Convex Interval Taylorization in Constrained Global Optimization

Ignacio Araya, Gilles Trombettoni, Bertrand Neveu

UTFSM (Chile), INRIA, I3S, Université Nice–Sophia (France), Imagine LIGM
Université Paris–Est (France)
iaraya@inf.utfsm.cl,Gilles.Trombettoni@inria.fr,neveub@certis.enpc.fr

Abstract. Interval taylorisation has been proposed in the sixties by the interval analysis community for relaxing and filtering continuous constraint systems. Unfortunately, it generally produces a nonconvex relaxation of the solution set. A recent interval Branch & Bound for global optimization, called `IbexOpt`, generates a convex (polyhedral) approximation of the system at each node of the search tree by performing a specific interval taylorization. Following the works by Lin and Stadtherr, the idea is to select a *corner* of the studied domain/box as expansion point, instead of the usual midpoint.
This paper studies how to better exploit this interval convexification. We first show that selecting the corner which produces the tightest relaxation is NP-hard. We then propose a greedy corner selection heuristic, a variant using several corners simultaneously and an interval Newton that iteratively calls this interval convexification. Experiments on a constrained global optimization benchmark highlight the best variants and allow a first comparison with affine arithmetic.

1 Motivation

Interval Newton is an operator often used by interval methods to contract/filter the search space [11]. Interval Newton uses *interval taylorization* 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 nonconvex 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 [20, 14, 19].) Contracting optimally a box containing this nonconvex relaxation has been proven to be NP-hard [15]. This explains why the interval analysis community has worked a lot on this problem for decades [11].

Only a few polynomial subclasses have been studied. The most interesting one has been first described by Oettli and Prager in the sixties [23] 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 2^n subproblems/quadrants before falling in this interesting subclass [1, 5, 8]. Hansen and Bliek propose independently a sophisticated and beautiful algorithm for avoiding to explicitly handle the 2^n quadrants [13, 7]. However, the method requires the system be first preconditioned (i.e., the interval Jacobian matrix must be multiplied by

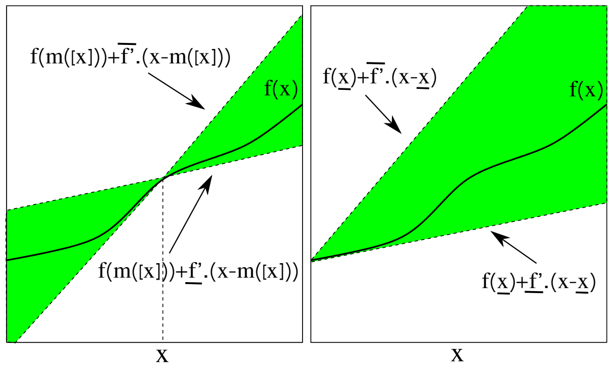

Fig. 1. Relaxation of a function f over the real numbers by a function $g : \mathbb{R} \rightarrow \mathbb{IR}$ using interval taylorization (graph in green). **Left:** Midpoint taylorization, using a midpoint evaluation $f(m([x]))$, the maximum derivative $\overline{f'}$ of f inside the interval $[x]$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

the inverse matrix of its midpoint). It is restricted to $n \times n$ (square) systems of equations (no inequalities). The preconditioning has a cubic time complexity, implies an overestimate of the relaxation and requires non-singularity conditions often met only at the bottom of the search tree.

In 2004, Lin & Stadtherr [17] have proposed to select a *corner* of the studied box/domain, instead of the usual midpoint. Graphically, it produces a convex cone, as shown in Fig. 1-right. The main drawback of this *extremal* interval taylorization 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 an interior point algorithm or, in practice, by a Simplex algorithm: two calls to a Simplex algorithm can compute the minimum (resp. maximum) value for each of the n variables (see Section 3).

A basic corner-based interval taylorization has been recently embedded in an interval branch and bound for constrained global optimization [25]. Let us introduce definitions and background before describing this simple convexification/linearization method.

Intervals

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

An **interval** $[x_i] = [\underline{x_i}, \overline{x_i}]$ defines the set of reals x_i s.t. $\underline{x_i} \leq x_i \leq \overline{x_i}$, where $\underline{x_i}$ and $\overline{x_i}$ are floating-point numbers. $\mathbb{IR}$ denotes the set of all intervals. The size or **width** of $[x_i]$ is $w([x_i]) = \overline{x_i} - \underline{x_i}$. 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 **hull** of a subset S of $\mathbb{R}^n$ is the smallest n-dimensional box enclosing S.

Interval arithmetic [18] 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 $f : \mathbb{R}^n \rightarrow \mathbb{R}$.
$[f] : \mathbb{IR}^n \rightarrow \mathbb{IR}$ is said to be an **extension** *of f to intervals iff:*

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

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** [18], 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: $\frac{\partial f}{\partial x_1}(x_1, x_2) = 6x_1 + x_2$, $[\frac{\partial f}{\partial x_1}]_n([-1, 3], [-1, 5]) = [-7, 23]$, $\frac{\partial f}{\partial x_2}(x_1, x_2) = x_1 + 2x_2$, $[\frac{\partial f}{\partial x_2}]_n([x_1], [x_2]) = [-3, 13]$. The interval Taylor evaluation with $\dot{x} = m([x]) = (1, 2)$ yields: $[f]_t([x_1], [x_2]) = 9 + [-7, 23] * [-2, 2] + [-3, 13] * [-3, 3] = [-76, 94]$.

A simple convex interval taylorization

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 implemented in our interval B&B 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 $y_i^r = x_i - \overline{x_i}$.

A *corner* of the box is chosen: $\underline{x}$ 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.

The correction of relation (1) – see for instance [25, 17] – lies on the simple fact that any variable y_i^l is positive since its domain is $[0, d_i]$, with $d_i = w([y_i^l]) =$

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

$w([x_i]) = \overline{x_i} - \underline{x_i}$. 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 the 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 [21], using interval arithmetic, must be added to guarantee that no solution is lost by the Simplex algorithm.

Endpoint taylorization in constrained global optimization

This simple convexification algorithm has been implemented in a new interval B&B called `IbexOpt` [25]. This global optimizer minimizes an objective function $f(x)$ subject to inequality and equality constraints, i.e.:

$$\min_{x \in [x]} f(x) \;\; \textit{subject to} \;\; g(x) \leq 0, \; h(x) = [-\epsilon_{eq}, +\epsilon_{eq}].$$

Note that equations are relaxed with a (tiny) admissible precision error ϵ_{eq}, (e.g., ϵ_{eq}=`1e-8`). That is, all the constraints are viewed as inequalities: $\{g(x) \leq 0, h(x) - \epsilon_{eq} \leq 0, -h(x) - \epsilon_{eq} \leq 0\}$.

`IbexOpt` proposes recent and new algorithms to handle constrained optimization problems. Contraction steps are achieved by the `Mohc` interval constraint propagation algorithm [3]. The upper bounding phase uses original algorithms for extracting *inner regions* inside the feasible search space, i.e., zones in which all points satisfy the inequality and (relaxed) equality constraints. The cost of any point inside an inner region can improve the upper bound.

Also, at each node of the B&B, the endpoint interval taylorization introduced above (called `OuterLinearization` in [25]) is used to produce a polytope enclosing all the constraints and the objective function. A lower bound is obtained by one call to a Simplex algorithm minimizing the linearized objective function on this polytope. The first implementation was very simple: every inequality constraint was linearized with the form (1). The present paper elaborates on this first choice by exploiting the ideas presented in Sections 2 and 3.

Related Work and contributions

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 [20]) for computing a range enclosure (see Def. 1) of a univariate function. Neumaier calls this the *linear boundary value form*. At page 211 of the same book, the step (4) of the presented pseudocode also uses an endpoint interval taylorization for contracting a system of equations. The aim is not to produce a polyhedral relaxation (which is not mentioned), but to use as expansion point the farthest point from a current point followed by the algorithm in the domain.[2]

Lin & Stadtherr proposed in 2004 an interval Newton based on an endpoint interval taylorization [17] close to the one presented in Section 2. Their interval

[2] The contraction is not obtained by calls to a Simplex algorithm but by an interval Gauss-Seidel iteration that also works for *nonconvex* systems of equations with linear coefficients and does not necessarily converge in polynomial-time.

Newton is 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.

Our interval Newton can treat under-constrained systems (with less equations than variables and with inequalities) encountered in constrained global optimization. The preconditioning of Lin & Stadtherr is not relevant in this more general context and we have rather sought for a corner selection heuristic that brings the best filtering of the solution set. We prove in Section 2 that the choice of the best expansion corner for any constraint is an NP-hard problem and propose a first greedy algorithm to select a corner in a heuristic way. We finally underline that several corners can be selected for every constraint in order to produce a tighter polytope. Tighter interval partial derivatives can also be produced by a Hansen's recursive variant of interval taylorization. Section 3 describes an eXtremal interval Newton algorithm (`X-Newton`) that iteratively computes a convex interval taylorization.

Section 5 shows experiments in global optimization that highlight the best variants of convex interval taylorization and `X-Newton`. This work provides an alternative to the two existing reliable (interval) convexification methods used in global optimization. The `Quad` [16] method is an interval reformulation-linearization technique that produces a polyhedral approximation of the quadratic terms of constraints. *Affine arithmetic* produces a polytope by replacing in the constraint expressions every basic operator with specific affine forms [10, 26, 4]. It has been recently implemented in an efficient interval B&B [22]. Experiments provide a first comparison between this affine arithmetic and our corner-based taylorization.

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 [12, 11]. 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 Hansen's variant of the form (1), for instance, in which the scalar coefficients $\underline{a_i}$ are:

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

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 x^c such that:

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

where: $z_i = \overline{a_i}(x_i - \overline{x_i})$ iff $x_i^c = \overline{x_i}$, and $z_i = \underline{a_i}(x_i - \underline{x_i})$ 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 depends on x_i but 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:

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

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 $f : \mathbb{R}^n \to \mathbb{R}$, 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 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)[3] 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.
- 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 $x_1'^j * x_2'^j * x_3'^j = 0$.
- 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 $-\overline{a_i}$ 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 tranformation 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 tranformation in which the derivative part is constant, i.e., $\forall_i \; \underline{a_i} = -\overline{a_i}$. Thus:

Proposition 1 *(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,

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

is an NP-hard problem.

[3] 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.

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 [24]. `BALANCED-3SAT` does not belong to none of these 6 subclasses.[4]

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 $\frac{\partial f}{\partial x_i'}([x])$ 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., $x_k'^j = (1 - x_i')$ and $[-1, 0] = -1 * [0, 1] * [0, 1]$), and equal to $[0, 1]$ if x_i appears as a negative literal (i.e., $x_k'^j = x_i'$ 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 $\forall_i \ \underline{a_i} = -\overline{a_i}$ □

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. More precisely, different polytopes computed at different search nodes achieve a type of intersection of polytopes (see Section 2.3).

We also propose a greedy heuristic for the corner selection, based on Expression (4). 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: $max_{x^c \in X^c} \quad \sum_i fh(x_i^c) + 0.5 \sum_i a_i^c \, d_i$ that can be maximized componentwise by computing the sign of the following quantity:

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

[4] 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...).

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 $x_i^c := \overline{x_i}$,
- if $g(i) < 0$, then $x_i^c := \underline{x_i}$.

2.3 Intersection of extremal taylorization relaxations

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

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 $[x]$. This algorithm is parameterized by:

- the interval gradient procedure used: standard or Hansen's variant (see Section 2.1),
- the c-set of functions producing the c different hyperplanes for a given inequality. For instance, the set {`random`, `greedy`} means that two corners are used to produce two hyperplanes: one selected at random and one selected with the greedy heuristic described above.

3 eXtremal interval Newton

We first describe in Section 3.1 a well-known algorithm for computing the (box) hull of the polytope produced by `X-Taylorization`. We then detail in Section 3.2 how this `X-NewIter` procedure is iteratively called in the `X-Newton` algorithm until a quasi-fixpoint is reached in terms of contraction.

3.1 X-Newton iteration

Algorithm 1 describes a well-known algorithm used in several solvers (see for instance [16, 4]). A specifity is the use of a corner-based interval taylorization (`X-Taylorization`) for computing the polytope.

Algorithm 1 `X-NewIter` (f, x, $[x]$,`Hansen?`,C): $[x]$

```
for j from 1 to m do
  polytope ← polytope ∪ {X-Taylorization(f_j,x,[x],Hansen?,C)}
end for
for i from 1 to n do
  /* Two calls to a Simplex algorithm: */
  x̲_i ← min x_i subject to polytope
  x̄_i ← 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 $f = (f_1, ..., f_j, ..., f_m)$. $x = (x_1, ..., x_i, ..., x_n)$ denotes the set of variables with domains $[x]$. `Hansen?` is a boolean indicating whether Hansen's variant is used to compute the partial derivatives. C is a c-list of corner selection procedures.

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 our interval B&B, `X-NewIter` allows the strategy to not only improve the lower bound of the objective function but also contract the current box in (potentially) all the dimensions.

To deal with round-off errors made by the Simplex algorithm, a cheap post-processing using interval arithmetic is added to guarantee that no solution is lost [21].

3.2 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` $= \perp$.

Algorithm 2 `X-Newton` (f, x, $[x]$, `Hansen?`, C, `ratio_fp`, `CP-contractor`): $[x]$

```
repeat
  [x]_save ← [x]
  [x] ← X-NewIter (f, x, [x], Hansen?, C)
  if CP-contractor ≠ ⊥ and gain([x],[x]_save) > 0 then
    [x] ← CP-contractor(f,x,[x])
  end if
until empty([x]) or gain([x],[x]_save) < ratio_fp)
return [x]
```

`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 `X-NewIter` has reduced the domain, then the `Mohc` algorithm [3] can further contract the box, before waiting the next choice point.[5] The guard $\texttt{gain}([x], [x]_{save}) > 0$ guarantees that `CP-contractor` be not called twice if `X-NewIter` does not contract the box.

4 Discussion

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.[6] It is however possible to switch from an endpoint taylorization to a midpoint one and thus be able to obtain quadratic convergence, as detailed in [2].

`X-Taylorization` is a simple algorithm for computing a polyhedral relaxation. It is fast and the simplest version can be encoded in 50 lines of codes. However, the power of contraction/filtering needs be evaluated, especially because the interval taylorization is known to generally produce a tight relaxation on small boxes. First experimental results shown below provide a first comparison with affine arithmetic in terms of contraction.

Overall, compared to the first `X-Taylorization` implemented in our interval B&B [25], we have described in this paper 6 ways to better contract the domains: (a) Hansen's variant for partial derivatives computation, (b) a greedy or (diversifying) random corner selection, (c) intersection of extreme taylorizations, (d) contraction on all the variables, instead of only the lower bound of the objective function (`X-NewIter`), (e) `X-Newton` iterations, (f) the `Mohc` algorithm interleaved with `X-NewIter` inside `X-Newton`. Experiments shown below try to underline the best variants.

5 First experiments

We have selected a sample of global optimization systems among those tested by our best competitor, an interval Branch and Bound called here `IBBA+` [22]. `IBBA+` uses contraint propagation and a sophisticated variant of affine arithmetic. From their benchmark, we have extracted the 27 systems that required more than 1 second to be solved by the first version of `IbexOpt` (column Rand).[7] 3 systems (`ex6_2_5`, `ex6_2_7` and `ex6_2_13`) are removed from the benchmark because they

[5] 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`.

[6] Quadratic convergence can be achieved on $n \times n$ systems of equations by the standard interval Newton when very strict conditions are met, a necessary condition being the existence of a unique solution inside the box, which generally occurs at the bottom of the search tree [14, 8].

[7] Our first interval B&B used in fact the corner $\underline{x}$ (form 1), but a random corner provides a slightly better contraction due to the various polytopes produced, as mentioned in Section 2.

are not solved by any solver. Table 1 corresponds to the 11 systems solved by this first version in less than 11 seconds. Table 2 includes the 13 systems solved in more than 11 seconds. The reported results have been obtained on a same computer (`Intel X86`, `3Ghz`). We have implemented the different algorithms in the Interval-Based EXplorer `Ibex` [9].

Table 1. First experimental results on mean-difficult global optimization systems

System	n	No	Rand	R+R	R+op	Greedy	Best	B+op	XIter	XNewt	Ibex'	Ibex"	IBBA+
ex2_1_8	24	TO	10.50	10.27	9.32	3.27	TO	TO	8.43	8.92	47.96	TO	26.78
			3605	2739	2444	989			1068	418	38988		1916
ex3_1_1	8	MO	1.91	1.75	1.28	2.05			1.24	1.87	MO	121	116
			2429	1877	1529	2544	1851	1516	676	428		36689	131195
ex6_1_4	6	MO	1.74	1.48	1.10	2.18			1.40	1.55	1.82	2.30	2.70
			1844	1359	1069	2100	1830	1097	796	540	4218	2215	1622
ex6_2_14	4	2.16	1.74	1.68	1.58	1.55			1.58	1.49	44.53	65.26	208
		1421	1290	1264	1247	1242	1369	1237	1066	742	109745	104483	95170
ex7_2_1	7	883	1.23	1.28	1.22	1.20			0.49	0.45	13.74	5.45	24.72
		1.2e+6	1410	1314	1280	1262	1636	1336	260	153	33478	5139	8419
ex7_2_6	3	10.52	9.42	6.63	1.24	10.61			4.22	2.74	0.11	0.16	1.23
		71447	31601	20874	3425	36566	37026	12179	9211	4272	570	436	1319
ex7_3_4	12	39.08	1.11	1.33	1.28	1.19			1.66	2.25	TO	TO	TO
		38291	818	793	770	808	789	760	441	334			
ex14_2_1	5	7.57	1.04	1.09	0.95	1.12			0.68	0.88	8.97	21.20	36.73
		7374	768	689	619	740	749	604	336	198	14476	22720	16786
ex14_2_3	6	20.21	2.82	3.20	2.91	3.44			1.75	2.62	64.22	30.81	TO
		11557	1203	1150	1081	1305	1533	979	525	376	55347	19410	
ex14_2_4	5	0.96	1.09	1.33	1.04	1.08			0.65	1.09	35.32	36.80	128
		657	588	490	471	521	545	481	229	220	34240	28249	30002
ex14_2_6	5	1.11	1.20	1.21	1.24	1.40			1.05	1.21	42.61	72.52	238
		689	578	459	501	611	578	484	368	234	74630	32675	74630
Sum			33.80	31.25	23.16	29.09			23.15	25.07	147	203	638
			46134	33308	14436	48688			14976	7915	229402	208268	227948
Gain			1	1.02	1.71	1.14			1.50	1.40			

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+`. Cases of memory overflow (MO) sometimes occur. For each method m, the last line includes an average gain among the different systems. For a given system, the gain w.r.t. the basic method (column Rand) is $\frac{CPU\,time(Rand)}{CPU\,time(m)}$. The last 10 columns of Tables 1 and 2 compare different variants of `X-Taylorization` 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 2.1). The gain is generally slight but Hansen's variant is more robust: for instance `ex_7_2_3` cannot be solved with the basic interval gradient calculation.

In the column *No*, the convexification operator is removed from our interval B&B, which underlines its significant benefits in practice. In columns 4 to 9, one produces a polytope for lower bounding the objective function (only one call to a Simplex algorithm) while in columns 10 and 11, $2n$ calls to a Simplex algorithm are achieved.

The column *Rand* corresponds to an X-Taylorization performed with one corner randomly picked for every constraint. The next column (R+R) corre-

Table 2. First experimental results on difficult constrained global optimization systems

System	n	No	Rand	R+R	R+op	Greedy	Best	B+op	XIter	XNewt	Ibex'	Ibex"	IBBA+
ex2_1_7	20	TO	42.96 20439	43.17 16492	40.73 15477	58.42 28858	TO	TO	7.74 1344	10.58 514	TO	TO	16.75 1574
ex2_1_9	10	MO	40.09 49146	29.27 30323	22.29 23232	44.66 49039	57560	26841	9.07 5760	9.53 1910	46.58 119831	103 100987	154.02 60007
ex6_1_1	8	MO	20.44 21804	19.08 17104	17.23 14933	23.81 23378	24204	15078	31.24 14852	38.59 13751	TO	633 427468	TO
ex6_1_3	12	TO	1100 522036	711 269232	529 205940	3014 1.3e+6	TO	TO	262.5 55280	219 33368	TO	TO	TO
ex6_2_6	3	TO	162 172413	175 168435	169 163076	173 171231	171235	162844	172 140130	136 61969	1033 1.7e+6	583 770332	1575 922664
ex6_2_8	3	97.10 119240	121 117036	119 105777	110 97626	129.2 117047	117062	97580	78.1 61047	59.3 25168	284 523848	274 403668	458 265276
ex6_2_9	4	25.20 27892	33.0 27892	36.7 27826	35.82 27453	35.63 27881	27881	27457	42.34 27152	43.74 21490	455 840878	513 684302	523 203775
ex6_2_10	6	TO	3221 1.6e+6	2849 1.2e+6	1924 820902	TO	1.1e+6	820611	2218 818833	2697 656360	TO	TO	TO
ex6_2_11	3	10.57 17852	19.31 24397	7.51 8498	7.96 8851	12.48 14669	5606	27016	13.26 12253	11.08 6797	41.21 93427	11.80 21754	140.51 83487
ex6_2_12	4	2120 2e+6	232 198156	160 113893	118.6 86725	265 197462	191390	86729	51.31 31646	22.20 7954	122 321468	187 316675	112.58 58231
ex7_3_5	13	TO	44.7 45784	54.9 44443	60.3 50544	47.2 42553	45352	42453	29.88 6071	28.91 5519	TO	TO	TO
ex14_1_7	10	TO	433 223673	445 172671	406 156834	334 158077	165327	109685	786 179060	938 139111	TO	TO	TO
ex14_2_7	6	93.10 35517	94.16 25802	102.2 21060	83.6 16657	94.1 21084	20273	18126	66.39 12555	97.36 9723	TO	TO	TO
Sum			5564 3.1e+6	4752 2.2e+6	3525 1.7e+6	7831 3.7e+6			3767 1.4e+6	4311 983634	1982 3.6e+6	1672 2.3e+6	2963 1.6e+6
Gain			1	1.21	1.39	0.94			2.23	1.78			
ex7_2_3	8	MO	MO	MO	MO	MO			544 611438	691 588791	TO	719 681992	TO

sponds to a tighter polytope computed with two randomly chosen corners. The gain is slight w.r.t. *Rand*. The next column (R+op) highlights the best `X-Taylorization` variant where are chosen a random corner c and the opposite corner of c. The success of this combination is explained by the fact that the two opposite convexifications are very different, picking opposite bounds in the linear system with interval coefficients. Several other combinations with more than 2 corners appeared to be counter-productive and are not reported here.

The column *Greedy* shows the generally poor results obtained by our greedy corner selection heuristic. After having tried numerous variants (including a hill-climbing method), we have performed a very informative experiment whose results are shown in columns *Best* and *B+op*: an exponential algorithm selects the best corner, maximizing the expression (4), among the 2^n ones.[8] 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 diversifying random corner (resp. *R+op*). Therefore, the combination *R+op* has been kept in all the remaining variants.

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

[8] We could not thus compute the number of branching nodes of systems with more than 12 variables because they reached the timeout.

The last three columns report a first comparison between AA (affine arithmetic; Ninin et al.'s implementation) 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` [6] 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.[9]

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 can compensate the lack of contraction of `X-Taylorization`. Finally, the performances of `Ibex'` and `Ibex''` are better than `IBBA+` one, but one cannot conclude if it comes from a faster convexification method or from a better implementation of the whole strategy.

6 Conclusion

By choosing a corner of the studied box, extremal interval taylorization produces 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 would have opened the door to the search for efficient heuristics. However, we have experimentally shown that the best corner often brings a very small additional contraction w.r.t. a random corner *and* its opposite corner.

This convex interval taylorization 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 a sample of constrained global optimization systems. Compared to our first version, our optimizer endowed with `X-NewIter` can solve one additional system before the timeout and obtains an average speedup of more than 2 on the other difficult systems.

Compared to affine arithmetic, first experiments suggest that our convex interval taylorization produces a looser relaxation in less CPU time. Also, the additional job achieved by `X-Newton` can compensate this lack of filtering (so that one can solve one additional 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.

[9] We have removed the call to `Mohc` inside the `X-Newton` loop (i.e., `CP-contractor=`$\perp$) because this constraint propagation algorithm is not a convexification method.

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 and G. Trombettoni. Convex Interval Taylorization: some theoretical and algorithmical results. In *Research report in preparation*, 2011.
3. I. Araya, G. Trombettoni, and B. Neveu. Exploiting Monotonicity in Interval Constraint Propagation. In *Proc. AAAI*, pages 9–14, 2010.
4. A. Baharev, T. Achterberg, and E. Rév. Computation of an Extractive Distillition Column with Affine Arithmetic. *AIChE Journal*, 55(7):1695–1704, 2009.
5. O. Beaumont. *Algorithmique pour les intervalles.* PhD thesis, Université de Rennes, 1997.
6. F. Benhamou, F. Goualard, L. Granvilliers, and J.-F. Puget. Revising Hull and Box Consistency. In *Proc. ICLP*, pages 230–244, 1999.
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 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. E. Hansen. *Global Optimization using Interval Analysis.* Marcel Dekker inc., 1992.
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. R. B. Kearfott. *Rigorous Global Search: Continuous Problems.* Kluwer Academic Publishers, 1996.
15. V. Kreinovich, A.V. Lakeyev, J. Rohn, and P.T. Kahl. *Computational Complexity and Feasibility of Data Processing and Interval Computations.* Kluwer, 1997.
16. 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.
17. Y. Lin and M. Stadtherr. LP Strategy for the Interval-Newton Method in Deterministic Global Optimization. *Industrial & engineering chemistry research*, 43:3741–3749, 2004.
18. R. E. Moore. *Interval Analysis.* Prentice-Hall, 1966.
19. R.E. Moore, R. B. Kearfott, and M.J. Cloud. *Introduction to Interval Analysis.* SIAM, 2009.
20. A. Neumaier. *Interval Methods for Systems of Equations.* Cambridge Univ. Press, 1990.
21. A. Neumaier and O. Shcherbina. Safe Bounds in Linear and Mixed-Integer Programming. *Mathematical Programming*, 99:283–296, 2004.
22. J. Ninin, F. Messine, and P. Hansen. A Reliable Affine Relaxation Method for Global Optimization. *Accepted for publication in Mathematical Programming*, 2011.
23. W. Oettli. On the Solution Set of a Linear System with Inaccurate Coefficients. *SIAM J. Numerical Analysis*, 2(1):115–118, 1965.
24. T. J. Schaefer. The Complexity of Satisfiability Problems. In *Proc. STOC, ACM symposium on theory of computing*, pages 216–226, 1978.
25. G. Trombettoni, I. Araya, B. Neveu, and G. Chabert. Inner Regions and Interval Linearizations for Global Optimization. In *AAAI, accepted for publication*, 2011.
26. 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.