Taylorisation par intervalles convexe : premiers résultats

Ignacio Araya, Gilles Trombettoni
UTFSM, Chili; INRIA, I3S, Université de Nice-Sophia, France
iaraya@inf.utfsm.cl, Gilles.Trombettoni@inria.fr

Résumé

La taylorisation par intervalles est un outil mathématique important proposé dans les années 1960 par Ramon E. Moore et la communauté de l'analyse par intervalles. Elle permet de borner de manière élégante le reste dans l'approximation polynomiale d'une fonction non convexe. La taylorisation par intervalles est l'ingrédient de base des algorithmes de Newton sur intervalles qui peuvent résoudre de manière fiable les systèmes de contraintes non convexes, en prenant en compte les arrondis sur les nombres flottants et les incertitudes dans les données.

Malheureusement, à chaque itération de l'algorithme du Newton sur intervalles, l'approximation de l'ensemble des solutions générée par la taylorisation par intervalles au premier ordre demeure non convexe. On ne peut donc a priori pas produire d'enveloppe convexe optimale en temps polynomial. Les seules sous-classes polynomiales connues ont peu d'intérêt en pratique. C'est pourquoi d'autres méthodes de convexification connaissent un succès croissant, notamment l'arithmétique affine.

Il se trouve qu'une taylorisation par intervalles convexe a été ignorée pendant des décennies, au moins dans son exploitation pratique. En choisissant un *coin* de la boîte étudiée comme point d'expansion, la taylorisation par intervalles *extrême* produit une relaxation convexe (polyédrale) du système dont on peut produire une enveloppe optimale en temps polynomial. Elle permet de construire une variante de l'algorithme de Newton sur intervalles, sans préconditionnement, qui peut contracter le domaine en de nombreux nœuds de l'arbre de recherche. Nous montrons que le choix du coin produisant la relaxation la plus fine est NP-difficile ainsi que des premières expérimentations en optimisation globale.

1 Motivation

Interval methods use an elegant set-based reasoning to handle nonconvex continuous systems of constraints despite errors from rounding and uncertain data. The main and historical (type of) algorithm from interval analysis is *interval Newton*, an adaptation to intervals of the multivariate Newton algorithm [20, 12]. Interval Newton has beautiful properties such as quadratic convergence when regularity properties are met in the studied domain [22, 8]. In practice however, these properties are fullfilled when the domain is small, i.e., at the bottom of the search tree followed by the combinatorial search. Therefore, to better filter/contract the domain on the top of the search tree, modern interval solvers also resort to other algorithms coming from constraint programming [6] and mathematical programming. Constraint programming techniques include shaving/slicing methods [19, 30] and interval constraint propagation algorithms, such as `HC4` [5], `Box` [31, 5], `Mohc` [2]. Several convex relaxation techniques have been made robust to round-off errors with intervals, such as the reformulation-linearization-technique `Quad` [18] and affine arithmetic [11, 32, 3, 24]. Note that `Quad` and affine arithmetic are the only two existing interval linearization methods that can produce in polynomial-time a polyhedral, and thus convex, relaxation of a nonconvex constraint system.

Interval Newton uses interval taylorization to iteratively produce a linear system with interval coefficients. The main issue is that the system thus generated is *not* convex. Restricted to a single constraint, the linear relaxation forms a nonconvex "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 [22, 16, 21].) Contracting optimally the box containing this nonconvex relaxation has been proven to be NP-hard [17]. This explains why the interval analysis community has worked a lot on this problem for decades [22, 12].

Only a few polynomial subclasses have been studied. The most interesting one has been first described by Oettli and Prager in the sixties [25] and occurs when the variables are all nonnegative or nonpositive. Unfortunately, when the Taylor expansion point is chosen strictly inside the domain (the midpoint typically), the studied box must be previously split into (at most) 2^n subproblems before falling in this interesting subclass [1, 4, 8].

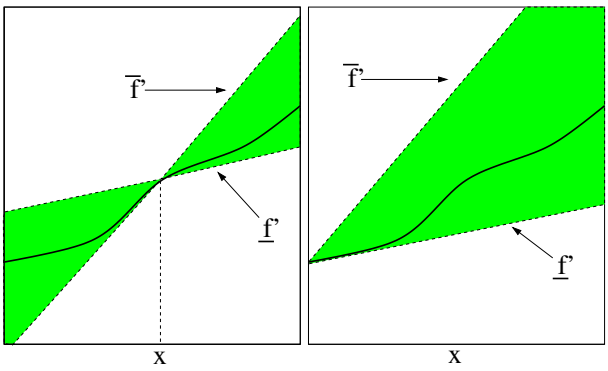

FIG. 1 – Relaxation of the range of $f(x)$ ($f : \mathbb{R} \to \mathbb{R}$) by an interval function (graph in green) using interval taylorization. **Left :** Midpoint taylorization, using a midpoint evaluation $f(m([x]))$, the maximum derivative $\overline{f'}$ of f inside the interval $[x]$ and the minimum derivative $\underline{f'}$. **Right :** Extremal taylorization, using an endpoint evaluation $f(\underline{x})$, $\overline{f'}$ and $\underline{f'}$.

It is worthwhile observing that interval taylorisation can be achieved at any expansion point in the studied domain [16] and most of the existing algorithms select the midpoint. This paper presents an interval taylorisation that has been ignored during decades by the interval analysis community, at least in its practical exploitation : our interval taylorization is performed at a *corner* of the studied box/domain. Graphically, it produces a cone, as shown in Fig. 1-right. A cone is less romantic than a butterfly and has a priori some drawbacks (see below), but it is convex and leads to interesting opportunities as well.

Let us introduce definitions and background before describing the simplest version of this convex *extremal* interval taylorization.

Intervals

An **interval** $[x_i] = [\underline{x_i}, \overline{x_i}]$ defines the set of reals x_i s.t. $\underline{x_i} \leq x_i \leq \overline{x_i}$. $\mathbb{IR}$ denotes the set of all intervals. The size or **width** of $[x_i]$ is $w([x_i]) = \overline{x_i} - \underline{x_i}$. A **box** $[x]$ is the Cartesian product of intervals $[x_1] \times ... \times [x_i] \times ... \times [x_n]$. Its width is defined by $\max_i w([x_i])$. $m([x])$ denotes the middle of $[x]$. The **magnitude** of an interval $[x_i]$ is defined by $|[x_i]| := \max_{x_i \in [x_i]} |x_i|$. The **hull** of a subset S of $\mathbb{R}^n$, denoted by $\texttt{Hull}(S)$, is the smallest n-dimensional box enclosing S.

Interval arithmetic [20] has been defined to extend to $\mathbb{IR}$ elementary functions over $\mathbb{R}$. For instance, the interval sum is defined by $[x_1]+[x_2] = [\underline{x_1}+\underline{x_2}, \overline{x_1}+\overline{x_2}]$. 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 $f : \mathbb{R}^n \to \mathbb{R}$.

$[f] : \mathbb{IR}^n \to \mathbb{IR}$ *is an* **extension** *of f to intervals if :*

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

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** [20], defined as follows :

$$[f]_t([x]) = f(\dot{x}) + \sum_i \left[\frac{\partial f}{\partial x_i}\right]_n ([x]) * ([x_i] - \dot{x}_i)$$

where $\dot{x}$ denotes any point in $[x]$, e.g., $m([x])$. Equivalently, we have $\forall x \in [x],\ \underline{[f]_t([x])} \leq f(x) \leq \overline{[f]_t([x])}$.

Example. Consider $f(x_1, x_2) = 3x_1^2 + x_2^2 + x_1 * x_2$ 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 : $\frac{\partial f}{\partial x_1}(x_1, x_2) = 6x_1 + x_2$, $[\frac{\partial f}{\partial x_1}]_n([-1, 3], [-1, 5]) = [-17, 23]$, $\frac{\partial f}{\partial x_2}(x_1, x_2) = x_1 + 2x_2$, $[\frac{\partial f}{\partial x_2}]_n([x_1], [x_2]) = [-3, 13]$. The interval Taylor evaluation with $\dot{x} = m([x]) = (1, 2)$ yields : $[f]_t([x_1], [x_2]) = 9 + [-17, 23] * [-2, 2] + [-3, 13] * [-3, 3] = [-76, 94]$.

A simple convex interval taylorization

A particular first order interval taylorization has been recently embedded in an interval branch and bound for global optimization [29]. Consider a function $f : \mathbb{R}^n \to \mathbb{R}$ 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 was to lower tighten $f(x)$ with one of the following interval linear forms :

$$\forall x \in [x], f(\underline{x}) + \underline{a_1} * y_1^l + ... + \underline{a_n} * y_n^l \ \leq f(x) \quad (1)$$
$$\forall x \in [x], f(\overline{x}) + \overline{a_1} * y_1^r + ... + \overline{a_n} * y_n^r \ \leq f(x) \quad (2)$$

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

One can use any expansion point inside the box to achieve the taylorization, and researchers in interval analysis often take the midpoint $m([x])$. Instead, we have chosen a *corner* of the box : $\underline{x}$ in form (1) or $\overline{x}$ in form (2). When applied to a set of inequality and equality[1] constraints, we obtain a polytope enclosing the solution set. This polytope can then be hulled in polynomial-time by an interior point algorithm or, in practice, by a Simplex algorithm : two calls to a Simplex algorithm can compute the minimum (resp. maximum) value of $\underline{x_i}$ (resp. $\overline{x_i}$) for each of the n variables (see Algorithm `X-NewtonIter` in Section 3 or, for instance [17], or page 81 of [15]).

[1] An equation $f(x) = 0$ can be viewed as two inequality constraints : $0 \leq f(x) \leq 0$.

Proposition 1 *The interval taylorizations (1) and (2) are correct and safe (reliable), i.e., they are robust to computation errors over floating point numbers.*

Proof. Safety is ensured by the *interval-based* taylorisation [22]. The correction of relation (1) lies on the simple fact that any variable y_i^l is positive since its domain is $[0, d_i]$, with $d_i = w([y_i^l]) = w([x_i]) = \overline{x_i} - \underline{x_i}$. Therefore, minimizing each term $[a_i] * y_i^l$ for any point $y_i^l \in [0, d_i]$ is 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 $\overline{a_i}$. □

Note that, eventhough our polytope computation is safe, the floating-point round-off errors made by the Simplex algorithm could render the hull of the polytope unsafe. A cheap postprocessing proposed in [23], using interval arithmetic, must be added to guarantee that no solution is lost by the Simplex algorithm.

Related work

The simple idea introduced above has a common point with the principle described by Oettli & Pragger [25] for characterizing their polynomial subclass. In their work, the bounds $\underline{a_i}$ and $\overline{a_i}$ can also be used for lower tightening a function because all the terms $y_i = x_i - \dot{x_i}$ are nonnegative or nonpositive. However, this occurs at the cost of previously splitting the initial domains $[y_i]$ at 0 and combining the different subintervals, thus building at most 2^n subdomains (*quadrants*) of the initial box [1, 4]. In particular, when the midpoints $\dot{x_i} = m([x_i])$ are chosen in the interval taylorization, the initial intervals $[y_i]$ are centered around 0.

Hansen and Bliek propose independently a sophisticated algorithm for avoiding to explicitly handle the 2^n quadrants [14, 7, 27]. The method handles a square $n * n$ system of equations the interval Jacobian matrix $[J]$ of which must be first preconditioned : $[J]$ must be multiplied by the inverse matrix of its midpoint. The obtained interval matrix is centered around the identity matrix I, thus creating a symmetry that avoids to consider the 2^n quadrants. The method can compute the optimal hull of the preconditioned interval linear system (i.e., intersections of butterflies) in polynomial-time, which is a beautiful result. A first limit is that the preconditioning overestimates the solution set of the initial interval linear system. In other words, the Hansen–Bliek method computes the optimal hull of an overestimated intersection of butterflies. A more important limit lies in the preconditioning that requires $[J]$ be *strongly regular* [22, 8]. This non-singularity condition is very restrictive and holds for instance if the system contains only one solution in the studied domain. This occurs at a low level of the search tree when solving nonconvex systems.

By choosing a corner as expansion point of our extremal interval taylorization, we have in every term $y_i = x_i - \underline{x_i} \geq 0$ or $y_i = x_i - \overline{x_i} \leq 0$. Thus, *the interval linear system produced by the extremal interval taylorization entirely belongs to a single quadrant and the system does not need be preconditioned.*

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 [22]) for computing a range enclosure (see Def. 1) of a univariate function. Neumaier calls this the *linear boundary value form*. However, to our knowledge, the presented convex interval taylorization seems to have never been exploited in practice for handling nonconvex constraint systems and is not described in the main reference books [20, 26, 22, 12, 16, 17, 15, 6, 21].

Characteristics of the extremal interval taylorization

Two reasons related to intervals can maybe explain why extremal interval taylorization has not been exploited in the past.

First, the overestimate implied by y_i on a term $[ai] * y_i$ appearing in an interval Taylor form (with $y_i = (x_i - \dot{x_i})$, $d_i = w([y_i]) = w([x_i])$) depends on the magnitude of $[y_i]$. Observe that $|[y_i]| = d_i$ (the maximum possible value) when $\dot{x_i} = \underline{x_i}$ or $\dot{x_i} = \overline{x_i}$, whereas it is only $|[y_i]| = \frac{d_i}{2}$ (i.e., $[y_i] = [-d_i/2, d_i/2]$) when $\dot{x_i} = m([x_i])$. This explains for instance why the cone depicted in Fig 1-right leads to a larger system relaxation surface than the butterfly appearing in Fig 1-left.

The midpoint of $[x]$ also permits an interesting convergence speed of interval Newton onto the unique solution when the box belongs to the convergence basin (see for instance page 52 of [16]).

Concerning the speed of convergence and the strong regularity requirement of the Hansen–Bliek method, it should be highlighted that this can be fulfilled in a very few number of nodes in the search tree solving systems of equations (no inequality). This explains why constraint programming methods are often used to contract the box in the other search tree nodes. The `X-Newton` algorithm presented in this paper, built upon extremal interval taylorization, is in a sense specialized outside the convergence basin. We will see in Section 3.4 that we can easily switch from an endpoint interval taylorization to a midpoint one.

To sum up, the main virtue of the extremal interval taylorization is that *the solution set of the interval linear system, although large, belongs to a unique quadrant and is convex*. This relaxed convex solution set of a nonconvex system can be optimally hulled in *polynomial-time, without preconditioning, at any node of the search tree.*

Outline

Section 2 deals with the extremal interval taylorization. We prove that the choice of the best expansion corner is an NP-hard problem and propose a first

greedy algorithm to make this choice in a heuristic way. Section 3 describes an eXtremal interval Newton algorithm, built upon our convex interval taylorization, for contracting a nonconvex system. Section 4 shows first experiments using extremal interval taylorization in global optimization. We compare different corner selection methods and with a strategy based on affine arithmetic.

2 Extremal interval taylorization

2.1 Preliminary interval linearization

Recall that the linear forms (1) and (2) shown in introduction use the bounds of the interval *gradient*, given by $\forall i \in \{1,...,n\}, [a_i] = \left[\frac{\partial f}{\partial x_i}\right]_n([x])$.

Eldon Hansen proposed in 1968 a famous variant in which the taylorization is achieved recursively, one variable after the other [13, 12]. The variant amounts in producing the following tighter interval coefficients :

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

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

By following Hansen's recursive principle, one can produce a Hansen variant of the form (1), for instance, in which the scalar coefficients are :

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

2.2 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 $\overline{x_i}$ in a term $\overline{a_i} * (x_i - \overline{x_i})$, 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 ? More precisely, we want to select a corner such that :

$$max_{x^c \in X^c} \int_{x_1=\underline{x_1}}^{\overline{x_1}} ... \int_{x_n=\underline{x_n}}^{\overline{x_n}} (f(x^c) + \sum_i z_i)\, dx_n * ... * dx_1 \quad (3)$$

where :

- $z_i = \overline{a_i}(x_i - \overline{x_i})$ iff $x_i^c = \overline{x_i}$,
- $z_i = \underline{a_i}(x_i - \underline{x_i})$ iff $x_i^c = \underline{x_i}$.

If we consider an inequality $f(x) \leq 0$, Expression (3) defines the tightest/highest hyper-plane $f^l(x)$ allowing one to enclose the solution set : $f^l(x) \leq f(x) \leq 0$.

Expression (3) means that we want to find a corner x^c that maximizes the Taylor form for *all* the points $x = \{x_1, ..., x_n\} \in [x]$, by adding their different contributions. Since :

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

Expression (3) is equivalent 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 $a_i^c = -\overline{a_i}$.
We simplify by the positive factor $\prod_i d_i$ and obtain :

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

2.2.1 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 $f : \mathbb{R}^n \to \mathbb{R}$, 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 in other communities (studying Diophantine equations for instance), but we are interested here in the polynomial transformation.

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)[2] 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 $\{x_1, ..., x_i, ..., x_n\}$ 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.

[2] We "restrict" the class to polynomial functions, otherwise the problem would not belong to *NP*.

– Note that we have chosen the bound $B = 0$.

It is straightforward (a) to check that this tranformation 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 $x'^{j}_1 * x'^{j}_2 * x'^{j}_3 = 0$.
– 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 $0.5 \sum_i a^c_i \, d_i$ in Expression (4) by selecting the maximum value among $\underline{a_i}$ and $-\overline{a_i}$ in every term.

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

Proposition 2 *(Corner selection is NP-hard) 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,*

$$max_{x^c \in X^c} \; -(f(x^c) + 0.5 \sum_i a^c_i \, d_i)$$
$$(or \; min_{x^c \in X^c} \; f(x^c) + 0.5 \sum_i a^c_i \, d_i)$$

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 [28]. BALANCED-3SAT does not belong to none of these 6 subclasses.[3]

A second difference with Lemma 1 is the bound B chosen for $f(x^c) + 0.5 \sum_i a^c_i \, d_i \leq B$. 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 $f(x^c) + 0.5 \sum_i a^c_i \, d_i \leq -0.5 \sum_i n_i$. Each term $x'^{j}_1 * x'^{j}_2 * x'^{j}_3$ of I' implies a partial derivative $\frac{\partial f}{\partial x'_i}([x])$ equal to 0 if x'_i does not appear in the term, equal to $[-1, 0]$ if x_i appears as a positive literal in I (i.e., $x'^{j}_k = (1 - x'_i)$ and $[-1, 0] = -1 * [0, 1] * [0, 1]$), and equal to $[0, 1]$ if x_i appears as a negative literal (i.e., $x'^{j}_k = x'_i$ 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 $\forall_i \; \underline{a_i} = -\overline{a_i}$ □

[3] 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 ($j \in \{1...d\}$) are dummy additional boolean variables (appearing d times as a negative literal and d times as a positive literal in round-robin...).

2.2.2 A first greedy corner selection

The previous section has shown that, assuming $P \neq NP$, no polynomial time corner selection algorithm exists for computing the tightest relaxation (by extremal interval taylorization) of an inequality $f(x) \leq 0$. This justifies the use of heuristics for selecting a "good" corner. The simplest heuristic method consists in choosing between $\underline{x_i}$ and $\overline{x_i}$ at random. When used at each node of a search tree, the *random corner selection* has the advantage of "diversifying" the computed relaxation.

We present hereafter a first greedy heuristic for the corner selection. Since Lemma 1 highlights that the difficult part is the maximization of $f(x^c)$, we use a heuristic approximation $fh(x_1, ..., x_n)$ of $f(x_1, ..., x_n)$: $fh(x_1, ..., x_n) = \sum_i fh(x_i)$, where $fh(x_i)$ reflects the impact of x_i on the range $fh(x_1, ..., x_n)$.We have chosen $fh(x_i) = 1/n \; fh'(x_i)$, with :

$$fh'(x_i) = f(m([x_1]), ..., m([x_{i-1}]), x_i, m([x_{i+1}], ..., m([x_n])$$

Note that the approximation is exact in the midpoint of the box, i.e., $fh(m([x])) = f(m([x]))$.

The heuristic variant of Expression (4) becomes : $max_{x^c \in X^c} \; \sum_i fh(x^c_i) + 0.5 \sum_i a^c_i \, d_i$ that can be maximized componentwise by computing the sign of the following quantity :

$$g(i) = (fh(\overline{x_i}) - 0.5 \, \overline{a_i} \, d_i) - (fh(\underline{x_i}) + 0.5 \, \underline{a_i} \, d_i)$$

where $d_i := w([x_i])$. Hence :

$$g(i) = (1/n * (fh'(\overline{x_i}) - fh'(\underline{x_i})) - 0.5 \, d_i \, (\underline{a_i} + \overline{a_i}).$$

We select the adequate bound $\underline{x_i}$ or $\overline{x_i}$ as follows :

– if $g(i) \geq 0$, then $x^c_i := \overline{x_i}$,
– if $g(i) < 0$, then $x^c_i := \underline{x_i}$.

2.3 Intersection of outer linear approximations

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, the obtained polytope corresponds to the intersection of these $c * m$ half-spaces.

Little care must be paid to put the whole linear system in the form $Ax - b$ before running the Simplex algorithm. We just show an example in which a function $f(x)$ is lower tightened by expressions (1) and (2) simultaneously. Both expressions can be written as :

– $f(\underline{x}) + \sum_i \underline{a_i}(x_i - \underline{x_i})$
– $f(\overline{x}) + \sum_i \overline{a_i}(x_i - \overline{x_i})$

We expand the expressions and obtain :

– $f(\underline{x}) + \sum_i \underline{a_i} x_i - \underline{a_i}\,\underline{x_i} = \sum_i \underline{a_i} x_i + f(\underline{x}) - \sum_i \underline{a_i}\,\underline{x_i}$
– $f(\overline{x}) + \sum_i \overline{a_i} x_i - \overline{a_i x_i} = \sum_i \overline{a_i} x_i + f(\overline{x}) - \sum_i \overline{a_i}\,\overline{x_i}$

Note that, to remain safe, the computation of constant terms $\underline{a_i}\,\underline{x_i}$ (resp. $\overline{a_i}\,\overline{x_i}$) must be achieved with degenerate intervals : $[\underline{a_i}, \underline{a_i}] * [\underline{x_i}, \underline{x_i}]$ (resp. $[\overline{a_i}, \overline{a_i}] * [\overline{x_i}, \overline{x_i}]$).

We end up with an `X-Taylorization` algorithm (`X-Taylorization` stands for *eXtremal interval taylorization*) able to produce c linear expressions lower tightening a given function $f : \mathbb{R}^n \to \mathbb{R}$ on a given domain $[x]$.

Algorithm 1 `X-Taylorization` (f, x, $[x]$, `IntervalGradient`, c, r) : HP=(f_1^l,...,f_c^l)

```
[G_f] ← IntervalGradient(f,[x])
HP ← ∅
for k from 1 to c do
  if k ≤ r then
    f_k^l ← greedy_k([G_f], f, x, [x])
  else
    f_k^l ← random([G_f])
  end if
  HP ← HP ∪ {f_k^l}
end for
return HP
```

r is the number of used greedy algorithms. For the moment $r = 1$, the only proposed `greedy_1` algorithm being described in Section 2.2.2. The function `IntervalGradient` computes the interval gradient $[G_f] = \{[a_1], ..., [a_i], ..., [a_n]\}$ that will be used to generate the (non-interval) linear expression (see Section 2.1). The function `random` corresponds to the simple random corner selection mentioned in the beginning of Section 2.2.2.

3 eXtremal interval Newton

Before describing the `X-Newton` algorithm, let us first recall the principle of the standard interval Newton, called `I-Newton` hereafter, applied to a square system of constraints.

3.1 Standard interval Newton

Let $f = f_1, ..., f_j, ..., f_m : \mathbb{R}^n \to \mathbb{R}^m$ be the set of considered functions, x be a vector of variables and $[x] = [x_1] \times ... \times [x_i] \times ... \times [x_n]$ its domain. Let $[A]$ be the interval Jacobian matrix (Hansen's variant) obtained by a midpoint-oriented interval taylorization, i.e., a matrix in which every element is the interval :

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

One iteration of interval Newton 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 taylorization.
   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 [7, 12, 27]. Also, if after step 4, $[x'] \subseteq [x]$, then it is guaranteed that a unique solution exists inside $[x']$ and that further iterations will quadratically converge to this solution [22].

3.2 X-Newton iteration

Let us describe now an `X-Newton` iteration, called `X-NewIter`, that contracts the box using an extremal first-order interval taylorization.

The standard interval Newton is rather adapted to equality constraints. On the contrary, `X-Newton` approximates nonconvex systems with polytopes and is thus devoted to inequalities. All equations $g(x) = 0$ are therefore transformed into two inequalities $0 \leq g(x) \leq 0$. All the constraints appear as inequality constraints $f_j(x) \leq 0$ in the vector $f = (f_1, ..., f_j, ..., f_m)$.

Algorithm 2 `X-NewIter` (f, x, $[x]$, `Gradient`, c, r) : $[x]$

```
for j from 1 to m do
  polytope ← polytope ∪
        {X-Taylorization(f_j,x,[x],Gradient, c, r)}
end for
for i from 1 to n do
  /* Two calls to a Simplex algorithm : */
  x_i ← min x_i subject to polytope
  x̄_i ← max x_i subject to polytope
end for
return [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.

Recall that, due to round-off errors made by the Simplex algorithm, a cheap postprocessing using interval arithmetic must be added to guarantee that no solution is lost [23].

3.3 X-Newton

The procedure `X-NewIter` allows one to build the main `X-Newton` algorithm.

Algorithm 3 `X-Newton` (f, x, $[x]$, `Gradient`, c, r, `ratio_fp`, `CP-contractor`) : $[x]$

repeat
  $[x]_{save} \leftarrow [x]$
  $[x] \leftarrow$ `X-NewIter` (f, x, $[x]$, `Gradient`, c, r)
  if `CP-contractor` $\neq \perp$ **and** `gain`$([x],[x]_{save}) > 0.001$ **then**
    $[x] \leftarrow$ `CP-contractor`$(f,x,[x])$
  end if
until `empty`$([x])$ **or** `gain`$([x],[x]_{save}) <$ `ratio_fp`)
return $[x]$

Consider first the basic variant in which `CP-contractor` $= \perp$. `X-NewIter` is iteratively run until a quasi fixed-point is reached in terms of contraction. More precisely, `ratio_fp` is a user-defined percentage of interval size and :

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

We also permit the use of a contraction algorithm, typically issued from constraint programming, inside the main loop. For instance, if the user has specified `CP-contractor=Mohc` and if the X-Newton iteration has reduced the domain, then the `Mohc` algorithm [2] can further contract the box.[4] The guard `gain`$([x],[x]_{save}) > 0.001$ guarantees that `CP-contractor` be not called twice if `X-NewIter` does not contract the box.

3.4 I-Newton and X-Newton for square systems

In the case where square constraint systems are handled, `X-Newton` can be specialized. Indeed, as mentioned above, a drawback of an interval Newton achieved with an extremal interval taylorization is the lack of quadratic convergence when we fall inside a convergence basin. We propose hereafter a hybrid version `Square-X-Newton` of `X-Newton` that can switch from a endpoint taylorization to a midpoint one when a necessary condition $[x]_{save} \subset [x]$ holds. (The possibility of calling `CP-contractor` is not considered in this section for the sake of clarity.)

We consider here that $f = (f_1, ..., f_j, ..., f_m)$ is the set of functions involved in the set of equations $f_j(x) = 0$ handled by the algorithm.

The inclusion test $[x] \subset [x]_{save}$ is a necessary condition for the existence and unicity of a solution inside $[x]$.[5] 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 [8, 22].

Algorithm 4 `Square-X-Newton` (f, x, $[x]$, `Gradient`, c, r, `ratio_fp`) : $[x]$

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

Therefore, in practice, each time the inclusion test is true, the function `InverseMidPointJacobian` resorts to the first two steps shown in Section 3.1 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 [8, 22]. Hence the last call to `I-Newton` (see Section 3.1).

3.5 Existence test

Algorithm 4 described above, due to the switch to `I-Newton`, can sometimes prove the existence of real-valued solutions inside the tiny boxes returned at the end. However, this algorithm is restricted to square constraint systems with zero-dimensional variety and a finite number of solutions. We propose in this section an existence test that can tackle indifferently under-constrained and well-constrained systems. Inspired by our work in global optimization [29], we propose to post-process the boxes returned by a search strategy using `X-Newton`, rather adapted to inequalities (see Algorithm 3), as follows.

All the constraints in a user-defined constraint system are first partitioned into three subsets. The inequality constraints are left unchanged and will be handled as such by our existence test. A second subset comprises "thick" equations, i.e., equations with a non zero-dimensional set of solutions. This occurs 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 π. Provided that the bounded uncertainties are encoded by interval constants, these thick equations $f_j(x) = 0$ are transformed, without loss of information, into two inequalities $0 \leq f_j(x) \leq 0$. The third class includes "true" equalities $f_k(x) = 0$.

[4] If the CP contractor is a constraint propagation algorithm, then a non incremental version should be run. That is, all the contraints must be initially pushed in the propagation queue, or at most the constraints involving the variables the interval of which has been reduced by `X-NewIter`.

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

Every equation in this subset is relaxed into one thick equation $f_k(x) = [-\epsilon_{eq}, +\epsilon_{eq}]$, i.e, two inequalities $f_k(x) - \epsilon_{eq} \leq 0$, $-f_k(x) - \epsilon_{eq} \leq 0$. The precision value ϵ_{eq} is typically tiny, e.g., $\epsilon_{eq} =$ `1.e-8`. We end up with a constraint system S made of a set of inequalities, in which Algorithm 5 tries to guarantee the existence of a (floating-point) solution. Note that the existence test may fail although one such solution exists in the box, like every other existence test. Of course the unicity property is also lost, even in presence of only true equalities, due to the relaxation with ϵ_{eq}.

Algorithm 5 `Existence` (S=(f, x, $[x]$)) : boolean

return f(`RandomProbing`([x])) ≤ 0 **or**
`InHC4`(S) **or**
`InnerLinearization`(S)

The test first randomly picks an n-dimensional point x inside the box $[x]$ (see `RandomProbing` in Algorithm 5). 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.*

Without detailing, `InHC4` and `InnerLinearization` are recent heuristical algorithms able to sometimes extract respectively an inner box and an inner polyhedral region inside a given box [9, 29]. Note that `InnerLinearization` uses a dual extremal interval taylorization to extract an inner polytope [29]. In case of success of one of both inner region extraction algorithms, the existence test succeeds.

There are two different policies for handling equalities in nonconvex systems in presence of roundoff errors due to floating point numbers : either (a) find *approximately* a point that satisfies *exactly* the constraints, or (b) find *exactly* a point that satisfies *approximately* the constraints. The first option is the one usually adopted by the interval community. Their solvers return the best solution as a tiny box of width ϵ_{sol} in which the existence of a real-valued point is (often) guaranteed by interval Newton methods. We have followed the second option by relaxing true equations by two inequalities with an imprecision ϵ_{eq}.

Notice that the two policies are of equal status regarding reliability, although we should highlight that a precision error ϵ_{eq} on the *images* of functions f better corresponds to the defined feasibility problem than a precision ϵ_{sol} on the unknowns [29].

4 First experiments

We have selected a sample of global optimization systems among those tested by an interval Branch and Bound, called here `IBBA+` [24]. `IBBA+` uses contraint propagation and a sophisticated variant of affine arithmetic. From their sample, we extracted (a) the 15 benchmarks that `IBBA+` could not solve in one hour,[6] (b) the 14 most difficult instances, in terms of branching points obtained by `IBBA+`, one in each subserie `exa_b_*` and (c) all the serie `ex6_2_*` in which every system contains a complicated (highly) non polynomial objective function. The reported results have been obtained on very similar computers (`Intel X86`, `3Ghz`).

We have implemented `X-Taylorization` (Algorithm 1) in the Interval-Based EXplorer `Ibex` [10]. We compare several variants of `X-Taylorization` implemented in our recent optimization strategy called `IbexOpt` [29]. The convex taylorization is achieved once at each node of the search tree and is used to reduce (only) the domain of the objective function, using a call to a Simplex algorithm. The columns 2, 3 and 4 of Tables 1 and 2 compare different policies for the corner selection. All three use Hansen's variant to compute the interval gradient (see Section 2.1). All three choose two different corners for computing the interval taylorization, which appeared to be a good compromise on the selected benchmark.[7] The column 2 (2 Rand) includes the results of `X-Taylorization` when two corners are randomly chosen. The column 3 (Gr-Rand) chooses one corner at random and a second one with our first greedy heuristic (see Section 2.2.2). The column 4 (Inf-Sup) chooses $\underline{x}$ and $\overline{x}$. The experiments show that our first greedy heuristic is not very efficient and, surprisingly, that the best results on average are obtained with $\underline{x}$ and $\overline{x}$! This could be explained by a bias related to the modeling of the systems. These results are very preliminary and we have to perform deeper studies to find out an efficient corner selection heuristic.

In the column 5 (Taylor), $\underline{x}$ and $\overline{x}$ have also been chosen, but with the standard interval gradient calculation. Although worse than Hansen's variant on average, the difference is slight in terms of CPU time.

The last two columns report a first comparison between AA (affine arithmetic ; Ninin et al.'s implementation) and `X-Taylorization`. To try to be the fairest, we have transformed `IbexOpt` into a variant `IbexOpt'` very close to `IBBA+` : `IbexOpt'` uses a non incremental version of `HC4` [5] that loops only once on the variables, and a *largest-first* branching strategy. It also picks one point in the current box of each search

[6] Ninin et al. have recently corrected a bug in their affine arithmetic so that they can solve 2 additional systems before the timeout, as shown in the tables.

[7] N.B. The selection of two corners instead of only one mainly explains the improvement, on average, w.r.t. the results reported in the companion paper [29] on the same benchmark.

node to try to improve the upper bound. Therefore we guess that only the convexification method differs between the two competitors. Three main conclusions can be drawn. First, `X-Taylorization` seems not competitive with AA on 4 benchmarks : `ex2_1_7`, `ex2_1_10`, `ex3_1_1`, `ex5_4_2`. Second, the comparison in the number of branching points underlines that AA contracts generally more than `X-Taylorization`, except in four cases, e.g., `ex14_2_6`. Third, except for the 4 bad results mentioned above, `IbexOpt'` endowed with `X-Taylorization` is generally faster than `IBBA+`. Future studies will determine whether it comes from implementation issues or from the fact that `X-Taylorization` is a fast algorithm for computing a polyhedral relaxation.

Remarks

`X-Taylorization` (called `OuterLinearization` in [29]) is one of the five ingredients of our interval branch and bound. A qualitative experimental study based on Tables 1 and 2 of the companion paper [29] concludes that `X-Taylorization` is the ingredient that has the most significant impact on performance.

We have just implemented a first version of `X-Newton`, close to the pseudo-code shown in Algorithm 3. `X-Newton` seems not useful/efficient for square systems, but seems more robust in global optimization than `X-Taylorization` alone. Recall that `X-Newton` can contract the domain of all the variables, perform several `X-NewIter` iterations and call an interleaved CP contractor, i.e., `Mohc` [2]. `IbexOpt` endowed with `X-Newton` is sometimes worse (less than a factor 2 on difficult systems), but is also even faster than `IbexOpt` endowed with `X-Taylorization` on several systems that can thus be solved within the time limit.

System	2 Rand	Gr-Rand	Inf-Sup	Taylor	IbexOpt'	IBBA+
hs071	0.15	0.16	0.11	0.23	0.24	1.04
	363	350	273	578	1020	804
ex2_1_7	46.9	56.5	6.75	4.58	TO	16.75
	16479	18971	2320	2320		1574
ex2_1_9	31.24	35.0	31.58	213.5	54.5	154
	30355	31632	30425	268845	132358	60007
ex2_1_10	1.1	1.39	0.39	0.25	MO	5.91
	384	420	142	142		636
ex3_1_1	1.74	2.23	1.36	3.79	TO	115.92
	1896	2198	1516	4249		131195
ex4_1_8	0.01	0.01	0.01	0.01	0.01	0.11
	16	19	13	13	72	128
ex5_2_4	0.41	0.46	0.33	0.3	38.13	11.3
	579	652	479	479	112381	9848
ex5_4_2	0.26	0.46	0.23	0.44	TO	121.45
	353	557	344	643		201630
ex6_1_1	22.36	24.21	18.07	18.27	TO	TO
	18480	18603	14725	16980		
ex6_1_3	752.8	983.4	587.9	459	TO	TO
	269729	310843	204439	213582		
ex6_1_4	1.63	1.75	1.25	1.09	1.69	2.7
	1397	1380	1097	1125	3723	1622
ex6_2_5	TO	TO	TO	TO	TO	TO
ex6_2_6	932.2	940.6	917.8	799.2	1108	1575
	873263	806654	858983	863060	1714493	922664
ex6_2_7	TO	TO	TO	TO	TO	TO
ex6_2_8	126.8	134.8	118.1	105.7	306.68	458
	105889	105703	97554	100110	501356	265276

Tab. 1 – First experimental results. The first column contains the name of the handled system. 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 timeouts (TO = 1 hour) have been used by `IbexOpt` and `IBBA+`. Cases of memory overflow (MO) sometimes occur.

5 Conclusion

The adaptation of the first-order Taylor form to intervals produces a nonconvex approximation of the solution set, which has caused loss of performance in interval solvers for decades.

By choosing a corner of the studied box, the extremal interval taylorization addresses this issue by producing a convex approximation that can be hulled in polynomial time. We have proven that the selection of the best corner, allowing the tightest relaxation, is NP-hard, which opens the door to the search for efficient heuristics. Extremal interval taylorization can be used to build an eXtremal interval Newton for handling non convex constraint satisfaction and optimization problems. We hope these convex interval taylorization and extremal interval Newton will be able to complement affine arithmetic, interval reformulation-linearization techniques like `Quad` and the standard interval Newton.

Acknowledgments

We would like to particularly thank G. Chabert and B. Neveu for useful discussions about existing interval analysis results and for the detailed proof-reading of the NP-hardness result.

Références

[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 Distillition Column with Affine Arithmetic. *AIChE Journal*, 55(7) :1695–1704, 2009.

[4] O. Beaumont. *Algorithmique pour les intervalles*. PhD thesis, Université de Rennes, 1997.

System	2 Rand	Gr-Rand	Inf-Sup	Taylor	IbexOpt'	IBBA+
ex6_2_9	38.44	41.32	38.78	33.61	477.9	522
	27809	27775	27461	27898	753449	203775
ex6_2_10	TO	TO	TO	TO	TO	TO
ex6_2_11	10.66	18.83	23.28	23.47	24.99	140
	11487	19088	24264	28455	58911	83487
ex6_2_12	167.6	183.4	128.5	108.6	130.3	113
	113629	115764	86722	87047	322984	58231
ex6_2_13	TO	TO	TO	TO	TO	TO
ex6_2_14	1.77	1.75	1.67	1.54	47.7	207
	1266	1243	1237	1237	111903	95170
ex7_2_1	1.34	1.24	1.22	1.38	212.7	24.7
	1328	1144	1197	1410	51162	8419
ex7_2_3	TO	TO	TO	TO	TO	TO
ex7_3_1	0.01	0.01	0.01	0	0.3	3.5
	11	11	11	11	1087	1536
ex7_3_4	1.41	1.46	1.38	1.33	MO	TO
	797	762	775	872		
ex7_3_5	TO	1355	197.5	330.5	MO	TO
		147509	895907	231425		
ex8_1_7	0.78	0.69	0.77	0.96	1.5	2.64
	1429	1185	1393	1883	4789	1432
ex9_2_1	0	0	0.01	0	0.25	0.26
	9	11	9	9	666	64
ex9_2_2	0.02	0.04	0.02	0.02	MO	TO
	24	75	23	23		
ex9_2_6	0.08	0.26	0.07	0.07	MO	TO
	67	294	57	57		
ex14_1_2	0.53	0.61	0.44	0.54	18.32	54.58
	518	555	442	666	30227	24166
ex14_1_7	472.5	364.1	497.3	598.9	TO	TO
	168372	121730	179993	284106		
ex14_2_6	1.33	1.51	1.37	1.28	42.5	237
	501	531	515	649	27706	74630
ex14_2_7	105.8	121.8	89.98	90.4	TO	TO
	20109	21971	16759	26816		

TAB. 2 – First experimental results.

[5] F. Benhamou, F. Goualard, L. Granvilliers, and J.-F. Puget. Revising Hull and Box Consistency. In *Proc. ICLP*, pages 230–244, 1999.

[6] F. Benhamou and L. Granvilliers. *Continuous and Interval Constraints*. Elsevier, 2006. chapter 16 of : Handbook of Constraint Programming.

[7] C. Bliek. *Computer Methods for Design Automation*. PhD thesis, MIT, 1992.

[8] G. Chabert. *Techniques d'intervalles pour la résolution de systèmes d'intervalles*. PhD thesis, Université de Nice–Sophia, 2007.

[9] G. Chabert and N. Beldiceanu. Sweeping with Continuous Domains. In *Proc. CP, LNCS 6308*, pages 137–151, 2010.

[10] G. Chabert and L. Jaulin. Contractor Programming. *Artificial Intelligence*, 173 :1079–1100, 2009.

[11] L. de Figueiredo and J. Stolfi. Affine Arithmetic : Concepts and Applications. *Numerical Algorithms*, 37(1–4) :147–158, 2004.

[12] E. Hansen. *Global Optimization using Interval Analysis*. Marcel Dekker inc., 1992.

[13] E.R. Hansen. On Solving Systems of Equations Using Interval Arithmetic. *Mathematical Comput.*, 22 :374–384, 1968.

[14] E.R. Hansen. Bounding the Solution of Interval Linear Equations. *SIAM J. Numerical Analysis*, 29(5) :1493–1503, 1992.

[15] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis*. Springer, 2001.

[16] R. B. Kearfott. *Rigorous Global Search : Continuous Problems*. Kluwer Academic Publishers, 1996.

[17] V. Kreinovich, A.V. Lakeyev, J. Rohn, and P.T. Kahl. *Computational Complexity and Feasibility of Data Processing and Interval Computations*. Kluwer, 1997.

[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] O. Lhomme. Consistency Techniques for Numeric CSPs. In *IJCAI*, pages 232–238, 1993.

[20] R. E. Moore. *Interval Analysis*. Prentice-Hall, 1966.

[21] R.E. Moore, R. B. Kearfott, and M.J. Cloud. *Introduction to Interval Analysis*. SIAM, 2009.

[22] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge Univ. Press, 1990.

[23] A. Neumaier and O. Shcherbina. Safe Bounds in Linear and Mixed-Integer Programming. *Mathematical Programming*, 99 :283–296, 2004.

[24] J. Ninin, F. Messine, and P. Hansen. A Reliable Affine Relaxation Method for Global Optimization. *Accepted for publication in Mathematical Programming*, 2011.

[25] W. Oettli. On the Solution Set of a Linear System with Inaccurate Coefficients. *SIAM J. Numerical Analysis*, 2(1) :115–118, 1965.

[26] H. Ratschek and J. Rokne. *New Computer Methods for Global Optimization*. Halsted Press New York, 1988.

[27] J. Rohn. Cheap and Tight Bounds : The Recent Result by E. Hansen can be Made More Efficient. *Interval Computations*, 1 :13–21, 1993.

[28] T. J. Schaefer. The Complexity of Satisfiability Problems. In *Proc. STOC, ACM symposium on theory of computing*, pages 216–226, 1978.

[29] G. Trombettoni, I. Araya, B. Neveu, and G. Chabert. Régions intérieures et linéarisations par intervalles en optimisation globale. In *JFPC (submitted)*, 2011.

[30] G. Trombettoni and G. Chabert. Constructive Interval Disjunction. In *Proc. CP, LNCS 4741*, pages 635–650, 2007.

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

[32] X.-H. Vu, D. Sam-Haroud, and B. Faltings. Enhancing Numerical Constraint Propagation using Multiple Inclusion Representations. *Annals of Math. and Artificial Intelligence*, 55(3–4) :295–354, 2009.