2008 International Symposium on Nonlinear Theory and its Applications
NOLTA'08, Budapest, Hungary, September 7-10, 2008

Capabilities of Constraint Programming in Rigorous Global Optimization

Michel Rueher†
Alexandre Goldsztejn✠, Yahia Lebbah‡,† and Claude Michel†

†Université de Nice Sophia Antipolis, CNRS, 06903 Sophia Antipolis, France,
✠CNRS, Université de Nantes, 44322 Nantes, France
‡Université d'Oran Es-Senia B.P. 1524 EL-M'Naouar, 31000 Oran, Algeria
Email: michel.rueher@gmail.com, alexandre.goldsztejn@univ-nantes.fr,ylebbah@gmail.com,cpjm@polytech.unice.fr

Abstract—We investigate the capabilities of constraints programming techniques to boost rigorous global optimization methods, and thus, to reduce the gap between efficient but unsafe systems like Baron[1], and slow but safe global optimization approaches. We show how constraint programming filtering techniques can be used to implement optimality-based reduction in a safe and efficient way, and thus to take advantage of the known bounds of the objective function to reduce the domain of the variables, and to speed up the search of a global optimum. We describe an efficient strategy to compute very accurate approximations of feasible points. This strategy takes advantage of the Newton method for under-constrained systems of equations and inequalities to compute efficiently a promising upper bound. Experiments on the COCONUT benchmarks demonstrate that these different techniques drastically improve the performances.

1. Introduction

We consider here the global optimization problem $\mathcal{P}$ to minimize an objective function under nonlinear equalities and inequalities,

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) = 0,\ i \in \{1,..,k\} \\ & h_j(x) \leq 0,\ j \in \{1,..,m\} \end{array} \quad (1)$$

with $x \in \mathbf{x}$, $f : \boldsymbol{R}^n \rightarrow \boldsymbol{R}$, $g_i : \boldsymbol{R}^n \rightarrow \boldsymbol{R}$ and $h_j : \boldsymbol{R}^n \rightarrow \boldsymbol{R}$; Functions f , g_i and h_j are nonlinear and continuously differentiable on some vector $\mathbf{x}$ of intervals of $\boldsymbol{R}$. For convenience, in the sequel, $g(x)$ (resp. $h(x)$) will denote the vector of $g_i(x)$ (resp. $h_j(x)$) functions.

The difficulties in such global optimization problems come mainly from the fact that many local minimizers may exist but only few of them are global minimizers [7].

Optimality-based reduction (OBR) has been introduced by Ryoo and Sahinidis in [8] to take advantage of the known bound of the objective function to reduce the size of the domains of the variables. This technique uses a well known property of the saddle point to compute new bounds for the domain of a variable taking into account the known bounds of the objective function. However, the basic OBR algorithm is unsafe[2]

We have show in [10] that constraint programming techniques can be used in a simple and elegant way to safely refute the potential non-solution boxes identified by the OBR method. Roughly speaking, filtering techniques are used to reduce these boxes to empty boxes, and thus, to prove that they do not contain any feasible point. These constraint programming techniques do not suffer from the same limitations as Kearfott's method. The first experiments show that they are also much more efficient.

In global optimization problems the feasible region may be disconnected. Thus, finding feasible points is a critical issue in safe Branch and Bound algorithms for continuous global optimization. Standard strategies use local search techniques to provide a reasonable approximation of an upper bound and try to prove that a feasible solution actually exists within the box around the guessed global optimum. Practically, finding a guessed point for which the proof succeeds is often a very costly process.

In [1] we have introduced a new strategy to compute very accurate approximations of feasible points. This strategy takes advantage of the Newton method for under-constrained systems of equations and inequalities. More precisely, this procedure exploits the optimal solution of a linear relaxation of the problem to compute efficiently a promising upper bound. Experiments on the COCONUT benchmarks demonstrate that the combination of this procedure with a safe branch and bound algorithm drastically improves the performances.

The rest of this paper is organized as follows. The first section provides the overall schema of a safe branch and bound process for global optimization. The next section describes the OBR method and introduces our safe implementation based on constraint techniques. Next, we describe our new strategy to compute very accurate approximations of feasible points

We do not recall here the capabilities of consistency techniques to speed up the initial convergence of the interval narrowing algorithms. Neither do we show how linear relaxations can be used in such a CP framework to rigor-

[1] See http://www.andrew.cmu.edu/user/ns1b/baron/baron.html

[2] Kearfott [3, 2] has proposed a safe implementation of OBR which is based on a valid bounding of the dual solution but this method suffers from strong limitations and is rather slow.

Algorithm 1 Branch and Bound Algorithm

Function BB(IN $\mathbf{x}$, ϵ; OUT $\mathcal{S}$, $[L, U]$)

% $\mathcal{S}$: set of proven feasible points
% $\mathbf{f}_{\mathbf{x}}$ denotes the set of possible values for f in $\mathbf{x}$
% *nbStarts*: number of starting points in the first upper-bounding
$\mathcal{L} \leftarrow \{\mathbf{x}\}$; $(L, U) \leftarrow (-\infty, +\infty)$;
$\mathcal{S} \leftarrow UpperBounding(\mathbf{x}', nbStarts)$;
while $w([L, U]) > \epsilon$ **do**
  $\mathbf{x}' \leftarrow \mathbf{x}''$ such that $\underline{\mathbf{f}}_{\mathbf{x}''} = min\{\underline{\mathbf{f}}_{\mathbf{x}''} : \mathbf{x}'' \in \mathcal{L}\}$; $\mathcal{L} \leftarrow \mathcal{L} \setminus \{\mathbf{x}\}'$;
  $\overline{\mathbf{f}}_{\mathbf{x}'} \leftarrow min(\overline{\mathbf{f}}_{\mathbf{x}'}, U)$;
  $\mathbf{x}' \leftarrow Prune(\mathbf{x}')$; $\underline{\mathbf{f}}_{\mathbf{x}'} \leftarrow LowerBound(\mathbf{x}')$;
  $\mathcal{S} \leftarrow \mathcal{S} \cup UpperBounding(\mathbf{x}', 1)$;
  if $\mathbf{x}' \neq \emptyset$ **then** $(\mathbf{x}'_1, \mathbf{x}'_2) \leftarrow Split(\mathbf{x}')$; $\mathcal{L} \leftarrow \mathcal{L} \cup \{\mathbf{x}'_1, \mathbf{x}'_2\}$;
  if $\mathcal{L} = \emptyset$ **then** $(L, U) \leftarrow (+\infty, -\infty)$
    else $(L, U) \leftarrow (min\{\underline{\mathbf{f}}_{\mathbf{x}''} : \mathbf{x}'' \in \mathcal{L}\}, min\{\overline{\mathbf{f}}_{\mathbf{x}''} : \mathbf{x}'' \in \mathcal{S}\})$
endwhile

ously bound the global optima as well as its location. A detailed discussion of these concepts and techniques can be found in [4, 5, 9].

2. The Branch and Bound schema

The algorithms (see Algorithm 1) we describe here are derived from the well known Branch and Bound schema introduced by Horst and Tuy for finding a global minimizer. Interval analysis techniques are used to ensure rigorous and safe computations whereas constraint programming techniques are used to improve the reduction of the feasible space.

Algorithm 1 computes enclosers for minimizers and safe bounds of the global minimum value within an initial box $\mathbf{x}$. Algorithm 1 maintains two lists : a list $\mathcal{L}$ of boxes to be processed and a list $\mathcal{S}$ of proven feasible boxes. It provides a rigorous encloser $[L, U]$ of the global optimum with respect to a tolerance ϵ.

Algorithm 1 starts with $UpperBounding(\mathbf{x}, nbStarts)$ which computes a set of feasible boxes by calling a local search with *nbStarts* starting points and a proof procedure.

The box around the local solution is added to $\mathcal{S}$ if it is proved to contain a feasible point. At this stage, if the box $\mathbf{x}'$ is empty then, either it does not contain any feasible point or its lower bound $\underline{\mathbf{f}}_{\mathbf{x}'}$ is greater than the current upper bound U. If $\mathbf{x}'$ is not empty, the box is split along one of the variables[3] of the problem.

In the main loop, Algorithm 1 selects the box with the lowest lower bound of the objective function. The *Prune* function applies filtering techniques to reduce the size of the box $\mathbf{x}'$. In the framework we have implemented, *Prune* just uses a 2B-filtering algorithm [6]. Then, $LowerBound(\mathbf{x}')$ computes a rigorous lower bound $\underline{\mathbf{f}}_{\mathbf{x}'}$ using a linear programming relaxation of the initial problem.

[3] Various heuristics are used to select the variable the domain of which has to be split.

Actually, function *LowerBound* is based on the linearization techniques of the Quad-framework [5]. *LowerBound* computes a safe minimizer $\underline{\mathbf{f}}_{\mathbf{x}'}$ thanks to the techniques introduced by Neumaier et al.

Algorithm 1 maintains the lowest lower bound L of the remaining boxes $\mathcal{L}$ and the lowest upper bound U of proven feasible boxes. The algorithm terminates when the space between U and L becomes smaller than the given tolerance ϵ.

The next section is devoted to OBR techniques. We first recall the basic definitions, then we describe the method proposed by Kearfott, and finally we introduce our safe algorithm for computing OBR.

3. Optimality-based reduction

3.1. Basics of optimality-based reduction

Optimality-based reduction has been introduced by Ryoo and Sahinidis in [8]. It takes advantage of a property of the saddle point to reduce the domains of the variables of the problem to optimize. Optimality-based reduction relies on the following two theorems to improve the bounds of the domain of one variable:

Theorem 1 *Let U be a known upper bound of the original problem P, let L be a known lower bound of a convex relaxation R of P, and assume that the constraint $x_i - \overline{x}_i \leq 0$ is active at the optimal solution of R and has a corresponding multiplier $\lambda_i^* > 0$. Then*

$$x_i \geq x'_i \text{ with } x'_i = \overline{x}_i - \frac{U - L}{\lambda_i^*}. \tag{2}$$

Thus, if $x'_i > \underline{x}_i$, the domain of x_i can be set to $[x'_i, \overline{x}_i]$ without loss of any global optima.

λ_i^* denotes the dual solution of R. Roughly speaking, a constraint $A_i x_i \leq b_i$ is active if $A_i x_i = b_i$. This equality may be difficult to be checked over the floating-point numbers.

Theorem 2 *Let U be a known upper bound of the original problem P, let L be a known lower bound of a convex relaxation R of P, and assume that the constraint $\underline{x}_i - x_i \leq 0$ is active at the optimal solution of R and has a corresponding multiplier $\lambda_i^* > 0$. Then*

$$x_i \leq x''_i \text{ with } x''_i = \underline{x}_i + \frac{U - L}{\lambda_i^*}. \tag{3}$$

Thus, if $x''_i < \overline{x}_i$, the domain of x_i can be set to $[\underline{x}_i, x''_i]$ without loss of any global optima.

The first theorem provides a test to improve the lower bound of the domain of a variable while the second theorem provides a test to improve the upper bound of the domain of a variable.

Moreover, these valid inequalities have been generalized to the other constraints. The following theorem is the most general one :

Theorem 3 *Let U be a known upper bound of the original problem P, let L be a known lower bound of a convex relaxation R of P, and assume that the constraint $g_i(x) \leq 0$ is active at the optimal solution of R and has a corresponding multiplier $\lambda_i^* > 0$. Then*

$$g_i(x) \geq -\frac{U-L}{\lambda_i^*}. \quad (4)$$

This last theorem enables to enforce some constraints, and thus to reduce the domains of the variables.

All these theorems are more detailed and proved in [8].

3.2. A safe implementation of OBR based on constraint filtering techniques

As said before, the critical issue in the OBR method comes from the unsafe dual solution provided by the simplex algorithm. In other words, due to the rounding errors, we may lose the global optima when we use formula (2) to shrink the domain of some variable x_i.

The essential observation is that we can use filtering techniques to prove that no feasible point exists when the domain of x_i is reduced to $[\underline{x_i}, x_i']$. Indeed, if the constraint system

$$\begin{array}{l} f(x) \leq U \\ g_i(x) = 0, \ i = 1..k \\ g_j(x) \leq 0, \ j = k+1..m \end{array} \quad (5)$$

does not have any solution when the domain of x is set to $[\underline{x_i}, x_i']$, then the domain reduction computed by the OBR method is valid; if the filtering cannot prove that no solution exits inside the considered box, we have just to add this box to $\mathcal{L}$, the list of boxes to be processed (See algorithms 1 and 2).

The same reasoning holds for the reduction of the domain of $\mathbf{f}$, i.e., when algorithm 2 attempts to reduce the lower bound of the variables of the problem by means of formula (3).

Algorithm 2 details the new process of the computation of the lower bound. Note that Algorithm 1 remains almost unchanged : we have just to replace the call $\underline{\mathbf{f}}_{\mathbf{x}'} \leftarrow$ *LowerBound*($\mathbf{x}'$) by ($\underline{\mathbf{f}}_{\mathbf{x}'}, x', \mathcal{L}$)$\leftarrow$*LowerBound*($\mathbf{x}', L, U, \mathcal{L}$).

The constraint-based approach introduced here is about five time faster than Kearfott's approach. In fact, our approach introduces a negligible overhead since the proof process mostly relies on a $2B$-consistency which is an effective technique here[4].

The Upper-bounding step (see Algorithm 3) performs a multistart strategy where a set of *nbStarts* starting points are provided to a local optimization solver. The solutions computed by the local solver are then given to a function *InflateAndProve* which uses an existence proof procedure based on the Borsuk test. However, the proof procedure may fail to prove the existence of a feasible point within box $\mathbf{x}_p$. The most common source of failure is that the "guess" provided by the local search lies too far from the feasible region.

[4]That is why the more costly *Quad*-filtering is almost never used in these examples.

Algorithm 2 Computation of a safe lower bound with OBR

Function `LowerBound(IN` $\mathbf{x}, L, U, \mathcal{L}$; `OUT (`$\underline{\mathbf{f}}_{\mathbf{x}'}, \mathcal{L}$`) )`

$\mathcal{L}_r \leftarrow \emptyset$ % $\mathcal{L}_r$: set of potential non-solution boxes
Compute $\underline{\mathbf{f}}$ with `Quad` in $\mathbf{x}$
for each variable $\mathbf{x}$ **do**
  Apply formula 2 of OBR
    and add the generated potential non-solution boxes to $\mathcal{L}_r$
for each box $\mathbf{B_i}$ in $\mathcal{L}_r$ **do**
  $\mathbf{B_i'} \leftarrow 2B$-filtering($\mathbf{B_i}$)
  if $\mathbf{B_i'} = \emptyset$ **then** reduce the domain of x_i
  else $\mathbf{B_i''} \leftarrow$ `Quad`-filtering($\mathbf{B_i'}$)
    if $\mathbf{B_i''} = \emptyset$ **then** reduce the domain of x_i
      else add $\mathbf{B_i''}$ to $\mathcal{L}$; **endif**
  endif
Apply formula (3) of OBR to reduce the lower bound of the variables
Use $2B$-filtering and `Quad`-filtering to validate the reduction

4. A new upper bounding strategy

The standard upper bounding procedure relies on a local search to provide a "guessed" feasible point lying in the neighborhood of a local optima. However, the effects of floating point computation on the provided local optima are hard to predict. As a result, the local optima might lie outside the feasible region and the proof procedure might fail to build a proven box around this point.

We propose here a new upper bounding strategy which attempts to take advantage of the solution of a linear outer approximation of the problem. The lower bound process uses such an approximation to compute a safe lower bound of $\mathcal{P}$. When the LP is solved, a solution x_{LP} is always computed and, thus, available for free. This solution being an optimal solution of an outer approximation of $\mathcal{P}$, it lies outside the feasible region. Thus, x_{LP} is not a feasible point. Nevertheless, x_{LP} may be a good starting point to consider for the following reasons:

- At each iteration, the branch and bound process splits the domain of the variables. The smaller the box is, the nearest x_{LP} is from the actual optima of $\mathcal{P}$.
- The proof process inflates a box around the initial guess. This process may compensate the effect of the distance of x_{LP} from the feasible region.

However, while x_{LP} converges to a feasible point, the process might be quite slow. To speed up the upper bounding process, we have introduced a light weight, though efficient, procedure which compute a feasible point from a point lying in the neighborhood of the feasible region. This procedure which is called *FeasibilityCorrection* will be detailed in the next subsection.

Algorithm 3 Upper bounding build from the LP optimal solution x^*_{LP}

Function `UpperBounding(IN` $\mathbf{x}$, x^*_{LP}, $nbStarts$; `OUT` $\mathcal{S}'$`)`

% $\mathcal{S}'$: list of proven feasible boxes;
% $nbStarts$: number of starting points
% x^*_{LP}: the optimal solution of the LP relaxation of $\mathcal{P}(\mathbf{x})$
$\mathcal{S}' \leftarrow \emptyset$; $x^*_{corr} \leftarrow$FeasibilityCorrection(x^*_{LP});
$\mathbf{x}_p \leftarrow$InflateAndProve(x^*_{corr}, $\mathbf{x}$);
if $\mathbf{x}_p \neq \emptyset$ **then** $\mathcal{S}' \leftarrow \mathcal{S}' \cup \mathbf{x}_p$
return $\mathcal{S}'$

Algorithm 3 describes how an upper bound may be build from the solution of the linear problem used in the lower bounding procedure.

4.1. Computing pseudo-feasible points

This section introduces an adaptation of the Newton method to under-constrained systems of equations and inequalities which provides very accurate approximations of feasible points at a low computational cost. When the system of equations $g(x) = 0$ is under-constrained there is a manifold of solutions. $l(x)$, the linear approximation[5] around $x^{(0)}$ is still valid in this situation, but the linear system of equations $l(x) = 0$ is now under-constrained, and has therefore an affine space of solutions. So we have to choose a solution $x^{(1)}$ of the linearized equation $l(x) = 0$ among the affine space of solutions. As $x^{(0)}$ is supposed to be an approximate solution of $g(x) = 0$, the best choice is certainly the solution of $l(x) = 0$ which is the closest to $x^{(0)}$. This solution can easily be computed with the Moore-Penrose inverse: $x^{(1)} = x^{(0)} - A_g^+(x^{(0)})g(x^{(0)})$, where $A_g^+ \in \boldsymbol{R}^{n\times m}$ is the Moore-Penrose inverse of $A_g \in \boldsymbol{R}^{m\times n}$, the solution of the linearized equation which minimizes $\|x^{(1)} - x^{(0)}\|$. Applying previous relation recursively leads to a sequence of vectors which converges to a solution close to the initial approximation, provided that this latter is accurate enough.

The Moore-Penrose inverse can be computed in several ways: a singular value decomposition can be used, or in the case where A_g has full row rank (which is the case for $A_g(x^{(0)})$ if $x^{(0)}$ is non-singular) the Moore-Penrose inverse can be computed using $A_g^+ = A_g^T(A_g A_g^T)^{-1}$.

Inequality constraints are changed to equalities by introducing slack variables: $h_j(x) \leq 0 \iff h_j(x) = -s_i^2$. So the Newton method for under-constrained systems of equations can be applied.

5. Conclusion

Constraint programming filtering techniques can be used to implement optimality-based reduction in a safe and efficient way. Thanks to constraint programming, the branch and bound algorithm can take advantage of the OBR through a simple but efficient refutation process. Preliminary experiments have shown that our procedure compares well to the Kearfott's procedure. Using constraint-based refutation, OBR is up to five times faster than with Kearfott's procedure. As a result, constraint-based OBR can significantly improve the branch and bound process.

Constraint-based refutation has allowed a safe embedding of the OBR in a very simple way. This approach seems to be general enough to be applied to other unsafe methods. Our next work is thus to test this approach on other unsafe methods in order to improve the branch and bound process.

To evaluate our new upper bounding strategy we have performed experiments on a significant set of benchmarks[6]. All the benchmarks come from the collection of benchmarks of the COCONUT project. We have selected 35 benchmarks where Icos succeeds to find the global minimum while relying on an unsafe local search.

We did compare our new upper bounding strategy with various upper bounding strategies. Our new upper bounding strategy is the best strategy: 31 benchmarks are now solved within the 30s time out; moreover, almost all benchmarks are solved in much less time and with a greater amount of proven solutions. This new strategy improves drastically the performance of the upper bounding procedure and competes well with a local search.

References

[1] C. M. Alexandre Goldsztejn, Yahia Lebbah and M. Rueher. Revisiting the upper bounding process in a safe branch and bound algorithm. In S. Verlag, editor, *Proc. CP2008,http://www.cs.mu.oz.au/cp2008/*, 2008.

[2] R. B. Kearfott. Validated probing with linear relaxations. *submitted to Journal of Global Optimization*, 2005.

[3] R. B. Kearfott. Discussion and empirical comparisons of linear relaxations and alternate techniques in validated deterministic global optimization. *journal of Optimization Methods and Software*, pages 715–731, Oct. 2006.

[4] Y. Lebbah and O. Lhomme. Accelerating filtering techniques for numeric CSPs. *Artificial Intelligence*, 139(1):109–132, 2002.

[5] Y. Lebbah, C. Michel, M. Rueher, D. Daney, and J.-P. Merlet. Efficient and safe global constraints for handling numerical constraint systems. *SIAM Journal on Numerical Analysis*, 42(5):2076–2097, 2004.

[6] O. Lhomme. Consistency techniques for numeric CSPs. In *Proceedings of IJCAI'93*, pages 232–238, Chambéry(France), 1993.

[7] A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 2004.

[8] H. S. Ryoo and N. V. Sahinidis. A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, pages 107–138, 1996.

[9] P. Van-Hentenryck, D. Mc Allester, and D. Kapur. Solving polynomial systems using branch and prune approach. *SIAM Journal on Numerical Analysis*, pages 34(2):797–827, 1997.

[10] C. M. Yahia Lebbah and M. Rueher. Using constraint techniques for a safe and fast implementation of optimality-based reduction. In ACM, editor, *Proceedings of SAC'07*, pages 326 – 331, 2007.

[5] $l(x) := g(x^{(0)}) + J_g(x^{(0)}) \cdot (x - x^{(0)})$ where J_g is the Jacobian matrix of g, i.e. $J_{g_{ij}} = \partial g_i / \partial x_j$.

[6] Detailled results can be found in http://www.i3s.unice.fr/%7Emh/RR/2008/RR-08.11-A.GOLDSZTEJN.1.pdf