Upper Bounding in Inner Regions for Global Optimization under Inequality Constraints

Ignacio Araya,[1] Gilles Trombettoni,[2] Bertrand Neveu,[3] and Gilles Chabert[4]

[1]*UTFSM, Universitad Federico Santa Maria, Valparaiso, Chile* iaraya@inf.utfsm.cl

[2]*IRIT, I3S, INRIA, Université Nice–Sophia, France* Gilles.Trombettoni@inria.fr

[3]*Imagine LIGM Université Paris–Est, France* Bertrand.Neveu@enpc.fr

[4]*LINA, Ecole de Mines de Nantes, France* Gilles.Chabert@emn.fr

Abstract

In deterministic constrained global optimization, upper bounding the objective function generally resorts to local minimization at the nodes of the branch and bound. The local minimization process is sometimes costly when constraints must be respected.

We propose in this paper an alternative approach when the constraints are inequalities or relaxed equalities so that the feasible space has a non-null volume. First, we extract an inner region, i.e., an (entirely feasible) convex polyhedron or box in which all points satisfy the constraints. Second, we select a point inside the extracted inner region and update the upper bound with its cost.

We use two inner region extraction algorithms implemented in our interval B&B called Ibex-Opt [7]. This upper bounding shows good performance in medium-sized systems proposed in the COCONUT suite.

Keywords: global optimization, upper bounding, intervals, branch and bound, inner regions

1. Upper bounding in inner regions

In deterministic constrained global optimization, upper bounding the objective function consists in finding a feasible point that improves the best cost already found in the branch and bound. Most global optimizers resort to local minimization[1] using a Lagrangian relaxation. The considered function is sometimes big, which may render the local minimization slow.

This paper describes an alternative approach for global optimization under inequality constraints defined by: $\min_{x\in[x]} f(x)$ *subject to* $g(x) \leq 0$, where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is the real-valued objective function and $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is a vector-valued function. $x = \{x_1, ..., x_i, ...x_n\}$ is a vector of variables varying in a box $[x]$.[2] x is said to be feasible if it satisfies the constraints.

The main idea is to exploit so called inner regions, i.e., subsets of the search space in which all points are feasible.

Definition 1. Consider a system $(f, g, x, [x])$ comprising only inequality constraints. An inner region r^{in} is a feasible subset of $[x]$, i.e., $r^{in} \subset [x]$ and all points $x \in r^{in}$ satisfy $g(x) \leq 0$.

At every node (iteration) of our interval B&B named IbexOpt [7], the cost is bounded above by using two inner region extraction algorithms, called InHC4 and InnerPolytope. InHC4 is described in Section 2. It tries to extract an inner box from the current outer box. If it fails, one simply picks a point randomly inside the outer box and checks its feasibility. If it succeeds, a simple monotonicity analysis of f replaces the intervals of the monotonic variables by the adequate bounds in the found inner box and the other values are randomly chosen. Then, InnerPolytope [7] builds a hyperplane for every inequality constraint. The hyperplane is produced by a special convex form of interval Taylor where the expansion point is chosen

[1]We consider minimization in this paper without loss of generality.

[2]An interval $[x_i] = [\underline{x_i}, \overline{x_i}]$ defines the set of reals x_i s.t. $\underline{x_i} \leq x_i \leq \overline{x_i}$. A box $[x]$ is the Cartesian product of intervals $[x_1] \times ... \times [x_i] \times ... \times [x_n]$.

at a corner of the studied outer box. If it succeeds in building an inner polytope, the point minimizing the linearized form of the objective function is used to update the upper bound.

Contribution and limits

Contrarily to existing approaches, the proposed inner region extraction algorithms separate the feasibility part (handled first, by inner region extraction) and the computation of the cost (handled next, inside the found inner region).

It is important to highlight that, like the other inner region extraction algorithms, ours can fail in finding an inner region even if one such region exists. However, they are rather inexpensive. In particular, InHC4 is faster when it fails in finding an inner region for a given constraint because the loop on all the constraints can be prematurely stopped (see below).

This upper bounding based on inner region extraction could also apply to "thick" and relaxed equations that define a feasible space with a non-null volume. A thick equation has at least one coefficient that can be modeled by an interval constant. This parameter corresponds to a bounded uncertainty, e.g., an imprecision on a measurement, or an irrational constant, like π. A pure equality $f_k(x) = 0$ can also be handled with a relaxation as a thick equation $f_k(x) \in [-\epsilon_{eq}, +\epsilon_{eq}]$, i.e, two inequalities $-\epsilon_{eq} \leq f_k(x) \leq \epsilon_{eq}$. In this case of course, we can only guarantee the global optimum of the relaxed system, but ϵ_{eq} can often be chosen almost arbitrarily small.

2. The InHC4 inner box extraction algorithm

InHC4 follows the simple and general scheme proposed in [1, 4]. A main loop handles every constraint once in sequence and intersects incrementally the different boxes built.[3]

- The handling of the j^{th} constraint uses as input the inner box returned by the handling of constraint $g_{j-1}(x) \leq 0$. This box is inner w.r.t. the first $j-1$ contraints. Handling the first constraint $g_1(x) \leq 0$ is achieved with the outer box.
- Handling the constraint $g_j(x) \leq 0$ consists in finding, inside the input box under construction, a box which is inner w.r.t. this single constraint.

Thus, if a box is returned by the handling of the last constraint, this box is inner w.r.t. all the constraints.

The handling of an individual constraint in InHC4 is radically different from [4, 1]. Contrarily to their refutation process, our InHC4-Revise procedure tries to extract an inner region at each operator of the constraint. Like the main procedure of the state-of-the-art constraint propagation algorithm HC4 [2, 5], our InHC4-Revise procedure (InHC4R) works with a tree representation of the constraint, as illustrated in Fig. 1.

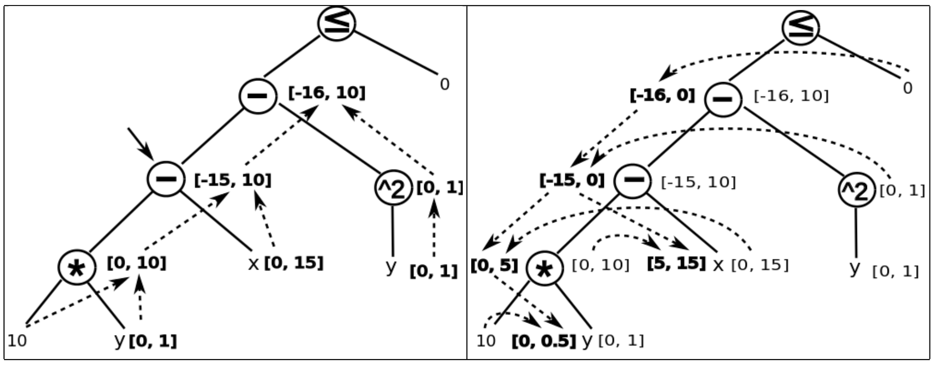

Figure 1. Binary tree representation of the constraint $10y-x-y^2 \leq 0$. Left: First bottom-up evaluation phase. Right: top-down inner projection phase.

Let us denote by $[x]$ the input box and $g_j(x) \leq 0$ the constraint. Each node of the tree is associated to an interval, the intervals related to the leaves are initialized with the corresponding values in $[x]$. Then, the following two phases are performed:

[3]This scheme radically differs from constraint propagation achieved by HC4 that can handle a constraint several times.

- Bottom-up evaluation (see Fig. 1–left): The tree is traversed from the leaves to the root and intervals associated to an operator are computed with interval arithmetics. For example, the node pointed by the arrow is initialized with the interval $[0, 10]-[0, 15] = [-15, 10]$. Thus, every node contains an interval corresponding to the natural interval evaluation of the subexpression.
- Top-down inner projection (see Fig. 1–right): In each node related to a binary operator *op* and to an interval $[z]$, the 2-dimensional box corresponding to its children x_1 and x_2 is reduced to an inner box $[x_1]^{in} \times [x_2]^{in}$ such that:
$$\forall (x_1, x_2) \in [x_1]^{in} \times [x_2]^{in} : \quad x_1 \; op \; x_2 \in [z] \tag{1}$$
If *op* corresponds to a unary operator, its unique child is reduced to $[x]^{in}$, such that:
$$\forall x \in [x]^{in} : \quad op(x) \in [z] \tag{2}$$

If the inner projection returns an empty box (i.e., no box satisfies relation (1) or (2)), then the top-down process is interrupted. It means that InHC4R has failed in finding a box that is inner w.r.t. $g_j(x) \leq 0$. Since the approach is not complete because not all the feasible space is extracted during the top-down traversal, an inner box could be indeed missed by the process.

Consider the product operator of Fig. 1–right and its two children. The reduced intervals appear in bold in the left side of each node. After the reduction of the product operator, its interval becomes $[0, 5]$. Before reduction, its children are associated to the intervals $[10, 10]$ and $[0, 1]$. They are then reduced to $[10, 10]$ and $[0, 0.5]$ respectively. The reduction agrees with relation (1), i.e, $\forall y \in [0, 0.5] : \; 10 * y \in [0, 5]$.

2.1 Inner projection for unary and binary basic operators

For unary, monotonic and continous operators, like log and exp, the inner projection is trivial and computes the (maximum) inner interval (i.e, no feasible point is lost, modulo roundoffs). It is very close to a standard projection in HC4R. However, for managing floating-point roundoff errors, the outward rounding of HC4R is replaced by inward rounding.

For non monotonic unary operator like x^2 or $sinus$, a union of intervals is computed by HC4R, before returning the hull of these intervals. For an inner projection in InHC4R instead, only a single interval is kept since holes between these intervals contain inconsistent points.

For binary operators, Chabert and Beldiceanu in [3] proposed inner projections, but with a case-by-case approach. We have extended their approach and built a more generic projection based on monotonicity properties. There usually exists an infinite number of maximal boxes (as depicted in Fig. 2), and we have succeeded in designing inner projection operators that select randomly one maximal inner box. Note that these inner projections lead to heuristic choices since a single box cannot include the whole inner/feasible space. Also note that the two inequalities $\underline{z} \leq (x_1 \; op \; x_2) \leq \overline{z}$ are handled in sequence, the inner box computed for one inequality being used as input of the second one.

For binary (or n-ary) operators that are monotonic w.r.t. each of their variables, a generic procedure, called MonoMaxInnerBox, can compute randomly one maximal inner box, if one such box exists, as shown in Fig. 2. This procedure is of course used for the addition and subtraction operators. It is also used for handling several (monotonic) subcases of the non monotonic binary operators: the multiplication and the division. Fig. 3 illustrates the two main cases for the multiplication $x_1 * x_2 \in [z]$, depending whether 0 belongs or not to $[z]$.

Handling the division operator amounts in rewriting $x_1/x_2 = x_1 * \frac{1}{x_2} \in [z]$, although a direct implementation would also be possible.

2.2 Properties

We have proven that every implemented unary and binary operator computes a maximal inner box, modulo the loss involved by inward roundoffs. In case a constraint contains only a single occurrence of each variable, InHC4R thus computes a maximal inner box, when one such box is found. The result finally holds for a system of inequality constraints handled by InHC4.

3. Experiments

We have tested our original upper bounding procedure on a sample of about thirty constrained global optimization found in the COCONUT benchmark suite. 24 of them correspond to the

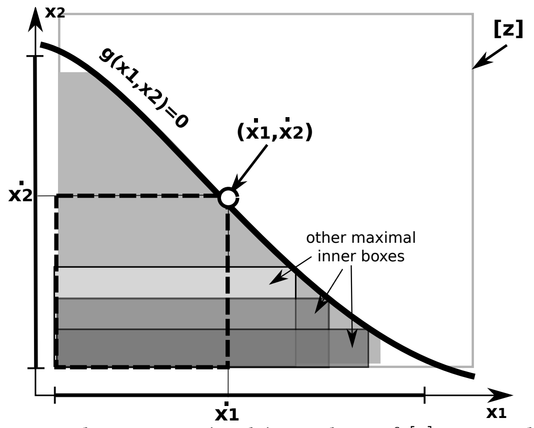

Figure 2. The dotted box corresponds to a maximal inner box of $[z]$ w.r.t. the constraint $g(x_1, x_2) \leq 0$. A point $\dot{x_1}$ is randomly picked inside the range of allowed values illustrated by the horizontal segment. Only one remaining value $\dot{x_2}$ can then make the computed inner box maximal.

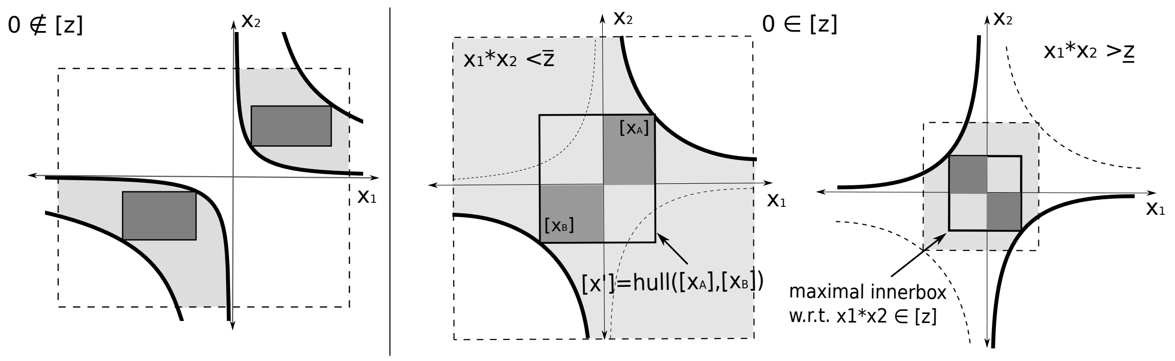

Figure 3. Inner projection for the binary multiplication. Left: Two maximal boxes that can indifferently be computed by MonoMaxInnerBox in the two disjoint inner regions (quadrants) defined by the operator $x_1 * x_2 \in [z] \geq 0$. Middle and right: Maximal box computed for $x_1 * x_2 \in [z] \ni 0$ ($\overline{z} \geq -\underline{z}$) with four calls to MonoMaxInnerBox (boxes in grey).

most difficult systems selected by Ninin et al. [6]. Equations $f_k(x) = 0$ are relaxed by inequalities $-\epsilon_{eq} \leq f_k(x) \leq \epsilon_{eq}$, with $\epsilon_{eq} =$1.e-8. The main results are the following. A first experiment highlights the benefits of this upper bounding, compared to a simple probing in every explored outer box. A second experiment underlines that, in a large majority of the tested systems, the upper bounding is satisfactory since the upper bound converges faster than the lower bound towards the final value. Third, a qualitative study determines which of the two inner region extraction heuristics is the most useful in every system. A last study analyzes the size of the outer boxes in which the algorithms succeed in extracting an inner region.

References

[1] F. Benhamou and F. Goualard. Universally Quantified Interval Constraints. In Proc. CP, Constraint Programming, LNCS 1894, pages 67–82, 2004.

[2] F. Benhamou, F. Goualard, L. Granvilliers, and J.-F. Puget. Revising Hull and Box Consistency. In Proc. ICLP, pages 230–244, 1999.

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

[4] H. Collavizza, F. Delobel, and M. Rueher. Comparing Partial Consistencies. Reliable Comp., 5(3):213–228, 1999.

[5] F. Messine. Méthodes d'Optimisation Globale basées sur l'Analyse d'Intervalle pour la Résolution des Problèmes avec Contraintes. PhD thesis, LIMA-IRIT-ENSEEIHT-INPT, Toulouse, 1997.

[6] J. Ninin, F. Messine, and P. Hansen. A Reliable Affine Relaxation Method for Global Optimization. Technical Report RT-APO-10-05, IRIT, 2010.

[7] G. Trombettoni, I. Araya, B. Neveu, and G. Chabert. Inner Regions and Interval Linearizations for Global Optimization. In AAAI, pages 99–104, 2011.