Proceedings of GOW 2014, pp. 1 – 4.

Node Selection Heuristics Using the Upper Bound in Interval Branch and Bound*

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

[1]*LIGM, Université Paris Est, France,* Bertrand.Neveu@enpc.fr

[2]*LIRMM, Université Montpellier, France,* Gilles.Trombettoni@lirmm.fr

[3]*Pontificia Universidad Católica, Valparaiso, Chile,* rilianx@gmail.com

Abstract We present in this article a new strategy for selecting the current node in an interval Branch and Bound algorithm for constrained global optimization. The standard best-first strategy selects the node with the lowest lower bound of the objective estimate. We propose in this article new node selection policies where an *upper* bound of each node/box is also taken into account. The good accuracy of this upper bound achieved by several operators leads to a good performance of the criterion. These new strategies obtain better experimental results than classical best-first search on difficult instances.

Keywords: Global Optimization, Interval Branch and Bound, Node Selection

1. Introduction

The paper deals with continuous global optimization (nonlinear programming) deterministically handled by interval branch and bound (B&B). Several works have been performed for finding good branching heuristics [3], but very little work for the node selection itself. The solvers generally follow a best first search strategy (BFS), with some studies for limiting its exponential memory growth [1, 6]. Different variants of BFS have been proposed for discrete problems, such as K-Best-First Search [4].

To our knowledge, only Casado et al. in [1] and Csendes in [2] proposed node selection heuristics for interval B&Bs. One criterion to maximize, called C_3 in [5], and suitable for unconstrained global optimization is:

$$\frac{f^* - lb}{ub - lb}$$

where $[lb, ub] = [f]_N([x])$ is the interval obtained by the natural interval evaluation of the real-valued objective function f in the current box $[x]$: $[lb, ub]$ is a range interval including all real images of any point in $[x]$ by f. f^* is the optimum. When f^* is not known, $\widetilde{f}$, the cost of the best feasible point found so far can be used as an approximation of f^*. This criterion favors small boxes (i.e., a small interval width ($ub - lb$)) and nodes with good lb. For constrained optimization, another criterion (to maximize) called C_5 is equal to $C_3 \times fr$. It takes into account a feasibility ratio fr computed from all the inequality constraints. The criterion C_7 proposes to minimize $\frac{lb}{C_5}$.

This paper proposes two other policies taking into account an accurate upper bound of the optimum cost.

*Supported by the ANR project Ficolofo

2. Standard Interval B&B

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 a Cartesian product of intervals $[x_1] \times ... \times [x_i] \times ... \times [x_n]$.

The paper deals with continuous 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 domain/box $[x]$. x is said to be *feasible* if it satisfies the constraints.

A standard interval (or spatial) B&B scheme for continuous constrained global optimization (also known as nonlinear programming) is described in algorithms below. The B&B maintains two main types of information during search: $\widetilde{f}$, which is the cost of the best feasible point found so far, and f_{min} the minimum value of the lower bounds $lb([x])$ of the boxes/nodes $[x]$ to explore. In other terms, in every box $[x]$, there is a guarantee that no feasible point exists with a cost lower than $lb([x])$.

Let us first ignore the bold-faced instructions corresponding to the new strategy and detailed in Section 3. The algorithm is launched with the set g of constraints, the objective function f and with the initial box put into a list B of boxes to be handled. ϵ_{obj} is the required precision on the objective cost and is used as stopping criterion. We add a variable x_{obj} in the system (the vector x of variables) corresponding to the image (cost) of the objective function, and a constraint $f(x) = x_{obj}$. The criterion generally used in existing interval B&Bs is denoted

```
IntervalBranch&Bound(B, g, f) {
    While (B ≠ ∅ and f̃ − f_min > ε_obj) {
        criterion := criterionChoice(LBBox, UBBox, UBProb)
        [x] := bestBox (B, criterion); B := B \ [x]
        ([x]_1, [x]_2) := bisect ([x])
        [x]_1 := Contract&Bound([x]_1, g, f)
        [x]_2 := Contract&Bound([x]_2, g, f)
        B := B ∪ {[x]_1} ∪ {[x]_2}
        f_min := min_{[x]∈B} lb([x])
    }
}
```

```
Contract&Bound([x], g, f) {
    UBBox := f̃ − ε_obj + 0.1ε_obj
    g' := g ∪ {x_obj ≤ UBBox }
    [x] := contraction([x], g', f)
    if ([x] ≠ ∅) {
        (x_ub, cost):= FeasibleSearch([x],g')
        if (cost < f̃) {
            f̃ := cost
            ub([x]) := f̃ − ε_obj
        }
    }
    Return [x]
}
```

in this paper by LBBox. It consists in selecting a node/box $[x]$ with a minimum lower bound estimate of the objective function ($lb([x])$). The selected box $[x]$ is then split into two sub-boxes $[x]_1$ and $[x]_2$ along one dimension (selected by another and more studied branching heuristic). Both sub-boxes are then handled by the Contract&Bound procedure.

A constraint $x_{obj} \leq$ UBBox is first added to the system for decreasing the upperbound of the objective function in the box. This aims at finding a solution better than the current best feasible point. The procedure then *contracts* the handled box without loss of feasible part. In other words, some infeasible parts at the bounds of the domain are discarded by constraint programming (CP) and convexification algorithms. Since this contraction works on the extended box including the objective variable x_{obj}, it may improve both bounds of the objective on the current box.

The last part of the procedure carries out upperbounding. *FeasibleSearch* calls one or several heuristics searching for a feasible point that improves the best cost found so far.

Note that the search tree is traversed in best-first order so that an exponential memory may be required to store the nodes to handle.

3. New Strategies Using Upper Bounds

In optimization, the selection of the next node to expand is crucial for obtaining a good performance. The best node we can choose is such that it will improve the most the upperbound. Indeed, the upperbound improvement reduces *globally* the feasible space due to the constraint:

$x_{obj} \leq \widetilde{f}$. There exist two phases in a branch and bound: a phase where we try to find the optimal solution, and a second phase where we prove that this solution is optimal, which requires us expand all the remaining nodes. Therefore the node selection matters only in the first phase.

We define new strategies aggregating two criteria for selecting the current box:

1. LBBox: The well known criterion used by BFS and minimizing $lb([x])$ (for all boxes $[x]$ in the set B). This criterion is optimistic since we hope to find a solution with cost f_{min}, in which case the search would end. For each box, $lb([x])$ is computed by the Contract&Bound procedure and the computed value labels the node stored into the set B of boxes.
2. UBBox: This criterion selects the node having the smallest goal upperbound. Thus, if a feasible point was found inside this box, it would more likely improve the best cost found so far.

The UBBox criterion is symmetric to the LBBox one: for every box, $ub([x])$ is computed by the Contract&Bound procedure and labels the node before storing it in the set B of boxes. In particular, constraint programming techniques like 3BCID [7] can improve $ub([x])$ by discarding small slices at the upper bound of $[x_{obj}]$ (shaving process).

We think that a key of success of the UBBox criterion is that it evaluates more accurately the objective upperbound than the natural interval evaluation would do (i.e., ub computed by $[lb, ub] := [f]_N([x])$).

We propose two main ways to aggregate these two criteria.

LB+UBBox. This strategy selects the node $[x]$ with the lowest value of the sum $lb([x]) + ub([x])$. This corresponds to minimizing both criteria with the same weight, i.e. minimizing the middle of the interval of the objective estimate in the box.

Alternating both criteria. In this second strategy, the next box to handle is chosen using *one* of the two criteria. A random choice is made by the $criterionChoice$ function at each node selection, with a probability UBProb of choosing UBBox. If UBBox (resp. LBBox) is chosen and several nodes have the same cost $ub([x])$ (resp. $lb([x])$), then we use the other criterion LBBox (resp. UBBox) to tie breaks.

Experiments showed that the performance is not sensitive to a fine tuning of the UBProb parameter provided it remains between 0.2 and 0.8, so that the parameter has been fixed to 0.5. The experiments in Section 5 highlight the positive impact of this criterion on performance.

These results suggest that it is important to invest both in intensification (UBBox) and diversification (LBBox). In other words, the use of a second criterion allows the search to avoid the drawback of using one criterion alone, i.e. (for LBBox) choosing promising boxes with no feasible point and (for UBBox) going deeply in the search tree where only slightly better solutions will be found trapped inside a local minimum.

3.1 Details on the criterion UBBox

All candidate boxes in the set B, have a UB label depending on the best cost found so far ($\widetilde{f}$) when the boxes were handled by Contract&Bound. All labels fall in four main cost ranges categories given by $\widetilde{f}$ and explaining with which priority the boxes are chosen using the UBBox criterion. The label is:

1. lower than $\widetilde{f} - \epsilon_{obj}$ if the contraction procedure reduced the maximum estimate of the objective in the box,
2. equal to $\widetilde{f} - \epsilon_{obj}$, if the box is a descendant of the box containing the current best feasible point $\widetilde{f}$,
3. equal to $\widetilde{f} - 0.9\,\epsilon_{obj}$ if the box was handled after the last update of best cost,
4. greater than $\widetilde{f} - 0.9\,\epsilon_{obj}$ in the remaining case.

As shown in the Contract&Bound pseudocode, the additional term $0.1\,\epsilon_{obj}$ allows penalizing the boxes that are not issued by a bisection from boxes where the current best feasible point was found.

4. Implementation of the Set B of Boxes

The set B was initially implemented by a heap structure ordered on the LBBox criterion. The implementation `s1-05` keeps this unique data structure for taking into account the two criteria in the randomized strategy, but the node selection using UBBox comes at a linear cost in the number N of nodes. In practice, this takes about 10% of the total cost when N exceeds 50,000. We have then built a variant `s1-05-01` changing dynamically the probability UBProb in the following way: UBProb $= 0.5$ if $N \leq 50,000$ and UBProb $= 0.1$ if $N > 50,000$.

Finally, we tried a cleaner implementation, `s2-nodiv`, with two heaps, one for each criterion. All the operations are then in $log_2(N)$, except for the heap filtering process launched occasionnally during search. (This "garbage collector" is performed in all our implementations and removes from B all the nodes with $lb([x]) > \widetilde{f}$.)

The `s2-nodiv` implementation brings less diversification than the first one which reconstructs the heap at each criterion change. Indeed, there often exist several nodes with the same UBBox and LBBox values, so it is useful to periodically rebuild the data structures randomly for breaking the ties. So we propose variants that use a second parameter corresponding to a diversification period. We obtained good results by fixing it to 50 in the `s2-50` variant or 100 in the `s2-100` variant, the first parameter UBProb being still fixed to 0.5. To be fair, we also applied the first strategy with the minUB+LB aggregative criterion running heap filtering every 100 nodes (`s0-100`).

5. Experiments

We have run experiments on 82 problems, issued from the series 1 and 2 of the Coconut benchmark. The best strategies `s2-50` and `s2-100` obtain a gain of about 40% w.r.t. the total time and 23% on average, meaning that greater gains are obtained on difficult problems.

6. Summary

The node selection policy is a promising line of research to improve performance of interval B&B. We have obtained good results by taking into account for each node a lower bound but also an upper bound, provided that this upper bound is made accurate by box contraction operations, by a random selection between both criteria and by a work on heap data structures. In a short term, we are going to investigate how relevant components of criteria proposed by Markot and al. in [5] can improve the current policies, especially the feasibility ratio.

References

[1] L.G. Casado, J.A. Martinez, and I. Garcia. Experiments with a New Selection Criterion in a Fast Interval Optimization Algorithm. *Journal of Global Optimization*, 19:247–264, 2001.

[2] T. Csendes. New Subinterval Selection Criteria for Interval Global Optimization. *Journal of Global Optimization*, 19:307–327, 2001.

[3] T. Csendes and D. Ratz. Subdivision Direction Selection in Interval Methods for Global Optimization. *SIAM Journal on Numerical Analysis*, 34(3), 1997.

[4] A. Felner, S. Kraus, and R. E. Korf. KBFS: K-Best-First Search. *Annals of Mathematics and Artificial Intelligence*, 39, 2003.

[5] M.C. Markot, J. Fernandez, L.G. Casado, and T. Csendes. New Interval Methods for Constrained Global Optimization. *Mathematical Programming*, 106:287–318, 2006.

[6] J. Ninin and F. Messine. A Metaheuristic Methodology Based on the Limitation of the Memory of Interval Branch and Bound Algorithms. *Journal of Global Optimization*, 50:629–644, 2011.

[7] G. Trombettoni and G. Chabert. Constructive Interval Disjunction. In *Proc. CP*, volume 4741 of *LNCS*, pages 635–650. Springer, 2007.