Hybrid Constraints over Continuous Domains: introduction

Michel RUEHER

University of Nice Sophia-Antipolis / I3S – CNRS, France

June, 2011

ACP Summer School

"Hybrid Methods for Constraint Programming"

Turunç

Outline

Motivations

Interval Programming

Constraint Programming

Applications

Why do we need intervals?

- **Modelling uncertainty**
  - Error in Measurement or uncertainty in measurements
  - Uncertainty when estimating unknown values

- **Safe Computations** with floating-point numbers
  - Rounding errors
  - Cancellation, ...

What Every Computer Scientist Should Know About Floating-Point Arithmetic, Goldberg, 1991

Problem with floating-point computations

Examples

(in simple precision)

- **Absorption:** $10^7 + 0.5 = 10^7$
- **Cancellation:** $((1 - 10^{-7}) - 1) * 10^7 = -1.192...(\neq -1)$
- Operations are **not associative:** $(10000001 - 10^7) + 0.5 \neq 10000001 - (10^7 + 0.5)$
- **No exact representation:** $0.1 = 0.000110011001100\ldots$

Rump polynomial

- `RumpFunc[x_,y_]:=`$(1335/4 - x^2)y^6 + x^2(11x^2y^2 - 121y^4 - 2) + (11/2)y^8 + x/(2y)$
- Value computed with rational numbers: `RumpFunc[77617, 33096]`$= -\frac{54767}{66192} = -0.827396$
- Value with floating point numbers: **0**
- Value with floating point numbers when $11/2$ is replaced by 5.5 in the polynomial: 1.18059×10^{21}

What is an interval?

An **interval** $[a, b]$ describes a set of real numbers x such that: $a \leq x \leq b$

Assumption:

a and b belong to **finite set** of numbers representable on a computer: **floating-point numbers**, subset of integers, rational numbers, ...

A **Box** denotes a Cartesian product of intervals

$\rightarrow$ a box is a vector of intervals that defines the **search space** of problem,
i.e., the space in which are the values of the variables

Interval Programming

- Basics on **interval arithmetic**
- **Interval Newton-like methods** for solving a multi-variate system of non-linear equations

Interval arithmetic: notations

$c_j(x_1, \ldots, x_n)$	A relation over the real numbers
x, y	Real variables or vectors
X or $\mathbf{x}$ or D_x	The domain of variable x (i.e. intervals)
$X = [\underline{X}, \overline{X}]$	The set of real numbers x verifying
$\mathbf{x} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$	$\underline{X} \leq x \leq \overline{X}$ (resp. $\underline{\mathbf{x}} \leq x \leq \overline{\mathbf{x}}$)
$\mathcal{C}$	The set of constraints
$\mathcal{D}$	The set of domains of all the variables
$\mathcal{R}$	The set of real numbers
$\mathcal{R}^{\infty}$	$\mathcal{R} \cup \{-\infty, +\infty\}$, set of real numbers extended with infinity symbols
$\mathcal{F}$	The set of floating point numbers
a^{+} (resp. a^{-})	The smallest (resp. largest) number of $\mathcal{F}$ strictly greater (resp. lower) than a
$\tilde{k}$	smallest interval containing real number k
$\Phi_{cstc}(P)$	closure (filtering) by consistency of CSP P
	$cstc$ stands for $2B, Box, 3B, Bound$

Interval arithmetic: basic definitions (1)

Interval arithmetic (Moore-1966)
is based on the representation of variables as intervals

Let f be a real-valued function of n unknowns $\{x_1, \ldots, x_n\}$, an **interval evaluation** F of f for given ranges $\mathbf{X} = \{X_1, \ldots, X_n\}$ for the unknowns is an interval Y such that

$$\forall \{v_1, \ldots, v_n\} \in \{X_1, \ldots, X_n\} : \ \underline{Y} \leq f(v_1, \ldots, v_n) \leq \overline{Y}$$

$\underline{Y}, \overline{Y}$: lower and upper bounds for the values of f when the values of the unknowns are restricted to the box $\mathbf{X}$

Interval arithmetic: basic definitions (2)

Let $C : \mathcal{I}^n \to \mathcal{B}ool$ be a **relation** over the intervals

C is an **interval extension** of the relation $c : \mathcal{R}^n \to \mathcal{B}ool$ iff:

$$\forall X_1, \ldots, X_n \in \mathcal{I} : \exists v_1 \in X_1 \wedge \ldots \wedge \exists v_n \in X_n \wedge c(v_1, \ldots, v_n) \Rightarrow C(X_1, \ldots, X_n)$$

For instance, $X_1 \doteq X_2 \Leftrightarrow (X_1 \cap X_2) \neq \emptyset$ is an interval extension of the relation $x_1 = x_2$ over the real numbers

Example:

Relation $X_1 \doteq X_2$ holds if $X_1 = [0, 17.5]$ and $X_2 = [17, 27.5]$

Natural interval extension (1)

- In general, it is not possible to compute **the exact enclosure** of the range for an arbitrary function over the real numbers

  $\longrightarrow$ The interval extension of a function is an interval function that computes an **outer approximation** of the range of the function over a domain

Natural interval extension (2)

F the **natural** interval extension of a real function f is obtained by replacing:

- Each constant k by its natural interval extension $\tilde{k}$
- Each variable by a variable over the intervals
- Each mathematical operator in f by its **optimal** interval extension

Optimal extensions for basic operators:

- $[a,b] \ominus [c,d] = [a-d, b-c]$
- $[a,b] \oplus [c,d] = [a+c, b+d]$
- $[a,b] \otimes [c,d] =$
  $[\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)]$
- $[a,b] \oslash [c,d] = [\min(\frac{a}{c}, \frac{a}{d}, \frac{b}{c}, \frac{b}{d}), \max(\frac{a}{c}, \frac{a}{d}, \frac{b}{c}, \frac{b}{d})]$
  if $0 \notin [c,d]$
  otherwise $\rightarrow [-\infty, +\infty]$

Natural interval extension: Example

Let $f = (x + y) - (y \times x)$ be a real function

Let be $X = [-2, 3]$, $Y = [-9, 1]$

$$
\begin{aligned}
F &= (X \oplus Y) \ominus (Y \otimes X) \\
&= ([-2, 3] \oplus [-9, 1]) \ominus ([-9, 1] \otimes [-2, 3]) \\
&= [-11, 4] \ominus \\
&\qquad [\min(18, -27, -2, 3), \max(18, -27, -2, 3)] \\
&= [-11, 4] \ominus [-27, 18] \\
&= [-29, 31]
\end{aligned}
$$

Interval extension: properties

- If $\mathbf{0} \notin \mathbf{F}(\mathbf{X})$, then no value exists in the box $\mathbf{X}$ such that $f(X) = 0$ $\longrightarrow$
  Equation $f(x)$ does not have any root in the box X
- Interval arithmetic can be implemented taking into account **round-off errors**
- No monotonicity but interval arithmetic preserves **inclusion monotonicity**: $Y \subseteq X \quad \Rightarrow \quad F(Y) \subseteq F(X)$
- No distributivity but interval arithmetic is **sub-distributive**: $X(Y + X) \subseteq XY + XZ$

Why is the image of an interval function not optimal ?

- **Outward Rounding** (required for safe computations with floating point numbers)
- **Non continuity** of interval functions: the image of an interval is in general not an interval
  $\rightarrow$ The **wrapping effect**, which overestimates by a unique vector the image of an interval vector
  Example:
  $$f(x) = \frac{1}{x} \text{ with } X = [-1, 1]$$
  $$F([-1,1]) = \frac{1}{[-1,1]} = [-\infty, -1] \cup [1, +\infty]$$
  $$\rightarrow = [-\infty, +\infty]$$
- **Dependency problem** when a variable has multiple occurrences

Interval extension: dependency problem

- The **dependency problem**, which is due to the **independence** of the different occurrences of a variable during the interval evaluation of an expression

- **Example:**

  Consider $X = [0, 5]$
  $X - X = [0 - 5, 5 - 0] = [-5, 5]$ instead of $[0, 0]$!
  $X^2 - X = [0, 25] - [0, 5] = [-5, 25]$
  $X(X - 1) = [0, 5]([0, 5] - [1, 1]) = [0, 5][-1, 4] = [-5, 20]$

Interval arithmetic: notations

$c_j(x_1, \ldots, x_n)$	A relation over the real numbers
x, y	Real variables or vectors
X or $\mathbf{x}$ or D_x	The domain of variable x (i.e. intervals)
$X = [\underline{X}, \overline{X}]$	The set of real numbers x verifying
$\mathbf{x} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$	$\underline{X} \leq x \leq \overline{X}$ (resp. $\underline{\mathbf{x}} \leq x \leq \overline{\mathbf{x}}$)
$\mathcal{C}$	The set of constraints
$\mathcal{D}$	The set of domains of all the variables
$\mathcal{R}$	The set of real numbers
$\mathcal{R}^{\infty}$	$\mathcal{R} \cup \{-\infty, +\infty\}$, set of real numbers extended with infinity symbols
$\mathcal{F}$	The set of floating point numbers
a^{+} (resp. a^{-})	The smallest (resp. largest) number of $\mathcal{F}$ strictly greater (resp. lower) than a
$\tilde{k}$	smallest interval containing real number k
$\Phi_{cstc}(P)$	closure (filtering) by consistency of CSP P; $cstc$ stands for $2B, Box, 3B, Bound$

Interval arithmetic: basic definitions (1)

Interval arithmetic (Moore-1966)
is based on the representation of variables as intervals

Let f be a real-valued function of n unknowns $\{x_1, \ldots, x_n\}$, an **interval evaluation** F of f for given ranges $\mathbf{X} = \{X_1, \ldots, X_n\}$ for the unknowns is an interval Y such that

$$\forall \{v_1, \ldots, v_n\} \in \{X_1, \ldots, X_n\} : \ \underline{Y} \leq f(v_1, \ldots, v_n) \leq \overline{Y}$$

$\underline{Y}, \overline{Y}$: lower and upper bounds for the values of f when the values of the unknowns are restricted to the box $\mathbf{X}$

Interval arithmetic: basic definitions (2)

Let $C : \mathcal{I}^n \rightarrow \mathcal{B}ool$ be a **relation** over the intervals

C is an **interval extension** of the relation $c : \mathcal{R}^n \rightarrow \mathcal{B}ool$ iff:

$$\forall X_1, \ldots, X_n \in \mathcal{I} : \exists v_1 \in X_1 \wedge \ldots \wedge \exists v_n \in X_n \wedge c(v_1, \ldots, v_n) \Rightarrow C(X_1, \ldots, X_n)$$

For instance, $X_1 \doteq X_2 \Leftrightarrow (X_1 \cap X_2) \neq \emptyset$ is an interval extension of the relation $x_1 = x_2$ over the real numbers

Example:

Relation $X_1 \doteq X_2$ holds if $X_1 = [0, 17.5]$ and $X_2 = [17, 27.5]$

Natural interval extension (1)

- In general, it is not possible to compute **the exact enclosure** of the range for an arbitrary function over the real numbers

  $\longrightarrow$ The interval extension of a function is an interval function that computes an **outer approximation** of the range of the function over a domain

Natural interval extension (1)

F the **natural** interval extension of a real function f is obtained by replacing:

- Each constant k by its natural interval extension $\tilde{k}$
- Each variable by a variable over the intervals
- Each mathematical operator in f by its **optimal** interval extension

Optimal extensions for basic operators:

- $[a, b] \ominus [c, d] = [a - d, b - c]$
- $[a, b] \oplus [c, d] = [a + c, b + d]$
- $[a, b] \otimes [c, d] =$
  $[\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)]$
- $[a, b] \oslash [c, d] = [\min(\frac{a}{c}, \frac{a}{d}, \frac{b}{c}, \frac{b}{d}), \max(\frac{a}{c}, \frac{a}{d}, \frac{b}{c}, \frac{b}{d})]$
  if $0 \notin [c, d]$
  otherwise $\rightarrow [-\infty, +\infty]$

Natural interval extension: Example

Let $f = (x + y) - (y \times x)$ be a real function

Let be $X = [-2, 3]$, $Y = [-9, 1]$

$$
\begin{aligned}
F &= (X \oplus Y) \ominus (Y \otimes X) \\
&= ([-2, 3] \oplus [-9, 1]) \ominus ([-9, 1] \otimes [-2, 3]) \\
&= [-11, 4] \ominus \\
&\qquad [\min(18, -27, -2, 3), \max(18, -27, -2, 3)] \\
&= [-11, 4] \ominus [-27, 18] \\
&= [-29, 31]
\end{aligned}
$$

Interval extension: properties

- If $\mathbf{0} \notin \mathbf{F}(\mathbf{X})$, then no value exists in the box $\mathbf{X}$ such that $f(X) = 0$ $\longrightarrow$
  Equation $f(x)$ does not have any root in the box X
- Interval arithmetic can be implemented taking into account **round-off errors**
- No monotonicity but interval arithmetic preserves **inclusion monotonicity**: $Y \subseteq X \quad \Rightarrow \quad F(Y) \subseteq F(X)$
- No distributivity but interval arithmetic is **sub-distributive**: $X(Y + X) \subseteq XY + XZ$

Why is the image of an interval function not optimal ?

- **Outward Rounding** (required for safe computations with floating point numbers)
- **Non continuity** of interval functions: the image of an interval is in general not an interval
  $\rightarrow$ The **wrapping effect**, which overestimates by a unique vector the image of an interval vector
  Example:
  $$f(x) = \frac{1}{x} \text{ with } X = [-1, 1]$$
  $$F([-1, 1]) = \frac{1}{[-1,1]} = [-\infty, -1] \cup [1, +\infty]$$
  $$\rightarrow = [-\infty, +\infty]$$
- **Dependency problem** when a variable has multiple occurrences

Interval extension: dependency problem

- The **dependency problem**, which is due to the **independence** of the different occurrences of a variable during the interval evaluation of an expression

- **Example:**

  Consider $X = [0, 5]$
  $X - X = [0 - 5, 5 - 0] = [-5, 5]$ instead of $[0, 0]$!
  $X^2 - X = [0, 25] - [0, 5] = [-5, 25]$
  $X(X - 1) = [0, 5]([0, 5] - [1, 1]) = [0, 5][-1, 4] = [-5, 20]$

Interval extension: using different literal forms (1)

- **Factorized form** (Horner for polynomial system) or distributed form

- **First-order Taylor development** of f

$$F_{\text{tay}}(X) = f(x) + J(X).(X - x)$$

with $\forall x \in X$, $J()$ being the Jacobian of f

Interval extension: using different literal forms (2)

- In general, first order Taylor extensions yield a better enclosure than the natural extension on **small intervals**
- Taylor extensions have a **quadratic convergence** whereas the natural extension has a linear convergence
- In general, neither F_{nat} nor F_{tay} won't allow to compute the exact range of a function f

Interval extension: using different literal forms (3)

Consider $f(x) = 1 - x + x^2$, and $X = [0, 2]$

$$f_{\text{tay}}([0,2]) = f(x) + (2X - 1)(X - x) = f(1) + (2[0,2] - 1)([0,2] - 1) = [-\mathbf{2}, \mathbf{4}]$$

$$f_{\text{nat}}([0,2]) = 1 - X + X^2 = [1,1] - [0,2] + [0,2]^2 = [-\mathbf{1}, \mathbf{5}]$$

$$f_{\text{factor}}([0,2]) = 1 + X(X - 1) = [1,1] + [0,2]([0,2] - [1,1]) = [-\mathbf{1}, \mathbf{3}]$$

whereas the range of f over $X = [0, 2]$ is $[0.75, 3]$

Interval Analysis methods

Goal: to determine the zeros of a system of n equations $f_i(x_1, \ldots, x_n)$ in n unknowns x_i inside the interval vector

$$X = \{X_1, \ldots, X_n\} \text{ with } x_i \in X_i \text{ for } i = 1, \ldots, n$$

- **Gauss-Seidel iterative method**
- **Interval Newton algorithm**

Gauss-Seidel iterative method

Consider the case of interval linear equations:

$$\mathbf{A}.x = \mathbf{b}$$

with $\mathbf{A}$ an interval matrix and $\mathbf{b}$ an interval vector
For each unknowns X_i, the Gauss-Seidel algorithm is defined by the following iterative process:

$$X_i^{k+1} \leftarrow [(\mathbf{b}_i - \sum_{j=1}^{i-1} \mathbf{A}_{i,j} X_j^{k+1} - \sum_{j=i+1}^{n} \mathbf{A}_{i,j} X_j^{k}) / \mathbf{A}_{i,i}] \cap X_i^k$$

Pre-conditioning $\rightarrow$ *to shrink the width* of the intervals

Interval Newton algorithm (1)

Principle of the Newton operator:

Consider $f : \mathcal{R} \to \mathcal{R}$, the mean value theorem says:
$\exists a \in [v, u] : f(u) - f(v) = (u - v)f'(a)$ and thus,

$$v = u - \frac{f(u)}{f'(a)} \text{ if } v \text{ is a zero of } f$$

If $a \in I$ then $f(a) \in F(I)$, and $v \in \tilde{u} - \frac{F(\tilde{u})}{F'(I)} = N(F, F', \tilde{u}, I)$
If v is a zero of f then $v \in I_n \quad (n \geq 1)$ where

$$I_0 = I$$
$$I_{i+1} = N(F, F', center(I_i), I) \cap I_i$$
$$\dots$$
$$I_n = I_{n+1}$$

Interval Newton algorithm (2)

The **Interval Newton algorithm** is used to solve **non-linear systems** with

$$X_{k+1} = N(\tilde{x}_k, X_k) \cap X_k \text{ with } N(\tilde{x}_k, X_k) = \tilde{x}_k - A.f(\tilde{x}_k)$$

where $A = [F^{'}(X_k)]^{-1}$
and $\tilde{x}_k \in X_k$ (e.g., the mid-point of X_k)

Properties:

- If $N(\tilde{x}_k, X_k) \cap X_k = \emptyset$, then the system F **does not have any solution** in X_k
- if $N(\tilde{x}_k, X_k)_k \subset X_k$, there exists **at least one solution** in X_{k+1}

Matrix $A = [F^{'}(X_k)]^{-1}$ may be costly to compute
... to determine $N(\tilde{x}_k, X_k) \rightarrow$ solve the linear system:

$$F^{'}(X_k)(N(\tilde{x}_k, X_k) - \tilde{x}_k) = -f(\tilde{x}_k)$$

Constraint Programming

Numeric CSP $(\mathcal{X}, \mathcal{D}, \mathcal{C})$**:**

- $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of variables
- $\mathcal{D} = \{D_{x_1}, \ldots, D_{x_n}\}$ is a set of domains
  (D_{x_i} contains all acceptable values for variable x_i)
- $\mathcal{C} = \{c_1, \ldots, c_m\}$ is a set of constraints

Overall scheme

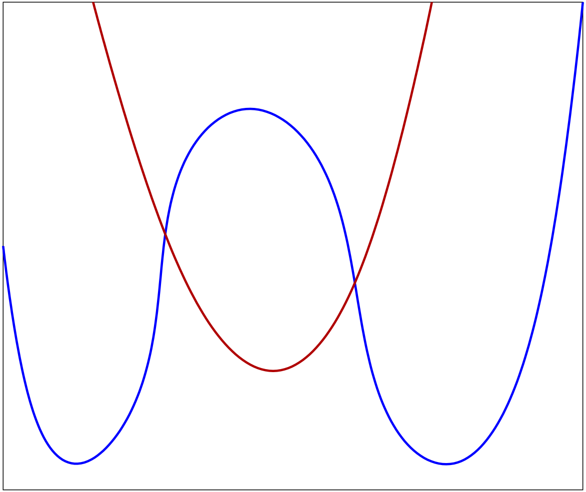
The constraint programming framework is based on a **branch & prune** schema which is best viewed as an iteration of two steps:

1. **Pruning the search space**
2. **Making a choice to generate two (or more) sub-problems**

- The pruning step $\longrightarrow$ **reduces an interval** when it can prove that the upper bound or the lower bound does not satisfy some constraint
- The branching step $\longrightarrow$ **splits the interval** associated to some variable in two intervals (often with the same width)

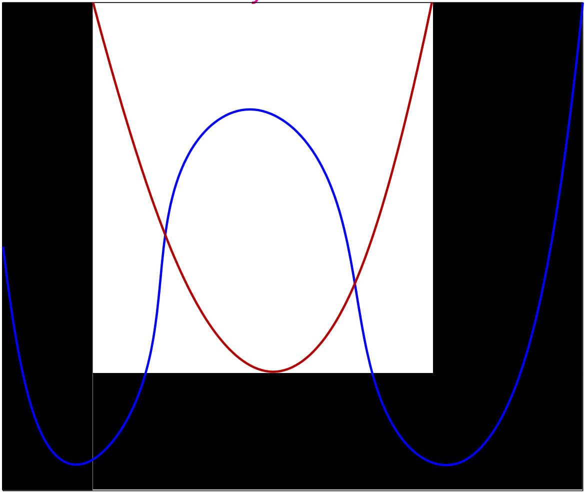
Filtering & Solving process (example)

Courtesy to Gilles Trombettoni



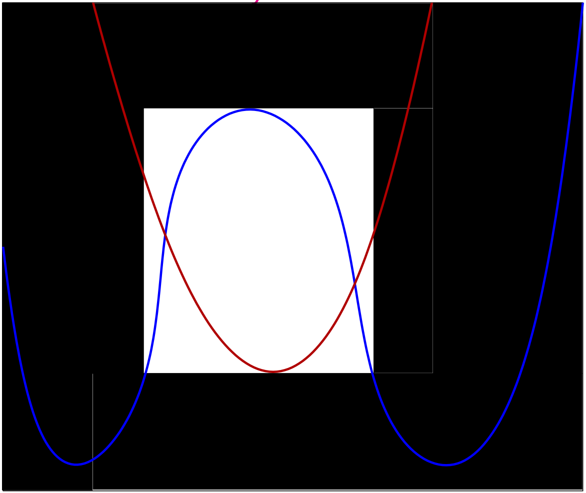
Filtering & Solving process (example)

Courtesy to Gilles Trombettoni



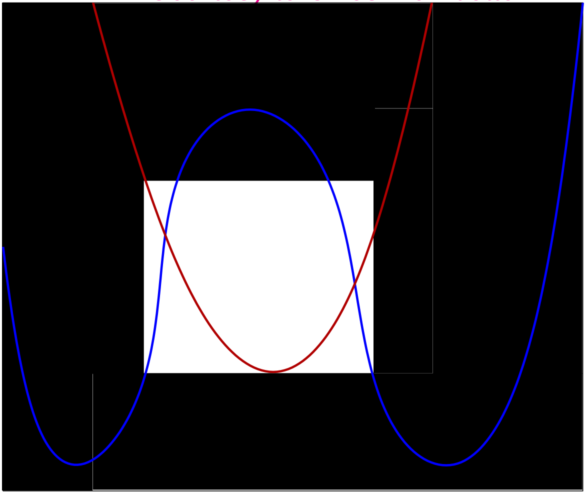
Filtering & Solving process (example)

Courtesy to Gilles Trombettoni



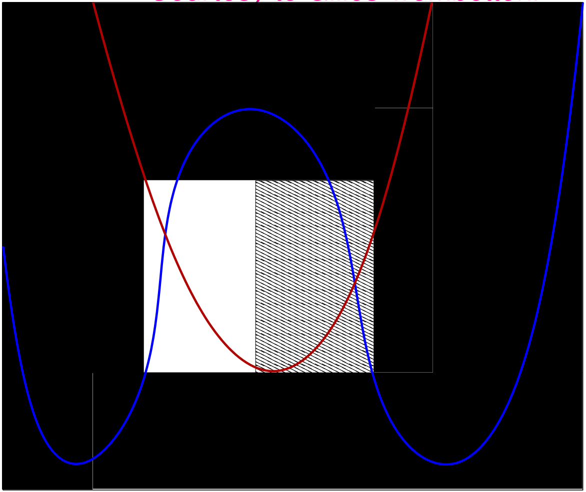
Filtering & Solving process (example)

Courtesy to Gilles Trombettoni



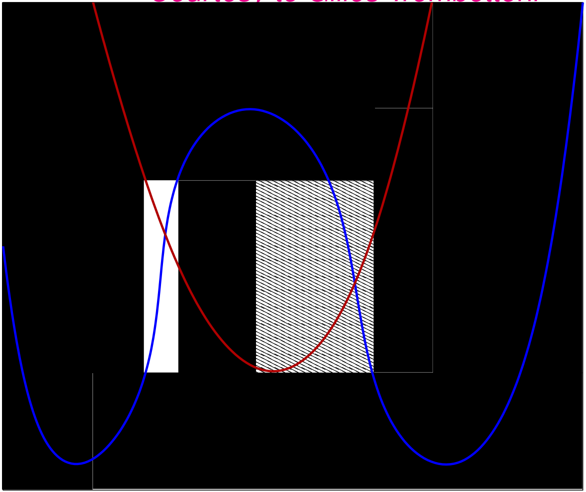
Filtering & Solving process (example)

Courtesy to Gilles Trombettoni



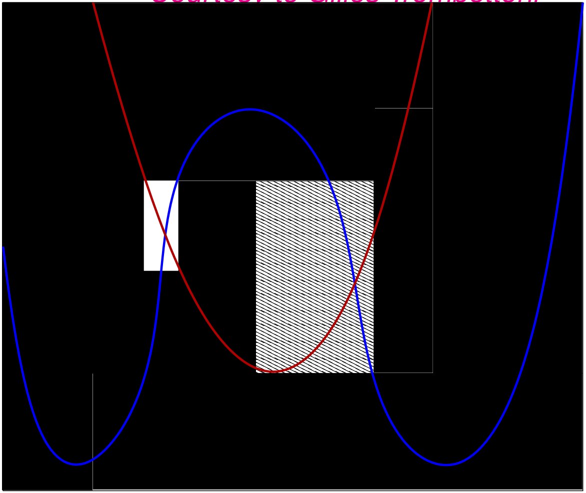
Filtering & Solving process (example)

Courtesy to Gilles Trombettoni



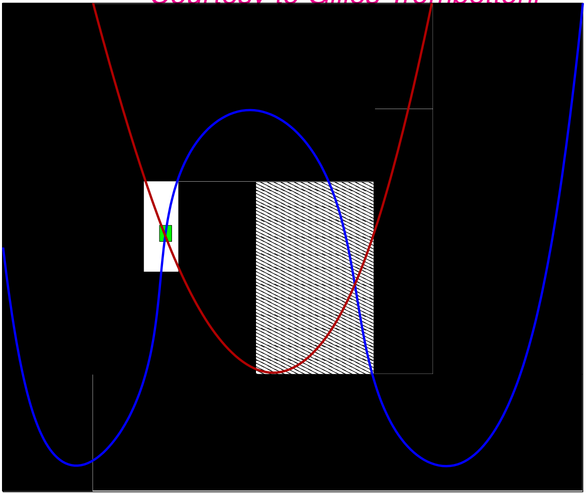
Filtering & Solving process (example)

Courtesy to Gilles Trombettoni



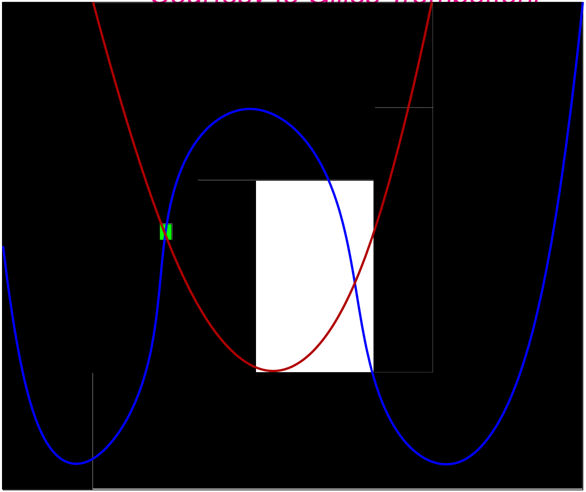
Filtering & Solving process (example)

Courtesy to Gilles Trombettoni



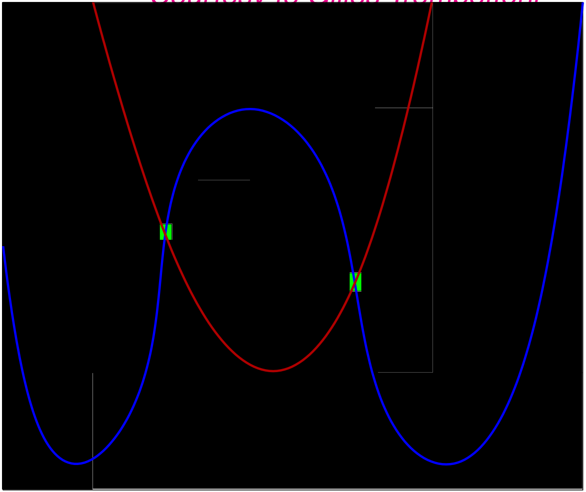
Filtering & Solving process (example)

Courtesy to Gilles Trombettoni



Filtering & Solving process (example)

Courtesy to Gilles Trombettoni



Local consistencies (1)

- Informally speaking, a constraint system C satisfies a partial consistency property if **a relaxation of C is consistent**

- Consider $X = [\underline{x}, \overline{x}]$ and $C(x, x_1, \ldots, x_n) \in \mathcal{C}$: if $C(x, x_1, \ldots, x_n)$ does not hold for any values $a \in [\underline{x}, x']$, then X may be shrunken to $X = [x', \overline{x}]$

Local consistencies (2)

- A **constraint C_j is *AC*-like-consistent** if for any variable x_i in $\mathcal{X}_j$, **the bounds $\underline{D_i}$ and $\overline{D_i}$ have a support** in the domains of all other variables of $\mathcal{X}_j$
- *AC*-like local consistencies are used in `BNR-prolog`, `Interlog`, `CLP(BNR)`, `PrologIV`, `UniCalc`, `Ilog Solver`, `Numerica`, `Icos`, `RealPaver`, `IBEX`,...

Local consistencies (3)

Local consistencies are conditions that filtering algorithms must satisfy

$\longrightarrow$ fixed point algorithm defined by the sequence $\{\mathcal{D}_k\}$ of domains generated by the iterative application of an operator

$$Op : I(R)^n \longrightarrow I(R)^n$$

$$\boxed{\mathcal{D}_k = \begin{cases} \mathcal{D} & \text{if } k = 0 \\ Op(\mathcal{D}_{k-1}) & \text{if } k > 0 \end{cases}}$$

Local consistencies (4)

Properties of the operator *Op*:

- $Op(\mathcal{D}) \subseteq \mathcal{D}$ (inclusion)
- *Op* is conservative; that is, it cannot remove any solution
- $\mathcal{D}' \subseteq \mathcal{D} \Rightarrow Op(\mathcal{D}') \subseteq Op(\mathcal{D})$ (monotonicity)

The limit of the sequence $\{\mathcal{D}_k\}$ corresponds to the greatest fixed point of the operator *Op*

Local consistencies (5)

- **2B–consistency** (also known as hull consistency) only requires to check the Arc–Consistency property **for each bound** of the intervals
- **Box–consistency** is a coarser relaxation of Arc–Consistency than 2B–consistency ... but Box–consistency algorithms **may achieve a stronger filtering than 2B–consistency**
- **KB–consistency**... used when no bound of the domains can be removed with a local consistency filtering algorithm
- **Implementation issues are critical** → HC4-Revise, Mohc-Revise

2B–consistency (1)

Variable x is 2B–consistency for constraint $f(x, x_1, \ldots, x_n) = 0$ if the lower (resp. upper) bound of the domain X is the smallest (resp. largest) solution of $f(x, x_1, \ldots, x_n)$

Definition: 2B–consistency

Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a CSP and $C \in \mathcal{C}$ a k-ary constraint over $(X_1, \ldots, X_k)$

C is 2B–consistency iff :

$$\forall i, X_i = \Box\{\tilde{x}_i \,|\, \exists \tilde{x}_1 \in X_1, \ldots, \exists \tilde{x}_{i-1} \in X_{i-1}, \exists \tilde{x}_{i+1} \in X_{i+1}, \ldots, \exists \tilde{x}_k \in X_k \;:\; c(\tilde{x}_1, \ldots, \tilde{x}_{i-1}, \tilde{x}_i, \tilde{x}_{i+1} \ldots, \tilde{x}_k)\}$$

A CSP is 2B–consistent iff all its constraints are consistent

Box–consistency (1)

Variable x is Box–Consistent for constraint $f(x, x_1, \ldots, x_n) = 0$ if the bounds of the domain of x correspond to the **leftmost** and the **rightmost** **zero** of the **optimal interval extension** $F(X, X_1, \ldots, X_n)$ of $f(x, x_1, \ldots, x_n)$

Definition: Box–consistency

Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$be a CSP and $C \in \mathcal{C}$ a k-ary constraint over $(X_1, \ldots, X_k)$

C is Box–Consistent if, for all X_i the following relations hold :

1. $C(X_1, \ldots, X_{i-1}, [\underline{X_i}, \underline{X_i}^+), X_{i+1}, \ldots, X_k)$
2. $C(X_1, \ldots, X_{i-1}, (\overline{X_i}^-, \overline{X_i}], X_{i+1}, \ldots, X_k)$

Local consistency filtering (1)

Algorithms that achieve a local consistency filtering are based upon projection functions

- **Solution functions** express the variable x_i in terms of the other variables of the constraint. The solution functions of $x + y = z$ are:
  $f_x = z - y, f_y = z - x, f_z = x + y$
- An approximation of the projection of the constraint over X_i given a domain $\mathcal{D}$ can be computed with any interval extension of this solution function $\rightarrow$ we have a way to compute $\pi_{j,i}(\mathcal{D})$
- For complex constraints, **no analytic solution function may exist**
  Example: $x + log(x) = 0$

Local consistency filtering (2)

Analytic functions always exist when the variable to express in terms of the others appears only once in the constraint

$\rightarrow$ **Considers that each occurrence is a different new variable**

For $x + log(x) = 0$ we obtain $x_1 + \log(x_2) = 0$
Thus $f_{x_1} = -\log(x_2)$, $f_{x_2} = \exp^{-x_1}$
and $\pi_{x+log(x)=0,x}(X) = -\log(X) \cap \exp^{-X}$

- This approach is used for computing **2B–consistency filtering** (the initial constraints are decomposed into primitive constraints)
- Decomposition does not change the semantics of the initial constraints system but **it amplifies the dependency problem**

Local consistency filtering (3)

Transformation of the constraint $C_j(x_{j_1}, ...x_{j_k})$ into k **mono-variable constraints** by substituting all variables but one by their intervals

- The two extremal zeros of $C_{j,l}$ can be found by a **dichotomy algorithm** combined with a mono-variable version of the **interval Newton method**
- This approach is **well adapted for Box–consistency filtering**

Local consistency filtering (4)

Use the Taylor extension to transform the constraint into an interval linear constraint

- Equation $f(X) = 0$ becomes an interval linear equation in X, which does not contain multiple occurrences
- Solving the **squared interval linear system** allows much more **precise approximations** of projections to be computed

3B–Consistency (1)

3B–Consistency, a relaxation of path consistency

$\longrightarrow$

checks whether 2B–Consistency can be enforced when the domain of a variable is **reduced to the value of one of its bounds** in the whole system

3B–Consistency (2)

Definition: 3B–Consistency

Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a CSP and x a variable of $\mathcal{X}$ with $D_x = [a, b]$.

Let also:

- Let $P_{D_x^1 \leftarrow [a, a^+)}$ be the CSP derived from P by substituting D_x in $\mathcal{D}$ with $D_x^1 = [a, a^+)$
- Let $P_{D_x^2 \leftarrow (b^-, b]}$ be the CSP derived from P by substituting D_x in $\mathcal{D}$ with $D_x^2 = (b^-, b]$

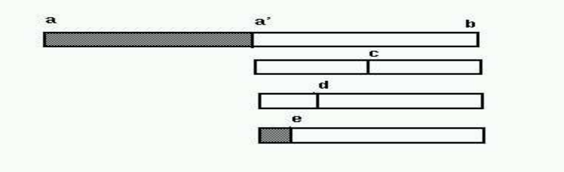
X is 3B–Consistent iff

$$\Phi_{2B}(P_{D_x^1 \leftarrow [a, a^+)}) \neq P_\emptyset \text{ and } \Phi_{2B}(P_{D_x^2 \leftarrow (b^-, b]}) \neq P_\emptyset$$

3B–Consistency (3)

Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a CSP and $D_x = [a, b]$, if $\Phi_{2B}(P_{D_x \leftarrow [a, \frac{a+b}{2}]}) = \emptyset$

- then the part $[a, \frac{a+b}{2})$ of D_x will be removed and the filtering process continues on the interval $[\frac{a+b}{2}, b]$
- otherwise, the filtering process continues on the interval $[a, \frac{3a+b}{4}]$.



Global Constraints (1)

- **"Syntactical"** approach

- **"Semantic"** approach

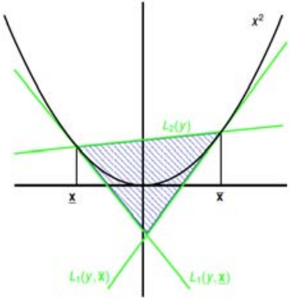
Global Constraints (1)

- **A global constraint** to handle a tight approximation of the constraint system with an LP solver
- ***Combines***
  - **safe and rigorous** linear relaxations
  - **local consistencies** and **interval methods**

Linearisation of x^2

Example: relaxation of $\mathbf{x}^2$ with $\mathbf{x} \in [-4, 5]$

- Introduce a **new variable**
  - Replace $\mathbf{x}^2$ by $\mathbf{y}$
  - Domain of $\mathbf{y}$: $[0, 25]$
- Add **redudant** constraints



- $y \geq 2\alpha x - \alpha^2$ with $\alpha \in [-4, 5]$

  $\mathbf{y} \geq -\mathbf{8x} - \mathbf{16}$
  $\mathbf{y} \geq \mathbf{10x} - \mathbf{25}$

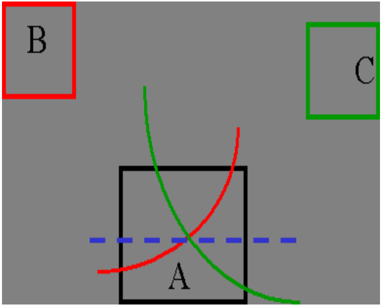
- $y \leq (\underline{x} + \overline{x})x - \underline{x} * \overline{x}$

  $\mathbf{y} \leq \mathbf{x} + \mathbf{20}$

Global Constraints (2)

“Semantic” approach

→ **Distance constraint**



Applications

- **Safe global Optimisation**
- **Program verification**

Global Numerical Optimization: problem

We consider the continuous global optimisation problem

$$\mathcal{P} \equiv \begin{cases} \min & f(x) \\ \text{s.c.} & g_i(x) = 0,\ j = 1..k \\ & g_j(x) \leq 0,\ j = k+1..m \\ & \underline{\mathbf{x}} \leq x \leq \overline{\mathbf{x}} \end{cases}$$

with

- $\mathbf{X} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$: a vector of intervals of R
- $f : R^n \rightarrow R$ and $g_j : R^n \rightarrow R$
- Functions f and g_j: are continuously differentiable on $\mathbf{X}$

Trends in global optimisation

- **Performance**

  Most successful systems (Baron, αBB, ...) use local methods and linear relaxations
  $\rightarrow$ **not rigorous** (work with floats)

- **Rigour**

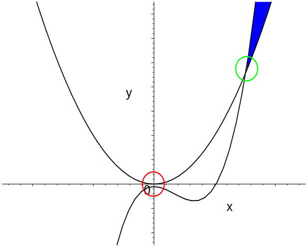
  Mainly rely on interval computation
  ... available systems (e.g., Globsol) are **quite slow**

- **Challenge:** to combine the advantages of both approaches in an **efficient** and **rigorous** global optimisation framework

Example of flaw due to a lack of rigour

Consider the following optimisation problem:

$$\begin{array}{ll} \min & x \\ \text{s. t.} & y - x^2 \geq 0 \\ & y - x^2 * (x-2) + 10^{-5} \leq 0 \\ & x, y \in [-10, +10] \end{array}$$



Baron 6.0 and Baron 7.2 find 0 as the minimum ...

Branch and Bound Algorithm

- **BB Algorithm:**
  While $\mathcal{L} \neq \emptyset$ do   %$\mathcal{L}$ initialized with the input box
  - **Select** a box B from the set of current boxes $\mathcal{L}$
  - **Reduction** (filtering or tightening) of B
  - **Lower bounding** of f in box B
  - **Upper bounding** of f in box B
  - **Update** of $\underline{\mathbf{f}}$ and $\overline{\mathbf{f}}$
  - **Splitting** of B (if not empty)

- **Upper Bounding – Critical issue:** to prove the **existence** of a feasible point in a small box
  $\rightarrow$ Using **CP refutation capabilities**

- **Lower Bounding – Critical issue:** to achieve an **efficient pruning**
  $\rightarrow$ Using Hybrid constraints **to boost safe OBR**

A challenging finite-domain optimization application

Handling software upgradeability problems

- A **critical issue** in modern operating systems
  - → Finding the "best" solution to install, remove or upgrade packages in a given installation.
  - → The complexity of the upgradeability problem itself is **NP complete**
  - → modern OS contain a huge number of packages (often more than **20 000** packages in a Linux distribution)

- **Mancoosi** (European project FP7/2007-2013)
  `http://www.mancoosi.org/`

Solving software upgradeability problems

Computing a final package configuration from an initial one

- A configuration states which package is installed and which package is not installed:
  - **Problem** (in CUDF): list of package descriptions (with their status) & a set of packages to install/remove/upgrade
  - **Final configuration**: list of installed packages (uninstalled packages are not listed)
- **Expected Answer**: **best solution** according to multiple criteria

  Several optimisation criteria have to be considered, e.g., stability, memory efficiency, network efficiency
- Difficult to tackle with CP tools

CP-based Bounded program verification

- **Goals of BMC**
  - Mechanically check **properties** of models
  - Widely used in **hardware and software** verification
  - Automatic generation of **counterexamples**

- **Principles of BMC**
  - **Bounded** program verification: the array lengths, the variable values and the loops are bounded
  - Falsification of a property is checked for a given **bound** $\rightarrow$ **program is unwound k** times,

CP-based BMC

- **Constraint stores** to represent the specification and the program
  → Program is translated in constraints **on the fly**
- Program is **partially correct** if the constraint store implies the post-conditions
- A **list of solvers** tried in sequence (LP, MILP, Boolean, CP)
- Non deterministically exploration of execution paths

Refining AI-based Approximations

Programs are run on the floats but:

- **Specification**, **properties** of programs
  → **Reasoning with real numbers**
- **Programs** are sometimes written with a the semantics of real numbers "in mind"

Abstract Interpretation

- **Differences** between real numbers and floats reveal **problems with floats**

  → **Approximations** over **floats** and over the **real numbers**

Objective & Approach

- **Goal:** **refine the approximations** computed by abstract interpretation for domains of the program variables
- **Method:** Using **local consistencies** over real numbers and floating-point numbers to **"shave"** the domains