



# A Contractor Based on Convex Interval Taylor

Ignacio Araya, Gilles Trombettoni, Bertrand Neveu

► **To cite this version:**

Ignacio Araya, Gilles Trombettoni, Bertrand Neveu. A Contractor Based on Convex Interval Taylor. [Research Report] RR-7887, INRIA. 2012, pp.23. <hal-00673447>

**HAL Id: hal-00673447**

**<https://hal.inria.fr/hal-00673447>**

Submitted on 23 Feb 2012

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



# A Contractor Based on Convex Interval Taylor

Ignacio Araya, Gilles Trombettoni, Bertrand Neveu

**RESEARCH  
REPORT**

**N° 7887**

February 2012

Project-Team COPRIN

ISRN INRIA/RR--7887--FR+ENG

ISSN 0249-6399





## A Contractor Based on Convex Interval Taylor

Ignacio Araya<sup>\*</sup>, Gilles Trombettoni<sup>†</sup>, Bertrand Neveu<sup>‡</sup>

Project-Team COPRIN

Research Report n° 7887 — February 2012 — 23 pages

### Abstract:

Interval Taylor has been proposed in the sixties by the interval analysis community for relaxing non-convex continuous constraint systems. However, it generally produces a non-convex relaxation of the solution set. A simple way to build a convex polyhedral relaxation is to select a *corner* of the studied domain/box as expansion point of the interval Taylor form, instead of the usual midpoint. The idea has been proposed by Neumaier to produce a sharp range of a single function and by Lin and Stadtherr to handle  $n \times n$  (square) systems of equations.

This paper presents an interval Newton-like operator, called *X-Newton*, that iteratively calls this interval convexification based on an endpoint interval Taylor. This general-purpose contractor uses no preconditioning and can handle any system of equality and inequality constraints. It uses Hansen's variant to compute the interval Taylor form and uses two opposite corners of the domain for every constraint.

The *X-Newton* operator can be rapidly encoded, and produces good speedups in constrained global optimization and non-convex constraint satisfaction. First experiments compare *X-Newton* with affine arithmetic.

**Key-words:** intervals, Taylor, convex polyhedral relaxation, global optimization

---

<sup>\*</sup> UTFSM, Chile. Email: [iaraya@inf.utfsm.cl](mailto:iaraya@inf.utfsm.cl)

<sup>†</sup> INRIA, I3S, Université Nice–Sophia, France. Email: [Gilles.Trombettoni@inria.fr](mailto:Gilles.Trombettoni@inria.fr)

<sup>‡</sup> Imagine LIGM Université Paris–Est, France. Email: [Bertrand.Neveu@enpc.fr](mailto:Bertrand.Neveu@enpc.fr)

**RESEARCH CENTRE  
SOPHIA ANTIPOLIS – MÉDITERRANÉE**

2004 route des Lucioles - BP 93  
06902 Sophia Antipolis Cedex

## Un contracteur basé sur une forme de Taylor sur intervalles convexe

**Résumé :** Une forme de Taylor sur intervalles a été proposée dans les années 1960 par la communauté de l'analyse par intervalles pour relaxer les systèmes de contraintes continues non convexes. Cependant, celle-ci produit généralement une relaxation non convexe de l'espace solution. Un moyen simple de produire une relaxation polyédrale convexe est de sélectionner un *coin* du domaine/boîte étudié comme point d'expansion de la forme de Taylor, en place du point milieu couramment utilisé. L'idée a été proposée par Neumaier pour calculer un intervalle étroit d'une fonction d'inclusion et par Lin et Stadtherr pour traiter des systèmes d'équations  $n \times n$  (carrés).

Cet article présente un opérateur de Newton sur intervalles, appelé X-Newton, qui appelle itérativement cette convexification utilisant une forme de Taylor sur intervalles extrémales. Ce contracteur généraliste n'utilise pas de préconditionnement et peut traiter pratiquement n'importe quel système de contraintes d'égalité et d'inégalité. Il utilise la variante de E. R. Hansen pour calculer la forme de Taylor et utilise deux coins opposés du domaine pour chaque contrainte.

L'opérateur X-Newton peut se coder rapidement et apporte d'importantes accélérations en optimisation globale sous contraintes et en satisfaction de contraintes non convexes. Des premières expérimentations comparent X-Newton à l'arithmétique affine.

**Mots-clés :** intervalles, Taylor, relaxation polyédrale convexe, optimisation globale

## Contents

<b>1</b>	<b>Motivation</b>	<b>3</b>
<b>2</b>	<b>Background</b>	<b>5</b>
<b>3</b>	<b>Extremal interval Taylor form</b>	<b>7</b>
3.1	Corner selection for a tight convexification . . . . .	7
3.2	Preliminary interval linearization . . . . .	10
<b>4</b>	<b>eXtremal interval Newton</b>	<b>10</b>
4.1	X-Newton iteration . . . . .	11
4.2	X-Newton . . . . .	11
<b>5</b>	<b>Experiments</b>	<b>12</b>
5.1	Experiments in constrained global optimization . . . . .	12
5.2	Experiments in constraint satisfaction . . . . .	15
<b>6</b>	<b>Conclusion</b>	<b>18</b>
<b>A</b>	<b>X-Newton and square systems of equations</b>	<b>21</b>
A.1	Standard interval Newton . . . . .	21
A.2	I-Newton and X-Newton for square systems . . . . .	21
<b>B</b>	<b>Existence test for systems of inequality constraints</b>	<b>22</b>
B.1	Adaptation to equality constraints . . . . .	23

## 1 Motivation

Interval B&B algorithms are used to solve continuous constraint systems and to handle constrained global optimization problems in a *reliable* way, i.e., they provide an optimal solution and its cost with a bounded error or a proof of infeasibility. The functions taken into account may be non-convex and can include many (piecewise) differentiable operators like arithmetic operators (+, −, ·, /), power, log, exp, sinus, etc.

Interval Newton is an operator often used by interval methods to contract/filter the search space [14]. The interval Newton operator uses an *interval Taylor* form to iteratively produce a linear system with interval coefficients. The main issue is that this system is *not* convex. Restricted to a single constraint, it forms a non-convex cone (a “butterfly”), as illustrated in Fig. 1-left. An  $n$ -dimensional constraint system is relaxed by an intersection of butterflies that is not convex either. (Examples can be found in [24, 15, 23].) Contracting optimally a box containing this non-convex relaxation has been proven to be NP-hard [16]. This explains why the interval analysis community has worked a lot on this problem for decades [14].

Only a few polynomial time solvable subclasses have been studied. The most interesting one has been first described by Oettli and Prager in the sixties [27] and occurs when the variables are all non-negative or non-positive. Unfortunately, when the Taylor expansion point is chosen strictly inside the domain (the midpoint typically), the studied box must be previously split into  $2^n$  sub-problems/quadrants before falling in this interesting subclass [1, 4, 7]. Hansen and Bliek independently proposed a sophisticated and beautiful algorithm for avoiding explicitly handling the  $2^n$  quadrants [13, 6].

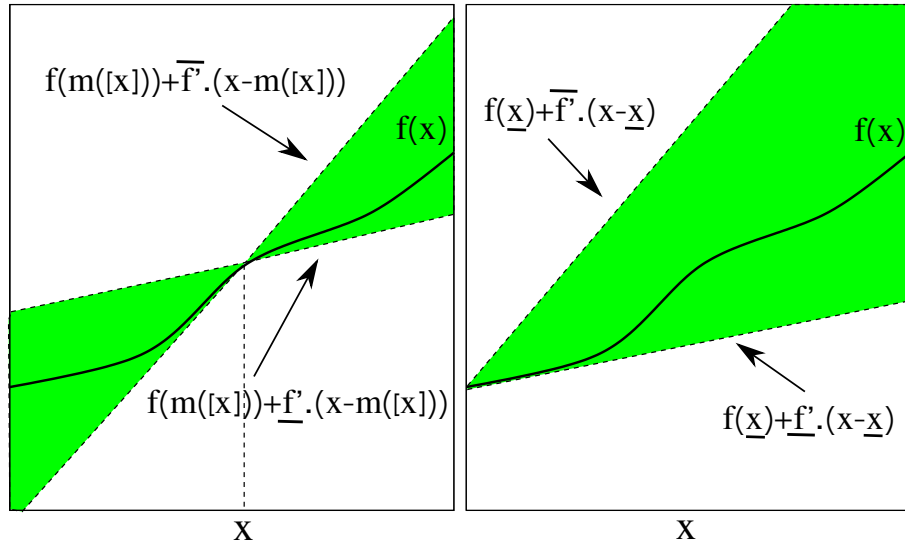


Figure 1: Relaxation of a function  $f$  over the real numbers by a function  $g : \mathbb{R} \rightarrow \mathbb{IR}$  using an interval Taylor form (graph in gray). **Left:** Midpoint Taylor form, using a midpoint evaluation  $f(m([x]))$ , the maximum derivative  $\overline{f'}$  of  $f$  inside the interval  $[x]$  and the minimum derivative  $\underline{f'}$ . **Right:** Extremal Taylor form, using an endpoint evaluation  $f(\underline{x})$ ,  $\overline{f'}$  and  $\underline{f'}$ .

However, the method is restricted to  $n \times n$  (square) systems of equations (no inequalities).<sup>1</sup> Also, the method requires the system be first preconditioned (i.e., the interval Jacobian matrix must be multiplied by the inverse matrix of the domain midpoint). The preconditioning has a cubic time complexity, implies an overestimate of the relaxation and requires non-singularity conditions often met only on small domains, at the bottom of the search tree.

In 2004, Lin & Stadtherr [19] proposed to select a *corner* of the studied box, instead of the usual midpoint. Graphically, it produces a convex cone, as shown in Fig. 1-right. The main drawback of this *extremal* interval Taylor form is that it leads to a larger system relaxation surface. The main virtue is that the solution set belongs to a unique quadrant and is convex. It is a polytope that can be (box) hulled in polynomial-time by a linear programming (LP) solver: two calls to an LP solver compute the minimum and maximum values in this polytope for each of the  $n$  variables (see Section 4). Upon this extremal interval Taylor, they have built an interval Newton restricted to square  $n \times n$  systems of *equations* for which they had proposed in a previous work a specific preconditioning. They have presented a corner selection heuristic optimizing their preconditioning. The selected corner is common to all the constraints.

The idea of selecting a corner as Taylor expansion point is mentioned, in dimension 1, by A. Neumaier (see page 60 and Fig. 2.1 in [24]) for computing a range enclosure (see Def. 1) of a univariate function. Neumaier calls this the *linear boundary value form*. The idea has been exploited by Messine and Laganouelle for lower bounding the objective function in a Branch & Bound algorithm for unconstrained global optimization [21].

<sup>1</sup>It could be applied to under-constrained systems of equations by using techniques described in [11]

McAllester et al. also mention this idea in [20] (end of page 2) for finding cuts of the box in constraint systems. At page 211 of Neumaier’s book [24], the step (4) of the presented pseudo-code also uses an endpoint interval Taylor form for contracting a system of equations.<sup>2</sup>

## Contributions

We present in this paper a new contractor, called X-Newton (for eXtremal interval Newton), that iteratively achieves an interval Taylor form on a corner of the studied domain. X-Newton does not require the system be preconditioned and can thus reduce the domains higher in the search tree. It can treat well-constrained systems as well as under-constrained ones (with fewer equations than variables and with inequalities), as encountered in constrained global optimization. The only limit is that the domain must be bounded, although the considered intervals, i.e., the initial search space, can be very large.

This paper experimentally shows that such a contractor is crucial in constrained global optimization and is also useful in continuous constraint satisfaction where it makes the whole solving strategy more robust.

After the background introduced in the next section, we show in Section 3 that the choice of the best expansion corner for any constraint is an NP-hard problem and propose a simple selection policy choosing two opposite corners of the box. Tighter interval partial derivatives are also produced by Hansen’s recursive variant of interval Taylor. Section 4 details the extremal interval Newton operator that iteratively computes a convex interval Taylor form. Section 5 highlights the benefits of X-Newton in satisfaction and constrained global optimization problems.

This work provides an alternative to the two existing *reliable* (interval) convexification methods used in global optimization. The Quad [18, 17] method is an interval *reformulation-linearization technique* that produces a convex polyhedral approximation of the quadratic terms in the constraints. *Affine arithmetic* produces a polytope by replacing in the constraint expressions every basic operator by specific affine forms [10, 33, 3]. It has been recently implemented in an efficient interval B&B [26]. Experiments provide a first comparison between this affine arithmetic and the corner-based interval Taylor.

## 2 Background

Intervals allow reliable computations on computers by managing floating-point bounds and outward rounding.

### Intervals

An **interval**  $[x_i] = [x_i, \bar{x}_i]$  defines the set of reals  $x_i$  s.t.  $x_i \leq x_i \leq \bar{x}_i$ , where  $x_i$  and  $\bar{x}_i$  are floating-point numbers.  $\mathbb{IR}$  denotes the set of all intervals. The size or **width** of  $[x_i]$  is  $w([x_i]) = \bar{x}_i - x_i$ . A **box**  $[x]$  is the Cartesian product of intervals  $[x_1] \times \dots \times [x_i] \times \dots \times [x_n]$ .

<sup>2</sup>The aim is not to produce a convex polyhedral relaxation (which is not mentioned), but to use as expansion point the farthest point in the domain from a current point followed by the algorithm. The contraction is not obtained by calls to an LP solver but by the general purpose Gauss-Seidel without taking advantage of the convexity.



Its width is defined by  $\max_i w([x_i])$ .  $m([x])$  denotes the middle of  $[x]$ . The **hull** of a subset  $S$  of  $\mathbb{R}^n$  is the smallest  $n$ -dimensional box enclosing  $S$ .

*Interval arithmetic* [22] has been defined to extend to  $\mathbb{IR}$  elementary functions over  $\mathbb{R}$ . For instance, the interval sum is defined by  $[x_1] + [x_2] = [x_1 + x_2, \bar{x}_1 + \bar{x}_2]$ . When a function  $f$  is a composition of elementary functions, an *extension* of  $f$  to intervals must be defined to ensure a conservative image computation.

**Definition 1 (Extension of a function to  $\mathbb{IR}$ ; inclusion function; range enclosure)**

Consider a function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ .

$[f] : \mathbb{IR}^n \rightarrow \mathbb{IR}$  is said to be an **extension** of  $f$  to intervals iff:

$$\begin{aligned} \forall [x] \in \mathbb{IR}^n \quad [f]([x]) &\supseteq \{f(x), x \in [x]\} \\ \forall x \in \mathbb{R}^n \quad f(x) &= [f](x) \end{aligned}$$

The **natural extension**  $[f]_N$  of a real function  $f$  corresponds to the mapping of  $f$  to intervals using interval arithmetic. The outer and inner interval linearizations proposed in this paper are related to the first-order **interval Taylor extension** [22], defined as follows:

$$[f]_T([x]) = f(\dot{x}) + \sum_i [a_i] \cdot ([x_i] - \dot{x}_i)$$

where  $\dot{x}$  denotes any point in  $[x]$ , e.g.,  $m([x])$ , and  $[a_i]$  denotes  $\left[\frac{\partial f}{\partial x_i}\right]_N([x])$ .

Equivalently, we have:  $\forall x \in [x], [f]_T([x]) \leq f(x) \leq \overline{[f]_T([x])}$ .

**Example.** Consider  $f(x_1, x_2) = 3x_1^2 + x_2^2 + x_1x_2$  in the box  $[x] = [-1, 3] \times [-1, 5]$ . The natural evaluation provides:  $[f]_N([x_1], [x_2]) = 3[-1, 3]^2 + [-1, 5]^2 + [-1, 3][-1, 5] = [0, 27] + [0, 25] + [-5, 15] = [-5, 67]$ . The partial derivatives are:  $\frac{\partial f}{\partial x_1}(x_1, x_2) = 6x_1 + x_2$ ,  $\left[\frac{\partial f}{\partial x_1}\right]_N([-1, 3], [-1, 5]) = [-7, 23]$ ,  $\frac{\partial f}{\partial x_2}(x_1, x_2) = x_1 + 2x_2$ ,  $\left[\frac{\partial f}{\partial x_2}\right]_N([x_1], [x_2]) = [-3, 13]$ . The interval Taylor evaluation with  $\dot{x} = m([x]) = (1, 2)$  yields:  $[f]_T([x_1], [x_2]) = 9 + [-7, 23][-2, 2] + [-3, 13][-3, 3] = [-76, 94]$ .

## A simple convexification based on interval Taylor

Consider a function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  defined on a domain  $[x]$ , and the inequality constraint  $f(x) \leq 0$ . For any variable  $x_i \in x$ , let us denote  $[a_i]$  the interval partial derivative  $\left[\frac{\partial f}{\partial x_i}\right]_N([x])$ . The first idea is to lower tighten  $f(x)$  with one of the following interval linear forms that hold for all  $x$  in  $[x]$ .

$$f(\underline{x}) + \underline{a}_1 y_1^l + \dots + \underline{a}_n y_n^l \leq f(x) \tag{1}$$

$$f(\bar{x}) + \bar{a}_1 y_1^r + \dots + \bar{a}_n y_n^r \leq f(x) \tag{2}$$

where:  $y_i^l = x_i - \underline{x}_i$  and  $y_i^r = x_i - \bar{x}_i$ .

A *corner* of the box is chosen:  $\underline{x}$  in form (1) or  $\bar{x}$  in form (2). When applied to a set of inequality and equality<sup>3</sup> constraints, we obtain a polytope enclosing the solution set.

The correctness of relation (1) – see for instance [31, 19] – lies on the simple fact that any variable  $y_i^l$  is non-negative since its domain is  $[0, d_i]$ , with  $d_i = w([y_i^l]) = w([x_i]) = \bar{x}_i - \underline{x}_i$ . Therefore, minimizing each term  $[a_i]y_i^l$  for any point  $y_i^l \in [0, d_i]$  is

<sup>3</sup>An equation  $f(x) = 0$  can be viewed as two inequality constraints:  $0 \leq f(x) \leq 0$ .

obtained with  $\underline{a}_i$ . Symmetrically, relation (2) is correct since  $y_i^r \in [-d_i, 0] \leq 0$ , and the minimal value of a term is obtained with  $\bar{a}_i$ .

Note that, even though the polytope computation is safe, the floating-point round-off errors made by the LP solver could render the hull of the polytope unsafe. A cheap post-processing proposed in [25], using interval arithmetic, is added to guarantee that no solution is lost by the Simplex algorithm.

### 3 Extremal interval Taylor form

#### 3.1 Corner selection for a tight convexification

Relations (1) and (2) consider two specific corners of the box  $[x]$ . We can remark that every other corner of  $[x]$  is also suitable. In other terms, for every variable  $x_i$ , we can indifferently select one of both bounds of  $[x_i]$  and combine them in a combinatorial way: either  $\underline{x}_i$  in a term  $\underline{a}_i(x_i - \underline{x}_i)$ , like in relation (1), or  $\bar{x}_i$  in a term  $\bar{a}_i(x_i - \bar{x}_i)$ , like in relation (2).

A natural question then arises: Which corner  $x^c$  of  $[x]$  among the  $2^n$ -set  $X^c$  ones produces the tightest convexification? If we consider an inequality  $f(x) \leq 0$ , we want to compute a hyperplane  $f^l(x)$  that approximates the function, i.e., for all  $x$  in  $[x]$  we want:  $f^l(x) \leq f(x) \leq 0$ .

Following the standard policy of linearization methods, for every inequality constraint, we want to select a corner  $x^c$  whose corresponding hyperplane is the closest to the non-convex solution set, i.e., adds the smallest volume. This is exactly what represents Expression (3) that maximizes the Taylor form for *all* the points  $x = \{x_1, \dots, x_n\} \in [x]$  and adds their different contributions: one wants to select a corner  $x^c$  from the set of corners  $X^c$  such that:

$$\max_{x^c \in X^c} \int_{x_1=\underline{x}_1}^{\bar{x}_1} \dots \int_{x_n=\underline{x}_n}^{\bar{x}_n} (f(x^c) + \sum_i z_i) dx_n \dots dx_1 \quad (3)$$

where:  $z_i = \bar{a}_i(x_i - \bar{x}_i)$  iff  $x_i^c = \bar{x}_i$ , and  $z_i = \underline{a}_i(x_i - \underline{x}_i)$  iff  $x_i^c = \underline{x}_i$ .

Since:

- $f(x^c)$  is independent from the  $x_i$  values,
- any point  $z_i$  depends on  $x_i$  but does not depend on  $x_j$  (with  $j \neq i$ ),
- $\int_{\underline{x}_i}^{\bar{x}_i} \underline{a}_i(x_i - \underline{x}_i) dx_i = \underline{a}_i \int_{y_i=0}^{d_i} y_i dy_i = \underline{a}_i 0.5 d_i^2$ ,
- $\int_{\underline{x}_i}^{\bar{x}_i} \bar{a}_i(x_i - \bar{x}_i) dx_i = \bar{a}_i \int_{-d_i}^0 y_i dy_i = -0.5 \bar{a}_i d_i^2$ ,

Expression (3) is equal to:

$$\max_{x^c \in X^c} \prod_i d_i f(x^c) + \prod_i d_i \sum_i 0.5 a_i^c d_i$$

where  $d_i = w([x_i])$  and  $a_i^c = \underline{a}_i$  or  $a_i^c = -\bar{a}_i$ .

We simplify by the positive factor  $\prod_i d_i$  and obtain:

$$\max_{x^c \in X^c} f(x^c) + 0.5 \sum_i a_i^c d_i \quad (4)$$

### Tightest corner convexification is NP-hard

Unfortunately, we can prove that this maximization problem (4) is NP-hard. The following lemma underlines that the difficult part is to maximize  $f(x^c)$ .

**Lemma 1** *Consider a polynomial function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , with rational coefficients, and defined on a domain  $[x] = [0, 1]^n$ . Let  $X^c$  be the  $2^n$ -set of corners, i.e., in which every element is a bound 0 or 1. Then,*

*$\max_{x^c \in X^c} -f(x^c)$  (or  $\min_{x^c \in X^c} f(x^c)$ ) is an NP-hard problem.*

The result is probably well-known but we are interested here in the reduction.

**Proof.** We prove that the (minimization) problem of finding a corner  $x^c \in X^c$  such that  $f(x^c) \leq B$  (where  $B$  is a rational bound)<sup>4</sup> is as hard as the well-known NP-complete 3SAT problem. The polynomial reduction from a 3SAT instance  $I$  to a corner selection instance  $I'$  is the following:

- An instance  $I$  of 3SAT is given by a set of  $n$  boolean variables  $\{x_1, \dots, x_i, \dots, x_n\}$  and a BNF boolean formula, i.e., a conjunction of clauses  $C_I = \bigwedge_j (l_1^j \vee l_2^j \vee l_3^j)$ , where  $l_k^j$  denotes a positive literal  $x_i$  or a negative literal  $\neg x_i$ .
- For every boolean variable  $x_i$  in  $I$ , a rational variable  $x'_i$  is generated in  $I'$  with domain  $[0, 1]$ .
- A boolean formula  $C_I$  is reduced to a polynomial inequality made of a sum of products:  $\sum_j (x_1^j x_2^j x_3^j) \leq 0$ . For every clause  $c_j = (l_1^j \vee l_2^j \vee l_3^j)$  of  $C_I$ , we generate a term  $(x_1^j x_2^j x_3^j)$  where:
  - $x_k^j = 1 - x'_i$  if  $l_k^j = x_i$  is a positive literal in  $c_j$ ,
  - $x_k^j = x'_i$  if  $l_k^j = \neg x_i$  is a negative literal in  $c_j$ .
- Note that we have chosen the bound  $B = 0$ .

It is straightforward (a) to check that this transformation is polynomial, (b) to check in polynomial-time the existence of a solution of  $I'$  and (c) that a solution of an instance  $I$  is equivalent to a solution of an instance  $I'$ . Indeed:

- A boolean variable  $x_i$  is true (resp. false) iff  $x'_i = 1$  (resp.  $x'_i = 0$ ).
- A literal in a clause  $c_j$  is true iff the corresponding term  $x_1^j x_2^j x_3^j = 0$ .
- The conjunction  $C_I$  is satisfiable iff all terms in  $I'$  are null ( $f(x^c) \leq 0$ ).

□

On the other hand, it is easy to maximize the other term  $0.5 \sum_i a_i^c d_i$  in Expression (4) by selecting the maximum value among  $\underline{a}_i$  and  $-\bar{a}_i$  in every term.

The difficulty is thus to determine the computational complexity of the problem (4) that combines  $f(x^c)$  (NP-hard) and  $0.5 \sum_i a_i^c d_i$  (in  $P$ ). In order to prove the NP-hardness of the problem (4), our first (failed) idea was to achieve a polynomial transformation in which the derivative part  $0.5 \sum_i a_i^c d_i$  would be always negligible over its counterpart in  $f(x^c)$ . Instead, we propose a polynomial reduction in which the derivative part is constant, i.e.,  $\forall_i \underline{a}_i = -\bar{a}_i$ . Thus:

<sup>4</sup>We “restrict” the class to polynomial functions, otherwise the corresponding decision problem would not belong to NP. Indeed, verifying the satisfaction of a constraint with, e.g., trigonometric operators cannot be achieved in polynomial-time due to considerations related to floating-point calculation.

**Proposition 1** (Corner selection is NP-hard)

Consider a polynomial <sup>5</sup>  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , with rational coefficients, and defined on a domain  $[x] = [0, 1]^n$ . Let  $X^c$  be the  $2^n$ -set of corners, i.e., in which every component is a bound 0 or 1. Then,

$$\begin{aligned} & \max_{x^c \in X^c} - (f(x^c) + 0.5 \sum_i a_i^c d_i) \\ & \text{(or } \min_{x^c \in X^c} f(x^c) + 0.5 \sum_i a_i^c d_i) \end{aligned}$$

is an NP-hard problem.

**Proof.** The polynomial reduction have similarities with the reduction shown in Lemma 1. The main difference is that we consider a subclass of 3SAT, called here BALANCED-3SAT. In an instance of BALANCED-3SAT, each boolean variable  $x_i$  occurs  $n_i$  times in a negative literal and  $n_i$  times in a positive literal. We know that BALANCED-3SAT is NP-complete thanks to the dichotomy theorem by Thomas J. Schaefer who identified the only 6 subclasses of SAT that are in  $P$  [29]. BALANCED-3SAT does not belong to none of these 6 subclasses.<sup>6</sup>

Considering  $f(x^c) + 0.5 \sum_i a_i^c d_i \leq B$ , a second difference with Lemma 1 is the chosen bound  $B$ . We choose  $B = 0.5 \sum_i d_i (-n_i) = -0.5 \sum_i n_i$  (recall that  $\forall i, d_i = 1$ ).

It is less trivial to check that a solution of an instance  $I$  of BALANCED-3SAT is equivalent to a solution of an instance  $I'$  of  $f(x^c) + 0.5 \sum_i a_i^c d_i \leq -0.5 \sum_i n_i$ . Each term  $x_1^j x_2^j x_3^j$  of  $I'$  implies a partial derivative  $\frac{\partial f}{\partial x_i^j}([x])$  equal to 0 if  $x_i^j$  does not appear in the term, equal to  $[-1, 0]$  if  $x_i$  appears as a positive literal in  $I$  (i.e.,  $x_k^j = (1 - x_i^j)$ ) and  $[-1, 0] = -1 [0, 1] [0, 1]$ , and equal to  $[0, 1]$  if  $x_i$  appears as a negative literal (i.e.,  $x_k^j = x_i^j$  and  $[0, 1] = 1 [0, 1] [0, 1]$ ). Thus, by adding all these intervals in the different terms, we obtain  $[a_i] = [-n_i, n_i]$  and thus  $\forall_i \underline{a}_i = -\bar{a}_i \square$

## Using two opposite corners

Even more annoying is that experiments presented in Section 5 suggest that the criterion (4) is not relevant in practice. Indeed, even if the best corner was chosen (by an oracle), the gain in box contraction brought by this strategy w.r.t. a random choice of corner would be not significant. This renders pointless the search for an efficient and fast corner selection heuristic.

This study suggests that this criterion is not relevant and leads to explore another criterion. We should notice that when a hyperplane built by endpoint interval Taylor removes some inconsistent parts from the box, the inconsistent subspace more often includes the selected corner  $x_c$  because the approximation at this point is exact. However, the corresponding criterion includes terms mixing variables coming from all the dimensions simultaneously, and makes difficult the design of an efficient corner selection heuristic. This qualitative analysis nevertheless provides us rationale to adopt the following policy.

To obtain a better contraction, it is also possible to produce *several*, i.e.,  $c$ , linear expressions lower tightening a given constraint  $f(x) \leq 0$ . Applied to the whole system

<sup>5</sup>We cannot prove anything on more complicated, e.g., transcendental, functions that make the problem undecidable.

<sup>6</sup>A straightforward reduction from 3SAT to BALANCED-3SAT could also be followed: add to the 3SAT instance  $d$  “dummy” clauses, one for each “missing” literal; for one such literal, e.g.,  $\neg x_i$ , the corresponding clause is  $\neg x_i \vee b_j \vee \neg b_{j-1}$ ; the  $b_j$  variables ( $j \in \{1 \dots d\}$ ) are dummy additional boolean variables (appearing  $d$  times as a negative literal and  $d$  times as a positive literal in round-robin...).

with  $m$  inequalities, the obtained polytope corresponds to the intersection of these  $cm$  half-spaces. Experiments (see Section 5.2) suggest that generating two hyperplanes (using two corners) yields a good ratio between contraction (gain) and number of hyperplanes (cost). Also, choosing opposite corners tends to minimize the redundancy between hyperplanes since the hyperplanes remove from the box preferably the search subspaces around the selected corners.

Note that, for managing several corners simultaneously, an expanded form must be adopted to put the whole linear system in the form  $Ax - b$  before running the Simplex algorithm. For instance, if we want to lower tighten a function  $f(x)$  by expressions (1) and (2) simultaneously, we must rewrite:

1.  $f(\underline{x}) + \sum_i \underline{a}_i(x_i - \underline{x}_i) = f(\underline{x}) + \sum_i \underline{a}_i x_i - \underline{a}_i \underline{x}_i = \sum_i \underline{a}_i x_i + f(\underline{x}) - \sum_i \underline{a}_i \underline{x}_i$
2.  $f(\bar{x}) + \sum_i \bar{a}_i(x_i - \bar{x}_i) = f(\bar{x}) + \sum_i \bar{a}_i x_i - \bar{a}_i \bar{x}_i = \sum_i \bar{a}_i x_i + f(\bar{x}) - \sum_i \bar{a}_i \bar{x}_i$

Also note that, to remain safe, the computation of constant terms  $\underline{a}_i \underline{x}_i$  (resp.  $\bar{a}_i \bar{x}_i$ ) must be achieved with degenerate intervals:  $[\underline{a}_i, \underline{a}_i] [\underline{x}_i, \underline{x}_i]$  (resp.  $[\bar{a}_i, \bar{a}_i] [\bar{x}_i, \bar{x}_i]$ ).

### 3.2 Preliminary interval linearization

Recall that the linear forms (1) and (2) proposed by Neumaier and Lin & Stadtherr use the bounds of the interval *gradient*, given by  $\forall i \in \{1, \dots, n\}, [a_i] = \left[ \frac{\partial f}{\partial x_i} \right]_N([x])$ .

Eldon R. Hansen proposed in 1968 a variant in which the Taylor form is achieved recursively, one variable after the other [12, 14]. The variant amounts in producing the following tighter interval coefficients:

$$\forall i \in \{1, \dots, n\}, [a_i] = \left[ \frac{\partial f}{\partial x_i} \right]_N([x_1] \times \dots \times [x_i] \times x_{i+1} \times \dots \times x_n)$$

where  $x_j \in [x_j]$ , e.g.,  $x_j = m([x_j])$ .

By following Hansen's recursive principle, we can produce Hansen's variant of the form (1), for instance, in which the scalar coefficients  $\underline{a}_i$  are:

$$\forall i \in \{1, \dots, n\}, \underline{a}_i = \left[ \frac{\partial f}{\partial x_i} \right]_N(\underline{[x_1]} \times \dots \times \underline{[x_i]} \times x_{i+1} \times \dots \times x_n).$$

We end up with an X-Taylor algorithm (X-Taylor stands for *eXtremal interval Taylor*) producing 2 linear expressions lower tightening a given function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  on a given domain  $[x]$ . The first corner is randomly selected, the second one is opposite to the first one.

## 4 eXtremal interval Newton

We first describe in Section 4.1 an algorithm for computing the (box) hull of the polytope produced by X-Taylor. We then detail in Section 4.2 how this X-Newton procedure is iteratively called in the X-Newton algorithm until a quasi-fixpoint is reached in terms of contraction.

## 4.1 X-Newton iteration

Algorithm 1 describes a well-known algorithm used in several solvers (see for instance [18, 3]). A specificity here is the use of a corner-based interval Taylor form (X-Taylor) for computing the polytope.

---

**Algorithm 1** X-NewIter ( $f, x, [x]$ ):  $[x]$

---

```

for  $j$  from 1 to  $m$  do
  polytope  $\leftarrow$  polytope  $\cup$  {X-Taylor( $f_j, x, [x]$ )}
end for
for  $i$  from 1 to  $n$  do
  /* Two calls to a Simplex algorithm: */
   $\underline{x}_i \leftarrow$  min  $x_i$  subject to polytope
   $\overline{x}_i \leftarrow$  max  $x_i$  subject to polytope
end for
return  $[x]$ 

```

---

All the constraints appear as inequality constraints  $f_j(x) \leq 0$  in the vector/set  $f = (f_1, \dots, f_j, \dots, f_m)$ .  $x = (x_1, \dots, x_i, \dots, x_n)$  denotes the set of variables with domains  $[x]$ .

The first loop on the constraints builds the polytope while the second loop on the variables contracts the domains, without loss of solution, by calling a Simplex algorithm twice per variable. When embedded in an interval B&B for constrained global optimization, X-NewIter is modified to also compute a lower bound of the objective in the current box: an additional call to the Simplex algorithm minimizes an X-Taylor relaxation of the objective on the same polytope.

Heuristics mentioned in [3] indicate in which order the variables can be handled, thus avoiding in practice to call  $2n$  times the Simplex algorithm.

## 4.2 X-Newton

The procedure X-NewIter allows one to build the X-Newton operator (see Algorithm 2). Consider first the basic variant in which CP-contractor =  $\perp$ . X-NewIter

---

**Algorithm 2** X-Newton ( $f, x, [x], \text{ratio\_fp}, \text{CP-contractor}$ ):  $[x]$

---

```

repeat
   $[x]_{\text{save}} \leftarrow [x]$ 
   $[x] \leftarrow$  X-NewIter ( $f, x, [x]$ )
  if CP-contractor  $\neq \perp$  and gain( $[x], [x]_{\text{save}}$ )  $> 0$  then
     $[x] \leftarrow$  CP-contractor( $f, x, [x]$ )
  end if
until empty( $[x]$ ) or gain( $[x], [x]_{\text{save}}$ )  $<$  ratio_fp
return  $[x]$ 

```

---

is iteratively run until a quasi fixed-point is reached in terms of contraction. More precisely, ratio\_fp is a user-defined percentage of the interval size and:

$$\text{gain}([x'], [x]) := \max_i \frac{w([x_i]) - w([x'_i])}{w([x_i])}.$$

We also permit the use of a contraction algorithm, typically issued from constraint programming, inside the main loop. For instance, if the user specifies CP-contractor=Mohc

and if `X-NewIter` reduces the domain, then the `Mohc` constraint propagation algorithm [2] can further contract the box, before waiting for the next choice point. The guard  $\text{gain}([x], [x]_{\text{save}}) > 0$  guarantees that `CP-contractor` will not be called twice if `X-NewIter` does not contract the box.

Note that the *X-Newton* operator does not require the system be preconditioned so that this contractor can cut branches early during the tree search (see Section 5.2). In this sense, it is closer to a reliable convexification method like `Quad` [18, 17] or affine arithmetic [26].

## Quadratic convergence

Compared to a standard interval Newton, a drawback of *X-Newton* is the loss of quadratic convergence when the current box belongs to a convergence basin. It is however possible to switch from an endpoint Taylor form to a midpoint one and thus be able to obtain quadratic convergence, as detailed in Section A.

## Existence test for systems of inequalities

The *X-Newton* operator is a contractor but does not provide any guarantee that a solution exists inside the returned box. Since this operator is adapted to systems of inequality constraints, we detail in Section B an original existence test that applies to this class of systems.

# 5 Experiments

We have applied *X-Newton* to constrained global optimization and to constraint satisfaction problems.

## 5.1 Experiments in constrained global optimization

We have selected a sample of global optimization systems among those tested by Ninin et al. [26]. They have proposed an interval Branch and Bound, called here `IBBA+`, that uses constraint propagation and a sophisticated variant of affine arithmetic. From their benchmark of 74 polynomial and non polynomial systems (without trigonometric operators), we have extracted the 27 ones that required more than 1 second to be solved by the simplest version of `IbexOpt` (column 4). Table 1 shows the 11 systems solved by this first version in a time comprised between 1 and 11 seconds. Table 2 includes the 13 systems solved in more than 11 seconds.<sup>7</sup> Three systems (`ex6_2_5`, `ex6_2_7` and `ex6_2_13`) are removed from the benchmark because they are not solved by any solver. The reported results have been obtained on a same computer (`Intel X86`, 3Ghz).

We have implemented the different algorithms in the Interval-Based `EXplorer` [9] (`Ibex`). Reference [31] details how our interval B&B, called `IbexOpt`, handles constrained optimization problems by using recent and new algorithms. Contraction steps are achieved by the `Mohc` interval constraint propagation algorithm [2] (that also lower bounds the range of the objective function). The upper bounding phase uses original algorithms for extracting *inner regions* inside the feasible search space, i.e., zones in

<sup>7</sup>Note that most of these systems are also difficult for the *non* reliable state-of-the-art global optimizer `Baron` [30], i.e., they are solved in a time comprised between 1 second and more than 1000 seconds (time out).

Table 1: Experimental results on medium difficult global optimization systems

System	$n$	No	Rand	R+R	R+op	RRRR	Best	B+op	XIter	XNewt	Ibex'	Ibex''	IBBA+
ex2_1_8	24	TO	10.50 3605	10.27 2739	9.32 2444	12.29 2200	TO	TO	<b>8.43</b> 1068	8.92 418	47.96 38988	TO	26.78 1916
ex3_1_1	8	MO	1.91 2429	1.75 1877	1.28 1529	1.75 1556	1851	1516	<b>1.24</b> 676	1.87 428	MO	121 36689	116 131195
ex6_1_4	6	MO	1.74 1844	1.48 1359	<b>1.10</b> 1069	1.59 1146	1830	1097	1.40 796	1.55 540	1.82 4218	2.30 2215	2.70 1622
ex6_2_14	4	2.16 1421	1.74 1290	1.68 1264	1.58 1247	1.79 1239	1369	1237	1.58 1066	<b>1.49</b> 742	44.53 109745	65.26 104483	208 95170
ex7_2_1	7	883 1.2e+6	1.23 1410	1.28 1314	1.22 1280	1.57 1276	1636	1336	0.49 260	<b>0.45</b> 153	13.74 33478	5.45 5139	24.72 8419
ex7_2_6	3	10.52 71447	9.42 31601	6.63 20874	1.24 3425	3.65 9412	3.7e+5	1.2e+5	4.22 9211	2.74 4272	<b>0.11</b> 570	0.16 436	1.23 1319
ex7_3_4	12	39.08 38291	<b>1.11</b> 818	1.33 793	1.28 770	1.56 685	789	760	1.66 441	2.25 334	TO	TO	TO
ex14_2_1	5	7.57 7374	1.04 768	1.09 689	0.95 619	1.28 587	749	604	<b>0.68</b> 336	0.88 198	8.97 14476	21.20 22720	36.73 16786
ex14_2_3	6	20.21 11557	2.82 1203	3.20 1150	2.91 1081	3.82 1017	1533	979	<b>1.75</b> 525	2.62 376	64.22 55347	30.81 19410	TO
ex14_2_4	5	0.96 657	1.09 588	1.33 490	1.04 471	1.35 437	545	481	<b>0.65</b> 229	1.09 220	35.32 34240	36.80 28249	128 30002
ex14_2_6	5	1.11 689	1.20 578	1.21 459	1.24 501	1.51 424	578	484	<b>1.05</b> 368	1.21 234	42.61 74630	72.52 32675	238 74630
Sum			33.80 46134	31.25 33308	<b>23.16</b> 14436	32.16 19979			<b>23.15</b> 14976	25.07 7915	147 229402	203 208268	638 227948
Gain			1	1.02	<b>1.71</b>	1.03			1.50	1.40			

which all points satisfy the inequality and relaxed equality constraints.<sup>8</sup> The cost of any point inside an inner region may improve the upper bound. Also, at each node of the B&B, the X-Taylor algorithm is used to produce hyperplanes for each inequality constraints and the objective function. On the obtained convex polyhedron, two types of tasks can be achieved: either the lower bounding of the cost with one call to a Simplex algorithm (results reported in columns 4 to 13), or the lower bounding and the contraction of the box, with X-NewIter (i.e.,  $2n + 1$  calls to a Simplex algorithm; results reported in column 10) or X-Newton (columns 11, 13). The bisection heuristic is a variant of Kearfott's Smear function described in [31].

The first two columns contain the name of the handled system and its number of variables. Each entry contains generally the CPU time in second (first line of a multi-line) and the number of branching nodes (second line). The same precision on the cost ( $1.e-8$ ) and the same timeout (TO = 1 hour) have been used by IbexOpt and IBBA+.<sup>9</sup> Cases of memory overflow (MO) sometimes occur. For each method  $m$ , the last line includes an average gain on the different systems. For a given system, the gain w.r.t. the basic method (column 4) is  $\frac{CPU\ time(Rand)}{CPU\ time(m)}$ . The last 10 columns of Table 2 compare different variants of X-Taylor and X-Newton. The differences between variants are clearer on the most difficult instances. All use Hansen's variant to compute the interval gradient (see Section 3.2). The gain is generally small but

<sup>8</sup>An equation  $h_j(x) = 0$  is relaxed by two inequality constraints:  $-\varepsilon \leq h_j(x) \leq +\varepsilon$ .

<sup>9</sup>The results obtained by IBBA+ on a similar computer are taken from [26].



Hansen’s variant is more robust: for instance `ex_7_2_3` cannot be solved with the basic interval gradient calculation.

In the column 3, the convexification operator is removed from our interval B&B, which underlines its significant benefits in practice. The column 4 corresponds to an

Table 2: Experimental results on difficult constrained global optimization systems

1	2	3	4	5	6	7	8	9	10	11	12	13	14
System	n	No	Rand	R+R	R+op	RRRR	Best	B+op	XIter	XNewt	Ibex'	Ibex''	IBBA+
ex2_1_7	20	TO	42.96 20439	43.17 16492	40.73 15477	49.48 13200	TO	TO	<b>7.74</b> 1344	10.58 514	TO	TO	16.75 1574
ex2_1_9	10	MO	40.09 49146	29.27 30323	22.29 23232	24.54 19347	57560	26841	<b>9.07</b> 5760	9.53 1910	46.58 119831	103 100987	154.02 60007
ex6_1_1	8	MO	20.44 21804	19.08 17104	<b>17.23</b> 14933	22.66 14977	24204	15078	31.24 14852	38.59 13751	TO	633 427468	TO
ex6_1_3	12	TO	1100 522036	711 269232	529 205940	794 211362	TO	TO	262.5 55280	<b>219</b> 33368	TO	TO	TO
ex6_2_6	3	TO	162 172413	175 168435	169 163076	207 163967	1.7e5	1.6e5	172 140130	<b>136</b> 61969	1033 1.7e6	583 770332	1575 922664
ex6_2_8	3	97.10 1.2e5	121 117036	119 105777	110 97626	134.7 98897	1.2e5	97580	78.1 61047	<b>59.3</b> 25168	284 523848	274 403668	458 265276
ex6_2_9	4	<b>25.20</b> 27892	33.0 27892	36.7 27826	35.82 27453	44.68 27457	27881	27457	42.34 27152	43.74 21490	455 840878	513 684302	523 203775
ex6_2_10	6	TO	3221 1.6e6	2849 1.2e6	<b>1924</b> 820902	2905 894893	1.1e6	8.2e5	2218 818833	2697 656360	TO	TO	TO
ex6_2_11	3	10.57 17852	19.31 24397	<b>7.51</b> 8498	7.96 8851	10.82 10049	5606	27016	13.26 12253	11.08 6797	41.21 93427	11.80 21754	140.51 83487
ex6_2_12	4	2120 2e6	232 198156	160 113893	118.6 86725	155 90414	1.9e5	86729	51.31 31646	<b>22.20</b> 7954	122 321468	187 316675	112.58 58231
ex7_3_5	13	TO	44.7 45784	54.9 44443	60.3 50544	75.63 43181	45352	42453	29.88 6071	<b>28.91</b> 5519	TO	TO	TO
ex14_1_7	10	TO	433 223673	445 172671	<b>406</b> 156834	489 125121	1.7e5	1.1e5	786 179060	938 139111	TO	TO	TO
ex14_2_7	6	93.10 35517	94.16 25802	102.2 21060	83.6 16657	113.7 15412	20273	18126	<b>66.39</b> 12555	97.36 9723	TO	TO	TO
Sum			5564 3.1e6	4752 2.2e6	<b>3525</b> 1.7e6	5026 1.7e6			3767 1.4e6	4311 983634	1982 3.6e6	1672 2.3e6	2963 1.6e6
Gain			1	1.21	1.39	1.07			<b>2.23</b>	1.78			
ex7_2_3	8	MO	MO	MO	MO	MO			<b>544</b> 611438	691 588791	TO	719 681992	TO

X-Taylor performed with one corner randomly picked for every constraint. The next column (R+R) corresponds to a tighter polytope computed with two randomly chosen corners per inequality constraint. The gain is small w.r.t. *Rand*. The column 6 (R+op) highlights the best X-Taylor variant where a random corner is chosen along with its opposite corner. Working with more than 2 corners appeared to be counter-productive, as shown by the column 7 (RRRR) that corresponds to 4 corners randomly picked.

We have performed a very informative experiment whose results are shown in columns 8 (*Best*) and 9 (*B+op*): an exponential algorithm selects the best corner, maximizing the expression (4), among the  $2^n$  ones.<sup>10</sup> The reported number of branching nodes shows that the best corner (resp. *B+op*) sometimes brings no additional contraction and often brings a very small one w.r.t. a random corner (resp. *R+op*). Therefore, the combination *R+op* has been kept in all the remaining variants (columns 10 to 14).

The column 10 (*XIter*) reports the results obtained by X-Newton. It shows the best performance on average while being robust. In particular, it avoids the memory

<sup>10</sup>We could not thus compute the number of branching nodes of systems with more than 12 variables because they reached the timeout.

overflow on `ex7_2_3`. X-Newton, using `ratio_fp=20%`, is generally slightly worse, although a good result is obtained on `ex6_2_12` (see column 11).

The last three columns report a first comparison between AA (affine arithmetic; Ninin et al.’s AF2 variant) and our convexification methods. Since we did not encode AA in our solver due to the significant development time required, we have transformed `IbexOpt` into two variants `Ibex’` and `Ibex’’` very close to IBBA+: `Ibex’` and `Ibex’’` use a non incremental version of HC4 [5] that loops only once on the constraints, and a *largest-first* branching strategy. The upper bounding is also the same as IBBA+ one. Therefore we guess that only the convexification method differs from IBBA+: `Ibex’` improves the lower bound using a polytope based on a random corner and its opposite corner; `Ibex’’` builds the same polytope but uses X-Newton to better contract on all the dimensions.<sup>11</sup>

First, `Ibex’` reaches the timeout once more than IBBA+; and IBBA+ reaches the timeout once more than `Ibex’’`. Second, the comparison in the number of branching points (the line *Sum* accounts only the systems that the three strategies solve within the timeout) underlines that AA contracts generally more than `Ibex’`, but the difference is smaller with the more contracting `Ibex’’` (that can also solve `ex7_2_3`). This suggests that the job on all the variables compensates the relative lack of contraction of X-Taylor. Finally, the performances of `Ibex’` and `Ibex’’` are better than IBBA+ one, but it is probably due to the different implementations.

## 5.2 Experiments in constraint satisfaction

We have also tested the X-Newton contractor in constraint satisfaction, i.e., for solving well constrained systems having a finite number of solutions. These systems are generally square systems ( $n$  equations and  $n$  variables). The constraints correspond to non linear differentiable functions (some systems are polynomial, others are not). We have selected from the COPRIN benchmark<sup>12</sup> all the systems that can be solved by one of the tested algorithms in a time between 10 s and 1000 s: we discarded easy problems solved in less than 10 seconds, and too difficult problems that no method can solve in less than 1000 seconds. The timeout was fixed to one hour. The required precision on the solution is  $10^{-8}$ . Some of these problems are scalable. In this case, we selected the problem with the greatest size (number of variables) that can be solved by one of the tested algorithms in less than 1000 seconds.

We compared our method with the state of art algorithm for solving such problems in their original form (we did not use rewriting of constraints and did not exploit common subexpressions). We used as reference contractor our best contractor `ACID(Mohc)`, an adaptive version of CID [32] with `Mohc` [2] as basic contractor, that exploits the monotonicity of constraints. We used the same bisection heuristic as in optimization experiments. Between two choice points in the search tree, we called one of the following contractors (see Table 3).

- `ACID(Mohc)`: see column 3 (Ref),
- `X-NewIter`: `ACID(Mohc)` followed by one call to Algorithm 1 (column 4, Xiter),
- `X-Newton`: the most powerful contractor with `ratio_fp=20%`, and `ACID(Mohc)` as internal CP contractor (see Algorithm 2).

<sup>11</sup>We have removed the call to `Mohc` inside the X-Newton loop (i.e., `CP-contractor=⊥`) because this constraint propagation algorithm is not a convexification method.

<sup>12</sup><http://www-sop.inria.fr/coprin/logiciels/ALIAS/Benches/benches.html>

For X-Newton, we have tested 5 ways for selecting the corners (see columns 5–9):

- Rand: one random corner,
- R+R: two random corners,
- R+op: one random corner and its opposite,
- RRRR: four random corners,
- 2R+op: four corners, i.e., two random corners and their two respective opposite ones.

We can observe that, as for the optimization problems, the corner selection R+op yields the lowest sum of solving times and often good results. The performance profile 2 (and the last line of Table 3) highlights that all the 24 systems can be solved in 1000 s by X-Newton R+op, while only 18 systems are solved in 1000 s by the reference algorithm with no convexification method. Each entry in Table 3 contains the CPU

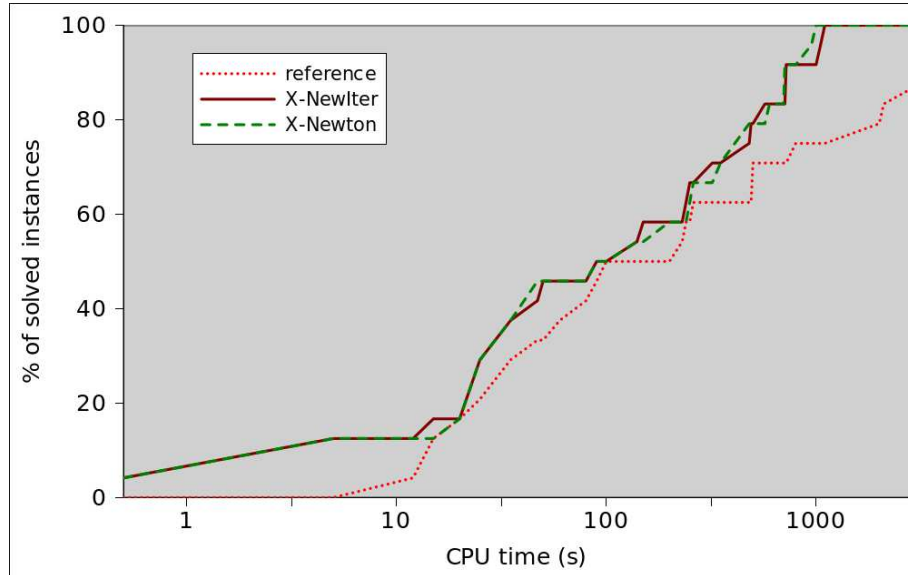


Figure 2: **Performance profile.** The curves show, for a given algorithm, the percentage of systems solved as a function of the CPU time in second.

time in second (first line of a multi-line) and the number of branching nodes (second line). We have reported in the last column (Gain) the gains obtained by the best corner selection strategy R+op as the ratio w.r.t. the reference method (column 3 Ref), i.e.,  $\frac{CPU\ time(R+op)}{CPU\ time(Ref)}$ . Note that we used the inverse gain definition compared to the one used in optimization (see 5.1) in order to manage the problems reaching the timeout. We can also observe that our new algorithm X-Newton R+op is efficient and robust: we can obtain significant gains (small values in bold) and lose never more than 39% in CPU time.

We have finally tried, for the scalable systems, to solve problems of bigger size. We could solve Katsura-30 in 4145 s, and Yamamura1-16 in 2423 s (instead of 33521 s with the reference algorithm). We can remark that, for these problems, the gain grows with the size.

Table 3: Experimental results on difficult constraint satisfaction problems. The best results and the gains ( $< 1$ ) appear in bold.

1	2	3	4	5	6	7	8	9	10
System	n	Ref	Xiter	Rand	R+R	R+op	RRRR	2R+op	Gain
Bellido	9	10.04 3385	3.88 1273	4.55 715	3.71 491	<b>3.33</b> 443	<b>3.35</b> 327	<b>3.28</b> 299	<b>0.33</b>
Bratu-60	60	494 9579	<b>146</b> 3725	306 4263	218 3705	190 3385	172 3131	357 5247	<b>0.38</b>
Brent-10	10	<b>25.31</b> 4797	28 4077	31.84 3807	33.16 3699	34.88 3507	37.72 3543	37.11 3381	1.38
Brown-10	10	TO	<b>0.13</b> 67	0.17 49	0.17 49	0.17 49	0.17 49	0.18 49	<b>0</b>
Butcher8-a	8	<b>233</b> 40945	246 39259	246 36515	248 35829	242 35487	266 33867	266 33525	1.06
Butcher8-b	8	<b>97.9</b> 26693	123 23533	113.6 26203	121.8 24947	122 24447	142.4 24059	142.2 24745	1.26
Design	9	<b>21.7</b> 3301	23.61 3121	22 2793	22.96 2549	22.38 2485	25.33 2357	25.45 2365	1.03
Direct Kinematics	11	85.28 1285	<b>81.25</b> 1211	84.96 1019	83.52 929	84.28 915	86.15 815	85.62 823	<b>0.99</b>
Dietmaier	12	3055 493957	1036 152455	<b>880</b> 113015	979 96599	960 93891	1233 85751	1205 83107	<b>0.31</b>
Discrete integral-16 2nd form.	32	TO	480 57901	<b>469</b> 57591	<b>471</b> 57591	<b>472</b> 57591	478 57591	476 57591	<b>0</b>
Eco9	8	<b>12.85</b> 4573	14.19 3595	14.35 3491	14.88 2747	15.05 2643	17.48 2265	17.3 2159	1.17
Ex14-2-3	6	45.01 3511	3.83 291	4.39 219	3.88 177	<b>3.58</b> 181	3.87 145	3.68 139	<b>0.08</b>
Fredtest	6	74.61 18255	47.73 12849	54.46 11207	47.43 8641	44.26 7699	42.67 6471	<b>40.76</b> 6205	<b>0.59</b>
Fourbar	4	<b>258</b> 89257	317 83565	295 79048	319 73957	320 75371	366 65609	367 67671	1.24
Geneig	6	57.32 3567	46.1 3161	46.25 2659	41.33 2847	40.38 2813	<b>38.4</b> 2679	<b>38.43</b> 2673	<b>0.7</b>
I5	10	<b>17.21</b> 5087	20.59 4931	19.7 5135	20.53 4885	20.86 4931	23.23 4843	23.43 4861	1.21
Katsura-25	26	TO	711 9661	1900 17113	1258 7857	<b>700</b> 4931	1238 5013	1007 4393	<b>0</b>
Pramanik	3	<b>14.69</b> 18901	20.08 14181	19.16 14285	20.31 11919	20.38 11865	24.58 11513	25.15 12027	1.39
Synthesis	33	<b>212</b> 9097	235 7423	264 7135	316 6051	259 4991	631 7523	329 3831	1.22
Trigexp2-17	17	<b>492</b> 27403	568 27049	533 26215	570 25805	574 25831	630 25515	637 25055	1.17
Trigo1-14	14	2097 8855	1062 5229	1314 4173	1003 2773	910 2575	865 1991	<b>823</b> 1903	<b>0.43</b>
Trigonometric	5	33.75 4143	30.99 3117	<b>30.13</b> 2813	<b>30.11</b> 2265	30.65 2165	31.13 1897	31.75 1845	<b>0.91</b>
Virasoro	8	760 32787	715 35443	729 33119	<b>704</b> 32065	<b>709</b> 32441	713 30717	715 27783	<b>0.93</b>
Yamamura1-14	14	1542 118021	<b>407</b> 33927	628 24533	557 23855	<b>472</b> 14759	520 13291	<b>475</b> 11239	<b>0.26</b>
Sum		>42353 >1.8e6	6431 531044	8000 477115	7087 432232	<b>6185</b> 415396	7588 382862	7131 382916	
Gain		1	0.75	0.77	0.78	0.76	0.9	0.85	
Solved in 1000 s		18	22	22	22	<b>24</b>	22	22	

## 6 Conclusion

Endowing a solver with a reliable convexification algorithm is useful in constraint satisfaction and crucial in constrained global optimization. This paper has presented the probably simplest way to produce a reliable convexification of the solution space and the objective function. X-Taylor can be encoded in 100 lines of codes and calls a standard Simplex algorithm. It rapidly computes a polyhedral convex relaxation following Hansen's recursive principle to produce the gradient and using two corners as expansion point of Taylor: a corner randomly selected and the opposite corner.

This convex interval Taylor form can be used to build an eXtremal interval Newton. The X-NewIter variant contracting all the variable intervals once provides on average the best performance on constrained global optimization systems. For constraint satisfaction, both algorithms yield comparable results.

Compared to affine arithmetic, preliminary experiments suggest that our convex interval Taylor produces a looser relaxation in less CPU time. However, the additional job achieved by X-Newton can compensate this lack of filtering at a low cost, so that one can solve one additional tested system in the end. Therefore, we think that this reliable convexification method has the potential to complement affine arithmetic and Quad.

## Acknowledgment

We would like to particularly thank G. Chabert for useful discussions about existing interval analysis results.

## References

- [1] O. Aberth. The Solution of Linear Interval Equations by a Linear Programming Method. *Linear Algebra and its Applications*, 259:271–279, 1997.
- [2] I. Araya, G. Trombettoni, and B. Neveu. Exploiting Monotonicity in Interval Constraint Propagation. In *Proc. AAAI*, pages 9–14, 2010.
- [3] A. Baharev, T. Achterberg, and E. Rév. Computation of an Extractive Distillation Column with Affine Arithmetic. *AIChE Journal*, 55(7):1695–1704, 2009.
- [4] O. Beaumont. *Algorithmique pour les intervalles*. PhD thesis, Université de Rennes, 1997.
- [5] F. Benhamou, F. Goualard, L. Granvilliers, and J.-F. Puget. Revising Hull and Box Consistency. In *Proc. ICLP*, pages 230–244, 1999.
- [6] C. Bliek. *Computer Methods for Design Automation*. PhD thesis, MIT, 1992.
- [7] G. Chabert. *Techniques d'intervalles pour la résolution de systèmes d'intervalles*. PhD thesis, Université de Nice–Sophia, 2007.
- [8] G. Chabert and N. Beldiceanu. Sweeping with Continuous Domains. In *Proc. CP, LNCS 6308*, pages 137–151, 2010.
- [9] G. Chabert and L. Jaulin. Contractor Programming. *Artificial Intelligence*, 173:1079–1100, 2009.

- 
- [10] L. de Figueiredo and J. Stolfi. Affine Arithmetic: Concepts and Applications. *Numerical Algorithms*, 37(1–4):147–158, 2004.
- [11] A. Goldsztejn and L. Granvilliers. A New Framework for Sharp and Efficient Resolution of NCSP with Manifolds of Solutions. *Constraints (Springer)*, 15(2):190–212, 2010.
- [12] E.R. Hansen. On Solving Systems of Equations Using Interval Arithmetic. *Mathematical Comput.*, 22:374–384, 1968.
- [13] E.R. Hansen. Bounding the Solution of Interval Linear Equations. *SIAM J. Numerical Analysis*, 29(5):1493–1503, 1992.
- [14] E.R. Hansen. *Global Optimization using Interval Analysis*. Marcel Dekker inc., 1992.
- [15] R. B. Kearfott. *Rigorous Global Search: Continuous Problems*. Kluwer Academic Publishers, 1996.
- [16] V. Kreinovich, A.V. Lakeyev, J. Rohn, and P.T. Kahl. *Computational Complexity and Feasibility of Data Processing and Interval Computations*. Kluwer, 1997.
- [17] Y. Lebbah, C. Michel, and M. Rueher. An Efficient and Safe Framework for Solving Optimization Problems. *J. Computing and Applied Mathematics*, 199:372–377, 2007.
- [18] Y. Lebbah, C. Michel, M. Rueher, D. Daney, and J.P. Merlet. Efficient and safe global constraints for handling numerical constraint systems. *SIAM Journal on Numerical Analysis*, 42(5):2076–2097, 2005.
- [19] Y. Lin and M. Stadtherr. LP Strategy for the Interval-Newton Method in Deterministic Global Optimization. *Industrial & engineering chemistry research*, 43:3741–3749, 2004.
- [20] D. McAllester, P. Van Hentenryck, and D. Kapur. Three Cuts for Accelerated Interval Propagation. Technical Report AI Memo 1542, Massachusetts Institute of Technology, 1995.
- [21] F. Messine, , and J.-L. Laganouelle. Enclosure Methods for Multivariate Differentiable Functions and Application to Global Optimization. *Journal of Universal Computer Science*, 4(6):589–603, 1998.
- [22] R. E. Moore. *Interval Analysis*. Prentice-Hall, 1966.
- [23] R.E. Moore, R. B. Kearfott, and M.J. Cloud. *Introduction to Interval Analysis*. SIAM, 2009.
- [24] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge Univ. Press, 1990.
- [25] A. Neumaier and O. Shcherbina. Safe Bounds in Linear and Mixed-Integer Programming. *Mathematical Programming*, 99:283–296, 2004.
- [26] J. Ninin, F. Messine, and P. Hansen. A Reliable Affine Relaxation Method for Global Optimization. *Submitted (research report RT-APO-10-05, IRIT, march 2010)*, 2010.

- [27] W. Oettli. On the Solution Set of a Linear System with Inaccurate Coefficients. *SIAM J. Numerical Analysis*, 2(1):115–118, 1965.
- [28] J. Rohn. Cheap and Tight Bounds: The Recent Result by E. Hansen can be Made More Efficient. *Interval Computations*, 1:13–21, 1993.
- [29] T. J. Schaefer. The Complexity of Satisfiability Problems. In *Proc. STOC, ACM symposium on theory of computing*, pages 216–226, 1978.
- [30] M. Tawarmalani and N. V. Sahinidis. A Polyhedral Branch-and-Cut Approach to Global Optimization. *Mathematical Programming*, 103(2):225–249, 2005.
- [31] G. Trombettoni, I. Araya, B. Neveu, and G. Chabert. Inner Regions and Interval Linearizations for Global Optimization. In *AAAI*, pages 99–104, 2011.
- [32] G. Trombettoni and G. Chabert. Constructive Interval Disjunction. In *Proc. CP, LNCS 4741*, pages 635–650, 2007.
- [33] X.-H. Vu, D. Sam-Haroud, and B. Faltings. Enhancing Numerical Constraint Propagation using Multiple Inclusion Representations. *Annals of Mathematics and Artificial Intelligence*, 55(3–4):295–354, 2009.

## A X-Newton and square systems of equations

In the case where square constraint systems are handled, the standard interval Newton operator, called I-Newton hereafter, sometimes detects cases where the system falls in a convergence basin and obtains quadratic convergence. The X-Newton operator cannot have the same property a priori, but we propose a hybrid version Square-X-Newton of X-Newton that can switch from an endpoint Taylor form to a midpoint one when a necessary condition holds. (The possibility of calling CP-contractor is not considered in this section for the sake of clarity.)

Let us first recall the principle of the standard interval Newton operator.

### A.1 Standard interval Newton

We consider here that  $f = (f_1, \dots, f_j, \dots, f_m)$  is the set of functions involved in the set of equations  $f_j(x) = 0$  handled by the algorithm. Let  $x$  be a vector of variables and  $[x] = [x_1] \times \dots \times [x_i] \times \dots \times [x_n]$  its domain. Let  $[A]$  be the interval Jacobian matrix (Hansen's variant) obtained with a midpoint interval Taylor form, i.e., a matrix in which every element is the interval:

$$[a_{i,j}] = \left[ \frac{\partial f_j}{\partial x_i} \right]_N ([x_1] \times \dots \times [x_i] \times m([x_{i+1}]) \times \dots \times m([x_n])).$$

One iteration of the interval Newton operator contracts the current box. It returns a box  $[x']$  and intersects it with the current box  $[x]$ , as follows:

1. Compute the Jacobian matrix  $[A]$  of  $f$  in  $[x]$  with a midpoint interval Taylor form.  
Compute the vector of values  $b := -f(m([x]))$ .
2. Compute  $P := m([A])^{-1}$ .
3. Preconditioning:  $[A'] := P.[A]$ ;  $b' := P.b$ .
4. Compute the hull  $[x']$  of the solution set of the interval linear system:  $[A'][x] = b'$ .
5.  $[x] := [x] \cap [x']$

Several such iterations are launched until a quasi fixed-point is reached in terms of contraction.

The step 4 of an interval Newton iteration can be performed by several methods, such as an interval Gauss-Seidel or the Hansen-Bliek method mentioned above [6, 14, 28]. Also, if after step 4 we have  $[x'] \subseteq [x]$ , then it is guaranteed that a unique solution exists inside  $[x']$  and that further iterations will quadratically converge to this solution [24].

### A.2 I-Newton and X-Newton for square systems

In the case where square constraint systems are handled, X-Newton can be specialized to a Square-X-Newton variant that can theoretically obtain sometimes quadratic convergence.



---

**Algorithm 3** Square-X-Newton ( $f, x, [x], \text{ratio\_fp}$ ):  $[x]$

---

```

repeat
   $[x]_{\text{save}} \leftarrow [x]$ 
   $[x] \leftarrow \text{X-NewIter}(f, x, [x])$ 
until ( $\text{empty}([x])$  or
   $[x] \subset [x]_{\text{save}}$  and  $\perp \neq P := \text{InverseMidPointJacobian}(f, x, [x])$  or
   $\text{gain}([x], [x]_{\text{save}}) < \text{ratio\_fp}$ )
if !  $\text{empty}([x])$  and  $[x] \subset [x]_{\text{save}}$  and  $P \neq \perp$  then
  return  $\text{I-Newton}(f, x, [x])$ 
else
  return  $[x]$ 
end if

```

---

The inclusion test  $[x] \subset [x]_{\text{save}}$  is a necessary condition for the existence and unicity of a solution inside  $[x]$ .<sup>13</sup> A second condition makes the test sufficient: the condition that the midpoint of the Jacobian matrix  $[A]$  be invertible. This implies a so-called *strong regularity* condition on  $[A]$  that implies its regularity [7, 24].

Therefore, in practice, each time the inclusion test is true, the function `InverseMidPointJacobian` resorts to the first two steps shown in Section A.1 for computing the preconditioning matrix  $P = m([A])^{-1}$ . It returns  $P = \perp$  when  $m([A])$  is not invertible.

Both conditions prove the existence and unicity of a solution in the box. They also imply quadratic convergence onto the linear solution set [7, 24]. Hence the last call to `I-Newton`.

## B Existence test for systems of inequality constraints

We consider here a constraint system  $S$  made of a set of inequalities, in which Algorithm 4 tries to guarantee the existence of a (floating-point) solution. Note that this existence test may fail although one such solution exists in the box, like every other existence test.

---

**Algorithm 4** InequalitiesExistence ( $S=(f, x, [x])$ ): boolean

---

```

return  $f(\text{RandomProbing}([x])) \leq 0$  or
   $\text{InHG4}(S)$  or
   $\text{InnerLinearization}(S)$ 

```

---

The test first randomly picks an  $n$ -dimensional point  $x$  inside the box  $[x]$  (see `RandomProbing` in Algorithm 4). If this floating-point number satisfies the constraints, i.e.,  $f(x) \leq 0$ , then the existence test succeeds. This sometimes works in practice at the end of the combinatorial search because bisection and contraction operations have reduced the box  $[x]$  around solutions. Otherwise, the test continues with more original tests based on inner boxes and inner polyhedral regions.

**Definition 2** Consider a system made of only inequality constraints  $f(x) \leq 0$ , studied in a box  $[x]^{out}$ . An **inner region**  $r^{in}$  is a feasible subset of  $[x]^{out}$ , i.e.,  $r^{in} \subset [x]^{out}$  and all points  $x \in r^{in}$  satisfy  $f(x) \leq 0$ . An **inner box**  $[x]^{in}$  is an inner region which is a box.

<sup>13</sup>The strict inclusion must hold in our case because the domain/bound constraints imposed by  $[x]$  (i.e.,  $\underline{x}_i \leq x_i \leq \bar{x}_i$ ) are yielded to the Simplex algorithm via the procedure `X-NewIter`.

Without detailing, `InHC4` and `InnerLinearization` are recent heuristical algorithms able to sometimes extract respectively an inner box and an inner polytope inside a given box [8, 31]. Note that `InnerLinearization` uses a dual extremal interval Taylor form to extract an inner polytope [31]. In case of success of one of both inner region extraction algorithms, the existence test succeeds.

## B.1 Adaptation to equality constraints

This existence test could also hold for “*thick*” equations, i.e., equations with a non zero-dimensional set of solutions and for relaxed equations.

A thick equation is common in practice when at least one coefficient of the equation is known with a bounded uncertainty, e.g., an imprecision on a measured distance. This also appears in equations with irrational constants, like  $\pi$ . Provided that the bounded uncertainties and the irrational constants are encoded by interval constants, these thick equations  $f_j(x) = 0$  are transformed, without loss of information, into two inequalities  $0 \leq f_j(x) \leq 0$ .

A “true” equality  $f_k(x) = 0$  can also be handled with a relaxation as a thick equation  $f_k(x) \in [-\varepsilon_{eq}, +\varepsilon_{eq}]$ , i.e. two inequalities  $-\varepsilon_{eq} \leq f_k(x)$  and  $f_k(x) \leq \varepsilon_{eq}$ . This is inspired by our interval B&B where the default precision value  $\varepsilon_{eq} = 1. \text{e-}8$  for the equalities is tiny. In this case, our existence test holds for the relaxed system but not for the original one. To reach reliability for the original system, non square Newton operators, proposed by the interval analysis community (see e.g. [11]), should be applied in the end to check the existence of a real-valued solution of the system inside a given box. In practice, such a non square Newton algorithm would work with a box enclosing the inner point, box or polytope returned by Algorithm 4. For instance, starting from the inner box returned by `InHC4`, the size of the considered box would be increased (e.g., doubled) on all its dimensions until the existence of a solution is guaranteed inside it or until a maximum number of iterations is reached.



**RESEARCH CENTRE  
SOPHIA ANTIPOLIS – MÉDITERRANÉE**

2004 route des Lucioles - BP 93  
06902 Sophia Antipolis Cedex

Publisher  
Inria  
Domaine de Voluceau - Rocquencourt  
BP 105 - 78153 Le Chesnay Cedex  
[inria.fr](http://inria.fr)

ISSN 0249-6399