AbsTaylor: Finding Inner Regions for Nonlinear Constraint Systems with Linearizations and Absolute Values

Ignacio Araya[1,a)] and Victor Reyes[1,b)]

[1]*Pontificia Universidad Católica de Valparaíso, Escuela de Ingeniería Informática, Chile.*

[a)]Corresponding author: ignacio.araya@pucv.cl
[b)]victor.reyes.r@mail.pucv.cl

Abstract.
In this paper we propose a simple and cheap method for extracting *inner polytopes*, i.e., entirely feasible convex regions in which all points satisfy the constraints. The method performs an inner linearization of a set of nonlinear constraints by using a Taylor form. Unlike a previous proposal, the expansion point of the Taylor form is not limited to the bounds of the domains; it can be given by any point inside the studied region producing, in general, a tighter approximation.

The approach was used as an upper bounding method in a state-of-the-art global branch & bound optimizer. In the studied instances, the new method finds in average much more inner regions (in 20% of the processed nodes) than the original approach (in 5% of the nodes).

INTRODUCTION

This paper deals with Nonlinear Continuous global Optimization Problems (NCOPs) handled by interval branch and bound. The problem is defined by:

$$\min_{x \in \boldsymbol{x}} f(x) \;\; s.t. \;\; g(x) \leq 0 \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the real-valued objective function and $g : \mathbb{R}^n \to \mathbb{R}^m$ the inequality constraints. f and g may be nonlinear (convex or non-convex) functions. $x = (x_1, ..., x_i, ...x_n)$ is a vector of variables varying in a domain (i.e., a box) $\boldsymbol{x}$.[1]

When solving NCOP problems, interval branch & bound solvers generally start from an initial box $\boldsymbol{x}$. Then, a search tree is build by interleaving bisection and filtering methods. *Bisection* methods are in charge of dividing the current box into two (or more) sub-boxes by splitting the domain of one variable. *Filtering methods* attempt to reduce the boxes by eliminating inconsistent values from their bounds without loss of solutions. If all the values in some domain are filtered, then the corresponding branch of the search tree is discarded. Solvers also include *upper-bounding procedures* which attempt to find new feasible solutions with costs (upper bounds) more and more close to the optimal cost. A lower bound of the objective function is also kept and updated taking into account the current unprocessed boxes.

The search finishes when the difference between the upper and the lower bound of the objective function is lower than a required precision ϵ. The solution related to the upper bound is called an *ϵ-optimal solution* of the problem because its cost differs in at most ϵ from the optimal cost. Otherwise, if all the nodes of the search tree have been removed and no feasible solution has been found, then the treated problem is unfeasible.

[1]An interval $\boldsymbol{x}_i = [\underline{x_i}, \overline{x_i}]$ defines the set of reals x_i, such that $\underline{x_i} \leq x_i \leq \overline{x_i}$. $\text{mid}(\boldsymbol{x}) = \frac{\underline{x_i}+\overline{x_i}}{2}$ denotes the midpoint of the interval $\boldsymbol{x}$. A box $\boldsymbol{x}$ is a Cartesian product of intervals $\boldsymbol{x}_1 \times ... \times \boldsymbol{x}_i \times ... \times \boldsymbol{x}_n$.

Upper bounding is a crucial component of NCOP solvers. Finding a new best solution x' with cost $ub = f(x')$, implies a reduction of the feasible space through the addition of the auxiliary constraint: $f(x) \leq ub - \epsilon$.[2] Thus, solvers should put considerable effort into finding solutions with near-optimal costs. However, once an ϵ-optimal solution has been found, any effort spent in finding better solutions becomes useless.

In interval solvers, low-cost upper-bounding techniques are generally used. They range from simply trying the midpoint point of the box [1] to more sophisticated methods, such as applying the Newton method [2], or extracting convex inner regions [3].

In [3], we proposed an upper bounding method which builds an *inner linearization* of each nonlinear constraint in the system. In other words, for each inequality $g_j(x) \leq 0$, we generate a constraint $h_j(x) \leq 0$, such that $g_j(x) \leq h_j(x)$ for any x in the current box $\boldsymbol{x}$. If it succeeds in building the inner convex region or polytope, we minimize a linearization of the objective function using a simplex algorithm. Finally, the found *feasible* solution is used to update the upper bound. The inner linearization is obtained by a first order interval Taylor form of the nonlinear function, i.e., for all $x \in \boldsymbol{x}$, we have:

$$g_j(x) \leq h_j(x) = g_j(\underline{x}) + \sum_{i}^{n} \overline{J_{ij}} \cdot (x_i - \underline{x_i}) \tag{2}$$

where $\boldsymbol{J}$ is an interval Jacobian matrix of g in the box $\boldsymbol{x}$, i.e., any element $\boldsymbol{J}_{ij}$ in the matrix corresponds to an interval overestimation of the image of the partial derivative $\frac{\partial g_j}{\partial x_i}(x)$ over $\boldsymbol{x}$. Fig. 1-left shows an example of the inner linearization for an univariate function.

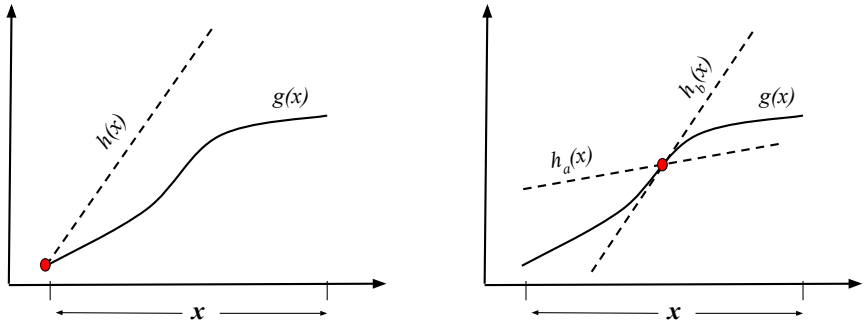

FIGURE 1. Inner linearizations by using the Taylor form: (left) expansion point in a bound of the interval, (right) expansion point in the midpoint of the interval.

Note that a corner of the box ($\underline{x}$) is chosen as expansion point of the Taylor form. Choosing the midpoint of the box would generate two linear constraints in the univariate case (see Fig. 1-right). However, in the general case it would generate 2^n constraints[3], thus it is not a viable option.

A TAYLOR-BASED ABSOLUTE VALUE LINEARIZATION

We propose a simple Taylor-based inner linearization which uses any point in the box (e.g., the midpoint) as the expansion point but without generating 2^n constraints. To do that, we first decompose the interval partial derivatives into two values: $\boldsymbol{J}_{ij} = c_{ij} + \boldsymbol{d}_{ij}$, where the real value c_{ij} is the midpoint of the interval derivative and $\boldsymbol{d}_{ij} = \boldsymbol{J}_{ij} - c_{ij}$ (note that $\overline{d_{ij}} = -\underline{d_{ij}}$).

Then, we propose the following inner linearization with absolute values:

$$g_j(x) \leq h_j(x) = g_j(x') + \sum_{i}^{n} \left(c_{ij} \cdot (x_i - x'_i) + \overline{d_{ij}} \cdot |x_i - x'_i| \right) \tag{3}$$

[2] After finding a solution with cost ub, we are not interested in finding more solutions with costs in $[ub - \epsilon, ub]$, because the solver just guarantees that *one* ϵ-optimal solution is found.

[3] i.e., the set of constraints $g_j(\text{mid}(\boldsymbol{x})) + a \cdot (x - \text{mid}(\boldsymbol{x})) \leq 0$, for all $a \in \{\underline{J_{1j}}, \overline{J_{1j}}\} \times \{\underline{J_{2j}}, \overline{J_{2j}}\} \times ... \times \{\underline{J_{nj}}, \overline{J_{nj}}\}$.

$x' \in \mathbb{R}^n$ can be any point inside the box $\boldsymbol{x}$, for instance $x' = \text{mid}(\boldsymbol{x})$. The feasible region of the constraint $h_j(x) \leq 0$ is equivalent to the feasible region achieved by the 2^n constraints generated by the linearization (2) with x' as the expansion point. In the example of Fig. 1-right, $h(x)$ is equivalent to $h_a(x)$ when $x \leq \text{mid}(\boldsymbol{x})$ and to $h_b(x)$ when $x > \text{mid}(\boldsymbol{x})$.

FINDING A SOLUTION IN THE POLYTOPE

Similar to that proposed in [3], for finding a feasible solution we propose to minimize a linearization of the objective function subject to the constraint system generated by (3), i.e.,

$$\begin{array}{ll} \min\limits_{x\in\boldsymbol{x}} & f(x') + \sum\limits_{i}^{n} \text{mid}(\boldsymbol{J}_{ij}) \cdot (x_i - x'_i) \\ \text{s.t} & g_j(x') + \sum\limits_{i}^{n} \left(c_{ij} \cdot (x_i - x'_i) + \overline{d_{ij}} \cdot |x_i - x'_i| \right) \leq 0 \end{array} \tag{4}$$

In order to deal with the absolute value functions, we replace the expressions $|x_i - x'_i|$ by auxiliary variables u_i. Then, as the partial derivative of the constraint functions is positive w.r.t. the auxiliary variables (i.e., $\overline{d_{ij}} \geq 0$), we add the constraints $u_i \geq x_i - x'_i$ and $u_i \geq -(x_i - x'_i)$. We obtain the equivalent linear program:

$$\begin{array}{ll} \min\limits_{x\in\boldsymbol{x}} & f(x') + \sum\limits_{i}^{n} \text{mid}(\boldsymbol{J}_{ij}) \cdot (x_i - x'_i) \\ \text{s.t} & g_j(x') + \sum\limits_{i}^{n} \left(c_{ij} \cdot (x_i - x'_i) + \overline{d_{ij}} \cdot u_i \right) \leq 0, \quad \forall j = 1..m \\ & u_i \geq x_i - x'_i, \quad \forall i = 1..n \\ & u_i \geq -(x_i - x'_i), \quad \forall i = 1..n \end{array} \tag{5}$$

The linear program can be solved by a simplex algorithm. Whenever a solution is found, this solution can be used for updating the current upper bound of the solver.

EXPERIMENTS

The presented approach was implemented in IbexOpt [4]. We call AbsTaylor to the strategy using our absolute-value-based linearization approach (using $x' = \text{mid}(\boldsymbol{x})$) and TwiceTaylor to the strategy using both, XTaylor (the classical approach presented in the introduction) and AbsTaylor for finding upper bounds. All strategies also use a cheap additional upper bounding method based on the extraction of feasible boxes [3].

The experiments were run on a computer with an Intel Core i7-4700MQ CPU 2.40GHz and 8GB RAM, running on Ubuntu 16.04. The set of instances were selected from the COCONUT benchmarks for global optimization[4]. We selected all the instances solved by the strategy using XTaylor for upper bounding in a time comprised between 1 and 3600 seconds (56 instances).

Definition 1 **Gain in relative time.** *We define as* **relative time** $t_r(a, b, \pi)$ *the ratio between the mean time* ($\bar{t}$) *taken by a strategy a and the sum of mean times taken by the strategies a and b in solving an instance* π*, i.e.,* $t_r(a, b, \pi) = \frac{\bar{t}(a,\pi)}{\bar{t}(a,\pi)+\bar{t}(b,\pi)}$*. Thus, the* **gain in relative time** *of a strategy a w.r.t. a strategy b is given by* $\frac{\sum_{\pi\in\Pi} t_r(b,a,\pi)}{\sum_{\pi\in\Pi} t_r(a,b,\pi)}$*, where* Π *is the set of the considered instances.*

Table 1 reports a comparison between the time spent by the strategies, using different values for the precision ϵ of the objective function. The last row of each column corresponds to the gain in relative time w.r.t. the reference strategy (i.e., XTaylor). The table only considers the instances with time differences larger than 10% (in parentheses we report the number of considered instances for each strategy).

AbsTaylor and TwiceTaylor seem to converge more quickly to low precision values (e.g., 10^{-1} and 10^{-3}) with gains of 1.21 and 1.17 respectively when $\epsilon = 10^{-1}$. For higher precision values AbsTaylor and XTaylor report similar results. On the other hand, TwiceTaylor outperforms both strategies when $\epsilon = 10^{-5}$. Note that TwiceTaylor reports worse results than AbsTaylor when $\epsilon = 10^{-3}$. This does not,

[4] http://www.mat.univie.ac.at/~neum/glopt/coconut/Benchmark/Benchmark.html

however, mean that TWICETAYLOR is less effective than ABSTAYLOR on these precision but that it is more expensive, i.e., it processes less nodes but at a higher cost.

TABLE 1. CPU times when the best solution found so far reaches a precision of 10^{-1}, 10^{-3} or 10^{-5} for each strategy and a selected set of benchmark instances.

	XTAYLOR			ABSTAYLOR			TWICETAYLOR		
Benchmarks	10^{-1}	10^{-3}	10^{-5}	10^{-1}	10^{-3}	10^{-5}	10^{-1}	10^{-3}	10^{-5}
ex5_4_4	**14.54**	**120.65**	**221.85**	45.02	279.82	279.82	44.08	224.54	225.68
ex6_1_3	1.55	66.37	149.92	**1.16**	**59.2**0	**118.11**	1.24	59.74	127.55
ex7_2_4	0.12	3.22	13.80	**0.05**	3.20	11.60	0.09	**2.49**	**7.83**
ex7_2_8	0.18	2.50	14.12	**0.04**	**1.55**	**10.22**	**0.04**	1.72	10.90
ex8_5_2	2.30	11.17	11.17	**2.12**	**4.65**	**8.86**	3.42	6.04	9.15
ex14_2_7	28.40	87.80	91.66	4.96	91.59	92.34	**4.77**	**69.58**	**70.13**
immun	**7.96**	**10.93**	**12.10**	13.63	13.68	15.37	44.14	44.29	44.29
ramsey	1.72	1.72	10.03	**0.11**	**0.23**	14.29	**0.11**	0.84	**6.68**
hs113	10.37	20.12	27.76	5.77	**19.67**	**23.25**	**4.74**	22.60	26.79
Gain in rel. time				1.21 (21)	1.15 (29)	1.04 (39)	1.17 (22)	1.07 (40)	1.15 (31)

In a second series of experiments we evaluated the effectiveness of ABSTAYLOR and XTAYLOR by counting the number of times each method finds a new upper bound inside the TWICETAYLOR approach. We observed that, in average, ABSTAYLOR finds new upper bounds 63% of the times and XTAYLOR only a 24%. We also counted the number of times each strategy was successful in finding a feasible region. ABSTAYLOR was successful in 20% of the cases while XTAYLOR was successful only in 5% of them (we did not consider the 17 instances where both strategies were always successful).

Given that ABSTAYLOR is able to generate feasible regions by using any point as the expansion point, we plan to incorporate iterative methods, such as Frank-Wolfe [5] or trust region algorithms [6], in order to find better upper bounds faster in the current box.

REFERENCES

[1] J. Ninin, F. Messine, and P. Hansen, "A reliable affine relaxation method for global optimization," *4OR*, vol. 13, no. 3, pp. 247–277, 2015.

[2] Y. Lebbah, "Icos: a branch and bound based solver for rigorous global optimization," *Optimization Methods & Software*, vol. 24, no. 4-5, pp. 709–726, 2009.

[3] I. Araya, G. Trombettoni, B. Neveu, and G. Chabert, "Upper bounding in inner regions for global optimization under inequality constraints," *Journal of Global Optimization*, vol. 60, no. 2, pp. 145–164, 2014.

[4] G. Chabert and L. Jaulin, "Contractor Programming," *Artificial Intelligence*, vol. 173, pp. 1079–1100, 2009.

[5] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval research logistics quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[6] Y.-x. Yuan, "A review of trust region algorithms for optimization," in *Iciam*, vol. 99, pp. 271–282, 2000.