Constraint Propagation and Constraint Solving

Jorge Cruz

DI/FCT/UNL

2020

Constraint Propagation

Constraint Propagation

Fixed-Points of Narrowing Functions

Contraction Obtained by Applying a Narrowing Function

Constraint Propagation Algorithm and its Properties

Local Consistencies

Arc-Consistency and Interval-Consistency

Hull-Consistency and Box-Consistency

Higher Order Consistencies

Shaving and Probing

Constraint Propagation

Propagation is a successive reduction of variables domains by successive application of narrowing functions

An important concept is the notion of a fixed point

Fixed-Points. Let P=(X,D,C) be a CCSP. Let NF be a narrowing function associated with a constraint of C. Let A be an element of Domain$_{NF}$. A is a fixed-point of NF iff:

$$NF(A) = A.$$

The set of all fixed-points of NF within A, denoted Fixed-Points$_{NF}$(A), is the set:

$$\text{Fixed-Points}_{NF}(A) = \{ A_i \in \text{Domain}_{NF} \mid A_i \subseteq A \ \wedge NF(A_i) = A_i \}$$ ❑

The union of all fixed-points of a monotonic narrowing function within A is a fixed-point which is the greatest fixed-point within A

Union of Fixed-Points. Let P=(X,D,C) be a CCSP. Let NF be a monotonic narrowing function associated with a constraint of C, and A an element of its domain. The union of all fixed-points of NF within A is the greatest fixed-point of NF in A:

$$\cup\text{Fixed-Points}_{NF}(A) \in \text{Fixed-Points}_{NF}(A)$$

$$\forall A_i \in \text{Fixed-Points}_{NF}(A) \ A_i \subseteq \cup\text{Fixed-Points}_{NF}(A)$$ ❑

Constraint Propagation

The contraction resulting from applying a monotonic narrowing function to A is limited by the greatest fixed-point within A:

> No value combination included in the greatest fixed-point may be discarded in the contraction
>
> If the monotonic narrowing function is idempotent, the result of the contraction is precisely the greatest fixed-point within A

Contraction Applying a Narrowing Function. Let P=(X,D,C) be a CCSP. Let NF be a monotonic narrowing function associated with a constraint of C and A an element of its domain. The greatest fixed-point of NF within A is included in the element obtained by applying NF to A:

$$\cup\text{Fixed-Points}_{NF}(A) \subseteq NF(A)$$

In particular, if NF is also idempotent then:

$$\cup\text{Fixed-Points}_{NF}(A) = NF(A)$$

❑

Constraint Propagation

The propagation algorithm applies successively each narrowing function until a fixed-point is attained:

```
function prune(a set Q of narrowing functions, an element A of the domains lattice)
      S ← ∅ ;
      while Q ≠ ∅ do
          choose NF ∈ Q;
          A' ← NF(A) ;
          if A' = ∅ then return ∅ ;
          P ← { NF' ∈ S: ∃x∈Relevant_NF' A[x] ≠ A'[x] } ;
          Q ← Q ∪ P ; S ← S \ P ;
          if A' = A then Q ← Q \ {NF} ; S ← S ∪ {NF} end if;
          A ← A' ;
      end while
      return A ;
end function
```

The algorithm is an adaptation of the original propagation algorithm AC3 used for solving CSPs with finite domains

Constraint Propagation

From the properties of the narrowing functions it is possible to prove that the propagation algorithm terminates and is correct

If all the narrowing functions are monotonic then it is confluent (the result is independent from the selection criteria) and computes the greatest common fixed-point included in the initial domains

Properties of the Propagation Algorithm. Let $P=(X,D,C)$ be a CCSP. Let set S_0 be a set of narrowing functions (obtained from the set of constraints C). Let A_0 be an element of Domain$_{NF}$ (where $NF \in S_0$) and d an element of D ($d \in D$). The propagation algorithm $prune(S_0, A_0)$ terminates and is correct:

$$\forall_{d \in A_0}\ d \text{ is a solution of the CCSP} \Rightarrow d \in prune(S_0, A_0)$$

If S_0 is a set of monotonic narrowing functions then the propagation algorithm is confluent and computes the greatest common fixed-point included in A_0. ❑

The selection criterion is irrelevant for the pruning obtained but it may be very important for the efficiency of the propagation

Local Consistency

The fixed-points of the narrowing functions associated with a constraint characterize a local property enforced on its variables

Such property is called local consistency:

- depends only on the narrowing functions associated with one constraint (local)
- defines the value combinations that are not pruned by them (consistent)

Local consistency is a partial consistency:

- imposing it on a constraint is not sufficient to remove all inconsistent value combinations among its variables

Local Consistency

Local consistencies used continuous domains are approximations of arc-consistency developed for finite domains

Arc-Consistency and Interval-Consistency

A constraint is said to be arc-consistent wrt a set of value combinations iff, within this set, for each value of each variable there is a value combination that satisfy the constraint:

Arc-Consistency. Let $P=(X,D,C)$ be a CSP. Let $c=(s,\rho)$ be a constraint of the CSP. Let A be an element of the power set of D ($A \in 2^D$). The constraint c is arc-consistent wrt A iff:

$$\forall_{x_i \in s} \forall_{d_i \in A[x_i]} \exists_{d \in A[s]} \; (d[x_i]=d_i \wedge d \in \rho)$$

which, is equivalent to:

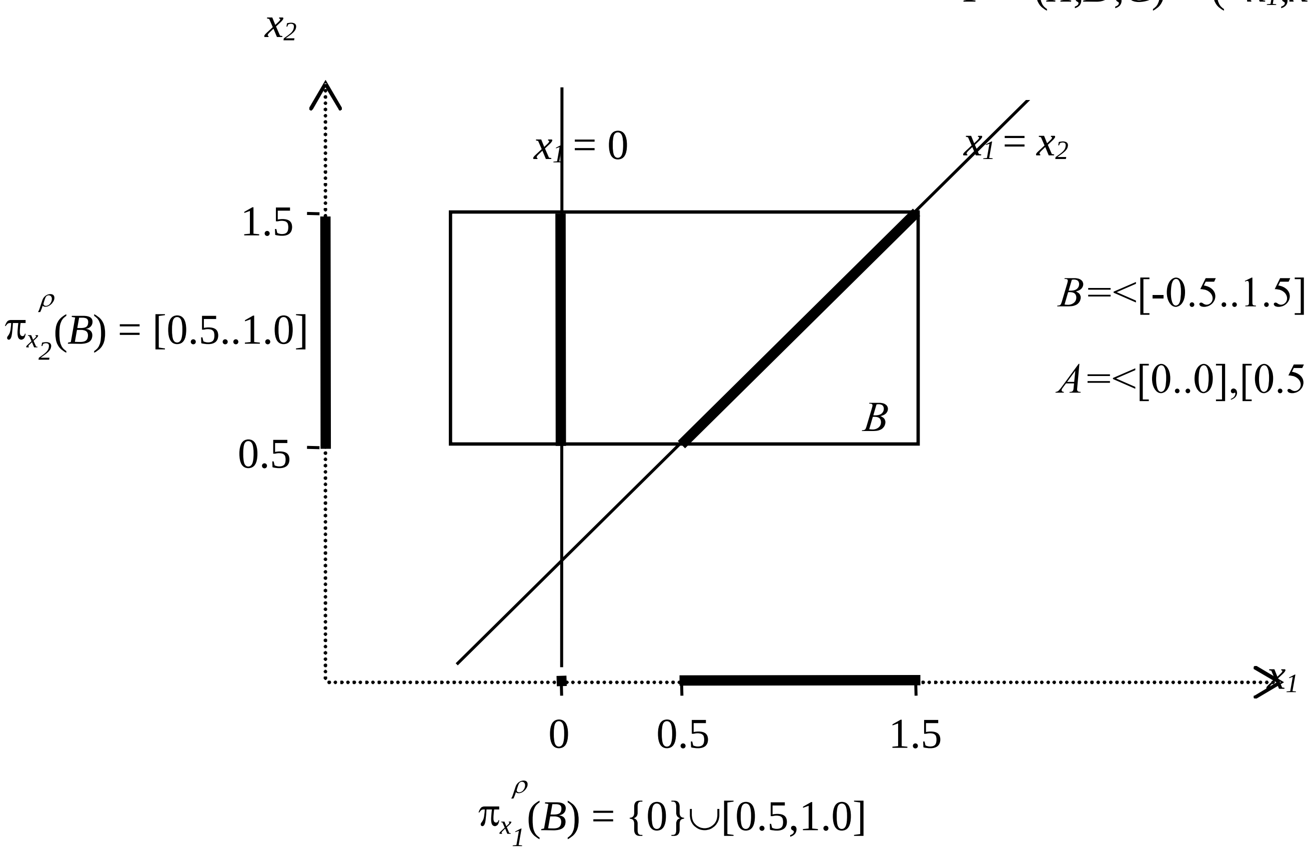
$$\forall_{x_i \in s} A[x_i] = \{ d[x_i] \mid d \in \rho \cap A[s] \} = \pi^{\rho}_{x_i}(A[s]) \qquad \square$$

Local Consistency

Example

$P = (X,D,C) = (\langle x_1,x_2\rangle, D_1 \times D_2, \{c\})$

$c \equiv x_1 \times (x_2 - x_1) = 0$



$B = \langle [-0.5..1.5],[0.5..1.5] \rangle$

$A = \langle [0..0],[0.5..1.5] \rangle \cup \langle [0.5..1.5],[0.5..1.5] \rangle$

B is not arc-consistent (ex: if x_1=0.25 there is no value for x_2 to satisfy c)

A is arc-consistent ($\pi^{\rho}_{x_1}(A) = A[x_1]$ and $\pi^{\rho}_{x_2}(A) = A[x_2]$)

Local Consistency

Arc-Consistency and Interval-Consistency

In continuous domains, arc-consistency cannot be obtained in general due to machine limitations for representing real numbers

The best approximation of arc-consistency wrt a set of real valued combinations is the set approximation of each variable domain

A constraint is interval-consistent wrt a set of value combinations iff for each canonical *F*-interval representing a variable sub-domain there is a value combination satisfying the constraint

Interval-Consistency. Let P=(X,D,C) be a CCSP. Let c=(s,ρ) be a constraint of the CCSP ($c\in C$). Let A be an element of the power set of D ($A\in 2^D$). The constraint c is interval-consistent wrt A iff:

$$\forall_{x_i\in s}\ \forall_{[a..a^+]\subseteq A[x_i]}\ \exists_{d\in A[s]}\ (d[x_i]\in(a..a^+)\ \wedge\ d\in\rho)\ \wedge$$

$$\forall_{[a]\subseteq A[x_i]}\ \exists_{d\in A[s]}\ (d[x_i]\in(a^-..a^+)\ \wedge\ d\in\rho)\qquad \text{(where } a \text{ is an } F\text{-number)}$$

which is equivalent to:

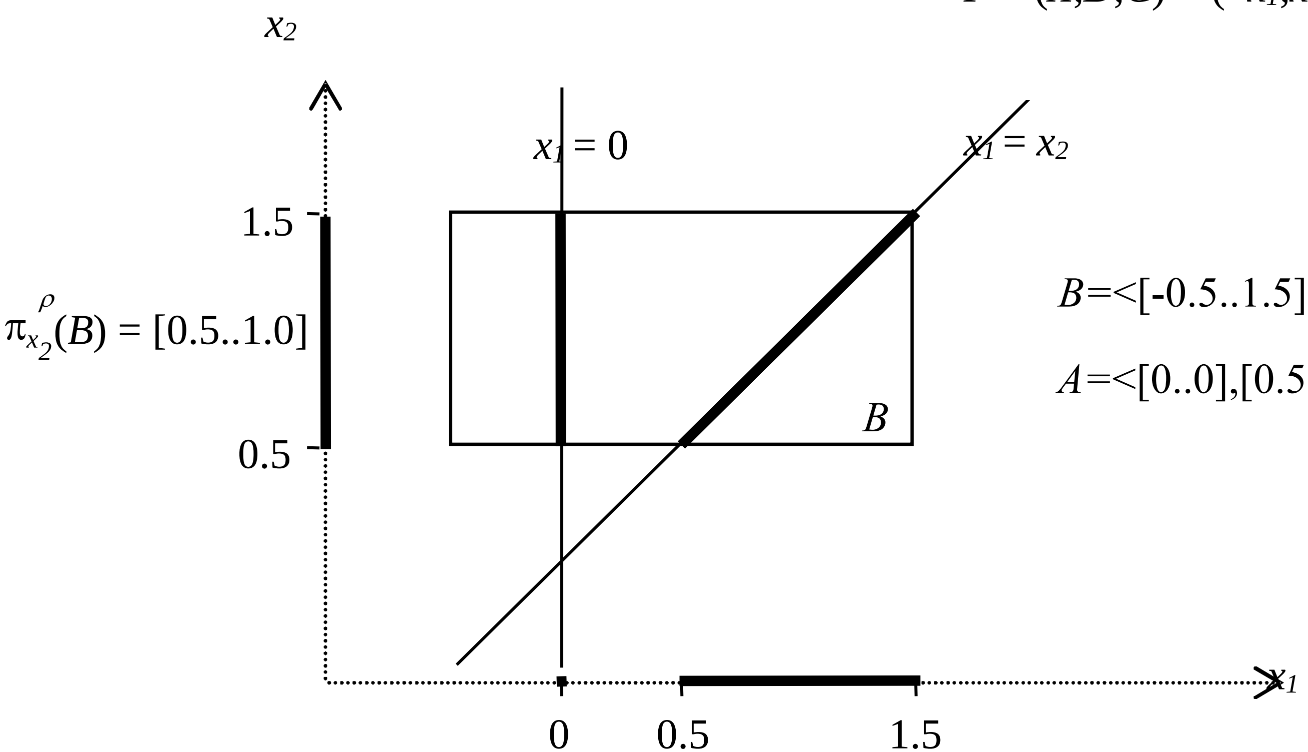
$$\forall_{x_i\in s}\ A[x_i] = S_{apx}(\{\ d[x_i] \mid d\in\rho\cap A[s]\ \}) = S_{apx}(\pi^{\rho}_{x_i}(A[s]))$$ ❑

Local Consistency

Example

$P = (X,D,C) = (<x_1,x_2>, D_1 \times D_2, \{c\})$

$c \equiv x_1 \times (x_2 - x_1) = 0$



B=<[-0.5..1.5],[0.5..1.5]>

A=<[0..0],[0.5..1.5]>∪<[0.5..1.5],[0.5..1.5]>

$\pi^{\rho}_{x_1}(B) = \{0\} \cup [0.5,1.0]$

B is not interval-consistent (if $x_1 \in [0.250,0.251]$ there is no x_2 satisfying c)

A is interval-consistent ($S_{apx}(\pi_{x_1}{}^{\rho}(A)) = A[x_1]$ and $S_{apx}(\pi_{x_2}{}^{\rho}(A)) = A[x_2]$)

Local Consistency

Arc-Consistency and Interval-Consistency

Interval-consistency can only be enforced on primitive constraints where the set approximation of the projection function can be obtained using extended interval arithmetic

Structures (not F-intervals) must be considered for representing each variable domain as a non-compact set of real values

In practice, the enforcement of interval-consistency can be applied only to small problems:

the number of non-contiguous F-intervals may grow exponentially, requiring an unreasonably number of computations for each narrowing function.

The approximations of arc-consistency most widely used in continuous domains assume the convexity of the variable domains, in order to represent them by single F-intervals

Local Consistency

Hull-Consistency

Hull-consistency (or 2B-consistency) requires the satisfaction of the arc-consistency property only at the bounds of the *F*-intervals that represent the variable domains

A constraint is said to be hull-consistent wrt an *F*-box iff, for each bound of the *F*-interval representing the domain of a variable there is a value combination satisfying the constraint:

Hull-Consistency. Let $P=(X,D,C)$ be a CCSP. Let $c=(s,\rho)$ be a constraint of the CCSP ($c\in C$). Let B be an *F*-box which is an element of the power set of D ($B\in 2^D$). The constraint c is hull-consistent wrt B iff:

$$\forall_{x_i\in s}\ \exists_{d_l\in B[s]}\,(d_l[x_i]\in[a..a^+)\wedge d_l\in\rho)\ \wedge$$

$$\exists_{d_r\in B[s]}\,(d_r[x_i]\in(b^-..b]\wedge d_r\in\rho) \qquad \text{(where } B[x_i]=[a..b])$$

which is equivalent to:

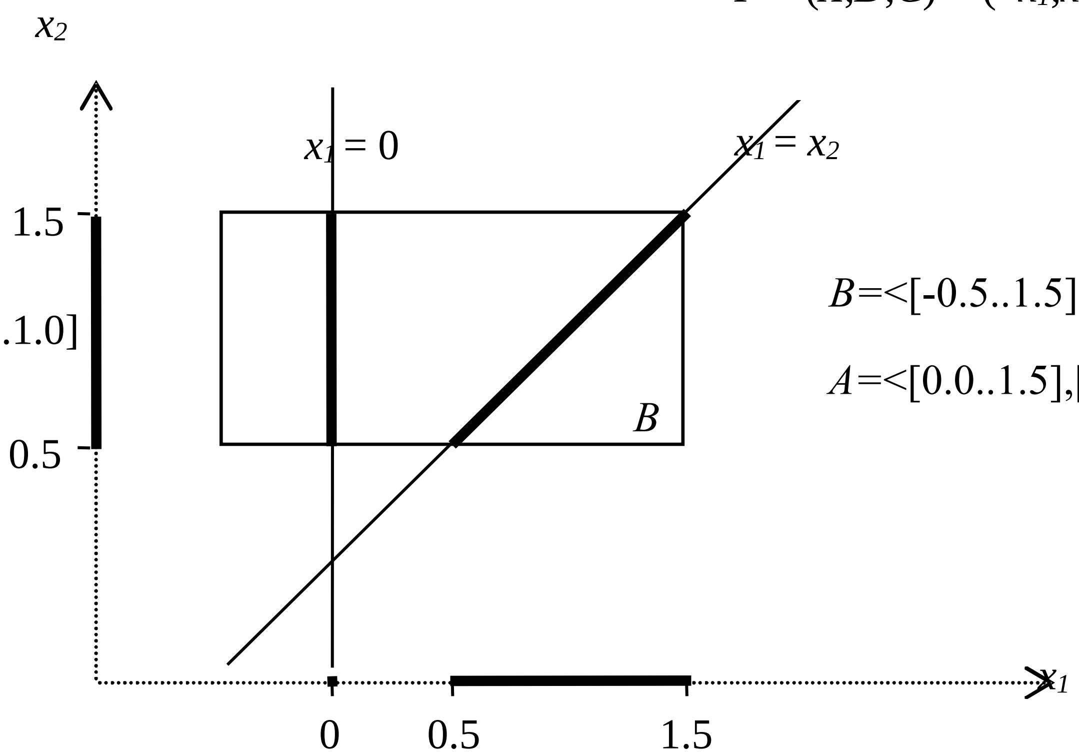
$$\forall_{x_i\in s}\ B[x_i]=I_{hull}(\{\,d[x_i]\mid d\in\rho\cap B[s]\,\})=I_{hull}(\pi^{\rho}_{x_i}(B[s])) \qquad \square$$

Local Consistency

Example

$P = (X,D,C) = (<x_1,x_2>, D_1 \times D_2, \{c\})$

$c \equiv x_1 \times (x_2 - x_1) = 0$



$\pi_{x_2}^{\rho}(B) = [0.5..1.0]$

$B = <[-0.5..1.5],[0.5..1.5]>$

$A = <[0.0..1.5],[0.5..1.5]>$

$\pi_{x_1}^{\rho}(B) = \{0\} \cup [0.5,1.0]$

B is not hull-consistent (if $x_1 \in [-0.5,-0.499]$ there is no x_2 satisfying c)

A is hull-consistent ($I_{hull}(\pi_{x_1}^{\rho}(A)) = A[x_1]$ and $I_{hull}(\pi_{x_2}^{\rho}(A)) = A[x_2]$)

Local Consistency

Hull-Consistency

HC3-Revise and HC4-Revise enforce Hull-consistency on a constraint by explicitly (HC3-Revise) or implicitly (HC4-Revise) decomposing it into primitive constraints

The major drawback of any decomposition approach is the worsening of the dependency problem:

- the satisfaction of a local property on each primitive constraint does not imply the existence of value combinations satisfying simultaneously all of them

HC3-Revise and HC4-Revise are particularly ineffective if the original constraint contain multiple occurrences of variables

Local Consistency

Box-Consistency

The drawbacks of the decomposition approach motivated the constraint Newton method, which can be applied directly to complex constraints

A constraint is said to be box-consistent wrt an F-box iff, for each bound of the F-interval representing the domain of a variable there is a box (bound+other F-intervals) that satisfies the interval projection condition:

Box-Consistency. Let $P=(X,D,C)$ be a CCSP. Let $c=(s,\rho)$ be a constraint of the CCSP ($c\in C$) expressed in the form $e_c\diamond 0$ (with $\diamond\in\{\leq,=,\geq\}$ and e_c a real expression). Let F_E be an interval expression representing an interval extension F of the real function represented by e_c. Let B be an F-box which is an element of the power set of D ($B\in 2^D$). c is box-consistent wrt B and F_E iff:

$$\forall x_i\in s\ \exists r_1\in F_E(B_1)\ r_1\diamond 0 \wedge \exists r_2\in F_E(B_2)\ r_2\diamond 0$$

where B_1 and B_2 are two F-boxes such as:

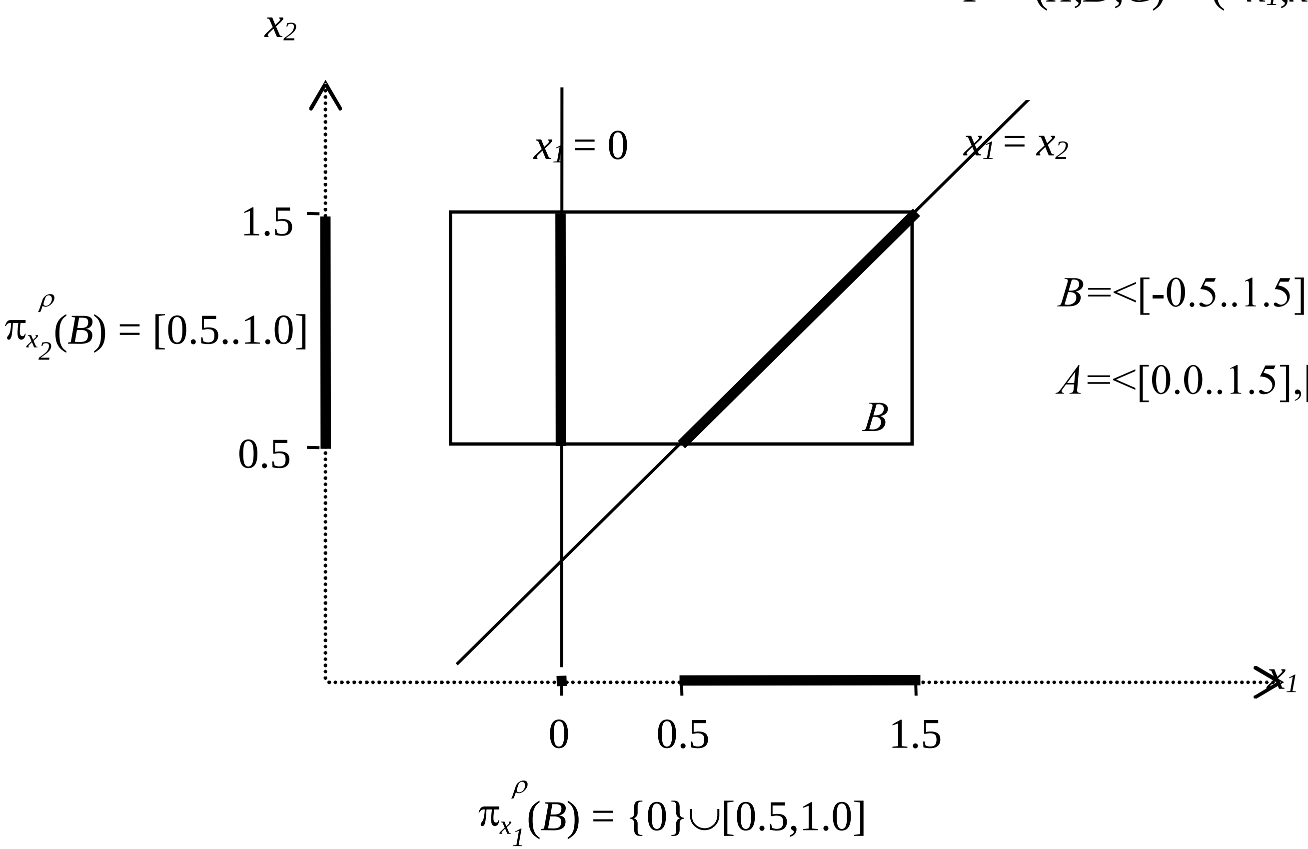
$$B_1[x_i]=cleft(B[x_i]),\ B_2[x_i]=cright(B[x_i]) \text{ and } \forall_{x_j\in s}\ (x_j\neq x_i\Rightarrow B_1[x_j]=B_2[x_j]=B[x_i]).$$ ❑

Local Consistency

Example

$P = (X,D,C) = (<x_1,x_2>, D_1 \times D_2, \{c\})$

$c \equiv x_1 \times (x_2 - x_1) = 0$



$B = <[-0.5..1.5],[0.5..1.5]>$

$A = <[0.0..1.5],[0.5..1.5]>$

B is not box-consistent $(0 \notin [-0.5,-0.499] \times ([0.5,1.5]-[-0.5,-0.499]) = [-1,-0.498])$

A is box-consistent:

$0 \in [0..0.001] \times ([0.5..1.5]-[0..0.001])$ and $0 \in [1.499..1.5] \times ([0.5..1.5]-[1.499..1.5])$

$0 \in [0..1.5] \times ([0.5..0.501]-[0..1.5])$ and $0 \in [0..1.5] \times ([1.499..1.5]-[0..1.5])$

Local Consistency

Box-Consistency

Although box-consistency is weaker than hull-consistency for the same constraint, the enforcement of box-consistency may achieve better pruning since it may be directly applied to complex constraints with BC3-Revise

For primitive constraints box-consistency and hull-consistency are equivalent (with infinite precision)

For complex constraints box-consistency is stronger than hull-consistency applied on the primitive constraints obtained by decomposition

Consistency Enforcement

Local Consistency and Higher Order Consistencies

Generalising the concept of local consistency from a constraint to the set of constraints:

> a CCSP is locally consistent (interval, hull or box-consistent) wrt a set A of real valued combinations iff all its constraints are locally consistent wrt A

Since the propagation algorithm obtains the greatest common fixed-point (of the monotonic narrowing functions) included in the original domains, then applying it to a set A results in the largest subset $A' \subseteq A$ for which each constraint is locally consistent.

Local-Consistency. Let $P=(X,D,C)$ be a CCSP. Let A be an element of the power set of D ($A \in 2^D$). P is locally-consistent wrt A iff:

$\forall_{c \in C}$ c is locally-consistent wrt A

Let S be a set of monotonic narrowing functions associated with the constraints in C which enforce a particular local consistency by constraint propagation:

P is locally-consistent wrt prune(S,A)

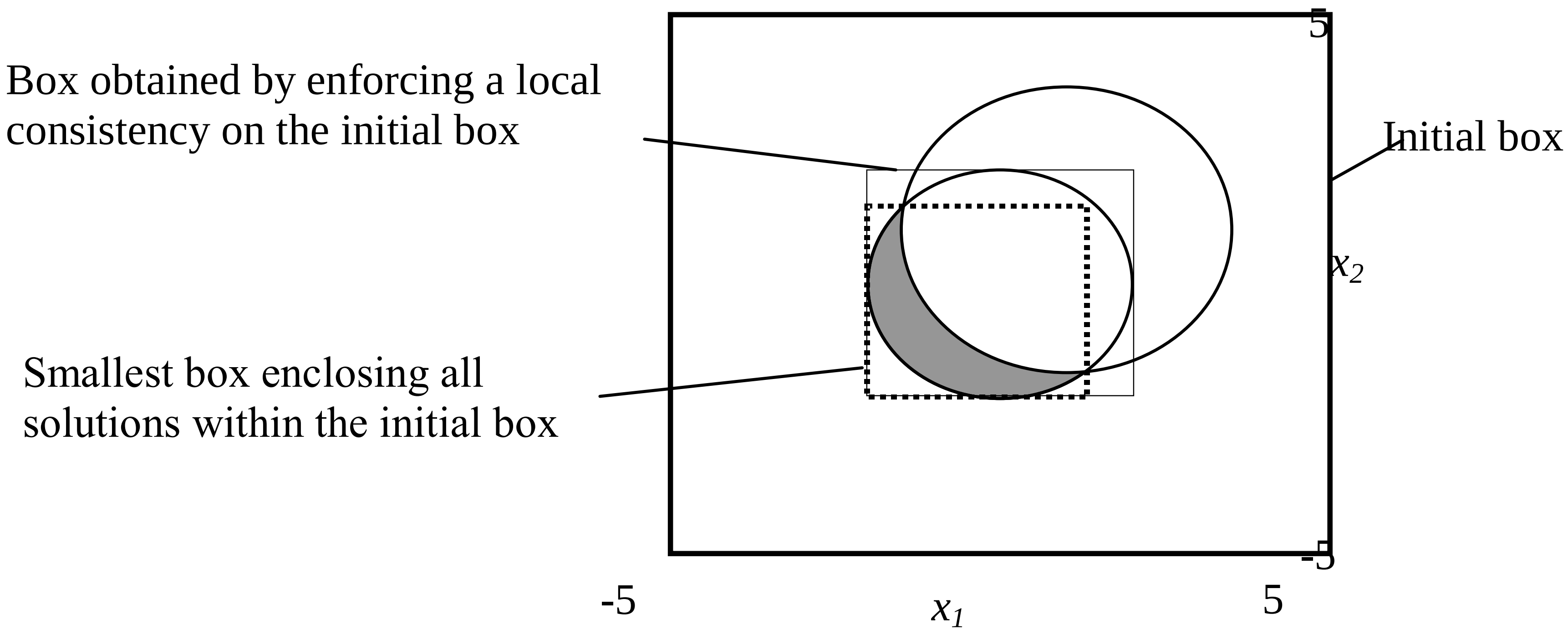
$\forall_{A' \subseteq A}$ (P is locally-consistent wrt $A' \Rightarrow A' \subseteq$ prune(S,A)) ❑

Consistency Enforcement

Local Consistency and Higher Order Consistencies

When only local consistency techniques are applied to non-trivial problems the achieved reduction of the search space is often poor

$c_1 \equiv x_1^2 + x_2^2 - 2^2 \leq 0$ $\quad$ $c_2 \equiv (x_1 - 1)^2 + (x_2 - 1)^2 - 2.5^2 \geq 0.$



Higher Order Consistencies

Better pruning of the variable domains may be achieved if, complementary to a local property, some (global) properties are also enforced on the overall constraint set

Higher order consistency types used in continuous domains are approximations of strong k-consistency (with k>2) restricted to the bounds of the variable domains:

> A CSP is k-consistent (k≥2) iff any consistent instantiation of k-1 variables can be extended by instantiating any of the remaining variables.
> A CSP is strongly k-consistent if it is i-consistent for all i≤k.

Strong 2-consistency corresponds to arc-consistency and hull-consistency is an approximation of strong 2-consistency restricted to the bounds of the variable domains

Higher Order Consistencies

3B-consistency and Bound-consistency, are generalisations of hull and box-consistency respectively:

- if the domain of one variable is reduced to one of its bounds then the obtained F-box must contain a sub-box for which the CCSP is locally consistent.

The following is a generic definition for the consistency types used in continuous domains (local consistency is just a special case with k=2):

***kB*-Consistency.** Let P=(X,D,C) be a CCSP. Let A be an element of the power set of D ($A \in 2^D$) and k an integer number.
P is 2B-Consistent wrt A iff P is locally-consistent wrt A
$\forall_{k>2}$ P is kB-Consistent wrt A iff
$\forall_{x_i \in X}$ ($\exists_{A_1 \subseteq B_1}$ P is (k-1)B-Consistent wrt $A_1 \wedge \exists_{A_2 \subseteq B_2}$ P is (k-1)B-Consistent wrt A_2)
where B_1 and B_2 are two elements of the power set of D such that:
$B_1[x_i]=cleft(B[x_i])$, $B_2[x_i]=cright(B[x_i])$ and $\forall_{x_j \in X}$ $(x_j \neq x_i \Rightarrow B_1[x_j]=B_2[x_j]=B[x_i])$. ❑

Higher Order Consistencies

The algorithms to enforce higher order consistencies interleave constraint propagation with search techniques

Shaving and Probing implement strong consistencies

- one variable is instantiated
- slices are contracted with respect to the complete CCSP

The growth in computational cost of the enforcing algorithms may limit the practical applicability of such criteria

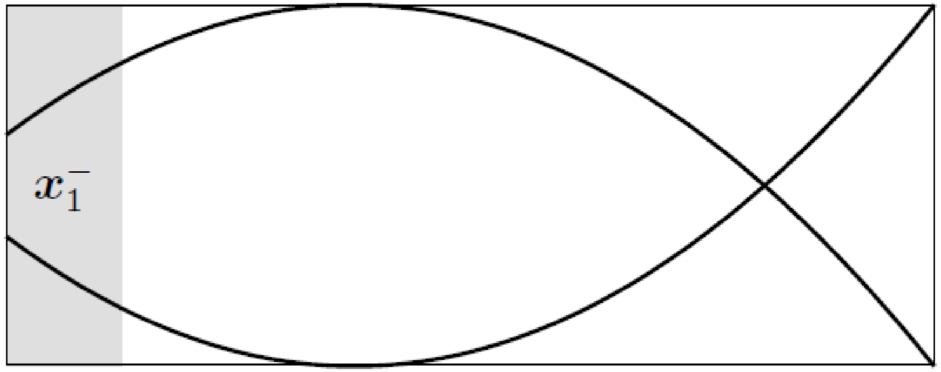
Higher Order Consistencies

Shaving

Iteratively discard slices on the boundaries of an interval domain using local consistency based operators on all the constraints

A value is temporarily assigned to a variable and a partial consistency is enforced to the CCSP. If an inconsistency is obtained then the value can be safely removed from the domain of the variable. Otherwise, the value is kept in the domain.

Shaving contracts one facet at a time



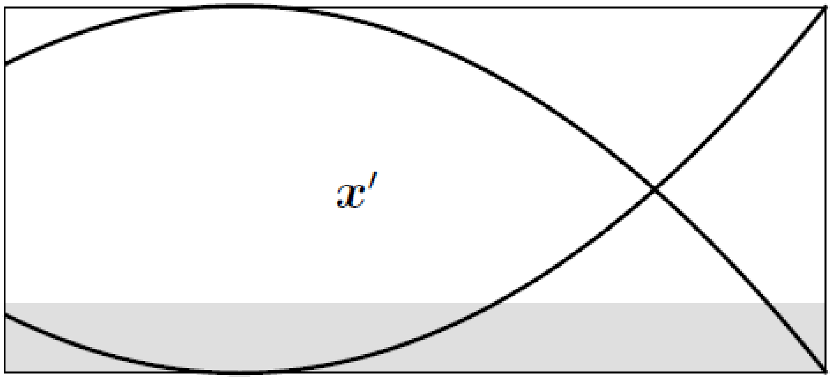
Higher Order Consistencies

Shaving

Iteratively discard slices on the boundaries of an interval domain using local consistency based operators on all the constraints

A value is temporarily assigned to a variable and a partial consistency is enforced to the CCSP. If an inconsistency is obtained then the value can be safely removed from the domain of the variable. Otherwise, the value is kept in the domain.

Shaving contracts one facet at a time

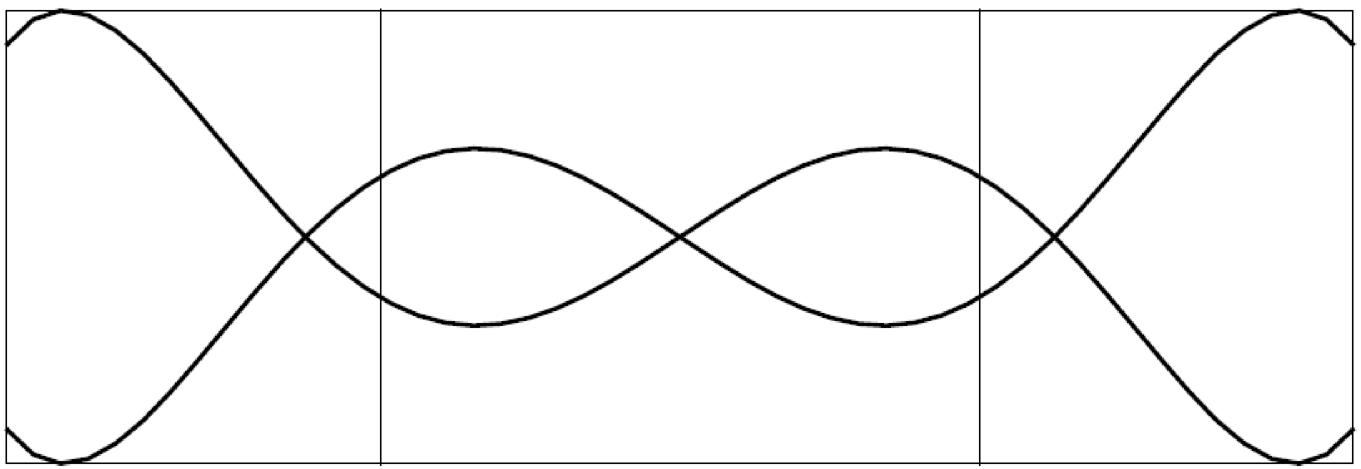


Higher Order Consistencies

Probing

Split the domain of one variable in several parts and contract every sub-box using local consistency operators on all the constraints

The result is the union hull of contracted sub-boxes

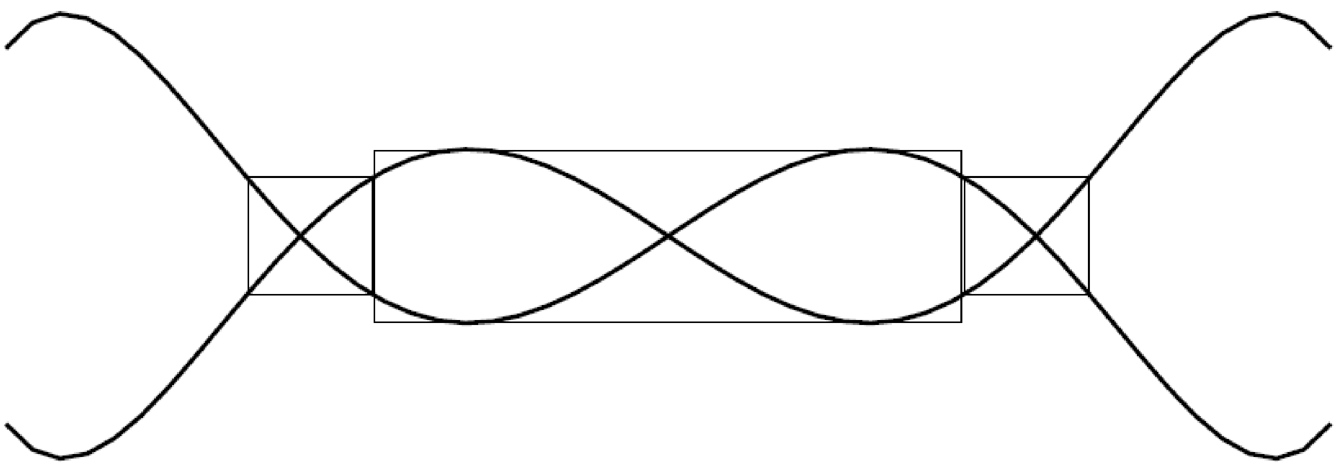


Higher Order Consistencies

Probing

Split the domain of one variable in several parts and contract every sub-box using local consistency operators on all the constraints

The result is the union hull of contracted sub-boxes

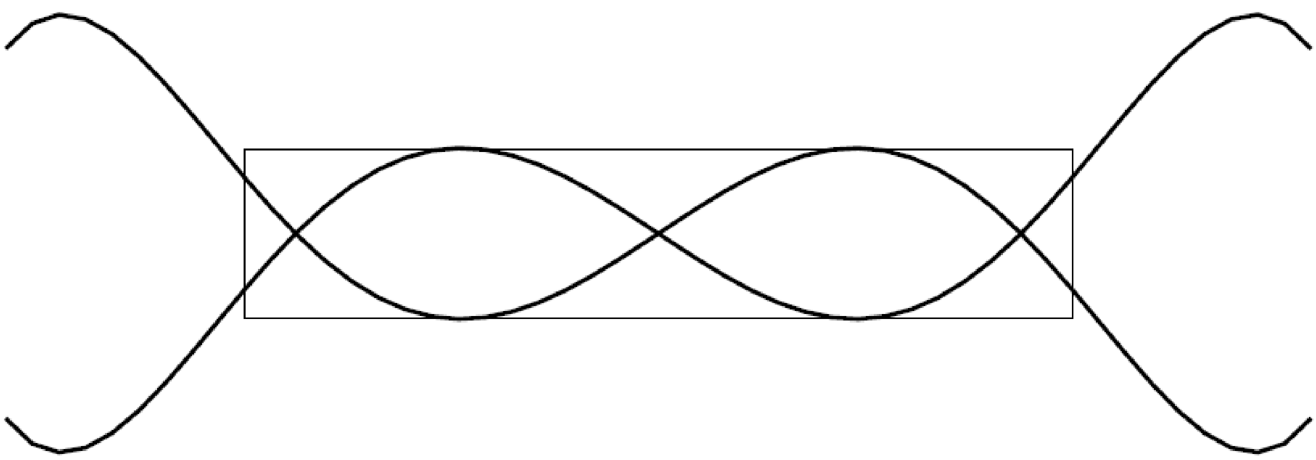


Higher Order Consistencies

Probing

Split the domain of one variable in several parts and contract every sub-box using local consistency operators on all the constraints

The result is the union hull of contracted sub-boxes



With probing all the variable domains are potentially contracted

Higher Order Consistencies

Probing

Split the domain of one variable in several parts and contract every sub-box using local consistency operators on all the constraints

The result is the union hull of contracted sub-boxes

CID (Constructive Interval Disjunction): Is a contractor based on this technique. CID(HC4) propagates HC4-revise onto each sub-box

3BCID: Is a hybrid algorithm mixing constructive interval disjunction and shaving. 3BCID(HC4), 3BCID(Mohc)

ACID: Is an adaptive variant of 3BCID which computes dynamically during search the value of its parameters. ACID(HC4), ACID(Mohc)

Consistency Enforcement

Local Consistency and Higher Order Consistencies

All the consistency criteria used in continuous domains, either local or higher order consistencies, are partial consistencies

The adequacy of a partial consistency for a particular CCSP must be evaluated taking into account the trade-off between the pruning it achieves and its execution time

It is necessary to be aware that the filtering process is performed within a larger procedure for solving the CCSP and it may be globally advantageous to obtain faster, if less accurate, results

Constraint Solving

Branch-and-Prune Algorithm

- Enclosure of the Feasible Space

Branching Strategies

- Bisection
- Variable selection: round-robin, largest-first, smear-based

Branch-and-Bound Algorithm

- Optimization

Modelling Techniques

- Dependency Reduction
- Variable Elimination
- System Scaling

Branch-and-Prune

Solving Continuous Constraint Satisfaction Problems

When the objective is to find all the solutions, interval branch-and-prune algorithms alternate contraction and branching steps until reaching small enough boxes containing the feasible space:

- it starts with *D,* the box of the initial domains to process
- at each iteration, a box of the paving to process is selected and contracted according to the constraints
- if it becomes empty, it is discarded
- if is proved that it contains only solutions, it is added to the set of inner boxes (and not further processed)
- else, if it has reached the prescribed precision, it is added to the set of boundary boxes (and not further processed)
- otherwise it is split into sub-boxes to be further processed

Branch-and-Prune

Branching Strategies

Box bisection is the usual split strategy:

- choose one variable x_i from the box with its interval domain larger than the prescribed precision
- generate two sub-boxes equal to the original box except in the x_i domain: one with the left part (from lower bound to midpoint); the other with the right part (from midpoint to upper bound)

The most commonly used variable selection strategies are:

- round-robin
- largest-first
- smear-based strategies

Branch-and-Prune

Branching Strategies

Round-robin variable selection strategy:

- is a fair strategy where each variable is regularly chosen
- the goal is to refrain from neglecting any variable
- a bad initial ordering of variables can lead to bed performance

Largest-first variable selection strategy:

- selects the variable with the largest domain
- intervals with large domains penalize the contracting methods
- is also a fair strategy

Branch-and-Prune

Branching Strategies

Smear-based variable selection strategies:

- estimate the impact of the variables on each constraint using the partial derivatives and the sizes of the variable domains
- aggregate these values to estimate the impact of each variable on the whole system
- select the variable with the greatest impact

The impact of x_i on a function g_j is computed by the smear value:

$$smear(x_i, g_j) = width([x_i]) \times mag(\frac{\delta g_j}{\delta x_i}[x]) \quad \text{with} \quad mag([a,b]) = \max(|a|,|b|)$$

the smear value estimates the reduction of the image size of g_j consequent upon a reduction of the domain size of x_i

Branch-and-Bound

Optimization

When the objective is to solve a continuous constraint optimization problem, interval branch-and-bound algorithms are used, that alternate branching, bounding and contraction steps

- branching is similar to the branch-and-prune algorithms
- upper bounding consists in finding feasible solutions within a box with costs lower than the best found solution (local optimization techniques are used for upper bounding)
- lower bounds may be computed with interval methods or linear methods applied to safe relaxations (used for selecting the next box to process and in stopping criteria)
- contraction is similar to the branch-and-prune algorithms
- additional constraints on the bounds of the objective function and first-order optimality conditions

Modelling Techniques

Frequently a single problem may be modelled by several equivalent CCSPs

The behaviour of a constraint solver may change drastically even with equivalent CCSPs

To choose the *best* model for a particular problem is important to understand the underlying constraint propagation algorithms

Some modelling techniques are commonly adopted for improving the accuracy and efficiency of the continuous constraint solvers

Modelling Techniques

Dependency Reduction

A fundamental problem of interval arithmetic is the dependency problem (see lecture 2).

> **Dependency Problem.** In the interval arithmetic evaluation of an interval expression, each occurrence of the same variable is treated as a different variable. The dependency between the different occurrences of a variable in an expression is lost. ❑

Some expressions may be rewritten into equivalents that minimize the dependency problem

Examples:

- Factorize as much as possible polynomial expressions:
  - Instead of using constraint $x^2y^2+xy^2+xy=0$ use constraint $xy(y(x+1)+1)=0$
- Use better interval extensions (mean value form, Taylor form,…):
  - Instead of using constraint $x-x^2=0$ use constraint $0.25-(x-0.5)^2=0$

Modelling Techniques

Variable Elimination

Continuous constraint solvers rely on the efficiency of branch and prune algorithms for enforcing consistency on the CCSP variables

Precision and efficiency may be improved if the number of variables is reduced

Sometimes a set of constraints may be rewritten into an equivalent set with less variables

Example:

Instead of using the constraint system: $\begin{cases} x_1 + x_2 + x_3 = -1 \\ (x_1 + x_1 x_2 + x_2 x_3)\, x_4 = c_1 \\ (x_2 + x_1 x_3)\, x_4 = c_2 \\ x_3 x_4 = c_3 \end{cases}$

Consider $x_4 = c_3/x_3$ and use the constraint system: $\begin{cases} x_1 + x_2 + x_3 = -1 \\ (x_1 + x_1 x_2 + x_2 x_3) c_3 = c_1 x_3 \\ (x_2 + x_1 x_3) c_3 = c_2 x_3 \end{cases}$

Modelling Techniques

System Scaling

Continuous constraint solvers rely on interval techniques for dealing with numerical errors.

A consequence of numerical errors is the amplification of the variable domains and poor pruning results

Two major sources of numerical errors are:

- operations with large numbers (lower density of F-Numbers at this ranges)
- operands with different magnitudes

Scaling the system and making some variable substitutions may avoid such situations as much as possible

Example:

Instead of using constraint: $10^{-20}x^2+3x+2\times10^{20}=0$

Consider $x = 10^{20}y$ and use the constraint: $y^2+3y+2=0$