LABORATOIRE

INFORMATIQUE, SIGNAUX ET SYSTÈMES
DE SOPHIA ANTIPOLIS
UMR 6070

REVISITING THE UPPER BOUNDING PROCESS IN A SAFE BRANCH AND BOUND ALGORITHM

Alexandre GOLDSZTEJN, Yahia LEBBAH, Claude MICHEL, Michel RUEHER

Projet CEP

Rapport de recherche
ISRN I3S/RR–2008-11–FR

Juin 2008

Laboratoire d'Informatique de Signaux et Systèmes de Sophia Antipolis - UNSA-CNRS
2000, rte.des Lucioles – Les Algorithmes – Bât Euclide B – B.P. 121 – 06903 Sophia-Antipolis Cedex – France
Tél.: 33 (0)4 92 94 27 01 – Fax: 33 (0)4 92 94 28 98 – www.i3s.unice.fr
UMR6070

Revisiting the upper bounding process in a safe Branch and Bound algorithm

Alexandre Goldsztejn[1], Yahia Lebbah[2,3], Claude Michel[3], and Michel Rueher[3]

[1] CNRS / Université de Nantes
2, rue de la Houssinière
44322 Nantes, France
alexandre.goldsztejn@univ-nantes.fr
[2] Département Informatique
Université d'Oran Es-Senia
B.P. 1524 EL-M'Naouar, 31000 Oran, Algeria
ylebbah@gmail.com
[3] I3S (CNRS/UNSA)
930, Route des Colles - BP 145
06903 Sophia Antipolis Cedex
{cpjm, rueher}@polytech.unice.fr

Abstract. Finding feasible points is a critical issue in continuous global optimization problems. Safe Branch and Bound algorithms 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 this paper, we introduce 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.
First experiments on the Coconuts benchmarks demonstrate that the combination of this procedure with a safe Branch and Bound algorithm drastically improves the performances.

1 Introduction

Global optimization is a critical issue in numerous applications of AI ranging from robot arm design [8], safety verification problems [1] to chemical equilibrium problems [3] and scheduling [1]. Many famous hard optimization problems, such as the traveling salesman problem or the protein folding problem, are global optimization problems. We consider here the global optimization problem $\mathcal{P}$ to minimize an objective function under nonlinear Equalities and inequalities,

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

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

Optimization problems are a challenge for CP in finite domains, they are also a big challenge for CP on continuous domains. The point is that CP solvers are much more slower than classical (non-safe) mathematical methods on non-linear constraint problems as soon as we consider optimization problems. The techniques introduced in this paper try to boost constraints techniques on these problems and thus, to reduce the gap between efficient but unsafe systems like BARON[4], and the slow but safe constraint based approaches.

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 [10]. Moreover, the feasible region may be disconnected.

Finding feasible points is a critical issue in safe Branch and Bound algorithms for continuous global optimization. Indeed, a good upper bound allows us to prune the search tree and to speed up the convergence to the global optima. 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 this paper, we introduce 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.

First experiments on the Coconuts[5] benchmarks demonstrate that the combination of this procedure with a safe Branch and Bound algorithm drastically improves the performances.

The paper is organized as follows. Section 2 recalls the general Branch and Bound schema and introduces the classical upper bounding function. Our new upper bounding procedure is introduced in Section 3. The computation of pseudo feasible points is detailed in Section 4. Experiment results on different strategies are discussed in Section 5.

2 The Branch and Bound schema

The algorithm we describe here is derived from the well known Branch and Bound schema introduced by Horst and Tuy[6] for finding a global minimizer. We use interval analysis techniques to ensure rigorous and safe computations and constraint programming techniques to improve the reduction of the feasible space.

[4] See http://www.andrew.cmu.edu/user/ns1b/baron/baron.html
[5] See http://www.mat.univie.ac.at/~neum/glopt/coconut/Benchmark/Benchmark.html.

Algorithm 1 Branch and Bound algorithm

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

% $\mathcal{S}$: set of proven feasible points
% $\mathbf{f_x}$ 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 $\underline{\mathbf{f}}_{\mathbf{x}''} = min\{\underline{\mathbf{f}}_{\mathbf{x}''} : \mathbf{x}'' \in \mathcal{L}\}$
  $\mathcal{L} \leftarrow \mathcal{L} \backslash \mathbf{x}'$
  $\overline{\mathbf{f}}_{\mathbf{x}'} \leftarrow min(\overline{\mathbf{f}}_{\mathbf{x}'}, U)$
  $\mathbf{x}' \leftarrow Prune(\mathbf{x}')$
  $\underline{\mathbf{f}}_{\mathbf{x}'} \leftarrow LowerBound(\mathbf{x}')$
  $\mathcal{S} \leftarrow \mathcal{S} \cup UpperBounding(\mathbf{x}', 1)$
  if $\mathbf{x}' \neq \emptyset$ **then**
    $(\mathbf{x}'_1, \mathbf{x}'_2) \leftarrow Split(\mathbf{x}')$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{\mathbf{x}'_1, \mathbf{x}'_2\}$
  endif
  if $\mathcal{L} = \emptyset$ **then**
    $(L,U) \leftarrow (+\infty, -\infty)$
  else
    $(L,U) \leftarrow (min\{\underline{\mathbf{f}}_{\mathbf{x}''} : \mathbf{x}'' \in \mathcal{L}\}, min\{\overline{\mathbf{f}}_{\mathbf{x}''} : \mathbf{x}'' \in \mathcal{S}\})$
  endif
endwhile

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 given 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. This function is detailed in Section 2.1.

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 [9]. Then, $LowerBound(\mathbf{x}')$ computes a rigorous lower bound $\underline{\mathbf{f}}_{\mathbf{x}'}$ using a linear programming relaxation of the initial problem. Actually, function *LowerBound* is based on the linearization techniques of the `Quad`-framework [7]. *LowerBound* computes a safe minimizer $\underline{\mathbf{f}}_{\mathbf{x}'}$ thanks to the techniques introduced in [11].

Function $UpperBounding(\mathbf{x}', 1)$ calls the local solver only once. 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. In both cases, we say that the box is fathomed. If $\mathbf{x}'$ is not empty, the box is split along one of the problem variables[6]. 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. Of course a proven optimum cannot always be found, and thus, algorithm 1 has to be stopped in some cases to get the feasible boxes which may have been found.

2.1 Upper-bounding

The Upper-bounding step (see Algorithm 2) performs a multistart strategy where a set of $nbStarts$ starting points are provided to a local optimization solver. The first initial point is the midpoint of the box, and the next ones are generated randomly.

The solutions computed by the local solver are then given to the function $InflateAndProve$ which tries to identify within $\mathbf{x}$ a box $\mathbf{x}_p$ containing proven feasible points.

Procedure $Prove$ applies an existence proof procedure based on the Borsuk test, a proof procedure similar to the one of Krawczyck and Miranda but more efficient [4]. We rely also on the techniques introduced by Hansen in [5] to handle under-determined systems.

Each time $InflateAndProve$ succeeds and returns a box $\mathbf{x}_p$ which is proved to contain a feasible point, a new upper bound can be computed for box $\mathbf{x}$. This computation is achieved using a simple interval evaluation of the objective function f over box $\mathbf{x}_p$. Of course, the upper bound linked to $\mathbf{x}$ is updated only if the newly computed upper bound is better than the older one[7].

Procedure $Prove$ may fail to prove the existence of a feasible point within box $\mathbf{x}_p$. Though difficulties like a singularity might prevent the proof mechanism to work correctly, a more common source of failure is that the "guess" provided by the local search lies too far from the feasible region.

In the next section, we introduce a new upper bounding strategy which takes advantage of the Newton method for under-constrained systems of equations and inequalities. More precisely, this strategy exploits the optimal solution of a linear relaxation of the problem to compute efficiently and accurately a promising upper bound.

3 A new upper bounding strategy

Finding feasible points, i.e. points x that satisfy $g(x) = 0$ and $h(x) \leq 0$, is a critical issue of Branch and Bound algorithms. As said before, this is usually

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

[7] Note that the initial upper bound is computed by means of an interval evaluation of the objective function f over the whole box $\mathbf{x}$.

Algorithm 2 Upper bounding

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

```
% S': list of proven feasible boxes
% nbStarts: number of starting points
S' ← ∅
for i←1 to nbStarts do
    x_0 ←Select a starting point in x
    x* ←LS(x_0)   % x* is a guess point computed by the local solver LS
    x_p ←InflateAndProve(x*, x)
    if x_p ≠ ∅ then
        S' ←S' ∪ x_p
    endif
endfor
return S'
```

Function `InflateAndProve`(`IN` x^*, $\mathbf{x}$; `OUT` $\mathbf{x}_p$)

```
% x*: a guessed optimun within x
ε ← 10^-9
x'_p ←∅
while ε ≤ 1.0 do
    x_p ←[x* − ε, x* + ε] ∩ x   % x_p is a box of size 2ε around x*
    if (x_p ⊆ x'_p) then
        return ∅
    else
        x'_p ←x_p
    endif
    if Prove(x_p) then
        return x_p
    endif
    ε ← 10 * ε
endwhile
return ∅
```

done by inflating a box around a "guess point" (estimated feasible point) and by applying an existence theorem to prove that there exists a feasible point inside this inflated box. Therefore, it is crucial to be able to obtain some guess points of good quality.

The upper bounding procedure described in the previous section relies on a local search to provide a "guessed" feasible point lying in the neighborhood of a local optima. However, the behavior of the local search is not always clear and 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.

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}) % Improving x^*_{LP} feasibility
$\mathbf{x}_p \leftarrow$InflateAndProve(x^*_{corr}, $\mathbf{x}$)
if $\mathbf{x}_p \neq \emptyset$ **then**
  $\mathcal{S}' \leftarrow \mathcal{S}' \cup \mathbf{x}_p$
endif
return $\mathcal{S}'$

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 section 4.

Algorithm 3 describes how an upper bound may be build from the solution of the linear problem used in the lower bounding procedure. Here, instead of calling a local solver, the procedure exploits directly the solution x^*_{LP} computed by a linear program solver. So the first step is to correct x^*_{LP} to build a feasible solution x^*_{corr} by means of function $FeasibilityCorrection$. As in algorithm 2, the $InflateAndProve$ procedure then tries to prove that a feasible point exists within a box build around x^*_{corr}.

4 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.

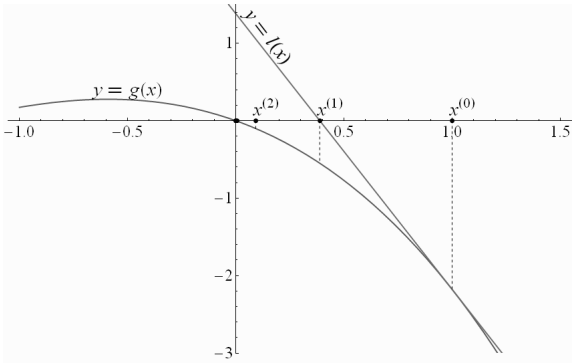

Fig. 1. Illustration of the Newton iteration

4.1 Newton for square systems of equations

The basic idea of the Newton method [13] applied to a system of equations $g(x) = 0$, where $g = (g_1, \ldots, g_n) : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, is to improve an estimated solution $x^{(0)} \in \mathbb{R}^n$ in the following way: the function g is linearized around $x^{(0)}$ leading to the linear approximation $l(x) \approx g(x)$ with

$$l(x) := g(x^{(0)}) + J_g(x^{(0)}) \cdot (x - x^{(0)}), \tag{2}$$

where J_g is the Jacobian matrix of g, i.e. $J_{g_{ij}} = \partial g_i / \partial x_j$. Then, the linearized system of equations $l(x) = 0$ is solved, leading to a new estimation

$$x^{(1)} = x^{(0)} - J_g^{-1}(x^{(0)}) g(x^{(0)})$$

of the solution of $g(x) = 0$. Provided that the approximated solution is close enough of the actual solution, we can expect $x^{(1)}$ to be much more accurate than $x^{(0)}$ (cf. Figure 1). This process is repeated to obtain a sequence $x^{(k)}$ which converges generally to the exact solution of the original equation (cf Figure 1).

Although the convergence of the Newton iteration is difficult to foresee, it converges well in practice for good initial approximations. Provided that the exact solution $\tilde{x}$ to be approximated is non-singular, i.e. $\det J_g(\tilde{x}) \neq 0$, the Newton iteration has a quadratic order of convergence, that is to say, the number of correct digits is multiplied by two at each iteration.

4.2 Newton for under-constrained systems of equations

When the system of equations $g(x) = 0$ is under-constrained, i.e. $g = (g_1, \ldots, g_m) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ with $m < n$, there is a manifold of solutions. The linear approximation (2) 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 (see Figure 2).

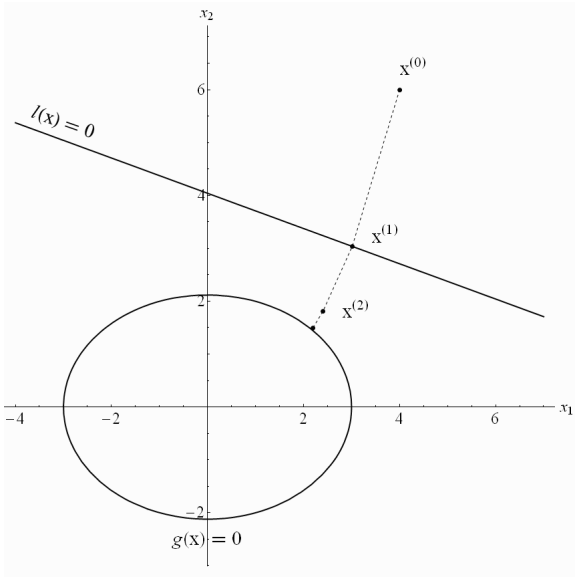

Fig. 2. Illustration of the Newton method adapted to under-constrained systems of equations

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 $x^{(0)}$, as illustrated by Figure 2. This solution can easily be computed with the Moore-Penrose inverse [14] (also called the generalized inverse or the pseudo inverse):

$$x^{(1)} = x^{(0)} - A_g^+(x^{(0)})g(x^{(0)}), \tag{3}$$

where $A_g^+ \in I\!R^{n\times m}$ is the Moore-Penrose inverse of $A_g \in I\!R^{m\times n}$, the solution of the linearized equation which minimizes $||x^{(1)} - x^{(0)}||$. Applying (3) recursively leads to a sequence of vectors which converges to a solution close to the initial approximation, provided that this latter is accurate enough (cf. Figure 2).

The Moore-Penrose inverse can be computed in several ways:

- A singular value decomposition can be used[8].
- 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 $A_g^+ = A_g^T(A_g A_g^T)^{-1}$.

[8] In this case, the Newton method presented in this Section is probably similar to the local search presented in [2] although the authors do not provide a detailed description of their method.

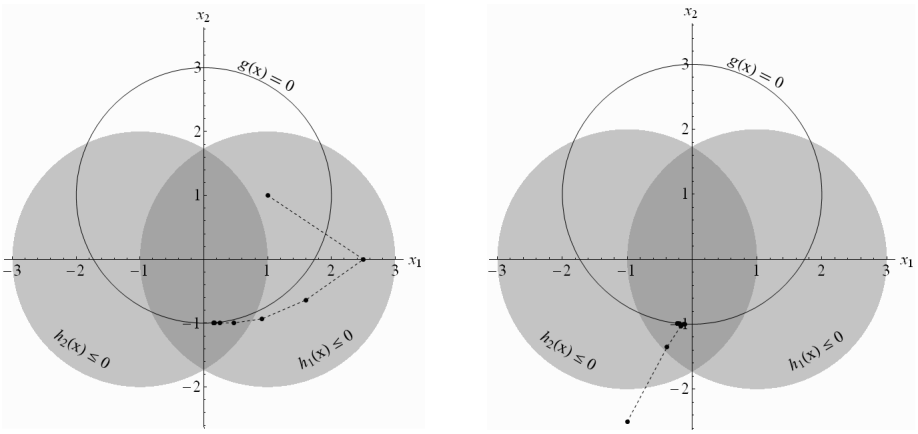

Fig. 3. Illustration of the Newton method adapted to under-constrained systems of equations and inequalities

4.3 Dealing with inequality constraints

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. Some initial values for the slack variables has to be provided in order to start the Newton method. It is not clear whether some optimal choice exists. Experiments have shown that 0 is not a good initial value because of the symmetry of the problem ($s_i^2 = (-s_i)^2$). A slightly positive value, e.g. $s_i = 0.1$, seems to break this symmetry while showing a good convergence. Figure 3 shows the convergence of this method on an example with one equality constraint and two inequality constraints for two different starting points.

5 Experiments

In this Section, we report the results of the experiments with our new upper bounding strategy on a significant set of benchmarks.

All the benchmarks come from the collection of benchmarks of the Coconuts project[9]. Note that all the problems are detailed on the Coconuts website. From this library, benchmarks with more than 100 variables or 100 constraints[10] have been rejected, as well as benchmarks using functions that are not yet handled

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

[10] Problems with up to 100 variables and 100 constraints are classified as hard problems for rigorous global optimizer in [12].

by Icos, like power[11] with real numbers (i.e. x^y where y is not a positive integer) or binary variables. Due to the lack of space, we report here only 35 selected benchmarks where Icos succeeds to find the global minimum while relying on an unsafe local search. Indeed, this unsafe upper bounding procedure could let the Branch and Bound procedure produce an empty encloser of the global minima even when the benchmark is known to have a global optima. This did actually happen: the upper bound computed with this strategy was lower than the lower bound proposed on the Coconuts project website for more than 40 of the tested problems.

We compare our new upper bounding strategy with two different upper bounding strategies. These strategies relies on a local search called within the Branch and Bound process. They handle the local search guess in the two following ways:

S1: This strategy directly uses the guess from the local search, i.e. this strategy uses a simplified version of algorithm 2 where the proof procedure has been dropped. As a consequence, it does not suffer from the difficulty to prove the existence of a feasible point. However, this strategy is unsafe and may produce wrong results.

S2: This strategy attempts to prove the existence of a feasible point within a box build around the local search guess. In other words, strategy S2 is nothing but algorithm 2. Here, all provided solutions are safe and the solving process is rigorous.

Strategies S1 and S2 are compared with our upper bounding strategy, name S3, where the upper bounding relies on the optimal solution of the problem linear relaxation to build a box proved to hold a feasible point. A call to the correction procedure attempts to compensate the effect of the outer approximation. Thus, strategy S3 is nothing but algorithm 3.

To sum up, the first strategy shows that the problem is much easier to solve without proving the existence of a feasible point. This is an unsafe, and thus unreliable, strategy. The two other strategies are safe and rigorous: strategy S2 relies on a local search while strategy S3 relies on the optimal solution of the linear relaxation.

All the tests have been ran with Icos[12] on a core duo T7700 laptop under linux. A time out of 30s is used to bound the running time. Icos relies on Ipopt[13] for the local search and on Coin LP[14] to solve the linear relaxation. The proof of existence is based on an implementation of the Borsuk theorem [4].

Table 1 reports the results of our experiments. First column gives the name of the benchmark. The second column gives the number n of variables and the number m of constraints. The next three couples of columns reports the number of found solutions and the time required to find the global optima for the

[11] Obviously, Icos handles x^n where n is a positive integer.
[12] See http://ylebbah.googlepages.com/icos.
[13] See http://www.coin-or.org/projects/Ipopt.xml.
[14] See http://www.coin-or.org/projects/Clp.xml.

		LS in BB				UB from LB	
		S1: Unsafe		S2: Safe		S3: Safe & Corr	
Name	(n,m)	n_{sols}	$t(s)$	n_{sols}	$t(s)$	n_{sols}	$t(s)$
alkyl	(14, 7)	1	0.30	0	-	5	1.54
circle	(3, 10)	1	2.30	1	1.98	50	0.84
ex14_1_2	(6, 9)	3	0.16	0	-	77	1.74
ex14_1_3	(3, 4)	1	0.09	7	-	56	0.42
ex14_1_6	(9, 15)	3	0.89	0	-	48	12.44
ex14_1_8	(3, 4)	4	0.28	0	-	1	-
ex2_1_1	(5, 1)	2	0.11	2	0.09	4	0.04
ex2_1_2	(6, 2)	1	0.05	0	-	20	0.24
ex2_1_3	(13, 9)	2	0.11	0	-	10	1.32
ex2_1_4	(6, 5)	1	0.07	2	0.52	2	0.43
ex2_1_6	(10, 5)	2	0.26	2	1.61	1	0.35
ex3_1_3	(6, 6)	2	1.08	1	1.03	1	0.29
ex3_1_4	(3, 3)	1	5.78	2	6.51	6	0.14
ex4_1_2	(1, 0)	1	20.52	1	18.84	12	17.03
ex4_1_6	(1, 0)	2	0.11	2	0.11	290	14.28
ex4_1_7	(1, 0)	2	0.07	2	0.07	7	0.01
ex5_4_2	(8, 6)	1	9.69	0	-	9	18.15
ex6_1_2	(4, 3)	2	0.46	2	0.51	10	0.52
ex6_1_4	(6, 4)	1	6.82	2	7.45	21	8.92
ex7_3_5	(13, 15)	3	17.03	0	-	0	-
ex7_3_6	(17, 17)	0	8.53	0	7.04	0	0.46
ex8_1_6	(2, 0)	2	2.64	0	-	11	0.39
ex9_1_1	(13, 12)	1	0.43	0	-	5	-
ex9_1_10	(14, 12)	1	0.23	1	-	9	3.76
ex9_1_4	(10, 9)	1	0.12	0	-	2	0.49
ex9_1_5	(13, 12)	1	0.13	0	-	1	2.68
ex9_1_8	(14, 12)	1	0.22	0	-	9	3.76
ex9_2_1	(10, 9)	2	0.68	0	-	7	0.68
ex9_2_4	(8, 7)	1	2.28	1	2.94	13	0.69
ex9_2_5	(8, 7)	3	0.68	0	-	0	-
ex9_2_7	(10, 9)	2	0.62	0	-	7	0.68
ex9_2_8	(6, 5)	1	0.08	0	-	2	0.53
house	(8, 8)	1	0.74	0	-	1	0.90
like	(9, 3)	0	4.06	0	4.10	0	4.08
nemhaus	(5, 5)	1	0.00	1	0.02	1	0.01

Table 1. Comparing the UB from LB strategy with a local search

three explored upper bounding strategies. The columns n_{sols} give the number of solutions found by Icos. Note that for strategy S1, these solutions are unproven solutions generated by the local search. The columns $t(s)$ gives in seconds the amount of time required to find a global minima. When the amount of time required to find a global minima is higher than 30 seconds, this value is replaced by a "-".

Algorithm 4 Upper bounding with correction

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

% $\mathcal{S}'$: list of proven feasible boxes;
% $nbStarts$: number of starting points
$\mathcal{S}' \leftarrow \emptyset$
for $i \leftarrow 1$ *to* $nbStarts$ **do**
  $x_0 \leftarrow$Select a starting point in $\mathbf{x}$
  $x^* \leftarrow \text{LS}(x_0)$ % x^* is a guess point computed by the local solver LS
  $x^*_{corr} \leftarrow$FeasibilityCorrection(x^*) % Improving x^* feasibility
  $\mathbf{x}_p \leftarrow$InflateAndProve(x^*_{corr}, $\mathbf{x}$)
  if $\mathbf{x}_p \neq \emptyset$ **then**
    $\mathcal{S}' \leftarrow \mathcal{S}' \cup \mathbf{x}_p$
  endif
endfor
return $\mathcal{S}'$

Of course, S1 is the fastest strategy but the reader must remember that strategy S1 is unsafe, and that the selected benchmarks are the one where strategy S1 succeeds. For example, with strategy S1, Icos reports that the *ex14_1_4* benchmark has no solution while it computes the global optima with all other strategies. Thus, we cannot rely on strategy S1. However, it emphasises the computational cost to provide a proven global optima. As a matter of fact, strategy S2 failed to provide the global optima within 30s for 20 benchmarks. This is more than half of the benchmarks of table 1.

Strategy S3 shows the benefits of our new upper bounding strategy: 31 benchmarks are now solved within the 30s time out, i.e., 16 more benchmarks than with S2. Moreover, almost all benchmarks are solved in much less time and with a greater amount of proven solutions. Only *ex4_1_6* and *ex6_1_4* need more time to be solved with strategy S3 than with strategy S2. The time required to solve these two problems is probably due to a slow convergence of the correction procedure. However, as a whole, strategy S3 improves drastically the performance of the upper bounding procedure and provides a serious alternative to the more traditional local search. It competes well with a local search solving 31 benchmarks while strategy S2 solves only 15 benchmarks within 30s.

To emphasize the benefit of this new strategy, we have explored two other approaches: the first one uses the proposed correction but with a local search whereas the second one exploits the optimal solution of the linear outer approximation but without the proposed correction. Table 2 shows that strategy S3 outperforms these approaches.

Strategy S4 applies the correction procedure to the output of the local search in an attempt to improve the approximate solution given by a local search. Algorithm 4 adds to the previous upper bounding procedure a call to the *Feasibility Correction* function to correct the local solver "guess". The corrected approximation x^*_{corr} is then directly given to the proof procedure which attempts to

		LS in BB		UB from LB		Combination	
		S4: Safe & Corr		S5: Safe		S6: S2+S3	
Name	(n,m)	n_{sols}	$t(s)$	n_{sols}	$t(s)$	n_{sols}	$t(s)$
alkyl	(14, 7)	0	-	1	-	5	1.66
circle	(3, 10)	1	2.02	3	0.78	1	6.11
ex14_1_2	(6, 9)	0	-	0	-	77	2.30
ex14_1_3	(3, 4)	7	-	11	4.06	56	0.70
ex14_1_6	(9, 15)	0	-	9	-	48	18.37
ex14_1_8	(3, 4)	0	-	0	-	1	-
ex2_1_1	(5, 1)	2	0.09	3	-	3	0.10
ex2_1_2	(6, 2)	0	-	1	-	20	0.31
ex2_1_3	(13, 9)	2	0.82	2	-	11	1.52
ex2_1_4	(6, 5)	2	0.55	0	-	2	0.61
ex2_1_6	(10, 5)	2	1.58	1	-	1	0.48
ex3_1_3	(6, 6)	1	1.04	1	-	1	1.18
ex3_1_4	(3, 3)	2	4.14	11	0.28	3	4.28
ex4_1_2	(1, 0)	1	17.57	12	16.88	2	18.60
ex4_1_6	(1, 0)	2	0.11	290	14.25	2	0.12
ex4_1_7	(1, 0)	2	0.07	7	0.00	3	0.07
ex5_4_2	(8, 6)	0	-	0	-	9	19.32
ex6_1_2	(4, 3)	2	0.48	10	1.05	2	0.59
ex6_1_4	(6, 4)	2	7.43	34	12.67	5	8.28
ex7_3_5	(13, 15)	0	-	0	-	0	-
ex7_3_6	(17, 17)	0	2.92	0	0.60	0	2.71
ex8_1_6	(2, 0)	0	-	8	-	7	2.78
ex9_1_1	(13, 12)	0	-	0	-	5	-
ex9_1_10	(14, 12)	1	-	4	0.73	9	6.12
ex9_1_4	(10, 9)	0	-	0	-	2	0.70
ex9_1_5	(13, 12)	1	28.97	1	0.86	1	7.00
ex9_1_8	(14, 12)	1	-	4	0.70	4	6.15
ex9_2_1	(10, 9)	0	-	0	-	7	1.31
ex9_2_4	(8, 7)	1	3.07	7	0.50	13	2.61
ex9_2_5	(8, 7)	0	-	0	-	0	-
ex9_2_7	(10, 9)	0	-	0	-	7	1.23
ex9_2_8	(6, 5)	0	-	2	0.24	2	0.69
house	(8, 8)	0	-	0	-	1	1.68
like	(9, 3)	0	4.20	0	4.10	0	4.33
nemhaus	(5, 5)	1	0.02	0	-	1	0.02

Table 2. Other upper bounding strategies

build a proven box around it. Unfortunately, this strategy only succeeds to solve 2 more benchmarks than strategy S2.

Strategy S5 relies directly on the optimal solution of the problem linear relaxation to build a box proved to hold a feasible point. Such a solution is unfeasible as the linear relaxation is an outer approximation of the problem. However, when the splitted boxes of the Branch and Bound become small enough, the

epsilon inflation used by the proving procedure to build a box proved to contain a feasible point compensates the effect of the outer approximation. Thus, such a process should converge to the solution. In other words, strategy S5 differs from algorithm 3 by the fact that it does not call the correction procedure. Unfortunately, the narrowing of the boxes splitted within the Branch and Bound process does not sufficiently compensate for the effect of the outer approximation. As a result, S5 solves only 15 problems within 30s. Thus, S5 underlines the key role of the correction procedure in our strategy.

Finally, we have combined S2 and S3 in an attempt to get the benefit of two different upper bounding strategies. However, this combination does not seem to be successful here. Moreover, it tends to increase the time required to solve the problems.

6 Conclusion

Finding feasible points is a critical issue in safe Branch and Bound algorithms. Classical approaches compute a "guess" point with a unsafe local method and try to prove that a feasible point actually exists inside a box around this "guess" point. Practically, this strategy is often very costly, and worse, it may even fail in some cases.

In this paper, we have introduced a new strategy which uses as starting point the optimal solution of a linear relaxation of the initial problem. A correction –based on the Newton method for under-constrained systems of equations and inequalities– is used to bring this starting point inside the feasible region. First experiments demonstrate that this strategy drastically improves the performances of the upper-bounding process.

Our current work aims at improving and generalizing this framework and its implementation. In particular, we would like to exploit the sequence of values of the different variables provided by the lower bonding process to find potential feasible points and to drive the inflating process when the Newton method does not converge.

References

1. Venkataramanan Balakrishnan and Stephen P. Boyd. Global optimization in control system analysis and design. In C.T. Leondes, editor, *Control and Dynamic Systems: Advances in Theory and Applications*, volume 53. Academic Press, New York, New York, 1992.
2. J. Cruz and P. Barahona. Global Hull Consistency with Local Search for Continuous Constraint Solving. In *EPIA '01: Proceedings of the10th Portuguese Conference on Artificial Intelligence on Progress in Artificial Intelligence, Knowledge Extraction, Multi-agent Systems, Logic Programming and Constraint Solving*, pages 349–362, 2001.
3. Christodoulos A. Floudas. Deterministic global optimization in design, control, and computational chemistry. In *IMA Proceedings: Large Scale Optimization with Applications. Part II: Optimal Design and Control*, pages 129–184, 1997.

4. Andreas Frommer and Bruno Lang. Existence tests for solutions of nonlinear equations using borsuk's theorem. Technical Report BUW-SC 2004/2, Department of Mathematics, Faculty of Mathematics and Natural Sciences, University of Wuppertal, MathePrisma, 2004.
5. Eldon R. Hansen. *Global Optimization Using Interval Analysis*. Marcel Dekker, New York, 2004.
6. Rainer Horst and Hoang Tuy. *Global Optimization: Deterministic Approches*. Springer-Verlag, 1993.
7. Yahia Lebbah, Claude Michel, Michel Rueher, David Daney, and Jean-Pierre Merlet. Efficient and safe global constraints for handling numerical constraint systems. *SIAM Journal on Numerical Analysis*, 42(5):2076–2097, 2004.
8. Eric Lee and Constantinos Mavroidis. Solving the geometric design problem of spatial 3R robot manipulators using polynomial homotopy continuation. *Journal of Mechanical Design*, 124(4):652–661, December 2002.
9. Olivier Lhomme. Consistency techniques for numeric CSPs. In *Proceedings of IJCAI'93*, pages 232–238, Chambéry(France), 1993.
10. Arnold Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 2004.
11. Arnold Neumaier and Oleg Shcherbina. Safe bounds in linear and mixed-integer programming. *Mathematical Programming*, pages 283–296, 2004.
12. Arnold Neumaier, Oleg Shcherbina, Waltraud Huyer, and Tamás Vinkó. A comparison of complete global optimization solvers. *Math. Program.*, 103(2):335–356, 2005.
13. J.M. Ortega and W.C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. SIAM, 1970.
14. C. R. Rao and S. K. Mitra. *Generalized Inverse of Matrices and Its Applications*. New York: Wiley, 1971.