Constraint propagation on quadratic constraints

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

March 19, 2008

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.

Keywords. Constraint propagation, constraint programming, continuous constraints, quadratic constraint satisfaction problems, rounding error control, verified computation, preprocessing, 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 C. Grandon & Papegay [10], Merlet [35], localization and map building Jaulin [27], Jaulin et al. [28], biomedicine Cruz & Barahona [13], or the protein folding problem Krippahl & Barahona [32].

Note that solving constrained global optimization problems is in practice 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.

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 bound 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 Neumaier [38]. For filtering, relaxation, branching and other techniques also see. Domes [16, 17], Jaulin [26], Sahinidis & Tawarmalani [40], Y. Lebbah [46].

Filtering techniques which tighten a box – the Cartesian product of intervals defined by the bounds on the variables – are called **constraint propagation** if they are based on single constraints 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 bound 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 techniques.

A number of software packages for solving constraint satisfaction problems make extensive use of constraint propagation. The `Numerica` software Hentenryck [22], Hentenryck et al. [24] 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. The `PaLM` system by Jussien & Barichard [30] uses explanation-based constraint programming, which allows to propagate constraints of the problem, learning from failure and from the solver. The price winning solver `Baron` by Sahinidis & Tawarmalani [41] also uses constraint propagation techniques. Initiated by the development of interval analysis on DAGs (Directed Acyclic Graphs) by Schichl & Neumaier [43], advanced constraint propagation techniques for solving numerical constraint satisfaction problems have been given in Vu et al. [44, 45], which today belong to the most efficient implementations of basic constraint propagation algorithms for individual operations. It is included in the global optimization software platform `COCONUT` Environment [3, 42].

Historically, constraint propagation was pioneered in constraint logic Cleary [12], Older & Vellino [39], first for discrete constraints, later for continuous constraints (see Babichev et al. [4], Benhamou & Older [8], Chen & van Emden [11], F. Benhamou & Hentenryck [19], Hager [20], Hentenryck [22], Hentenryck et al. [23, 24], Hyvönen & Pascale [25], Kearfott [31]), but has also forerunners in presolve techniques in mathematical programming (Anderson & Anderson [1], Lodwick [34]). They can be modeled by narrowing Benhamou [5] 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. [6, 7], Jermann et al. [29]. 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 [38] 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. The new methods are implemented as a part of the `GloptLab` environment (Domes [16, 17]).

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, and 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 only increases the number of variables, but allows one 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.

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 componentwise, 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.

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, while in Section 5, we give an effective method to bound and eliminate bilinear entries from a constraint. This allows the reduction of the non-separable constraints to separable ones, and thus in Section 6 we can introduce the 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.

2.1 Example. 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 53 and 15. Therefore, we find that $\mathbf{f} = [-1, 53]$.

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 $u = b^2/(-4a)$, 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` $= \overline{x}$):

2.2 Algorithm: (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

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

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

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

except that finding the interior extrema is more involved. It can be done with closed formulas for $n \le 5$ (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 [37].

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 [21] (who only treat the solution of a quadratic equation with interval coefficients).

3.1 Example. 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

$$x^2 - 2x + 10 = (x-1)^2 \ge 0,$$

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

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

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

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

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

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

which can be nested such that only two compares 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

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

where

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

These formulas are numerically stable, and can be nested such that only three compares 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 $b \geq 0$, and with rounding upwards if $b \leq 0$. (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 $ac \geq 0$, so that $\Delta \geq 0$ 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,

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

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

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

can be reduced to the previous task since

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

The sets

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

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 $\underline{x}_0 \geq 0$, by negating x, $\mathbf{x}_0$, and $\mathbf{b}$ if $\overline{x}_0 \leq 0$, 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).

3.2 Algorithm: (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.

4.1 Example. 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 $f(x) \in [-1, 53]$, 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

$$f(x) := x^2 - 2x \in [-1, 8], \quad x \in [-2, 3]$$

where the bounds on f and x are tighter as 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,

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

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

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

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

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

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

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

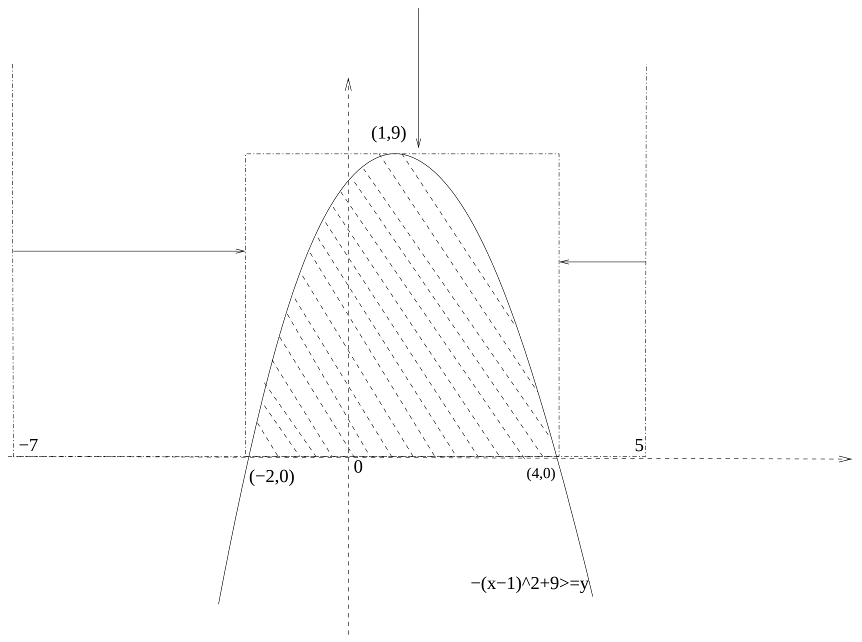

Figure 1: Improving the bound constraints in Example 4.2.

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

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

Then we sum the found intervals and obtain

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

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

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

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

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

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

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

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

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

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, $\gamma_i = \hat{c} - \infty + \infty$ 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

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

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

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

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.

4.3 Example. 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 each for the above three possibilities and put $\hat{c} = 0$. 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

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

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 each case 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 $\hat{\mathbf{x}}_i$ on the variable x_i. If we cut it with the original bound on x_i we obtain $x_i \in \hat{\mathbf{x}}_i \cap \mathbf{x}_i$. Since we approximate all univariate expression $p_i(x_i)$, $i \in \{1, \dots, n\}$, we obtain the new bound constraints

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

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 Non-separable quadratic constraints

In this section we discuss a method for removing a bilinear term from non-separable 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.

5.1 Example. (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

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

For $k = 3$, we find $x_1^2 + x_2^2 + x_3^2 \leq 1$, 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 $k \leq 2$, (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 another paper Domes & Neumaier [18], 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 $x_j x_k$ by the interval evaluation of $\mathbf{x}_j \mathbf{x}_k$ and obtain $2x_j x_k \in [-10, 50]$. In this case (17) reduces to

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

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 $i = 1, \dots, 3$

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

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

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

$$\sum_k (a_k x_k^2 + b_k x_k) + \sum_{\substack{j,k \\ j>k}} b_{jk} x_j x_k \geq c, \quad x \in \mathbf{x}, \tag{20}$$

where the $a_k x_k^2$ are the quadratic, the $b_k x_k$ the linear and $b_{ik} x_i x_k$ the bilinear terms.

As in Example 5.1 (ii), we find rigorous bounds for a bilinear term $b_{jk} x_j x_k$ 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 [36]) and obtain

$$\mathbf{x}_i \mathbf{x}_j = [\min(\underline{\mathbf{x}_i \mathbf{x}_j}, \underline{\mathbf{x}_i}\overline{\mathbf{x}_j}, \overline{\mathbf{x}_i}\underline{\mathbf{x}_j}, \overline{\mathbf{x}_i \mathbf{x}_j}), \max(\underline{\mathbf{x}_i \mathbf{x}_j}, \underline{\mathbf{x}_i}\overline{\mathbf{x}_j}, \overline{\mathbf{x}_i}\underline{\mathbf{x}_j}, \overline{\mathbf{x}_i \mathbf{x}_j})]. \tag{21}$$

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

Alternatively, when a bound for x_j or x_k is infinite or huge, 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 (20), 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 $b_{jk} x_j x_k$ by a multiple of $a_j x_j^2 + a_k x_k^2$; this is the case if, for some constant q_{jk},

$$q_{jk}(a_j x_j^2 + a_k x_k^2) - b_{jk} x_j x_k = d_{jk}(x_j - \beta x_k)^2 \tag{22}$$

with nonnegative d_{jk}. This is achieved for the following choice.

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

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

Then

$$0 \le d_{jk}(x_j - \beta_{jk} x_k)^2 = d_{jk} x_j^2 + \frac{b_{jk}\beta_{jk}}{2} x_k^2 - b_{jk} x_j x_k \tag{23}$$

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

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

and

$$d_{jk}(x_j - \beta_{jk} x_k)^2 = \frac{b_{jk}}{2\beta_{jk}}\frac{a_j}{a_j} x_j^2 + \frac{b_{jk}}{2\beta_{jk}}\frac{a_k}{a_j} x_k^2 - b_{jk} x_j x_k = q_{jk}(a_j x_j^2 + a_k x_k^2) - b_{jk} x_j x_k.$$

$\square$

Since the first method tends to give better bounds on $x_j x_k$ 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, for bilinear terms with nonzero coefficients b_{jk} we proceed as follows. First, we factor the quadratic and the bilinear terms which depend on the variables x_j and x_k from (20) and obtain

$$c(x) + a_k x_k^2 + a_j x_j^2 + b_{jk} x_j x_k + d_{jk}(x_j - \beta_{jk} x_k)^2 \ge c, \tag{24}$$

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

$$c(x) := \sum_{\substack{i \\ i\neq k, i\neq j}} p_i(x_i) + \sum_{\substack{m,i \\ m>i \\ (mi)\neq(jk)}} b_{mi}x_m x_i.$$

Then we handle two different cases:

1. If one of the bounds $\underline{x}_j$, $\overline{x}_j$, $\underline{x}_k$, $\overline{x}_k$ is infinite and both a_k and a_j is negative we quadratically approximate the bilinear term $b_{jk}x_jx_k$. First we add

   $$d_{jk}(x_j - \beta_{jk}x_k)^2 \text{ with } \beta_{jk} := \operatorname{sign}(b_{jk})\sqrt{\frac{a_k}{a_j}}, \quad d_{jk} := \frac{b_{jk}}{2\beta_{jk}}$$

   to the left hand side of (24) and obtain

   $$c(x) + a_kx_k^2 + a_jx_j^2 + b_{jk}x_jx_k + d_{jk}(x_j - \beta_{jk}x_k)^2 \geq c. \tag{25}$$

   Then, by the inequality (23) we have

   $$c(x) + (a_k + d_{jk})x_k^2 + (a_j + \frac{b_{jk}\beta_{jk}}{2})x_j^2 \geq c, \tag{26}$$

   showing that the bilinear term $b_{jk}x_jx_k$ has been eliminated from (24). We have separated the variables x_j and x_k in (25), ending up in

   $$c(x) + \hat{a}_kx_k^2 + \hat{a}_jx_j^2 \geq c, \tag{27}$$

   with the new quadratic coefficients

   $$\hat{a}_k := a_k + d_{jk}, \ \hat{a}_j := a_j + \frac{b_{jk}\beta_{jk}}{2}.$$

2. If all bounds $\underline{x}_j$, $\overline{x}_j$, $\underline{x}_k$, $\overline{x}_k$ are finite we approximate the bilinear term $b_{jk}x_jx_k$ by interval arithmetic. This changes the right hand side of the inequality (24) by the supremum of $b_{jk}x_jx_k$ resulting in

   $$c(x) + a_kx_k^2 + a_jx_j^2 \geq \hat{c}, \tag{28}$$

   where

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

   So we have separated the variables x_j and x_k.

The above method results in the new separable system

$$\sum_k \hat{a}_kx_k^2 + b_kx_k \geq \hat{c}, \quad x \in \mathbf{x}. \tag{29}$$

If we have additional uncertainties $a_k \in \mathbf{a}_k$ and $b_{jk} \in \mathbf{b}_{jk}$ in the constraint coefficients, it is easy to verify that the upper bound of (20) is attained if we set $a_k = \overline{a}_k$ and $b_{jk} = \overline{b}_{jk}$ in all the above formulas.

All above bounds should be computed with upward rounding.

6 The constraint propagation method

The constraint satisfaction problems considered in this paper consist 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

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

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 $A_{i:}q(x) \in \mathbf{F}_i$ $(i = 1 \dots m)$, 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 $m \times n$ 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 the quadratic constraint satisfaction problem in the form

$$Aq(x) \in \mathbf{F}, \quad x \in \mathbf{x}, \quad A \in \mathbf{A}. \tag{30}$$

We now summarize the constraint propagation method for the quadratic constraint satisfaction problem (30).

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 not used by the constraints, and the entries corresponding to the removed variables from the vector $q(x)$. In this step the dimensions of the coefficient matrix A and the box $\mathbf{x}$ may change.

Resolving the two sided constraints. We resolve the two-sided constraints of (30) 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

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

The system of quadratic inequalities

$$A'q(x) \geq c', \quad x \in \mathbf{x}, \quad A' \in \mathbf{A}' := [-A^n, A^p], \tag{31}$$

is another representation of the quadratic constraint satisfaction problem (30). 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. Since the ith row of (31) 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 (20), with

$$a_k := \overline{A}'_{i,kn+k}, \quad b_k := \overline{A}'_{i,k}, \quad b_{jk} := \overline{A}'_{i,jn+k} \text{ and } c := c'_i. \tag{32}$$

we can use the results of Section 5 to remove all bilinear entries from each constraint. We note that in (32) we used the upper bounds to take account of the uncertainties of the constraint coefficients. Depending on the removal method we have applied either the quadratic coefficients or the bound c' or both of them has been changed, ending up in a new system

$$A''q(x) \geq c'', \quad x \in \mathbf{x}, \quad A'' \in \mathbf{A}'' := [-A^n, A^p] \tag{33}$$

In (33) all bilinear coefficients are zero, therefore from this point on, the system is separable.

Forward and backward propagation. Since for the ith row of (33) 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 been found feasible, can apply the backward propagation step (by using the theory from Section 3), which may yield tighter bounds on the variables.

After forward and backward propagating all the constraints we obtain

$$A''q(x) \geq c''', \quad x \in \hat{\mathbf{x}}, \tag{34}$$

with new lower bounds c''' on the constraints and a new bounds $\hat{\mathbf{x}}$ on the variables.

Postprocessing. Finally we merge the inequalities into two sided constraints, and obtain

$$Aq(x) \in \hat{\mathbf{F}}, \quad x \in \hat{\mathbf{x}}, \quad A \in \mathbf{A}. \tag{35}$$

with the original constraint coefficient matrix $\mathbf{A}$,

$$\underline{\hat{F}}_{I_i} = c_i''', \quad i = 1 \dots n_I, \quad \overline{\hat{F}}_{J_j} = -c_j''', \quad j = (n_I + 1) \dots (n_I + n_J)$$

and a new bound constraints $x \in \hat{\mathbf{x}}$.

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] H. Schichl at al. The COCONUT Environment, 2000-2008. URL `http://www.mat.univie.ac.at/coconut-environment`. Software.

[4] 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.

[5] 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.

[6] 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`.

[7] 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`.

[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 Hentenrck, 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] D. Daney C. Grandon 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`.

[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 et al, editor, *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. Verified global optimization with Gloptlab, 2007. URL `http://www.mat.univie.ac.at/~dferi/ICIAM07.pdf`.

[17] F. Domes. GloptLab, a configurable framework for the rigorous global solution of quadratic constraint satisfaction problems. in preparation, 2007-2008.

[18] F. Domes and A. Neumaier. Directed cholesky factorizations and applications. submitted, 2008.

[19] D. McAllister F. Benhamou and P. Van Hentenryck. Clp(intervals) revisited. In *Proc. International Symposium on Logic Programming*, pp. 1–21 or 124–138. MIT Press, 1994.

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

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

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

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

[24] P. Van Hentenryck, L. Michel, and Y. Deville. *Numerica. A Modeling Language for Global Optimization.* MIT Press, 1997.

[25] 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.

[26] 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`.

[27] 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/\%7Eneum/glopt/gicolag/talks/jaulin.pdf`.

[28] 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`.

[29] 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.

[30] N. Jussien and V. Barichard. The PaLM system: explanation-based constraint programming. In *Proceedings of TRICS: Techniques foR Implementing Constraint programming Systems, a post-conference workshop of CP 2000*, pp. 118–133, September 2000. URL `http://www.emn.fr/jussien/publications/jussien-WCP00.pdf`.

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

[35] 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`.

[36] A. Neumaier. *Interval Methods for Systems of Equations*, vol. 37 of *Encyclopedia of Mathematics and its Applications*. Cambridge Univ. Press, Cambridge, 1990.

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

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

[39] 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.

[40] 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.

[41] 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`.

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

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

[44] Xuan-Ha Vu, H. Schichl, and D. Sam-Haroud. Using directed acyclic graphs to coordinate 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`.

[45] 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. URL `http://www.mat.univie.ac.at/~herman/papers/FBPD-Hermann.pdf`. to appear.

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