Quadratic Optimization Through the Lens of Adjustable Robust Optimization

Abbas Khademi
School of Mathematics, Statistics and Computer Science, College of Science, University of Tehran, Tehran, Iran, abbaskhademi@ut.ac.ir, abbaskhademi92@gmail.com

Ahmadreza Marandi
Department of Industrial Engineering and Innovation Sciences, Eindhoven University of Technology, Eindhoven, The Netherlands, a.marandi@tue.nl
Faculty of Management, University of British Columbia, Kelowna, BC, Canada,
Eindhoven Artificial Intelligence Systems Institute, Eindhoven, The Netherlands,

Abstract. Quadratic optimization (QO) has been studied extensively in the literature due to its applicability in many practical problems. While practical, it is known that QO problems are generally NP-hard. So, researchers developed many approximation methods to find good solutions. In this paper, we analyze QO problems using robust optimization techniques. To this end, we first show that any QO problem can be reformulated as a disjoint bi-convex QO problem. Then, we provide an equivalent adjustable robust optimization (ARO) reformulation and leverage the methods available in the literature on ARO to approximate this reformulation. More specifically, we show that using a so-called decision rule technique to approximate the ARO reformulation is equivalent to using a reformulation-linearization technique on the original QO problem. Additionally, we design an algorithm that can find a close-to-optimal solution based on our new reformulations. Our numerical results demonstrate the efficiency of our algorithm to find near-optimal solutions, particularly for large-sized instances, compared with off-the-shelf solvers and state-of-art approaches.

Key words: Quadratic Optimization, Adjustable Robust Optimization, Duality, Affine Decision Rule, Reformulation-Linearization Technique

1. Introduction

Various practical problems in different domains, including financial mathematics (Markowitz 1952), machine learning (Cevikalp and Polikar 2008), resource allocation (Ibaraki and Katoh 1988), computer vision (Bhanja et al. 2016), game theory (Bomze 2002b), robotic systems (Khadivar et al. 2023), graph theory (Gibbons et al. 1997), and image processing (Bulo et al. 2011), to mention few, can be formulated as quadratic optimization problems. Thus, developing efficient techniques to solve general quadratic optimization problems is of great importance.

Let us consider a quadratic optimization (QO) problem of the form:

$$\min_{x\in\mathcal{X}} \; x^\top Q x + c^\top x, \tag{QO}$$

where $\mathcal{X}\subseteq\mathbb{R}^{n_x}$ is a nonempty convex set, $Q\in\mathbb{R}^{n_x\times n_x}$ is a real matrix, and $c\in\mathbb{R}^{n_x}$ is a real vector. Without loss of generality, we assume that Q is a symmetric matrix. If Q is a positive semi-definite matrix, we have a convex QO, which is solvable in polynomial time (Kozlov et al. 1980, Renegar 2001). In contrast,

even when Q has only one negative eigenvalue, (QO) is NP-hard (Pardalos and Vavasis 1991). Besides, identifying local minimizers of (QO) over a polyhedron is not simpler than finding global minimizers from a complexity perspective (Ahmadi and Zhang 2022).

Due to the NP-hardness of indefinite QO problems, there has been a lot of research on constructing upper bounds via finding "good" solutions (Bentobache et al. 2022, Cuong et al. 2024), and lower bounds to identify the quality of a candidate solution, which are mainly based on linear or conic approximations (Mitchell et al. 2014, Rostami et al. 2023, Zamani 2023). A customary way to approximate a QO problem is by relaxing it into linear optimization problems, which is achieved through Reformulation-Linearization Techniques (RLT) (Anstreicher 2009, Sherali and Tuncbilek 1995). For an overview of RLTs, we refer the reader to the chapter (Sherali and Liberti 2009) and the references therein.

Among the conic relaxations, copositive relaxations have been considered the most powerful as it was shown that they result in tight bounds (Bomze 2015, Burer 2009). In such relaxations, the primary computational challenge shifts to deal with the copositive cone using tractable inner and outer approximations (Bundfuss and Dur 2009, Gouveia et al. 2020, Kim et al. 2020), or use a KKT-based branch-and-bound method (Chen and Burer 2012).

Another important conic relaxation for QO problems is the positive semi-definite relaxations. In the last thirty years, the field of semi-definite optimization (SDO) has undergone significant and swift advancement (Wolkowicz et al. 2012). Due to their efficiency, the SDO framework has led to many semi-definite relaxations; these relaxations are reviewed and compared in (Bao et al. 2011, Wang and Kılınç-Karzan 2022, Zheng et al. 2011). Moreover, Burer and Vandenbussche (2008, 2009) develop branch-and-bound approaches based on semi-definite relaxations to solve a QO problem.

In addition to directly approximating QO problems, a research direction is to reformulate them into other well-studied problems. Hu et al. (2012) and Xia et al. (2020) show how a QO problem is reformulated as a mixed-integer linear optimization (MILO) problem. Furthermore, it is possible to transform an indefinite QO problem with quadratic constraints into a bi-linear problem with additional variables and constraints (Hansen and Jaumard 1992). Moreover, since any quadratic function can be written as the difference between two convex quadratic functions, a QO can be reformulated as a difference-of-convex (DC) optimization problem (Anstreicher and Burer 2005, Fampa et al. 2017). For a review of DC optimization, we refer the reader to Horst and Thoai (1999), Lipp and Boyd (2016).

Next to methods developed for general QO problems, there are techniques to solve or approximate special classes. One class is when the matrix Q has a few negative eigenvalues. In Cen and Xia (2021), the authors propose a solution scheme that involves solving a series of convex QO problems over the original feasible region. Additionally, Luo et al. (2019) introduces an alternative direction-based method to solve QO problems in this class.

Another class is standard QO problems, where the feasible region is the unit simplex. For more details on lower bound approximations for this class of QO problems, we refer the reader to Bomze and De Klerk (2002), Bomze et al. (2008), Bonami et al. (2019), Gökmen and Yıldırım (2022), Gondzio and Yıldırım (2021), Selvi et al. (2023).

In this paper, we focus on the relation between QO problems and adjustable robust optimization problems. The adjustable robust optimization (ARO) framework, initially introduced in Ben-Tal et al. (2004), has gained significant attention among researchers due to its ability to handle decision-making problems in the presence of uncertain parameters. This approach involves adaptive decision-making by considering two types of decision variables: static and adjustable decisions. Static (or 'here-and-now') decisions are made based on available information, while adjustable (or 'wait-and-see') decisions are made in response to the actual values of uncertain parameters. In recent years, the ARO framework has been successfully applied to tackle complex optimization problems such as convex maximization, bi-linear optimization, and convex-non-convex quadratic optimization (Selvi et al. 2022, Zhen et al. 2022, Bomze and Gabl 2021).

There are algorithms and approaches developed in the literature to solve an ARO problem exactly. Fourier-Motzkin elimination is such an approach that can be used in ARO problems with fixed recourse (Zhen et al. 2018). If all adjustable decision variables are eliminated, a static solution can be obtained by solving the robust counterpart. Even if not all adjustable decision variables can be eliminated, the resulting exact reformulation is likely to lead to stronger bounds after applying approximations based on the mentioned techniques. Furthermore, there are iterative approaches to solve such problems exactly either by partitioning the uncertainty set (Bertsimas and Dunning 2016, Postek and Hertog 2016) or applying piecewise decision rules (Thomä et al. 2024).

To obtain an approximate solution for an ARO problem, various techniques, such as the finite scenario approach (Hadjiyiannis et al. 2011), finite adaptability approach (Bertsimas and Caramanis 2010), and decision rules (El Housni and Goyal 2021) can be employed, particularly in the case of linear ARO problems. Using these methods, one can estimate the optimal value or obtain an approximated solution for the original problem. For more information on ARO, we refer to the tutorial by Delage and Iancu (2015) and the survey paper by Yanıkoğlu et al. (2019).

While there has been a lot of research in approximating linear ARO problems, the literature sparsely covers non-linear ARO problems due to their inherent complexity. De Ruiter et al. (2023) shows a class of non-linear ARO problems featuring a polyhedral uncertainty set that can be transformed into an equivalent linear ARO problem, thereby enabling the application of the existing approximations techniques for linear cases. In a recent study, Khademi et al. (2024) employed Fenchel's duality to convert a non-linear ARO problem into its dual formulation and introduce a cutting-plane algorithm to find locally robust solutions.

In this paper, we make a four-fold contribution to the literature to connect the two fields of quadratic optimization and adjustable robust optimization. First, we show that any QO problem can be reformulated

as a disjoint bi-convex quadratic optimization problem. Using this new reformulation, we further show that any QO problem can be reformulated as an ARO problem, where the objective functions and constraints are convex quadratic in the decision variables This structure allows us to employ customary techniques to relax the ARO reformulation, thereby enabling us to derive a near-optimal solution to the original problem.

Second, we show how one can interpret an approximation of the ARO reformulation on the original QO problem. More specifically, we prove that applying a structured affine decision rule to approximate the ARO formulation is equivalent to applying an RLT to approximate the original problem.

Third, we design an algorithm to construct a bound on the optimal value of (QO). More specifically, we apply a decision-rule approximation to obtain a lower bound. Then, based on the solution and the structure of the ARO problem, we construct "good" feasible solutions. In the final step, we apply the mountain-climbing procedure to improve the quality of the solution.

Finally, we conduct an extensive numerical experiment to illustrate the efficiency of our algorithm in obtaining near-optimal solutions. Based on the numerical results, we see that the solution obtained from the algorithm is close to optimum and, in most cases, has an optimality gap of 1% for concave QO problems (the optimality gap is measured as the percentage relative gap between lower and upper bounds on the optimal value). Regarding speed, our algorithm is computationally efficient and significantly outperforms the available off-the-shelf solvers and state-of-the-art approaches for large-sized instances.

The rest of the paper is structured as follows: in Section 1.1, we define the notation used throughout the paper. Section 2 introduces the reformulation of a QO problem as a bi-convex optimization problem and outlines its equivalent ARO problem. In Section 3, we approximate this problem using available techniques and prove the equivalence to an RLT for the original QO problem. Subsequently, in Section 4, we design an algorithm that provides a near-optimal solution for a QO problem using the ARO reformulation. Section 5 presents numerical results, demonstrating the efficiency of our ARO-based algorithm, particularly for large-sized instances. Finally, in Section 6, we summarize our findings and present our conclusions.

1.1. Notation

In this section, we introduce notations used in the paper. For a symmetric matrix B, we use $B \succeq 0$ ($B \succ 0$) to show B is positive semi-definite (positive definite), i.e., it has non-negative (positive) eigenvalues. The smallest and largest eigenvalues of a symmetric matrix B are denoted by $\lambda_{min}(B)$ and $\lambda_{max}(B)$, respectively. For a given matrix B, and indices i and j, we denote by B_i, B^j, and B_{ij}, the i-th row, the j-th column, and the ij-th entry of B, respectively. For a matrix B, vec(B) denotes the vector formed by concatenating all of the rows of the matrix B. For a square matrix B, diag(B) is a vector containing the main diagonal elements of the matrix B. For a vector v, Diag(v) is a matrix with the elements of the vector v placed on its main diagonal, and zeros elsewhere. The notation $B \geq 0$ for a matrix B indicates that all its elements are non-negative. We use $\mathcal{S}^n$ to refer to the space of $n \times n$ symmetric matrices. We use $(\cdot)^\top$ to

refer to the transpose operator for both matrices and vectors. We denote the $n \times n$ identity matrix by I_n, the vector of all ones by e, and the i-th unit vector by e_i. To avoid overcomplicating notation, we do not specify the dimensions of e and e_i but make sure they are always evident from the context. We misuse the notation and denote the real number zero, the vector of all zeroes, and the matrix of all zeroes by 0.

We use $\mathbb{R}^n$ to refer to the n-dimensional real-valued Euclidean space, where $\|\cdot\|_2$ is the Euclidean norm. The standard or unit simplex in $\mathbb{R}^n$, given by $\{x \in \mathbb{R}^n : e^\top x = 1, x \geq 0\}$, is denoted by Δ.

2. New Reformulations for Quadratic Optimization Problems

This section proposes two reformulations for a quadratic optimization problem (QO). We first show how we can reformulate (QO) as a disjoint bi-convex quadratic optimization problem. Using this reformulation, we further provide an equivalent adjustable robust optimization problem. So, we start with the following theorem.

THEOREM 1. *Let $Q^+, -Q^- \succeq 0$, and $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ be an arbitrary set. Then,*

$$\min_{x \in \mathcal{X}} \; x^\top (Q^+ + Q^-)x + c^\top x \tag{1}$$

is equivalent to

$$\min_{x,y \in \mathbb{R}^{n_x}} \left\{ \tfrac{1}{2}x^\top Q^+ x + \tfrac{1}{2}y^\top Q^+ y + x^\top Q^- y + \tfrac{1}{2}c^\top x + \tfrac{1}{2}c^\top y : \; x, y \in \mathcal{X} \right\}. \tag{Bi-QO}$$

Proof. It is clear that

$$\begin{aligned}
&\min_{x \in \mathcal{X}} x^\top (Q^+ + Q^-)x + c^\top x \\
&\quad = \min_{x,y \in \mathbb{R}^{n_x}} \left\{ \tfrac{1}{2}x^\top Q^+ x + \tfrac{1}{2}y^\top Q^+ y + x^\top Q^- y + \tfrac{1}{2}c^\top x + \tfrac{1}{2}c^\top y : \; x = y, \; x, y \in \mathcal{X} \right\} \\
&\quad \geq \min_{x,y \in \mathbb{R}^{n_x}} \left\{ \tfrac{1}{2}x^\top Q^+ x + \tfrac{1}{2}y^\top Q^+ y + x^\top Q^- y + \tfrac{1}{2}c^\top x + \tfrac{1}{2}c^\top y : \; x, y \in \mathcal{X} \right\},
\end{aligned}$$

where the inequality is due to the fact that the feasible region of the last optimization problem is contained in the feasible region of the middle optimization problem.

To show "$\leq$", we use the negative semi-definiteness of Q^-. For any given x and y in $\mathbb{R}^{n_x}$, because $Q^- \preceq 0$, we have $(x-y)^\top Q^-(x-y) \leq 0$. Hence, $x^\top Q^- x + y^\top Q^- y \leq 2x^\top Q^- y$. This implies that for any $x, y \in \mathbb{R}^{n_x}$,

$$x^\top (Q^+ + Q^-)x + y^\top (Q^+ + Q^-)y \leq x^\top Q^+ x + y^\top Q^+ y + 2x^\top Q^- y.$$

So,

$$x^\top (Q^+ + Q^-)x + y^\top (Q^+ + Q^-)y + c^\top x + c^\top y \leq x^\top Q^+ x + y^\top Q^+ y + 2x^\top Q^- y + c^\top x + c^\top y.$$

Now, by taking the minimum over $x, y \in \mathcal{X}$, we have

$$\begin{aligned}\min_{x\in\mathcal{X}} \left\{x^\top (Q^+ + Q^-)x + c^\top x\right\} + \min_{y\in\mathcal{X}} \left\{y^\top (Q^+ + Q^-)y + c^\top y\right\} \\ \leq \min_{x,y\in\mathcal{X}} \left\{x^\top Q^+ x + y^\top Q^+ y + 2x^\top Q^- y + c^\top x + c^\top y\right\}.\end{aligned}$$

The fact that

$$\min_{x\in\mathcal{X}} \left\{x^\top (Q^+ + Q^-)x + c^\top x\right\} = \min_{y\in\mathcal{X}} \left\{y^\top (Q^+ + Q^-)y + c^\top y\right\},$$

completes the proof. □

It is worth noting that the proof of Theorem 1 does not rely on the specific structure of the feasible set $\mathcal{X}$. If $\mathcal{X}$ is convex, then the proposition asserts that any indefinite QO can be reformulated as a disjoint bi-convex quadratic optimization problem, where the variables x and y are linked only in the objective function. Furthermore, it can be demonstrated that if x^* is an optimal solution for (QO), then the pair (x^*, x^*) is an optimal solution for (Bi-QO). Conversely, when $Q^+ \succeq 0$ and $-Q^- \succ 0$, if $(\hat{x}, \hat{y})$ is an optimal pair for (Bi-QO), then both $\hat{x}$ and $\hat{y}$ are optimal solutions for (QO), and we have $\hat{x} = \hat{y}$. For more details, see Appendix A.

From now on, let us restrict the feasible region of (QO) to polytopes i.e., $\mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid Ax = b,\ x \geq 0\}$ for some $A \in \mathbb{R}^{m_x \times n_x}$ and $b \in \mathbb{R}^{m_x}$, so that $\mathcal{X}$ is compact. In the next theorem, we show that we can reformulate (QO) problem to an adjustable robust optimization problem.

THEOREM 2. *Let $Q = Q^+ + Q^-$ where $Q \in \mathbb{R}^{n_x \times n_x}$, and $Q^+, -Q^- \succeq 0$. Assume that $\mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid Ax = b,\ x \geq 0\}$ is non-empty compact. Then, the optimal value of (QO) is equal to the optimal value of the following problem:*

$$\begin{aligned}&\max_{\tau\in\mathbb{R}} \ \tau \\ &\text{s.t. } \forall x \in \mathcal{X},\ \exists (u_x, w_x): \begin{cases} \frac{1}{2}x^\top Q^+ x + \frac{1}{2}c^\top x - \frac{1}{2}u_x^\top Q^+ u_x + b^\top w_x \geq \tau, \\ A^\top w_x - Q^+ u_x \leq Q^- x + \frac{1}{2}c. \end{cases}\end{aligned} \tag{ARO-QO}$$

Proof. Based on the assumption, we have that (QO) is equivalent to

$$\min_{x\in\mathcal{X}} x^\top (Q^+ + Q^-)x + c^\top x,$$

which is, using Theorem 1, equivalent to

$$\min_{x,y\in\mathbb{R}^{n_x}} \left\{\tfrac{1}{2}x^\top Q^+ x + \tfrac{1}{2}y^\top Q^+ y + x^\top Q^- y + \tfrac{1}{2}c^\top x + \tfrac{1}{2}c^\top y :\ x, y \in \mathcal{X}\right\}. \tag{2}$$

We can write (2) as

$$\min_{x\in\mathcal{X}} \left\{\tfrac{1}{2}x^\top Q^+ x + \tfrac{1}{2}c^\top x + \min_{y\in\mathcal{X}} \tfrac{1}{2}y^\top Q^+ y + x^\top Q^- y + \tfrac{1}{2}c^\top y\right\}. \tag{3}$$

We consider the inner minimization problem over y for a given $x \in \mathcal{X}$. Since $\mathcal{X}$ non-empty compact, we can apply Dorn duality (Dorn 1960), and rewrite (3) as follows:

$$\begin{aligned} \min_{x\in\mathcal{X}} \tfrac{1}{2}x^\top Q^+ x + \tfrac{1}{2}c^\top x + \max_{u_x,w_x} & -\tfrac{1}{2}u_x^\top Q^+ u_x + b^\top w_x \\ \text{s.t. } & A^\top w_x - Q^+ u_x \le Q^- x + \tfrac{1}{2}c. \end{aligned} \tag{4}$$

Let $x \in \mathcal{X}$. If the inner maximization is infeasible, its optimal value is $-\infty$, implying that (4) is unbounded. So, in this case, (QO) is unbounded, which contradicts the compactness of $\mathcal{X}$. So, for any $x \in \mathcal{X}$, there is a feasible (u_x, w_x) for the inner maximization. Thus, using the epigraph reformulation of the objective function, we can rewrite (4) as

$$\max_{\tau} \left\{ \tau \;\middle|\; \forall x \in \mathcal{X},\ \exists (u_x, w_x): \begin{array}{l} \tfrac{1}{2}x^\top Q^+ x + \tfrac{1}{2}c^\top x - \tfrac{1}{2}u_x^\top Q^+ u_x + b^\top w_x \ge \tau, \\ A^\top w_x - Q^+ u_x \le Q^- x + \tfrac{1}{2}c. \end{array} \right\}, \tag{5}$$

which completes the proof. □

Problem (ARO-QO) is a quadratic ARO problem with fixed recourse and right-hand-side uncertainty. In this problem, τ is the static variable, $x \in \mathcal{X}$ is the uncertain parameter, and (u_x, w_x) is the adjustable variable. The adjustable variables can be seen as functions of x, and are known as decision policies (Khademi et al. 2024, Khademi 2024, Yanıkoğlu et al. 2019).

It is important to note that a concave QO problem, i.e., when dealing with a negative semi-definite matrix Q, is NP-hard. This complexity primarily arises from the crucial relationship between achieving optimality and enumerating the extreme points within the feasible region (Pardalos and Schnitger 1988). Predominant strategies for addressing concave QO problems typically involve cutting plane methods, branch and bound approaches, or iterative computational techniques (Audet et al. 2005, Andrianova et al. 2016, Chinchuluun et al. 2005, Phillips and Rosen 1988). Furthermore, recent studies in this area have focused on establishing bounds from a robust optimization perspective (Selvi et al. 2022), and some have adopted approaches that begin with convex optimization techniques to find starting points and then proceed to gradient descent principles (Ben-Tal and Roos 2022); the application of these techniques has been instrumental in deriving high-quality bounds for the optimal solution. In the subsequent corollary, we present the ARO reformulation for a concave QO problem.

COROLLARY 1. *Let Q be a negative semi-definite matrix. Assume that $\mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid Ax = b,\ x \ge 0\}$ is non-empty compact. Then, the optimal value of (QO) is equal to the optimal value of the following problem:*

$$\begin{aligned} & \max_{\tau\in\mathbb{R}} \ \tau \\ & \text{s.t. } \forall x \in \mathcal{X},\ \exists w_x: \begin{cases} \tfrac{1}{2}c^\top x + b^\top w_x \ge \tau, \\ A^\top w_x \le Qx + \tfrac{1}{2}c. \end{cases} \end{aligned} \tag{6}$$

Proof. From Theorem 2 by setting $Q^+ := 0$ and $Q^- := Q$. □

Note that (6) is a linear adjustable robust optimization problem and all techniques in the literature can be used to solve or approximate it.

REMARK 1. In Table 6 of Appendix C, we present the equivalent ARO formulations if the polytope $\mathcal{X}$ is formulated in another form than canonical. □

To approximate (ARO-QO), we can use customary techniques to deal with adjustable variables, such as eliminating the adjustable variables via Fourier-Motzkin Elimination or using decision rules to approximate the adjustable variables. In the next section, we focus on such approximation methods.

3. ARO Based Approximations

In this section, we recap some of the available customary approaches to approximate our (ARO-QO) problem from literature and discuss their interpretations concerning (QO).

3.1. Decision Rules

In (ARO-QO) problem, the adjustable variables u_x and w_x are, in essence, functions of the uncertain parameter x. One of the popular methods to approximate an ARO problem is by restricting the adjustable variables to belong to a specific class of functions. For example, we can restrict them to be constants, resulting in a static formulation, or to be affine, known as affine decision rule (ADR), which is a good approximation for linear ARO problems (see, e.g., Bertsimas and Bidkhori (2015) and Bertsimas et al. (2015, 2010)).

Since (ARO-QO) contains a non-linear convex term $u_x^\top Q^+ u_x$, using ADR to approximate u_x results in an intractable approximation. Therefore, we apply a hybrid decision rule to have a tractable approximation. More specifically, we restrict u_x to be constant and w_x to be affine:

$$u_x := u \text{ and } w_x := z + Zx,$$

where $u \in \mathbb{R}^{n_x}$, $z \in \mathbb{R}^{m_x}$, and $Z \in \mathbb{R}^{m_x \times n_x}$ are static variables. Using this decision rule in (ARO-QO) leads to the following static robust counterpart, which gives a lower bound on the optimal value of (QO):

$$\max_{u,z,Z,\tau} \left\{ \tau \;\middle|\; \begin{array}{ll} \frac{1}{2}x^\top Q^+ x + \frac{1}{2}c^\top x - \frac{1}{2}u^\top Q^+ u + b^\top (z + Zx) \geq \tau, & \forall x \in \mathcal{X} \\ A^\top (z + Zx) - Q^+ u \leq Q^- x + \frac{1}{2}c, & \forall x \in \mathcal{X} \end{array} \right\}, \tag{7}$$

where u, z, and Z are simultaneously optimized together with the static decision variable τ.

In the previous section, we demonstrated that the (QO) problem is equivalent to both the (Bi-QO) and (ARO-QO) problems. In the rest of this section, we explore the relationship between the reformulation-linearization technique (RLT) and the hybrid decision rule.

The RLT was originally proposed by Sherali and Alameddine (1992) and was further developed in Sherali and Tuncbilek (1995). RLT works in two steps: reformulation and linearization. The reformulation step creates additional constraints by multiplying the existing ones. The linearization step then replaces each unique product of variables with a new continuous variable. By applying RLT to approximate the problem

with a convex problem, we can find a lower bound on the solution of the original non-convex minimization problem.

The following theorem demonstrates that (7) is equivalent to applying RLT to the non-convex part of the objective function in (QO) when representing Q as $Q^+ + Q^-$, and linearizing $x^\top Q^- x$.

THEOREM 3. *Let $Q = Q^+ + Q^-$ where $Q \in \mathbb{R}^{n_x \times n_x}$, and $Q^+, -Q^- \succeq 0$. Assume that $\mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid Ax = b,\ x \geq 0\}$ is non-empty compact. Then, the optimal value of (7) is equal to the optimal value*

$$\begin{aligned} \min_{\substack{x\in\mathbb{R}^{n_x}\\ \gamma\in\mathcal{S}^{n_x}}} \ & x^\top Q^+ x + c^\top x + \sum_{i,j=1}^{n_x} Q^-_{ij}\gamma_{ij} \\ \text{s.t.} \ & Ax = b, \\ & A\gamma = bx^\top, \\ & x \geq 0,\ \gamma \geq 0. \end{aligned} \tag{8}$$

Proof. We can rewrite (7) as

$$\max_{u,z,Z,\tau} \left\{ \tau \;\middle|\; \begin{array}{ll} \min\limits_{x\in\mathcal{X}} \left\{ \frac{1}{2} x^\top Q^+ x + (\frac{1}{2}c^\top + b^\top Z)x \right\} + b^\top z - \frac{1}{2}u^\top Q^+ u \geq \tau, & \\ \min\limits_{x\in\mathcal{X}} \left\{ (-A^\top Z + Q^-)_i x \right\} + (\frac{1}{2}c + Q^+ u - A^\top z)_i \geq 0, & i = 1,\ldots,n_x \end{array} \right\}. \tag{9}$$

Since $\mathcal{X}$ is a polytope, the inner minimizations are convex optimization problems. Since $\mathcal{X}$ is non-empty and compact, strong duality holds (Boyd and Vandenberghe 2004, Dorn 1960). Therefore, (9) is equivalent to

$$\begin{aligned} \max_{u,z,Z,\tau} \ & \tau \\ \text{s.t.} \ & \max_{\alpha,\beta} \left\{ b^\top \beta - \tfrac{1}{2}\alpha^\top Q^+ \alpha \;\middle|\; A^\top \beta - Q^+ \alpha \leq (b^\top Z)^\top + \tfrac{1}{2}c \right\} + b^\top z - \tfrac{1}{2}u^\top Q^+ u \geq \tau, \\ & \max_{\theta^i} \left\{ b^\top \theta^i \;\middle|\; A^\top \theta^i \leq \left((-A^\top Z + Q^-)_i\right)^\top \right\} + (\tfrac{1}{2}c + Q^+ u - A^\top z)_i \geq 0, \quad i = 1,\ldots,n_x. \end{aligned} \tag{10}$$

We can omit the inner maximization operator in the above constraints since the maximizations are bounded. Thus, we have

$$\begin{aligned} \max_{u,z,Z,\alpha,\beta,\theta} \ & b^\top \beta - \tfrac{1}{2}\alpha^\top Q^+ \alpha + b^\top z - \tfrac{1}{2}u^\top Q^+ u \\ \text{s.t.} \ & A^\top \beta - Q^+ \alpha \leq (b^\top Z)^\top + \tfrac{1}{2}c, \\ & b^\top \theta^i + (\tfrac{1}{2}c + Q^+ u - A^\top z)_i \geq 0, \qquad i = 1,\ldots,n_x, \\ & A^\top \theta^i \leq \left((-A^\top Z + Q^-)_i\right)^\top, \qquad i = 1,\ldots,n_x. \end{aligned} \tag{11}$$

Now, consider the following RLT reformulation of (Bi-QO) problem

$$\begin{aligned}\min_{\gamma,x,y}\ & \tfrac{1}{2}\left(x^\top Q^+ x + y^\top Q^+ y + c^\top x + c^\top y\right) + \sum_{i,j=1}^{n_x} Q^-_{ij}\gamma_{ij}\\ \text{s.t.}\ & Ax = b,\\ & Ay = b,\\ & A\gamma = by^\top,\\ & A\gamma^\top = bx^\top,\\ & x \geq 0,\ y \geq 0,\ \gamma \geq 0.\end{aligned} \tag{12}$$

We first show that if $(\hat{x},\hat{y},\hat{\gamma})$ is an optimal solution to (12), then $\left(\frac{\hat{x}+\hat{y}}{2}, \frac{\hat{x}+\hat{y}}{2}, \frac{\hat{\gamma}+\hat{\gamma}^\top}{2}\right)$ is also an optimal solution. Due to the optimality of $(\hat{x},\hat{y},\hat{\gamma})$, it is clear that $(\hat{y},\hat{x},\hat{\gamma}^\top)$ is also an optimal solution. So, the feasibility of $\left(\frac{\hat{x}+\hat{y}}{2}, \frac{\hat{x}+\hat{y}}{2}, \frac{\hat{\gamma}+\hat{\gamma}^\top}{2}\right)$ is evident. Setting $\Phi(x,y,\gamma) := \frac{1}{2}\left(x^\top Q^+ x + y^\top Q^+ y + c^\top x + c^\top y\right) + \sum_{i,j=1}^{n_x} Q^-_{ij}\gamma_{ij}$, we know the optimal value is $\Phi(\hat{x},\hat{y},\hat{\gamma}) = \Phi(\hat{y},\hat{x},\hat{\gamma}^\top)$. Furthermore, the convexity of $x^\top Q^+ x$ in x implies that

$$2\Phi\left(\tfrac{\hat{x}+\hat{y}}{2}, \tfrac{\hat{x}+\hat{y}}{2}, \tfrac{\hat{\gamma}+\hat{\gamma}^\top}{2}\right) \leq \Phi(\hat{x},\hat{y},\hat{\gamma}) + \Phi(\hat{y},\hat{x},\hat{\gamma}^\top) = 2\Phi(\hat{x},\hat{y},\hat{\gamma}).$$

Therefore $\left(\frac{\hat{x}+\hat{y}}{2}, \frac{\hat{x}+\hat{y}}{2}, \frac{\hat{\gamma}+\hat{\gamma}^\top}{2}\right)$ is an optimal solution to (12).

We have thus shown that there exists an optimal solution to (12) with the structure $x = y$ and $\gamma = \gamma^\top$. We now show that (12) and (8) are equivalent. To demonstrate this, it suffices to show that any optimal solution to (12) corresponds to an optimal solution to (8) with the same objective value. Suppose that $(\hat{x},\hat{y},\hat{\gamma})$ is optimal for (12). Then $\left(\bar{x} = \frac{\hat{x}+\hat{y}}{2}, \bar{x} = \frac{\hat{x}+\hat{y}}{2}, \bar{\gamma} = \frac{\hat{\gamma}+\hat{\gamma}^\top}{2}\right)$ is also optimal for (12). We claim that $(\bar{x},\bar{\gamma})$ is an optimal solution to (8). If the claim is not true, then there exists a feasible solution $(\tilde{x},\tilde{\gamma})$ such that

$$\tilde{x}^\top Q^+ \tilde{x} + c^\top \tilde{x} + \sum_{i,j=1}^{n_x} Q^-_{ij}\tilde{\gamma}_{ij} < \bar{x}^\top Q^+ \bar{x} + c^\top \bar{x} + \sum_{i,j=1}^{n_x} Q^-_{ij}\bar{\gamma}_{ij}.$$

This implies that, $(\tilde{x},\tilde{x},\tilde{\gamma})$ is feasible for (12) and achieves a better objective value than $(\hat{x},\hat{y},\hat{\gamma})$, which is a contradiction.

Now, we show that (11) is the Dorn dual problem of (12) (which is equivalent to (8)). To do this, we first write (12) in the matrix form:

$$\begin{aligned}\min_{\text{vec}(\gamma),x,y}\ & \tfrac{1}{2}\begin{pmatrix} x \\ y \\ \text{vec}(\gamma)\end{pmatrix}^\top \begin{pmatrix} Q^+ & 0 & 0 \\ 0 & Q^+ & 0 \\ 0 & 0 & 0\end{pmatrix}\begin{pmatrix} x \\ y \\ \text{vec}(\gamma)\end{pmatrix} + \begin{pmatrix} \frac{c}{2} \\ \frac{c}{2} \\ \text{vec}(Q^-)\end{pmatrix}^\top \begin{pmatrix} x \\ y \\ \text{vec}(\gamma)\end{pmatrix}\\ \text{s.t.}\ & \begin{pmatrix} A & 0 & 0 \\ 0 & A & 0 \\ 0 & B & C \\ B & 0 & D\end{pmatrix}\begin{pmatrix} x \\ y \\ \text{vec}(\gamma)\end{pmatrix} = \begin{pmatrix} b \\ b \\ 0 \\ 0\end{pmatrix},\\ & \begin{pmatrix} x \\ y \\ \text{vec}(\gamma)\end{pmatrix} \geq 0,\end{aligned} \tag{13}$$

where $B := -\begin{pmatrix} b_1 I_{n_x} \\ b_2 I_{n_x} \\ \vdots \\ b_{m_x} I_{n_x} \end{pmatrix}$, $C := \begin{pmatrix} A_{11} I_{n_x} & A_{12} I_{n_x} & \dots & A_{1 n_n} I_{n_x} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m_x 1} I_{n_x} & A_{m_x 2} I_{n_x} & \dots & A_{m_x n_x} I_{n_x} \end{pmatrix}$, and $D := \begin{pmatrix} A_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & A_1 \\ \vdots & & \vdots \\ A_{m_x} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & A_{m_x} \end{pmatrix}$. The dual of (13) is

$$\begin{aligned} \max_{Y,W} \; & -\tfrac{1}{2} Y^\top \begin{pmatrix} Q^+ & 0 & 0 \\ 0 & Q^+ & 0 \\ 0 & 0 & 0 \end{pmatrix} Y + \begin{pmatrix} b \\ b \\ 0 \\ 0 \end{pmatrix}^\top W \\ \text{s.t.} \; & -\begin{pmatrix} Q^+ & 0 & 0 \\ 0 & Q^+ & 0 \\ 0 & 0 & 0 \end{pmatrix} Y + \begin{pmatrix} A & 0 & 0 \\ 0 & A & 0 \\ 0 & B & C \\ B & 0 & D \end{pmatrix}^\top W \leq \begin{pmatrix} \frac{c}{2} \\ \frac{c}{2} \\ \text{vec}(Q^-) \end{pmatrix}. \end{aligned} \tag{14}$$

Setting

$$Y \equiv \begin{pmatrix} \alpha \\ u \\ Y^3 \end{pmatrix}, \quad W \equiv \begin{pmatrix} \beta \\ z \\ \text{vec}(\theta) \\ \text{vec}(Z) \end{pmatrix},$$

(14) is the matrix form of (11). Hence, (12) is the dual of the deterministic reformulation of (7). Observe that, due to the property of Dorn duality at the optimal solution, we have $Y^* = \begin{pmatrix} x^* \\ y^* \\ \text{vec}(\gamma^*) \end{pmatrix}$, i.e., $x^* = \alpha^*$ and $y^* = u^*$. □

Theorem 3 establishes that a convex relaxation of the original QO through an RLT is equivalent to approximating its ARO reformulation via hybrid decision rules.

The literature has also considered RLTs to approximate an ARO problem. More specifically, it is shown in Ardestani-Jaafari and Delage (2021) that a linear ARO problem can be reformulated as a bi-linear optimization problem using duality techniques. The authors then show that using an RLT to approximate the bi-linear optimization reformulation is equivalent to applying ADR to the original problem. In Zhen et al. (2022), the same results are shown for disjoint bi-linear problems with convex feasible regions.

Theorem 3 bridges the gap between existing analytical results for approximating ARO problems and those for approximating (QO). This allows us to extend the analytical techniques from one domain to the other. For example, Bertsimas and Bidkhori (2015) provide us with analytical bounds on the quality of the ADR approximation of a linear ARO problem, which can directly be translated to analytical bounds on the quality of (8) to approximate a concave QO problem.

In (QO), we can assume, without loss of generality, that the matrix Q is symmetric; otherwise, we can replace the objective function with $x^\top (\frac{Q^\top + Q}{2}) x + c^\top x$. Now, for a symmetric matrix Q, we know that the eigenvalues are real (O'Nan 1971). So, for an indefinite matrix Q, we can construct the matrices in Theorem

1 in many ways, including the following representations:

Representation 1:

$$Q^+ := Q - (\lambda_{min}(Q) - \epsilon)I, \text{ and } Q^- := (\lambda_{min}(Q) - \epsilon)I,$$

Representation 2:

$$Q^+ := (\lambda_{max}(Q) + \epsilon)I, \text{ and } Q^- := Q - (\lambda_{max}(Q) + \epsilon)I,$$

Representation 3:

$$Q^+ := V\Lambda^+V^\top, \text{ and } Q^- := V\Lambda^-V^\top,$$

where $V \in \mathbb{R}^{n_x \times n_x}$ is an orthonormal matrix of eigenvectors of Q, $\Lambda^+ = \text{Diag}(d_1^+, \ldots, d_{n_x}^+)$, $\Lambda^- = \text{Diag}(d_1^-, \ldots, d_{n_x}^-)$, with $d_i^+ = \max\{\lambda_i + \epsilon, \epsilon\}$, $d_i^- = \min\{\lambda_i - \epsilon, -\epsilon\}$, and λ_i being the $i-$th eigenvalue of Q, $i = 1, ..., n_x$. In all representations, a small positive constant ϵ is chosen to ensure that Q^+ and $-Q^- \succ 0$.

The representations above offer a natural method to decompose arbitrary indefinite symmetric matrices by expressing them as the sum of one positive and one negative semi-definite matrix, as also discussed in (Bomze 2002a, Section 5.2).

Considering Representation 1, we see that Q^- is a diagonal matrix, but Q^+ has a similar density as Q. Therefore, in (ARO-QO), all entries of u_x are linked together via Q^+u_x. However, in Representation 2, Q^+ is a diagonal matrix, implying that the entries of u_x are only linked together via $u_x^\top Q^+ u_x$ and are not linked in the constraints. In the numerical result section, we will use Representation 2 and Representation 3, showcasing their effectiveness in practical scenarios and their impact on computational efficiency and solution accuracy.

3.2. Fourier–Motzkin Elimination

In linear ARO problems with fixed recourse, an adjustable variable may be eliminated by employing Fourier-Motzkin elimination (EME). This approach effectively handles problems involving a limited number of adjustable variables (Zhen et al. 2018).

Note that for a given $x \in \mathcal{X}$, in (ARO-QO), we have the ability to eliminate the adjustable variable $w_x \in \mathbb{R}^{m_x}$. We assume without loss of generality that $b \geq 0$. Let $k \in \{1, \ldots, m_x\}$. To eliminate w_{x_k}, the k-th component of the vector w_x, we first isolate it in the constraints:

$$\begin{cases} b_k w_{x_k} \geq \tau - \frac{1}{2}x^\top Q^+ x - \frac{1}{2}c^\top x + \frac{1}{2}u_x^\top Q^+ u_x - \sum\limits_{\substack{j=1 \\ j\neq k}}^{m_x} b_j w_{x_j}, \\ A_{ki} w_{x_k} \leq (Q^- x + \frac{1}{2}c + Q^+ u_x)_i - \sum\limits_{\substack{j=1 \\ j\neq k}}^{m_x} A_{ji} w_{x_j}, \qquad i = 1, \ldots, m_x. \end{cases} \tag{15}$$

Since $\mathcal{X} = \{x |\ Ax = b,\ x \geq 0\}$ is non-empty, so we cannot have $b_k > 0$ and $A_{ki} \leq 0$ for any $i = 1, \ldots, m_x$.

If $A_{ki} \neq 0$ and $b_k > 0$, then both sides of their respective constraints can be divided by A_{ki} and b_k. This yields an equivalent representation of the feasible region, involving the following constraints:

$$
\begin{aligned}
& w_{x_k} \geq \tfrac{1}{b_k}(\tau - \tfrac{1}{2}x^\top Q^+ x - \tfrac{1}{2}c^\top x + \tfrac{1}{2}u_x^\top Q^+ u_x - \sum_{\substack{j=1\\ j\neq k}}^{m_x} b_j w_{x_j}) && \text{if } b_k > 0,\\
& 0 \geq \tau - \tfrac{1}{2}x^\top Q^+ x - \tfrac{1}{2}c^\top x + \tfrac{1}{2}u_x^\top Q^+ u_x - \sum_{\substack{j=1\\ j\neq k}}^{m_x} b_j w_{x_j} && \text{if } b_k = 0,\\
& w_{x_k} \geq \tfrac{1}{A_{ki}}((Q^- x + \tfrac{1}{2}c + Q^+ u_x)_i - \sum_{\substack{j=1\\ j\neq k}}^{m_x} A_{ji} w_{x_j}) && \text{for } i = 1, \ldots, m_x, \text{ where } A_{ki} < 0,\\
& \tfrac{1}{A_{kr}}((Q^- x + \tfrac{1}{2}c + Q^+ u_x)_i - \sum_{\substack{j=1\\ j\neq k}}^{m_x} A_{jr} w_{x_j}) \geq w_{x_k} && \text{for } r = 1, \ldots, m_x, \text{ where } A_{kr} > 0,\\
& (Q^- x + \tfrac{1}{2}c + Q^+ u_x)_i - \sum_{\substack{j=1\\ j\neq k}}^{m_x} A_{js} w_{x_j} \geq 0 && \text{for } s = 1, \ldots, m_x, \text{ where } A_{ks} = 0.
\end{aligned}
$$

After the adjustable variable w_{x_k} is eliminated, the feasible set becomes:

$$
\begin{aligned}
& \tfrac{1}{A_{kr}}((Q^- x + \tfrac{1}{2}c + Q^+ u_x)_i - \sum_{\substack{j=1\\ j\neq k}}^{m_x} A_{ji} w_{x_j}) \geq \\
& \qquad \tfrac{1}{b_k}(\tau - \tfrac{1}{2}x^\top Q^+ x - \tfrac{1}{2}c^\top x + \tfrac{1}{2}u_x^\top Q^+ u_x - \sum_{\substack{j=1\\ j\neq k}}^{m_x} b_j w_{x_j}) && r = 1, \ldots, m_x,\\
& && \text{where } b_k > 0 \text{ and } A_{kr} > 0,\\
& 0 \geq \tau - \tfrac{1}{2}x^\top Q^+ x - \tfrac{1}{2}c^\top x + \tfrac{1}{2}u_x^\top Q^+ u_x - \sum_{\substack{j=1\\ j\neq k}}^{m_x} b_j w_{x_j} && \text{where } b_k = 0,\\
& \tfrac{1}{A_{kr}}((Q^- x + \tfrac{1}{2}c + Q^+ u_x)_r - \sum_{\substack{j=1\\ j\neq k}}^{m_x} A_{jr} w_{x_j}) \geq \\
& \qquad \tfrac{1}{A_{ki}}((Q^- x + \tfrac{1}{2}c + Q^+ u_x)_i - \sum_{\substack{j=1\\ j\neq k}}^{m_x} A_{ji} w_{x_j}) && i, r = 1, \ldots, m_x,\\
& && \text{where } A_{ki} < 0 \text{ and } A_{kr} > 0,\\
& (Q^- x + \tfrac{1}{2}c + Q^+ u_x)_s - \sum_{\substack{j=1\\ j\neq k}}^{m_x} A_{js} w_{x_j} \geq 0 && s = 1, \ldots, m_x, \text{ where } A_{sk}^\top = 0.
\end{aligned}
$$

By continuing the process of FME, the adjustable variable w_x (or some part of it) is eliminated, resulting in a problem with fewer adjustable variables but potentially many more constraints. If the number of constraints in (QO) is limited, then it is computationally efficient to eliminate w_x.

4. Solution Method

In the previous section, we explained how to obtain a lower bound using the techniques from ARO literature. This section provides an algorithm to obtain a feasible solution and construct an upper bound.

After solving the approximated problem (7), we use the obtained solution to extract worst-case scenarios from each constraint of the robust counterpart problem (7). Among these scenarios, we select the one that yields the best objective value for the original (QO) problem. After identifying the most favorable scenario, our attention is redirected to the bi-convex reformulation of (QO) problem. Given the selected scenario, we employ the mounting claiming algorithm (Algorithm 1) for (Bi-QO) to improve the quality of the solution. This process ultimately leads us to an upper bound for (QO) problem. The mountain climbing procedure was initially introduced by Konno (1976) for bi-linear optimization problems, but can easily be extended to bi-convex problems, wherein the variable is partitioned into disjoint sets. For further information regarding the convergence theory of the mountain climbing procedure, we refer to Gorski et al. (2007), Grippo and Sciandrone (2000).

Algorithm 1 Mountain Climbing Procedure

Input: Matrix Q and starting point x^0.

Initialization: Decompose $Q = Q^+ + Q^-$ such that $Q^+, -Q^- \succeq 0$.

Repeat: Execute the following steps:

$$x^{(k+1)} \leftarrow \arg\min_{x \in \mathbb{R}^{n_x}} \left\{ \tfrac{1}{2} x^\top Q^+ x + x^\top Q^- x^{(k)} + \tfrac{1}{2} c^\top x : x \in \mathcal{X} \right\}.$$

Until: No further improvement is possible.

Output: Solution candidate $x^{(end)}$.

By employing the ARO reformulation, bi-convex reformulation, and mounting claiming method, we can efficiently explore and improve the solution space, thereby obtaining an upper bound that closely approaches the optimal value. This approach allows us to make significant progress in refining the solution quality while mitigating computational challenges often associated with large-scale optimization problems. Algorithm 2 presents the pseudo-code of the approach discussed above.

5. Numerical Experiments

In this section, we conduct a comprehensive numerical experiment to evaluate the efficacy of Algorithm 2, which we call ARO-QO Algorithm. The efficiency of a particular bound on the optimal value of a mathematical optimization problem is influenced by two key aspects: the precision of the generated bound and the required computational time.

We implement the numerical experiments using MATLAB 2022a. The computations are executed on a laptop equipped with an Intel(R) Core(TM) i5-3210M CPU at 2.50 GHz and 8 GB of RAM. We use YALMIP to pass optimization problems to suitable solvers (Löfberg 2004).

We emphasize that the computational times reported in our experiments exclude the time required by YALMIP to build the model and pass it to solvers, and we merely consider the time consumed by the

Algorithm 2 ARO-Based Algorithm to Obtain an Upper Bound for QO

Input: Matrix Q, vector c, matrix A, and vector b.

Initialization: Decompose $Q = Q^+ + Q^-$ such that $Q^+, -Q^- \succeq 0$.

- **(Step 1) Lower Bound:** Compute the lower bound for the approximated problem based on the ARO formulation of the QO and hybrid decision rule (see Section 3).
- **(Step 2) Generation of Worst-Case Scenarios:** Generate a finite set of worst-case scenarios by substituting the optimal decision rule into (7).
- **(Step 3) Set Initial Point:** Select from these scenarios the one that yields the best objective value for the original QO problem. Denote this point by $x^{(0)}$.
- **(Step 4) Improve the Initial Solution:** Execute the mountain climbing algorithm starting with the initial solution $x^{(0)}$:

$$x^{(k+1)} \longleftarrow \arg\min \left\{ \tfrac{1}{2} x^\top Q^+ x + x^\top Q^- x^{(k)} + \tfrac{1}{2} c^\top x : \; x \in \mathcal{X} \right\}.$$

- **(Step 5) Termination:** Continue (Step 4) until no further improvement is observed.

Output: Final solution candidate $x^* := x^{(end)}$, and the corresponding upper-bound value $UB := (x^*)^\top Q x^* + c^\top x^*$.

solvers themselves. In what follows, we present the numerical experiments, specifically focusing on concave quadratic minimization, standard quadratic optimization, and general indefinite quadratic optimization. All the instances and the code are available using this link: [Link].

We use off-the-shelf global solvers to solve the QO problems, namely Gurobi (Gurobi Optimization 2023, version 10.0) and CPLEX (IBM 2019, version 12.9). We specifically use CPLEX version 12.9, as Zhen et al. (2022) reported a bug in the solver of newer versions for solving non-convex QO problems. It is crucial to highlight that these solvers aim to achieve global solutions. Moreover, MOSEK (MOSEK ApS 2023, version 10.1.15) is used to solve second-order cone optimization problems.

5.1. Concave Quadratic Minimization

Let us consider a concave quadratic minimization over a polyhedron

$$\begin{aligned} \min_{x \geq 0} \; & x^\top Q x + c^\top x \\ \text{s.t.} \; & Ax \geq b, \end{aligned} \tag{16}$$

where $Q \in \mathbb{R}^{n_x \times n_x}$, and $-Q \succeq 0$, $c \in \mathbb{R}^{n_x}$, $A \in \mathbb{R}^{m_x \times n_x}$, and $b \in \mathbb{R}^{m_x}$ are given. From Corollary 1 and Table 6 in Appendix C, we have the following linear ARO reformulation of (16):

$$\begin{aligned} \max_{\tau \in \mathbb{R}} \; & \tau \\ \text{s.t.} \; \forall x \in \mathcal{X}, \; \exists w_x : & \begin{cases} \frac{1}{2} c^\top x + b^\top w_x \geq \tau, \\ A^\top w_x \leq Q x + \frac{1}{2} c, \\ w_x \geq 0, \end{cases} \end{aligned} \tag{17}$$

where $\mathcal{X} := \{x \in \mathbb{R}^{n_x} | Ax \geq b,\ x \geq 0\}$. To obtain a lower bound, we consider the following decision rule:

$$w_x := \begin{pmatrix} z + Zx \\ w \end{pmatrix},$$

where for a given $r \in \{1, 2, \ldots, m_x\}$, $z \in \mathbb{R}^r$, $Z \in \mathbb{R}^{r \times n_x}$, and $w \in \mathbb{R}^{(m_x - r)}$ are static variables. It is important to note that for $r = m_x$, we obtain a full affine decision rule, while for $r = 0$, we have a static decision rule. For other values, we have a partial affine decision rule. Each decision rule type has its own advantages and disadvantages, which we will address later in this section.

In Selvi et al. (2022), the authors propose an approximation solution approach for solving a concave minimization problem via ARO by providing upper and lower bounds, where the lower bound is formulated as a second-order cone optimization problem. Furthermore, the solution method proposed by Ben-Tal and Roos (2022) finds near-optimal solutions to concave minimization problems. This method consists of two main phases. In the initial phase, the method employs several approaches to approximate the original problem with a tractable convex optimization problem. Subsequently, in the second phase, the method employs conditional gradient descent and iterates through various starting points generated in the initial phase, thereby enhancing the quality of the final solution. It is worth mentioning that this method only provides an upper bound on the problem. In this section, we compare the quality of the solution obtained by ARO-QO Algorithm with Gurobi, CPLEX, Selvi et al. (2022), and Ben-Tal and Roos (2022). In all of our numerical experiments, we set a maximum time limit of 3,000 seconds.

We analyze the performance of the upper and lower bound in terms of the optimality gap, which is measured as follows:

$$\text{OGap}(\%) = \left(\frac{\text{UB} - \text{LB}}{|\text{UB}| + 10^{-4}} \right) \times 100,$$

where LB is the lower bound and UB is the upper bound for a given instance. Adding the small constant 10^{-4} in the denominator ensures the prevention of division by zero. Additionally, we analyze the performance of the upper bounds in terms of the solution gap, measured as:

$$\text{SGap}(\%) = \left(\frac{\text{UB} - \text{UB}^{(best)}}{|\text{UB}| + 10^{-4}} \right) \times 100,$$

where $\text{UB}^{(best)}$ represents the best obtained upper bound among all approaches, and UB is the upper bound for a given method.

Problem Instances First, we consider the seven test instances from Section 4.3 of Selvi et al. (2022). We undertake a detailed comparison of three versions of ARO-QO Algorithm (static, partial, and fully affine), Selvi et al. (2022), Ben-Tal and Roos (2022), and global solvers Gurobi and CPLEX. In the lower bound approximation of the ARO-QO Algorithm, applying fully static, partially affine (restricting the first $r = [\frac{m_x}{7}] + 1$ of w_x to be affine and the remaining $m_x - r$ to be constant), and fully affine decision rules has distinct effects on the optimality gaps.

Table 1 demonstrates a clear correlation between the type of decision rules and the tightness of optimality gaps within our proposed ARO-QO Algorithm. The fully affine ARO-QO Algorithm consistently achieves the smallest optimality gaps among its variants, providing the tightest bounds for problems it can solve within the time limit. However, this precision comes at the cost of significantly longer solver times, particularly for larger problems. For instance, Problem 5 required 900.48 seconds, while Problems 6 and 7 exceeded the 3000-second time limit. In contrast, the static and partial ARO-QO Algorithms offer a balance between solution quality and speed, achieving competitive optimality gaps within reasonable time frames, especially for larger problems like Problem 7. Global solvers Gurobi and CPLEX demonstrate acceptable performance on smaller instances (Problems 1-6), consistently achieving optimality. However, they struggle with the largest problem (Problem 7), where both solvers reach time limits.

While the method of Ben-Tal and Roos (2022) offers a notable balance between solution quality and speed across all problem sizes, by consistently achieving reasonable solution gaps with competitive computation times, the fully static version of our algorithm outperforms their method in all instances. The method of Selvi et al. (2022), while providing relatively quick solutions, typically leads to larger optimality gaps compared to the ARO-QO Algorithm due to its less tight lower bounds (see Table 7 in Appendix D for more details). Moreover, the solver times for the Selvi et al. (2022) method are generally longer than those of the static ARO-QO Algorithm. The results of our analysis indicate that the static ARO-QO Algorithm demonstrates superior performance compared to the method proposed by Selvi et al. (2022) when considering both optimality gap and computational efficiency.

Even though the static ARO-QO Algorithm yields the highest optimality gap among other decision rules, it stands out for its minimal computation time required to derive both lower and upper bounds. In the static ARO-QO Algorithm, based on the structure of problem (17), the upper bound can be calculated by directly identifying worst-case scenarios and then improving the best one using the mountain climbing procedure to reach a candidate solution. Furthermore, we incorporate the worst-case scenario corresponding to each constraint in (17), resulting in a linear optimization problem that yields the lower bound. Remarkably, in all seven instances, the calculated upper bound is obtained computationally efficiently with good solution quality. Compared to alternative approaches—such as partial or full affine ARO-QO Algorithms, or using the methods of Selvi et al. (2022) and Ben-Tal and Roos (2022)—the static ARO-QO Algorithm demonstrates a faster computation process in reaching a candidate solution.

After considering the seven test instances of Selvi et al. (2022), we randomly generate large-size instances. For a meaningful comparison of the mentioned approaches, we evaluate the quality of the bounds on the objective value of problem (16) using 15 Groups of random instances, with the dimension n_x taking value in $\{50, 100, \ldots, 600, 700\}$ with the number of constraints $m_x = n_x + 50$ and $m_x = n_x + 100$. Each group contains five instances of the same size, which are generated similarly to those created in Selvi et al. (2022).

Table 1 Solution and optimality gaps for concave quadratic minimization instances from Selvi et al. (2022)

Problem	Static ARO-QO			Partial affine ARO-QO			Full affine ARO-QO			Selvi et al. (2022)			Ben-Tal and Roos (2022)		Gurobi			CPLEX		
	SGap	OGap	Time	SGap	OGap	Time	SGap	OGap	Time	SGap	OGap	Time	SGap	Time	SGap	OGap	Time	SGap	OGap	Time
#1 ($m_x = 10, n_x = 20$)	0.00	30.38	0.05	0.00	27.97	0.15	0.00	0.02	1.05	0.00	77.85	0.17	0.00	6.78	0.00	0.00	0.11	0.00	0.00	0.12
#2 ($m_x = 10, n_x = 20$)	0.00	9.13	0.05	0.00	8.94	0.15	0.00	0.01	1.06	0.00	34.73	0.15	0.00	7.67	0.00	0.00	0.12	0.00	0.00	0.11
#3 ($m_x = 15, n_x = 10$)	0.00	0.42	0.03	0.00	0.29	0.06	0.00	0.00	0.12	0.00	1.68	0.15	0.00	8.18	0.00	0.00	0.03	0.00	0.01	0.06
#4 ($m_x = 62, n_x = 50$)	0.00	0.12	0.13	0.00	0.09	0.48	0.00	0.00	11.46	0.00	1.10	1.32	0.00	6.95	0.00	0.00	0.56	0.00	0.00	1.56
#5 ($m_x = 130, n_x = 100$)	0.00	0.08	0.32	0.00	0.07	4.81	0.00	0.00	900.48	0.00	2.15	8.27	0.00	7.77	0.00	0.00	12.03	0.00	0.00	43.64
#6 ($m_x = 240, n_x = 200$)	0.00	0.02	0.95	0.00	0.02	44.83	-	-	3000*	0.00	1.52	59.60	0.00	11.27	0.00	0.00	528.99	0.00	0.00	1165.38
#7 ($m_x = 280, n_x = 240$)	0.00	0.04	1.36	0.00	0.04	104.90	-	-	3000*	0.00	5.71	80.90	0.00	13.06	12.48	15.22	3000*	0.00	16.51	3000*

Notes. In this table, 'OGap' represents the optimality gap, which measures the relative difference between the upper and lower bounds of the solution. 'SGap' represents the solution gap, which measures the relative difference between a given solution's upper bound and the best upper bound found across all methods. The symbol "-" indicates that it was not possible to determine the bound within the 3000-second time limit.

Table 2 presents a comparison of various methods for 15 Groups of increasing size. For each group, the table lists the mean optimality gap, solution gap, and solver time, with standard deviations shown between brackets (see Table 8 in Appendix D for more details). Boldface numbers highlight the best solution gap, and underlined numbers highlight the best optimality gap in each group.

Our proposed (static) ARO-QO Algorithm demonstrates remarkable consistency, maintaining strong performance across all problem dimensions. It achieves the best optimality gaps for larger problems (Groups 6-15), with the average optimality gap being less than 2% for all groups, and solves all instances within the time limit with very low solution gaps. Notably, the ARO-QO Algorithm exhibits the lowest computation time compared to other methods in each group, underscoring its efficiency in balancing time and gap management across these instances. The method by Selvi et al. (2022) displays more consistent optimality gaps across all problem groups, despite being significantly higher than those of Gurobi and CPLEX for the smaller instances. The method of Ben-Tal and Roos (2022) consistently achieves the best average solution gap, which is always below 0.05%, with a maximum standard deviation of 0.12% across all problem sizes, though it cannot provide optimality gaps.

Global solvers Gurobi and CPLEX perform exceptionally well for smaller problems (Groups 1-3), with Gurobi achieving very low optimality gaps. However, they begin to struggle with larger problems from Group 4 onwards. Gurobi and CPLEX have acceptable solution gaps up to Group 7, but Gurobi could not find feasible solutions for instances in Groups 14 and 15, while CPLEX failed to find feasible solutions for instances in Groups 8-15, all within the given 3,000-second time limit. This highlights the challenges these solvers face with large-scale instances.

5.2. Standard Quadratic Optimization

Let us consider a standard quadratic optimization problem

$$\min_{x \in \Delta} \; x^\top \tilde{Q} x + c^\top x.$$

In general, a standard QO problem is NP-hard (Bomze and De Klerk 2002). We remark that the quadratic function $x^\top \tilde{Q} x + c^\top x$ over the unit-simplex can be described as a homogeneous quadratic function $x^\top Q x$,

Table 2 Statistic of solution and optimality gaps and solver times for randomly generated concave minimization instances

Group (Dimension)	Static ARO-QO			Selvi et al. (2022)			Ben-Tal and Roos (2022)		Gurobi			CPLEX		
	OGap	SGap	Time	OGap	SGap	Time	SGap	Time	OGap	SGap	Time	OGap	SGap	Time
#1 ($m_x = 100, n_x = 50$)	1.54 [0.35]	0.03 [0.07]	0.20 [0.01]	13.32 [0.94]	**0.00** [0.00]	2.20 [0.28]	**0.00** [0.00]	9.47 [3.81]	0.01 [0.00]	**0.00** [0.00]	66.53 [44.43]	0.01 [0.00]	**0.00** [0.00]	16.93 [2.14]
#2 ($m_x = 150, n_x = 100$)	1.49 [0.35]	0.03 [0.04]	0.49 [0.01]	13.81 [0.88]	**0.00** [0.00]	9.00 [0.40]	**0.00** [0.00]	10.05 [0.73]	0.01 [0.00]	**0.00** [0.00]	776.08 [407.71]	0.02 [0.01]	**0.00** [0.00]	382.64 [181.74]
#3 ($m_x = 200, n_x = 100$)	1.71 [0.36]	0.05 [0.08]	0.72 [0.03]	16.37 [0.97]	0.01 [0.01]	17.83 [1.46]	**0.00** [0.00]	12.50 [0.55]	0.02 [0.02]	**0.00** [0.00]	1602.76 [1035.26]	0.12 [0.15]	**0.00** [0.00]	1285.16 [1256.08]
#4 ($m_x = 250, n_x = 200$)	1.40 [0.37]	0.01 [0.02]	1.43 [0.11]	14.71 [2.03]	**0.00** [0.01]	51.25 [4.76]	**0.00** [0.00]	11.31 [0.12]	10.69 [10.33]	0.15 [0.33]	3000*	0.22 [0.35]	0.15 [0.33]	2927.72 [161.62]
#5 ($m_x = 300, n_x = 200$)	1.33 [0.13]	**0.00** [0.00]	2.60 [0.11]	17.02 [1.03]	0.01 [0.01]	101.50 [22.87]	**0.00** [0.00]	12.96 [0.18]	25.93 [3.34]	**0.00** [0.00]	3000*	0.43[6.26]	**0.00** [0.00]	3000*
#6 ($m_x = 350, n_x = 300$)	1.62 [0.19]	**0.01** [0.01]	2.89 [0.09]	16.38 [1.18]	0.02 [0.03]	154.01 [10.49]	0.05 [0.12]	16.31 [0.19]	23.85 [3.49]	0.12 [0.12]	3000*	1365.22 [1636.18]	0.12 [0.12]	3000*
#7 ($m_x = 400, n_x = 300$)	1.53 [0.28]	**0.00** [0.00]	5.40 [0.19]	18.83 [2.21]	0.73 [1.62]	256.42 [30.29]	**0.00** [0.00]	19.06 [0.29]	37.02 [2.57]	0.06 [0.11]	3000*	4104.90 [236.04]	0.06 [0.11]	3000*
#8 ($m_x = 450, n_x = 400$)	1.70 [0.25]	**0.00** [0.01]	4.81 [0.15]	16.21 [1.83]	**0.00** [0.00]	295.57 [12.37]	**0.00** [0.00]	22.28 [0.23]	118.90 [88.99]	23.26 [22.36]	3000*	-	-	3000*
#9 ($m_x = 500, n_x = 400$)	1.89 [0.20]	**0.00** [0.00]	9.29 [0.37]	19.38 [0.77]	0.02 [0.02]	556.61 [19.42]	0.01 [0.02]	26.50 [0.70]	171.95 [103.12]	8.76 [18.96]	3000*	-	-	3000*
#10 ($m_x = 550, n_x = 500$)	1.79 [0.08]	**0.00** [0.00]	7.24 [0.17]	16.92 [0.56]	0.01 [0.01]	256.74 [15.37]	**0.00** [0.00]	29.42 [0.65]	2521.58 [5213.52]	71.26 [46.24]	3000*	-	-	3000*
#11 ($m_x = 600, n_x = 500$)	1.64 [0.19]	**0.00** [0.00]	17.66 [0.47]	19.24 [0.66]	0.01 [0.01]	504.47 [15.83]	**0.00** [0.00]	34.92 [1.26]	5420.98 [4697.11]	75.31 [35.19]	3000*	-	-	3000*
#12 ($m_x = 650, n_x = 600$)	1.57 [0.12]	**0.00** [0.00]	13.09 [0.51]	17.17 [0.89]	0.01 [0.01]	424.63 [43.45]	**0.00** [0.00]	37.81 [1.27]	8616.82 [5403.16]	158.20 [22.93]	3000*	-	-	3000*
#13 ($m_x = 700, n_x = 600$)	1.48 [0.29]	**0.00** [0.00]	23.83 [0.57]	18.89 [1.50]	0.02 [0.01]	821.51 [75.41]	0.01 [0.02]	45.35 [1.73]	10233.75 [1598.93]	91.68 [3.34]	3000*	-	-	3000*
#14 ($m_x = 750, n_x = 700$)	1.79 [0.39]	0.10 [0.23]	16.92 [0.53]	17.33 [1.08]	0.02 [0.02]	503.12 [36.98]	**0.00** [0.00]	47.67 [1.18]	-	-	3000*	-	-	3000*
#15 ($m_x = 800, n_x = 700$)	1.48 [0.21]	**0.00** [0.00]	31.60 [1.23]	19.21 [1.47]	0.01 [0.02]	1164.43 [515.86]	**0.00** [0.00]	56.33 [0.86]	-	-	3000*	-	-	3000*

Notes: This table categorizes problems into groups in the first column. The subsequent columns display "mean [standard deviation]" values of each subgroup's solution and optimality gaps and solver time. The symbol "-" indicates that it was not possible to determine upper bounds for all instances of the corresponding group within the maximum time limit. Bold numbers indicate the best solution gap, while underlined numbers show the best optimality gap per group.

where $Q := \tilde{Q} + \frac{1}{2}ec^\top + \frac{1}{2}ce^\top$. Hence, without loss of generality, the standard QO problem can be represented as follows:

$$\min_{x\in\Delta} \; x^\top Q x. \tag{StQO}$$

Let $Q \in \mathbb{R}^{n_x \times n_x}$ be an indefinite symmetric matrix. The (StQO) problem is equivalent to the following problem

$$\begin{aligned} &\max_{\tau\in\mathbb{R}} \; \tau \\ &\text{s.t. } \forall x \in \Delta,\; \exists (u_x \in \mathbb{R}^{n_x}, w_x \in \mathbb{R}) : \begin{cases} \frac{1}{2}x^\top Q^+ x - \frac{1}{2}u_x^\top Q^+ u_x + w_x \geq \tau, \\ -Q^+ u_x + e w_x \leq Q^- x, \end{cases} \end{aligned} \tag{ARO-StQO}$$

where τ is the static variable, $x \in \Delta$ is the uncertain parameter, and $(u_x, w_x) \in \mathbb{R}^{n_x} \times \mathbb{R}$ is the adjustable variable.

According to Theorem 3, applying a hybrid decision rule (i.e., $u_x = u$ and $w_x = Zx + z$) to (ARO-StQO) is equivalent to:

$$\begin{aligned} \min_{\substack{\gamma\in\mathcal{S}^{n_x} \\ x\in\mathbb{R}^{n_x}}} \; & x^\top Q^+ x + \sum_{i,j=1}^{n_x} Q^-_{ij}\gamma_{ij} \\ \text{s.t.} \; & e^\top x = 1, \\ & e^\top \gamma = x^\top, \\ & x \geq 0,\; \gamma \geq 0. \end{aligned} \tag{18}$$

Selvi et al. (2023) show that any optimal solution (x^*, γ^*) of (18) satisfies the sparsity pattern $\gamma^* = \text{Diag}(x^*)$. Consequently, the number of decision variables in (18) can be reduced. Moreover, they show that adding a semi-definite constraint $\gamma \succeq x^\top x$ offers no advantage over the relaxed problem (18). Therefore, the lower bound obtained from the hybrid decision rule on (ARO-StQO) is equivalent to

$$\begin{aligned} \min_{x} \; & x^\top Q^+ x + \text{diag}(Q^-)^\top x \\ \text{s.t.} \; & e^\top x = 1, \\ & x \geq 0. \end{aligned} \tag{19}$$

Another equivalent formulation to (StQO) is to apply FME and remove the wait-and-see variable w_x:

$$\begin{aligned} &\max_{\tau\in\mathbb{R}} \ \tau \\ &\text{s.t. } \forall x\in\Delta,\ \exists u_x:\ \tfrac{1}{2}x^\top Q^+ x - \tfrac{1}{2}u_x^\top Q^+ u_x + (Q^-)_i x + (Q^+)_i u_x \geq \tau,\ \ i=1,\dots,n_x. \end{aligned} \tag{FME-StQO}$$

The next proposition establishes a connection between the above ARO reformulation of (StQO) and its standard reformulation-linearization technique.

PROPOSITION 1. *In* (FME-StQO) *problem, applying the decision rule $u_x = x$ is equivalent to applying standard reformulation-linearization technique relaxation to* (StQO)*, i.e.,*

$$\begin{aligned} \min_{\gamma,x}\ & \sum_{i,j=1}^{n_x} Q_{ij}\gamma_{ij} \\ \text{s.t.}\ \ & e^\top x = 1, \\ & e^\top \gamma = x^\top, \\ & x\geq 0,\ \gamma\geq 0, \end{aligned} \tag{RLT-StQO}$$

with the optimal value of $\min_{1\leq i,j\leq n_x} Q_{ij}$.

Proof. Applying the decision rule $u_x = x$ results in

$$\begin{aligned} &\max_{\tau\in\mathbb{R}} \tau \\ &\text{s.t.}\quad Q_i x \geq \tau, \quad \forall x\in\Delta, \quad i=1,\dots,n_x, \end{aligned} \tag{20}$$

which is equivalent to

$$\begin{aligned} &\max_{\tau\in\mathbb{R}} \tau \\ &\text{s.t.}\quad \min_{x\in\Delta} Q_i x \geq \tau, \quad i=1,\dots,n_x. \end{aligned} \tag{L1-StQO}$$

The extreme points of the unit simplex Δ are represented by $\{e_j\}_{j=1}^{n_x}$. Since the optimal values are among the extreme point, (L1-StQO) is equivalent to $\min_{1\leq i,j\leq n_x} Q_{ij}$. Note that the optimal value of (RLT-StQO) is also $\min_{1\leq i,j\leq n_x} Q_{ij}$, since

$$\sum_{i,j=1}^{n_x} Q_{ij}\gamma_{ij} \geq \sum_{i,j=1}^{n_x} \left(\min_{1\leq i,j\leq n_x} Q_{ij}\right)\gamma_{ij} = \left(\min_{1\leq i,j\leq n_x} Q_{ij}\right)\sum_{i,j=1}^{n_x}\gamma_{ij} = \min_{1\leq i,j\leq n_x} Q_{ij},$$

where the first inequality holds because each Q_{ij} is at least as large as $\min_{1\leq i,j\leq n_x} Q_{ij}$, the second equality holds because $\min_{1\leq i,j\leq n_x} Q_{ij}$ is a constant factor, and the last equality holds due to the constraints $e^\top\gamma = x^\top$ and $e^\top x = 1$, which together enforce that the sum of all entries of γ must equal 1. Thus, $\min_{1\leq i,j\leq n_x} Q_{ij}$ is a lower bound on the optimal value of (RLT-StQO). Let $Q_{rs} := \min_{1\leq i,j\leq n_x} Q_{ij}$. Clearly, the solution $(\bar{\gamma},\bar{x})$, where

$$\bar{\gamma}_{ij} := \begin{cases} 1, & \text{if } i=r \text{ and } j=s \\ 0, & \text{otherwise} \end{cases} \qquad \text{and} \qquad \bar{x} := e_r,$$

is feasible for (RLT-StQO) with the objective value of Q_{rs}, hence (RLT-StQO) has the optimal value of $\min_{1\leq i,j\leq n_x} Q_{ij}$ which is the same applying the deicion rule $u_x = x$ to (FME-StQO). □

In Proposition 1, we see that applying the special linear decision rule $u_x = x$ yields a simple lower bound (LB $= \min_{1\le i,j\le n_x} Q_{ij}$) to the optimal value of (StQO) problem. This simple lower bound can be tight if and only if the minimum entry of Q is located on the diagonal; otherwise, the approximation is not tight.

Adding the additional semi-definite constraint $\gamma \succeq x^\top x$ in (RLT-StQO) results in a progressively tighter bound. However, this comes at the expense of significant computational demands (Selvi et al. 2023). For large-sized problems, this bound frequently encounters "out-of-memory" errors during the solving process, making it generally inapplicable.

To implement the ARO-QO Algorithm, we use (L1-StQO) in Step 1 as a lower bound approximation. In Step 2 of the ARO-QO Algorithm, we collect a finite set of worst-case scenarios in two ways. **Scenario Based on L1-StQO**. To select scenarios we can use the following optimization problems

$$\bar{x}^i \in \arg\min_{x\in\Delta} \{Q_i x\}, \quad i = 1,\dots,n_x. \tag{21}$$

However, we do not need to solve linear optimization problems in (21) to find $\{\bar{x}^i\}_{i=1}^{n_x}$, as we only need to consider the extreme points of the unit-simplex set, i.e., $\{e_i\}_{i=1}^{n_x}$. These points provide the natural upper bound (i.e., ${e_i}^\top Q e_i = Q_{ii}$), which exists in the literature, see (Gondzio and Yıldırım 2021, Lemma 2.1).

To collect additional worst-case scenarios beyond the extreme points of the unit simplex, we also use (FME-StQO), where applying a constant decision rule on u_x (i.e., $u_x = u$) results in

$$\begin{aligned} \max_{\tau\in\mathbb{R},u\in\mathbb{R}^{n_x}} \quad & \tau \\ \text{s.t.} \quad & \tfrac{1}{2}x^\top Q^+ x - \tfrac{1}{2}u^\top Q^+ u + (Q^-)_i x + (Q^+)_i u \ge \tau. \quad \forall x\in\Delta \quad i = 1,\dots,n_x \end{aligned} \tag{L2-StQO}$$

Scenario Based on L2-StQO. Using this approximation, we can use the following problem to generate scenarios

$$\hat{x}^i \in \arg\min_{x\in\Delta} \left\{\tfrac{1}{2}x^\top Q^+ x + Q_i^- x\right\}, \quad i = 1,\dots,n_x. \tag{22}$$

We denote by $x^{(0)}$ the scenario with the lowest objective value from all scenarios, i.e.,

$$x^{(0)} \in \arg\min_{x} \left\{x^\top Q x \,\middle|\, x \in \{\hat{x}^i\}_{i=1}^{n_x} \cup \{e_i\}_{i=1}^{n_x}\right\}. \tag{23}$$

After choosing $x^{(0)}$ from either of the approaches above, we move to Step 4 of the Algorithm. In this step by using this initial point, we can improve this initial solution, to get the candidate solution, with its corresponding objective value serving as an upper bound.

Gondzio and Yıldırım (2021) propose an exact mixed-integer linear optimization (MILO) reformulation for (StQO):

$$\begin{aligned} \min_{x,y,z,\alpha} \quad & \alpha \\ \text{s.t.} \quad & e^\top x = 1, \\ & Qx \le e\alpha + z, \\ & 0 \le x \le y, \\ & 0 \le z_i \le (1-y_i)\, M_i, \quad i = 1,\dots,n_x \\ & y \in \{0,1\}^{n_x}, \end{aligned} \tag{MILO-StQO}$$

where non-negative parameter $M \in \mathbb{R}^{n_x}$ depends on a lower bound of the optimal value of (StQO), ensuring that the optimal value of (MILO-StQO) is equivalent to the optimal value of (StQO).

We will explore how to enhance the solution quality of the (MILO-StQO) formulation by integrating the ARO-QO Algorithm.

Problem Instances It is of paramount importance to note that with a high probability, global solutions of randomly generated (StQO) instances are located either at vertices or edges of the standard simplex (Bomze et al. 2018). In order to make a fair comparison, we do not generate naive random instances in our study, as our upper-bound methodology would be optimal in these cases. Instead, we concentrate on using instances from well-known datasets or employing their patterns to generate new instances, as outlined by Liuzzi et al. (2019).

Detailed Results We consider 15 groups of instances with the dimension n_x taking values in $\{500, 600, 700, 800, 900\}$. These test problems were generated using the pattern described in Liuzzi et al. (2019) and Scozzari and Tardella (2008). Another important factor affecting the performance of different solution methods, is the density of Matrix Q. We examine three density values for each dimension: 50%, 75%, and 90%. For further details, we refer the reader to (Scozzari and Tardella 2008).

In the initial comparison, we evaluate the quality of the ARO-based upper bound approximation against CPLEX local search. Setting `cplex.optimalitytarget` option to 2 allows CPLEX to solve indefinite quadratic optimization problems locally. This configuration guarantees solutions that satisfy first-order optimality conditions, but not necessarily global optimality.

Table 3 provides statistics on the solution gap as well as the time it takes for ARO-QO and CPLEX to obtain a solution. As one can see, the ARO-QO Algorithm demonstrates frequently better performance compared to CPLEX local search in terms of upper bound quality and computational efficiency. More specifically, the ARO-QO Algorithm outperforms CPLEX local search in 46 out of 75 instances, while CPLEX local search produces better upper bounds in only 28 instances, with one instance resulting in equal performance (see Table 9 in Appendix D). This comparison highlights the ARO-QO Algorithm's consistent ability to generate good upper bounds for most test instances. Overall, across all instances, the average solution gaps are 4.15% and 5.75% (with the maximum value of 14.53% and 19.73%), respectively for ARO-QO and CPLEX. From a solution time perspective, the ARO-QO Algorithm also exhibits better performance. It consistently solves instances more quickly. In Table 3, we see that ARO-QO Algorithm outperforms CPLEX Local in solution quality for 11 out of 15 problem groups, particularly excelling in larger instances.

Given that our approach obtained a good solution, we use it to improve the performance of (MILO-StQO) reformulation. Specifically, since the objective function of (MILO-StQO) is linear, we can restrict the objective function to only get values between the valid lower and upper bounds, i.e., LB $\leq \alpha \leq$ UB,

Table 3 Comparison of solution gaps and solver times for standard quadratic optimization instances between ARO-QO Algorithm and CPLEX local search.

	ARO-QO Algorithm		CPLEX local search	
Group (Dimension, density)	SGap	Time	SGap	Time
#1 ($n_x = 500, d = 0.50$)	**5.91** [3.14]	7.38 [0.76]	13.69 [4.94]	6.22 [1.17]
#2 ($n_x = 500, d = 0.75$)	5.07 [2.76]	6.30 [1.08]	**2.88** [3.81]	4.94 [0.79]
#3 ($n_x = 500, d = 0.90$)	2.96 [2.06]	5.87 [0.90]	**2.87** [1.36]	4.91 [1.00]
#4 ($n_x = 600, d = 0.50$)	**4.57** [5.09]	13.12 [4.34]	5.85 [3.76]	10.56 [2.54]
#5 ($n_x = 600, d = 0.75$)	**3.19** [2.58]	8.79 [1.96]	3.69 [1.96]	11.30 [2.63]
#6 ($n_x = 600, d = 0.90$)	5.02 [2.45]	10.72 [3.83]	**3.66** [1.29]	9.73 [2.11]
#7 ($n_x = 700, d = 0.50$)	**6.05** [2.89]	15.00 [3.75]	7.26 [3.16]	14.10 [1.28]
#8 ($n_x = 700, d = 0.75$)	**4.92** [1.48]	10.61 [1.13]	5.42 [1.71]	13.25 [0.82]
#9 ($n_x = 700, d = 0.90$)	2.67 [1.58]	10.55 [3.19]	**2.56** [1.50]	13.82 [0.94]
#10 ($n_x = 800, d = 0.50$)	**3.65** [2.03]	25.05 [9.89]	9.36 [1.94]	23.81 [4.52]
#11 ($n_x = 800, d = 0.75$)	**2.88** [2.15]	12.51 [3.96]	5.26 [4.03]	22.67 [5.01]
#12 ($n_x = 800, d = 0.90$)	**1.59** [1.30]	12.10 [1.91]	2.76 [2.19]	27.28 [7.75]
#13 ($n_x = 900, d = 0.50$)	**9.34** [3.62]	20.31 [0.95]	12.31 [5.90]	34.32 [5.61]
#14 ($n_x = 900, d = 0.75$)	**2.79** [1.73]	14.23 [2.01]	3.99 [2.06]	33.61 [5.05]
#15 ($n_x = 900, d = 0.90$)	**1.73** [2.16]	13.91 [1.04]	4.68 [2.01]	35.05 [9.31]

Notes: This table categorizes problems into groups in the first column. The subsequent columns display "mean [standard deviation]" values of each subgroup's solution gap and solver time. Bold numbers indicate the best solution gap of these two approaches per group.

potentially accelerating convergence and improving solution quality by guiding the solver towards more promising areas of the search space. It is worth mentioning that Gondzio and Yıldırım (2021) used UB := $\min_{1\leq i\leq n_x} Q_{ii}$ as a natural upper bound, which corresponds to the best objective value achievable for (StQO) when restricted to the vertices of the unit simplex. For the lower bound, they used a tight SDO-based bound. However, in our instances, due to out-of-memory errors, this type of bound is not applicable, so we use (L1-StQO) as a valid lower bound. We remark that there exist other lower bounds in the literature (Bomze et al. 2008, Liuzzi et al. 2019).

Based on the upper bound UB obtained from ARO-QO Algorithm, we first generate a feasible solution $(\bar{x}, \bar{y}, \bar{z}, \bar{\alpha})$ for (MILO-StQO) such that $\bar{\alpha} \leq$ UB. To do so, we use CPLEX, since it has fast heuristic algorithms to find a feasible solution for MILOs. We then use this solution as a warm starting point for (MILO-StQO) and add the extra objective cut LB$\leq \alpha$.

In this part of the numerical results, we compare the following approaches to solve (StQO): the original (MILO-StQO), its variant with the additional LB and UB cuts (MILO-L-U), where UB is the natural upper bound (like as in Gondzio and Yıldırım (2021)), and its variant with LB cut and ARO-based warm start (ARO-QO-MILO). Additionally, we compare these with global solvers Gurobi and CPLEX. All (MILO-StQO) versions are solved using Gurobi as the MILO solver.

Table 4 shows the statistics of the results. In this table, the first column represents the group of instances, including the number of variables (n_x) and the density (d). The subsequent columns are organized into

five different solution methods. For each method, three performance metrics are reported: optimality gap, solution gap, and solver time in seconds. Each cell contains the mean value and standard deviation [between brackets]. Bold numbers indicate the best (lowest) solution gap, while underlined numbers show the best (lowest) optimality gap for each group. An asterisk (*) in the time column indicates that the method is terminated due to the time restriction.

Table 4 Statistic of solution and optimality gaps and solver times for standard quadratic optimization instances

	MILO			MILO-L-U			ARO-QO-MILO			Gurobi			CPLEX		
Group (Dimension, density)	OGap	SGap	Time	OGap	SGap	Time	OGap	SGap	Time	OGap	SGap	Time	OGap	SGap	Time
#1 ($n_x = 500, d = 0.50$)	0.00 [0.00]	**0.00** [0.00]	1929.83 [87.33]	0.00 [0.00]	**0.00** [0.00]	826.85 [133.51]	0.00 [0.00]	**0.00** [0.00]	887.20 [129.78]	1972.38 [1850.29]	7.31 [3.44]	3000*	61772219.07 [54491734.55]	957.04 [766.36]	3000*
#2 ($n_x = 500, d = 0.75$)	2.55 [2.33]	**0.00** [0.00]	2930.34 [102.62]	0.00 [0.00]	**0.00** [0.00]	2124.75 [296.42]	0.00 [0.00]	**0.00** [0.00]	2191.32 [263.73]	26.98 [4.43]	2.77 [2.21]	3000*	92672786.53 [82304244.51]	987.47 [781.11]	3000*
#3 ($n_x = 500, d = 0.90$)	7.32 [2.78]	**0.00** [0.00]	3000*	4.85 [2.91]	**0.00** [0.00]	2960.18 [89.04]	4.93 [2.96]	**0.00** [0.00]	2872.41 [285.29]	26.23 [4.02]	3.81 [2.78]	3000*	111064986.98 [98903826.58]	1010.48 [820.30]	3000*
#4 ($n_x = 600, d = 0.50$)	176.76 [5.92]	0.25 [0.34]	3000*	0.00 [0.00]	**0.00** [0.00]	2160.33 [177.22]	0.00 [0.00]	**0.00** [0.00]	2241.34 [181.76]	2687.31 [1691.32]	8.69 [2.58]	3000*	86173185.08 [60661256.69]	929.23 [581.79]	3000*
#5 ($n_x = 600, d = 0.75$)	55.02 [80.93]	0.55 [0.58]	3000*	11.17 [0.92]	0.18 [0.37]	3000*	12.74 [4.64]	**0.00** [0.00]	3000*	1955.29 [2667.52]	4.54 [2.33]	3000*	129433672.53 [91250582.32]	967.01 [617.40]	3000*
#6 ($n_x = 600, d = 0.90$)	15.91 [1.43]	0.04 [0.08]	3000*	12.55 [1.32]	0.36 [0.62]	3000*	12.02 [1.14]	**0.03** [0.07]	3000*	27.81 [5.44]	4.42 [3.08]	3000*	155111853.68 [109405777.34]	982.44 [622.62]	3000*
#7 ($n_x = 700, d = 0.50$)	182.14 [3.99]	0.91 [1.18]	3000*	8.84 [3.04]	0.03 [0.06]	3000*	9.59 [1.95]	**0.00** [0.00]	3000*	9426.52 [955.51]	5.06 [4.97]	3000*	374534570.36 [600759655.77]	2827.11 [4394.98]	3000*
#8 ($n_x = 700, d = 0.75$)	206.24 [4.23]	1.36 [1.58]	3000*	16.35 [1.99]	**0.60** [0.56]	3000*	17.21 [1.49]	1.18 [1.57]	3000*	15809.10 [4620.84]	2.10 [1.95]	3000*	561860313.36 [900673927.18]	2893.43 [4487.49]	3000*
#9 ($n_x = 700, d = 0.90$)	213.46 [2.59]	1.14 [1.12]	3000*	16.56 [1.16]	**0.00** [0.00]	3000*	17.93 [3.01]	0.28 [0.43]	3000*	15804.48 [1280.84]	6.25 [2.32]	3000*	673291316.71 [1078887498.54]	2905.61 [4492.97]	3000*
#10 ($n_x = 800, d = 0.50$)	185.63 [2.43]	0.29 [0.46]	3000*	20.33 [2.14]	0.30 [0.67]	3000*	20.79 [1.69]	**0.17** [0.15]	3000*	17214.45 [718.18]	7.21 [3.56]	3000*	213944685.06 [256201132.95]	1264.89 [1390.58]	3000*
#11 ($n_x = 800, d = 0.75$)	210.48 [2.12]	0.86 [0.77]	3000*	61.74 [88.49]	41.09 [89.05]	3000*	20.51 [0.90]	**0.07** [0.13]	3000*	24206.49 [3539.76]	5.06 [2.85]	3000*	321304714.38 [385347544.93]	1313.44 [1458.89]	3000*
#12 ($n_x = 800, d = 0.90$)	220.40 [3.61]	1.07 [0.99]	3000*	19.61 [0.77]	0.30 [0.60]	3000*	22.49 [5.68]	**0.12** [0.28]	3000*	26275.01 [4764.03]	4.33 [1.16]	3000*	385112553.77 [461880487.68]	1339.63 [1495.70]	3000*
#13 ($n_x = 900, d = 0.50$)	185.11 [2.21]	0.21 [0.44]	3000*	25.17 [0.70]	0.31 [0.41]	3000*	24.48 [1.43]	**0.03** [0.07]	3000*	23731.50 [1504.65]	8.58 [4.06]	3000*	187345502.54 [68002456.84]	913.82 [300.76]	3000*
#14 ($n_x = 900, d = 0.75$)	212.18 [3.16]	1.34 [1.33]	3000*	23.67 [1.11]	0.51 [0.66]	3000*	23.54 [0.95]	**0.00** [0.00]	3000*	33231.50 [1583.20]	3.40 [2.49]	3000*	281072681.59 [102072591.74]	939.96 [307.88]	3000*
#15 ($n_x = 900, d = 0.90$)	225.00 [1.14]	2.15 [1.09]	3000*	21.93 [1.51]	**0.47** [0.71]	3000*	22.68 [0.64]	0.61 [1.36]	3000*	39919.59 [571.37]	3.18 [0.99]	3000*	336989667.40 [122404416.52]	956.45 [307.51]	3000*

Notes: This table categorizes problems into groups in the first column. The subsequent columns display "mean [standard deviation]" values of each subgroup's solution and optimality gaps and solver time. Bold numbers indicate the best solution gap, while underlined numbers show the best optimality gap per group.

For small instances ($n_x = 500$), all MILO variants perform well, often with zero gaps. MILO-L-U and ARO-QO-MILO are typically faster. As problem size increases, all methods reach the time limit, and performance differences become clearer. ARO-QO-MILO significantly improves solution quality, consistently achieving the best optimality and solution gaps, even for larger problems. Detailed results are in Table 9, Appendix D.

The ARO-QO-MILO's benefits are evident in its consistent outperformance of MILO-L-U, especially regarding solution quality. In contrast, both Gurobi and CPLEX struggle with larger instances, consistently producing higher optimality and solution gaps compared to the MILO variants. CPLEX, in particular, shows extremely large optimality gaps for all problem sizes. These results suggest that ARO-QO-MILO is promising for enhancing the performance of MILO reformulations for quadratic optimization problems. This method is particularly valuable for larger instances where global solvers may struggle.

5.3. General Quadratic Optimization

In our final experiment, we consider a general quadratic optimization problem

$$\begin{aligned} \min_{x \geq 0} \ & x^\top Q x + c^\top x \\ \text{s.t.} \ & Ax = b, \end{aligned} \tag{24}$$

where $Q \in \mathbb{R}^{n_x \times n_x}$ is an indefinite matrix, $c \in \mathbb{R}^{n_x}$, $A \in \mathbb{R}^{m_x \times n_x}$, and $b \in \mathbb{R}^{m_x}$ are given.

From Theorem 2, we know that the above problem can be rewritten as

$$\begin{aligned} & \max_{\tau \in \mathbb{R}} \ \tau \\ & \text{s.t.} \ \forall x \in \mathcal{X}, \ \exists (u_x, w_x) : \begin{cases} \frac{1}{2} x^\top Q^+ x + \frac{1}{2} c^\top x - \frac{1}{2} u_x^\top Q^+ u_x + b^\top w_x \geq \tau, \\ A^\top w_x - Q^+ u_x \leq Q^- x + \frac{1}{2} c. \end{cases} \end{aligned} \tag{ARO-QO}$$

We examine three distinct versions of the ARO-QO Algorithm, denoted as static ARO-QO, linear ARO-QO, and affine ARO-QO, each employing different decision policies. Static ARO-QO uses a static decision rule with $u_x = u$ and $w_x = w$, linear ARO-QO employs a decision rule with $u_x = x$ and $w_x = w$, while affine ARO-QO utilizes a special affine decision rule with $u_x = x$ and $w_x = Zx + z$, where $u \in \mathbb{R}^{n_x}$, $w \in \mathbb{R}^{m_x}$, $Z \in \mathbb{R}^{m_x \times n_x}$, and $z \in \mathbb{R}^{m_x}$ are static variables. To implement the ARO-QO Algorithm and its variants, we use MOSEK as a convex optimization solver.

In Xia et al. (2020), the authors propose an exact MILO reformulation of problem (24)

$$\begin{array}{ll} \min\limits_{x,\mu,z,\lambda} & \frac{1}{2}(c^\top x - b^\top \mu) \\ \text{s.t.} & Ax = b, \\ & 2Qx + c + A^\top \mu - \lambda = 0, \\ & 0 \le x_i \le z_i U_i, \qquad\qquad i = 1, \cdots, n_x \\ & 0 \le \lambda_i \le (1 - z_i)\, V_i, \qquad i = 1, \cdots, n_x \\ & z \in \{0,1\}^{n_x}. \end{array} \tag{MILO-QO}$$

This reformulation is based on the KKT conditions. The authors introduce binary variables z and big-M parameters (U and V) to linearize the non-convex complementarity constraints in the KKT conditions. The parameters $U \in \mathbb{R}^{n_x}$ and $V \in \mathbb{R}^{n_x}$ are determined by solving a series of linear optimization problems. For a more detailed explanation of this reformulation technique, see Xia et al. (2020). In the remainder of this section, we consider (MILO-QO) as a benchmark approach and utilize Gurobi to solve it.

Problem Instances We generate a square matrix Q using the formula $Q = P\Lambda P^\top$, where P is an orthogonal matrix obtained from the QR decomposition of a random matrix, and Λ is a diagonal matrix with components sampled uniformly from the interval $[-5, 5]$. We know that (QO) is NP-hard even when Q has only one negative eigenvalue (Pardalos and Vavasis 1991). To explore the problem's behavior across different scenarios, we generate Q in three cases based on its convexity rank (which we define as the number of positive eigenvalues). We consider low, medium, and high convexity ranks, corresponding to 10%, 50%, and 90% of the eigenvalues being positive, respectively. We also generate the matrix A and the vector c with components from the interval $[-5, 5]$ uniformly. To ensure the feasible region is bounded and feasible, we add a row of ones to the matrix A and set $b := A(\frac{1}{2}e)$.

Detailed Results We consider 15 groups each of which contains five instances, with the dimension n_x taking values in $\{20, 50, 100, 200, 1000\}$. To ensure a large feasible set, we set the number of constraints m_x to $\frac{n_x}{5}$. These test problems were generated using the mentioned pattern with the same convexity rank in each group.

The statistics of the results of solution gaps, optimality gaps, and solver times are presented in Table 5, where the first column in each represents the group of instances with the number of variables (n_x), number of constraints (m_x), and convexity rank of the matrix Q (p), respectively.

Static ARO-QO variants consistently outperform the CPLEX local search in terms of solution quality. Static ARO-QO with Representation 2 and Representation 3 achieve average solution gaps of 2.80% and

2.08% respectively, compared to 11.15% for CPLEX local search across all instances. The superiority of Static ARO-QO is further emphasized by the maximum solution gaps observed: 34.26% for Representation 2, 28.66% for Representation 3, versus a substantial 92.53% for CPLEX local search.

Table 5 Statistics of solution and optimality gaps and solver times for general quadratic optimization instances

	Static ARO-QO R2			Static ARO-QO R3			Linear ARO-QO			Affine ARO-QO		
Group (n_x, m_x, p)	OGap	SGap	Time	OGap	SGap	Time	OGap	SGap	Time	OGap	SGap	Time
#1 (20, 4, 0.10)	137.61 [48.82]	**0.00** [0.00]	0.06	19.04 [5.43]	**0.00** [0.00]	0.06	19.03 [11.54]	2.05 [4.34]	0.06	0.02 [0.05]	**0.00** [0.00]	0.07
#2 (20, 4, 0.50)	280.19 [73.76]	2.32 [5.18]	0.08	48.11 [20.31]	2.32 [5.18]	0.07	70.53 [37.49]	3.74 [7.77]	0.08	23.02 [22.51]	5.16 [7.81]	0.10
#3 (20, 4, 0.90)	833.48 [493.99]	0.08 [0.17]	0.15	75.88 [41.82]	0.08 [0.17]	0.08	541.08 [381.12]	0.08 [0.17]	0.14	374.41 [376.78]	**0.00** [0.00]	0.13
#4 (50, 10, 0.10)	244.47 [26.88]	0.98 [1.36]	0.17	50.27 [8.82]	0.69 [1.33]	0.17	43.87 [8.87]	0.55 [1.03]	0.17	3.92 [5.12]	0.76 [1.48]	0.74
#5 (50, 10, 0.50)	449.48 [108.27]	9.94 [13.52]	0.25	78.66 [12.53]	0.89 [1.27]	0.20	92.78 [26.70]	3.13 [5.39]	0.26	21.83 [19.78]	4.01 [6.41]	0.73
#6 (50, 10, 0.90)	808.97 [502.13]	1.89 [2.59]	0.36	116.58 [63.07]	0.88 [1.97]	0.23	368.43 [225.03]	1.43 [3.19]	0.35	179.62 [130.55]	1.01 [2.25]	0.81
#7 (100, 20, 0.10)	340.16 [22.89]	1.48 [2.63]	0.42	77.63 [11.08]	0.30 [0.44]	0.46	70.12 [10.31]	**0.09** [0.12]	0.43	18.41 [14.43]	4.01 [6.77]	226.32
#8 (100, 20, 0.50)	643.25 [107.03]	7.36 [4.96]	0.76	144.03 [28.07]	6.93 [6.82]	0.66	139.20 [15.40]	**0.03** [0.06]	0.66	56.04 [25.31]	6.84 [10.65]	103.21
#9 (100, 20, 0.90)	1276.85 [336.77]	**0.00** [0.00]	0.77	172.77 [40.76]	5.73 [12.82]	0.73	573.13 [169.95]	4.88 [10.92]	0.78	266.75 [93.59]	**0.00** [0.00]	32.96
#10 (200, 40, 0.10)	534.64 [12.50]	**0.99** [2.05]	1.67	144.53 [6.94]	1.80 [2.47]	1.95	130.91 [5.08]	1.01 [2.25]	1.71	60.60 [6.98]	4.76 [6.09]	3000*
#11 (200, 40, 0.50)	930.02 [93.46]	9.51 [14.14]	1.95	208.84 [21.82]	5.56 [8.31]	2.84	225.05 [35.68]	7.49 [5.33]	2.27	104.62 [31.57]	9.03 [10.92]	3000*
#12 (200, 40, 0.90)	1942.98 [277.56]	**0.00** [0.00]	2.56	259.38 [40.42]	1.16 [2.58]	4.08	788.83 [153.96]	6.83 [15.26]	2.37	390.98 [66.68]	**0.00** [0.00]	3000*
#13 (1000, 200, 0.10)	1307.11 [34.89]	1.89 [1.97]	236.02	389.64 [12.03]	1.18 [1.78]	248.70	357.28 [14.00]	**0.12** [0.22]	263.10	-	-	3000*
#14 (1000, 200, 0.50)	2018.56 [113.24]	5.61 [4.09]	310.89	511.58 [19.65]	3.69 [3.73]	381.05	496.19 [12.05]	6.87 [2.93]	320.57	-	-	3000*
#15 (1000, 200, 0.90)	3882.46 [240.03]	**0.00** [0.00]	293.19	538.49 [37.86]	0.02 [0.04]	359.27	1275.71 [108.36]	0.38 [0.85]	279.80	-	-	3000*

	MILO-QO			Gurobi			CPLEX			CPLEX Local	
	OGap	SGap	Time	OGap	SGap	Time	OGap	SGap	Time	SGap	Time
#1 (20, 4, 0.10)	0.00 [0.00]	**0.00** [0.00]	0.39	0.00 [0.00]	**0.00** [0.00]	0.11	0.00 [0.00]	**0.00** [0.00]	0.13	30.14 [21.57]	0.06
#2 (20, 4, 0.50)	0.00 [0.00]	**0.00** [0.00]	0.40	0.00 [0.00]	**0.00** [0.00]	0.19	0.00 [0.00]	**0.00** [0.00]	0.35	15.82 [22.17]	0.03
#3 (20, 4, 0.90)	0.00 [0.00]	**0.00** [0.00]	0.51	0.01 [0.00]	**0.00** [0.00]	6.38	0.00 [0.00]	**0.00** [0.00]	0.19	**0.00** [0.00]	0.03
#4 (50, 10, 0.10)	0.00 [0.00]	**0.00** [0.00]	360.72	0.00 [0.00]	**0.00** [0.00]	6.22	0.00 [0.00]	**0.00** [0.00]	13.43	31.01 [18.93]	0.06
#5 (50, 10, 0.50)	0.00 [0.00]	**0.00** [0.00]	394.57	0.00 [0.00]	0.01 [0.02]	35.55	0.00 [0.00]	0.01 [0.02]	468.09	12.37 [12.55]	0.05
#6 (50, 10, 0.90)	30.85 [68.99]	10.55 [23.60]	1508.81	0.02 [0.03]	**0.00** [0.00]	1050.84	73.72 [84.39]	0.88 [1.97]	3000*	1.89 [2.59]	0.04
#7 (100, 20, 0.10)	61940.57 [4389.58]	88.45 [24.09]	3000*	73.66 [39.67]	16.44 [16.27]	3000*	239.12 [293.14]	11.88 [11.41]	3000*	33.38 [35.71]	0.22
#8 (100, 20, 0.50)	-	-	3000*	1144.23 [324.05]	10.08 [7.76]	3000*	3225.46 [423.43]	8.42 [7.30]	3000*	5.67 [6.78]	0.11
#9 (100, 20, 0.90)	-	-	3000*	1234.60 [466.02]	**0.00** [0.00]	3000*	10943.59 [3055.57]	**0.00** [0.00]	3000*	**0.00** [0.00]	0.09
#10 (200, 40, 0.10)	-	-	3000*	4463.24 [403.29]	16.02 [14.95]	3000*	1124.49 [62.94]	10.67 [8.23]	3000*	16.56 [5.91]	1.13
#11 (200, 40, 0.50)	-	-	3000*	11959.44 [1780.21]	5.40 [5.89]	3000*	8056.26 [1444.42]	5.31 [6.89]	3000*	**5.01** [5.03]	0.57
#12 (200, 40, 0.90)	-	-	3000*	25562.79 [3323.87]	1.41 [3.15]	3000*	33881.59 [3944.54]	2.95 [4.05]	3000*	1.41 [3.15]	0.32
#13 (1000, 200, 0.10)	-	-	3000*	-	-	3000*	-	-	3000*	10.88 [4.80]	387.96
#14 (1000, 200, 0.50)	-	-	3000*	-	-	3000*	-	-	3000*	**3.08** [3.54]	186.89
#15 (1000, 200, 0.90)	-	-	3000*	-	-	3000*	-	-	3000*	0.02 [0.04]	49.56

Notes. Problems are grouped by dimensions (n_x, m_x) and convexity rank (p) in the first column. Columns report "mean [standard deviation]" for solution/optimality gaps and solver times per subgroup. The symbol "–" indicates bounds not computed within the time limit. Bold denotes the best solution gap; underlined denotes the best optimality gap per group.

For smaller instances (Groups 1-6), all methods perform well, with MILO-QO, Gurobi, and CPLEX often achieving optimal solutions. The ARO methods show progressive improvement as their complexity increases from static to linear and then affine variants, specifically lower bound for low convexity rank groups.

For medium-sized instances (Groups 7-12), a significant performance shift becomes apparent. MILO-QO struggles, often failing to find solutions within the time limit. While Gurobi and CPLEX do find solutions, they generally exhibit higher optimality gaps than the ARO-QO family. However, on larger problems, Gurobi and CPLEX outperform MILO-QO, frequently finding solutions where MILO-QO fails. The ARO-QO family consistently achieves lower optimality gaps more rapidly. For example, in Group 7, the ARO-QO family consistently outperforms all others in terms of both optimality and solution gaps while maintaining reasonable computation times. Specifically, Affine ARO-QO achieves the best optimality gap, and Linear ARO-QO achieves the best solution gap. In contrast, MILO-QO, Gurobi, and CPLEX struggle to find good solutions within the given time limit, demonstrating the superiority of the ARO-QO family for this group of instances.

The ARO-QO Algorithm variants begin to outperform the MILO-QO reformulation and the global solvers Gurobi and CPLEX for medium-sized instances. This trend becomes even more evident for large instances

(Groups 13-15), where only Static ARO-QO and Linear ARO-QO consistently provide solutions and lower bounds within acceptable solver times, while MILO-QO, Gurobi, and CPLEX fail to provide any solutions for these instances.

The impact of convexity rank is evident across all problem sizes. With increasing p, the function approaches convexity, making it easier to obtain a good solution, as shown in Table 5. In contrast, non-convex QO problems with a lower convexity rank (i.e., a higher number of negative eigenvalues in Q) are generally more challenging to solve due to their complex landscape and the potential presence of multiple local optima, which complicates the search for the global optimum. However, a small value of p in ARO-QO often yields the best solutions as well. According to Section 5.1, we know that for $p = 0$, ARO outperforms to finding near-optimal solutions and acceptable optimality gaps. The results demonstrate that for $p = 0.1$, the optimality gap achieved by the ARO approaches remains small across all instances. As the problem size increases, and with lower p values, ARO methods maintain their effectiveness while other methods struggle or fail to provide viable solutions. This highlights the robustness of ARO approaches, especially in challenging scenarios where global solvers and MILO-QO reformulation falter. The consistent ability of ARO methods to deliver high-quality solutions with reasonable computational effort across diverse problem characteristics underscores their value in addressing complex quadratic optimization challenges.

6. Conclusions

We introduce a novel reformulation technique that enables the Quadratic Optimization problem (QO) to be recast as an Adjustable Robust Optimization problem (ARO). This process begins by demonstrating that any QO problem can be transformed into a disjoint bi-convex QO problem. Following this, we propose an equivalent ARO reformulation. Specifically, we illustrate that employing a so-called decision rule technique to approximate the ARO reformulation equates to using a reformulation-linearization technique on original QO problem. The ARO reformulation offers a new approach to solving non-convex QO problems by transferring the complexity from the original problem to its equivalent ARO counterpart. Specifically, in the concave QO problem, our ARO model transforms into a linear ARO, whereas in the indefinite QO problem, it becomes a non-linear ARO. Moreover, we develop an algorithm capable of identifying near-optimal solutions using our novel reformulations. We demonstrate the effectiveness of our ARO-based method in solving a class of quadratic optimization problems through numerical experiments, showing that it can yield high-quality solutions with reasonable computational costs. This established connection between QO and ARO provides a new perspective on addressing the challenges of non-convex QO problems and opens up new possibilities for further research in the field of mathematical optimization.

Acknowledgement

The authors would like to express their appreciation to Dick Den Hertog for the insightful discussions and feedback. Additionally, the first author is grateful for the guidance and support provided by Majid Soleimani-damaneh throughout this research.

Appendices

This appendix is divided into four sections. In the first section, we establish a relationship between the optimal solutions of (QO) and (Bi-QO) problems. In the second section, we illustrate when the finite scenario approach of ARO reformulation is tight. The third section summarizes all Adjustable Robust Optimization (ARO) reformulations of QO problems based on their feasible regions. The final section contains elaborated tables related to the numerical experiments.

A. On (Bi-QO) Refotmulation

The next proposition aims to establish a relation between the optimal solutions of (QO) and (Bi-QO) problems, showcasing how solutions from one problem can be used to obtain optimal solutions for the other.

PROPOSITION 2. *Let $Q = Q^+ + Q^-$ where $Q \in \mathbb{R}^{n_x \times n_x}$, and $Q^+, -Q^- \succ 0$. If x^* is an optimal solution of (QO), then (x^*, x^*) is an optimal solution of (Bi-QO). Moreover, if $(\hat{x}, \hat{y})$ is an optimal solution of (Bi-QO), then $\hat{x}$ and $\hat{y}$ are both optimal solutions of (QO) as well as $\hat{x} = \hat{y}$.*

Proof. Suppose x^* is an optimal solution to (QO). It is clear that (x^*, x^*) is also an optimal solution to (Bi-QO). To prove the reverse direction, assume that $(\hat{x}, \hat{y})$ is an optimal solution of (Bi-QO). Thus,

$$\begin{aligned} \tfrac{1}{2}\left(\hat{x}^\top Q^+ \hat{x} + \hat{y}^\top Q^+ \hat{y} + c^\top \hat{x} + c^\top \hat{y}\right) + \hat{x}^\top Q^- \hat{y} &\le \tfrac{1}{2}\left(\hat{x}^\top Q^+ \hat{x} + \hat{x}^\top Q^+ \hat{x} + c^\top \hat{x} + c^\top \hat{x}\right) + \hat{x}^\top Q^- \hat{x}, \\ \tfrac{1}{2}\left(\hat{x}^\top Q^+ \hat{x} + \hat{y}^\top Q^+ \hat{y} + c^\top \hat{x} + c^\top \hat{y}\right) + \hat{x}^\top Q^- \hat{y} &\le \tfrac{1}{2}\left(\hat{y}^\top Q^+ \hat{y} + \hat{y}^\top Q^+ \hat{y} + c^\top \hat{y} + c^\top \hat{y}\right) + \hat{y}^\top Q^- \hat{y}. \end{aligned} \tag{25}$$

Summing these inequalities results in

$$2\hat{x}^\top Q^- \hat{y} \le \hat{x}^\top Q^- \hat{x} + \hat{y}^\top Q^- \hat{y}. \tag{26}$$

Now, note that $\hat{x}^\top(-Q^-)\hat{y} = \left((-Q^-)^{\frac{1}{2}}\hat{x}\right)^\top \left((-Q^-)^{\frac{1}{2}}\hat{y}\right)$, where $(-Q^-)^{\frac{1}{2}}$ is the square roots of the matrix $(-Q^-)$. Therefore, we can apply the Cauchy-Schwarz inequality, which implies that

$$\begin{aligned} 2\hat{x}^\top(-Q^-)\hat{y} &\le 2\left\|(-Q^-)^{\frac{1}{2}}\hat{x}\right\| . \left\|(-Q^-)^{\frac{1}{2}}\hat{y}\right\| \\ &= 2\sqrt{\hat{x}^\top(-Q^-)\hat{x}}\ \sqrt{\hat{y}^\top(-Q^-)\hat{y}} \\ &\le \hat{x}^\top(-Q^-)\hat{x} + \hat{y}^\top(-Q^-)\hat{y}, \end{aligned} \tag{27}$$

where the reason for the last inequality is that, for any two non-negative scalars a and c, $2\sqrt{ac} \le (a+c)$. Hence, we have

$$2\hat{x}^\top Q^- \hat{y} \ge \hat{x}^\top Q^- \hat{x} + \hat{y}^\top Q^- \hat{y}. \tag{28}$$

Thus, by (26) and (28), we obtain that

$$2\hat{x}^\top Q^- \hat{y} = \hat{x}^\top Q^- \hat{x} + \hat{y}^\top Q^- \hat{y}, \tag{29}$$

which is equivalent

$$(\hat{x} - \hat{y})^\top Q^- (\hat{x} - \hat{y}) = 0. \tag{30}$$

From $-Q^- \succ 0$, we have $\hat{x} - \hat{y} = 0$, i.e. $\hat{x} = \hat{y}$. So, $\hat{x}$ is an optimal solution of (QO). □

The following example shows that if instead of $-Q^- \succ 0$, we have $-Q^- \succeq 0$ in the above proposition, then for an optimal solution $(\hat{x}, \hat{y})$ of (Bi-QO), we do not necessarily have $\hat{x} = \hat{y}$.

EXAMPLE 1. Consider an instance of (Bi-QO) with $c = (0,0,0)^\top$, $Q^+ = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$, $-Q^- = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \succeq 0$, and $\mathcal{X} = \{x \in \mathbb{R}^3 : 0 \le x \le e\}$. Then $(\hat{x}, \hat{y}) = ((0,0,1)^\top, (0,1,1)^\top)$ is an optimal solution to (Bi-QO) with optimal value -1; however, $\hat{x} \ne \hat{y}$. □

B. Finite Scenario Approach

One of the approximation approaches for the ARO problem is the Finite Scenario Approach (FSA). In this approach, we restrict ourselves to only finite scenarios in the uncertainty set, resulting in an upper bound to the optimal objective value of the ARO problem. The (FSA) is computationally efficient because it only considers a finite number of scenarios, rather than trying to optimize over all possible scenarios. This reduces the number of variables and constraints in the optimization problem, making it easier to solve. By identifying a set of potential scenarios, we are able to utilize this technique, which results in the following deterministic convex optimization problem:

$$\max_{\{u^k\}_k,\{w^k\}_k,\tau} \left\{ \tau \;\middle|\; \begin{array}{ll} \frac{1}{2}(x^k)^\top Q^+ x^k + \frac{1}{2}c^\top x^k + b^\top w^k - \frac{1}{2}(u^k)^\top Q^+ u^k \geq \tau, & k=1,\ldots,|\mathcal{W}| \\ -Q^+ u^k + A^\top w^k \leq Q^- x^k + \frac{1}{2}c, & k=1,\ldots,|\mathcal{W}| \end{array} \right\}, \tag{FSA-QO}$$

where $\mathcal{W} = \{x^1, \ldots, x^r\}$ is a finite sub-set of $\mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid Ax = b,\ x \geq 0\}$.

The following proposition states that if the optimal value of (FSA-QO) for a finite subset of scenarios is identical to the optimal value of (QO), then the optimal solution for (QO) must be included within that subset.

PROPOSITION 3. *Let $Q = Q^+ + Q^-$ where $Q \in \mathbb{R}^{n_x \times n_x}$, and $Q^+, -Q^- \succ 0$. If the optimal value of (FSA-QO) for a given finite subset of scenarios is equal to the optimal value of (QO), then the finite subset of scenarios contains the optimal solution for (QO).*

Proof. Let $\left(\bar{\tau}, \{\bar{u}^k\}_k, \{\bar{w}^k\}_k\right)$ be an optimal solution of problem (FSA-QO). Based on the definition of the optimality, there exists $x^s \in \mathcal{W} \subsetneq \mathcal{X}$ for which the following constraint of (FSA-QO) is binding:

$$\tfrac{1}{2}(x^s)^\top Q^+ x^s + \tfrac{1}{2}c^\top x^s + b^\top \bar{w}^s - \tfrac{1}{2}(\bar{u}^s)^\top Q^+ \bar{u}^s = \bar{\tau}.$$

We claim that x^s is an optimal solution of (QO). To show this, we have

$$\begin{aligned} \bar{\tau} &= \tfrac{1}{2}(x^s)^\top Q^+ x^s + \tfrac{1}{2}c^\top x^s + \max_{w,u} \left\{ b^\top w - \tfrac{1}{2}u^\top Q^+ u : -Q^+ u + A^\top w \leq Q^- x^s + \tfrac{1}{2}c \right\} \\ &= \tfrac{1}{2}(x^s)^\top Q^+ x^s + \tfrac{1}{2}c^\top x^s + \min_{z} \left\{ \tfrac{1}{2}z^\top Q^+ z + \tfrac{1}{2}c^\top z + z^\top Q^- x^s \mid z \in \mathcal{X} \right\}, \end{aligned} \tag{31}$$

The validity of the last equality follows from the fact that strong duality holds. If we denote the optimal solution of the above minimization problem by z^s, then we can conclude that (x^s, z^s) is optimal for (Bi-QO). Furthermore, according to Proposition 2, we have $x^s = z^s$, and therefore x^s is optimal for the original (QO) problem. □

C. Alternative Forms

A quadratic optimization problem can be expressed in different formulations, depending on how the feasible region is formulated, leading to variations in its ARO reformulation. Table 6 summarizes some of these formulations.

D. Detailed Results of Numerical Experiments

This appendix presents the outcomes derived from the numerical experiments conducted in the respective sections. The results showcased herein provide detailed insights into the findings obtained through various computational analyses and experiments discussed throughout this paper.

Table 6 Other Classes of ARO Reformulations of Indefinite QO Problem

Feasible Region	Type	ARO Problem
	I	
$\mathcal{X}=\{x\in\mathbb{R}^{n_x} \mid Ax=b,\ x\geq 0\}$		$\max\limits_{\tau\in\mathbb{R}}\ \tau$ $\text{s.t. } \forall x\in\mathcal{X},\ \exists(u_x,w_x): \begin{cases} \frac{1}{2}x^\top Q^+x+\frac{1}{2}c^\top x-\frac{1}{2}u_x^\top Q^+u_x+b^\top w_x\geq\tau, \\ A^\top w_x-Q^+u_x\leq Q^-x+\frac{1}{2}c. \end{cases}$
	II	
$\mathcal{X}=\{x\in\mathbb{R}^{n_x} \mid Ax=b\}$		$\max\limits_{\tau\in\mathbb{R}}\ \tau$ $\text{s.t. } \forall x\in\mathcal{X},\ \exists(u_x,w_x): \begin{cases} \frac{1}{2}x^\top Q^+x+\frac{1}{2}c^\top x-\frac{1}{2}u_x^\top Q^+u_x+b^\top w_x\geq\tau, \\ A^\top w_x-Q^+u_x=Q^-x+\frac{1}{2}c. \end{cases}$
	III	
$\mathcal{X}=\{x\in\mathbb{R}^{n_x} \mid Ax\geq b,\ x\geq 0\}$		$\max\limits_{\tau\in\mathbb{R}}\ \tau$ $\text{s.t. } \forall x\in\mathcal{X},\ \exists(u_x,w_x): \begin{cases} \frac{1}{2}x^\top Q^+x+\frac{1}{2}c^\top x-\frac{1}{2}u_x^\top Q^+u_x+b^\top w_x\geq\tau, \\ A^\top w_x-Q^+u_x\leq Q^-x+\frac{1}{2}c, \\ w_x\geq 0. \end{cases}$
	IV	
$\mathcal{X}=\{x\in\mathbb{R}^{n_x} \mid Ax\geq b\}$		$\max\limits_{\tau\in\mathbb{R}}\ \tau$ $\text{s.t. } \forall x\in\mathcal{X},\ \exists(u_x,w_x): \begin{cases} \frac{1}{2}x^\top Q^+x+\frac{1}{2}c^\top x-\frac{1}{2}u_x^\top Q^+u_x+b^\top w_x\geq\tau, \\ A^\top w_x-Q^+u_x=Q^-x+\frac{1}{2}c, \\ w_x\geq 0. \end{cases}$

Table 7 Detailed result on concave quadratic minimization instances from Selvi et al. (2022)

	Static ARO-QO			Partial affine ARO-QO			Full affine ARO-QO			Selvi et al. (2022)			Ben-Tal and Roos (2022)		Gurobi			CPLEX		
Problem	LB	UB	Time	LB	UB	Time	LB	UB	Time	LB	UB	Time	UB	Time	LB	UB	Time	LB	UB	Time
#1 $(m_x=10,n_x=20)$	-514.68	-394.75	0.05	-505.16	-394.75	0.15	-394.83	-394.75	1.05	-702.05	-394.75	0.17	-394.75	6.78	-394.75	-394.75	0.11	-394.75	-394.75	0.12
#2 $(m_x=10,n_x=20)$	-965.52	-884.75	0.05	-963.87	-884.75	0.15	-884.83	-884.75	1.06	-1192.05	-884.75	0.15	-884.75	7.67	-884.75	-884.75	0.12	-884.75	-884.75	0.11
#3 $(m_x=15,n_x=10)$	-4694.10	-4674.68	0.03	-4688.44	-4674.68	0.06	-4674.68	-4674.68	0.12	-4753.10	-4674.68	0.15	-4674.68	8.18	-4674.83	-4674.68	0.03	-4674.92	-4674.68	0.06
#4 $(m_x=62,n_x=50)$	-175920.41	-175705.59	0.13	-175869.24	-175705.59	0.48	-175705.59	-175705.59	11.46	-177638.35	-175705.59	1.32	-175705.57	6.95	-175707.22	-175705.59	0.56	-175707.22	-175705.59	1.56
#5 $(m_x=130,n_x=100)$	-693146.47	-692613.05	0.32	-693068.49	-692613.05	4.81	-692613.05	-692613.05	900.48	-707519.84	-692613.05	8.27	-692613.05	7.77	-692633.48	-692613.05	12.03	-692633.48	-692613.05	43.64
#6 $(m_x=240,n_x=200)$	-6022194.41	-6020787.42	0.95	-6021978.92	-6020787.42	44.83	NA	-	3000*	-6112433.52	-6020787.42	59.60	-6020787.41	11.27	-6020887.35	-6020787.42	528.99	-6020887.35	-6020787.42	1165.38
#7 $(m_x=280,n_x=240)$	-1856557.05	-1855739.98	1.36	-1856443.10	-1855733.06	104.90	NA	-	3000*	-1961723.39	-1855733.06	80.90	-1855739.98	13.06	-1900962.70	-1649853.80	3000*	-2162137.04	-1855739.98	3000*

Notes. The first column in this table presents the problem numbers and their corresponding dimensions. In each approach, the 'LB' column represents the lower bound values, and the 'UB' column displays the upper bounds. 'NA' indicates the bound was not determined within the 3000-second time limit.

Table 8 Detailed result on concave quadratic minimization

Problem (Dimension)	ARO-QO Algorithm			Selvi et al. (2022)			Ben-Tal and Roos (2022)		Gurobi			CPLEX		
	LB	UB	Time	LB	UB	Time	UB	Time	LB	UB	Time	LB	UB	Time
# 1 ($m_x = 100, n_x = 50$)	-21226.05	-20906.11	0.19	-23918.61	-20906.11	2.19	-20906.11	16.24	-20907.98	-20906.11	46.51	-20908.66	-20906.11	20.31
# 2 ($m_x = 100, n_x = 50$)	-19999.47	-19801.12	0.22	-22281.13	-19801.12	2.21	-19801.12	7.59	-19801.12	-19801.12	142.38	-19803.73	-19801.12	16.31
# 3 ($m_x = 100, n_x = 50$)	-21924.71	-21516.59	0.20	-24468.84	-21514.54	1.93	-21516.59	8.26	-21518.39	-21516.59	36.52	-21520.18	-21516.59	16.50
# 4 ($m_x = 100, n_x = 50$)	-19796.92	-19504.56	0.20	-22192.55	-19504.56	2.01	-19504.56	7.20	-19505.00	-19504.46	37.70	-19508.39	-19504.56	14.43
# 5 ($m_x = 100, n_x = 50$)	-19938.54	-19588.64	0.20	-22004.72	-19617.58	2.65	-19617.58	8.04	-19619.28	-19617.58	69.52	-19620.01	-19617.58	17.07
# 6 ($m_x = 150, n_x = 100$)	-43503.68	-42817.73	0.49	-48917.86	-42817.73	8.97	-42817.73	10.31	-42821.91	-42817.73	398.47	-42824.07	-42817.73	421.30
# 7 ($m_x = 150, n_x = 100$)	-44023.17	-43288.39	0.50	-49478.74	-43295.39	9.09	-43295.39	10.04	-43298.91	-43295.39	1138.38	-43304.86	-43295.39	523.09
# 8 ($m_x = 150, n_x = 100$)	-44509.83	-43708.24	0.51	-50215.65	-43746.44	9.61	-43749.35	11.08	-43753.45	-43749.45	832.73	-43756.27	-43749.36	578.56
# 9 ($m_x = 150, n_x = 100$)	-40043.22	-39464.10	0.49	-44570.54	-39479.17	8.72	-39479.17	9.69	-39479.86	-39477.46	1195.30	-39489.34	-39479.17	228.30
# 10 ($m_x = 150, n_x = 100$)	-50419.80	-49987.08	0.49	-56409.90	-49987.08	8.59	-49987.08	9.11	-49991.33	-49987.08	315.52	-49992.08	-49987.08	161.97
# 11 ($m_x = 200, n_x = 100$)	-36365.81	-35824.95	0.71	-41378.67	-35824.95	18.51	-35824.95	11.76	-35828.39	-35824.95	484.97	-35829.88	-35824.95	366.40
# 12 ($m_x = 200, n_x = 100$)	-36273.69	-35835.66	0.72	-41252.42	-35828.10	18.44	-35835.66	12.46	-35839.10	-35835.66	726.90	-35841.34	-35835.67	501.42
# 13 ($m_x = 200, n_x = 100$)	-34758.56	-34055.57	0.69	-39847.86	-34055.40	19.59	-34055.76	13.19	-34057.88	-34055.76	1636.05	-34135.30	-34055.76	301.26
# 14 ($m_x = 200, n_x = 100$)	-35864.55	-35262.45	0.76	-41288.01	-35289.22	16.38	-35289.70	12.23	-35310.24	-35289.70	3000*	-35410.68	-35289.49	3000*
# 15 ($m_x = 200, n_x = 100$)	-39351.10	-38553.98	0.73	-45260.71	-38619.98	16.24	-38624.93	12.85	-38628.29	-38624.93	2165.88	-38631.30	-38624.93	2256.70
# 16 ($m_x = 250, n_x = 200$)	-114818.87	-113258.29	1.32	-129138.55	-113258.34	48.23	-113258.34	11.15	-113269.93	-113258.34	3000*	-113282.68	-113258.34	3000*
# 17 ($m_x = 250, n_x = 200$)	-92364.58	-90802.03	1.39	-103508.71	-90833.68	47.94	-90848.24	11.46	-109407.37	-90185.44	3000*	-91059.89	-90848.24	3000*
# 18 ($m_x = 250, n_x = 200$)	-102592.36	-101763.08	1.42	-115061.49	-101763.08	48.82	-101763.08	11.27	-102331.34	-101763.08	3000*	-101797.00	-101763.08	2638.61
# 19 ($m_x = 250, n_x = 200$)	-110848.56	-109342.56	1.42	-124919.22	-109342.56	52.02	-109342.56	11.29	-121382.85	-109342.56	3000*	-109354.94	-109342.56	3000*
# 20 ($m_x = 250, n_x = 200$)	-124939.05	-122854.09	1.61	-145278.15	-122854.09	59.25	-122854.09	11.39	-148108.59	-122854.09	3000*	-123861.93	-122854.09	3000*
# 21 ($m_x = 300, n_x = 200$)	-90379.80	-89076.58	2.78	-104968.70	-89054.41	83.60	-89076.13	13.17	-115882.61	-89076.79	3000*	-89728.32	-89076.86	3000*
# 22 ($m_x = 300, n_x = 200$)	-83366.49	-82259.34	2.63	-95965.30	-82256.29	97.87	-82259.34	12.77	-102793.52	-82259.34	3000*	-82661.52	-82259.34	3000*
# 23 ($m_x = 300, n_x = 200$)	-82027.18	-81016.91	2.49	-93726.28	-81008.71	138.95	-81016.91	13.12	-97987.79	-81016.91	3000*	-81178.61	-81016.91	3000*
# 24 ($m_x = 300, n_x = 200$)	-84667.10	-83459.67	2.51	-98686.53	-83457.97	102.20	-83459.67	12.93	-105898.98	-83459.67	3000*	-83953.38	-83459.67	3000*
# 25 ($m_x = 300, n_x = 200$)	-81894.50	-80966.99	2.59	-94430.77	-80966.99	82.63	-80966.99	12.81	-102639.56	-80966.99	3000*	-81059.07	-80966.99	3000*
# 26 ($m_x = 350, n_x = 300$)	-171137.64	-168437.60	2.98	-194844.52	-168438.92	170.15	-168444.69	16.46	-210038.17	-168194.52	3000*	-5605813.93	-168194.52	3000*
# 27 ($m_x = 350, n_x = 300$)	-164141.28	-161444.77	2.88	-186697.76	-161462.55	158.70	-161463.50	16.15	-205946.65	-161142.30	3000*	-5128394.30	-161463.49	3000*
# 28 ($m_x = 350, n_x = 300$)	-158477.21	-156089.67	2.92	-184476.06	-156064.84	146.55	-156089.67	16.55	-196949.54	-156089.67	3000*	-372519.82	-156089.67	3000*
# 29 ($m_x = 350, n_x = 300$)	-163020.17	-159968.26	2.75	-186943.98	-159857.19	149.84	-159567.38	16.22	-192162.18	-159559.82	3000*	-295635.77	-159985.69	3000*
# 30 ($m_x = 350, n_x = 300$)	-179027.10	-176570.47	2.92	-203873.39	-176564.79	144.81	-176570.47	16.15	-211848.25	-176570.47	3000*	-694837.69	-176570.47	3000*
# 31 ($m_x = 400, n_x = 300$)	-130525.34	-128324.26	5.55	-151936.50	-123832.77	262.92	-128322.77	18.56	-171188.37	-128323.25	3000*	-5164971.42	-128310.81	3000*
# 32 ($m_x = 400, n_x = 300$)	-122185.75	-120302.84	5.34	-142516.23	-120310.31	230.07	-120302.84	19.06	-165499.48	-120287.00	3000*	-5232838.82	-120308.58	3000*
# 33 ($m_x = 400, n_x = 300$)	-120417.62	-118873.22	5.21	-139741.15	-118868.50	220.00	-118873.22	19.16	-163195.52	-118573.02	3000*	-5201602.88	-118873.22	3000*
# 34 ($m_x = 400, n_x = 300$)	-121135.65	-118904.01	5.22	-140455.45	-118869.00	281.14	-118904.01	19.24	-166977.00	-118898.47	3000*	-5223379.15	-118898.58	3000*
# 35 ($m_x = 400, n_x = 300$)	-135783.77	-134180.30	5.64	-157376.81	-134180.24	287.99	-134180.30	19.28	-182483.43	-134135.73	3000*	-5207186.87	-134180.24	3000*
# 36 ($m_x = 450, n_x = 400$)	-239708.24	-236691.08	4.73	-267620.18	-236691.08	292.47	-236691.08	21.99	-308141.60	-236632.22	3000*	-1.02268×10^{9}	NA	3000*
# 37 ($m_x = 450, n_x = 400$)	-219201.30	-215007.32	4.96	-252589.43	-215005.17	308.21	-215005.17	22.54	-401192.82	-213758.53	3000*	-1.02521×10^{9}	NA	3000*
# 38 ($m_x = 450, n_x = 400$)	-231523.46	-227609.94	4.72	-265032.88	-227619.86	307.32	-227619.86	22.49	-322874.67	-169857.26	3000*	-1.02639×10^{9}	NA	3000*
# 39 ($m_x = 450, n_x = 400$)	-225364.42	-221535.08	4.65	-258200.84	-221537.39	291.19	-221562.86	22.20	-623239.89	-169708.85	3000*	-1.02626×10^{9}	NA	3000*
# 40 ($m_x = 450, n_x = 400$)	-248061.41	-243641.36	4.98	-286303.36	-243639.15	278.68	-243641.36	22.20	-353455.02	-161208.64	3000*	-1.02815×10^{9}	NA	3000*
# 41 ($m_x = 500, n_x = 400$)	-162882.47	-160300.05	9.57	-191340.70	-160278.14	559.34	-160307.27	25.89	-604066.17	-159909.39	3000*	-1.02268×10^{9}	NA	3000*
# 42 ($m_x = 500, n_x = 400$)	-172886.97	-169806.25	8.77	-202411.67	-169805.29	586.96	-169806.25	26.33	-267361.99	-169267.40	3000*	-1.02521×10^{9}	NA	3000*
# 43 ($m_x = 500, n_x = 400$)	-161267.75	-158235.56	9.03	-187056.39	-158223.56	555.60	-158232.58	25.86	-425931.05	-110899.09	3000*	-102664×10^{9}	NA	3000*
# 44 ($m_x = 500, n_x = 400$)	-173508.69	-170210.36	9.61	-203828.67	-170199.68	534.90	-170210.96	26.95	-350152.46	-169666.30	3000*	-1.02626×10^{9}	NA	3000*
# 45 ($m_x = 500, n_x = 400$)	-177296.71	-173532.58	9.46	-208668.91	-173433.65	546.23	-173461.45	27.46	-404418.89	-173127.83	3000*	-1.02815×10^{9}	NA	3000*
# 46 ($m_x = 550, n_x = 500$)	-254620.02	-250228.59	7.25	-292466.45	-250227.15	269.22	-250218.18	29.10	-386062.82	-165730.74	3000*	-2.00381×10^{9}	NA	3000*
# 47 ($m_x = 550, n_x = 500$)	-283033.14	-278009.50	7.13	-326626.69	-277934.32	263.27	-278009.50	29.14	-13261942.44	-111018.29	3000*	-1.99773×10^{9}	NA	3000*
# 48 ($m_x = 550, n_x = 500$)	-267476.07	-262965.87	7.35	-305398.64	-262965.87	270.19	-262965.87	29.83	-738574.46	-153507.88	3000*	-2.00129×10^{9}	NA	3000*
# 49 ($m_x = 550, n_x = 500$)	-268766.57	-264084.34	7.46	-309881.77	-264016.01	244.89	-264084.34	28.69	-449419.26	-197858.04	3000*	-2.00369×10^{9}	NA	3000*
# 50 ($m_x = 550, n_x = 500$)	-293060.47	-287527.81	7.03	-335443.85	-287498.36	236.11	-287547.38	30.33	-423238.78	-191541.98	3000*	-2.00595×10^{9}	NA	3000*
# 51 ($m_x = 600, n_x = 500$)	-217416.92	-213705.89	17.21	-256282.73	-213683.16	515.61	-213705.87	34.51	-1127074.64	-166459.89	3000*	-2.00381×10^{9}	NA	3000*
# 52 ($m_x = 600, n_x = 500$)	-230918.43	-227145.96	18.40	-270398.12	-227144.69	514.71	-227138.76	34.60	-525624.12	-151916.19	3000*	-1.99773×10^{9}	NA	3000*
# 53 ($m_x = 600, n_x = 500$)	-211637.44	-207961.86	17.79	-247423.31	-207918.89	484.56	-207961.86	37.11	-11832043.00	-111613.71	3000*	-2.00129×10^{9}	NA	3000*
# 54 ($m_x = 600, n_x = 500$)	-240876.42	-236788.53	17.57	-283777.93	-236783.30	517.46	-236788.86	34.50	-8878852.73	-111854.65	3000*	-2.00369×10^{9}	NA	3000*
# 55 ($m_x = 600, n_x = 500$)	-220691.08	-217845.10	17.35	-257822.62	-217845.30	490.03	-217845.10	33.87	-8732517.01	-108571.64	3000*	-2.00595×10^{9}	NA	3000*
# 56 ($m_x = 650, n_x = 600$)	-391433.86	-384919.90	13.98	-455449.56	-384880.07	483.69	-384919.90	39.44	-4971784.16	-134292.50	3000*	-3.45702×10^{9}	NA	3000*
# 57 ($m_x = 650, n_x = 600$)	-403904.96	-398018.32	12.81	-463747.58	-398018.32	445.72	-398018.32	36.42	-14155023.89	-146383.96	3000*	-3.45533×10^{9}	NA	3000*
# 58 ($m_x = 650, n_x = 600$)	-329953.17	-324832.60	12.92	-381469.39	-324788.61	416.11	-324832.60	38.78	-3241488.76	-127909.30	3000*	-3.45744×10^{9}	NA	3000*
# 59 ($m_x = 650, n_x = 600$)	-344749.19	-339970.74	12.99	-394538.13	-339907.86	406.65	-339937.88	37.55	-18755129.41	-134434.84	3000*	-3.46041×10^{9}	NA	3000*
# 60 ($m_x = 650, n_x = 600$)	-344458.62	-338774.49	12.72	-397930.12	-338767.55	370.98	-338801.08	36.88	-20611553.36	-150155.61	3000*	-3.46327×10^{9}	NA	3000*
# 61 ($m_x = 700, n_x = 600$)	-282346.92	-278274.80	24.64	-332396.86	-278196.53	910.18	-278119.92	44.49	-12550163.40	-146157.71	3000*	-3.45702×10^{9}	NA	3000*
# 62 ($m_x = 700, n_x = 600$)	-255125.93	-252049.75	23.19	-297435.47	-252019.04	734.42	-252019.04	44.34	-13717479.02	-135118.87	3000*	-3.45533×10^{9}	NA	3000*
# 63 ($m_x = 700, n_x = 600$)	-268578.12	-265180.15	23.93	-310071.57	-265179.26	751.89	-265180.15	46.80	-17742893.76	-136924.49	3000*	-3.45744×10^{9}	NA	3000*
# 64 ($m_x = 700, n_x = 600$)	-271077.81	-265854.53	24.04	-321255.53	-265784.91	847.24	-265854.53	47.55	-13671678.23	-137967.52	3000*	-3.46041×10^{9}	NA	3000*
# 65 ($m_x = 700, n_x = 600$)	-265786.65	-261964.40	23.37	-312133.44	-261960.96	863.81	-261989.02	43.56	-13511151.74	-134274.22	3000*	-3.46327×10^{9}	NA	3000*
# 66 ($m_x = 750, n_x = 700$)	-453938.25	-445745.52	16.26	-522337.64	-445739.49	500.72	-445745.52	48.47	-30556778.77	NA	3000*	-5.49163×10^{9}	NA	3000*
# 67 ($m_x = 750, n_x = 700$)	-392258.28	-385657.82	17.00	-449380.91	-385644.74	455.94	-385653.69	47.43	-29261626.30	-161772.44	3000*	-5.48510×10^{9}	NA	3000*
# 68 ($m_x = 750, n_x = 700$)	-446958.57	-441833.49	17.71	-512918.35	-441818.13	559.76	-441829.69	46.59	-29099763.10	NA	3000*	-5.48474×10^{9}	NA	3000*
# 69 ($m_x = 750, n_x = 700$)	-437897.81	-430075.17	16.99	-507811.80	-429904.39	501.20	-430076.75	46.58	-28450230.87	NA	3000*	-5.48958×10^{9}	NA	3000*
# 70 ($m_x = 750, n_x = 700$)	-396137.17	-388867.26	16.65	-461346.54	-388657.60	497.99	-388867.26	49.26	-28394163.71	NA	3000*	-5.50234×10^{9}	NA	3000*
# 71 ($m_x = 800, n_x = 700$)	-311760.73	-306985.03	31.70	-366078.98	-306837.21	938.07	-306985.03	57.35	-28336162.67	NA	3000*	-5.49163×10^{9}	NA	3000*
# 72 ($m_x = 800, n_x = 700$)	-331384.65	-326264.31	33.21	-392327.23	-326243.53	950.99	-326264.31	55.41	-29520962.09	NA	3000*	-5.48510×10^{9}	NA	3000*
# 73 ($m_x = 800, n_x = 700$)	-302347.42	-297341.90	32.34	-358956.95	-297338.12	939.47	-297347.39	55.90	-28577654.37	NA	3000*	-5.48474×10^{9}	NA	3000*
# 74 ($m_x = 800, n_x = 700$)	-317298.58	-312825.99	30.31	-371677.77	-312807.92	2086.76	-312825.99	57.14	-29767062.12	NA	3000*	-5.48958×10^{9}	NA	3000*
# 75 ($m_x = 800, n_x = 700$)	-325810.25	-322123.47	30.46	-376710.09	-322132.66	906.84	-322132.66	55.85	-28979308.79	NA	3000*	-5.50234×10^{9}	NA	3000*

Notes. In the table, we applied the ARO-QO Algorithm with a static decision rule in the lower bound approximation step. The 'NA' indicates that the solver could not find any feasible solution within the time limit, which was set at 3,000 seconds.

Table 9 Detailed result on standard quadratic optimization

	ARO-QO			MILO			MILO-L-U			ARO-QO-MILO			Gurobi			CPLEX			CPLEX local	
Problem	LB	UB	Time	LB	UB	Time	LB	UB	Time	LB	UB	Time	LB	UB	Time	LB	UB	Time	UB	Time
#1 ($n_x = 500, d = 0.50$)	-9.8882	-6.4012	6.72	-6.9657	-6.9657	1961.92	-6.9657	-6.9657	916.67	-6.9657	-6.9657	983.48	-191.4577	-6.3477	3000*	-510133.3318	1.2865	3000*	-6.3715	4.53
#2 ($n_x = 500, d = 0.50$)	-9.9270	-6.6279	7.80	-7.0841	-7.0841	2000.15	-7.0841	-7.0841	730.09	-7.0841	-7.0841	768.65	-9.0589	-6.6493	3000*	-506289.0935	0.3183	3000*	-5.9169	5.92
#3 ($n_x = 500, d = 0.50$)	-9.9067	-6.7358	6.53	-6.8667	-6.8667	2014.08	-6.8667	-6.8667	951.72	-6.8667	-6.8667	998.48	-8.8296	-6.6624	3000*	-510373.7349	1.2783	3000*	-6.1619	7.71
#4 ($n_x = 500, d = 0.50$)	-9.8942	-6.6229	8.35	-6.8399	-6.8399	1839.07	-6.8399	-6.8399	893.02	-6.8399	-6.8399	960.37	-275.1602	-6.4848	3000*	-494369.5259	1.6020	3000*	-6.2371	6.77
#5 ($n_x = 500, d = 0.50$)	-9.9156	-6.5620	7.48	-7.1282	-7.1282	1833.92	-7.1282	-7.1282	642.74	-7.1282	-7.1282	725.01	-180.8172	-6.3804	3000*	-506389.7396	1.2845	3000*	-6.0268	6.16
#6 ($n_x = 500, d = 0.75$)	-9.9257	-6.8795	7.49	-7.6294	-7.3103	3000*	-7.3103	-7.3103	2020.58	-7.3110	-7.3103	2175.48	-8.9277	-7.2121	3000*	-762239.6625	1.2865	3000*	-6.7017	5.45
#7 ($n_x = 500, d = 0.75$)	-9.9270	-6.8710	6.97	-7.5763	-7.2660	3000*	-7.2660	-7.2660	2443.64	-7.2660	-7.2660	2550.61	-9.0610	-7.0364	3000*	-762805.0408	0.3183	3000*	-6.9959	5.40
#8 ($n_x = 500, d = 0.75$)	-9.9067	-7.2615	5.29	-7.2736	-7.2736	2772.31	-7.2736	-7.2736	1994.84	-7.2736	-7.2736	1952.91	-8.8130	-7.2737	3000*	-752040.9615	1.2783	3000*	-7.2736	5.53
#9 ($n_x = 500, d = 0.75$)	-9.8942	-6.8162	6.72	-7.2698	-7.2698	2879.39	-7.2698	-7.2698	1751.84	-7.2698	-7.2698	1930.82	-9.0686	-7.0285	3000*	-743920.3594	1.6020	3000*	-7.2698	4.65
#10 ($n_x = 500, d = 0.75$)	-9.9229	-6.8076	5.04	-7.5511	-7.2511	3000*	-7.2511	-7.2511	2412.83	-7.2511	-7.2511	2346.78	-9.0567	-6.8525	3000*	-761558.1498	1.2845	3000*	-7.1457	3.69
#11 ($n_x = 500, d = 0.90$)	-9.9257	-7.1770	5.50	-8.0331	-7.3852	3000*	-7.9052	-7.3852	3000*	-7.9044	-7.3852	3000*	-8.9426	-7.2882	3000*	-912849.6789	1.2865	3000*	-7.1905	4.90
#12 ($n_x = 500, d = 0.90$)	-9.9270	-7.1588	7.28	-7.7535	-7.5623	3000*	-7.5623	-7.5623	2800.91	-7.5623	-7.5623	2362.07	-8.9811	-7.2046	3000*	-915735.5099	0.3183	3000*	-7.3436	4.28
#13 ($n_x = 500, d = 0.90$)	-9.9067	-7.2765	4.96	-7.9589	-7.2875	3000*	-7.8110	-7.2875	3000*	-7.8290	-7.2875	3000*	-9.0566	-7.0170	3000*	-897979.1777	1.2783	3000*	-7.2372	6.39
#14 ($n_x = 500, d = 0.90$)	-9.8942	-7.1644	6.20	-7.9589	-7.3137	3000*	-7.6804	-7.3137	3000*	-7.7055	-7.3137	3000*	-8.8940	-7.2341	3000*	-891704.7924	1.6020	3000*	-7.0148	5.22
#15 ($n_x = 500, d = 0.90$)	-9.9439	-7.1231	5.41	-7.9481	-7.4094	3000*	-7.7801	-7.4094	3000*	-7.7693	-7.4094	3000*	-9.0567	-6.8725	3000*	-910365.1227	1.2845	3000*	-7.1457	3.78
#16 ($n_x = 600, d = 0.50$)	-9.9464	-6.8819	8.62	-19.2924	-7.1110	3000*	-7.1110	-7.1110	2045.93	-7.1110	-7.1110	1997.78	-284.0831	-6.7010	3000*	-734938.2000	0.8135	3000*	-6.6918	7.42
#17 ($n_x = 600, d = 0.50$)	-9.9207	-7.0434	9.65	-19.1496	-7.0434	3000*	-7.0434	-7.0434	2018.48	-7.0434	-7.0434	2458.40	-9.2987	-6.6313	3000*	-731040.2752	1.3523	3000*	-6.9571	12.26
#18 ($n_x = 600, d = 0.50$)	-9.9393	-6.8264	16.12	-19.2418	-6.9538	3000*	-6.9972	-6.9972	2042.46	-6.9972	-6.9972	2190.21	-167.2108	-6.3736	3000*	-725959.4255	1.2299	3000*	-6.3754	13.52
#19 ($n_x = 600, d = 0.50$)	-9.9371	-6.6957	12.32	-19.7394	-6.9021	3000*	-6.9442	-6.9442	2412.77	-6.9442	-6.9442	2380.36	-273.4919	-6.3655	3000*	-723597.9867	0.3822	3000*	-6.7507	11.07
#20 ($n_x = 600, d = 0.50$)	-9.9282	-6.1743	18.91	-19.4424	-6.9955	3000*	-6.9955	-6.9955	2282.01	-6.9955	-6.9955	2179.94	-164.3706	-6.2329	3000*	-729596.0179	1.9107	3000*	-6.4092	8.55
#21 ($n_x = 600, d = 0.75$)	-9.9464	-6.9293	10.30	-8.5760	-7.2844	3000*	-8.1162	-7.3282	3000*	-8.0788	-7.3899	3000*	-9.2723	-6.9656	3000*	-1099994.2014	0.8135	3000*	-7.1644	9.32
#22 ($n_x = 600, d = 0.75$)	-9.9207	-7.2915	7.91	-8.5603	-7.2867	3000*	-8.1567	-7.2867	3000*	-8.8122	-7.2915	3000*	-9.5096	-7.0131	3000*	-1100460.7270	1.3523	3000*	-7.0273	10.57
#23 ($n_x = 600, d = 0.75$)	-9.9393	-6.9892	11.43	-21.7273	-7.2478	3000*	-8.0810	-7.2890	3000*	-8.0940	-7.2890	3000*	-402.6448	-7.2394	3000*	-1090334.6306	1.2299	3000*	-6.8130	8.58
#24 ($n_x = 600, d = 0.75$)	-9.9371	-7.2231	7.16	-8.7843	-7.3271	3000*	-8.0614	-7.3271	3000*	-8.0896	-7.3271	3000*	-9.2341	-6.9121	3000*	-1088254.3172	0.3822	3000*	-7.1402	13.32
#25 ($n_x = 600, d = 0.75$)	-9.9282	-6.9840	7.16	-8.6369	-7.1835	3000*	-8.1193	-7.2321	3000*	-8.1062	-7.2321	3000*	-294.3743	-6.8262	3000*	-1094719.9823	1.9107	3000*	-7.0937	14.73
#26 ($n_x = 600, d = 0.90$)	-9.9643	-6.9793	16.93	-8.5489	-7.4746	3000*	-8.3991	-7.4746	3000*	-8.3959	-7.4746	3000*	-8.9978	-7.4085	3000*	-1317637.6549	0.8135	3000*	-7.3390	13.17
#27 ($n_x = 600, d = 0.90$)	-9.9207	-7.1750	8.48	-8.8202	-7.5206	3000*	-8.4155	-7.4149	3000*	-8.3602	-7.5206	3000*	-9.2295	-7.0848	3000*	-1320873.1874	1.3523	3000*	-7.1866	10.27
#28 ($n_x = 600, d = 0.90$)	-9.9393	-6.9889	9.07	-8.5916	-7.5124	3000*	-8.3136	-7.5124	3000*	-8.3078	-7.5124	3000*	-9.0171	-7.1633	3000*	-1305440.4038	1.2299	3000*	-7.1971	8.84
#29 ($n_x = 600, d = 0.90$)	-9.9371	-7.1185	7.36	-8.6395	-7.4096	3000*	-8.3242	-7.4226	3000*	-8.3436	-7.4103	3000*	-9.3006	-6.8496	3000*	-1304554.1044	0.3822	3000*	-7.0915	8.01
#30 ($n_x = 600, d = 0.90$)	-9.9282	-7.2576	11.77	-8.6059	-7.3595	3000*	-8.3660	-7.3330	3000*	-8.3475	-7.3595	3000*	-9.0805	-7.2268	3000*	-1310172.9769	1.9107	3000*	-7.1630	8.35
#31 ($n_x = 700, d = 0.50$)	-9.9535	-6.8554	11.62	-19.9975	-7.0875	3000*	-7.8082	-7.0875	3000*	-7.8213	-7.0976	3000*	-581.7589	-7.0743	3000*	-992790.3462	1.9870	3000*	-6.9621	15.71
#32 ($n_x = 700, d = 0.50$)	-9.9423	-6.8287	18.77	-19.9495	-7.0978	3000*	-7.9777	-7.0978	3000*	-7.9033	-7.0978	3000*	-687.3291	-6.5275	3000*	-991913.5841	0.8314	3000*	-6.5229	15.23
#33 ($n_x = 700, d = 0.50$)	-9.8993	-6.9250	19.33	-19.9400	-7.0874	3000*	-7.5893	-7.2341	3000*	-7.7323	-7.2341	3000*	-688.0122	-7.2341	3000*	-989990.5545	0.0685	3000*	-6.7706	12.81
#34 ($n_x = 700, d = 0.50$)	-9.9196	-6.6119	13.27	-19.9242	-7.1775	3000*	-7.6468	-7.1775	3000*	-7.7715	-7.1775	3000*	-582.0707	-6.4569	3000*	-990737.2155	0.4787	3000*	-6.5411	13.44
#35 ($n_x = 700, d = 0.50$)	-9.9604	-6.4637	12.01	-20.0099	-6.9343	3000*	-7.8200	-7.0960	3000*	-7.8922	-7.0960	3000*	-699.3185	-6.7537	3000*	-995054.2807	1.8646	3000*	-6.5121	13.31
#36 ($n_x = 700, d = 0.75$)	-9.9535	-6.9743	9.38	-22.3853	-7.3986	3000*	-8.5175	-7.3684	3000*	-8.4993	-7.3986	3000*	-1449.6412	-7.4271	3000*	-1485228.3068	1.9870	3000*	-7.2063	13.51
#37 ($n_x = 700, d = 0.75$)	-9.9423	-7.0034	11.18	-22.3412	-7.2658	3000*	-8.5911	-7.2092	3000*	-8.5616	-7.2884	3000*	-1270.8686	-7.2887	3000*	-1491678.2390	0.8314	3000*	-6.8830	12.05
#38 ($n_x = 700, d = 0.75$)	-9.8993	-7.1934	9.42	-22.2591	-7.3929	3000*	-8.5297	-7.3139	3000*	-8.6085	-7.2557	3000*	-889.5585	-7.1530	3000*	-1484429.9024	0.0685	3000*	-7.0102	14.02
#39 ($n_x = 700, d = 0.75$)	-9.9273	-6.9695	11.26	-22.2867	-7.1705	3000*	-8.5786	-7.3523	3000*	-8.5860	-7.3523	3000*	-1454.5193	-7.1306	3000*	-1488303.0870	0.4787	3000*	-7.0094	12.81
#40 ($n_x = 700, d = 0.75$)	-9.9604	-7.0969	11.79	-22.4141	-7.2477	3000*	-8.5301	-7.5046	3000*	-8.5657	-7.2423	3000*	-703.8169	-7.2129	3000*	-1491721.6015	1.8546	3000*	-6.9618	13.86
#41 ($n_x = 700, d = 0.90$)	-9.9535	-7.2967	8.86	-22.9162	-7.3986	3000*	-8.6791	-7.4785	3000*	-8.6857	-7.4024	3000*	-1199.5401	-6.8760	3000*	-1780157.9382	1.9870	3000*	-7.3376	13.51
#42 ($n_x = 700, d = 0.90$)	-9.9423	-7.3353	8.60	-23.2553	-7.3661	3000*	-8.7111	-7.3736	3000*	-9.0656	-7.3661	3000*	-1208.4181	-7.0783	3000*	-1792548.9534	0.8314	3000*	-7.3221	14.76
#43 ($n_x = 700, d = 0.90$)	-9.8993	-7.2548	10.20	-22.8163	-7.2265	3000*	-8.6946	-7.4066	3000*	-8.6839	-7.3853	3000*	-1044.4092	-7.1261	3000*	-1778302.0147	0.0685	3000*	-7.0803	12.82
#44 ($n_x = 700, d = 0.90$)	-9.9273	-7.1481	16.16	-23.0565	-7.3386	3000*	-8.6788	-7.4870	3000*	-8.7011	-7.4870	3000*	-1068.3227	-6.8956	3000*	-1784629.0506	0.4787	3000*	-7.2375	13.15
#45 ($n_x = 700, d = 0.90$)	-9.9604	-7.2503	8.95	-23.4102	-7.5046	3000*	-8.6531	-7.5046	3000*	-8.6618	-7.5046	3000*	-1053.8475	-7.0932	3000*	-1785773.3406	1.8546	3000*	-7.3479	14.86
#46 ($n_x = 800, d = 0.50$)	-9.9253	-6.9309	18.34	-20.2061	-7.0692	3000*	-8.4827	-7.1461	3000*	-8.5821	-7.1268	3000*	-1204.9105	-7.0090	3000*	-1304988.0970	1.5730	3000*	-6.6751	20.49
#47 ($n_x = 800, d = 0.50$)	-9.9557	-6.8142	30.94	-20.2037	-7.1738	3000*	-8.4906	-7.1817	3000*	-8.4552	-7.1598	3000*	-1107.3697	-6.4334	3000*	-1293347.9485	1.4440	3000*	-6.5371	20.67
#48 ($n_x = 800, d = 0.50$)	-9.9517	-7.0402	39.40	-20.2053	-7.0690	3000*	-8.5381	-7.0690	3000*	-8.6646	-7.0690	3000*	-1107.3036	-6.5765	3000*	-1295449.7290	1.9782	3000*	-6.4990	24.58
#49 ($n_x = 800, d = 0.50$)	-9.9648	-6.6772	15.65	-20.1933	-7.0085	3000*	-8.4541	-7.0270	3000*	-8.4978	-7.0085	3000*	-1198.3929	-6.4603	3000*	-1293039.4367	0.1937	3000*	-6.4603	21.96
#50 ($n_x = 800, d = 0.50$)	-9.9623	-6.7852	20.92	-20.2523	-7.0628	3000*	-8.6028	-6.9578	3000*	-8.5880	-7.0628	3000*	-1116.2836	-6.6515	3000*	-1305956.9435	0.7937	3000*	-6.2876	31.34
#51 ($n_x = 800, d = 0.75$)	-9.9272	-7.1289	11.23	-22.5872	-7.3001	3000*	-8.8705	-7.3051	3000*	-8.8762	-7.2833	3000*	-1360.1920	-6.8978	3000*	-1955093.6379	1.5730	3000*	-7.0998	31.40
#52 ($n_x = 800, d = 0.75$)	-9.9719	-7.2198	10.60	-22.5954	-7.2698	3000*	-8.8500	-7.2523	3000*	-8.8483	-7.3588	3000*	-1823.0260	-6.8874	3000*	-1940785.8404	1.4440	3000*	-6.6591	18.60
#53 ($n_x = 800, d = 0.75$)	-9.9517	-7.3767	10.34	-22.6151	-7.2549	3000*	-8.8208	7.3485	3000*	-8.8234	-7.3767	3000*	-1520.4457	-6.9469	3000*	-1943358.3883	1.9782	3000*	-6.8387	21.03
#54 ($n_x = 800, d = 0.75$)	-9.9648	-6.9863	10.81	-22.6194	-7.3558	3000*	-8.8657	-7.1858	3000*	-8.8218	-7.3558	3000*	-1976.3724	-6.9167	3000*	-1944031.3414	0.1937	3000*	-7.0193	21.65
#55 ($n_x = 800, d = 0.75$)	-9.9682	-7.0128	19.58	-22.6747	-7.2447	3000*	-8.8357	-7.2514	3000*	-8.8740	-7.3387	3000*	-1827.0103	-7.3417	3000*	-1956718.3229	0.7937	3000*	-7.3223	20.66
#56 ($n_x = 800, d = 0.90$)	-9.9309	-7.4501	11.61	-23.7523	-7.3560	3000*	-8.9372	-7.4393	3000*	-9.0032	-7.4501	3000*	-2189.8735	-7.2454	3000*	-2344643.1172	1.5730	3000*	-7.2556	21.80
#57 ($n_x = 800, d = 0.90$)	-9.9719	-7.4105	10.46	-23.7465	-7.3273	3000*	-8.9223	-7.4194	3000*	-9.9719	-7.5203	3000*	-2186.6381	-7.0898	3000*	-2328728.9883	1.4440	3000*	-7.5203	40.13
#58 ($n_x = 800, d = 0.90$)	-9.9517	-7.3767	10.21	-23.7194	-7.4337	3000*	-8.9099	-7.4767	3000*	-8.9209	-7.4767	3000*	-1452.6958	-7.1811	3000*	-2327206.7644	1.9782	3000*	-7.3156	26.14
#59 ($n_x = 800, d = 0.90$)	-9.9648	-7.2703	14.17	-23.7265	-7.5342	3000*	-8.9252	-7.5342	3000*	-8.9435	-7.4877	3000*	-1626.0559	-7.2361	3000*	-2330158.9267	0.1937	3000*	-7.0993	20.68
#60 ($n_x = 800, d = 0.90$)	-9.9832	-7.3396	14.04	-23.7133	-7.3861	3000*	-8.9419	-7.4486	3000*	-8.9572	-7.4486	3000*	-2003.3481	-7.1269	3000*	-2343845.3291	0.7937	3000*	-7.2450	27.63
#61 ($n_x = 900, d = 0.50$)	-9.9404	-6.7907	19.76	-20.2443	-7.1815	3000*	-8.9124	-7.1702	3000*	-8.8664	-7.1815	3000*	-1628.8530	-6.7064	3000*	-1650849.1216	0.6707	3000*	-6.0438	38.05
#62 ($n_x = 900, d = 0.50$)	-9.9158	-6.3653	19.21	-20.2287	-7.0768	3000*	-8.8025	-7.0512	3000*	-8.8463	-7.0660	3000*	-1609.7388	-6.1243	3000*	-1634675.1466	1.0180	3000*	-6.5870	35.71
#63 ($n_x = 900, d = 0.50$)	-9.9407	-6.6766	20.07	-20.2575	-7.0959	3000*	-8.9101	-7.0984	3000*	-8.9041	-7.0984	3000*	-1513.6661	-6.6296	3000*	-1635890.4096	0.8904	3000*	-6.2114	31.76
#64 ($n_x = 900, d = 0.50$)	-9.9265	-6.5910	21.61	-20.2182	-7.1089	3000*	-8.8908	-7.1089	3000*	-8.7942	-7.1798	3000*	-1523.9258	-6.6394	3000*	-1639838.4390	0.6390	3000*	-6.1844	25.95
#65 ($n_x = 900, d = 0.50$)	-9.9226	-6.1556	20.89	-20.2961	-7.0485	3000*	-8.8906	-7.0485	3000*	-8.8697	-7.0501	3000*	-1530.5476	-6.7123	3000*	-1652826.0587	1.8410	3000*	-6.7211	40.11
#66 ($n_x = 900, d = 0.75$)	-9.9404	-7.1241	17.70	-22.6327	-7.1474	3000*	-9.0512	-7.4088	3000*	-9.0727	-7.4088	3000*	-2442.4223	-6.9131	3000*	-2473501.4954	0.6707	3000*	-7.0606	30.80
#67 ($n_x = 900, d = 0.75$)	-9.9630	-7.1454	12.50	-22.6840	-7.2707	3000*	-9.0554	-7.3517	3000*	-9.0352	-7.3517	3000*	-2259.1470	-7.2346	3000*	-2452282.8912	1.0180	3000*	-6.9308	29.39
#68 ($n_x = 900, d = 0.75$)	-9.9407	-7.1086	13.33	-22.6615	-7.2547	3000*	-9.0214	-7.2077	3000*	-9.0223	-7.2946	3000*	-2274.8173	-6.9916	3000*	-2456608.9126	0.8904	3000*	-6.9429	40.92
#69 ($n_x = 900, d = 0.75$)	-9.9265	-7.3870	13.70	-22.6445	-7.3589	3000*	-9.0490	-7.2958	3000*	-9.2288	-7.3870	3000*	-2434.8416	-7.3222	3000*	-2462770.4364	0.6390	3000*	-7.3222	30.10
#70 ($n_x = 900, d = 0.75$)	-9.9242	-7.0212	13.90	-22.6587	-7.2586	3000*	-9.0712	-7.3255	3000*	-9.0722	-7.3325	3000*	-2442.9979	-7.1197	3000*	-2476961.2207	1.8410	3000*	-7.1197	36.84
#71 ($n_x = 900, d = 0.90$)	-9.9404	-7.3829	15.13	-23.9074	-7.3536	3000*	-9.1379	-7.4484	3000*	-9.1697	-7.4772	3000*	-2931.5765	-7.1555	3000*	-2966156.3480	0.6707	3000*	-7.1397	51.17
#72 ($n_x = 900, d = 0.90$)	-9.9630	-7.4819	14.75	-23.9652	-7.4185	3000*	-9.1211	-7.4819	3000*	-9.2142	-7.4951	3000*	-2896.0428	-7.2951	3000*	-2944100.5505	1.0180	3000*	-7.1328	28.59
#73 ($n_x = 900, d = 0.90$)	-9.9407	-7.4195	12.99	-23.9818	-7.3573	3000*	-9.1050	-7.4827	3000*	-9.1269	-7.4887	3000*	-2910.6695	-7.2515	3000*	-2942237.0584	0.8904	3000*	-6.9536	29.41
#74 ($n_x = 900, d = 0.90$)	-9.9265	-7.4590	12.76	-23.9073	-7.3427	3000*	-9.1412	-7.3870	3000*	-9.1927	-7.5135	3000*	-2920.5837	-7.3785	3000*	-2952666.4505	0.6390	3000*	-7.3444	34.63
#75 ($n_x = 900, d = 0.90$)	-9.9242	-7.2383	13.90	-23.9091	-7.3492	3000*	-9.1392	-7.6379	3000*	-9.1613	-7.4127	3000*	-2928.5207	-7.3745	3000*	-2967545.7925	1.8410	3000*	-7.3745	31.45

Table 10 Detailed result on general quadratic optimization

Problem (Dimensions, convexity rank)	static ARO-QO R2			static ARO-QO R3			linear ARO-QO			affine ARO-QO			MILO			Gurobi			CPLEX			CPLEX local	
	LB	UB	Time	LB	UB	Time	LB	UB	Time	LB	UB	Time	LB	UB	Time	LB	UB	Time	LB	UB	Time	UB	Time
# 1 ($n_x = 20, m_x = 4, p = 0.10$)	-502.2654	-209.7567	0.06	-246.4913	-209.7567	0.06	-236.8930	-209.7567	0.06	-209.7567	-209.7567	0.08	-209.7567	-209.7567	0.41	-209.7567	-209.7567	0.24	-209.7567	-209.7567	0.22	-133.1002	0.16
# 2 ($n_x = 20, m_x = 4, p = 0.10$)	-485.1619	-209.7015	0.06	-249.5338	-209.7015	0.06	-247.7615	-208.7302	0.07	-209.7015	-209.7015	0.07	-209.7015	-209.7015	0.42	-209.7015	-209.7015	0.09	-209.7015	-209.7015	0.11	-149.6372	0.05
# 3 ($n_x = 20, m_x = 4, p = 0.10$)	-372.6161	-116.9514	0.06	-149.6097	-116.9514	0.07	-147.5531	-106.5136	0.06	-117.0847	-116.9514	0.07	-116.9514	-116.9514	0.38	-116.9514	-116.9514	0.07	-116.9514	-116.9514	0.13	-88.2729	0.03
# 4 ($n_x = 20, m_x = 4, p = 0.10$)	-327.4302	-164.5331	0.06	-193.5341	-164.5331	0.06	-191.4127	-164.5331	0.06	-164.5331	-164.5331	0.08	-164.5331	-164.5331	0.41	-164.5331	-164.5331	0.12	-164.5331	-164.5331	0.08	-136.5830	0.03
# 5 ($n_x = 20, m_x = 4, p = 0.10$)	-518.6406	-259.7843	0.06	-293.9354	-259.7843	0.06	-282.2194	-259.7843	0.06	-259.7843	-259.7843	0.07	-259.7843	-259.7843	0.32	-259.7843	-259.7843	0.05	-259.7843	-259.7843	0.09	-259.7843	0.04
# 6 ($n_x = 20, m_x = 4, p = 0.50$)	-346.6186	-85.0160	0.06	-120.0366	-85.0160	0.06	-133.4730	-72.2828	0.11	-87.3275	-72.2828	0.13	-85.0160	-85.0160	0.35	-85.0160	-85.0160	0.07	-85.0160	-85.0160	0.20	-72.2828	0.03
# 7 ($n_x = 20, m_x = 4, p = 0.50$)	-374.0329	-92.5225	0.06	-147.5837	-92.5225	0.06	-143.2189	-103.2390	0.06	-105.6209	-103.2390	0.08	-103.2390	-103.2390	0.38	-103.2390	-103.2390	0.07	-103.2390	-103.2390	0.16	-103.2390	0.03
# 8 ($n_x = 20, m_x = 4, p = 0.50$)	-267.2395	-60.5018	0.17	-95.5146	-60.5018	0.10	-125.3089	-59.8427	0.11	-80.6305	-55.9289	0.12	-60.5018	-60.5018	0.37	-60.5038	-60.5018	0.41	-60.5018	-60.5018	1.00	-55.9289	0.03
# 9 ($n_x = 20, m_x = 4, p = 0.50$)	-316.2491	-125.4634	0.06	-145.1906	-125.4634	0.06	-154.7701	-125.4634	0.06	-125.4634	-125.4634	0.07	-125.4634	-125.4634	0.43	-125.4634	-125.4634	0.11	-125.4634	-125.4634	0.08	-81.8444	0.03
# 10 ($n_x = 20, m_x = 4, p = 0.50$)	-308.4996	-78.0629	0.07	-129.7636	-78.0629	0.06	-153.4202	-78.0629	0.06	-115.4057	-78.0629	0.08	-78.0629	-78.0629	0.47	-78.0629	-78.0629	0.27	-78.0629	-78.0629	0.31	-78.0629	0.03
# 11 ($n_x = 20, m_x = 4, p = 0.90$)	-151.4653	-13.0050	0.14	-19.9855	-13.0050	0.07	-89.3734	-13.0050	0.17	-57.1525	-13.0050	0.14	-13.0050	-13.0050	0.56	-13.0060	-13.0050	4.39	-13.0050	-13.0050	0.17	-13.0050	0.03
# 12 ($n_x = 20, m_x = 4, p = 0.90$)	-170.8914	-18.6757	0.17	-45.8072	-18.6757	0.09	-112.3453	-18.6757	0.10	-69.8442	-18.7477	0.10	-18.7477	-18.7477	0.58	-18.7495	-18.7477	6.87	-18.7477	-18.7477	0.33	-18.7477	0.03
# 13 ($n_x = 20, m_x = 4, p = 0.90$)	-159.7397	-29.5241	0.21	-52.5509	-29.5241	0.09	-99.3066	-29.5241	0.14	-57.0078	-29.5241	0.15	-29.5241	-29.5241	0.40	-29.5269	-29.5241	1.34	-29.5241	-29.5241	0.19	-29.5241	0.04
# 14 ($n_x = 20, m_x = 4, p = 0.90$)	-163.6100	-39.8986	0.11	-54.2365	-39.8986	0.08	-128.1053	-39.8986	0.11	-96.5843	-39.8986	0.11	-39.8986	-39.8986	0.38	-39.9026	-39.8986	1.62	-39.8986	-39.8986	0.14	-39.8986	0.03
# 15 ($n_x = 20, m_x = 4, p = 0.90$)	-142.7646	-8.7232	0.13	-14.5267	-8.7232	0.08	-109.8431	-8.7232	0.19	-98.1281	-8.7232	0.14	-8.7232	-8.7232	0.61	-8.7241	-8.7232	17.67	-8.7232	-8.7232	0.13	-8.7232	0.03
# 16 ($n_x = 50, m_x = 10, p = 0.10$)	-1836.3495	-486.8317	0.16	-736.4046	-486.8318	0.16	-696.7586	-486.8317	0.15	-486.8688	-486.8317	1.00	-486.8318	-486.8318	438.53	-486.8688	-486.8318	4.08	-486.8318	-486.8318	5.18	-413.2632	0.06
# 17 ($n_x = 50, m_x = 10, p = 0.10$)	-1799.3320	-497.5446	0.16	-767.5660	-497.5446	0.17	-733.6390	-497.5446	0.16	-533.2391	-497.5048	0.66	-499.5314	-499.5314	423.95	-499.5314	-499.5314	6.62	-499.5314	-499.5314	15.91	-362.7922	0.06
# 18 ($n_x = 50, m_x = 10, p = 0.10$)	-1464.9765	-420.5130	0.16	-653.7156	-413.2770	0.16	-644.8400	-416.0061	0.16	-458.6711	-411.9358	0.58	-425.8887	-425.8887	424.70	-425.9004	-425.8792	13.35	-425.8887	-425.8887	30.09	-392.1561	0.05
# 19 ($n_x = 50, m_x = 10, p = 0.10$)	-2051.5417	-657.5263	0.16	-917.6506	-678.7335	0.16	-886.3562	-678.7335	0.17	-678.7335	-678.7335	0.82	-678.7335	-678.7335	174.47	-678.7335	-678.7335	2.88	-678.7335	-678.7335	3.80	-430.6644	0.07
# 20 ($n_x = 50, m_x = 10, p = 0.10$)	-1639.8792	-507.4875	0.23	-773.5436	-507.4875	0.20	-726.5178	-507.4875	0.22	-512.7718	-507.4875	0.66	-507.4875	-507.4875	359.02	-507.4875	-507.4875	4.15	-507.4875	-507.4875	12.18	-380.5690	0.05
# 21 ($n_x = 50, m_x = 10, p = 0.50$)	-1281.8535	-211.5395	0.25	-403.1994	-215.6970	0.24	-468.3377	-215.6970	0.29	-278.4177	-215.6969	0.83	-221.7102	-221.7102	385.71	-221.7156	-221.7102	42.26	-221.7102	-221.7102	414.57	-221.7102	0.05
# 22 ($n_x = 50, m_x = 10, p = 0.50$)	-1248.1573	-175.5579	0.23	-455.7560	-232.3267	0.22	-500.0422	-232.3267	0.33	-306.5403	-201.4618	0.78	-232.3267	-232.3267	502.24	-232.3267	-232.3267	61.40	-232.3267	-232.3267	1804.33	-185.2477	0.05
# 23 ($n_x = 50, m_x = 10, p = 0.50$)	-1214.0754	-239.3386	0.20	-439.2754	-265.0968	0.18	-473.2546	-239.3386	0.19	-303.4787	-265.0968	0.64	-269.3328	-269.3328	304.59	-269.2300	-269.2116	28.81	-269.1975	-269.1975	40.80	-241.2824	0.05
# 24 ($n_x = 50, m_x = 10, p = 0.50$)	-1623.4898	-341.2661	0.34	-583.2424	-341.2661	0.18	-522.1243	-341.4167	0.19	-342.3302	-341.2661	0.69	-341.4167	-341.4167	209.12	-341.4178	-341.4167	4.14	-341.4167	-341.4167	9.65	-341.2661	0.05
# 25 ($n_x = 50, m_x = 10, p = 0.50$)	-1236.5919	-276.3289	0.24	-479.7365	-276.3289	0.17	-498.2328	-275.4528	0.32	-311.6615	-275.4528	0.69	-276.3289	-276.3289	571.18	-276.3289	-276.3289	41.13	-276.3289	-276.3289	71.11	-221.4508	0.04
# 26 ($n_x = 50, m_x = 10, p = 0.90$)	-787.9402	-46.9926	0.31	-135.8380	-46.9926	0.21	-352.7599	-46.9926	0.33	-215.7176	-46.9926	0.83	-78.2194	-30.7618	3001.13	-47.0322	-46.9926	3000.01	-133.8373	-46.9926	3000*	-46.9926	0.04
# 27 ($n_x = 50, m_x = 10, p = 0.90$)	-684.1419	-61.4566	0.29	-163.8091	-61.4566	0.22	-390.2469	-59.8840	0.25	-234.3891	-64.1607	0.80	-64.1607	-64.1607	2743.57	-64.1646	-64.1599	1445.78	-149.5522	-61.4566	3000*	-61.4566	0.04
# 28 ($n_x = 50, m_x = 10, p = 0.90$)	-701.2366	-142.7176	0.36	-198.9443	-142.7176	0.24	-380.6214	-142.7176	0.32	-238.1651	-142.7176	0.76	-142.7176	-142.7176	566.87	-142.7282	-142.7176	203.45	-151.8085	-142.7176	3000*	-142.7176	0.04
# 29 ($n_x = 50, m_x = 10, p = 0.90$)	-768.4358	-97.4185	0.29	-223.2464	-102.3150	0.19	-429.4734	-102.3150	0.35	-243.4236	-97.4185	0.82	-102.3150	-102.3150	883.43	-102.3181	-102.3150	277.30	-137.0214	-102.3150	3000*	-97.4185	0.04
# 30 ($n_x = 50, m_x = 10, p = 0.90$)	-790.1741	-166.4317	0.56	-282.4093	-166.4318	0.28	-421.6547	-166.4318	0.51	-261.2522	-166.4317	0.82	-166.4318	-166.4318	349.07	-166.4451	-166.4318	327.68	-167.7462	-166.4318	3000*	-166.4318	0.05
# 31 ($n_x = 100, m_x = 20, p = 0.10$)	-5411.5125	-1337.5543	0.41	-2139.8609	-1337.5543	0.46	-2053.4413	-1337.5543	0.43	-1349.1224	-1340.6499	998.22	-363142.6475	-579.8700	3000*	-1471.5252	-1340.6499	3000*	-1349.1224	-1078.6421	3000*	-696.3344	0.15
# 32 ($n_x = 100, m_x = 20, p = 0.10$)	-5304.6295	-1194.0588	0.43	-2176.3615	-1194.0588	0.49	-2081.9089	-1194.0588	0.46	-1341.2385	-1193.6543	27.64	-379640.8257	-690.4465	3000*	-1956.3072	-1013.8984	3000*	-1420.8368	-1132.6198	3000*	-1116.9634	0.34
# 33 ($n_x = 100, m_x = 20, p = 0.10$)	-4574.7349	-1000.7006	0.40	-1871.8744	-1000.7006	0.45	-1803.8469	-1000.7006	0.43	-1187.3002	-961.3245	28.76	-352935.4716	-559.2105	3000*	-1485.4819	-739.6329	3000*	-5702.6154	-807.2805	3000*	-724.8201	0.10
# 34 ($n_x = 100, m_x = 20, p = 0.10$)	-4727.2246	-1091.9423	0.40	-2020.5892	-1091.9423	0.45	-1930.0486	-1103.5279	0.43	-1331.2346	-953.6755	43.35	-386552.8451	-619.7630	3000*	-1744.6087	-854.5058	3000*	-1346.2577	-1043.6252	3000*	-1049.3367	0.34
# 35 ($n_x = 100, m_x = 20, p = 0.10$)	-4775.7327	-1033.9351	0.44	-1903.2068	-1094.9161	0.47	-1834.9146	-1094.9160	0.42	-1269.8908	-1094.9160	33.61	-406642.4624	-605.9432	3000*	-1762.1319	-1097.2681	3000*	-6689.7564	-1097.2681	3000*	-883.2602	0.16
# 36 ($n_x = 100, m_x = 20, p = 0.50$)	-4065.3259	-494.3490	0.74	-1268.0711	-494.3490	0.71	-1213.0641	-535.5468	0.74	-747.1404	-535.5468	397.33	-440196.6816	NA	3000*	-4928.6120	-513.0554	3000*	-13828.3489	-486.6684	3000*	-526.0419	0.11
# 37 ($n_x = 100, m_x = 20, p = 0.50$)	-3507.0491	-443.8409	0.92	-1182.7427	-434.3577	0.58	-1274.8630	-501.7429	0.64	-791.5316	-401.2651	28.32	-453772.6158	NA	3000*	-6349.4625	-421.4734	3000*	-20067.4319	-502.4029	3000*	-502.4029	0.14
# 38 ($n_x = 100, m_x = 20, p = 0.50$)	-3554.0026	-419.8498	0.63	-1122.7639	-464.9687	0.60	-1165.9439	-464.9687	0.59	-684.4929	-464.9687	27.07	-431371.6631	-380.5227	3000*	-5817.5665	-413.3563	3000*	-13774.2537	-407.3387	3000*	-427.9473	0.12
# 39 ($n_x = 100, m_x = 20, p = 0.50$)	-3561.3538	-579.8560	0.65	-1170.0446	-591.1865	0.71	-1297.6821	-591.6483	0.80	-778.1755	-579.8560	30.29	-431873.8340	NA	3000*	-4919.0847	-591.8320	3000*	-18416.9632	-582.1224	3000*	-582.1224	0.09
# 40 ($n_x = 100, m_x = 20, p = 0.50$)	-3362.1602	-522.8475	0.84	-1221.2673	-484.6747	0.68	-1314.5130	-535.8351	0.54	-811.9004	-501.0470	33.06	-460744.5237	NA	3000*	-7103.6419	-468.7373	3000*	-14963.6642	-460.9429	3000*	-460.9429	0.10
# 41 ($n_x = 100, m_x = 20, p = 0.90$)	-2292.0111	-165.6611	0.55	-383.0496	-165.6611	0.54	-948.3959	-165.6611	0.70	-544.8511	-165.6611	33.29	-289854.3498	NA	3000*	-2362.8867	-165.6611	3000*	-15785.5636	-165.6611	3000*	-165.6611	0.11
# 42 ($n_x = 100, m_x = 20, p = 0.90$)	-1911.1078	-101.4331	0.61	-311.2408	-101.4331	0.56	-934.7435	-101.4331	0.61	-538.3615	-101.4331	42.25	-294426.9809	NA	3000*	-785.6434	-101.4331	3000*	-16574.7090	-101.4331	3000*	-101.4331	0.11
# 43 ($n_x = 100, m_x = 20, p = 0.90$)	-2065.1321	-146.8353	0.56	-384.3917	-146.8353	0.63	-897.8807	-146.8353	0.58	-479.7671	-146.8353	30.26	-279911.3556	NA	3000*	-2675.2889	-146.8353	3000*	-15813.3065	-146.8353	3000*	-146.8353	0.10
# 44 ($n_x = 100, m_x = 20, p = 0.90$)	-2022.8094	-160.5975	0.80	-403.9767	-124.8196	0.77	-986.9888	-129.0902	0.69	-562.1642	-160.5975	27.06	-288697.1158	NA	3000*	-1496.9144	-160.5976	3000*	-15912.6669	-160.5976	3000*	-160.5976	0.06
# 45 ($n_x = 100, m_x = 20, p = 0.90$)	-1982.4773	-208.5480	1.31	-501.2663	-208.5481	1.15	-1033.4454	-208.5481	1.32	-620.0111	-208.5480	31.92	-351111.7441	NA	3000*	-3583.0449	-208.5481	3000*	-18082.7880	-208.5481	3000*	-208.5481	0.07
# 46 ($n_x = 200, m_x = 40, p = 0.10$)	-14172.2598	-2307.8918	1.62	-5232.5192	-2213.9900	2.19	-4951.1568	-2213.9900	2.14	-3311.7565	-1929.7471	3000*	-3797556.4250	NA	3000*	-85827.7832	-2080.0788	3000*	-23829.4334	-1918.5359	3000*	-1912.3629	0.84
# 47 ($n_x = 200, m_x = 40, p = 0.10$)	-14603.6859	-2306.5321	1.49	-5504.6901	-2306.5321	1.91	-5265.5428	-2306.5321	1.59	-3489.7806	-2240.8294	3000*	-4015372.6328	-510.3066	3000*	-85514.7184	-1650.7289	3000*	-25046.6875	-1980.2090	3000*	-1963.1169	1.22
# 48 ($n_x = 200, m_x = 40, p = 0.10$)	-12926.4627	-2098.3095	1.82	-5070.5783	-2009.6560	2.05	-4866.0407	-2098.3104	1.78	-3229.4880	-2098.3698	3000*	-3692832.6827	NA	3000*	-83617.8148	-1926.7162	3000*	-22967.3792	-1780.4717	3000*	-1947.1680	0.91
# 49 ($n_x = 200, m_x = 40, p = 0.10$)	-13829.5618	-2129.2767	1.54	-5287.6078	-2129.2768	1.76	-5036.7351	-2129.2767	1.52	-3387.8967	-2126.9469	3000*	-3912959.8028	NA	3000*	-91939.4061	-2058.2538	3000*	-24541.5220	-2054.5611	3000*	-1713.0363	1.55
# 50 ($n_x = 200, m_x = 40, p = 0.10$)	-14049.5491	-2220.7663	1.88	-5498.1688	-2221.9175	1.83	-5181.2309	-2212.6053	1.50	-3562.0374	-2192.2582	3000*	-4220693.1549	NA	3000*	-89886.7389	-1911.3396	3000*	-26286.6294	-2323.9666	3000*	-1978.0106	1.13
# 51 ($n_x = 200, m_x = 40, p = 0.50$)	-10326.4231	-1138.5124	1.84	-3230.3688	-1185.7654	2.18	-3194.8031	-1185.7654	2.13	-1968.8449	-1185.7654	3000*	-4530998.7825	NA	3000*	-118495.9328	-1137.1170	3000*	-65761.0090	-1137.117	3000*	-1136.0751	0.42
# 52 ($n_x = 200, m_x = 40, p = 0.50$)	-10200.0407	-954.0831	1.89	-3153.6780	-954.0832	2.56	-3283.2052	-918.0892	1.95	-2047.0008	-810.2982	3000*	-4796065.9215	NA	3000*	-128156.6178	-892.3848	3000*	-93204.9866	-993.3569	3000*	-1027.3875	0.57
# 53 ($n_x = 200, m_x = 40, p = 0.50$)	-10153.7995	-948.4658	1.82	-2955.7007	-954.0641	3.26	-3067.8925	-913.3022	2.05	-1894.3573	-932.1833	3000*	-4402349.7428	NA	3000*	-119544.7045	-901.2258	3000*	-77908.6617	-939.2971	3000*	-896.4181	0.49
# 54 ($n_x = 200, m_x = 40, p = 0.50$)	-10136.3166	-888.9150	1.95	-3136.0148	-1000.7620	2.73	-3299.1975	-1057.9986	3.39	-2040.7484	-1064.6908	3000*	-4668549.3168	NA	3000*	-122458.3506	-1193.4899	3000*	-94076.2750	-1017.6693	3000*	-1057.9986	0.66
# 55 ($n_x = 200, m_x = 40, p = 0.50$)	-9995.3057	-1037.8015	2.23	-3301.0946	-1037.8016	3.46	-3386.8654	-966.4569	1.82	-2110.8066	-1007.2145	3000*	-4794925.1890	NA	3000*	-123389.9880	-1029.0947	3000*	-84535.4339	-1046.6587	3000*	-1031.7948	0.73
# 56 ($n_x = 200, m_x = 40, p = 0.90$)	-6362.6179	-3120.6672	2.58	-1155.7491	-312.0674	3.26	-2492.6803	-232.6648	1.98	-1448.4865	-312.0673	3000*	-3329063.1027	NA	3000*	-74844.6915	-291.5432	3000*	-97089.0354	-291.5432	3000*	-291.5432	0.35
# 57 ($n_x = 200, m_x = 40, p = 0.90$)	-5808.3465	-257.6580	2.29	-1022.8180	-243.5871	6.38	-2446.7099	-257.6581	2.98	-1423.7264	-257.6581	3000*	-3328355.5249	NA	3000*	-74057.8393	-257.6581	3000*	-97719.9254	-257.6581	3000*	-257.6581	0.18
# 58 ($n_x = 200, m_x = 40, p = 0.90$)	-6359.3660	-338.3065	2.02	-1064.6585	-338.3065	2.56	-2402.9843	-338.3065	2.46	-1410.3324	-338.3064	3000*	-3199092.5491	NA	3000*	-75319.6508	-338.3065	3000*	-96211.0995	-338.3065	3000*	-338.3065	0.30
# 59 ($n_x = 200, m_x = 40, p = 0.90$)	-5823.8613	-246.3108	3.57	-886.7774	-246.3108	4.88	-2378.4661	-246.3108	2.34	-1403.6177	-246.3107	3000*	-3190587.9946	NA	3000*	-71964.0594	-246.3109	3000*	-92761.4664	-246.3109	3000*	-246.3109	0.41
# 60 ($n_x = 200, m_x = 40, p = 0.90$)	-5774.6444	-344.2266	2.34	-1142.5446	-344.2267	3.34	-2572.3896	-344.2267	2.10	-1554.0242	-344.2267	3000*	-3460715.7847	NA	3000*	-77171.9787	-344.2267	3000*	-104124.5446	-319.5897	3000*	-344.2267	0.37
# 61 ($n_x = 1000, m_x = 200, p = 0.10$)	-155715.6472	-10609.7458	238.20	-53614.7610	-10607.9336	220.75	-50531.8327	-10659.1266	269.51	-	-	3000*	-	-	3000*	-	-	3000*	-56592600000.0000	NA	3000*	-1024.2155	422.13
# 62 ($n_x = 1000, m_x = 200, p = 0.10$)	-160238.2209	-11585.6497	246.61	-56653.0814	-11662.1772	250.34	-53391.4479	-12164.3577	301.36	-	-	3000*	-	-	3000*	-	-	3000*	-6067290000.0000	NA	3000*	-10491.6510	414.95
# 63 ($n_x = 1000, m_x = 200, p = 0.10$)	-160654.4228	-11584.0843	222.50	-56629.4801	-11836.9278	231.57	-53666.2354	-11860.3793	249.77	-	-	3000*	-	-	3000*	-	-	3000*	-5793880000.0000	NA	3000*	-10759.0251	248.72
# 64 ($n_x = 1000, m_x = 200, p = 0.10$)	-160183.4599	-11398.0990	253.52	-56359.3681	-11294.5133	291.70	-53102.4374	-11340.8389	270.37	-	-	3000*	-	-	3000*	-	-	3000*	-6012650000.0000	NA	3000*	-10428.5112	426.48
# 65 ($n_x = 1000, m_x = 200, p = 0.10$)	-158939.8365	-11414.6882	219.26	-55617.4063	-11597.1014	249.12	-52454.0849	-11587.5294	224.51	-	-	3000*	-	-	3000*	-	-	3000*	-5934110000.0000	NA	3000*	-10092.0115	427.50
# 66 ($n_x = 1000, m_x = 200, p = 0.50$)	-116975.9950	-5317.9019	270.49	-33183.0764	-5269.4018	327.13	-31628.9882	-5442.7567	230.72	-	-	3000*	-	-	3000*	-	-	3000*	-7220530000.0000	NA	3000*	-5728.7229	193.93
# 67 ($n_x = 1000, m_x = 200, p = 0.50$)	-118668.7591	-6157.2543	276.11	-35825.4796	-5853.5543	270.59	-33286.7187	-5576.3871	328.89	-	-	3000*	-	-	3000*	-	-	3000*	-7706360000.0000	NA	3000*	-6025.8465	287.95
# 68 ($n_x = 1000, m_x = 200, p = 0.50$)	-117177.6061	-5566.4007	395.45	-34742.5932	-5573.4174	426.88	-32704.7745	-5340.9563	360.80	-	-	3000*	-	-	3000*	-	-	3000*	-7539740000.0000	NA	3000*	-5827.5965	140.19
# 69 ($n_x = 1000, m_x = 200, p = 0.50$)	-116456.5644	-5371.9010	274.04	-34496.3325	-5619.0413	511.34	-32794.3840	-5447.5039	412.59	-	-	3000*	-	-	3000*	-	-	3000*	-7701260000.0000	NA	3000*	-5190.5103	128.09
# 70 ($n_x = 1000, m_x = 200, p = 0.50$)	-116472.6824	-5311.4202	338.38	-34133.0236	-5896.4911	369.30	-32607.6084	-5540.4352	269.83	-	-	3000*	-	-	3000*	-	-	3000*	-7521460000.0000	NA	3000*	-5617.4839	184.27
# 71 ($n_x = 1000, m_x = 200, p = 0.90$)	-72472.0345	-1877.2144	367.37	-11205.6507	-1875.3918	379.63	-23884.8720	-1877.2164	272.24	-	-	3000*	-	-	3000*	-	-	3000*	-5294870000.0000	NA	3000*	-1875.3921	32.92
# 72 ($n_x = 1000, m_x = 200, p = 0.90$)	-68654.8710	-1624.5308	295.06	-11025.3635	-1624.5312	380.04	-24226.4690	-1594.0721	253.06	-	-	3000*	-	-	3000*	-	-	3000*	-5415910000.0000	NA	3000*	-1624.5315	38.11
# 73 ($n_x = 1000, m_x = 200, p = 0.90$)	-69472.2549	-1638.6911	276.98	-11225.3516	-1638.6914	363.61	-23910.8115	-1638.6911	314.87	-	-	3000*	-	-	3000*	-	-	3000*	-5385570000.0000	NA	3000*	-1638.6915	77.36
# 74 ($n_x = 1000, m_x = 200, p = 0.90$)	-69793.0139	-1887.7666	244.51	-11623.6630	-1887.7670	313.09	-24406.5590	-1887.7669	238.96	-	-	3000*	-	-	3000*	-	-	3000*	-5557750000.0000	NA	3000*	-1887.7670	58.88
# 75 ($n_x = 1000, m_x = 200, p = 0.90$)	-70060.2848	-1801.5314	282.02	-11198.0583	-1801.5318	359.99	-24039.0511	-1801.5320	319.89	-	-	3000*	-	-	3000*	-	-	3000*	-5395900000.0000	NA	3000*	-1801.5321	40.54

Note. The symbol "-" indicates that it not possible to determine bound within the maximum time limit. The 'NA' indicates that the solver could not find any feasible solution within the time limit,

References

Ahmadi, A. A. and Zhang, J. (2022). On the complexity of finding a local minimizer of a quadratic function over a polytope. *Mathematical Programming*, 195(1):783–792.

Andrianova, A., Korepanova, A., and Halilova, I. (2016). One algorithm for branch and bound method for solving concave optimization problem. In *IOP Conference Series: Materials Science and Engineering*, page 012005. IOP Publishing.

Anstreicher, K. M. (2009). Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. *Journal of Global Optimization*, 43:471–484.

Anstreicher, K. M. and Burer, S. (2005). DC versus copositive bounds for standard QP. *Journal of Global Optimization*, 33(2):299–312.

Ardestani-Jaafari, A. and Delage, E. (2021). Linearized robust counterparts of two-stage robust optimization problems with applications in operations management. *INFORMS Journal on Computing*, 33(3):1138–1161.

Audet, C., Hansen, P., and Savard, G. (2005). *Essays and Surveys in Global Optimization*, volume 7. Springer Science & Business Media, New York.

Bao, X., Sahinidis, N. V., and Tawarmalani, M. (2011). Semidefinite relaxations for quadratically constrained quadratic programming: A review and comparisons. *Mathematical Programming*, 129:129–157.

Ben-Tal, A., Goryashko, A., Guslitzer, E., and Nemirovski, A. (2004). Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376.

Ben-Tal, A. and Roos, E. (2022). An algorithm for maximizing a convex function based on its minimum. *INFORMS Journal on Computing*, 34(6):3200–3214.

Bentobache, M., Telli, M., and Mokhtari, A. (2022). New LP-based local and global algorithms for continuous and mixed-integer nonconvex quadratic programming. *Journal of Global Optimization*, 82(4):659–689.

Bertsimas, D. and Bidkhori, H. (2015). On the performance of affine policies for two-stage adaptive optimization: a geometric perspective. *Mathematical Programming*, 153(2):577–594.

Bertsimas, D. and Caramanis, C. (2010). Finite adaptability in multistage linear optimization. *IEEE Transactions on Automatic Control*, 55(12):2751–2766.

Bertsimas, D. and Dunning, I. (2016). Multistage robust mixed-integer optimization with adaptive partitions. *Operations Research*, 64(4):980–998.

Bertsimas, D., Goyal, V., and Lu, B. Y. (2015). A tight characterization of the performance of static solutions in two-stage adjustable robust linear optimization. *Mathematical Programming*, 150(2):281–319.

Bertsimas, D., Iancu, D. A., and Parrilo, P. A. (2010). Optimality of affine policies in multistage robust optimization. *Mathematics of Operations Research*, 35(2):363–394.

Bhanja, S., Karunaratne, D., Panchumarthy, R., Rajaram, S., and Sarkar, S. (2016). Non-boolean computing with nanomagnets for computer vision applications. *Nature Nanotechnology*, 11(2):177–183.

Bomze, I. and Gabl, M. (2021). Interplay of non-convex quadratically constrained problems with adjustable robust optimization. *Mathematical Methods of Operations Research*, 93(1):115–151.

Bomze, I. M. (2002a). Branch-and-bound approaches to standard quadratic optimization problems. *Journal of Global Optimization*, 22:17–37.

Bomze, I. M. (2002b). Regularity versus degeneracy in dynamics, games, and optimization: a unified approach to different aspects. *SIAM review*, 44(3):394–414.

Bomze, I. M. (2015). Copositive relaxation beats lagrangian dual bounds in quadratically and linearly constrained quadratic optimization problems. *SIAM Journal on Optimization*, 25(3):1249–1275.

Bomze, I. M. and De Klerk, E. (2002). Solving standard quadratic optimization problems via linear, semidefinite and copositive programming. *Journal of Global Optimization*, 24(2):163–185.

Bomze, I. M., Locatelli, M., and Tardella, F. (2008). New and old bounds for standard quadratic optimization: dominance, equivalence and incomparability. *Mathematical Programming*, 115(1):31–64.

Bomze, I. M., Schachinger, W., and Ullrich, R. (2018). The complexity of simple models—a study of worst and typical hard cases for the standard quadratic optimization problem. *Mathematics of Operations Research*, 43(2):651–674.

Bonami, P., Lodi, A., Schweiger, J., and Tramontani, A. (2019). Solving quadratic programming by cutting planes. *SIAM Journal on Optimization*, 29(2):1076–1105.

Boyd, S. P. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Bulo, S. R., Pelillo, M., and Bomze, I. M. (2011). Graph-based quadratic optimization: A fast evolutionary approach. *Computer Vision and Image Understanding*, 115(7):984–995.

Bundfuss, S. and Dur, M. (2009). An adaptive linear approximation algorithm for copositive programs. *SIAM Journal on Optimization*, 20(1):30–53.

Burer, S. (2009). On the copositive representation of binary and continuous nonconvex quadratic programs. *Mathematical Programming*, 120(2):479–495.

Burer, S. and Vandenbussche, D. (2008). A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations. *Mathematical Programming*, 113(2):259–282.

Burer, S. and Vandenbussche, D. (2009). Globally solving box-constrained nonconvex quadratic programs with semidefinite-based finite branch-and-bound. *Computational Optimization and Applications*, 43(2):181–195.

Cen, X. and Xia, Y. (2021). A new global optimization scheme for quadratic programs with low-rank nonconvexity. *INFORMS Journal on Computing*, 33(4):1368–1383.

Cevikalp, H. and Polikar, R. (2008). Local classifier weighting by quadratic programming. *IEEE Transactions on Neural Networks*, 19(10):1832–1838.

Chen, J. and Burer, S. (2012). Globally solving nonconvex quadratic programming problems via completely positive programming. *Mathematical Programming Computation*, 4(1):33–52.

Chinchuluun, A., Pardalos, P. M., and Enkhbat, R. (2005). Global minimization algorithms for concave quadratic programming problems. *Optimization*, 54(6):627–639.

Cuong, T. H., Lim, Y., and Yen, N. D. (2024). On a solution method in indefinite quadratic programming under linear constraints. *Optimization*, 73(4):1087–1112.

De Ruiter, F. J., Zhen, J., and Den Hertog, D. (2023). Dual approach for two-stage robust nonlinear optimization. *Operations Research*, 71(5):1794–1799.

Delage, E. and Iancu, D. A. (2015). Robust multistage decision making. In *The Operations Research Revolution*, chapter 2, pages 20–46. INFORMS.

Dorn, W. S. (1960). Duality in quadratic programming. *Quarterly of Applied Mathematics*, 18(2):155–162.

El Housni, O. and Goyal, V. (2021). On the optimality of affine policies for budgeted uncertainty sets. *Mathematics of Operations Research*, 46(2):674–711.

Fampa, M., Lee, J., and Melo, W. (2017). On global optimization with indefinite quadratics. *EURO Journal on Computational Optimization*, 5(3):309–337.

Gibbons, L. E., Hearn, D. W., Pardalos, P. M., and Ramana, M. V. (1997). Continuous characterizations of the maximum clique problem. *Mathematics of Operations Research*, 22(3):754–768.

Gökmen, Y. G. and Yıldırım, E. A. (2022). On standard quadratic programs with exact and inexact doubly nonnegative relaxations. *Mathematical Programming*, 193(1):365–403.

Gondzio, J. and Yıldırım, E. A. (2021). Global solutions of nonconvex standard quadratic programs via mixed integer linear programming reformulations. *Journal of Global Optimization*, 81(2):293–321.

Gorski, J., Pfeuffer, F., and Klamroth, K. (2007). Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research*, 66(3):373–407.

Gouveia, J., Pong, T. K., and Saee, M. (2020). Inner approximating the completely positive cone via the cone of scaled diagonally dominant matrices. *Journal of Global Optimization*, 76:383–405.

Grippo, L. and Sciandrone, M. (2000). On the convergence of the block nonlinear gauss–seidel method under convex constraints. *Operations Research Letters*, 26(3):127–136.

Gurobi Optimization (2023). LLC: Gurobi optimizer reference manual. Version 10.0.0.

Hadjiyiannis, M. J., Goulart, P. J., and Kuhn, D. (2011). A scenario approach for estimating the suboptimality of linear decision rules in two-stage robust optimization. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 7386–7391. IEEE.

Hansen, P. and Jaumard, B. (1992). Reduction of indefinite quadratic programs to bilinear programs. *Journal of Global Optimization*, 2:41–60.

Horst, R. and Thoai, N. V. (1999). DC programming: overview. *Journal of Optimization Theory and Applications*, 103:1–43.

Hu, J., Mitchell, J. E., and Pang, J.-S. (2012). An LPCC approach to nonconvex quadratic programs. *Mathematical Programming*, 133(1-2):243–277.

Ibaraki, T. and Katoh, N. (1988). *Resource allocation problems: algorithmic approaches*. MIT press, Cambridge.

IBM (2019). IBM ILOG CPLEX optimization studio User's Manual. Version 12.9.0 (1987-2019).

Khademi, A. (2024). *Nonlinear adjustable robust optimization*. PhD thesis, University of Tehran.

Khademi, A., Marandi, A., and Soleimani-damaneh, M. (2024). A new dual-based cutting plane algorithm for nonlinear adjustable robust optimization. *Journal of Global Optimization*, 89(3):559–595.

Khadivar, F., Chatzilygeroudis, K., and Billard, A. (2023). Self-correcting quadratic programming-based robot control. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Kim, S., Kojima, M., and Toh, K.-C. (2020). A geometrical analysis on convex conic reformulations of quadratic and polynomial optimization problems. *SIAM Journal on Optimization*, 30(2):1251–1273.

Konno, H. (1976). A cutting plane algorithm for solving bilinear programs. *Mathematical Programming*, 11(1):14–27.

Kozlov, M. K., Tarasov, S. P., and Khachiyan, L. G. (1980). The polynomial solvability of convex quadratic programming. *USSR Computational Mathematics and Mathematical Physics*, 20(5):223–228.

Lipp, T. and Boyd, S. (2016). Variations and extension of the convex–concave procedure. *Optimization and Engineering*, 17:263–287.

Liuzzi, G., Locatelli, M., and Piccialli, V. (2019). A new branch-and-bound algorithm for standard quadratic programming problems. *Optimization Methods and Software*, 34(1):79–97.

Löfberg, J. (2004). YALMIP: A toolbox for modeling and optimization in MATLAB. In *In Proceedings of the CACSD Conference*, Taipei, Taiwan.

Luo, H., Bai, X., Lim, G., and Peng, J. (2019). New global algorithms for quadratic programming with a few negative eigenvalues based on alternative direction method and convex relaxation. *Mathematical Programming Computation*, 11(1):119–171.

Markowitz, H. M. (1952). Protfilio selection. *The Journal of Finance*, 7(1):77–91.

Mitchell, J. E., Pang, J.-S., and Yu, B. (2014). Convex quadratic relaxations of nonconvex quadratically constrained quadratic programs. *Optimization Methods and Software*, 29(1):120–136.

MOSEK ApS (2023). The MOSEK optimization toolbox for MATLAB manual. Version 10.1.15.

O'Nan, M. (1971). *Linear Algebra*. Eagle mathematics series. Harcourt Brace Jovanovich, New York.

Pardalos, P. M. and Schnitger, G. (1988). Checking local optimality in constrained quadratic programming is NP-hard. *Operations Research Letters*, 7(1):33–35.

Pardalos, P. M. and Vavasis, S. A. (1991). Quadratic programming with one negative eigenvalue is NP-hard. *Journal of Global Optimization*, 1(1):15–22.

Phillips, A. T. and Rosen, J. B. (1988). A parallel algorithm for constrained concave quadratic global minimization. *Mathematical Programming*, 42:421–448.

Postek, K. and Hertog, D. d. (2016). Multistage adjustable robust mixed-integer optimization via iterative splitting of the uncertainty set. *INFORMS Journal on Computing*, 28(3):553–574.

Renegar, J. (2001). *A mathematical view of interior-point methods in convex optimization*. SIAM, Philadelphia.

Rostami, B., Errico, F., and Lodi, A. (2023). A convex reformulation and an outer approximation for a large class of binary quadratic programs. *Operations Research*, 71(2):471–486.

Scozzari, A. and Tardella, F. (2008). A clique algorithm for standard quadratic programming. *Discrete Applied Mathematics*, 156(13):2439–2448.

Selvi, A., Ben-Tal, A., Brekelmans, R., and den Hertog, D. (2022). Convex maximization via adjustable robust optimization. *INFORMS Journal on Computing*, 34(4):2091–2105.

Selvi, A., den Hertog, D., and Wiesemann, W. (2023). A reformulation-linearization technique for optimization over simplices. *Mathematical Programming*, 197(1):427–447.

Sherali, H. and Liberti, L. (2009). *Reformulation-linearization methods for global optimization. In Encyclopedia of Optimization*, pages 3263–3268. Springer, Boston, MA.

Sherali, H. D. and Alameddine, A. (1992). A new reformulation-linearization technique for bilinear programming problems. *Journal of Global Optimization*, 2:379–410.

Sherali, H. D. and Tuncbilek, C. H. (1995). A reformulation-convexification approach for solving nonconvex quadratic programming problems. *Journal of Global Optimization*, 7:1–31.

Thomä, S., Walther, G., and Schiffer, M. (2024). Designing tractable piecewise affine policies for multi-stage adjustable robust optimization. *Mathematical Programming*, pages 1–56.

Wang, A. L. and Kılınç-Karzan, F. (2022). On the tightness of SDP relaxations of QCQPs. *Mathematical Programming*, 193(1):33–73.

Wolkowicz, H., Saigal, R., and Vandenberghe, L. (2012). *Handbook of semidefinite programming: theory, algorithms, and applications*, volume 27. Springer Science & Business Media, Kluwer Academic Publishers, Boston.

Xia, W., Vera, J. C., and Zuluaga, L. F. (2020). Globally solving nonconvex quadratic programs via linear integer programming techniques. *INFORMS Journal on Computing*, 32(1):40–56.

Yanıkoğlu, İ., Gorissen, B. L., and den Hertog, D. (2019). A survey of adjustable robust optimization. *European Journal of Operational Research*, 277(3):799–813.

Zamani, M. (2023). New bounds for nonconvex quadratically constrained quadratic programming. *Journal of Global Optimization*, 85(3):595–613.

Zhen, J., Den Hertog, D., and Sim, M. (2018). Adjustable robust optimization via Fourier–Motzkin elimination. *Operations Research*, 66(4):1086–1100.

Zhen, J., Marandi, A., de Moor, D., den Hertog, D., and Vandenberghe, L. (2022). Disjoint bilinear optimization: A two-stage robust optimization perspective. *INFORMS Journal on Computing*, 34(5):2410–2427.

Zheng, X. J., Sun, X. L., and Li, D. (2011). Convex relaxations for nonconvex quadratically constrained quadratic programming: matrix cone decomposition and polyhedral approximation. *Mathematical Programming*, 129(2):301–329.