Extraction of inner regions to improve the majorant in global optimization under constraints

Ignacio Araya, Gilles Trombettoni, Bertrand Neveu, Gilles Chabert

UTFSM (Chile);
LIRMM, IRIT, Univ. Montpellier 2 (France);
Imagine, LIGM, Univ. Paris-Est (France);
LINA, EMN (France)

JFPC, Aix-en-Pce, June 2013
Outline

1. Inner regions in interval Branch & Bounds
2. Inner polytope extraction (InnerPolytope – IP)
3. Inner box extraction (InHC4)
4. Experiments
Plan

1. Inner regions in interval Branch & Bounds
2. Inner polytope extraction (InnerPolytope – IP)
3. Inner box extraction (InHC4)
4. Experiments
### Intervals

<table>
<thead>
<tr>
<th>Interval</th>
<th>$[x_i] = [x_i, \overline{x}_i]$</th>
<th>${x_i \in \mathbb{R}, x_i \leq x_i \leq \overline{x}_i}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_i$ et $\overline{x}_i$</td>
<td>Floating-point bounds</td>
<td></td>
</tr>
<tr>
<td>$\mathbb{I} \mathbb{R}$</td>
<td>Set of all the intervals</td>
<td></td>
</tr>
<tr>
<td>$m([x_i])$</td>
<td><strong>Midpoint</strong> of $[x_i]$</td>
<td></td>
</tr>
<tr>
<td>$w([x_i]) := \overline{x}_i - x_i$</td>
<td><strong>Width</strong> or size of $[x_i]$</td>
<td></td>
</tr>
</tbody>
</table>

### Boxes

<table>
<thead>
<tr>
<th>Box</th>
<th>$[x] = [x_1] \times \ldots \times [x_i] \times \ldots \times [x_n]$ (explored <strong>search space</strong>)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w([x])$</td>
<td>$\max_n(w([x_i]))$</td>
</tr>
</tbody>
</table>

### Interval arithmetic and (natural) interval interval extension

<table>
<thead>
<tr>
<th>Interval arithmetic</th>
<th>$[1, 2] + [-5, 0] = [-4, 2]; \quad [-4, 2]^2 = [0, 16]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interval extension $[f]$</td>
<td>$f(x_1, x_2) := (x_1 + x_2)^2; \quad [f]([1, 2], [-5, 0]) = [0, 16]$</td>
</tr>
</tbody>
</table>
Constrained global optimization

\[ \min_{x \in [x] \subset \mathbb{R}^n} f(x) \quad \text{s.t.} \quad g(x) \leq 0 \land h(x) = 0 \]
Example of allowed operators: Coconut ex_6_2_9

Variables
x2, x3, x4, x5 in [1e-7, 0.5];

Minimize
(31.4830434782609*x2 + 6*x4)*log(4.8274*x2 + 0.92*x4) -
1.36551138119385*x2 + 2.8555953099828*x4 + 11.5030434782609*x2*
log(x2/(4.8274*x2 + 0.92*x4)) + 20.98*x2*log(x2/(4.196*x2 + 1.4*
x4)) + 7*x4*log(x4/(4.196*x2 + 1.4*x4)) + (4.196*x2 + 1.4*x4)*
log(4.196*x2 + 1.4*x4) + 1.62*x2*log(x2/(7.52678200680961*x2 +
0.443737968424621*x4)) + 0.848*x2*log(x2/(7.52678200680961*x2 +
0.443737968424621*x4)) + 1.728*x2*log(x2/(1.82245052351472*x2 +
1.4300083598626*x4)) + 1.4*x4*log(x4/(0.504772348000588*x2 + 1.4*
x4)) + (31.4830434782609*x3 + 6*x5)*log(4.8274*x3 + 0.92*x5) -
1.36551138119385*x3 + 2.8555953099828*x5 + 11.5030434782609*x3*
log(x3/(4.8274*x3 + 0.92*x5)) + 20.98*x3*log(x3/(4.196*x3 + 1.4*
x5)) + 7*x5*log(x5/(4.196*x3 + 1.4*x5)) + (4.196*x3 + 1.4*x5)*
log(4.196*x3 + 1.4*x5) + 1.62*x3*log(x3/(7.52678200680961*x3 +
0.443737968424621*x5)) + 0.848*x3*log(x3/(7.52678200680961*x3 +
0.443737968424621*x5)) + 1.728*x3*log(x3/(1.82245052351472*x3 +
1.4300083598626*x5)) + 1.4*x5*log(x5/(0.504772348000588*x3 + 1.4*
x5)) - 35.6790434782609*x2*log(x2) - 7.4*x4*log(x4) -
35.6790434782609*x3*log(x3) - 7.4*x5*log(x5);

Subject to
x2 + x3 = 0.5;  x4 + x5 = 0.5;
Interval *Branch & Bound* embedding:

- **combinatorial exploration**: boxes are bisected on one dimension
- **contraction**: reduction of the box with no loss of solution
- search of a good lower bound of the cost (**lower bounding**): convexification
- search of a feasible point with a “good” cost (**upper bounding**): local search (quasi-Newton)
Comparison between IbexOpt [AAAI 2011] and Baron

![Graph](attachment:image.png)
Ingredients in IbexOpt

- Branching heuristic: variant of the smear function ([Kearfott 2010])


- Lower bounding: X-Newton [CPAIOR 2012]: contractor based on a convex polyhedral (first-order) interval interval Taylor

- Upper bounding: inner region extraction in the feasible space... with a non null volume
Relaxing equations in IbexOpt

Inspired by the IBBA solver [Ninin, Messine, Hansen 2010], pure equalities $h_j(x) = 0$ are relaxed by inequalities:

$-\epsilon_{eq} \leq h_j(x) \leq +\epsilon_{eq}$.

Three categories of global optimizers w.r.t. rigor:

- Most of deterministic global optimizers are not "exact"/valid/reliable/rigorous.

- Reliability of a small relaxation of the problem:
  Output: a floating-point vector $x$ $\epsilon$-minimizing:
  $f(x)$ s.t. $g(x) \leq 0 \land (-\epsilon_{eq} \leq h(x) \leq +\epsilon_{eq})$.

- Reliability of the problem:
  Output: a tiny box guaranteed to contain a real-valued vector $x$ $\epsilon$-minimizing:
  $f(x)$ s.t. $g(x) \leq 0 \land h(x) = 0$. 
Contribution: **upper bounding** in inner regions

\[
\arg min_{x \in [x] \subseteq \mathbb{R}^n} f(x) \text{ s.t. } g(x) \leq 0 \land (-\epsilon_{eq} \leq h(x) \leq +\epsilon_{eq})
\]

Original computation of a feasible upper bound

1. Extraction of an **inner region**
   - Inner region \(\equiv\) box or polytope in which all points are feasible.

2. Minimization of the objective function in these entirely feasible regions.
Plan

1. Inner regions in interval Branch & Bounds
2. Inner polytope extraction (InnerPolytope – IP)
3. Inner box extraction (InHC4)
4. Experiments
Example of inner hyperplane construction

\[ g_1(x_1, x_2) = x_1^3 + \cos(x_1) - \sin(x_2) - 0.15 \leq 0 \]

Domaine \([x_1] \times [x_2] = [-0.32, 0.52] \times [0.90, 1.06]\)
Consider a function $f : \mathbb{R}^n \to \mathbb{R}$.
For all point $x$ in the studied box $[x]$, we have:

$$f(x) \leq f(x) + \sum_i \bar{a}_i(x_i - \underline{x}_i), \text{ where } [a_i] = \left[ \frac{\partial f}{\partial x_i} \right]([x]).$$

If an inequality $g_j(x) \leq 0$ is handled, this builds a hyper-plane $g_j'(x)$ s.t. $g_j(x) \leq g_j'(x) \leq 0$. 

In Inner regions in interval Branch & Bounds
Inner polytope extraction (InnerPolytope – IP)
Inner box extraction (InHC4)
Experiments
Plan

1. Inner regions in interval Branch & Bounds
2. Inner polytope extraction (InnerPolytope – IP)
3. Inner box extraction (InHC4)
4. Experiments
Extraction of an inner box constraint per constraint

A main loop handles every inequality constraint $g_j$ ($j$ ranging from 1 to $m$) in sequence:

1. Handling $g_1(x) \leq 0$ extracts an inner box (w.r.t. this constraint) from the outer box.

2. Handling $g_j(x) \leq 0$ extracts an inner box inside the (inner) box returned by the handling of constraint $g_{j-1}(x) \leq 0$.
   $\Rightarrow$ This box is inner w.r.t. the first $j$ constraints.

3. Successfully handling $g_m(x) \leq 0$ computes an inner box (w.r.t. the whole system of constraints).
Extracting an inner box w.r.t. one constraint

Close to the main HC4-Revise procedure [Benhamou et al., Messine]. Extends the case-by-case approach by [Chabert and Beldiceanu, CP 2010].

1. Bottom-up evaluation (like in HC4-Revise)
2. Top-down “projection”:
   - For every unary operator $op$, compute an inner box $[x]^{in}$ s.t. $\forall x \in [x]^{in} : op(x) \in [z]$ ($z \leq op(x) \leq \bar{z}$)
   - For every binary operator $op$, compute an inner box $[x_1]^{in} \times [x_2]^{in}$ s.t. $\forall (x_1, x_2) \in [x_1]^{in} \times [x_2]^{in} : x_1 \; op \; x_2 \in [z]$
Inner boxes for primitive constraints

Case 2: $z = op(x)$ is not monotonic w.r.t. $x$ in $[x] \times [z]$ (e.g., $x^2$, sine): piecewise monotonicity analysis: choice of one interval in a union of intervals

Remark: outward rounding $\Rightarrow$ inward rounding
Case 4: MonoMaxInnerBox used in several monotonic subcases of the multiplication (or division) operator: $x_1 \cdot x_2 \in [z]$
Properties

- **Primitive constraint**: Every implemented unary and binary operator computes a maximal inner box, modulo the loss involved by inward roundoffs.

- **Single constraint**: If a constraint contains only a single occurrence of each variable, InHC4-Revise computes a maximal inner box, when one such box is found.

*InnerPolytope and InHC4 are heuristics!!*
Plan

1. Inner regions in interval Branch & Bounds
2. Inner polytope extraction (InnerPolytope – IP)
3. Inner box extraction (InHC4)
4. Experiments
**Implementation and benchmark**

- **IbexOpt** implemented in (and enriches) the C++ library **Ibex** (Interval Based EXplorer) developed by Chabert (rather architecture) and colleagues (rather algorithms).

- Benchmark: from the serie 1 of the Coconut constrained global optimization benchmark:
  - all the 59 systems
  - solved in a runtime between 1 second and 1 hour
  - by IbexOpt (with IN: upper bounding based on INner region)
  - or by IbexOpt-- (without IN; with random probing)
Sample of 59 instances

<table>
<thead>
<tr>
<th>name</th>
<th>n</th>
<th>name</th>
<th>n</th>
<th>name</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>alkyl (rr)</td>
<td>14</td>
<td>ex6.1_4</td>
<td>6</td>
<td>ex9.2_6</td>
<td>16</td>
</tr>
<tr>
<td>bearing</td>
<td>14</td>
<td>ex6.2_6</td>
<td>3</td>
<td>ex14.1_2</td>
<td>6</td>
</tr>
<tr>
<td>ex2.1.3</td>
<td>13</td>
<td>ex6.2_8</td>
<td>3</td>
<td>ex14.1_6</td>
<td>9</td>
</tr>
<tr>
<td>ex2.1.5</td>
<td>10</td>
<td>ex6.2_9</td>
<td>4</td>
<td>ex14.1_7</td>
<td>10</td>
</tr>
<tr>
<td>ex2.1.6</td>
<td>10</td>
<td>ex6.2_10</td>
<td>6</td>
<td>ex14.2_1</td>
<td>5</td>
</tr>
<tr>
<td>ex2.1.7</td>
<td>20</td>
<td>ex6.2_11</td>
<td>3</td>
<td>ex14.2_3</td>
<td>6</td>
</tr>
<tr>
<td>ex2.1.8</td>
<td>24</td>
<td>ex6.2_12</td>
<td>4</td>
<td>ex14.2_4</td>
<td>6</td>
</tr>
<tr>
<td>ex2.1.9</td>
<td>10</td>
<td>ex6.2_14</td>
<td>4</td>
<td>ex14.2_6</td>
<td>5</td>
</tr>
<tr>
<td>ex2.1.10</td>
<td>20</td>
<td>ex7.2_1</td>
<td>7</td>
<td>ex14.2_7</td>
<td>6</td>
</tr>
<tr>
<td>ex3.1.1</td>
<td>8</td>
<td>ex7.2_3</td>
<td>9</td>
<td>haverly</td>
<td>12</td>
</tr>
<tr>
<td>ex3.1.3</td>
<td>6</td>
<td>ex7.2_7 (rr)</td>
<td>4</td>
<td>hhfair</td>
<td>28</td>
</tr>
<tr>
<td>ex5.2.2_c1</td>
<td>9</td>
<td>ex7.2_8 (rr)</td>
<td>8</td>
<td>himmel11</td>
<td>9</td>
</tr>
<tr>
<td>ex5.2.2_c2</td>
<td>9</td>
<td>ex7.2_9 (rr)</td>
<td>10</td>
<td>himmel16</td>
<td>18</td>
</tr>
<tr>
<td>ex5.2.2_c3</td>
<td>9</td>
<td>ex7.3_4</td>
<td>12</td>
<td>house</td>
<td>8</td>
</tr>
<tr>
<td>ex5.2.4</td>
<td>7</td>
<td>ex7.3_5</td>
<td>13</td>
<td>hydro</td>
<td>30</td>
</tr>
<tr>
<td>ex5.3.2</td>
<td>22</td>
<td>ex8.1_8</td>
<td>6</td>
<td>immun (rr)</td>
<td>21</td>
</tr>
<tr>
<td>ex5.4.2</td>
<td>8</td>
<td>ex8.5_1</td>
<td>6</td>
<td>launch</td>
<td>38</td>
</tr>
<tr>
<td>ex5.4.3</td>
<td>16</td>
<td>ex8.5_2</td>
<td>6</td>
<td>meanvar</td>
<td>7</td>
</tr>
<tr>
<td>ex6.1.1</td>
<td>8</td>
<td>ex8.5_3</td>
<td>5</td>
<td>process</td>
<td>10</td>
</tr>
<tr>
<td>ex6.1.3</td>
<td>12</td>
<td>ex8.5_6</td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Performance profile
Gains

Gain = \frac{\text{time(IbexOpt)} - \text{time}(X)}{\text{time}(X)}

X = \text{IN (IbexOpt) or InHC4 or IP (InnerPolytope)}

<table>
<thead>
<tr>
<th>Gains</th>
<th>IN</th>
<th>InHC4</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 0.69</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.69 – 0.9</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>0.9 – 1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1 – 1.1</td>
<td>0</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>1.1 – 2</td>
<td>14</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>2 – 10</td>
<td>19</td>
<td>15</td>
<td>22</td>
</tr>
<tr>
<td>10 – 100</td>
<td>11</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>&gt; 100</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>MO</td>
<td>9</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>Solved with no MO</td>
<td>59</td>
<td>57</td>
<td>59</td>
</tr>
</tbody>
</table>
Conclusion

- Non costly heuristics run at each node of the interval branch and bound.

- InHC4 is useful for better upperbounding; InnerPolytope is also useful; InnerPolytope seems more useful than InHC4; but InnerPolytope+InHC4 is robust: better than IP and InHC4 individually.

- Upper bounding the cost in inner regions seems relevant for systems with size less than 40.
Qualitative study

Study using other measures on the most difficult systems

- Indicator 1: $\frac{\text{upperBound} - \text{bestCost}}{\text{upperBound} - \text{lowerBound}}$ (average on all the nodes once a first upper bound has been found)

- Indicator 2: the mean size of boxes and the number of times where InHC4 (only) and IP (only) improve the upper bound

- Other observation: variability in runtime for IbexOpt with InHC4 only (due to the random choices).
Case 3: For binary (or n-ary) operators that are monotonic w.r.t. each of their variables, a generic MonoMaxInnerBox procedure can compute randomly one maximal inner box, if one such box exists.

Example: $x_1 + x_2 \leq z \ (g(x_1, x_2) := x_1 + x_2 - z \leq 0)$. 
Inner boxes for primitive constraints

Monotonicity analysis (four cases)
Case 1: $z = \text{op}(x)$ is continuous and monotonic w.r.t. $x$ in $[x] \times [z]$ (e.g., $\log$, $\exp$): “inverse” operator (like in HC4-Revise, but with inward rounding)