Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence

Inner Regions and Interval Linearizations for Global Optimization

Gilles Trombettoni,[1] Ignacio Araya,[2] Bertrand Neveu,[3] Gilles Chabert[4]
Gilles.Trombettoni@inria.fr, iaraya@inf.utfsm.cl, neveub@certis.enpc.fr, Gilles.Chabert@emn.fr
[1]INRIA, I3S, Université Nice–Sophia (France), [2]UTFSM (Chile),
[3]Imagine LIGM Université Paris–Est (France), [4] LINA, EMN (France)

Abstract

Researchers from interval analysis and constraint (logic) programming communities have studied intervals for their ability to manage infinite solution sets of numerical constraint systems. In particular, *inner* regions represent subsets of the search space in which *all* points are solutions. Our main contribution is the use of recent and new inner region extraction algorithms in the *upper bounding* phase of constrained global optimization.

Convexification is a major key for efficiently *lower bounding* the objective function. We have adapted the convex interval taylorization proposed by Lin & Stadtherr for producing a reliable outer and inner polyhedral approximation of the solution set and a linearization of the objective function. Other original ingredients are part of our optimizer, including an efficient interval constraint propagation algorithm exploiting monotonicity of functions.

We end up with a new framework for reliable continuous constrained global optimization. Our interval `B&B` is implemented in the interval-based explorer `Ibex` and extends this free `C++` library. Our strategy significantly outperforms the best reliable global optimizers.

1 Introduction

Interval B&B algorithms are used to solve constrained global optimization problems[1] in a *reliable* way, i.e., they provide an optimal solution and its cost with a bounded error or a proof of infeasibility. The story of interval B&B started with interval analysis (Moore 1966). Numerous pioneering ideas are detailed in books like (Moore 1966), (Hansen 2003), (Kearfott 1996), to name a few. In the middle of the nineties, Kearfott designed the `GlobSol` solver, and researchers from the constraint programming community designed the solvers `Numerica` (Van Hentenryck, Michel, and Deville 1997) and `Icos` (Lebbah, Michel, and Rueher 2007) that introduce interval constraint propagation algorithms and safe linear relaxations respectively. More recently, the mathematical programming community has also contributed with a solver, called here `IBBA+`, that integrates constraint propagation and affine arithmetic (Ninin, Messine, and Hansen 2011).

For ensuring reliability at a good performance, the interval paradigm is faced with two main difficulties.

[1]We consider minimization in this paper w.l.o.g.

Upper bounding in the feasible space. *Local search* is the most used approach for finding a *feasible* point[2] (i.e., a solution satisfying the constraints) that improves the best value of the objective function f. However, to ensure reliability in presence of equality constraints, the exploration of a search space containing feasible and unfeasible points requires an additional iterative correction of the unfeasible points found during the local search and their certification with expensive interval analysis techniques (Lebbah, Michel, and Rueher 2007). This additional time-consuming iterative correction makes it simply impossible the competition, in terms of performance, with state-of-the-art *non* reliable global optimizers like `Baron` (Tawarmalani and Sahinidis 2005).

In this paper, we propose a radically different approach where the search effort is spent only inside inner regions of the search space, i.e., regions in which all points are feasible. Several researchers from interval constraint programming have intensively studied inner boxes for paving the solution set (Collavizza, Delobel, and Rueher 1999; Benhamou and Goualard 2000). Inner boxes have also been used for minimizing the number of violated numerical inequality constraints (Normand et al. 2010). However, the paradigm remained not exploited in general global optimization under inequality and equality constraints.

Lower bounding with reliable convexification. All the existing solvers compute, at each node of the B&B, a convex, generally polyhedral, outer approximation of the solution set. The best solution of the obtained relaxation (of the initial problem) yields a lower bound of the cost that is needed to terminate the search. Most of the linear relaxations are sophisticated, rendering tedious the task of making them conservative. Indeed, the outer approximation must enclose the solution set in spite of floating-point calculation errors. Specific Reformulation-Linearization Techniques (Sherali and Adams 1999) are presented in (Kearfott 1996) and (Lebbah, Michel, and Rueher 2007) that add new variables equal to powers or products in the system, and define linear constraints between them. Ninin et al. use affine

[2]A second approach resorts to a satisfaction problem by looking for points where the gradient of f is null. The minimum of f is obtained at a solution of this problem or at a bound of the domain. Taking into account the constrains in this formulation requires Lagrangian machinery and Kuhn-Tucker theorem. This can lead to a huge aggregate function unadapted for interval computations (Hansen 2003).

arithmetic for computing a safe linearization of each operator (Ninin, Messine, and Hansen 2011). Instead, we propose in this paper a reliable linearization based on a first-order interval Taylor. The simplicity of this interval relaxation also leads to a dual version that can extract an inner polyhedral region inside the solution set and improve the upper bound.

1.1 Intervals and constrained global optimization

An interval $[x_i] = [\underline{x_i}, \overline{x_i}]$ defines the set of reals x_i s.t. $\underline{x_i} \le x_i \le \overline{x_i}$. $\mathbb{IR}$ 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 $\max_i w([x_i])$. $\mathrm{Mid}([x])$ denotes the middle of $[x]$. A numerical or continuous constrained global optimization problem is defined as follows.

Definition 1 (Constrained global optimization) *Consider a vector of variables $x = \{x_1, ..., x_i, ...x_n\}$ varying in a box $[x]$, a real-valued function $f : \mathbb{R}^n \to \mathbb{R}$, vector-valued functions $g : \mathbb{R}^n \to \mathbb{R}^m$ and $h : \mathbb{R}^n \to \mathbb{R}^p$.*

Given the system $S = (f, g, h, x, [x])$, the constrained global optimization problem consists in finding:

$$\min_{x \in [x]} f(x) \ \ subject\ to\ \ g(x) \le 0 \wedge h(x) = 0.$$

f denotes the **objective function***; g and h are inequality and equality* **constraints** *respectively. x is said to be* **feasible** *if it satisfies the constraints.*

Our interval optimizer extracts *inner boxes* and *inner regions* inside classical (outer) boxes.

Definition 2 *Consider a system $(f, g, \emptyset, x, [x]^{out})$ comprising only inequality constraints. 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 $g(x) \le 0$.*

An **inner box** *$[x]^{in}$ is an inner region which is a box.*

Due to the *inner linearizations* achieved by our strategy, the considered inner regions are polytopes.

Interval arithmetic (Moore 1966) extends to $\mathbb{IR}$ elementary functions over $\mathbb{R}$. For instance, the interval sum (i.e., $[x_1] + [x_2] = [\underline{x_1} + \underline{x_2}, \overline{x_1} + \overline{x_2}]$) encloses the image of the sum function over its arguments, and this enclosing property basically defines what we call an *interval extension*.

Definition 3 (Extension of a function to $\mathbb{IR}$) *Consider a function $f : \mathbb{R}^n \to \mathbb{R}$.*
$[f] : \mathbb{IR}^n \to \mathbb{IR}$ is said to be an **extension** *of f to intervals if:*

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

In our context, the expression of a function f is always a composition of elementary functions. The **natural extension** $[f]_N$ is then simply a composition of the corresponding interval operators.

The outer and inner interval linearizations proposed in this paper are related to the first-order **interval Taylor extension** (Moore 1966), 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., $\mathrm{Mid}([x])$.

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 $\dot{x} = (1, 2)$ yields: $[f]_t([x_1], [x_2]) = 9 + [-7, 23] * [-2, 2] + [-3, 13] * [-3, 3] = [-76, 94]$.

1.2 Handling equations as inequality constraints

To handle equalities, in a first option followed by the interval community, one finds *approximately* a point that satisfies *exactly* the constraints. The solvers return a tiny box of width ϵ_{sol} in which the existence of a real-valued point is (often) guaranteed by interval Newton methods. In a second option, one finds *exactly* a point that satisfies *approximately* the constraints. Equations are handled with a (tiny) admissible precision error ϵ_{eq}, i.e., a feasible floating-point x verifies $h(x) \in [-\epsilon_{eq}, +\epsilon_{eq}]$. All the constraints can thus be viewed as inequalities: $\{g(x) \le 0,\ h(x) - \epsilon_{eq} \le 0,\ -h(x) - \epsilon_{eq} \le 0\}$. Ninin et al. were guided to this choice by their affine arithmetic, but we believe that this is a relevant approach for any global optimization solver. First, both policies are of equal status regarding reliability. Second, a precision error ϵ_{eq} on the *images* of functions h better fits the original feasibility problem than a precision ϵ_{sol} on the unknowns. Third, most of the equations defined by interval practitioners are already "thick" and do not require a relaxation with ϵ_{eq}. Indeed, constraints often have coefficients known with a bounded uncertainty (e.g., an imprecision on a measured distance) and sometimes contain irrational constants, like π, that can be specified by tiny intervals. Finally, our experiments give an evidence that handling thick equalities can work efficiently in practice. The reason behind this good surprise is that this policy allows optimizers to extract inner regions in continua of solutions. *Efficient inner region extraction and filtering algorithms can then focus the search in the tiny solution set defined by thick equations.*

2 Description of our interval B&B

Our `IbexOpt` strategy follows the well-known branch and bound schema described in (Horst and Tuy 1966) to solve a constrained global optimization problem. Starting from an initial box, the algorithm splits it recursively until a solution minimizing the objective function is found. During the search, a current (generally non feasible) *lower bound* of the objective function is computed for each box in the list managed by the algorithm. We call lb (for lower bound) the minimum value of these lower bounds. Also, ub (for upper bound) is the cost of the current best feasible point found during the search. A termination occurs when $ub - lb$ reaches a precision ϵ_{obj},[3] and the floating-point vector x_{ub} of cost ub is returned. Note that boxes the width of which does not exceed a precision ϵ_{sol} are not put again in the list and their lower bound accounts in the computation of lb.

At each iteration, the algorithm selects in the list the box $[x]$ with the lowest lower bound, thus following a best-first

[3]Following standard implementations, ϵ_{obj} is a percentage of ub if $|ub| \ge 1$; ϵ_{obj} is an absolute distance if $|ub| \le 1$.

search. It chooses a branching variable $x_i \in x$ heuristically, bisects $[x_i]$ and applies the main `Contract&Bound` procedure on the two sub-boxes. Note that the search tree, i.e., the "list" of boxes to be handled, is managed by a *heap* data structure to access to the minimum lower bound in constant time. More details about the overall schema can be found in (Ninin, Messine, and Hansen 2011).

Like for instance in Numerica (Van Hentenryck, Michel, and Deville 1997), the first task of our optimizer is to automatically introduce a new variable y in the input system $(f, g, h, x, [x])$. This variable is linked to the others by an additional constraint $y = f(x)$. The domain $[y]$ is therefore an interval that encompasses the image of the objective function on $[x]$. It can be used as a simple way to propagate and retro-propagate contractions between $[x]$ and global bounds on the minimum. Hence, the extended box $[x] \times [y]$ defines the backtrackable state of the optimizer, and three variables shared by all nodes in the search tree are updated globally during the search: the current best candidate x_{ub}, its cost ub ($f(x_{ub}) = ub$) and the minimum of the lower bounds lb.

2.1 A variant of the smear branching strategy

At each search node, a variant of the well-known *smear function* (Kearfott and Novoa III 1990) selects the next variable to be split. Given a system $(f, g, h, x, [x])$, the standard smear-based strategy selects the variable x_i in x with the greatest value $\mathtt{smearMax}(x_i) = Max_{f_j}\ \mathtt{smear}(x_i, f_j)$ or $\mathtt{smearSum}(x_i) = \sum_{f_j} \mathtt{smear}(x_i, f_j)$, according to two different variants, where f_j ranges over all the functions (f and the components of g, h).

$\mathtt{smear}(x_i, f_j)$ reflects an impact of the variable x_i on function f_j. It depends on the partial derivative of f_j w.r.t. x_i and on the width of $[x_i]$. More precisely:

$$\mathtt{smear}(x_i, f_j) = \left| \left[\frac{\partial f_j}{\partial x_i}\right]_N ([x]) \right| * w([x_i]).$$

We propose a variant $\mathtt{smearRel}(x_i, f_j)$ of $\mathtt{smear}(x_i, f_j)$ that simply measures a *relative* impact falling in $[0, 1]$:

$$\mathtt{smearRel}(x_i, f_j) = \frac{\mathtt{smear}(x_i, f_j)}{\sum_{x_k \in x} \mathtt{smear}(x_k, f_j)}.$$

Finally, our branching strategy `SmearSumRel` selects the variable x_i in x with the greatest impact:

$$\mathtt{smearSumRel}(x_i) = \sum_{f_j} \mathtt{smearRel}(x_i, f_j).$$

Although not always the best, this strategy appears to be more robust than its competitors on the tested benchmark.

2.2 The Contract&Bound procedure

The main algorithm `Contract&Bound` (see Algorithm 1) is called at each node of our B&B. The first line introduces in the current system the best cost ub ever found (like in any `B&B`). The procedure `OuterContractLB` filters the domains and improves the lower bound. `InnerExtractUB` extracts inner regions inside $[x]$, chooses a point x in those regions (if any), and potentially replaces x_{ub} by x and its cost ub by $f(x)$. `OuterContractLB` calls two main procedures. First, the `Mohc` algorithm (Araya, Trombettoni, and Neveu 2010) contracts the box $[x] \times [y]$. It can thus also lower bound the objective function. This recent interval constraint propagation algorithm exploits monotonicity of functions. It uses an efficient *Revise* procedure that can optimally contract the box w.r.t. a single constraint (e.g., $g_j(x) \leq 0$), when $g_j(x)$ is monotonic w.r.t. every variable in the box, even if $g_j(x)$ contains multiple occurrences of variables. Note that the smaller a box is, i.e., the deeper in the search tree, the likelier functions are monotonic w.r.t. variables. Second, `OuterContractLB` calls an interval linearization, called here `OuterLinearization`, for lower bounding the objective function, i.e., for increasing $\underline{y}$.

Algorithm 1 `Contract&Bound` (**in** S, $[x]$; **in-out** ub)

$\overline{y} \leftarrow ub - \epsilon_{obj}$
`OuterContractLB` (S, $[x] \times [y]$) /* contraction */
if $[x] \times [y] = \emptyset$ **then** exit **endif** /* no solution */
`InnerExtractUB` (S, $[x]$, ub, x_{ub}) /* inner regions */

2.3 Outer interval linearization

A safe polyhedral convexification is built upon a straightforward adaptation of a specific first order interval Taylor form of a nonlinear function (Lin and Stadtherr 2004). Consider a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ defined on a domain $[x]$. For any variable $x_i \in x$, let $[a_i]$ be $\left[\frac{\partial f}{\partial x_i}\right]_N([x])$. The idea is to (lower) tighten $f(x)$ with linear functions like:

$$\forall x \in [x], f(\underline{x}) + \underline{a_1} * x_1^l + ... + \underline{a_n} * x_n^l \ \leq f(x) \qquad (1)$$

$$\forall x \in [x], f(\overline{x}) + \overline{a_1} * x_1^r + ... + \overline{a_n} * x_n^r \ \leq f(x) \qquad (2)$$

where: $x_i^l = x_i - \underline{x_i}$ and $x_i^r = x_i - \overline{x_i}$. The first-order interval Taylor form can select any expansion point $\dot{x}$ inside the box to achieve the linearization. Instead of the usual midpoint, a *corner* of the box is chosen here: $\underline{x}$ in form (1) or $\overline{x}$ in form (2). If we consider an inequality $g_j(x) \leq 0$, Expression (1) or (2) defines a hyper-plane $g_j^l(x)$ bounding the solution set from below: $g_j^l(x) \leq g_j(x) \leq 0$. Applying, for instance, the form (1) to the objective function $f(x)$ and to the inequalities $g_j(x) \leq 0$ ($j = 1...m$), we can derive a linear problem LP^{lb}:

$$\begin{array}{lll} LP^{lb} = \min & & f(\underline{x}) + \underline{a_1} * x_1^l + ... + \underline{a_n} * x_n^l \\ \mathit{subject\ to:} & \forall j & g_j(\underline{x}) + \underline{a_1^j} * x_1^l + ... + \underline{a_n^j} * x_n^l \leq 0 \\ & & \forall i \ \ 0 \leq x_i^l, \ \ x_i^l \leq w([x_i]) \\ \mathit{where:} & & x_i^l = x_i - \underline{x_i} \end{array}$$

`OuterLinearization` calls a Simplex algorithm to solve LP^{lb} and returns infeasibility or the optimal value y^l. Infeasibility means that $[x]$ contains no solution and can be discarded. Otherwise, if $y^l \geq \underline{y}$, then the best lower bound of the box is updated: $\underline{y} \leftarrow y^l$.

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

Safety is ensured by the *interval-based* taylorisation (Neumaier 1990). The correction of relation (1) lies on the fact that any variable x_i^l is positive since its domain is $[0, d_i]$, with $d_i = w([x_i]) = \overline{x_i} - \underline{x_i}$. Therefore, minimizing each term $[a_i] * x_i^l$ for any point $x_i^l \in [0, d_i]$ is obtained with $\underline{a_i}$. Symmetrically, relation (2) is correct since $x_i^r \in [-d_i, 0] \leq 0$, and the minimal value of a term is obtained with $\overline{a_i}$ (Lin and Stadtherr 2004).

Note that, even though our linearizations are safe, the floating-point calculation errors made by the Simplex algorithm could make its output y^l unsafe. A cheap postprocessing proposed in (Neumaier and Shcherbina 2004), using interval arithmetic, has been added to certify the solution.

An improvement has been brought to this outer convexification for computing a tighter polytope. We lower tighten a function $f(x)$ with expressions (1) and (2) simultaneously, using an expanded form:

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

2.4 Upper bounding with inner regions

The call to `OuterContractLB` is followed by a call to `InnerExtractUB` (see Algorithm 2). The procedure first calls an adaptation of a recent algorithm (Chabert and Beldiceanu 2010), named here `InHC4`, for extracting an inner box from the outer box $[x]^{out}$.[4] For a single constraint, `InHC4` returns a box that is inner w.r.t. that constraint. The different boxes returned for all the constraints are intersected to obtain an inner box. Like `HC4` (Benhamou et al. 1999), the algorithm reasons on the syntactical tree of the constraints and uses *projections* for unary operators, with inward rounding however. Furthermore, in case of unions (e.g., in x^2 and $sinus$ operators), one single interval is kept since holes contain inconsistent points, making the whole algorithm heuristic. For binary operators, the projections in the *backward phase* are different and also lead to heuristics choices. Details can be found in (Chabert and Beldiceanu 2010), Section 3.

If an inner box $[x]^{in}$ is found by `InHC4`, then `MonotonicityAnalysis` analyzes the monotonicity of the objective function f w.r.t. every variable x_i. If the partial derivative $[a_i] = \left[\frac{\partial f}{\partial x_i}\right]_N ([x]^{in}) \geq 0$, then f is increasing and $[x_i]$ is replaced by the degenerate interval $[\underline{x_i}, \underline{x_i}]$ in $[x]^{in}$ to minimize $f(x)$ over $[x]^{in}$. If $[a_i] \leq 0$, f is decreasing and $[x_i]$ is replaced by $[\overline{x_i}, \overline{x_i}]$ in $[x]^{in}$.

Next, we pick randomly a point x inside the box[5] and replaces x_{ub} by x if x satisfies the constraints and improves the best cost ub. Two different cases may occur. If an inner box has been extracted by `InHC4`, then a point is selected inside $[x]^{in}$. The feasibility does not need be checked since $[x]^{in}$ contains only feasible points. If no inner box is available, a random point is however picked in the outer box $[x]^{out}$, and the constraints must be checked. Replacing this simple probing by a gradient descent did not improve the strategy.

[4] The published algorithm handles in fact a dual problem of finding unfeasible boxes, i.e., boxes in which all points satisfy the negation of the constraints...

[5] Selecting several points instead of just one turned out to be experimentally counter-productive.

Algorithm 2 `InnerExtractUB` (**in:** S, $[x]^{out}$; **in-out:** ub, x_{ub})

> $[x]^{in} \leftarrow$ `InHC4` $(S, [x]^{out})$ /* Inner box extraction */
> **if** $[x]^{in} \neq \emptyset$ **then**
> $[x]^{in} \leftarrow$ `MonotonicityAnalysis` $(f, [x]^{in})$
> $x \leftarrow$ `RandomProbing`$([x]^{in})$
> **else**
> $x \leftarrow$ `RandomProbing`$([x]^{out})$
> **end if**
> $cost \leftarrow \overline{f([x,x])}$ /* Cost evaluation */
> **if** $cost < ub$ **and** $([x]^{in} \neq \emptyset$ **or** $g(x) \leq 0)$ **then**
> $ub \leftarrow cost$; $x_{ub} \leftarrow x$
> **end if**
> $LP^{ub} \leftarrow$ `InnerLinearization` $(S, [x]^{out})$
> $x^l \leftarrow$ `Simplex`(LP^{ub})
> **if** $x^l \neq \perp$ **then**
> $cost \leftarrow \overline{f([x^l, x^l])}$
>   **if** $cost < ub$ **then** $ub \leftarrow cost$; $x_{ub} \leftarrow x^l$ **end if**
> **end if**

This is easy to understand in presence of equations since the inner boxes are tiny. This was more surprising for optimization problems under inequality constraints only. The last part of `InnerExtractUB` performs an *inner* linearization of the system for extracting a polyhedral inner region.

2.5 Inner interval linearization

Symmetrically to the relation (1) used in outer linearization:

$$\forall x \in [x],\ f(x) \leq f^l(x) = f(\underline{x}) + \sum_i \overline{a_i} * (x_i - \underline{x_i}). \quad (3)$$

If an inequality $f(x) \leq 0$ is handled, relation (3) enables us to build a hyper-plane $f^l(x)$ s.t. $f(x) \leq f^l(x) \leq 0$. That is to say, the linear function $f^l(x)$ can be used to define an inner region of $[x]$. Applying this idea to the objective function $f(x)$ and to the inequalities $g_j(x) \leq 0$, we can derive the linear program LP^{ub}:

$$\begin{array}{lll} LP^{ub} = \min & & f(\underline{x}) + \sum_i \overline{a_i} * (x_i - \underline{x_i}) \\ subject\ to: & \forall j & g_j(\underline{x}) + \sum_i \overline{a_i^j} * (x_i - \underline{x_i}) \leq 0 \\ & \forall i & \underline{x_i} \leq x_i \wedge x_i \leq \overline{x_i} \end{array}$$

A Simplex algorithm solves LP^{ub} and returns infeasibility or the optimal solution x^l (see Algorithm 2). Infeasibility proves nothing because the linearized system is more constrained than the original system, so that one could still find solutions in the original one. If the Simplex algorithm returns an optimal solution of the inner approximation, then x^l is also a solution of the original system, maybe not the optimal one. We evaluate the (original) objective function at the point x^l and potentially update x_{ub} and ub.

3 Experiments

We have implemented our strategy in the Interval-Based EXplorer `Ibex` (Chabert and Jaulin 2009). This free `C++` library has facilitated the implementation of our global optimizer by providing us a direct access to the `Mohc` algorithm, different branching strategies, automatic differentiation, etc. All parameters have been fixed to a given set of adequate values common to all the tested instances. The precision has been fixed to $\epsilon_{obj} = $ `1.e-8` Also, $\epsilon_{sol} = \frac{\epsilon_{obj}}{10}$. Finally, the admissible precision error ϵ_{eq} in thick equations $h(x) \in [-\epsilon_{eq}, +\epsilon_{eq}]$ has been fixed to $\epsilon_{eq} =$ `1.e-8` for all the experiments.

Tests have been performed on the benchmark of 74 systems used by our best competitor `IBBA+` (Ninin, Messine, and Hansen 2011). Table 1 presents a qualitative study analyzing which ingredients improve the performance.

Table 1: **Qualitative study.** The columns include the number of systems whose loss/gain in performance $\frac{cputime(strategy \backslash \{ingredient\})}{cputime(strategy)}$, caused by the removal of a single ingredient from our strategy, belongs to a given range (first line). The tested removals are: `Mohc` replaced by `HC4` (Mohc/HC4); `OuterLinearization` (OuterLinear.); `InnerExtractUB` replaced by a simple probing in the outer box (Inner/Probing); `InnerLinearization` (InnerLinear.); `InHC4`; `SmearSumRel` replaced resp. by `SmearMax` (SSR/SM); `Round Robin` (SSR/RR); `LF` (SSR/LF); the largest-first heuristic selects the variable with the largest interval.

Gain	0.02	[0.1, 0.5]	[0.5, 2]	[2, 10]	[10, 100]	>100
Mohc/HC4	0	1	62	5	0	2
OuterLinear.	0	1	35	9	5	20
Inner/Probing	0	0	33	24	9	4
InnerLinear.	0	0	62	7	0	1
InHC4	0	0	66	4	0	0
SSR/SM	0	2	59	4	1	4
SSR/RR	0	1	42	13	11	3
SSR/LF	1	0	40	9	16	4

Interesting observations can be drawn. First, all five original devices proposed in our strategy appear to be useful in practice. Second, our simple outer linearization seems very helpful in the lower bounding phase. A future study should compare this convex interval taylorization with affine arithmetic and with the `Quad` RLT operator used by `Icos`. Third, extracting inner regions is also very useful in the upper bounding phase. Table 1 underlines that it is often sufficient to endow the strategy with `InHC4` *or* `InnerLinearization`, although putting both devices is sometimes beneficial and never counter-productive on the selected benchmark.

We have also compared our strategy to the available and maintained reliable global optimizers: `Globsol`, `Icos` and `IBBA+`,[6] and with the *non reliable* deterministic solver `Baron`. Be aware that `Baron` does not guarantee its returned best solution that is sometimes not feasible and may have a too low cost.

Fig.1 shows the performance profiles of `IbexOpt`, `Baron` and our best reliable competitor `IBBA+`. We give details on the 28 systems that are solved by `IbexOpt` in more than one second. Table 2 corresponds to the 12 systems solved by `IbexOpt` in less than 10 seconds. Table 3 includes the 13 systems solved in more than 10 seconds. Three systems (`ex6_2_5`, `ex6_2_7` and `ex7_2_3`) are removed from this table because they are not solved by any solver, including `Baron`. The results for `Globsol`, `IBBA+`, `Icos` and `IbexOpt` have

[6] `IBBA+` corresponds to the most efficient strategy described in (Ninin, Messine, and Hansen 2011).

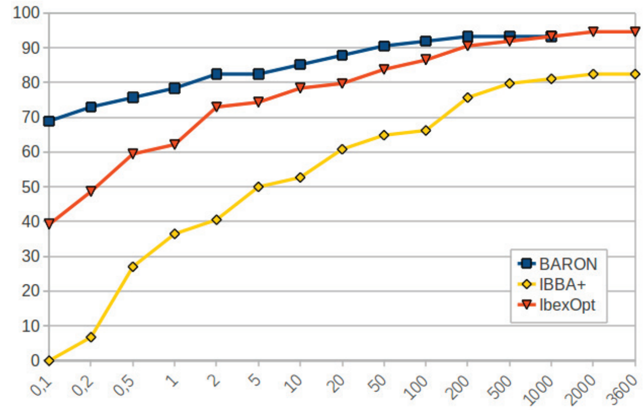

Figure 1: **Performance profiles.** For a given strategy, a point (t, p) on the corresponding curve indicates that p percent of the systems have been solved in less than t seconds.

been obtained on very similar computers (`Intel X86, 3Ghz`). `Baron` 9.0.7 has been run on the `Neos` server (see `www.neos-server.org/neos/`) also on a `X86`, thus making the comparison rather fair.

The figure and tables show that `IbexOpt` often outperforms its reliable competitors by one or several orders of magnitude. The performance profile illustrates that `IbexOpt` show performances which are intermediary between `IBBA+` and `Baron` and that it can solve the same systems as `Baron` in 1000 seconds. The results obtained by `Baron` are impressive, although it should be noticed that several instances are solved during a pre-processing (the number of branching nodes is 1 in the tables).

Note that `IbexOpt` is better than `Baron` on 5 of the 28 difficult systems (see Tables 2 and 3), especially on the series `ex6_2_*` having huge non polynomial objective functions. To our knowledge, no reliable solver could compete with `Baron` on non trivial instances that `Baron` solve in seconds or more.

We have also tested a variant of our strategy where `Mohc` is replaced by `3BCID(Mohc)` (Trombettoni and Chabert 2007). Although generally counterproductive in terms of performances, the variant is more robust and can solve the `ex7_2_3` instance in 38 seconds with 6235 branching nodes, while `Baron` raises a memory overflow.

4 Conclusion

We have proposed a new framework for reliable global optimization that exploits inner regions in the upper bounding phase, thus avoiding the recourse to local search. Provided that equations can be defined with a tiny admissible precision error, the approach is also relevant for handling equality constraints. Our strategy is endowed with five significant devices. Three of them, i.e., `Mohc`, `InHC4` and `OuterLinearization` have never been used in global optimization. Two of them, i.e., `SmearSumRel` and `InnerLinearization`, are new. All five have proven their efficiency in a sample of non trivial constrained global optimization systems. They confirm the relevance of inner region exploitation and polyhedral approximations based on convex interval taylorization.

Due to the number of novel ingredients, there is still significant space for improvement in the hope of reaching `Baron` performance in a long-term.

Table 2: **Comparison on mean-difficult systems.** The first two lines indicate the name of the competitor with the used precision ϵ_{obj} on the cost. Each entry contains generally the CPU time in second (first line of a multi-line) and the number of branching nodes (second line). A timeout of one hour (>3600) is shared by `IBBA+`, `GlobSol` and `IbexOpt`. It is 10 min (>600) for `Icos`, 1000 seconds for `Baron` (imposed by the `Neos` server). An empty entry indicates that the information is not available. In particular, `GlobSol` restricts itself to problems having less than 9 variables.

System ϵ_{obj}	n	Baron 1.e-8	GlobSol 1.e-8	IBBA+ 1.e-8	Icos 1.e-3	IbexOpt 1.e-3	IbexOpt 1.e-8
ex2_1_7	20	0.33 89		16.75 1574	>600	5.52 2102	6.24 2320
ex2_1_8	24	0.07 7		26.78 1916	>600	5.78 1540	6.50 1702
ex3_1_1	8	0.51 453	>3600	116 131195	180 8930	0.48 605	1.31 1516
ex6_1_4	6	0.25 242	14	2.70 1622	4.28 1109	0.37 471	1.11 1053
ex6_2_14	4	**5.2** 1824	32	208 95170	>600	0.77 765	**1.59** 1237
ex7_2_1	7	0.05 1		24.72 8419	>600	0.80 825	1.17 1197
ex7_2_6	3	0.06 7	1	1.23 1319	2.68 986	0.02 73	5.35 16171
ex7_3_4	12	0.93 268		>3600	>600	1.27 771	1.31 775
ex14_2_1	5	0.03 1	4	36.73 16786	>600	0.82 533	1.09 704
ex14_2_3	6	0.03 1	11	173 46673	>600	2.57 996	2.92 1048
ex14_2_4	5	0.03 1		127 30002	>600	0.95 435	1.02 449
ex14_2_6	5	0.03 1		237 74630	>600	1.20 498	1.29 515

Table 3: **Comparison with competitors on difficult systems.** In case of time out, the second line contains the precision obtained at this time.

System ϵ_{obj}	n	Baron 1.e-8	GlobSol 1.e-8	IBBA+ 1.e-8	Icos 1.e-3	IbexOpt 1.e-3	IbexOpt 1.e-8
ex2_1_9	10	1.52 2050		154 60007	59.9 1549	13 13370	30 30444
ex6_1_1	8	7.64 5616	3203	>3600	>600	13 12811	17 14725
ex6_1_3	12	19.2 11217		>600	>600	46.74 26137	540 204439
ex6_2_6	3	26 26765	306	1575 922664	>600	36.75 34318	173 163227
ex6_2_8	3	19 29469	220	458 265276	>600	29.40 27513	111 97554
ex6_2_9	4	**170** 92143	465	522 203775	>600	12.94 9873	**37** 27461
ex6_2_10	6	**>1000** **2.e-3**	>3600	>3600	>600	**431** 224484	1955 820902
ex6_2_11	3	**55** 45085	273	140 83457	>600	4.02 4487	**22** 24264
ex6_2_12	4	30 19182	193	113 58231	> 600	4.37 4173	122 86722
ex6_2_13	6	**>1000** **2.e-2**	>3600	>3600	>600	**1099** 545676	> 3600 2.e-4
ex7_3_5	13	1.11 309		>3600	136 3699	50.50 40936	55 44147
ex14_1_7	10	1.27 181		>3600	> 600	451 177464	464 181136
ex14_2_7	6	0.03 1		>3600	>600	84.73 17463	85 16759

References

Araya, I.; Trombettoni, G.; and Neveu, B. 2010. Exploiting Monotonicity in Interval Constraint Propagation. In *Proc. AAAI*, 9–14.

Benhamou, F., and Goualard, F. 2000. Universally Quantified Interval Constraints. In *Proc. CP*, 67–82.

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

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

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

Collavizza, H.; Delobel, F.; and Rueher, M. 1999. Extending Consistent Domains of NCSP. In *IJCAI*, 406–413.

Hansen, E. R. 2003. *Global Optimization Using Interval Analysis*. Marcel Dekker Inc.

Horst, R., and Tuy, H. 1966. *Global Optimization: Deterministic Approaches*. Springer.

Kearfott, R., and Novoa III, M. 1990. INTBIS, a portable interval Newton/Bisection package. *ACM Trans. on Mathematical Software* 16(2):152–157.

Kearfott, R. B. 1996. *Rigourous Global Search: Continuous Problems*. Kluwer Academic Publishers.

Lebbah, Y.; Michel, C.; and Rueher, M. 2007. An Efficient and Safe Framework for Solving Optimization Problems. *J. of Computational and Applied Mathematics* 199:372–377.

Lin, Y., and Stadtherr, M. 2004. LP Strategy for the Interval-Newton Method in Deterministic Global Optimization. *Ind. & eng. chemistry research* 43:3741–3749.

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

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

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

Ninin, J.; Messine, F.; and Hansen, P. 2011. A Reliable Affine Relaxation Method for Global Optimization. *Mathematical Programming, accepted for publication*.

Normand, J.-M.; Goldsztejn, A.; Christie, M.; and Benhamou, F. 2010. A Branch and Bound Algorithm for Numerical Max-CSP. *Constraints* 15(2):213–237.

Sherali, H., and Adams, W. 1999. *Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*. Kluwer Academic Publishers.

Tawarmalani, M., and Sahinidis, N. V. 2005. A Polyhedral Branch-and-Cut Approach to Global Optimization. *Mathematical Programming* 103(2):225–249.

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

Van Hentenryck, P.; Michel, L.; and Deville, Y. 1997. *Numerica : A Modeling Lang. for Global Optim.* MIT Press.