Approximation Techniques for Non-linear Problems with Continuum of Solutions

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

Artificial Intelligence Laboratory, Institute of Core Computing Science,
School of Computer and Communication Sciences, Swiss Federal Institute of Technology,
CH-1015, Lausanne, Switzerland
{xuan-ha.vu, jamila.sam, marius.silaghi}@epfl.ch
http://liawww.epfl.ch

Abstract. Most of the working solvers for numerical constraint satisfaction problems (NCSPs) are designed to delivering *point-wise* solutions with an arbitrary accuracy. When there is a *continuum of feasible points* this might lead to prohibitively verbose representations of the output. In many practical applications, such large sets of solutions express equally relevant alternatives which need to be identified as completely as possible. The goal of this paper is to show that by using appropriate approximation techniques, explicit representations of the solution sets, preserving both accuracy and completeness, can still be proposed for NCSPs with continuum of solutions. We present a technique for constructing *concise* inner and outer approximations as unions of interval boxes. The proposed technique combines a *new splitting strategy* with the *extreme vertex representation* of orthogonal polyhedra [1–3], as defined in computational geometry. This allows for compacting the representation of the approximations and improves efficiency.

1 Introduction

Numerical constraints can naturally model a wide range of real-world problems. In practice, process descriptions, cost restrictions, chemical or mechanical models are most often expressed using this type of constraints. A numerical constraint satisfaction problem (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 constraints $\mathcal{C}$. The constraints can be equalities or inequalities of arbitrary type and arity, usually expressed using arithmetic expressions. The goal is to assign values to the variables so that all the constraints are satisfied. Such an assignment is then called a solution. The completeness of a solving procedure means its ability to find a solution to the NCSP if any, or else, to prove that there are no solutions to the problem. Completeness is essential in many real-world situations since it is the only way to guarantee that all inconsistencies are avoided, that all relevant alternatives can be provided and that an eventual global optima can be identified. When devising complete solving techniques for NCSPs, a fundamental issue is the representation of the actual solution sets. The spectrum of possible representations ranges from the implicit one, given by the arithmetic expressions of constraints, to the explicit one, given by the enumeration of all individual solutions. The former representation is

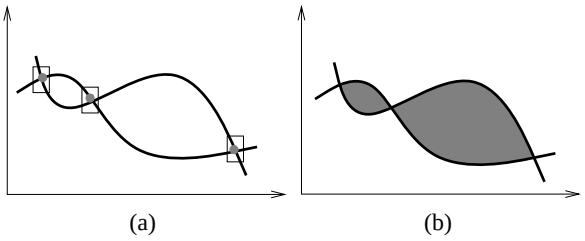

Fig. 1. (a) Numerical CSP with three point-wise solutions (grey dots); (b) Numerical CSP with a continuum of solutions (grey regions)

traditionally used by mathematical programming solvers. It is compact but difficult to query. As a consequence, even the state-of-the-art solving techniques cannot guarantee completeness in the general case [4]. The second representation is usually constructed by constraint programming solvers [5, 6]. It is complete and trivial to query but can be exceedingly verbose, and therefore unpractical, when the NCSP has a continuum of solutions (see Figure 1). Such a situation often occurs in real-world applications since under-constraint problems or problems with inequalities are ubiquitous in practice. The goal of this paper is to show that by using appropriate approximation techniques, explicit representations, preserving both accuracy and completeness, can still be proposed for NCSPs with non-isolated solutions. We propose to use the *Extreme Vertex Representation* (EVR) of orthogonal polyhedra [1–3] as defined in computational geometry, coupled with adapted branching strategies, to compute inner and outer approximations of the solution sets under the form of *unions of interval boxes*. The resulting technique applies to general constraint systems. The preliminary experiments show that it improves efficiency as well as the compactness and quality of the explicit representation of the solution sets.

2 Background and Motivation

We address the issue of solving non-linear NCSPs with *continuum of solutions* (Figure 1). In its most general form, a continuum of solutions 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 a numerical CSP, where the domains of the variables are intervals over the reals, and generate a set of boxes which *conservatively* enclose each solution (no solution is lost). They have proven particularly efficient in solving challenging instances of numerical CSPs with non-linear constraints but are commonly designed to deliver point-wise solutions. As a consequence, when applied to our target problems, the approximations they provide for the complete solution set are, in most cases, prohibitively verbose. As an example, let us consider the following NCSP with four non-linear inequality constraints and three 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 point-wise techniques,[1] the computation had to be stopped after 1 hour and produced more than 90000 small boxes.[2] A natural alternative to the point-wise approach is to try to cover the spectrum of non-isolated solutions using a *reduced* number of subsets from $\mathbb{R}^n$. Usually, these subsets are chosen with known and simple properties (e.g. interval boxes, polytopes, ellipsoids) [7]. In recent years, several authors have proposed set covering algorithms with interval boxes [7–10]. These algorithms are based on domain splitting and have one of the following limitations: they are designed for inequality constraints only [8–10], they only apply to polynomials [9], they uniformly enforce dichotomous splitting on all variables [7]. Moreover, most of these techniques produce verbose approximations of the boundaries [7–9]. As a consequence, either their applicability is restricted or the tractability limits are rapidly reached. The alternative technique we propose is based on the following observations. Firstly, 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 an alternative scheme based on splitting around the negation of feasible regions. Secondly, the union of boxes produced by the complete solving of numerical CSPs with continuum of solutions can be seen as an orthogonal polyhedron. Enhanced representations from computational geometry can be used to reduce the verbosity of such geometrical object. We propose to use the *extreme vertex representation* of orthogonal polyhedra [1–3] for this purpose.

3 Definitions and Notations

3.1 Interval Arithmetic

The finite nature of computers precludes an exact representation of the reals. The set $\mathbb{R}$, extended with the two infinity symbols, and then denoted by $\mathbb{R}_\infty = \mathbb{R} \cup \{-\infty, +\infty\}$, is in practice approximated by a finite subset $\mathbb{F}_\infty$ containing $-\infty$, $+\infty$ and 0. In interval-based constraint solvers, $\mathbb{F}_\infty$ usually corresponds to the floating-point numbers used in the implementation. Let $<$ be the natural extension to $\mathbb{R}_\infty$ of the order relation $<$ over $\mathbb{R}$. For each l in $\mathbb{F}_\infty$, we denote by l^+ the smallest element in $\mathbb{F}_\infty$ greater than l, and by l^- the greatest element in $\mathbb{F}_\infty$ smaller than l.

The set of intervals with bounds in $\mathbb{F}_\infty$, denoted by $\mathbb{I}$, is ordered by set inclusion. In the rest of the paper, intervals are written uppercase, reals or floats are lowercase, vectors in boldface and sets in uppercase calligraphic letters. An *interval box* (henceforth referred to *box*) $\mathbf{B} = I_1 \times \ldots \times I_n$ is a Cartesian product of n intervals in $\mathbb{I}$. A *canonical interval* is a non-empty interval of the form $[l..l]$ or of the form $[l..l^+]$. A *canonical box* is a Cartesian product of canonical intervals.

We use the following notations for set theoretic operations on boxes: $\mathbf{A} \sqcup \mathbf{B} = \mathbf{A} \cup \mathbf{B}$, $\mathbf{A} \sqcap \mathbf{B} = cl(int(\mathbf{A}) \cap int(\mathbf{B}))$, $\neg\mathbf{A} = cl(\sim \mathbf{A})$, where cl and int are the topological closure and interior operations, and $\sim$ is the complementary operation.

[1] The implementation was based on ILOG solver 5.1 (see Section 6).

[2] The alternative technique we propose could reduce the complete output to 1373 boxes and produced the result in 5.63 seconds (see Table 1).

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 satisfying c. The relation defined by the negation, $\neg c$, of c is given by $\mathbb{R}^n \setminus \rho_c$. 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, or more coarsely by the smallest box, containing it. In the rest of the paper, we will use the following definitions and notations:

Definition 1 (The Minimal Outer Box, $\mathbf{OB}$). *Let ρ be a relation from $\mathbb{R}^n$, the minimal outer box of ρ, denoted by $\mathbf{OB}(\rho)$, is defined by:*

$$\mathbf{OB}(\rho) \in \mathbb{I}^n, \mathbf{OB}(\rho) \supseteq \rho : \forall \mathbf{B} \in \mathbb{I}^n, \mathbf{B} \supseteq \rho \Rightarrow \mathbf{OB}(\rho) \subseteq \mathbf{B} \tag{1}$$

Definition 2 (The Best Outer Approximation, $\mathbf{OA}$). *Let ρ be a relation from $\mathbb{R}^n$, the best outer approximation of ρ, denoted by $\mathbf{OA}(\rho)$, is defined by:*

$$\mathbf{OA}(\rho) = \bigcup_{r \in \rho} \mathbf{OB}(\{r\}) \tag{2}$$

Definition 3 (The Best Inner Approximation, $\mathbf{IA}$). *Let ρ be a relation from $\mathbb{R}^n$, the best inner approximation of ρ, denoted by $\mathbf{IA}(\rho)$, is defined by:*

$$\mathbf{IA}(\rho) = \bigcup_{\mathbf{B} \in \mathbb{I}^n, \mathbf{B} \subseteq \rho} \mathbf{B} \tag{3}$$

Definition 4 (The Best Undiscernible Approximation, $\mathbf{UA}$). *Let ρ be a relation from $\mathbb{R}^n$, the best undiscernible approximation of ρ, denoted by $\mathbf{UA}(\rho)$, is the difference between $\mathbf{OA}(\rho)$ and $\mathbf{IA}(\rho)$: $\mathbf{UA}(\rho) = \mathbf{OA}(\rho) \setminus \mathbf{IA}(\rho)$.*

Figure 2 illustrates Definitions 1, 2, 3 and 4 on a simple example. The relation to approximate contains all the points lying inside the circle and on its boundary.

Proposition 1. *Given a relation, $\rho \subseteq \mathbb{R}^n$, these properties hold: (1) $\mathbf{OB}(\rho)$ exists uniquely; (2) $\mathbf{IA}(\rho) \subseteq \rho \subseteq \mathbf{OA}(\rho)$; (3) $\mathbf{OA}(\rho)$, $\mathbf{UA}(\rho)$ are minimal and $\mathbf{IA}(\rho)$ is maximal, i.e. $\forall \mathcal{S}_i, \mathcal{S}_o \in \mathcal{P}(\mathbb{I}^n), \mathcal{U}_i = \bigcup_{\mathbf{B} \in \mathcal{S}_i} \mathbf{B}, \mathcal{U}_o = \bigcup_{\mathbf{B} \in \mathcal{S}_o} \mathbf{B}$:*

$$\mathcal{U}_i \subseteq \rho \subseteq \mathcal{U}_o \Rightarrow \mathcal{U}_i \subseteq \mathbf{IA}(\rho) \subseteq \mathbf{OA}(\rho) \subseteq \mathcal{U}_o, \mathbf{UA}(\rho) \subseteq \mathcal{U}_o \setminus \mathcal{U}_i \tag{4}$$

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. In this paper we use the notion of outer-bound contracting operator, defined as follows:

Definition 5 (Outer-bound Contracting Operator, $\mathbf{OC}$). *An outer-bound contracting operator is a function $\mathbf{OC} : \mathbb{I}^n \times \mathcal{P}(\mathbb{R}^n) \rightarrow \mathbb{I}^n$ such that $\forall \mathbf{B} \in \mathbb{I}^n$, $\rho \in \mathcal{P}(\mathbb{R}^n)$ these properties hold:* [3]

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

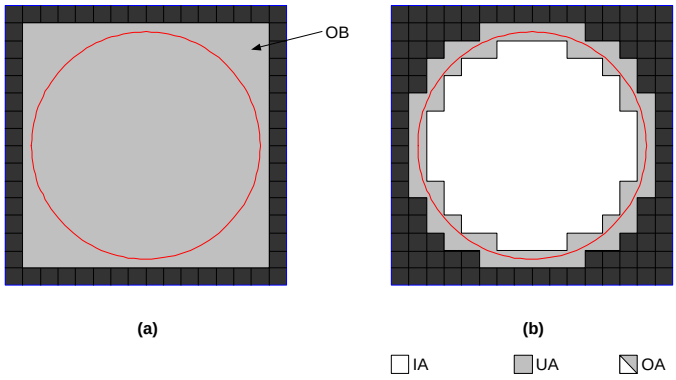

Fig. 2. The Best Approximations: (a) $\mathbf{OB}$; (b) $\mathbf{IA}$ (white), $\mathbf{UA}$ (gray) and $\mathbf{OA}$ (white & gray). The relation to approximate contains all the points lying inside the circle and on its boundary

$$\textit{(1)}\ \mathbf{OC}(\mathbf{B},\rho) \subseteq \mathbf{B} \quad \textit{(Contractiveness)}$$
$$\textit{(2)}\ \mathbf{OC}(\mathbf{B},\rho) \supseteq \mathbf{B} \cap \rho \ \textit{(Completeness)}$$

Often, a monotonicity condition is also required to guarantee confluence. We do not consider this restriction for the moment. In numerical domains, the outer-bound contracting operators usually enforce either *Box*, *Hull*, *kB* or *Bound* consistency [11, 5], generally referred to as bound-consistency in the rest of the paper.

3.3 Union Approximations

In this paper we consider the problem of computing $\mathbf{IA}(\rho)$ and $\mathbf{OA}(\rho)$ approximations of a relation $\rho \subseteq \mathbb{R}^n$ under the form of *unions of disjoint boxes*[4].

Definition 6 (Outer Union Approximation, $\mathbf{Union}^{\mathcal{O}}(\rho)$). $\mathbf{Union}^{\mathcal{O}}(\rho)$ *is a set of disjoint boxes* $\mathcal{U} \in \mathcal{P}(\mathbb{I}^n)$ *such that:*

$$\bigcup_{\mathbf{B}\in\mathcal{U}} \mathbf{B} \supseteq \mathbf{OA}(\rho) \tag{5}$$

Definition 7 (Inner Union Approximation, $\mathbf{Union}^{\mathcal{I}}(\rho)$). $\mathbf{Union}^{\mathcal{I}}(\rho)$ *is a set of disjoint boxes* $\mathcal{U} \in \mathcal{P}(\mathbb{I}^n)$ *such that:*

$$\bigcup_{\mathbf{B}\in\mathcal{U}} \mathbf{B} \subseteq \mathbf{IA}(\rho) \tag{6}$$

Definition 8 (Undiscernible Union Approximation, $\mathbf{Union}^{\mathcal{U}}(\rho)$). $\mathbf{Union}^{\mathcal{U}}(\rho)$ *is a set of disjoint boxes* $\mathcal{U} \in \mathcal{P}(\mathbb{I}^n)$ *such that:*[5]

$$\bigcup_{\mathbf{B}\in\mathcal{U}} \mathbf{B} = cl\Big(\bigcup_{\mathbf{B}\in\mathbf{Union}^{\mathcal{O}}(\rho)} \mathbf{B} \setminus \bigcup_{\mathbf{B}\in\mathbf{Union}^{\mathcal{I}}(\rho)} \mathbf{B}\Big) \tag{7}$$

[4] Two boxes, $\mathbf{B_1}$ and $\mathbf{B_2}$, are said disjoint if $\mathbf{B_1} \neq \mathbf{B_2} \Rightarrow \mathbf{B_1} \sqcap \mathbf{B_2} = \emptyset$.

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

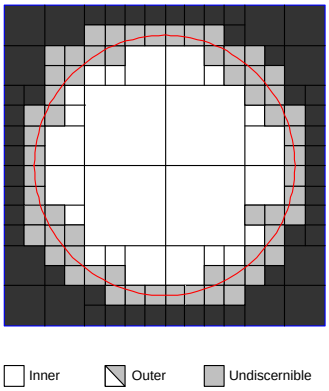

Fig. 3. The Union Approximations: $\mathbf{Union}^{\mathcal{O}}(\rho)$ (white & gray), $\mathbf{Union}^{\mathcal{I}}(\rho)$ (white) and $\mathbf{Union}^{\mathcal{U}}(\rho)$ (gray)

Figure 3 illustrates these definitions on the example of Figure 2.

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, etc. 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] constructs the union algebraically using Bernstein polynomials, which makes it possible to use guaranteed inclusion tests for boxes. The approach is restricted to polynomial constraints. A technique to extend consistent domains of particular class of constraints has also been proposed in [12]. Finally, [10] has addressed the issue of computing $\mathbf{Union}^{\mathcal{I}}$ approximations for universally quantified constraints.

4 Back-Boxing and EVR

Interval-based search techniques for NCSPs are essentially dichotomous. 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 outer-bound contracting operator to the overall constraint system, after each split. In all the known algorithms, the general policy is to perform splitting until canonical intervals are reached and as long as the error inherent to the outer-bound contracting operator is smaller than the interval to split. This policy, referred to as `DMBC` (dichotomous maintaining bound-consistency) in the rest of the paper, works generally well for systems with isolated solutions but leaves room for improvement when there is a con-

tinuum of feasible points. The improvements we propose are presented in the two next subsections.

4.1 Better Splitting Decisions Using Back-Boxing

In order to reduce as much as possible the number of disjoint boxes required for covering the solution space, we first try to avoid the split of completely feasible boxes. To achieve this goal, we use a feasibility (soundness) test for boxes [10],[6] which allows better splitting decisions. Given a relation, ρ, and a box, $\mathbf{B}$, the feasibility test checks whether the whole box is contained in ρ or not. It is based on the following obvious property: $\mathbf{B} \cap \neg\rho = \emptyset \Leftrightarrow \mathbf{B} \subseteq \rho$. We use this property to implement a contracting operator, called *back-boxing contracting* operator and a splitting operator, called *back-box splitting* operator.

Definition 9 (Back-Boxing Contracting Operator, $\mathbf{BBC}$). *A back-boxing contracting operator w.r.t. an* $\mathbf{OC}$ *operator is a function* $\mathbf{BBC} : \mathbb{I}^n \times \mathcal{P}(\mathbb{R}^n) \rightarrow \mathbb{I}^n$ *such that*

$$\forall \mathbf{B} \in \mathbb{I}^n, \rho \in \mathcal{P}(\mathbb{R}^n) : \mathbf{BBC}(\mathbf{B}, \rho) = \mathbf{OC}(\mathbf{B}, \neg\rho)$$

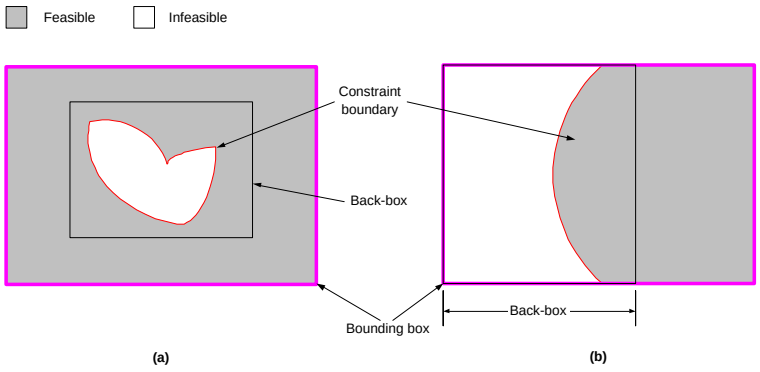

Fig. 4. Two examples of back-boxing contractions

For simplicity, given a finite set of constraints $\mathcal{C} = \{c_1, ..., c_n\}$, we denote $\mathbf{BBC}(\mathbf{B}, \rho_C)$ by $\mathbf{BBC}(\mathbf{B}, c_1, ..., c_n)$ or $\mathbf{BBC}(\mathbf{B}, \mathcal{C})$.

The following properties characterize back-box contracting operators.

Proposition 2. *Given a relation,* ρ*, and a box,* $\mathbf{B}$*, if there is some* $\mathbf{BBC}$ *operator that contracts* $(\mathbf{B}, \rho)$ *to an empty set, then* $\mathbf{B}$ *is completely contained in* ρ*, i.e.*

$$\exists \mathbf{BBC} : \mathbf{BBC}(\mathbf{B}, \rho) = \emptyset \Rightarrow \mathbf{B} \subseteq \rho$$

[6] In [10] such a feasibility test is used for individual constraints. We use it for conjunction of constraints.

Proof. We have $\mathbf{BBC}(\mathbf{B}, \rho) = \emptyset$, $\mathbf{BBC}(\mathbf{B}, \rho) = \mathbf{OC}(\mathbf{B}, \neg\rho)$, and $\mathbf{OC}(\mathbf{B}, \neg\rho) \supseteq \mathbf{B} \cap \neg\rho$, then $\mathbf{B} \cap \neg\rho = \emptyset$, this implies that $\mathbf{B} \subseteq \rho$.

Corollary 1. *Given a finite set of constraints, $\mathcal{C} = \{c_1, ..., c_n\}$, and a bounding box, $\mathbf{B}$. The box $\mathbf{B}$ is completely feasible (w.r.t $\mathcal{C}$) if there is some $\mathbf{BBC}$ operator that contracts $(\mathbf{B}, \mathcal{C})$ to an empty set, i.e.*

$$\exists \mathbf{BBC} : \mathbf{BBC}(\mathbf{B}, \mathcal{C}) = \emptyset \Rightarrow \mathbf{B} \textit{ is feasible (w.r.t } \mathcal{C})$$

This corollary implies that back-boxing makes it possible to isolate completely feasible boxes with respect to some constraints. Figure 4 illustrates the behavior of a back-boxing contracting operator. A back-box results from the application of a $\mathbf{BBC}$ operator to a box $\mathbf{B}$ and a relation ρ_c identified by a constraint c. When applying a back-boxing contracting operator to a box with respect to a constraint results in a empty set, it can be deduced that the box completely satisfies that constraint. Similarly, when applying a back-boxing contracting operator to a box with respect to the whole constraint set results in a empty set, the box can be stated as completely feasible. We then define a splitting operator based on back-boxing, which consists of splitting around back-boxes:

Definition 10 (Back-Box Splitting Operator: $\mathbf{BBS}$). *A back-box splitting operator is a function $\mathbf{BBS} : \mathbb{I}^n \times \mathbb{I}^n \rightarrow \mathcal{P}(\mathbb{I}^n)$ splitting a bounding box, $\mathbf{B}$, along the faces of a back-box, $\mathbf{BB}$.*

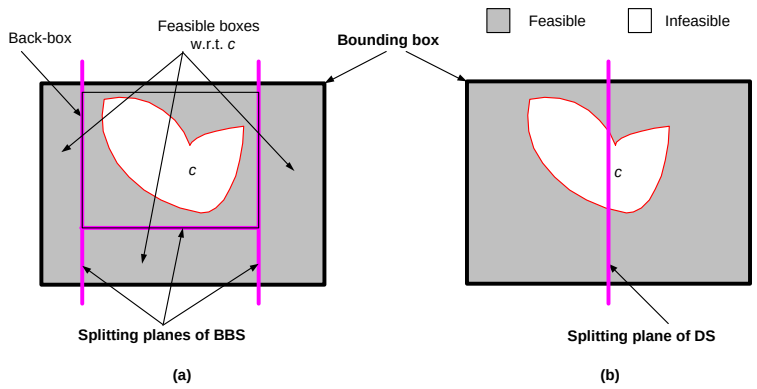

Fig. 5. (a) Back-Box Splitting: splitting around back-boxes; (b) Dichotomous Splitting: splitting the original domain of a variable into two halves

In the algorithms we propose, back-box splitting is applied in combination with dichotomous splitting. The latter is used either when back-boxing produces no reduction or when back-box splitting results in too small boxes (i.e. it is performed close to the boundaries). Figure 5 illustrates the notion of back-box splitting.

4.2 Concise Representations of The Boundaries Using EVR

Back-boxing is mainly intended to reduce the number of boxes of *inner* union approximations. We now address the issue of producing more compact *undiscernible* union approximations. Box-covering, as usually implemented, often produces a significant number of nearly aligned boxes along the boundary of the constraints. These boxes introduce artificial convexity deficiencies which are only due to the orthogonal splitting policy.

Better Alignment of Boxes. We observe that a better alignment can be obtained by closely controlling the application of the contracting operators during search. More precisely, whenever some dimension, i, of a box, $\mathbf{B}$, reaches precision ε_i, one can prevent the contracting operator to contract $\mathbf{B}$ over this dimension in order to obtain better alignments and performances.

Definition 11 (Active/Inactive Dimension). *Given a box, $\mathbf{B}$, a set of constraints, $\mathcal{C}$, and a precision vector, ε. A dimension, i, of $\mathbf{B}$ is called* active dimension *if the size of $\mathbf{B}$ in dimension i exceeds ε_i and if the corresponding variable, v_i, occurs in some constraint of $\mathcal{C}$. Otherwise, it is said to be an* inactive dimension.

A contracting operator working on the active dimensions of a box only will be called a *restricted-dimensional contracting operator*. Hereafter, we denote as $\mathbf{OC_{rd}}$ (respectively, $\mathbf{BBC_{rd}}$) the restricted-dimensional contracting operators corresponding to $\mathbf{OC}$ (respectively, $\mathbf{BBC}$) operators. There are several ways of implementing $\mathbf{OC_{rd}}$ and $\mathbf{BBC_{rd}}$ depending on which $\mathbf{OC}$ operator is used. The first way, consists of using some classical $\mathbf{OC}$ operators working in full-dimension. After a box $\mathbf{B}$ has been contracted over all dimensions, the inactive dimensions are simply restored to their original sizes. The gain is then only a better alignment of boxes. The second way consists of using an $\mathbf{OC}$ operators which directly allows a restricted-dimensional contracting.[7] A box $\mathbf{B}$ will then only be contracted over the active dimensions. Such an $\mathbf{OC}$ operator does not require the returned box to be bound-consistent in the inactive dimensions. The returned box will therefore be usually larger than the one returned by an $\mathbf{OC}$ operator working in full dimension. However, not only better alignments can be obtained but also better performances. The third possible way consists of applying the $\mathbf{OC}$ operator to the *projection* of the relation (or constraints) over the active dimensions only. The result is composed back to the full-dimensional box by adding the inactive dimensions with the data of the original box. This alternative can only be used for the constraints that can be easily projected symbolically.

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 of boxes. We propose to use the Extreme Vertex Representation of orthogonal polyhedra for that purpose. The basic idea is that the finite unions of boxes delivered by box-covering solver define *orthogonal polyhedra* for which improved representations can be used. Informally, *orthogonal polyhedra* are the ones whose facets are axis-parallel hyper-rectangles. They

[7] ILOG Solver 5.1 provides such $\mathbf{OC_{rd}}$ operators. We use them in our implementation.

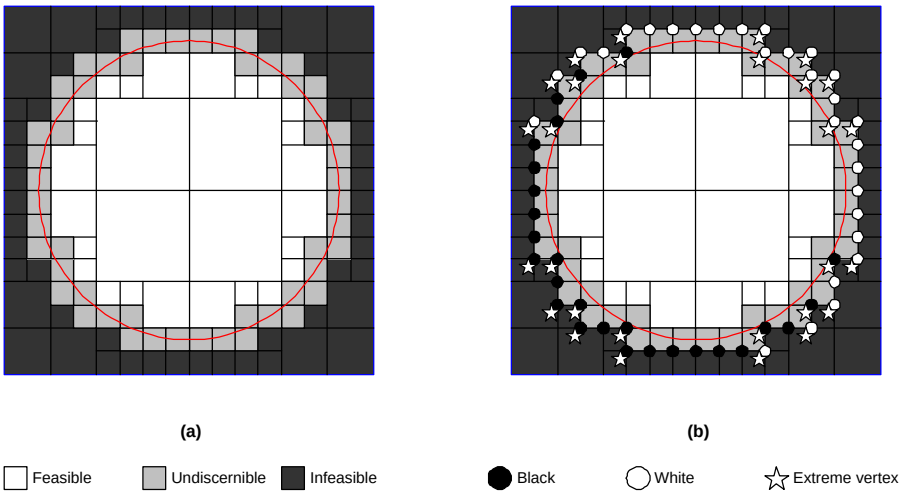

Fig. 6. (a) DBR of union approximations; (b) EVR of $\mathbf{Union}^{\mathcal{O}}$

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 *Extreme Vertex Representation* (EVR) is a way of compacting a DBR representation. It was first proposed in [1] for 3-dimensional orthogonal polyhedra and generalized to n-dimensional orthogonal polyhedra in [2, 3]. We now recall some basic concepts related to EVR. We refer the reader to [2, 3] for further details. The concepts we present relate to a particular type of orthogonal polyhedra, called *griddy polyhedra*. Informally, 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 appropriate stretching and translation, the results on EVR are not affected by this simplification. In fact they even do not depend on an orthogonal basis. For simplicity, we assume that the polyhedra 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 integer 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 12 (Color Function). *Let P be a griddy polyhedron. The color function c: $\mathbf{X} \rightarrow \{0, 1\}$ is defined as follows: if $\mathbf{x}$ is a grid point then $c(\mathbf{x}) = 1$ iff $\mathbf{B}(\mathbf{x}) \subseteq P$; 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 $\Psi : \mathcal{E} \rightarrow 2^{\mathcal{B}}$.

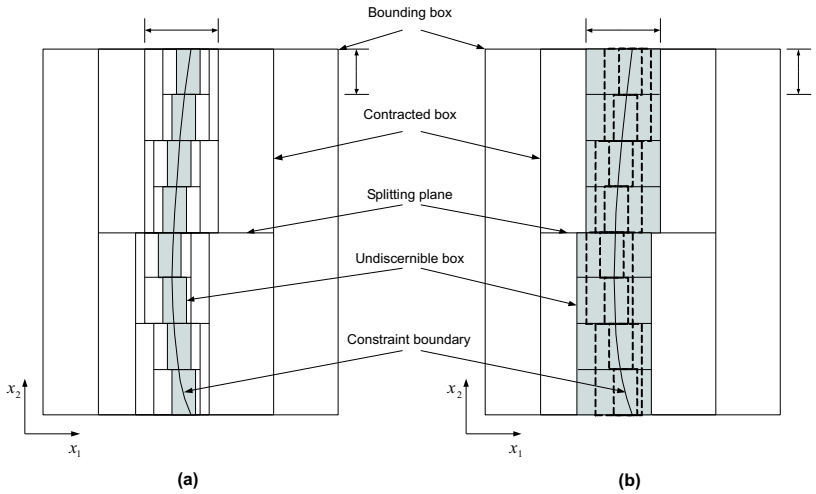

Fig. 7. Constraint boundary in a bounding box: (a) unaligned boxes produced by standard covering; (b) enlarged and aligned boxes using EVR

Definition 13 (Extreme Vertex). *A grid point* **x** *is said to be extreme if* τ*(***x***) = 1, where* τ*(***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* **x***).*

Figure 6 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 [3]. It also enables the design of efficient algorithms for fundamental operations on orthogonal polyhedra (e.g. membership, set-theoretic operations). In particular, effective transformation between DBR and EVR can be proposed for low dimension and/or small size (i.e. m is small) polyhedron [1]. For example, in three dimensions, 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, for a 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 aligned contiguous boxes using the following procedure:

1. Produce a better alignment of the boxes along the boundary of the constraints. This is done by preventing the unnecessary application of contracting operators over the inactive dimensions. Figure 7 shows the better alignment produced for a set of nearly aligned boxes of an undiscernible approximation. The original set of 8 small boxes (Figure 7(a)) reduces to two groups of 4 aligned boxes (Figure 7(b)) without altering the predefined 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, this procedure guarantees that $\mathcal{S}_2$ has a more concise size than $\mathcal{S}_1$. Figure 7(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 dimensions. Due to the efficiency of EVR in low dimensions, we however restrict its application to low dimensional or small size sub-regions of the search space in our implementation (see Section 5.2).

5 Algorithms

We now present two algorithms that compute outer and inner union approximations for non-linear NCSPs, These algorithms, called `UCA6` and `UCA6-Plus` are referred to as `UCA6*` when they have the same properties/operations. A preliminary version of `UCA6` was presented in [13] we will therefore mainly focus on the `UCA6-Plus` algorithm. Given a NCSP $P = (\mathcal{V}, \mathcal{C}, \mathcal{D})$, $\mathbf{B}$ will denote the bounding box of the relation defined by P. Originally, this bounding box is set to $\mathcal{D}$. For convenience, we denote $\mathbf{Union}^{\mathcal{X}}(\mathbf{B} \cap \rho_{\mathcal{C}})$ as $\mathbf{Union}^{\mathcal{X}}(\mathbf{B}, \mathcal{C})$, where $\mathcal{X} \in \{\mathcal{O}, \mathcal{I}, \mathcal{U}\}$. `UCA6*` 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*` proceeds by repeating three main operations: *(i)* using outer-bound contracting operators to contract the current bounding box to a tighter bounding box;[8] *(ii)* using back-boxing contracting operators to get a list of back-boxes w.r.t. each active constraint and w.r.t the contracted box in *(i)*, the constraints that makes the corresponding back-box empty are removed; finally, *(iii)* combining dichotomous splitting with back-box splitting. When the chosen strategy, at a given moment, is back-box splitting, the $\mathbf{BBS}$ operator is used to split around the best back-box (details are given later). The constraint corresponding to the chosen back-box is then removed from all the surrounding boxes resulting from the $\mathbf{BBS}$.[9] In Figure 8 and 9, $\mathcal{S}_{inn}$ and $\mathcal{S}_{und}$, which are global variables, denote the sets of boxes 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 sub-problems waiting to be processed. `WList` can be handled as a queue or a stack. This allows for breadth-first search in the former case and to depth-first search in the latter. $chooseTheBest(\mathbf{B}, \{\mathbf{BB}_c | c \in \mathcal{C}\})$ is a function choosing the best back-box and the respective constraint based on some criteria to maximize the space surrounding the back-box. $enlarge(\mathbf{B}, \mathbf{BB}_c, ZeroPlus)$ is a function extending $\mathbf{BB}_c$ to $\mathbf{BB}$ by $ZeroPlus$ (considered as a sufficiently small positive number) such that the result is still in $\mathbf{B}$. This will guarantee that no point satisfying $\neg c$ is on the boundary of $\mathbf{BB}$ except the points on the boundary of $\mathbf{B}$. Figures 8 and 9 give the algorithm `UCA6-plus`.

5.1 Splitting Strategies

$getSplit()$ is a function which returns the splitting mode to be used for splitting the current box. `UCA6*` uses a combination of the $\mathbf{BBS}$ and $\mathbf{DS}$ operators. The current

[8] Standard operator for `UCA6` and restricted-dimensional one for `UCA6-Plus`.

[9] These boxes are known to be feasible due to the properties of back-boxing.

```
function UCA6Plus( B_0, C_0, ε, OC_rd, BBC_rd, OC, BBC, D_stop)
    S_inn := ∅; S_und := ∅; WList := ∅;
    if solveQuickly(B_0, C_0, ε, OC_rd, BBC_rd, OC, BBC, WList, D_stop) then return;
    while WList ≠ ∅ do
        < B, C > := get(WList);
        foreach c ∈ C do
            BB_c := BBC_rd(B, c);
            if BB_c = ∅ then
                C := C \ {c};
            end
        end
        if C = ∅ then
            store(S_inn, B);
            continue; /* while loop */
        end
        Split = getSplit();
        if Split = 'BBS' then
            BB_c := chooseTheBest(B, {BB_c | c ∈ C});
            BB := enlarge(B, BB_c, ZeroPlus);
            < B_1,...,B_k > := BBS(B, BB);
        else
            < B_1,...,B_k > := DS(B);
        end
        for i = 1 to k do
            if Split = 'BBS' and B_i ∩ BB_c = ∅ then
                if C = {c} then
                    store(S_inn, B_i);
                    continue; /* for loop */
                else
                    C := C \ {c};
                end
            end
            solveQuickly(B_i, C, ε, OC_rd, BBC_rd, OC, BBC, WList, D_stop);
        end
    end
end /* UCA6Plus */
```

Fig. 8. The `UCA6-plus` algorithm

splitting mode returned by $getSplit()$ is inferred from information on the history of the current box. The simplest implementation uses the information concerning the splitting mode of the parent box, for example whether or not the current box is a back-box obtained from splitting the parent box.

In contrast to `DMBC`, the $\mathbf{DS}$ operator used for `UCA6*` only tries to dichotomize over the active dimensions. This avoids splitting boxes into a huge number of tiny boxes. Moreover, in `UCA6*` constraints are removed gradually whenever an empty back-box is computed w.r.t. those constraints. The dimension with the greatest size is preferred

for **DS** split. For the pruning to be efficient, **BBS** splits along some face of a back-box only if the splitting plane produces sufficiently large boxes, the back-box itself excepted. This estimation is done using a pre-determined *fragmentation ratio*.

5.2 Applying EVR

function $solveQuickly$(**B**, $\mathcal{C}$, ε, $\mathbf{OC_{rd}}$, $\mathbf{BBC_{rd}}$, **OC**, **BBC**, WList, D_{stop})
  if B *has no active dimension* **then**
    $\mathbf{B}'$:= **OC**(**B**, $\mathcal{C}$);
    if $\mathbf{B}' = \emptyset$ **then return** True;
    if BBC(**B**, $\mathcal{C}$) = $\emptyset$ **then**
      $store(\mathcal{S}_{inn}, \mathbf{B})$;
      return True;
    end
    $store(\mathcal{S}_{und}, \mathbf{B})$;
    return True;
  end
  $\mathbf{B}'$:= $\mathbf{OC_{rd}}$(**B**, $\mathcal{C}$);
  if $\mathbf{B}' = \emptyset$ **then return** True;
  if $\mathbf{B}'$ *has no active dimension* **then**
    if BBC($\mathbf{B}'$, $\mathcal{C}$) = $\emptyset$ **then**
      $store(\mathcal{S}_{inn}, \mathbf{B}')$;
      return True;
    end
    $store(\mathcal{S}_{und}, \mathbf{B}')$;
    return True;
  end
  if $\mathbf{B}'$ *has at most* D_{stop} *active dimensions* **then**
    $< \mathcal{S}'_{inn}, \mathcal{S}'_{und} >$:= $DimStopSolver$($\mathbf{B}'$, $\mathcal{C}$, ε, $\mathbf{OC_{rd}}$, $\mathbf{BBC_{rd}}$, **OC**, **BBC**);
    $store(\mathcal{S}_{inn}, combine(\mathcal{S}'_{inn}))$;
    $store(\mathcal{S}_{und}, combine(\mathcal{S}'_{und}))$;
    return True;
  end
  put(WList,$< \mathbf{B}', \mathcal{C} >$);
  return False;
end /* solveQuickly */

Fig. 9. The function $SolveQuickly$

The function $solveQuickly$ (Figure 9) is of the main novelties in `UCA6-Plus` w.r.t. `UCA6`. This function constructs $\mathbf{Union}^{\mathcal{I}}$ and $\mathbf{Union}^{\mathcal{U}}$ approximations for low-dimensional sub-problems with at most D_{stop} active dimensions. The output is compacted using EVR. The second novelty lies in the fact that `UCA6-Plus` uses $\mathbf{OC_{rd}}$ and $\mathbf{BBC_{rd}}$ instead of **OC** and **BBC** for the purpose of narrowing.[10] By doing so

[10] **OC** and **BBC** are still used for checking the feasibility of ε-bounded boxes.

UCA6-Plus gains in performance and produces a better alignment of boxes along the boundaries. This allows for using EVR to combine the aligned contiguous boxes. $solveQuickly$ (Figure 9) proceeds by using the following tests: *(i)* check if the bounding box is infeasible; *(ii)* check if the contracted box has no active dimension, then check if it is feasible or undiscernible w.r.t. $\mathcal{C}$; *(iii)* check if the contracted box has at most D_{stop} active dimensions, if so, use an appropriate technique to solve the sub-problem in that box using restricted-dimensional contracting operators, *(iv)* Otherwise, the contracted box is put into the waiting list to be further processed. For efficiency purposes, $solveQuickly$ allows resorting to a secondary search technique, $DimStopSolver$, to solve the low-dimensional sub-problems 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[11]. Variants of DMBC or UCA6 using the restricted-dimensional contracting operators can alternatively be used. For a given sub-problem, $DimStopSolver$ constructs the sets $\mathcal{S}'_{inn}$ and $\mathcal{S}'_{und}$ which are the $\mathbf{Union}^{\mathcal{I}}$ and $\mathbf{Union}^{\mathcal{U}}$ of the sub-problem, respectively. These two sets are represented in DBR. They are converted to EVR and then back to DBR to combine each group of aligned contiguous boxes into a bigger equivalent box. This operation is represented by the function $combine$ in Figure 9. The results after combination are stored in the set of boxes of $\mathbf{Union}^{\mathcal{I}}(\mathbf{B}_0, \mathcal{C}_0)$ and $\mathbf{Union}^{\mathcal{U}}(\mathbf{B}_0, \mathcal{C}_0)$.

Proposition 3. *Let $P = (\mathcal{V}, \mathcal{C}, \mathcal{D})$ be a numeric CSP, the* UCA6 *and* UCA6-Plus *algorithms compute an inner union approximation ($\mathbf{Union}^{\mathcal{I}}$) and outer union approximation ($\mathbf{Union}^{\mathcal{O}}$) for $\rho_{\mathcal{C}}$ w.r.t. the predefined precision.*

Sketch of proof: The rigorous proof of this proposition is out of scope of the paper. However, we can see informally that given the properties of the contracting operators used for these algorithm: *(i)* the outer-bound contracting operator always produces a box which contains the active relation, and *(ii)* the back-boxing contracting operator always produces a back-box which contains the negation of the active relation, then it is safe to remove the active relation (defined by one or many constraints) from all the regions surrounding the back-box and contained in the bounding box.

6 Preliminary Experiments

Only a small amount of work exists on computing inner and outer union approximations for numerical CSPs with non-linear constraints. Often these problems are recast as optimization problems, with artificial optimization criteria, to fit the solvers. Hence, no significant set of benchmarks is presently available in this area. In this section we present a preliminary evaluation on the following small set of typical problems (with different types of solution space).

CD (column design) and FD (fatigue design) are two engineering design examples. Their complete descriptions are available at *http://imacsg4.epfl.ch:8080/PGSL/*. In Table 1, the considered instance of *CD* is the one that finds $(a, b, e) \in [0.01, 2] \times [0.01, 1] \times$

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

$[0.05, 0.1]$ given the eccentric load $P = 400kN$, the height of column $H = 6m$ and the eccentricity load $L = 1m$, where a and b denote the width and depth of the column cross section (in meter) respectively, e the thickness (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 qf is the permissible load and Z is the section modulus (scaled up 100 times in unit).

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.
$WP = \{20 < \sqrt{x^2 + y^2} < 50,\ 12y/\sqrt{(x-12)^2 + y^2} < 10,\ x \in [-50,\ 50],\ y \in [0,\ 50]\}$.

SB describes structural and safety restrictions for the components of a floor consisting of a concrete slab on steel beams.
$SB = \{u + c_1 w^{1.5161} - p = 0,\ u - (c_6 h_s + c_7)s \leq 0,\ c_2 - c_3 s + c_4 s^2 - c_5 s^3 - h_s \leq 0,\ c_8(pw^2)^{0.3976} - h_b \leq 0,\ c_9(pw^3)^{0.2839} - h_b,\ w \in [5800,\ 13500],\ p \in [15,\ 50],\ h_b \in [150,\ 223],\ s \in [1900,\ 2200],\ u \in [10,\ 60],\ h_s \in [75,\ 200]\}$.

$P1 = \{x_0 = x_1 + 1,\ x_2 + 1 = x_0 + x_1,\ x_2 \geq x_0 + 2,\ x_1 + 2x_3 \geq x_4,\ x_2 - x_3 \leq 3,\ x_i \in [-10,\ 10]\ \}$.

$P2 = \{x^2 \leq y,\ \ln y + 1 \geq z,\ xz \leq 1,\ x \in [-15,\ 15],\ y \in [1,\ 200],\ z \in [-10,\ 10]\}$.

$P3 = \{x^2 \leq y,\ \ln y + 1 \geq z,\ xz \leq 1,\ x^{3/2} + \ln(1.5z + 1) \leq y + 1,\ x \in [-15,\ 15],\ y \in [1,\ 200],\ z \in [-10,\ 10]\}$.

For comparison and 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 version of `DMBC` including the feasibility test. This enhanced version, called `DMBC+`, can therefore check whether a box is completely feasible or not. Our experiments discard `DMBC` as a reasonable candidate for this kind of problems. It always produces a huge number of boxes, each of which is ε-bounded. The tests shown in Table 1 were run with a fragmentation ratio of 0.25 and D_{stop} = 1. The standard $\mathbf{OC}$ operator was implemented with ILOG Solver 5.1 [6]. Each cell in the table has two rows. The first shows time and the second the number of boxes in the $\mathbf{Union}^{\mathcal{I}}$ and $\mathbf{Union}^{\mathcal{U}}$ respectively. $Bprec$ is the precision for the contracting operators, $prec$ is the precision of the algorithms. The secondary search technique used for `UCA6-Plus` in Table 1 is the simple grid-based one.

Other tests were carried out on tens of similar problems. They showed that the best gains, in running time and number of boxes, of `UCA6*` over `DMBC+` are obtained for problems with low-arity constraints. `UCA6*` remains better than `DMBC+` in running time in the other cases. The experiments also showed that `UCA6-Plus` is always better than `UCA6` in running time and number of boxes. The best gains are obtained when the non-linear constraints contain some nearly axis-parallel sub-regions.

The preliminary experiments are therefore encouraging enough to warrant further investigations and in-depth evaluations.

Table 1. Test results

Problem	$prec$	$Bprec$	DMBC+	UCA6	UCA6-Plus
$P1$	0.01	0.1	$>$ 1h $>$ 0/90000	94.89s 0/11465	66.17s 0/6415
$P2$	0.1	0.2	$>$ 1h $>$ 30000/80000	267.38s 18721/55062	21.49s 1813/3224
$P3$	0.1	1	$>$ 1h $>$ 20000/70000	142.14s 12113/38808	5.63s 403/970
WP	0.1	1	33.48s 1683/3692	12.27s 1521/2859	9.95s 949/1699
SB	0.01	1	$>$ 1h $>$ 0/100000	15.35s 0/4932	11.93s 0/2743
CD	0.01	1	2497.32s 2871/28665	959.93s 9301/26568	705.30s 1086/13414
FD	0.1	1	2638.82s 16437/92681	440.21s 26330/70218	273.77s 10324/35134

7 Conclusion

Interval-constraint based solvers are usually designed to deliver point-wise solutions. Their techniques are complete and work efficiently for problems with isolated solutions, but might alter both efficiency and compactness of the output representation for many problems with continuum of feasible points. In this paper, we propose a technique for computing inner and outer approximations of numerical CSPs that remedies this state of affairs. The approach works for general non-linear CSPs with equality and inequality constraints. It combines an enhanced splitting policy with the extreme vertex representation of orthogonal polyhedra, as defined in computational geometry. This allows for gains in performance and space requirements. The quality of the output is also enhanced since the inner approximations provide *guaranteed feasible boxes*. In practice, NCSPs with continuum of solutions often occur as sub-problems of higher dimensional NCSPs. In future work, we therefore plan to investigate collaboration strategies between our approximation 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 require intermediate conversion steps to the disjoint box representation).

8 Acknowledgments

Support for this research was 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), Switzerland (2000)
4. Nocedal, J., Wright, S.: Numerical Optimization - Series in Operations Research. Springer (2000)
5. Van Hentenryck, P.: A gentle introduction to Numerica. (1998)
6. ILOG: ILOG Solver. Reference Manual. (2001)
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 6^{th} International Conference on Principles and Practice of Constraint Programming (CP'2000). (2000) 67–82
11. Lhomme, O.: Consistency techniques for numeric CSPs. In: Proceedings of IJCAI'93. (1993)
12. Collavizza, H., Delobel, F., Rueher, M.: Extending consistent domains of numeric CSP. In: Proceedings IJCAI'99. (1999)
13. 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)