J Glob Optim
DOI 10.1007/s10898-016-0462-0

A branch and bound algorithm for quantified quadratic programming

F. Domes[1] · **A. Goldsztejn**[2]

Received: 27 November 2015 / Accepted: 2 September 2016

Abstract The aim of this paper is to find the global solutions of uncertain optimization problems having a quadratic objective function and quadratic inequality constraints. The bounded epistemic uncertainties in the constraint coefficients are represented using either universal or existential quantified parameters and interval parameter domains. This approach allows to model non-controlled uncertainties by using universally quantified parameters and controlled uncertainties by using existentially quantified ones. While existentially quantified parameters could be equivalently considered as additional variables, keeping them as parameters allows maintaining the quadratic problem structure, which is essential for the proposed algorithm. The branch and bound algorithm presented in the paper handles both universally and existentially quantified parameters in a homogeneous way, without branching on their domains, and uses some dedicated numerical constraint programming techniques for finding a robust, global solution. Several examples clarify the theoretical parts and the tests demonstrate the usefulness of the proposed method.

Keywords Branch and bound · Robust optimization · Quadratic problem · First order conditions

1 Introduction

This paper addresses optimization problems where some parameters are uncertain. The current theory about robust optimization [3–6,15,16] considers universally quantified parameters (allowing decisions that are robust with respect to these uncertainties), and convex prob-

✉ A. Goldsztejn
Alexandre.Goldsztejn@gmail.com

F. Domes
Ferenc.Domes@univie.ac.at

1 Faculty of Mathematics, University of Vienna, Vienna, Austria

2 CNRS, IRCCYN (UMR 6597), Nantes, France

Published online: 08 September 2016

lems with concave parameter dependence of the constraints (so that their maximization is tractable).

In this paper for non-convex quadratic problems, we address a more general treatment of parameters where some of them can be existentially quantified, but restrict our investigations to problems where the constraints depend only linearly on the parameters. Universally quantified parameters correspond to the usual non-controlled uncertainties (e.g. manufacturing errors in a design process), while existentially quantified parameters correspond to controlled uncertainties (e.g. a quality tuning that is performed during the manufacturing process, after the manufacturing errors have been measured). Although, the existentially quantified parameters could be included in the model as variables, this approach has two serious drawbacks: it increases the number of variables and it breaks the problem structure.

In this paper, we analyze (non-convex) quantified quadratic constraints (QQC) and quantified quadratic optimization problems, develop specific pruning, feasibility checking and branching methods, and integrate these in a branch and bound algorithm. The following example illustrates the impact of quantified parameters on the solution set of a quadratic constraint.

Example 1 Consider the quantified quadratic optimization problem

$$\begin{array}{ll} \min & x_1 + x_2 \\ \text{s.t.} & Q_{11}a_{11} \in [2,4], \quad Q_{22}a_{22} \in [2,6], \quad Q_{12}a_{12} \in [-3,1], \\ & a_{11}x_1^2 + a_{22}x_2^2 + a_{12}x_1x_2 \leq 1. \end{array} \tag{1}$$

where Q_{11}, Q_{22} and Q_{12} are arbitrary quantifiers. As proved in Sect. 3, the order of the quantification is irrelevant inside QQCs. Its feasible set, denoted by

$$\Omega_{Q_{11},Q_{22},Q_{12}}, \tag{2}$$

and its optimal solution are displayed in Fig. 1 for different choices of Q_{11}, Q_{22} and Q_{12}. It is easily seen that the quadratic function $a_{11}x_1^2 + a_{22}x_2^2 + a_{12}x_1x_2$ is convex for every value of the parameters inside their domains, therefore the sets

$$\Omega(a_{11}, a_{22}, a_{12}) := \{x \in \mathbb{R}^2 : a_{11}x_1^2 + a_{22}x_2^2 + a_{12}x_1x_2 \leq 1\} \tag{3}$$

are convex for all these parameter values. The feasible set $\Omega_{\forall,\forall,\forall}$ is the intersection of all convex sets $\Omega(a_{11}, a_{22}, a_{12})$ and therefore convex. However, when some quantifiers are existential, $\Omega_{Q_{11},Q_{22},Q_{12}}$ involves some unions of convex sets which is in general non convex (see Fig. 1).

The quantifiers Q_{11} and Q_{22} can be eliminated from (1) by the quantifier elimination method presented in Sect. 3, and we find that (1) is equivalent to

$$\begin{array}{ll} \min & x_1 + x_2 \\ \text{s.t.} & Q_{12}a_{12} \in [-3,1], \ (3+\epsilon_{11})x_1^2 + (4+2\epsilon_{22})x_2^2 + a_{12}x_1x_2 \leq 1, \end{array} \tag{4}$$

where $\epsilon_{ij} = \epsilon_{Q_{ij}} \in \{-1, 1\}$ is defined by Definition 1 p. 5. Note that depending on the quantifiers the coefficients of the quadratic terms are instantiated to either the upper or lower bound, i.e. $3 + \epsilon_{11} \in \{2, 4\}$ and $4 + 2\epsilon_{22} \in \{2, 6\}$. Therefore the quadratic function $(3 + \epsilon_{11})x_1^2 + (4 + 2\epsilon_{22})x_2^2 + a_{12}x_1x_2$ is convex for all values of a_{12} inside its domain, for all $\epsilon_{ii} \in \{-1, 1\}$. As a consequence, the feasible set $\Omega_{Q_{11},Q_{22},\forall}$ is convex for arbitrary quantifiers Q_{11} and Q_{22}, as also illustrated in Fig. 1. This obviously holds more generally for QQCs which have convex feasible sets provided that the quadratic function is convex for all parameter values inside their domains and that off diagonal parameters are universally quantified.

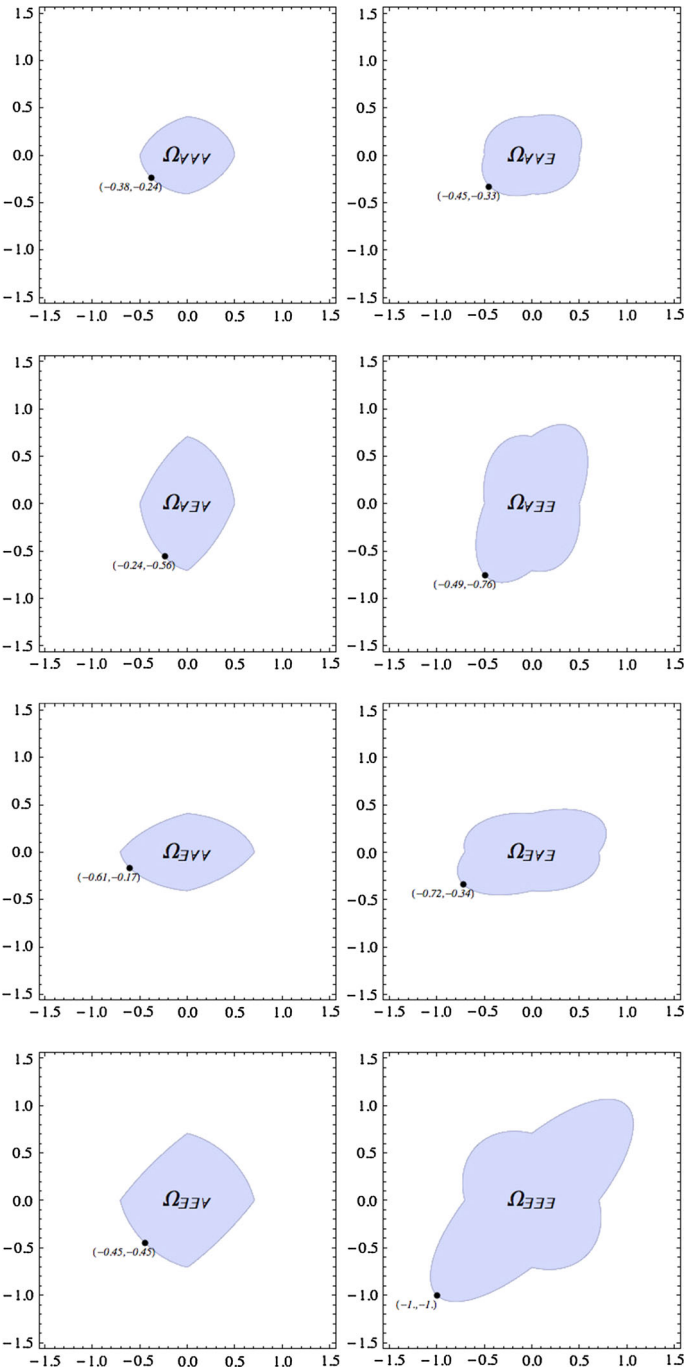

Fig. 1 The feasible sets and the optimal solutions of (1) for each possible quantification

Using the quantifier elimination method presented in Sect. 3 the quantifier Q_{12} can also be eliminated from (4), leading to the quantifier free equivalent problem

$$\begin{array}{ll} \min & x_1 + x_2 \\ \text{s.t.} & (3+\epsilon_{11})x_1^2 + (4+2\epsilon_{22})x_2^2 - x_1x_2 + 2\epsilon_{12}|x_1x_2| \leq 1. \end{array} \tag{5}$$

Although useful in proofs and for checking feasibility, the quantifier free formulation (5) of QQCs increases the number of occurrences of the variables, is non quadratic, non smooth, therefore not suitable for several local optimization techniques and numerical constraint programming, and conclusively global optimization. On the other hand, the quantifier free expression (5) shows that once the signs of the variables are decided inside a domain, the absolute values are simplified and (5) recovers a non parametric quadratic form, where monomial coefficients are instantiated to some bounds of their domains depending on both the quantifier and the variable signs. This observation will motivate the theory developed in the following sections.

QQCs have not been studied yet in the literature, but robust quadratic constraints, where all quantifiers are universal, are a special case of QQCs and the theory and algorithm presented in this paper does improve the state of the art on this restricted topic. In particular, [30] tackles exactly this class of robust quadratic problems, and proposes the same quantifier free expression of the robust constraint in the restricted case where variables have constant signs (see Sect. 3.3 for more details). Several approaches have been proposed to handle general non-convex universally quantified optimization problems (also called semi-infinite problems): e.g., [8,38] use a finite number of parameter samples, which is increased in order to converge to feasible and optimal solutions and [31,32] use adaptive relaxation or approximations into NLPs, which converge to the optimal solution.

Outline of the paper Section 2 gives the basic notation, summarizes the concept of interval analysis, explains rigorous rounding of the inputs for parameter domains, and formally defines the problem class treated in the paper. The structure of QQCs is exploited in Sect. 3: in Sect. 3.1 the main theorem for eliminating quantified parameters is presented, as well as a corollary of the theorem providing a quantifier free equivalent expression of QQCs. An expression of the generalized gradient of a QQC is derived in Sect. 3.2. The related work about robust quadratic constraints and their generalized gradients is reviewed in Sect. 3.3. The main branch and bound algorithm is discussed in Sect. 4. In particular Sect. 4.1 presents a generic branch and bound algorithm, for which the pruning, proving feasibility and branching steps (specially tailored for solving quantified quadratic global optimization problems) are discussed in the Sects. 4.2–4.4. Finally in Sect. 5, experiments solving artificial, quantified, quadratic global optimization problems, using the proposed branch and bound algorithm are presented.

2 Preliminaries

2.1 Notation

For quantifiers the following definition, is used throughout the paper:

Definition 1 Let Q be a quantifier. We define the *quantifier sign function* as

$$\epsilon_Q := \begin{cases} 1 & \text{if } Q = \forall, \\ -1 & \text{if } Q = \exists. \end{cases} \tag{6}$$

For quantifiers Q_i (respectively Q_{ij}) instead of ϵ_{Q_i} (respectively $\epsilon_{Q_{ij}}$) we will simply write ϵ_i (respectively ϵ_{ij}).

We also define the relative deviation between x and y, which will be used within the stoping criteria of the branch and bound algorithm:

Definition 2 The relative distance between scalar and vectors are defined as follows:

$$\mathrm{d_r}(a,b) := \frac{a-b}{\max\{1,|a|,|b|\}} \quad \text{where } a,b \in \mathbb{R} \tag{7}$$

$$\mathrm{d_r}(x,y) := \max_i(\mathrm{d_r}(x_i,y_i)) \quad \text{where } x,y \in \mathbb{R}^n. \tag{8}$$

The convex hull of a set $E \subseteq \mathbb{R}^n$ is denoted by $\mathrm{co}\, E$.

Finally we note that the concept of uncertainty used in this paper is unrelated to the uncertainty in a stochastic process; it is bounded epistemic uncertainty (not knowing more than bounds for a number), not aleatoric uncertainty (for which a probability distribution exists).

2.2 Interval analysis

Interval analysis is a branch of numerical analysis that was born in the 1960's. It consists of computing with intervals of reals instead of reals, providing a framework for handling uncertainties and verified computations (see e.g. [2,25,33,34] for a survey). Interval analysis is a key ingredient for numerical constraint programming (see e.g. [21]) and global optimization (see e.g. [20,26]).

Intervals, interval vectors and interval matrices are denoted by boldface symbols. For the notation in interval analysis we mostly follow [23]. An interval is a closed connected subset of $\mathbb{R}$, the set of intervals being denoted by $\mathbb{IR}$. Intervals are denoted by boldface symbols, e.g. $\mathbf{x} \subseteq \mathbb{R}$. There are two equivalent ways of defining interval vectors, the set of n dimensional interval vectors being denoted by $\mathbb{IR}^n$. On the one hand, being given two vectors $\underline{x} \leq \overline{x} \in \mathbb{R}^n$ (where the inequality is defined component-wise), an interval of vectors is obtained by considering $\mathbf{x} := \{x \in \mathbb{R}^n : \underline{x} \leq x \leq \overline{x}\}$. On the other hand, being given intervals $\mathbf{x}_i \in \mathbb{IR}$ for $i \in \{1,\ldots,n\}$, a vector of intervals is obtained by considering $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)^T := \{x \in \mathbb{R}^n : \forall i \in \{1,\ldots n\}, x_i \in \mathbf{x}_i\}$. These two definitions are obviously equivalent and will be used interchangeably. Interval matrices are defined similarly to interval vectors as either intervals of matrices or matrices of intervals, the set of $n \times m$ interval matrices being denoted by $\mathbb{IR}^{n\times m}$. A real number $x \in \mathbb{R}$ (respectively a real vector $x \in \mathbb{R}^n$ or a real matrix $A \in \mathbb{R}^{n\times m}$) will be identified with the degenerate interval $[x,x]$ (respectively a degenerated interval vector $[x,x]$ or a degenerated interval matrix $[A,A]$).

Operations $\circ \in \{+,\times,-,\div\}$ are extended to intervals in the following way: $\mathbf{x} \circ \mathbf{y} := \{x \circ y : x \in \mathbf{x}, y \in \mathbf{y}\}$. The division is defined for intervals $[\underline{y},\overline{y}]$ that do not contain zero. Unary elementary functions $f(x)$ like $\exp(x)$, $\ln(x)$, $\sin(x)$, etc., are also extended to intervals similarly: $f(\mathbf{x}) = \{f(x) : x \in \mathbf{x}\}$. All these elementary interval extensions form the interval arithmetic (IA). As real numbers are identified to degenerated intervals, the IA actually generalizes the real arithmetic, and mixed operations like $1 + [1,2] = [2,3]$ are interpreted as interval operations, e.g. in this case $[1,1]+[1,2] = [2,3]$. An interval function $\mathbf{f} : \mathbb{IR}^n \longrightarrow \mathbb{IR}^m$ is an interval extension of the real function $f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ if for all $\mathbf{x} \in \mathbb{IR}^n$ we have $\mathbf{f}(\mathbf{x}) \supseteq \{f(x) : x \in \mathbf{x}\}$. Thus interval extensions allow computing enclosures of real functions range over boxes. So called natural interval extensions of a function are obtained by evaluating an expression of this function for interval arguments using the interval arithmetic.

When every variable has one unique occurrence in the function's expression the natural interval extension computes the exact function range. However, when the expression of a function has several occurrences of some variable, its interval evaluation may be pessimistic. The pessimism of interval evaluation is one of the critical issue to be tackled when applying interval analysis. Finally, arithmetic operations on interval vectors and matrices are extended to intervals using interval arithmetic, e.g. $\mathbf{A}\,\mathbf{x} = \mathbf{y}$ with $\mathbf{y}_i = \sum_k \mathbf{A}_{ik}\mathbf{x}_k$ with the enclosure property $\mathbf{y} \supseteq \{Ax : A \in \mathbf{A}, x \in \mathbf{x}\}$ (note that since the expression of y_i contains only one occurrence of each involved interval, $\mathbf{y}$ is actually the smallest interval vector satisfying this enclosure).

The *midpoint* and *radius* of an interval vector $\mathbf{x} \in \mathbb{IR}^n$ (intervals correspond to $n = 1$) and an interval matrix $\mathbf{A} \in \mathbb{IR}^{n\times n}$ are defined as follows:

$$\mathrm{mid}(\mathbf{x}) = (\overline{x} + \underline{x})/2 \in \mathbb{R}^n, \quad \mathrm{rad}(\mathbf{x}) = (\overline{x} - \underline{x})/2 \in \mathbb{R}^n, \tag{9}$$

$$\mathrm{mid}(\mathbf{A}) = (\overline{A} + \underline{A})/2 \in \mathbb{R}^{n\times n}, \quad \mathrm{rad}(\mathbf{A}) = (\overline{A} - \underline{A})/2 \in \mathbb{R}^{n\times n}. \tag{10}$$

Midpoints will usually be shortly written $\hat{x}$ or $\hat{A}$, and radii by $\check{x}$ or $\check{A}$.

Rounded computations As real numbers are approximately represented by floating point numbers [17], the interval arithmetic (IA) cannot match the real definitions of interval extensions exactly. In order to preserve the inclusion property, the IA has to be implemented using an outward rounding. For example, $[1, 3]/[10, 10] = [0.1, 0.3]$ while both 0.1 and 0.3 cannot be exactly represented with standard floating point numbers. Therefore, the computed result will be $[0.1^-, 0.3^+]$ where 0.1^- (respectively 0.3^+) is a floating point number smaller than 0.1 (respectively greater than 0.3). We expect actually the greatest floating point number smaller than 0.1 and the smallest floating point number greater than 0.3, which is often achieved by interval arithmetic implementations. Among other implementations of IA, we can cite the C/C++ libraries PROFIL/BIAS [27] and Gaol [19], the Matlab toolbox INTLAB [35] and Mathematica [40].

As mentioned above, the algorithm remains correct when the floating point interval arithmetic is implemented using outward rounding. However, some special care has to be given to the numerical inputs: inputs for solvers are usually given in decimal format, while such decimal numbers are generally not exactly represented in binary format: while integers as well as sums of negative powers of two like 0.5 or 0.25 of reasonable size are exactly represented, decimal numbers like 0.1 have to be rounded. In the context of non-parametric global optimization, such inputs are used for domain bounds and constant constants, where outer rounding allows safely enclosing the real optimum.

Here, such decimal number inputs are also used to define the domains of the universally and existentially quantified parameters. A rigorous process of these non machine-representable data is required. This is illustrated by the following example.

Example 2 Consider the quantified constraint $\forall a \in [0.1, 1], ax \geq 1$, whose solution set is $[10, +\infty)$. This solution set can be computed using the equivalent constraint $\forall a \in [0.1, 1], x \geq \frac{1}{a}$, the domain of $\frac{1}{a}$ being exactly $\frac{1}{[0.1,1]} = [1, \frac{1}{0.1}]$. Since 0.1 is not representable in the standard floating format, the value of 10 could be rejected from the solution set due to rounding errors of the parameter domains.

More generally, outer rounding of universally quantified or inner rounding of existentially quantified parameter domains will reduce the feasible set. On the other hand, inner rounding of universally quantified or outer rounding of existentially quantified parameter domains will enlarge the feasible set. In both cases, the minimizer of the problem could become wrong.

Therefore, each parameter domain bound is rounded both upward and downward in order to be enclosed in a tiny interval that will be used every time a parameter domain bound is required. For example, in the quantifier-free expression (21), each mid and rad operation is computed using the tiny interval for parameter domain bounds, giving rise to a rigorous enclosure of the actual mid and rad even in the presence of decimal input rounding errors for these domains. This process will not be explicitly described in the rest of the paper, where parameter domain bounds will be assumed to be reals.

2.3 Problem definition

In this section the formal definition of the problem class treated the paper is given. Let

$$g(A, b, c, x) := \sum_{(i,j)\in\mathcal{A}} a_{ij} x_i x_j + \sum_{i\in\mathcal{B}} b_i x_i + c = x^T A x + b^T x + c \tag{11}$$

be a quadratic function, with the index sets $\mathcal{A} = \{(i, j) : 1 \leq i \leq j \leq n\}$ and $\mathcal{B} = \{i : 1 \leq i \leq n\}$. Because of the structure of the index set $\mathcal{A}$ the matrix A is upper triangular. When in addition to (11), for each parameter a_{ij} an interval domain $\mathbf{a}_{ij} \in \mathbb{IR}$ and a quantifier $Q_{ij} \in \{\forall, \exists\}$, and for each b_i an interval domain $\mathbf{b}_i \in \mathbb{IR}$ and a quantifier $Q_i \in \{\forall, \exists\}$ are given, a *quantified quadratic constraint* (QQC) is defined as

$$Q_{\mathcal{A}} a_{\mathcal{A}} \in \mathbf{a}_{\mathcal{A}}, Q_{\mathcal{B}} b_{\mathcal{B}} \in \mathbf{b}_{\mathcal{B}}, g(A, b, c, x) \leq 0. \tag{12}$$

The terms $Q_{\mathcal{A}} a_{\mathcal{A}} \in \mathbf{a}_{\mathcal{A}}$ and $Q_{\mathcal{B}} b_{\mathcal{B}} \in \mathbf{b}_{\mathcal{B}}$ denote the sequence of quantifications $Q_{ij} a_{ij} \in \mathbf{a}_{ij}, (i, j) \in \mathcal{A}$, and $Q_i b_i \in \mathbf{b}_i, i \in \mathcal{B}$, respectively. No domain or quantifier is associated to c since they could be eliminated trivially. Usually, the order of the quantifications has an impact on the semantic of the constraint but, as proved later in the paper in Sect. 3, the quantifiers all commute in case the constraints are quadratic, and therefore the quantification order does not need to be specified.

A *quantified quadratic optimization problem* consists of minimizing a quadratic objective function f over the domain Ω defined by the m quantified quadratic constraints, formally

$$\min_{x\in\Omega} f(x), \tag{13}$$

where

$$\Omega := \cap_{k=1}^{m} \Omega^{(k)}, \tag{14}$$

$$\Omega^{(k)} := \{x \in \mathbb{R}^n \mid Q_{\mathcal{A}^{(k)}} a_{\mathcal{A}} \in \mathbf{a}_{\mathcal{A}}^{(k)}, Q_{\mathcal{B}^{(k)}} b_{\mathcal{B}} \in \mathbf{b}_{\mathcal{B}}^{(k)}, g(A, b, c^{(k)}, x) \leq 0\}. \tag{15}$$

Note that parameters are all independent: Correlated parameters generally break the structure of the constraint and require more specific treatments which are out of the scope of this paper. Parameters in the objective function $f(x)$ can be handled adding the constraint $f(x) \leq y$ and minimizing the auxiliary variable y.

3 Eliminating quantifiers from quantified quadratic constraints

3.1 Quantifier free expression

Theorem 1 provides a quantifier free expression equivalent to a general class of quantified constraints, which are linear with respect to parameters.

Lemma 1 *Let $u, \alpha, \beta \in \mathbb{R}$ and $\epsilon \in \{-1, 1\}$. Then*

$$\alpha u + \epsilon\beta|u| = \begin{cases} (\alpha+\beta)u & \text{if } \epsilon u \geq 0 \\ (\alpha-\beta)u & \text{otherwise.} \end{cases} \tag{16}$$

Proof Considering the four cases depending on the sign of u and ϵ, the expression $\alpha u + \epsilon\beta|u|$ is equal to $(\alpha+\beta)u$, $(\alpha-\beta)u$, $(\alpha-\beta)u$ or $(\alpha+\beta)u$ when respectively $u \geq 0 \wedge \epsilon = 1$, $u \geq 0 \wedge \epsilon = -1$, $u \leq 0 \wedge \epsilon = 1$ or $u \leq 0 \wedge \epsilon = -1$. This matches the factored expression given in the statement. □

Theorem 1 *Let $h_k(x) : \mathbb{R}^n \to \mathbb{R}$ for $i \in \{0, \ldots, p\}$, $\mathbf{u} \in \mathbb{IR}^p$ and Q a quantifier vector, i.e., $Q_i \in \{\exists, \forall\}$ for $i \in \{1, \ldots, p\}$. Consider the quantified constraint*

$$Q_p u_p \in \mathbf{u}_p, \ldots, Q_2 u_2 \in \mathbf{u}_2, Q_1 u_1 \in \mathbf{u}_1, h_0(x) + \sum_{k=1}^{p} u_k h_k(x) \leq 0, \tag{17}$$

where the quantifier order is fixed. Then:

(i) (17) *is equivalent to*

$$h_0(x) + \sum_{k=1}^{p} \hat{u}_k h_k(x) + \sum_{k=1}^{p} \epsilon_k \check{u}_k |h_k(x)| + c \leq 0, \tag{18}$$

where $\hat{u}_k = \text{mid}(\mathbf{u}_k)$, $\check{u}_k = \text{rad}(\mathbf{u}_k)$ and ϵ_k is the quantifier sign function as defined by Definition 1.

(ii) Quantifiers commute in (17).

Proof For (i), we use the facts that $\exists u \in \mathbf{u}, g(u) \leq 0 \iff \min_{u\in\mathbf{u}} g(u) \leq 0$ and $\forall u \in \mathbf{u}, g(u) \leq 0 \iff \max_{u\in\mathbf{u}} g(u) \leq 0$. Defining $\text{opt}^k := \min$ if $Q_k = \exists$, or $\text{opt}^k := \max$ if $Q_k = \forall$, we can eliminate the quantifiers from (17) starting from Q_1 until Q_s:

$$\begin{array}{cccccc} Q_p u_p \in \mathbf{u}_p & \ldots & Q_3 u_3 \in \mathbf{u}_3 & Q_2 u_2 \in \mathbf{u}_2 & \text{opt}^1_{u_1\in\mathbf{u}_1} & h_0(x) + \sum_{k=1}^{p} u_k h_k(x) \leq 0, \\ Q_p u_p \in \mathbf{u}_p & \ldots & Q_3 u_3 \in \mathbf{u}_3 & \text{opt}^2_{u_2\in\mathbf{u}_2} & \text{opt}^1_{u_1\in\mathbf{u}_1} & h_0(x) + \sum_{k=1}^{p} u_k h_k(x) \leq 0, \\ & & & \vdots & & \\ \text{opt}^s_{u_s\in\mathbf{u}_s} & \ldots & \text{opt}^3_{u_3\in\mathbf{u}_3} & \text{opt}^2_{u_2\in\mathbf{u}_2} & \text{opt}^1_{u_1\in\mathbf{u}_1} & h_0(x) + \sum_{k=1}^{p} u_k h_k(x) \leq 0. \end{array} \tag{19}$$

Since summation is increasing with respect to each variable, this is equivalent to

$$h_0(x) + \sum_{k=1}^{p} \text{opt}^k_{u_k\in\mathbf{u}_k} u_k h_k(x) \leq 0. \tag{20}$$

Finally, depending on the sign of $g(x)$ and on the quantifier Q_k, $\text{opt}^k_{u_k\in\mathbf{u}_k} u_k h_k(x)$ is either $\overline{u}_k h_k(x) = (\hat{u}_k + \check{u}_k)h_k(x)$ or $\underline{u}_k h_k(x) = (\hat{u}_k - \check{u}_k)h_k(x)$, which can be checked to be compactly written as $\hat{u}_k h(x) + \epsilon_k \check{u}_k |h_k(x)|$ using Lemma 1.

For (ii), just note that while the quantifier order is fixed in (17), the quantifier free expression (18) does not depend on it (since addition is commutative and associative), so any quantifier order will lead to the same quantifier free expression. □

The quantifier free expression of the quantified quadratic constraint directly follows from Theorem 1.

Corollary 1 *For given $Q_{\mathcal{A}}$, $\mathbf{a}_{\mathcal{A}}$, $Q_{\mathcal{B}}$, $\mathbf{b}_{\mathcal{B}}$ and c the quantified quadratic constraint (12) is equivalent to $g^\bullet(x) \leq 0$ with*

$$g^\bullet(x) = \sum_{1\leq i\leq j\leq n} \hat{a}_{ij}x_ix_j + \sum_{1\leq i\leq j\leq n} \epsilon_{ij}\check{a}_{ij}|x_ix_j| + \sum_{1\leq i\leq n} \hat{b}_ix_i + \sum_{1\leq i\leq n} \epsilon_i\check{b}_i|x_i| + c \quad (21)$$

where $\hat{a}_{ij} := \text{mid}(\mathbf{a}_{ij})$, $\check{a}_{ij} := \text{rad}(\mathbf{a}_{ij})$, $\hat{b}_i := \text{mid}(\mathbf{b}_i)$, $\check{b}_{ij} = \text{rad}(\mathbf{b}_i)$, ϵ_{ij} is the quantifier sign function as defined by Definition 1.

Proof Apply Theorem 1 with $h_0(x) = c$ and either $h_k(x) = x_ix_j$ and $u_k = a_{ij}$, or $h_k(x) = x_i$ and $u_k = b_i$. □

The quadratic terms $\hat{a}_{ii}x_i^2 + \epsilon_{ii}\check{a}_{ii}|x_i^2|$ in (21) can be simplified to $(\hat{a}_{ii} + \epsilon_{ii}\check{a}_{ii})x_i^2$ where no absolute value occurs anymore, resulting in the following expression of $g^\bullet(x)$:

$$\sum_{1\leq i\leq n} (\hat{a}_{ii} + \epsilon_{ii}\check{a}_{ii})x_i^2 + \sum_{1\leq i<j\leq n} (\hat{a}_{ij}x_ix_j + \epsilon_{ij}\check{a}_{ij}|x_ix_j|) + \sum_{1\leq i\leq n} (\hat{b}_ix_i + \epsilon_i\check{b}_i|x_i|) + c. \quad (22)$$

The quantifier-free expression (22) can be used within a local optimizer to search for good feasible points. It will also be used to obtain the generalized gradient of the quantified constraint in Sect. 3.2, which can be of great interest for rejection tests based on first order conditions (see Sect. 4.2.2). However, (22) is not well suited to numerical constraint programming because:

- the number of occurrences of variables has been significantly increased,
- the quadratic structure of the problem has been broken.

For these reasons, the efficiency of several filtering techniques may be seriously degraded when using (22) directly.

3.2 Generalized gradients

The quantifier-free expression of Corollary 1 allows stating first order necessary conditions for quantified quadratic programming using generalized gradients (see e.g. [10]).

Lemma 2 *Let $h : \mathbb{R}^n \to \mathbb{R}$ be Lipschitz continuous, and $D \subseteq \mathbb{R}^n$ be the (open) set where it is differentiable. Suppose that there exist finitely many non-overlapping open sets E_i, $i \in \mathcal{I}$, forming a partition of D. Define $H_i : E_i \to \mathbb{R}^n$ by $H_i(x) = \nabla h(x)$, and suppose that $H_i(x)$ can be extended continuously to $\overline{E}_i$. Then*

$$\partial h(x) = \text{co}\{H_i(x) : x \in \overline{E}_i\}. \quad (23)$$

Proof The generalized gradient $\partial g(x)$ is the convex hull of the limit set of all sequences $(\nabla g(x_k))_{k\in\mathbb{N}}$ such that $(x_k)_{k\in\mathbb{N}}$ converges toward x and g differentiable at x_k. Let Y be this limit set, it is sufficient to prove that $Y = \{H_i(x) : x \in \overline{E}_i\}$ so that their respective convex hulls coincide.

We first prove that $Y \subseteq \{H_i(x) : x \in \overline{E}_i\}$. Consider an arbitrary $y \in Y$ which is a limit point of $(x_k)_{k\in\mathbb{N}}$, $x_k \in D$ and $\lim x_k = x$. Since there are finitely many sets E_i in the partition of D, there exists $i \in \mathcal{I}$ and a subsequence $(y_k)_{k\in\mathbb{N}}$ of $(x_k)_{k\in\mathbb{N}}$ such that $x_k \in E_i$, so $x \in \overline{E}_i$, and $\lim \nabla h(x_k) = y$. Now, $\nabla h(x_k) = H_i(x_k)$ because $x_k \in E_i$, and $\lim H_i(x_k) = H_i(x)$ because H_i is continuous by hypothesis. We have proved that there exists $i \in \mathcal{I}$ such that $x \in \overline{E}_i$ and $y = H_i(x)$.

Second, we prove that $\{H_i(x) : x \in \overline{E}_i\} \subseteq Y$. Consider an arbitrary $i \in \mathcal{I}$ such that $x \in \overline{E}_i$. Since $x \in \overline{E}_i$, there exists $(x_k)_{k\in\mathbb{N}}$ such that $x_k \in E_i$ and $\lim x_k = x$. Now since H_i is continuous, we have $\lim H_i(x_k) = H_i(x)$, or equivalently $\lim \nabla h(x_k) = H_i(x)$, which proves that $H_i(x) \in Y$. □

The following theorem provides an expression of the generalized gradient of the quantified quadratic constraint defined by (12) or (21). For this purpose, let us define

$$\operatorname{sign}(x) := \big\{\alpha \in \{-1, 1\}^n : \forall i \in \{1, \ldots, n\}, \alpha_i x_i \geq 0\big\}, \tag{24}$$

then $\operatorname{sign}(x)$ characterizes every (closed) orthant to whom x belong: the (closed) orthant associated to $\alpha \in \{-1, 1\}^n$ is defined by $\{x \in \mathbb{R}^n : \alpha_i x_i \geq 0\}$.

Theorem 2 *The generalized gradient $\partial g^\bullet(x)$ of (22) is*

$$\operatorname{co}\{(\hat{A} + \hat{A}^T + (\check{A} + \check{A}^T) \circ (\alpha\alpha^T) \circ \Sigma)x \ + \ \hat{b} + \check{b} \circ \alpha \circ \epsilon : \alpha \in \operatorname{sign}(x)\} \tag{25}$$

where co *is the convex hull, $\circ$ is the entry-wise product, $\Sigma_{ij} = \epsilon_{ij}$ if $j \geq i$, $\Sigma_{ij} = \epsilon_{ji}$ otherwise, and $\epsilon = (\epsilon_i)$, are quantifiers sign functions defined in Definition 1.*

Proof We apply Lemma 2 to $g^\bullet$ using $\mathcal{I} = \{-1, 1\}^n$, and $E_\alpha = \{x \in \mathbb{R}^n : x_i\alpha_i > 0\}$ for $\alpha \in \mathcal{I}$. Then assuming $x \in E_\alpha$ allows simplifying (21) to

$$g^\bullet(x) = \sum_{1\leq i\leq j\leq n} (\hat{a}_{ij} + \alpha_i\alpha_j\epsilon_{ij}\check{a}_{ij})x_i x_j + \sum_{1\leq i\leq n} (\hat{b}_i + \alpha_i\epsilon_i)x_i + c. \tag{26}$$

Since $\hat{A}$ and $\check{A}$ are upper triangular, and that $\alpha\alpha^T$ and Σ are symmetric, the restriction of the gradient of $g^\bullet$ to E_α is

$$\begin{aligned} H_\alpha &= \big((\hat{A} + \check{A} \circ (\alpha\alpha^T) \circ \Sigma) + (\hat{A} + \check{A} \circ (\alpha\alpha^T) \circ \Sigma)^T\big)x + \hat{b} + \check{b} \circ \alpha \circ \epsilon & (27)\\ &= (\hat{A} + \hat{A}^T + (\check{A} + \check{A}^T) \circ (\alpha\alpha^T) \circ \Sigma)x \ + \ \hat{b} + \check{b} \circ \alpha \circ \epsilon, & (28)\end{aligned}$$

which can be extended to $\overline{E}_\alpha$ using the same expression. Finally, Lemma 2 leads to

$$\partial h(x) = \operatorname{co}\{H_\alpha(x) : x \in \overline{E}_\alpha\}, \tag{29}$$

and the proof is concluded noting that $x \in \overline{E}_\alpha$ if and only if $\alpha \in \operatorname{sign}(x)$. □

Note that if k entries of x are zero then $\partial g^\bullet(x)$ is the convex hull of 2^k vectors which is usually intractable. The following corollary provides a tractable interval enclosure of $\partial g^\bullet(x)$.

Corollary 2 $\partial g^\bullet(x) \subseteq (\mathbf{A}^\bullet + \mathbf{A}^{\bullet T})x + \mathbf{b}^\bullet$ *with*

$$\mathbf{a}^\bullet_{ij} = \begin{cases} \overline{a}_{ij} & \text{if } i \leq j \wedge x_i x_j \epsilon_{ij} > 0 \\ \underline{a}_{ij} & \text{if } i \leq j \wedge x_i x_j \epsilon_{ij} < 0 \\ \mathbf{a}_{ij} & \text{if } i \leq j \wedge x_i x_j = 0 \\ 0 & \text{otherwise,} \end{cases}$$

$$\mathbf{b}^\bullet_i = \begin{cases} \overline{b}_i & \text{if } x_i\epsilon_i > 0 \\ \underline{b}_i & \text{if } x_i\epsilon_i < 0 \\ \mathbf{b}_i & \text{otherwise.} \end{cases} \tag{30}$$

Proof Consider an arbitrary vector in the convex hull (25)

$$\sum \lambda_k \Big((\hat{A} + \hat{A}^T + (\check{A} + \check{A}^T) \circ (s_k s_k^T) \circ \Sigma)x \ + \ \hat{b} + \check{b} \circ s_k \circ \epsilon\Big), \tag{31}$$

with $\sum \lambda_k = 1$, summations being performed for each $s_k \in \text{sign}(x)$. Factoring and using the fact that $\sum \lambda_k = 1$ leads to

$$\Big(\hat{A} + \hat{A}^T + (\check{A} + \check{A}^T) \circ \sum \lambda_k (s_k s_k^T \circ \Sigma)\Big) x \; + \; \hat{b} + \check{b} \circ \sum \lambda_k (s_k \circ \epsilon). \tag{32}$$

Let A^λ and b^λ be the matrix and vector involved in the previous expression, so that an arbitrary vector of (25) is written $A^\lambda x + b^\lambda$ with $\sum \lambda_i = 1$. We need to prove that $A^\lambda \in \mathbf{A}^\bullet + \mathbf{A}^{\bullet T}$ and $b^\lambda \in \mathbf{b}^\bullet$. For proving $A^\lambda \in \mathbf{A}^\bullet + \mathbf{A}^{\bullet T}$, first note that $(s_k s_k^T \circ \Sigma)_{ij} = s_{ki} s_{kj} \epsilon_{ij} \in \{-1, 1\}$ so $\sum \lambda_k (s_k s_k^T \circ \Sigma)_{ij} \in [-1, 1]$. Therefore, $A^\lambda_{ij} \in \hat{a}_{ij} + \hat{a}_{ji} + [-1, 1]\check{a}_{ij} + [-1, 1]\check{a}_{ji} = \mathbf{a}_{ij} + \mathbf{a}_{ji}$. Second, suppose that $x_i x_j \epsilon_{ij} > 0$, so that $(s_k s_k^T \circ \Sigma)_{ij} = s_{ki} s_{kj} \epsilon_{ij} = 1$ whatever is k, and $\sum \lambda_k (s_k s_k^T \circ \Sigma)_{ij} = 1$. Therefore $A^\lambda_{ij} = \hat{a}_{ij} + \hat{a}_{ji} + \check{a}_{ij} + \check{a}_{ji} = \overline{a}_{ij} + \overline{a}_{ji}$. Similarly, $x_i x_j \epsilon_{ij} < 0$ implies $A^\lambda_{ij} = \hat{a}_{ij} + \hat{a}_{ji} - \check{a}_{ij} - \check{a}_{ji} = \underline{a}_{ij} + \underline{a}_{ji}$. Finally $b^\lambda \in \mathbf{b}^\bullet$ is proved similarly. □

Example 3 Consider minimizing or maximizing x_1 subject to the universally quantified constraint

$$\forall a_{ij} \in [-1, 1]\,, \; 2n \sum_{1 \le i \le n} x_i^2 + \sum_{1 \le i < j \le n} a_{ij} x_i x_j \le 2n. \tag{33}$$

It is convex since the involved quadratic function is positive definite for all parameter values inside their domains and all quantifiers are universal. By Corollary 1, (33) is equivalent to

$$2n \sum_{1 \le i \le n} x_i^2 + \sum_{1 \le i < j \le n} |x_i x_j| \le 2n. \tag{34}$$

The points $x_\pm = (\pm 1, 0, \ldots, 0)$ satisfy the constraint, and the constraints are active at both x_+ and x_-. Furthermore $\partial g^\bullet(x_\pm)$ is the convex hull of 2^{n-1} vectors, formally

$$\partial g^\bullet(x_\pm) = \text{co}\{(\pm 4n, e_2, \ldots, e_n)^T \in \mathbb{R}^n : e_i \in \{-1, 1\}\}, \tag{35}$$

which using interval notation, can be written as

$$\partial g^\bullet(x_\pm) = (\pm 4n, [-1, 1], \ldots, [-1, 1])^T. \tag{36}$$

The interval matrix $\mathbf{A}^\bullet$ involved in the tractable inclusion is given by $\mathbf{a}_{ii} = 4n$ and $\mathbf{a}_{ij} = [-1, 1]$ for $i \ne j$, while $\mathbf{b}^\bullet = 0$, thus $\mathbf{A}^\bullet x_\pm + \mathbf{b}^\bullet = \partial g^\bullet(x_\pm)$ in this case. It is also easy to see that for $\lambda = -\frac{1}{4n}$ the first order optimality condition $0 \in (\pm 1, 0, \ldots, 0) + \lambda \partial g^\bullet(x_\pm)$ holds, and thus $x_\pm$ are respectively the global minimum and the global maximum of the problem.

As the example shows the enclosure provided by Corollary 2 is well suited for interval evaluation, especially in the form $\partial g^\bullet(\mathbf{x}) \subseteq (\mathbf{A}^\bullet + \mathbf{A}^{\bullet T})\mathbf{x} + \mathbf{b}^\bullet$ with

$$\mathbf{a}^\bullet_{ij} = \begin{cases} \overline{a}_{ij} & \text{if } i \le j \wedge \mathbf{x}_i \mathbf{x}_j \epsilon_{ij} > 0, \\ \underline{a}_{ij} & \text{if } i \le j \wedge \mathbf{x}_i \mathbf{x}_j \epsilon_{ij} < 0, \\ \mathbf{a}_{ij} & \text{if } i \le j \wedge 0 \in \mathbf{x}_i \mathbf{x}_j, \\ 0 & \text{otherwise,} \end{cases}$$

$$\mathbf{b}^\bullet_i = \begin{cases} \overline{b}_i & \text{if } \mathbf{x}_i \epsilon_i > 0, \\ \underline{b}_i & \text{if } \mathbf{x}_i \epsilon_i < 0, \\ \mathbf{b}_i & \text{otherwise,} \end{cases} \tag{37}$$

Since (37) is easy to evaluate, this formulation is useful for pruning methods, in particular for the first order rejection test discussed in Sect. 4.2.1 and illustrated in Example 6 and in Sect. 5.

3.3 Related work

A similar quantifier-free expression for robust quadratic problem has been provided in [30]. Its scope is however different. On the one hand, in the restricted case where all parameter domains are positive, the expression in [30] handles uncertainties on variables, which is out of the scope of Corollary 1. On the other hand, when parameter signs are not positive inside the domain, no uncertainties on variables are handled while the expression in [30] is restricted to positive variable domains and universally quantified parameters. In this case, it is equivalent to the quantifier-free expression of Corollary 1. Note that in Appendix B of [30], the authors claim that although their quantifier-free expression is restricted to positive variable domains, a change of basis can be performed to handle non-positive variable domains. However, this seems to be wrong since the quantifier-free expression of Corollary 1 (as well as Example 1) shows that the feasible set is generally non-smooth when variable signs are not constant inside the domain, while the equivalence after changing the basis claimed in [30] seems to prove its smoothness.[1]

Some necessary and sufficient first and second order conditions were derived in [22] for a different class of robust quadratic problems, namely the problem of minimizing $\frac{1}{2}x^T Ax+a^T x$ subject to one single robust quadratic constraint

$$\forall \mu \in [\mu_1, \mu_2], \quad \forall \delta \in [\delta_1, \delta_2], \quad g(\mu, \delta, x) \leq 0, \tag{38}$$

where $g(\mu, \delta, x) = \frac{1}{2}x^T(B_1+\mu B_2)x+(b_1+\delta b_2)^T x+\beta$, with $A, B_1, B_2 \in \mathbb{R}^{n\times n}$ symmetric real matrices and $b_1, b_2 \in \mathbb{R}^n$. This robustness requirement is different from the one considered here, where we consider the full interval matrix $[B_1, B_2]$ and the full interval vector $[b_1, b_2]$. Nevertheless, the universal quantifiers can be eliminated from (38) using Theorem 1: since $g(\mu, \delta, x) = h_0(x)+\mu h_1(x)+\delta h_2(x)$ with $h_0(x) = \frac{1}{2}x^T B_1 x+b_1^T x$, $h_1(x) = \frac{1}{2}x^T B_2 x$ and $h_2(x) = b_2^T x$, Theorem 1 proves that (38) is equivalent to

$$\begin{cases} \frac{1}{2}x^T B_1 x + \frac{1}{2}\mu_2 x^T B_2 x + b_1^T x + \delta_2 b_2^T x + \beta \leq 0 & \text{if } \ x^T B_2 x \geq 0 \wedge b_2^T x \geq 0 \\ \frac{1}{2}x^T B_1 x + \frac{1}{2}\mu_1 x^T B_2 x + b_1^T x + \delta_2 b_2^T x + \beta \leq 0 & \text{if } \ x^T B_2 x \leq 0 \wedge b_2^T x \geq 0 \\ \frac{1}{2}x^T B_1 x + \frac{1}{2}\mu_2 x^T B_2 x + b_1^T x + \delta_1 b_2^T x + \beta \leq 0 & \text{if } \ x^T B_2 x \geq 0 \wedge b_2^T x \leq 0 \\ \frac{1}{2}x^T B_1 x + \frac{1}{2}\mu_1 x^T B_2 x + b_1^T x + \delta_1 b_2^T x + \beta \leq 0 & \text{if } \ x^T B_2 x \leq 0 \wedge b_2^T x \leq 0 \end{cases} \tag{39}$$

Finally using Lemma 1 the four cases above can be grouped proving that (38) is equivalent to $g^\bullet(x) \leq 0$ with

$$g^\bullet(x) := \frac{1}{2}x^T(B_1 + \hat{\mu}B_2)x + \check{\mu}|x^T B_2 x| + (b_1 + \hat{\delta}b_2)^T x + \check{\delta}|b_2^T x| + \beta \tag{40}$$

where $\hat{\mu} = \text{mid}([\mu_1, \mu_2])$, $\check{\mu} = \text{rad}([\mu_1, \mu_2])$, $\hat{\delta} = \text{mid}([\delta_1, \delta_2])$ and $\check{\delta} = \text{rad}([\delta_1, \delta_2])$. From (40) we get the following general expression of the constraint generalized gradient (see e.g. [10]):

$$\partial g^\bullet(x) = \text{co}\{(B_1 + (\hat{\mu} \pm \check{\mu})B_2)x + (b_1 + (\hat{\delta} \pm \check{\delta})b_2)^T\} \tag{41}$$

$$= \{(B_1 + \mu B_2)x + (b_1 + \delta b_2) : \mu \in [\mu_1, \mu_2] \wedge \delta \in [\delta_1, \delta_2]\}, \tag{42}$$

which is the convex hull of four vectors. Using Lagrange conditions for nonsmooth problems (see e.g. [10]) we obtain the following necessary condition of the robust problem tackled

[1] The inconsistency seems to come from the fact that the new constraint after the change of variable involved in Appendix B of [30] contains several occurrences of universally quantified parameters, hence breaking the structure of the problem.

in [22]: under some constraint qualification (e.g. the $0 \notin \partial g^\bullet(x)$ whenever $g^\bullet(x) = 0$), if x_* is a local minimizer of the problem, then there exists $\lambda \geq 0$ such that $0 \in \nabla f(x) + \lambda \partial g^\bullet(\mathbf{x}_*)$, i.e. there exist $\mu \in [\mu_1, \mu_2]$ and $\delta \in [\delta_1, \delta_2]$ such that $(A + \lambda(B_1 + \mu B_2)x) + (a + \lambda(b_1 + \delta b_2))^T = 0$. This matches the condition provided in [22], although this result is weaker than the one provided in [22] where necessary and sufficient conditions are derived exploiting the specific problem structure as well as other technical requirements.

4 The branch and bound algorithm

Section 4.1 presents a generic branch and bound algorithm, for which the pruning, proving feasibility and branching steps (specially tailored for solving quantified quadratic global optimization problems) are discussed in Sects. 4.2, 4.3 and 4.4 respectively.

4.1 Generic branch and bound algorithm

Branch and bound is a common strategy for solving global optimization problems. It is based on successive subdivision[2] of the search space into smaller parts, pruning (eliminating or reducing) these parts until only those who tightly enclose the global solution(s) remain. On the other hand an efficient branch and bound also requires finding and verifying feasible points. Finding feasible points allows decreasing the objective upper bound $\overline{f}$, while pruning allows increasing its lower bound $\underline{f}$. Under some mild assumptions (on the constraints filtering operators and feasibility checker, and on the splitting strategy) the branch and bound algorithm makes the lower and upper bounds asymptotically converge toward the global minimum (however, usually implementations include some pragmatic stopping criteria). Algorithm 1 shows the description of the generic branch and bound algorithm.

Algorithm 1: Branch and bound algorithm

Input: $f : \mathbb{R}^n \to \mathbb{R}, \mathcal{C} = \{c_1, \ldots, c_m\}, \mathbf{x}^0 \in \mathbb{IR}^n, \epsilon > 0, \tau > 0$
Result: $\mathcal{E} \subseteq \mathbb{IR}^n$, $\underline{f} \in \mathbb{R} \cup \{-\infty\}$, $\overline{f} \in \mathbb{R} \cup \{+\infty\}$

$\mathcal{L} \leftarrow \{\mathbf{x}^0\}$; $\mathcal{E} \leftarrow \emptyset$; $\overline{f} \leftarrow +\infty$; $\underline{f} \leftarrow -\infty$;
while $\mathcal{L} \neq \emptyset$ **do**
  $\mathbf{x} \leftarrow \texttt{Extract}(\mathcal{L})$;
  if $\mathrm{d_r}(\overline{x}, \underline{x}) \leq \epsilon$ **or** $\mathrm{d_r}(\inf \mathbf{f}(\mathbf{x}), \overline{f}) \leq \tau$ **then**
    $\mathcal{E} \leftarrow \mathcal{E} \cup \mathbf{x}$;
  else
    $u \leftarrow \texttt{SafeUpperBound}(\mathbf{x})$;
    $\overline{f} \leftarrow \min\{\overline{f}, u\}$;
    $\mathbf{x} \leftarrow \texttt{Prune}_{\mathcal{C}, f, \overline{f}}(\mathbf{x})$;
    if $\mathbf{x} \neq \emptyset$ **then** $\mathcal{L} \leftarrow \mathcal{L} \cup \texttt{Split}(\mathbf{x})$;
  end
  $\underline{f} \leftarrow \min\{\inf \mathbf{f}(\mathbf{y}) : \mathbf{y} \in \mathcal{L} \cup \mathcal{E}\}$;
  $\mathcal{L} \leftarrow \texttt{Cleanup}_f(\mathcal{L}, \overline{f})$; $\mathcal{E} \leftarrow \texttt{Cleanup}_f(\mathcal{E}, \overline{f})$;
end
return $\mathcal{E}, \underline{f}, \overline{f}$;

[2] Also called splitting or branching.

The list of boxes $\mathcal{L}$ contains the boxes to be further processed by the algorithm, while the list of boxes $\mathcal{E}$ contains the boxes not to be processed anymore. The global minimizer is always contained inside the union of $\mathcal{L}$ and $\mathcal{E}$, and as $\mathcal{L}$ is empty at the end of the algorithm $\mathcal{E}$ is then an enclosure of the minimizer(s).

The **while** loop in the algorithm proceeds as follows: a new box $\mathbf{x}$ is extracted from $\mathcal{L}$ at Line 3 (usually, the one with the smallest objective lower bound $\inf \mathbf{f}(\mathbf{x})$). This box is not processed but stored in $\mathcal{E}$ if it satisfies the stopping criteria at Line 4: either it is considered too small (a positive ϵ is a typical safeguard enforcing the algorithm to halt), or proved not to be able to improve the objective enclosure more than the requirement $\mathrm{d_r}(\underline{f}, \overline{f}) \leq \tau$. It is otherwise processed as follows: first, one or several feasible points are searched inside $\mathbf{x}$, leading to a certified upper-bound u at Line 7 (which can be inf if no feasible point is found inside $\mathbf{x}$), allowing potentially improving the upper bound $\overline{f}$ at Line 8. Parts of the box $\mathbf{x}$ are filtered using techniques allowing proving the absence of candidate for global minimality (e.g. infeasibility, objective value greater than $\overline{f}$, or inconsistent first order conditions) at Line 9. Finally, in case the box is not fully filtered, the remaining part is split and inserted inside $\mathcal{L}$ at Line 10. After this, the lower-bound $\underline{f}$ is updated to the lowest lower-bound of the boxes in the list $\mathcal{L}$ at Line 12, and boxes of $\mathcal{L}$ and $\mathcal{E}$ whose lower bound is higher than $\overline{f}$ are removed at Line 13 in order to keep these lists as small as possible.

Remark 1 Operations performed at Line 3, Line 12 and Line 13 can be performed efficiently provided that both $\mathcal{L}$ and $\mathcal{E}$ contains not only boxes $\mathbf{x}$, but instead couples $(\mathbf{x}, \inf \mathbf{f}(\mathbf{x}))$ sorted according to $\inf \mathbf{f}(\mathbf{x})$.

4.2 Pruning

Pruning stands for reducing (or eliminating) a box $\mathbf{x}$ without loosing the global minimizer. Three necessary conditions for global optimality are usually used for pruning: the global minimizer x^* is feasible (hence non-feasible points that do not satisfy the constraints $\mathcal{C}$ can be pruned), every feasible point $\tilde{x}$ satisfies $f(x^*) \leq f(x)$ (hence given a feasible point $\tilde{x}$, the constraint $f(x) \leq f(\tilde{x})$ can be enforced), and it is a local minimizer (allowing using some first order conditions based pruning).

There are several different pruning methods; here we mention constraint propagation (e.g., see [7,9,29] or [12] for pruning dedicated to quadratic constraints) linear or convex relaxations (e.g., see [1,28,37]), rejection tests based on monotonicity or optimality conditions (e.g., see [18]) and exclusion regions based on the Krawczyk operator (e.g., see [36]). Since these methods often complement each other, they are usually combined as a single pruning step.

Constraint propagation handles naturally equality and inequality constraints, and is used for pruning the feasibility and the upper bound constraints. Section 4.2.1 describes how it can be efficiently applied to quantified quadratic constraints by relaxing it through partial quantifier elimination. Section 4.2.2 briefly shows how first order rejection tests can be applied. The other pruning methods mentioned above (relaxation and exclusion regions) have not been adapted to quantified quadratic constraints, and are not in the scope of the present paper.

4.2.1 Pruning quantified quadratic constraints

Numerical constraint pruning algorithms can naturally handle constraint expressions (including interval domains for parameters) by just performing interval operations with them. For

the existentially quantified constraint $\exists A \in \mathbf{A}, \exists b \in \mathbf{b}, g(A, b, c, x) \leq 0$ the domains of the variables $x \in \mathbf{x}$ are pruned by using

$$g(\mathbf{A}, \mathbf{b}, c, x) \leq 0, \tag{43}$$

where parameters are replaced by their interval domains $\mathbf{A} \in \mathbb{IR}^{n \times n}$ and $\mathbf{b} \in \mathbb{IR}^n$. This way pruning will enclose all $x \in \mathbf{x}$ which is a solution for any of the parameter values in their domains. Since a universal quantification implies an existential one, the interval constraint (43) is a relaxation of the QQC (12) for all $x \in \mathbf{x}$ and therefore (43) is called the *existential relaxation* for the rest of the paper. Note that (43) is usually only a crude relaxation because universal quantifiers are much more constraining than existential ones.

The following proposition describes the general approach for relaxing the quantified constraint (12), by instantiating the parameters to fixed numbers or intervals:

- If some variables have fixed signs inside the considered box, the corresponding parameters can be equivalently instantiated to one of the bounds of their interval domains using Theorem 1.
- If the sign of a variable is not decided, then no equivalent quantifier-free expression can be used, therefore existentially quantified parameter domains are kept unchanged, while universally quantified parameter domains are instantiated to their midpoint in order to decrease the over-relaxation.

Proposition 1 *Let* $\mathbf{x} \in \mathbb{IR}^n$. *For an arbitrary* $x \in \mathbf{x}$, *the quantified quadratic constraint* (12) *implies*

$$g(\mathbf{A}', \mathbf{b}', c, x) \leq 0, \tag{44}$$

where

$$\mathbf{a}'_{ij} = \begin{cases} \overline{a}_{ij} & \text{if } \epsilon_{ij}\mathbf{x}_i\mathbf{x}_j \geq 0 \\ \underline{a}_{ij} & \text{if } \epsilon_{ij}\mathbf{x}_i\mathbf{x}_j \leq 0 \\ \operatorname{mid}(\mathbf{a}_{ij}) & \text{if } 0 \in \operatorname{int} \mathbf{x}_i\mathbf{x}_j \wedge \epsilon_{ij} = 1 \\ \mathbf{a}_{ij} & \text{if } 0 \in \operatorname{int} \mathbf{x}_i\mathbf{x}_j \wedge \epsilon_{ij} = -1 \end{cases} \tag{45}$$

and

$$\mathbf{b}'_i = \begin{cases} \overline{b}_i & \text{if } \epsilon_i\mathbf{x}_i \geq 0 \\ \underline{b}_i & \text{if } \epsilon_i\mathbf{x}_i \leq 0 \\ \operatorname{mid}(\mathbf{b}_{ij}) & \text{if } 0 \in \operatorname{int} \mathbf{x}_i \wedge \epsilon_i = 1 \\ \mathbf{b}_i & \text{if } 0 \in \operatorname{int} \mathbf{x}_i \wedge \epsilon_i = -1 \end{cases}. \tag{46}$$

Proof Quantifiers of parameters a_{ij} satisfying $0 \notin \mathbf{x}_i\mathbf{x}_j$ and parameters b_i satisfying $0 \notin \mathbf{x}_i$ are eliminated like in the proof of Corollary 1, leading to a an equivalent reformulation of (12). The remaining universal quantifiers are eliminated by fixing the parameters to the midpoint of their domain, leading to a relaxation of (12). □

As shown in the following example, the relaxation (44) is in general a much better approximation of (12) than (43).

Example 4 We consider the constraint of Example 1. We have

$$\exists a_{11} \in [2, 4], \quad \exists a_{22} \in [2, 6], \quad \forall a_{12} \in [-3, 1], x^T Ax \leq 1, \tag{47}$$

where $A = (a_{ij})_{1 \leq i,j \leq 2}$ is upper triangular (i.e., $a_{21} = 0$), and the box $\mathbf{x} = ([-1.5, 1.5], [-1.5, 1.5])^T$. The quantifier free expression of this constraint obtained applying Corollary 1 is $g^\bullet(x) = 2x_1^2 + 2x_2^2 - x_1x_2 + 2|x_1x_2| \leq 1$, and its feasible set is depicted in

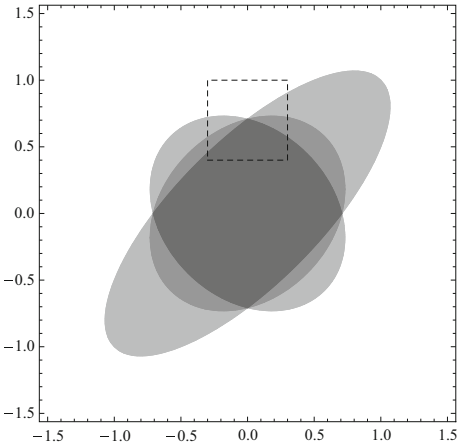

Fig. 2 The solution set of the quantified quadratic constraint from Example 4 and the solution set of its relaxations. From *darker* to *lighter gray*: the exact quantified quadratic constraint, its relaxation obtained using Proposition 1, and its existential relaxation. The *dashed box* is the domain used in Example 5

Fig. 2. The existential relaxation is $[2,4]x_1^2 + [2,6]x_2^2 + [-3,1]x_1x_2 \leq 1$, which is depicted in light gray on Fig. 2. Applying Proposition 1 instantiates $\mathbf{a}_{11}$ and $\mathbf{a}_{22}$ to their lower bound 2, and $\mathbf{a}_{12}$ to its midpoint -1, and gives rise to the relaxation $2x_1^2 + 2x_2^2 - x_1x_2 \leq 1$, which is much better than the existential relaxation. The different relaxations are visualized in Fig. 2.

4.2.2 Pruning using first order rejection tests

Rejection tests based on first order conditions, applicable also to non-smooth constraints (e.g., the ones defined in Corollary 1) have been proposed in [18]. One of them encloses the generalized gradients of the objective function and the potentially active constraints, and checks the rank[3] of the interval matrix built from these vectors (see [18] for more details). This method can be applied for pruning the QQC (13) since in (30) gives the required enclosure of the generalized gradients of the constraints.

Example 5 Consider again the constraint defined in Example 4, the objective function $f(x) = x_1$ and the domain $\mathbf{x} = ([-0.3, 0.3], [0.1, 0.7])^T$ (marked with a dashed line in Fig. 2). The gradient of the objective function is $\partial f(\mathbf{x}) = (1, 0)^T$ and using (37) we find that the interval enclosure of the generalized gradient is $\partial g^\bullet(\mathbf{x}) = ([-4.2, 2.2], [2.4, 6])^T$. Since there is no $\lambda \neq 0$ such that $\lambda \frac{\partial f(x)}{\partial x_2} \in g^\bullet(\mathbf{x})_2 = [2.4, 6]$ the vectors $\partial f(\mathbf{x})$ and $\partial g^\bullet(\mathbf{x})$ are linearly independent, proving that $\mathbf{x}$ contains no local optimizer.

In spite of its simplicity, the first order rejection test drastically improves the convergence of the algorithm by fighting the so called cluster effect (as shown in [18] but also demonstrated by the experiments in Sect. 5).

4.3 Finding and verifying feasible points

Finding and verifying feasible points (e.g., see [13,24]) is an essential part of the branch and bound strategy since it allows to rigorously assert an upper bound $\overline{f}$ on the minimal objective function within the feasible set, resulting in the additional constraint $f(x) \leq \overline{f}$. During the

[3] Rigorously checking that an interval matrix is full rank can be done e.g. by using the interval Gauss elimination or by preconditioning the matrix using some approximation of generalized inverse of is midpoint.

resolution, this constraint will remove parts of the search space which do not contain a global solution. Initially $\overline{f}$ is set to ∞.

Verifying feasibility of an arbitrarily $\tilde{x} \in \mathbf{x}$ is done by evaluating the quantifier-free expression of the constraints using interval arithmetic.[4] For selecting an $\tilde{x} \in \mathbf{x}$ usually two strategies are considered: choosing a suitable x from $\mathbf{x}$ by some heuristics (a common choice is using random vectors or $\mathrm{mid}(\mathbf{x})$) or using a local solver to look for a local minimizer of the problem within the box $\mathbf{x}$. The local solver may use the quantifier-free expression of the constraints, although it is non smooth so standard local solvers will face difficulties when the local minimizer is around point where the quantifier-free expression is non-smooth.

4.4 Branching

In a branch and bound scheme, the search space of a problem with n variables, is usually represented by a single n-dimensional box $\mathbf{x}$. This box is then successively subdivided (split) into smaller n-dimensional boxes, in general by choosing a variable index k (e.g., one with $\mathrm{rad}(\mathbf{x}_k) \geq \mathrm{rad}(\mathbf{x}_i)$ for all $1 \leq i \leq n$) and then splitting the box at the midpoint of the kth component into two sub-boxes. More sophisticated strategies may have a more complicated variable selection method, select more than one components, or subdivide each box into several sub-boxes (see e.g., [11]).

Here we consider an arbitrary splitting strategy, e.g., split the largest domain at its midpoint. The strategy is then modified in order to fully benefit of the parameters instantiations and the first order rejection tests: Since we require the sign of the variables become fixed as soon as possible, if some variable domains contain zero, we split the largest one, say $\mathbf{x}_i$, at zero. While this works for constraint propagation (since it uses the condition $\pm\mathbf{x}_i \leq 0$ for instantiating parameters), it does not for the first order rejection test, since they require the stronger condition $\pm\mathbf{x}_i < 0$. Therefore, if first order rejection tests are to be applied, the box $\mathbf{x}_i$ is not only bisected at zero, but split into three subintervals $[\underline{x}_i, -\epsilon]$, $[-\epsilon, \epsilon]$ and $[\epsilon, \overline{x}_i]$, where ϵ is the smallest strictly positive floating point number.

Example 6 Consider again the QQC from Example 4 (page 16), and suppose we wish to minimize $f(x) := x_2$, whose gradient within the initial domain $\mathbf{x} := ([-1, 1], [-2, 0])^T$ is $\nabla f = (0, 1)^T$. If we split $\mathbf{x}$ into the two subdomains $\mathbf{x}' := ([-2, 0], [-2, 0])^T$ and $\mathbf{x}'' := ([0, 2], [-2, 0])^T$, then the enclosure of the generalized gradient of the constraint can be computed using (37) and we obtain:

$$\partial g^\bullet(\mathbf{x}') = ([-9, 3], [-5, 1])^T \tag{48}$$

$$\partial g^\bullet(\mathbf{x}'') = ([-1, 11], [-7, -1])^T. \tag{49}$$

Sadly both enclosures contain vectors $y_\lambda := -\frac{1}{\lambda}\nabla f$ for some $\lambda > 0$, therefore none of them can be rejected using the first order optimality conditions.

On the other hand if $\mathbf{x}$ is split into $\mathbf{x}' = ([-2, -\epsilon], [-2, 0])^T$, $\mathbf{x}'' = ([\epsilon, 2], [-2, 0])^T$ and $\mathbf{x}''' = ([-\epsilon, \epsilon], [-2, 0])^T$, the first component of $\mathbf{x}'$ and $\mathbf{x}''$ does not contain 0, an therefore the generalized gradient enclosure is much tighter:

$$\partial g^\bullet(\mathbf{x}') = ([-5, -0.5], [-5, -2])^T \tag{50}$$

$$\partial g^\bullet(\mathbf{x}'') = ([1.5, 7], [-7, -2])^T. \tag{51}$$

$$\partial g^\bullet(\mathbf{x}''') = ([-1, 3], [-4, -2])^T, \tag{52}$$

[4] Necessary for rigorously handling rounding errors.

and both $\mathbf{x}'$ and $\mathbf{x}''$ are rejected by the first order rejection test from [18]. The radius of the first component of $\mathbf{x}'''$ is ϵ, and since ϵ is the smallest strictly positive floating point number this component will never split again. Conclusively for the global optimizer x^* we find that $x_1^* \in [-\epsilon, \epsilon]$.

5 Experiments

In this section the branch and bound method proposed in this paper is tested on academic, quantified quadratic, optimization problems. The test compares the performance of three different methods based on the generic branch and bound algorithm (Algorithm 1):

- `PINST`: The method proposed in this paper, where in the pruning step the parameters are instantiated depending on the variable signs as described in Proposition 1, and in branching step the components of the processed boxes containing zero are split into three subintervals: $[\underline{x}_i, -\epsilon]$, $[-\epsilon, \epsilon]$ and $[\epsilon, \overline{x}_i]$ (cf. Sect. 4.4).
- `PELIM`: The standard branch and bound algorithm applied to the quantifier free expression given in Corollary 1, and the arising quantifier free expression is solved using standard pruning (including first order test) and midpoint splitting strategy.
- `PINST_NFO`: The same as `PINST` but the first order optimality condition tests are not used.

All three methods use depth-first, maximum-width-first branching. During the solution process the local solver IPOPT [39] is used to find local optima which are then (if successfully verified) used to improve the upper bound on the minimal objective function by finding feasible points with good objective evaluation (cf. Sect. 4.3). The test were performed on a `Lenovo W530` Laptop with `Intel Core i7-3610QM CPU, 2.30 GHz` and `24GB RAM` under Windows 7, using a custom Java interval package similar to the implementation of IAMath: http://interval.sourceforge.net/interval/java/ia_math/README.html.

5.1 Test problems

The test problems (53) and (55) described below are n-dimensional, having a quadratic quantified inequality constraint and a linear, unquantified objective function. They are suitable for testing our algorithms for the following reasons:

- They can be tested for any dimension $n \geq 2$.
- They contain $n(n+1)/4$ universally and $n(n+1)/4$ existentially quantified parameters.
- The global minimizer x^* lies on the boundary of the feasible domain (and therefore the constraint is active at x^*), making the problem typical in constrained optimization.[5]
- The global minimizer lies on the smooth portion of the boundary. This is important since we have no access to a non-smooth, constrained local solver, and IPOPT has serious performance issues if a local optimum lies near to a non-smooth point (cf. Sect. 4.3). On the other hand, as shown in [18], minimizers lying on a non-smooth point of the constraint boundary seem to reduce the cluster effect, hence simplifying the overall solving process.

[5] constrained optimization problems usually have at least one active constraint at a minimizer.

5.1.1 Convex problem

$$\begin{array}{ll} \min & f(x) := \sum_{i=1}^{n} x_i \\ \text{s.t.} & x_i \in [-2, 3], \\ & Q_{ij} a_{ij} \in [0, 1], \ \sum_{i=1}^{n} (a_{ii} + 2n) x_i^2 + \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{ij} x_i x_j \le 2n, \end{array} \tag{53}$$

where

$$Q_{ij} = \begin{cases} \forall & \text{if } (n-1)(i-1) + j \text{ is odd,} \\ \exists & \text{otherwise.} \end{cases} \tag{54}$$

5.1.2 Non-convex problem

$$\begin{array}{ll} \min & f(x) := \sum_{i=1}^{n} x_i \\ \text{s.t.} & x_i \in [-2, 3], \\ & Q_{ij} a_{ij} \in [-2n, 2n], \ \sum_{i=1}^{n} \sum_{j=i}^{n} a_{ij} x_i x_j \le 2n, \end{array} \tag{55}$$

where

$$Q_{ij} = \begin{cases} \forall & \text{if } i + j \text{ is even,} \\ \exists & \text{otherwise.} \end{cases} \tag{56}$$

5.2 Test settings

The generic branch and bound algorithm (Algorithm 1) for `PINST`, `PELIM` and `PINST_NFO` is tuned as follows. For given positive tolerances $0 < \tau \ll 1$ and $0 < \epsilon \ll 1$, Algorithm 1 returns the interval enclosure $[\underline{f}, \overline{f}]$ of the global minimum with $\mathrm{d_r}(\overline{f}, \underline{f}) \le \tau$, as well as a set of boxes $\mathcal{E}$, which contains the global minimizer. In our test we have chosen $\tau = 10^{-5}$ and $\epsilon = 10^{-7}$.

5.3 Test results

The test results are summarized in Table 1, and the log scaled plots of the computation times and pruning steps taken are shown in Fig. 3. The entries in the tables are for each dimension (column `n`): the number of pruning steps taken (in the column `steps`), the computation times in seconds (in the column `time`), the number of boxes removed by the first order optimality test (in the column `RFO`) and the enclosure of the global minimum (in the column `fRange`, with e.g., $-1.265\_$ denoting the interval $[-1.2659, -1.2650]$). The enclosures of the global minima are not given separately for each method since they were the same (in the shown accuracy) for each of them (hinting that the implementations were correct).

Comparison of the test results for the methods `PINST` and `PELIM` show that (although already quite useful) solving the parameter free form is less preferable than using the parametric form and instantiating the parameter bounds during the solution process. Not only `PINST` takes less pruning steps, but also the time spent in single pruning step[6] is approxi-

[6] Computed by averaging over the times divided by the number of pruning steps.

Table 1 Test results summary for Problem (53) (upper part, $n = 2, \ldots, 11$) and for Problem (55) (lower part, $n = 2, \ldots, 6$)

n	fRange	PINST			PELIM			PINST-NFO	
		Steps	Time	RFO	Steps	Time	RFO	Steps	Time
2	−1.265_	22	0.26	18	22	0.29	17	109	0.32
3	−1.607_	85	0.43	76	84	0.53	72	17,440	8.1
4	−1.861_	207	0.45	182	237	1.6	200	2,501,762	1431
5	−2.078_	406	0.81	351	715	3.9	592	-	-
6	−2.282_	1440	2.8	1236	2035	11	1528	-	-
7	−2.476_	3227	7.5	2538	5812	31	4393	-	-
8	−2.649_	17,527	48	14,781	19,248	114	14,459	-	-
9	−2.808_	36,617	134	28,784	68,794	479	52,089	-	-
10	−2.962_	177,800	809	152,043	267,377	2237	211,250	-	-
11	−3.112_	491,579	2601	422,595	874,324	8547	833,762	-	-
2	−2.00_	43	0.07	12	46	0.08	14	337	0.61
3	−2.34_	616	0.64	345	796	0.7	440	744,101	284
4	−2.82_	11,424	8.86	6233	19,525	10.71	10,313	-	-
5	−3.10_	224,235	198.7	124,916	448,829	344.3	210,129	-	-
6	−3.46_	5,703,963	4082	2,554,711	17,804,016	12,246	6,171,161	-	-

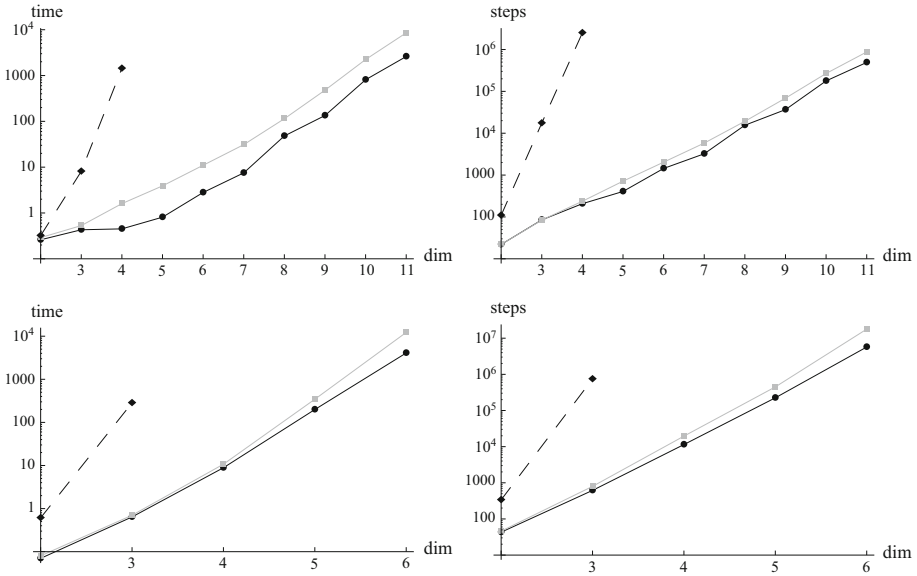

Fig. 3 Test results for problem (53) (*upper two figures*) and Problem (55) (*lower two figures*). Log scaled plot of the computation times (*left figures*) and pruning steps taken (*right figures*) plotted against the problem dimensions. *Black* is PINST (*dashed* is PINST-NFO), and *gray* is PELIM

mately two times lower than in PELIM. This behavior was expected since in the parametric form (if all the parameter bounds are already instantiated) the constraint remain quadratic and therefore the pruning steps made use of the dedicated quadratic constraint propagation

method. For both methods we see that due to the cluster effect[7] there were a large number of boxes which needed to be processed, even when the first order optimality test removed a great amount of them.

The test results for `PINST_NFO` applied to Problem (53) and Problem (55) (for dimensions 2–4 and dimensions 2–3 respectively, higher dimensions took too long to test) demonstrate that the first order optimality test (or another method able reduce the cluster effect) is essential for efficiently solving problems in higher dimensions. Since the first order optimality test need the three subintervals splitting strategy from Sect. 4.4, the overhead arising from this splitting method is justified by the test results.

6 Conclusion

In this paper quantified quadratic constraints have been investigated. For these constraints general quantifier free expressions, as well as expressions of their generalized gradients have been proposed. Since the quantifier free expressions loose the nice quadratic structure of the initial problem, a partial quantifier elimination has been proposed, allowing to keep the useful properties of the quadratic form.

The partial quantifier elimination has been integrated into a branch and bound algorithm, based on numerical constraint programming for domain reduction. The bisection strategy of the branch and bound algorithm has also been tuned to support the partial quantifier elimination. Using the generalized gradients, first order rejection tests have been integrated into the new solution method.

Experiments on random academic benchmarks have shown that the new method is approximately three times faster than simply solving the quantifier free expression. Furthermore, the first order rejection tests have greatly enhanced the resolution process.

References

1. Adjiman, C.S., Androulakis, I.P., Maranas, C.D., Floudas, C.A.: A global optimization method αBB for process design. Comput. Chem. Eng. **20**, 419–424 (1996)
2. Alefeld, G., Herzberger, J.: Introduction to Interval Computations. Computer Science and Applied Mathematics (1974)
3. Ben-Tal, A., El Ghaoui, L., Nemirovski, A.: Robust Optimization, Princeton Series in Applied Mathematics. Princeton University Press, Princeton (2009)
4. Ben-Tal, A., Nemirovski, A.: Robust solutions of uncertain linear programs. Oper. Res. Lett. **25**(1), 1–13 (1999)
5. Ben-Tal, A., Nemirovski, A.: Robust solutions of linear programming problems contaminated with uncertain data. Math. Program. **88**(3), 411–424 (2000)
6. Ben-Tal, A., Nemirovski, A.: On tractable approximations of uncertain linear matrix inequalities affected by interval uncertainty. SIAM J. Optim. **12**(3), 811–833 (2002)
7. Benhamou, F., Goualard, F., Granvilliers, L., Puget, J.F.: Revising hull and box consistency. In: International Conference on Logic Programming, pp. 230–244 (1999)
8. Blankenship, J.W., Falk, J.E.: Infinitely constrained optimization problems. J. Optim. Theory Appl. **19**(2), 261–281 (1976)
9. Chabert, G., Jaulin, L.: Hull consistency under monotonicity. In: Principle and practices of constraint programming—CP2009, pp. 188–195 (2009)
10. Clarke, F.H.: Optimization and Nonsmooth Analysis. Society for Industrial and Applied Mathematics, Philadephia (1990)

[7] The cluster effect is a usual phenomenon when solving global optimization problems, see e.g., [14].

11. Csendes, T., Ratz, D.: Subdivision direction selection in interval methods for global optimization. SIAM J. Numer. Anal. **34**, 922–938 (1997)
12. Domes, F., Neumaier, A.: Quadratic constraint propagation. Constraints, pp. 404–429 (2010)
13. Domes, F., Neumaier, A.: Rigorous verification of feasibility. J. Global Optim. **61**(2), 255–278 (2015)
14. Du, K., Kearfott, R.B.: The cluster problem in multivariate global optimization. J. Global Optim. **5**, 253–265 (1994)
15. Ghaoui, L.E., Lebret, H.: Robust solutions to least-squares problems with uncertain data. SIAM J. Matrix Anal. Appl. **18**(4), 1035–1064 (1997)
16. Ghaoui, L.E., Oustry, F., Lebret, H.: Robust solutions to uncertain semidefinite programs. SIAM J. Optim. **9**(1), 33–52 (1998)
17. Goldberg, D.: What every computer scientist should know about floating-point arithmetic. Comput. Surv. **23**(1), 5–48 (1991)
18. Goldsztejn, A., Domes, F., Chevalier, B.: First order rejection tests for multiple-objective optimization. J. Global Optim. **58**(4), 653–672 (2013)
19. Goualard, F.: GAOL 3.1.1: Not Just Another Interval Arithmetic Library, 4.0 edn. Laboratoire d'Informatique de Nantes-Atlantique, Nantes (2006)
20. Hansen, E.: Global Optimization Using Interval Analysis, 2nd edn. Marcel Dekker, New York (1992)
21. Jaulin, L., Kieffer, M., Didrit, O., Walter, E.: Applied Interval Analysis with Examples in Parameter and State Estimation, Robust Control and Robotics. Springer, Berlin (2001)
22. Jeyakumar, V., Li, G.: Robust solutions of quadratic optimization over single quadratic constraint under interval uncertainty. J. Global Optim. **55**(2), 209–226 (2013)
23. Kearfott, R., Nakao, M., Neumaier, A., Rump, S., Shary, S., van Hentenryck, P.: Standardized notation in interval analysis. In: Proceedings of XIII Baikal International School-seminar "Optimization methods and their applications" (Vol. 4, pp. 106–113). Irkutsk: Institute of Energy Systems, Baikal (2005)
24. Kearfott, R.B.: On proving existence of feasible points in equality constrained optimization problems. Math. Program. **83**(1–3), 89–100 (1995)
25. Kearfott, R.B.: Interval Computations: Introduction, Uses, and Resources. Euromath Bull. **2**(1), 95–112 (1996)
26. Kearfott, R.B.: Interval computations, rigour and non-rigour in deterministic continuous global optimization. Optim. Methods Softw. **26**(2), 259–279 (2011)
27. Knueppel, O.: PROFIL/BIAS: a fast interval library. Computing **53**(3–4), 277–287 (1994)
28. Lebbah, Y., Michel, C., Rueher, M.: A rigorous global filtering algorithm for quadratic constraints. Constraints **10**, 47–65 (2005)
29. Lhomme, O.: Consistency techniques for numeric CSPS. IJCAI **1**, 232–238 (1993)
30. Li, M., Gabriel, S., Shim, Y., Azarm, S.: Interval uncertainty-based robust optimization for convex and non-convex quadratic programs with applications in network infrastructure planning. Netw. Spat. Econ. **11**, 159–191 (2011)
31. Mitsos, A.: Global optimization of semi-infinite programs via restriction of the right-hand side. Optimization **60**(10–11), 1291–1308 (2011)
32. Mitsos, A., Tsoukalas, A.: Global optimization of generalized semi-infinite programs via restriction of the right hand side. J. Global Optim. **61**(1), 1–17 (2015)
33. Moore, R.: Interval Analysis. Prentice-Hall, Upper Saddle River (1966)
34. Neumaier, A.: Interval Methods for Systems of Equations. Cambridge Univ. Press, Cambridge (1990)
35. Rump, S.: INTLAB—INTerval LABoratory. In: Csendes, T. (ed.) Developments in Reliable Computing, pp. 77–104. Kluwer Academic Publishers, Dordrecht (1999)
36. Schichl, H., Neumaier, A.: Exclusion regions for systems of equations. SIAM J. Numer. Anal. **42**(1), 383–408 (2004)
37. Sherali, H., Adams, W.: A Reformulation–Linearization Technique for Solving Discrete and Continuous Nonconvex Problems. Kluwer Academic Publ., Dordrecht (1999)
38. Tsoukalas, A., Rustem, B.: A feasible point adaptation of the blankenship and falk algorithm for semi-infinite programming. Optim. Lett. **5**(4), 705–716 (2011)
39. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006)
40. Wolfram Research Inc.: Mathematica 7.0. Wolfram Research Inc., Champaign (2008)