Project SCooP-Sakura

Presentation, Progress, Perspectives

NII, Japan, and LINA, France

Martine Ceberio

LINA, Nantes France / University of Texas at El Paso

Outline of the presentation

- *Context and general presentation of the project*
- *What is this about?, and what are our objectives?*
- *Besides constraints, other issues we will have to tackle*
- *Conclusion and future plans*

Getting acquainted with:

SCooP-Sakura

What is SCooP-Sakura?

SCooP

- . Soft and Continuous Constraint Programming
- Cooperative project between NII (Tokyo) and LINA (Nantes)
- Part of the MOU between these institutes
- Objectives of SCooP: integrate all projects related to soft and / or continuous constraints between NII and LINA

What is SCooP-Sakura?

SCooP

- . Soft and Continuous Constraint Programming
- Cooperative project between NII (Tokyo) and LINA (Nantes)
- Part of the MOU between these institutes
- Objectives of SCooP: integrate all projects related to soft and / or continuous constraints between NII and LINA

Projects integrated to SCooP

- Speculative Constraint Processing in Multi-Agent Systems

  Members: **Ken Satoh**, *Hiroshi Hosobe, Makoto Yokoo, Katsutoshi Hirayama, Martine Ceberio*

What is SCooP-Sakura?

SCooP

- . Soft and Continuous Constraint Programming
- Cooperative project between NII (Tokyo) and LINA (Nantes)
- Part of the MOU between these institutes
- Objectives of SCooP: integrate all projects related to soft and / or continuous constraints between NII and LINA

Projects integrated to SCooP

- Speculative Constraint Processing in Multi-Agent Systems
- Dynamic CSP

  Members: **Ken Satoh**, *Hiroshi Hosobe, Narendra Jussien*

What is SCooP-Sakura?

SCooP

- . Soft and Continuous Constraint Programming
- Cooperative project between NII (Tokyo) and LINA (Nantes)
- Part of the MOU between these institutes
- Objectives of SCooP: integrate all projects related to soft and / or continuous constraints between NII and LINA

Projects integrated to SCooP

- Speculative Constraint Processing in Multi-Agent Systems
- Dynamic CSP
- project Sakura

More about Sakura

Continuous Soft Constraints and Graphical Applications

Funded by:

- the French Ministry of Foreign Affairs (program: Égide PAI)
- the Japan Society for the Promotion of Science (JSPS)

More about Sakura

Continuous Soft Constraints and Graphical Applications

Funded by:

- the French Ministry of Foreign Affairs (program: Égide PAI)
- the Japan Society for the Promotion of Science (JSPS)

Members:

- NII: **Ken Satoh**, Hiroshi Hosobe
- LINA: **Frédéric Benhamou**, Christophe Jermann, Martine Ceberio

More about Sakura

Continuous Soft Constraints and Graphical Applications

Funded by:

- the French Ministry of Foreign Affairs (program: Égide PAI)
- the Japan Society for the Promotion of Science (JSPS)

Members:

- NII: **Ken Satoh**, Hiroshi Hosobe
- LINA: **Frédéric Benhamou**, Christophe Jermann, Martine Ceberio

Began since first semester of 2004.

The nuts and bolts of:

Project Sakura

- Motivation
- Objectives
- Framework

Constraint satisfaction

Solving a (set of) constraint(s):

determining the instanciations of the variables (within the specified domains) that meet/*satisfy* the constraint(s)

Constraint satisfaction

Solving a (set of) constraint(s):
determining the instanciations of the variables (within the specified domains) that meet/*satisfy* the constraint(s)

= **Splitting the search space** (*i.e.,* the variables' domains) into:

- the elements satisfying the constraints (*consistent* elements)
- the elements violating the constraints (*inconsistent* elements)

Why be flexible?

1. **because constraints may not be compatible:**

   i.e., no instanciation meets the constraints

   e.g., the model $\neq$ expected picture of the situation

   or too ambitious to be consistent

   etc.

Why be flexible?

1. **because constraints may not be compatible:**

   i.e., no instanciation meets the constraints

   e.g., the model $\neq$ expected picture of the situation

   or too ambitious to be consistent

   etc.

2. **because it may make no sense to divide accurately the search space into two precisely bounded parts**

   e.g., when data not accurate enough, due to machine error

   $\rightsquigarrow$ meaningless to define an accurate frontier between elements

Organizing a meeting with n participants

Example

Organizing a meeting with n participants

- all participants have a tight schedule
- we know the schedule of each of them: meeting times, time to go from one meeting to the other (known with uncertainty $\rightarrow$ interval), etc.

? is there a time slot, on day d, s.t. all can meet?

Example

Organizing a meeting with n participants

- all participants have a tight schedule
- we know the schedule of each of them: meeting times, time to go from one meeting to the other (known with uncertainty $\longrightarrow$ interval), etc.

? is there a time slot, on day d, s.t. all can meet?

Most probably not...

Example

Organizing a meeting with n participants

- all participants have a tight schedule
- we know the schedule of each of them: meeting times, time to go from one meeting to the other (known with uncertainty $\rightarrow$ interval), etc.

? is there a time slot, on day d, s.t. all can meet?

Most probably not...

On the other hand:

- this meeting may be held without everybody attending?
- there may be possible other changes in the schedules of participants?
- etc.

Example

Organizing a meeting with n participants

- all participants have a tight schedule
- we know the schedule of each of them: meeting times, time to go from one meeting to the other (known with uncertainty $\rightarrow$ interval), etc.

? is there a time slot, on day d, s.t. all can meet?

Most probably not...

On the other hand:

- this meeting may be held without everybody attending?
- there may be possible other changes in the schedules of participants?
- etc.

Many possible adjustments...

Objectives of Sakura

- Implement a library for handling such inconsistent problems
  - objectives of such a tool

  NOT to provide the user with expert advise

Objectives of Sakura

- Implement a library for handling such inconsistent problems
  - objectives of such a tool

  NOT to provide the user with expert advise

  BUT to determine solutions[a] corresponding to his/her relaxation

[a] to be defined later

Objectives of Sakura

- Implement a library for handling such inconsistent problems
  - objectives of such a tool

  NOT to provide the user with expert advise

  BUT to determine solutions corresponding to his/her relaxation
    - *we may also make suggestions: using qualitative information about constraints (cf. Christophe Jermann's work)*

Objectives of Sakura

- Implement a library for handling such inconsistent problems
  - objectives of such a tool

  NOT to provide the user with expert advise

  BUT to determine solutions corresponding to his/her relaxation
    - *we may also make suggestions: using qualitative information about constraints (cf. Christophe Jermann's work)*
- Design a user-friendly interface

Objectives of Sakura

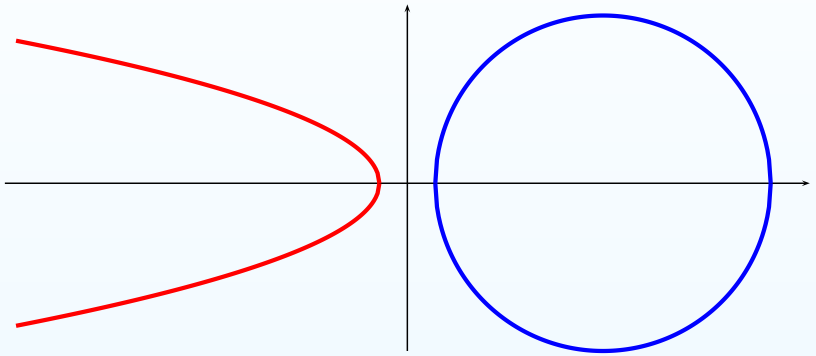
- Implement a library for handling such inconsistent problems
  - objectives of such a tool

  NOT to provide the user with expert advise

  BUT to determine solutions corresponding to his/her relaxation
    - *we may also make suggestions: using qualitative information about constraints (cf. Christophe Jermann's work)*
- Design a user-friendly interface

But then: **what kind of flexibility?**

Sakura: What kind of flexibility?

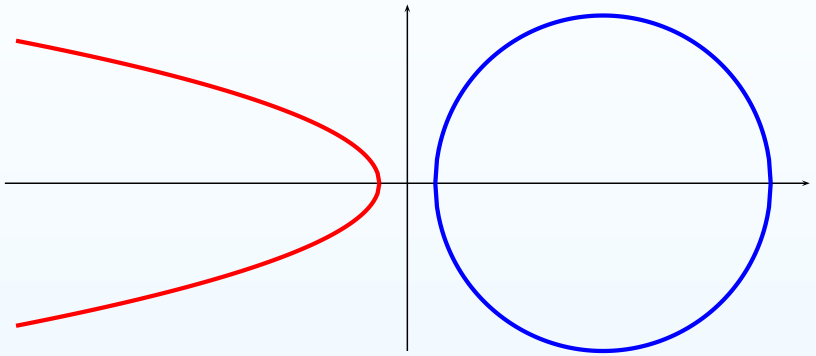


— Constraint c_1 : $(x - \frac{7}{4})^2 + y^2 \leq \left(\frac{3}{2}\right)^2$

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

Solution set $= \emptyset$

Sakura: What kind of flexibility?



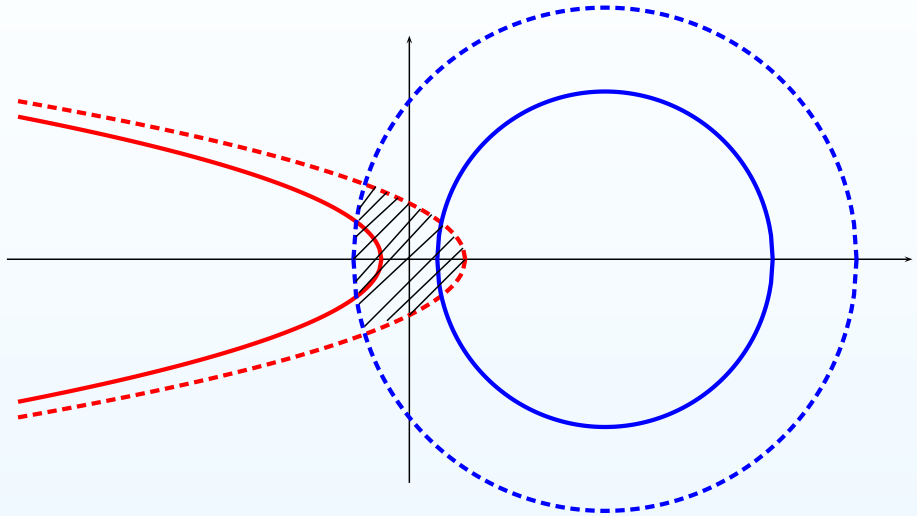
— Constraint c_1 : $(x - \frac{7}{4})^2 + y^2 \leq \left(\frac{3}{2}\right)^2$

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

Solution set $= \emptyset$

- stretch the constraints

Sakura: What kind of flexibility?



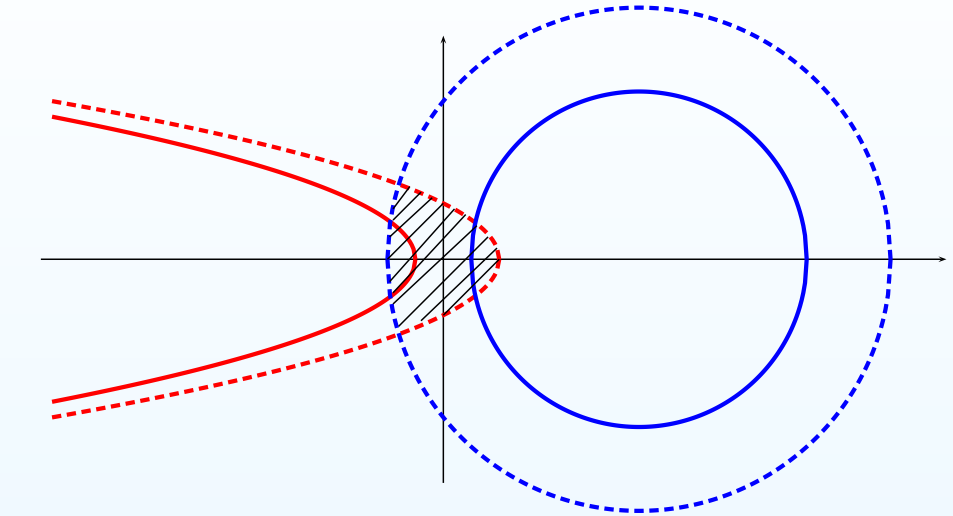
— Constraint c_1' : $(x - \frac{7}{4})^2 + y^2 \leq \left(\frac{9}{4}\right)^2$

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

Solution set $= \emptyset$ $\rightsquigarrow$ “Extended” solution set

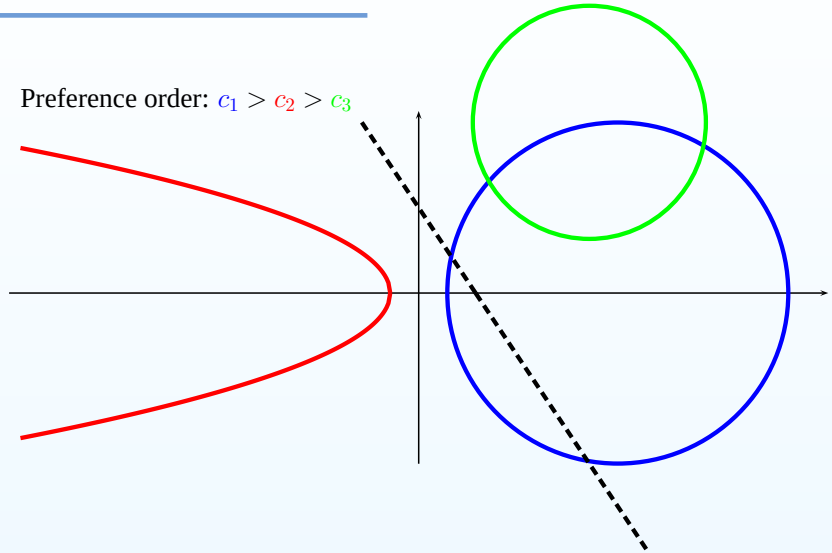
- stretch the constraints

Sakura: What kind of flexibility?



- stretch the constraints
- establish orders, with possible interpretations such as:
  - *preferences over the constraints with possible distances*

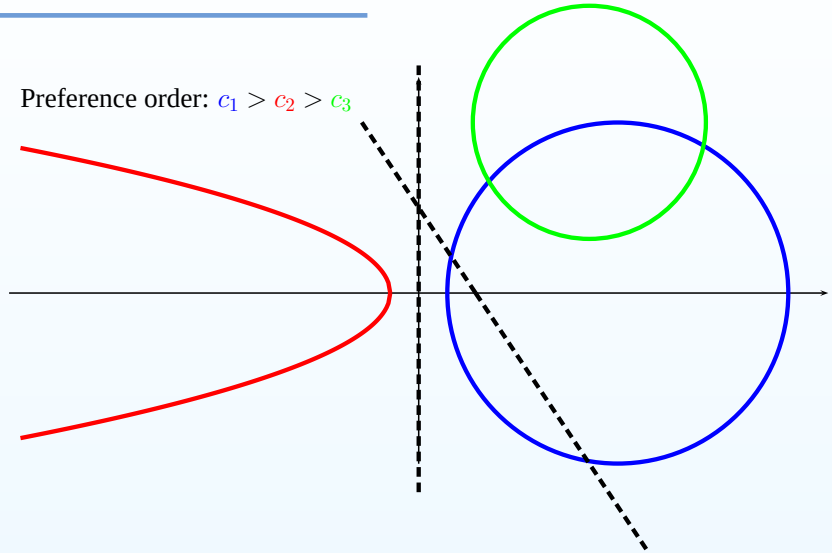
Sakura: What kind of flexibility?



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

- stretch the constraints
- establish orders, with possible interpretations such as:
  - *preferences over the constraints with possible distances*

Sakura: What kind of flexibility?



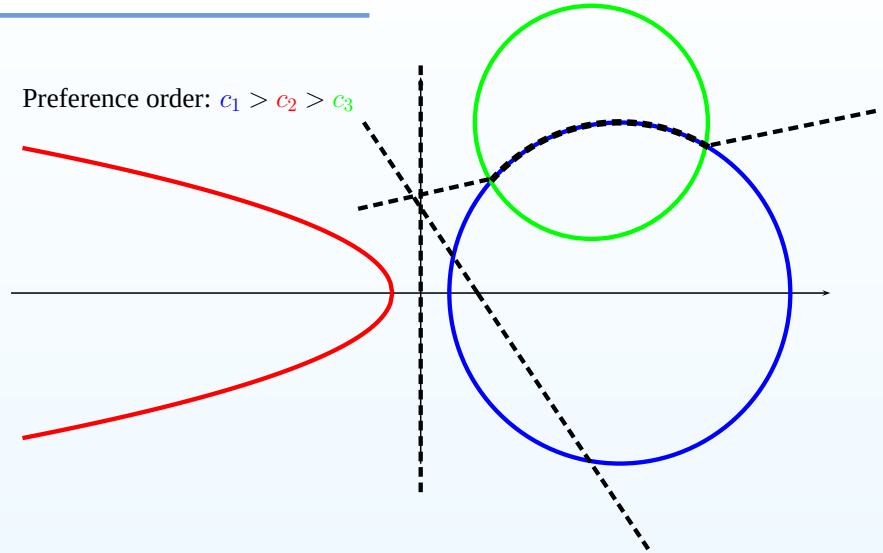
Constraint c_1 : $(x - \frac{7}{4})^2 + y^2 \leq \left(\frac{3}{2}\right)^2$

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

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

- stretch the constraints
- establish orders, with possible interpretations such as:
  - *preferences over the constraints with possible distances*

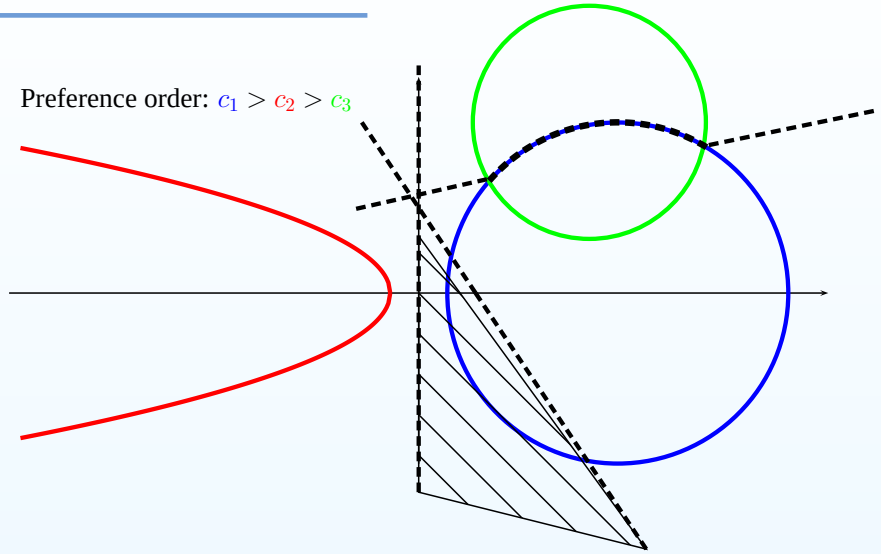
Sakura: What kind of flexibility?



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

- stretch the constraints
- establish orders, with possible interpretations such as:
  - *preferences over the constraints with possible distances*

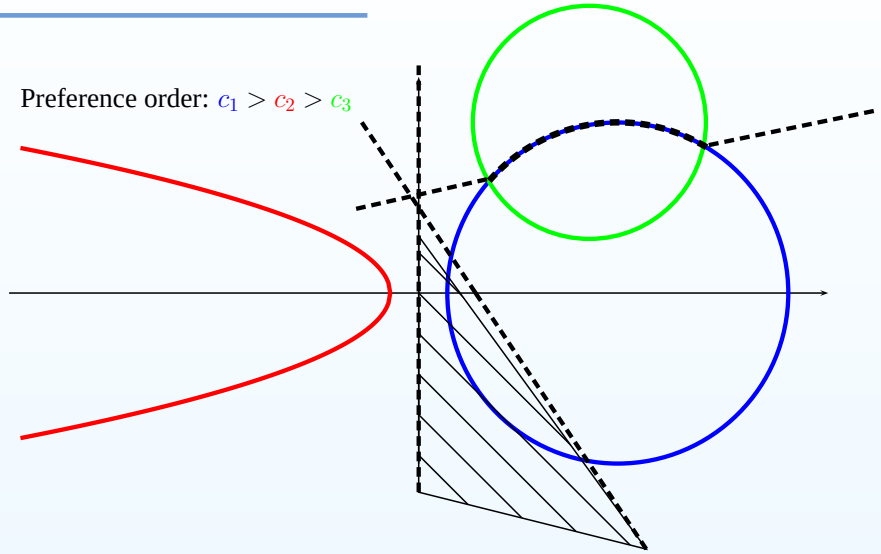
Sakura: What kind of flexibility?



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

- stretch the constraints
- establish orders, with possible interpretations such as:
  - *preferences over the constraints with possible distances*

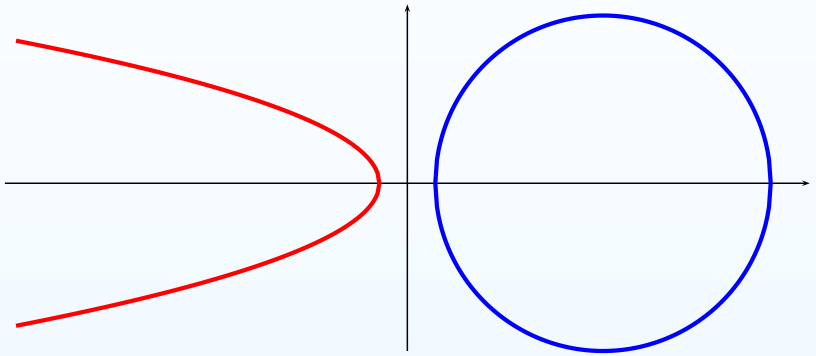
Sakura: What kind of flexibility?



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

- stretch the constraints
- establish orders, with possible interpretations such as:
  - *preferences over the constraints with possible distances*
  - *over the constraints as crisp constraints*

Sakura: What kind of flexibility?



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 $= \emptyset$

- stretch the constraints
- establish orders, with possible interpretations such as:
  - *preferences over the constraints with possible distances*
  - *over the constraints as crisp constraints*
  - *with "if then else" meaning*

Sakura: What kind of flexibility? (cont'd)

- **stretch the constraints**
- **strong preferences:** *order over the satisfaction of the constraints*
- **weak preferences:** *order met only if possible*
- **priorities**
- **"if then else"** orders: *if* c_1 *then* $c_1 \wedge c_2$, *else* c_3
- **hierarchies**
- *etc.*

Sakura: What kind of flexibility? (cont'd)

- **stretch the constraints**
- **strong preferences:** *order over the satisfaction of the constraints*
- **weak preferences:** *order met only if possible*
- **priorities**
- **"if then else"** orders: *if* c_1 *then* $c_1 \wedge c_2$, *else* c_3
- **hierarchies**
- *etc.*

Objective:	be general enough
Practically:	we focus on graphical applications
	(cf. Hosobe-sensei's presentation)
Later:	we will enrich our application fields

Sakura: aim of our work

- Based on constraint solving and optimization methods:
  - optimization: minimization of the violation
  - constraint solving: to maintain a preference order + for when non-zero distances to constraints are not allowed
- Be general enough to be able to model many kinds of soft constraints
  - yet we will restrict ourselves to some of them
- Use state-of-the-art solvers to benefit from:
  - their properties
  - the implementation... already done, tested, optimized

  $\rightsquigarrow$ connection between our framework and classical problems

Sakura: a glance at the framework we currenlty use

$$\begin{array}{lcl} c & \rightsquigarrow & (c, f) \\ C & \rightsquigarrow & (C, A, F, t) \end{array}$$

where, at the constraint's level:

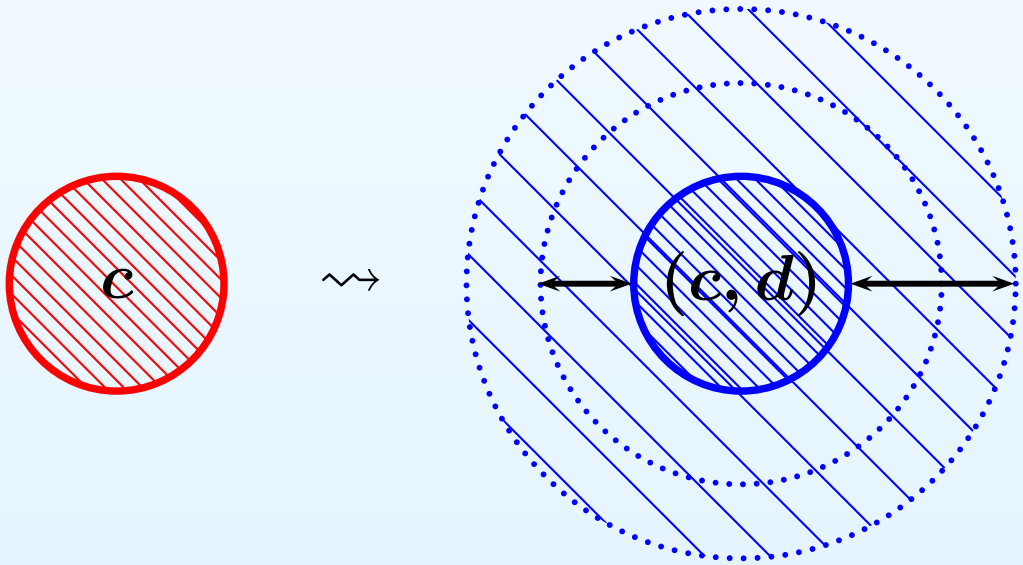
- f stands for any error function: *distance, stair function, etc.*

Sakura: a glance at the framework we currenlty use

$$\begin{array}{lcl} c & \rightsquigarrow & (c, f) \\ C & \rightsquigarrow & (C, A, F, t) \end{array}$$

where, at the constraint's level:

- f stands for any error function: *distance, stair function, etc.*

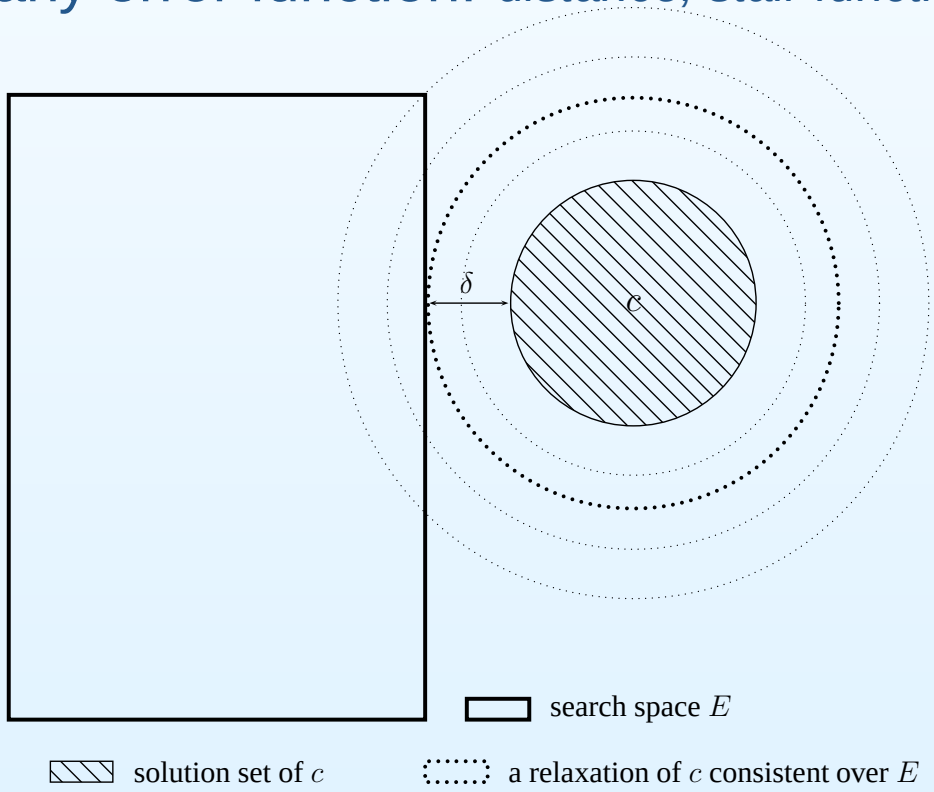


Sakura: a glance at the framework we currenlty use

$$
\begin{array}{lcl}
c & \rightsquigarrow & (c, f) \\
C & \rightsquigarrow & (C, A, F, t)
\end{array}
$$

where, at the constraint's level:

- f stands for any error function: *distance, stair function, etc.*

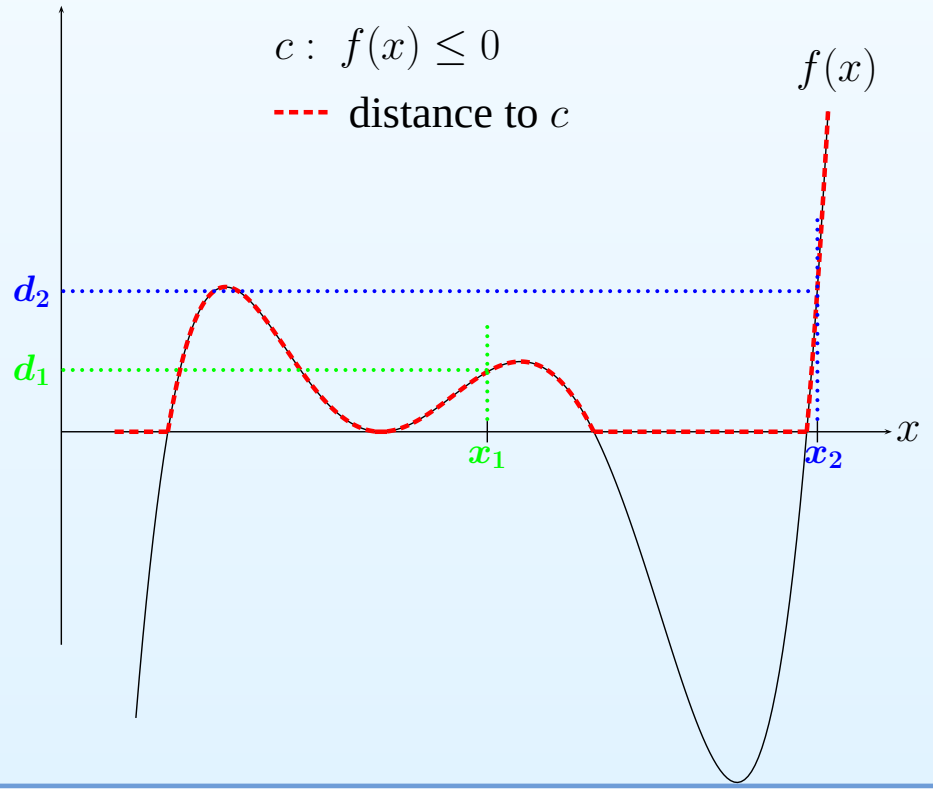


Sakura: a glance at the framework we currenlty use

$$\begin{array}{lcl} c & \rightsquigarrow & (c, f) \\ C & \rightsquigarrow & (C, A, F, t) \end{array}$$

where, at the constraint's level:

- f stands for any error function: *distance, stair function, etc.*



Sakura: a glance at the framework we currenlty use

$$\begin{array}{lcl} c & \rightsquigarrow & (c, f) \\ C & \rightsquigarrow & (C, A, F, t) \end{array}$$

where, at the constraint's level:

- f stands for any error function: *distance, stair function, etc.*

Optimization: minimization of error

Sakura: a glance at the framework we currenlty use

$$\begin{array}{lcl} c & \rightsquigarrow & (c, f) \\ C & \rightsquigarrow & (C, A, F, t) \end{array}$$

where, at the constraint's level:

- f stands for any error function: *distance, stair function, etc.*

and when it comes to sets of constraints $C = (c_1, \ldots, c_n)$:

- $F = (f_1, \ldots, f_n)$

Sakura: a glance at the framework we currenlty use

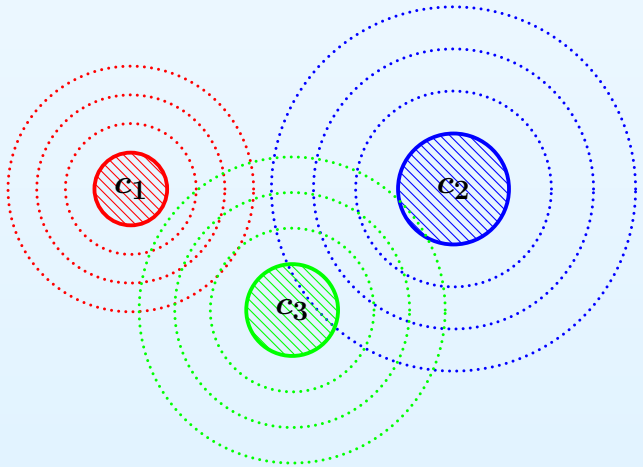
$$\begin{array}{lcl} c & \rightsquigarrow & (c, f) \\ C & \rightsquigarrow & (C, A, F, t) \end{array}$$

where, at the constraint's level:

- f stands for any error function: *distance, stair function, etc.*

and when it comes to sets of constraints $C = (c_1, \ldots, c_n)$:

- $F = (f_1, \ldots, f_n)$



Sakura: a glance at the framework we currenlty use

$$\begin{array}{lcl} c & \rightsquigarrow & (c, f) \\ C & \rightsquigarrow & (C, A, F, t) \end{array}$$

where, at the constraint's level:

- f stands for any error function: *distance, stair function, etc.*

and when it comes to sets of constraints $C = (c_1, \ldots, c_n)$:

- $F = (f_1, \ldots, f_n)$
- A is a vector of aggregation operators

  e.g., $a = f_1 + \cdots + f_n$

Optimization: minimization of the global error $a(f_1, \ldots, f_n)$

Sakura: a glance at the framework we currenlty use

$$\begin{array}{lcl} c & \rightsquigarrow & (c, f) \\ C & \rightsquigarrow & (C, A, F, t) \end{array}$$

where, at the constraint's level:

- f stands for any error function: *distance, stair function, etc.*

and when it comes to sets of constraints $C = (c_1, \ldots, c_n)$:

- $F = (f_1, \ldots, f_n)$
- A is a vector of aggregation operators
- t is a tree representing the order over constraints

Constrained optimization:

minimization of the error while satisfying the order

Sakura: a glance at the framework... (cont'd)

A few more words on A and t...

Sakura: a glance at the framework... (cont'd)

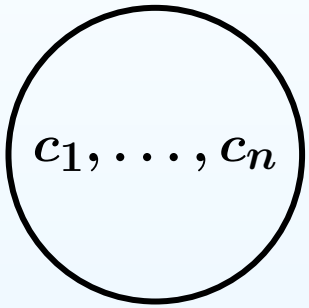
A few more words on A and t...

- *t a tree of (sets of) constraints, defining the order over constraints*

Sakura: a glance at the framework... (cont'd)

A few more words on A and t...

- *t a tree of (sets of) constraints, defining the order over constraints*

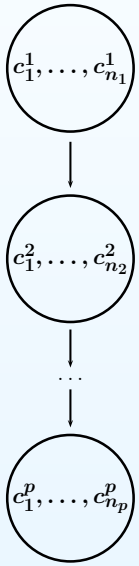


  - **No order**

Sakura: a glance at the framework... (cont'd)

A few more words on A and t...

- *t a tree of (sets of) constraints, defining the order over constraints*

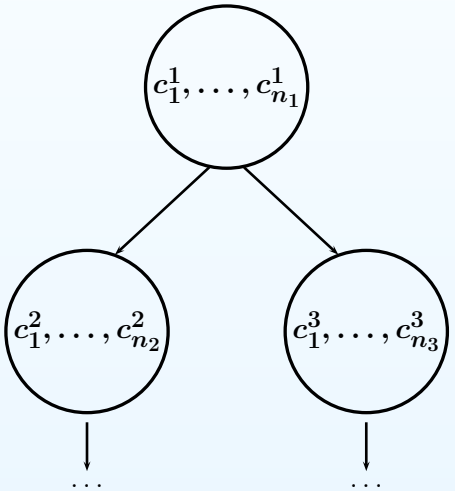


  - **(Linear) Hierarchies of constraints:** *level 1 preferred to level 2, and so on and so forth...*
  - **Priorities over (sets of) constraints:** *if level 1 can be satisfied, then try to satisfy level 2 too, etc.* **if level 1 inconsistent, no solution...**

Sakura: a glance at the framework... (cont'd)

A few more words on A and t...

- *t a tree of (sets of) constraints, defining the order over constraints*



  - **(More complex) Hierarchies:** *general tree*
    each edge stands for the parent being preferred to the children

$$\text{for } k = 2, 3,\ \forall i \in \{1, \ldots, n_1\},\ j_k \in \{1, \ldots, n_k\},\ f_i^1(x) \leqslant f_{j_k}^k(x)$$

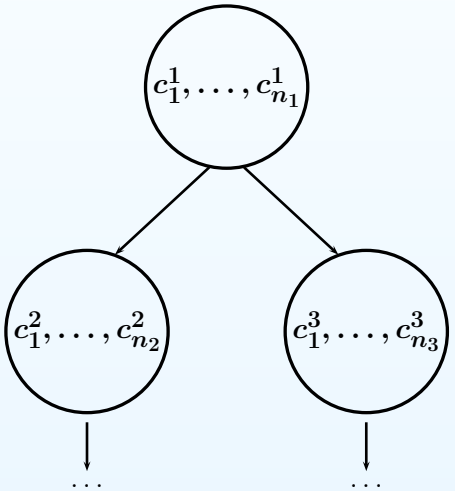
and / or:

$$\text{for } k = 2, 3,\ a_1(f_1^1(x), \ldots, f_{n_1}^1) \leqslant a_k(f_1^k(x), \ldots, f_{n_k}^k)$$

Sakura: a glance at the framework... (cont'd)

A few more words on A and t...

- *t a tree of (sets of) constraints, defining the order over constraints*



  - **"if . . . then . . . else . . . " constraint structure: binary tree**

$$\text{if } c_1^1 \wedge \cdots \wedge c_{n_1}^1 \quad \text{then} \quad (c_1^1 \wedge \cdots \wedge c_{n_1}^1) \wedge (c_1^2 \wedge \cdots \wedge c_{n_2}^2)$$
$$\text{else} \quad c_1^3 \wedge \cdots \wedge c_{n_3}^3$$

Problem with this scheme: *may lead to inconsistencies too...*

Sakura: a glance at the framework... (cont'd)

A few more words on A and t...

- *t a tree of (sets of) constraints, defining the order over constraints*
- *A a vector of aggregation operators*

$$A = (a_0, a_1, \ldots, a_p)$$

  - *a_0 is a global aggregation operator, possibly not defined ($\rightsquigarrow$ flag "?")*
  - *$(a_1, \ldots, a_p)$ is a vector of additional aggreg. op., possibly empty*

Sakura: a glance at the framework... (cont'd)

A few more words on A and t...

- *t a tree of (sets of) constraints, defining the order over constraints*
- *A a vector of aggregation operators*

  For instance:

  - *$A = (a_0)$ when there is no order, i.e., t=single node tree*
  - *$A = (?, a_1, \ldots, a_p)$ when t consists of p nodes*

    e.g., constraint hierarchies
  - *$A = (a_0)$ and t has p nodes: strong hierarchy*
  - *$A = (?)$ and t has p nodes: depending on the structure of t,*

    interpreted as "linear" priority or "if . . . then . . . else . . . "

Sakura: examples

Stretching the constraints:

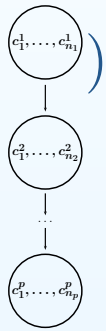
$$(C, (a_0 = \sum), (d_1, \ldots, d_n), t_1 = \overset{c_1,\ldots,c_n}{\bigcirc})$$

Sakura: examples

Stretching the constraints: $(C, (a_0 = \sum), (d_1, \ldots, d_n), t_1 =$ 

$)$

Hierarchy of constraints:

$$(C, (?, a_1, \ldots, a_p), (d_1, \ldots, d_n), t_2 =$$ 

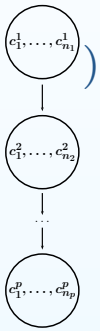
$)$

here each level has its aggregated value smaller than lower levels

Sakura: examples

Stretching the constraints: $(C, (a_0 = \sum), (d_1, \ldots, d_n), t_1 =$ 

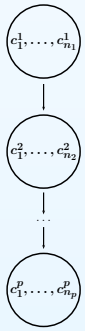
$)$

Hierarchy of constraints: 1. $(C, (?, a_1, \ldots, a_p), (d_1, \ldots, d_n), t_2 =$ 

$)$

here each level has its aggregated value smaller than lower levels

Hierarchy of constraints:

$$(C, (a_0), (d_1, \ldots, d_n), t_2 = \text{  }$$

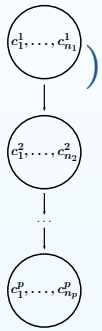
$)$

here we minimize $a_0(x)$ under $d_i^j(x) \leqslant d_k^l(x)$ for all $i < j$:
stronger than former hierarchy scheme

Sakura: examples

Stretching the constraints: $(C, (a_0 = \sum), (d_1, \ldots, d_n), t_1 =$ 

$)$

Hierarchy of constraints: 1. $(C, (?, a_1, \ldots, a_p), (d_1, \ldots, d_n), t_2 =$ 

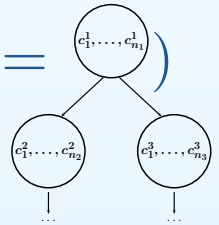
$)$

here each level has its aggregated value smaller than lower levels

2. $(C, (a_0), (d_1, \ldots, d_n), t_2)$

here we minimize $a_0(x)$ under $d_i^j(x) \leqslant d_k^l(x)$ for all $i < j$

Tree structures:

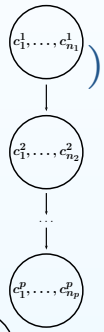
$(C, (?, a_1, \ldots, a_p), (d_1, \ldots, d_n), t_3 =$ 

$)$
or (a_0)

extension of the former linear hierarchy schemes (1, 2)

Sakura: examples

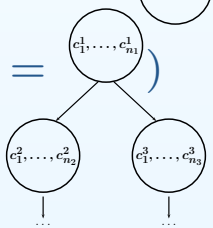
Stretching the constraints: $(C, (a_0 = \sum), (d_1, \ldots, d_n), t_1 =$  $)$

Hierarchy of constraints: 1. $(C, \mathbf{(?, a_1, \ldots, a_p)}, (d_1, \ldots, d_n), t_2 =$  $)$

here each level has its aggregated value smaller than lower levels

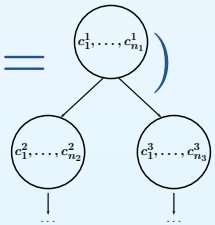
2. $(C, \mathbf{(a_0)}, (d_1, \ldots, d_n), t_2)$

here we minimize $a_0(x)$ under $d_i^j(x) \leqslant d_k^l(x)$ for all $i < j$

Tree structures: $(C, \mathbf{(?, a_1, \ldots, a_p) or (a_0)}, (d_1, \ldots, d_n), t_3 =$  $)$

extension of the former linear hierarchy schemes (1, 2)

"If then else" structures:

$(C, \mathbf{(?)}, (d_1, \ldots, d_n), t_3 =$  $)$

constraint satisfaction only: degenerate optimization problem

We can still...

- set weights on constraints

$$(C, \sum, (d_{w_1}, \ldots, d_{w_n}), t = t_1)$$

- value the elements of the search space

  all appropriate functions f associated to constraints are allowed

Remaining issues

Issues

- Modeling language
- New aggregation operators
- Provide a semantics to the solution set of:
  - prioritized clusters of constraints,
  - partial orders over prioritized constraints
- What kind of output?

Modeling language

- choose a standard language (*i.e.,* AMPL or any other language commonly used as input of classical solvers):
  - we may need to extend such a language, yet keeping easily useable and standard

- expressive enough to model soft constraints precisely (no ambiguity left)
  - possibly adjust the framework (data structure)
  - we have to make clear the correspondence between modeling and solutions: the user should get what s/he expects

Expressiveness of the modeling language

Associating values to constraints is ambiguous

- does it define the aggregation operator?
  weights $\rightsquigarrow$ weighted sum, valued CSP

$$a(d_1(x), \ldots, d_n(x)) = w_1 \times d_1(x) + \cdots + w_n \times d_n(x)$$

- or does it define an order : as preference levels for instance

$$(c_1, 5), (c_2, 3) \quad \stackrel{?}{\Leftrightarrow} \quad c_1 \succ c_2$$

but then what kind of preference: crisp? soft?

Expressiveness of the modeling language

Associating values to constraints is ambiguous

- does it define the aggregation operator?
  weights $\rightsquigarrow$ weighted sum, valued CSP

$$a(d_1(x), \ldots, d_n(x)) = w_1 \times d_1(x) + \cdots + w_n \times d_n(x)$$

- or does it define an order : as preference levels for instance

$$(c_1, 5), (c_2, 3) \quad \stackrel{?}{\Leftrightarrow} \quad c_1 \succ c_2$$

but then what kind of preference: crisp? soft?

We need slightly more detail, *e.g.,* a "code" specifying the use of an order and the kind of this order

is a tree expressive enough?

New aggregation operators

- Idea: benefit from the Multi-Criteria Decision Making (MCDM) techniques
  - no linear aggregation: weighted sum assume linear independence
  - criteria are aggregated using integrals (Choquet, Sugeno), usually discretized

Improve the semantics of prioritized constraints

Generalize the total-order scheme to more complex ones.

Improve the semantics of prioritized constraints

Generalize the total-order scheme to more complex ones.

Problems so far:

if more than one branch of priority (or a tree for instance), what is a solution?

- *a max-length consistent branch?*
- *all consistent branches?*
- *should we establish an order over different branches? but then wouldn't we restrict ourselves too much?*

Improve the semantics of prioritized constraints

Generalize the total-order scheme to more complex ones.

Problems so far:

if more than one branch of priority (or a tree for instance), what is a solution?

- *a max-length consistent branch?*
- *all consistent branches?*
- *should we establish an order over different branches? but then wouldn't we restrict ourselves too much?*

These points constitute perspectives for this project

What kind of output?

- solutions of the optimization problem

What kind of output?

- solutions of the optimization problem
- what if no solution? or too many?
  - *should we provide users with suggestions? use of qualitative information* **before** *the solving process? or* **after***?*
  - *should we run additional models in case of inconsistency?*

What kind of output?

- solutions of the optimization problem
- what if no solution? or too many?
  - *should we provide users with suggestions? use of qualitative information* **before** *the solving process? or* **after***?*
  - *should we run additional models in case of inconsistency?*

Need for clear definition of the output:

may be adjusted when experiments are carried out on real problems

Conclusion
and
Plans for future work

Conclusion

Already done

- definition of the framework for soft constraints we are going to use in this project
  - *yet, adjustments are still possible*
- prototype using this framework:
  - *tested on inconsistent camera positioning problems*
  - *using classical constraint and optimization solvers: we benefi t from their algorithmic power*

Plans for future work in Sakura

- *implement a common prototype*
  - *probably on top of* elisa: *constraint solver implemented at LINA*
- *determine the user needs: experiments on GUI problems*
  - *perform experiments on Pr. Hosobe's set of GUI problems*
  - *adjust our solving algorithms w.r.t. these experiments*
  - *get preliminary feed-back*
- *explore possible connections with methods from MCDM*
  - *expressiveness may be improved using such techniques*

Plans for future work in Sakura (cont'd)

- **apply models different from hierarchies to GUI problems**
- **extend our target application field (GUI)**
- **other projects of SCooP**
  - *integrate flexibility in speculative constraint processing*
  - *benefi t from the work performed in the area of dynamic constraints (e.g., explanations)*
- **user interface**
  - *appropriate modelling language*
  - *easy-to-use interface*

The end

Thank you for your attention

Martine Ceberio
UTEP, Computer Science Department
LINA, Nantes France
e-mail. mceberio@cs.utep.edu
http://www.cs.utep.edu/mceberio