A Contractor Based on Convex Interval Taylor
Ignacio Araya, Gilles Trombettoni, Bertrand Neveu

To cite this version:

HAL Id: hal-00733848
https://hal-enpc.archives-ouvertes.fr/hal-00733848
Submitted on 20 Sep 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
A Contractor Based on Convex Interval Taylor

Ignacio Araya, Gilles Trombettoni, Bertrand Neveu

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

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

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

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

1 Motivation

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

Interval Newton is an operator often used by interval methods to contract/filter the search space [12]. The interval Newton operator uses an interval Taylor form to iteratively produce a linear system with interval coefficients. The main issue is that this system is not convex. Restricted to a single constraint, it forms a non-convex cone (a “butterfly”), as illustrated in Fig. 1-left. An n-dimensional constraint system is relaxed by an intersection of butterflies that is not convex either. (Examples can be found in [24, 15, 23].) Contracting optimally a box containing this non-convex relaxation has been proven to be NP-hard [16]. This explains why the interval analysis community has worked a lot on this problem for decades [12].
Fig. 1. Relaxation of a function $f$ over the real numbers by a function $g : \mathbb{R} \to \mathbb{R}$ using an interval Taylor form (graph in gray). \textbf{Left}: Midpoint Taylor form, using a midpoint evaluation $f(m([x]))$, the maximum derivative $f'$ of $f$ inside the interval $[x]$ and the minimum derivative $f'$. \textbf{Right}: Extremal Taylor form, using an endpoint evaluation $f(x)$, $f'$ and $f'$. Only a few polynomial time solvable subclasses have been studied. The most interesting one has been first described by Oettli and Prager in the sixties [27] and occurs when the variables are all non-negative or non-positive. Unfortunately, when the Taylor expansion point is chosen strictly inside the domain (the midpoint typically), the studied box must be previously split into $2^n$ subproblems/quadrants before falling in this interesting subclass [1, 5, 8]. Hansen and Bliek independently proposed a sophisticated and beautiful algorithm for avoiding explicitly handling the $2^n$ quadrants [14, 7]. However, the method is restricted to $n \times n$ (square) systems of equations (no inequalities). Also, the method requires the system be first preconditioned (i.e., the interval Jacobian matrix must be multiplied by the inverse matrix of the domain midpoint). The preconditioning has a cubic time complexity, implies an overestimate of the relaxation and requires non-singularity conditions often met only on small domains, at the bottom of the search tree.

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

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

McAllester et al. also mention this idea in [20] (end of page 2) for finding cuts of the box in constraint systems. At page 211 of Neumaier’s book [24], the step (4) of the presented pseudo-code also uses an endpoint interval Taylor form for contracting a system of equations.\(^1\)

**Contributions**

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

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

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

This work provides an alternative to the two existing reliable (interval) convexification methods used in global optimization. The Quad [18, 17] method is an interval reformulation-linearization technique that produces a convex polyhedral approximation of the quadratic terms in the constraints. Affine arithmetic produces a polytope by replacing in the constraint expressions every basic operator by specific affine forms [10, 32, 4]. It has been recently implemented in an efficient interval B&B [26]. Experiments provide a first comparison between this affine arithmetic and the corner-based interval Taylor.

\(^1\) The aim is not to produce a convex polyhedral relaxation (which is not mentioned), but to use as expansion point the farthest point in the domain from a current point followed by the algorithm. The contraction is not obtained by calls to an LP solver but by the general purpose Gauss-Seidel without taking advantage of the convexity.
2 Background

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

Intervals

An interval \([x_i] = [x_{i1}, x_{i2}]\) defines the set of reals \(x_i\) s.t. \(x_{i1} \leq x_i \leq x_{i2}\), where \(x_{i1}\) and \(x_{i2}\) are floating-point numbers. \(\mathbb{R}\) denotes the set of all \(\mathbb{R}\) intervals. The size or \textbf{width} of \([x_i]\) is \(w([x_i]) = x_{i2} - x_{i1}\). A box \([x]\) is the Cartesian product of intervals \([x_1] \times \ldots \times [x_i] \times \ldots \times [x_n]\). Its width is defined by \(\max_i w([x_i])\). \(m([x])\) denotes the middle of \([x]\). The \textbf{hull} of a subset \(S\) of \(\mathbb{R}^n\) is the smallest \(n\)-dimensional box enclosing \(S\).

Interval arithmetic [22] has been defined to extend to \(\mathbb{R}\) elementary functions over \(\mathbb{R}\). For instance, the interval sum is defined by \(\left[a + b, c + d\right] = [a, c] + [b, d]\). When a function \(f\) is a composition of elementary functions, an \textbf{extension} of \(f\) to intervals must be defined to ensure a conservative image computation.

\begin{definition}[Extension of a function to \(\mathbb{R}\); inclusion function; range enclosure]
Consider a function \(f : \mathbb{R}^n \to \mathbb{R}\).
\[ [f] : \mathbb{R}^n \to \mathbb{R} \]
\(\mathbb{R}\) is said to be an \textbf{extension} of \(f\) to intervals iff:
\[ \forall [x] \in \mathbb{R}^n \quad [f](|[x]|) \supseteq \{f(x), x \in [x]\} \]
\[ \forall x \in \mathbb{R}^n \quad f(x) = [f](x) \]
\end{definition}

The \textbf{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 \textbf{interval Taylor extension} [22], defined as follows:
\[ [f]_T([x]) = f(\hat{x}) + \sum_i [a_i] \cdot (|[x_i] - \hat{x}_i|) \]

where \(\hat{x}\) denotes any point in \([x]\), e.g., \(m([x])\), and \([a_i]\) denotes \(\left[\frac{\partial f_{N}}{\partial x_i}\right]([x])\).

Equivalently, we have: \(\forall x \in [x], \quad [f]_T([x]) \leq f(x) \leq [f]_T([x])\).

\begin{example}
Consider \(f(x_1, x_2) = 3x_1^2 + x_2^2 + x_1x_2\) in the box \([x] = [-1, 3] \times [-1, 5]\). The natural evaluation provides: \([f]_N([x_1], [x_2]) = 3[-1, 3]^2 + [-1, 5]^2 + [-1, 3][-1, 5] = [0, 27] + [0, 25] + [-5, 15] = [-5, 67]\). The partial derivatives are:
\[ \partial_{x_1} f(x_1, x_2) = 6x_1 + x_2, \quad \partial_{x_2} f(x_1, x_2) = x_1 + 2x_2, \]
\[ \frac{\partial f}{\partial x_1}([x_1], [x_2]) = [-3, 13] \]
\end{example}

The interval Taylor evaluation with \(\hat{x} = m([x]) = (1, 2)\) yields: \([f]_T([x_1], [x_2]) = 9 + [-7, 23][-2, 2] + [-3, 13][-3, 3] = [-76, 94]\).
A simple convexification based on interval Taylor

Consider a function $f : \mathbb{R}^n \to \mathbb{R}$ 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 $\left[ \frac{\partial f}{\partial x_i} \right] (x)$. The first idea is to lower tighten $f(x)$ with one of the following interval linear forms that hold for all $x$ in $[x]$.

$$f(x) + a_1 y_l^i + \ldots + a_n y_l^n \leq f(x) \quad (1)$$

$$f(x) + a_1 y_r^i + \ldots + a_n y_r^n \leq f(x) \quad (2)$$

where: $y_l^i = x_i - x_i$ and $y_r^i = x_i - x_i$.

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

The correctness of relation (1) – see for instance [30, 19] – lies on the simple fact that any variable $y_l^i$ is non-negative since its domain is $[0, d_i]$, with $d_i = w([y_l^i]) = w([x_i]) = \overline{x}_i - x_i$. Therefore, minimizing each term $[a_i] y_l^i$ for any point $y_l^i \in [0, d_i]$ is obtained with $a_i$. Symmetrically, relation (2) is correct since $y_r^i \in [-d_i, 0] \leq 0$, and the minimal value of a term is obtained with $\overline{a}_i$.

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

3 Extremal interval Taylor form

3.1 Corner selection for a tight convexification

Relations (1) and (2) consider two specific corners of the box $[x]$. We can remark that every other corner of $[x]$ is also suitable. In other terms, for every variable $x_i$, we can inductively select one of both bounds of $[x_i]$ and combine them in a combinatorial way: either $x_i$ in a term $a_i (x_i - x_i)$, 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? If we consider an inequality $f(x) \leq 0$, we want to compute a hyperplane $f^f(x)$ that approximates the function, i.e., for all $x$ in $[x]$ we want: $f^f(x) \leq f(x) \leq 0$.

Following the standard policy of linearization methods, for every inequality constraint, we want to select a corner $x^c$ whose corresponding hyperplane is the closest to the non-convex solution set, i.e., adds the smallest volume. This is exactly what represents Expression (3) that maximizes the Taylor form for

---

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

all the points \( x = \{ x_1, ..., x_n \} \in [x] \) and adds their different contributions: one wants to select a corner \( x^c \) from the set of corners \( X^c \) such that:

\[
\max_{x^c \in X^c} \int_{x_1 = x_{11}}^{x_{1}} ... \int_{x_n = x_{n2}}^{x_{n}} (f(x^c) + \sum_i z_i) \, dx_n ... \, dx_1 \tag{3}
\]

where: \( z_i = a_i(x_i - \overline{x}_i) \) iff \( x_i^c = \overline{x}_i \), and \( z_i = a_i(x_i - \overline{x}_i) \) iff \( x_i^c = x_i \).

Since:
- \( f(x^c) \) is independent from the \( x_i \) values,
- any point \( z_i \) depends on \( x_i \) but does not depend on \( x_j \) (with \( j \neq i \)),
- \( \int_{x_i = x_{i1}}^{x_{i}} a_i(x_i - \overline{x_i}) \, dx_i = a_i \int_{y_i = -\overline{x}_i}^{y_i} y_i \, dy_i = a_i \cdot 0.5 \, d_i^2 \),
- \( \int_{x_i = x_{i1}}^{x_{i}} a_i(x_i - \overline{x}_i) \, dx_i = a_i \int_{-\overline{d}_i}^{d_i} y_i \, dy_i = -0.5 \, \overline{a}_i \, d_i^2 \).

Expression (3) is equal to:

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

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

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

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

Unfortunately, we have proven that this maximization problem (4) is NP-hard.

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

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

\[
\max_{x^c \in X^c} (f(x^c) + 0.5 \sum_i a_i^c \, d_i)
\]

(or \( \min_{x^c \in X^c} f(x^c) + 0.5 \sum_i a_i^c \, d_i \)) is an NP-hard problem.

The extended paper [3] shows straightforward proofs that maximizing the first term of Expression 4 (\( f(x^c) \)) is NP-hard and maximizing the second term \( 0.5 \sum_i a_i^c \, d_i \) is easy, by selecting the maximum value among \( a_i \) and \( -\overline{a}_i \) in every term. However, proving Proposition 1 is not trivial (see [3]) and has been achieved with a polynomial reduction from a subclass of 3SAT, called BALANCED-3SAT.\(^4\)

\(^3\) We cannot prove anything on more complicated, e.g., transcendental, functions that make the problem undecidable.

\(^4\) In an instance of BALANCED-3SAT, each Boolean variable \( x_i \) occurs \( n_i \) times in a negative literal and \( \overline{n}_i \) times in a positive literal. We know that BALANCED-3SAT is NP-complete thanks to the dichotomy theorem by Thomas J. Schaefer [28].
Even more annoying is that experiments presented in Section 5 suggest that the criterion (4) is not relevant in practice. Indeed, even if the best corner was chosen (by an oracle), the gain in box contraction brought by this strategy w.r.t. a random choice of corner would be not significant. This renders pointless the search for an efficient and fast corner selection heuristic.

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

This qualitative analysis nevertheless provides us rationale to adopt the following policy.

**Using two opposite corners**

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

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

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

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

### 3.2 Preliminary interval linearization

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

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

$$
\forall i \in \{1, \ldots, n\}, [a_i] = \left[ \frac{\partial f}{\partial x_i} \right]_{N} ([x_1] \times \ldots \times [x_i] \times [x_{i+1}] \times \ldots \times [x_n])
$$
where \( x_j \in [x_j] \), e.g., \( x_j = m([x_j]) \).

By following Hansen’s recursive principle, we can produce Hansen’s variant of the form (1), for instance, in which the scalar coefficients \( a_i \) are:

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

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

## 4 eXtremal interval Newton

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

### 4.1 X-Newton iteration

Algorithm 1 describes a well-known algorithm used in several solvers (see for instance [18, 4]). A specificity here is the use of a corner-based interval Taylor form (X-Taylor) for computing the polytope.

---

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

for \( j \) from 1 to \( m \) do

polytope \( \leftarrow \) polytope \( \cup \) \{X-Taylor\(f_j, x, [x]\)\}

end for

for \( i \) from 1 to \( n \) do

/* Two calls to a Simplex algorithm: */

\( x_i \leftarrow \min x_i \) subject to polytope

\( x_i \leftarrow \max x_i \) subject to polytope

end for

return \([x]\)

---

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

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

Heuristics mentioned in [4] indicate in which order the variables can be handled, thus avoiding in practice to call 2n times the Simplex algorithm.

4.2 X-Newton

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

\begin{algorithm}[h]
\caption{X-Newton \((f, x, [x], \text{ratio}_\text{fp}, \text{CP-contractor}): [x]\)}
\begin{algorithmic}
\REPEAT
\STATE \(\{x\}_\text{save} \leftarrow [x]\)
\STATE \(\{x\} \leftarrow \text{X-NewIter}(f, x, [x])\)
\IF{\text{CP-contractor} \neq \bot \\AND \text{gain}(\{x\}, \{x\}_\text{save}) > 0} \STATE \(\{x\} \leftarrow \text{CP-contractor}(f, x, [x])\) \ENDIF
\UNTIL{\text{empty}(\{x\}) \\OR \text{gain}(\{x\}, \{x\}_\text{save}) < \text{ratio}_\text{fp}}
\RETURN{\{x\}}
\end{algorithmic}
\end{algorithm}

\texttt{X-NewIter} is iteratively run until a quasi fixed-point is reached in terms of contraction. More precisely, \texttt{ratio}_\text{fp} is a user-defined percentage of the interval size and:

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

We also permit the use of a contraction algorithm, typically issued from constraint programming, inside the main loop. For instance, if the user specifies \texttt{CP-contractor} = Mohc and if \texttt{X-NewIter} reduces the domain, then the Mohc constraint propagation algorithm [2] can further contract the box, before waiting for the next choice point. The guard \texttt{gain}(\{x\}, \{x\}_\text{save}) > 0 guarantees that \texttt{CP-contractor} will not be called twice if \texttt{X-NewIter} does not contract the box.

\textbf{Remark}

Compared to a standard interval Newton, a drawback of \texttt{X-Newton} is the loss of quadratic convergence when the current box belongs to a convergence basin. It is however possible to switch from an endpoint Taylor form to a midpoint one and thus be able to obtain quadratic convergence, as detailed in [3].

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

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

5.1 Experiments in constrained global optimization

We have selected a sample of global optimization systems among those tested by Ninin et al. [26]. They have proposed an interval Branch and Bound, called here IBBA+, that uses constraint propagation and a sophisticated variant of affine arithmetic. From their benchmark of 74 polynomial and non polynomial systems (without trigonometric operators), we have extracted the 27 ones that required more than 1 second to be solved by the simplest version of IbexOpt (column 4). In the extended paper [3], a table shows the 11 systems solved by this first version in a time comprised between 1 and 11 seconds. Table 1 includes the 13 systems solved in more than 11 seconds.\(^5\) Three systems (ex6\_2\_5, ex6\_2\_7 and ex6\_2\_13) are removed from the benchmark because they are not solved by any solver. The reported results have been obtained on a same computer (Intel X86, 3Ghz).

We have implemented the different algorithms in the Interval-Based EXplorer Ibex [9]. Reference [30] details how our interval B&B, called IbexOpt, handles constrained optimization problems by using recent and new algorithms. Contraction steps are achieved by the Mohc interval constraint propagation algorithm [2] (that also lower bounds the range of the objective function). The upper bounding phase uses original algorithms for extracting inner regions inside the feasible search space, i.e., zones in which all points satisfy the inequality and relaxed equality constraints.\(^6\) The cost of any point inside an inner region may improve the upper bound. Also, at each node of the B&B, the X-Taylor algorithm is used to produce hyperplanes for each inequality constraints and the objective function. On the obtained convex polyhedron, two types of tasks can be achieved: either the lower bounding of the cost with one call to a Simplex algorithm (results reported in columns 4 to 13), or the lower bounding and the contraction of the box, with X-NewIter (i.e., \(2n + 1\) calls to a Simplex algorithm; results reported in column 10) or X-Newton (columns 11, 13). The bisection heuristic is a variant of Kearfot’s Smear function described in [30].

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+.\(^7\) Cases of memory overflow (MO) sometimes occur. For each

\(^5\) Note that most of these systems are also difficult for the non reliable state-of-the-art global optimizer Baron [29], i.e., they are solved in a time comprised between 1 second and more than 1000 seconds (time out).

\(^6\) An equation \(h_j(x) = 0\) is relaxed by two inequality constraints: \(-\epsilon \leq h_j(x) \leq +\epsilon\).

\(^7\) The results obtained by IBBA+ on a similar computer are taken from [26].
A Contractor Based on Convex Interval Taylor

method $m$, the last line includes an average gain on the different systems. For a
given system, the gain w.r.t. the basic method (column 4) is $\frac{CPU\ time(m)}{CPU\ time(Rand)}$. The
last 10 columns of Table 1 compare different variants of X-Taylor and X-Newton.
The differences between variants are clearer on the most difficult instances. All
use Hansen’s variant to compute the interval gradient (see Section 3.2). The gain
is generally small but Hansen’s variant is more robust: for instance $ex_7.2.3$
cannot be solved with the basic interval gradient calculation.

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

<table>
<thead>
<tr>
<th>System</th>
<th>n</th>
<th>Rand</th>
<th>R+R</th>
<th>R+op</th>
<th>RRRR</th>
<th>Best</th>
<th>Brop</th>
<th>XIter</th>
<th>XNewt</th>
<th>Ibex'</th>
<th>Ibex</th>
<th>Ibex</th>
<th>IBBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>ex2.1_7</td>
<td>20</td>
<td>TO</td>
<td>42.96</td>
<td>43.17</td>
<td>40.73</td>
<td>49.48</td>
<td>TO</td>
<td>TO</td>
<td>7.74</td>
<td>10.58</td>
<td>1344</td>
<td>514</td>
<td>1574</td>
</tr>
<tr>
<td>ex2.2_9</td>
<td>10</td>
<td>MO</td>
<td>30.79</td>
<td>29.27</td>
<td>24.94</td>
<td>26.74</td>
<td>9.07</td>
<td>9.53</td>
<td>56.60</td>
<td>59.55</td>
<td>1189833</td>
<td>1095481</td>
<td>548723</td>
</tr>
<tr>
<td>ex3.1_1</td>
<td>8</td>
<td>MO</td>
<td>20.44</td>
<td>19.08</td>
<td>17.23</td>
<td>22.66</td>
<td>31.24</td>
<td>38.59</td>
<td>TO</td>
<td>633</td>
<td>14852</td>
<td>13751</td>
<td>427468</td>
</tr>
<tr>
<td>ex3.1_3</td>
<td>12</td>
<td>TO</td>
<td>1100</td>
<td>711</td>
<td>529</td>
<td>794</td>
<td>262.5</td>
<td>219</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>ex6.2_5</td>
<td>9</td>
<td>TO</td>
<td>162</td>
<td>175</td>
<td>169</td>
<td>207</td>
<td>172</td>
<td>136</td>
<td>1033</td>
<td>583</td>
<td>1675</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex6.2_8</td>
<td>3</td>
<td>97.10</td>
<td>170.06</td>
<td>119</td>
<td>110</td>
<td>134.7</td>
<td>110</td>
<td>134.7</td>
<td>110</td>
<td>134.7</td>
<td>110</td>
<td>134.7</td>
<td>110</td>
</tr>
<tr>
<td>ex6.2_9</td>
<td>8</td>
<td>25.26</td>
<td>33.10</td>
<td>36.71</td>
<td>38.82</td>
<td>44.48</td>
<td>42.34</td>
<td>43.73</td>
<td>455</td>
<td>513</td>
<td>532</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex6.2_10</td>
<td>9</td>
<td>TO</td>
<td>3221</td>
<td>2849</td>
<td>1924</td>
<td>2095</td>
<td>2218</td>
<td>2697</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td></td>
</tr>
<tr>
<td>ex6.2_11</td>
<td>10</td>
<td>10.57</td>
<td>19.31</td>
<td>7.51</td>
<td>7.53</td>
<td>10.92</td>
<td>12.26</td>
<td>11.06</td>
<td>41.20</td>
<td>11.06</td>
<td>140.51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex6.2_12</td>
<td>4</td>
<td>2120</td>
<td>232</td>
<td>160</td>
<td>118.6</td>
<td>155</td>
<td>51.31</td>
<td>22.20</td>
<td>122</td>
<td>187</td>
<td>112.58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex7.2_5</td>
<td>13</td>
<td>TO</td>
<td>44.7</td>
<td>54.9</td>
<td>60.3</td>
<td>75.86</td>
<td>29.88</td>
<td>28.91</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td></td>
</tr>
<tr>
<td>ex14_1_7</td>
<td>10</td>
<td>TO</td>
<td>433</td>
<td>445</td>
<td>406</td>
<td>489</td>
<td>786</td>
<td>938</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td></td>
</tr>
<tr>
<td>ex14.2_7</td>
<td>6</td>
<td>93.10</td>
<td>94.16</td>
<td>102.2</td>
<td>83.6</td>
<td>113.7</td>
<td>6071</td>
<td>5519</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>564</td>
<td>691</td>
<td>588791</td>
<td>611438</td>
<td>5858791</td>
<td>362538</td>
<td>315171</td>
<td>658791</td>
<td>681992</td>
<td>60007</td>
<td>1910</td>
<td>83487</td>
<td></td>
</tr>
</tbody>
</table>

The column 4 corresponds to an $X$-Taylor performed with one corner randomly
picked for every constraint. The next column (R+R) corresponds to a
tighter polytope computed with two randomly chosen corners per inequality
constraint. The gain is small w.r.t. Rand. The column 6 (R+op) highlights the
best $X$-Taylor variant where a random corner is chosen along with its opposite
corner. Working with more than 2 corners appeared to be counter-productive, as
shown by the column 7 ($RRRR$) that corresponds to 4 corners randomly picked.
We have performed a very informative experiment whose results are shown in columns 8 (Best) and 9 (B+op): an exponential algorithm selects the best corner, maximizing the expression (4), among the $2^n$ ones. The reported number of branching nodes shows that the best corner (resp. $B+op$) sometimes brings no additional contraction and often brings a very small one w.r.t. a random corner (resp. $R+op$). Therefore, the combination $R+op$ has been kept in all the remaining variants (columns 10 to 14).

The column 10 (XIter) reports the results obtained by $X$-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 11).

The last three columns report a first comparison between AA (affine arithmetic; Ninin et al.’s AF2 variant) and our convexification methods. Since we did not encode AA in our solver due to the significant development time required, we have transformed IbexOpt into two variants Ibex' and Ibex'' very close to IBBA+: Ibex' and Ibex'' use a non incremental version of HC4 [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.

First, Ibex' reaches the timeout once more than IBBA+; and IBBA+ reaches the timeout once more than Ibex''. Second, the comparison in the number of branching points (the line Sum accounts only the systems that the three strategies solve within the timeout) underlines that AA contracts generally more than Ibex', but the difference is smaller with the more contracting Ibex'' (that can also solve $ex7_{2,3}$). This suggests that the job on all the variables compensates the relative lack of contraction of $X$-Taylor. Finally, the performances of Ibex' and Ibex'' are better than IBBA+ one, but it is probably due to the different implementations.

5.2 Experiments in constraint satisfaction

We have also tested the $X$-Newton contractor in constraint satisfaction, i.e., for solving well constrained systems having a finite number of solutions. These systems are generally square systems ($n$ equations and $n$ variables). The constraints correspond to non linear differentiable functions (some systems are polynomial, others are not). We have selected from the COPRIN benchmark all the systems that can be solved by one of the tested algorithms in a time between 10 s and 1000 s: we discarded easy problems solved in less than 10 seconds, and too difficult problems that no method can solve in less than 1000 seconds. The timeout

---

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

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

10. [Link to COPRIN benchmark](http://www-sop.inria.fr/coprin/logiciels/ALIAS/Benches/benches.html)
was fixed to one hour. The required precision on the solution is $10^{-8}$. Some of these problems are scalable. In this case, we selected the problem with the greatest size (number of variables) that can be solved by one of the tested algorithms in less than 1000 seconds.

We compared our method with the state of art algorithm for solving such problems in their original form (we did not use rewriting of constraints and did not exploit common subexpressions). We used as reference contractor our best contractor $\text{ACID(Mohc)}$, an adaptive version of $\text{CID [31]}$ with $\text{Mohc [2]}$ as basic contractor, that exploits the monotonicity of constraints. We used the same bisection heuristic as in optimization experiments. Between two choice points in the search tree, we called one of the following contractors (see Table 2).

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

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

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

We can observe that, as for the optimization problems, the corner selection $\text{R+op}$ yields the lowest sum of solving times and often good results. The last line of Table 2 highlights that all the 24 systems can be solved in 1000 s by $\text{X-Newton R+op}$, while only 18 systems are solved in 1000 s by the reference algorithm with no convexification method. Each entry in Table 2 contains the CPU time in second (first line of a multi-line) and the number of branching nodes (second line). We have reported in the last column (Gain) the gains obtained by the best corner selection strategy $\text{R+op}$ as the ratio w.r.t. the reference method (column 3 Ref), i.e., $\frac{\text{CPU time(Ref)}}{\text{CPU time(R+op)}}$. Note that we used the inverse gain definition compared to the one used in optimization (see 5.1) in order to manage the problems reaching the timeout. We can also observe that our new algorithm $\text{X-Newton R+op}$ is efficient and robust: we can obtain significant gains (small values in bold) and lose never more than 39% in CPU time.

We have finally tried, for the scalable systems, to solve problems of bigger size. We could solve $\text{Katsura-30}$ in 4145 s, and $\text{Yamamura1-16}$ in 2423 s (instead of 33521 s with the reference algorithm). We can remark that, for these problems, the gain grows with the size.
Table 2. Experimental results on difficult constraint satisfaction problems: the best results and the gains (< 1) appear in bold

<table>
<thead>
<tr>
<th>System</th>
<th>Ref</th>
<th>Xiter</th>
<th>Rand</th>
<th>R+R</th>
<th>R+op</th>
<th>RR</th>
<th>R+op</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Belido</td>
<td>9</td>
<td>10.04</td>
<td>3.88</td>
<td>4.53</td>
<td>3.71</td>
<td>3.35</td>
<td>3.35</td>
<td>3.28</td>
</tr>
<tr>
<td>Bratu-60</td>
<td>60</td>
<td>494</td>
<td>146</td>
<td>306</td>
<td>218</td>
<td>196</td>
<td>172</td>
<td>357</td>
</tr>
<tr>
<td>Brent-10</td>
<td>10</td>
<td>25.31</td>
<td>28</td>
<td>31.84</td>
<td>33.16</td>
<td>34.88</td>
<td>37.72</td>
<td>37.11</td>
</tr>
<tr>
<td>Brown-10</td>
<td>10</td>
<td>TO</td>
<td>0.13</td>
<td>0.17</td>
<td>0.17</td>
<td>0.17</td>
<td>0.17</td>
<td>0.18</td>
</tr>
<tr>
<td>Butcher8-a</td>
<td>8</td>
<td>233</td>
<td>246</td>
<td>248</td>
<td>242</td>
<td>266</td>
<td>266</td>
<td>1.06</td>
</tr>
<tr>
<td>Butcher8-b</td>
<td>8</td>
<td>97.9</td>
<td>123</td>
<td>113.6</td>
<td>122</td>
<td>142.4</td>
<td>122</td>
<td>1.26</td>
</tr>
<tr>
<td>Design</td>
<td>9</td>
<td>21.7</td>
<td>23.61</td>
<td>22</td>
<td>22.38</td>
<td>25.33</td>
<td>25.45</td>
<td>1.03</td>
</tr>
<tr>
<td>Direct Kinematics</td>
<td>11</td>
<td>85.28</td>
<td>84.96</td>
<td>83.52</td>
<td>84.28</td>
<td>86.15</td>
<td>85.62</td>
<td>0.99</td>
</tr>
<tr>
<td>Dietmaier</td>
<td>12</td>
<td>3055</td>
<td>1036</td>
<td>880</td>
<td>979</td>
<td>1233</td>
<td>1205</td>
<td>0.31</td>
</tr>
<tr>
<td>Discrete integral-16</td>
<td>52</td>
<td>TO</td>
<td>480</td>
<td>469</td>
<td>471</td>
<td>472</td>
<td>478</td>
<td>476</td>
</tr>
<tr>
<td>2nd form.</td>
<td></td>
<td>57901</td>
<td>57991</td>
<td>57991</td>
<td>57591</td>
<td>57591</td>
<td>57591</td>
<td>57591</td>
</tr>
<tr>
<td>Eco9</td>
<td>8</td>
<td>12.85</td>
<td>14.19</td>
<td>14.35</td>
<td>14.88</td>
<td>17.48</td>
<td>17.3</td>
<td>1.17</td>
</tr>
<tr>
<td>Ex14-2-3</td>
<td>6</td>
<td>45.01</td>
<td>3.83</td>
<td>4.39</td>
<td>3.88</td>
<td>3.58</td>
<td>3.87</td>
<td>3.68</td>
</tr>
<tr>
<td>Fredtest</td>
<td>6</td>
<td>71.61</td>
<td>47.73</td>
<td>54.46</td>
<td>47.14</td>
<td>42.67</td>
<td>40.76</td>
<td>0.59</td>
</tr>
<tr>
<td>Fourbar</td>
<td>4</td>
<td>258</td>
<td>317</td>
<td>295</td>
<td>319</td>
<td>320</td>
<td>366</td>
<td>1.24</td>
</tr>
<tr>
<td>Geneig</td>
<td>17</td>
<td>51.32</td>
<td>46.1</td>
<td>46.25</td>
<td>41.33</td>
<td>38.4</td>
<td>38.43</td>
<td>0.7</td>
</tr>
<tr>
<td>I5</td>
<td>19</td>
<td>17.21</td>
<td>20.59</td>
<td>19.7</td>
<td>20.53</td>
<td>20.86</td>
<td>23.23</td>
<td>23.43</td>
</tr>
<tr>
<td>Katsura-25</td>
<td>28</td>
<td>TO</td>
<td>711</td>
<td>1900</td>
<td>1258</td>
<td>700</td>
<td>1238</td>
<td>1007</td>
</tr>
<tr>
<td>Pramanik</td>
<td>9</td>
<td>14.69</td>
<td>20.98</td>
<td>19.16</td>
<td>20.31</td>
<td>20.38</td>
<td>24.58</td>
<td>25.15</td>
</tr>
<tr>
<td>Trigexp2-17</td>
<td>17</td>
<td>492</td>
<td>568</td>
<td>533</td>
<td>570</td>
<td>574</td>
<td>630</td>
<td>0.95</td>
</tr>
<tr>
<td>Trigo1-14</td>
<td>17</td>
<td>2097</td>
<td>1062</td>
<td>1314</td>
<td>1003</td>
<td>910</td>
<td>865</td>
<td>0.43</td>
</tr>
<tr>
<td>Trigonometric</td>
<td>33</td>
<td>33.75</td>
<td>30.99</td>
<td>30.13</td>
<td>30.11</td>
<td>30.65</td>
<td>31.1</td>
<td>31.75</td>
</tr>
<tr>
<td>Virasoro</td>
<td>8</td>
<td>670</td>
<td>715</td>
<td>729</td>
<td>704</td>
<td>709</td>
<td>713</td>
<td>0.93</td>
</tr>
<tr>
<td>Yamamura-14</td>
<td>14</td>
<td>1542</td>
<td>409</td>
<td>628</td>
<td>555</td>
<td>472</td>
<td>520</td>
<td>0.26</td>
</tr>
<tr>
<td>Sum</td>
<td></td>
<td>42353</td>
<td>8341</td>
<td>800</td>
<td>7087</td>
<td>6185</td>
<td>7688</td>
<td>7131</td>
</tr>
<tr>
<td>Gain</td>
<td></td>
<td>0.75</td>
<td>0.77</td>
<td>0.78</td>
<td>0.76</td>
<td>0.9</td>
<td>0.85</td>
<td></td>
</tr>
<tr>
<td>Solved in 1000s</td>
<td></td>
<td>18</td>
<td>22</td>
<td>22</td>
<td>24</td>
<td>22</td>
<td>22</td>
<td></td>
</tr>
</tbody>
</table>
6 Conclusion

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

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

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

Acknowledgment

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

References