbb{C}^{M_S}} \quad & t_S^H A t_S \\ \text{s. t.} \quad & t_S^H Q_k t_S \le 1, \quad k = 1, \ldots, K, \\ & t_S^H t_S \le P_{S,\max}, \end{aligned} \tag{51}$$

where Q_k is equal to Q_k^1 and Q_k^2 in Scenarios 1 and 2, respectively. Note that both A and Q_k are Hermitian positive semidefinite matrices, thus problem (51) is NP-hard in general. In [48], authors tackled problem (51) by using SDP relaxation methods, and proposed a randomized algorithm based on SDP relaxation that finds a near-optimal solution to problem (51) when $K \ge 3$.

Next, we reformulate problem (51) as an equivalent concave QCQP with $2M_S$ real variables. Let $t_S = x + iy$ with $x, y \in \mathbb{R}^{M_S}$, $A = B + iC$ and $Q_k = B_k + iC_k$ with $B, C, B_k, C_k \in \mathbb{R}^{M_S \times M_S}$. Since both A and each Q_k are Hermitian, we have $B^T = B$, $C^T = -C$, $B_k^T = B_k$ and $C_k^T = -C_k$ for all k. It is then easy to check that $t_S^H t_S = (x - iy)^T (x + iy) = z^T z$, and

$$\begin{aligned} t_S^H A t_S &= x^T B x - x^T C y + y^T C x + y^T B y = z^T \hat{A} z, \\ t_S^H Q_k t_S &= x^T B_k x - x^T C_k y + y^T C_k x + y^T B_k y = z^T \hat{Q}_k z, \end{aligned}$$

where $z = \begin{pmatrix} x \\ y \end{pmatrix}$, $\hat{A} = \begin{pmatrix} B & -C \\ C & B \end{pmatrix}$ and $\hat{Q}_k = \begin{pmatrix} B_k & -C_k \\ C_k & B_k \end{pmatrix}$. Since A and each Q_k are Hermitian positive semidefinite matrices, we can see that $\hat{A}$ and $\hat{Q}_k$ are real symmetric and positive semidefinite matrices. Therefore, problem (51) can be reformulated as the following equivalent concave QCQP problem with $2M_S$ real variables:

$$\begin{aligned} \max_{z \in \mathbb{R}^{2M_S}} \quad & z^T \hat{A} z \\ \text{s. t.} \quad & z^T \hat{Q}_k z \le 1, \quad k = 1, \ldots, K, \\ & z^T z \le P_{S,\max}, \end{aligned} \tag{52}$$

which can be solved by the ADMBB algorithm. It should be pointed out that for problem (52), the constraint (23) in problem (22) can be replaced by

$$\sum_{i=1}^{r} \frac{s_i}{\lambda_i} \le P_{S,\max}.$$

Now, we simulate the network with one secondary link and K primary links. In our experiment, we consider two sets of links:

- Set 1: Each link has five transmit antennae and five receive antennae;
- Set 2: Each link has ten transmit antennae and ten receive antennae.

As in [48], we assume a Rayleigh fading and rich scattering environment, i.e., the entries of the channel matrices are independently and identically distributed complex Gaussian random variables with zero mean and unit variance. The parameters of $P_{S,\max}, \alpha_{S,S}, \alpha_{S,k}, \alpha_{k,S}, t_k$ and r_k are set in the same fashion as in Section VII of [48]. N_0 is set to 0.05, ϵ_k is set to 1, and δ_k is set to 1%. For each scenario, we generate 40 test instances of problem (52) with different number K of primary links. We use the ADMBB algorithm and the global optimization package BARON [36] to find the global optimal solution for test instances.

In Tables 8 and 9, we summarize the average numerical results of ADMBB and BARON for ten instances of problem (52) in Set 1 and Set 2, respectively, where K denotes the number of primary links, n denotes the number of variable, and r denotes the number of negative eigenvalues of the matrix $\hat{A}$. We see from Tables 8 and 9 that for each scenario, both ADMBB and BARON can find the global optimal solution for all test problems, while ADMBB is more effective than BARON. It is worth mentioning that our ADMBB algorithm is quite different from the algorithm in [48], which provides a high quality approximate solution, but the globally optimality can not be guaranteed. In comparison with BARON, the ADMBB algorithm can find a global solution in less CPU time. This indicates that the ADMBB algorithm is promising for real applications in telecommunication systems.

Table 8 Average performance of ADMBB versus BARON for ten instances of problem (52) in Set 1

Scenario	Size			ADMBB			BARON	
	K	n	r	Opt.val	CPU	Iter	Opt.val	CPU
1	10	10	10	−0.4886	0.29	16	−0.4886	0.80
	20	10	10	−0.4977	0.31	12	−0.4977	1.21
	30	10	10	−0.5278	0.45	16	−0.5278	1.39
	40	10	10	−0.4730	0.55	16	−0.4730	1.79
2	10	10	10	−0.6093	0.28	7	−0.6093	0.74
	20	10	10	−0.4766	0.67	20	−0.4766	1.06
	30	10	10	−0.4752	0.94	24	−0.4752	1.42
	40	10	10	−0.4572	1.81	42	−0.4572	1.77

Table 9 Average performance of ADMBB versus BARON for ten instances of problem (52) in Set 2

Scenario	Size			ADMBB			BARON	
	K	n	r	Opt.val	CPU	Iter	Opt.val	CPU
	10	20	20	−1.2751	0.77	32	−1.2751	1.99
1	20	20	20	−1.2089	1.02	35	−1.2089	3.54
	30	20	20	−1.2024	1.13	27	−1.2024	5.06
	40	20	20	−1.1680	1.34	29	−1.1680	6.84
	10	20	20	−1.1672	1.56	28	−1.1672	1.86
2	20	20	20	−1.1786	2.19	21	−1.1786	3.81
	30	20	20	−1.1650	7.26	67	−1.1650	5.16
	40	20	20	−1.1451	7.64	57	−1.1451	7.77

5.2 Numerical results of the GSA algorithm

In this subsection, we present numerical results of the GSA algorithm for the QP1NE problem. We consider three classes of randomly generated test problems: the clique problem, the rank-one quadratic nonconvex program, and the convex quadratically constrained QP1NE problem.

5.2.1 The clique problem

Given an undirected graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices, and E is the set of edges. For any integer k, the clique problem is to decide whether the graph has a subset of size k whose vertices are all connected to each other.

In [33], the clique problem was shown to be NP-hard, and can be reformulated as the following QP1NE problem:

$$\min \quad z - w^2$$

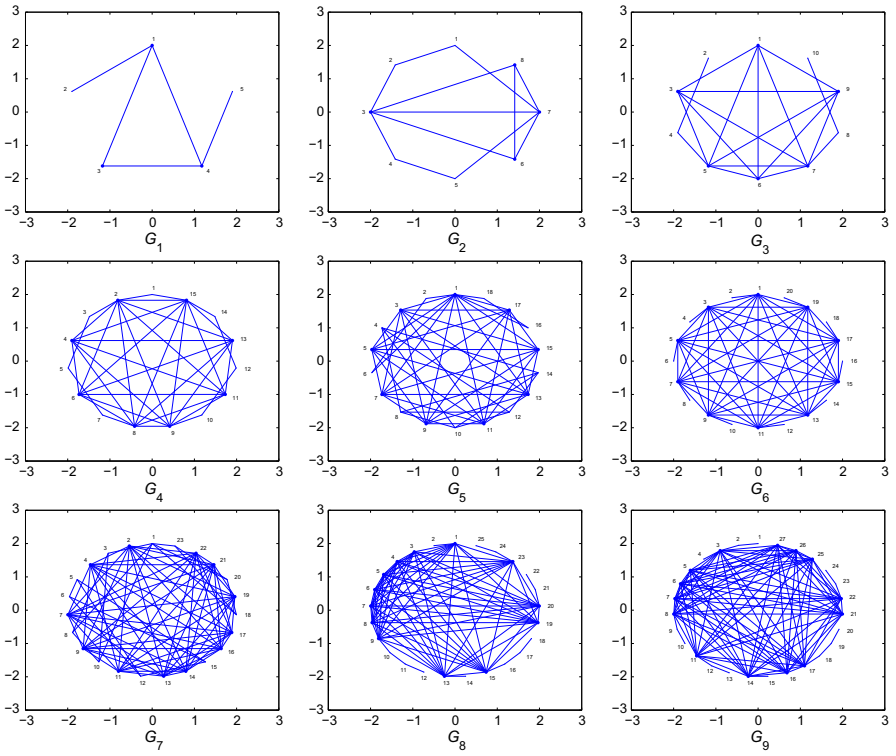

Fig. 2 Randomly generated graphs

$$
\begin{aligned}
s.t.\quad & w = \sum_{i=1}^{n} b^i x_i, \\
& z = \sum_{i=1}^{n} b^{2i} x_i + \sum_{1 \le i < j \le n} 2b^{i+j} y_{ij}, \\
& y_{ij} \ge x_i + x_j - 1, \quad y_{ij} \ge 0, \quad 1 \le i < j \le n, \\
& 0 \le x_i \le 1, \quad i = 1, \ldots, n, \\
& x_i + x_j \le 1, \quad (i, j) \notin E, \\
& \sum_{i=1}^{n} x_i = k,
\end{aligned} \tag{53}
$$

where $b = 4$. It was shown in [33] that the graph G has a clique of size k if and only if the global optimum of the problem (53) has a value zero.

To test the performance of the GSA algorithm on clique problems, we randomly generate 9 graphs. For illustration, we also depict these graphs in Fig. 2 where the corresponding clique in every graph is highlighted.

Table 10 Numerical results of GSA for random graphs

Case	n	k	GSA				
			Opt.val	CPU	Iter	T_{LSA}	Opt.solution
G_1	5	3	0.0000	0.08	1	0	$x^*_{G1} = (1, 0, 1, 1, 0)^T$
G_2	8	4	0.0000	0.08	2	2	$x^*_{G2} = (0, 0, 1, 0, 0, 1, 1, 1)^T$
G_3	10	6	0.0000	0.07	1	0	$x^*_{G3} = (1, 0, 1, 0, 1, 1, 1, 0, 1, 0)^T$
G_4	15	8	0.0000	0.08	1	0	x^*_{G4}
G_5	18	9	0.0000	0.09	2	2	x^*_{G5}
G_6	20	10	0.0000	0.04	3	4	x^*_{G6}
G_7	23	12	0.0000	0.08	1	0	x^*_{G7}
G_8	25	13	0.0000	0.09	1	0	x^*_{G8}
G_9	27	14	0.0000	0.09	1	0	x^*_{G9}

$x^*_{G4} = (0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1)^T$,
$x^*_{G5} = (1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1)^T$,
$x^*_{G6} = (1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1)^T$,
$x^*_{G7} = (0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0)^T$,
$x^*_{G8} = (1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0)^T$,
$x^*_{G9} = (0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1)^T$

The numerical results of the GSA for these randomly generated clique problems (53) are summarized in Table 10, where "opt.solution" denotes the optimal solution of problem (53) obtained by GSA. From Table 10, we see that given a graph $G = (V, E)$ and an integer k, a clique of size k can be effectively found by using the GSA to solve problem (53) associated with G and k when the size of the underlying graph is small ($n \leq 27$). However, for large scale graph, we had a numerical challenge in solving the linearized approximation of problem (53) caused by the extraordinarily large parameter values in the linearized model.

5.2.2 The rank-one nonconvex quadratic program

In this sub-subsection, we test the GSA algorithm on the following rank-one nonconvex quadratic programming problems (see [13]):

$$\begin{aligned} \min \quad & c_1^T x c_2^T x \\ \text{s. t.} \quad & x \in \mathcal{P} = \{x \in \mathbb{R}^n : Ax \leq b, \quad x \geq 0\}, \end{aligned} \tag{54}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $c_1, c_2 \in \mathbb{R}^n$. This problem is known to be NP-hard [27]. It is easy to check that problem (54) can be rewritten as the following equivalent problem

$$\min_{x\in\mathcal{P}} \frac{1}{4} x^T \left((c_1 + c_2)(c_1 + c_2)^T - (c_1 - c_2)(c_1 - c_2)^T \right) x,$$

which can be viewed as a specific case of the QP1NE problem and solved by the GSA or ADMBB algorithm.

To test the performance of the GSA algorithm, we randomly generate $c_1, c_2 \in \mathbb{R}^n$ and then normalize the random vectors such that $\|c_1\|_2 = \|c_2\|_2 = 1$. The constraint set is constructed by using the random procedure described in Sect. 5.1.4 without the quadratic term in the constraint functions. In Table 11, we compare the average performance of GSA and ADMBB for five random instances of problem (54) in the same size. As one can see from the table, the GSA algorithm is indeed more efficient than ADMBB in finding the global optimal solution for all test problems in terms of the CPU time. The average CPU time of GSA is much fewer than ADMBB. Moreover, ADMBB can not find the global solution within 3600 s for all test instances with $n \geq 4000$.

5.2.3 The quadratically constrained QP1NE problem

We finally test the GSA algorithm on generic convex quadratically constrained QP1NE problems:

$$\begin{aligned} \min \quad & x^T Q_+ x + q^T x - \left(c^T x\right)^2 \\ s.t. \quad & x^T Q_i x + q_i^T x \leq d_i, \quad i = 1, \ldots, m, \\ & Ax \leq b, \ x \in [0,1]^n, \end{aligned} \tag{55}$$

where Q_+ is an $n \times n$ symmetric positive semi-definite matrix and $c \in \mathbb{R}^n$ derived from the singular value decomposition of the $n \times n$ symmetric indefinite matrix Q with one negative eigenvalue, each Q_i is an $n \times n$ symmetric positive semidefinite matrix, $q, q_i \in \mathbb{R}^n$, $d_i \in \mathbb{R}$, $i = 1, \ldots, m$, $A \in \mathbb{R}^{l \times n}$, $b \in \mathbb{R}^l$. For numerical comparison, we also use the global optimization package BARON [36] to find the global optimal solution for some test problems in small scale.

The parameters in the test problems are randomly generated in the same fashion as in Sect. 5.1.4. The average numerical results for five test problems of the same size are summarized in Tables 12 and 13. In Table 12, we report comparison numerical results among GSA, ADMBB and BARON. One can see that for small size problem, ADMBB, GSA and BARON can find the global optimal solution. However, BARON usually takes much longer time than both ADMBB and GSA, while GSA takes less time than ADMBB. We also observe that the CPU time of BARON increases very fast as the size of the test problem grows. Moreover, for numerous test instances with $n = 20, 25$, BARON only reported the best solution obtained within 3600 s and failed to verify the global optimality of the obtained solution.

In Table 13, we report the testing results of the GSA and ADMBB methods for medium and large scale QP1NE instances. As one can see from Table 13, the GSA algorithm is able to effectively find the global optimal solution for all the test problems and takes much fewer time than ADMBB. Again, It_{ADM} is very small.

Table 11 Average performance of GSA versus ADMBB for five random instances of problem (54)

n_{lc}	n	ADMBB					GSA					
		Opt. val	CPU	Iter	Val_{ADM}	T_{ADM}	Opt. val	CPU	Iter	T_{LSA}	It_{ADM}	t_{ADM}
100	500	− 44.2743	2.93	11	− 44.2743	5	− 44.2743	0.65	1	0	3	0.32
200	1000	− 90.6546	25.37	13	− 90.6546	5	− 90.6546	2.04	1	0	3	1.22
300	1500	− 132.9847	54.46	11	− 132.9847	6	− 132.9847	5.27	1	0	3	3.27
400	2000	− 177.9142	231.83	11	− 177.9142	5	− 177.9142	12.26	1	0	3	8.11
500	2500	− 227.8830	343.54	14	− 227.8830	7	− 227.8830	18.47	1	0	3	10.81
600	3000	− 261.9717	711.57	14	− 261.9717	5	− 261.9717	34.08	1	0	3	21.52
700	3500	− 306.1330	1985.29	7	− 306.1330	3	− 306.1330	48.62	1	0	3	29.55
800	4000	–	–	–	–	–	− 339.8857	78.27	1	0	3	51.01
900	4500	–	–	–	–	–	− 375.4616	97.07	1	0	3	58.36
1000	5000	–	–	–	–	–	− 427.8434	166.09	1	0	3	116.86

"–" stands for the method failed to find the global solution within 3600 s at all cases

Table 12 Average performance of ADMBB & GSA versus BARON for five instances of problem (55)

n_{qc}	n_{lc}	n	ADMBB		GSA		BARON	
			Opt.val	CPU	Opt.val	CPU	Opt.val	CPU
0	10	5	− 22.9332	0.14	− 22.9332	0.13	− 22.9332	1.55
0	10	10	− 37.9158	0.19	− 37.9158	0.18	− 37.9158	3.74
0	10	15	− 112.6887	0.17	− 112.6887	0.13	− 112.6887	310.17
0	10	20	− 24.0924	0.17	− 24.0924	0.16	− 21.7404 (1)	1729.46
0	10	25	− 17.9142	0.16	− 17.9142	0.16	− 20.0023 (1)	1324.53
1	10	5	− 12.4054	0.23	− 12.4054	0.21	− 12.4054	1.12
1	10	10	− 21.3522	0.23	− 21.3522	0.19	− 21.3522	27.61
1	10	15	− 65.8701	0.33	− 65.8701	0.27	− 65.8701	74.93
1	10	20	− 16.7500	0.34	− 16.7500	0.28	− 22.4836 (3)	2219.44
1	10	25	− 46.7023	0.43	− 46.7023	0.37	− 56.1634 (2)	2293.63
3	10	5	− 40.0461	0.16	− 40.0461	0.15	− 40.0461	0.57
3	10	10	− 36.0863	0.26	− 36.0863	0.20	− 36.0863	23.96
3	10	15	− 20.9488	4.21	− 20.9488	0.34	− 20.9488	808.66
3	10	20	− 45.9831	0.41	− 45.9831	0.30	− 56.7575 (3)	2616.12
5	10	5	− 12.4860	0.18	− 12.4860	0.16	− 12.4860	2.75
5	10	10	− 33.9447	0.30	− 33.9447	0.23	− 33.9447	55.89
5	10	15	− 16.6016	0.41	− 16.6016	0.29	− 20.3717 (1)	1814.08
5	10	20	− 47.4871	0.54	− 47.4871	0.34	− 53.3468 (2)	2105.74

The number in parentheses stands for the number of instances for which BARON can not verify the global optimality of the solution within 3600 s

6 Conclusions

In this paper, we considered the QCQP problem with a few negative eigenvalues that arises from various disciplines and is known to be NP-hard. By combining the classical alternative direction method, modern convex relaxation, initialization and branch-and-bound technique, we developed the ADMBB algorithm to find a global optimal solution to the underlying QP. Especially, without resorting to the B&B framework, we also developed a new algorithm (GSA) to find a global optimal solution to the QP1NE problem by integrating ADM, convex relaxation and new line search technique. Under mild conditions, we established the global convergence of both ADMBB and GSA and estimated their complexity. Preliminary experiments illustrated that both ADMBB and GSA can effectively find the global optimal solution for all test problems described in this paper. But for the large-scale QP1NE instances, GSA outperforms ADMBB in terms of the CPU time. We also observed that though the theoretical worst-case complexity of the GSA algorithm is slightly worse than that of ADMBB for the QP1NE problem, the practical complexity of the GSA algorithm is almost constant, which is much better than its worst-case estimate. We also noted that the complexity of ADMBB grows exponentially in terms of the number of negative eigenvalues of the Hessian matrix in the objective function. This indicates that ADMBB may not be effective for

Table 13 Average performance of GSA versus ADMBB for five random instances of problem (55)

n_{qc}	n_{lc}	n	ADMBB					GSA					
			Opt. val	CPU	Iter	Val_{ADM}	T_{ADM}	Opt. val	CPU	Iter	T_{LSA}	It_{ADM}	t_{ADM}
0	100	500	− 54.7096	5.74	10	− 54.7096	4	− 54.7096	3.27	0	0	2	2.37
0	200	1000	− 105.6664	32.20	11	− 105.6664	3	− 105.6664	26.53	1	0	3	21.24
0	300	1500	− 157.1839	113.53	10	− 157.1839	5	− 157.1839	75.65	0	0	2	60.62
0	400	2000	− 216.2709	245.17	10	− 216.2709	6	− 216.2709	131.56	0	0	2	103.25
0	500	2500	− 259.9254	488.33	12	− 259.9254	5	− 259.9254	382.26	1	0	3	287.29
0	600	3000	− 310.8992	917.58	11	− 310.8992	6	− 310.8992	559.30	0	0	2	456.41
0	700	3500	− 362.4223	1338.05	11	− 362.4223	5	− 362.4223	897.58	0	0	2	693.33
0	800	4000	− 427.7947	2082.56	10	− 427.7947	8	− 427.7947	1048.83	0	0	2	767.09
1	100	500	− 51.6040	74.89	5	− 51.6036	1	− 51.6037	21.11	0	0	2	17.26
1	200	1000	− 102.0536	560.01	5	− 102.0521	1	− 102.0528	137.35	1	0	3	102.96
1	300	1500	− 159.7015	2347.81	5	− 159.6988	1	− 159.7011	507.97	0	0	2	425.64
1	400	2000	–	–	–	–	–	− 214.2295	1150.12	0	0	2	953.24
1	500	2500	–	–	–	–	–	− 261.7671	2297.79	0	0	2	1788.06
3	100	500	− 53.8077	298.75	7	− 53.8067	3	− 53.8069	53.08	0	0	2	43.45
3	120	600	− 62.6677	395.22	6	− 62.6674	1	− 62.6676	96.37	1	0	3	77.58
3	140	700	− 73.8259	753.71	5	− 73.8248	2	− 73.8254	195.24	1	0	3	141.85
3	160	800	− 84.4212	1267.87	7	− 84.4198	1	− 84.4207	271.23	1	0	3	227.22
3	180	900	− 94.0743	1625.77	6	− 94.0735	1	− 94.0737	349.74	1	0	3	282.43
3	200	1000	− 107.3750	2435.92	6	− 107.3730	2	− 107.3734	600.77	1	0	3	481.95
5	100	500	− 55.4183	515.84	6	− 55.4176	3	− 55.4178	91.74	0	0	2	75.86
5	120	600	− 61.1282	1202.09	6	− 61.1263	3	− 61.1273	186.21	0	0	2	158.81
5	140	700	− 75.3344	1803.62	6	− 75.3336	1	− 75.3341	384.97	1	0	3	277.87
5	160	800	− 81.3586	2334.55	5	− 81.3583	1	− 81.3583	533.25	0	0	2	455.62

Table 13 continued

n_{qc}	n_{lc}	n	ADMBB					GSA					
			Opt. val	CPU	Iter	Val_{ADM}	T_{ADM}	Opt. val	CPU	Iter	T_{LSA}	It_{ADM}	t_{ADM}
5	180	900	− 94.5335	3178.93	5	− 94.5312	1	− 94.5322	599.53	0	0	2	485.69
5	200	1000	–	–	–	–	–	− 110.0029	1418.33	1	0	3	1021.15

"–" stands for the method fails to find the global solution within 3600 s at all cases

QCQP instances with many negative eigenvalues. One interesting direction for future research is to study whether we can develop effective global algorithms for generic QPs based on NADM and other branching and bound or local search techniques. Another direction is to investigate whether similar global approaches can be developed for generic quadratic programming problems with non-convex quadratic constraints.

Acknowledgements We would like to thank all the anonymous reviewers and the associate editor for their useful suggestions that has helped to substantially improve the presentation of this work.

References

1. Anjos, M .F., Lasserre, J .B.: Handbook of Semidefinite, Conic and Polynomial Optimization: Theory, Algorithms, Software and Applications. International Series in Operational Research and Management Science, p. 166. Springer, Berlin (2001)
2. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn. **3**(1), 1–122 (2010)
3. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
4. Burer, S., Vandenbussche, D.: A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations. Math. Program. **113**(2), 259–282 (2008)
5. Burer, S., Vandenbussche, D.: Globally solving box-constrained nonconvex quadratic programs with semidefinite-based finite branch-and-bound. Comput. Optim. Appl. **43**(2), 181–195 (2009)
6. Cambini, R., Salvi, F.: A branch and reduce approach for solving a class of low rank d.c. programs. J. Comput. Appl. Math. **233**, 492–501 (2009)
7. Cambini, R., Salvi, F.: Solving a class of low rank d.c. programs via a branch and bound approach: a computational experience. Oper. Res. Lett. **38**(5), 354–357 (2010)
8. Cambini, R., Sodini, C.: A finite algorithm for a particular d.c. quadratic programming problem. Ann. Oper. Res. **117**(1), 33–49 (2002)
9. Chen, J., Burer, S.: Globally solving nonconvex quadratic programming problems via completely positive programming. Math. Program. Comput. **4**, 33–52 (2012)
10. Floudas, C .A., Visweswaran, V.: Quadratic optimization. In: Horst, R., Pardalos, P.M. (eds.) Handbook of Global Optimization, pp. 217–270. Kluwer Academic Publishers, Boston (1994)
11. Gorski, J., Pfeuffer, F., Klamroth, K.: Biconvex sets and optimization with biconvex functions: a survey and extensions. Math. Methods Oper. Res. **66**, 373–407 (2007)
12. Gould, N .I .M., Toint, P .L.: Numerical methods for large-scale non-convex quadratic programming. In: Siddiqi, A.H., Kocvara, M. (eds.) Trends in Industrial and Applied Mathematics. Applied Optimization, vol. 72, pp. 149–179. Springer, Boston (2002)
13. Goyal, V., Genc-Kaya, L., Ravi, R.: An FPTAS for minimizing the product of two non-negative linear cost functions. Math. Program. **126**, 401–405 (2011)
14. Horst, R., Thoai, N.V.: DC programming: overview. J. Optim. Theory Appl. **103**(1), 1–43 (1999)
15. IBM ILOG CPLEX. IBM ILOG CPLEX 12.3 User's Manual for CPLEX, 89 (2011)
16. Kim, S., Kojima, M.: Exact solutions of some nonconvex quadratic optimization problems via SDP and SOCP relaxations. Comput. Optim. Appl. **26**, 143–154 (2003)
17. Kojima, M., Tunçel, L.: Cones of matrices and successive convex relaxations of nonconvex sets. SIAM J. Optim. **10**, 750–778 (2000)
18. Kojima, M., Tunçel, L.: Discretization and localization in successive convex relaxation methods for nonconvex quadratic optimization. Math. Program. **89**(1), 79–111 (2000)
19. Konno, H.: A cutting plane algorithm for solving bilinear programs. Math. Program. **11**, 14–27 (1976)
20. Lasserre, J.: Global optimization with polynomials and the problem of moments. SIAM J. Optim. **11**(3), 796–817 (2001)
21. Laurent, M.: A comparison of the sherali-adams, lovász-schrijver, and lasserre relaxations for 0–1 programming. Math. Oper. Res. **28**(3), 470–496 (2003)
22. Le Thi, H.A., Pham, T.: Dinh. Solving a class of linearly constrained indefinite quadratic problems by D.C. algorithms. J. Glob. Optim. **11**, 253–285 (1997)

23. Le Thi, H.A., Pham Dinh, T.: A branch and bound method via d.c. optimization algorithms and ellipsoidal technique for box constrained nonconvex quadratic problems. J. Glob. Optim. **13**, 171–206 (1998)
24. Le Thi, H.A.: An efficient algorithm for globally minimizing a quadratic function under convex quadratic constraints. Math. Program. Ser. A **87**(3), 401–426 (2000)
25. Loridan, P.: Necessary conditions for ϵ-optimality. Math. Program. Stud. **19**, 140–152 (1982)
26. Lovász, L., Schrijver, A.: Cones of matrices and set-functions and 0–1 optimization. SIAM J. Optim. **2**(1), 166–190 (1991)
27. Matsui, T.: NP-hardness of linear multiplicative programming and related problems. J. Glob. Optim. **9**(2), 113–119 (1996)
28. Nie, J.: Optimality conditions and finite convergence of lasserre's hierarchy. Math. Program. **146**(1–2), 97–121 (2014)
29. Palacios-Gomez, F., Lasdon, L., Enquist, M.: Nonlinear optimization by successive linear programming. Manag. Sci. **28**(10), 1106–1120 (1982)
30. Pardalos, P.M., Rodgers, G.: Computational aspects of a branch and bound algorithm for quadratic zero-one programming. Computing **45**, 131–144 (1990)
31. Pardalos, P.M.: Global optimization algorithms for linearly constrained indefinite quadratic problems. Comput. Math. Appl. **21**, 87–97 (1991)
32. Pardalos, P.M., Schnitger, G.: Checking local optimality in constrained quadratic programming is NP-hard. Oper. Res. Lett. **7**(1), 33–35 (1988)
33. Pardalos, P.M., Vavasis, S.A.: Quadratic programming with one negative eigenvalue is NP-hard. J. Glob. Optim. **1**, 15–22 (1991)
34. Peng, J., Zhu, T.: A nonlinear semidefinite programming approch for the worst-case linear optimization under uncertainties. Math. Program. **152**(1), 593–614 (2015)
35. Pham Dinh, T., Le Thi, H. A.: Recent advances in DC programming and DCA. In: Transactions on Computational Intelligence XIII, pp. 1–37. Springer, Berlin (2014)
36. Sahinidis, N.V.: BARON: a general purpose global optimization software package. J. Glob. Optim. **8**, 201–205 (1996)
37. Saxena, A., Bonami, P., Lee, J.: Convex relaxation of non-convex mixed integer quadratically constrained programs: extended formulations. Math. Program. Ser. B **124**, 383–411 (2010)
38. Saxena, A., Bonami, P., Lee, J.: Convex relaxation of nonconvex mixed integer quadratically constrained programs: projected formulations. Math. Program. Ser. A **130**, 359–413 (2011)
39. Sherali, H., Adams, W.: A hierarchy of relaxation between the continuous and convex hull representations. SIAM J. Discret. Math. **3**, 411–430 (1990)
40. Sherali, H.D., Tuncbilek, C.H.: A reformulation-convexification approach for solving nonconvex quadratic programming problems. J. Global Optim. **7**(1), 1–31 (1995)
41. Vandenbussche, D., Nemhauser, G.: A branch-and-cut algorithm for nonconvex quadratic programming with box constraints. Math. Program. **102**, 559–575 (2005)
42. Vandenbussche, D., Nemhauser, G.: A polyhedral study of nonconvex quadratic programs with box constraints. Math. Program. **102**(3), 531–557 (2005)
43. Vavasis, S.A.: Approximation algorithms for indefinite quadratic programming. Math. Program. **57**, 279–311 (1992)
44. Visweswaran, V., Floudas, C.: New properties and computational improvement of the GOP algorithm for problems with quadratic objective function and constraints. J. Glob. Optim. **3**(3), 439–462 (1993)
45. Wolkowicz, H., Saigal, R., Vandenberghe, L.: Handbook of Semidefinite Programming. Kluwer Academic Publishers, Boston (2000)
46. Ye, Y.: On the complexity of approximating a KKT point of quadratic programming. Math. Program. **80**, 195–211 (1998)
47. Zhang, S.: Quadratic maximization and semidefinite relaxation. Math. Program. **87**, 453–465 (2000)
48. Zhang, Y.J., So, A.M.-C.: Optimal spectrum sharing in MIMO cognitive radio networks via semidefinite programming. IEEE J. Sel. Areas Commun. **29**, 362–373 (2011)

Affiliations

Hezhi Luo[1] · Xiaodi Bai[2] · Gino Lim[3] · Jiming Peng[3]

Hezhi Luo
hzluo@zjut.edu.cn

Xiaodi Bai
xdbai@zjut.edu.cn

Gino Lim
ginolim@uh.edu

1 Department of Management Science & Engineering, College of Economics and Management, Zhejiang University of Technology, Hangzhou 310032, Zhejiang, P. R. China

2 Department of Applied Mathematics, College of Science, Zhejiang University of Technology, Hangzhou 310032, Zhejiang, P. R. China

3 Department of Industrial Engineering, University of Houston, Houston, TX 77204, USA