Soft Global Constraints

Tutorial - CP 2009

Willem-Jan van Hoeve

Tepper School of Business
Carnegie Mellon University

Outline

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Constraint programming in brief
- Soft constraints
- Soft *alldifferent* constraint
- Soft *global cardinality* constraint
- Soft *regular* constraint
- Other soft global constraints
- Constraint-based local search
- Conclusions and perspectives

Main references for this tutorial

Carnegie Mellon Tepper SCHOOL OF BUSINESS

T. Petit, J.-C. Régin, and C. Bessière. Specific Filtering Algorithms for Over-Constrained Problems. In *Proceedings of the Seventh International Conference on Principles and Practice of Constraint Programming (CP)*, volume 2239 of *Lecture Notes in Computer Science*, pages 451–463. Springer, 2001.

W.-J. van Hoeve, G. Pesant, and L.-M. Rousseau. On Global Warming: Flow-Based Soft Global Constraints. *Journal of Heuristics*, 12(4):347–373, 2006.

W.-J. van Hoeve. Over-Constrained Problems. In M. Milano and P. Van Hentenryck (eds.), *Hybrid Optimization: the 10 years of CPAIOR*, chapter 6. To appear.

(contains many more references)

Constraint Programming

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Example:

variables/domains	$x_1 \in \{1,2\}$, $x_2 \in \{0,1,2,3\}$, $x_3 \in \{2,3\}$
constraints	$x_1 > x_2$
	$x_1 + x_2 = x_3$
	alldifferent(x_1,x_2,x_3)

Carnegie Mellon Tepper SCHOOL OF BUSINESS

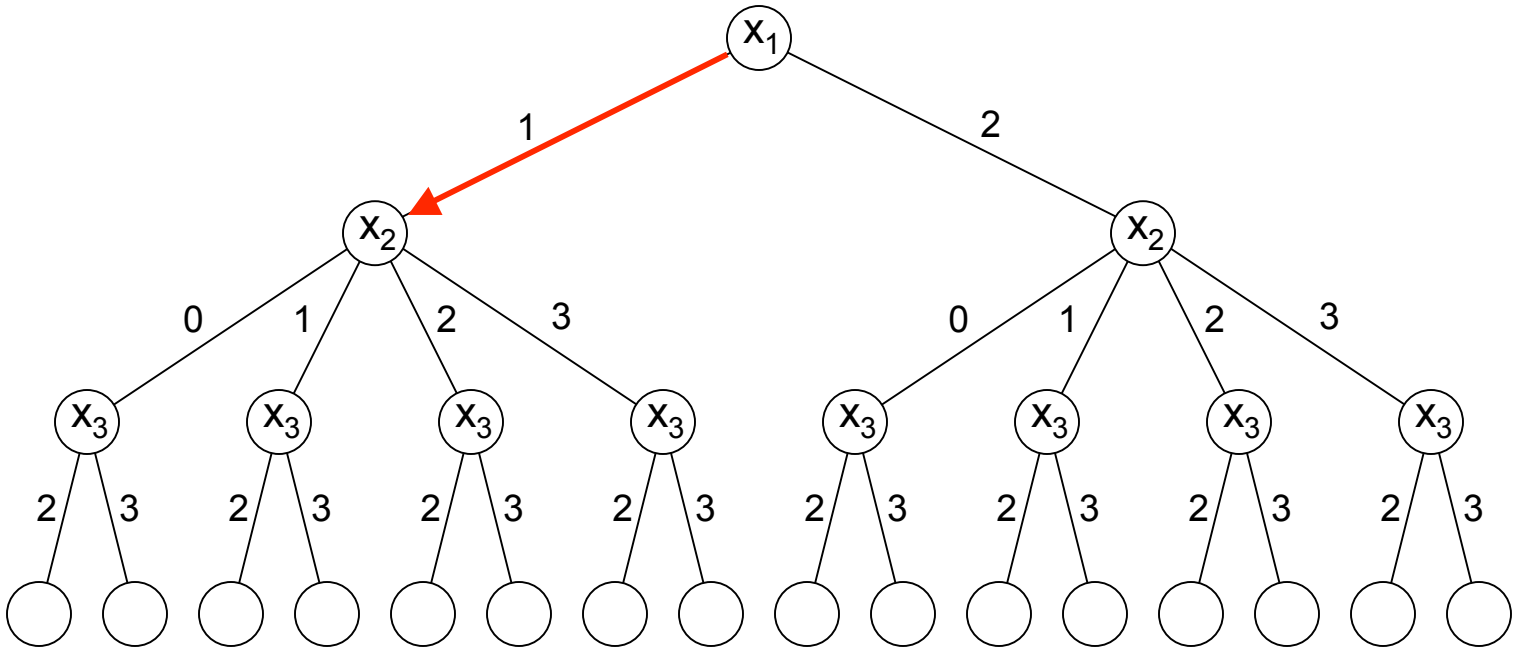
Example:

variables/domains $x_1 \in \{1,2\}$, $x_2 \in \{0,1,2,3\}$, $x_3 \in \{2,3\}$

constraints $x_1 > x_2$

$x_1 + x_2 = x_3$

alldifferent(x_1,x_2,x_3)



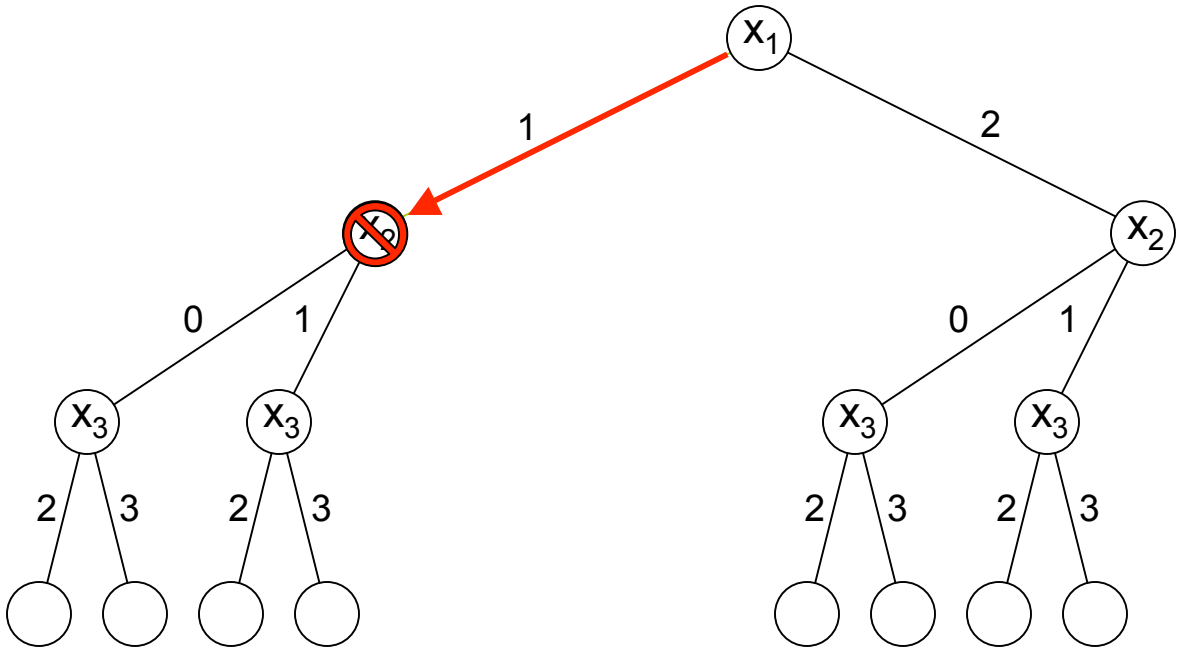
Example:

variables/domains $x_1 \in \{1\}$, $x_2 \in \{0,1\}$, $x_3 \in \{2,3\}$

constraints $x_1 > x_2$

$x_1 + x_2 = x_3$

alldifferent(x_1,x_2,x_3)



Carnegie Mellon Tepper SCHOOL OF BUSINESS

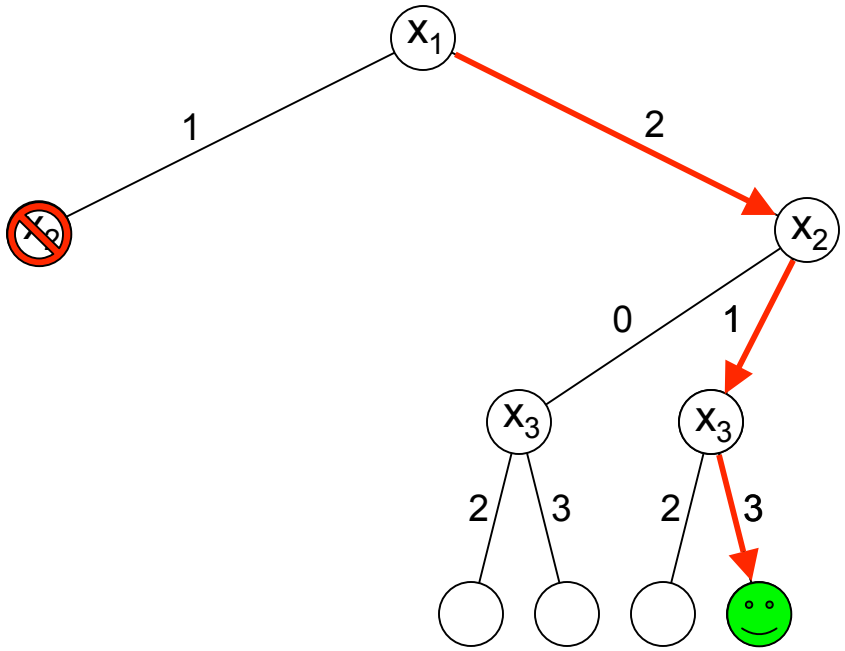
Example:

variables/domains $x_1 \in \{2\}, x_2 \in \{0,1\}, x_3 \in \{2,3\}$

constraints $x_1 > x_2$

$x_1 + x_2 = x_3$

alldifferent(x_1,x_2,x_3)



Domain Filtering

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Goal: Remove as many inconsistent values as efficiently possible for each constraint individually

More filtering: Group constraints together?

Problem: Solving arbitrary conjunction of constraints is NP-hard

Solution:

- group constraints together that *occur frequently in applications*, and capture *tractable* structure
- result is called a *global* constraint
- Examples: *alldifferent*, *cumulative*, *global-cardinality*, *among*, *sequence*, *circuit*, *element*, *regular*, ...

Soft Constraints

Over-constrained problems

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Assign seats for overbooked airplane; no solution that carries all passengers
- Create roster for employees with conflicting preferences
- Factory wants to satisfy demands of all customers, but has limited resources

  (Many industrial problems are essentially over-constrained)

A CP solver will report that no solution exists. How to find acceptable 'solution'?

- Soften (some of) the constraints of the problem
- Compute solution that minimizes conflicts or maximizes satisfaction

Soft constraints

Carnegie Mellon Tepper SCHOOL OF BUSINESS

A hard constraint must always be satisfied.

A soft constraint does not need to be satisfied, but we would like it to be.

Example:

Over-constrained CSP with two soft constraints:

$x_1 \in \{1,2\}$, $x_2 \in \{2,3\}$, $x_3 \in \{2,3\}$

(i) $x_1 > x_2$

(ii) $x_1 + x_2 = x_3$

$(x_1,x_2,x_3) = (1,2,2)$ both constraints violated

$(x_1,x_2,x_3) = (1,2,3)$ constraint (i) violated ← minimum violation or maximum satisfaction

$(x_1,x_2,x_3) = (1,3,2)$ both constraints violated

$(x_1,x_2,x_3) = (1,3,3)$ both constraints violated

$(x_1,x_2,x_3) = (2,2,2)$ both constraints violated

$(x_1,x_2,x_3) = (2,2,3)$ both constraints violated

$(x_1,x_2,x_3) = (2,3,2)$ both constraints violated

$(x_1,x_2,x_3) = (2,3,3)$ both constraints violated

'Traditional' approaches to soft CSPs

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Partial CSP [Freuder & Wallace, 1992]

  maximize number of satisfied constraints (previous example)

- Weighted CSP [Larossa, 2002]

  associate a weight to each constraint

  maximize weighted sum of satisfied constraints

- Possibilistic CSP [Schiex, 1992]

  associate weight to each constraint representing its importance

  hierarchical satisfaction of most important constraints, i.e. maximize smallest weight over all satisfied constraints

- Fuzzy CSP [Dubois et al., 1993] [Ruttkay, 1994]

  associate weight to each tuple of every constraint

  maximize smallest preference level (over all constraints)

These approaches can be modeled using valued CSPs [Schiex, Fargier, Verfaillie, 1995] or semi-rings [Bistarelli, Montanari, Rossi, 1997]

Some drawbacks

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Previous approaches typically use a functional representation or an explicit representation (tuples), and usually are limited to constraints of small arity (unary, binary, sometimes ternary)

- If constraints are represented explicitly (as in e.g., Fuzzy CSPs), domain filtering may become computationally very expensive, or not very effective
- Difficult to exploit domain structure using global constraints
- Standard CP solvers (ILOG, Comet, Eclipse, Gecode, Minion,...) are not implemented to apply the previous methods

Ideally, we would like to apply soft constraints in a CP solver just as any other constraint; using an implicit representation where possible (for example using global constraints), with efficient and effective domain filtering algorithms

From soft constraints to hard optimization constraints

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Cost-based approach [Petit, Régin, and Bessiere, 2000] (see also [Baptiste et al., 1998]):

- Introduce a cost variable for each soft constraint
- This variable represents some violation measure of the constraint
- Optimize aggregation of all cost variables (e.g., take their sum, or max)
- Use upper bound on cost variable to apply cost-based filtering (with back-propagation)

In this way

- soft global constraints become hard optimization constraints
- soft CSPs become hard COPs
- the cost variables can be used in other (meta-)constraints!

  if ($z_1 > 0$) then ($z_2 = 0$)
- we can apply classical constraint programming solvers

Note: In this framework, we can also encode the previous approaches [Petit et al., 2000]

From soft constraints to hard optimization constraints

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Example: $x \in [9000,10000]$ (x,y integer)

$y \in [0,20000]$

$x \leq y$

Let's make the constraint '$x \leq y$' soft by introducing a 'cost' variable z representing the amount of violation, as the gap between x and y.

Suppose that we impose $z \in [0,5]$. By looking at the bounds of x and y, we can immediately deduce that $y \in [8995,20000]$.

Observations:

- We can exploit the semantics of this soft constraint for efficient domain filtering algorithm (looking at the values one by one would have take at least 8995 steps)
- We can perform back-propagation of the cost variable to the other variables

Softening global constraints

Carnegie Mellon Tepper SCHOOL OF BUSINESS

This approach also works for *global* constraints

General recipe for global constraint C(X), where $X = \{x_1, x_2, \ldots, x_n\}$:

1. define violation measure μ: $D(x_1) \times D(x_2) \times \cdots \times D(x_n) \rightarrow \mathbb{R}_{\geq}$ such that $\mu=0$ iff C is satisfied and $\mu>0$ otherwise
2. define cost variable z
3. then soften the constraint C as:

   soft_C(X,z,μ) := ($\mu(X) \leq z$)

Previous example:

define violation measure $\mu_{\leq}(x,y) = \max\{x - y , 0\}$

$soft_{\leq}(x,y,z,\mu)$:= ($\mu(x,y) \leq z$), in other words, $x - y \leq z$

Remark: All information is now embedded in μ

Filtering soft global constraints

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Basic cost-based filtering algorithm for soft_C(X,z,μ) = (μ(X) ≤ z) :

for all x∈X and d∈D(x) {
  compute minimum of μ with x=d
  if minimum > max(D(z)) remove d from D(x)
  if D(x) empty **return** inconsistent
}
update min(D(z)) ≥ min(all minima of μ)
if D(z) empty **return** inconsistent
else return consistent

Always try to improve this:
Separate consistency check from filtering

Notes:

- Similar to classical filtering for 'optimization constraints'
- Filtering establishes domain consistency on soft_C, i.e., we remove all domain values That are not part of a solution to the constraint (this is the best possible)
- Different violation measures define different soft constraints, and thus different filtering algorithms
- Time complexity depends on computation of minimum of μ

Softening global constraints

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Remark: We define soft constraint as

soft_C(X,z,μ) := (μ(X) ≤ z)

Why not

soft_C(X,z,μ) := (μ(X) = z) ?

Because then domain consistency becomes often NP-hard (for 'reasonable' violation measure μ), while inequality makes it often tractable

Alternative view:

- use equality in definition, but
- establish domain consistency w.r.t. X and 'bound consistency' w.r.t. z (effectively the same approach as with inequality)

Softening global constraints

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Our goal is to design efficient filtering algorithms for soft global constraints, that establish domain consistency, using cost-based approach

Methodology:

1. Choose a global constraint
2. Define useful violation measure(s)
3. Design efficient filtering algorithm

Note: We will omit the objective 'minimize z' in our examples, and let it depend on the problem at hand what to do with z

Violation measures

Carnegie Mellon Tepper SCHOOL OF BUSINESS

General violation measures [Petit et al., 2001]:

- variable-based μ_{var}
  - minimum number of variables that need to change their value in order to satisfy the constraint (minimum Hamming distance to any solution)
  - Example for alldifferent(x_1,x_2,x_3,x_4): $\mu_{var}(1,2,2,2) = 2$
- decomposition-based μ_{dec}
  - number of violated constraints in binary decomposition

    e.g., $x_i \neq x_j$ for all $i<j$ is binary decomposition of alldifferent(x_1,...,x_n)
  - Example for alldifferent (x_1,x_2,x_3,x_4): $\mu_{dec}(1,2,2,2) = 3$

Other, more problem-specific violation measures [Beldiceanu and Petit, 2004]:

- object-based
  - for example, number of late activities in scheduling problem
- graph-based
  - for example, number of SCCs

Outlook

Carnegie Mellon Tepper SCHOOL OF BUSINESS

constraint	violation measure	consistency check	domain consistency	reference
`alldifferent`		$O(m\sqrt{n})$	$O(m)$	[Régin, 1994]
`soft_alldifferent`	variable-based	$O(m\sqrt{n})$	$O(m)$	[Petit et al., 2001]
`soft_alldifferent`	decomposition-based	$O(mn)$	$O(m)$	[van Hoeve, 2004]
`gcc`		$O(m\sqrt{n})$	$O(m)$	[Quimper et al., 2004]
`soft_gcc`	variable-based	$O(m\sqrt{n})$	$O(m)$	[Zanarini et al., 2006]
`soft_gcc`	value-based	$O(m\sqrt{n})$	$O(m)$	[Zanarini et al., 2006]
`regular`		$O(m)$	$O(m)$	[Pesant, 2004]
`soft_regular`	variable-based	$O(m)$	$O(m)$	[van Hoeve et al., 2006a]
`soft_regular`	edit-based	$O(m)$	$O(m)$	[van Hoeve et al., 2006a]

Here n is the number of variables, m is proportional to the sum of the domain sizes.

Time complexity of soft versions similar to hard versions!

Filtering algorithms for the soft alldifferent constraint

1. variable-based violation measure
2. decomposition-based violation measure

Softening alldifferent

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Example: $x_1 \in \{a,b\}$, $x_2 \in \{a,b\}$,
$x_3 \in \{a,b\}$, $x_4 \in \{b,c\}$,
alldifferent(x_1,x_2,x_3,x_4)

Over-constrained CSP; we need to soften the alldifferent constraint:

introduce cost variable z

define some violation measure, say μ

Result: $x_1 \in \{a,b\}$, $x_2 \in \{a,b\}$,
$x_3 \in \{a,b\}$, $x_4 \in \{b,c\}$,
$z \in \{0,1,2,3,....\}$
soft_alldifferent(x_1,x_2,x_3,x_4,z,μ)

We will consider the variable-based and graph-based violation measures

Violation measures

variable-based violation measure μ_{var}:

"minimum number of variables that need to change value in order to satisfy the constraint"

$x_1 \in \{a,b\}$, $x_2 \in \{a,b\}$,

$x_3 \in \{a,b\}$, $x_4 \in \{b,c\}$,

alldifferent(x_1,x_2,x_3,x_4)

x_1	x_2	x_3	x_4	μ_{var}
a	a	a	b	2
a	a	b	b	2
a	a	b	c	1
a	b	a	b	2
a	b	a	c	1
...	...	...	...	...

Violation measures

decomposition-based violation measure μ_{dec}:

"number of violated constraints in binary decomposition"

$x_1 \in \{a,b\}$, $x_2 \in \{a,b\}$,
$x_3 \in \{a,b\}$, $x_4 \in \{b,c\}$,
alldifferent(x_1,x_2,x_3,x_4) ⟶ $x_1 \neq x_2, x_1 \neq x_3, x_1 \neq x_4,$ $x_2 \neq x_3, x_2 \neq x_4, x_3 \neq x_4$

x_1	x_2	x_3	x_4	μ_{var}	μ_{dec}
a	a	a	b	2	3
a	a	b	b	2	2
a	a	b	c	1	1
a	b	a	b	2	2
a	b	a	c	1	1
...	...	...	...	...	...

distinction

Filtering algorithm for soft alldifferent using variable-based violation measure

Filtering for variable-based violation

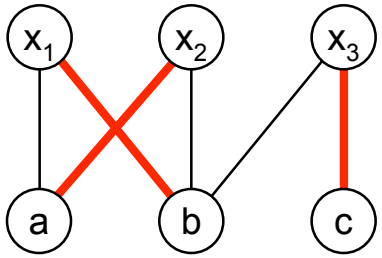
Carnegie Mellon Tepper SCHOOL OF BUSINESS

Theorem [Régin, 1994]:

solution to *hard* alldifferent ⇔ maximum matching in value graph

$x_1 \in \{a,b\}$, $x_2 \in \{a,b\}$, $x_3 \in \{b,c\}$

alldifferent(x_1,x_2,x_3)



matching in graph:

subset of non-touching edges

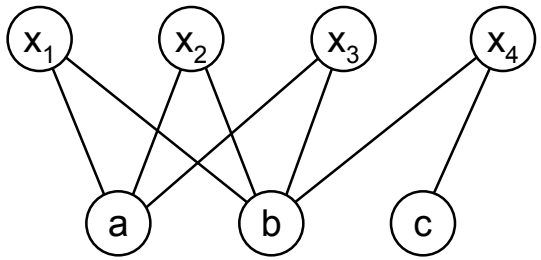
Filtering for hard alldifferent: remove all edges (and corresponding domain values) that are not in any maximum matching

But what can we do if there is no solution?

$x_1 \in \{a,b\}$, $x_2 \in \{a,b\}$,

$x_3 \in \{a,b\}$, $x_4 \in \{b,c\}$,

alldifferent(x_1,x_2,x_3,x_4)



Filtering for variable-based violation

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Fortunately, we can re-use these techniques

Theorem [Petit et al., 2001]: for alldifferent

minimum value of $\mu_{var}(x_1,x_2,...,x_n)$

=

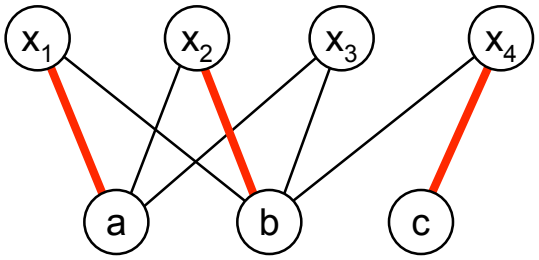
n - value of maximum matching in value graph

Example:

$x_1 \in \{a,b\}$, $x_2 \in \{a,b\}$,

$x_3 \in \{a,b\}$, $x_4 \in \{b,c\}$,

alldifferent(x_1,x_2,x_3,x_4)



n - value of maximum matching = 4 – 3 = 1,

minimum value for μ_{var} is 1 (see before)

Filtering for variable-based violation

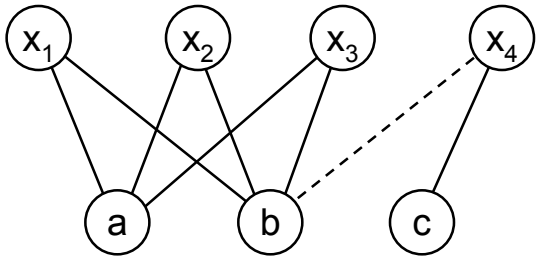
Filtering rules for $x_1,x_2,...,x_n$ in soft_alldifferent($x_1,x_2,...,x_n,z,\mu_{var}$):

Let M be maximum matching in value graph

- if (n - |M|) > max(D(z)) then constraint is inconsistent
- if (n - |M|) < max(D(z)) then *all* domain values are consistent
  (namely, if we change the value of one variable μ_{var} can increase at most one)
- if (n - |M|) = max(D(z)) then domain value d ∈ D(x) is consistent iff
  edge (x,d) belongs to some maximum matching in value graph

Example:

$x_1 \in \{a,b\}$, $x_2 \in \{a,b\}$,
$x_3 \in \{a,b\}$, $x_4 \in \{b,c\}$, $z \in \{0,1\}$
soft_alldifferent($x_1,x_2,x_3,x_4,z,\mu_{var}$)



|M| = 3, minimum μ_{var} = 4 - 3 = 1, which is equal to max(D(z))
edge (x_4,b) not in maximum matching
note that also value 0 in D(z) is inconsistent

Filtering for variable-based violation

Carnegie Mellon Tepper SCHOOL OF BUSINESS

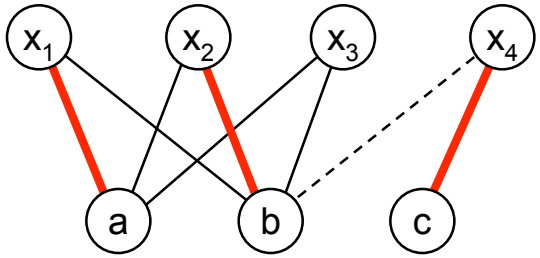
To find consistent edges (same as hard alldifferent):

Theorem [Petersen, 1891]:

Given maximum matching M, any edge e is in some maximum matching iff

- e in M, or
- e on even-length M-alternating path starting from M-free vertex, or
- e on even-length M-alternating circuit

$x_1 \in \{a,b\}$, $x_2 \in \{a,b\}$,
$x_3 \in \{a,b\}$, $x_4 \in \{b,c\}$, $z \in \{0,1\}$
soft_alldifferent(x_1,x_2,x_3,x_4,z,μ_{var})



even-length M-alternating path starting from M-free vertex: $x_3 - b - x_2 - a - x_1$
even-length M-alternating circuit: $x_1 - a - x_2 - b - x_1$
again, edge (x_4,b) not in maximum matching

Filtering for variable-based violation

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Algorithm to make soft_alldifferent(x_1,x_2,...,x_n,z,μ_{var}) domain consistent

[Petit et al., 2001]:

compute maximum matching M in value graph ← O(m$\sqrt{}$ n) [Hopcroft/Karp, 1973]

if (n - |M| > max(D(z)) **return** inconsistent

else if (n - |M| < max(D(z)) **return** consistent

else { ← O(m+n) [Tarjan, 1972]

remove all edges (and corresponding domain values) not in any maximum size matching

if some domain is empty **return** inconsistent

}

update min(D(z)) ≥ n - |M|

if D(z) is empty **return** inconsistent

else return consistent

Remarks: Separation between constraint check and filtering; incrementality

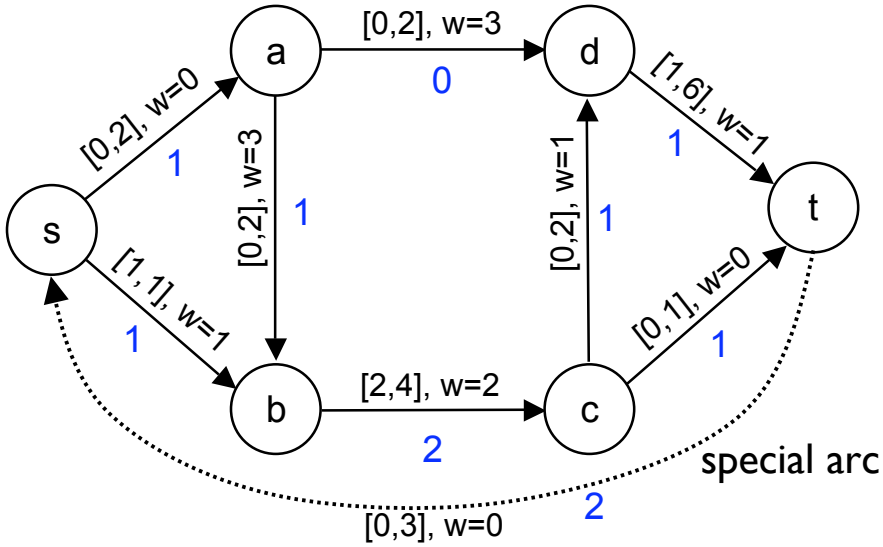
Intermezzo: Network Flows

Let G=(V,A) be a directed graph with vertex set V and arc set A. To each arc $a \in A$ we assign a capacity function [d(a),c(a)] and a weight function w(a).

Let $s,t \in V$. A function $f: A \to \mathbb{R}$ is called an s-t flow (or a flow) if

- $f(a) \geq 0$ for all $a \in A$
- $\sum_{a \text{ enters } v} f(a) = \sum_{a \text{ leaves } v} f(a)$ for all $v \in V$ (flow conservation)
- $d(a) \leq f(a) \leq c(a)$ for all $a \in A$

Define the cost of flow f as $\sum_{a \in A} w(a)f(a)$. A minimum-cost flow is a flow with minimum cost.



flow (in blue) with cost 10

special arc to ensure flow conservation

Alternative graph representation for alldifferent

Fact: matching in bipartite graph ⇔ integer flow in directed bipartite graph

Step 1: direct edges from X to D(X)

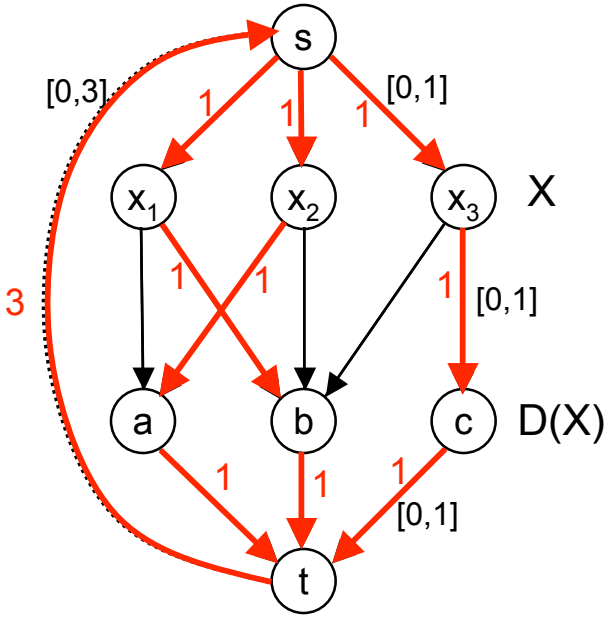
Step 2: add a source s and sink t

Step 3: connect s to X, and D(X) to t

Step 4: add special arc (t,s)

all arcs have capacity [0,1] and weight 0

except arc (t,s) with capacity [0, min{|X|,|D(X)|}]



Decomposition-based violation for alldifferent

Start from basic network:

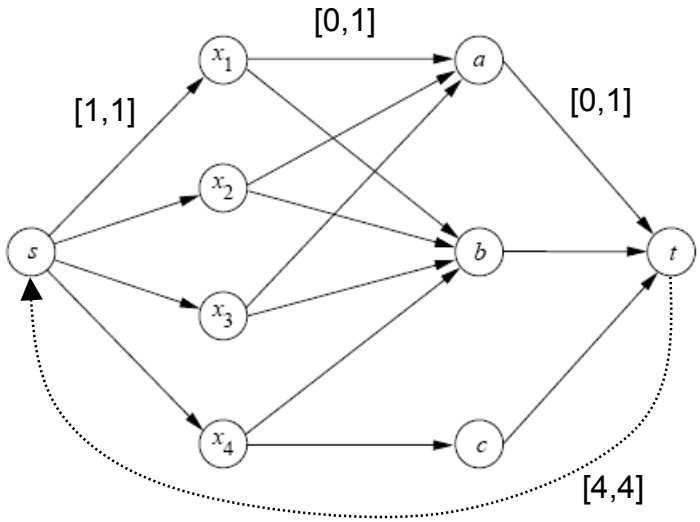
Example:

$x_1 \in \{a,b\}$, $x_2 \in \{a,b\}$,

$x_3 \in \{a,b\}$, $x_4 \in \{b,c\}$, $z \in \{0,1\}$

soft_alldifferent(x_1,x_2,x_3,x_4,z,μ_{dec})

e.g. x_1=x_2=x_3=x_4=b should give violation cost 6



What can we do?

Options:

- manipulate capacities
- add/remove arcs
- add weights to some arcs

Graph representation

We can do the following [v.H., 2004]:

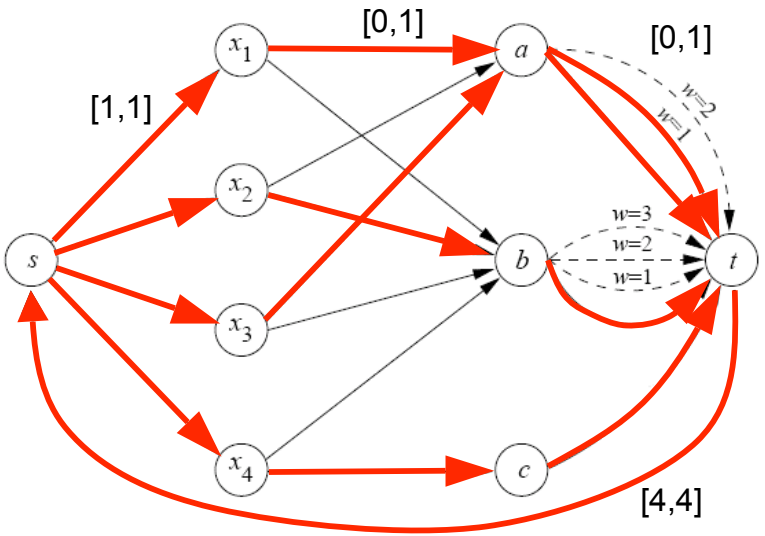
- for all $d \in D(X)$ with k incoming arcs: add arcs $(d,t)_1$, $(d,t)_2$,...,$(d,t)_{k-1}$, with capacity [0,1], and weight i for arc $(d,t)_i$ (i=1,...,k-1)
- for all $x \in X$: set capacity of arc (s,x) to [1,1], i.e. force x into solution
- minimum-cost flow ⇔ solution to soft_alldifferent minimizing μ_{dec}

Example:

$x_1 \in \{a,b\}$, $x_2 \in \{a,b\}$,

$x_3 \in \{a,b\}$, $x_4 \in \{b,c\}$, $z \in \{0,1\}$

soft_alldifferent(x_1,x_2,x_3,x_4,z,μ_{dec})



Note: minimum-cost flow in this graph can be computed in O(nm) time

Filtering for decomposition-based violation

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Algorithm to make soft_alldifferent(x_1,x_2,...,x_n,z,μ_{dec}) domain consistent:

for all arcs (x_i,d) {

set capacity of (x_i,d) to [1,1] // force flow to represent x_i=d

compute min-cost flow f

if cost(f) > max(D(z)) remove d from D(x_i)

if D(x_i) is empty **return** inconsistent

}

update min(D(z)) ≥ min(cost(f))

if D(z) is empty **return** inconsistent

else return consistent

Improvements: Can use flow theory to make incremental, and to separate consistency check (O(mn) time), and filtering (O(m) time)

Filtering algorithms for the soft global cardinality constraint

1. variable-based violation measure
2. value-based violation measure

Global cardinality constraint

Carnegie Mellon Tepper SCHOOL OF BUSINESS

A global cardinality constraint is defined as gcc(X, *l*, *u*)

where X is set of variables $\{x_1,x_2,...,x_n\}$

l and *u* are arrays of (constant) integers, such that

domain value d is used between l_d and u_d times.

$x_1 \in \{0,1\}$, $x_2 \in \{0,1,2\}$, $x_3 \in \{1,2\}$, $x_4 \in \{1,2,3\}$

gcc($\{x_1,x_2,x_3,x_4\}$, [0,1,0,1], [1,2,2,1])

a solution: $x_1=0$, $x_2=1$, $x_3=2$, $x_4=3$

Efficient domain consistency filtering algorithm for gcc in O(mn) time [Régin, 1996], improved to $O(m\sqrt{n})$ time [Quimper et al., 2004]

Special case: alldifferent = gcc with *l*=[0,0,...,0] and *u*=[1,1,...,1]

Softening global cardinality constraint

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Example:

$x_1 \in \{1,2\}$, $x_2 \in \{1\}$, $x_3 \in \{1,2\}$, $x_4 \in \{1\}$

gcc($\{x_1,x_2,x_3,x_4\}$, [1,3], [2,5])

Over-constrained CSP; we need to soften the gcc:

introduce cost variable z

define some violation measure, say μ

Result:

$x_1 \in \{1,2\}$, $x_2 \in \{1\}$, $x_3 \in \{1,2\}$, $x_4 \in \{1\}$

$z \in \{0,1,2,3,....\}$

soft_gcc($\{x_1,x_2,x_3,x_4\}$, [1,3], [2,5], z, μ)

What is a good violation measure?

Violation measures for gcc

Carnegie Mellon Tepper SCHOOL OF BUSINESS

General violation measures:

- variable-based μ_{var}:
  - minimum number of variables that need to change their value in order to satisfy the constraint
- decomposition-based μ_{dec}:
  - minimum number of violated constraints in binary decomposition

We can apply μ_{var}, but what is binary decomposition of gcc?

Example:

$x_1 \in \{1,2\}$, $x_2 \in \{1\}$, $x_3 \in \{1,2\}$, $x_4 \in \{1\}$

gcc($\{x_1,x_2,x_3,x_4\}$, [1,3], [2,5])

$\mu_{var}(1,1,1,1) = 3$ (e.g. change x_1, x_2 and x_3 to 2)

$\mu_{var}(2,1,2,1) = 1$ (e.g. change x_2 to 2)

Violation measures for gcc

Rewrite μ_{var} in terms of 'shortage' and 'excess' functions [v.H. et al., 2006].
Let $X = \{x_1, x_2, ..., x_n\}$ and $d \in D(X)$. Then

$$s(X,d) = \begin{cases} l_d - |\{x_i \mid x_i = d\}| & \text{if } |\{x_i \mid x_i = d\}| \leq l_d \\ 0 & \text{otherwise} \end{cases}$$

'shortage' of value d

$$e(X,d) = \begin{cases} |\{x_i \mid x_i = d\}| - u_d & \text{if } |\{x_i \mid x_i = d\}| \geq u_d \\ 0 & \text{otherwise} \end{cases}$$

'excess' of value d

We have $\mu_{\text{var}}(X) = \max\left(\sum_d s(X,d), \sum_d e(X,d) \right)$

Example: $x_1 \in \{1,2\}$, $x_2 \in \{1\}$, $x_3 \in \{1,2\}$, $x_4 \in \{1\}$
gcc($\{x_1,x_2,x_3,x_4\}$, [1,3], [2,5])

for X=(1,1,1,1): s(X,1)=0, s(X,2)=3, e(X,1)=2, e(X,2)=0,
hence μ_{var}(1,1,1,1) = max(3,2)=3

Violation measures for gcc

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Example: $x_1 \in \{1,2\}$, $x_2 \in \{1\}$, $x_3 \in \{1,2\}$, $x_4 \in \{1\}$

gcc($\{x_1,x_2,x_3,x_4\}$, [2,3], [3,5])

for X=(1,1,1,1): s(X,1)=0, s(X,2)=3, e(X,1)=1, e(X,2)=0,

hence $\mu_{var}(1,1,1,1) = \max(3,1)=3$

but this is wrong! we can never assign 2+3=5 values to only 4 variables!

Problem: μ_{var} only applicable if $\sum_d l_d \le |X| \le \sum_d u_d$

Remedy: Define new 'value-based' violation measure

$$\mu_{\text{val}}(X) = \sum_d \left(s(X,d) + e(X,d)\right)$$

Example (cont'd): $\mu_{val}(1,1,1,1) = 0+3+1+0=4$

Filtering algorithm for soft gcc using variable-based violation measure

Filtering for variable-based violation

Carnegie Mellon Tepper SCHOOL OF BUSINESS

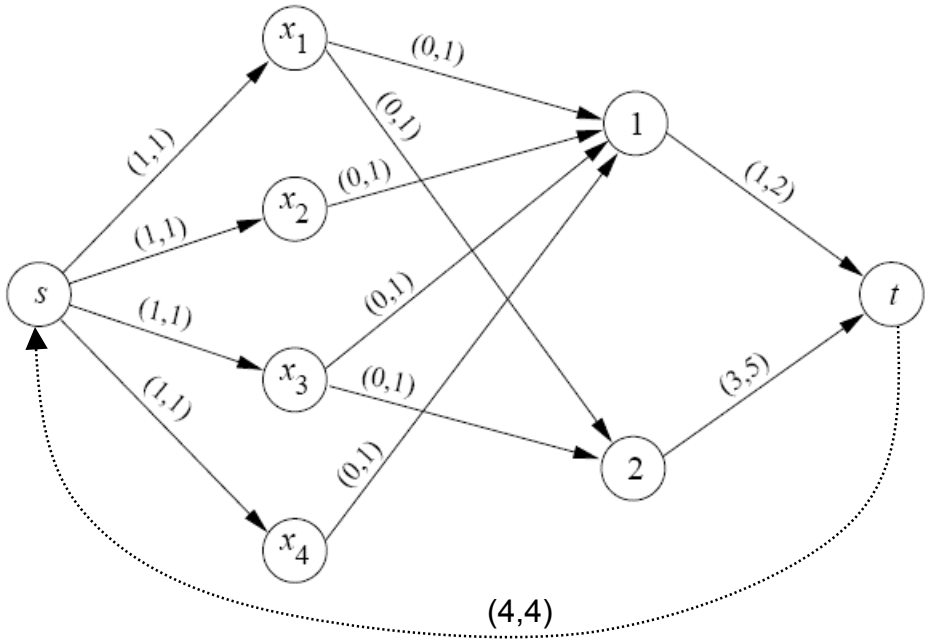
Theorem [Régin, 1996]:

solution to hard gcc ⇔ flow in particular network

Example:

$x_1 \in \{1,2\}$, $x_2 \in \{1\}$, $x_3 \in \{1,2\}$, $x_4 \in \{1\}$

gcc($\{x_1,x_2,x_3,x_4\}$, [1,3], [2,5])



for each arc a, capacity indicated as (d(a), c(a))

over-constrained gcc:

no flow exists

what can we do?

insert weighted arcs?

Filtering for variable-based violation

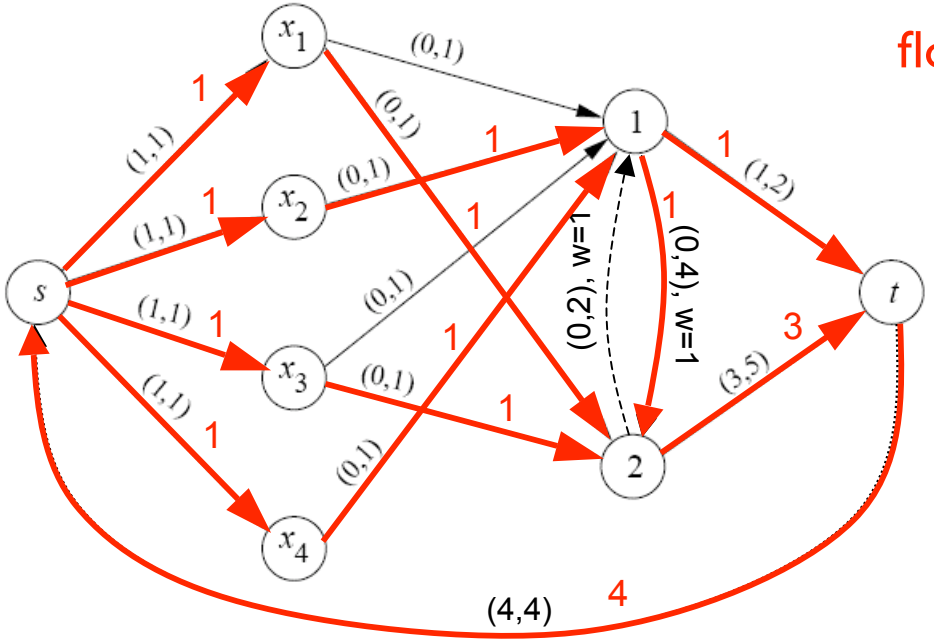
We can do the following [v.H., Pesant, and Rousseau, 2006]:

- for all pairs u,v ∈ D(X): add arcs (u,v) with capacity [0,|X|], and weight 1
- minimum-cost flow ⇔ solution to soft_gcc minimizing μ_{var}

Example:

$x_1 \in \{1,2\}$, $x_2 \in \{1\}$, $x_3 \in \{1,2\}$, $x_4 \in \{1\}$

soft_gcc($\{x_1,x_2,x_3,x_4\}$, [1,3], [2,5], z, μ_{var})



flow with cost 1

Filtering for variable-based violation

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Domain consistency filtering algorithm for soft_gcc(X,*l*,*u*,z,μ_{var}) :

- Similar to network-based soft_alldifferent
- First compute minimum flow to check for consistency, in O(n(m+n log n)) time
- Filter all inconsistent arcs, in O(n(m+n log n)) time (using flow theory)

This can be improved to O(m$\sqrt{}$n) for consistency check and O(m) for filtering (will show later)

Filtering algorithm for soft gcc using value-based violation measure

Filtering for value-based violation (gcc)

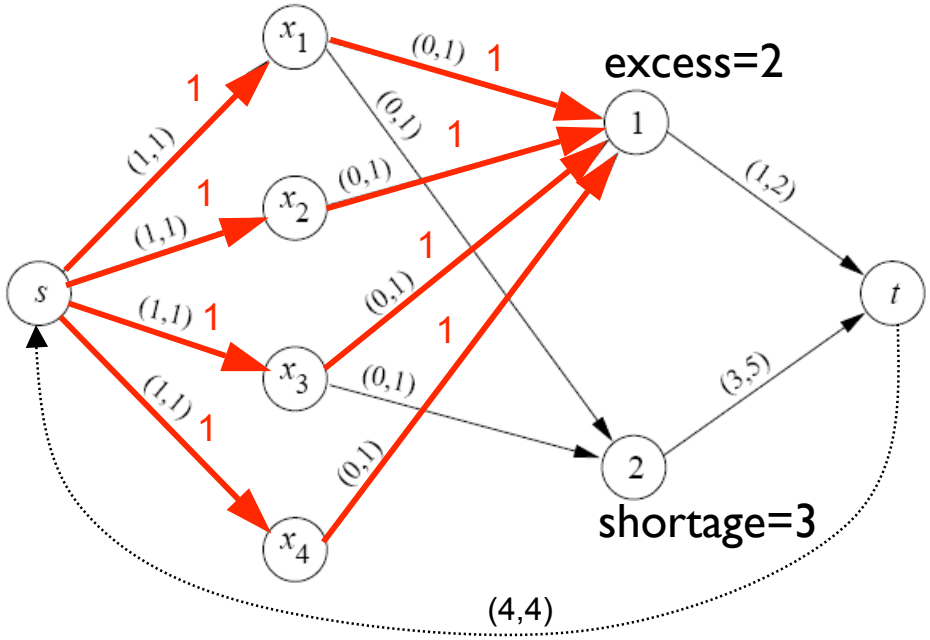
Carnegie Mellon Tepper SCHOOL OF BUSINESS

Again, start from basic network

Example: $x_1 \in \{1,2\}$, $x_2 \in \{1\}$, $x_3 \in \{1,2\}$, $x_4 \in \{1\}$

gcc($\{x_1,x_2,x_3,x_4\}$, [1,3], [2,5])

e.g. when all variables are assigned to 1, μ_{val}=2+3=5



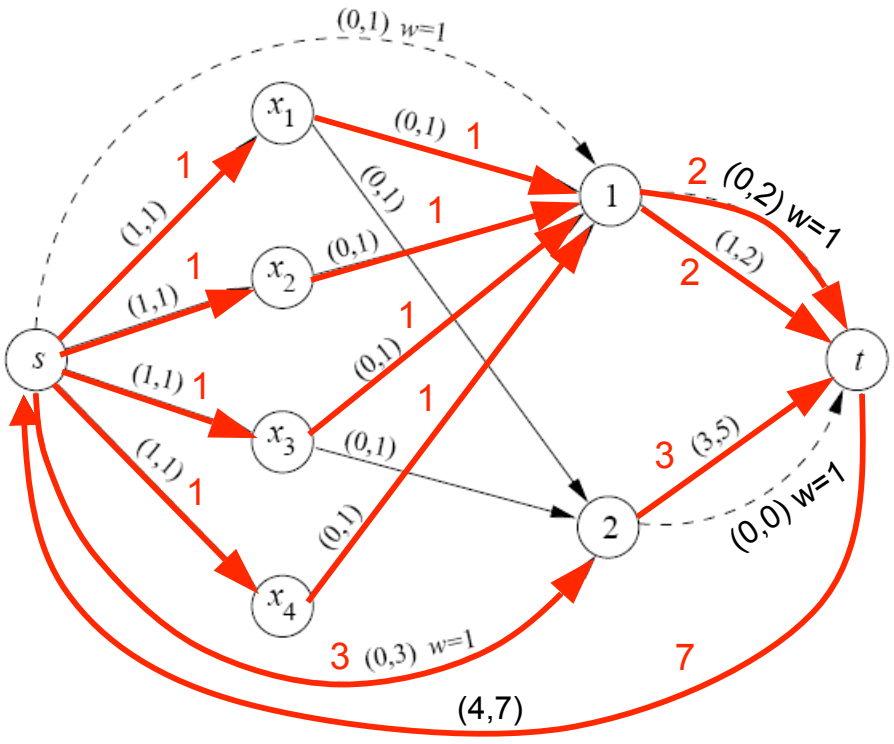
where to insert weighted arcs?

$$\mu_{\text{val}}(X) = \sum_{d} (s(X,d) + e(X,d))$$

Filtering for value-based violation (gcc)

We can do the following [v.H. et al., 2006]:

- for all d∈D(X): add arcs (s,d) with capacity [0,l_d], and weight 1
- for all d∈D(X): add arcs (d,t) with capacity [0,max{|X|-u_d,0}], and weight 1
- minimum-cost flow ⇔ solution to soft_gcc minimizing μ_{val}



flow with cost 5

Filtering for value-based violation

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Domain consistency filtering algorithm for soft_gcc(X,*l*,*u*,z,μ_{val}) :

- Again, similar to network-based soft_alldifferent and var-based soft_gcc
- First compute minimum flow to check for consistency,
  in O((n+k)(m+n log n)) time, where k = |D(X)|
- Filter all inconsistent arcs, in O(n(m+n log n)) time (using flow theory)

Also this can be improved to O(m$\sqrt{}$n) for consistency check and O(m) for filtering

Alternative approach

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Zanarini, Milano and Pesant [2006] present an alternative approach that improves the running time to $O(m\sqrt{n})$. Their approach extends the algorithm of Petit et al. [2001] for the variable-based soft alldifferent:

- Apply result of Quimper et al. [2004] to compute two maximum flows in the value graph (they correspond to 'capacitated matchings'):
  - one flow w.r.t. lower bounds l: f_l (i.e., capacity of node i is l_i)
  - one flow w.r.t. upper bounds u: f_u (i.e., capacity of node i is u_i)
- Minimum shortage is
  - 0 if $\sum_d l_d = f_l$ i.e., all lower bounds are respected
  - $\sum_d l_d - f_l$ otherwise
- Minimum excess is
  - 0 if $|X| = f_u$ i.e., all upper bounds are respected
  - $|X| - f_u$ otherwise

Alternative approach

- Theorem: minimum shortage and minimum excess can be combined to form μ_{val} and μ_{val}
- Filtering does case analysis, and apply rules:
  - forcing x=d increases μ_{var} at most 1
  - forcing x=d increases μ_{val} at most 2

Filtering algorithms for the soft regular constraint

1. variable-based violation measure
2. edit-based violation measure

Regular constraint

A regular language can be represented by a deterministic finite automaton (DFA):

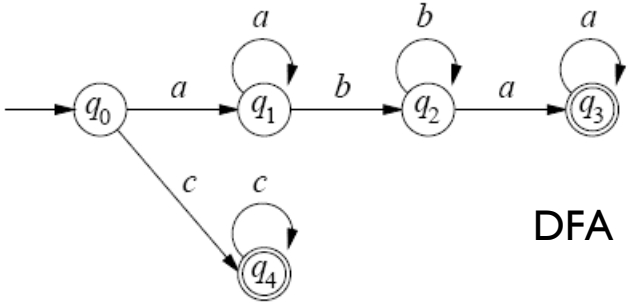
automaton accepts string ⇔ string belongs to regular language

Example:

q_0 is start state, q_3 and q_4 are end states

arrow is transition from one state to another

each transition has a label



DFA

e.g.string aabbaa and ccc accepted, string caabbac not accepted

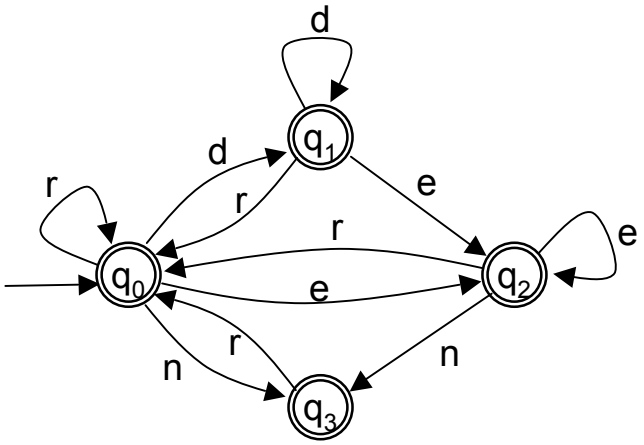
Given a DFA, the constraint regular(x_1,x_2,...,x_n,DFA) imposes that the

'string' $x_1x_2\cdots x_n$ is accepted by DFA [Pesant, 2004]

Regular constraint - application

Consider the problem to roster nurses in a hospital

- each nurse works at most one shift a day
- each shift contains 8 consecutive hours
  - day shift: 8am-4pm
  - evening shift: 4pm-12am
  - night shift: 12am-8am
- after a night shift, nurse needs to take one day rest
- after an evening shift, nurse may not work a day shift



Feasible roster (7 days) for a nurse: day - day - evening - night - rest - day – day

For each nurse, introduce variables $X = \{x_1, x_2, ..., x_7\}$ representing shift on day 1,2,...,7 with domains $D(x) = \{r,d,e,n\}$ for all $x \in X$

Model the pattern with regular(X, DFA)

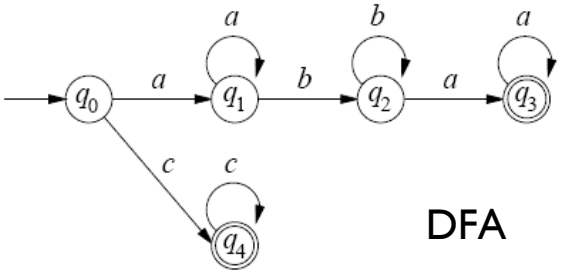
Filtering hard regular constraint

Theorem [Pesant, 2004]:

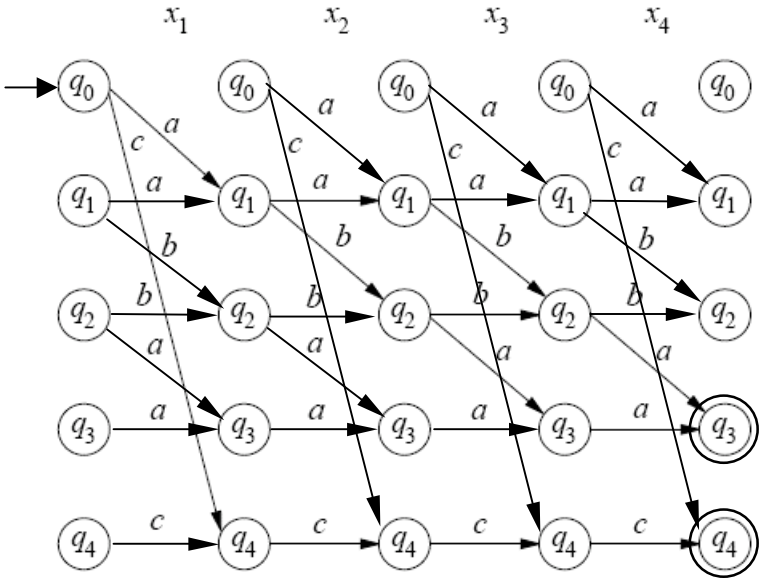
solution to regular ⇔ path from q_0 to 'end vertex' in layered graph

$x_1 \in \{a,b,c\}$, $x_2 \in \{a,b,c\}$, $x_3 \in \{a,b,c\}$, $x_4 \in \{a,b,c\}$

regular(x_1,x_2,x_3,x_4,DFA)



DFA



Filtering: remove all arcs whose label is not supported by domain value and vice versa (linear time in size of graph)

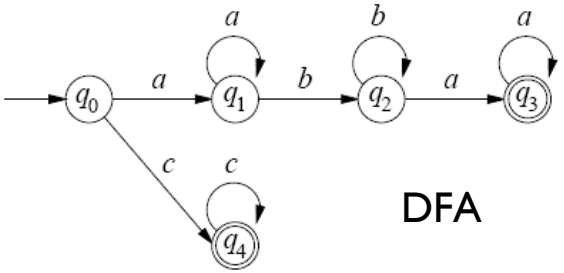
Soft regular constraint

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Example: over-constrained CSP

$x_1 \in \{a,c\}$, $x_2 \in \{a,b,c\}$, $x_3 \in \{a,b,c\}$, $x_4 \in \{b,d\}$

regular(x_1,x_2,x_3,x_4,DFA)



Soften the regular constraint, as usual:

$x_1 \in \{a,c\}$, $x_2 \in \{a,b,c\}$, $x_3 \in \{a,b,c\}$, $x_4 \in \{b,d\}$, $z \in \{0,1,2,...\}$

soft_regular(x_1,x_2,x_3,x_4,DFA,z,μ)

Again, what violation measure μ can we apply?

Variable-based violation measure? Yes, can work: This corresponds to the smallest Hamming distance of the string to any string in the language

Violation measures for regular

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Example: Consider the regular language of strings that consist of alternating pairs of *a* and *b*, for example *aabbaabb*.

The string *abbaabbaab* does not belong to the regular language, and the corresponding 'variable-based violation' would be 5 (the length of the string divided by 2).

For such cases, the Hamming distance is a bad measure, as it depends on the *size* of the string rather than its structure.

Alternative: Edit distance. Smallest number of insertions, deletions, and substitutions to change one string in another.

Edit-based violation measure [v.H. et al. 2006]: Minimum edit distance of given string to any string in the regular language

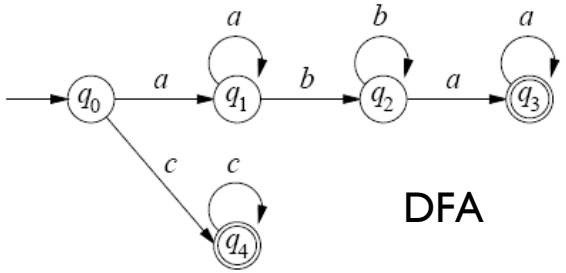
Example above: Edit-based violation is 2.

Filtering algorithm for soft regular constraint using variable-based violation measure

Filtering for variable-based violation

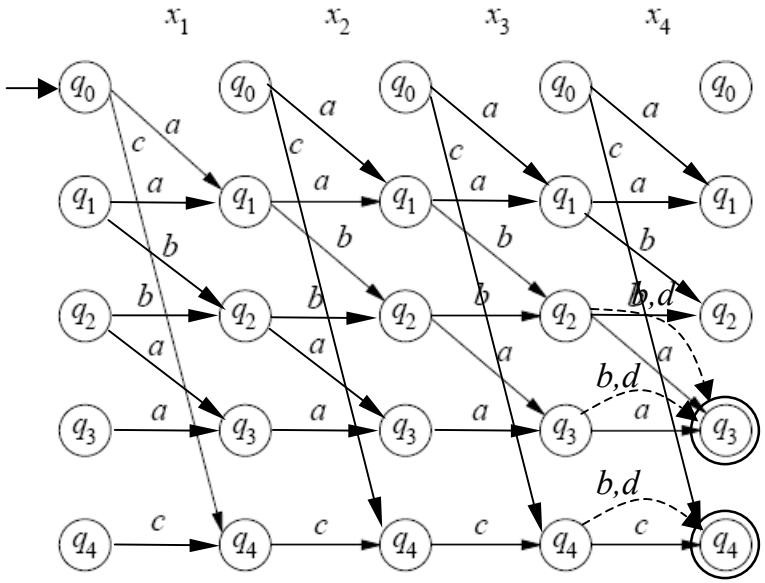
$x_1 \in \{a,b,c\}$, $x_2 \in \{a,b,c\}$, $x_3 \in \{a,b,c\}$, $x_4 \in \{b,d\}$

regular(x_1,x_2,x_3,x_4,DFA)



Can we add weighted arcs to the unfolded graph?

- e.g., we want to use value b or d for x_4
- we must start from q_0
- we must end in end vertex q_3 or q_4
- we want to capture μ_{var}:
  - allowed path must have cost of μ_{var}



For clarity, remove clearly infeasible arcs without touching the domains

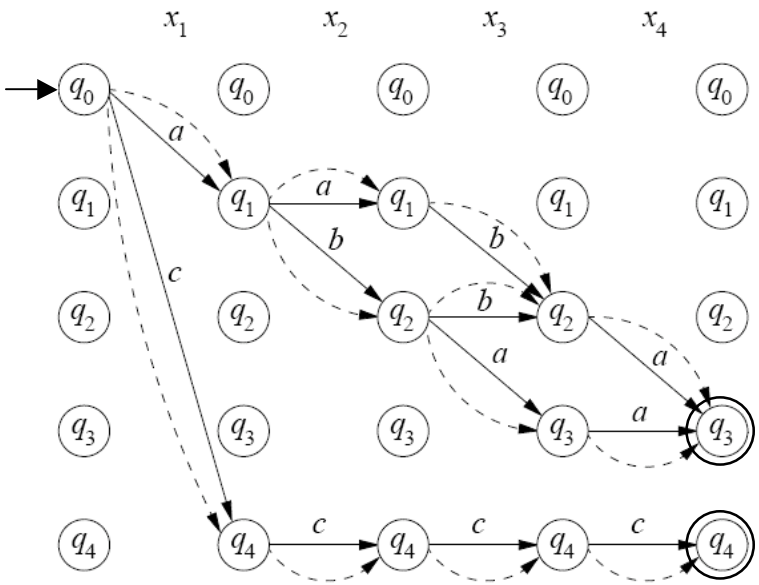
Can we insert weighted arcs now?

Filtering for variable-based violation

More formally [v.H. et al., 2006]:

- for every arc in the unfolded graph, add parallel arc with 'void' label and weight 1
- minimum-cost path from q_0 to end vertex ⇔ solution to soft_regular minimizing μ_{var}

filtered unfolded graph



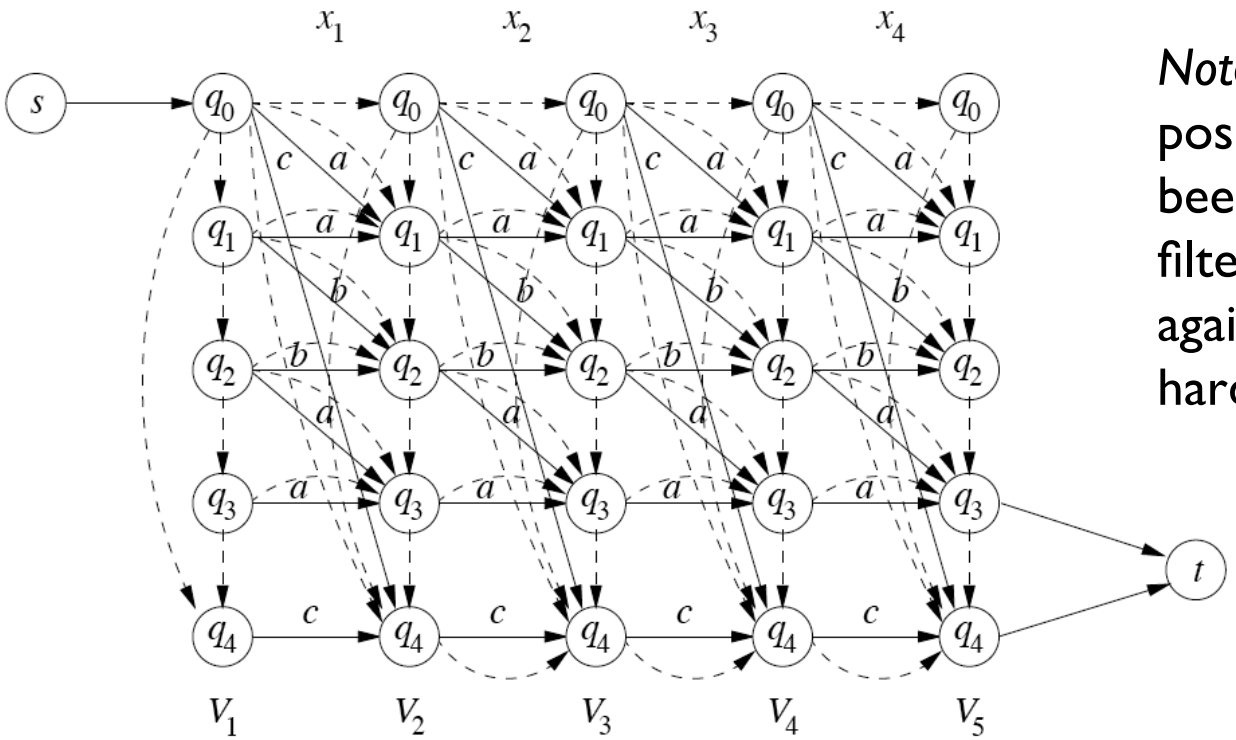
Filtering (domain consistency) same time complexity as hard regular

Filtering algorithm for soft regular constraint using edit-based violation measure

Filtering for edit-based violation

Add following arcs with unit weight [v.H. et al. 2006]

- substitution: parallel arcs to existing ones (as before)
- insertion: arcs connecting state with itself in next layer (if no substitution yet)
- deletion: for all arcs in DFA, add arc *within* layer
- minimum-cost path from q_0 to end vertex ⇔ solution to soft_regular minimizing μ_{edit}



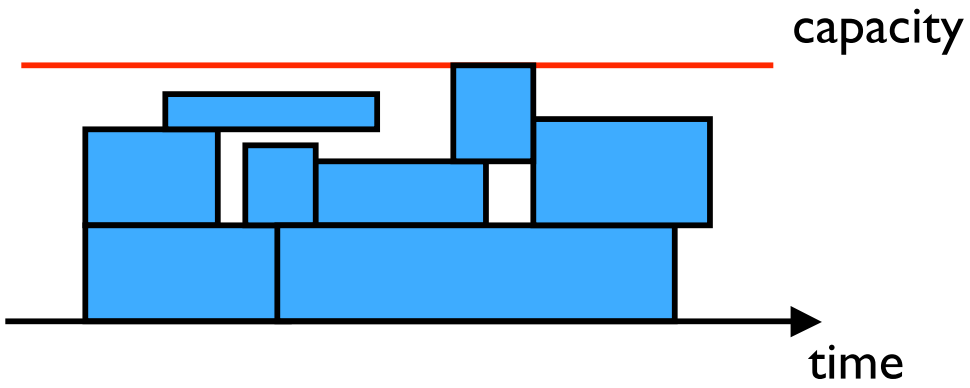
Note: Directed circuits (with positive weight) may have been formed. However, filtering (domain consistency) again same time complexity as hard regular.

Other soft global constraints

Soft Cumulative

Cumulative constraint can be used to model resource constraints in scheduling and packing (see Petr Vilim's talk yesterday)

Given set of activities, each with processing time, resource consumption, earliest start time and latest end time, assign an execution time to each activity so that a given resource does not exceed its capacity



Violation measures:

- Number of late activities [Baptiste, Le Pape, and Péridy, 1998]
  First soft global constraint, 'heuristic' filtering to enforce on time/late.
- Variable-based, on overcapacity of resource [Petit and Poder 2008]
  Provide implementation and compare against value-based approach.

Other scheduling-related soft global constraints

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Soft precedence constraint [Lesaint, Mehta, O'Sullivan, Quesada, and Wilson, 2009]

- Groups together hard and soft precedence constraint for telecommunications application
- Violation based on total weight of violated soft precedence constraints.
- NP-hard, but provide filtering rules for upper and lower bound

Soft constraints for timetabling application [Cambazard,Hebrard,O'Sullivan, and Papadopoulos, 2008]

- Three problem-specific soft constraints
- Soft constraints are used to exploit good bounds for this problem class

More soft global constraints

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Soft sequence constraint [Maher, Narodytska, Quimper, and Walsh, 2008]

- Given ordered set of variables, each subsequence of q consecutive variables must take at least l and at must u values from a special set V
- Violation measure sums the deviation from l or u for each subsequence
- Domain consistency filtering using minimum-cost network flow

Soft slide constraint [Bessiere, Hebrard, Hnich,Kiziltan, Quimper, Walsh, 2007]

- Slide is extension of sequence, special case of 'cardinality-path' constraint
- Edit-based and variable-based violation measures
- Reformulated as hard slide constraints using additional variables

Still more soft global constraints

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Soft context-free grammar constraint [Katsirelos, Narodytska, Walsh, 2008]

- Extension of regular constraint [Sellmann 2006], [Quimper & Walsh, 2006]
- Special case of weighted context free grammar constraint
- Variable-based (Hamming distance) and edit-based violation measures

Soft all-equal constraint [Hebrard, O'Sullivan, and Razgon, 2008]

- 'Inverse' of decomposition-based soft alldifferent
- Extended work presented this CP [Hebrard, Marx, O'Sullivan, Razgon, 2009]

Soft open global constraints

- Open constraints have a scope that will be defined the during search for a solution
- Presented this CP [Maher, 2009]

Soft global constraint for weighted CSP

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Sigma-alldifferent, Sigma-Gcc, Sigma-regular [Métivier, Boizumault, Loudni, 2007, 2009]

- Model preferences among variables and constraints
- Weight associated to each variable, and each constraint (for example, to each not-equal constraint in the alldifferent)
- Domain consistency algorithm for variable-based sigma-alldifferent, using network flow
- For decomposition-based sigma-alldifferent, domain consistency is NP-hard; filtering based on relaxations is proposed
- Filtering for decomposition-based sigma-gcc and a distance-based sigma-regular, similar to the network flow approach of [v.H. et al., 2006]

Weighted CSPs [Lee and Leung, 2009]

- Costs are associated to tuples of the constraint
- Apply flow-based filtering algorithms of [v.H. et al., 2006] in this context

Constraint-Based Local Search

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Aim: Model the problem using variables and constraints (as in CP), and apply an automatically-derived Local Search method to solve the model

[Van Hentenryck and Michel, 2005], [Galinier and Hao, 2000,2004], Bohlin [2004, 2005]

Essential to CBLS is that the solution method can be derived from the constraints

- Local Search evaluates current assignment and then moves to an (improving) assignment in its neighborhood
- Neighborhoods as well as evaluation functions can be based on combinatorial properties of the constraints
- Global constraints can be particularly useful for this purpose

[Nareyek, 2001]

Soft global constraints for CBLS [Van Hentenryck and Michel 2005]

- Instead of domain filtering, the task is to measure the additional amount of violation (gradient) if we were to assign a variable to a certain value
- Violation measures are given for alldifferent, atmost, atleast, multi-knapsack, sequence, systems of not-equal constraints, and weighted constraint systems

Conclusions and perspectives

Conclusions

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Since the introduction of the first soft global constraint in 1998, and the cost-based framework for soft global constraints, many soft global constraints have been introduced
- For many of these constraints, domain filtering is as efficient for the soft version as for its hard counterpart
- It has become an 'established' approach
  - for many global constraints, a soft version is presented at the same time as the hard version (e.g., sequence, slide, weighted-grammar, etc.)
  - soft global constraints are being implemented and designed specifically for applications
  - it influences other approaches (weighted CSPs, Local Search)

Perspectives

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Is the topic closed?

- No! It is as essential to CP as global constraints are in general
- Number of new soft global constraints and new applications grows rapidly

Many (most) industrial problems are essentially over-constrained

- Soft global constraints allow to model and solve these problems using CP
- Huge potential

Task for (industrial) CP solvers to adopt soft global constraints?

- Present in Comet 2.0, but what about IBM ILOG, Eclipse, Gecode?