Constraint propagation on quadratic constraints

Ferenc Domes, Arnold Neumaier
Faculty of Mathematics, University of Vienna
Nordbergstrasse 15, A-1090 Vienna, Austria

April 5, 2010

Abstract. This paper considers constraint propagation methods for continuous constraint satisfaction problems consisting of linear and quadratic constraints. All methods can be applied after suitable preprocessing to arbitrary algebraic constraints.

The basic new techniques consist in eliminating bilinear entries from a quadratic constraint, and solving the resulting separable quadratic constraints by means of a sequence of univariate quadratic problems. Care is taken to ensure that all methods correctly account for rounding errors in the computations.

Various tests and examples illustrate the advantage of the presented method.

Keywords. Constraint propagation, constraint programming, continuous constraints, quadratic constraint satisfaction problems, rounding error control, verified computation, quadratic programming, constrained optimization.

1 Introduction

Context. This paper contributes some new solution techniques for continuous constraint satisfaction problems. A constraint satisfaction problem is the task of finding one or all points satisfying a given family of equations and/or inequalities, called constraints. Many real word problems are continuous constraint satisfaction problems, often high dimensional ones. Typical applications include robotics Grandon et al. [19], Merlet [36], localization and map building Jaulin [28], Jaulin et al. [29], biomedicine Cruz & Barahona [13], or the protein folding problem Krippahl & Barahona [32].

Solving constrained global optimization problems is typically reduced to solving a sequence of constraint satisfaction problems, each obtained by adding a constraint $f(x) \leq f_{\text{best}}$ to the original constraints, where f is the objective function and f_{best} the function value of the best feasible point found. Thus all techniques for solving constraint satisfaction problems have immediate impact on global optimization (see Neumaier [39]).

Constraint satisfaction problems are solved in practice by a combination of a variety of techniques, almost always involving as key components constraint propagation combined with either some form of stochastic search or a branch and prune scheme for a complete search. These techniques are often complemented by filtering or reduction techniques based on techniques borrowed from optimization, such as convex relaxations Lebbah et al. [34]. For filtering, relaxation, branching and other techniques also see Jaulin [27], Sahinidis & Tawarmalani [41], [34].

Filtering techniques that tighten a box – the Cartesian product of intervals defined by the bounds on the variables – are called *constraint propagation* if they are based on a sequence of steps, each using a single constraint only. *Forward propagation* uses the bound constraints to improve the bounds on the general constraints; *backward propagation* uses the bounds on the general constraints to improve the bounds on the variables. In order to avoid a loss of feasible points, constraint propagation methods are usually implemented with rigorous error control, taking care that all reductions are valid even though the calculations are done with floating-point arithmetic only.

In practice, constraint propagation repeats the reduction of a box by means of a suitably chosen constraint, navigating through the network of constraints connected by the variables, until no further significant reduction takes place. In particular, if the initial search box is unbounded but the feasible domain is bounded, constraint propagation methods may be able to find finite bounds on all variables. Since many methods require finite box constraints, this makes constraint propagation a valuable preprocessing tool.

In a stochastic search procedure, constraint propagation on the initial box may result in a much smaller search domain. In a branch and prune procedure, where a tree of subboxes is generated, constraint propagation may result in a quick elimination of subboxes, or a significant reduction before more complex reduction techniques are applied. This shows that constraint propagation has a wide range of applicability, and is a very useful optimization technique.

Prior work. A number of software packages for solving constraint satisfaction problems make extensive use of constraint propagation. The `Numerica` software HENTENRYCK [23], HENTENRYCK et al. [25] uses branch and prune methods and interval constraint programming to solve constraint satisfaction problems. The `ICOS` solver by LEBBAH [33] is a software package for solving nonlinear and continuous constraints, based on constraint programming and interval analysis techniques. `Realpaver` by GRANVILLIERS & BENHAMOU [20] combines interval methods, with constraint satisfaction techniques to solve systems given by sets of equations or inequality constraints over integer and real variables. CEBERIO & GRANVILLIERS [10] solves nonlinear systems by using interval extension and constraint inversion.

The price winning solver `Baron` by SAHINIDIS & TAWARMALANI [42] also uses constraint propagation techniques. Initiated by the development of interval analysis on DAGs (Directed Acyclic Graphs) by SCHICHL & NEUMAIER [45], advanced constraint propagation techniques for solving numerical constraint satisfaction problems have been given in VU et al. [47, 48], which is an efficient implementation of basic constraint propagation algorithms for individual operations. It is included in the global optimization software platform COCONUT Environment [43, 44].

Historically, constraint propagation was pioneered in constraint logic, first for discrete constraints by CLEARY [12], later for continuous constraints (OLDER & VELLINO [40], see also [3, 7, 8, 11, 14, 21, 23–26, 31]), but has also forerunners in presolve techniques in mathematical programming (ANDERSON & ANDERSON [1], LODWICK [35]). They can be modeled by narrowing BENHAMOU [4] or chaotic iterations APT [2], i.e., sequences of application of contracting and monotonic functions on domains. The level of work involved and quality obtained in constraint propagation methods may be characterized by local consistency notions; see BENHAMOU et al. [5, 6], JERMANN et al. [30]. An in-depth treatment of continuous constraint propagation from the point of view of constraint programming can be found in the COCONUT report (BLIEK et al. [9]). The global optimization survey of NEUMAIER [39] also discusses continuous constraint propagation without the need to decompose the constraints into single operations.

Contents. In this paper, we consider constraint propagation methods for continuous constraint satisfaction problems consisting of linear and quadratic constraints. Care is taken to ensure that all methods correctly account for rounding errors in the computations. We only present techniques for improving the inferences from single constraints. These techniques can be easily combined with our new constraint propagation method. In general constraint propagation on single constraints only is not efficient enough for solving constraint satisfaction problems. As is well known (see, e.g., [34]), future significant improvements can be obtained by using the information from several constraints simultaneously. This can be done in various ways, e.g., by using linear relaxations ([33]), probing ([46]) or other, new methods in our GloptLab environment ([16]) where all constraint propagation methods introduced in this paper are implemented, too.

All our methods can be applied after suitable preprocessing to arbitrary algebraic constraints. We can always transform a polynomial constraint to a collection of quadratic constraints by introducing explicit intermediate variables. The same holds for constraints involving roots, provided that we also add nonnegativity constraints to the intermediate variables representing the roots. Rewriting an algebraic constraint satisfaction problem as an equivalent problem with linear and quadratic constraints increases the number of variables and can result in loss of structural information which is used by some constraint propagation techniques but makes possible to apply the methods discussed in this paper. Of course, all techniques can be applied to the subset of quadratic (or algebraic) constraints in an arbitrary constraint satisfaction problem. However, in practice care should be taken that the dimension of the transformed problem does not exceed the reasonable maximum.

We represent simple bounds as box constraint $x \in \mathbf{x}$. A *box* (or interval vector) is a Cartesian product

$$\mathbf{x} = [\underline{x}, \overline{x}] := (\mathbf{x}_1, \dots, \mathbf{x}_n)^T$$

of (bounded or unbounded) closed, real intervals $\mathbf{x}_i := [\underline{x}_i, \overline{x}_i]$. Thus the condition $x \in \mathbf{x}$ is equivalent to the collection of simple bounds

$$\underline{x}_i \le x_i \le \overline{x}_i \quad (i = 1, \dots, n),$$

or, with inequalities on vectors and matrices interpreted component-wise, to the two-sided vector inequality $\underline{x} \le x \le \overline{x}$. Apart from two-sided constraints, this includes with $\mathbf{x}_i = [a, a]$ variables x_i fixed at a particular value $x_i = a$, with $\mathbf{x}_i = [a, \infty]$ lower bounds $x_i \ge a$, with $\mathbf{x}_i = [-\infty, a]$ upper bounds $x_i \le a$, and with $\mathbf{x}_i = [-\infty, \infty]$ free variables. A bound is *large*, if its absolute value is larger than a configurable constant (whose default value is 10^6). Decisions are based on 'if a bound is large' rather than on 'if a bound is infinite'.

In Section 2 we derive rigorous bounds for univariate quadratic expressions, relevant for forward propagation, while in Section 3 we find bounds on the arguments in a constraint, relevant for backward propagation. The propagation of separable quadratic constraints (containing no bilinear entries) is discussed in Section 4, and in Section 5, we give an effective method to bound and eliminate bilinear entries from a constraint. This allows the reduction of the nonseparable constraints to separable ones, leading in Section 6 to a constraint propagation method for quadratic constraint satisfaction problems.

2 Bounds for univariate quadratic expressions

For use in forward propagation, we derive rigorous bounds for univariate quadratic expressions. We analyze the possible extrema of the expression inside a given interval, and derive the general solution of the problem.

Example. 2.1 *Let $f(x)$ denote an univariate quadratic expression, with*

$$f(x) = x^2 - 2x, \quad x \in [-7, 5] := \mathbf{x}. \tag{1}$$

We look for the best interval $\mathbf{f}$ *such that for all* $x \in \mathbf{x}$*,* $f(x) \in \mathbf{f}$ *holds: we find that the minimum of* f *is attained at* $x = 1$ *which is inside of the interval* $\mathbf{x}$*. The maximum of* f *must be attained at the boundary of* $\mathbf{x}$*. We have* $f(1) = -1$ *and the function values on the boundary of* $\mathbf{x}$ *are* 63 *and* 15*. Therefore, we find that* $\mathbf{f} = [-1, 63]$*.*

In general, we want to find a rigorous upper bound on

$$u = \sup\{ax^2 + bx \mid x \in \mathbf{x}\}.$$

We note that $u = \max\{\underline{x}(a\underline{x} + b), \overline{x}(a\overline{x} + b)\}$, except in case that $ax^2 + bx$ attains its global maximum in the interior of $\mathbf{x}$. This is the case iff $a < 0$ and $t = -b/(2a)$ is in the interior of $\mathbf{x}$, in which case $u = b^2/(-4a)$ is attained at t.

If $\underline{x} \geq 0$, we get a rigorous upper bound in finite precision arithmetic by computing with upward rounding as follows (`xl` $= \underline{x}$, `xu` $= \overline{x}$):

Algorithm: 2.2 (Rigorous upper bound for a univariate quadratic expression)

```
roundup;
if a == 0,
  u=max(xl*b,xu*b);
else
  u=max(xl*(a*xl+b),xu*(a*xu+b));
  s=b/2; t=s/(-a);
  if t>xl,
    r=(-2*a)*xu;
    if r>b, u=max(u,s*t); end;
  end;
end;
```

With some extra analysis, it could be determined in most cases which of the three cases is the worst case; however, if the unconstrained maximum of the quadratic is very close to a bound (or to both bounds), two (or three) of the cases might apply due to uncertainty caused by rounding errors.

Finding a rigorous enclosure for the interval

$$\mathbf{c} = \{\sup\{ax^2 + bx \mid x \in \mathbf{x}\} \mid a \in \mathbf{a}, b \in \mathbf{b}\}$$

can be reduced to the above for $\underline{x} \geq 0$, using

$$\overline{c} = \sup\{\overline{a}x^2 + \overline{b}x \mid x \in \mathbf{x}\}, \quad \underline{c} = -\sup\{-\underline{a}x^2 - \underline{b}x \mid x \in \mathbf{x}\}.$$

The case $\overline{x} \leq 0$ can be reduced to this by changing the sign of x, and the general case by splitting $\mathbf{x}$ at zero if necessary.

Essentially the same analysis holds for rigorous upper bounds on

$$u = \sup\left\{ \sum_{i=1}^{n} a_i x^i \;\middle|\; x \in \mathbf{x} \right\}$$

and for rigorous enclosures of

$$\mathbf{c} = \sup\left\{ \sum_{i=1}^{n} a_i x^i \;\middle|\; x \in \mathbf{x},\ a \in \mathbf{a} \right\},$$

except that finding the interior extrema is more involved. It can be done with closed formulas for $n \leq 5$ (though already $n = 4$ is quite cumbersome and not recommended). In general, for $n > 3$ we recommend to use a root enclosure algorithm for the derivative, such as that in NEUMAIER [38].

3 Solving univariate quadratic expressions

If we have bounds on an univariate quadratic expression, we can find the range for the variable, for which the expression satisfies the given bounds. An in depth analysis of quadratic equations leads to the general solution, relevant for backward propagation.

The present approach, based on directed rounding only, provides an efficient alternative to the interval arithmetic based procedures discussed by DIMITROVA & MARKOV [15, Section 4] and later by HANSEN & WALSTER [22] (who only treat the solution of a quadratic equation with interval coefficients).

Example. 3.1 *Let $f(x)$ denote an univariate quadratic expression, with*

$$f(x) = x^2 - 2x \in [-1, 8]. \tag{2}$$

We look for the best interval $\mathbf{x}$ such that for all $x \in \mathbf{x}$, (2) holds. The inequality

$$x^2 - 2x + 1 = (x-1)^2 \geq 0,$$

arising from the lower bound, always holds, while the inequality

$$x^2 - 2x - 8 \leq 0,$$

arises from the upper bound. Therefore we find that $\mathbf{x} = [-2, 4]$. We note that, while $\mathbf{x}$ is by definition always an interval, sometimes the set of all x satisfying the constraint may be strictly smaller, containing an interior gap.

In general, we want to find the set

$$X = \{x \geq 0 \mid ax^2 + 2bx \geq c\},$$

and we proceed as follows. If $a = 0$, the constraint is in fact linear, and we have

$$X = \begin{cases} \emptyset & \text{if } b \leq 0, c > 0, \\ [0, 0.5c/b] & \text{if } b < 0, c \leq 0, \\ [0.5c/b, \infty] & \text{if } b > 0, c \geq 0, \\ [0, \infty] & \text{if } b \geq 0, c \leq 0, \end{cases}$$

which can be nested such that only two comparisons are needed in any particular case. For a rigorous enclosure in finite precision arithmetic, rounding must be downwards in the second case, and upwards in the third case.

If $a \neq 0$, the behavior is governed by the zeros of the quadratic equation $ax^2 + 2bx - c = 0$, given by

$$t_1 = \frac{-b - \sqrt{\Delta}}{a} = \frac{c}{b - \sqrt{\Delta}}, \qquad t_2 = \frac{-b + \sqrt{\Delta}}{a} = \frac{c}{b + \sqrt{\Delta}},$$

where $\Delta := b^2 + ac$. If $\Delta \geq 0$, the zeros are real, and the nonnegative zeros determine

$$X = \begin{cases} [0, \infty] \setminus \;]t_1, t_2[& \text{if } a > 0, \\ [0, \infty] \cap [t_2, t_1] & \text{if } a < 0. \end{cases}$$

Depending on the signs of a, b and c we find

$$X = \begin{cases} \emptyset & \text{if} \quad a<0, b\le 0, c>0, & \text{(case 1)} \\ [z/a, \infty] & \text{if} \quad a>0, b\le 0, c>0, & \text{(case 2)} \\ [0, -(c/z)] & \text{if} \quad a<0, b\le 0, c\le 0, & \text{(case 3)} \\ [0, -(c/z)][\cup [z/a, \infty] & \text{if} \quad a>0, b\le 0, c\le 0, & \text{(case 4)} \\ [-((-c)/z), z/(-a)] & \text{if} \quad a<0, b\ge 0, c>0, & \text{(case 5)} \\ [-((-c)/z), \infty] & \text{if} \quad a>0, b\ge 0, c>0, & \text{(case 6)} \\ [0, z/(-a)] & \text{if} \quad a<0, b\ge 0, c\le 0, & \text{(case 7)} \\ [0, \infty] & \text{if} \quad a>0, b\ge 0, c\le 0, & \text{(case 8)} \end{cases}$$

where

$$z = |b| + \sqrt{\Delta}.$$

These formulas are numerically stable, and can be nested such that only three comparisons are needed in any particular case. (There are avoidable overflow problems for huge $|b|$, which can be cured by using for huge $|b|$ instead of $\sqrt{b^2+ac}$ the formula $|b|\sqrt{1+ac/b^2}$.)

Rigorous results in the presence of rounding errors are obtained if lower bounds are rounded downwards, and upper bounds are rounded upwards. With the bracketing as given, this happens if all computations (including those of $\Delta = \sqrt{b^2+ac}$ and $z = |b| + \sqrt{\Delta}$) are done with rounding upwards if $b \ge 0$, and with rounding downwards if $b \le 0$. (However, this does *not* hold for the version guarded against overflow, where further care is needed for the directed rounding of $\sqrt{\Delta} = |b|\sqrt{1+ac/b^2}$.)

If (the exact) Δ is negative, there is no real solution, and X is empty if $c > 0$ and $[0, \infty]$ otherwise. The case when the sign of Δ cannot be determined due to rounding errors needs special consideration. In the first and last case, the conclusion holds independent of the sign of Δ, so that the latter need only be computed for cases 2–7 in the definition of X. In the cases 2, 3, 6, and 7 we have $ac \ge 0$, so that $\Delta \ge 0$ automatically. This leaves cases 4 and 5. Now it is easily checked that with the recommended rounding and, in place of cases 4 and 5,

$$X = \begin{cases} [0, -(c/z)] \cup [z/a, \infty] & \text{if } a>0,\ b\le 0,\ c\le 0,\ \Delta \ge 0, \\ [0, \infty] & \text{if } a>0,\ b\le 0,\ c\le 0,\ \Delta < 0, \\ \emptyset & \text{if } a<0,\ b\ge 0,\ c>0,\ \Delta < 0, \\ [-((-c)/z), z/(-a)] & \text{if } a<0,\ b\ge 0,\ c>0,\ \Delta \ge 0, \end{cases} \tag{3}$$

a rigorous enclosure is computed in all cases. Finding the set

$$X' = \{x \ge 0 \mid ax^2 + 2bx \in \mathbf{c} \text{ for any } a \in \mathbf{a}, b \in \mathbf{b}\}$$

can be reduced to the previous task since

$$X' = \{x \ge 0 \mid \underline{a}x^2 + 2\underline{b}x \le \overline{c}\} \cap \{x \ge 0 \mid \overline{a}x^2 + 2\overline{b}x \ge \underline{c}\}.$$

The sets

$$X'' = \{x \in \mathbf{x}_0 \mid ax^2 + 2bx \ge c\}$$

and

$$X''' = \{x \in \mathbf{x}_0 \mid ax^2 + 2bx \in \mathbf{c} \text{ for some } a \in \mathbf{a}, b \in \mathbf{b}\}$$

can be obtained by intersecting the result of the above tasks with $\mathbf{x}_0$ if $\underline{x}_0 \ge 0$, by negating x, $\mathbf{x}_0$, and $\mathbf{b}$ if $\overline{x}_0 \le 0$, and by splitting $\mathbf{x}_0$ at zero if 0 is in the interior of $\mathbf{x}_0$. By modifying the code appropriately, one can also avoid computing roots which can be seen to lie outside $\mathbf{x}_0$.

With minor changes, these formulas also apply for strict inequalities and interior enclosures. Also, it is clear that polynomial inequalities and inclusions of interval polynomials can be solved by a straightforward adaptation of the above arguments.

We end the section with MATLAB code for computing the enclosure

$$X = [\mathtt{xl}, \mathtt{xu}] \cup [\mathtt{x2l}, \mathtt{x2u}] \setminus \{\infty\}$$

according to (3).

Algorithm: 3.2 (Solving an univariate quadratic expression)

```
xl=0;xu=inf;
x2l=inf;x2u=inf;
if b>=0,
  roundup;
  if c>0,
    % case b>=0, c>0
    Delta=b^2+a*c;
    if Delta<0,
      xl=inf;
    elseif a==0 & b==0,
      xl=inf;
    else
      z=b+sqrt(Delta);
      xl=-((-c)/z);
      if a<0, xu=z/(-a); end;
    end
  else
    % case b>=0, c<=0
    if a<0,
      Delta=b^2+a*c;
      z=b+sqrt(Delta);
      xu=z/(-a);
    end;
  end;
else
  rounddn;
  if c>0,
    % case b<0, c>0
    if a>0,
      Delta=b^2+a*c;
      z=-b+sqrt(Delta);
      xl=z/a;
    else
      xl=inf;
    end
  else
    % case b<0, c<=0
    Delta=b^2+a*c;
    if Delta>=0,
      z=-b+sqrt(Delta);
      xu=-(c/z);
      if a>0, x2l=z/a; end;
    end;
  end;
end;
```

4 Propagating separable quadratic constraints

We now combine the results of the previous two sections.

Example. 4.1 *Let $f(x)$ denote an univariate quadratic expression, with*

$$f(x) := x^2 - 2x \in [-10, 8], \quad x \in [-7, 5]. \tag{4}$$

Example 2.1 produced from the bound on x the new bound $f(x) \in [-1, 63]$, hence $f(x) \in [-1, 8]$. Example 3.1 produced from these bounds on f the new bounds $x \in [-2, 4]$. Thus we end up with the new problem

$$f(x) := x^2 - 2x \in [-1, 8], \quad x \in [-2, 4]$$

where the bounds on f and x are tighter than in the original problem (4).

This combination of forward and backward propagation for a univariate quadratic expression can be extended without difficulties to a method of constraint propagation for separable quadratic constraints in several variables,

$$\sum_{k=1}^{n} p_k(x_k) \geq c, \quad x \in \mathbf{x}, \tag{5}$$

where each term

$$p_k(x_k) := a_k x_k^2 + b_k x_k \tag{6}$$

depends on a single variable x_k and may have uncertain coefficients,

$$a_k \in \mathbf{a}_k, \quad b_k \in \mathbf{b}_k.$$

Example. 4.2 *We demonstrate separable quadratic constraint propagation on the constraint*

$$-x_1^2 + 2x_1 - x_2 \geq -8 \quad \textit{with } x_1 \in [-7,5],\ x_2 \in [0,\infty]$$

step by step (see Figure 1):

• *We find that for* $x_1 \in [-7,5]$, $x_2 \in [0,\infty]$, $-x_1^2 + 2x_1 \in [-63,1]$ *and* $-x_2 \in [-\infty,0]$ *holds.*

• *Therefore, we have* $-x_1^2 + 2x_1 - x_2 \leq 1$.

• *From* $0 \geq -x_2$, *the inequality* $-x_1^2 + 2x_1 \geq -8 - 0 = -8$ *follows.*

• *Since* $1 \geq -x_1^2 + 2x_1$, *the inequality* $-x_2 \geq -8 - 1 = -9$ *holds.*

• *Then from* $-x_1^2 + 2x_1 \geq -8$, $x_1 \in [-2,4]$, *and from* $-x_2 \geq -9$, $x_2 \in [-\infty,9]$ *follows.*

• *Finally, we cut the bounds on the variables with the original bounds, and obtain*

$$-x_1^2 + 2x_1 - x_2 \in [-8,1] \quad \textit{with } x_1 \in [-2,4],\ x_2 \in [0,9].$$

For $x_k \in \mathbf{x}_k$ we denote the enclosure of the quadratic univariate term $p_k(x_k)$ by

$$p_k(x_k) \in \mathbf{p}_k = [\underline{p_k}, \overline{p_k}]. \tag{7}$$

To find the $\mathbf{p}_k$, if $0 \in \mathbf{x}_k$, we split $\mathbf{x}_k$ at zero into a positive part $\mathbf{x}_k^p$ and a negative part $\mathbf{x}_k^n$, with $\underline{x}_k^p \geq 0$, $\underline{x}_k^n \geq 0$, $\mathbf{x}_k^p \cap -\mathbf{x}_k^n = \{0\}$ and $\mathbf{x}_k^p \cup -\mathbf{x}_k^n = \mathbf{x}_k$. If $\underline{x}_k \geq 0$ then we set $\mathbf{x}_k^p := \mathbf{x}_k$ and $\mathbf{x}_k^n := \emptyset$, while if $\underline{x}_k \geq 0$ then we set $\mathbf{x}_k^p := \emptyset$ and $\mathbf{x}_k^n := -\mathbf{x}_k$. Then we define the bounds

$$\begin{array}{ll} \overline{c}_k^p := \sup\{\overline{a}x_k^2 + \overline{b}x_k \mid x_k \in \mathbf{x}_k^p\}, & \underline{c}_k^p := -\sup\{-\underline{a}x_k^2 - \underline{b}x_k \mid x_k \in \mathbf{x}_k^p\}, \\ \overline{c}_k^n := \sup\{\overline{a}x_k^2 - \overline{b}x_k \mid x_k \in \mathbf{x}_k^n\}, & \underline{c}_k^n := -\sup\{-\underline{a}x_k^2 + \underline{b}x_k \mid x_k \in \mathbf{x}_k^n\}. \end{array} \tag{8}$$

Each bound $\overline{c}_k^p$, $\underline{c}_k^p$, $\overline{c}_k^n$, $\underline{c}_k^n$ in (8) can be found by applying the results of Section 2 to them separately. Then the enclosure of $p_k(x_k)$ is

$$\begin{array}{lll} \mathbf{p}_k = [\,\min(\underline{c}_k^p, \underline{c}_k^n), \max(\overline{c}_k^p, \overline{c}_k^n)] & \text{if} & 0 \in \mathbf{x}_k, \\ \mathbf{p}_k = \mathbf{c}_k^p & \text{if} & \underline{x}_k \geq 0, \\ \mathbf{p}_k = \mathbf{c}_k^n & \text{if} & \overline{x}_k < 0. \end{array}$$

Then we sum the found intervals and obtain

$$\sum_{k=1}^{n} p_k(x_k) \in \mathbf{e} \text{ with } \mathbf{e} := \sum_{k=1}^{n} \mathbf{p}_k = \Big[\sum_{k=1}^{n} \underline{p}_k, \sum_{k=1}^{n} \overline{p}_k\Big]. \tag{9}$$

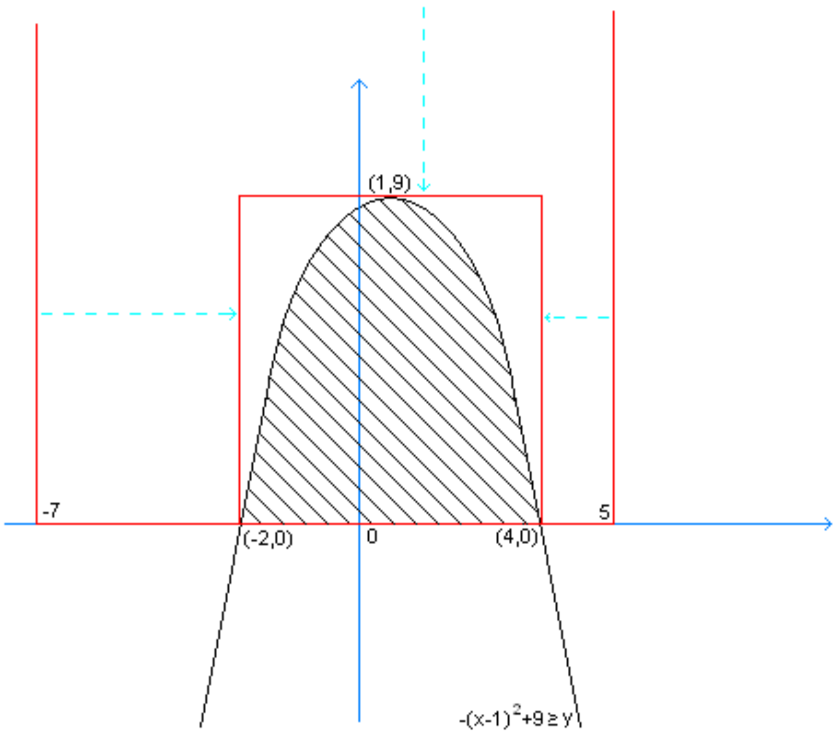

Figure 1: Improving the bound constraints in Example 4.2.

We can then use the bounds $\mathbf{p}_k$ and $\mathbf{e}$ to check the consistency of the constraint and obtain a new bound for it (forward propagation), and to get better box constraints (backward propagation).

Forward propagation. By (5) and (9), we have

$$\overline{e} \geq \sum_{k=1}^{n} p_k(x_k) \geq c. \tag{10}$$

Therefore, if (10) does not hold, the constraint is inconsistent. Following this, we pose the *inconsistency condition*:

$$\text{If } \overline{e} - c < 0 \text{ then the constraint (5) is inconsistent.} \tag{11}$$

If the constraint is consistent, by (5) and (9) both

$$\sum_{k=1}^{n} p_k(x_k) \geq c \text{ and } \sum_{k=1}^{n} p_k(x_k) \geq \underline{e}$$

are satisfied, giving us the combined lower bound

$$\sum_{k=1}^{n} p_k(x_k) \geq c' := \max(c, \underline{e}). \tag{12}$$

Backward propagation. For all $i \in \{1, \ldots, n\}$ and all $k \neq i$, by (7) and (12) we obtain

$$\sum_{k=1,\ k\neq i}^{n} \overline{p}_k + p_i(x_i) \geq \sum_{k=1,\ k\neq i}^{n} p_k(x_k) + p_i(x_i) \geq c'.$$

Bringing the upper bounds on the $p_k(x_i)$ to the left hand side and by (6) we get

$$a_i x_i^2 + b_i x_i \geq -\gamma_i, \quad \text{where } \gamma_i := \sum_{k\neq i} \overline{p}_k - c'. \tag{13}$$

The arrangement of the operations is such that upward rounding still gives correct results. Since the above approximation must be done for each univariate term in the constraint, time can be saved when $n > 2$ by avoiding unnecessary work in the summations. The paper by Dallwig et al. [14] proposes to remove the summations completely, using instead the identity

$$\gamma_i = \overline{e} - c' - \overline{p}_i. \tag{14}$$

Again, the arrangement of the operations is such that upward rounding still gives correct results. While fast, this identity must be used with caution: If $\overline{p}_i$ is infinite, $\gamma_i = c' - \infty + \infty$ is undefined. And if $|\overline{p}_i|$ is very large, cancellation (together with the always necessary directed rounding) may lead to unnecessarily pessimistic bounds. Below we give some examples which demonstrate this behavior. To eliminate these problems, we recommend the use of the formulas

$$\gamma_i = \begin{cases} \gamma' + p_{min} & \text{if} \quad p_i = p_{min} = \infty, \\ \gamma' + p_{min} - e & \text{if} \quad d + e > 0, \\ \gamma' + p_{max} - d & \text{if} \quad d + e \leq 0, \end{cases} \tag{15}$$

where $\underline{i}, \overline{i}$ are distinct indices with $p_{\underline{i}} = p_{min}, p_{\overline{i}} = p_{max}$,

$$\gamma' := \sum_{k \neq \underline{i}, \overline{i}} \overline{p}_k - c',$$

with nonnegative numbers $d := \overline{p}_i - p_{min}$ and $e := \overline{p}_i - p_{max}$. Again, the arrangement of the operations is such that upward rounding still gives correct results.

Remark. 4.3 *Alternatively we could rewrite* (6) *as*

$$p_k(x_k) = a_k(x_k + b_k/2a_k)^2 - b_k^2/4a_k$$

and use interval arithmetic to enclose ranges (forward propagation) and bounds on x (backward propagation) by using this form. However this would only work if $0 \notin \mathbf{a}_k$ *and even in this case it yields non-optimal bounds if* $\mathbf{a}_k$ *is a proper interval and* $\mathbf{b}_k$ *is not zero.*

Example. 4.4 *We denote the sum* γ_i *from* (13) *as* γ_i'*, the sum from* (14) *as* γ_i'' *and the sum from* (15) *as* γ_i'''*. We give three examples, one for each of the above three possibilities and put* $c' = 0$*. For simplicity, we perform all calculations with 16 digit decimal arithmetic, doing the sums from left to right.*

Case 1: For $\overline{p} = (1, 1, \infty)$*, we get*

$$\begin{aligned} \gamma_3' &= 1 + 1 = 2, \\ \gamma_3'' &= (1 + 1 + \infty) - \infty = NaN, \\ \gamma_3''' &= 1 + 1 = 2. \end{aligned}$$

Case 2: For $\overline{p} = (8 \cdot 10^{-8}, 9 \cdot 10^{-8}, 10^9)$*, we get*

$$\begin{aligned} \gamma_3' &= 8 \cdot 10^{-8} + 9 \cdot 10^{-8} = 1.7 \cdot 10^{-7}, \\ \gamma_3'' &= (8 \cdot 10^{-8} + 9 \cdot 10^{-8} + 10^9) - 10^9 = 0, \\ \gamma_3''' &= 9 \cdot 10^{-8} + 8 \cdot 10^{-8} - (10^9 - 10^9) = 1.7 \cdot 10^{-7}. \end{aligned}$$

Case 3: For $\overline{p} = (-10^9, 8 \cdot 10^{-8}, 9 \cdot 10^{-8}, 10^9)$*, we get*

$$\begin{aligned}
\gamma_3' &= -10^9 + 8 \cdot 10^{-8} + 10^9 = 10^{-6}, \\
\gamma_3'' &= (-10^9 + 8 \cdot 10^{-8} + 9 \cdot 10^{-8} + 10^9) - 9 \cdot 10^{-8} = -9 \cdot 10^{-8}, \\
\gamma_3''' &= (8 \cdot 10^{-8} + 9 \cdot 10^{-8}) + 10^9 - (9 \cdot 10^{-8} + 10^9) = 0.
\end{aligned}$$

In the two first cases our formula (15) *reproduces* (13)*, while the formula* (14) *fails in the first case, and suffers from severe cancellation in the second case. In the third case all formulas suffer from cancellation.*

Since (13) is a univariate, quadratic expression, the results of Section 3 can be applied. This may result in an improved bound $\mathbf{x}_i'$ on the variable x_i. If we cut it with the original bound on x_i we obtain $x_i \in \mathbf{x}_i' \cap \mathbf{x}_i$. Since we approximate all univariate expression $p_i(x_i)$, $i \in \{1, \ldots, n\}$, we obtain the new bound constraints

$$x \in \mathbf{x}' \cap \mathbf{x}.$$

In general, separable quadratic constraints can be written as

$$\sum_{k=1}^{n} p_k(x_k) \in \mathbf{c}, \tag{16}$$

since they have both a lower bound $\underline{c}$ and an upper bound $\overline{c}$. The inequalities

$$\sum_{k=1}^{n} p_k(x_k) \geq \underline{c} \text{ and } \sum_{k=1}^{n} -p_k(x_k) \geq -\overline{c},$$

represent (16), and for them all the results of this section can be applied.

5 Nonseparable quadratic constraints

In this section we discuss a method for removing a bilinear term from nonseparable quadratic constraints. This is important since by removing all bilinear terms in turn, the problem is transformed to a separable one, to which the results of the previous sections can be applied.

Example. 5.1 *(i) For some positive constant k, we consider the quadratic constraint*

$$f(x) := kx_1^2 + kx_2^2 + kx_3^2 + 2x_1x_2 + 2x_1x_3 + 2x_2x_3 \leq 1. \tag{17}$$

This constraint defines a bounded ellipsoid when $k > 1$, while for $k \leq 1$, an unbounded domain results. Indeed, the Hessian

$$G = 2 \begin{pmatrix} k & 1 & 1 \\ 1 & k & 1 \\ 1 & 1 & k \end{pmatrix}$$

has the principal sub–determinants $G_{11} = 2k > 0$, $\det G_{1:2,1:2} = 2^2(k^2 - 1)$ *and* $\det G = 8(k^3 - 3k + 2) = 2^3(k-1)^2(k+2)$*; hence G is positive definite iff $k > 1$.*

If we rewrite $f(x)$ as

$$f(x) = (k-2)(x_1^2 + x_2^2 + x_3^2) + (x_1 + x_2)^2 + (x_1 + x_3)^2 + (x_2 + x_3)^2$$

and drop the (nonnegative) quadratic terms, we find the separable quadratic inequality

$$(k-2)(x_1^2+x_2^2+x_3^2) \le 1. \tag{18}$$

For $k=3$, we find $x_1^2+x_2^2+x_3^2 \le 1$, and our separable constraint propagation gives the bounds $x_i \in [-1,1]$ for $i=1,\dots,3$. Similarly, any $k>2$ leads to a finite box, which gets arbitrarily large as k tends to 2. However, for any value of $k \le 2$, (18) *is a trivial, non–informative inequality. On the other hand, we have seen that the original constraint* (18) *defines a bounded domain when $k>1$. Thus, for $k \in]1,2]$, the above method of eliminating bilinear terms is not able to exploit the full power of* (18).

In Domes & Neumaier *[17], we describe the ellipsoid hull technique, which always yields optimal bounds based on more expensive (and much more difficult to rigorously analyze) linear algebra. For example, when $k=2$, we get the finite bounds $x_i \in [-0.86606, 0.86606]$ for $i=1,\dots,3$.*

(ii) If we add the bound constraints $\mathbf{x}_i=[-1,5]$, $i=1,\dots,3$ to the constraint (17) *and set $k=2$, we can approximate the bilinear terms x_jx_k by the interval evaluation of $\mathbf{x}_j\mathbf{x}_k$ and obtain $2x_jx_k \in [-10,50]$. In this case* (17) *reduces to*

$$2x_1^2+2x_2^2+2x_3^2 \le 31. \tag{19}$$

Since $x_k \in [-1,5]$, we find that $x_k^2 \in \mathbf{p}_k := [0,25]$ for all $k=1,\dots,3$. Therefore for all $i=1,\dots,3$

$$2x_i^2 \le (31 - \sum_{k\neq i} \underline{p}_k) = 31$$

holds, yielding the bound $x_i \le \sqrt{31/2} \le 3.94$.

(iii) Suppose we have the same bound constraints as in (ii). We approximate the bilinear terms by linear and constant ones. Since $x_i \in [-1,5]$ for each bilinear term $2x_jx_k$ the inequality

$$2x_j+2x_k-26 \le 2x_jx_k,$$

holds. Therefore by (17) *we obtain*

$$kx_1^2+kx_2^2+kx_3^2+4x_1+4x_2+4x_3 \le 79. \tag{20}$$

Since $x_k \in [-1,5]$, we find that $2x_k^2+4x_k \in \mathbf{p}_k := [-2,70]$ for all $k=1,\dots,3$. Then for all $i=1,\dots,3$

$$2x_i^2+4x_i \le (79 - \sum_{k\neq i} \underline{p}_k) = 83$$

holds, yielding the bound $x_i \le -1+\sqrt{85/2} \le 5.52$.

The example shows that approximating the bilinear entries in different ways can lead to different results.

We now formalize and extend the methods used in the preceding example. We consider an arbitrary multivariate quadratic inequality constraint, which we write without loss of generality in the form

$$\sum_k (a_kx_k^2+b_kx_k) + \sum_{\substack{j,k\\ j>k}} b_{jk}x_jx_k \ge c, \quad x \in \mathbf{x}, \tag{21}$$

where the $a_kx_k^2$ are the quadratic, the b_kx_k the linear and $b_{jk}x_jx_k$ the bilinear terms.

5.1 Approximation by constants

As in Example 5.1 (ii), we find rigorous bounds for a bilinear term $b_{jk}x_jx_k$ when x_j and x_k are bounded. We evaluate $\mathbf{x}_i\mathbf{x}_j$ by using the rule for multiplying the intervals $\mathbf{x}_i$ and $\mathbf{x}_j$ (see, e.g., Neumaier [37]) and obtain

$$b_{jk}\mathbf{x}_i\mathbf{x}_j \le \begin{cases} b_{jk}\min\left(\underline{\mathbf{x}}_i\underline{\mathbf{x}}_j, \underline{\mathbf{x}}_i\overline{\mathbf{x}}_j, \overline{\mathbf{x}}_i\underline{\mathbf{x}}_j, \overline{\mathbf{x}}_i\overline{\mathbf{x}}_j\right) & \text{if } b_{jk} < 0 \\ b_{jk}\max\left(\underline{\mathbf{x}}_i\underline{\mathbf{x}}_j, \underline{\mathbf{x}}_i\overline{\mathbf{x}}_j, \overline{\mathbf{x}}_i\underline{\mathbf{x}}_j, \overline{\mathbf{x}}_i\overline{\mathbf{x}}_j\right) & \text{if } b_{jk} \ge 0. \end{cases} \tag{22}$$

Directed rounding when evaluating the right hand side ensures that no feasible points can be lost during this process.

5.2 Approximation by linear terms

As in Example 5.1 (iii), we approximate each bilinear term $b_{jk}x_jx_k$ with $x_j \in \mathbf{x}_j$ and $x_k \in \mathbf{x}_k$, by the linear expression

$$q_{jk}(x_{jk}) := g_{jk}x_j + e_{jk}x_k + h_{jk}. \tag{23}$$

If for all $x \in \mathbf{x}$ the inequality

$$g_{jk}x_j + e_{jk}x_k + h_{jk} \ge b_{jk}x_jx_k, \tag{24}$$

holds, by (21) we get

$$\sum_k (a_kx_k^2 + b_kx_k) + \sum_{\substack{j,k \\ j>k}} (g_{jk}x_j + e_{jk}x_k + h_{jk}) \ge \sum_k (a_kx_k^2 + b_kx_k) + \sum_{\substack{j,k \\ j>k}} b_{jk}x_jx_k \ge c,$$

for all $x \in \mathbf{x}$. In order to get an optimal $q_{jk}(x_{jk})$ we set

$$g_{jk} := b_{jk}z_k, \quad e_{jk} := b_{jk}z_j$$

for some $z_k \in \mathbf{x}_k$. By (24) term h_{jk} can be obtained by finding the upper bound of

$$f(x_j, x_k) := b_{jk}x_jx_k - (e_{jk}x_k + g_{jk}x_j) = (b_{jk}x_j - e_{jk})x_k - g_{jk}x_j \tag{25}$$

for $x_k \in \mathbf{x}_k$ and $x_j \in \mathbf{x}_j$. There, again the direct interval evaluation of (25) can be used, but to save computational time we propose:

Proposition. 5.2 *Let $f(x,y)$ be monotone in x and y, and suppose that $x \in \mathbf{x}$ and $y \in \mathbf{y}$. Then*

$$\square\{f(x,y) \mid x \in \mathbf{x},\ y \in \mathbf{y}\} = \square\{f(\underline{x},\underline{y}) \cup f(\overline{x},\underline{y}) \cup f(\underline{x},\overline{y}) \cup f(\overline{x},\overline{y})\}$$

holds.

Proof.

$$\begin{aligned} &\square\{f(x,y) \mid x \in \mathbf{x},\ y \in \mathbf{y}\} = \square\{(\square\{f(x,y) \mid x \in \mathbf{x}\}) \mid y \in \mathbf{y}\} \\ &= \square\{(\square\{f(\underline{x},y) \cup f(\overline{x},y)\}) \mid y \in \mathbf{y}\} = \square\{\square\{f(\underline{x},y) \mid y \in \mathbf{y}\} \cup \square\{f(\overline{x},y) \mid y \in \mathbf{y}\}\} \\ &= \square\{f(\underline{x},\mathbf{y}) \cup f(\overline{x},\mathbf{y})\} = \square\{f(\underline{x},\underline{y}) \cup f(\overline{x},\underline{y}) \cup f(\underline{x},\overline{y}) \cup f(\overline{x},\overline{y})\} \end{aligned} \tag{26}$$

holds. $\square$

In floating point arithmetics, we have to ensure correct upward rounding and therefore we compute

$$\begin{array}{llll} u_1 = \Delta f(\underline{x}, \underline{y}), & u_2 = \Delta f(\overline{x}, \underline{y}), & u_3 = \Delta f(\underline{x}, \overline{y}), & u_4 = \Delta f(\overline{x}, \overline{y}), \\ l_1 = \nabla f(\underline{x}, \underline{y}), & l_2 = \nabla f(\overline{x}, \underline{y}), & l_3 = \nabla f(\underline{x}, \overline{y}), & l_4 = \nabla f(\overline{x}, \overline{y}). \end{array}$$

and obtain

$$\square\{f(x,y) \mid x \in \mathbf{x},\ y \in \mathbf{y}\} = [\min_i l_i,\ \max_i u_i].$$

Applied to (25), noting that $f(x,y)$ is monotone in both x and y, we find the upper bound

$$h_{jk} = \max_i u_i$$

with

$$\begin{array}{ll} u_1 = \Delta b_{jk}((\underline{x}_j - z_j)\underline{x}_k - z_k\underline{x}_j), & u_2 = \Delta b_{jk}((\overline{x}_j - z_j)\underline{x}_k - z_k\overline{x}_j), \\ u_3 = \Delta b_{jk}((\underline{x}_j - z_j)\overline{x}_k - z_k\underline{x}_j), & u_4 = \Delta b_{jk}((\overline{x}_j - z_j)\overline{x}_k - z_k\overline{x}_j). \end{array}$$

5.3 Approximation by separable quadratic terms

Alternatively, when a bound for x_j or x_k is large, but the coefficients a_j and a_k of the corresponding quadratic terms have negative sign, it is usually better to proceed as in Example 5.1 (i) and relax the bilinear entries by quadratic ones. Note that when the constraint in that example is rewritten in the form (21), the coefficients of the quadratic terms become negative. This is a necessary condition for the constraint to lead to a bounded feasible set.

To ensure good scaling behavior, we want to bound $b_{jk}x_jx_k$ by a multiple of $a_jx_j^2 + a_kx_k^2$:

Proposition. 5.3 *Suppose that $a_k < 0$ and $a_j < 0$, and put*

$$v_{jk} := \operatorname{sign}(b_{jk})\sqrt{\frac{a_k}{a_j}}, \quad d_{jk} := \frac{b_{jk}}{2v_{jk}}.$$

Then

$$b_{jk}x_jx_k \le d_{jk}x_j^2 + \frac{b_{jk}v_{jk}}{2}x_k^2. \tag{27}$$

Proof. Since b_{jk} and v_{jk} have the same sign, $d_{jk} = \frac{b_{jk}}{2v_{jk}} \ge 0$ holds, and thus $d_{jk}(x_j - v_{jk}x_k)^2 \ge 0$ follows. In addition to this,

$$d_{jk}(x_j - v_{jk}x_k)^2 = d_{jk}x_j^2 + d_{jk}v_{jk}^2x_k^2 - 2d_{jk}v_{jk}x_jx_k = d_{jk}x_j^2 + \frac{b_{jk}v_{jk}}{2}x_k^2 - b_{jk}v_{jk}x_jx_k$$

and the inequality (27) follows from

$$\begin{aligned} 0 \le d_{jk}(x_j - v_{jk}x_k)^2 &= \frac{b_{jk}}{2v_{jk}}\frac{a_j}{a_j}x_j^2 + \frac{b_{jk}}{2v_{jk}}\frac{a_k}{a_j}x_k^2 - b_{jk}x_jx_k = \\ &\frac{d_{jk}}{a_j}(a_jx_j^2 + a_kx_k^2) - b_{jk}x_jx_k = d_{jk}x_j^2 + \frac{b_{jk}v_{jk}}{2}x_k^2 - b_{jk}x_jx_k. \end{aligned}$$

□

5.4 Combining the approximation methods

Since (22) tends to give better bounds on x_jx_k than (23) or (27) if the boxes $\mathbf{x}_j$ and $\mathbf{x}_k$ are not too wide, but infinite ones if the width of one of the boxes is infinite we combine the different methods and proceed as follows for each bilinear term with nonzero coefficients b_{jk}: First, we factor the quadratic, bilinear and linear terms which depend on the variables x_j and x_k from (21) and obtain

$$c(x) + a_kx_k^2 + a_jx_j^2 + b_{jk}x_jx_k + b_kx_k + b_jx_j \geq c, \tag{28}$$

The entries which do not depend on x_j or x_k are collected in

$$c(x) := \sum_{\substack{i \\ i\neq k, i\neq j}} (a_ix_i^2 + b_ix_i) + \sum_{\substack{m,i \\ m>i \\ (mi)\neq(jk)}} b_{mi}x_mx_i.$$

Then we handle the following cases:

1. If one of the bounds $\underline{x}_j$, $\overline{x}_j$, $\underline{x}_k$, $\overline{x}_k$ is large and both a_k and a_j are negative, we quadratically approximate the bilinear term $b_{jk}x_jx_k$ as described in Subsection 5.3. By the inequality (27) we obtain the relaxation

$$c(x) + (a_k + d_{jk})x_k^2 + \Big(a_j + \frac{b_{jk}v_{jk}}{2}\Big)x_j^2 + b_kx_k + b_jx_j \geq c, \tag{29}$$

showing that the bilinear term $b_{jk}x_jx_k$ has been eliminated from (28). We have separated the variables x_j and x_k in (28), ending up in

$$c(x) + a'_kx_k^2 + a'_jx_j^2 + b_kx_k + b_jx_j \geq c, \tag{30}$$

with the new quadratic coefficients

$$a'_k := a_k + d_{jk},\ a'_j := a_j + \frac{b_{jk}v_{jk}}{2}. \tag{31}$$

The linear and constant coefficients remain unchanged. If we have uncertainties in the coefficients; $a_k \in \mathbf{a}_k$ and $b_{jk} \in \mathbf{b}_{jk}$ then (31) is changes to

$$a'_k \in \mathbf{a}'_k, \quad a'_j \in \mathbf{a}'_j,$$

with

$$\mathbf{a}'_k = \mathbf{a}_k + \overline{d}_{jk}, \quad \mathbf{a}'_j = \mathbf{a}_j + \sup\frac{\mathbf{b}_{jk}\mathbf{v}_{jk}}{2}, \quad \mathbf{v}_{jk} = \operatorname{sign}(\mathbf{b}_{jk})\sqrt{\frac{\mathbf{a}_k}{\mathbf{a}_j}}, \quad \mathbf{d}_{jk} = \frac{\mathbf{b}_{jk}}{2\mathbf{v}_{jk}}.$$

2. If we have no large bounds on the variables x_j and x_k, the expression

$$p(x) := a_kx_k^2 + a_jx_j^2 + b_{jk}x_jx_k + b_kx_k + b_jx_j$$

can be bounded from above by a convex function only if it has a finite global maximum. This requires that the Hessian

$$H = \begin{pmatrix} 2a_k & b_{jk} \\ b_{jk} & 2a_j \end{pmatrix}$$

is negative semidefinite. If $b_{jk} \neq 0$ this implies that a_k and a_j are negative. Therefore, the constraint propagation on this constraint is useful only in this case.

3. If all bounds $\underline{x}_j$, $\overline{x}_j$, $\underline{x}_k$, $\overline{x}_k$ are not large we approximate the bilinear term $b_{jk}x_jx_k$ by applying the results of Subsection 5.1. In addition to this, if we assume that we have uncertainties in the coefficient $b_{jk} \in \mathbf{b}_{jk}$, we can add the supremum of $b_{jk}x_jx_k$ to the right hand side of the inequality (28) obtaining

$$c(x) + a_kx_k^2 + a_jx_j^2 + b_kx_k + b_jx_j \geq c', \tag{32}$$

where

$$c' := c + \begin{cases} \overline{b}_{jk} \max\{\underline{x}_j\underline{x}_k, \underline{x}_j\overline{x}_k, \overline{x}_j\underline{x}_k, \overline{x}_j\overline{x}_k\} & \text{if } \overline{b}_{jk} \geq -\underline{b}_{jk} \\ -\underline{b}_{jk} \max\{(-\underline{x}_j)\underline{x}_k, (-\underline{x}_j)\overline{x}_k, (-\overline{x}_j)\underline{x}_k, (-\overline{x}_j)\overline{x}_k\} & \text{if } \overline{b}_{jk} < -\underline{b}_{jk}. \end{cases}$$

Thus we have separated the variables x_j and x_k in (28). The quadratic and linear coefficients remain unchanged. Note that the signs in the above expression are intended to save rounding mode switches.

4. If the bounds $\underline{x}_j$, $\overline{x}_j$, $\underline{x}_k$, $\overline{x}_k$ are not large, for special applications (e.g. for computing linear relaxations as in Domes & Neumaier [18]), it can be suitable to approximate the bilinear terms by linear expressions. We apply the results of Subsection 5.2; we choose a $z \in \mathbf{x}$ (e.g., $z = (\overline{x} + \underline{x})/2$ is a good choice) and approximate each $b_{jk}x_jx_k$ with $x_j \in \mathbf{x}_j$ and $x_k \in \mathbf{x}_k$ by

$$b_{jk}z_j + b_{jk}z_k + d \geq b_{jk}x_jx_k, \quad d := \max_i u_i$$

where

$$\begin{array}{ll} u_1 = \Delta b_{jk}((\underline{x}_j - z_j)\underline{x}_k - z_k\underline{x}_j), & u_2 = \Delta b_{jk}((\overline{x}_j - z_j)\underline{x}_k - z_k\overline{x}_j), \\ u_3 = \Delta b_{jk}((\underline{x}_j - z_j)\overline{x}_k - z_k\underline{x}_j), & u_4 = \Delta b_{jk}((\overline{x}_j - z_j)\overline{x}_k - z_k\overline{x}_j). \end{array} \tag{33}$$

Then by (28) we have

$$\begin{aligned} &c(x) + a_kx_k^2 + a_jx_j^2 + (b_{jk}z_j + b_k)x_k + (b_{jk}z_k + b_j)x_j + d \\ &\geq c(x) + a_kx_k^2 + a_jx_j^2 + b_{jk}x_jx_k + b_kx_k + b_jx_j \geq c. \end{aligned}$$

Therefore we successfully separated the the variables x_j and x_k in (28) and obtain

$$c(x) + a_kx_k^2 + a_jx_j^2 + b'_kx_k + b'_jx_j \geq c' \tag{34}$$

with the new linear and constant coefficients

$$b'_k := b_{jk}z_j + b_k,\ b'_j := b_{jk}z_k + b_j,\ c' := c - \max_i u_i. \tag{35}$$

The quadratic coefficients remain unchanged. If we have the uncertainties $b_{jk} \in \mathbf{b}_{jk}$ and $b_k \in \mathbf{b}_k$ in the coefficients, (35) changes to

$$b'_k \in \mathbf{b}'_k, \quad b'_j \in \mathbf{b}'_j, \quad c' = c - \max_i u_i,$$

with

$$\begin{array}{ll} \mathbf{b}'_k = \mathbf{b}_{jk}z_j + \mathbf{b}_k, & \mathbf{b}'_j = \mathbf{b}_{jk}z_k + \mathbf{b}_j, \\ u_1 = \sup(\mathbf{b}_{jk}((\underline{x}_j - z_j)\underline{x}_k - z_k\underline{x}_j)), & u_2 = \sup(\mathbf{b}_{jk}((\overline{x}_j - z_j)\underline{x}_k - z_k\overline{x}_j)), \\ u_3 = \sup(\mathbf{b}_{jk}((\underline{x}_j - z_j)\overline{x}_k - z_k\underline{x}_j)), & u_4 = \sup(\mathbf{b}_{jk}((\overline{x}_j - z_j)\overline{x}_k - z_k\overline{x}_j)). \end{array}$$

Applying the above on (28) for all indexes $j \in \{1, \dots, n\}$ and $k \in \{1, \dots, n\}$ with $j < k$, we obtain the new separable system

$$\sum_k a'_kx_k^2 + b'_kx_k \geq c', \quad x \in \mathbf{x}. \tag{36}$$

All above bounds should be computed with upward rounding.

6 Constraint propagation in GloptLab

This section discusses how the new techniques presented above are implemented in the GloptLab environment to solve algebraic constraint satisfaction problems. The problems treated in GloptLab consist (after preliminary transformations) of simple bounds, linear constraint, and quadratic constraints. We represent simple bounds as box constraint $x \in \mathbf{x}$. The linear and quadratic constraints are represented in a sparse matrix notation. The linear, quadratic, and bilinear monomials occurring in at least one of the constraint (but not the constant term) are collected into an n_q-dimensional column vector $q(x)$. There we choose

$$q(x) = (x_1, \ldots, x_n,\ x_1^2, \ldots, x_1x_n,\ \ldots\ x_nx_1, \ldots, x_n^2)^T$$

The coefficients of the ith constraint in the resulting monomial basis are collected in the ith row of a (generally sparse) matrix A, and any constant term (if present) is moved to the right hand side. Thus the linear and quadratic constraints take the form $A_{i:}q(x) \in \mathbf{F}_i$ ($i = 1 \ldots m$), where $\mathbf{F}_i$ is a closed interval, and $A_{j:}$ denotes the jth row of A.

As in the case of simple bounds, this includes equality constraints and one-sided constraints by choosing for the corresponding $\mathbf{F}_i$ degenerate or unbounded intervals. In compact vector notation, the constraints take the form $Aq(x) \in \mathbf{F}$.

While traditionally the coefficients in a constraint are taken to be exactly known, we allow them to vary in (narrow) intervals, to be able to rigorously account for uncertainties due to measurements of limited accuracy, conversion errors from an original representation to our normal form, and rounding errors when creating new constraints by relaxation techniques. Thus the coefficient matrix A is allowed to vary arbitrarily within some interval matrix $\mathbf{A}$. The $m \times n_q$ interval matrix $\mathbf{A}$ with closed and bounded interval components $\mathbf{A}_{ik} = [\underline{A}_{ik}, \overline{A}_{ik}]$, is interpreted as the set of all $A \in \mathbb{R}^{m\times n}$ such that $\underline{A} \le A \le \overline{A}$, where $\underline{A}$ and $\overline{A}$ are the matrices containing the lower and upper bounds of the components of $\mathbf{A}$.

We therefore pose the general quadratic constraint satisfaction problem in the form

$$Aq(x) \in \mathbf{F}, \quad x \in \mathbf{x}, \quad A \in \mathbf{A}. \tag{37}$$

We now summarize the constraint propagation method for the quadratic constraint satisfaction problem (37).

Problem simplification. First we simplify the problem; we remove the constraints of the form $bx_j \in \mathbf{F}_i$ and modify the corresponding bounds on the variable x_j. We also remove the variables which are fixed from the constraints, and the entries corresponding to the removed variables from the vector $q(x)$. The dimensions of the coefficient matrix A and the box $\mathbf{x}$ may change in this step.

Resolving the two-sided constraints. We resolve the two-sided constraints of (37) into inequalities. We define

$$A^n := \begin{pmatrix} -\underline{A}_{I,:} \\ \overline{A}_{J,:} \end{pmatrix}, \quad A^p := \begin{pmatrix} \overline{A}_{I,:} \\ -\underline{A}_{J,:} \end{pmatrix}, \quad c := \begin{pmatrix} \underline{F}_I \\ -\overline{F}_J \end{pmatrix},$$

where

$$I := \{i \in 1, \ldots, m \mid \underline{F}_i > -\infty\} \text{ and } J := \{i \in 1, \ldots, m \mid \overline{F}_i < \infty\}.$$

The system of quadratic inequalities

$$Aq(x) \ge c, \quad x \in \mathbf{x}, \quad A \in \mathbf{A} := [-A^n, A^p], \tag{38}$$

is another representation of the quadratic constraint satisfaction problem (37). The matrix A is $m \times n_q$ dimensional, where $m := n_I + n_J$ depends on the length n_I of the index set I and on the length n_J of the index set J.

Separating the constraints. We transform the quadratic constraint satisfaction problem into a separable one. The ith row of (38) matches the form of

$$\sum_k (a_k x_k^2 + b_k x_k) + \sum_{\substack{j,k \\ j>k}} b_{jk} x_j x_k \geq c, \quad x \in \mathbf{x},$$

of (21), with

$$a_k := \overline{A}_{i,kn+k}, \quad b_k \in \mathbf{A}_{i,k}, \quad b_{jk} \in \mathbf{A}_{i,jn+k} \text{ and } c := c_i. \tag{39}$$

Here we used the upper bounds for the quadratic terms since the sign of x_k^2 is known. Then we use the results of Section 5 to remove all bilinear entries from each constraint, obtaining the new coefficients

$$a'_k \in \mathbf{a}'_k, \quad b'_k \in \mathbf{b}_k, \quad b_{jk} = 0 \text{ and } c = c'_i. \tag{40}$$

Depending on the removal method we have applied either the quadratic coefficients or the bound c' or both of them have been changed, ending up in a new system

$$A'q(x) \geq c', \quad x \in \mathbf{x}, \quad A' \in \mathbf{A}' := [-A'^n, A'^p] \tag{41}$$

In (41) all bilinear coefficients are zero, therefore from this point on, the system is separable.

Forward and backward propagation. Since the ith row of (41) matches the form

$$\sum_{k=1}^{n} (a_k x_k^2 + b_k x_k) \geq c, \quad x \in \mathbf{x}, \quad a_k \in \mathbf{a}_k, \quad b_k \in \mathbf{b}_k$$

of (5), we can apply the forward and the backward propagation steps from Section 4.

We compute the enclosure $\mathbf{p}_k$ of each univariate quadratic term $p_k(x_k) := a_k x_k^2 + b_k x_k$ by using the theory developed in Section 2, where the uncertainties $\mathbf{a}_k$ and $\mathbf{b}_k$ of the constraint coefficients are also taken into account.

Then we use the $\mathbf{p}_k$ to verify that the constraint is feasible, to get a new bound on each $p_k(x_k)$ and to find a new lower bound for the constraint.

If the constraint has not yet been detected as infeasible, we can apply the backward propagation step (by using the theory from Section 3), which may yield tighter bounds on the variables.

7 Tests and Comparison

In this section we compare our quadratic constraint propagation method (QCP) with elementary constraint propagation (ECP).

The forward propagation step of the elementary constraint propagation finds the range of each expression in the constraints individually, then for all expressions in a constraint uses the ranges of all other expressions to get new bounds on them. The backward propagation step uses the inverse of the expressions to get new bounds on the variables.

Example. 7.1 *To demonstrate the elementary constraint propagation and simultaneously compare it to the method presented in this paper we again solve the problem*

$$-x_1^2 + 2x_1 - x_2 \geq -8 \quad \textit{with } x_1 \in [-7,5],\ x_2 \in [0,\infty].$$

from Example 4.2 and give the step by step comparison of the two different approaches:

ECP:	**QCP:**
• *For* $x_1 \in [-7,5]$, $x_2 \in [0,\infty]$, *we have* $-x_1^2 \in [-49,0]$, $2x_1 \in [-14,10]$ *and* $-x_2 \in [-\infty,0]$.	• *For* $x_1 \in [-7,5]$, $x_2 \in [0,\infty]$, *we have* $-x_1^2 + 2x_1 \in [-63,1]$ *and* $-x_2 \in [-\infty,0]$.
• *Therefore,* $-x_1^2 + 2x_1 - x_2 \leq 10$.	• *Therefore,* $-x_1^2 + 2x_1 - x_2 \leq 1$.
• *From* $0 \geq -x_2$, $0 \geq -x_1^2$ *and* $10 \geq 2x_1$, *the inequalities* $-x_1^2 \geq -8-0-10 = -18$ *and* $2x_1 \geq -8-0-0 = -8$ *follows.*	• *From* $0 \geq -x_2$, *the inequality* $-x_1^2 + 2x_1 \geq -8-0 = -8$ *follows.*
• *Since* $10 \geq -x_1^2 + 2x_1$, *the inequality* $-x_2 \geq -8-10 = -18$ *holds.*	• *Since* $1 \geq -x_1^2 + 2x_1$, *the inequality* $-x_2 \geq -8-1 = -9$ *holds.*
• *From* $-x_1^2 \geq -18$, $x_1 \in [-\sqrt{18},\sqrt{18}]$, *from* $2x_1 \geq -8$, $x_1 \in [-4,\infty]$, *and from* $-x_2 \geq -18$, $x_2 \in [-\infty,18]$ *follows.*	• *From* $-x_1^2 + 2x_1 \geq -8$, $x_1 \in [-2,4]$, *and from* $-x_2 \geq -9$, $x_2 \in [-\infty,9]$ *follows.*
• *Intersecting the bounds on the variables with the original bounds, we obtain* $-x_1^2 + 2x_1 - x_2 \in [-8,10]$ *with* $x_1 \in [-4,\sqrt{18}]$, $x_2 \in [0,18]$.	• *Intersecting the bounds on the variables with the original bounds, we obtain* $-x_1^2 + 2x_1 - x_2 \in [-8,1]$ *with* $x_1 \in [-2,4]$, $x_2 \in [0,9]$.

As the example shows for this problem the quadratic constraint propagation presented in this paper gives significantly tighter bounds than the elementary constraint propagation used as the pruning step of several state-of-the-art constraint propagation methods (e.g. [10, 20, 48]) with hardly any extra work. Probing ([46], also called slicing or shaving) would yield the same results as QCP, but at a much higher cost.

Strategy. 7.2 (Test strategies)
In order to compare our method with the traditional approaches we reimplemented the elementary constraint propagation in MATLAB, *integrated in* GLOPTLAB *[16] and used branch and prune to solve random problems.* GLOPTLAB *executes configurable strategies, for the comparison we used the following one, with default tuning parameter settings (which are configurable in* GLOPTLAB*).*

```
01: Read Problem
02: Begin While
03:    Propagate*
04: End While
05: Begin Split
06:    Begin While
07:       Propagate*
08:    End While
09: End Split
11: Finish
```

Test strategies	
name	`Propagate*` *method selection*
ELEM	*elementary constraint propagation*
SCON	*quadratic CP with constant bilinear approximation*
SLIN	*quadratic CP with linear bilinear approximation*
SQUA	*quadratic CP with quadratic bilinear approximation*
SAUT	*quadratic CP with automatic bilinear approximation*

Each strategy from 7.2 first reads the problem then accomplishes a single propagation step (each of them using different method) until the gain is less than 20% of the original box or until the number of iterations exceeds 20. Then the branching process follows; the box is split at the midpoint of a selected component then the same sequence of constraint propagation is applied to subboxes as before. Infeasible boxes are discarded, feasible boxes are split again if their maximum width is more than 0.001. Boxes of maximum width smaller than 0.001 are not split but saved for the final output.

In the first test we use the strategies from 7.2 to test three test sets of 50 random, infeasible problems. The problems are 2 dimensional in the first, 5 dimensional in the second, and 10 dimensional in the third test set. Each problem in the test consists of a single conic inequality constraint with random coefficients and random bound constraints $x \in \mathbf{x}$, choosen such that $\mathbf{x}_i \subseteq [-1, 1]$ and the problem is infeasible (infeasibility was verified by using a more complicated strategy; Strategy 5.2 from DOMES [16], Section 5.2). The table below shows the median of the solution times (in seconds) and the median of the splits required to solve the problems contained in each test set.

Branch and prune test results.

`dimension`	`n = 2`		`n = 5`		`n = 10`	
`method`	`time`	`splits`	`time`	`splits`	`time`	`splits`
`ELEM`	0.003	0	0.255	3.00	22.58	47.5
`SCON`	0.001	0	0.033	1.75	0.722	47.3
`SLIN`	0.003	0	0.039	1.50	0.984	55.0
`SQUA`	0.001	0	0.045	2.00	1.500	95.5
`SAUT`	0.002	0	0.035	1.75	0.813	47.3

As the results show, verifying in higher dimensions that the search space does not contain points of single conic inequality constraint consisting of bilinear terms using constraint propagation is a non-trivial task. The reason is that the approximation error of the bilinear terms (and in case of the elementary constraint propagation also the approximation of the separable quadratic expressions) makes the CP incapable to discard the regions of the search space which are close to region defined by the constraint. Only a division of the search space into several subboxes leads to a solution.

The elementary constraint propagation is slower than the quadratic constraint propagation, due to the need of more rounding mode switches in the interval arithmetic and the greater approximation error (see Example 7.1). Since the width of the bound constraint box is small, the constant approximation performs better than the linear or the quadratic ones. The automatic method is only slightly slower than the constant approximation, but has the advantage that it is also performs good when the bound constraints are large.

The following tests show how the constraint propagation method presented in this paper scales favorably with the complexity of the constraints. The test problems are 9 dimensional, having linear equality constraints (depending on the variables x_i, x_{i+1} and x_{i+2}, $i = 1, \dots, 7$) but the fifth constraint also has some quadratic and bilinear terms in 3 (`Test 1`), in 4 (`Test 2`), in 5 (`Test 3`), or in 6 variables (`Test 4`). In `Test 1-4` the fifth constraint is convex, while in `Test 1'-4'` non-convex. We have chosen 20 random bound constraints $x \in \mathbf{x}$ such that $\underline{x}_i \in [-b, 0]$ and $\overline{x}_i = \underline{x}_i + 2b$ for $i = 1, \dots n$, and listed the different bs in the `bound` column of the tables. We added the random bounds to the test problems, and solved them using the `ELEM` and the `SAUT` strategy. The median of the solution times (in seconds) are shown in the following tables:

Median of the solution times for convex (`Test 1-4`) and non-convex (`Test 1'–4'`) problems. Problems in each test run: 20, strategy: `ELEM`.

`bound`	`Test 1`	`Test 1'`	`Test 2`	`Test 2'`	`Test 3`	`Test 3'`	`Test 4`	`Test 4'`
1	0.082	0.055	0.094	0.081	0.257	0.171	0.170	0.348
10	1.391	1.031	1.823	1.605	2.772	2.589	4.229	4.597
100	1.416	1.008	1.969	1.894	2.624	2.862	4.688	4.414
1000	1.883	1.481	2.098	2.085	3.217	3.245	5.496	4.749

Median of the solution times for convex (`Test 1-4`) and non-convex (`Test 1'–4'`) problems. Problems in each test run: 20, strategy: `SAUT`.

`bound`	`Test 1`	`Test 1'`	`Test 2`	`Test 2'`	`Test 3`	`Test 3'`	`Test 4`	`Test 4'`
1	0.017	0.028	0.027	0.021	0.025	0.021	0.023	0.040
10	0.244	0.282	0.293	0.256	0.325	0.319	0.339	0.404
100	0.256	0.289	0.286	0.298	0.294	0.335	0.338	0.360
1000	0.406	0.348	0.696	0.390	0.694	0.339	0.919	0.677

Note that the convex problems without additional bound constraints can be solved in less then 0.12 seconds by adding the ellipsoid hull enclosure method presented in DOMES & NEUMAIER [17] to the above strategy.

Remark. 7.3 *Testing and comparing the above methods with other constraint propagation methods on standard benchmarks only makes sense when the method is integrated in the same strategy (combined with branch and bound, shaving, relaxations etc.). Constraint propagation alone is not powerful enough to solve the most real life problems and the type and quality of the auxiliary methods does count a lot if we would like to compare our method with the constraint propagation methods implemented in other solvers like* ICOS *or* REALPAVER.
An implementation of the above methods in other programming languages (e.g., in C++, *which is significantly faster than* MATLAB*) or using other representations of the problem (e.g. using DAGs) should yield a significant reduction of the solution times. However, we expect that the relative quality of the different methods will approximately remain the same. A* C++ *implementation for the* COCONUT *Environment is in preparation.*

Acknowledgment

This research was supported through the research grant FSP 506/003 of the University of Vienna. Numerous suggestions by the referees, which markedly improved the presentation of the paper, are gratefully acknowledged.

References

[1] E.D. Anderson and K.D. Anderson. Presolving in linear programming. *Math. Program.*, 71:221–245, 1995.

[2] K. R. Apt. The essence of constraint propagation. *Theoretical Computer Science*, 221 (1):179–210, 1999.

[3] A.B. Babichev, O.B. Kadyrova, T.P. Kashevarova, A.S. Leshchenko, and A.L. Semenov. UniCalc, a novel approach to solving systems of algebraic equations. *Interval Computations*, 3:29–47, 1993.

[4] F. Benhamou. Heterogeneous constraint solving. In Michael Hanus and Mario Rodrguez-Artalejo, editors, *Proceedings of ALP'96, 5th International Conference on Algebraic and Logic Programming*, vol. 1139 of *Lecture Notes in Computer Science*, pp. 62–76, Aachen, Germany, 1996. Springer-Verlag.

[5] F. Benhamou, F. Goualard, L. Granvilliers, and J.F. Puget. Revising hull and box consistency. In *International Conference on Logic Programming*, pp. 230–244, 1999. URL `citeseer.ist.psu.edu/benhamou99revising.html`.

[6] F. Benhamou, L. Granvilliers, and F. Goualard. Interval constraints: Results and perspectives. In *New Trends in Constraints*, pp. 1–16, 1999. URL `citeseer.ist.psu.edu/benhamou99interval.html`.

[7] F. Benhamou, D. McAllister, and P. Van Hentenryck. CLP(intervals) revisited. In *Proc. International Symposium on Logic Programming*, pp. 124–138. MIT Press, 1994.

[8] F. Benhamou and W.J. Older. Applying interval arithmetic to real, integer, and boolean constraints. *J. Logic Program*, 32:1–24, 1997.

[9] C. Bliek, P. Spelucci, L.N. Vicente, A. Neumaier, L. Granvilliers, E. Monfro, F. Benhamou, E. Huens, P. Van Hentenryck, D. Sam-Haround, and B. Faltings. Algorithms for solving nonlinear constrained and optimization problems: the state of the art, 2000. URL `http://www.mat.univie.ac.at/~neum/ms/StArt.pdf`.

[10] M. Ceberio and L. Granvilliers. Solving nonlinear systems by constraint inversion and interval arithmetic. In *AISC*, pp. 127–141, 2000. URL `http://link.springer.de/link/service/series/0558/bibs/1930/19300127.htm`.

[11] H.M. Chen and M.H. van Emden. Adding interval constraints to the Moore–Skelboe global optimization algorithm. In V. Kreinovich, editor, *Extended Abstracts of APIC'95 of the International Workshop on Applications of Interval Computations*, pp. 54–57, 1995.

[12] J.G. Cleary. Logical arithmetic. *Future Computing Systems*, 2:125–149, 1987.

[13] J. Cruz and P. Barahona. Constraint reasoning in deep biomedical models. *Journal of Artificial Intelligence in Medicine*, 34:77–88, 2005. URL `http://ssdi.di.fct.unl.pt/~pb/papers/ludi_constraints.pdf`.

[14] S. Dallwig, A. Neumaier, and H. Schichl. GLOPT - a program for constrained global optimization. In I. Bomze, T. Csendes, R. Horst, and P. M. Pardalos, editors, *Developments in Global Optimization*, pp. 19–36. Kluwer, Dordrecht, 1997.

[15] N.S. Dimitrova and S.M. Markov. Über die intervall-arithmetische Berechnung des Wertebereichs einer Funktion mit Anwendungen, Freiburger Intervall-Berichte. *Univ. Freiburg*, 81:1–22, 1981.

[16] F. Domes. GloptLab – a configurable framework for the rigorous global solution of quadratic constraint satisfaction problems. *Optimization Methods and Software*, 24: 727–747, 2009. URL `http://www.mat.univie.ac.at/~dferi/publ/Gloptlab.pdf`.

[17] F. Domes and A. Neumaier. Rigorous enclosures of ellipsoids and directed Cholesky factorizations. submitted, 2009. URL `http://www.mat.univie.ac.at/~dferi/publ/Cholesky.pdf`.

[18] F. Domes and A. Neumaier. Rigorous filtering using linear relaxations. in preparation, 2010. URL `http://www.mat.univie.ac.at/~dferi/publ/`.

[19] C. Grandon, D. Daney, and Y. Papegay. Combining CP and interval methods for solving the direct kinematic of a parallel robot under uncertainties. IntCP 06 Workshop, 2006. URL `ftp://ftp-sop.inria.fr/coprin/daney/articles/intcp06.pdf`.

[20] L. Granvilliers and F. Benhamou. Realpaver: An interval solver using constraint satisfaction techniques. *ACM Transactions on Mathematical Software*, 32:38–156, 2006. URL `http://www.sciences.univ-nantes.fr/info/perso/permanents/granvil/realpaver/`.

[21] G.D. Hager. Solving large systems of nonlinear constraints with application to data modeling. *Interval Computations*, 3:169–200, 1993.

[22] E. R. Hansen and G. W. Walster. Sharp bounds on interval polynomial roots. *Reliable Computing*, 8:115–122, 2002.

[23] P. Van Hentenryck. A Gentle Introduction to Numerica. *Artifical Intelligence*, 103: 209–235, 1998.

[24] P. Van Hentenryck, L. Michel, and F. Benhamou. Newton: constraint programming over non-linear constraints. *Sci. Program*, 30:83–118, 1997.

[25] P. Van Hentenryck, L. Michel, and Y. Deville. *Numerica. A modeling language for global optimization.* MIT Press, 1997.

[26] E. Hyvönen and S. De Pascale. Interval computations on the spreadsheet. In R. B. Kearfott and V. Kreinovich, editors, *Applications of Interval Computations*, pp. 169–209. Kluwer, 1996.

[27] L. Jaulin. Interval constraint propagation with application to bounded-error estimation. *Automatica*, 36:1547–1552, 2000. URL `https://www.ensieta.fr/e3i2/Jaulin/hull.pdf`.

[28] L. Jaulin. Interval constraints propagation techniques for the simultaneous localization and map building of an underwater robot, 2006. URL `http://www.mat.univie.ac.at/~neum/glopt/gicolag/talks/jaulin.pdf`.

[29] L. Jaulin, M. Kieffer, I. Braems, and E. Walter. Guaranteed nonlinear estimation using constraint propagation on sets. *International Journal of Control*, 74:1772–1782, 1999. URL `https://www.ensieta.fr/e3i2/Jaulin/observer.pdf`.

[30] C. Jermann, Y. Lebbah, and D. Sam-Haroud. Interval analysis, constraint propagation and applications. In F. Benhamou, N. Jussien, and B. O'Sullivan, editors, *Trends in Constraint Programming*, chapter 4, pp. 223–259. ISTE, 2007.

[31] R. B. Kearfott. Decomposition of arithmetic expressions to improve the behavior of interval iteration for nonlinear systems. *Computing*, 47:169–191, 1991.

[32] L. Krippahl and P. Barahona. PSICO: Solving protein structures with constraint programming and optimization. *Constraints*, 7:317–331, 2002. URL `http://ssdi.di.fct.unl.pt/~pb/papers/ludi_constraints.pdf`.

[33] Y. Lebbah. iCOs – Interval COnstraints Solver, 2003. URL `http://ylebbah.googlepages.com/icos`.

[34] Y. Lebbah, C. Michel, and M. Rueher. A rigorous global filtering algorithm for quadratic constraints. *Constraints*, 10:47–65, 2005. URL `http://ylebbah.googlepages.com/research`.

[35] W. A. Lodwick. Constraint propagation, relational arithmetic in ai systems and mathematical programs. *Ann. Oper. Res*, 21:143–148, 1989.

[36] J-P. Merlet. Solving the forward kinematics of a Gough-type parallel manipulator with interval analysis. *Int. J. of Robotics Research*, 23(3):221–235, 2004. URL `http://www-sop.inria.fr/coprin/equipe/merlet/Papers/IJRR2004.pdf`.

[37] A. Neumaier. *Interval methods for systems of equations*, vol. 37 of *Encyclopedia of Mathematics and its Applications*. Cambridge Univ. Press, Cambridge, 1990.

[38] A. Neumaier. Enclosing clusters of zeros of polynomials. *J. Comput. Appl. Math.*, 156: 389–401, 2003.

[39] A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 1004:271–369, 2004.

[40] W. Older and A. Vellino. Constraint arithmetic on real intervals. In F. Benhameou and A. Colmerauer, editors, *Constrained Logic Programming: Selected Research*. MIT Press, 1993.

[41] N. Sahinidis and M. Tawarmalani. *Convexification and global optimization in continuous and mixed–integer nonlinear programming: theory, algorithms, software, and applications*. Kluwer Academic Pub., 2003.

[42] N. V. Sahinidis and M. Tawarmalani. *BARON 7.2.5: global optimization of mixed-integer nonlinear programs,* User's Manual, 2005. URL `http://www.gams.com/dd/docs/solvers/baron.pdf`.

[43] H. Schichl. Mathematical modeling and global optimization, habilitation thesis, 2003. to appear. URL `http://www.mat.univie.ac.at/~herman/papers/habil.pdf`.

[44] H. Schichl, M. C. Markót, A. Neumaier, O. Shcherbina, E. Monfroy, B. Pajot, Xuan-Ha Vu, B. Toth, T. Vinko, K. Petras, and C. Keil. The COCONUT Environment, 2000-2010. Software. URL `http://www.mat.univie.ac.at/coconut-environment`.

[45] H. Schichl and A. Neumaier. Interval Analysis on Directed Acyclic Graphs for Global Optimization. *Journal of Global Optimization*, 33(4):541–562, 2005.

[46] M. H. van Emden. Computing functional and relational box consistency by structured propagation in atomic constraint systems. *CoRR*, cs.PL/0106008, 2001.

[47] Xuan-Ha Vu, H. Schichl, and D. Sam-Haroud. Using directed acyclic graphs to co-ordinate propagation and search for numerical constraint satisfaction problems. In *In Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004)*, pp. 72–81, 2004. URL `http://www.mat.univie.ac.at/~herman/papers/ICTAI2004.pdf`.

[48] Xuan-Ha Vu, H. Schichl, and D. Sam-Haroud. Interval propagation and search on directed acyclic graphs for numerical constraint solving. *Journal of Global Optimization*, p. 39, 2007. to appear. URL `http://www.mat.univie.ac.at/~herman/papers/FBPD-Hermann.pdf`.