Box-Set Consistency for interval-based constraint problems

Gilles Chabert, Gilles Trombettoni and Bertrand Neveu
PROJET COPRIN I3S-INRIA-CERTIS
2004 route des Lucioles BP 93
06902 Sophia Antipolis Cedex, FRANCE
{chabert, trombe, neveu}@sophia.inria.fr

ABSTRACT

As opposed to finite domain CSPs, arc consistency cannot be enforced, in general, on CSPs over the reals, including very simple instances. In contrast, a stronger property, the so-called *box-set consistency*, that requires a no-split condition in addition to arc consistency, can be obtained on a much larger number of problems.

To obtain this property, we devise a lazy algorithm that combines hull consistency filtering, interval union projection, and intelligent domain splitting. It can be applied to any numerical CSP, and achieves box-set consistency if constraints are redundancy-free in terms of variables. This holds even if the problem is not interval-convex. The main contribution of our approach lies in the way we bypass the non-convexity issue, which so far was a synonym for either a loss of accuracy or an unbounded growth of label size.

We prove the correctness of our algorithm and through experimental results, we show that, as compared to a strategy based on a standard bisection, it may lead to gains while never producing an overhead.

1. INTRODUCTION

Constraint programming (CP) is a complementary approach of interval-based numerical methods to solve systems of equations over the reals. Instead of approximating solutions with mathematical reasoning on the entire system, equations are treated as independent compatibility relations (called *constraints*) between variables. In this perspective, domains are filtered by removing values for which a relation cannot be satisfied individually. On the one hand, these methods make hardly use of semantics, but on the other hand, they are general-purpose. A mix of CP and interval analysis has given rise to the best solvers for satisfying or optimizing constraint systems.

When the domains of the variables are stable for all the constraints, i.e., when all the values can be used to satisfy any constraint, we see that we cannot go further with "local" reasoning, and we talk about *arc consistency*. A lot of algorithms have been designed to enforce *arc consistency* in discrete constraint systems (where domains have a finite number of values). This key property in CP can be obtained as well with real variables, but usually a few steps of a specific bisection are required. Thus, we obtain the so-called *box-set consistency*, a stronger property which yields a set of arc consistent boxes (i.e., products of intervals).

In this paper, we show in practice how to enforce box-set consistency for a class of projection-friendly problems. First, we recall some classical definitions about the CSPs and a precise description of what the box-set consistency is. Second, we give our algorithm, detailing the important projection operators. Third, we expound the theoretical properties of our algorithm, and finally deal with performances.

2. BACKGROUND

2.1 CSPs over the reals

A **numerical constraint satisfaction problem** (NCSP) is a 3-uple (C,V,B). C is a set of constraints $c_1, ..., c_m$ (equations or inequations) relating a set V of variables $x_1, ..., x_n$. Each variable is given an initial interval of real values $D_{x_1}, .., D_{x_n}$, and the problem is to find all the n-tuples of values $(v_1, ..., v_n)$, $v_i \in D_{x_i}$ $(1 \leq i \leq n)$, such that constraints are all satisfied when simultaneously each variable x_i is assigned v_i. $B = D_{x_1} \times ... \times D_{x_n}$ is the initial *box* of the problem. In fact, algorithms and properties introduced in this paper require a more general representation of domains: a union of intervals. So hereafter, D_x designates a union of intervals, and B the cartesian product of interval unions.

As said above, solving an NCSP with constraint programming often relies on *local filtering* techniques, i.e., techniques that only use properties of the system related to a single constraint. For instance, let (c) be the equation $x^2 = y$. Whatever are the other constraints, if $D_x = [-10, 10]$ and $D_y = [1, 4]$, we can safely reduce D_x to $[-2, -1] \cup [1, 2]$ right away. This narrowing operation is called *projection of c over x*. Here is a formal definition of projection, and the related property of arc consistency.

DEFINITION 2.1 (PROJECTION & ARC CONSISTENCY).
Let $P = (C, V, B = D_{x_1} \times ... \times D_{x_n})$ be a NCSP.
Let c be a constraint relating $\{y_1, ..., y_k\} \subset V$.
The **projection** *of c over y_i applied on B is the following set:*
$\Pi^c_{y_i}(B) = \{v_i \in D_{y_i} \mid \exists\ v_1 \in D_{y_1}, ..., v_{i-1} \in D_{y_{i-1}}, v_{i+1} \in D_{y_{i+1}}, ..., v_k \in D_{y_k}$ *such that c is satisfied with* $y_1 = v_1, ..., y_k = v_k\ \}$
P is **arc consistent** *iff $\forall < c, x > \in C \times V$ with x related by c, $D_x = \Pi^c_x(B)$*

hull consistency [1] (or 2B-consistency [2]) is an easy-to-apply approximation of arc consistency, for which efficient implementations exist, such as HC4 [3].

The contribution of this paper is to study an algorithm, and we need to consider various levels of complexity for the problems to state its correctness.

- **NCSPs with primitive constraints only**. A *primitive* constraint is a basic binary or ternary relation such as $z = x + y$. As the projections are mathematically known, they can be hard coded. The set of primitive constraints is given in Figure 1 below.
- **Simple NCSPs**. A constraint c is *simple* if the syntax of c does not include multiple occurrences of the same variable. For instance, $z = (x + y)^2$ is simple but $z = x^2 + y^2 + 2xy$ is not. A simple constraint is a combination of several primitive constraints. A NCSP is *simple* if all the constraints are simple.
- **General case**. Any constraint that is not simple falls into the general case. Intuitively, it is a hard problem to infer projections from the syntax of a constraint when variables occur more than once.

$p_1(x, y)$:	$x = y$	$p_6(x, y, z)$:	$z = x * y$
$p_2(x, y, z)$:	$z = x + y$	$p_7(x, y, z)$:	$z = x/y$
$p_3(x, y, z)$:	$z = x - y$	$p_8(x, y, a)$:	$y = x^a$
$p_4(x, y)$:	$y = exp(x)$	$p_9(x, y)$:	$y = cos(x)$
$p_5(x, y)$:	$y = ln(x)$	$p_{10}(x, y)$:	$y = sin(x)$

Figure 1: Primitive constraints

Finally, we need to recall a last definition, introduced in [5], to express a *no-disjunction* property:

DEFINITION 2.2 (INTERVAL-CONVEXITY). *Let c be a constraint relating variables $x_1, ..., x_k$, Let B be a box.*
c is interval-convex on B iff $\forall i \in \{1..k\}$ $\Pi^c_{x_i}(B)$ *is a single interval (e.g.,* $[-2,-1] \cup [1, 2]$ *is not a single interval)* .

2.2 The Box-Set Consistency

In general, obtaining an arc consistent NCSP from an initial box is not possible with continuous variables. Projections can split domains of variables (see example $y = x^2$ in Section 2), and the number of splits can grow indefinitely. See the extended version of this paper [6] for a precise description of this issue. But in contrast, it is easy to look for the arc consistent sub-boxes. Let us insist on the term *boxes*: a box is a cartesian product of intervals. This means that all the values inside a box are considered, that is, no gap is allowed. The difference is illustrated on the following example:

EXAMPLE 2.1.
Let $P = (\{c_1, c_2, c_3, c_4\}, \{x, y, z\},]-\infty, +\infty[^3)$ *be a NCSP.*
(c_1) $1 \leq (x - 2)^2 \leq 4$ $\quad (c_2)$ $y = x$
(c_3) $|z| = |x|$ $\quad (c_4)$ $z \geq y - 4$

Arc consistency is achieved with the following domains :
$D_x = D_y = [0, 1] \cup [3, 4]$ and $D_z = [-4, -3] \cup [-1, 1] \cup [3, 4]$. But $D_x \times D_y \times D_z$ is not a box because domains are not intervals, but unions of intervals (there are “gaps”). The 2 maximal arc consistent sub-boxes of P are $[0, 1] \times [0, 1] \times [-1, 1]$ and $[3, 4] \times [3, 4] \times [3, 4]$. Interval $I_z = [-4, -3]$ for z is discarded because there do not exist intervals I_x and I_y such that $I_x \times I_y \times I_z$ is arc consistent. Formally:

DEFINITION 2.3 (BOX-SET CONSISTENCY).
Let $P = (C, V, B)$ be a NCSP. The box-set consistency of P is the set $\{B'\}$ of maximal subboxes of B such that (C, V, B') is an arc consistent NCSP.

Here is a generic method that computes the box-set consistency of a problem. It is also the sketch of our algorithm `HC4+TAC` below, and corresponding lines are given in brackets.

[2] Apply a hull consistency filtering on the system (no split is done). Meanwhile, detect constraints that are not interval-convex.

[7-16] Search a gap, by examining constraints suspected to be not interval-convex, and by applying projections [14].

[18-21] If a projection discloses a gap, call recursively the procedure for the sub-systems obtained by instantiating the current variable to each sub-interval formed by the gap(s). This phase is called **natural splitting**.

We call this generic method `Lazy_BoxSet`. Box-set consistency formalizes the result of Hyvönen’s algorithm [7], and `Lazy_BoxSet` can be viewed as a lazy version of the latter. In both algorithms, the number of boxes produced is bounded by $(p \times a)^n$, where p is the maximum number of intervals obtained by *one* projection, and a is the arity of the variable. See [6] for details.

3. ALGORITHM

`Lazy_BoxSet` is a theoretical method because it is based on a projection operator, and nothing is said about the way to implement it. This operator is called at two different steps : during the hull consistency filtering, and during the gap search.

We are going to present in this section a way to work out this operator, what *de facto* will give us a ready-to-use version of a box-set filtering algorithm. We will say that a projection operator is *exact* when it respects the definition 2.1.

3.1 Overview

For performance reasons, we chose `HC4` [3] for our implementation of hull consistency. But the projection operator embedded in `HC4`, called `HC4Revise`, is exact only for primitive constraints. Despite this restriction, we will see that it is possible to obtain the box-set consistency of simple NCSPs, providing that projections for the gap search are exact. That is why we use a different operator for the gap search, called `TAC`.

Algorithm 1 `HC4+TAC`(NCSP (C, V, B), in-out `results`)

```
 2: HC4($(C, V, B)$)

 4: if $B$ is empty then
      return
 6: $C' \leftarrow C$
    gap ← false
 8: while (not gap) and ($C' \neq \emptyset$) do
      pop $c$ from $C'$
10:   if not (interval-convex($c$)) then
        $V' \leftarrow$ Variables($c, V$)
12:     while (not gap) and ($V' \neq \emptyset$) do
          pop $x$ from $V'$
14:       $U \longleftarrow$ TAC($c, x, B$)
          if $U \subsetneq D_x$ then
16:         gap ← true

18: if gap then
      for all intervals $I$ in $U$ do
20:     $D_x \longleftarrow I$          // $D_x$ is the domain of $x$ in $B$
        HC4+TAC($(C, V, B)$, results)
22: else
      add $B$ to results
```

Hence, we do not use the same operator in both places we have mentioned (hull consistency filtering and gap search), and this is why our algorithm is sumed up in the acronym HC4+TAC.

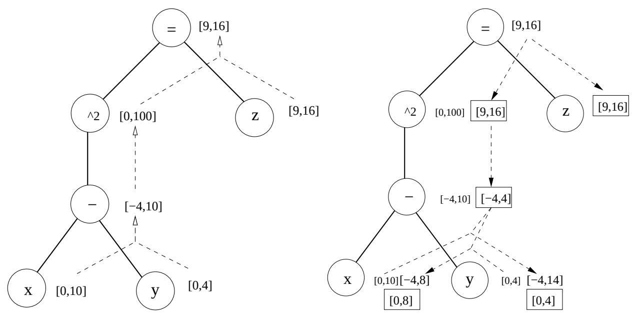
Note that line 15 hides a subtlety: gap may be true either when U is a union (as in the original algorithm), or when bounds of D_x can be narrowed. The latter case arises due to the limitation of HC4Revise evoked before. We will get back to this issue in 4.2. The rest of this section describes the projection operators, HC4Revise and TAC. Both originate from [3].

3.2 HC4Revise

First, let us summarize how HC4Revise works through an example. See [3] for a complete description.

Let $c : (x - y)^2 = z$ be a constraint relating $x \in [0, 10], y \in [0, 4]$ and $z \in [9, 16]$. Constraints are represented by their abstract syntax tree, where each node stands for a sub-expression.

Figure 2(a) depicts the first step of the retro-propagation algorithm, an upward traversal of the tree that computes an evaluation of every sub-expression in the tree, using interval arithmetics.



(a) upward propagation (b) downward propagation

Figure 2: HC4Revise

Figure 2(b) shows the second step, the downward traversal of the tree that computes the projection over every sub-expression. Actually, a node is linked to its child nodes by a primitive constraint, and this step simply consists in applying cascading projections of primitive constraints.

Finally, we notice that HC4Revise has reduced the domain of x from $[0, 10]$ to $[0, 8]$.

Except with primitive constraints, HC4Revise does not compute an exact projection, as defined in Section 2, but a conservative approximation. Indeed, when a disjunction (i.e., a union of intervals) occurs during the second step, the hull is computed so that inconsistent values are introduced in place of gaps. In our example, the exact projection of $[9, 16]$ over the node labeled with "–" is $\{[-4, -3], [3,\ 4]\}$, and only the hull $[-4, 4]$ is kept.

Note that before handling a constraint c, HC4Revise sets the interval-convexity boolean of c to true. If a disjunction appears somewhere in the tree, this boolean is set to false.

3.3 TAC

The algorithm TAC (**T**ree **A**rc **C**onsistency) computes an exact [1] projection with simple NCSPs. It is the union-variant of HC4Revise. The exactly same backward propagation idea is used, except that interval unions are stored in place of intervals. This variant has already been evoked in [3].

[1]This term is employed regardless of the rounding due to the machine representation of reals (not considered here).

We preclude from our discussion the particular case of trigonometric functions with an infinite number of acceptable periods, such as $sin(1/x)$ with $0 \in D_x$. We will see later a way to tackle such constraints.

TAC is exactly HC4Revise except that unions of intervals are authorized. In our example, the node labeled with "-" is assigned a union of intervals : $\{[-4, -3], [3,\ 4]\}$. Projecting this union over x leads to a union for x, as shown in Figure 3. Note that a gap cannot appear with primitive constraints in the left column of Figure 1 (functions are monotonous), as opposed to the constraints in the right column. Note also that, except with p_7, only the downward propagation requires union labeling.

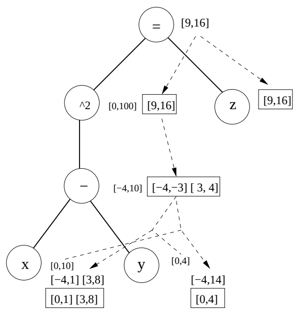


Figure 3: Downward step of TAC

By limiting the scope of propagation to a single constraint, an infinite loop as described in [6] cannot occur. Indeed, a necessary condition for such a loop is the presence of cycles in the constraint network. Even more, the label size (i.e., the number of intervals used to represent domains) is bounded, this bound being inherent to the problem and easily computable (see [7]).

Remark: We also use TAC in presence of multiple variable occurrences in a same constraint. Each occurrence is treated as a different and independent variable. Of course, exactness of the output is no more guaranteed. In Section 4, we explain what we can expect from TAC in this case.

3.4 Dealing with infinity of disjunctions

In this case, we need to introduce a bound for the label size, say MAX. If at some point the label size of a projection result exceeds MAX, we simply merge some of the intervals. The projection is not exact any longer, but as long as two intervals are isolated, the other ones can be merged and their split is just postponed. If the system has a finite number of solutions, the process will end. This implementation trick prevents the solver from failing in case of overflow. In all our benchmarks, we never observed an overflow with a bound set to 10.

4. THEORETICAL PROPERTIES

In this section, we clarify the properties of our algorithm because projections are not exact in all the situations. We start from primitive constraints, go on with simple NCSPs and finally deal with the general case.

PROPOSITION 4.1. *Let P be a NCSP.*
If constraints in P are all primitive, then HC4+TAC *enforces the box-set consistency of P.*

PROOF. It is shown in [6] that Lazy_BoxSet solves the box-set consistency of a NCSP. Then, since both TAC and HC4Revise compute an exact projection of a primitive constraint, the box-set consistency is ensured by HC4+TAC. □

4.1 Simple NCSPs

The previous result can be extended to systems with arbitrary complex expressions, assuming that variables appear at most once inside a same constraint.

PROPOSITION 4.2. *Let P be a simple NCSP.* HC4+TAC *enforces the box-set consistency of P.*

The proof is not as straightforward as for the proposition 4.1 because HC4Revise does not always compute exact projections (see 4.1.3), and consequently hull consistency is not guaranteed by HC4 anymore. But fortunately, the situations where HC4 fails to achieve hull consistency are intermediate steps of the general algorithm, so that this limitation does not prevent HC4+TAC from giving the box-set consistency.

First, we are going to show that in this case TAC still computes exact projections.

4.1.1 Exactness of TAC

Let c be an arbitrary constraint. We call *decomposition* of c the sub-system of primitive constraints equivalent to c.

EXAMPLE 4.1 (DECOMPOSITION). $c : (x-y)^2 = z$ *is equivalent to a sub-system S of 3 primitive constraints relating 5 variables (x, y, z, w, t):*

$$c : (x-y)^2 = z \Longleftrightarrow (S) \begin{cases} p_3(x, y, w) \\ p_8(w, t, 2) \\ p_1(t, z) \end{cases} \qquad (1)$$

PROPOSITION 4.3. *Let c be a simple constraint.* TAC *computes exact projections of c.*

PROOF. Let S be the decomposition of c.

1- The constraint network of S is a tree (the same as the syntax tree of c).

2- Each time a union is computed for a node in TAC, there is an equivalent projection in S over the implicit variable (like w and t in example 4.1) representing this node. Initial domains of implicit variables are $]-\infty, +\infty[$.

3- The projection of a constraint in S over a given variable is exact (proposition 4.1), i.e., every value in the resulting domain of this variable has a support in the domains of the other variables. Such projection is equivalent to a step of *directed arc consistency* in finite domain [8].

4- Faltings showed in [9] that a two-step *directed arc consistency filtering* applied on a tree-structured graph leads to an arc consistent labeling. Each step of TAC is equivalent to a step of directed arc consistency, from the leaves of the tree (the variables of c) up to the root and in the other way around.

5- arc consistency of a tree-structured graph is equivalent to global consistency [10]. So TAC gives the global consistency of S. In other words, every value in the domain of a variable in c belongs to a solution of S, i.e., satisfies the constraint c, since c and S are semantically equivalent. Therefore, global consistency of S is equivalent to arc consistency of c. This justifies the name (**T**ree **A**rc-**C**onsistency) of this method. Hence, this two-step process computes exact projections over all the involved variables.

□

4.1.2 Proof of proposition 4.2

Consider the leaves in the search tree of HC4+TAC.

Leaves are arc consistent: If no disjunction occurs during a call to HC4Revise, no intermediate hull is computed and HC4Revise behaves rigorously like TAC. Assume a disjunction occurs during a call to HC4Revise. The constraint is marked as "not interval-convex", and it will be projected by TAC. But TAC cannot perform any reduction with this constraint, since we are dealing with a leaf (any reduction entails a subsequent call to HC4+TAC). Therefore, any call to HC4Revise is necessarily equivalent to a projection by TAC. Since we have proven in 4.1.1 that TAC is exact, once the fixpoint is reached, arc consistency is achieved.

Leaves are maximal: Let B be the initial box. Let B' be the box obtained by HC4+TAC just before the first natural split, and B_1,...,B_n the child boxes obtained just after. It is clear that the box-set consistency of B is also the box-set consistency of B'. We need to prove that the box-set consistency of B' is the union of the box-set consistencies of B_1,...,B_n. Assume that it is not. We can find a box X in the box-set consistency of a child box, for instance B_1, that is not maximal. Therefore X can be enlarged, i.e., an "adherent" value can be added in the domain of a variable (a value stuck to X). As B' and B_1 differ only in the domain of the split variable (say x), by a simple reasoning on propagation, the added value requires an extra value in the domain of x. But D_x cannot be enlarged, because an adherent value of D_x would be inside the gap, and necessarily inconsistent. Hence, by a straightforward induction, the box-set consistency of the initial box is the union of the box-set consistencies of the leaves. We have seen that leaves are arc consistent, so they are maximal.

4.1.3 Remark: The diffi culty with HC4

HC4Revise does not always compute an exact projection, because of intermediate approximations while projecting over implicit variables. The consequence is that HC4 does not enforce the hull consistency of the system anymore, but only the hull consistency of the decomposition of the system [3].

EXAMPLE 4.2. *Let P be the following CSP:*
$D_x = [0, 1]$, $D_y = [-1, 1]$
$(x \times y)^2 = 1$
Let P' be the decomposition of P, i.e. :
$D_x = [0, 1]$, $D_y = [-1, 1]$, $D_w =]-\infty, +\infty[$
$x \times y = w$
$w^2 = 1$

P is not hull consistent because the bound 0 for x has no support in y, and it is easy to check that HC4 cannot reduce this bound. Indeed, the projection over w (the subexpression $x \times y$) results in the interval $[-1, 1]$ that includes the bad support $w = 0$ for $x = 0$. On the other hand, hull consistency of P' is achieved with an *extension* of B, i.e., a box $B' = B \times D'_w$. This extension is $[0, 1] \times [-1, 1] \times [-1, 1]$.

So we could have expected HC4+TAC to enforce the box-set consistency of the decomposition of a problem. This cannot be true because arc consistency of a system is a weaker property than arc consistency of its decomposition.

EXAMPLE 4.3. *Let P be the following CSP:*
$D_x = [0, 4]$, $D_y = [0, 4]$
$(x - y)^2 = 4$
Let P' be the decomposition of P, i.e. :
$D_x = [0, 4]$, $D_y = [0, 4]$, $D_w =] - \infty, +\infty[$
$x - y = w$
$w^2 = 4$

P is arc consistent. But arc consistency of P' cannot be achieved with an extension of B, i.e., with $D_x = [0,4]$ and $D_y = [0,4]$. Indeed, the box-set consistency of P' includes two arc consistent boxes : $\{[0,2] \times [2,4] \times [-2,-2]$ and $[2,4] \times [0,2] \times [2,2]\}$. This is not a big surprise since, by decomposing a system, it becomes possible to split the domains of the implicit variables (w) and therefore, to obtain smaller boxes.

This hybrid situation can be summarized with this pseudo-formula:
(1) **hull consistency $\Longrightarrow$ hull consistency of the decomposition**
(2) **arc consistency $\Longleftarrow$ arc consistency of the decomposition**

See [11] for (1). The flip in the implications means *a priori* that a box returned by HC4+TAC has not been filtered enough to be hull consistent (regarding P), and not been split enough to be arc consistent (regarding P'). However, we have seen how to fix the problem with a weaker split condition (line 15 of HC4+TAC).

4.2 General Case

To describe the property of HC4+TAC in this case, follows the definition of another kind of transformation called *renaming* [11], where each occurrence of the same variable is substituted by a new variable [2].

EXAMPLE 4.4 (RENAMING).

$$c : (x-y)^2 - x = 0 \Longleftrightarrow (S) \begin{cases} (x-y)^2 - z = 0 \\ z = x \end{cases} \qquad (2)$$

x and z in (S) are the aliases of x in c.

Note that renaming produces an intermediate system between the original one and the decomposition, in terms of size.

PROPOSITION 4.4. *Let P be a NCSP. Let P' be the renaming of P*
HC4+TAC *applied on P enforces the box-set consistency of P'.*

Thus, the box-set consistency cannot be enforced by HC4+TAC in the general case.

PROOF. With multiple occurrences of variables, HC4+TAC behaves as if it was applied on the renaming of the system. For instance, the first operation made by HC4Revise and TAC with a constraint is to assign to the aliases of a same variable x the domain of x. So, if aliases of x are noted $z_1, ..., z_n$, the first step of these operators is equivalent to a projection of the non-root constraints $z_i = x$ over the aliases z_i in the renaming. One precaution however with multiple occurrences : we must push back the current constraint in the propagation queue after a call to HC4Revise.

So, HC4+TAC provides the same result as if it was applied on the renaming of the problem, which is a simple NCSP. But we know that HC4+TAC applied on a simple problem enforces box-set consistency (proposition 4.2), so HC4+TAC enforces the box-set consistency of the renaming. □

5. PRACTICAL TIME COMPLEXITY

In practice, box-set consistency can be used to find solutions of a NCSP. Instead of collecting the resulting boxes (see line 23 in HC4+TAC), we use them in a depth-first process as new choice points. In this way, the advantage of box-set consistency is twofold: First, the property itself may be exploited, and this approach is still under research. Second, as natural splitting is driven by the semantics of the problem, it is sharper and often more efficient than other bisection heuristics (e.g., round-robin or largest-domain first).

On 20 problems found in [12] and ALIAS solver homepage[3], We have compared HC4 and bisection, with HC4+TAC and bisection. In 15 of them, computation times are equivalent; in 3 others HC4+TAC provides a gain up to 10%. A 20% gain was obtained on the Broyden banded function problem [12], and on the Sierpinski's distance equations problem [13], which include both 30 nonlinear equations. This ratio is maintained even by introducing higher order consistencies [14] in the solving process.

6. CONCLUSION

We have shown in practice how to enforce box-set consistency (hence arc consistency) in a lazy way, without spoiling the performances. The keystone of this implementation is an operator that manages gaps, and the number of calls to this costly operator is minimized by using a standard hull consistency algorithm as a prefiltering process.

The main contribution was to establish precisely the properties of the algorithm. We have proven that box-set consistency is obtained with simple NCSPs, and a relaxation in the general case.

Our future works include the possible integration of box-set consistency within existing methods, especially from interval analysis. Indeed, we suspect some techniques to converge faster with arc consistent boxes.

7. REFERENCES

[1] Benhamou, F., McAllester, D., Van Hentenryck, P.: Clp(intervals) revisited. In: International Symposium on Logic programming, MIT Press (1994) 124–138

[2] Lhomme, O.: Contribution à la résolution de contraintes sur les réels par propagation d'intervalles. Phd thesis, University of Nice-Sophia Antipolis (1994)

[3] Benhamou, F., Goualard, F., Granvilliers, L., Puget, J.F.: Revising hull and box consistency. In: International Conference on Logic Programming. (1999) 230–244

[4] Benhamou, F., Older, W.: Applying interval arithmetic to real, integer and boolean constraints. Journal of Logic Programming **32** (1997) 1–24

[5] Chabert, G., Trombettoni, G., Neveu, B.: New light on arc consistency over continous domains. Research report, Institut National de Recherche en Informatique et en Automatique (2004)

[6] Hyvönen, E.: Constraint reasoning based on interval arithmetic—The tolerance propagation approach. Artifi cial Intelligence **58** (1992) 71–112

[7] Dechter, R., Pearl, J.: Network-based heuristics for constraint satisfaction problems. Artifi cial Intelligence **34** (1987) 1–38

[8] Faltings, B.: Arc-consistency for continuous variables. Artifi cial Intelligence **65** (1994)

[9] Freuder, E.: A suffi cient condition for backtrack-free search. Journal of the ACM **29** (1982) 24–32

[10] Delobel, F., Collavizza, H., Rueher, M.: Comparing partial consistencies. In: Reliable Computing. Kluwer (1999) 213–228

[11] Moré, J., Garbow, B., Hillstrom, K.: Testing unconstrained optimization software. ACM Trans. Math. Softw. **7** (1981) 17–41

[12] Jermann, C., Trombettoni, G., Neveu, B.: Inter-block backtracking: Exploiting the structure in continuous csps. In: 2nd International Workshop on Global Constrained Optimization and Constraint Satisfaction (Cocos'03). (2003)

[13] Lhomme, O.: Consistency techniques for numeric csps. In: Proc. of the 13th IJCAI. (1993) 232–238

[2] It seems that no dedicated terms exist in literature to distinguish *renaming* from *decomposition*. Depending on the paper, *decomposition* may either refer to the first or the latter.

[3] http://www-sop.inria.fr/coprin/logiciels/ALIAS/Benches