Continuous Constraints: An Overview

Martine Ceberio

University of Texas at El Paso

Logic Programming and Computational Logic

New Mexico State University

Outline of the presentation

- *Continuous constraints: definition and solving process*
- *An example of under and over-constrained problems*
- *Important notions*
- *Some research directions*
- *Conclusion*

Outline of the presentation

- Continuous constraints: definition and solving process
- *An example of under and over-constrained problems*
- *Important notions*
- *Some research directions*
- *Conclusion*

Continuous constraints in a nutshell

- Continuous constraints are...

Continuous constraints in a nutshell

- Continuous constraints are... **CONSTRAINTS**

Continuous constraints in a nutshell

- Continuous constraints are... **CONSTRAINTS**
- Continuous constraints define **RELATIONS** between variables

  ⋆ *domains of variables: intervals = continuous ranges of possible values*

  ⋆ *constraints restrict the possible combinations of values = define a subset of the search space*

Continuous constraints in a nutshell

- Continuous constraints are... **CONSTRAINTS**
- Continuous constraints define **RELATIONS** between variables

  ⋆ *domains of variables: intervals = continuous ranges of possible values*

  ⋆ *constraints restrict the possible combinations of values = define a subset of the search space*
- CSP or Constraint systems are defined by:

  ⋆ *a finite set of variables*

  ⋆ *a finite set of domains: continuous ranges of possible values*

  ⋆ *a finite set of continuous constraints*

Continuous constraints in a nutshell

- Continuous constraints are... **CONSTRAINTS**
- Continuous constraints define **RELATIONS** between variables

  ⋆ *domains of variables: intervals = continuous ranges of possible values*

  ⋆ *constraints restrict the possible combinations of values = define a subset of the search space*
- CSP or Constraint systems are defined by:

  ⋆ *a finite set of variables*

  ⋆ *a finite set of domains: continuous ranges of possible values*

  ⋆ *a finite set of continuous constraints*
- A solution of a constraint system is:

  a complete assignment of all the variables, satisfying all constraints at the same time

How to solve continuous constraints?

- Enumeration is not an option...

How to solve continuous constraints?

- Enumeration is not an option...
- Algorithms based on intervals (as detailed later)

How to solve continuous constraints?

- Enumeration is not an option...
- Algorithms based on intervals (as detailed later)
  - ⋆ *Branch and Bound (B&B):*

    http://www-sop.inria.fr/coprin/logiciels/ALIAS/Movie/film_license.mpg
  - ⋆ *More sophisticated consistency algorithms: Box / Hull-consistencies and their combinations*

    result in Branch and Prune algorithms (B&P)

Solving algorithm: a skeleton

Suppose you solve (C,X,D)

```
S ← Initial domain            // S is the store of domains to be visited
Solutions ← ∅
while (S ≠ ∅) {
   take D out of S            // usually D is the first available domain
   D' ← narrow(D,C)           // apply a consistency technique on D
   if (D' ≠ ∅) and (D' is still too large) then
      split(D',D1,D2)         // splitting in halves is not compulsory
      S ← S ∪ {D1, D2}
      else store D' in Solutions
}
return Solutions              // What does Solutions contain?
```

Solving algorithm: narrow(D,C)

Here we look at the details of narrow($D_1 \times \cdots \times D_n, \{c_1, \ldots, c_p\}$)

S $\leftarrow \{c_1, \ldots, c_p\}$ // S is the store of constraints, no duplicates
while (S $\neq \varnothing$) {
take c out of S // usually c is the first available constraint
for all $i \in \{1, \ldots, n\}$ {
$D'_i \leftarrow$ consistency(D_i,c)
// apply a consistency technique on D_i w.r.t. c
if ($D'_i = \varnothing$) then return $\varnothing$
if ($D'_i \neq D_i$) then
S $\leftarrow$ S $\cup \{c_j,\ j \in J\}$
// c_j are the constraints that share variable i with c
}
return $\times_{1 \leqslant i \leqslant n} D'_i$ // What is $\times_{1 \leqslant i \leqslant n} D'_i$?

Recap'

- Continuous constraints: *very similar in definition to discrete constraints*
- Solving algorithms: *quite different to ensure completeness, but similar structures*
- In the following: *discussion of different flavors of constraint solving*

Outline of the presentation

- *Continuous constraints: definitions and solving process*
- *An example of under and over-constrained problems*
- *Important notions*
- *Some research directions*
- *Conclusion*

Outline of the presentation

- *Continuous constraints: definitions and solving process*
- An example of under and over-constrained problems
- *Important notions*
- *Some research directions*
- *Conclusion*

Example (1/3)

$\boxed{\text{Problem to be solved: } y(t) = f(x,t)}$

Example (1/3)

$\boxed{\text{Problem to be solved: } y(t) = f(x,t)}$

$$\Downarrow$$

the radioactive decay of radium

[*Pierre and Marie Curie (1898)*]

$$y(t) = \exp^{-xt}$$

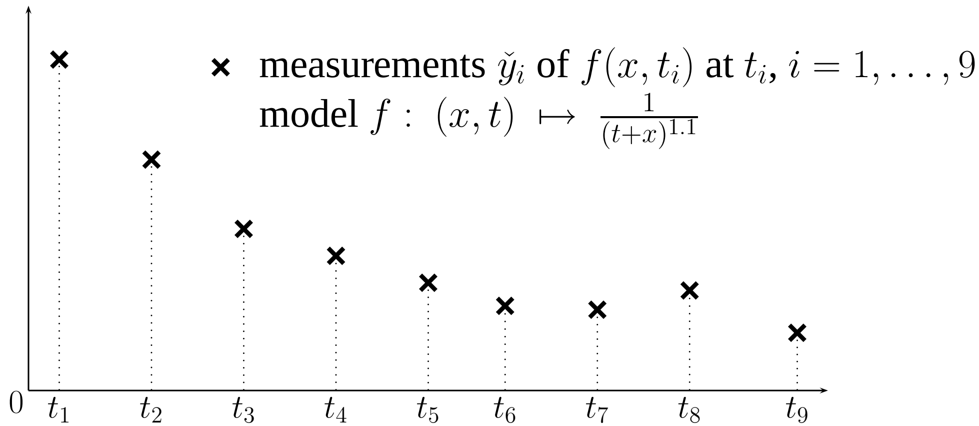
Example (1/3)

$\boxed{\text{Problem to be solved: } y(t) = f(x,t)}$

Knowing: y, t, the model (f)

Given: measurements $\check{y}_i$ of $f(x, t_i)$ at instants t_i

Find: parameter x



Example (1/3)

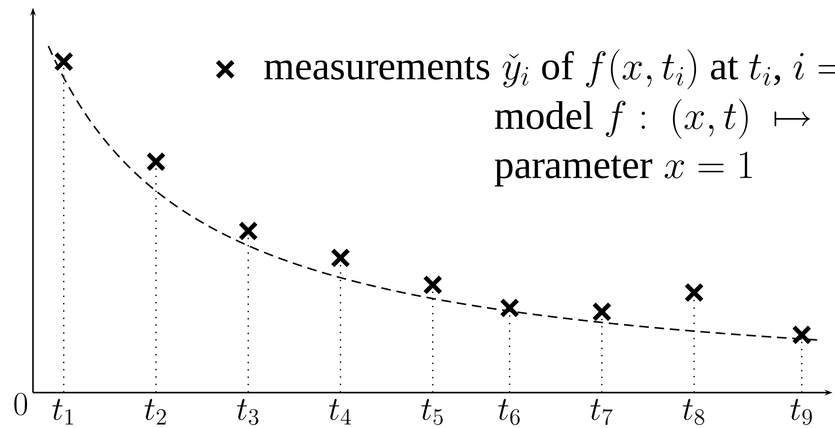
$\boxed{\text{Problem to be solved: } y(t) = f(x,t)}$

Knowing: y, t, the model (f)

Given: measurements $\check{y}_i$ of $f(x,t_i)$ at instants t_i

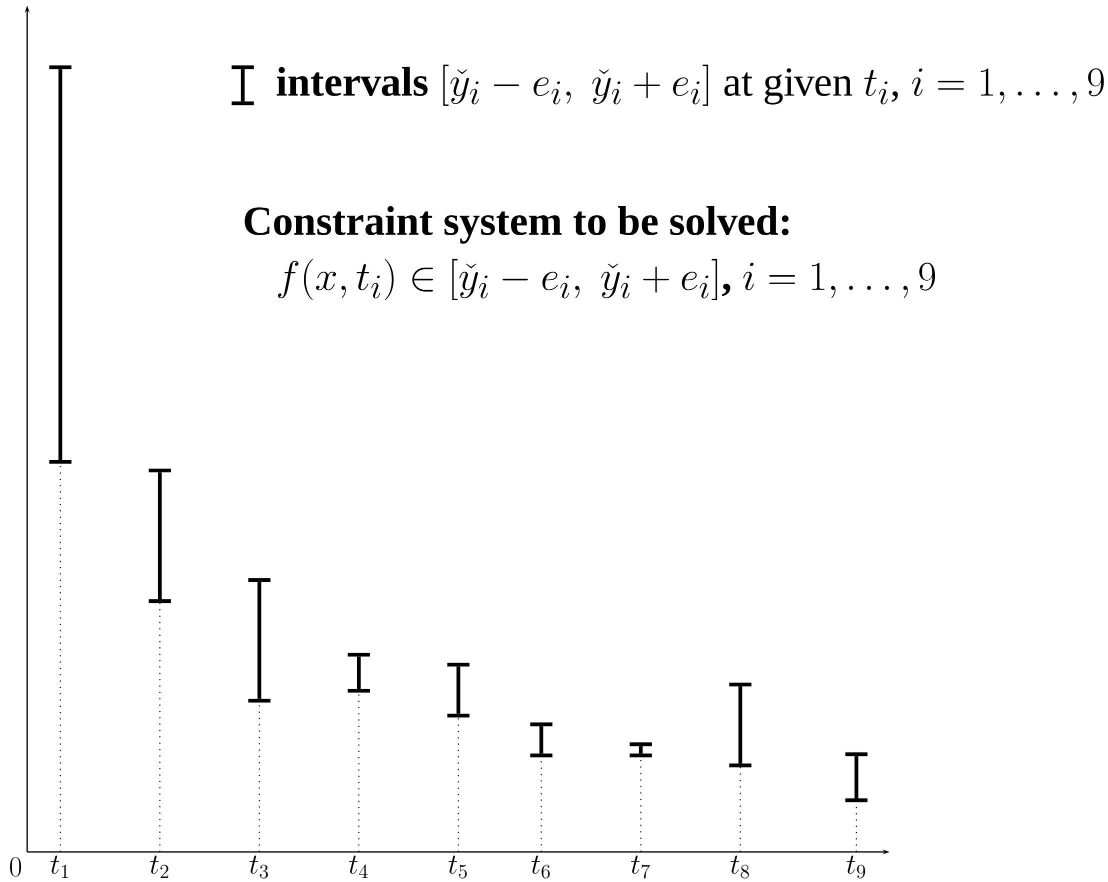
Find: parameter x

Classical solving method: *least squares* $\min_x \sum_{i=1}^{n} (\check{y}_i - f(x,t_i))^2$



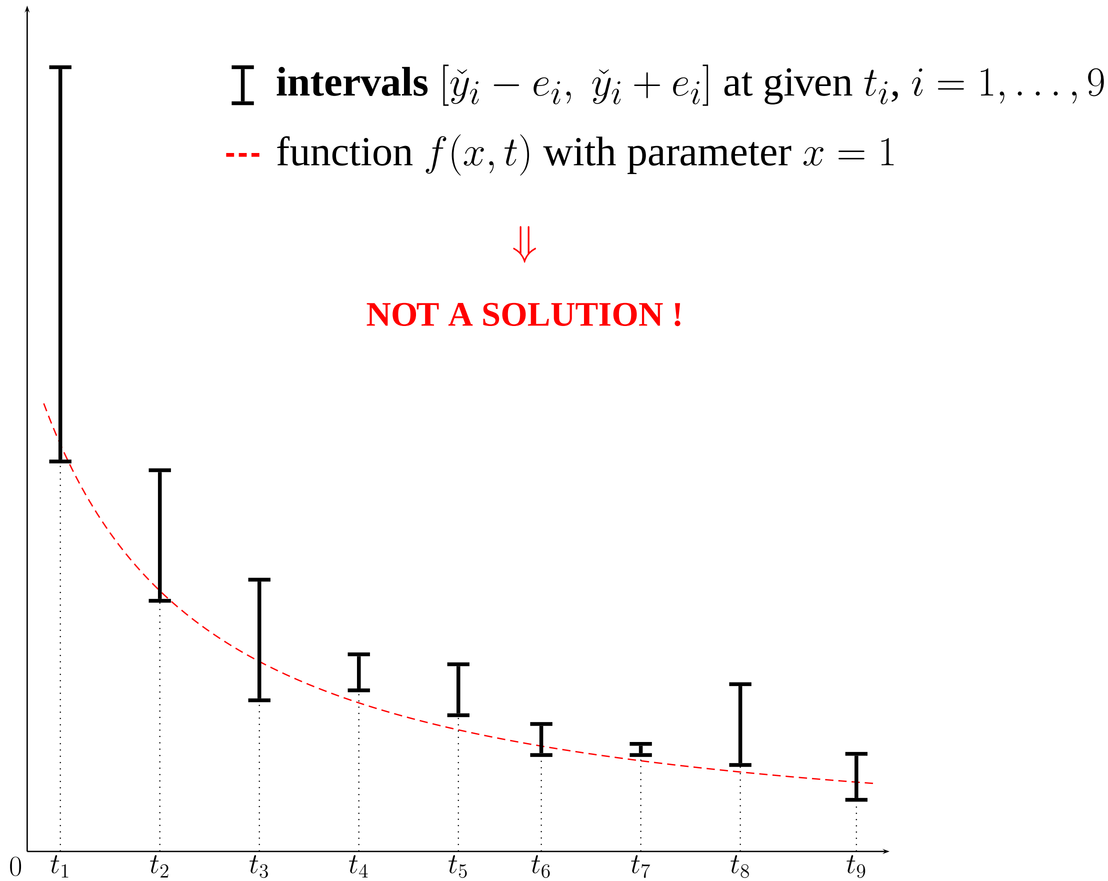
Example (2/3)

Taking inaccuracy into account



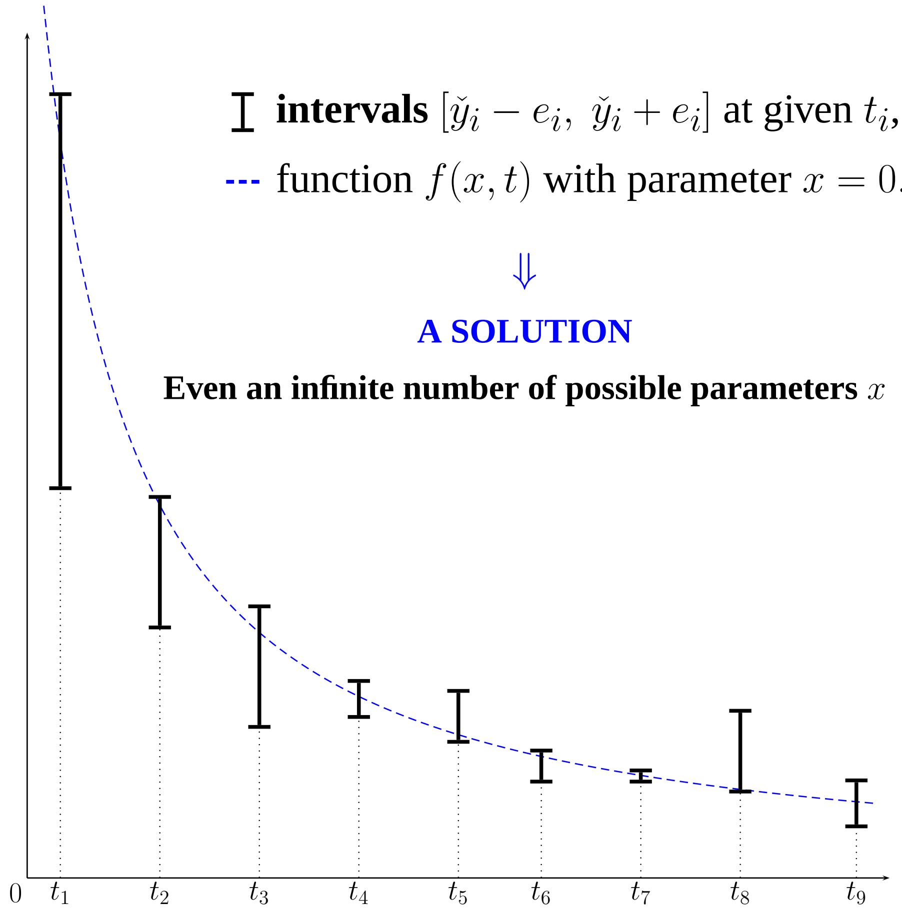
Example (2/3)

Taking inaccuracy into account



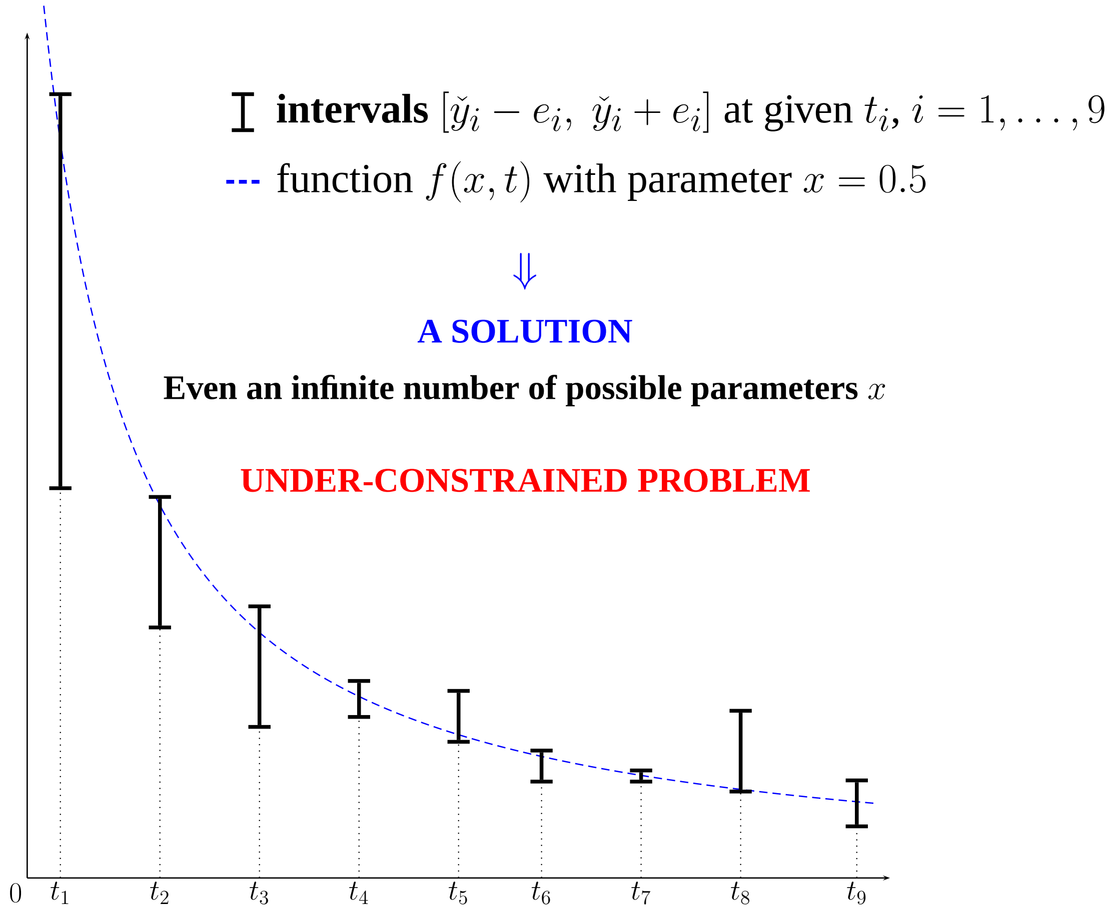
Example (2/3)

Taking inaccuracy into account



Example (2/3)

Taking inaccuracy into account



Example (2/3)

Taking inaccuracy into account

$$\boxed{\textbf{Under-constrained problem}}$$

$$\Downarrow$$

Definition of an appropriate criterion to be optimized

i.e., discrimination over the solution set

Example (2/3)

Taking inaccuracy into account

$\boxed{\textbf{Under-constrained problem}}$

$\Downarrow$

Definition of an appropriate criterion to be optimized

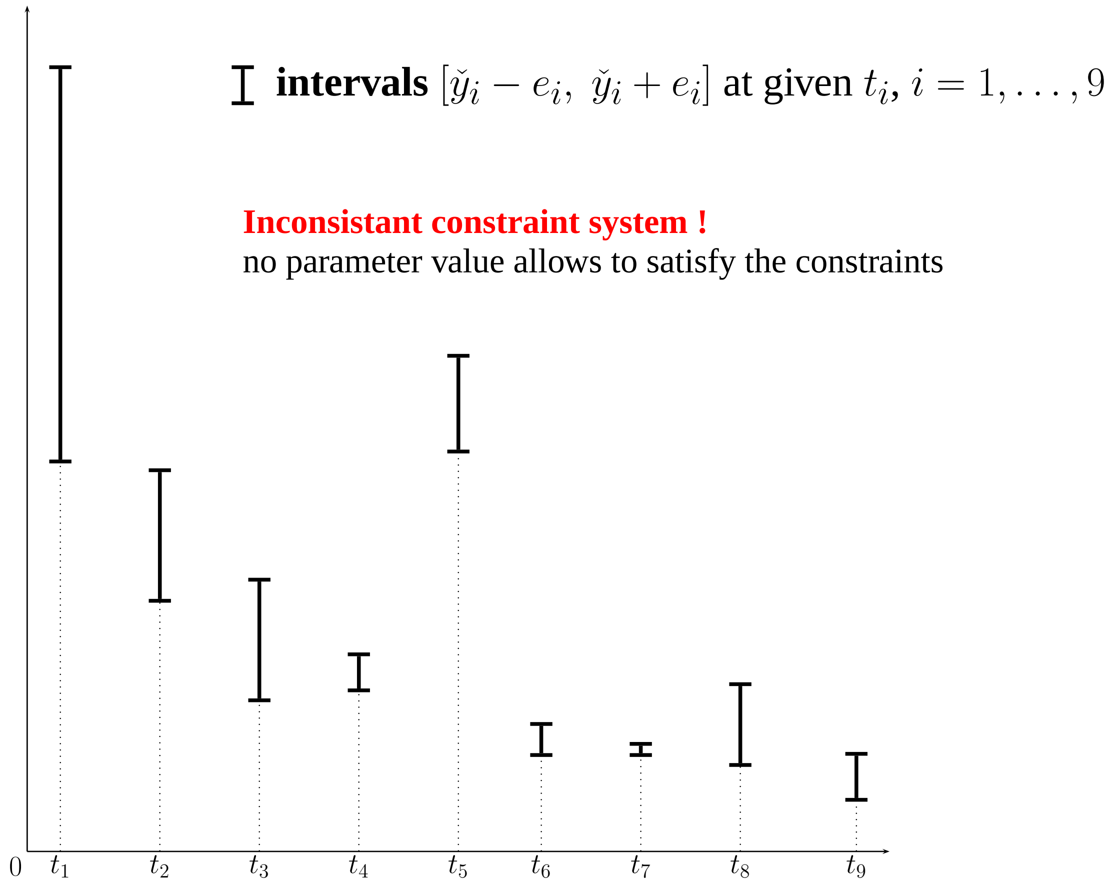
i.e., discrimination over the solution set

$\equiv$

$\boxed{\textbf{Constrained global optimization}}$

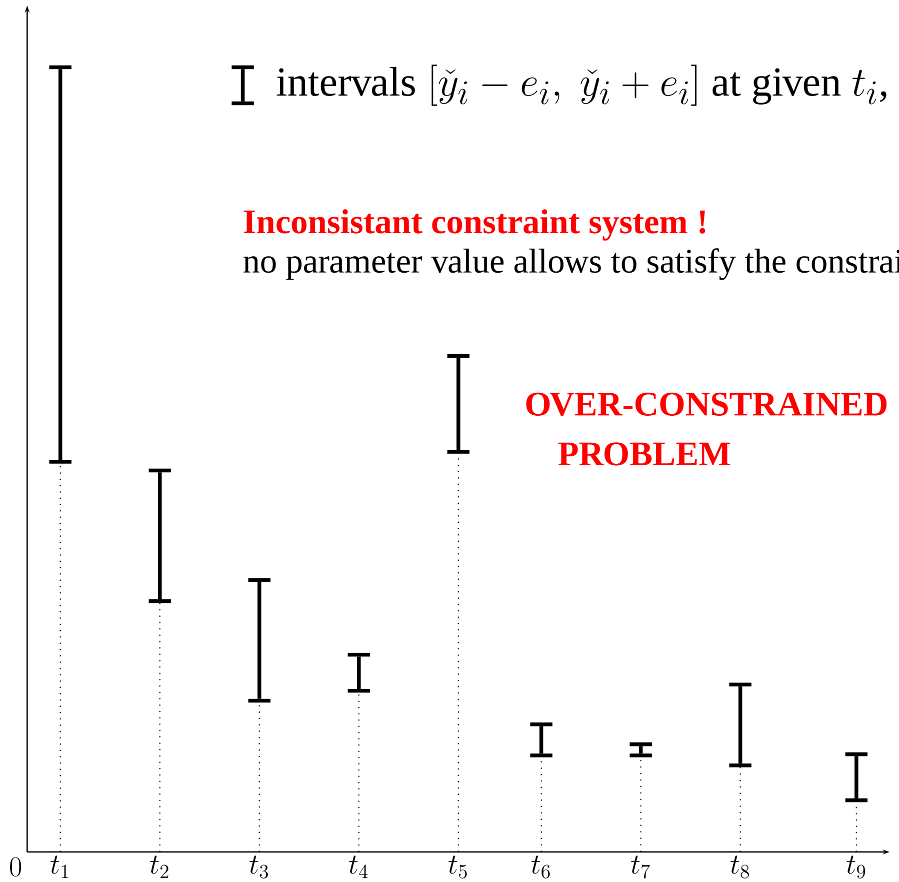
Example (3/3)

Taking erroneous measurements into account



Example (3/3)

Taking erroneous measurements into account



Example (3/3)

Taking erroneous measurements into account

$\boxed{\textbf{Over-constrained problem}}$

$\Downarrow$

Need of solution $\rightsquigarrow$ weaker constraints *i.e.,* need for flexibility

Example (3/3)

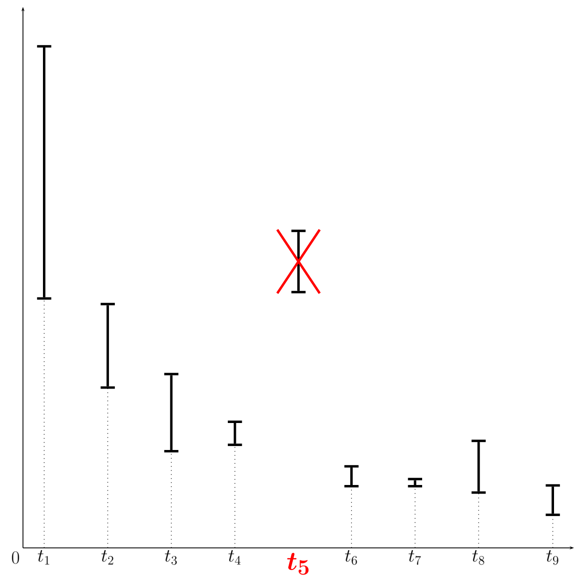
Taking erroneous measurements into account

$\boxed{\textbf{Over-constrained problem}}$

$\Downarrow$

Need of solution $\rightsquigarrow$ weaker constraints *i.e.,* need for flexibility

Ex. deletion of the measure at t_5



Example (3/3)

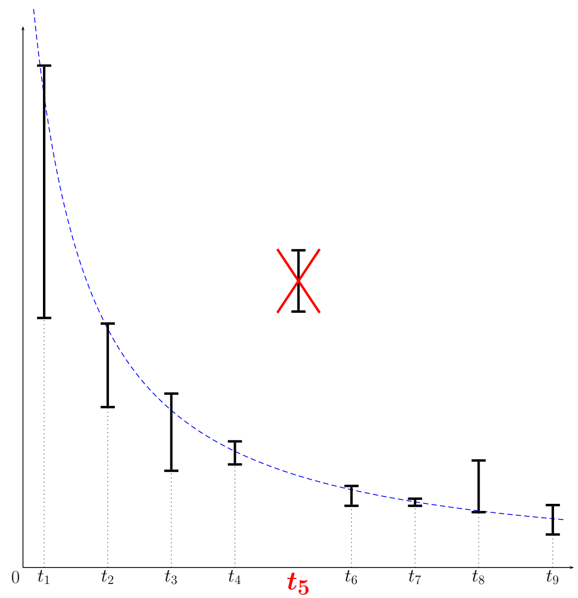
Taking erroneous measurements into account

$\boxed{\textbf{Over-constrained problem}}$

$\Downarrow$

Need of solution $\rightsquigarrow$ weaker constraints *i.e.,* need for flexibility

Ex. deletion of the measure at t_5, *i.e.,* deletion of a constraint



Example (3/3)

Taking erroneous measurements into account

$\boxed{\textbf{Over-constrained problem}}$

$\Downarrow$

Need of solution $\rightsquigarrow$ weaker constraints *i.e.,* need for flexibility

$\equiv$

$\boxed{\textbf{Soft constraints}}$

Outline of the presentation

- *Continuous constraints: definitions and solving process*
- *An example of under and over-constrained problems*
- *Important notions*
- *Some research directions*
- *Conclusion*

Outline of the presentation

- *Continuous constraints: definitions and solving process*
- *An example of under and over-constrained problems*
- Important notions
  - *Intervals*
  - *Global optimization*
  - *Soft constraints*
- *Some research directions*
- *Conclusion*

Important notions

Intervals

Global optimization

Soft constraints

Real intervals

Definition 2 (Real interval [Moore, 1966]). *A real interval $\boldsymbol{x}$ is a closed and connected set of real numbers, noted $[a, b]$.*

$$\boldsymbol{x} = \{x \in \mathbb{R} \mid a \leqslant x \leqslant b\} \qquad \underline{\boldsymbol{x}} = a \qquad \overline{\boldsymbol{x}} = b$$

$\mathbb{IR}$ *is the set of all real intervals.*

Real intervals

Definition 2 (Real interval [Moore, 1966]). *A real interval $\boldsymbol{x}$ is a closed and connected set of real numbers, noted $[a, b]$.*

$$\boldsymbol{x} = \{x \in \mathbb{R} \mid a \leqslant x \leqslant b\} \qquad \underline{\boldsymbol{x}} = a \qquad \overline{\boldsymbol{x}} = b$$

$\mathbb{IR}$ is the set of all real intervals.

Some useful notions.

Width of $\boldsymbol{x}$: $\quad w(\boldsymbol{x}) = \overline{\boldsymbol{x}} - \underline{\boldsymbol{x}}$

Interval hull of $\rho \subset \mathbb{R}$: $\quad \mathrm{Hull}\,(\rho) = [\inf \rho, \sup \rho] = \Box\rho$

Real interval arithmetic

Definition 3 (Interval arithmetic (IA)). *Usual arithmetic-like arithmetic where handled items are intervals* (and no longer reals)

Real interval arithmetic

Definition 3 (Interval arithmetic (IA)). *Usual arithmetic-like arithmetic where handled items are intervals* (and no longer reals)

General formula of IA. *Let* $\diamond \in \{+, -, \times, /\}$

$$\boldsymbol{x} \diamond \boldsymbol{y} = \Box \ \{x \diamond y \,|\, x \in \boldsymbol{x},\ y \in \boldsymbol{y}\}$$

Real interval arithmetic

Definition 3 (Interval arithmetic (IA)). *Usual arithmetic-like arithmetic where handled items are intervals* (and no longer reals)

General formula of IA. *Let* $\diamond \in \{+, -, \times, /\}$

$$\boldsymbol{x} \diamond \boldsymbol{y} = \square\ \{x \diamond y \,|\, x \in \boldsymbol{x},\ y \in \boldsymbol{y}\}$$

Properties.

- *associativity*
- *commutativity*
- *sub-distributivity:* $x \times (y + z) \subset x \times y + x \times z$

  $\rightsquigarrow$ *interval arithm. is expression-dependent*

  $=$ *the DEPENDENCY PROBLEM*

Real interval arithmetic

Definition 3 (Interval arithmetic (IA)). *Usual arithmetic-like arithmetic where handled items are intervals* (and no longer reals)

General formula of IA. *Let* $\diamond \in \{+, -, \times, /\}$

$$\boldsymbol{x} \diamond \boldsymbol{y} = \square \{x \diamond y \mid x \in \boldsymbol{x},\ y \in \boldsymbol{y}\}$$

Properties.

- *associativity* $\rightsquigarrow$ *No longer valid!*
- *commutativity*
- *sub-distributivity:* $x \times (y + z) \subset x \times y + x \times z$

  $\rightsquigarrow$ *interval arithm. is expression-dependent*

  $=$ *the DEPENDENCY PROBLEM*

Interval extensions

IA Principle: *provides outer approximations of real quantities being looked for*

$\rightsquigarrow$ *used for the evaluation of the* **ranges of functions**

Interval extensions

IA Principle: *provides outer approximations of real quantities being looked for*

$\rightsquigarrow$ *used for the evaluation of the* **ranges of functions**

Definition 5 (Interval extension). *Let f be a real function defined over $E \subset \mathbb{R}^n$. Any interval function $\boldsymbol{\phi}$ is an interval extension of f provided that:*

$$\forall \boldsymbol{x} \subset \mathbb{IR}^n, \{f(x) \mid x \in \boldsymbol{x} \cap E\} \subset \boldsymbol{\phi}(\boldsymbol{x}).$$

Interval extensions

IA Principle: *provides outer approximations of real quantities being looked for*

$\rightsquigarrow$ *used for the evaluation of the* **ranges of functions**

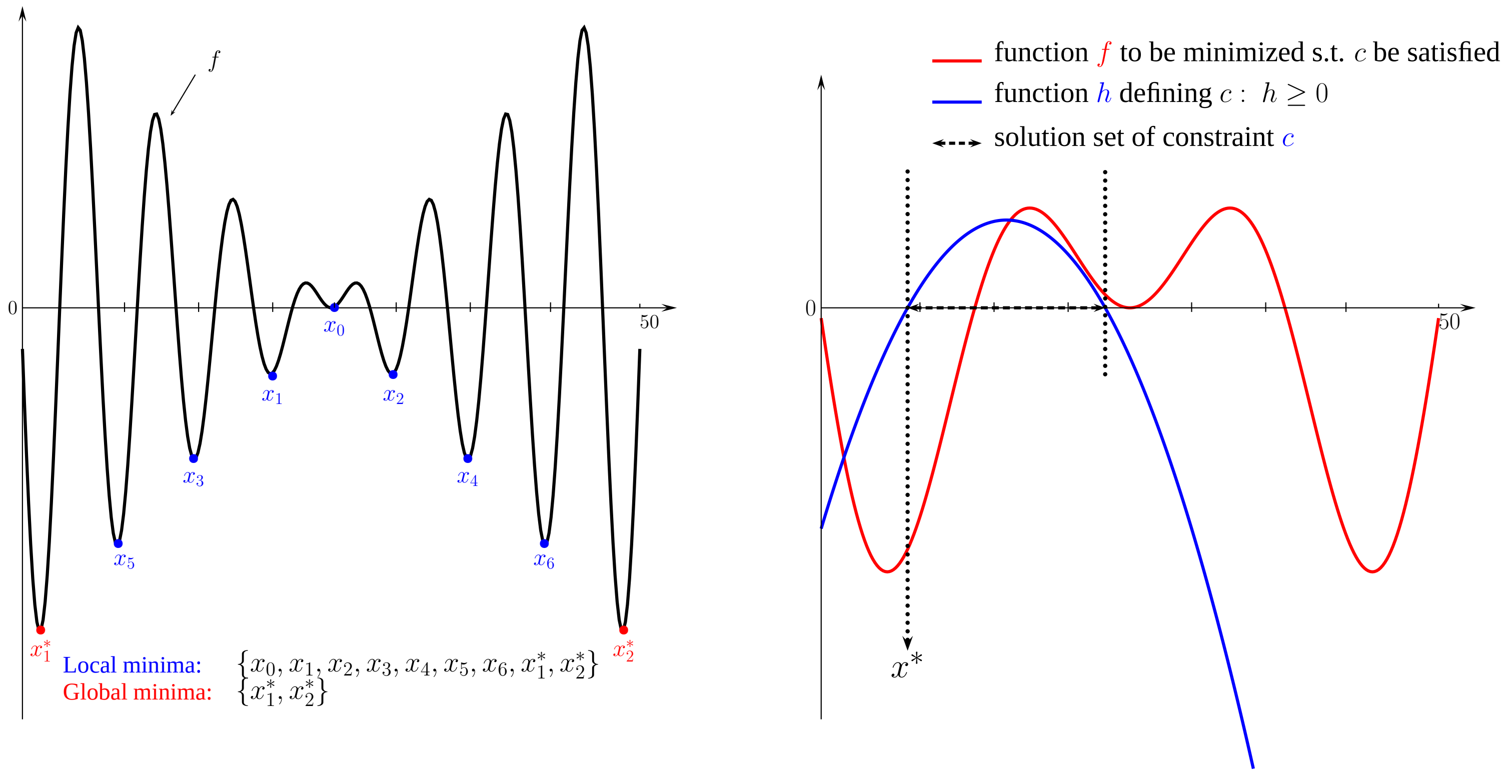
Definition 5 (Interval extension). *Let* f *be a real function defined over* $E \subset \mathbb{R}^n$*. Any interval function* ϕ *is an interval extension of* f *provided that:*

$$\forall \boldsymbol{x} \subset \mathbb{IR}^n,\ \{f(x) \mid x \in \boldsymbol{x} \cap E\} \subset \boldsymbol{\phi}(\boldsymbol{x}).$$

Examples. *possibility of an infinite number of interval extensions*

rough extension:	$\phi_{\boldsymbol{f}} : \boldsymbol{x} \mapsto [-\infty, +\infty]$	*totally useless*
ideal extension:	$\phi_{\boldsymbol{f}} : \boldsymbol{x} \mapsto \Box\{f(x) \mid x \in \boldsymbol{x}\}$	*extremely rare*
natural extension:	$\phi_{\boldsymbol{f}} : \boldsymbol{x} \mapsto \boldsymbol{f}(\boldsymbol{x})$	*syntactic interval extension*

Definition 1 (Unconstrained and constrained global optimization).



Optimization: solving methods (1/2)

Optimality conditions [Fritz, 1948] [Hiriart-Urruty, 1995&1996]

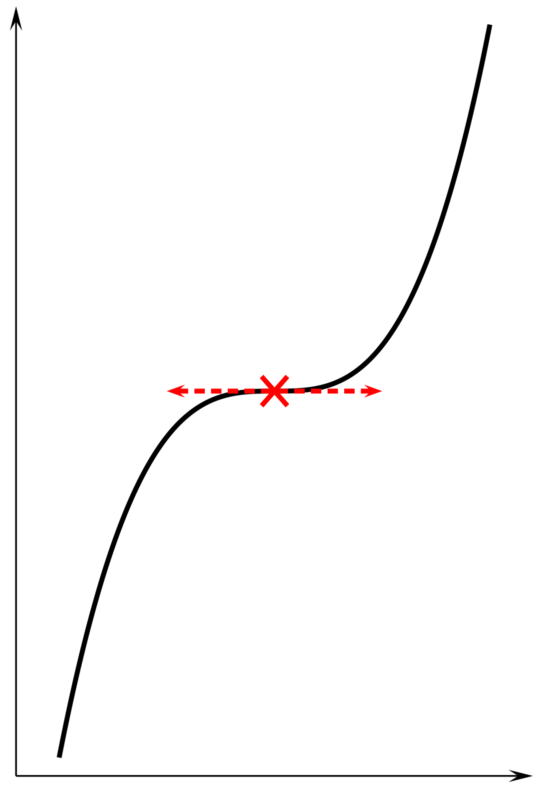
- Optimization problem $\rightsquigarrow$ constraint satisfaction problem

Optimization: solving methods (1/2)

Optimality conditions [Fritz, 1948] [Hiriart-Urruty, 1995&1996]

- Optimization problem $\rightsquigarrow$ constraint satisfaction problem

  ex. for unconstrained optimization, slope$= 0$



Optimization: solving methods (1/2)

Optimality conditions [Fritz, 1948] [Hiriart-Urruty, 1995&1996]

- Optimization problem $\rightsquigarrow$ constraint satisfaction problem

  ex. for unconstrained optimization, slope$= 0$

$\rightsquigarrow$ not necessarily an optimum, nor a global one (*except if the problem is convex*)

$\rightsquigarrow$ necessary but not sufficient conditions (*Lagrange, Fritz-John, Karush-Kuhn-Tucker*)

Optimization: solving methods (1/2)

Optimality conditions [Fritz, 1948] [Hiriart-Urruty, 1995&1996]

- Optimization problem $\rightsquigarrow$ constraint satisfaction problem

Penalty-based methods [Joines & Houck, 1994] [Michalewicz & al., 1995&1996]

- Constrained optimization problem $\rightsquigarrow$ unconstrained optimization problem

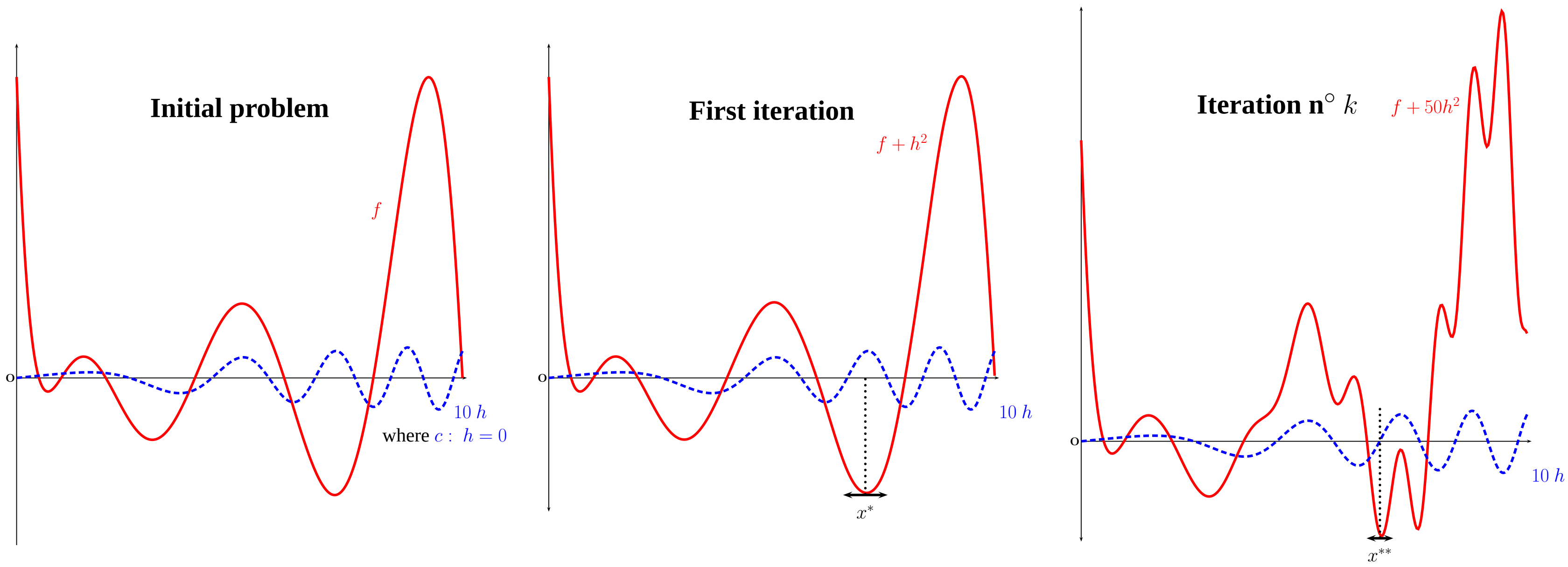
Optimization: solving methods (1/2)

Optimality conditions [Fritz, 1948] [Hiriart-Urruty, 1995&1996]

- Optimization problem $\rightsquigarrow$ constraint satisfaction problem

Penalty-based methods [Joines & Houck, 1994] [Michalewicz & al., 1995&1996]

- Constrained optimization problem $\rightsquigarrow$ unconstrained optimization problem



Optimization: solving methods (1/2)

Optimality conditions [Fritz, 1948] [Hiriart-Urruty, 1995&1996]

- Optimization problem ⇝ constraint satisfaction problem

Penalty-based methods [Joines & Houck, 1994] [Michalewicz & al., 1995&1996]

- Constrained optimization problem ⇝ unconstrained optimization problem

⇝ number of iterations uncontrolled, optimization process to be performed

⇝ no guarantee about the globality of the solutions

Optimization: solving methods (1/2)

Optimality conditions [Fritz, 1948] [Hiriart-Urruty, 1995&1996]

- Optimization problem $\rightsquigarrow$ constraint satisfaction problem

Penalty-based methods [Joines & Houck, 1994] [Michalewicz & al., 1995&1996]

- Constrained optimization problem $\rightsquigarrow$ unconstrained optimization problem

Meta-heuristics [Goldberg, 1989] [Michalewicz, 1996]

- genetic, evolutionary algorithms, tabu search, simulated annealing, clustering,

etc.

Optimization: solving methods (1/2)

Optimality conditions [Fritz, 1948] [Hiriart-Urruty, 1995&1996]

- Optimization problem $\rightsquigarrow$ constraint satisfaction problem

Penalty-based methods [Joines & Houck, 1994] [Michalewicz & al., 1995&1996]

- Constrained optimization problem $\rightsquigarrow$ unconstrained optimization problem

Meta-heuristics [Goldberg, 1989] [Michalewicz, 1996]

- genetic, evolutionary algorithms, tabu search, simulated annealing, clustering,

etc.

$\Downarrow$

$\boxed{\textit{Incomplete methods}}$

i.e., no guarantee about the solution set: minimum, globality, completeness

Optimization: solving methods (2/2)

Objective: a complete method $=$ globality, and no loss of solutions

Optimization: solving methods (2/2)

Objective: a complete method $=$ *globality, and no loss of solutions*

Continuation methods [Chen & Harker, 1993]

- series of auxiliary problems leading continuously to the initial problem to be solved

$\star$ global information, completeness

Optimization: solving methods (2/2)

Objective: a complete method $=$ *globality, and no loss of solutions*

Continuation methods [Chen & Harker, 1993]

- series of auxiliary problems leading continuously to the initial problem to be solved

$\star$ global information, completeness

$\dagger$ uneffective for high-order problems, and apply only to polynomial expressions

Optimization: solving methods (2/2)

Objective: a complete method $=$ *globality, and no loss of solutions*

Continuation methods [Chen & Harker, 1993]

- series of auxiliary problems leading continuously to the initial problem to be solved

$\star$ global information, completeness

† uneffective for high-order problems, and apply only to polynomial expressions

Interval methods [Hansen, 1992] [Kearfott, 1996]

- real quantities bounded by intervals, controlled rounding-errors

$\star$ global information, completeness

Optimization: solving methods (2/2)

Objective: a complete method $=$ *globality, and no loss of solutions*

Continuation methods [Chen & Harker, 1993]

- series of auxiliary problems leading continuously to the initial problem to be solved

⋆ global information, completeness

† uneffective for high-order problems, and apply only to polynomial expressions

Interval methods [Hansen, 1992] [Kearfott, 1996]

- real quantities bounded by intervals, controlled rounding-errors

⋆ global information, completeness

† more expensive computations (higher complexity)

† loss of accuracy

Interval optimization

Classical algorithms. *Branch-and-Bound / Prune algorithms*

[Hansen, 1992] [Kearfott, 1996] [VanHentenryck et al., 1995&1997]

= *upper-bound update and domain tightening processes*

Interval optimization

Classical algorithms. *Branch-and-Bound / Prune algorithms*

[Hansen, 1992] [Kearfott, 1996] [VanHentenryck et al., 1995&1997]

2 stable traits: *(interval)* **evaluation** *and* **constraint solving**

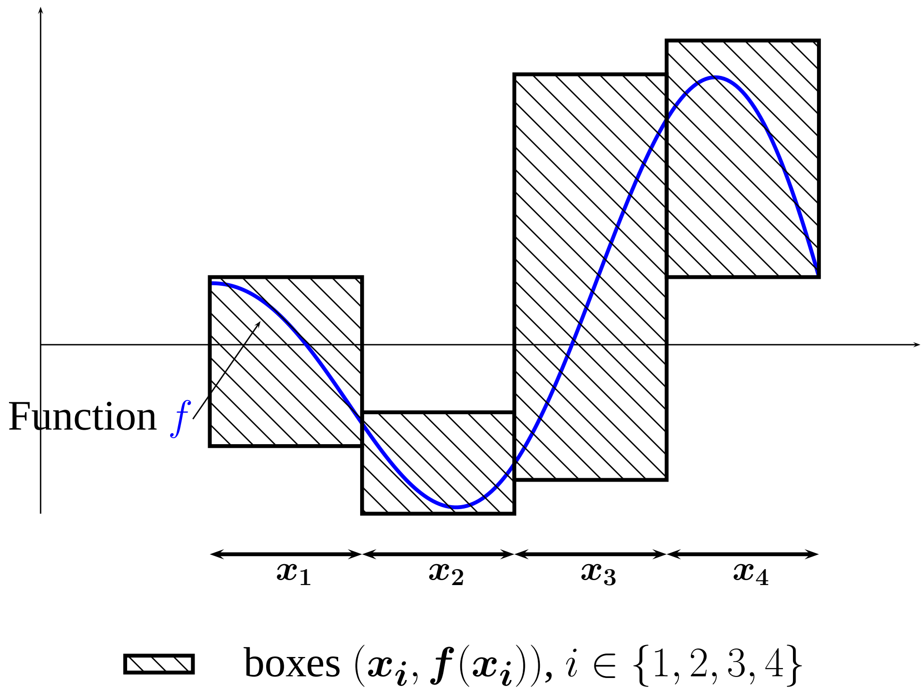
Interval optimization

Classical algorithms. *Branch-and-Bound / Prune algorithms*

[Hansen, 1992] [Kearfott, 1996] [VanHentenryck et al., 1995&1997]

2 stable traits: *(interval)* **evaluation** *and* **constraint solving**

Interval evaluation.



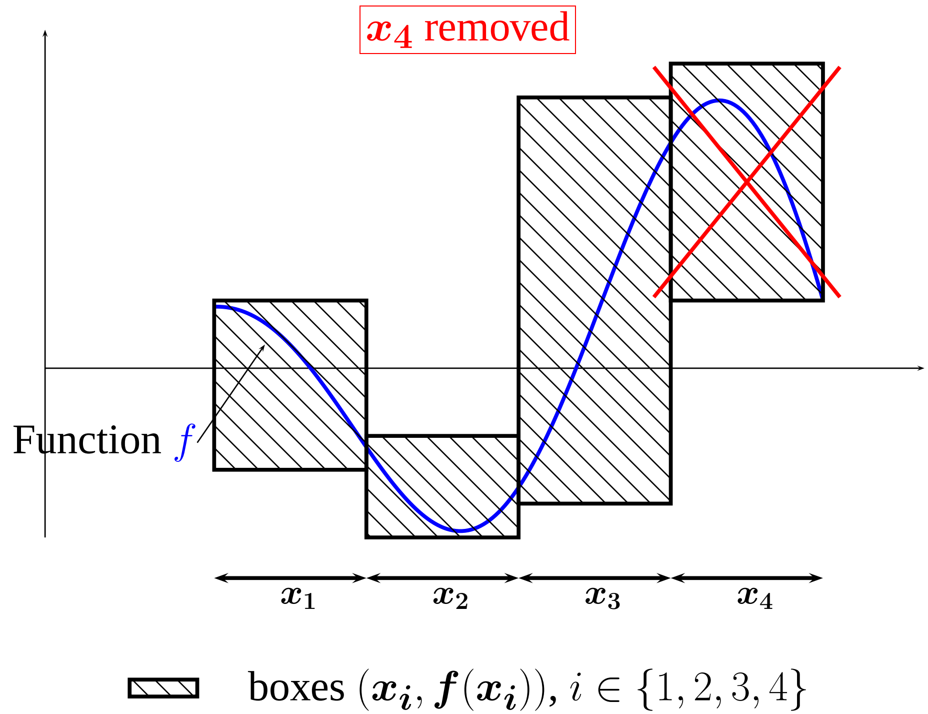
Interval optimization

Classical algorithms. *Branch-and-Bound / Prune algorithms*

[Hansen, 1992] [Kearfott, 1996] [VanHentenryck et al., 1995&1997]

2 stable traits: *(interval)* **evaluation** *and* **constraint solving**

Interval evaluation.

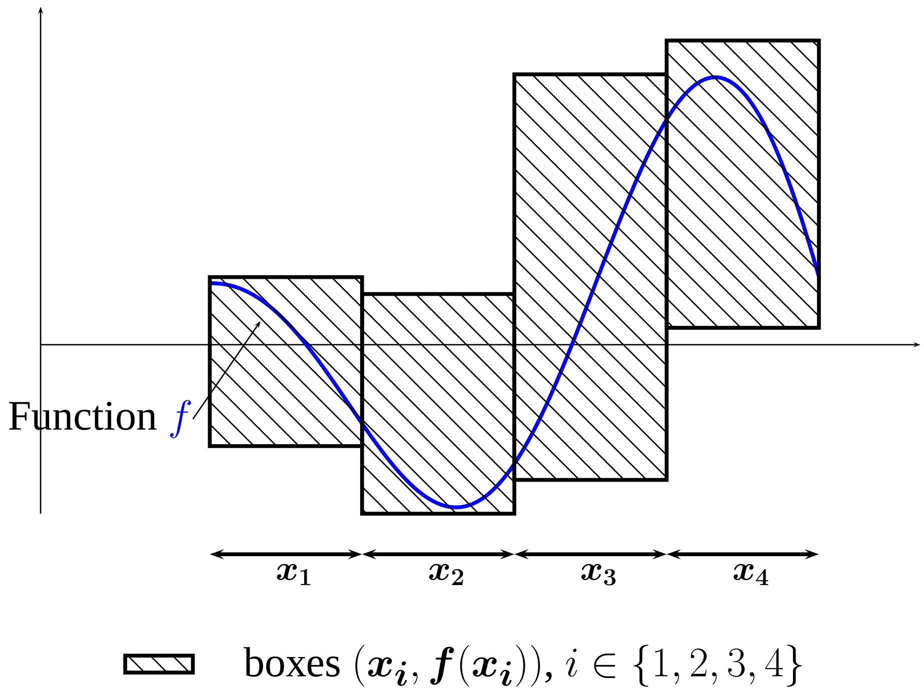


Classical algorithms. *Branch-and-Bound / Prune algorithms*

[Hansen, 1992] [Kearfott, 1996] [VanHentenryck et al., 1995&1997]

2 stable traits: *(interval)* **evaluation** *and* **constraint solving**

Interval evaluation.



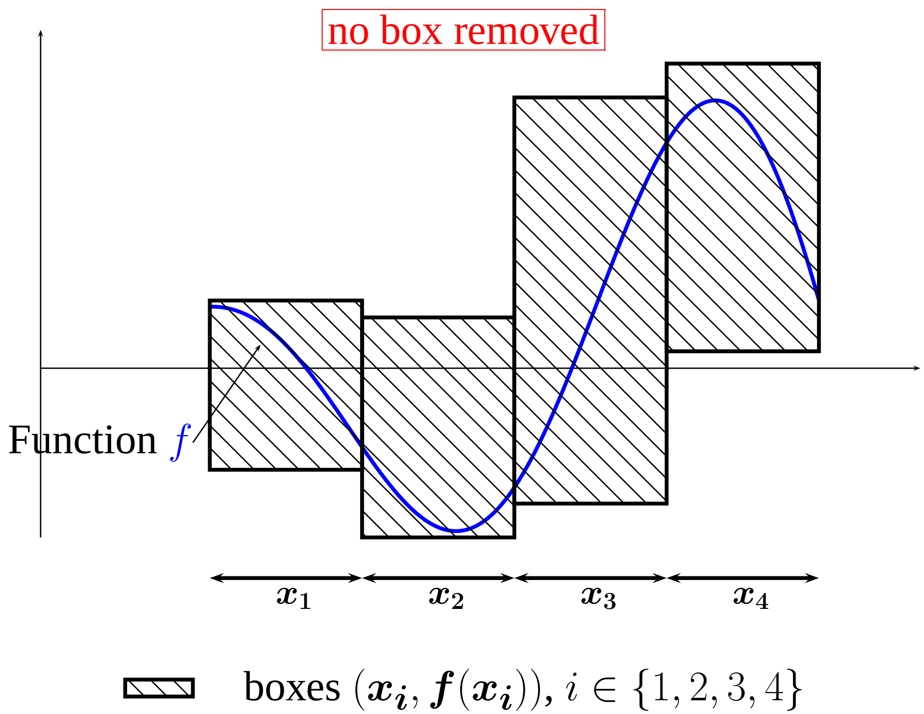
Interval optimization

Classical algorithms. *Branch-and-Bound / Prune algorithms*

[Hansen, 1992] [Kearfott, 1996] [VanHentenryck et al., 1995&1997]

2 stable traits: *(interval)* **evaluation** *and* **constraint solving**

Interval evaluation.



Interval optimization

Classical algorithms. *Branch-and-Bound / Prune algorithms*

[Hansen, 1992] [Kearfott, 1996] [VanHentenryck et al., 1995&1997]

2 stable traits: *(interval)* **evaluation** *and* **constraint solving**

Interval evaluation.

overestimation $=$ **dependency problem**

Interval optimization

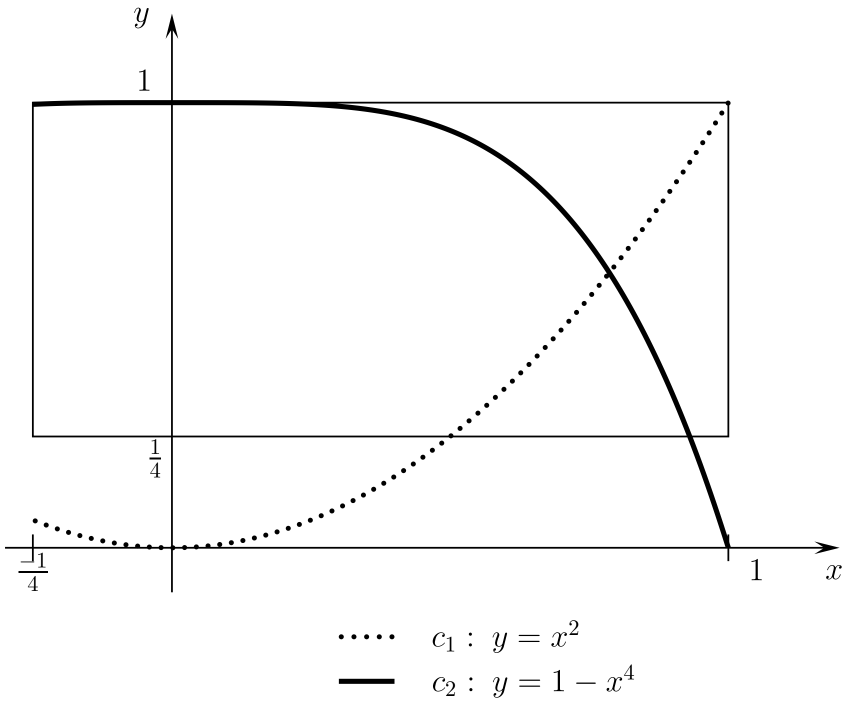
Classical algorithms. *Branch-and-Bound / Prune algorithms*

[Hansen, 1992] [Kearfott, 1996] [VanHentenryck et al., 1995&1997]

2 stable traits: *(interval)* **evaluation** *and* **constraint solving**

Interval evaluation. *dependency problem*

Constraint solving.



Interval optimization

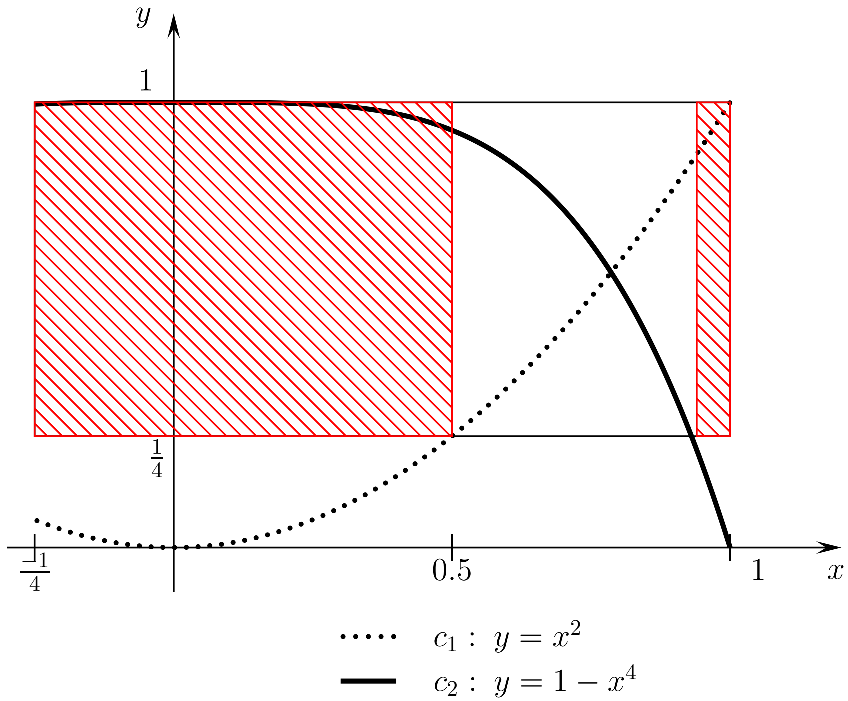
Classical algorithms. *Branch-and-Bound / Prune algorithms*

[Hansen, 1992] [Kearfott, 1996] [VanHentenryck et al., 1995&1997]

2 stable traits: *(interval)* **evaluation** *and* **constraint solving**

Interval evaluation. *dependency problem*

Constraint solving.



Interval optimization

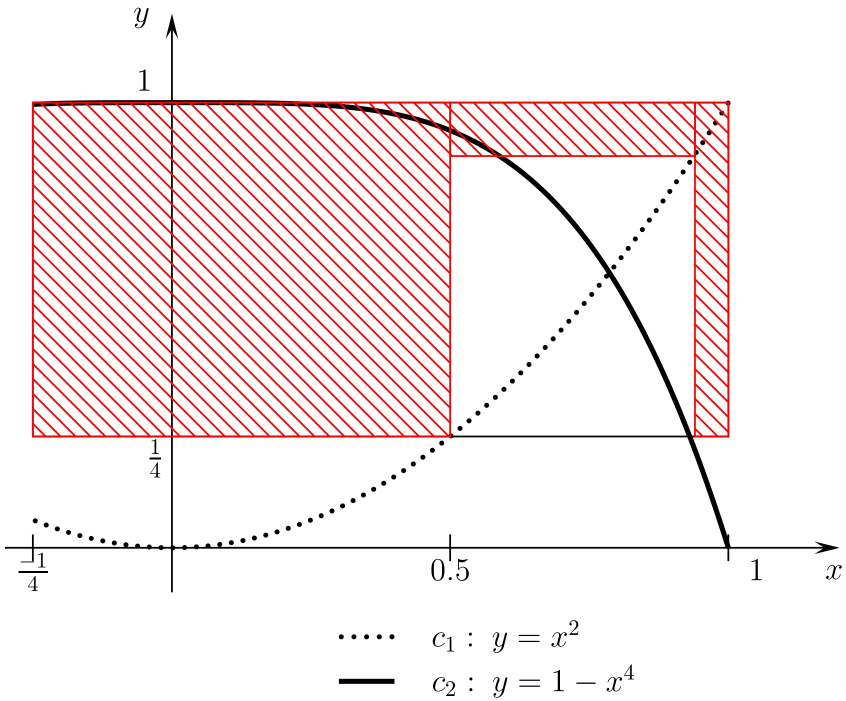
Classical algorithms. *Branch-and-Bound / Prune algorithms*

[Hansen, 1992] [Kearfott, 1996] [VanHentenryck et al., 1995&1997]

2 stable traits: *(interval)* **evaluation** *and* **constraint solving**

Interval evaluation. *dependency problem*

Constraint solving.



Interval optimization

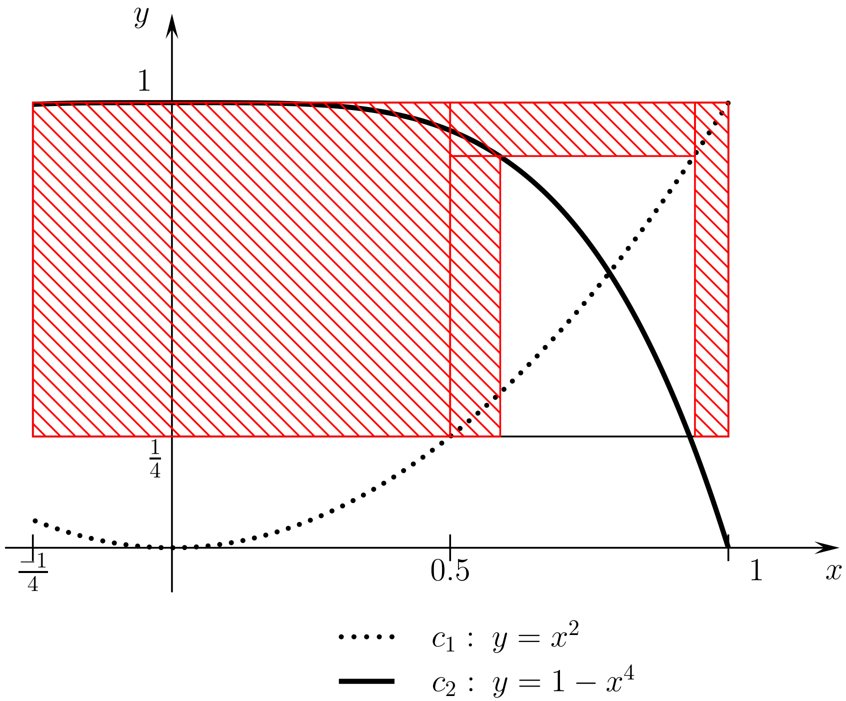
Classical algorithms. *Branch-and-Bound / Prune algorithms*

[Hansen, 1992] [Kearfott, 1996] [VanHentenryck et al., 1995&1997]

2 stable traits: *(interval)* **evaluation** *and* **constraint solving**

Interval evaluation. *dependency problem*

Constraint solving.



Interval optimization

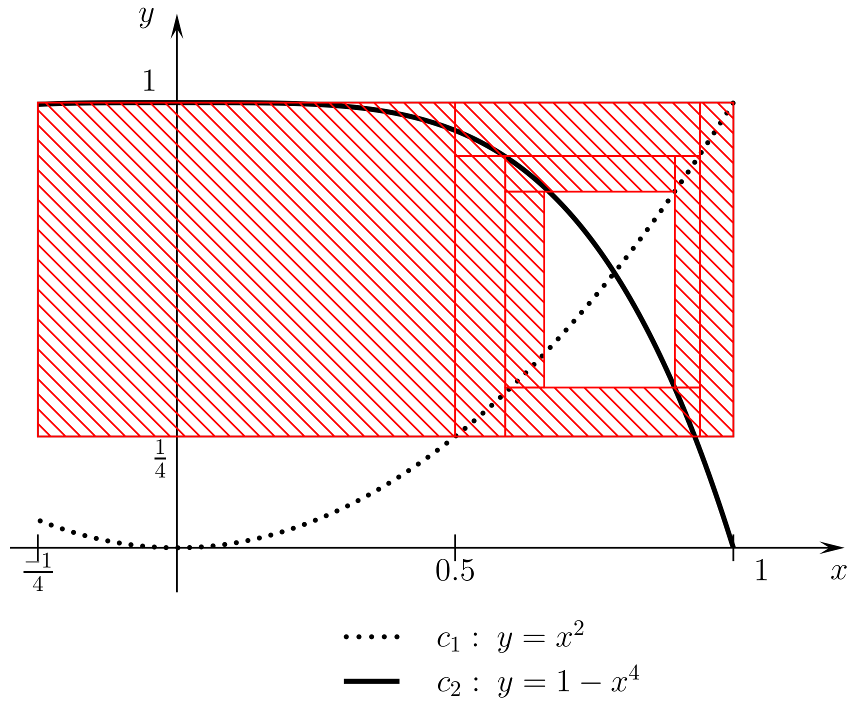
Classical algorithms. *Branch-and-Bound / Prune algorithms*

[Hansen, 1992] [Kearfott, 1996] [VanHentenryck et al., 1995&1997]

2 stable traits: *(interval)* **evaluation** *and* **constraint solving**

Interval evaluation. *dependency problem*

Constraint solving.



Interval optimization

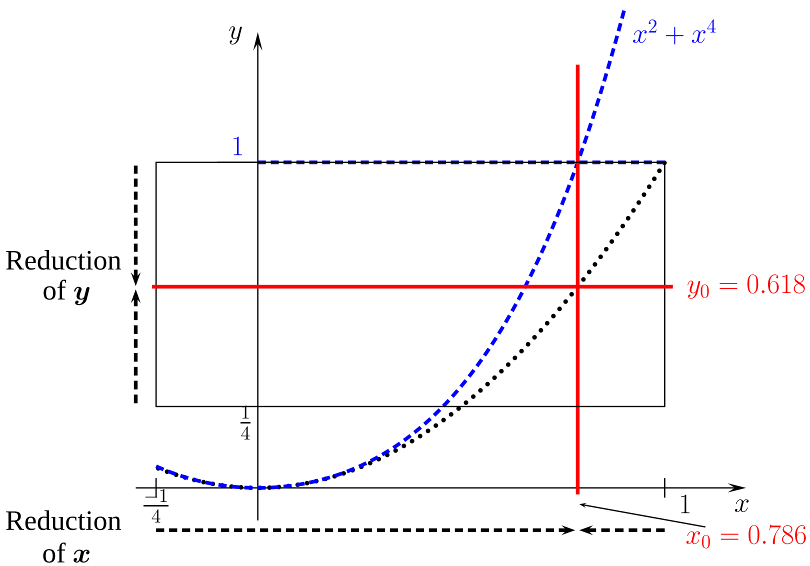
Classical algorithms. *Branch-and-Bound / Prune algorithms*

[Hansen, 1992] [Kearfott, 1996] [VanHentenryck et al., 1995&1997]

2 stable traits: *(interval)* **evaluation** *and* **constraint solving**

Interval evaluation. *dependency problem*

Constraint solving.



$$\boxed{\begin{array}{l} c_1 : \; y = x^2 \\ c_2 : \; y = 1 - x^4 \end{array}}$$

$$\Downarrow$$

$$\boxed{\begin{array}{l} c_1 : \; y = x^2 \\ c_2' : \; x^2 = 1 - x^4 \end{array}}$$

Interval optimization

Classical algorithms. *Branch-and-Bound / Prune algorithms*

[Hansen, 1992] [Kearfott, 1996] [VanHentenryck et al., 1995&1997]

2 stable traits: *(interval)* **evaluation** *and* **constraint solving**

Interval evaluation. *dependency problem*

Constraint solving.

locality of reasonings

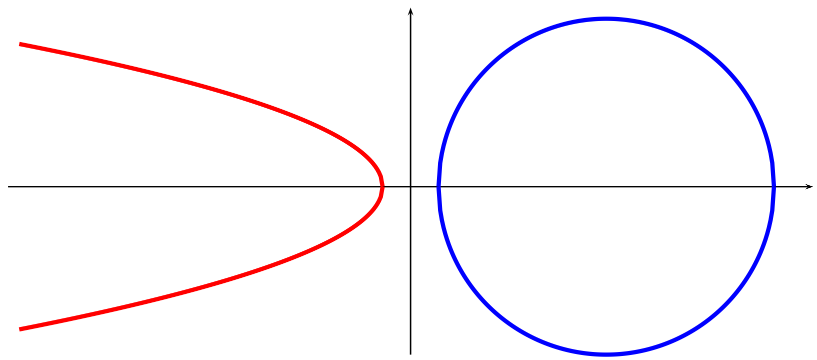
Soft constraints

Definition 6 (Soft constraint). *Given a constraint c over a set of variables V, defining a relation ρ. A soft constraint $\widehat{c}$ resulting from c is a constraint defining a relation $\widehat{\rho}$ over V s.t. $\rho \subset \widehat{\rho}$.*

Soft constraints

Definition 6 (Soft constraint). *Given a constraint c over a set of variables V, defining a relation ρ. A* *soft constraint* *$\widehat{c}$ resulting from c is a constraint defining a relation $\widehat{\rho}$ over V s.t. $\rho \subset \widehat{\rho}$.*

Considering softness... some possible treatments



— Constraint $c_1 : \ (x - \frac{7}{4})^2 + y^2 \leq \left(\frac{3}{2}\right)^2$

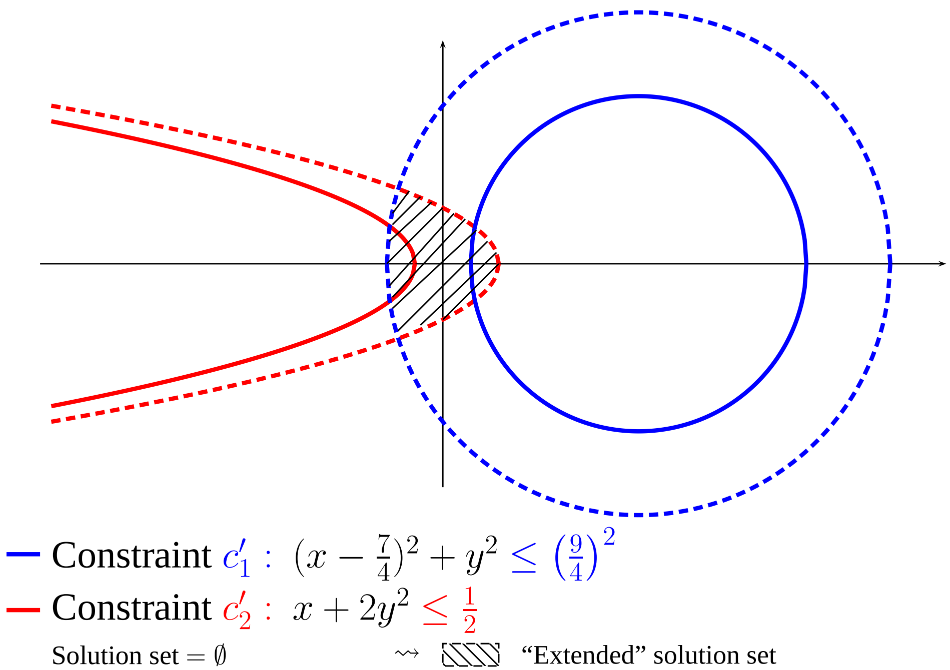
— Constraint $c_2 : \ x + 2y^2 \leq -\frac{1}{4}$

Solution set $= \emptyset$

Soft constraints

Definition 6 (Soft constraint). *Given a constraint c over a set of variables V, defining a relation ρ. A soft constraint $\widehat{c}$ resulting from c is a constraint defining a relation $\widehat{\rho}$ over V s.t. $\rho \subset \widehat{\rho}$.*

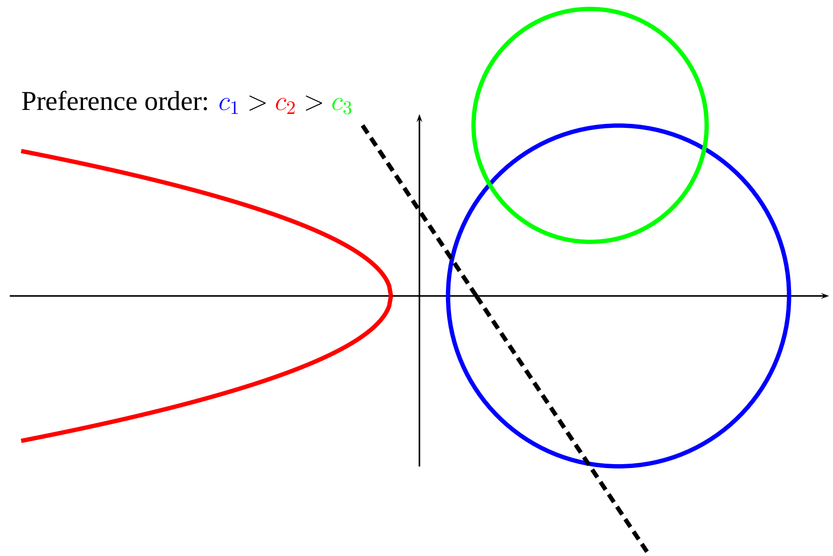
Considering softness... some possible treatments



Soft constraints

Definition 6 (Soft constraint). *Given a constraint c over a set of variables V, defining a relation ρ. A* *soft constraint* *$\widehat{c}$ resulting from c is a constraint defining a relation $\widehat{\rho}$ over V s.t. $\rho \subset \widehat{\rho}$.*

Considering softness... some possible treatments



Constraint c_1 : $(x - \frac{7}{4})^2 + y^2 \leq \left(\frac{3}{2}\right)^2$

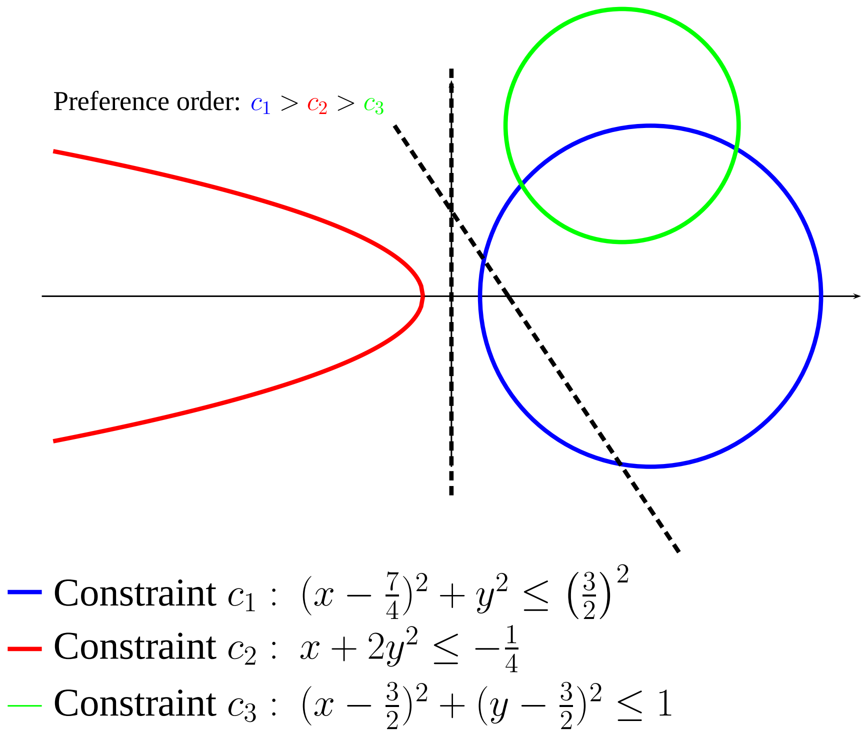
Constraint c_2 : $x + 2y^2 \leq -\frac{1}{4}$

Constraint c_3 : $(x - \frac{3}{2})^2 + (y - \frac{3}{2})^2 \leq 1$

Soft constraints

Definition 6 (Soft constraint). *Given a constraint c over a set of variables V, defining a relation ρ. A* *soft constraint* *$\widehat{c}$ resulting from c is a constraint defining a relation $\widehat{\rho}$ over V s.t. $\rho \subset \widehat{\rho}$.*

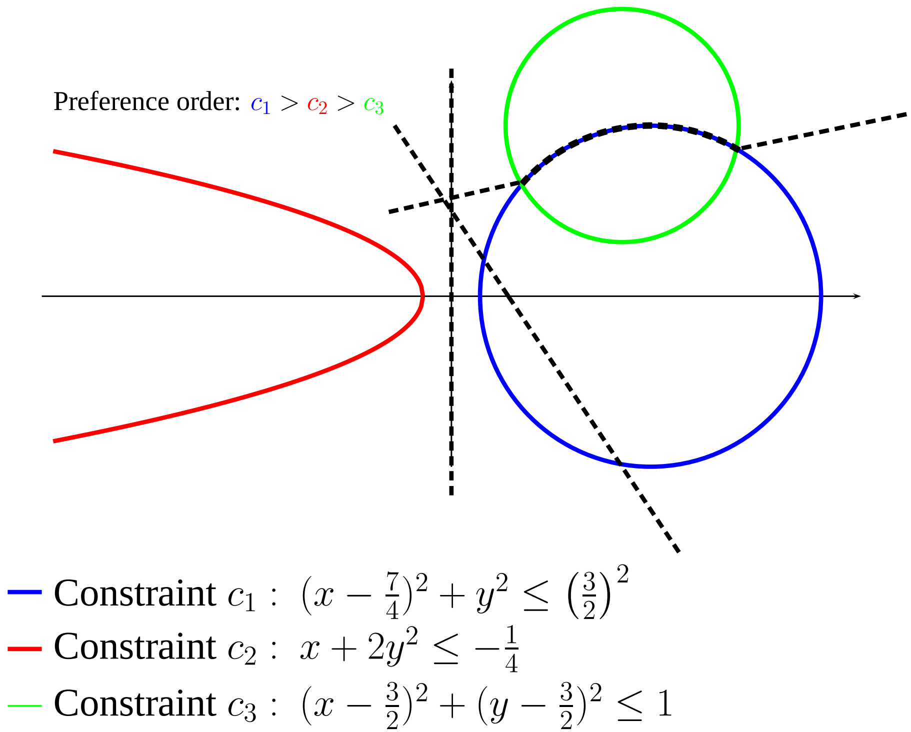
Considering softness... some possible treatments



Soft constraints

Definition 6 (Soft constraint). *Given a constraint c over a set of variables V, defining a relation ρ. A soft constraint $\widehat{c}$ resulting from c is a constraint defining a relation $\widehat{\rho}$ over V s.t. $\rho \subset \widehat{\rho}$.*

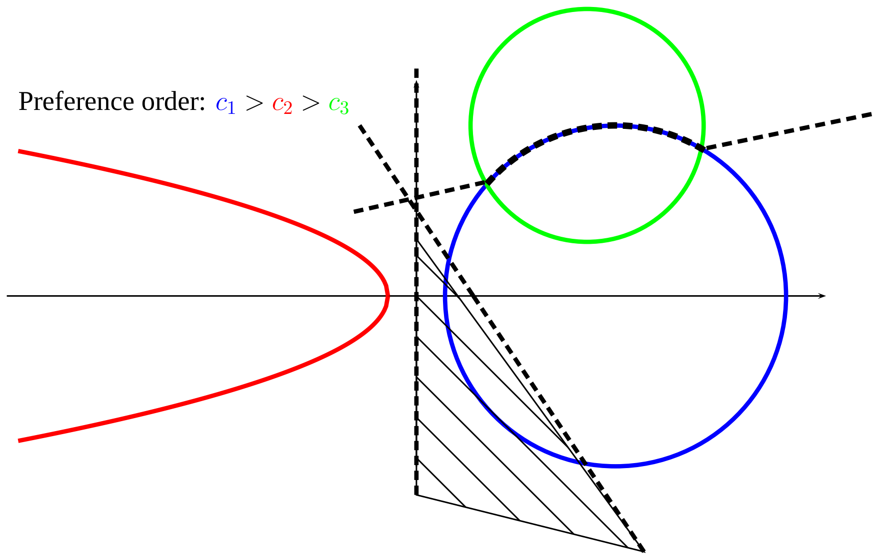
Considering softness... some possible treatments



Soft constraints

Definition 6 (Soft constraint). *Given a constraint c over a set of variables V, defining a relation ρ. A* *soft constraint* *$\widehat{c}$ resulting from c is a constraint defining a relation $\widehat{\rho}$ over V s.t. $\rho \subset \widehat{\rho}$.*

Considering softness... some possible treatments



Constraint c_1 : $(x - \frac{7}{4})^2 + y^2 \leq \left(\frac{3}{2}\right)^2$

Constraint c_2 : $x + 2y^2 \leq -\frac{1}{4}$

Constraint c_3 : $(x - \frac{3}{2})^2 + (y - \frac{3}{2})^2 \leq 1$

Soft constraints: frameworks

Hierarchical CSP [Borning et al., 1988,1989&1992] [Wilson, 1993]

- the set of constraints is ordered (hierarchical)

  objective: determining the instanciations *satisfying the hierarchy*

Soft constraints: frameworks

Hierarchical CSP [Borning et al., 1988,1989&1992] [Wilson, 1993]

- the set of constraints is ordered (hierarchical)

  objective: determining the instanciations *satisfying the hierarchy*

⋆ preferences over the constraints and over the search space

Soft constraints: frameworks

Hierarchical CSP [Borning et al., 1988,1989&1992] [Wilson, 1993]

- $\star$ preference over the constraints and over the search space

Partial CSP [Freuder & Wallace, 1995]

- given P to be solved, and some distance d, $(\mathcal{P}, d)$ ordered set of problems
  objective: determining *the closest problem* P' and solving it

Soft constraints: frameworks

Hierarchical CSP [Borning et al., 1988,1989&1992] [Wilson, 1993]

- ⋆ preference over the constraints and over the search space

Partial CSP [Freuder & Wallace, 1995]

- given P to be solved, and some distance d, $(\mathcal{P}, d)$ ordered set of problems
  objective: determining *the closest problem* P' and solving it
- ⋆ preference over the space of problems

Soft constraints: frameworks

Hierarchical CSP [Borning et al., 1988,1989&1992] [Wilson, 1993]

- ⋆ preference over the constraints and over the search space

Partial CSP [Freuder & Wallace, 1995]

- ⋆ preference over the space of problems

Semiring-based CSP [Bistarelli, Montanari & Rossi, 1997&1999]

- each instanciation x is valuated w.r.t. each constraint
  valuations are combined, and express the quality of x
  objective: determining *the best quality instanciation*

Soft constraints: frameworks

Hierarchical CSP [Borning et al., 1988,1989&1992] [Wilson, 1993]

- ⋆ preference over the constraints and over the search space

Partial CSP [Freuder & Wallace, 1995]

- ⋆ preference over the space of problems

Semiring-based CSP [Bistarelli, Montanari & Rossi, 1997&1999]

- each instanciation x is valuated w.r.t. each constraint
  valuations are combined, and express the quality of x
  objective: determining *the best quality instanciation*
- ⋆ preference over the search space

Soft constraints: frameworks

Hierarchical CSP [Borning et al., 1988,1989&1992] [Wilson, 1993]

- ⋆ preference over the constraints and over the search space

Partial CSP [Freuder & Wallace, 1995]

- ⋆ preference over the space of problems

Semiring-based CSP [Bistarelli, Montanari & Rossi, 1997&1999]

- ● each instanciation x is valuated w.r.t. each constraint

  valuations are combined, and express the quality of x

  objective: determining *the best quality instanciation*
- ⋆ preference over the search space
- † the *qualitative* aspect is drowned out by the (*quantitative*) combination

Soft constraints: frameworks

Hierarchical CSP [Borning et al., 1988,1989&1992] [Wilson, 1993]

- ⋆ preference over the constraints and over the search space

Partial CSP [Freuder & Wallace, 1995]

- ⋆ preference over the space of problems

Semiring-based CSP [Bistarelli, Montanari & Rossi, 1997&1999]

- ⋆ preference over the search space

Valued CSP [Bistarelli, Montanari & Rossi, 1997&1999]

- constraints are valuated (weighted)

  instanciations are valued through the constraint valuation

  objective: determining *the best quality instanciation*

  equivalent to SCSP

Soft constraints: frameworks

Hierarchical CSP [Borning et al., 1988,1989&1992] [Wilson, 1993]

⋆ preference over the constraints and over the search space

Partial CSP [Freuder & Wallace, 1995]

⋆ preference over the space of problems

Semiring-based CSP [Bistarelli, Montanari & Rossi, 1997&1999]

⋆ preference over the search space

Valued CSP [Bistarelli, Montanari & Rossi, 1997&1999]

- constraints are valuated (weighted)

  instanciations are valued through the constraint valuation

  objective: determining *the best quality instanciation*

  equivalent to SCSP

⋆ a kind of preferences

Soft constraints: frameworks

Hierarchical CSP [Borning et al., 1988,1989&1992] [Wilson, 1993]

- ⋆ preference over the constraints and over the search space

Partial CSP [Freuder & Wallace, 1995]

- ⋆ preference over the space of problems

Semiring-based CSP [Bistarelli, Montanari & Rossi, 1997&1999]

- ⋆ preference over the search space

Valued CSP [Bistarelli, Montanari & Rossi, 1997&1999]

- ⋆ equivalent to SCSP

Fuzzy CSP [Dubois, Fargier & Prade, 1996] [Moura Pires, 2000]

- integrated in the SCSP framework

Soft constraints: frameworks

Hierarchical CSP [Borning et al., 1988,1989&1992] [Wilson, 1993]

- ⋆ preference over the constraints and over the search space

Partial CSP [Freuder & Wallace, 1995]

- ⋆ preference over the space of problems

Semiring-based CSP [Bistarelli, Montanari & Rossi, 1997&1999]

- ⋆ preference over the search space

Valued CSP [Bistarelli, Montanari & Rossi, 1997&1999]

- ⋆ equivalent to SCSP

Fuzzy CSP [Dubois, Fargier & Prade, 1996] [Moura Pires, 2000]

- integrated in the SCSP framework

  ex: priorities, discrimin (leximin)

Soft constraints: frameworks

Hierarchical CSP [Borning et al., 1988,1989&1992] [Wilson, 1993]

- ⋆ preference over the constraints and over the search space

Partial CSP [Freuder & Wallace, 1995]

- ⋆ preference over the space of problems

Semiring-based CSP [Bistarelli, Montanari & Rossi, 1997&1999]

- ⋆ preference over the search space

Valued CSP [Bistarelli, Montanari & Rossi, 1997&1999]

- ⋆ equivalent to SCSP

Fuzzy CSP [Dubois, Fargier & Prade, 1996] [Moura Pires, 2000]

- integrated in the SCSP framework
- ⋆ allows to express priorities and preferences

Recap’

There is room for improvement:

- the **dependency problem** of interval computations;
- the **locality of reasonings** arising in constraint solving;

In the following, we also present:

- a **unifying framework** for modeling and **solving** soft constraints.
- and a way to address some problems in **distributed constraint solving**

Outline of the presentation

- Continuous constraints: definitions and solving process
- An example of under and over-constrained problems
- Important notions
- Some research directions
- Conclusion

Outline of the presentation

- *Continuous constraints: definitions and solving process*
- *An example of under and over-constrained problems*
- *Important notions*
- Some research directions
  - *Interval evaluation: the dependency problem*
  - *Constraint solving: the locality of reasonings*
  - *Soft constraints: a unifying hard framework*
  - *Distributed constraints: speculating to solve faster*
- *Conclusion*

Some research directions

The dependency problem

The locality of reasonings

A unifying framework for soft constraints

Distributed constraints: speculations

1. The dependency problem

The workings of this problem

Classical treatments and their limits

Another factorization method

The workings

1. Independency of the occurrences.

2 occurrences of the **same** *variable "behave" as if they were* **different** *variables*

The workings

1. Independency of the occurrences.

2 occurrences of the **same** *variable "behave" as if they were* **different** *variables*

$$x = [-1, 1] \rightsquigarrow x \times x = [-1, 1] \text{ instead of } [0, 1]$$

$$= [\underline{x}\overline{x}, \overline{x}\underline{x}]$$

$$= x \times y, \text{ where } y = x$$

The workings

1. Independency of the occurrences.

2 occurrences of the **same** *variable "behave" as if they were* **different** *variables*

⋆ *limiting the number of occurrences [Hong & Stahl, 1994][Ceberio & Granvilliers, 2000]*

The workings

1. Independency of the occurrences.

2 occurrences of the **same** *variable “behave” as if they were* **different** *variables*

$\star$ *limiting the number of occurrences [Hong & Stahl, 1994][Ceberio & Granvilliers, 2000]*

2. Monotonicities.

occurrences are independent $\rightsquigarrow$ *respecting monotonicities is crucial for the computations to be performed on* **the proper bounds**

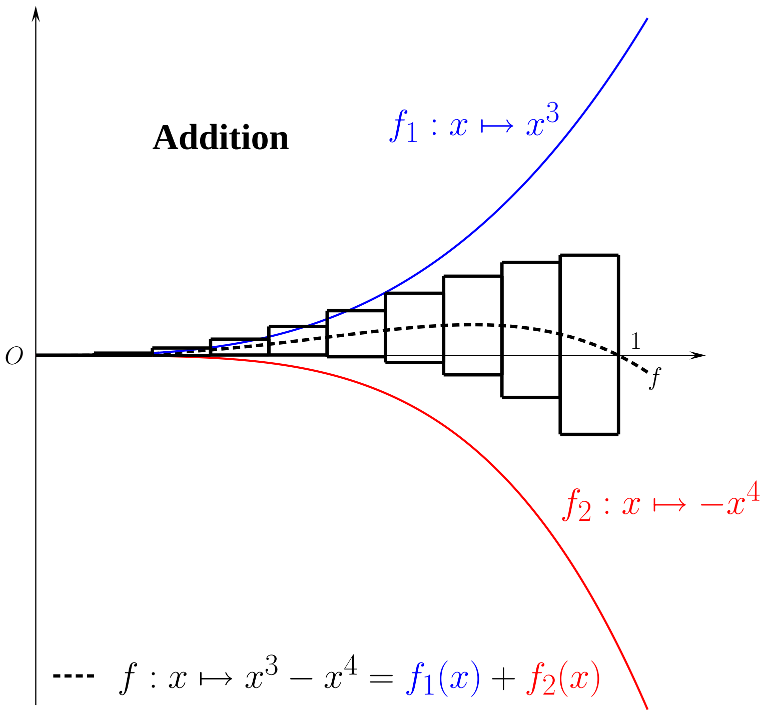
The workings

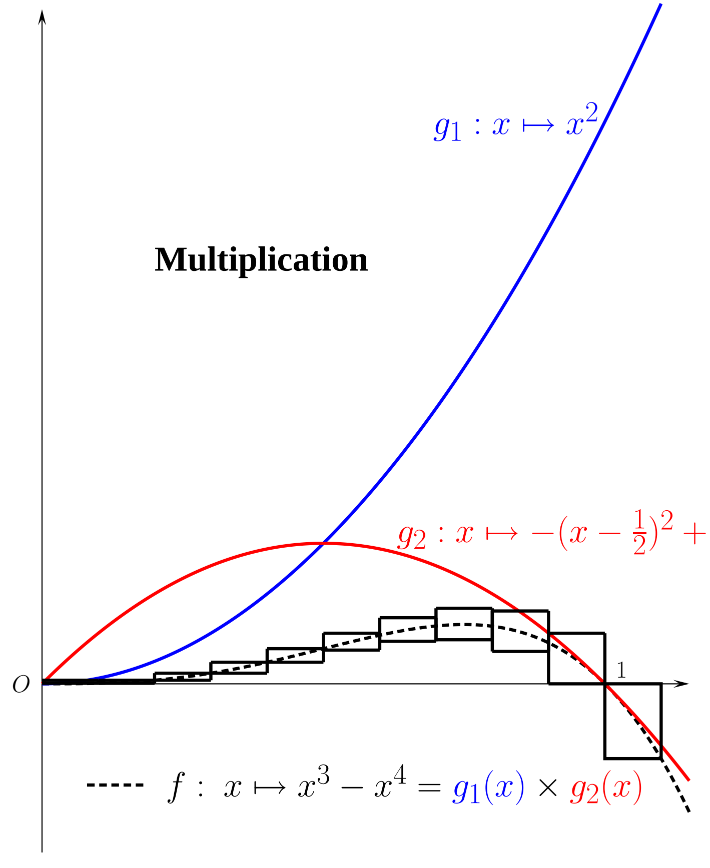
1. Independency of the occurrences.

2 occurrences of the **same** *variable "behave" as if they were* **different** *variables*

⋆ *limiting the number of occurrences [Hong & Stahl, 1994][Ceberio & Granvilliers, 2000]*

2. Monotonicities.





The workings

1. Independency of the occurrences.

2 occurrences of the **same** *variable "behave" as if they were* **different** *variables*

⋆ *limiting the number of occurrences [Hong & Stahl, 1994][Ceberio & Granvilliers, 2000]*

2. Monotonicities.

occurrences are independent ⇝ *monotony is to be respected so that computations are performed on* **the proper bounds**

† *difficult to determine the monotonicities*

⋆ *at least, we try to respect some properties:*

multiplications *are easier to handle and control,*

sub-distributivity *of IA*

The workings

1. Independency of the occurrences.

2 occurrences of the **same** *variable "behave" as if they were* **different** *variables*

$\star$ *limiting the number of occurrences* *[Hong & Stahl, 1994][Ceberio & Granvilliers, 2000]*

2. Monotonicities.

occurrences are independent $\rightsquigarrow$ *monotony is to be respected so that computations are performed on* **the proper bounds**

† *difficult to determine the monotonicities*

$\star$ *at least, we try to respect some properties:*

multiplications *are easier to handle and control,*

sub-distributivity *of IA*

$\rightsquigarrow$ **factorized forms**

Interval Horner form. *[Shih-Chieh, 1303][Horner, 1819][Stahl, 1995]*

Let p *be a polynomial defined by:* $a_0 + \sum_{i=1}^{n} a_i x^{\alpha_i}$

$$\boxed{\boldsymbol{h_p}(\boldsymbol{x}) = \boldsymbol{a}_0 + \boldsymbol{x}^{d_1}\left(\cdots + \boldsymbol{x}^{d_{n-1}}(\boldsymbol{a}_{n-1} + \boldsymbol{a}_n \boldsymbol{x}^{d_n})\cdots\right)}$$

Def. *Intermediate polynomials:*

$$\begin{cases} p_n(x) & = & a_n \\ p_i(x) & = & x^{d_{i+1}} p_{i+1}(x) + a_i \quad i = n-1, n-2, \ldots, 0 \end{cases}$$

Classical treatments and their limits

for univariate polynomials

Interval Horner form. *[Shih-Chieh, 1303][Horner, 1819][Stahl, 1995]*

Let p *be a polynomial defined by:* $a_0 + \sum_{i=1}^{n} a_i x^{\alpha_i}$

$$\boxed{\boldsymbol{h_p}(\boldsymbol{x}) = \boldsymbol{a}_0 + \boldsymbol{x}^{d_1}\left(\cdots + \boldsymbol{x}^{d_{n-1}}(\boldsymbol{a}_{n-1} + \boldsymbol{a}_n \boldsymbol{x}^{d_n})\cdots\right)}$$

$=$ **optimal w.r.t. factorization:**

1. *made of only multiplications and additions of constants* $\rightsquigarrow$ **monotonicity**

2. *completely nested* $\rightsquigarrow$ **sub-distributivity**

Interval Horner form. *[Shih-Chieh, 1303][Horner, 1819][Stahl, 1995]*

Let p *be a polynomial defined by:* $a_0 + \sum_{i=1}^{n} a_i x^{\alpha_i}$

$$\boxed{\boldsymbol{h_p}(\boldsymbol{x}) = \boldsymbol{a}_0 + \boldsymbol{x}^{d_1}\left(\cdots + \boldsymbol{x}^{d_{n-1}}(\boldsymbol{a}_{n-1} + \boldsymbol{a}_n \boldsymbol{x}^{d_n})\cdots\right)}$$

1. Monotonicity.

Let $O_p = \square\{$ *all the zeros of the intermediate polynomials of* $h_p \cup \{0\}$ $\}$

$$\boxed{\forall \boldsymbol{x} \in \mathbb{IR}^n \text{ s.t. } \mathring{\boldsymbol{x}} \cap O_p = \emptyset, \; \boldsymbol{h_p}(\boldsymbol{x}) = \{p(x) \mid x \in \boldsymbol{x}\}}$$

Classical treatments and their limits

for univariate polynomials

Interval Horner form. *[Shih-Chieh, 1303][Horner, 1819][Stahl, 1995]*

Let p *be a polynomial defined by:* $a_0 + \sum_{i=1}^{n} a_i x^{\alpha_i}$

$$\boxed{\boldsymbol{h_p}(\boldsymbol{x}) = \boldsymbol{a}_0 + \boldsymbol{x}^{d_1}\left(\cdots + \boldsymbol{x}^{d_{n-1}}(\boldsymbol{a}_{n-1} + \boldsymbol{a}_n \boldsymbol{x}^{d_n})\cdots\right)}$$

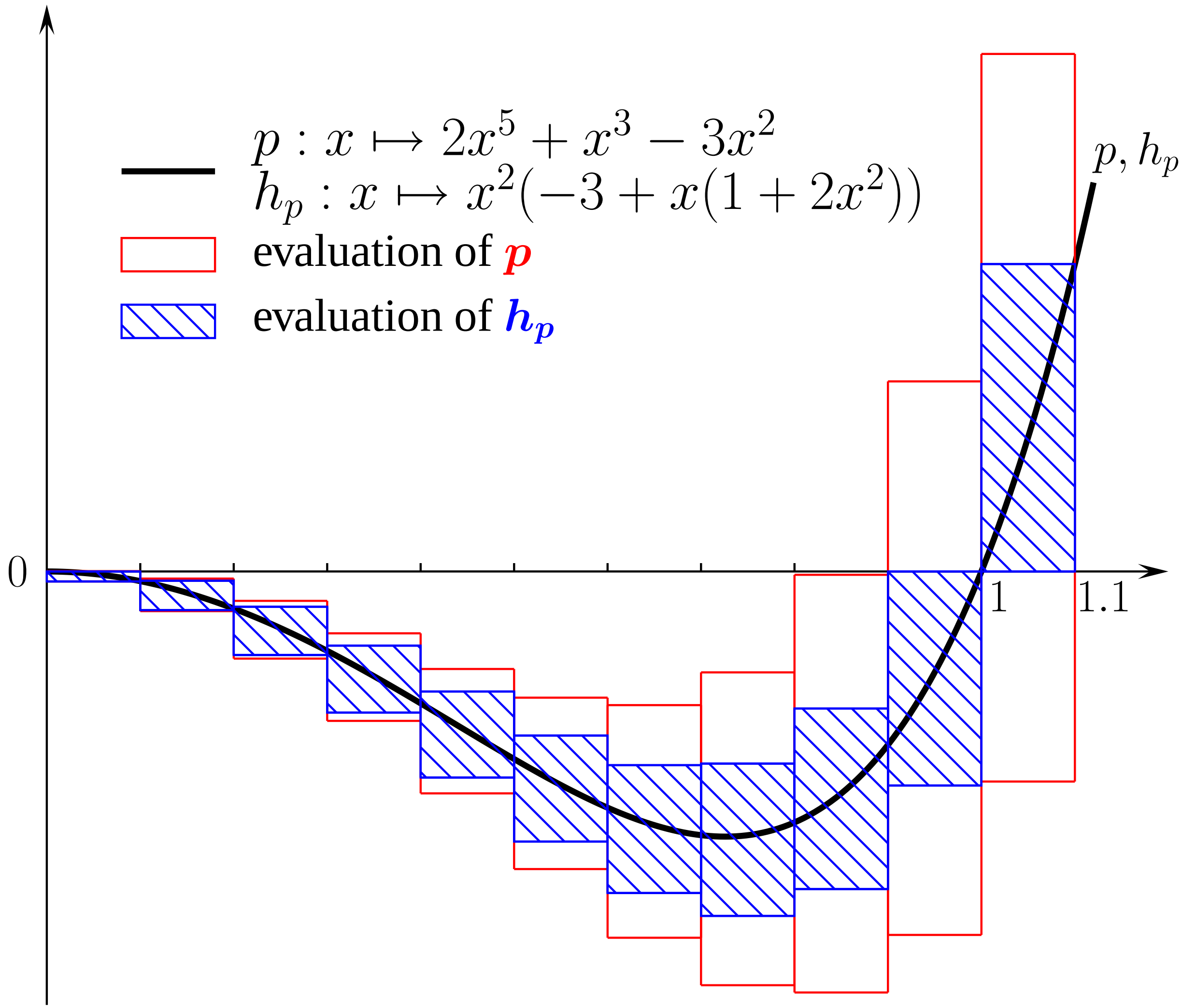
1. Monotonicity.

Let $O_p = \square\{$ *all the zeros of the intermediate polynomials of* $h_p \cup \{0\}$ $\}$

$$\boxed{\forall \boldsymbol{x} \in \mathbb{IR}^n \text{ s.t. } \mathring{\boldsymbol{x}} \cap O_p = \varnothing,\ \boldsymbol{h_p}(\boldsymbol{x}) = \{p(x) \mid x \in \boldsymbol{x}\}}$$

† *beyond this condition, no guarantee.*

Horner



$p : x \mapsto 2x^5 + x^3 - 3x^2$
$h_p : x \mapsto x^2(-3 + x(1 + 2x^2))$
evaluation of p
evaluation of h_p
p, h_p
0
1
1.1

Classical treatments and their limits

for univariate polynomials

Interval Horner form. *[Shih-Chieh, 1303][Horner, 1819][Stahl, 1995]*

Let p *be a polynomial defined by:* $a_0 + \sum_{i=1}^{n} a_i x^{\alpha_i}$

$$\boxed{\boldsymbol{h_p}(\boldsymbol{x}) = \boldsymbol{a}_0 + \boldsymbol{x}^{d_1}\left(\cdots + \boldsymbol{x}^{d_{n-1}}(\boldsymbol{a}_{n-1} + \boldsymbol{a}_n \boldsymbol{x}^{d_n})\cdots\right)}$$

1. Monotonicity.

Let $O_p = \square\{$ *all the zeros of the intermediate polynomials of* $h_p \cup \{0\}$ $\}$

$$\boxed{\forall \boldsymbol{x} \in \mathbb{IR}^n \text{ s.t. } \mathring{\boldsymbol{x}} \cap O_p = \varnothing,\ \boldsymbol{h_p}(\boldsymbol{x}) = \{p(x) \mid x \in \boldsymbol{x}\}}$$

† *beyond this condition, no guarantee.*

† *pb. with the decomposition of powers*

Classical treatments and their limits

for univariate polynomials

Interval Horner form. *[Shih-Chieh, 1303][Horner, 1819][Stahl, 1995]*

Let p *be a polynomial defined by:* $a_0 + \sum_{i=1}^{n} a_i x^{\alpha_i}$

$$\boxed{\boldsymbol{h_p}(\boldsymbol{x}) = \boldsymbol{a}_0 + \boldsymbol{x}^{d_1}\left(\cdots + \boldsymbol{x}^{d_{n-1}}(\boldsymbol{a}_{n-1} + \boldsymbol{a}_n \boldsymbol{x}^{d_n})\cdots\right)}$$

2. Sub-distributivity.

$$\boldsymbol{a}_0 + \overbrace{\boldsymbol{x}\cdots\boldsymbol{x}}^{d_1 \text{ times}}(\cdots + \overbrace{\boldsymbol{x}\cdots\boldsymbol{x}}^{d_{n-1} \text{ times}}(\boldsymbol{a}_{n-1} + \boldsymbol{a}_n \overbrace{\boldsymbol{x}\cdots\boldsymbol{x}}^{d_n \text{ times}})\cdots) \subseteq \boldsymbol{a}_0 + \sum_{i=1}^{n} \boldsymbol{a}_i \overbrace{\boldsymbol{x}\cdots\boldsymbol{x}}^{\beta_i \text{ times}}$$

Interval Horner form. *[Shih-Chieh, 1303][Horner, 1819][Stahl, 1995]*

Let p *be a polynomial defined by:* $a_0 + \sum_{i=1}^{n} a_i x^{\alpha_i}$

$$\boxed{\boldsymbol{h_p}(\boldsymbol{x}) = \boldsymbol{a}_0 + \boldsymbol{x}^{d_1}\left(\cdots + \boldsymbol{x}^{d_{n-1}}(\boldsymbol{a}_{n-1} + \boldsymbol{a}_n \boldsymbol{x}^{d_n})\cdots\right)}$$

2. Sub-distributivity.

$$\boxed{\boldsymbol{a}_0 + \overbrace{\boldsymbol{x}\cdots\boldsymbol{x}}^{d_1 \text{ times}}(\cdots + \overbrace{\boldsymbol{x}\cdots\boldsymbol{x}}^{d_{n-1} \text{ times}}(\boldsymbol{a}_{n-1} + \boldsymbol{a}_n \overbrace{\boldsymbol{x}\cdots\boldsymbol{x}}^{d_n \text{ times}})\cdots) \subseteq \boldsymbol{a}_0 + \sum_{i=1}^{n} \boldsymbol{a}_i \overbrace{\boldsymbol{x}\cdots\boldsymbol{x}}^{\beta_i \text{ times}}}$$

$$\boldsymbol{a}_0 + \boldsymbol{x}^{d_1}\left(\cdots + \boldsymbol{x}^{d_{n-1}}(\boldsymbol{a}_{n-1} + \boldsymbol{a}_n \boldsymbol{x}^{d_n})\cdots\right) \overset{?}{\subseteq} \boldsymbol{a}_0 + \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{x}^{\alpha_i}$$

Interval Horner form. *[Shih-Chieh, 1303][Horner, 1819][Stahl, 1995]*

Let p *be a polynomial defined by:* $a_0 + \sum_{i=1}^{n} a_i x^{\alpha_i}$

$$\boxed{\boldsymbol{h_p}(\boldsymbol{x}) = \boldsymbol{a}_0 + \boldsymbol{x}^{d_1}\left(\cdots + \boldsymbol{x}^{d_{n-1}}(\boldsymbol{a}_{n-1} + \boldsymbol{a}_n \boldsymbol{x}^{d_n})\cdots\right)}$$

2. Sub-distributivity.

$$\begin{aligned} p(x) &= x + x^4 & \qquad h_p(x) &= x(x^3+1) \\ q(x) &= x + xxxx & \qquad r(x) &= x(xxx+1) \end{aligned}$$

Let $\boldsymbol{x} = [-2, 1]$*:*

$$\begin{aligned} \boldsymbol{p}(\boldsymbol{x}) &= [-2, 17] & \qquad \boldsymbol{h_p}(\boldsymbol{x}) &= [-7, 14] \\ \boldsymbol{q}(\boldsymbol{x}) &= [-10, 17] & \qquad \boldsymbol{r}(\boldsymbol{x}) &= [-10, 14] \end{aligned}$$

Classical treatments and their limits

for univariate polynomials

Interval Horner form. *[Shih-Chieh, 1303][Horner, 1819][Stahl, 1995]*

Let p *be a polynomial defined by:* $a_0 + \sum_{i=1}^{n} a_i x^{\alpha_i}$

$$\boxed{\boldsymbol{h}_{\boldsymbol{p}}(\boldsymbol{x}) = \boldsymbol{a}_0 + \boldsymbol{x}^{d_1}\left(\cdots + \boldsymbol{x}^{d_{n-1}}(\boldsymbol{a}_{n-1} + \boldsymbol{a}_n \boldsymbol{x}^{d_n})\cdots\right)}$$

Limits of Horner's form.

† *when intersecting the overestimation set: no guarantee*

† *does not benefit from the sub-distributivity property*

$\rightsquigarrow$ **Another factorization scheme**

Another factorization scheme

for univariate polynomials

Objectives:	*1. controlled decomposition of powers*
	2. priority to even powers

Another factorization scheme

for univariate polynomials

Objectives:	*1. controlled decomposition of powers*
	2. priority to even powers

Elementary scheme. *Given* $p(x) = ax^{\alpha+\gamma} + bx^{\alpha}$,

$$\boxed{\textit{Mcr}_p(x) = ax^{\alpha-\gamma}\,\left[\left(x^{\gamma} + \tfrac{b}{2a}\right)^2 - \left(\tfrac{b}{2a}\right)^2\right]}$$

with: a, $b \in \mathbb{R}^*$, $\alpha \geqslant \gamma$ *and* $\alpha + \gamma$ *even.*

Horner form of the same binomial: $h_p(x) = x^{\alpha}(b + ax^{\gamma})$

Another factorization scheme

for univariate polynomials

Objectives:	*1. controlled decomposition of powers*
	2. priority to even powers

Elementary scheme. *Given* $p(x) = ax^{\alpha+\gamma} + bx^{\alpha}$,

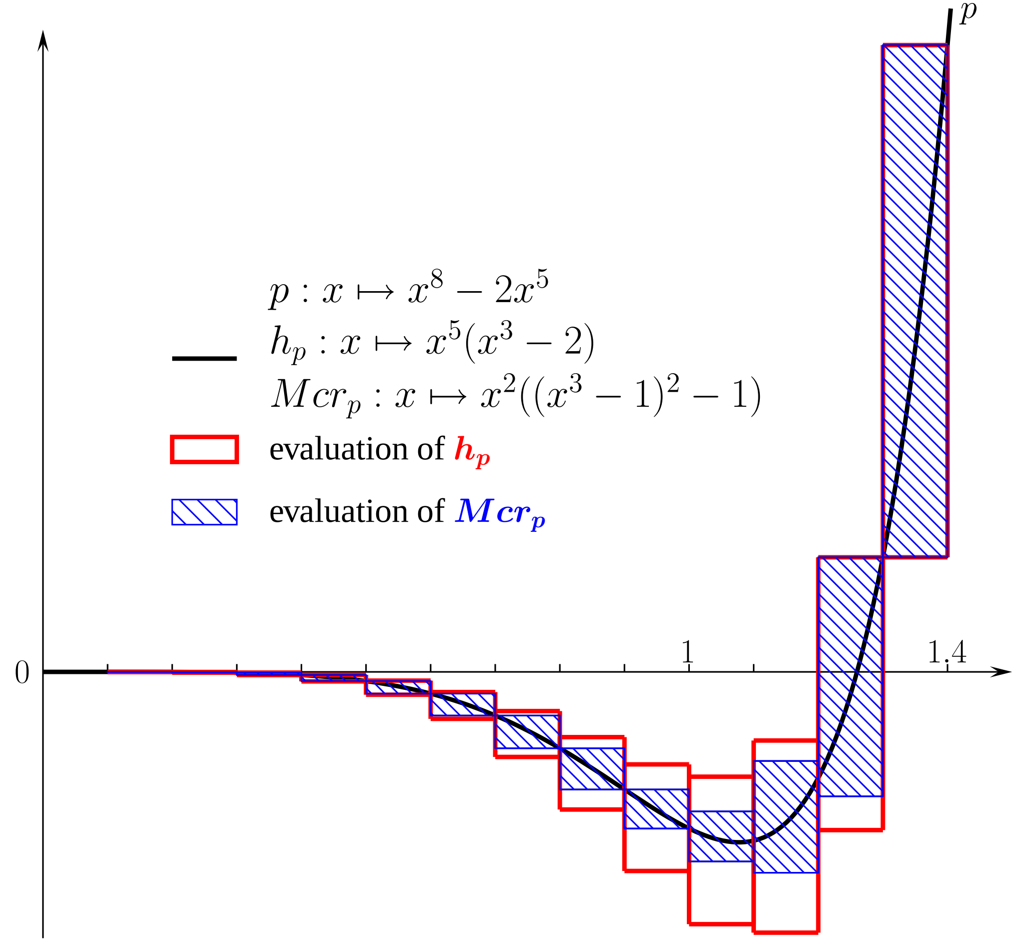
$$\boxed{\mathit{Mcr}_p(x) = ax^{\alpha-\gamma}\ \left[\left(x^{\gamma} + \frac{b}{2a}\right)^2 - \left(\frac{b}{2a}\right)^2\right]}$$

with: a, $b \in \mathbb{R}^*$, $\alpha \geqslant \gamma$ *and* $\alpha + \gamma$ *even.*

Main properties.

- $\forall \boldsymbol{x} \in \mathbb{IR},\ 0 \notin \boldsymbol{x}\ \rightarrow\ w(\mathit{Mcr}_{\boldsymbol{p}}(\boldsymbol{x})) \leqslant w(\boldsymbol{h_p}(\boldsymbol{x}))$
- $\left\{\begin{array}{l} (ab > 0 \text{ and } (\underline{\boldsymbol{x}} \geqslant 0 \text{ or } \overline{\boldsymbol{x}}^{\gamma} \leqslant -\frac{b}{a})) \\ \text{or } (ab < 0 \text{ and } (\overline{\boldsymbol{x}} \leqslant 0 \text{ or } \underline{\boldsymbol{x}}^{\gamma} \geqslant \frac{b}{a})) \end{array}\right. \rightarrow \mathit{Mcr}_p(\boldsymbol{x}) = \{p(x) \mid x \in \boldsymbol{x}\}$

Mcr_p vs. Horner



Another factorization scheme

for univariate polynomials

Generalization. *Given* $p(x) = \sum_{i=1}^{n} a_i x^i$, *we define:*

$$I = \{(i,j) \in \{0, \cdots, n\}^2 \mid a_i \neq 0,\ a_j \neq 0,\ i < j < 2i,\ j \text{ is even}\}$$

and $I' \subset I$ *without shared monomials*

Another factorization scheme

for univariate polynomials

Generalization. *Given* $p(x) = \sum_{i=1}^{n} a_i x^i$, *we define:*

$$I = \{(i,j) \in \{0, \cdots, n\}^2 \mid a_i \neq 0,\ a_j \neq 0,\ i < j < 2i,\ j \text{ is even}\}$$

and $I' \subset I$ *without shared monomials*

$\rightsquigarrow$ *we can rewrite* p *as follows:*

$$p(x) = r(x) + \sum_{(i,j) \in I'} \left(a_i x^i + a_j x^j\right) = r(x) + \sum_{(i,j) \in I'} p_{i,j}(x)$$

Another factorization scheme

for univariate polynomials

Generalization. *Given* $p(x) = \sum_{i=1}^{n} a_i x^i$, *we define:*

$$I = \{(i,j) \in \{0,\cdots,n\}^2 \mid a_i \neq 0,\ a_j \neq 0,\ i < j < 2i,\ j \text{ is even}\}$$

and $I' \subset I$ *without shared monomials*

$\rightsquigarrow$ *we can rewrite* p *as follows:*

$$p(x) = r(x) + \sum_{(i,j)\in I'} \left(a_i x^i + a_j x^j\right) = r(x) + \sum_{(i,j)\in I'} p_{i,j}(x)$$

and we finally factorize:

$$\boxed{\mathit{Mcr}_p(x) = r(x) + \sum_{(i,j)\in I'} \mathit{Mcr}_{p_{i,j}}(x)}$$

Another factorization scheme

for univariate polynomials

Generalization. *Given* $p(x) = \sum_{i=1}^{n} a_i x^i$, *we define:*

$$I = \{(i,j) \in \{0,\cdots,n\}^2 \mid a_i \neq 0,\ a_j \neq 0,\ i < j < 2i,\ j \text{ is even}\}$$

and $I' \subset I$ *without shared monomials*

$\rightsquigarrow$ *we can rewrite* p *as follows:*

$$p(x) = r(x) + \sum_{(i,j)\in I'} \left(a_i x^i + a_j x^j\right) = r(x) + \sum_{(i,j)\in I'} p_{i,j}(x)$$

and we finally factorize:

$$\boxed{\mathit{Mcr}_p(x) = r(x) + \sum_{(i,j)\in I'} \mathit{Mcr}_{p_{i,j}}(x)}$$

many possibilities $\rightsquigarrow$ *strategies are defined*

Strategies and tests

Main principles.

- *No decomposition of odd powers*
- *No decomposition of even powers into odd ones*
- *No introduction of odd powers / deletion of odd powers*

Strategies and tests

Main principles.

- *No decomposition of odd powers*
- *No decomposition of even powers into odd ones*
- *No introduction of odd powers / deletion of odd powers*

Two classes of strategies. *parsing the expressions in the increasing order of their powers*

1. *given a power i, another one is looked for between $i+1$ and $2i$*
2. *priority to the factorization of odd powers, i.e., schemes (i,j) where i is odd*

Strategies and tests

Main principles.

- *No decomposition of odd powers*
- *No decomposition of even powers into odd ones*
- *No introduction of odd powers / deletion of odd powers*

Two classes of strategies. *parsing the expressions in the increasing order of their powers*

$$p(x) = x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^9 + x^{12}$$

1. *given a power i, another one is looked for between $i+1$ and $2i$*
$\{(2,4),(3,6),(7,12),5,9\} \quad \{(2,4),(3,6),5,7,9,12\}$

2. *priority to the factorization of odd powers, i.e., schemes (i,j) where i is odd*
$\{(3,4),(5,6),(7,12),2,9\}$

Strategies and tests

Main principles.

- *No decomposition of odd powers*
- *No decomposition of even powers into odd ones*
- *No introduction of odd powers / deletion of odd powers*

Tests and results.

Sparse polynomials: *the greater α, the sparser $P_{\alpha,n}$*

$$P_{\alpha,n}(x) = (x^\alpha - 1)^n = \sum_{k=0}^{n} (-1)^{n-k} C_n^k x^{k\alpha}$$

Comparison of several forms to the exact range of $P_{\alpha,n}$ over $\boldsymbol{x} = [-0.5, 0.3]$

α	1	2	3	4	5
$s_1 \& s_2$	1.11	2.57	1.02	1.00	1.00
s'_1	1.11	4.86	1.07	1.05	1.00
horner	1.49	2.92	1.10	1.34	1.09
natural	1.15	2.92	1.08	1.34	1.05

Strategies and tests

Main principles.

- *No decomposition of odd powers*
- *No decomposition of even powers into odd ones*
- *No introduction of odd powers / deletion of odd powers*

Tests and results.

Sparse polynomials: *the greater α, the sparser $P_{\alpha,n}$*

$$P_{\alpha,n}(x) = (x^\alpha - 1)^n = \sum_{k=0}^{n} (-1)^{n-k} C_n^k x^{k\alpha}$$

Randomly generated polynomials: 500*-polynomial basis*

interval evaluations using Mcr are globally better than Horner's

Recap'

Best strategy:

- *second strategy* ($\boldsymbol{\varphi}$) *when* $\mathring{\boldsymbol{x}} \cap O_p \neq \varnothing$ $\approx 25\%$*-improvement (w.r.t. our tests)*
- *Horner otherwise*

$\longrightarrow$ *globally composition of Horner with our strategy on average*

Properties.

• *beyond the overestimation interval,* $\boldsymbol{h} \circ \varphi$ *is* **equivalent to** $\boldsymbol{p}$
• *otherwise,* $\boldsymbol{h} \circ \varphi_{\boldsymbol{b}}$ *globally* **improves the Horner form** *(w.r.t. our tests), while always keeping equivalent to* $\boldsymbol{p}$

Research directions

The dependency problem

The locality of reasonings

A unifying framework for soft constraints

Distributed constraints: speculations

2. The locality of reasonings

The workings of this problem

Classical treatments and their limits

Triangularization is an idea

The workings

- *the propagation stage only communicates locally consistent domains*
- *pieces of information are lost between constraints*

for instance the correspondance of bounds is lost, drowned out in the local reasonings

$\rightsquigarrow$ **a new symbolic representation**

to enhance the propagation stage

Classical treatments and their limits

Redundant constraints *[Marti & Rueher, 1995] [Benhamou & Granvilliers, 1998]*

[Van Emden, 1999]

Classical treatments and their limits

Redundant constraints *[Marti & Rueher, 1995] [Benhamou & Granvilliers, 1998]*

[Van Emden, 1999]

Linear constraint solving *and introduction of nonlinear constraints when their nonlinear*

variables are determined [Colmerauer, 1993]

Linearization of the nonlinear terms *[Yamamura et al., 1998]*

⋆ *these methods aim at improving the propagation stage*

† *no control of the accuracy of interval computations*

† *or no stopping control* $\rightsquigarrow$ *exponential in time and memory*

Classical treatments and their limits

Redundant constraints *[Marti & Rueher, 1995] [Benhamou & Granvilliers, 1998]*

[Van Emden, 1999]

Linear constraint solving *[Colmerauer, 1993]*

Linearization of the nonlinear terms *[Yamamura et al., 1998]*

Gaussian elimination

⋆ *generation of triangular systems, information totally shared is the system is totally triangular*

† *only for linear systems*

Classical treatments and their limits

Redundant constraints *[Marti & Rueher, 1995] [Benhamou & Granvilliers, 1998] [Van Emden, 1999]*

Linear constraint solving *[Colmerauer, 1993]*

Linearization of the nonlinear terms *[Yamamura et al., 1998]*

Gaussian elimination

control of the amount of transformations

$+$ **control of the interval computations accuracy**

$=$ **A new triangularization scheme**

Triangularization is an option

Consider the following nonlinear constraint system:

$$\boxed{C: \begin{cases} c_1: & x+y+x^2+xy+y^2 & = & 0 \\ c_2: & x+t+xy+t^2+x^2 & = & 0 \\ c_3: & y+z+x^2+z^2 & = & 0 \\ c_4: & x+z+x^2+y^2+z^2+xy & = & 0 \end{cases}}$$

defined over $E = [-100, 100]^4$*,*

4 *solutions reached in* 140*ms using* **realpaver** *[Granvilliers, 2002].*

- *difficult to remove nonlinear terms* $\rightsquigarrow$ *the nonlinear terms are abstracted*

Triangularization is an option

Abstraction phase: *equivalent system*

$$\boxed{\left\{\begin{array}{llllllllllll} lc_1: & x & +y & & & +u_1 & +u_2 & +u_3 & & & =0 \\ lc_2: & x & & & +t & +u_1 & +u_2 & & +u_4 & & =0 \\ lc_3: & & y & +z & & +u_1 & & & & +u_5 & =0 \\ lc_4: & x & & +z & & +u_1 & +u_2 & +u_3 & & +u_5 & =0 \end{array}\right.}$$

and the abstracted system: $\left\{\begin{array}{lll} u_1 & = & x^2 \\ u_2 & = & xy \\ u_3 & = & y^2 \\ u_4 & = & t^2 \\ u_5 & = & z^2 \end{array}\right.$

Triangularization is an option

Gaussian elimination phase:

$$\left\{\begin{array}{llllllllllll}
lc_1: & u_1 & +y & & & & & +x & +u_2 & + & u_3 & =0 \\
lc'_2: & & y & & & -t & -u_4 & & & + & u_3 & =0 \\
lc'_3: & & & -z & -u_5 & & & +x & +u_2 & + & u_3 & =0 \\
lc'_4: & & & & & -t & -u_4 & +x & +u_2 & & +2u_3 & =0
\end{array}\right.$$

Triangularization is an option

Gaussian elimination phase:

$$\boxed{\begin{cases} lc_1: & u_1 & +y & & & & & +x & +u_2 & + & u_3 & =0 \\ lc'_2: & & y & & & -t & -u_4 & & & + & u_3 & =0 \\ lc'_3: & & & -z & -u_5 & & & +x & +u_2 & + & u_3 & =0 \\ lc'_4: & & & & & -t & -u_4 & +x & +u_2 & & +2u_3 & =0 \end{cases}}$$

- *nonlinear terms are restored*

Triangularization is an option

Concretization phase:

$$
\boxed{\left\{\begin{array}{lcccccccccc}
lc_1: & x^2 & +y & & & & & +x & +xy & + \; y^2 & =0 \\
lc'_2: & & y & & & -t & -t^2 & & & + \; y^2 & =0 \\
lc'_3: & & & -z & -z^2 & & & +x & +xy & + \; y^2 & =0 \\
lc'_4: & & & & & -t & -t^2 & +x & +xy & +2y^2 & =0
\end{array}\right.}
$$

The new system is solved in 240*ms!!*

Triangularization is an option

Concretization phase:

$$\boxed{\left\{\begin{array}{llllllllllll}
lc_1: & x^2 & +y & & & & & +x & +xy & + & y^2 & =0\\
lc_2': & & y & & & -t & -t^2 & & & + & y^2 & =0\\
lc_3': & & & -z & -z^2 & & & +x & +xy & + & y^2 & =0\\
lc_4': & & & & & -t & -t^2 & +x & +xy & +2 & y^2 & =0
\end{array}\right.}$$

The new system is solved in 240*ms!!*

$\boxed{\textit{Strategies are designed}}$

Let us consider again the previous problem. We begin with the linearized system:

$$\boxed{\left\{\begin{array}{lcccccccccl} lc_1: & x & +y & & & +u_1 & +u_2 & +u_3 & & & =0 \\ lc_2: & x & & & +t & +u_1 & +u_2 & & +u_4 & & =0 \\ lc_3: & & y & +z & & +u_1 & & & & +u_5 & =0 \\ lc_4: & x & & +z & & +u_1 & +u_2 & +u_3 & & +u_5 & =0 \end{array}\right.}$$

Pivot. (lc_3, u_5)

A triangularization method

Strategies

First step of elimination

$$\left\{\begin{array}{lcccccccccccc} \boldsymbol{lc_3}: & & y & +z & & +u_1 & & & & +\boldsymbol{u_5} & = & 0 \\ lc_1: & x & +y & & & +u_1 & +u_2 & +u_3 & & & = & 0 \\ lc_2: & x & & & +t & +u_1 & +u_2 & & +u_4 & & = & 0 \\ lc_4': & -x & +y & & & & -u_2 & -u_3 & & & = & 0 \end{array}\right.$$

Control criterion: *controls the densification of the “linear” system*

User linear part: 0

Abstracted linear part: -2

Pivot. (lc_4', u_3)

Second step of elimination

$$\begin{cases} lc_3: & & y & +z & & +u_1 & & & & +u_5 & = & 0 \\ \boldsymbol{lc'_4}: & -x & +y & & & & -u_2 & -\boldsymbol{u_3} & & & = & 0 \\ lc'_1: & & 2y & & & +u_1 & & & & & = & 0 \\ lc_2: & x & & & +t & +u_1 & +u_2 & & +u_4 & & = & 0 \end{cases}$$

Control criterion: *controls the densification of the "linear" system*

User linear part: 0

Abstracted linear part: -1

Pivot. (lc'_1, u_1)

Third step of elimination

$$\left\{\begin{array}{lcccccccccc} lc_3: & & y & +z & & +u_1 & & & & +u_5 & = & 0 \\ lc'_4: & -x & +y & & & & -u_2 & -u_3 & & & = & 0 \\ \boldsymbol{lc'_1}: & & 2y & & & +\boldsymbol{u_1} & & & & & = & 0 \\ lc'_2: & -x & +2y & & -t & & -u_2 & & -u_4 & & = & 0 \end{array}\right.$$

Control criterion: *controls the densification of the "linear" system*

User linear part: $+1$

Abstracted linear part: -1

End of the elimination stage.

A triangularization method

Strategies

Triangularized system

$$
\boxed{\left\{\begin{array}{lcccccccccc}
lc'_2: & -t & -x & -u_2 & & -u_4 & & & & +2y & = 0\\
lc'_4: & & -x & -u_2 & -u_3 & & & & & +y & = 0\\
lc_3: & & & & & & +u_5 & +z & +u_1 & +\,y & = 0\\
lc'_1: & & & & & & & & +u_1 & +2y & = 0
\end{array}\right.}
$$

Concretization: *nonlinear terms are restored, using the abstracted system*

Concretization phase

$$\left\{\begin{array}{llllllllllll} c_1' : & -t & -x & -xy & & -t^2 & & & & +2y & = & 0 \\ c_2' : & & -x & -xy & -y^2 & & & & & +y & = & 0 \\ c_3' : & & & & & & z^2 & +z & +x^2 & +\,y & = & 0 \\ c_4' : & & & & & & & & x^2 & +2y & = & 0 \end{array}\right.$$

Post-processing: *simplification of the system using specific constraints*

$$x_i = f(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$$

$c_4' : \ -2y = x^2$ *is eligible for post-processing*

$-2y$ *is substituted for* x^2

Post-processing: $x^2 = -2y$

$$
C_T : \left\{
\begin{array}{rcccccccccc}
c_1^T : & -t & -x & -xy & & -t^2 & & & & +2y & = 0 \\
c_2^T : & & -x & -xy & -y^2 & & & & & +y & = 0 \\
c_3^{T'} : & & & & & & z^2 & +z & & -\,y & = 0 \\
c_4^T : & & & & & & & & +x^2 & +2y & = 0
\end{array}
\right.
$$

Solving stage: 4 *solutions reached in* 10*ms!*

Tests and results

Bratu's problem.

$$x_{k-1} - 2x_k + x_{k+1} + h\exp(x_k) = 0, \quad 1 \leqslant k \leqslant n$$

defined over $[-10^8, +10^8]^n$*, with* $x_0 = x_{n+1} = 0$ *and* $h = \frac{1}{(n+1)^2}$.

Tests and results

Bratu's problem.

$$x_{k-1} - 2x_k + x_{k+1} + h\exp(x_k) = 0, \quad 1 \leqslant k \leqslant n$$

defined over $[-10^8, +10^8]^n$*, with* $x_0 = x_{n+1} = 0$ *and* $h = \frac{1}{(n+1)^2}$*.*

The initial problem is transformed as follows into a **dense triangular system***:*

$$-(k+1)x_k + (k+2)x_{k+1} + h\sum_{i=1}^{k} i\exp(x_i) = 0, \quad 1 \leqslant k \leqslant n$$

Bratu's problem.

$$x_{k-1} - 2x_k + x_{k+1} + h\exp(x_k) = 0, \quad 1 \leqslant k \leqslant n$$

defined over $[-10^8, +10^8]^n$, *with* $x_0 = x_{n+1} = 0$ *and* $h = \frac{1}{(n+1)^2}$.

Problem	v	Initial Pb.		Triangul. Pb.	
		Time	Sol.	Time	Sol.
Bratu	7	1.10	3	0.60	4
	8	0.70	2	0.10	2
	10	2.30	2	0.10	2
	13	20.50	6	0.10	2
	14	46.40	11	0.20	2
	15	94.40	12	0.20	2

Recap'

Symbolic pre-processing of constraint systems is efficient:

Triangularization through abstraction and elimination

Related work

- **triangularization methods that cross even more sub-expressions**
- **elimination based on the tree representation**

Research directions

The dependency problem

The locality of reasonings

A unifying framework for soft constraints

Distributed constraints: speculations

3. Soft constraints

A unifying framework

Interval solving process

Applications

A unifying framework

Motivation.

• *providing a general framework, allowing to model explicitly the required flexibility*

• *exploiting the properties of well-known algorithms for classical problems*

(i.e., $\neq$ soft)

Framework.

• *based on distances/flexibility measures: the smallest flexibility is sought*
• *also integrates an order over the constraints*

A unifying framework

soft constraints

Given a constraint c *defined over* $E \subset \mathbb{R}^n$*, a soft constraint resulting from* c *is defined by a pair*

$$\widehat{\boldsymbol{c}} \;=\; (\boldsymbol{c}, \boldsymbol{d})$$

where: $\boldsymbol{d}$ *defines a distance between* $\boldsymbol{c}$ *and the elements of the search space.*

Given a constraint c defined over $E \subset \mathbb{R}^n$, a soft constraint resulting from c is defined by a pair

$$\widehat{\boldsymbol{c}} = (\boldsymbol{c}, \boldsymbol{d})$$

where: $\boldsymbol{d}$ *defines a distance between* $\boldsymbol{c}$ *and the elements of the search space.*

Properties of $\boldsymbol{d}$: *increasing function s.t.* $d(0) = 0$.

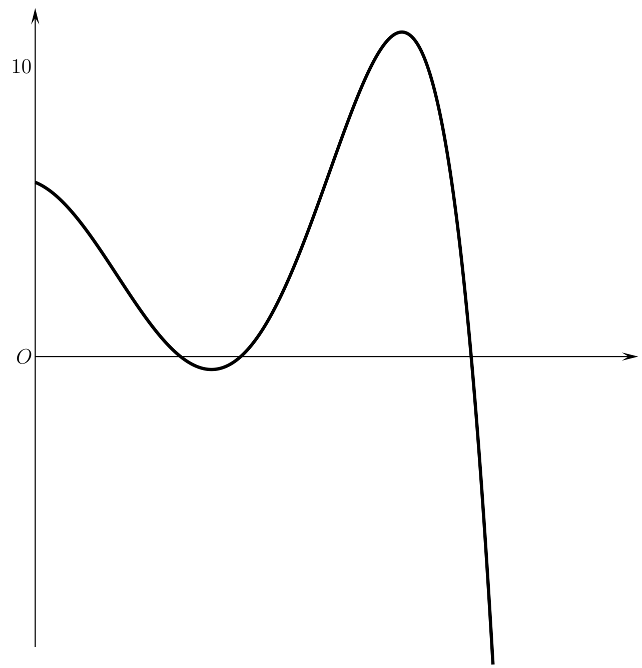
interprets the rough distance to c.

Given a constraint c defined over $E \subset \mathbb{R}^n$, a soft constraint resulting from c is defined by a pair

$$\widehat{c} = (c, d)$$

where: d defines a distance between c and the elements of the search space.

Properties of d:

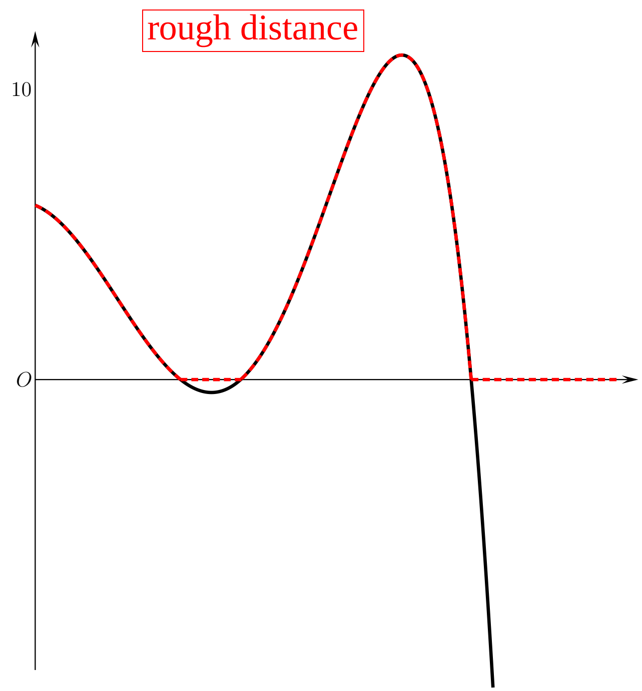


Given a constraint c defined over $E \subset \mathbb{R}^n$, a soft constraint resulting from c is defined by a pair

$$\widehat{\boldsymbol{c}} = (\boldsymbol{c}, \boldsymbol{d})$$

where: $\boldsymbol{d}$ defines a distance between $\boldsymbol{c}$ and the elements of the search space.

Properties of $\boldsymbol{d}$:

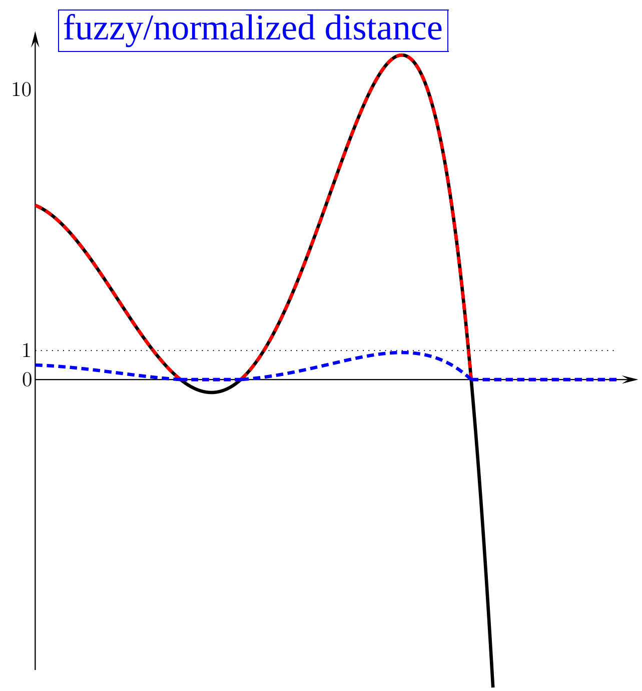


Given a constraint c defined over $E \subset \mathbb{R}^n$, a soft constraint resulting from c is defined by a pair

$$\widehat{\boldsymbol{c}} = (\boldsymbol{c}, \boldsymbol{d})$$

where: $\boldsymbol{d}$ *defines a distance between* $\boldsymbol{c}$ *and the elements of the search space.*

Properties of $\boldsymbol{d}$:



Given a constraint c defined over $E \subset \mathbb{R}^n$, a soft constraint resulting from c is defined by a pair

$$\widehat{c} = (c, d)$$

where: d *defines a distance between* c *and the elements of the search space.*

Properties of d: *increasing function s.t. $d(0) = 0$.*

interprets the rough distance to c.

instanciations $\leftrightarrow$ quality: to be maximized.

Given a constraint c defined over $E \subset \mathbb{R}^n$, a soft constraint resulting from c is defined by a pair

$$\widehat{\boldsymbol{c}} = (\boldsymbol{c}, \boldsymbol{d})$$

where: $\boldsymbol{d}$ defines a distance between $\boldsymbol{c}$ and the elements of the search space.

Solution set of $\widehat{c} =$ *the closest to c (w.r.t. d) subset of E.*

$$= \{x \in E \mid \forall y \in E, d(x,c) \leqslant d(y,c)\}$$

Given a constraint c defined over $E \subset \mathbb{R}^n$, a soft constraint resulting from c is defined by a pair

$$\widehat{\boldsymbol{c}} = (\boldsymbol{c}, \boldsymbol{d})$$

where: $\boldsymbol{d}$ *defines a distance between* $\boldsymbol{c}$ *and the elements of the search space.*

Solution set of $\widehat{c}$ $=$ *the closest to c (w.r.t. d) subset of E.*

$$= \{x \in E \mid \forall y \in E, d(x,c) \leqslant d(y,c)\}$$

Preferences over the search space

Given a CSP $C = \{c_1, \cdots, c_p\}$ defined over $E \subset \mathbb{R}^n$, a soft CSP resulting from C is defined by a tuple

$$\widehat{C} = (\boldsymbol{C}, \boldsymbol{d}, \boldsymbol{D}, \succ)$$

where: $\boldsymbol{D}$ is a set of distances corresponding to each constraint $\boldsymbol{c_i}$

$\boldsymbol{d}$ is a operator combining the values of the distances of $\boldsymbol{D}$

$\succ$ is an order over the set of constraints.

Given a CSP $C = \{c_1, \cdots, c_p\}$ *defined over* $E \subset \mathbb{R}^n$*, a soft CSP resulting from* C *is defined by a tuple*

$$\widehat{C} = (\boldsymbol{C}, \boldsymbol{d}, \boldsymbol{D}, \succ)$$

Properties of $\boldsymbol{d}$: *defined over* $(\mathbb{R}^+)^p$ = *combination of distances*

the same as those of each distance to a single constraint

increasing function w.r.t. each parameter

Given a CSP $C = \{c_1, \cdots, c_p\}$ *defined over* $E \subset \mathbb{R}^n$*, a soft CSP resulting from* C *is defined by a tuple*

$$\widehat{C} \; = \; (\boldsymbol{C}, \boldsymbol{d}, \boldsymbol{D}, \succ)$$

Properties of $\boldsymbol{d}$: *defined over* $(\mathbb{R}^+)^p$ *= combination of distances*

the same as those of each distance to a single constraint:

increasing function w.r.t. each parameter

Remark concerning $\boldsymbol{D}$: *up to now, all the distances are of the same type*

= commensurability problem

Given a CSP $C = \{c_1, \cdots, c_p\}$ *defined over* $E \subset \mathbb{R}^n$*, a soft CSP resulting from* C *is defined by a tuple*

$$\widehat{C} = (\boldsymbol{C}, \boldsymbol{d}, \boldsymbol{D}, \succ)$$

Properties of $\boldsymbol{d}$**:** *defined over* $(\mathbb{R}^+)^p$ *= combination of distances*

the same as those of each distance to a single constraint:

increasing function w.r.t. each parameter

Remark concerning $\boldsymbol{D}$**:** *up to now, all the distances are of the same type*

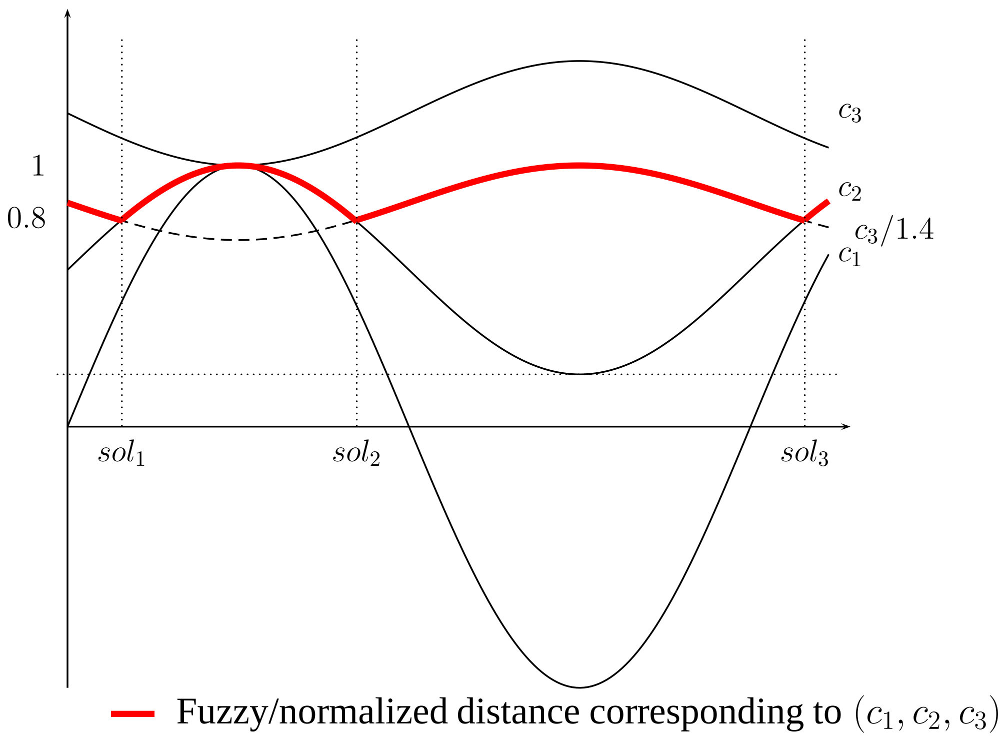
= commensurability problem

Order $\succ$**:** *establish the order instanciations are to satisfy*

- $\star$ *may be trivial*
- *otherwise, states new constraints* $\boldsymbol{C}_{\succ}$

Given a CSP $C = \{c_1, \cdots, c_p\}$ defined over $E \subset \mathbb{R}^n$, a soft CSP resulting from C is defined by a tuple

$$\widehat{C} = (\boldsymbol{C}, \boldsymbol{d}, \boldsymbol{D}, \succ)$$

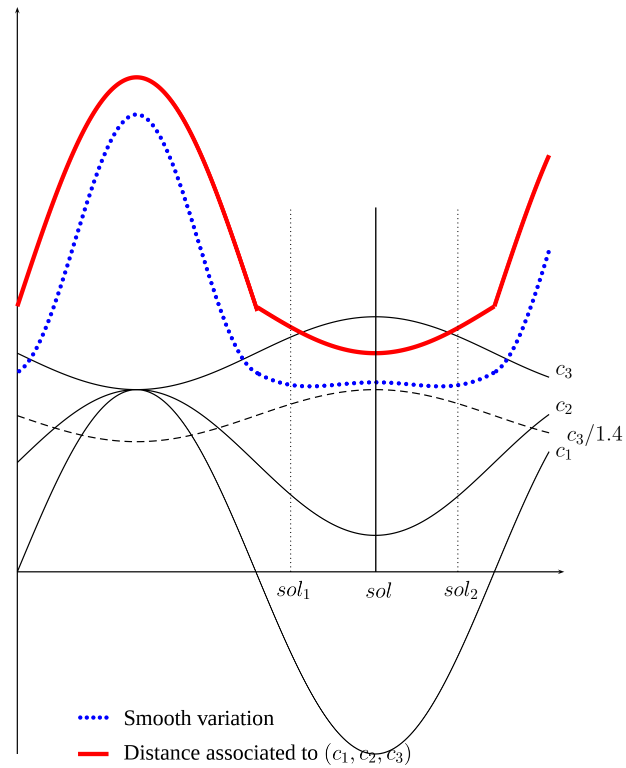


A unifying framework

soft CSP

Given a CSP $C = \{c_1, \cdots, c_p\}$ defined over $E \subset \mathbb{R}^n$, a soft CSP resulting from C is defined by a tuple

$$\widehat{C} = (\boldsymbol{C}, \boldsymbol{d}, \boldsymbol{D}, \succ)$$



Given a CSP $C = \{c_1, \cdots, c_p\}$ *defined over* $E \subset \mathbb{R}^n$*, a soft CSP resulting from* C *is defined by a tuple*

$$\widehat{C} = (\boldsymbol{C}, \boldsymbol{d}, \boldsymbol{D}, \succ)$$

Solution set of $\widehat{C}$ $=$ *the closest to* C *(w.r.t.* d*) subset of* E *satisfying* $C_\succ$.

$= \{x \in E \mid C_\succ$ *holds on* x

and $\forall y \in E, d(x, C) \leqslant d(y, C)\}$

Given a CSP $C = \{c_1, \cdots, c_p\}$ defined over $E \subset \mathbb{R}^n$, a soft CSP resulting from C is defined by a tuple

$$\widehat{C} = (\boldsymbol{C}, \boldsymbol{d}, \boldsymbol{D}, \succ)$$

Solution set of $\widehat{C}$ $=$ *the closest to C (w.r.t. d) subset of E satisfying $C_{\succ}$.*

$$= \{x \in E \mid C_{\succ} \text{ holds on } x \text{ and } \forall y \in E, d(x, C) \leqslant d(y, C)\}$$

For instance, preferences over the constraints establish an order over the satisfaction/violation of the constraints: $C_{\succ}$ expresses this order.

On the other hand, when trivial, $C_{\succ}$ holds on any $x \in E$

Preferences over the search space and over the constraints

Solving process

Given a **soft CSP** $\widehat{C} = (C, d, D, \succ)$ *defined over* $E \subset \mathbb{R}^n$, *the solution set of* $\widehat{C}$ *is the solution set of the following* **hard problem***:*

$$\boxed{\begin{array}{l} \min\limits_{x \in E} d(d_1(x), \cdots, d_p(x)) \\ \text{s.t. } x \text{ satisfies } C_{\succ} \end{array}}$$

Solving process

Given a **soft CSP** $\widehat{C} = (C, d, D, \succ)$ *defined over* $E \subset \mathbb{R}^n$*, the solution set of* $\widehat{C}$ *is the solution set of the following* **hard problem***:*

$$\boxed{\begin{array}{l} \min\limits_{x \in E} d(d_1(x), \cdots, d_p(x)) \\ \text{s.t. } x \text{ satisfies } C_{\succ} \end{array}}$$

Interval solving process: *distance functions are extended in the usual way.*

$\dagger$ *Pbs. with normalized distances:*

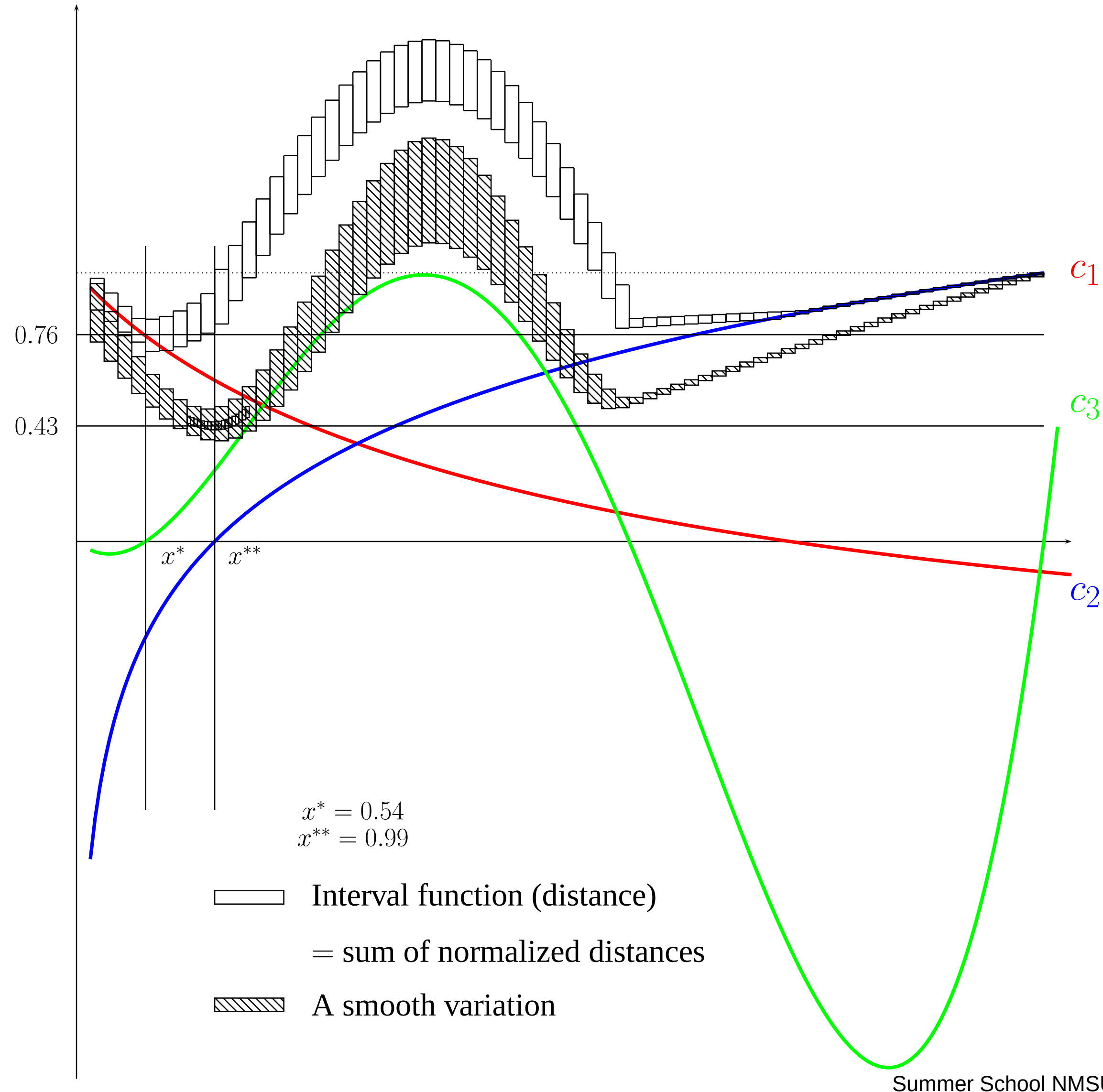
1. *maximum value of the rough distance = another optimization process!*

   $\rightsquigarrow$ **interval upper bound**

2. *but may be* ∞

   $\rightsquigarrow$ **variation** *of the normalized distance*

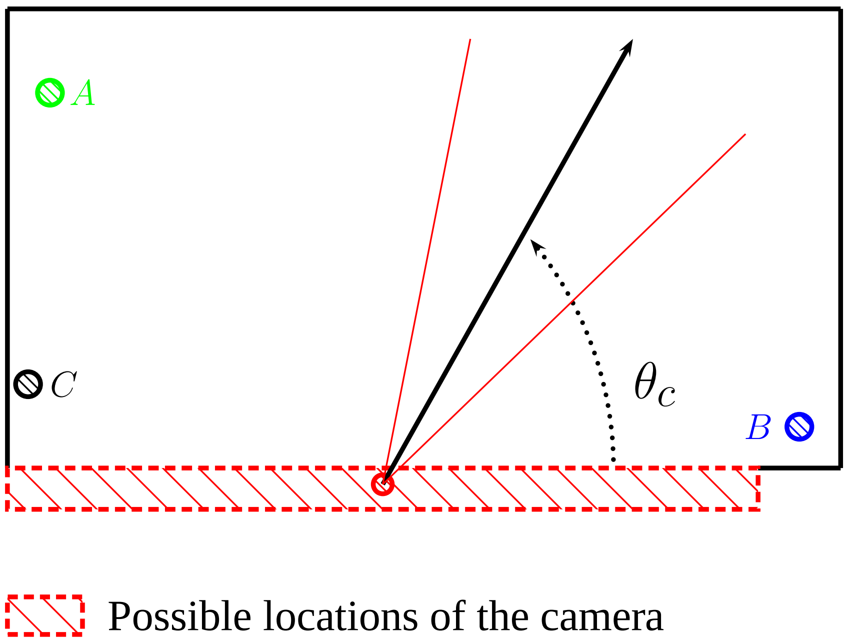
Solving process



Applications

Camera positioning problem.

- *given a camera, find a position and angle allowing to visualize given objects*
- *inconsistent (over-constrained) problem: solved using several soft models*



Applications

Camera positioning problem.

- *given a camera, find a position and angle allowing to visualize given objects*
- *inconsistent (over-constrained) problem: solved using several soft models*

Results.

- *1. soft positioning are easily reached using an optimization process*
- *2. some positioning are useless w.r.t. the camera problem:*

  no object is in the camera's scope

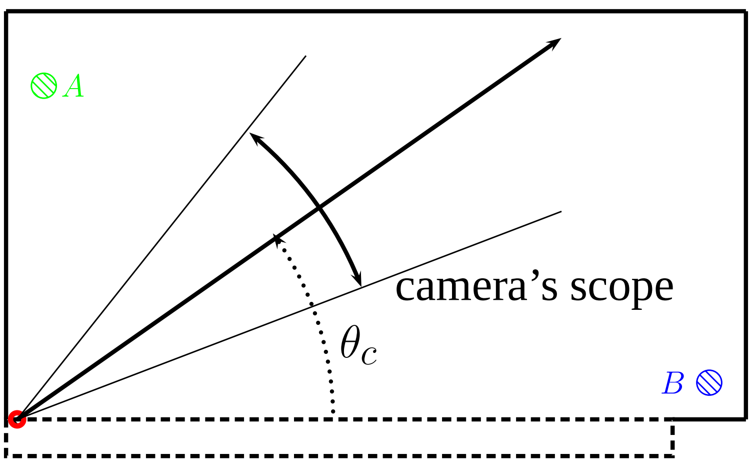
Applications

Camera positioning problem.

- *given a camera, find a position and angle allowing to visualize given objects*
- *inconsistent (over-constrained) problem: solved using several soft models*

Results.

- *1. soft positioning are easily reached using an optimization process*
- *2. some positioning are useless w.r.t. the camera problem:*



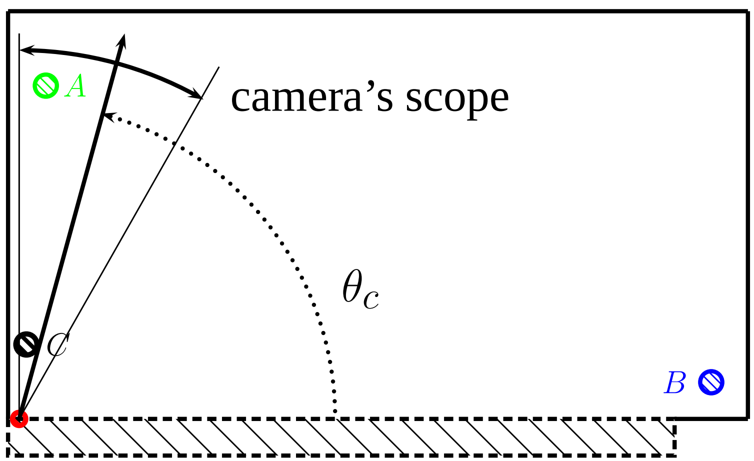
Applications

Camera positioning problem.

- *given a camera, find a position and angle allowing to visualize given objects*
- *inconsistent (over-constrained) problem: solved using several soft models*

Results.

- *1. soft positioning are easily reached using an optimization process*
- *2. some positioning are useless w.r.t. the camera problem:*



Applications

Camera positioning problem.

- *given a camera, find a position and angle allowing to visualize given objects*
- *inconsistent (over-constrained) problem: solved using several soft models*

Results.

- *1. soft positioning are easily reached using an optimization process*
- *2. some positioning are useless w.r.t. the camera problem:*

Conclusions.

- *soft constraints allowing violation degrees are useless when violated constraints are meaningless*
- *for specific problems, a priori knowledge is crucial to guarantee exploitable solutions*
- *the user is essential in the modelling stage*

Research directions

The dependency problem

The locality of reasonings

A unifying framework for soft constraints

Distributed constraints: speculations

What is a speculation?

Speculation *= a hypothesis that has been formed by speculating or conjecturing (usually with little hard evidence)*

What is a speculation?

Speculation *= a hypothesis that has been formed by speculating or conjecturing (usually with little hard evidence)*

e.g., ”speculations about the outcome of the election”

What is a speculation? (2)

Examples:

What is a speculation? (2)

Examples:

- you invite people at home, and you give them a choice among possible dates, but they don't reply immediately when they can come

What is a speculation? (2)

Examples:

- you invite people at home, and you give them a choice among possible dates, but they don't reply immediately when they can come
  - *instead of waiting for their replies, you may have a clue about the chosen date, and begin to prepare the party, based on this speculation*

What is a speculation? (2)

Examples:

- you invite people at home, and you give them a choice among possible dates, but they don't reply immediately when they can come
  - *instead of waiting for their replies, you may have a clue about the chosen date, and begin to prepare the party, based on this speculation*
- you plan a trip and ask Rose to take care about this, but you may not specify all your preferences: *e.g., only the date, and destination*

What is a speculation? (2)

Examples:

- you invite people at home, and you give them a choice among possible dates, but they don't reply immediately when they can come
  - *instead of waiting for their replies, you may have a clue about the chosen date, and begin to prepare the party, based on this speculation*
- you plan a trip and ask Rose to take care about this, but you may not specify all your preferences: *e.g., only the date, and destination*
  - *the travel agency will not wait until you specify your time preferences to begin and look for air fares*

Why perform speculative computations?

- Most studies on multi-agent systems (MAS) assume that the **communication** between agents **is guaranteed**

Why perform speculative computations?

- Most studies on multi-agent systems (MAS) assume that the **communication** between agents **is guaranteed**
- When an agent asks a question to another one, the **process** depending on the answer **is suspended** until some response is sent

Why perform speculative computations?

- Most studies on multi-agent systems (MAS) assume that the **communication** between agents **is guaranteed**
- When an agent asks a question to another one, the **process** depending on the answer **is suspended** until some response is sent

However...

Why perform speculative computations?

- Most studies on multi-agent systems (MAS) assume that the **communication** between agents **is guaranteed**
- When an agent asks a question to another one, the **process** depending on the answer **is suspended** until some response is sent

However...

- In real settings (e.g. internet), communication may fail

Why perform speculative computations?

- Most studies on multi-agent systems (MAS) assume that the **communication** between agents **is guaranteed**
- When an agent asks a question to another one, the **process** depending on the answer **is suspended** until some response is sent

However...

- In real settings (e.g. internet), communication may fail
- Agents may take time to send back a reply

What kind of problems can be considered?

- In the Constraint Logic Programming (CLP) world
e.g., organize a meeting, and determine when, where, and with whom

$$organize(large\_room, [a,b,c], D) \leftarrow meeting([a,b,c], D)$$
$$organize(small\_room, [X,Y], D) \leftarrow meeting([X,Y], D)$$
$$meeting([a,b], D) \leftarrow available(a,D), available(b,D), not\_available(c,D)$$
$$meeting([b,c], D) \leftarrow not\_available(a,D), available(b,D), available(c,D)$$
$$meeting([a,c], D) \leftarrow available(a,D), not\_available(b,D), available(c,D)$$
$$meeting([a,b,c], D) \leftarrow available(a,D), available(b,D), available(c,D)$$
$$\left.\begin{array}{rcl} available(P,D) & \leftarrow & free(P)@D \\ not\_available(P,D) & \leftarrow & busy(P)@D \end{array}\right\} \text{questions sent to agents}$$

$$?-organize(R,L,D).$$

What kind of problems can be considered?

- In the Constraint Logic Programming (CLP) world
- In the Constraint Solving world
  e.g., determine the geographical zone a robot can cover

$$(x - x_0)^2 + (y - y_0)^2 \leqslant (t_0 \cdot s_0)^2$$
$$x,\ y,\ \in\ [-10^8, 10^8]$$

$$\left.\begin{array}{l} x_0 = location(X) \\ y_0 = location(Y) \\ s_0 = speed(S) \\ d_0 = duration(D) \end{array}\right\} \text{questions sent to agents and transmitted to sensors}$$

SCP in a Master-Slave environment

Basic idea [Satoh, Prima 2003]:

- The program *(constraint problem, denoted by P)* is centralized at the master's level *(denoted by M)*

SCP in a Master-Slave environment

Basic idea [Satoh, Prima 2003]:

- The program *(constraint problem, denoted by P)* is centralized at the master's level *(denoted by M)*
- M begins to run the program / solve the constraint system

SCP in a Master-Slave environment

Basic idea [Satoh, Prima 2003]:

- The program *(constraint problem, denoted by P)* is centralized at the master's level *(denoted by M)*
- M begins to run the program / solve the constraint system
- When specific information is needed:

SCP in a Master-Slave environment

Basic idea [Satoh, Prima 2003]:

- The program *(constraint problem, denoted by P)* is centralized at the master's level *(denoted by M)*
- M begins to run the program / solve the constraint system
- When specific information is needed: *e.g.,*
  - *is person* a *available on day D?* $free(a)@D$
  - *where is the robot located?* $x_0 = location(X)$, $y_0 = location(Y)$
  - *etc.*

SCP in a Master-Slave environment

Basic idea [Satoh, Prima 2003]:

- The program *(constraint problem, denoted by P)* is centralized at the master's level *(denoted by M)*
- M begins to run the program / solve the constraint system
- When specific information is needed: M asks a slave S the corresponding question

SCP in a Master-Slave environment

- Before S answers, M continue the processing of P with some default value/constraint δ:

SCP in a Master-Slave environment

- Before S answers, M continue the processing of P with some default value/constraint δ: *e.g.,*
  - $\leftarrow D \in \{1,2\} || free(a)@D$
  - $x_0 \in [1,100]$, $y_0 \in [10,25]$

SCP in a Master-Slave environment

- Before S answers, M continue the processing of P with some default value/constraint δ: no time is wasted

SCP in a Master-Slave environment

- Before S answers, M continue the processing of P with some default value/constraint δ: no time is wasted
- When answers α come from S, M updates or reinforces its belief depending on whether:
  - α *entails* δ*:* $\alpha \subset \delta$
  - α *contradicts* δ*:* $\alpha \cap \delta = \varnothing$
  - α *is consistent with* δ *but does not entail it:* $\alpha \cap \delta \neq \varnothing$ *but* $\alpha \not\subset \delta$

MA belief revision in the case of yes/no questions

- What is speculative computation with MA belief revision?

MA belief revision in the case of yes/no questions

- What is speculative computation with MA belief revision?
  - *each agent can perform speculative computations*

MA belief revision in the case of yes/no questions

- What is speculative computation with MA belief revision?
  - *each agent can perform speculative computations*
  - *therefore, answers from slaves may not be certified: they are now likely to be default too*

MA belief revision in the case of yes/no questions

- Speculative computations with MA belief revision for yes/no questions [Satoh, AAMAS'03]

MA belief revision in the case of yes/no questions

- Speculative computations with MA belief revision for yes/no questions [Satoh, AAMAS'03]
  - *when S sends an answer δ_s, it may be a default S uses, instead of the actual certified answer from a person, or a sensor*

MA belief revision in the case of yes/no questions

- Speculative computations with MA belief revision for yes/no questions [Satoh, AAMAS'03]
  - *when S sends an answer* δ_s*, it may be a default S uses, instead of the actual certified answer from a person, or a sensor*
  - *therefore: different process management when answers come*

MA belief revision in the case of yes/no questions (2)

- There are only two possible cases:
  - *Entailment: default = answer*
  - *Contradiction: default = $\neg$ answer*

MA belief revision in the case of yes/no questions (2)

- There are only two possible cases:
  - *Entailment: default = answer*
  - *Contradiction: default =* $\neg$ *answer*

- When certified information comes, same situation as in [Satoh, Prima 2003]

MA belief revision in the case of yes/no questions (2)

- There are only two possible cases:
  - *Entailment: default = answer*
  - *Contradiction: default = $\neg$ answer*

- When certified information comes, same situation as in [Satoh, Prima 2003]

- Otherwise, complementary processes must not be killed:
  - *in case later answers contradicts the current one*
  - *instead, they are recorded*

Recap on speculative computations in MA systems

- Frameworks for speculative computations exist
- In master-slave, we can perform speculative constraint processing
- In general hierarchical systems, all agents can perform spec. computations in the case of yes/no questions

How to improve this?

Make it possible to:

- solve general constraints *(or ask more general questions)*...

How to improve this?

Make it possible to:

- solve general constraints *(or ask more general questions)*...
- ... in a general hierarchical MA system...

How to improve this?

Make it possible to:

- solve general constraints *(or ask more general questions)*...
- ... in a general hierarchical MA system...
- ... where all agents are enabled to perform speculations.

Outline of the presentation

- *Continuous constraints: definitions and solving process*
- *An example of under and over-constrained problems*
- *Important notions*
- *Some research directions*
- *Conclusion*

Outline of the presentation

- *Continuous constraints: definitions and solving process*
- *An example of under and over-constrained problems*
- *Important notions*
- *Some research directions*
- Conclusion

Conclusion

Now you know about:

- *Continuous constraints*
- *Variations: optimization, soft constraints*
- *Some issue about distributed constraint solving*
- *and their limitations / open problems*

You're ready to:

- *find new methods to address them*

Some ideas for doing this

Dependency problems: *Extension of factorization schemes*

- *to more generalized rules: elementary scheme greater than binomials*
- *to more general terms ($\sin$, $\cos$), integrated in schemes (more in-depth parsing)*
- *to more general terms: linearization, loss of accuracy needs to be evaluated*

Locality of Reasonings: *Cooperation of linearization processes*

or: Class of suitable problems

Soft constraints: *More expressivity*

Speculations: *Other social group organizations*

The end

Thank you for your attention

QUESTIONS?

Martine Ceberio

mceberio@utep.edu

www.constraintsolving.com

http://www.martineceberio.fr

University of Texas at El Paso