Box Consistency through Weak Box Consistency

Laurent Granvilliers, Frédéric Goualard, Frédéric Benhamou
IRIN, Université de Nantes
B.P. 92208, F-44322 Nantes Cedex 3, France
{Laurent.Granvilliers,Frederic.Goualard,Frederic.Benhamou}@irin.univ-nantes.fr

Abstract

Interval constraint solvers use local consistencies—among which one worth mentioning is box consistency—for computing verified solutions of real constraint systems. Though among the most efficient ones, the algorithm for enforcing box consistency suffers from the use of time-consuming operators. This paper first introduces box$_\varphi$ consistency, a weakening of box consistency; this new notion then allows us to devise an adaptive algorithm that computes box consistency by enforcing box$_\varphi$ consistency, decreasing the φ parameter as variables' domains get tightened, then achieving eventually box$_0$ consistency, which is equivalent to box consistency. A new propagation algorithm is also given, that intensifies the use of the most contracting pruning functions based on box$_\varphi$ consistency. The resulting algorithm is finally shown to outperform the original scheme for enforcing box consistency on a set of standard benchmarks.

1. Introduction

The use of interval analysis [12] for solving continuous constraint systems in the framework of logic programming is a step to reconcile logic and numerical computations. The starting point is the seminal work of Cleary [3] who introduced a relational form of interval arithmetic. Originating from these ideas, various extensions have been proposed and implemented in CLP(Intervals) systems like CLP(BNR), Interlog, Newton, PrologIV and DecLIC for solving *interval constraints* [4, 7, 9, 2].

The constraint solving algorithm is basically an iteration of two steps: first, an interval-based pruning operator associated to each constraint of the system to be solved discards from the variables' domains some of the values that are inconsistent with the constraint (*local consistency* such as arc consistency [11]); the domain modifications are then propagated to the other constraints for reinvocation of their pruning operator. When quiescence is reached, a *bisection* stage occurs—splitting of the domains—to separate solutions and obtain tighter domains that could have been obtained by constraint propagation alone.

The finite precision of machine arithmetic prevents from achieving arc consistency over continuous domains. Hence, coarser consistencies are used in practice, among which one may cite box consistency [2]. An algorithm enforcing box consistency was first implemented in Newton [13] by a dichotomic search combined with an interval Newton method. The consistency is enforced over a constraint system by propagation (BC3 algorithm) that combines the local approximations from every constraint. Two recent works have discussed the convergence of such algorithms. On the one hand, the authors [5] have proposed to weaken the precision of the local approximations computed by BC3 with the consequence that the final domains may not be box consistent. On the other hand, Lhomme *et al.* [10] have defined a new propagation algorithm for 2B consistency which essentially intensifies the application of the most contracting pruning functions.

This paper is an attempt to combine both above-mentioned techniques for implementing box consistency. More precisely, a weakening of box consistency called *box$_\varphi$ consistency* is introduced. Both notions are very close; hence, the implementation of an algorithm for box$_\varphi$ consistency is straightforward. Box$_\varphi$ consistency then allows us to devise a new adaptive propagation algorithm BC$_\varphi$ for achieving box consistency that iterates two steps: a selection of the most contracting pruning functions enforcing box$_\varphi$ consistency; and their application in sequence without any propagation. The key point is that the algorithm increases the precision of the computed consistency by decreasing the φ parameter the more variables' domains are tightened (that is, φ is an estimate of the distance to the global fixed-point). Hence, the computed consistency is eventually box consistency. Preliminary results from a prototype show a significant speed-up with respect to BC3.

The rest of the paper is organized as follows. Section 2 introduces the materials from constraints and interval analysis. Section 3 presents the notion of box consistency and the framework of constraint propagation. Section 4 defines the notion of box$_\varphi$ consistency, its implementation and the new propagation strategy. Section 5 analyses the experimental results. Finally, Section 6 discusses some directions for further developments.

2. Preliminaries

Let $\mathbb{R}$ be the set of real numbers and $\mathbb{F} \subseteq \mathbb{R}$ a set of binary floating-point numbers [8] (shortened thereafter to floats). Given a real number r, let $\lfloor \mathsf{r} \rfloor$ (resp. $\lceil \mathsf{r} \rceil$) be the greatest float smaller or equal (resp. lowest float greater or equal) than r. Given a float g, let g^- be the greatest float smaller than g, and g^+ the lowest float greater than g.

2.1. Interval Analysis

An interval I is a set of real numbers $\{\mathsf{r} \in \mathbb{R} \mid \underline{I} \leqslant \mathsf{r} \leqslant \overline{I}\}$ written $[\underline{I}, \overline{I}]$ where bounds $\underline{I}$ and $\overline{I}$ are elements of $\mathbb{F}$. Let $\mathbb{I}$ be the set of all intervals. A Cartesian product $\boldsymbol{I}$ of intervals $I_1 \times \cdots \times I_n$ is called a *box*. Thereafter, boldfaced letters denote either a vector or a Cartesian product, depending on the context. A nonempty interval $[a, b]$ is said *canonical* whenever $b \leqslant a^+$. Given $I = [a, b]$ and $\mathsf{r} = (a + b)/2$, the *center* of I is $\lfloor \mathsf{r} \rfloor$, the *interval midpoint* of I is $[\lfloor \mathsf{r} \rfloor, \lceil \mathsf{r} \rceil]$ and the *width* of I is $\lceil b - a \rceil$. The *width* of the box $\boldsymbol{I}$ is the maximum of the widths componentwise. The *distance* between $I = [a, b]$ and $J = [c, d]$ is $\lceil \lceil |c - a| \rceil + \lceil |b - d| \rceil \rceil$. The *distance* between $\boldsymbol{I} \in \mathbb{I}^n$ and $\boldsymbol{J} \in \mathbb{I}^n$ is the sum of the distances componentwise. Let $\mathsf{Hull}(\rho)$ be the smallest interval containing the relation $\rho \subseteq \mathbb{R}$.

Definition 1 (Interval extensions). *An inclusion monotonic function* $F\colon \mathbb{I}^n \to \mathbb{I}$ *is an* interval extension *of* $f\colon \mathbb{R}^n \to \mathbb{R}$ *if for every* $\boldsymbol{I} \in \mathbb{I}^n$, $\mathsf{r} \in \boldsymbol{I} \Rightarrow f(\mathsf{r}) \in F(\boldsymbol{I})$.
A relation $\Omega \subseteq \mathbb{I}^n$ *is an* interval extension *of* $\rho \subseteq \mathbb{R}^n$ *if for every* $\boldsymbol{I} \in \mathbb{I}^n$, $\boldsymbol{I} \cap \rho \neq \varnothing \Rightarrow \boldsymbol{I} \in \Omega$.

A *quasi-zero* of an interval function $F : \mathbb{I} \to \mathbb{I}$ is a canonical interval I verifying $0 \in F(I)$. Given an interval function $F(X_1, \ldots, X_n)$, an integer $k \in \{1, \ldots, n\}$, and an interval vector $\boldsymbol{I} \in \mathbb{I}^n$, $F[k, \boldsymbol{I}]$ corresponds to the interval function $X_k \mapsto F(I_1, \ldots, I_{k-1}, X_k, I_{k+1}, \ldots, I_n)$.

The univariate interval Newton method [12] is an adaptation based on interval analysis of the Newton root finding method for real functions derived from the mean value theorem (see [6] for a more complete presentation).

Theorem 1 (Mean value). *Let* x, x_0 *be real numbers and* $f : \mathbb{R} \to \mathbb{R}$ *a function with continuous first order derivative. Then, there exists* $\xi \in \mathbb{R}$ *between* x *and* x_0 *such that* $f(x) - f(x_0) = f'(\xi)(x - x_0)$.

If x_0 is a zero of f, Theorem 1 leads to $x_0 = x - f(x)/f'(\xi)$. Let F (resp. F') be an interval extension of f (resp. f') and I be an interval containing both x and x_0 such that $x \in \mathsf{midpoint}(I)$ and $0 \notin F'(I)$. Since ξ is in I, it follows $f'(\xi) \in F'(I)$, and then $x_0 \in N(I)$, where $N(I) = \mathsf{midpoint}(I) - F(\mathsf{midpoint}(I))/F'(I)$. By hypothesis, x_0 is in I; consequently $x_0 \in I \cap N(I)$. One may then devise an algorithm for finding a zero of f contained in an interval I^0 by applying the step $I^{i+1} := I^i \cap N(I^i)$ from I^0. Iterations converge since $\mathbb{I}$ is finite and each step is contracting. Let $N^*(F, F', I^0)$ be the fixed-point of this iterating process.

2.2. Constraints

A *constraint* is a first-order atomic formula over the structure of reals $\langle \mathbb{R}, \mathcal{O}, \{\leqslant, =, \geqslant\}\rangle$—where $\mathcal{O}$ is a set of operation symbols—and the set of variables $\{x_1, \ldots, x_n\}$. Given a constraint $c(x_{k1}, \ldots, x_{kl})$, let $\rho(c) = \{(a_1, \ldots, a_n) \in \mathbb{R}^n \mid \exists\, (a_{k1}, \ldots, a_{kl}) \in c\}$ be the relation obtained by an operation of cylindrification. A *constraint system* is a couple $C = \langle \mathcal{C}, \boldsymbol{I}\rangle$ made of a set of constraints $\mathcal{C} = \{c_1, \ldots, c_m\}$ and an interval vector $\boldsymbol{I}$ where I_k is the domain of possible values for x_k. The *solution set* of C is $\boldsymbol{I} \cap \rho(c_1) \cap \cdots \cap \rho(c_m)$.

3. Box consistency

Box consistency [2] is an approximation of arc consistency over intervals that is efficiently implemented by means of interval Newton methods in Newton [13] and Numerica [14]. It is defined in terms of quasi-zeros of some interval extensions of real relations.

Definition 2 (Box consistency). *A relation* $\Omega \subseteq \mathbb{I}^n$ *is* box consistent *in some* $k \in \{1, \ldots, n\}$ *and* $\boldsymbol{I} \in \mathbb{I}^n$ *if*

$$I_k = \mathsf{Hull}(\{a \in I_k \mid (I_1, \ldots, I_{k-1}, \mathsf{Hull}(\{a\}), I_{k+1}, \ldots, I_n) \in \Omega\})$$

As described in [1], solving a constraint system consists in iteratively applying some contracting functions called narrowing operators. A *narrowing operator* for a relation $\rho \subseteq \mathbb{R}^n$ is a contracting and monotonic function $\mathsf{nar} : \mathbb{I}^n \to \mathbb{I}^n$ such that $\boldsymbol{I} \cap \rho \subseteq \mathsf{nar}(\boldsymbol{I})$ for every $\boldsymbol{I} \in \mathbb{I}^n$ (correctness). Furthermore, a narrowing operator is said to be *univariate of rank* k if $\boldsymbol{I}$ and $\mathsf{nar}(\boldsymbol{I})$ only differ in the k-th component for every $\boldsymbol{I} \in \mathbb{I}^n$. A narrowing operator nar for the relation $\rho(c)$ of a constraint $c(x_{k1}, \ldots, x_{kl})$ is said *to depend on* the integers $k1, \ldots, kl$. *BC-narrowing operators* are narrowing operators devoted to the computation of box consistency.

Definition 3 (BC-narrowing operator). *Let* $\rho \subseteq \mathbb{R}^n$ *be a relation and* $\Omega \subseteq \mathbb{I}^n$ *an interval extension of* ρ. *A* BC-narrowing operator for Ω *is a univariate narrowing operator* nar *of rank* $k \in \{1, \ldots, n\}$ *for* ρ *that associates to*

Table 1. Box consistency algorithm BC3.

```
BC3 (𝒩, N : sets of BC-narrowing operators; I : 𝕀^n) : 𝕀^n
        % N: set of operators to re-apply
begin
  if N = ∅ or I = ∅ then return I
  else
    nar := one element of N  % selection
    k := rank of nar
    J := nar (I)  % contraction
    if J_k ≠ I_k then  % propagation
      M := N ∪ {nar′ ∈ 𝒩 | nar′ depends on k}
      % nar can be removed from M if it is idempotent
    else M := N \ {nar}  % nar(I) = I
    endif
    return BC3 (𝒩, M, J)
  endif
end
```

every $\boldsymbol{I} \in \mathbb{I}^n$ *the greatest sub-box* $\boldsymbol{J}$ *of* $\boldsymbol{I}$ *such that* Ω *is box consistent in* k *and* $\boldsymbol{J}$*. The vector* $\mathsf{nar}(\boldsymbol{I})$ *is denoted* $BC\,(k,\Omega,\boldsymbol{I})$.

Note that $BC\,(k,\Omega,\boldsymbol{I})$ is independent from the actual implementation of BC-narrowing operators. It characterizes the approximation computed by any narrowing operator that implements box consistency with respect to Ω and k. This is related to the notion of *maximal box consistency* in the literature.

Solving a constraint system $\langle \mathcal{C}, \boldsymbol{I}\rangle$ consists in two consecutive stages. A *compilation stage* creates a set of BC-narrowing operators $\mathcal{N}$ as follows: for every constraint $c \in \mathcal{C}$, for every interval extension[1] Ω of $\rho(c)$ and for every variable $x_k \in c$, the set $\mathcal{N}$ contains a BC-narrowing operator of rank k for Ω. The propagation algorithm BC3 described in Table 1 combines the approximations computed by all the narrowing operators from $\mathcal{N}$ until none of them is able to further tighten the domains. The final vector of domains is the greatest common fixed-point of the narrowing operators included in the initial domains.

The main feature of BC3 is to maintain the set $\mathcal{N} \setminus N$ of narrowing operators for which the current domains are necessarily a fixed-point. This process is easily implemented since the relation "*depends on*" is static and can be created at compile-time. However the updates of structures after each application of a narrowing operator are time-consuming (see [10]).

[1] For example the natural interval extension or the Taylor interval extension [14].

4. Box$_\varphi$ consistency

Box consistency effectiveness comes from its ability to locally cancel the so-called *dependency problem* of interval arithmetic [12]. However, its actual implementation suffers from some drawbacks: first, applying a BC-narrowing operator is computationally expensive and leads to the unnecessary computation of a local fixed-point per operator; second, BC-narrowing operators do not ensure in practice the same amount of domain tightening. We address these problems by first defining *box*$_\varphi$ *consistency*, a weakening of box consistency. Box$_\varphi$ consistency then allows us to devise a new propagation algorithm, called BC$_\varphi$, for computing box consistency.

4.1. Definitions

Box consistency ensures a property on canonical intervals at bounds of the variables' domains. In contrast, box$_\varphi$ consistency imposes a weaker condition by replacing the canonical intervals by intervals of width φ. Intuitively this corresponds to the replacement in Definition 2 of the hull of a (best possible approximation) by an interval included in I_k whose width is smaller than φ.

Definition 4 (box$_\varphi$ consistency). *Let* $\varphi \in \mathbb{R}^+$ *be a positive real number. A relation* $\Omega \subseteq \mathbb{I}^n$ *is* box$_\varphi$ consistent in some $k \in \{1,\ldots,n\}$ and $\boldsymbol{I} \in \mathbb{I}^n$ *if*

$$I_k = \mathsf{Hull}(\{a \in I_k \mid (I_1,\ldots,I_{k-1}, I_k \cap \mathsf{Hull}(\{a, a+\varphi\}), I_{k+1},\ldots,I_n) \in \Omega\})$$

Note that box consistency is equivalent to box$_\varphi$ consistency with $\varphi = 0$ since then $a+\varphi = a$ and $I_k \cap \mathsf{Hull}(\{a\}) = \mathsf{Hull}(\{a\})$ when $a \in I_k$. In practice box$_\varphi$ consistency is verified over the bounds of $I_k = [a,b]$ if the relation Ω contains $(I_1,\ldots,I_{k-1}, I_k \cap [a, \lceil a+\varphi\rceil^+], I_{k+1},\ldots,I_n)$ and $(I_1,\ldots,I_{k-1}, I_k \cap [\lfloor b-\varphi\rfloor^-, b], I_{k+1},\ldots,I_n)$.

Definition 5 (BC$_\varphi$-narrowing operator). *Let* $\rho \subseteq \mathbb{R}^n$ *be a relation and* $\Omega \subseteq \mathbb{I}^n$ *an interval extension of* ρ*. Given* $\varphi \in \mathbb{R}^+$*, a* BC$_\varphi$-narrowing operator for Ω *is a univariate narrowing operator* nar *of rank* $k \in \{1,\ldots,n\}$ *for* ρ *associating to every* $\boldsymbol{I} \in \mathbb{I}^n$ *a vector* $\boldsymbol{J} \subseteq \boldsymbol{I}$ *such that* Ω *is box*$_\varphi$ *consistent in* k *and* $\boldsymbol{J}$ *and* $BC\,(k,\Omega,\boldsymbol{I}) \subseteq \boldsymbol{J}$ *(weak box consistency).*

Unlike BC-narrowing operators, the implementation of BC$_\varphi$-narrowing operators may influence the computed approximations since the previous definition may be verified by a set of vectors $\boldsymbol{J}$. In order to model the possible variations of φ during propagation over a set of BC$_\varphi$-narrowing operators, we define the notion of *BC*Φ*-narrowing operator* which corresponds to a family of BC$_\varphi$-narrowing operators parameterized by φ.

Definition 6 (BCΦ-narrowing operator). *Let* $\Omega \subseteq \mathbb{I}^n$ *be a relation. A* BCΦ-narrowing operator *for* Ω *is a function* $\mathsf{nar}\colon \mathbb{R}^+ \times \mathbb{I}^n \to \mathbb{I}^n$ *such that for every* $\varphi \in \mathbb{R}^+$ *the function* $\boldsymbol{I} \mapsto \mathsf{nar}(\varphi, \boldsymbol{I})$ *is a* BC_φ*-narrowing operator for* Ω *and for every* $\varphi \geqslant \varphi'$, $\mathsf{nar}(\varphi', \boldsymbol{I}) \subseteq \mathsf{nar}(\varphi, \boldsymbol{I})$ *for every* $\boldsymbol{I} \in \mathbb{I}^n$.

It is worthwhile noting the links between the different kinds of narrowing operators: given a BCΦ-narrowing operator nar for $\Omega \subseteq \mathbb{I}^n$, the function $\boldsymbol{I} \mapsto \mathsf{nar}(0, \boldsymbol{I})$ is a BC_φ-narrowing operator for Ω with $\varphi = 0$, and a BC-narrowing operator for Ω.

4.2. Implementation of BC_φ-narrowing Operators

BC_φ-narrowing operators are implemented like BC-narrowing operators [2] except that the dichotomic search for quasi-zeros is stopped when the distance between the quasi-zero and one bound of the corresponding domain is smaller than φ. For the sake of clarity, we only consider thereafter the case of equational constraints of the form $f(\boldsymbol{x}) = 0$.

The function Narrowφ described in Table 2 is an implementation of BCΦ-narrowing operators. Given a real function f, an interval extension F of f, a variable x_k in f, an interval extension F' of $\partial f / \partial x_k$ and a box $\boldsymbol{I}$, the aim is to enclose the external (leftmost and rightmost) quasi-zeros in I_k of the interval function $F[k, \boldsymbol{I}]$. The new domain is then the hull of both quasi-zeros that verify the constraint $F[k, \boldsymbol{I}](X_k) = 0$. During the search for the leftmost (resp. rightmost) quasi-zero using the function Lnarrowφ (resp. Rnarrowφ) four cases are considered: if $0 \notin F(\boldsymbol{I})$ then $\boldsymbol{I}$ is declared inconsistent; if $0 \notin F'(\boldsymbol{I})$ then I_k contains at most one quasi-zero of $F[k, \boldsymbol{I}]$ in which case the Newton method is applied; if $0 \in F'(\boldsymbol{I})$ then I_k is returned if its left bound (resp. right bound) is box_φ consistent; otherwise, I_k is split into two parts and the process is iterated from the leftmost (resp. rightmost) sub-domain (dichotomic search for the quasi-zeros).

Figure 1 describes the process for a real function in an initial interval $[a, b]$: a BC-narrowing operator will compute $[c, d]$; a BC_φ-narrowing operator will compute $[c', d']$ providing the width of both light-grey boxes is smaller than φ.

Proposition 1. *Given* F, F' *two interval functions and* $k \in \{1, \dots, n\}$ *an integer, the function* $\mathsf{nar}\colon \varphi, \boldsymbol{I} \mapsto \mathsf{Narrow}\varphi(F, F', k, \varphi, \boldsymbol{I})$ *is a* k*-rank BCΦ-narrowing operator.*

Proof 1. *In [5] the function* $\boldsymbol{I} \mapsto \mathsf{Narrow}\varphi(F, F', k, \varphi, \boldsymbol{I})$ *is shown to be a* BC_φ*-narrowing operator of rank* k*. The second property (generalization for BCΦ-narrowing operators) follows from the monotonicity of interval extensions. Given* $\varphi \geqslant \varphi'$*, both computations with* φ *and* φ'

Table 2. Implementation of BCΦ-narrowing operators.

```
Narrowφ (F, F' : 𝕀^n → 𝕀; k: index of variable;
          φ : ℝ+; I : 𝕀^n) : 𝕀^n
begin
  if 0 ∉ F(I) then return ∅
  elif 0 ∉ F'(I) then
    return (I_1, ..., I_{k-1},
            N*(F[k,I], F'[k,I], I_k), I_{k+1}, ..., I_n)
  else
    J := Lnarrowφ (F, F', k, φ, I)
    K := Rnarrowφ (F, F', k, φ, I)
    return (I_1, ..., I_{k-1}, [lower(J), upper(K)], I_{k+1}, ..., I_n)
  endif
end
```

```
Lnarrowφ (F, F' : 𝕀^n → 𝕀; k: index of variable;
          φ : ℝ+; I : 𝕀^n) : 𝕀
begin
  if 0 ∉ F(I) then return ∅  % inconsistency
  elif 0 ∉ F'(I) then
    return N*(F[k,I], F'[k,I], I_k)  % box consistency
  else
    J := I_k ∩ [lower(I_k), ⌈lower(I_k) + φ⌉+]
    if 0 ∈ F[k,I](J) then return I_k  % box_φ consistency
    else
      I_k^l := [lower(I_k), center(I_k)]  % dichotomic search
      I_k^r := [center(I_k), upper(I_k)]
      K := Lnarrowφ (F, F', k, φ, (I_1, ..., I_{k-1}, I_k^l,
                                   I_{k+1}, ..., I_n))
      if K ≠ ∅ then return K
      else return Lnarrowφ (F, F', k, φ, (I_1, ..., I_{k-1},
                                           I_k^r, I_{k+1}, ..., I_n))
      endif
    endif
  endif
end
```

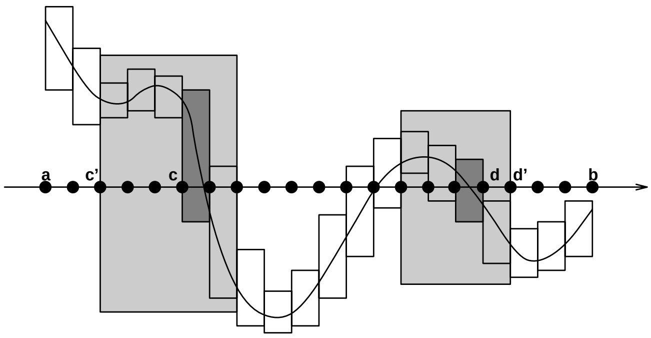

Figure 1. Searching for the external zeros of a real function in $[a, b]$.

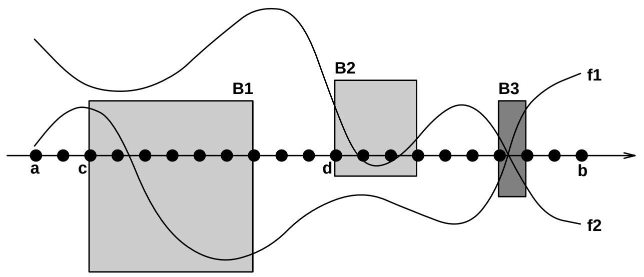

Figure 2. Searching for the common leftmost zero of two real functions in $[a, b]$.

only differ in the third test in Lnarrowφ. *Let* $J = I_k \cap [\underline{I_k}, \lceil \underline{I_k} + \varphi' \rceil^+]$ *and* $K = I_k \cap [\underline{I_k}, \lceil \underline{I_k} + \varphi \rceil^+] \supseteq J$. *Since* $F[k, \boldsymbol{I}](J) \subseteq F[k, \boldsymbol{I}](K)$ *by monotonicity of* F*, we have* $0 \in F[k, \boldsymbol{I}](J) \Rightarrow 0 \in F[k, \boldsymbol{I}](K)$. *The inclusion* Narrow$\varphi(F, F', k, \varphi', \boldsymbol{I}) \subseteq$ Narrow$\varphi(F, F', k, \varphi, \boldsymbol{I})$ *is then guaranteed, which ends the proof (the proof for* Rnarrowφ *is similar).*

The superiority of box$_\varphi$ consistency with respect to box consistency is illustrated in Figure 2 that describes the search for the leftmost common zero of two real functions in the interval $[a, b]$. The idea is to enclose a local zero by an interval of width $\varphi > 0$ (the box B1 is first computed from $[a, b]$ using f1, and then the box B2 from $[c, d]$ using f2). However, a common zero (in Box B3 computed from $[d, b]$ using f1) must be more tightly approximated. This remark motivates the introduction of the propagation algorithm presented in the next section.

Table 3. Box consistency algorithm BC$_\varphi$.

BC$_\varphi$ $(\mathcal{N}, N$: sets of BCΦ-narrowing operators; $\boldsymbol{I} : \mathbb{I}^n) : \mathbb{I}^n$
begin
  Snar := $\{$nar $\in N \mid$ nar$(0, \boldsymbol{I}) \neq \boldsymbol{I}\}$ % *first selection*
  $\boldsymbol{J}$:= RemoveBounds (*Snar*, $\boldsymbol{I}$)
  if *Snar* $= \varnothing$ **then**
    return $\boldsymbol{I}$ % *box consistency*
  else
    Index := $\{k \mid \exists$ nar $\in$ *Snar* $\wedge$ k is the rank of nar$\}$
    Snar' := Select a subset of *Snar* % *end of selection*
    $\boldsymbol{K}$:= Contractφ (*Snar'*, $\boldsymbol{J}$) % *intensification*
    if $\boldsymbol{K} = \varnothing$ **then**
      return $\varnothing$ % *inconsistency*
    else % *propagation*
      M := $\{$nar $\in \mathcal{N} \mid \exists k \in$ *Index* $\wedge$ nar depends on $k\}$
      return BC$_\varphi$ $(\mathcal{N}, M, \boldsymbol{K})$
    endif
  endif
end

4.3. A New Propagation Algorithm for Box Consistency

As said previously, the application of BC-narrowing operators is time-consuming while the precision they provide (enclosure of extreme quasi-zeros by canonical intervals) is often useless. We then propose a new propagation strategy for box consistency implemented by Algorithm BC$_\varphi$ described in Table 3. The main idea is to realize a selection among an input set of BCΦ-narrowing operators and to apply the selected ones without updating the internal structures. Furthermore, the value of φ decreases with respect to the distance from the current domains to the global fixed-point. More precisely the algorithm consists in an iteration of three consecutive stages:

Selection. The set *Snar* is the subset of narrowing operators in N for which $\boldsymbol{I}$ is not a fixed-point. If *Snar* is empty then $\boldsymbol{I}$ is a fixed-point of every nar $\in \mathcal{N}$ with $\varphi = 0$, which guarantees the computation of box consistency, and BC$_\varphi$ terminates. The condition nar$(0, \boldsymbol{I}) = \boldsymbol{I}$ can be efficiently tested since it is sufficient verifying that the canonical intervals at bounds of domains are quasi-zeros of the corresponding interval function. These canonical intervals are removed from $\boldsymbol{I}$ if the condition is not verified. The result is $\boldsymbol{J}$, computed by the function RemoveBounds.
The set *Index* contains all the indices of the variables whose domain is contracted by the application of one narrowing operator from *Snar*, i.e. the set of ranks of all the operators nar in *Snar*.

The set *Snar'* computed by Select contains one k-rank narrowing operator nar $\in$ *Snar* for every $k \in$ *Index*. The idea is to select the most contracting narrowing operator for each variable domain which is tightened. The contractance of an operator nar $\in$ *Snar* is the distance between $\boldsymbol{I}$ and $\mathsf{nar}(0, \boldsymbol{I})$;

Intensification. The aim of Contractφ is to apply in sequence—without updating internal structures—the selected narrowing operators from *Snar'*. No condition is required to stop this stage; in particular, the computation of a fixed-point is not mandatory. In practice, an efficient heuristic is to remove from *Snar'* every narrowing operator whose two last applications left the domains unchanged. In this case the end of the stage is detected when either *Snar'* or the domains are empty. BC_φ stops whenever the domains become empty.
The set *Snar'* is sorted into descending order of the weights of the narrowing operators it contains, where the *weight* of a k-rank narrowing operator is the number of narrowing operators in *Snar'* that depend on k. The aim is to maximize the size of contractions resulting from the interaction between narrowing operators. Intuitively, the application of a k-rank narrowing operator before the application of a j-rank narrowing operator not depending on k does not lead to more contraction of the domain of x_j;

Propagation. A property of both previous stages is that every narrowing operator nar $\in \mathcal{N}$ that does not depend on any k in *Index* is such that $\mathsf{nar}(0, \boldsymbol{K}) = \boldsymbol{K}$. The result is that M (next set of narrowing operators to consider) is composed of all narrowing operators from $\mathcal{N}$ depending on a k in *Index*. Hence, the propagation is independent from the intensification stage and the Select implementation.
An efficient implementation minimizing the costs of updates is to create first the list of all the constraints containing a variable x_k with $k \in$ *Index*. If one associates to each constraint the array of all the narrowing operators created from it, the set M may then be built by concatenating[2] all the arrays.

An important point is the decreasing of the φ parameter during the intensification stage. This parameter is an estimate of the distance from the current domains to the final domains. Intuitively, the precision of the approximations computed by the BC_φ-narrowing operators must increase when the current domains get closer to the global fixed-point. In other words, the value of φ must decrease when the widths of the last domains contractions decrease (see Figure 2). A method for computing the value of φ is as follows: to associate to each k-rank narrowing operator nar $\in$ *Snar'* (operator used in the intensification stage) the size w of the contraction of the domain of the variable x_k at the time of its last application. More precisely the initial value is $w = \mathsf{width}(I_k)$. After each application $\boldsymbol{J} := \mathsf{nar}(\boldsymbol{I})$, it becomes $w = \mathit{distance}(I_k, J_k)$. The value of φ is dynamically computed as $\max(0, \sqrt{W} - 0.125)$ where W is the average of the values w associated to all the narrowing operators in *Snar'*. Doing so, φ decreases when W decreases and it becomes 0 when $\sqrt{W}$ is smaller than 0.125, which prevents some slow convergence phenomena by computing box consistency (best possible approximation) when the global fixed-point is almost found.

[2] This operation is efficiently implemented in C by memcpy().

The following propositions state that BC_φ computes box consistency and may then replace BC3.

Proposition 2. $\mathsf{BC}_\varphi(\mathcal{N}, \mathcal{N}, \boldsymbol{I})$ *terminates for every set of BCΦ-narrowing operators $\mathcal{N}$ and every interval vector $\boldsymbol{I}$.*

Proof 2. *The current domains are returned whenever Snar is empty. If Snar is not empty, the application of* RemoveBounds *strictly contracts $\boldsymbol{I}$. Then BC_φ terminates since every non-terminal step is contracting and the search space is finite.*

Proposition 3. *Let $\mathcal{N}$ be a set of BCΦ-narrowing operators and $\mathcal{N}'$ the set of BC-narrowing operators obtained from $\mathcal{N}$ by replacing every function* nar *by $\boldsymbol{I} \mapsto \mathsf{nar}(0, \boldsymbol{I})$. Then for every $\boldsymbol{I} \in \mathbb{I}^n$, $\mathsf{BC}_\varphi(\mathcal{N}, \mathcal{N}, \boldsymbol{I})$ is equivalent to* $\mathsf{BC3}(\mathcal{N}', \mathcal{N}', \boldsymbol{I})$.

Proof 3. *The final vector of domains from BC_φ is a fixed-point of every function $\boldsymbol{I} \mapsto \mathsf{nar}(0, \boldsymbol{I})$ (narrowing operators used in* BC3*) which is the condition tested during the selection stage. The proof then follows since every intermediate approximation is necessarily included in the greatest common fixed-point included in the initial vector of domains.*

4.4. Bisection and Minimal Propagation

Constraint propagation algorithms tighten the domains of variables involved in constraint systems. In order to separate the solutions, they are generally embedded in a bisection algorithm that iteratively prunes by propagation the domains split into sub-domains if the required precision is not yet reached.

We present the constraint solving algorithm Bisect in Table 4 that uses the propagation algorithm BC_φ. The first call is $\mathsf{Bisect}(\mathcal{N}, \{1, \ldots, n\}, \varepsilon, \boldsymbol{I})$ where ε is the required precision of domains. All other calls after the bisection of the domain of the variable x_k initialize the propagation with only the narrowing operators depending on k. The function Divide splits J_k (in practice the component of $\boldsymbol{J}$ of greatest width) in l parts (in practice two or three parts) such that the width of J_k is greater than ε and $\boldsymbol{J} = \boldsymbol{J}_1 \cup \cdots \cup \boldsymbol{J}_l$. The

Table 4. The constraint solving algorithm.

Bisect ($\mathcal{N}$: set of BCΦ-narrowing operators;
    S : set of indices of variables;
    $\varepsilon : \mathbb{R}^+$; $\boldsymbol{I} : \mathbb{I}^n$) : set of $\mathbb{I}^n$
begin
  $\boldsymbol{J}$:= $\mathsf{BC}_\varphi(\mathcal{N}, \{\text{nar} \in \mathcal{N} \mid \exists k \in S \wedge \text{nar depends on } k\}, \boldsymbol{I})$
  if $\boldsymbol{J} = \varnothing$ **or** width($\boldsymbol{J}$) $\leqslant \varepsilon$ **then return** $\{\boldsymbol{J}\}$
  else
    $(k, \boldsymbol{J}_1, \ldots, \boldsymbol{J}_l)$:= Divide $(\boldsymbol{J}, \varepsilon)$
    return Bisect $(\mathcal{N}, \{k\}, \varepsilon, \boldsymbol{J}_1) \cup \cdots \cup$
        Bisect $(\mathcal{N}, \{k\}, \varepsilon, \boldsymbol{J}_l)$
  endif
end

result is a set of vectors of domains containing the solution set of the initial constraint system.

In practice, there exists some problems whose solving requires many bisections compared to the domains contractions computed by constraint propagation. In this case the propagation algorithm must be able to efficiently detect when the input domains given by the bisection algorithm are a fixed-point of the narrowing operators.

5. Experimental Results

The constraint solving algorithm has been tested on some examples from numerical analysis and CLP(Intervals) benchmarks [14, 5]. The input set of narrowing operators is computed from the natural interval extensions of the user's constraints. Table 5 presents the computational results obtained on a Sun UltraSparc 2 (166 MHz). Figures in columns are the computation times for different propagation algorithms: BC3 and BC_φ for the corresponding algorithms, BC_0 for BC_φ where the value of φ is always 0.0 (i.e. BC_φ using BC-narrowing operators and the new propagation strategy based on the intensification of some operators) and $\mathsf{BC}_{1/8}$ for BC_φ where the value of φ is always 0.125. The results of BC3, BC_0 and BC_φ are a fixed-point (computation of box consistency), while $\mathsf{BC}_{1/8}$ may stop before the computation of a fixed-point when all the distances between the interval functions' quasi-zeros and the bounds of domains are smaller than 0.125. The last column indicates the gain of BC_φ with respect to BC3.

Problem "Broyden v" is the Broyden-Banded functions with v variables:

$$\begin{cases} x_k(2+5x_k^2)+1-\sum_{j\in J_k} x_j(1+x_j)=0 \\ 1 \leqslant k \leqslant v \qquad x_i \in [-10^8, +10^8] \\ J_k = \{j \mid j \neq k, \max(1, i-5) \leqslant j \leqslant \min(v, i+1)\} \end{cases}$$

Table 5. Comparison between box consistency algorithms (times in seconds).

Problem	v	BC3	BC_0	$\mathsf{BC}_{1/8}$	BC_φ	$\nearrow$
Broyden	10	1.8	0.7	0.2	0.2	**9**
	20	4.2	1.5	0.4	0.3	**14**
	40	10.2	3.5	0.9	0.7	**14**
	80	25.5	8.7	2.0	1.8	**14**
	160	67.1	23.4	5.1	4.4	**15**
	320	190.2	81.2	14.7	12.6	**15**
Cosnard	10	2.0	0.6	0.2	0.1	**20**
	20	13.5	2.5	0.9	0.5	**27**
	40	102.2	12.7	4.6	2.9	**35**
	80	950.5	80.1	29.8	18.6	**51**
Griewank	2	5.0	4.5	1.8	2.5	**2**
Kearfott	4	5.3	3.1	2.3	2.1	**2**
i2×10	10	1.0	0.2	0.2	0.2	**5**
Powell	4	5.5	0.2	0.2	0.1	**55**

The v-dimensional problem "Cosnard v" corresponds to the Moré-Cosnard equations resulting from the discretization of a nonlinear integral equation:

$$\begin{cases} x_k + \frac{1}{2}[(1-t_k)\sum_{j=1}^{k} t_j(x_j+t_j+1)^3 + \\ \qquad t_k \sum_{j=k+1}^{v}(1-t_j)(x_j+t_j+1)^2] = 0 \\ 1 \leqslant k \leqslant v \quad t_j = jh \quad h = 1/(v+1) \quad x_i \in [-4, 5] \end{cases}$$

Constraint propagation using BC_φ-narrowing operators is able to compute the best possible approximations for the solutions of these two problems (bisection is useless). Their scalability permits illustrating the behaviours of the different algorithms when the dimension increases. In particular the time growth is almost linear for the Broyden problem and quadratic for the Cosnard problem. Benchmark "i2" is also a problem from interval analysis. The other problems ("Griewank", "Kearfott", "Powell") are used since the computation of the solutions requires bisection (the precision ε is set to 10^{-8}).

The comparison of BC3 and BC_0 which use the same narrowing operators permits exhibiting the efficiency of the new propagation strategy. The results for Broyden indicate that BC_0 is twice faster whatever the dimension is. The constant improvement factor comes from the problem regularity: a constant number of narrowing operators is selected and their costs are almost the same due to the smoothness of constraint expressions. This factor increases with the dimension for Cosnard (3 for 10 variables, and more than 10 for 80 variables) since the constraint expressions become more complex when the dimension increases.

The comparison of BC_0 and $\mathsf{BC}_{1/8}$ shows the need for implementing box$_\varphi$ consistency in order to prevent some slow convergence phenomena during the search for the local quasi-zeros of interval functions. The improvement factor is constant for Cosnard (about 3) while it linearly increases for Broyden (from 3.5 for 10 variables to 5.5 for 320 variables).

The comparison of $\mathsf{BC}_{1/8}$ and BC_φ demonstrates a small acceleration resulting from a good estimate of the distance from the current domains to the global fixed-point. The major improvement concerns the precision of the final domains. For example, the maximal precision (box consistency) is required for Griewank problem in order to limit the generation of a huge amount of sub-domains. $\mathsf{BC}_{1/8}$ is faster on this example but generates nine domains which do not contain any solution (and one enclosing the solution) in 315 bisections. BC_φ only computes one domain enclosing the solution in 163 bisections.

Finally the comparison of BC_φ and BC3 (last column) illustrates the efficiency of BC_φ resulting from the complementarity of the new techniques proposed in this paper (box$_\varphi$ consistency and propagation strategy). The improvement factor is almost constant for Broyden while it increases for Cosnard (from 20 for 10 variables to 51 for 80 variables). In practice, no benchmark illustrating a significant slowdown of BC_φ has been found.

6. Conclusion

In this paper, box$_\varphi$ consistency, a weakening of box consistency, has first been introduced; we have then described an efficient implementation of box consistency that relies on the above-mentioned coarser consistency notion and on a new propagation strategy intensifying the application of the most contracting narrowing operators. Experimental evidences show that the new algorithm for enforcing box consistency leads to significant speeds-up for some standard benchmarks.

The computation of φ and the intensification strategy are based on two heuristics. A possible refinement would come from the implementation of some techniques from local search in order to capture the quality (tightening ability) of narrowing operators that is dynamic in essence.

Due to lack of space, a main feature of BC_φ has not been discussed, namely its nice properties for parallelization. A first approach would exploit the independence of the selection stage computations that may easily be distributed.

Acknowledgments

Discussions with Olivier Lhomme are gratefully acknowledged.

References

[1] F. Benhamou. Heterogeneous Constraint Solving. In *Proc. of Algebraic and Logic Programming*, volume 1139 of *LNCS*, 1996.

[2] F. Benhamou, D. McAllester, and P. Van Hentenryck. CLP(Intervals) Revisited. In *Proc. of International Logic Programming Symposium*. MIT Press, 1994.

[3] J. G. Cleary. Logical Arithmetic. *Future Computing Systems*, 2(2):125–149, 1987.

[4] E. Davis. Constraint Propagation with Interval Labels. *Artificial Intelligence*, 32:281–331, 1987.

[5] F. Goualard, F. Benhamou, and L. Granvilliers. An Extension of the WAM for Hybrid Interval Solvers. *Journal of Functional and Logic Programming*, 1999.

[6] E. R. Hansen. *Global Optimization using Interval Analysis*. Marcel Dekker, 1992.

[7] E. Hyvönen. Constraint Reasoning based on Interval Arithmetic. The Tolerance Propagation Approach. *Artificial Intelligence*, 58:71–112, 1992.

[8] IEEE. IEEE standard for binary floating-point arithmetic. Technical Report IEEE Std 754-1985, Institute of Electrical and Electronics Engineers, 1985. Reaffirmed 1990.

[9] O. Lhomme. Consistency techniques for numeric CSPs. In *Proc. of International Joint Conference on Artificial Intelligence*, 1993.

[10] O. Lhomme, A. Gotlieb, and M. Rueher. Dynamic Optimization of Interval Narrowing Algorithms. *Journal of Logic Programming*, 37(1–2):165–183, 1998.

[11] A. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[12] R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1966.

[13] P. Van Hentenryck, L. Michel, and F. Benhamou. Newton - Constraint Programming over Nonlinear Constraints. *Science of Computer Programming*, 30(1–2):83–118, 1998.

[14] P. Van Hentenryck, L. Michel, and Y. Deville. *Numerica: a Modeling Language for Global Optimization*. MIT Press, 1997.