Using constraint techniques for a safe and fast implementation of optimality-based reduction

Yahia Lebbah
Département d'Informatique
Université d'Oran Es-Senia
B.P. 1524 EL-M'Naouar
31000 Oran, Algeria
and
Laboratoire I3S, UNSA-CNRS
2000, route des lucioles
BP.121
06903 Sophia Antipolis -
Cedex France
ylebbah@gmail.com

Claude Michel
Laboratoire I3S, UNSA-CNRS
2000, route des lucioles
BP.121
06903 Sophia Antipolis -
Cedex France
cpjm@polytech.unice.fr

Michel Rueher
Laboratoire I3S, UNSA-CNRS
2000, route des lucioles
BP.121
06903 Sophia Antipolis -
Cedex France
rueher@polytech.unice.fr

ABSTRACT

Optimality-based reduction attempts to take advantage of the known bounds of the objective function to reduce the domain of the variables, and thus to speed up the search of a global optimum. However, the basic algorithm is unsafe, and thus, the overall process may no longer be complete and may not reach the actual global optimum. Recently, Kearfott has proposed a safe implementation of optimality-based reduction. Unfortunately, his method suffers from some limitations and is rather slow. In this paper, we show how constraint programming filtering techniques can be used to implement optimality-based reduction in a safe and efficient way.

Categories and Subject Descriptors

G.1.6 [**Optimization**]: Global optimization; F.4.1 [**Mathematical Logic**]: Logic and constraint programming

General Terms

Reliability Algorithms

Keywords

Global optimization, Constraint programming, Reliable computing

1. INTRODUCTION

Global optimization is a critical issue in numerous application of AI ranging from robot arm design [17], safety ver-

ification problems [2] to chemical equilibrium problems [9] and scheduling [2]. 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,

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

with $x \in \mathbf{x}$, $f : I\!R^n \to I\!R$ and $g_j : I\!R^n \to I\!R$; Functions f and g_j are continuously differentiable on some vector $\mathbf{x}$ of intervals of $I\!R$.

The difficulties in such global optimization problems come mainly from the fact that there are generally many local minimizers but only few of them are global minimizers, and that the feasible region may be disconnected [23]. That is why it is often hard to solve continuous global optimization problems. Thus, different techniques have been proposed to improve the standard branch and bound methods. Optimality-based reduction is one of the most interesting of these techniques.

Optimality-based reduction (OBR) has been introduced by Ryoo and Sahinidis in [26] 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[1]. However, the basic OBR algorithm is unsafe, and thus, the overall process may no longer be complete and may not reach the actual global optimum.

Before going into further details, let us illustrate how OBR

---

[1]Actually, Ryoo and Sahinidis have introduced in [26] four tests separated in two different techniques : optimality-based reduction and probing. While optimality-based reduction appears as an effective technique, probing, in its attempt to reduce the bounds of a variable, is, according to their authors, less efficient.

is working on the small following optimization problem[2]:

$$\begin{aligned}
\text{minimize} \quad & -x_1 + x_1 x_2 - x_2 \\
\text{subject to} \quad & \\
& c_1 : -2x_1 + 2x_2 \leq 1 \\
& c_2 : 3x_1 - x_2 \leq 3; \\
& x_1, x_2 \in [0, 5]
\end{aligned} \quad (2)$$

There is only one nonlinear term $x_1 x_2$ which is linearized with bilinear relaxation. We obtain the following linear relaxation:

$$\begin{aligned}
\text{minimize} \quad & -x_1 - x_2 + x_3 \\
\text{subject to} \quad & \\
& c_1 : -2x_1 + 2x_2 \leq 1 \\
& c_2 : 3x_1 - x_2 \leq 3; \\
& c_3 : \overline{x}_2 x_1 + \overline{x}_1 x_2 - x_3 \leq \overline{x}_1 \overline{x}_2 \\
& c_4 : \underline{x}_2 x_1 + \underline{x}_1 x_2 - x_3 \leq \underline{x}_1 \underline{x}_2 \\
& x_1, x_2 \in [0, 5]; x_3 \in [0, 25]
\end{aligned} \quad (3)$$

where x_3 is a linearization variable which stands for the bilinear term $x_1 x_2$; $\overline{x}_i$ denotes the upper bound of variable x_i whereas $\underline{x}_i$ denotes its lower bound.

Solving relaxation (3) with the simplex algorithm yields the following dual multipliers $\lambda_1^* = 0$ and $\lambda_2^* = 0.1666$, and the lower bound $L = -1.08333$. Solving locally the original problem (2) with the local solver Minos [22] yields the upper bound $U = -1.005$ on the feasible point $x^* = (0.917, 1.062)$.

The OBR method (see section 4.1) uses relation $-2x_1 + 2x_2 - 1 \geq -(U - L)/\lambda_1^*$ on the active constraint c_1 to generate relation $x_1 \leq 4.57$ which enables to remove interval $[4.57, 5]$ from the domain of x_1. The critical issue is the safe computation of the dual solution of c_1. Recently, Kearfott [13, 12] has proposed a safe implementation of OBR which is based on a valid bounding of the dual solution. Unfortunately, his method suffers from strong limitations and is rather slow.

We show in this paper that constraint programming techniques can be used in a very simple 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.

The rest of this paper is organized as follows. The first section recalls basics of continuous filtering techniques required in the rest of paper. We describe also the principles of the Quad algorithm, which plays an essential role in our framework. Section 2 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. Finally, preliminary experimental results are discussed.

2. BASICS ON FILTERING TECHNIQUES OVER CONTINUOUS DOMAINS

The first sub-section recalls basics of continuous filtering techniques required in the rest of paper. The second sub-section recalls the intuition behind the Quad-filtering tech-

niques used in our algorithm. A detailed discussion of these concepts and techniques can be found in [4, 14, 16].

2.1 *2B*-filtering and *Box*-filtering

Arc-consistency plays a key role in constraint programming. Roughly speaking, a constraint c_j is arc-consistent [19] if for any variable x_i in c_j, each value in $\mathbf{x}_i$ has a support in the domains of all other variables of c_j. However, in general it is not possible to achieve arc-consistency on constraint systems over the reals. That is why different relaxations of arc consistency have been proposed. The most famous relaxations of arc consistency used in continuous domains are $2B$–consistency and box–consistency.

$2B$–consistency (also known as hull consistency) [7, 27, 5, 18] requires only to check the arc-consistency property for each bound of the intervals. The key point is that this relaxation is more easily verifiable than arc-consistency itself. Informally speaking, variable x is $2B$-consistent for constraint "$f(x, x_1, \ldots, x_n) = 0$" if the lower (resp., upper) bound of the domain of x is the smallest (resp., largest) solution of $f(x, x_1, \ldots, x_n) = 0$.

$2B$–consistency states a local property on the bounds of the domain of a variable at a single constraint level. A constraint c is $2B$-consistent if, for any variable x_i, there exists values in the domains of all other variables which satisfy c when x_i is fixed to $\underline{x}_i$ and $\overline{x}_i$.

Box–consistency [4] is a coarser relaxation (i.e., it allows less stringent pruning) of arc-consistency than $2B$–consistency. Variable x is box–consistent for constraint "$f(x, x_1, \ldots, x_n) = 0$" if the bounds of the domain of x correspond to the leftmost and rightmost zeros of the optimal interval extension of $f(x, x_1, \ldots, x_n)$. However, $2B$–consistency algorithms actually achieve a weaker filtering (i.e., a filtering that yields bigger intervals) than box–consistency when at least one variable occurs more than once in some constraint [8]. This is due to the fact that $2B$–consistency algorithms require a decomposition of the constraints with multiple occurrences of the same variable. Box–consistency generates a system of univariate interval functions which can be tackled by numerical methods such as interval Newton. In contrast to $2B$–consistency, box–consistency does not require any constraint decomposition and thus does not amplify the locality problem.

2.2 Quad-filtering

The Quad-filtering algorithm [16] is based on linear relaxation techniques. An LP solver is used to compute tight bounds of the domains of the variables. The Quad-filtering process consists of three main steps: reformulation, linearization, and pruning.

The reformulation step generates a set of implied linear constraints. More precisely, this set contains linear inequalities that approximate the semantics of nonlinear terms of the initial constraints.

The linearization process first decomposes each nonlinear term into sums and products of univariate terms; then it replaces nonlinear terms with their associated new variables. Nonlinear terms are approximated by tight linear convex relaxations.

The pruning step is just a fixed point algorithm that calls iteratively a LP solver to reduce the upper and lower bounds of every original variable. This process iterates until the difference between the upper and lower bound is below a given

---

[2]This problem derives from a problem found in [26] with a small variation on the constraint c_1.

ϵ. The algorithm converges and terminates if ϵ is greater than zero.

A safe procedure is a key issue in a process that involves linear relaxations. Thanks to the work of Neumaier and Shcherbina [24], a safe underestimation of a minimizer of a linear program is available. However, the computed minimizer is safe if and only if the problem itself is a conservative approximation of the initial problem. Michel et al. [20] have introduced corrections of the linear relaxations to ensure that the linear program is a conservative approximation of the initial program. This work has been extended by Borradaile and Van Hentenryck [6] to cover multivariate relaxations.

Therefore, the `Quad`-filtering algorithm offers an efficient and safe framework to prune the domains of the variables. It has shown its capability to achieve a more efficient pruning than $2B$−consistency or box−consistency. Moreover, the `Quad`-framework provides all the basics required to implement an efficient and safe global optimization solver based on linear relaxations.

Now, let us recall the overall branch and bound schema.

3. THE BRANCH AND BOUND SCHEMA

The well known branch and bound schema provides a very general framework to find a global minimizer. This framework is complete and can be implemented as a safe and rigorous platform [15]. We introduce directly the latter framework since the focus of this paper is on safe and rigorous approaches for continuous global optimization.

The branch and bound algorithm we use here combines interval analysis techniques and constraint programming techniques within the well known branch and bound schema described by Horst and Tuy in [11]. Interval analysis techniques enables to introduce safeguards that ensure rigorous and safe computations whereas constraint programming techniques improve the reduction of the feasible space.

The solver (see algorithm 1) computes enclosers for minimizers and safe bounds of the global minimum value within an initial box $\mathbf{x}$. The algorithm 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.

The algorithm 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. Then, *LowerBound*($\mathbf{x}'$) computes a rigorous lower bound of the objective within the box $\mathbf{x}'$ using a linear programming relaxation of the initial problem. Actually, function *LowerBound* is based on the linearization techniques of the `Quad` framework. The safe LP solver computes $\underline{\mathbf{f}}_{\mathbf{x}'}$.

UpperBox($\mathbf{x}$) computes a feasible box. A local search method helps to quickly find an approximate feasible point. Interval techniques are used to check the feasibility of the provided box[3]. If *UpperBox* succeeds to prove feasibility then the box $\mathbf{x}_p$ that contains this proven feasible point is added to the list $\mathcal{S}$. 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

---

[3]We rely on the techniques introduced by Hansen in [10] to handle under-determined systems

---

Algorithm 1 Branch and bound algorithm

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

% $\mathcal{S}$: set of proved feasible points
% $\mathbf{f}_{\mathbf{x}}$ denotes the set of possible values for f in $\mathbf{x}$
$\mathcal{L}\leftarrow\{\mathbf{x}\}$; $\mathcal{S}\leftarrow\emptyset$; $(L,U)\leftarrow(-\infty,+\infty)$;
while $w([L,U]) > \epsilon$ **do**
$\quad\mathbf{x}'\leftarrow\mathbf{x}''$ such that $\underline{\mathbf{f}}_{\mathbf{x}''} = min\{\underline{\mathbf{f}}_{\mathbf{x}''} : \mathbf{x}'' \in \mathcal{L}\}$; $\quad\mathcal{L}\leftarrow\mathcal{L}\backslash\mathbf{x}'$;
$\quad\overline{\mathbf{f}}_{\mathbf{x}'}\leftarrow min(\overline{\mathbf{f}}_{\mathbf{x}'},U)$;
$\quad\mathbf{x}'\leftarrow Prune(\mathbf{x}')$;
$\quad\underline{\mathbf{f}}_{\mathbf{x}'}\leftarrow LowerBound(\mathbf{x}')$;
$\quad(\overline{\mathbf{f}}_{\mathbf{x}'}, \mathbf{x}_p, Proved)\leftarrow UpperBox(\mathbf{x}')$;
$\quad$**if** $Proved$ **then** $\mathcal{S}\leftarrow\mathcal{S} \cup \{\mathbf{x}_p\}$; **endif**
$\quad$**if** $\mathbf{x}' \neq \emptyset$ **then**
$\quad\quad\quad(\mathbf{x}'_1,\mathbf{x}'_2)\leftarrow Split(\mathbf{x}')$; $\quad\mathcal{L}\leftarrow\mathcal{L} \cup \{\mathbf{x}'_1,\mathbf{x}'_2\}$;
$\quad$**endif**
$\quad$**if** $\mathcal{L} = \emptyset$ **then**
$\quad\quad\quad(L,U)\leftarrow(+\infty,-\infty)$;
$\quad$**else**
$\quad\quad\quad(L,U)\leftarrow(min\{\underline{\mathbf{f}}_{\mathbf{x}''} : \mathbf{x}'' \in \mathcal{L}\}, min\{\overline{\mathbf{f}}_{\mathbf{x}''} : \mathbf{x}'' \in \mathcal{S}\})$;
$\quad$**endif**
endwhile

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[4]. At each box selection and processing, the algorithm 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.

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.

4. OPTIMALITY-BASED REDUCTION

4.1 Basics of optimality-based reduction

Optimality-based reduction has been introduced by Ryoo and Sahinidis in [26]. It takes advantage of a property of the saddle point (see for instance Chap. 5 of [21]) 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*

$$x_i \geq x_i' \text{ with } x_i' = \overline{x}_i - \frac{U - L}{\lambda_i^*}. \qquad (4)$$

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.

---

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

328

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*

$$x_i \leq x_i'' \text{ with } x_i'' = \underline{x}_i + \frac{U - L}{\lambda_i^*}. \tag{5}$$

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*

$$g_i(x) \geq -\frac{U - L}{\lambda_i^*}. \tag{6}$$

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 [26].

4.2 Kearfott's approach: safe OBR based on bounding rigorously dual variables

The critical issue in optimality-based reduction formulae (4)(5)(6) comes from the unsafe dual solution provided by the Simplex method. Note that the techniques suggested by Neumaier et al [24] and used in the `Quad` framework cannot be applied to compute a safe solution of the dual problem. So, the validity of dual solutions must be proved to keep the branch and bound process rigorous. This approach has been investigated by Kearfott [12] in the specific case of linear relaxations. For the sake of simplicity, we consider the following linear relaxation formulated with lower inequalities:

$$\begin{array}{ll} min & d^T x \\ s.t. & Ax \leq b \end{array} \tag{7}$$

The dual of (7) is the following LP

$$\begin{array}{ll} max & b^T y \\ s.t. & A^T \lambda = d \end{array} \tag{8}$$

where λ denotes the dual variables required in OBR formulae (4)(5)(6). The Kuhn-Tucker system (KT)[5] is used to provide validated lower and upper bounds on the system (7) and (8).

$$(KT) \begin{cases} A^T \lambda - d & = 0 \\ \lambda_i(A_{i,:}x - b_i) & = 0, 1 \leq i \leq m \end{cases} \tag{9}$$

where $A_{i,:}$ is i-th row of A.

[5]For a detailed introduction to Kuhn-Tucker system, see [3].

It is well known that if the i-th constraint $A_i : x \leq b_i$ is inactive at a solution –i.e., strict inequality $x^* < b_i$ holds for x^*, the solution of the primal– then the corresponding dual variable (also called Lagrange multiplier) y_i is equal to zero. Thus, inactive constraints must be identified to make the whole system (9) linear. In this approach there are two main critical issues:

1. Some of the constraints of (9) are redundant due to the fact that each equality constraint is replaced by two inequality constraints. So, the Newton methods cannot be used for the validation process.

2. Inactive constraints are identified with the approximate dual solution provided by the Simplex method (i.e., inactive constraints have a null dual solution).

Kearfott handles these issues by weakening the relaxation. Consequently, the final validation method based on the Newton method applied to (9) does not always succeed and, when it succeeds, the bounds could be wide due to the fact that the relaxation has been weakened.

4.3 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 4 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

$$\begin{array}{ll} f(x) \leq U & \\ g_i(x) = 0, & i = 1..k \\ g_j(x) \leq 0, & j = k + 1..m \end{array} \tag{10}$$

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 objective function $\mathbf{f}$ by means of formula 6.

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 $(\underline{\mathbf{f}}_{\mathbf{x}'}, x', \mathcal{L}) \leftarrow LowerBound(\mathbf{x}', L, U, \mathcal{L})$.

5. EXPERIMENTAL RESULTS

To evaluate the contribution of the constraint-based safety proof we have compared its performance with the method proposed by Kearfott [12] on 78 benchmarks. 17 of them, the 'c-chem' problems, are described in [25]. 6 of them come from pages 140 to 147 of Audet thesis [1]. The 55 other problems come from COCONUT Library 1 benchmarks[6]. All

[6]http://www.mat.univie.ac.at/~neum/glopt/coconut/ Benchmark/Library1_new_v1.html The tested benchmarks are : alkyl, chance, circle, dispatch, ex14_1_1, ex14_1_2, ex14_1_3, ex14_1_4, ex14_1_5, ex14_1_6, ex14_1_8, ex14_1_9, ex14_2_2, ex14_2_5, ex2_1_1,

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 4 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 6 of OBR to reduce the
        lower bound of the objective function $\mathbf{f}$
Use $2B$-filtering and Quad-filtering to validate the reduction

---

these problems have less than 100 variables and 100 constraints.

The branch and bound algorithm has been implemented within the ICOS solver[7]. The linear relaxations for lower bounding are solved by means of the linear programming solver Coin/CLP[8]. The upper-bounding step uses the non-linear programming solver Coin/IPOPT[9] based on the interior point method. All experiments have been done on PC-Notebook/1Ghz.

We have performed the following experiments:

- a branch and bound without optimality-based reduction (labelled "no OBR" in the tables);

- a branch and bound with unsafe optimality-based reduction (labelled "unsafe OBR" in the tables);

- a branch and bound with safe optimality-based reduction based on Kearfott's approach (labelled "safe OBR Kearfott" in the tables);

- a branch and bound with safe optimality-based reduction on our approach (labelled "safe OBR CP" in the tables);

To refute the boxes rejected by OBR, a combination of a $2B$-filtering and $Quad$-filtering has been used. More precisely, $Quad$-filtering is only used when $2B$-filtering fails to reject the boxes identified by OBR.

Table 1 is a synthesis of the results obtained by the different methods with a timeout of 500s. The second column gives the total amount of time (in second) required to compute all the benches (with a timeout of 500s). The last column gives the percentage of time saved using one of the

ex2_1_10, ex2_1_2, ex2_1_4, ex2_1_5, ex2_1_6, ex3_1_2, ex3_1_3, ex3_1_4, ex4_1_1, ex4_1_3, ex4_1_4, ex4_1_5, ex4_1_6, ex4_1_7, ex4_1_8, ex4_1_9, ex5_2_2_case1, ex5_2_2_case3, ex5_4_2, ex6_1_2, ex6_1_4, ex7_2_2, ex7_2_5, ex7_2_6, ex7_2_10, ex7_3_1, ex7_3_2, ex7_3_3, ex8_1_1, ex8_1_5, ex8_1_6, ex8_1_8, ex9_1_6, ex9_1_7, ex9_1_9, ex9_2_8, himmel11, house, nemhaus, rbrock, sample, wall

[7] http://www.essi.fr/~lebbah/icos/index.html
[8] http://www.coin-or.org/cgi-bin/cvsweb.cgi/COIN/Clp/
[9] http://list.coin-or.org/mailman/listinfo/coin-ipopt

	$\Sigma_t(s)$	$\%saving$
no OBR	2384.36	-
unsafe OBR	881.51	63.03%
safe OBR Kearfott	1975.95	17.13%
safe OBR CP	454.73	80.93%

Table 1: Synthesis of the results (with a timeout of 500s)

name	safe OBR CP	safe OBR Kearfott	% saving
himmel11	2.4	-	-
ex5_2_2_case3	4.49	-	-
c-chem7	10.5	-	-
ex2_1_5	3.52	15.64	77.49%
ex7_2_5	2.19	7.14	69.32%
ex14_2_5	2.48	7.19	65.50%
ex7_2_10	0.16	0.41	60.97%
ex8_1_6	0.73	1.54	52.59%
ex14_2_2	3.05	6.38	52.19%
ex7_3_1	3.38	5.95	43.19%
ex2_1_1	0.25	0.21	-19.04%
ex9_1_9	0.11	0.09	-22.22%
ex2_1_4	1.25	1.01	-23.76%
ex9_2_8	0.05	0.04	-25%
c-chem18	4.83	3.57	-35.29%
ex3_1_2	0.19	0.14	-35.71%
c-audet147	0.61	0.44	-38.63%
c-audet140b	0.13	0.09	-44.44%
alkyl	32.45	20.55	-57.90%
ex5_2_2_case1	7.75	2.88	-169.09%

Table 2: Safe OBR CP vs safe OBR Kearfott

optimality-based method instead of the branch and bound alone.

In the first line, the OBR is used in an unsafe way. Thus, the result provided by the branch and bound process might be wrong. However, these experiments underline the potential benefit of the OBR in a branch and bound process. As a matter of fact, OBR might dramatically improve the speed of the search for a global optima.

Unfortunately, most of this benefit is lost by Kearfott's safe approach. A contrario, the constraint-based approach still increases this improvement. An essential observation is that the constraint-based approach managed to handle all these problems in less than $56s$ whereas Kearfott's method is stopped by a timeout of $500s$ for 3 of them.

The reader may be surprised by the fact that safe constraint-based OBR is even more efficient than the unsafe OBR. A careful analysis of the benchmarks was necessary to understand this point. Actually, this is due to the fact that wrong domains reductions achieved by the unsafe OBR prevent the upper-bounding process from improving the current upper bound.

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[10]. This is outlined by table 2 which compares the results of the constraint-based approach with the result of Kearfott's method. The 10 first benchmarks are the ones where constraint-based OBR has the best behavior compared to Kearfott's method. The 10 last benchmarks are the ones where constraint-based OBR has the worst behavior compared to Kearfott's method. The first column contains the name of the benchmark. The second column gives the amount of time (in seconds) required to solve the benchmark with the constraint-based approach. The third column gives the amount of time (in seconds) required to solve the benchmark with Kearfott's method. The last column gives the percentage of time saved using the constraint-based approach instead of Kearfott's method.

6. CONCLUSION

In this paper, we show how 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.

7. REFERENCES

[1] C. Audet. *Optimisation Globale Structurée: Propriétés, Équivalences et Résolution*. PhD thesis, École Polytechnique de Montréal, Quebec, 1997.

[2] V. Balakrishnan and S. P. Boyd. Global optimization in control system analysis and design. In C. Leondes, editor, *Control and Dynamic Systems: Advances in Theory and Applications*, volume 53. Academic Press, New York, New York, 1992.

[3] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming : Theory and Algorithms*. John Wiley & Sons, 1993.

[4] F. Benhamou, D. McAllester, and P. V. Hentenryck. Clp(intervals) revisited. In *Proc. of the ISLP'94*, pages 124–138, 1994.

[5] F. Benhamou and W. Older. Applying interval arithmetic to real, integer and boolean constraints. *Journal of Logic Programming*, pages 1–24, 1997.

[6] G. Borradaile and P. V. Hentenryck. Safe and tight linear estimators for global optimization. *Mathematical Programming*, 2005.

[7] J. G. Cleary. Logical arithmetic. *Future Computing Systems*, pages 125–149, 1987.

[8] H. Collavizza, F. Delobel, and M. Rueher. Comparing partial consistencies. *Reliable Computing*, pages 213–228, 1999.

[9] C. 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.

[10] E. R. Hansen. *Global Optimization Using Interval Analysis*. Marcel Dekker, New York, 2004.

[11] R. Horst and H. Tuy. *Global Optimization: Deterministic Approches*. Springer-Verlag, 1993.

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

[13] 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.

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

[15] Y. Lebbah, C. Michel, and M. Rueher. An efficient and safe framework for solving optimization problems. *JCAM (accepted for publication)*, 2005.

[16] 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.

[17] E. Lee and C. Mavroidis. Solving the geometric design problem of spatial 3R robot manipulators using polynomial homotopy continuation. *Journal of Mechanical Design*, 124(4):652–661, Dec. 2002.

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

[19] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, pages 99–118, 1977.

[20] C. Michel, Y. Lebbah, and M. Rueher. Safe embedding of the simplex algorithm in a CSP framework. In *Proc. of CPAIOR 2003, Montréal*, 2003.

[21] M. Minoux. *Mathematical Programming. Theory, Algorithms and Applications*. Wiley, 1986.

[22] B. A. Murtagh and M. A. Saunders. Minos 5.5 user's guide. Technical Report SOL 83-20R, Systems Optimization Laboratory, Dep. of Operations Research, Stanford University, July 1998.

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

[24] A. Neumaier and O. Shcherbina. Safe bounds in linear and mixed-integer programming. *Mathematical Programming*, pages 283–296, 2004.

[25] H. S. Ryoo and N. V. Sahinidis. Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers Chemical Engineering*, 19(5):551–566, 1995.

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

[27] D. Sam-Haroud and B. Faltings. Consistency techniques for continuous constraints. *Constraints*, 1(1/2):85–118, 1996.

---

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