Search techniques for non-linear constraint satisfaction problems with inequalities

Marius-Cătălin Silaghi, Djamila Sam-Haroud, and Boi Faltings

Artificial Intelligence Laboratory
Swiss Federal Institute of Technology 1015 Lausanne, Switzerland
{silaghi,haroud,faltings}@lia.di.epfl.ch

Abstract. In recent years, interval constraint-based solvers have shown their ability to efficiently solve challenging non-linear real constraint problems. However, most of the working systems limit themselves to delivering point-wise solutions with an arbitrary accuracy. This works well for equalities, or for inequalities stated for specifying tolerances, but less well when the inequalities express a set of equally relevant choices, as for example the possible moving areas for a mobile robot. In that case it is desirable to cover the large number of point-wise alternatives expressed by the constraints using a reduced number of sets, as interval boxes. Several authors [2, 1, 7] have proposed set covering algorithms specific to inequality systems. In this paper we propose a lookahead backtracking algorithm for inequality and *mixed equality/inequality* constraints. The proposed technique combines a set covering strategy for inequalities with classical interval search techniques for equalities. This allows for a more compact representation of the solution set and improves efficiency.

1 Introduction

A wide range of industrial problems require solving constraint satisfaction problems (CSPs) with numerical constraints. A numerical CSP (NCSP), (V, C, D) is stated as a set of variables V taking their values in domains D over the reals and subject to constraints C. In practice, 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. Interval constraint-based solvers (e.g. Numerica [8], Solver [4]) 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). While they have proven particularly efficient in solving challenging instances of numerical CSPs with non-linear constraints, they are commonly designed to deliver *punctual* solutions. This fits well the needs inherent to equality systems but is less adequate for several problems with inequalities. Inequalities can be used to state tolerances, like for example *beam-dimension* $= k \pm \varepsilon$, and it is then admissible to solve them using punctual solvers. However, in their most general form, they rather express spectra of equally relevant alternatives which need to be identified as precisely and exhaustively as possible. Such inequalities will, for example, define 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. In all these cases, it is not acceptable to arbitrarily focus on a speci£c solution, especially when this choice is forced by idiosyncrasies of the used solver.

A natural alternative to the punctual approach is to try to cover the spectrum of solutions for inequalities using a reduced number of subsets from $\mathbb{IR}^n$. Usually, these subsets are chosen with known and simple properties (interval boxes, polytopes, ellipsoid,..) [5]. In recent years, several authors have proposed set covering algorithms with intervals boxes [5, 2, 7, 1]. These algorithms, except for [5], are designed for inequality systems[1], are based on domain splitting and have one of the two following limitations. Either the constraint system is handled as an indivisible whole[2], or the splits are performed statically which means that their results are not, or only partially, further propagated to the related variables. In the £rst case the tractability limits are rapidly reached while in the second, the information resulting from a split is sub-optimally used. This paper proposes an algorithm for *dynamically* constructing an interval-box covering, for a set of equality/inequality constraints, according to a "maintaining local consistency" search schema. In numerical domains, local consistency usually takes the form of either Box, Hull, kB or Bound consistency [8, 6], generally referred to as bound consistency in the rest of the paper. Maintaining bound consistency (MBC) is a powerful lookahead search technique for numerical CSPs which allows the splits performed on a given variable domain to be propagated on domains of other variables, thus reducing the splitting effort. The proposed technique builds on the feasibility test proposed in [1]. This allows for robustly constructing sound boxes and devising ef£cient splitting heuristics for search. The output is a union of boxes which conservatively encloses the solution set. As shown by the preliminary experiments, the new algorithm improves ef£ciency as well as the compactness of the output representation. In order to reduce the space requirements, our algorithm can alternatively be used to compute a new form of consistency called $\varepsilon_1\varepsilon_2$-consistency. $\varepsilon_1\varepsilon_2$-consistency is a weakening of global consistency which only considers certain projections of the solution space. It can be used as a preprocessing technique for speeding up further queries.

2 Background

We start by recalling the necessary background and de£nitions. Parts of the material described in this section are presented in [1].

2.1 Interval arithmetic

Intervals The £nite nature of computers precludes an exact representation of the reals. The set $\mathbb{IR}$, extended with the two in£nity symbols, and then denoted by $\mathbb{IR}^\infty = \mathbb{IR} \bigcup \{-\infty, +\infty\}$, is in practice approximated by a £nite subset $\mathbb{F}^\infty$ containing $-\infty$, $+\infty$ and 0. In interval-based constraint solvers, $\mathbb{F}^\infty$ usually corresponds to the ¤oating point numbers used in the implementation. Let $<$ be the natural extension to $\mathbb{IR}^\infty$ of the

[1] In [7, 1] equalities are approximated by inequalities.

[2] All the variables are split uniformly [5], or the entire set of constraints must be algebraically reformulated [2].

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.

A closed interval $[l, u]$ with $l, u \in \mathbb{F}$ is the set of real numbers $\{r \in \mathbb{R} \mid l \leq r \leq u\}$. Similarly, an open/closed interval $[l, u)$ (respectively $(l, u]$) with $l, u \in \mathbb{F}$ is the set of real numbers $\{r \in \mathbb{R} \mid l \leq r < u\}$ (respectively $\{r \in \mathbb{R} \mid l < r \leq u\}$). The set of intervals, denoted by $\mathbb{I}$ is ordered by set inclusion. In the rest of the paper, intervals are written uppercase, reals or ¤oats are sans-serif lowercase, vectors in boldface and sets in uppercase calligraphic letters. A *box*, $\mathbf{B} = I_1 \times \ldots I_n$ is a Cartesian product of n intervals. 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.

Numerical Constraints Let $\mathcal{V}_\mathbb{R} = \{x_1 \ldots x_n\}$ be a set of variables taking their values over $\mathbb{R}$. Given $\sum_\mathbb{R} = \{\mathbb{R}, \mathcal{F}_\mathbb{R}, \mathcal{R}_\mathbb{R}\}$ a structure where $\mathcal{F}_\mathbb{R}$ denotes a set of operators and $\mathcal{R}_\mathbb{R}$ a set of relations de£ned in $\mathbb{R}$, a *real constraint* is de£ned as a £rst order formula built from $\sum_\mathbb{R}$ and $\mathcal{V}_\mathbb{R}$. Interval arithmetic methods [3] are the basis of interval constraint solving. They approximate real numbers by intervals and compute conservative enclosures of the solution space of real constraint systems.

Relations and Approximations Let $c(x_1, \ldots x_n)$ be a real constraint with arity n. The *relation* de£ned by c, denoted by ρ_c, is the set of tuples satisfying c. The relation de£ned by the negation, $\neg c$, of c is given by $\mathbb{R}^n \setminus \rho_c$ and is denoted by $\rho_{\overline{c}}$. The global *relation* de£ned by the conjunction of all the constraints of an NCSP, $\mathcal{C}$ is denoted $\rho_\mathcal{C}$. It can be approximated by a computer-representable superset or subset. In the £rst 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, $\mathbf{Union}\rho$, or more coarsely by the smallest box $\mathbf{Outer}\rho$, containing it. By using boxes included into ρ, sound (inner) approximations $\mathbf{Inner}\rho$ can also be de£ned. In [1], $\mathbf{Inner}\rho$ is de£ned as the set $\{r \in \mathbb{R}^n \mid \mathbf{Outer}\{r\} \subseteq \rho\}$. Figure 1 illustrates the different forms of approximations.

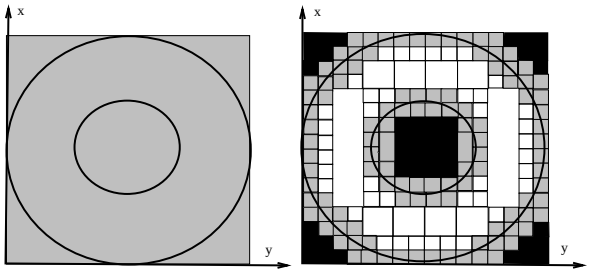

Fig. 1. a) is an $\mathbf{Outer}$ approximation; b) The set of white boxes give an $\mathbf{Inner}$ approximation, together with all the grey boxes they give a $\mathbf{Union}$ approximation.

The computation of these approximations relies on the notion of *contracting operators*. Basically, a contracting operator narrows down the variable domains by discarding

values that are locally inconsistent. This is often done using bound consistency. In this paper we use the notion of outer contracting operator, de£ned as follows:

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

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

Often, a monotonicity condition is also required [3].

2.2 Implementing approximations of type $\mathbf{Union}$

In this paper we consider the problem of computing $\mathbf{Union}$ approximations. Several authors have recently addressed this issue. In [5], 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 [7], a similar algorithm is presented. However, only binary or ternary subsets of variables are considered when performing the splits. This means that for problems of dimension n, only quadtrees or octrees need to be constructed instead of the entire 2^n-tree. The approach is restricted to classes of problems with convexity properties. The technique proposed in [2] constructs the union algebraically, using Bernstein polynomials which give formal guarantees on the result of the inclusion test. The approach is restricted to polynomial constraints. Finally, [1] has addressed the related problem of computing $\mathbf{Inner}$ approximations, which are also unions of boxes but entirely contained in the solution space.

3 Conservative Union Approximation

Interval-based search techniques for CSPs with equalities and inequalities 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 in two halves up to an established resolution. The most successful techniques enhance this process by applying an outer-contracting operator to the overall constraint system, after each split. In all the known algorithms, equalities and inequalities are treated the same way. Splitting is performed until canonical intervals are reached and as long as the error inherent to the outer-contracting operator is smaller than the interval to split. This policy, referred to as DMBC (Dichotomous MBC) in the rest of the paper, works generally well for equality systems but leaves place for improvement when inequalities are involved. Let us consider a small NCSP with the following constraints:

$$P_1 = \{x_0 = x_1 + 1, x_2 + 1 = x_0 + x_1, x_2 \geq x_0 + 2, x_1 + 2x_3 \geq x_4, x_2 - x_3 \leq 3\}$$

where the domains are [-10,10]. For this example, the usual technique, ef£ciently implemented in ILOG Solver, a popular constraint-based solver, generates 8280 small boxes

when all the solutions are explicitly asked for[3]. Using a set covering strategy for inequalities, the technique we propose delivers all the solutions, with the same precision, using only 199 boxes and with a speed up of three times.

The reason behind these results is that in the £rst case, the splitting is done blindly, without taking into account the topology of inequalities. Instead, the technique we propose includes a feasibility (soundness) test for boxes, which allows better splitting decisions. Given a constraint and a box, the feasibility test checks whether all the points in the box satisfy the constraint. Recently, an original idea was proposed in [1] for safely implementing such tests for general constraints. Given a constraint c and a box $\mathbf{B}$, it consists of proving that $\{r \in \mathbf{B} \mid r \in \rho_{\overline{c}}\} = \emptyset$. The proof is done by construction using DMBC on $\neg c$ and is valid due to the completeness of DMBC. We use a related approach for computing an outer approximation of type $\mathbf{Union}$. We de£ne a union conservative contracting operator as follows:

De£nition 2 (Union Conservative Contracting Operator). *Let ρ be an n-ary real relation. A union conservative contracting operator for ρ, $\mathbf{UC}_c : \mathbb{I}^n \rightarrow \mathcal{P}(\mathbb{I})^n$ veri£es:*

$$\forall \mathbf{B} : \mathbf{UC}_\rho(\mathbf{B}) \supseteq \mathbf{Union}(\mathbf{B} \bigcap \rho) \tag{1}$$

In this paper we use an outer contracting operator on inverted inequalities to avoid splitting completely feasible boxes. The goal is to generate a more compact output and to reduce the replication of search effort.

4 Algorithms

We now present an algorithm named UCA6 (Algorithm 1) that computes a $\mathbf{Union}$ approximation for numerical CSPs with equalities and inequalities. We note lists in the Prolog style $[Head|Tail]$. $\mathcal{B}$ denotes the list of children to be checked for a node, and $\mathcal{P}$ denotes the list of all $\mathcal{B}$. The algorithm presented is depth-£rst. Breadth-£rst and other heuristics can be obtained by treating the lists $\mathcal{B}$ and $\mathcal{P}$ as sets, $\mathcal{P}$ becoming respectively the union of all the sets of type $\mathcal{B}$. The algorithm UCA6 iteratively calls the function **getNext** which delivers a new $\mathbf{Outer}$ approximation for a subspace in the solution space. By construction, the new $\mathbf{Outer}$ approximation will not intersect with any previously computed box. The function **getNext** has two main components: a reduction operator, **reduc** (Algorithm 2), and a splitting operator, **split** (Algorithm 3). These operators are interleaved as in a classical maintaining bound consistency algorithm. Practically, it is preferable to stop the dichotomous split when the precision of the numeric search tool (splitting and contracting operators) can lead to unsafe solutions at a given precision ε. An unsafe solution is a box that may contain no real solution. **reduc**, checks this state using a function called *Indiscernible*$(\text{constraint}, \mathbf{Box}, \mathbf{OC}, \varepsilon)$, which is not discussed here in detail[4].

Each search node is characterized by the next structures:

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

[3] Typically, the algorithms are optimized for delivering the £rst solution.

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

* A box $\mathbf{B}$ de£ning the domains of the current NCSP.
* The current NCSP $\mathcal{C}$ containing only the constraints of the initial NCSP that can participate in pruning the search space. The constraints that are indiscernible or entirely feasible in $\mathbf{B}$ are eliminated.
* Each constraint q in a node is associated with a box, $\mathbf{B_q}$, such that all the space in $\mathbf{B} \setminus \mathbf{B_q}$ is feasible.

Each $\mathbf{B_q}$ is initially equal with the projection of the initial search space on the variables in the constraint q, after applying $\mathbf{OC}_{\rho_{\overline{q}}}$. One of the features of **reduc** is that it removes redundant completely feasible or indiscernible constraints. If the recent domain modi£cations of some inequality q have modi£ed $\mathbf{B_q}$, q is checked for feasibility at line 4, and eventually removed from the current CSP (line 6). Equalities are similarly eliminated at line 9 when they become indiscernible.

4.1 Splitting operator

The function **split** (Algorithm 3) allows for using three splitting strategies. The £rst one, **splitFeasible**, extracts sound subspaces for some inequality, as long as these subspaces fragment the search space in a ratio limited by a given *fragmentation threshold*, denoted by *frag* (line 4). The second and the third strategies (**splitIneq**, respectively **splitEq**), consist of choosing for dichotomous split, a variable involved in an inequality (respectively an equality) of the current NCSP $\mathcal{C}$. The heuristics used at lines 5, 6, 7, and 8 in Algorithm 3 can be based on the occurrence of variables in the constraints of $\mathcal{C}$, or according to a round robin technique. The domain of the chosen variable is then split in two halves. Techniques based on the occurrences of variables in constraints can also be used to devise heuristics on ordering the bounds at line 3 in **splitFeasible**. The criteria for choosing a constraint at line 2 can look for maximizing the size of the search space for which a given constraint is eliminated, minimize the number of children nodes, or maximize the number of constraints that can bene£t[5] from the split.

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

* $\mathbf{B}_{\mathbf{f}(\mathbf{x},\mathbf{b})[\mathbf{B_q},\mathbf{B}]}$ is the (feasible) box not containing $\mathbf{B_q}$ obtained from $\mathbf{B}$ by splitting the variable x in b.
* $\mathbf{B}_{\mathbf{u}(\mathbf{x},\mathbf{b})[\mathbf{B_q},\mathbf{B}]}$ is the (indiscernible) box containing $\mathbf{B_q}$ obtained from $\mathbf{B}$ by splitting the variable x in b.
* $\mathbf{B}_{\frac{\mathbf{1}}{\mathbf{2}}\mathbf{r}(\mathbf{x})[\mathbf{B}]}$ is the (indiscernible) box obtained from $\mathbf{B}$ by splitting the variable x in half and retaining its upper half.
* $\mathbf{B}_{\frac{\mathbf{1}}{\mathbf{2}}\mathbf{l}(\mathbf{x})[\mathbf{B}]}$ is the (indiscernible) box obtained from $\mathbf{B}$ by splitting the variable x in half and retaining its lower half.

These concepts are illustrated in the Figure 2.

Proposition 1. *Let $\mathcal{C} = (V, C, D)$ be an NCSP. UCA6 computes a union conservative contractive operator for $\rho_{\mathcal{C}}$.*

[5] The constraints for which the domains are split may propagate more when $\mathbf{OC}$ is applied.

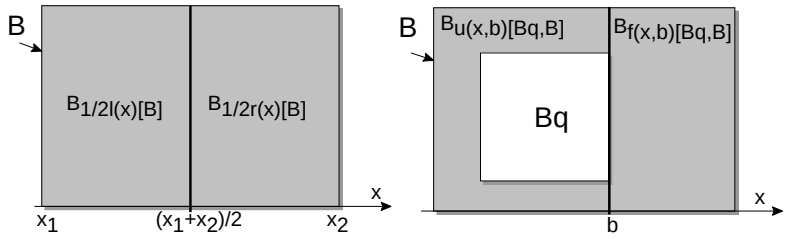

Fig. 2. Splitting operators

Sketch of proof Both the splitting and the contracting operators are complete and conservative. As invariant, the union of $\mathcal{P}$ with the set of already returned solutions corresponds to the output of a union conservative contractive operator. Therefore, when $\mathcal{P}$ is empty, the output solutions satisfy the property.

5 Handling space limitations

When a representation of all the solutions of a NCSP has to be built, or even its projection to a quite limited number of variables, the precision is the most constraining factor. The space required depends exponentially on this precision. The analytic representation itself is very ef£cient in space, but is less easy to visualize and offers less topological information. The amount of aggregation on solutions is a second factor that controls the required space. The improvements that can be achieved depend on the problem at hand. In order to characterize the representations that can be obtained we introduce the notion of $\varepsilon_1\varepsilon_2$-consistency which allows for constructing the representation of the solution space only for a given subset of variables.

De£nition 3 (ε-solution). *An ε-solution of a NCSP $\mathcal{N}$ is a box denoted by $\nu_{\mathcal{N},\varepsilon} = I_1 \times ... \times I_n$ (n is the number of variables in $\mathcal{N}$) such that the search tools with resolution ε (splitting and contracting operators) cannot reduce it or decide the absence of solutions inside it.*

De£nition 4 ($\varepsilon_1\varepsilon_2$-consistency). *A constraint c($x_1, ...x_k$) of a NCSP $\mathcal{N} = (V, C, D)$ is $\varepsilon_1\varepsilon_2$-consistent related to the variables in $X = \{x_1, ..., x_k\}$, $X \subseteq V$, iff:*

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

The procedure **UCA6** can be modi£ed for generating the boxes for representing an $\varepsilon_1\varepsilon_2$-consistent constraint on a set X of variables. This is done by £ltering out of $\mathcal{P}$, the portions of search space, closer to the found solution (line 1) than a distance ε_1. The distance is computed in the space de£ned by the variables in X.

6 Experiments

Only a small amount of work exists on computing union approximations for numerical problems with mixed equality/inequality constraints. Often these problems are recast

Algorithm 1: *Search*

procedure UCA6($\mathcal{C} = (V, C, D)$*: NCSP)*
  $\mathcal{P} = [[\{\mathbf{OC}_{\rho_\mathcal{C}}(D), C, \{\mathbf{B_q}(\mathbf{OC}_{\rho_\mathcal{C}}(D))\}\}]]$
1 **while** *(getNext(*$\mathcal{P}$*,* $\mathcal{C}$*, solution))* **do**
    $\mathcal{U} \leftarrow \{\text{solution}\} \cup \mathcal{U}$
  end
  return $\mathcal{U}$
end.
function getNext(inout:$\mathcal{P} = [\mathcal{B} = [\{\mathbf{B} \in \mathbb{I}^n, \mathcal{C} : \text{NCSP}, \{\mathbf{B_q} \in \mathbb{I}^n\}\} \mid T_\mathcal{B}] \mid T_\mathcal{P}]$*;*
in: $\mathcal{C}_\mathcal{G} \in \text{NCSP}$*;* **out:** *solution*$\in \mathbb{I}^n$*)* $\rightarrow$ *bool*
  forever do
2   **if** *(*$\mathcal{B} = [\,]$*)* **then**
3   **if** *(*$T_\mathcal{P} = [\,]$*)* **then**
4   return (false)
      else
5       $\mathcal{P} \leftarrow T_\mathcal{P}$
      end
6   continue
    end
7   ($\mathcal{C}'$,**B'**,{$\mathbf{B_q}$'}) $\leftarrow$ **reduc**($\mathcal{C}$, **B**, $\{\mathbf{B_q}\}$)
8   $\mathcal{B} \leftarrow T_\mathcal{B}$
9   **if** *(***B'** $<>$ $\emptyset$*)* **then**
10   **if** *(*$\mathcal{C}' = \emptyset$*)* **then**
11   solution $\leftarrow$ **B'**
12   return (true)
      end
13    $\mathcal{B}' \leftarrow$ split(**B'**,$\mathcal{C}'$,{$\mathbf{B_q}$'})
14    $\mathcal{P} \leftarrow [\mathcal{B}' \mid \mathcal{P}]$
    end
  end
end.

as optimization problems, with arti£cial optimization criteria, to £t the solvers. Hence, no signi£cant set of benchmarks is presently available in this area. In this section we present a preliminary evaluation on the following small set of problems.

WP is a 2D simpli£cation 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.

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

SB describes structural and safety restrictions for the components of a ¤oor consisting of a concrete slab on steel beams.

$$SB = \{u + c_1 w^{1.5161} - p = 0, u - (c_6 h_s + c_7)s \leq 0,$$
$$c_2 - c_3 s + c_4 s^2 - c_5 s^3 - h_s \leq 0, c_8 (pw^2)^{0.3976} - h_b \leq 0, c_9 (pw^3)^{0.2839} - h_b \leq 0\}$$

Finally, SC is a collision problem requiring some minimal distance between a trajectory and an object [1].

Algorithm 2: *Problem Reduction*

function reduc*(***in:** $\mathcal{C}$: NCSP, $\mathbf{B}\in \mathbb{I}^n$, $\{\mathbf{B_i}\in \mathbb{I}^n\}$*)* $\rightarrow$*(*NCSP, $\mathbb{I}^n$, $\{\mathbb{I}^n\}$*)*
1 **B'** $\leftarrow \mathbf{OC}_{\rho_{\mathcal{C}}}(\mathbf{B})$
2 **for all** *(*q=$\{$inequality$\}$, q$\in \mathcal{C}$, $\mathbf{B_q} \in \{\mathbf{B_i}\}$*)* **do**
3 **if** *(***B'**$\cap \mathbf{B_q} <> \mathbf{B_q}$*)* **then**
4 $\mathbf{B_q} \leftarrow \mathbf{OC}_{\rho_{\neg q}}$**(B'**$\cap \mathbf{B_q}$**)**
5 **if** *((*$\mathbf{B_q} = \emptyset$*)*$\vee$*Indiscernible(q,*$\mathbf{B_q}$*))* **then**
6 $\mathcal{C} \leftarrow \mathcal{C} \setminus \{\text{q}\}$, $\{\mathbf{B_i}\} \leftarrow \{\mathbf{B_i}\} \setminus \mathbf{B_q}$
end
end
end
7 **for all** *(*q=equality, q$\in \mathcal{C}$*)* **do**
8 **if** *(Indiscernible(q,***B'***))* **then**
9 $\mathcal{C} \leftarrow \mathcal{C} \setminus \{\text{q}\}$, $\{\mathbf{B_i}\} \leftarrow \{\mathbf{B_i}\} \setminus \mathbf{B_q}$
end
end
10 return ($\mathcal{C}$, **B'**, $\{\mathbf{B_i}\}$)
end.

$$SC = \{\forall t, \sqrt{(2.5\sin t - x)^2 + (2.5\cos t - y)^2} \geq 0.5, t : [-\pi..\pi], x, y : [-5..5]\}$$

On problems with exactly one inequality and no equalities, UCA6(**EqFeasibleIneq**) de£ned further is equivalent to ICAb5 presented in [1].

For the **splitFeasible** strategy our implementation chooses the inequality, q, whose split yields the child node with maximal search space where q is eliminated as completely feasible. For the other two splitting strategies, constraints and variables are chosen in a round robin fashion. We use *frag*=0.2. We have tested three combinations of these strategies: **EqIneq**:(splitEq,splitIneq), **IneqEq**:(splitIneq,splitEq), and **EqFeasibleIneq**:(splitEq,splitFeasible,splitIneq). The $\mathbf{OC}$ operator is 3B-consistency. DMBC is implemented with ILOG Solver. The obtained results are described in the following array:

Problem	DMBC (boxes / seconds)	EqIneq	IneqEq	EqFeasibleIneq
P_1 ($\varepsilon = .1$)	8280 / 3.38s	276 / 1.67s	410 / 1.47s	199 / 0.88s
SB ($\varepsilon = .02$)	67122 / 182.2s	122 / 0.47s	148 / 0.46s	92 / 0.35s
WP ($\varepsilon = .1$)	$>$100000 / $>$238s	5021 / 2.01s	5021 / 2.01s	5561 / 15.18s
SC ($\varepsilon = .1$)	16384 / 68.36s	3022 / 54.88s	3022 / 54.88s	2857 / 53s

7 Conclusion

Interval-constraint based solvers are usually designed to deliver punctual solutions. Their techniques work ef£ciently for problems with equalities, but might alter both ef£ciency and compactness of the output representation for many problems with inequalities. In this paper, we propose an algorithm for numerical CSPs with mixed equality/inequality constraints that remedies this state of affairs. The approach combines the classical interval search techniques for equalities with set covering strategies designed to reduce the number of boxes approximating inequalities.

Algorithm 3: *Splitting*

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

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

end.

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

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

end.

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

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

end.

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

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

end.

References

1. F. Benhamou and F. Goualard. Universally quanti£ed interval constraints. In *Procs. of CP'2000*, pages 67–82, 2000.
2. J. Garloff and B. Graf. Solving strict polynomial inequalities by Bernstein expansion. *Symbolic Methods in Control System Analysis and Design, London: IEE*, pages 339–352, 1999.
3. L. Granvilliers. *Consistances locales et transformations symboliques de contraintes d'intervalles*. PhD thesis, Université d'Orléans, déc 98.
4. ILOG. *Solver 4.4, Reference Manual*. ILOG, 1999.
5. L. Jaulin. *Solution globale et garantie de problÁmes ensemblistes ; Application Á l'estimation non linéaire et Á la commande robuste*. PhD thesis, Université Paris-Sud, Orsay, Feb 94.
6. O. Lhomme and M. Rueher. Application des techniques CSP au raisonnement sur les intervalles. *Revue d'intelligence arti£cielle*, 11(3):283–311, 97. Dunod.
7. D. Sam-Haroud and B. Faltings. Consistency techniques for continuous constraints. *Constraints, An International Journal,1*, pages 85–118, 96.
8. P. Van Hentenryck. A gentle introduction to Numerica. *AI*, 103:209–235, 98.