Numerical Constraint Satisfaction Problems with Non-isolated Solutions

Xuan-Ha Vu, Djamila Sam-Haroud, and Marius-Calin Silaghi

Artificial Intelligence Laboratory,
Swiss Federal Institute of Technology in Lausanne (EPFL),
CH-1015, Lausanne, Switzerland
{xuan-ha.vu, jamila.sam, marius.silaghi}@epfl.ch
http://liawww.epfl.ch

Abstract. In recent years, interval constraint-based solvers have shown their ability to efficiently solve complex instances of non-linear numerical CSPs. However, most of the working systems are designed to deliver *point-wise* solutions with an arbitrary accuracy. This works generally well for systems with isolated solutions but less well when there is a *continuum of feasible points* (e.g. under-constrained problems, problems with inequalities). In many practical applications, such large sets of solutions express equally relevant alternatives which need to be identified as completely as possible. In this paper, we address the issue of constructing *concise* inner and outer approximations of the complete solution set for non-linear CSPs. We propose a technique which combines the *extreme vertex representation* of orthogonal polyhedra [1–3], as defined in computational geometry, with adapted *splitting strategies* [4] to construct the approximations as unions of interval boxes. This allows for compacting the explicit representation of the complete solution set and improves efficiency.

1 Introduction

Many practical problems require solving constraint satisfaction problems (CSPs) with numerical constraints. A numerical CSP (NCSP), $(\mathcal{V}, \mathcal{C}, \mathcal{D})$, is stated as a set of variables $\mathcal{V}$ taking their values in domains $\mathcal{D}$ over the reals and subject to a finitely many set of constraints $\mathcal{C}$. In practice, the constraints can be equalities or inequalities of arbitrary type and arity, usually expressed using arithmetic expressions. In this paper we address the case of NCSPs with *non-isolated solutions*. Such a case is often encountered in real-world engineering applications where under-constrained problems, problems with inequalities or with universal quantifiers are ubiquitous. In practice, a set of non-isolated solutions often expresses a spectrum of equally relevant choices, as the possible moving areas of a mobile robot, the collision regions between objects in mechanical assembly, or different alternatives of shapes for the components of a kinematic chain. These alternatives need to be identified as precisely and completely as possible.

Interval constraint-based solvers (e.g. Numerica [5], ILOG Solver [6]) take as input an NCSP and generate a set of boxes which *conservatively* enclose each solution. They have proven particularly efficient in solving challenging instances of NCSPs with non-linear constraints. However, when applied to problems with non-isolated solutions

they provide enclosures that are either prohibitively verbose or poorly informative (see Section 2).

In contrast, a number of set-based approaches have been developed, notably in the areas of robust control, automation and robotics, which provide promising alternatives to the point-wise techniques. They consist in covering the spectrum of non-isolated solutions using a reduced number of subsets of $\mathbb{R}^n$. Usually, these subsets are chosen with known and simple properties (e.g. interval boxes, polytopes, ellipsoids). In recent years, several authors have proposed set covering algorithms with interval boxes [7–10]. Most existing box-covering algorithms are however limited by their restrictive applicability conditions or by their high average time and space complexities in the general case. The enhanced set-based technique we propose builds on the following observations. Firstly, the union of boxes produced by the complete interval-based solving of NCSPs can be seen as an *orthogonal polyhedron*[1]. Enhanced representations from computational geometry can be used to reduce the verbosity of such geometrical objects. We propose to use the *Extreme Vertex Representation* (EVR) of orthogonal polyhedra [1–3] for this purpose. Secondly, when there are non-isolated solutions, dichotomous splitting is not the most adapted branching strategy. It might lead to unnecessarily dividing entirely feasible regions. We propose to use another scheme based on splitting around the negation of feasible regions [4] which is an extension of the *negation test* proposed for universally quantified constraints in [10]. The resulting algorithm applies to general constraint systems. It produces inner and outer approximations of the feasible sets in the form of *unions of interval boxes*. The preliminary experiments show that it improves efficiency as well as the compactness and quality of the output representation.

2 Examples

We start by giving two small introductory examples which illustrate the inadequacy of point-wise approaches to the case of NCSPs with non-isolated solutions. The first example illustrates how the point-wise approach can be sometimes misused when applied to NCSPs with non-isolated solutions. Since point-wise techniques inherently assume the existence of isolated solutions, the interval splitting process they use for branching is sometimes prematurely stopped as soon as a solution is detected within an interval. This leads to poorly informative approximations of the complete solutions sets, as shown by the following example. The first example, called **WP**, is a 2D simplification of the design model for a kinematic pair consisting of a wheel and a pawl. The constraints determine the regions where the pawl can touch the wheel without blocking its motion. **WP** = $\{20 < \sqrt{x^2+y^2} < 50,\ 12y/\sqrt{(x-12)^2+y^2} < 10,\ x \in [-50, 50],\ y \in [0, 50]\}$. Figure 1 shows the output produced by a point-wise solver when the existence of point-wise solutions is abusively assumed.[2]

The second example consists of 4 non-linear inequality constraints involving 3 variables: **P3** = $\{x^2 \leq y,\ \ln y + 1 \geq z,\ xz \leq 1,\ x^{3/2} + \ln(1.5z + 1) \leq y + 1,\ x \in [-15, 15],\ y \in [1, 200],\ z \in [-10, 10]\}$. Using an efficient implementation of classical

[1] Informally, an orthogonal polyhedron is such that its facets are axis-parallel hyper-rectangles.

[2] It was solved using a combination of *IloGenerateBounds* and *IloSplit* in ILOG Solver 5.2.

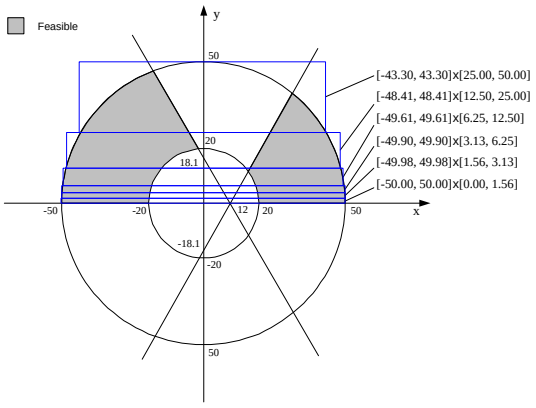

Fig. 1. The solution set of **WP** is approximated by 6 boxes (at precision = 2)

point-wise techniques,[3] the computation had to be stopped after 10 hours and produced more than 260000 small boxes. The alternative set-based technique we propose could reduce the complete output to 1376 boxes and produced the result in 1.41 seconds (see Table 1). This small example was one of our most successful and hence does not objectively illustrate the power of our technique, however it clearly illustrates how point-wise approaches can be unadapted to the complete solving of certain classes of problems.

3 Background and Definitions

3.1 Interval Arithmetic

The finite nature of computers precludes an exact representation of the *reals*. The real set $\mathbb{R}$ is in practice approximated by a finite set $\mathbb{F}_\infty = \mathbb{F} \cup \{-\infty, +\infty\}$, where $\mathbb{F}$ is a finitely many set of reals. In interval-based constraint solvers, $\mathbb{F}$ usually corresponds to the floating-point numbers. For each $l \in \mathbb{F}$, we denote $l^+ = \min\{f \in \mathbb{F}_\infty \mid l < f\}$, $l^- = \max\{f \in \mathbb{F}_\infty \mid f < l\}$. The set of intervals with bounds in $\mathbb{F}_\infty$, denoted by $\mathbb{I}$, is partially ordered by set inclusion. An *interval box*, or a *box* for short, $\mathbf{B} = I_1 \times \ldots \times I_n$ is a Cartesian product of n intervals in $\mathbb{I}$. We denote $\mathbf{B}|_i = I_i$. A *canonical interval* is a non-empty interval of the forms $[l, l]$, $[l^-, l]$ or $[l, l^+]$. Some extended definitions can be found in [10]. Two boxes, $\mathbf{A}$ and $\mathbf{B}$, are called disjoint if $\mathbf{A} \neq \mathbf{B}$ and $\exists i : \sup(\mathbf{A}|_i) \leq \inf(\mathbf{B}|_i) \vee \sup(\mathbf{B}|_i) \leq \inf(\mathbf{A}|_i)$. We denotes by $pts(S)$ the set of points represented by S, e.g. $pts(S) = \{x \mid x \in \mathbf{B} \in S\}$ if S is a set of boxes. Two sets of disjoint boxes, $\mathcal{U}_1$ and $\mathcal{U}_2$, are equivalent, denoted by $\mathcal{U}_1 \equiv \mathcal{U}_2$, if $pts(\mathcal{U}_1) = pts(\mathcal{U}_2)$.

3.2 Relations and Approximations

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 in $\mathbb{R}^n$ satisfying c. Let $vars(\rho_c) = vars(c) = \{x_1, \ldots, x_n\}$.

[3] The implementation was based on ILOG solver 5.2 (see Section 6).

The relation defined by the negation, $\neg c$, of c is given by $\mathbb{R}^n \setminus \rho_c$. In this paper, each box $\mathbf{B} \in \mathbb{I}^n$ and relation $\rho \subseteq \mathbb{R}^n$ is associated with n real variables, the projection of $\mathbf{B}$ on a subset, X, of its variables is denoted by $\mathbf{B}|_X$. We denote by $\mathcal{R}_n$ the set of relations defined on subsets of the n variables, ρ_P the relation defined by an NCSP, P, and $\rho_\mathcal{C}$ the *global relation* defined by the conjunction of constraints in a constraint set $\mathcal{C}$.

A relation can be approximated by a computer-representable superset or subset. The former is a *complete* approximation but may contain points that are not solutions. Conversely, the latter is a *sound* approximation but may lose certain solutions. In particular, a relation can be approximated conservatively by the smallest (w.r.t. set inclusion) union of boxes (called the *best outer approximation*), or more coarsely by the smallest box (called the *interval hull*), containing it. Beside that, the relation can also be approximated by the greatest union of boxes (called the *best inner approximation*) contained in it. The readers are referred to [11] for rigorous definitions.

The computation of these ideal approximations relies on the notion of *contracting operators*. An outer-box contracting operator [10] narrows down the variable domains by discarding values that are locally inconsistent using *Box* consistency. In this paper we use a generic notion defined as follows:

Definition 1 (Outer-bound Contracting Operator, $\mathtt{OC}$). *An outer-bound contracting operator is a function* $\mathtt{OC} : \mathbb{I}^n \times \mathcal{P}(\mathbb{R}^n) \rightarrow \mathbb{I}^n \cup \{\emptyset\}$ *such that* $\forall \mathbf{B} \in \mathbb{I}^n$, $\rho \in \mathcal{P}(\mathbb{R}^n)$ *these properties hold:* [4]

$$\textit{(i)}\ \ \mathtt{OC}(\mathbf{B}, \rho) \subseteq \mathbf{B} \qquad \textit{(Contractiveness)}$$
$$\textit{(ii)}\ \mathtt{OC}(\mathbf{B}, \rho) \supseteq \mathbf{B} \cap \rho \quad \textit{(Completeness)}$$

In numerical domains, the outer-bound contracting operators usually enforce either *Box*, *Hull*, *kB* or *Bound* consistency [12, 5], generally referred to as bound-consistency in the rest of the paper. For simplicity, given a set of constraints $\mathcal{C}$, we use $\mathcal{C}$ instead of $\rho_\mathcal{C}$ in the notation of contracting operators.

Proposition 1. *Given a set of constraints, $\mathcal{C}$, and a bounding box, $\mathbf{B}$. The box $\mathbf{B}$ is completely infeasible (w.r.t. $\mathcal{C}$) if there is some $\mathtt{OC}$ operator that contracts $(\mathbf{B}, \mathcal{C})$ to an empty set, i.e. $\exists \mathtt{OC} : \mathtt{OC}(\mathbf{B}, \mathcal{C}) = \emptyset \Rightarrow \mathbf{B}$ is infeasible (w.r.t. $\mathcal{C}$).*

3.3 Union Approximations

In general, the computation of the best inner and outer approximations is intractable. Therefore, in this paper we consider the problem of computing inner and outer approximations of a relation $\rho \subseteq \mathbb{R}^n$ in the form of *unions of disjoint boxes*.

Definition 2 (Outer Union Approximation, $\mathbf{Union}^\mathcal{O}$). $\mathbf{Union}^\mathcal{O}(\rho)$ *is a set of disjoint boxes* $\mathcal{U} \in \mathcal{P}(\mathbb{I}^n)$ *such that* $pts(\mathcal{U}) \supseteq \rho$.

Definition 3 (Inner Union Approximation, $\mathbf{Union}^\mathcal{I}$). $\mathbf{Union}^\mathcal{I}(\rho)$ *is a set of disjoint boxes* $\mathcal{U} \in \mathcal{P}(\mathbb{I}^n)$ *such that* $pts(\mathcal{U}) \subseteq \rho$.

[4] $\mathcal{P}(S)$ denotes the power set of S, i.e., the set $\{A \mid A \subseteq S\}$.

Definition 4 (Undiscernible Union Approximation, $\mathbf{Union}^{\mathcal{U}}$). *$\mathbf{Union}^{\mathcal{U}}(\rho)$ corresponding to $\mathbf{Union}^{\mathcal{O}}(\rho)$ and $\mathbf{Union}^{\mathcal{I}}(\rho)$ is a set of disjoint boxes $\mathcal{U} \in \mathcal{P}(\mathbb{I}^n)$ such that $pts(\mathcal{U}) = cl(pts(\mathbf{Union}^{\mathcal{O}}(\rho)) \setminus pts(\mathbf{Union}^{\mathcal{I}}(\rho)))$.*[5]

Several authors have recently addressed the issue of computing $\mathbf{Union}^{\mathcal{O}}$ approximations. In [7], a recursive dichotomous split is performed on the variable domains. Each box obtained by splitting is tested for inclusion using interval arithmetic tools. The boxes obtained are hierarchically structured as 2^k*-trees*. The authors have demonstrated the practical usefulness of such techniques in robotics, automation and robust control. In [8], a similar algorithm is presented. However, only binary or ternary subsets of variables are considered when performing the splits. The approach is restricted to classes of problems with convexity properties. The technique proposed in [9] algebraically constructs the unions using Bernstein polynomials which makes it possible to use guaranteed algebraic inclusion tests for boxes. The approach is restricted to polynomial constraints. A technique of extending consistent domains of a particular class of constraints has also been proposed in [13]. Finally, [10] has addressed the issue of computing $\mathbf{Union}^{\mathcal{I}}$ for universally quantified constraints using *negation tests* (see Section 4.1), and [4] has extended the negation test in combination with enhanced splitting strategies to computing $\mathbf{Union}^{\mathcal{O}}$ for classic NCSPs. Hereafter, we give abstractions of conventions in computing union approximations where $\mathbf{Union}^{\mathcal{U}}$ is assumed to exist.

Definition 5 (Feasibility Checker, FC). *A feasibility checker is a function, $\mathtt{FC} : \mathbb{I}^n \times \mathcal{R}_n \rightarrow \{\mathtt{feasible}, \mathtt{infeasible}, \mathtt{unknown}\}$ such that:*

(i) $\mathtt{FC}(\mathbf{B}, \rho) = \mathtt{feasible} \Rightarrow \mathbf{B}|_{vars(\rho)} \subseteq \rho$
(ii) $\mathtt{FC}(\mathbf{B}, \rho) = \mathtt{infeasible} \Rightarrow \mathbf{B}|_{vars(\rho)} \subseteq \neg\rho$
(iii) $\mathtt{FC}(\mathbf{B}, \rho) = \mathtt{unknown} \wedge \mathbf{B}|_{vars(\rho)} \subset \mathbf{B}'|_{vars(\rho)} \Rightarrow \mathtt{FC}(\mathbf{B}', \rho) = \mathtt{unknown}$

Definition 6 (Interval-Based Precision). *Given an NCSP, $P = (\mathcal{V}, \mathcal{C}, \mathcal{D})$, a precision (vector), ε, and a feasibility checker, FC. A search technique which computes the union approximations is called* having the precision ε (w.r.t. FC) *if there is some set, $\mathcal{U}$, of disjoint boxes whose sizes are not greater than ε (component-wise) such that:*

$$\mathcal{U} \equiv \mathbf{Union}^{\mathcal{U}}; \forall \mathbf{B} \in \mathcal{U} : \mathtt{FC}(\mathbf{B}, \rho_P) = \mathtt{unknown} \tag{1}$$

4 EVR and Complementary-Boxing

Interval-based search techniques for NCSPs are essentially bisectional. Variables are instantiated using intervals. When the search reaches an interval that contains no solutions it backtracks, otherwise the interval is recursively split into two halves up to an established resolution. The most successful techniques enhance this process by applying an OC operator to the overall constraint system, after each split. In most known algorithms, the general policy is to perform splitting intervals until canonical intervals are reached or their widths are not greater than a predefined precision, i.e. it simply has

[5] cl is the standard closure operator. Informally, $\mathbf{Union}^{\mathcal{U}}(\rho)$ is a set of undiscernible boxes enclosing the boundary of ρ.

a predefined interval-based precision. This policy, referred to as DMBC (dichotomous maintaining bound-consistency) in the rest of the paper, generates verbose outer and inner union approximations. The first reason is that the orthogonal splitting policy introduces artificial convexity deficiencies and generates a significant number of nearly aligned boxes along boundaries of constraints. The second reason is that entirely feasible boxes might be unnecessarily split. The improvements we propose are presented in the two next subsections.

4.1 Better Splitting Decisions Using Complementary-Boxing

We now recall the techniques which construct $\mathbf{Union}^{\mathcal{I}}$ [10] and $\mathbf{Union}^{\mathcal{O}}$ [4] by soon isolating feasible regions under new abstract concepts in order to be integrated with the techniques described in Section 4.2. Given a relation, ρ, and a box, $\mathbf{B}$, the *negation test* performs a kind of OC operator on $(\mathbf{B}, \neg\rho)$. A kind of splitting operator (called $\texttt{ICAb3}_c$) splitting around a box obtained by a negation test for a numeric constraint and dichotomizing this box was proposed in [10]. Herein, the proposed negation approach (called ICAb5) to universally quantified constraints recursively performs $\texttt{ICAb3}_c$ on the first active constraint until a predefined interval-based precision reached. In [4], a similar splitting operator was employed, which is based on the negation test, herein called $\mathbf{B_q}$, for numeric constraints. However, the approach (called UCA6), which was proposed for NCSPs, computes negation tests for all constraints and then chooses the best for the split. In addition, UCA6 has memorized old $\mathbf{B_q}$'s for computing new $\mathbf{B_q}$'s and performed some mixed splitting strategies based on equalities/inequalities. Hereafter, we employ the negation test to define a contracting operator, called *Complementary-Box contracting operator* and a splitting operator, called *Box splitting operator*. The followings give abstract definitions of those ones.

Definition 7 (Complementary-Box Contracting Operator, CBC). *A Complementary-Box contracting operator is a function* $\texttt{CBC} : \mathbb{I}^n \times \mathcal{P}(\mathbb{R}^n) \rightarrow \mathbb{I}^n \cup \{\emptyset\}$ *such that* $\forall \mathbf{B} \in \mathbb{I}^n$, $\rho \in \mathcal{P}(\mathbb{R}^n)$ *these properties hold:*[6]

(i) $\texttt{CBC}(\mathbf{B}, \rho) \subseteq \mathbf{B}$ *(Contractiveness)*
(ii) $\mathbf{B} \setminus \texttt{CBC}(\mathbf{B}, \rho) \subseteq \rho$ *(Complementariness)*

A box resulting from the application of a CBC operator to a bounding box, $\mathbf{B}$, and a relation, ρ, is called a *Complementary-Box* with respect to ρ within $\mathbf{B}$. *Complementary-Boxing* refers the process of identifying the Complementary-Box. The following properties characterize CBC operators.

Proposition 2. *Given a set of constraints, $\mathcal{C}$, and a bounding box, $\mathbf{B}$. The box $\mathbf{B}$ is completely feasible (w.r.t. $\mathcal{C}$) if there is some* CBC *operator that contracts $(\mathbf{B}, \mathcal{C})$ to an empty set, i.e.* $\exists \texttt{CBC} : \texttt{CBC}(\mathbf{B}, \mathcal{C}) = \emptyset \Rightarrow \mathbf{B}$ *is feasible (w.r.t. $\mathcal{C}$).*

Proposition 3. *Given an* OC *operator. The function* $f : \mathbb{I}^n \times \mathcal{P}(\mathbb{R}^n) \rightarrow \mathbb{I}^n \cup \{\emptyset\}$ *defined by* $f(\mathbf{B}, \rho) = \texttt{OC}(\mathbf{B}, \neg\rho)$ *is a* CBC *operator.*

[6] In [11], this operator was named "Back-Boxing Contracting operator". In this paper, we change its name and notation to avoid confusions with the namesake given in [14].

Proof. By definition $\mathbf{B}_f = f(\mathbf{B}, \rho) = \texttt{OC}(\mathbf{B}, \neg\rho)$. The contractiveness of OC operators implies $\mathbf{B}_f \subseteq \mathbf{B}$, i.e. the contractiveness of CBC operators. In addition to the completeness of OC operators we have $\mathbf{B} \cap \neg\rho \subseteq \mathbf{B}_f \subseteq \mathbf{B} \Rightarrow \mathbf{B} \setminus \mathbf{B}_f \subseteq \rho$. This implies the complementariness of CBC operators.

Definition 8 (Monotonicity). *A contracting operator (*OC *or* CBC*),* μ*, is called* monotonous *if* $\mathbf{B} \subseteq \mathbf{B}' \Rightarrow \mu(\mathbf{B}, \rho) \subseteq \mu(\mathbf{B}', \rho)$.

Proposition 4. *Given a set of* n OC *operators,* $\{\texttt{OC}_k\}$*, and a set of* n CBC *operators,* $\{\texttt{CBC}_k\}$*, which enforce the monotonicity, where* $\texttt{OC}_k$ *and* $\texttt{CBC}_k$ *are defined in* k*-dimension (*$k \leq n$*). The function* $f : \mathbb{I}^n \times \mathcal{R}_n \rightarrow \{\texttt{feasible}, \texttt{infeasible}, \texttt{unknown}\}$ *defined by the following rules is a feasibility checker:*

(i) $f(\mathbf{B}, \rho) = \texttt{infeasible} \Leftrightarrow \texttt{OC}_k(\mathbf{B}|_{vars(\rho)}, \rho) = \emptyset$*, where* $k = |vars(\rho)|$
(ii) $f(\mathbf{B}, \rho) = \texttt{feasible} \Leftrightarrow \texttt{CBC}_k(\mathbf{B}|_{vars(\rho)}, \rho) = \emptyset$*, where* $k = |vars(\rho)|$

Proof of Proposition 4 is straightforward due to Proposition 1, Proposition 2 and Definition 5. Proposition 4 gives a way to construct a feasibility checker from OC and CBC operators enforcing the monotonicity. Proposition 2 and Proposition 3 imply that CBC operators can be constructed by OC operators and that *Complementary-Boxing* makes it possible to isolate completely feasible regions, $\mathbf{B} \setminus \texttt{CBC}(\mathbf{B}, \rho)$, w.r.t. some constraints. When applying a CBC operator to a box with respect to a constraint results in an empty set, it can be deduced that the box completely satisfies that constraint. We then define a splitting operator based on Complementary-Boxes, which consists of splitting around Complementary-Boxes, to isolate the feasible regions.

Definition 9 (Box Splitting Operator: BS**).** *A Box splitting operator is a function* BS $: \mathbb{I}^n \times \mathbb{I}^n \rightarrow \mathcal{P}(\mathbb{I}^n)$*, which takes as input two boxes such that the former contains the latter, splitting the outer box along some facets of the inner one.*[7]

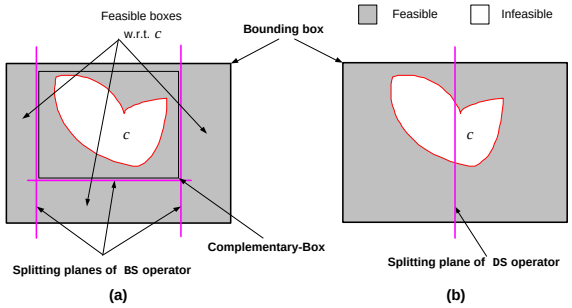

Fig. 2. (a) Box Splitting: splitting around a box (e.g. Complementary-Box); (b) Dichotomous Splitting: splitting the original domain of a variable into two halves

In the algorithm we propose, *Box splitting*, which partitions a region around a Complementary-Box, is applied in combination with dichotomous splitting. The latter is used either when Complementary-Boxing produces no reduction or when Box splitting results in too small boxes. Figure 2 illustrates the notion of Box splitting.

[7] This is a generic definition for partitioning a region around a box contained in it, given in [14].

4.2 Concise Approximations near Boundaries Using EVR

Stop Contracting over Inactive Dimensions. We first observe that a better alignment of boxes near boundaries of the solution space can be obtained by finely controlling applications of contracting operators during search. More precisely, whenever a dimension, i, of a box, $\mathbf{B}$, is bounded by ε_i, one can prevent contracting operators from contracting $\mathbf{B}$ over this dimension in order to obtain better alignments and performances. Given a box, $\mathbf{B}$, a constraint set, $\mathcal{C}$, and a precision vector, ε. A dimension, i, of $\mathbf{B}$ is called an *active dimension* if $\sup(\mathbf{B}|_i) - \inf(\mathbf{B}|_i) > \varepsilon_i$ and the corresponding variable, x_i, occurs in some active constraint. Otherwise, it is called an *inactive dimension*. A contracting operator which only works on active dimensions of boxes is called a *restricted-dimensional contracting operator* [11]. We denote by $\texttt{OC}_{\texttt{rd}}$ and $\texttt{CBC}_{\texttt{rd}}$ the restricted-dimensional contracting operators corresponding to OC and CBC operators, respectively.

Compacting Aligned Boxes. Once a better alignment is obtained, the question is how such a set of aligned boxes can be compacted into a smaller set. We propose to use the *Extreme Vertex Representation* (EVR) of orthogonal polyhedra for that purpose. The basic idea is that the finite unions of boxes delivered by a box-covering solver define *orthogonal polyhedra* for which improved representations can be used. An orthogonal polyhedron can be naturally represented as a finite union of disjoint boxes. Such a representation is called the *Disjoint Box Representation* (DBR) in computational geometry. The EVR is a way of compacting DBR [1–3]. We now recall some basic concepts related to EVR. We refer the reader to [2, 3] for further details. The concepts are presented for a particular type of orthogonal polyhedra, called *griddy polyhedra*. A griddy polyhedron [3] is generated from unit hyper-cubes with integer-valued vertices. Since arbitrary orthogonal polyhedra can be obtained from griddy ones by a bijection between vertex indices of the former and integer-valued vertices of the later, the results on EVR are not affected by this simplification. For simplicity, polyhedra are assumed to live inside a bounded subset $\mathbf{X} = [0, m]^d \subseteq \mathbb{R}^d$ (in fact, the results will hold also for $\mathbf{X} = \mathbb{R}^d_+$). Let $\mathbf{x} = (x_1, ..., x_d)$ be a grid point of the elementary grid $\mathcal{G} = \{0, 1, ..., m-1\}^d \subseteq \mathbb{N}^d$. For every point $\mathbf{x} \in \mathbf{X}$, $\lfloor \mathbf{x} \rfloor$ is the grid point corresponding to the integral part of the components of $\mathbf{x}$. The elementary box associated with $\mathbf{x}$ is the closed subset of $\mathbf{X}$ of the form $\mathbf{B}(\mathbf{x}) = [x_1, x_1 + 1] \times ... \times [x_d, x_d + 1]$. The set of all boxes is denoted by $\mathcal{B}$. A griddy polyhedron P is a union of elementary boxes, i.e. an elementary of $2^{\mathcal{B}}$.

Definition 10 (Color Function). *Let P be a griddy polyhedron. The color function $c : \mathbf{X} \to \{0, 1\}$ is defined as follows: if $\mathbf{x}$ is a grid point then $c(\mathbf{x}) = 1 \Leftrightarrow \mathbf{B}(\mathbf{x}) \subseteq P$; otherwise, $c(\mathbf{x}) = c(\lfloor \mathbf{x} \rfloor)$.*

We say that a grid point $\mathbf{x}$ is black (respectively, white) and that $\mathbf{B}(\mathbf{x})$ is full (respectively, empty) when $c(\mathbf{x}) = 1$ (respectively 0). A *canonical representation scheme* for $2^{\mathcal{B}}$ (or $2^{\mathcal{G}}$) is a set $\mathcal{E}$ of syntactic objects such that there is some bijective function $\Psi : \mathcal{E} \to 2^{\mathcal{B}}$.

Definition 11 (Extreme Vertex). *A grid point $\mathbf{x}$ is called an extreme if $\tau(\mathbf{x}) = 1$, where $\tau(\mathbf{x})$ denotes the parity of the number of black grid points in $\mathcal{N}(\mathbf{x}) = \{x_1 - 1, x_1\} \times ... \times \{x_d - 1, x_d\}$ (the neighborhood of $\mathbf{x}$).*

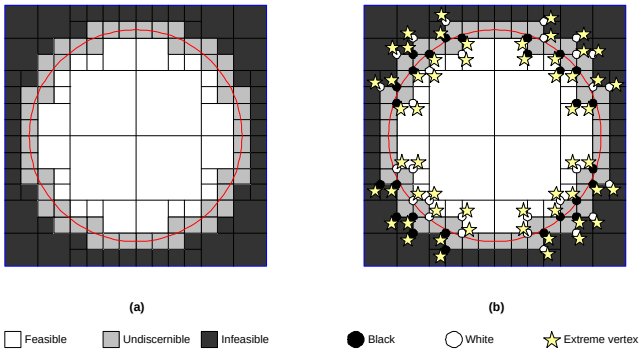

Fig. 3. (a) DBR and (b) Extreme vertices (with their colors) of union approximations

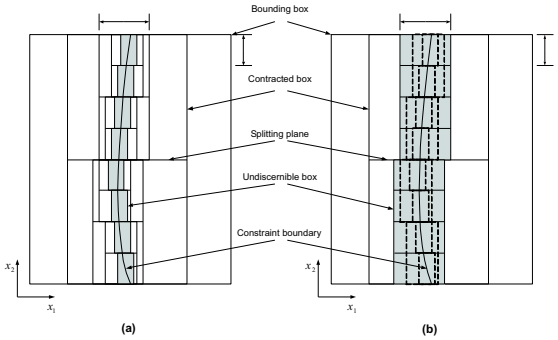

Fig. 4. Constraint boundary: (a) unaligned boxes produced by standard covering; (b) stop contracting over inactive dimensions and combine aligned boxes using EVR

Figure 3 illustrates the notion of EVR on a simple example. The fundamental theorem presented in [2, 3] shows that any griddy polyhedron can be canonically represented by the set of its extreme vertices (and their colors). The extreme vertex representation improves the space required for storing orthogonal polyhedra by an order of magnitude [1–3]. It also enables the design of efficient algorithms for fundamental operations on orthogonal polyhedra (e.g. membership, set-theoretic operations) [1–3]. In particular, effective transformation between DBR and EVR can be proposed for low-dimensional or small-size (i.e. m is small) polyhedra [1, 3]. For example, in three-dimension, the average experimental (time) complexity of converting an EVR to a DBR is far less than quadratic but slightly greater than linear in the number of extreme vertices [1]. Results in [3] also imply that, in fixed dimension, the time complexity of converting a DBR to an EVR using XOR operations is linear in the number of boxes in DBR. We propose to exploit these effective transformation schemes to produce a compact representation of contiguous aligned boxes using the following procedure:

1. Produce a better alignment of the boxes along the boundaries of constraints. This is done by preventing the unnecessary application of contracting operators over inactive dimensions. Figure 4 shows the better alignment produced for a set of nearly aligned boxes of an undiscernible approximation. The original set of 8 small

boxes (Figure 4-a) reduces to two groups of 4 aligned boxes (Figure 4-b) without altering the predefined interval-based precision.
2. The set of aligned boxes in each group, $\mathcal{S}_1$, is converted to EVR and then back to DBR to get a set of combined boxes, $\mathcal{S}_2$ (containing only one box in this case). Due to the properties of EVR, $\mathcal{S}_2$ is more concise than $\mathcal{S}_1$. Figure 4-b shows how this conversion procedure reduces the two groups of 4 boxes to two (gray) boxes.

Such a procedure can theoretically be applied in any dimension. Due to the efficiency of EVR in low dimension, we however restrict its application to low-dimensional or small-size regions of the search space in our implementation (see Section 5).

5 Algorithms

We now present an algorithm called UCA6-Plus (Figures 5 and 6). It takes as input a non-linear NCSP, $P = (\mathcal{V}, \mathcal{C}, \mathcal{D})$, and returns the $\mathbf{Union}^{\mathcal{I}}$ and $\mathbf{Union}^{\mathcal{U}}$ approximations of ρ_P. UCA6-Plus is an extension of UCA6 [4] to include the application of extreme vertex representation of orthogonal polyhedra and the use of restricted-dimensional contracting operators. Hereafter, $\mathbf{B}$ denotes a bounding box of the current subproblem. Originally, this bounding box is set to $\mathcal{D}$. For convenience, we denote $\mathbf{Union}^{\mathcal{X}}(\mathbf{B} \cap \rho_{\mathcal{C}})$ by $\mathbf{Union}^{\mathcal{X}}(\mathbf{B}, \mathcal{C})$, where $\mathcal{X} \in \{\mathcal{O}, \mathcal{I}, \mathcal{U}\}$. UCA6-Plus constructs the approximations $\mathbf{Union}^{\mathcal{I}}(\mathbf{B}, \mathcal{C})$ and $\mathbf{Union}^{\mathcal{U}}(\mathbf{B}, \mathcal{C})$, hence $\mathbf{Union}^{\mathcal{O}}(\mathbf{B}, \mathcal{C})$ can be computed as the union of these two approximations.

UCA6-Plus proceeds by recursively repeating three main steps:[8] *(i)* using $\mathtt{OC}_{\mathtt{rd}}$ operators to contract the current bounding box to a tighter one; *(ii)* using $\mathtt{CBC}_{\mathtt{rd}}$ operators to get a list of Complementary-Boxes w.r.t. each active constraint and w.r.t. the new bounding box, the constraints that make empty Complementary-Boxes are removed; finally, *(iii)* combining Dichotomous splitting (DS) with Box splitting (BS) for the whole set of active constraints. In practice, equalities usually define surfaces, we then do not need to perform step *(ii)* for such constraints (see Figure 4).

$getSplit()$ is a function returning the splitting mode to be used for splitting the current box. The current splitting mode can be inferred from the history of the current box (e.g. the splitting mode of the parent box). In contrast to DMBC, the DS operator used for UCA6-Plus only tries to dichotomize over the active dimensions. This avoids splitting boxes into a huge number of tiny boxes. Moreover, in UCA6-Plus constraints are removed gradually whenever an empty Complementary-Box is computed w.r.t. those constraints. The dimension with the greatest size is preferred for DS. For the pruning to be efficient, BS splits along some facet of a Complementary-Box only if that produces sufficiently large boxes, the Complementary-Box itself excepted. This estimation is done using a pre-determined *fragmentation ratio*. In Figure 5 and 6, $\mathcal{S}_{inn}$ and $\mathcal{S}_{und}$ are global variables denoting the set of boxes (and active constraints, if exist) of $\mathbf{Union}^{\mathcal{I}}(\mathbf{B}_0, \mathcal{C}_0)$ and $\mathbf{Union}^{\mathcal{U}}(\mathbf{B}_0, \mathcal{C}_0)$, respectively. We use a list, $WList$, to store the subproblems waiting to be processed.

$chooseTheBest()$ is a function choosing the best Complementary-Box and the respective constraint based on some criteria to maximize the space surrounding the

[8] UCA6 does the similar steps, except $solveQuickly()$, but it uses OC and CBC operators.

function $UCA6Plus(\mathbf{B}_0, \mathcal{C}_0, \varepsilon, \texttt{FC}, \texttt{OC}_{\texttt{rd}}, \texttt{CBC}_{\texttt{rd}}, D_{stop})$
  $\mathcal{S}_{inn} := \emptyset$; $\mathcal{S}_{und} := \emptyset$; $WList := \emptyset$; /* $\mathcal{S}_{inn}, \mathcal{S}_{und}$ are global lists to be return*/
  if $solveQuickly(\mathbf{B}_0, \mathcal{C}_0, \varepsilon, \texttt{FC}, \texttt{OC}_{\texttt{rd}}, \texttt{CBC}_{\texttt{rd}}, WList, D_{stop})$ **then return**;
  while $WList \neq \emptyset$ **do** /* Waiting list of subproblems is not empty */
    $(< \mathbf{B}, \mathcal{C} >, \{\mathbf{CB'}_c\}) := get(WList)$; /* set $\{\mathbf{CB'}_c\}$ was optionally memorized*/
    for each $c \in \mathcal{C}'$ **do** /* $\mathcal{C}' \subseteq \mathcal{C}$, it is set to $\mathcal{C}$ or dynamically computed based on $\{\mathbf{CB'}_c\}$ */
      $\mathbf{CB}_c := \texttt{CBC}_{\texttt{rd}}(\mathbf{B}, c)$ **or** $\texttt{CBC}_{\texttt{rd}}(\mathbf{B} \cap \mathbf{CB'}_c, c)$ **or** $\mathbf{B} \cap \mathbf{CB'}_c$; /* depends on $\mathcal{C}'$ */
      if $\mathbf{CB}_c = \emptyset$ **then** $\mathcal{C} := \mathcal{C} \setminus \{c\}$; /* c is redundant in $\mathbf{B}$ (Proposition 2) */
      if $\mathbf{CB}_c = \emptyset$ **or** $\mathbf{CB}_c = \mathbf{B}$ **then** $\mathcal{C}' := \mathcal{C}' \setminus \{c\}$;
    endforeach
    if $\mathcal{C} = \emptyset$ **then**
      $store(\mathbf{S}_{inn}, \mathbf{B})$; /* No active constraint, $\mathbf{B}$ is feasible */
      continue while; /* do the next loop of **while** */
    endif
    $Splitter := getSplit()$; /* Get a splitting mode, heuristics can be used */
    if $Splitter = \texttt{BS}$ **then** /* The splitting mode is Box Splitting */
      $\mathbf{CB}_c := chooseTheBest(\mathbf{B}, \{\mathbf{CB}_c \mid c \in \mathcal{C}'\})$; /* e.g. maximize surrounding regions*/
      $\mathbf{CB} := enlarge(\mathbf{B}, \mathbf{CB}_c, ZeroPlus)$; /* $\mathbf{CB}_c \subset \mathbf{CB} \subseteq \mathbf{B}$ or $\mathbf{CB}_c = \mathbf{CB} = \mathbf{B}$ */
      $< \mathbf{B}_1, \ldots, \mathbf{B}_k >:= \texttt{BS}(\mathbf{B}, \mathbf{CB})$; /* Box Splitting: failed or $\exists \mathbf{B}_i \supseteq \mathbf{CB}$ */
      if $\mathcal{C}' = \emptyset$ **or** BS *failed* **then** $Splitter := \texttt{DS}$;
    endif
    if $Splitter = \texttt{DS}$ **then** $< \mathbf{B}_1, \ldots, \mathbf{B}_k >:= \texttt{DS}(\mathbf{B})$; /* Dichotomous Splitting */
    for $i = 1$ **to** k **do**
      $\mathcal{C}_i := \mathcal{C}$; $\mathcal{C}'_i := \mathcal{C}'$;
      if $Splitter = \texttt{BS}$ **and** $\mathbf{B}_i \cap \mathbf{CB}_c = \emptyset$ **then**
        $\mathcal{C}_i := \mathcal{C}_i \setminus \{c\}$; $\mathcal{C}'_i := \mathcal{C}'_i \setminus \{c\}$ /* c is redundant (Complementariness of CBC) */
        if $\mathcal{C}_i = \emptyset$ **then**
          $store(\mathbf{S}_{inn}, \mathbf{B}_i)$; /* No active constraint, $\mathbf{B}_i$ is feasible */
          continue for; /* do the next loop of **for** */
        endif
      endif
      $solvable := solveQuickly(\mathbf{B}_i, \mathcal{C}_i, \varepsilon, \texttt{FC}, \texttt{OC}_{\texttt{rd}}, \texttt{CBC}_{\texttt{rd}}, WList, D_{stop})$;
      if not $solvable$ **then** $memorize(WList \leftarrow \{\mathbf{CB}_c \mid c \in \mathcal{C}'_i\})$; /* This is optional */
    endfor
  endwhile
end /* $UCA6Plus$ */

Fig. 5. The algorithm UCA6-Plus

Complementary-Box. The other Complementary-Boxes can be memorized for improving the Complementary-Boxing of child search nodes. $enlarge(\mathbf{B}, \mathbf{CB}_c, ZeroPlus)$ is a function extending $\mathbf{CB}_c$ to $\mathbf{CB}$ by $ZeroPlus$ (considered as a sufficiently small positive number) such that the result is still in $\mathbf{B}$. This guarantees that no point satisfying $\neg c$ is on the boundary of $\mathbf{CB}$ except the points on the boundary of $\mathbf{B}$.

The function $solveQuickly()$ (Figure 6) constructs $\mathbf{Union}^{\mathcal{I}}$ and $\mathbf{Union}^{\mathcal{U}}$ approximations for low-dimensional subproblems whose bounding box has D_{stop} active dimensions at the most. The output is compacted using EVR. The use of $\texttt{OC}_{\texttt{rd}}$ and $\texttt{CBC}_{\texttt{rd}}$ operators for the purpose of narrowing produces a better alignment of boxes along the boundaries. This allows for using EVR to combine the contiguous aligned boxes.

function $solveQuickly(\mathbf{B}, \mathcal{C}, \varepsilon, \texttt{FC}, \texttt{OC}_{\texttt{rd}}, \texttt{CBC}_{\texttt{rd}}, WList, D_{stop})$
  $\mathbf{B}' := \texttt{OC}_{\texttt{rd}}(\mathbf{B}, \mathcal{C})$;
  if $\mathbf{B}' = \emptyset$ **then return** TRUE; /* $\mathbf{B}$ is infeasible, the problem has been solved */
  if $isEpsilonBox(\mathbf{B}', \mathcal{C}, \varepsilon, \texttt{FC})$ **then return** TRUE; /* The problem has been solved */
  if $\mathbf{B}'$ *has at most* D_{stop} *active dimensions* **then** /* Resort to another technique */
    $< \mathcal{S}'_{inn}, \mathcal{S}'_{und} >:= DimStopSolver(\mathbf{B}', \mathcal{C}, \varepsilon, \texttt{FC}, \texttt{OC}_{\texttt{rd}}, \texttt{CBC}_{\texttt{rd}})$;
    /* $combine()$ does the conversions DBR → EVR → DBR in D_{stop}-dimension */
    $store(\mathcal{S}_{inn}, combine(\mathcal{S}'_{inn}))$; /* Store in the global list of feasible boxes */
    $store(\mathcal{S}_{und}, combine(\mathcal{S}'_{und}))$; /* Store in the global list of undiscernible boxes */
    return TRUE; /* The problem has been solved */
  endif
  $put(WList \leftarrow < \mathbf{B}', \mathcal{C} >)$; /* put the subproblem into the waiting list */
  return FALSE; /* The problem has not been solved yet */
end /* $solveQuickly$ */

function $isEpsilonBox(\mathbf{B}, \mathcal{C}, \varepsilon, \texttt{FC})$
  if $\mathbf{B}$ *has no active dimension* **then** /* $\mathbf{B}$ is an ε-bounded box in the variable space of $\mathcal{C}$ */
    switch $\texttt{FC}(\mathbf{B}, \rho_{\mathcal{C}})$ **:** /* Identify the feasibility of $\mathbf{B}$ w.r.t. $\mathcal{C}$ */
      case `feasible` : $store(\mathcal{S}_{inn}, \mathbf{B})$; /* $\mathbf{B}$ is feasible, store it */
      case `unknown` : $store(\mathcal{S}_{und}, < \mathbf{B}, \mathcal{C} >)$; /* $\mathbf{B}$ is undiscernible, store it */
    endswitch
    return TRUE;
  endif
  return FALSE;
end /* $isEpsilonBox$ */

Fig. 6. The function $solveQuickly()$

$solveQuickly()$ uses a feasibility checker, called FC, to check if an ε-bounded box is `feasible`, `infeasible` or `unknown` (then called *undiscernible*). Though the FC in our implementation uses OC and CBC operators for checking the feasibility of ε-bounded boxes, it is however not restricted to a specific feasibility checker.

For efficiency, $solveQuickly()$ allows resorting to a secondary search technique, $DimStopSolver()$, to solve the low-dimensional subproblems whose bounding box has at most D_{stop} active dimensions. Good candidates for small D_{stop} can be either the 2^k-tree based solver presented in [8] or a simple grid-based solver[9]. Variants of DMBC or UCA6 using the restricted-dimensional contracting operators can alternatively be used. For a given subproblem, $DimStopSolver()$ constructs the sets $\mathcal{S}'_{inn}$ and $\mathcal{S}'_{und}$ which are $\mathbf{Union}^{\mathcal{I}}$ and $\mathbf{Union}^{\mathcal{U}}$ of the subproblem, respectively. These two sets are represented in DBR. They are converted to EVR and then back to DBR to combine each group of contiguous aligned boxes into a bigger equivalent box. This operation is represented by the function $combine()$ in Figure 6.

Proposition 5. *Given a feasibility checker,* FC*. The algorithm* UCA6-Plus *computes the union approximations and has a predefined interval-based precision ε w.r.t.* FC*.*

[9] A simple grid-based solver splits variable domains into a grid and solves the problem in each grid element.

Sketch of proof: The conclusion can be deduced, informally, from the observations: *(i)* $\mathbf{Union}^{\mathcal{I}}$ and $\mathbf{Union}^{\mathcal{U}}$ are disjoint, $\mathbf{Union}^{\mathcal{O}} = \mathbf{Union}^{\mathcal{I}} \cup \mathbf{Union}^{\mathcal{U}}$; *(ii)* No solution is lost due to the completeness of `OC` operator; *(iii)* All the inner boxes (i.e. the boxes in $\mathbf{Union}^{\mathcal{I}}$) are sound due to Definition 7 and Proposition 2; *(iv)* $\mathbf{Union}^{\mathcal{U}}$ is equivalent to a union of the boxes (before applying EVR-DBR conversions) which have no active dimension (see the function $isEpsilonBox()$ in Figure 6) and cannot be classified as `feasible` or `infeasible` using the feasibility checker `FC`. That is due to the properties of EVR-DBR conversion and due to the fact that each box which has no active dimension has the same feasibility (under feasibility checkers) with its projection, an ε-bounded box, on the space defined by all the variables in the active constraints.

6 Preliminary Experiments

We now present a preliminary evaluation on the following small set of typical problems (with different properties of constraints and solution space).

CD (Column Design), **FD** (Fatigue Design) and **TD** (Truss Design) are three engineering design examples. Their complete descriptions are available at *http://imacsg4.epfl.ch:8080/PGSL/* and *http://liawww.epfl.ch/Coconut-benchs/*. In Table 1, the considered instance of **CD** is the one that finds $(a, b, e) \in [0.01, 2] \times [0.01, 1] \times [0.05, 0.1]$ given that $P = 400kN$, $H = 6m$ and $L = 1m$, where a and b are in meter, e is in decimeter. The **FD** instance considered is the one that finds $(L, qf, Z) \in [10, 30] \times [70, 90] \times [0.1, 10]$ for a given number of years to fatigue failure $years = 100$, where Z is scaled up 100 times in unit. The considered instance of **TD** is a simplified one that finds $x_1, y_1 \in [0.01, 10]$. **WP** and **P3** were described in Section 2. **P2** = $\{x^2 \leq y, \ln y + 1 \geq z, xz \leq 1, x \in [-15, 15], y \in [1, 200], z \in [-10, 10]\}$.

For evaluation purposes, we have implemented the algorithms `DMBC`, `UCA6`, `UCA6-Plus` using the same data structure and the same standard contracting operators. We have also implemented a direct migration, called `UCA5`, of `ICAb5` in [10] to solving NCSPs, and a version of `DMBC` including the negation test. This enhanced `DMBC`, called `DMBC+`, can therefore check whether a box is completely feasible or not. Our experiments discarded `DMBC` as a reasonable candidate for this kind of problems. It usually produces a huge number of boxes, each of which is ε-bounded.

The tests shown in Table 1 ran with *fragmentation ratio* = 0.25, $D_{stop} = 1$, and `FC` given by Proposition 4 (with the precision set to 1). The `OC` operator was implemented with *IloGenerateBounds* in ILOG Solver 5.2 [6]. The precision of the contracting operators used for narrowing bounding boxes was set to 1. Let ε be the interval-based precision for the algorithms. The secondary search technique used for `UCA6-Plus` is a simple grid-based one. The terms (`DS`) and (`BS+DS`) indicate the splitting strategies enforced upon the algorithms, `MEM` means the memorization of Complementary-Boxes. Each cell in the table has two rows. The first shows time and ratio of inner volume to total volume, the second the number of boxes in $\mathbf{Union}^{\mathcal{I}}$ and $\mathbf{Union}^{\mathcal{U}}$, respectively.

Other tests on tens of similar problems show that the best gains, in running time and number of boxes, of the algorithms `UCA5`, `UCA6` and `UCA6-Plus` over `DMBC+` are obtained for problems with low-arity constraints (w.r.t. arity of problems, e.g. **P2**, **P3**). In all the tests, `UCA6-Plus`(`BS+DS`) with either `MEM` or ¬`MEM` is better than the other algorithms in running time and number of boxes. The best gains are obtained

Table 1. Typical test results

Prob.	ε	DMBC+ (DS), ¬MEM	UCA6 (DS), MEM	UCA6-Plus (DS), MEM	UCA5 (BS+DS), ¬MEM	UCA6 (BS+DS), MEM	UCA6-Plus (BS+DS), ¬MEM
WP	0.1	22.07s / 0.992 2753 / 2620	6.73s / 0.992 2753 / 2620	5.98s / 0.991 2489 / 2147	5.11s / 0.994 1738 / 2788	4.77s / 0.994 1573 / 2791	3.97s / 0.993 1176 / 1585
TD	0.01	81.53s / 0.997 3900 / 2917	26.45s / 0.997 3900 / 2917	26.01s / 0.997 3895 / 1970	14.96s / 0.999 2832 / 3270	13.43s / 0.999 1313 / 3496	3.59s / 0.998 53 / 50
P3	0.1	>10h / n/a >110000 / 150000	615.98s / 0.907 33398 / 38006	530.16s / 0.912 30418 / 28229	87.09s / 0.980 10784 / 29888	135.28s / 0.980 12113 / 38808	1.41s / 0.919 406 / 970
P2	0.1	>10h / n/a >120000 / 180000	4959.76s / 0.973 108701 / 100027	4433.06s / 0.974 106320 / 78755	281.51s / 0.995 21872 / 55901	293.09s / 0.995 18722 / 55063	7.32s / 0.975 1873 / 3225
FD	0.1	3878.47s / 0.987 42178 / 66940	1278.82s / 0.987 42178 / 66940	740.05s / 0.984 42084 / 47138	394.29s / 0.992 51882 / 65536	439.07s / 0.992 26378 / 70170	211.91s / 0.986 10341 / 35126
CD	0.01	1308.63s / 0.638 8957 / 22132	468.21s / 0.638 8873 / 21974	389.12s / 0.619 8354 / 14857	538.21s / 0.830 12729 / 25079	493.57s / 0.828 9922 / 25344	352.96s / 0.616 2826 / 12922

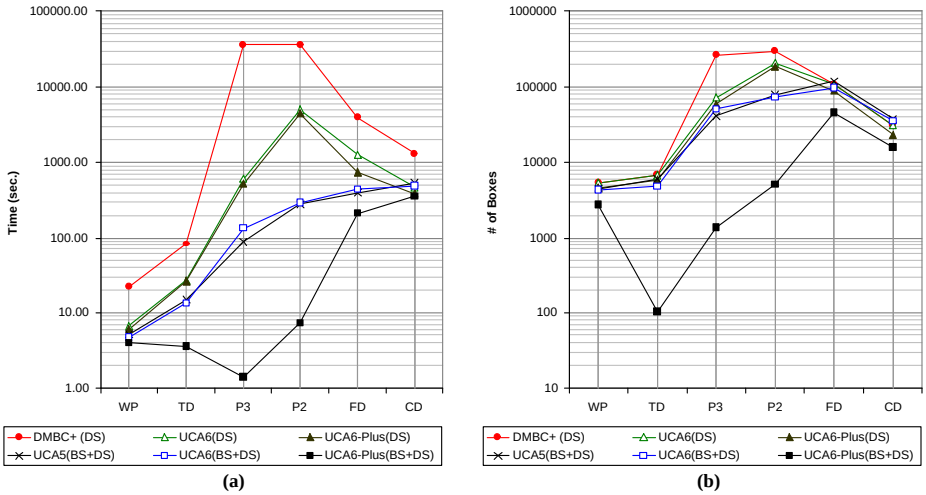

Fig. 7. Logarithmic charts for (a) running time, and (b) total number of boxes

when the non-linear constraints contain some nearly axis-parallel regions (e.g. **P2**, **P3**, **TD**). UCA6-Plus(BS+DS) is slightly less accurate than UCA5 and UCA6 in volume measure, but the situation is improved when ε is reduced (i.e. in higher precision).

The preliminary experiments are therefore encouraging enough to warrant further investigations and in-depth evaluations based on other combinations of the control parameters and higher values of D_{stop} in combination with variants of 2^k-tree solvers.

7 Conclusion

Interval-constraint based solvers are usually designed to deliver point-wise solutions. They are consequently inadequate for solving numerical problems with non-isolated solutions. In this paper, we propose a box-covering technique for computing inner and outer approximations of NCSPs that remedies this state of affairs. The approach works for general non-linear NCSPs with mixed equality/inequality constraints, especially for inequalities. It combines the compactness of extreme vertex representation of orthogonal polyhedra with an adapted splitting policy. This allows for gains in performance and

space requirements. The quality of the output is also enhanced by *guaranteed feasible boxes* when solution space is solid. In practice, NCSPs with non-isolated solutions often occur as subproblems of higher-dimensional problems. For such purposes, the feasibility checker can be relaxed or run with low precision, running time will hence be much improved. In future work, we therefore plan to investigate collaboration strategies between our techniques and standard point-wise interval-based solvers. We will also study alternative implementation schemes purely based on the extreme vertex representation of orthogonal polyhedra, i.e. those do not perform $\texttt{BS}(\mathbf{B}, \mathbf{CB})$ but consider $\mathbf{B} \setminus \mathbf{CB}$ as an orthogonal polyhedron in EVR, especially for consistencies and propagations.

8 Acknowledgments

We would like to thank Boi Faltings and Arnold Neumaier for their valuable discussions. Support for this research is provided by the European Commission and the Swiss Federal Education and Science Office (OFES) through the COCONUT project (IST-2000-26063).

References

1. Aguilera, A.: Orthogonal Polyhedra: Study and Application. PhD thesis, Universitat Politècnica de Catalunya, Barcelona, Spain (1998)
2. Bournez, O., Maler, O., Pnueli, A.: Orthogonal Polyhedra: Representation and Computation. F. Vaandrager and J. Van Schuppen (Eds.), Hybrid Systems: Computation and Control, LNCS 1569 (1999) 46–60 Springer.
3. Bournez, O., Maler, O.: On the Representation of Timed Polyhedra. In: Proceedings of International Colloquium on Automata Languages and Programming (ICALP'2000). (2000)
4. Silaghi, M.C., Sam-Haroud, D., Faltings, B.: Search techniques for non-linear CSPs with inequalities. In: Proceedings of the 14th Canadian Conference on AI. (2001)
5. Van Hentenryck, P.: A gentle introduction to Numerica. (1998)
6. ILOG: ILOG Solver. Reference Manual. (2002)
7. Jaulin, L.: Solution Globale et Guarantie de problèmes ensemblistes: application à l'estimation non linéaire et à la commande robuste. PhD thesis, Université Paris-Sud, Orsay (1994)
8. Sam-Haroud, D., Faltings, B.: Consistency techniques for continuous constraints. Constraints **1** (1996) 85–118
9. Garloff, J., Graf, B.: Solving strict polynomial inequalities by Bernstein expansion. In: Symbolic Methods in Control System Analysis and Design. (1999) 339–352
10. Benhamou, F., Goualard, F.: Universally Quantified Interval Constraints. In: Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming (CP'2000). (2000) 67–82
11. Vu, X.H., Sam-Haroud, D., Silaghi, M.C.: Approximation Techniques for Non-linear Problems with Continuum of Solutions. In: Proceedings of The 5th International Symposium on Abstraction, Reformulation and Approximation (SARA'2002), Canada (2002) 224–241
12. Lhomme, O.: Consistency techniques for numeric CSPs. In: Proceedings of IJCAI'93. (1993)
13. Collavizza, H., Delobel, F., Rueher, M.: Extending consistent domains of numeric CSP. In: Proceedings of IJCAI'99. (1999)
14. Van Iwaarden, R.J.: An Improved Unconstrainted Global Optimization Algorithm. PhD thesis, University of Colorado at Denver, USA (1996)