$\varepsilon_1\varepsilon_2\Phi$-Consistency

Technical Report No. TR-01/368

†Marius-Călin Silaghi, †Jamila Sam-Haroud, ‡Rainer Weigel,
†Xuan-Ha Vu, and †Boi Faltings

†Laboratoire d'Intelligence Artificielle
Département d'Informatique
EPFL, Ecublens
CH-1015 Lausanne

E-mail : { silaghi, haroud, xuanha, faltings } @lia.di.epfl.ch

Fax: ++41-21-693.52.25

‡Fachhochschule für Technik, Wirtschaft
und Soziale Arbeit St.Gallen (FHSG)
MIT IT CH-9001 St. Gallen
Tel. ++41 (0)71 288 99 72

E-mail : rainer.weigel@fhsg.ch

July, 2001

Many large sets of problems, specially alternatives of design problems, have important common components. Considerable simplifications to a set of queries are obtained when relevant information concerning large common subproblems can be compiled in advance. Such compiled information can also help human users to perform an analysis in order to first direct search toward the most promising alternatives of a design.

In this paper we describe $\varepsilon_1\varepsilon_2\Phi$-consistent constraints, a class of compiled representations with a predefined upper-bound of global approximation errors. To allow for increasing the resolution of the compiled information with manageable complexity, an $\varepsilon_1\varepsilon_2\Phi$-consistent constraint approximates projections of the solution space of a Numeric CSP on a subset of its variables. The upper bounds of approximation errors are parameterized by: a representation approximation error, ε_1; a vector ε_2 defining the splitting resolution on each variable of the constraint; and a set of contracting operators Φ that have reached a fix point.

1 Introduction

Many large sets of problems, specially alternatives of design problems, have important common components. Considerable simplifications to a set of queries are obtained when relevant information concerning large common subproblems can be compiled in advance. Such compiled information can also help human users to perform an analysis in order to first direct search toward the most promising alternatives of a design.

Here we describe $\varepsilon_1\varepsilon_2\Phi$-consistent constraints, a class of compiled representations with a predefined upper-bound of global approximation errors. To allow for increasing the resolution of the compiled information with manageable complexity, an $\varepsilon_1\varepsilon_2\Phi$-consistent constraint approximates projections of the solution space of a Numeric CSP on a subset of its variables. The upper bounds of approximation errors are parameterized by: a representation approximation error, ε_1; a vector ε_2 defining the splitting resolution on each variable of the constraint; and a set of contracting operators Φ that have reached a fix point.

2 Relations and Approximations

A real constraint $c(x_1, \ldots, x_n)$ has the form $f_1(x_1, \ldots, x_n) op f_2(x_1, \ldots, x_n)$, where $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ and $op \in \{<, >, =, \geq, \leq\}$. Let $c(x_1, \ldots, x_n)$ be a real constraint with arity n. The *relation* defined by c, denoted by ρ_c, is the set of tuples satisfying c. The relation defined by the negation, $\neg c$, of c is given by $\mathbb{R}^n \setminus \rho_c$ and is denoted by $\rho_{\overline{c}}$. The global *relation* defined by the conjunction of all the constraints of a NCSP, $\mathcal{C}$, is denoted $\rho_{\mathcal{C}}$. It can be approximated by a computer-representable superset or subset. In the first case the approximation is *complete* but may contain points that are not solutions. Conversely, in the second case, the approximation is *sound* but may lose certain solutions. A relation ρ can be approximated conservatively by the smallest (w.r.t set inclusion) union of boxes, **Union**ρ, or more coarsely by the smallest box **Outer**ρ, containing it. By using boxes included into ρ, sound (inner) approximations **Inner**ρ can also be defined. In [2], **Inner**ρ is defined as the set $\{r \in \mathbb{R}^n \mid \textbf{Outer}\{r\} \subseteq \rho\}$.

The computation of these approximations relies on the notion of *contracting operators*. Basically, a contracting operator narrows down the variable domains by discarding values that are locally inconsistent. This is often done using bound consistency. In this paper we use the notion of outer contracting operator, defined as follows:

Definition 1 (Outer contracting operator) *Let $\mathbb{I}$ be a set of intervals over $\mathbb{R}$ and ρ a real relation. The function $\mathbf{OC}_\rho : \mathbb{I}^n \to \mathbb{I}^n$ is a contracting operator for the relation ρ iff for any box $\mathbf{B}, \mathbf{B} \in \mathbb{I}^n$, the next properties are true:*

$$\begin{array}{llcll} (1) & \mathbf{OC}_\rho(\mathbf{B}) & \subseteq & \mathbf{B} & \textit{(Contractiveness)} \\ (2) & \rho \cap \mathbf{B} & \subseteq & \mathbf{OC}_\rho(\mathbf{B}) & \textit{(Completeness)} \end{array}$$

The contractiveness in unidimensional cases is illustrated in Figure 1. Often, a monotonicity condition is also required in some approaches [5, 1]. The

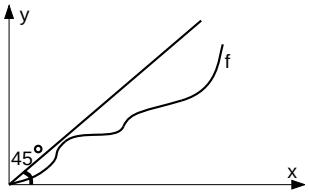


Figure 1: A function f in unidimensional cases is *contractive* if for positive arguments it is positive and lies below the identity function.

monotonicity ensures that a *maximal* fix-point is obtained. However, the ideal result is the *minimal* fix-point where the completeness is not lost. Reaching this minimal fix-point (as Hull-consistency) is impossible or prohibitive with most common operators (Bound or Box consistency). Figure 2 describes this situation. While in general one cannot guarantee that the wished *minimal* fix-point is reached, it is seldom useful to guarantee that the *maximal* fix-point is be obtained.

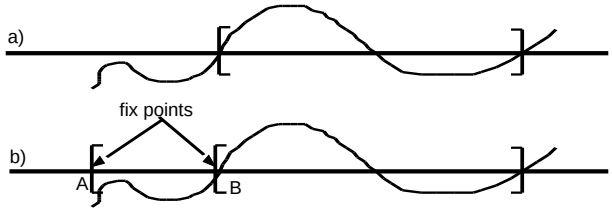


Figure 2: a) Monotonicity reaches the unique (wished) fix-point with Hull-consistency. b) With techniques like Box-consistency, monotonicity leads to the worst fixed point, A. Without monotonicity one may reach a smaller complete fix-point (e.g. B).

Let $\mathcal{V}_{\mathbb{IR}} = \{x_1 \dots x_n\}$ be a set of variables taking their values over $\mathbb{IR}$. Given $\sum_{\mathbb{IR}} = \{\mathbb{IR}, \mathcal{F}_{\mathbb{IR}}, \mathcal{R}_{\mathbb{IR}}\}$ a structure where $\mathcal{F}_{\mathbb{IR}}$ denotes a set of operators and $\mathcal{R}_{\mathbb{IR}}$ a set of relations defined in $\mathbb{IR}$, a *real constraint* is defined as a first order formula built from $\sum_{\mathbb{IR}}$ and $\mathcal{V}_{\mathbb{IR}}$. Interval arithmetic methods [5] are the basis of interval constraint solving. They approximate real numbers by intervals and compute conservative enclosures of the solution space of real constraint systems.

3 $\varepsilon_1\varepsilon_2\Phi$-Consistency

Consistencies are classes of representations which can be achieved using reduction operators. Such reduction operators are mostly used to simplify a (sub)problem during search. However, consistency can be use to simplify a subproblem that may occur often in the future problems. In this last case, the challenge is to find a representation which can be used efficiently in future con-

texts, and which contains as much important information as possible. Saving the whole solution space of a NCSP is out of discussion since even with floating point limitations, the space requirements to store all the solution points is usually intractable. In contrast, Hull consistent or Arc-consistent domains [4] are cheaper to store, but contain relatively little information.

A natural alternative to the punctual approach is to cover the spectrum of solutions for inequalities using a reduced number of subsets from $\mathbb{R}^n$. Usually, these subsets are chosen with known and simple properties (interval boxes, polytopes, ellipsoid) [6]. Several authors have proposed set covering algorithms with intervals boxes [6, 3, 8, 2]. Section 4 describes an algorithm for *dynamically* constructing an interval-box covering, for a set of equality/inequality constraints, according to a "maintaining local consistency" search schema. It builds on the feasibility test proposed in [2]. This allows for robustly constructing sound boxes and devising efficient splitting heuristics for search. The output is a union of boxes which conservatively encloses the solution set.

This section describes a class of outer approximations projected on a subset of variables and characterized by a property that we call $\varepsilon_1\varepsilon_2\Phi$-consistency. $\varepsilon_1\varepsilon_2\Phi$-consistency is a weaker version of global consistency, which only considers approximations of certain projections of the solution space. Such representations can be constructed as a preprocessing technique for speeding up further queries.

When a representation of all the solutions of a NCSP has to be built, or even its projection to a quite limited number of variables, the precision is the most constraining factor. The space required depends exponentially on this precision. The analytic representation itself is very efficient in space, but is less easy to visualize and offers less topological information. The amount of aggregation on solutions is a second factor that controls the required space. The improvements that can be achieved depend on the problem at hand. The next definition introduces the notion of $\varepsilon_1\varepsilon_2\Phi$-consistency which allows for characterizing the representation of the solution space for any projection on a given subset of variables.

Definition 2 ($\varepsilon\Phi$-solution) *An $\varepsilon\Phi$-solution of a NCSP $\mathcal{N}$ is a box denoted by $\nu_{\mathcal{N},\varepsilon} = I_1 \times ... \times I_n$ (n is the number of variables in $\mathcal{N}$) such that the search tool consisting of contracting operators Φ and splitting operators with resolution ε cannot reduce it or decide the absence of solutions inside it.*

An $\varepsilon\Phi$-solution is therefore either a completely feasible box, or a box in which Φ cannot find an infeasible box with all dimensions larger than ε. Techniques like those in Section 4 can be used to obtain boxes that are either completely feasible, or that have all dimensions lower than ε (see Figure 3). ε can be seen as a vector $\{\varepsilon^1, ..., \varepsilon^n\}$, where ε^i is the resolution chosen for x_i. Further we consider for simplicity that ε is a constant, and $\varepsilon^i = \varepsilon, \forall i$.

Definition 3 ($\varepsilon_1\varepsilon_2\Phi$-consistency) *A constraint $c(x_1, ...x_k)$ of a NCSP $\mathcal{N} = (V, C, D)$ is $\varepsilon_1\varepsilon_2\Phi$-consistent related to the variables in $X = \{x_1, ..., x_k\}$, $X \subseteq V$, iff:*

$$\rho_{N|_X} \subseteq \rho_c,\ \forall \overline{v} \in D_{x_1} \times ... \times D_{x_k},\quad \overline{v} \in \rho_c \Rightarrow \exists \nu_{N,\varepsilon_2},\quad \exists \overline{b} \in \nu_{\mathcal{N},\varepsilon_2|_X},\quad |\,\overline{v} - \overline{b}\,| < \varepsilon_1$$

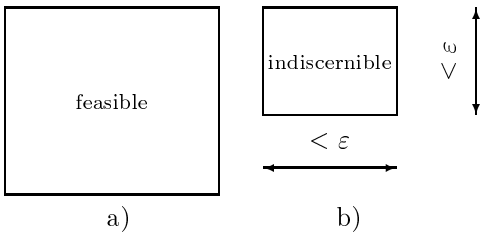


Figure 3: Example of $\varepsilon\Phi$-solutions: a) completely feasible box; b) indiscernible box.

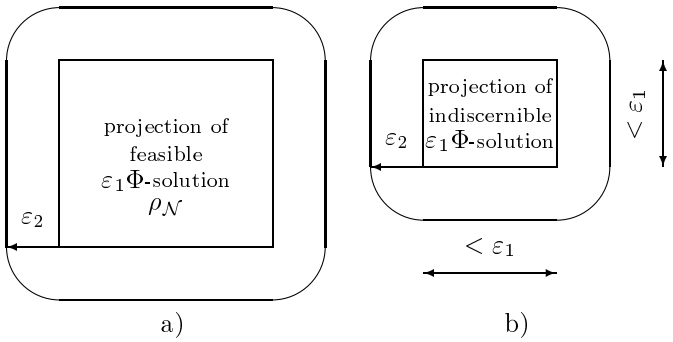


Figure 4: Simple example of an $\varepsilon_1\varepsilon_2\Phi$-consistent constraint C of a NCSP, $\mathcal{N}$. C is here a space surrounded by the box with rounded corners: a) building on a completely feasible box; b) building on an indiscernible box.

$\rho_{N|_X}$ represents the projection of the n dimension solution space of $\mathcal{N}$, denoted ρ_N, into the space X. When all the solutions of a NCSP, $\mathcal{N}$, are contained in an $\varepsilon_1\Phi$-solution (Figure 3), a corresponding worst approximation $\varepsilon_1\varepsilon_2\Phi$-consistent constraint C of $\mathcal{N}$ is obtained as in Figure 4.

Theorem 1 *Let C be an $\varepsilon_1\varepsilon_2\Phi$-consistent constraint for a NCSP, $\mathcal{N}$. Let S be a set of $\varepsilon\Phi$-solutions, $\rho_{\mathcal{N}} \subseteq S$. Any constraint C' such that $S\,|_{vars(C)} \subseteq \rho_{C'} \subseteq \rho_C$ is an $\varepsilon_1\varepsilon_2\Phi$-consistent constraint for $\mathcal{N}$.*

Proof. Referring to the projections of the same $\varepsilon\Phi$-solutions as C, the Euclidean distance condition ($a' \leq a \leq \varepsilon_1$) is respected by any element of $\rho_{C'}$ (Figure 5). □

Definition 4 *The conjunction of two constraints C_1, respectively C_2, is a constraint C, $C = C_1 \wedge C_2$, such that $\rho_C = \rho_{C_1} \cap \rho_{C_2}$.*

Definition 5 *The disjunction of two constraints C_1, respectively C_2, is a constraint C, $C = C_1 \vee C_2$, such that $\rho_C = \rho_{C_1} \cup \rho_{C_2}$.*

Let C be the conjunction of two $\varepsilon_1\varepsilon_2\Phi$-consistent constraints, C_1 for a NCSP, $\mathcal{N}_1$, respectively C_2 for a NCSP, $\mathcal{N}_2$. C is not necessarily an $\varepsilon_1\varepsilon_2\Phi$-consistent

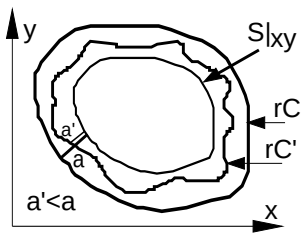


Figure 5: Any outer approximation constraint C', whose feasible set $\rho_{C'} = rC'$ is contained in an $\varepsilon_1\varepsilon_2\Phi$-consistent constraint C, $\rho_C = rC$, is $\varepsilon_1\varepsilon_2\Phi$-consistent.

constraint for the NCSP $\mathcal{N}_1 \cup \mathcal{N}_2$. Intuitively, the intersection of two $\varepsilon\Phi$-solutions in the two NCSPs may not be an $\varepsilon\Phi$-solution

Theorem 2 *Let C be the disjunction of two $\varepsilon_1\varepsilon_2\Phi$-consistent constraints, C_1 for a NCSP, $\mathcal{N}_1$, respectively C_2 for a NCSP, $\mathcal{N}_2$. C is an $\varepsilon_1\varepsilon_2\Phi$-consistent constraint for the NCSP $\mathcal{N}_1 \vee \mathcal{N}_2$.*

Proof. Any $\varepsilon\Phi$-solution of $\mathcal{N}_1$ or $\mathcal{N}_2$ is an $\varepsilon\Phi$-solution of $\mathcal{N}_1 \vee \mathcal{N}_2$. Therefore any element of $\rho_{C_1} \cup \rho_{C_2}$, belonging either to ρ_{C_1} or to ρ_{C_2}, is within Euclidean distance ε_2 from a an $\varepsilon\Phi$-solution. □

The next corollary is obvious:

Corollary 2.1 *An $\varepsilon_1\varepsilon_2\Phi$-consistent NCSP whose solution space is contained in a set, S, of ε_2-solutions can be obtained as the union of outer approximations of each box in S, where the precision of the approximation is $\varepsilon_1/\sqrt{2}$ (see Theorem 1).*

3.1 Example of $\varepsilon_1\varepsilon_2\Phi$-consistent constraint

We now illustrate a simple example of an $\varepsilon_1\varepsilon_2\Phi$-consistent constraint, namely *the $\varepsilon_1\varepsilon_2\Phi$-consistent binary bitmap constraint.* Let N be a NCSP with variables $\{x_1, x_2, ..., x_n\}$, each variable x_k taking its value in an interval $[a_k, b_k]$. Given two distinct variables, x_i and x_j, a projection of a numeric constraint, c_i, on x_i, x_j can be approximated with a bitmap representation. In [8, 7, 9], a cell in a tree or bitmap is feasible when assignments for x_i and x_j can be found such that c_i is satisfied. By constraint propagation, the global infeasibility of some cells can be detected. These representations help in solving N, but also in giving hints about the geometry of the solution space.

$\varepsilon_1\varepsilon_2\Phi$-consistency is a result of the separate consideration of these two different targets, namely solving and analyzing N. $\varepsilon_1\varepsilon_2\Phi$-consistency is not meant for solving N, but rather it is obtained as a result of solving. Instead, an $\varepsilon_1\varepsilon_2\Phi$-consistent constraint signal regions containing global solutions of N (somewhat similar to global consistency). It can be used as a redundant constraint for improving further solving of NCSPs of the type $N \wedge N'$ where N' is a NCSP. This is useful when several NCSP with a common part have to be solved successively.

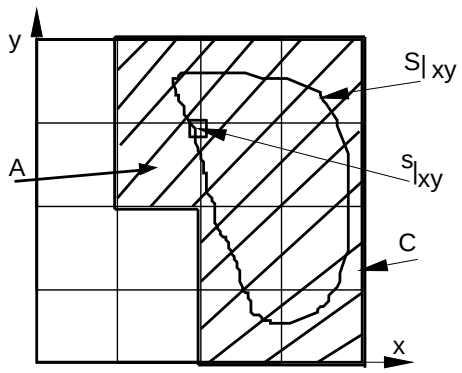


Figure 6: A bitmap $\varepsilon_1\varepsilon_2\Phi$-consistent constraint with feasible space C for a NCSP, N, whose solution space projection on x, y is $S_{|_{xy}}$. The grid resolution is $\varepsilon_1/\sqrt{2}$. The cell A belongs to C since it intersects the projection of an $\varepsilon\Phi$-solution s, $s_{|_{xy}}$.

4 Algorithms

We now present an algorithm named UCA6 (Algorithm 1) that computes a **Union** approximation for numerical CSPs with equalities and inequalities [10]. We note lists in the Prolog style $[Head|Tail]$. $\mathcal{B}$ denotes the list of children to be checked for a node, and $\mathcal{P}$ denotes the list of all $\mathcal{B}$. The algorithm presented is depth-first. Breadth-first and other heuristics can be obtained by treating the lists $\mathcal{B}$ and $\mathcal{P}$ as sets, $\mathcal{P}$ becoming respectively the union of all the sets of type $\mathcal{B}$. The algorithm UCA6 iteratively calls the function **getNext** which delivers a new **Outer** approximation for a subspace in the solution space. By construction, the new **Outer** approximation will not intersect with any previously computed box. The function **getNext** has two main components: a reduction operator, **reduc** (Algorithm 2), and a splitting operator, **split** (Algorithm 3). These operators are interleaved as in a classical maintaining bound consistency algorithm. Practically, it is preferable to stop the dichotomous split when the precision of the numeric search tool (splitting and contracting operators) can lead to unsafe solutions at a given precision ε. An unsafe solution is a box that may contain no real solution. **reduc**, checks this state using a function called *Indiscernible*(constraint, **Box**, **OC**, ε), which is not discussed here in detail[1]. Since we assume that **OC** and ε do not change during search, the actual call to *Indiscernible* is *Indiscernible*(constraint, **Box**).

Each search node is characterized by the next structures:

* The list $\mathcal{B}$ corresponds to a set of splits for the current search node. It defines the next branches of search. Each split correspond to a new node of the search.

[1]The simplest strategy consists of checking that all the intervals are smaller than ε, but more sophisticated techniques can be built by estimating computational errors.

```
    procedure UCA6(C = (V, C, D): NCSP) do
        P = [[{OC_ρC(D), C, {B_q(OC_ρC(D))}}]];
0.1     while (getNext(P, C, solution)) do
            U ← {solution}∪U;
        return U;
    function getNext(inout:P = [B = [{B ∈ 𝕀^n, C : NCSP, {B_q ∈ 𝕀^n}} |
    T_B] | T_P];
    in: C_G ∈ NCSP; out: solution∈ 𝕀^n) → bool
        for ever do
0.2         if (B = [ ]) then
0.3             if (T_P = [ ]) then
0.4                 return (false);
                else
0.5                 P ← T_P;
0.6             continue;
0.7         (C',B',{B_q'}) ← reduc(C, B, {B_q});
0.8         B ← T_B;
0.9         if (B' <> ∅) then
0.10            if (C' = ∅) then
0.11                solution ← B';
0.12                return (true);
0.13            B' ← split(B',C',{B_q'});
0.14            P ← [B' | P];
```

Algorithm 1: The Search procedure

```
    function reduc(in: C : NCSP, B∈ 𝕀^n, {B_i∈ 𝕀^n}) →(NCSP, 𝕀^n, {𝕀^n})
1.1     B' ← OC_ρC(B);
1.2     for all (q={inequality}, q∈ C, B_q ∈ {B_i}) do
1.3         if (B'∩B_q <> B_q) then
1.4             B_q ← OC_ρ¬q(B'∩B_q);
1.5             if ((B_q = ∅)∨Indiscernible(q,B_q)) then
1.6                 C ← C \ {q}, {B_i} ← {B_i} \ B_q;
1.7     for all (q=equality, q∈ C) do
1.8         if (Indiscernible(q,B')) then
1.9             C ← C \ {q}, {B_i} ← {B_i} \ B_q;
1.10    return (C, B', {B_i});
```

Algorithm 2: Problem Reduction

* A box $\mathbf{B}$ defining the domains of the current NCSP.

* The current NCSP $\mathcal{C}$ containing only the constraints of the initial NCSP that can participate in pruning the search space. The constraints that are indiscernible or entirely feasible in $\mathbf{B}$ are eliminated.

* Each constraint q in a node is associated with a box, $\mathbf{B_q}$, such that all the space in $\mathbf{B} \setminus \mathbf{B_q}$ is feasible.

Each $\mathbf{B_q}$ is initially equal with the projection of the initial search space on the variables in the constraint q, after applying $\mathbf{OC}_{\rho_{\overline{q}}}$. One of the features of **reduc** is that it removes redundant completely feasible or indiscernible constraints. If the recent domain modifications of some inequality q have modified $\mathbf{B_q}$, q is checked for feasibility at line 1.4, and eventually removed from the current CSP (line 1.6). Equalities are similarly eliminated at line 1.9 when they become indiscernible.

4.1 Splitting operator

The function **split** (Algorithm 3) allows for using three splitting strategies. The first one, **splitFeasible**, extracts sound subspaces for some inequality, as long as these subspaces fragment the search space in a ratio limited by a given *fragmentation threshold*, denoted by *frag* (line 2.4). The second and the third strategies (**splitIneq**, respectively **splitEq**), consist of choosing for dichotomous split, a variable involved in an inequality (respectively an equality) of the current NCSP $\mathcal{C}$. The heuristics used at lines 2.5, 2.6, 2.7, and 2.8 in Algorithm 3 can be based on the occurrence of variables in the constraints of $\mathcal{C}$, or according to a round robin technique. The domain of the chosen variable is then split in two halves. Techniques based on the occurrences of variables in constraints can also be used to devise heuristics on ordering the bounds at line 2.3 in **splitFeasible**. The criteria for choosing a constraint at line 2.2 can look for maximizing the size of the search space for which a given constraint is eliminated, minimize the number of children nodes, or maximize the number of constraints that can benefit[2] from the split.

Given two boxes $\mathbf{B}$ and $\mathbf{B_q}$, where $\mathbf{B}$ contains $\mathbf{B_q}$, and given a bound b in $\mathbf{B_q}$ for a variable x, we use the next notations:

* $\mathbf{B_{f(x,b)[B_q,B]}}$ is the (feasible) box not containing $\mathbf{B_q}$ obtained from $\mathbf{B}$ by splitting the variable x in b.

* $\mathbf{B_{u(x,b)[B_q,B]}}$ is the (indiscernible) box containing $\mathbf{B_q}$ obtained from $\mathbf{B}$ by splitting the variable x in b.

* $\mathbf{B_{\frac{1}{2}r(x)[B]}}$ is the (indiscernible) box obtained from $\mathbf{B}$ by splitting the variable x in half and retaining its upper half.

[2]The constraints for which the domains are split may propagate more when **OC** is applied.

function split(**in:** $\mathbf{B} \in \mathbb{I}^n$, $\mathcal{C}$: NCSP, $\{\mathbf{B_i} \in \mathbb{I}^n\}$) $\rightarrow [\{\mathbb{I}^n, \text{NCSP}, \{\mathbb{I}^n\}\} \mid \_]$

2.1 fun $\leftarrow$ choose appropriate(splitFeasible, splitIneq, splitEq);
$\mathcal{B} \leftarrow [\,]$;
fun($\mathbf{B}$, $\mathcal{C}$, $\{\mathbf{B_i}\}$, $\mathcal{B}$);
return $\mathcal{B}$;

procedure splitFeasible(**in:**$\mathbf{B}$,$\mathcal{C}$,$\{\mathbf{B_i}\}$;**inout:**$\mathcal{B} \in [\{\mathbb{I}^n, \text{NCSP}, \{\mathbb{I}^n\}\} \mid \_]$) **do**

2.2 q $\leftarrow$ choose {inequality}$\in \mathcal{C}$, $\mathbf{B_q} \in \{\mathbf{B_i}\}$;
2.3 **foreach** *(bound* b *of some variable* x *of* q *in* $\mathbf{B_q}$ *(e.q. in descending order of the relative distance* rd *to the corresponding bound in* $\mathbf{B}$*))* **do**
2.4 **if** (rd $<$*frag*) continue;
$\mathbf{B'} \leftarrow \mathbf{B_{f(x,b)[B_q,B]}}$;
$\mathbf{B} \leftarrow \mathbf{B_{u(x,b)[B_q,B]}}$;
$\mathcal{B} \leftarrow [\{\mathbf{B'}, \mathcal{C} \setminus \{q\}, \{\mathbf{B_i}\} \setminus \mathbf{B_q}\} \mid \mathcal{B}\,]$;
$\mathcal{B} \leftarrow [\{\mathbf{B}, \mathcal{C}, \{\mathbf{B_i}\}\} \mid \mathcal{B}\,]$;

procedure splitIneq(**in:**$\mathbf{B}$,$\mathcal{C}$,$\{\mathbf{B_i}\}$;**inout:**$\mathcal{B} \in [\{\mathbb{I}^n, \text{NCSP}, \{\mathbb{I}^n\}\} \mid \_]$) **do**

2.5 q $\leftarrow$ choose {inequality, $\mathbf{B_q} \in \mathbb{I}^n\} \in \mathcal{C}$;
2.6 x $\leftarrow$ choose variable of q given $\mathcal{C}$;
$\mathcal{B} \leftarrow [\{\mathbf{B_{\frac{1}{2}r(x)[B]}}, \mathcal{C}, \{\mathbf{B_i}\}\} \mid \mathcal{B}\,]$;
$\mathcal{B} \leftarrow [\{\mathbf{B_{\frac{1}{2}l(x)[B]}}, \mathcal{C}, \{\mathbf{B_i}\}\} \mid \mathcal{B}\,]$;

procedure splitEq(**in:**$\mathbf{B}$,$\mathcal{C}$,$\{\mathbf{B_i}\}$;**inout:**$\mathcal{B} \in [\{\mathbb{I}^n, \text{NCSP}, \{\mathbb{I}^n\}\} \mid \_]$) **do**

2.7 q $\leftarrow$ choose {equality} $\in \mathcal{C}$;
2.8 x $\leftarrow$ choose variable of q given $\mathcal{C}$;
$\mathcal{B} \leftarrow [\{\mathbf{B_{\frac{1}{2}r(x)[B]}}, \mathcal{C}, \{\mathbf{B_i}\}\} \mid \mathcal{B}\,]$;
$\mathcal{B} \leftarrow [\{\mathbf{B_{\frac{1}{2}l(x)[B]}}, \mathcal{C}, \{\mathbf{B_i}\}\} \mid \mathcal{B}\,]$;

Algorithm 3: Four splitting operators.

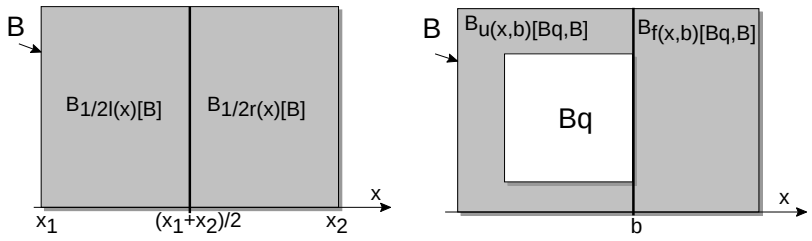


Figure 7: Splitting operators

* $\mathbf{B_{\frac{1}{2}l(x)[B]}}$ is the (indiscernible) box obtained from $\mathbf{B}$ by splitting the variable x in half and retaining its lower half.

These concepts are illustrated in the Figure 7. In Procedure *splitFeasible*, our

described implementation splits the current box q along all borders of B_q that do not fragment the search space with a ratio more than $frag$ (considering the ratio is computed such that it is always less or equal to 1). However, one may choose to split in a step only along the border which offer the highest ratio.

5 Building $\varepsilon_1\varepsilon_2\Phi$-consistent bitmap constraints

The procedure **UCA6** can be modified for generating the feasible cells for representing an $\varepsilon_1\varepsilon_2\Phi$-consistent bitmap constraint on a set X of variables. This is done by filtering out of $\mathcal{P}$, the boxes of search space (bitmap cells), whose points are closer to the found solution (line 0.1) than a distance ε_1. The distance is computed in the space defined by the variables in X. The procedure required for obtaining $\varepsilon_1\varepsilon_2\Phi$-consistent bitmap constraints is called **UCA7** and is given in Algorithm 4. Φ corresponds to the used **OC**, ε_1 is given by the size of the bitmap and ε_2 is given by the implementation of *Infeasible(c,Box)*.

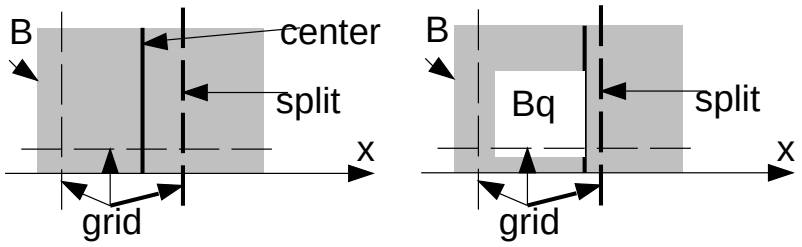
procedure UCA7$(\mathcal{C} = (V, C, D)$*: NCSP)* **do**
  $\mathcal{P} = [[\{\mathbf{OC}_{\rho_\mathcal{C}}(D), C, \{\mathbf{B_q}(\mathbf{OC}_{\rho_\mathcal{C}}(D))\}\}]]$;
3.1 **while** *(getNext(*$\mathcal{P}$*,* $\mathcal{C}$*, solution))* **do**
    $\mathcal{U} \leftarrow \{\text{solution}\} \cup \mathcal{U}$;
    mark as feasible the bitmap cells intersecting {solution};
3.2 **while** *non-empty(*$\mathcal{P}$*) and (marked-feasible-head(*$\mathcal{P}$*))* **do**
      $\mathcal{P} \leftarrow \text{tail}(\mathcal{P})$;
  return $\mathcal{U}$;

Algorithm 4: The Search procedure for bitmap $\varepsilon_1\varepsilon_2\Phi$-consistent representations.

In order to build an $\varepsilon_1\varepsilon_2\Phi$-consistent bitmap constraint, one has to filter out of $\mathcal{P}$ feasible bitmap cells. To avoid repeated return to cells found feasible and to avoid generating multiplications of the elements of the list, $\mathcal{P}$, during this filtering, the splitting of the boxes is preferably performed along the borders that separate the bitmap cells. Whenever $\varepsilon_1 >> \varepsilon_2$, there may be no appropriate cell border to reduce the size of (indiscernible) boxes. Boxes are then split along directions that are not cell borders, as in Figure 7. Filtering bitmap cells is performed by eliminating from $\mathcal{P}$ all boxes completely contained in cells that are marked as feasible.

For the examples in Figure 7, and given a bitmap generated by a grid with width $\varepsilon_1/\sqrt{2}$, the splitting strategies shown in Figure 8 are added to the previous ones. The chosen splitting line/hyper-plan is the grid line/hyper-plan that satisfies the splitting paradigm (bisection or separation of feasible regions) and is the closest to the ideal choice defined as in Figure 7. The split procedures are shown in Algorithm 5. The next notations, used in Algorithm 5, are defined as:

* $\mathbf{B}_{\frac{1}{2}\mathbf{rg(x)}[\mathbf{B}]}$ is the (indiscernible) box obtained from **B** by splitting the variable x along a grid line close to half and retaining its upper part.



a) b)

Figure 8: Splitting when bitmap cell boundaries intersect the current box, parallelly with the chosen split direction: a) Strategies with two indiscernible results; b) Strategies with one indiscernible result.

procedure splitFeasible(in:$\mathbf{B}, \mathcal{C}, \{\mathbf{B_i}\}$**;inout:**$\mathcal{B} \in [\{\mathbb{I}^n, \text{NCSP}, \{\mathbb{I}^n\}\} \mid \_])$ **do**

4.1 q ← choose {inequality}$\in \mathcal{C}$, $\mathbf{B_q} \in \{\mathbf{B_i}\}$;

4.2 **for** *(the closest external grid line to a bound* b *of some variable* x *of* q *in* $\mathbf{B_q}$ *(e.q. in descending order of the relative distance* rd *to the corresponding bound in* $\mathbf{B}$*))* **do**

4.3 **if** (rd <*frag*) continue;

$\mathbf{B'} \leftarrow \mathbf{B}_{\mathbf{f(x,b)[B_q,B]}}$;

$\mathbf{B} \leftarrow \mathbf{B}_{\mathbf{u(x,b)[B_q,B]}}$;

$\mathcal{B} \leftarrow [\{\mathbf{B'}, \mathcal{C} \setminus \{q\}, \{\mathbf{B_i}\} \setminus \mathbf{B_q}\} \mid \mathcal{B}\ ]$;

$\mathcal{B} \leftarrow [\{\mathbf{B}, \mathcal{C}, \{\mathbf{B_i}\}\} \mid \mathcal{B}\ ]$;

procedure splitIneq(in:$\mathbf{B}, \mathcal{C}, \{\mathbf{B_i}\}$**;inout:**$\mathcal{B} \in [\{\mathbb{I}^n, \text{NCSP}, \{\mathbb{I}^n\}\} \mid \_])$ **do**

4.4 q ← choose {inequality, $\mathbf{B_q} \in \mathbb{I}^n\} \in \mathcal{C}$;

4.5 x ← choose variable of q given $\mathcal{C}$;

$\mathcal{B} \leftarrow [\{\mathbf{B}_{\frac{1}{2}\mathbf{rg(x)}[\mathbf{B}]}, \mathcal{C}, \{\mathbf{B_i}\}\} \mid \mathcal{B}\ ]$;

$\mathcal{B} \leftarrow [\{\mathbf{B}_{\frac{1}{2}\mathbf{lg(x)}[\mathbf{B}]}, \mathcal{C}, \{\mathbf{B_i}\}\} \mid \mathcal{B}\ ]$;

procedure splitEq(in:$\mathbf{B}, \mathcal{C}, \{\mathbf{B_i}\}$**;inout:**$\mathcal{B} \in [\{\mathbb{I}^n, \text{NCSP}, \{\mathbb{I}^n\}\} \mid \_])$ **do**

4.6 q ← choose {equality} $\in \mathcal{C}$;

4.7 x ← choose variable of q given $\mathcal{C}$;

$\mathcal{B} \leftarrow [\{\mathbf{B}_{\frac{1}{2}\mathbf{rg(x)}[\mathbf{B}]}, \mathcal{C}, \{\mathbf{B_i}\}\} \mid \mathcal{B}\ ]$;

$\mathcal{B} \leftarrow [\{\mathbf{B}_{\frac{1}{2}\mathbf{lg(x)}[\mathbf{B}]}, \mathcal{C}, \{\mathbf{B_i}\}\} \mid \mathcal{B}\ ]$;

Algorithm 5: Splitting operators for bitmap $\varepsilon_1\varepsilon_2\Phi$-consistent constraints.

* $\mathbf{B}_{\frac{1}{2}\mathbf{lg(x)}[\mathbf{B}]}$ is the (indiscernible) box obtained from $\mathbf{B}$ by splitting the variable x along a grid line close to half and retaining its lower part.

The splitting strategies with the immediate subsequent filtering of $\mathcal{P}$ is not sufficient for guaranteeing that a cell detected as feasible will not be rummaged again. As shown in Figure 9, when the priority in splitting is not offered to

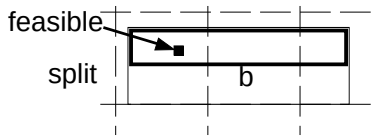


Figure 9: The filtering power is reduced if the splitting is not done along grid lines, when such lines intersect the current box. Here one cannot filter out the box b, so that the current feasible cell will be scanned again.

grid lines/hyper-plan, a cell may be reached again later. Such cases can still be cheaply detected relatively early, by checking if the current box is already marked feasible, whenever it is completely contained in a cell (not intersected by any grid line).

Lemma 1 *When the splitting is done along a grid line/hyper-plan whenever a grid line intersects the current box, the line 3.2 of procedure* **UCA7** *filters out of $\mathcal{P}$ all the boxes contained in the last feasible cell.*

Proof. Let us imagine that the bitmap cell, c, has just been found feasible due to an $\varepsilon\Phi$-solution B^*. We have to prove that no box B, whose projection is contained in c, is placed in $\mathcal{P}$ after a box B' which projects partly in another cell, c'.

Let us imagine that the previous affirmation does not hold. $\mathcal{P}$ is treated in FIFO order. It means that the box B strictly contained in c (does not contain B^*) has been split apart from a box B^0 containing B' and B^*. Therefore the boxes B^* and B^0 have been separated along a line/hyper-plan that is not a cell border, while a grid line/hyper-plan has intersected the current box across B^0 (the projection of B^0 intersects both c and c'). q.e.d. □

Theorem 3 *If the splitting is done along a grid line/hyper-plan whenever a grid line intersects the current box, then the search does never return to a bitmap cell found feasible in an $\varepsilon_1\varepsilon_2\Phi$-consistent bitmap constraint.*

Proof. In order to return to a feasible cell c, part of c should belong to a box, B in $\mathcal{P}$. If B is completely contained in c, it would have been filtered out in **UCA7** (Lemma 1). Therefore B contains parts of another bitmap cell, c'. B therefore intersects a grid line. Since c is marked feasible, it means that a box, B', intersecting c, has been proved to be $\varepsilon\Phi$-solution. When B' has been separated from B, the splitting action has separated c in two. Therefore that split was not performed along a grid line. Since B intersects a grid line, the hypothesis is contradicted and the proof is complete. □

6 Conclusions

We present $\varepsilon_1\varepsilon_2\Phi$-consistency, an approximation with bounded errors of global consistency for numeric CSPs. $\varepsilon_1\varepsilon_2\Phi$-consistent constraints are compiled representations that can provide relevant information concerning recurrent subproblems. An algorithm is then detailed, which allows for efficient computations of

$\varepsilon_1\varepsilon_2\Phi$-consistent bitmap constraints. It is based on a recent technique that we have proposed for reducing search effort in numeric CSPs with inequalities.

7 Acknowledgements

We want to thank Frederic Goualard for interesting feedback.

References

[1] Krzysztof R. Apt. The role of commutativity in constraint propagation algorithms. *ACM Transactions on Programming Languages and Systems*, TBD(TDB):1–35, 2001.

[2] F. Benhamou and F. Goualard. Universally quantified interval constraints. In *Procs. of CP'2000*, pages 67–82, 2000.

[3] J. Garloff and B. Graf. Solving strict polynomial inequalities by Bernstein expansion. *Symbolic Methods in Control System Analysis and Design, London: IEE*, pages 339–352, 1999.

[4] Esther M. Gelle. *On the Generation of Locally Consistent Solution Spaces in Mixed Dynamic Constraint Problems.* PhD thesis, EPF-Lausanne, 1998.

[5] L. Granvilliers. *Consistances locales et transformations symboliques de contraintes d'intervalles.* PhD thesis, Université d'Orléans, déc 98.

[6] L. Jaulin. *Solution globale et garantie de problèmes ensemblistes ; Application à l'estimation non linéaire et à la commande robuste.* PhD thesis, Université Paris-Sud, Orsay, Feb 94.

[7] Claudio Lottaz. *Collaborative Desgn Using Solution Spaces.* PhD thesis, EPF-Lausanne, 2000.

[8] D. Sam-Haroud and B. Faltings. Consistency techniques for continuous constraints. *Constraints, An International Journal,1*, pages 85–118, 96.

[9] M.-C. Silaghi, D. Sam-Haroud, and B. Faltings. Fractionnement intelligent de domaine pour CSPs avec domaines ordonnés. In *Proc. of RFIA2000*, 2000.

[10] M.-C. Silaghi, D. Sam-Haroud, and B. Faltings. Search techniques for nonliniar constraint satisfaction problems with inequalities. In *Proc. of AI2001*, Otawa, June 2001.