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*
• Enumeration is not an option...
• 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)

\[ S \leftarrow \text{Initial domain} \quad // \quad S \text{ is the store of domains to be visited} \]
\[ \text{Solutions} \leftarrow \emptyset \]
\[ \text{while } (S \neq \emptyset) \{ \]
\[ \text{take } D \text{ out of } S \quad // \quad \text{usually } D \text{ is the first available domain} \]
\[ D' \leftarrow \text{narrow}(D,C) \quad // \quad \text{apply a consistency technique on } D \]
\[ \text{if } (D' \neq \emptyset) \text{ and } (D' \text{ is still too large}) \text{ then} \]
\[ \text{split}(D',D_1,D_2) \quad // \quad \text{splitting in halves is not compulsory} \]
\[ S \leftarrow S \cup \{D_1,D_2\} \]
\[ \text{else store } D' \text{ in } \text{Solutions} \]
\[ \} \quad \]
\[ \text{return } \text{Solutions} \quad // \quad \text{What does } \text{Solutions} \text{ contain?} \]
Solving algorithm: narrow($D,C$)

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

\[
\begin{align*}
S &\leftarrow \{c_1, \ldots, c_p\} \quad \text{ // } S \text{ is the store of constraints, no duplicates} \\
\text{while } (S \neq \emptyset) \{ \\
& \text{take } c \text{ out of } S \quad \text{ // usually } c \text{ is the first available constraint} \\
& \text{for all } i \in \{1, \ldots, n\} \{ \\
& \quad D'_i \leftarrow \text{consistency}(D_i, c) \\
& \quad \text{ // apply a consistency technique on } D_i \text{ w.r.t. } c \\
& \quad \text{if } (D'_i = \emptyset) \text{ then return } \emptyset \\
& \quad \text{if } (D'_i \neq D_i) \text{ then} \\
& \quad \quad S \leftarrow S \cup \{c_j, \ j \in J\} \\
& \quad \quad \text{ // } c_j \text{ are the constraints that share variable } i \text{ with } c \\
& \text{\} } \\
\text{return } \times_{1 \leq i \leq n} D'_i \quad \text{ // What is } \times_{1 \leq i \leq n} D'_i? 
\end{align*}
\]
• 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
Problem to be solved: $y(t) = f(x,t)$
Problem to be solved: $y(t) = f(x, t)$

\[ \downarrow \]

the radioactive decay of radium

[Pierre and Marie Curie (1898)]

\[ y(t) = \exp^{-xt} \]
Problem to be solved: \( y(t) = f(x, t) \)

**Knowing:** \( y, t, \) the model \( f \)

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

**Find:** parameter \( x \)

---

\[
\text{measured data: } \tilde{y}_i \text{ of } f(x, t_i) \text{ at } t_i, \, i = 1, \ldots, 9
\]

model \( f : (x, t) \mapsto \frac{1}{(t+x)^{1.1}} \)

---

\[
\begin{array}{cccccccc}
& 0 & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 \\
\times & & & \times & & \times & & \times & & \times & \\
\end{array}
\]
Example (1/3)

Problem to be solved: \( y(t) = f(x, t) \)

**Knowing:** \( y, t, \) the model \( f \)

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

**Find:** parameter \( x \)

**Classical solving method:** least squares

\[
\min_x \sum_{i=1}^{n} (\tilde{y}_i - f(x, t_i))^2
\]
Taking inaccuracy into account

Intervals $[\hat{y}_i - e_i, \hat{y}_i + e_i]$ at given $t_i$, $i = 1, \ldots, 9$

Constraint system to be solved:
$$f(x, t_i) \in [\hat{y}_i - e_i, \hat{y}_i + e_i], i = 1, \ldots, 9$$
Taking *inaccuracy* into account

\[ \text{Intervals } [\hat{y}_i - e_i, \hat{y}_i + e_i] \text{ at given } t_i, \ i = 1, \ldots, 9 \]

\[ \text{Function } f(x, t) \text{ with parameter } x = 1 \]

\[ \downarrow \]

**NOT A SOLUTION!**
Taking **inaccuracy** into account

\[ \text{intervals } [\tilde{y}_i - e_i, \tilde{y}_i + e_i] \text{ at given } t_i, \ i = 1, \ldots, 9 \]

\[ \text{--- function } f(x, t) \text{ with parameter } x = 0.5 \]

\[ \downarrow \]

**A SOLUTION**

Even an infinite number of possible parameters \( x \)
Taking inaccuracy into account

\[
\mathcal{T} \text{ intervals } [\hat{y}_i - e_i, \hat{y}_i + e_i] \text{ at given } t_i, \ i = 1, \ldots, 9
\]

\[
\text{function } f(x, t) \text{ with parameter } x = 0.5
\]

\[
\downarrow
\]

A SOLUTION

Even an infinite number of possible parameters \(x\)

UNDER-CONSTRAINED PROBLEM
Taking inaccuracy into account

**Under-constrained problem**

Definition of an appropriate criterion to be optimized

*i.e.*, discrimination over the solution set
Taking inaccuracy into account

Under-constrained problem

Definition of an appropriate criterion to be optimized

i.e., discrimination over the solution set

≡

Constrained global optimization
Taking **erroneous measurements** into account

\[ \text{Intervals } [\hat{y}_i - e_i, \hat{y}_i + e_i] \text{ at given } t_i, \ i = 1, \ldots, 9 \]

**Inconsistent constraint system!**
no parameter value allows to satisfy the constraints
Taking erroneous measurements into account

Intervals $[\hat{y}_i - e_i, \hat{y}_i + e_i]$ at given $t_i, i = 1, \ldots, 9$

Inconsistent constraint system!
no parameter value allows to satisfy the constraints

OVER-CONSTRAINED PROBLEM
Taking erroneous measurements into account

Over-constrained problem

⇒

Need of solution ➞ weaker constraints  \textit{i.e.}, need for flexibility
Taking erroneous measurements into account

\[ \text{Over-constrained problem} \]

\[ \downarrow \]

Need of solution $\rightsquigarrow$ weaker constraints \textit{i.e.}, need for flexibility

\textit{Ex.} deletion of the measure at $t_5$
Taking **erroneous measurements** into account

**Over-constrained problem**

Need of solution $\leadsto$ weaker constraints \(i.e.,\) need for flexibility

*Ex.* deletion of the measure at \(t_5\), \(i.e.,\) deletion of a constraint
Taking **erroneous measurements** into account

**Over-constrained problem**

\[ \Downarrow \]

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

≡

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

$$x = \{ x \in \mathbb{R} \mid a \leq x \leq b \} \quad x = a \quad \overline{x} = b$$

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

$$x = \{x \in \mathbb{R} \mid a \leq x \leq b\} \quad \underline{x} = a \quad \overline{x} = b$$

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

Some useful notions.

Width of $x$:

$$w(x) = \overline{x} - \underline{x}$$

Interval hull of $\rho \subset \mathbb{R}$:

$$\text{Hull} (\rho) = [\inf \rho, \sup \rho] = \square \rho$$
Real interval arithmetic

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

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

$$x \diamond y = \square \{x \diamond y \mid x \in x, \ y \in y \}$$
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 $\diamond \in \{+, -, \times, /\}$

$$x \diamond y = \Box \{ x \diamond y \mid x \in x, y \in y \}$$

Properties.

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

$\implies$ interval arith. is expression-dependent

$= \text{the DEPENDENCY PROBLEM}$
Definition 3 (Interval arithmetic (IA)). Usual arithmetic-like arithmetic where handled items are intervals (and no longer reals)

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

$$x \diamond y = \Box \{ x \diamond y \mid x \in x, \ y \in y \}$$

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

$= \text{the DEPENDENCY PROBLEM}$
Interval extensions

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

→ used for the evaluation of the ranges of functions
IA Principle: provides outer approximations of real quantities being looked for
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 \( \phi \) is an interval extension of \( f \) provided that:

\[
\forall x \subset \mathbb{R}^n, \{ f(x) \mid x \in x \cap E \} \subset \phi(x).
\]
**Interval extensions**

**IA Principle:** provides outer approximations of real quantities being looked for 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 \( \phi \) is an interval extension of \( f \) provided that:

\[
\forall x \subset \mathbb{R}^n, \{ f(x) | x \in x \cap E \} \subset \phi(x).
\]

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

- **rough extension:** \( \phi_f : x \mapsto [-\infty, +\infty] \) totally useless
- **ideal extension:** \( \phi_f : x \mapsto \Box \{ f(x) | x \in x \} \) extremely rare
- **natural extension:** \( \phi_f : x \mapsto f(x) \) syntactic interval extension
**Definition 1** (Unconstrained and constrained global optimization).

Function $f$ to be minimized s.t. $c$ be satisfied

- **Local minima:** $\{x_0, x_1, x_2, x_3, x_4, x_5, x_6, x^*_1, x^*_2\}$
- **Global minima:** $\{x^*_1, x^*_2\}$

Solution set of constraint $c$

- Function $h$ defining $c$: $h \geq 0$
Optimality conditions [Fritz, 1948] [Hiriart-Urruty, 1995&1996]

- Optimization problem \(\rightarrow\) constraint satisfaction problem
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 $\leadsto$ constraint satisfaction problem
  
  ex. for unconstrained optimization, slope $= 0$
  
  $\leadsto$ not necessarily an optimum, nor a global one ({except if the problem is convex})
  
  $\leadsto$ 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 $\leadsto$ constraint satisfaction problem

Penalty-based methods [Joines & Houck, 1994] [Michalewicz & al., 1995&1996]
  • Constrained optimization problem $\leadsto$ unconstrained optimization problem
Optimization: solving methods (1/2)

**Optimality conditions** [Fritz, 1948] [Hiriart-Urruty, 1995&1996]
- Optimization problem \( \leadsto \) constraint satisfaction problem

**Penalty-based methods** [Joines & Houck, 1994] [Michalewicz & al., 1995&1996]
- Constrained optimization problem \( \leadsto \) unconstrained optimization problem

\[
\begin{align*}
O(f) & + h_1(x) = 0 \\
O(f) + h_2(x) & = 0
\end{align*}
\]

Initial problem

First iteration

Iteration \( n^k \)

where \( c \cdot h = 0 \)

\[
f + h^2
\]

\[
f + 50h^2
\]
Optimization: solving methods (1/2)

Optimality conditions [Fritz, 1948] [Hiriart-Urruty, 1995&1996]
- Optimization problem $\leadsto$ constraint satisfaction problem

Penalty-based methods [Joines & Houck, 1994] [Michalewicz & al., 1995&1996]
- Constrained optimization problem $\leadsto$ unconstrained optimization problem
  $\leadsto$ number of iterations uncontrolled, optimization process to be performed
  $\leadsto$ 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 $\leadsto$ constraint satisfaction problem

**Penalty-based methods** [Joines & Houck, 1994] [Michalewicz & al., 1995&1996]
- Constrained optimization problem $\leadsto$ unconstrained optimization problem

**Meta-heuristics** [Goldberg, 1989] [Michalewicz, 1996]
- genetic, evolutionary algorithms, tabu search, simulated annealing, clustering, etc.

↓

**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
  - 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
- ineffective 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
  - 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
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
  - † ineffective 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
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.

Function $f$

boxes $(x_i, f(x_i)), i \in \{1, 2, 3, 4\}$
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.

Boxes \((x_i, f(x_i)), i \in \{1, 2, 3, 4\}\)
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.**

Function $f$

boxes $(x_i, f(x_i)), i \in \{1, 2, 3, 4\}$
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.

Function $f$

boxes $(x_i, f(x_i)), i \in \{1, 2, 3, 4\}$
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.

c1: \( y = x^2 \)
c2: \( y = 1 - x^4 \)
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.

\[
c_1 : y = x^2  \\
c_2 : y = 1 - x^4
\]
**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.**

\[ \begin{align*}
  c_1 & : y = x^2 \\
  c_2 & : y = 1 - x^4
\end{align*} \]
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.

\[
\begin{align*}
&c_1 : y = x^2 \\
&c_2 : y = 1 - x^4
\end{align*}
\]
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.

\[ \begin{align*}
  c_1 : & \quad y = x^2 \\
  c_2 : & \quad y = 1 - x^4
\end{align*} \]
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.

\[ c_1 : y = x^2 \]
\[ c_2 : y = 1 - x^4 \]
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**
Definition 6 (Soft constraint). Given a constraint \( c \) over a set of variables \( V \), defining a relation \( \rho \). A soft constraint \( \hat{c} \) resulting from \( c \) is a constraint defining a relation \( \hat{\rho} \) over \( V \) s.t. \( \rho \subset \hat{\rho} \).
Definition 6 (Soft constraint). Given a constraint $c$ over a set of variables $V$, defining a relation $\rho$. A soft constraint $\hat{c}$ resulting from $c$ is a constraint defining a relation $\hat{\rho}$ over $V$ s.t. $\rho \subset \hat{\rho}$.

Considering softness... some possible treatments

![Graph showing constraints $c_1$ and $c_2$]

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

Considering softness... some possible treatments

---

**Constraint** $c_1'$: $\left(x - \frac{7}{4}\right)^2 + y^2 \leq \left(\frac{9}{4}\right)^2$

**Constraint** $c_2'$: $x + 2y^2 \leq \frac{1}{2}$

Solution set $= \emptyset$ $\implies$ “Extended” solution set
Definition 6 (Soft constraint). Given a constraint $c$ over a set of variables $V$, defining a relation $\rho$. A soft constraint $\hat{c}$ resulting from $c$ is a constraint defining a relation $\hat{\rho}$ over $V$ s.t. $\rho \subset \hat{\rho}$.

Considering softness... some possible treatments

Preference order: $c_1 > c_2 > c_3$

- 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 $c_3 : (x - \frac{3}{2})^2 + (y - \frac{3}{2})^2 \leq 1$
Definition 6 (Soft constraint). Given a constraint $c$ over a set of variables $V$, defining a relation $\rho$. A soft constraint $\hat{c}$ resulting from $c$ is a constraint defining a relation $\hat{\rho}$ over $V$ s.t. $\rho \subset \hat{\rho}$.

Considering softness... some possible treatments

Preference order: $c_1 > c_2 > c_3$

- Constraint $c_1$: $(x - \frac{7}{4})^2 + y^2 \leq (\frac{3}{2})^2$
- Constraint $c_2$: $x + 2y^2 \leq -\frac{1}{4}$
- Constraint $c_3$: $(x - \frac{3}{2})^2 + (y - \frac{3}{2})^2 \leq 1$
Definition 6 (Soft constraint). Given a constraint \( c \) over a set of variables \( V \), defining a relation \( \rho \). A soft constraint \( \hat{c} \) resulting from \( c \) is a constraint defining a relation \( \hat{\rho} \) over \( V \) s.t. \( \rho \subset \hat{\rho} \).

Considering softness... some possible treatments

Preference order: \( c_1 > c_2 > c_3 \)

- 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 \( c_3 : (x - \frac{3}{2})^2 + (y - \frac{3}{2})^2 \leq 1 \)
Definition 6 (Soft constraint). Given a constraint $c$ over a set of variables $V$, defining a relation $\rho$. A soft constraint $\hat{c}$ resulting from $c$ is a constraint defining a relation $\hat{\rho}$ over $V$ s.t. $\rho \subset \hat{\rho}$.

Considering softness... some possible treatments

Preference order: $c_1 > c_2 > c_3$

- 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

- 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 instantiations *satisfying the hierarchy*

- preferences over the constraints and over the search space
Soft constraints: frameworks


★ preference over the constraints and over the search space

Partial CSP [Freuder & Wallace, 1995]

★ given $P$ to be solved, and some distance $d$, $(P, d)$ ordered set of problems

objective: determining the closest problem $P'$ and solving it
Soft constraints: frameworks

- preference over the constraints and over the search space

Partial CSP [Freuder & Wallace, 1995]
- given $P$ to be solved, and some distance $d$, $(P, d)$ 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

★ 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

★ 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)
  - instantiations are valued through the constraint valuation
  - objective: determining *the best quality instantiation*
  - equivalent to SCSP
Soft constraints: frameworks

  ★ 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)
  ★ 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
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
1. Independency of the occurrences.

2 occurrences of the same variable “behave” as if they were different variables
1. Independency of the occurrences.

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

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

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

\[ = x \times y, \text{ where } y = x \]
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]
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 \(\leadsto\) 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.

---

Addition

\[
f_1 : x \mapsto x^3
\]

\[
f_2 : x \mapsto -x^4
\]

\[
f : x \mapsto x^3 - x^4 = f_1(x) + f_2(x)
\]

---

Multiplication

\[
g_1 : x \mapsto x^2
\]

\[
g_2 : x \mapsto -(x - \frac{1}{2})^2 + \frac{1}{4}
\]

\[
dd : x \mapsto x^3 - x^4 = g_1(x) \times g_2(x)
\]
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 $\leadsto$ 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

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

2. Monotoicities.

occurrences are independent $\leadsto$ 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

  $\leadsto$ factorized forms
Interval Horner form. [Shih-Chieh, 1303][Horner, 1819][Stahl, 1995]

Let \( p \) be a polynomial defined by:

\[
 p(x) = a_0 + \sum_{i=1}^{n} a_i x^{\alpha_i}
\]

\[
 h_p(x) = a_0 + x^{d_1} \left( \cdots + x^{d_{n-1}}(a_{n-1} + a_n x^{d_n}) \cdots \right)
\]

Def. Intermediate polynomials:

\[
 \begin{align*}
 p_n(x) &= a_n \\
 p_i(x) &= x^{d_i+1} p_{i+1}(x) + a_i & i = n - 1, n - 2, \ldots, 0
\end{align*}
\]
Interval Horner form. [Shih-Chieh, 1303][Horner, 1819][Stahl, 1995]

Let \( p \) be a polynomial defined by:

\[
 a_0 + \sum_{i=1}^{n} a_i x^{\alpha_i}
\]

\[
 h_p(x) = a_0 + x^{d_1} \left( \cdots + x^{d_{n-1}}(a_{n-1} + a_n x^{d_n}) \cdots \right)
\]

= optimal w.r.t. factorization:

1. made of only multiplications and additions of constants \( \leadsto \) monotonicity
2. completely nested \( \leadsto \) sub-distributivity
Interval Horner form. [Shih-Chieh, 1303][Horner, 1819][Stahl, 1995]

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

$$h_p(x) = a_0 + x^{d_1} \left( \cdots + x^{d_{n-1}} (a_{n-1} + a_n x^{d_n}) \cdots \right)$$

1. Monotonicity.

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

$$\forall x \in \mathbb{R}^n \text{ s.t. } x^* \cap O_p = \emptyset, \quad h_p(x) = \{ p(x) \mid x \in x \}$$
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:

$$a_0 + \sum_{i=1}^{n} a_i x^{\alpha_i}$$

$$h_p(x) = a_0 + x^{d_1} \left( \cdots + x^{d_{n-1}}(a_{n-1} + a_n x^{d_n}) \cdots \right)$$

1. Monotonicity.

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

$$\forall x \in \mathbb{R}^n \text{ s.t. } \bigcap x \cap O_p = \emptyset, \quad h_p(x) = \{p(x) \mid x \in x\}$$

† beyond this condition, no guarantee.
$p : x \mapsto 2x^5 + x^3 - 3x^2$

$h_p : x \mapsto x^2(-3 + x(1 + 2x^2))$

Evaluation of $p$

Evaluation of $h_p$
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: \( a_0 + \sum_{i=1}^{n} a_i x^{a_i} \)

\[
\begin{align*}
    h_p(x) &= a_0 + x^{d_1} \left( \cdots + x^{d_{n-1}}(a_{n-1} + a_n x^{d_n}) \cdots \right)
\end{align*}
\]

1. Monotonicity.

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

\[
\forall x \in \mathbb{R}^n \text{ s.t. } \mathring{x} \cap O_p = \emptyset, \quad h_p(x) = \{ p(x) \mid x \in x \}
\]

\( \dagger \) beyond this condition, no guarantee.

\( \dagger \) pb. with the decomposition of powers
Interval Horner form. [Shih-Chieh, 1303][Horner, 1819][Stahl, 1995]

Let \( p \) be a polynomial defined by:

\[
p(x) = a_0 + \sum_{i=1}^{n} a_i x^{\alpha_i}
\]

\[
h_p(x) = a_0 + x^{d_1} \left( \cdots + x^{d_{n-1}}(a_{n-1} + a_n x^{d_n}) \cdots \right)
\]

2. Sub-distributivity.
Interval Horner form. [Shih-Chieh, 1303][Horner, 1819][Stahl, 1995]

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

$$h_p(x) = a_0 + x^{d_1} \left( \cdots + x^{d_{n-1}}(a_{n-1} + a_n x^{d_n}) \cdots \right)$$

2. Sub-distributivity.

$$a_0 + x^{d_1} \left( \cdots + x^{d_{n-1}}(a_{n-1} + a_n x^{d_n}) \cdots \right) \subseteq a_0 + \sum_{i=1}^{n} a_i x^{\alpha_i}$$
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: $a_0 + \sum_{i=1}^{n} a_i x^{a_i}$

$$h_p(x) = a_0 + x^{d_1} (\cdots + x^{d_{n-1}}(a_{n-1} + a_n x^{d_n}) \cdots)$$

2. Sub-distributivity.

$$p(x) = x + x^4 \quad h_p(x) = x(x^3 + 1)$$
$$q(x) = x + xxxx \quad r(x) = x(xxxx + 1)$$

Let $x = [-2, 1]$:

$$p(x) = [-2, 17] \quad h_p(x) = [-7, 14]$$
$$q(x) = [-10, 17] \quad r(x) = [-10, 14]$$
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:

$$a_0 + \sum_{i=1}^{n} a_i x^{\alpha_i}$$

The Interval Horner form is:

$$h_p(x) = a_0 + x^{d_1} (\cdots + x^{d_{n-1}}(a_{n-1} + a_n x^{d_n}) \cdots)$$

Limits of Horner’s form.

† when intersecting the overestimation set: no guarantee

† does not benefit from the sub-distributivity property

~⇒ 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 \( p(x) = ax^{\alpha + \gamma} + bx^\alpha \),

\[
Mcr_p(x) = ax^{\alpha - \gamma} \left[ (x^\gamma + \frac{b}{2a})^2 - \left( \frac{b}{2a} \right)^2 \right]
\]

with: \( a, b \in \mathbb{R}^* \), \( \alpha \geq \gamma \) 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 \( p(x) = ax^{\alpha+\gamma} + bx^\alpha \),

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

with: \( a, b \in \mathbb{R}^*, \alpha \geq \gamma \) and \( \alpha + \gamma \) even.

Main properties.

\[
\forall x \in \mathbb{R}, \ 0 \not\in x \rightarrow w(\text{Mcr}_p(x)) \leq w(h_p(x))
\]

\[
\begin{align*}
\text{or } (ab > 0 \text{ and } (x \geq 0 \text{ or } x^{\gamma} \leq -\frac{b}{a})) \\
\text{or } (ab < 0 \text{ and } (x \leq 0 \text{ or } x^{\gamma} \geq \frac{b}{a}))
\end{align*}
\rightarrow \text{Mcr}_p(x) = \{p(x) \mid x \in x\}
$p : x \mapsto x^8 - 2x^5$

$h_p : x \mapsto x^5(x^3 - 2)$

$M_{cr_p} : x \mapsto x^2((x^3 - 1)^2 - 1)$

Evaluation of $h_p$

Evaluation of $M_{cr_p}$
Another factorization scheme
for univariate polynomials

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

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

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:

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

and $I' \subset I$ without shared monomials

$\leadsto$ we can rewrite $p$ as follows:

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

\[\Rightarrow \text{ we can rewrite } p \text{ as follows:} \]

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

and we finally factorize:

\[
\text{Mcr}_p(x) = r(x) + \sum_{(i, j) \in I'} \text{Mcr}_{p_{i,j}}(x)
\]
Another factorization scheme for univariate polynomials

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

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

and \( I' \subset I \) without shared monomials

\[\Rightarrow \text{we can rewrite} \ p \ \text{as follows:} \]

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

and we finally factorize:

\[
Mcr_p(x) = r(x) + \sum_{(i, j) \in I'} \text{Mcr}_{p_{i,j}}(x)
\]

many possibilities \( \Rightarrow \) strategies are defined
Main principles.

- No decomposition of odd powers
- No decomposition of even powers into odd ones
- No introduction of odd powers / deletion of odd powers
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 $\alpha$, 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 $x = [-0.5, 0.3]$
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 $\alpha$, 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
Best strategy:

- **second strategy** \( (\varphi) \) when \( \mathcal{O} \cap \mathcal{O}_p \neq \emptyset \) \( \approx 25\% \)-improvement (w.r.t. our tests)
- **Horner** otherwise

\[ \rightarrow \text{globally composition of Horner with our strategy on average} \]

Properties.

- beyond the overestimation interval, \( h \circ \varphi \) is equivalent to \( p \)
- otherwise, \( h \circ \varphi_b \) globally improves the Horner form (w.r.t. our tests), while always keeping equivalent to \( p \)
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

\[ \rightarrow \text{a new symbolic representation to enhance the propagation stage} \]
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 ~⇒ 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

\[ \text{control of the interval computations accuracy} \]

\[ \text{A new triangularization scheme} \]
Triangularization is an option

Consider the following nonlinear constraint system:

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

defined over \( E = [-100, 100]^4 \),

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

• difficult to remove nonlinear terms \( \leadsto \) the nonlinear terms are abstracted
Triangularization is an option

Abstraction phase: equivalent system

\[
\begin{align*}
lc_1 : & \quad x + y + u_1 + u_2 + u_3 = 0 \\
lc_2 : & \quad x + t + u_1 + u_2 + u_4 = 0 \\
lc_3 : & \quad y + z + u_1 + u_5 = 0 \\
lc_4 : & \quad x + z + u_1 + u_2 + u_3 + u_5 = 0
\end{align*}
\]

and the abstracted system:

\[
\begin{align*}
u_1 &= x^2 \\
u_2 &= xy \\
u_3 &= y^2 \\
u_4 &= t^2 \\
u_5 &= z^2
\end{align*}
\]
Triangularization is an option.

Gaussian elimination phase:

\[
\begin{align*}
lc_1 & : \quad u_1 + y + x + u_2 + u_3 = 0 \\
lc'_2 & : \quad y - t - u_4 + u_3 = 0 \\
lc'_3 & : \quad -z - u_5 + x + u_2 + u_3 = 0 \\
lc'_4 & : \quad -t - u_4 + x + u_2 + 2u_3 = 0
\end{align*}
\]
Triangularization is an option

Gaussian elimination phase:

\[
\begin{align*}
 lc_1 : & \quad u_1 + y + x + u_2 + u_3 = 0 \\
 lc'_2 : & \quad y - t - u_4 + u_3 = 0 \\
 lc'_3 : & \quad -z - u_5 + x + u_2 + u_3 = 0 \\
 lc'_4 : & \quad -t - u_4 + x + u_2 + 2u_3 = 0 \\
\end{align*}
\]

- nonlinear terms are restored
Triangularization is an option.

Concretization phase:

\[
\begin{align*}
lc_1 : & \quad x^2 + y + x + xy + y^2 = 0 \\
lc'_2 : & \quad y - t - t^2 + y^2 = 0 \\
lc'_3 : & \quad -z - z^2 + x + xy + y^2 = 0 \\
lc'_4 : & \quad -t - t^2 + x + xy + 2y^2 = 0
\end{align*}
\]

The new system is solved in 240ms!!
Triangularization is an option

Concretization phase:

\[
\begin{align*}
\ell c_1 & : \quad x^2 + y + x + xy + y^2 = 0 \\
\ell c_2' & : \quad y - t - t^2 + y^2 = 0 \\
\ell c_3' & : \quad -z - z^2 + x + xy + y^2 = 0 \\
\ell c_4' & : \quad -t - t^2 + x + xy + 2y^2 = 0
\end{align*}
\]

The new system is solved in 240ms!!

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

\[
\begin{align*}
lc_1 : & \quad x + y + u_1 + u_2 + u_3 = 0 \\
lc_2 : & \quad x + t + u_1 + u_2 + u_4 = 0 \\
lc_3 : & \quad y + z + u_1 + u_5 = 0 \\
lc_4 : & \quad x + z + u_1 + u_2 + u_3 + u_5 = 0
\end{align*}
\]

Pivot. \((lc_3, u_5)\)
A triangularization method

Strategies

First step of elimination

\[
\begin{align*}
lc_3 &: \quad y + z + u_1 + u_5 &= 0 \\
lc_1 &: \quad x + y + u_1 + u_2 + u_3 &= 0 \\
lc_2 &: \quad x + t + u_1 + u_2 + u_4 &= 0 \\
lc'_4 &: \quad -x + y - u_2 - u_3 &= 0
\end{align*}
\]

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

\[
\begin{align*}
lc_3 &: \quad y + z + u_1 + u_5 = 0 \\
lc'_4 &: \quad -x + y - u_2 - u_3 = 0 \\
lc'_1 &: \quad 2y + u_1 = 0 \\
lc_2 &: \quad x + t + u_1 + u_2 + u_4 = 0
\end{align*}
\]

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

User linear part: 0

Abstracted linear part: \(-1\)

Pivot. \((lc'_1, u_1)\)
A triangularization method

Strategies

Third step of elimination

\[
\begin{align*}
lc_3 &: \quad y + z + u_1 + u_5 = 0 \\
lc'_4 &: \quad -x + y - u_2 - u_3 = 0 \\
lc'_1 &: \quad 2y + u_1 = 0 \\
lc'_2 &: \quad -x + 2y - t - u_2 - u_4 = 0
\end{align*}
\]

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

\[
\begin{align*}
lc'_2 & : \quad -t \quad -x \quad -u_2 \quad -u_4 \quad +2y = 0 \\
lc'_4 & : \quad -x \quad -u_2 \quad -u_3 \quad +y = 0 \\
lc_3 & : \quad +u_5 \quad +z \quad +u_1 \quad +y = 0 \\
lc'_1 & : \quad +u_1 \quad +2y = 0
\end{align*}
\]

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

\[
\begin{align*}
     c_1' & : -t -x -xy -t^2 + 2y = 0 \\
     c_2' & : -x -xy -y^2 + y = 0 \\
     c_3' & : z^2 + z + x^2 + y = 0 \\
     c_4' & : x^2 + 2y = 0 
\end{align*}
\]

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 \(x^2\)
A triangularization method

Strategies

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

<table>
<thead>
<tr>
<th>Equation</th>
<th>Coefficients</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_T^1$</td>
<td>$-t$ $-x$ $-xy$ $-t^2$</td>
<td>$+2y = 0$</td>
</tr>
<tr>
<td>$c_T^2$</td>
<td>$-x$ $-xy$ $-y^2$</td>
<td>$+y = 0$</td>
</tr>
<tr>
<td>$c_T^3'$</td>
<td>$z^2$ $+z$ $-y$</td>
<td>$= 0$</td>
</tr>
<tr>
<td>$c_T^4$</td>
<td>$+x^2$ $+2y$</td>
<td>$= 0$</td>
</tr>
</tbody>
</table>

Solving stage: 4 solutions reached in 10ms!
Tests and results

Bratu’s problem.

\[ x_{k-1} - 2x_k + x_{k+1} + h \exp(x_k) = 0, \quad 1 \leq k \leq n \]

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

\[ x_{k-1} - 2x_k + x_{k+1} + h \exp(x_k) = 0, \quad 1 \leq k \leq n \]

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:

\[-(k + 1)x_k + (k + 2)x_{k+1} + h \sum_{i=1}^{k} i \exp(x_i) = 0, \quad 1 \leq k \leq n\]
Bratu’s problem.

\[ x_{k-1} - 2x_k + x_{k+1} + h \exp(x_k) = 0, \quad 1 \leq k \leq n \]

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

<table>
<thead>
<tr>
<th>Problem</th>
<th>(v)</th>
<th>Initial Pb.</th>
<th>Triangul. Pb.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bratu</td>
<td>7</td>
<td>1.10</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.70</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2.30</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>20.50</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>46.40</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>94.40</td>
<td>12</td>
</tr>
</tbody>
</table>
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
Motivation.

- providing a general framework, allowing to model explicitly the required flexibility
- exploiting the properties of well-known algorithms for classical problems (i.e., ≠ soft)

Framework.

- based on distances/flexibility measures: the smallest flexibility is sought
- also integrates an order over the constraints
Given a constraint $c$ defined over $E \subset \mathbb{R}^n$, a soft constraint resulting from $c$ is defined by a pair

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

where: $d$ defines a distance between $c$ 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

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

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

**Properties of $d$:** increasing function s.t. $d(0) = 0$. Interpret the rough distance to $c$. 

---

*Summer School NMSU, 27 July 2008 – p. 47/67*
Given a constraint $c$ defined over $E \subset \mathbb{R}^n$, a soft constraint resulting from $c$ is defined by a pair

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

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

Properties of $d$: 

![Graph showing the 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

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

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

\[
\hat{c} = (c, d)
\]

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

**Properties of \( d \):**

- **Fuzzy/Normalized Distance**

[Graph showing a curve with a peak, labeled as fuzzy/normalized distance.]
Given a constraint $c$ defined over $E \subset \mathbb{R}^n$, a soft constraint resulting from $c$ is defined by a pair

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

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

\[
\hat{c} = (c, d)
\]

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

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

\[
= \{ x \in E \mid \forall y \in E, d(x, c) \leq d(y, c) \}
\]
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

\[ \hat{c} = (c, d) \]

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

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

\[ = \{ x \in E \mid \forall y \in E, d(x, c) \leq d(y, c) \} \]

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

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

where: $D$ is a set of distances corresponding to each constraint $c_i$

$d$ is a operator combining the values of the distances of $D$

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

\[
\hat{C} = (C, d, D, \succ)
\]

Properties of \( d \): defined over \((\mathbb{R}^+)^p = \text{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

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

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

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

increasing function w.r.t. each parameter

Remark concerning $D$: 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

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

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

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

increasing function w.r.t. each parameter

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

= commensurability problem

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

* may be trivial

• otherwise, states new constraints $C_\succ$
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

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

$$\hat{C} = (C, d, D, \succeq)$$
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

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

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

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

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

\[
\hat{C} = (C, d, D, \succ)
\]

Solution set of \( \hat{C} \) is the closest to \( C \) (w.r.t. \( d \)) subset of \( E \) satisfying \( C \succ \).

\[
\{ x \in E \mid C \succ \text{ holds on } x \text{ and } \forall y \in E, d(x, C) \leq d(y, C) \}
\]

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

\[
\min_{x \in E} d(d_1(x), \ldots, d_p(x)) \quad \text{s.t. } x \text{ satisfies } C' \succ
\]
Given a soft CSP $\hat{C} = (C, d, D, \succ)$ defined over $E \subset \mathbb{R}^n$, the solution set of $\hat{C}$ is the solution set of the following hard problem:

$$\min_{x \in E} d(d_1(x), \ldots, d_p(x))$$

s.t. $x$ satisfies $C\succ$

---

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

† **Pbs. with normalized distances:**

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

   $\Rightarrow$ interval upper bound

2. but may be $\infty$

   $\Rightarrow$ variation of the normalized distance
Solving process

\[ x^* \times x^{**} = 0 \]

\[ 0.76 \times 0.99 = 0 \]

Interval function (distance) = sum of normalized distances

A smooth variation

\[ x^* = 0.54 \]
\[ x^{**} = 0.99 \]
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:

```
\[ A \]
\[ B \]
\[
\theta_c
\]
camera’s scope
```
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:
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:

6. 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:

1. 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

2. 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:

1. 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.

2. 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.
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.
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

\[
\begin{align*}
\text{organize}(\text{large\_room}, [a, b, c], D) & \leftarrow \text{meeting}([a, b, c], D) \\
\text{organize}(\text{small\_room}, [X, Y], D) & \leftarrow \text{meeting}([X, Y], D) \\
\text{meeting}([a, b], D) & \leftarrow \text{available}(a, D), \text{available}(b, D), \text{not\_available}(c, D) \\
\text{meeting}([b, c], D) & \leftarrow \text{not\_available}(a, D), \text{available}(b, D), \text{available}(c, D) \\
\text{meeting}([a, c], D) & \leftarrow \text{available}(a, D), \text{not\_available}(b, D), \text{available}(c, D) \\
\text{meeting}([a, b, c], D) & \leftarrow \text{available}(a, D), \text{available}(b, D), \text{available}(c, D) \\
\text{available}(P, D) & \leftarrow \text{free}(P)@D \\
\text{not\_available}(P, D) & \leftarrow \text{busy}(P)@D
\end{align*}
\]

\{ questions sent to agents \}

? \leftarrow \text{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

\[
(x - x_0)^2 + (y - y_0)^2 \leq (t_0 \cdot s_0)^2
\]

\[
x, y, \in [-10^8, 10^8]
\]

\[
x_0 = \text{location}(X)
\]
\[
y_0 = \text{location}(Y)
\]
\[
s_0 = \text{speed}(S)
\]
\[
d_0 = \text{duration}(D)
\]

\[
\text{questions sent to agents and transmitted to sensors}
\]
SCP in a Master-Slave environment

Basic idea [Satoh, Prima 2003]:

- The program \(\text{constraint problem, denoted by } P\) is centralized at the master’s level \(\text{denoted by } M\)
SCP in a Master-Slave environment

Basic idea [Satoh, Prima 2003]:

- The program \(\text{constraint problem, denoted by } P\) is centralized at the master’s level \(\text{denoted by } M\)
- \(M\) begins to run the program / solve the constraint system
SCP in a Master-Slave environment

Basic idea [Satoh, Prima 2003]:

1. The program \((\text{constraint problem, denoted by } P)\) is centralized at the master’s level \((\text{denoted by } M)\)
2. M begins to run the program / solve the constraint system
3. When specific information is needed:
SCP in a Master-Slave environment

Basic idea [Satoh, Prima 2003]:

1. The program (constraint problem, denoted by $P$) is centralized at the master’s level (denoted by $M$)
2. $M$ begins to run the program / solve the constraint system
3. When specific information is needed: *e.g.*, 
   - is person $a$ available on day $D$? $\text{free}(a)@D$
   - where is the robot located? $x_0 = \text{location}(X)$, $y_0 = \text{location}(Y)$
   - etc.
SCP in a Master-Slave environment

Basic idea [Satoh, Prima 2003]:

- The program \((\text{constraint problem, denoted by } P)\) is centralized at the master’s level \((\text{denoted by } M)\)
- 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 $\delta$: 
Before S answers, M continue the processing of P with some default value/constraint $\delta$: e.g.,

$\triangle \leftarrow D \in \{1, 2\} || free(a)@D$

$\triangle x_0 \in [1, 100], y_0 \in [10, 25]$
SCP in a Master-Slave environment

Before S answers, M continue the processing of P with some default value/constraint $\delta$: no time is wasted
SCP in a Master-Slave environment

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

When answers $\alpha$ come from S, M updates or reinforces its belief depending on whether:

- $\alpha$ entails $\delta$: $\alpha \subset \delta$
- $\alpha$ contradicts $\delta$: $\alpha \cap \delta = \emptyset$
- $\alpha$ is consistent with $\delta$ but does not entail it: $\alpha \cap \delta \neq \emptyset$ but $\alpha \not\subset \delta$
What is speculative computation with MA belief revision?
What is speculative computation with MA belief revision?

- each agent can perform speculative computations
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
Speculative computations with MA belief revision for yes/no questions [Satoh, AAMAS’03]
Speculative computations with MA belief revision for yes/no questions [Satoh, AAMAS’03]

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

- when S sends an answer $\delta_s$, 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
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 = ¬ 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:

1. solve general constraints *(or ask more general questions)*...
2. ... 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*
Thank you for your attention

QUESTIONS?

Martine Ceberio
mceberio@utep.edu
www.constraintsoving.com
http://www.martineceberio.fr
University of Texas at El Paso