

# Global Constraints

Nicolas Beldiceanu

## SICS

Lägerhyddsvägen 18  
75237 Uppsala  
nicolas@sics.se

## Goal of the Tutorial

- Overview of **different aspects** of global constraints,
- Emphasis on **common pitfalls**.

This is **subjective**, but I was asked to give **my** point of view !

# Outline of the Presentation

## A SHORT OVERVIEW

## A BRIEF HISTORY

## DECLARATIVE ASPECTS

- **Classification of Global Constraints**
- **Common Pitfalls when Creating Global Constraints**

## PROCEDURAL ASPECTS OF FILTERING ALGORITHMS

- **Common Pitfalls when Creating Filtering Algorithms**
- **Incremental Algorithms**
- **Geometrical Filtering Algorithms**
- **Hypothetical Propagation**

## CONCLUSION

## REFERENCES



## **A SHORT OVERVIEW**

### **A BRIEF HISTORY**

### **DECLARATIVE ASPECTS**

- **Classification of Global Constraints**
- **Common Pitfalls when Creating Global Constraints**

### **PROCEDURAL ASPECTS OF FILTERING ALGORITHMS**

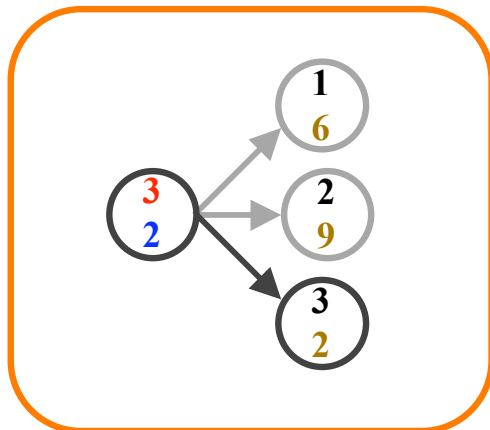
- **Common Pitfalls when Creating Filtering Algorithms**
- **Incremental Algorithms**
- **Geometrical Filtering Algorithms**
- **Hypothetical Propagation**

### **CONCLUSION**

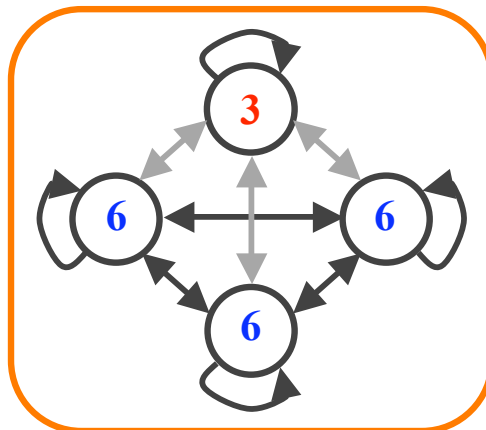
### **REFERENCES**

## A Short Overview of the Area: Declarative Aspect

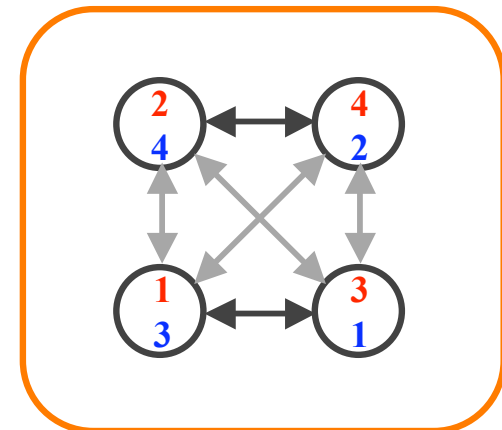
**GOAL:** Identify common structures in discrete combinatorial problems.



`element(3,{6,9,2},2)`



`nvalue(2,{6,3,6,6})`



`symetric_alldiff({3,4,1,2})`

## Declarative Aspect: Key Ideas

- A common structure is **not an entire problem** !  
⇒ first difference from catalog of problems.  
*(allows to come up with **components** which can be **reused** across different problems)*
- There should be an **explicit description** of these structures.  
⇒ second difference from catalog of problems.  
*(since different programs will exploit these structures describing things with **natural language is useless**)*

## Declarative Aspect: Describe your Problem

A problem should be **described** as  
a combination of common structures  
**regardless** of the solving paradigm (CP,LP,LS)

### ***n*-queens:**

- ***n*** variables,
- **three** *alldifferent* constraints.

### **Warehouse location:**

- ***c+3*** variables,
- **one** *gcc* constraint,
- **one** *swdv* constraint.

### **Traveling salesman:**

- ***c+1*** variables,
- **one** *circuit* constraint,
- **one** *mwa* constraint.

## A Short Overview of the Area: Procedural Aspect

Produce **efficient** algorithms within **various** contexts:

- **filtering** algorithms for constraint solving,
- generation of **explanations**,
- **visualization** algorithms for constraint debugging,
- generation of **cuts** for linear programming,
- incremental **moves** for local search.

## Procedural Aspect: Key Idea

- Don't reformulate the problem in a **unique formalism** (*k-sat, linear programming, ...*) with a general purpose solver,

### BUT RATHER

- Describe your problem in terms of **common structures** for which you have the best known algorithms.

Within a unique formalism you will **try to catch** some problem structure soon or later !

## A SHORT OVERVIEW



## A BRIEF HISTORY

### DECLARATIVE ASPECTS

- Classification of Global Constraints
- Common Pitfalls when Creating Global Constraints

### PROCEDURAL ASPECTS OF FILTERING ALGORITHMS

- Common Pitfalls when Creating Filtering Algorithms
- Incremental Algorithms
- Geometrical Filtering Algorithms
- Hypothetical Propagation

### CONCLUSION

### REFERENCES

## A Brief History: Key Phases

<b>1976</b> (pioneer)	<ul style="list-style-type: none"> <li><b>ALICE</b></li> </ul>	<i>alldifferent, circuit, tree</i>
<b>1987-1990</b> (research-lab)	<ul style="list-style-type: none"> <li>CHIP</li> </ul>	<i>element, atleast, atmost, between, lex_ordering, diff_pair, ...</i>
<b>1990-1992</b> (industry)	<ul style="list-style-type: none"> <li>Bull, Cosytec, Ilog</li> </ul>	<i>cumulative, cycle, diffn, distribute, ...</i>
<b>1993-2002</b> (everybody)	<ul style="list-style-type: none"> <li>BProlog, ECLAIR, ECLiPSe, FaCiLe, Figaro, ICEBERG, IF/Prolog, Mozart, SICStus, ... .</li> </ul>	<i>increasing, global cardinality, sequence, cumulatives, ...</i>

## A Breaf History: Key Points

1988	• Non-numerical constraint	<i>element: warehouse location</i>
1990	• Global constraint	<i>diff_pair: orthogonal latin squares</i>
1993	• Encapsulating OR	<i>scheduling, routing, metaheuristics</i>
2000	• Classification	<i>explicit description of global constraints</i>
2001	• Combinatorial structures for local search	<i>incremental moves for global constraints</i>

A SHORT OVERVIEW

A BRIEF HISTORY

DECLARATIVE ASPECTS



- **Classification of Global Constraints**
- Common Pitfalls when Creating Global Constraints

PROCEDURAL ASPECTS OF FILTERING ALGORITHMS

- Common Pitfalls when Creating Filtering Algorithms
- Incremental Algorithms
- Geometrical Filtering Algorithms
- Hypothetical Propagation

CONCLUSION

REFERENCES

## Motivations for a Classification of Global Constraints

- Find out the basic **constituents** of the global constraint,
- Classify the **properties** of each basic constituent,
- Understand how properties **interact**.

## Main Idea of the Classification

Global Constraints as:

**Graph Properties**

on **Structured Network**

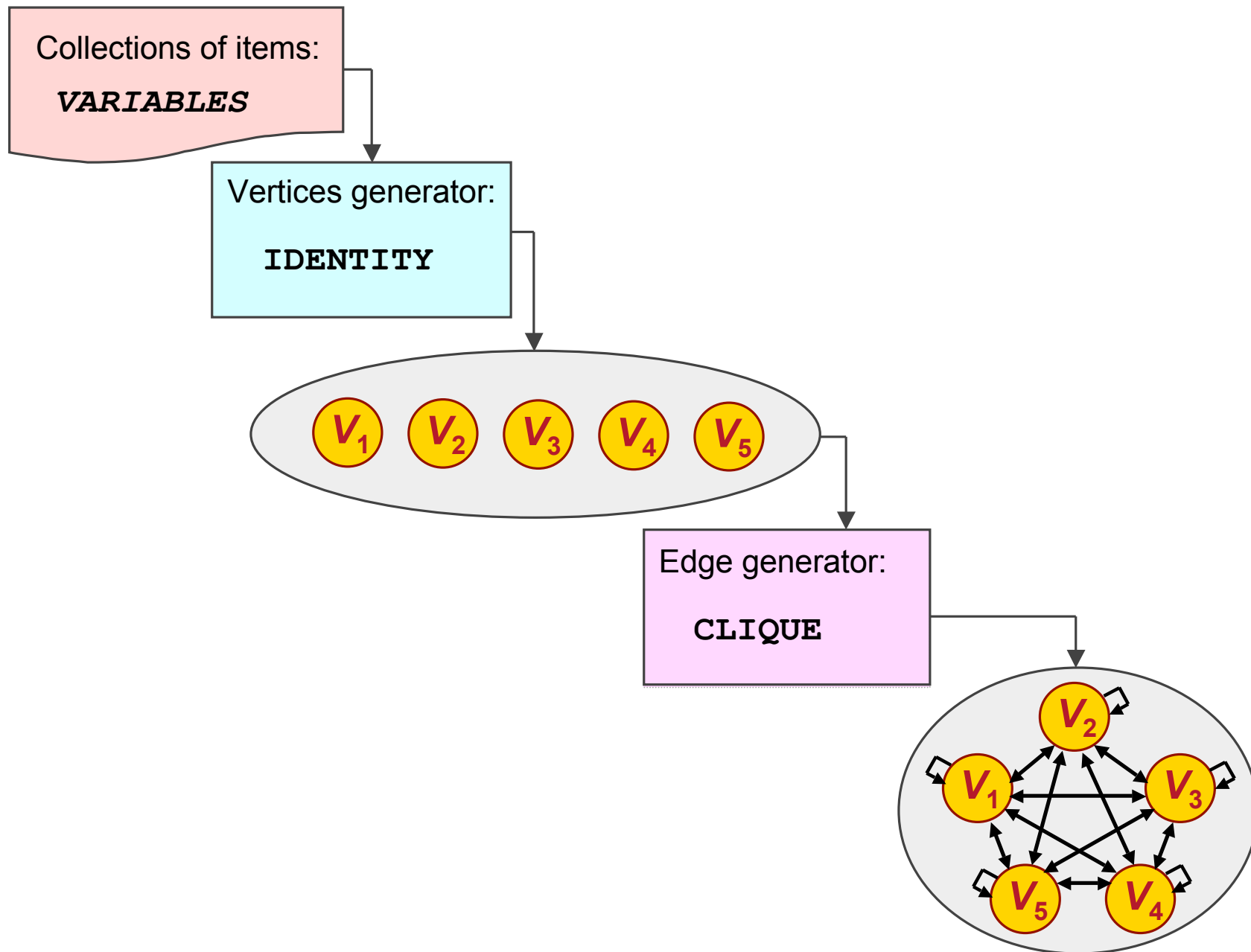
of **Elementary Constraints**

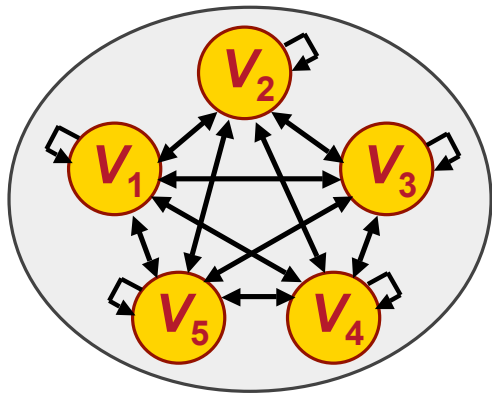
of the **Same Type**

## nvalue(NVAL, VARIABLES )

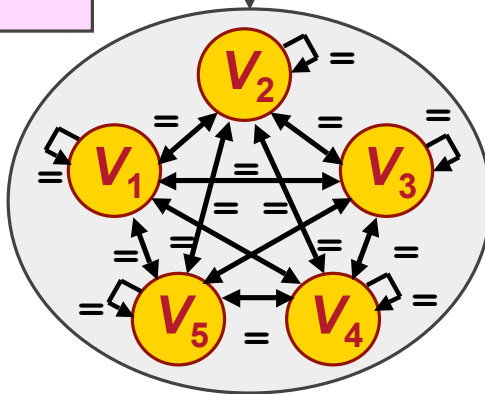
• ARGUMENT	:	NVAL	:	dvar	
					VARIABLES: collection(var-dvar)
• RESTRICTION(S)	:	NVAL ≥ 0			
					NVAL ≤  VARIABLES
					required(VARIABLES.var)
• VERTEX INPUT	:	VARIABLES			
• VERTEX GENERATOR	:	IDENTITY			
• EDGE INPUT	:	VARIABLES			
• EDGE GENERATOR	:	CLIQUE			
• EDGE ARITY	:	2			
• EDGE CONSTRAINT	:	VARIABLES.var[1] = VARIABLES.var[2]			
• GRAPH PROPERTY	:	NSCC = NVAL			

nvalue(4, { var-3, var-1, var-7, var-1, var-6 })

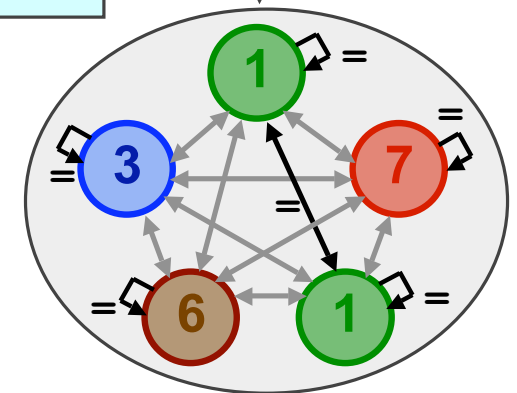




Edge constraint:  
=

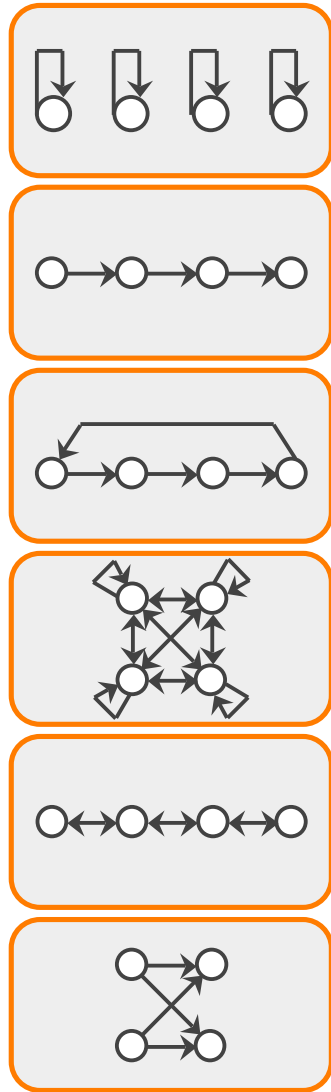


Graph property:  
 $NSCC=NVAL$

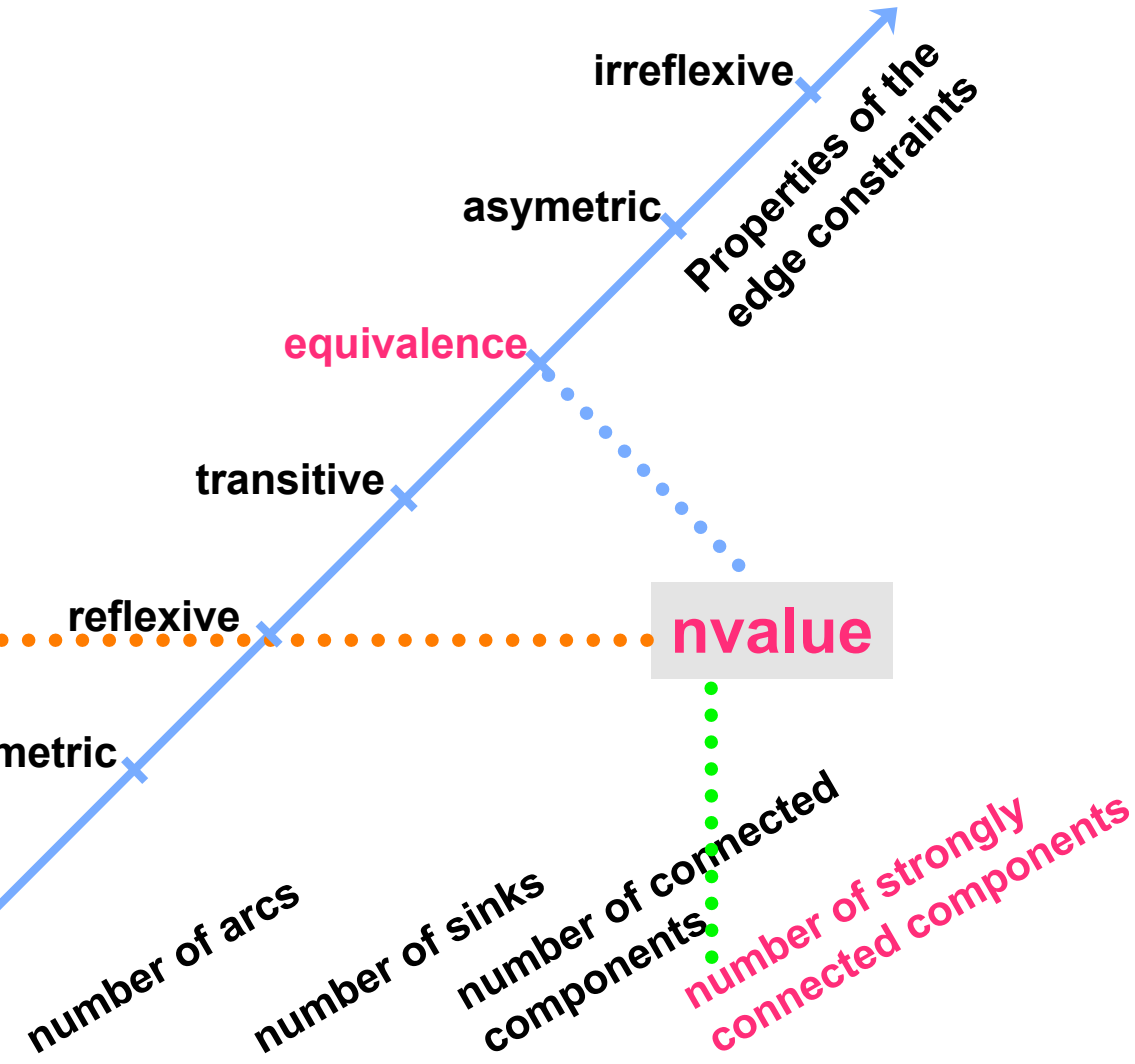


$nvalue(4, \{ var-3, var-1, var-7, var-1, var-6 \})$   
CP-2002, Cornell, USA

# Structure of the initial graph



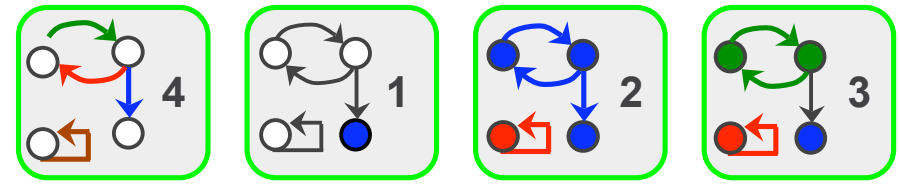
loop  
path  
circuit  
clique  
chain  
product



nvalue

number of arcs  
number of sinks  
number of connected components  
number of strongly connected components

## Global Constraint "Space"



Graph constraints

# A Catalog of Global Constraints

Alldifferent	Change_partition	Cycle_resource	Golomb	Minimum_pair
Alldifferent_except_0	Circuit	Cyclic_change	Graph_crossing	Nclass
Alldifferent_interval	Circuit_cluster	Cyclic_change_joker	Group	Nequivalence
Alldifferent_modulo	Circular_change	Cyclic_cumulative	Group_skip_isolated_item	Ninterval
Alldifferent_partition	Coloured_cumulative	Derangement	Inflexion	Notallegal
Alldifferent_same_value	Coloured_cumulatives	Diffn	Interval_and_count	Npair
Among	Common	Diff_2	Interval_and_sum	Number_of_rest
Among_interval	Common_interval	Diff_2_cyclic	Inverse	Nvalue
Among_modulo	Common_modulo	Diff_2_min_dist	Longest_change	Orchad
Among_seq	Common_partition	Disjoint	Map	Place_in_pyramid
Assign_and_count	Connect_points	Disjoint_tasks	Max_index	Polyomino
Assign_and_nvalue	Connected	Distance_change	Max_n	Relaxed_sliding_sum
Balance	Count	Distance_less	Max_nvalue	Same
Balance_modulo	Crossing	Distribute	Maximum	Same_interval
Balance_partition	Cumulative	Domain_constraint	Maximum_modulo	Same_modulo
Bin_packing	Cumulative_2d	Element	Maximum_pair	Same_partition
Binary_tree	Cumulative_product	Element_greatereq	Min_index	Sliding_card_skip0
Cardinality_atleast	Cumulatives	Element_lesseq	Min_n	Sliding_sum
Cardinality_atmost	Cycle	Element_sparse	Min_nvalue	Sliding_time_window
Change	Cycle_card_on_path	Elements	Minimum	Smooth
Change_continuity	Cycle_cover	Elements_alldifferent	Minimum_except_0	Soft_alldifferent_ctr
Change_pair	Cycle_or_accessibility	Global_cardinality	Minimum_modulo	Soft_alldifferent_var

# A Catalog of Global Constraints

Stretch  
Stretch\_circuit  
Stretch\_path  
Symmetric\_alldiff  
Temporal\_path  
Tree  
Tree\_resource  
Used\_by  
Used\_by\_interval  
Used\_by\_modulo  
Used\_by\_partition

- **Old constraints** which were not inside: element, ...
- **New constraints**: disjoint, ...
- Catalog **updated** as new constraints are presented:  
domain\_constraint,  
stretch,  
soft alldifferent.
- **Modifications** in the way of describing constraints:  
  
possibility to have constraint (like global cardinality, stretch) where  
different limits are associated to different values.  
  
avoid properties which can't be evaluated in polynomial time when all  
parameters of the constraint are fixed (e.g. maximum clique).

## A SHORT OVERVIEW

## A BRIEF HISTORY

## DECLARATIVE ASPECTS

- Classification of Global Constraints



- **Common Pitfalls when Creating Global Constraints**

## PROCEDURAL ASPECTS OF FILTERING ALGORITHMS

- Common Pitfalls when Creating Filtering Algorithms
- Incremental Algorithms
- Geometrical Filtering Algorithms
- Hypothetical Propagation

## CONCLUSION

## REFERENCES

## Declarative Pitfall

By using an **integer** as a parameter of a constraint you **restrict** artificially its use and modelling power.

`element(Index, [V1, V2, ..., Vn], Cost )`

`cumulative(Origin, Duration, Resource, Limit )`

`gcc(Assignment, Occurrences, Matrix, Cost )`

No specific **direction** for a parameter !

## Declarative Pitfall

A global constraint links usually different sets of variables / values;  
organize the constraint parameters in **collections** of objects.

### POOR DESIGN:

$\text{cumulative}(\{S_1, S_2, \dots, S_n\}, \{D_1, D_2, \dots, D_n\}, \{R_1, R_2, \dots, R_n\}, \text{Limit})$   
 $\text{stretch}(\{S_1, S_2, \dots, S_n\}, \tilde{\lambda}, \hat{\lambda}, \Pi, \gamma)$

### BETTER DESIGN:

$\text{cumulative}(\{\text{task}(S_1, D_1, R_1), \text{task}(S_2, D_2, R_2), \dots, \text{task}(S_n, D_n, R_n)\}, \text{Limit})$   
 $\text{stretch}(\{S_1, S_2, \dots, S_n\}, \{\text{value}(v_1, \tilde{\lambda}_1, \hat{\lambda}_1, \Pi_1), \text{value}(v_2, \tilde{\lambda}_2, \hat{\lambda}_2, \Pi_2), \dots, \text{value}(v_k, \tilde{\lambda}_k, \hat{\lambda}_k, \Pi_k)\}, \gamma)$

Put together what **belongs** together !

## Declarative Pitfall

**Artificial regroupment** of different class of constraints  
(*with the aim of doing better pruning*).

cycle/17	( <i>cycle + element</i> )
cumulative/8	( <i>cumulative + element</i> )
stretch/5	(span + permitted patterns)

Too **complicated** for standard users

## Declarative Pitfall

An **interface** to an **OR** algorithm.

An **efficient**, but **monolithic, adhoc mixture** of:

- handling basic constraints,
- constraint propagation,
- evaluating lower and upper bounds,
- domination criteria (*depends of the cost*),
- heuristics for selecting what to select next,
- metaheuristics,
- shaving.

Maximum Clique

Job-Shop

RCPSP

**Where** was the constraint ?  
**How** do I reuse this code ?

## A SHORT OVERVIEW

## A BRIEF HISTORY

## DECLARATIVE ASPECTS

- Classification of Global Constraints
- Common Pitfalls when Creating Global Constraints

## PROCEDURAL ASPECTS OF FILTERING ALGORITHMS



- **Common Pitfalls when Creating Filtering Algorithms**
- Incremental Algorithms
- Geometrical Filtering Algorithms
- Hypothetical Propagation

## CONCLUSION

## REFERENCES

# Complexity Pitfall

**What** do we measure ?

- the worst-case

- the average case

of

- an invocation to the filtering algorithm
- reaching a leaf without backtracking
- exploring a tree

Optimizing **one invocation** to the filtering algorithm is **not enough** !

## Complexity Pitfall

The complexity of a filtering algorithm is **not independent** of the complexity of the **services** of the constraint kernel.

for instance:

- access to a domain variable ( $\text{is\_in\_dom}(var, val)$ ,  $\text{next\_value}(var, val)$ )
- what kind of waking events are available?
- does the kernel assume saturation or not ?

Tarjan algorithm for strongly connected components (*circuit*)  
**keeps its complexity iff** can access to next element in  $O(1)$  !

## Complexity Pitfall

Avoid complexities depending on the **size of the domain**.

for instance:

- usual filtering algorithm for the *alldifferent* constraint.

$V_1, V_2, V_3$  in **1..100000**,  $\text{alldifferent}(\{V_1, V_2, V_3\})$

**The more values** you have, the **less pruning** you get !

## Complexity Pitfall

Be cautious if the complexity depends on the number of **removed values** for one invocation to the filtering algorithm.

for instance:

- complexity for one invocation to the *nvalue* constraint is  $O(n \cdot \log n + n \cdot p)$ , where  $n$  is the number of variables,  $p$  is the number of removed values.

Don't **rediscover** the same pruning **again and again** !  
(*within a path to a leaf or within a subtree* )

## Complexity Pitfall

Control the **amount of memory** used on a path to a leaf.

Backtrackable data structures with guarantee on memory use:

- should not trail the same address more than  $O(1)$ .

**Memory** is the biggest problem for scaling !

## Multi-Occurrences Pitfall

If the **same variable** is expected to **occur several times** in a constraint then you will have to revise your filtering algorithm.

Typical examples are:

- constraints involving *linear terms*,
- *alldifferent*,
- *non-overlapping rectangles* constraint.

The **worst** (even bugs) can happen if you ignore this issue !

## Pruning Pitfall

Since the **feasibility check** of a number of global constraints is **NP-hard**, adding deduction rules which cover special cases is a **never ending** task.

Typical examples are:

- *circuit*,
- *nvalue*,
- *cumulative*.

**Remember** that the worst-case complexity will be the worst complexity

among the algorithms associated to the different deduction rules.

## Pruning Pitfall

Add **cheap filters** which detect **early** that a particular filtering algorithm will not bring any domain reduction.

A typical example is:

- witness for the *non-overlapping rectangles* constraint.

**Most** of the invocations to a filtering algorithm are **useless** !

## Comparing two Incomplete Filtering Algorithms

Relate **worst-case** complexity to **amount** of pruning.

Does the **extra** effort **pay off** ?

## A SHORT OVERVIEW

## A BRIEF HISTORY

## DECLARATIVE ASPECTS

- Classification of Global Constraints
- Common Pitfalls when Creating Global Constraints

## PROCEDURAL ASPECTS OF FILTERING ALGORITHMS

- Common Pitfalls when Creating Filtering Algorithms
- **Incremental Algorithms**
- Geometrical Filtering Algorithms
- Hypothetical Propagation



## CONCLUSION

## REFERENCES

## Incremental Algorithms

Some filtering algorithms (*alldifferent*, *gcc*, *nvalue*) have **two phases**:

(1) Check feasibility,      (2) Prune.

Checking feasibility is usually done by **computing a particular solution**.

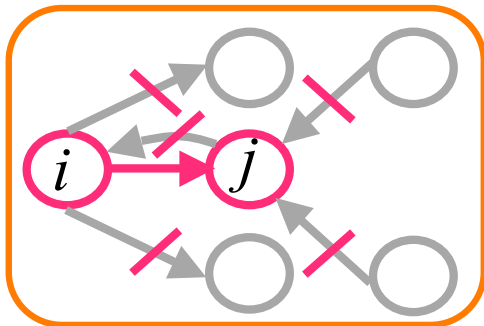
Restart from the **previous solution**,  
**removing** invalid assignments !

# Incremental Algorithms

Find the **right events** according to which you can be incremental.

For instance for the **circuit** constraint:

Not incremental according to **one single edge deletion** but according to a **family of edge deletions**.



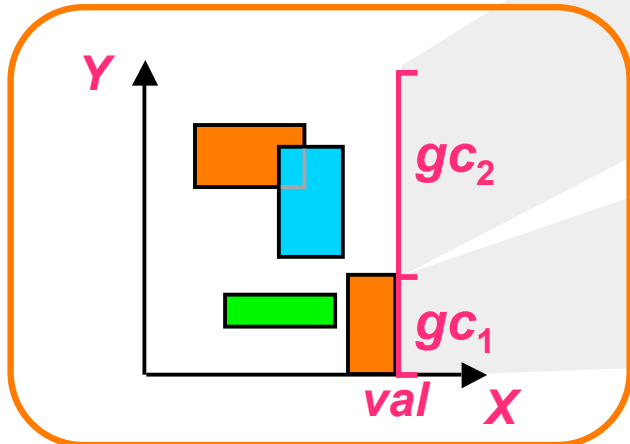
Removed values after  
an assignment:  $X_i = j$

**Reevaluating** from scratch may be better  
if **too many** changes !

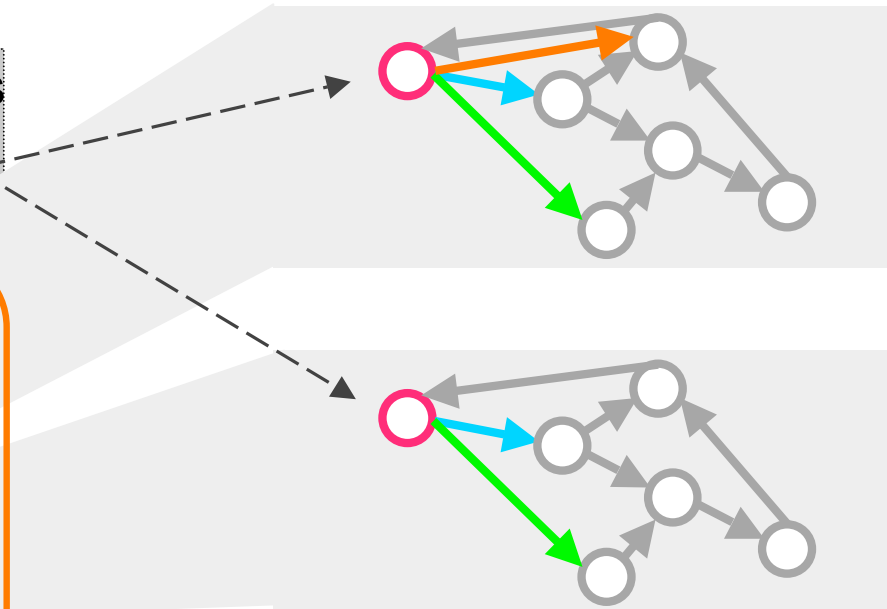
# Incremental Algorithms for Sweep

## PROPERTY

All removed or added arcs start from a given **vertex**.



$$\neg(l \leq gc_1 \leq u \vee l \leq gc_2 \leq u) \Rightarrow val \notin \text{dom}(X)$$



Evaluating a **graph characteristics** on a set of intervals  
**Event:** for an interval, an arc is removed or added  
**Check:** does there exist at least one interval  $i$  such that  $gc_i$  is located in some range?

# Incremental Algorithms for Cost Filtering: Context

$CTR_1(\{A_1, A_2, \dots, A_n\}, Cost)$

Lower bound for Cost:

$lb$

Regret matrix:

$A_i = val_j \Rightarrow Cost \geq lb + regret_{ij}$

	$val_1$	$val_2$	...	$val_m$
$A_1$				
$A_2$				
...				
$A_n$				

*regret<sub>ij</sub>*

$\wedge$

$CTR_2(\{A_1, A_2, \dots, A_n\})$   
(graph property)

Polynomial algorithm for checking a necessary condition:

$check\_CTR_2(\{A_1, A_2, \dots, A_n\})$ : (maybe, no)

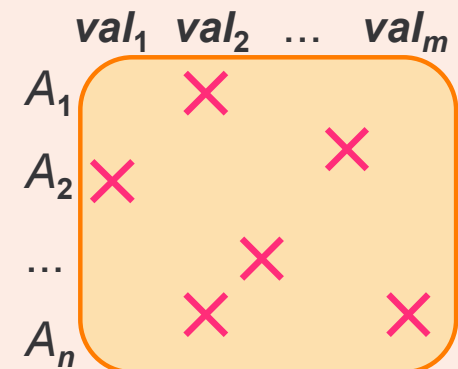
**How** to get a better evaluation of  $\min(Cost)$  according to  $CTR_2$  ?

## Incremental Algorithms for Cost Filtering: Method

Search smallest regret such that  $\text{check\_CTR}_2(\{A_1, A_2, \dots, A_n\})$  returns maybe.

$\text{regret}_{ij} \leq \text{treshold} \Rightarrow$

$A_i \neq \text{val}_j$  when  $\text{lb} + \text{regret}_{ij} \geq \max(\text{Cost})$



**Bisection search** of the smallest regret using  $\text{check\_CTR}_2(\{A_1, A_2, \dots, A_n\})$

**Avoid shaving!**

# Incremental Algorithms for Cost Filtering

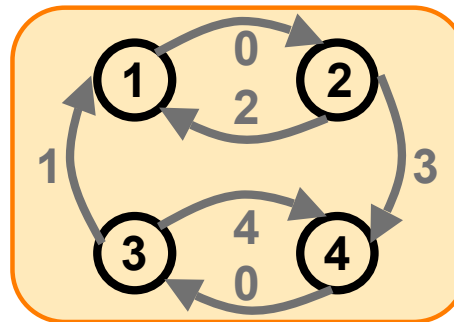
Assume  $\text{check\_CTR}_2(\{A_1, A_2, \dots, A_n\})$  checks **if no more than one strongly connected component** in the following graph:

$A_1 :: 2$   
 $A_2 :: 1, 3$   
 $A_3 :: 1, 4$   
 $A_4 :: 3$

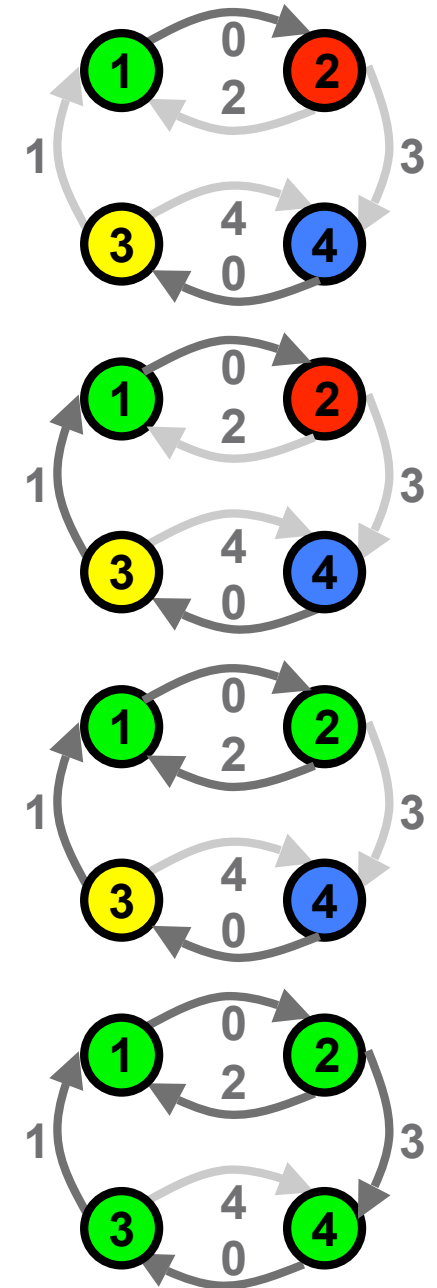
Assignment variables

	1	2	3	4
$A_1$	■	0	■	■
$A_2$	2	■	■	3
$A_3$	1	■	■	4
$A_4$	■	■	0	■

Regret matrix



Initial graph



Efficient algorithm ?

## A SHORT OVERVIEW

## A BRIEF HISTORY

## DECLARATIVE ASPECTS

- Classification of Global Constraints
- Common Pitfalls when Creating Global Constraints

## PROCEDURAL ASPECTS OF FILTERING ALGORITHMS

- Common Pitfalls when Creating Filtering Algorithms
- Incremental Algorithms
- **Geometrical Filtering Algorithms**
- Hypothetical Propagation



## CONCLUSION

## REFERENCES

# Variations around Sweep

**Pruning** for the following constraint patterns:

- A **conjunction** of constraints with **two shared** variables
- Several **conjunctions** of constraints with **two shared** variables
- A multi-resource **cumulatives** constraint
- The **non-overlapping** constraint between **polygons**
- The **cardinality** operator with **two shared** variables

## Forbidden and Safe Regions

**DEFINITION** *forbidden* region according to a constraint **Ctr** and 2 variables **X,Y** of Ctr :

**Two intervals**  $\text{inf}_x..sup_x$  and  $\text{inf}_y..sup_y$  such that:

$$\forall x \in \text{inf}_x..sup_x,$$

$\forall y \in \text{inf}_y..sup_y$ : **Ctr** with the assignment **X=x** and **Y=y** is **false**.

**DEFINITION** *safe* region according to a constraint **Ctr** and 2 variables **X,Y** of Ctr :

**Two intervals**  $\text{inf}_x..sup_x$  and  $\text{inf}_y..sup_y$  such that:

$$\forall x \in \text{inf}_x..sup_x,$$

$\forall y \in \text{inf}_y..sup_y$ : **Ctr** with the assignment **X=x** and **Y=y** is **true**.

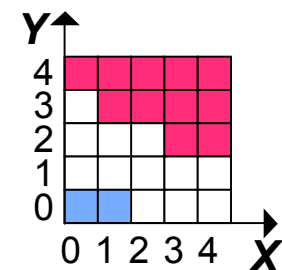
### EXAMPLE

$$0 \leq X \leq 4$$

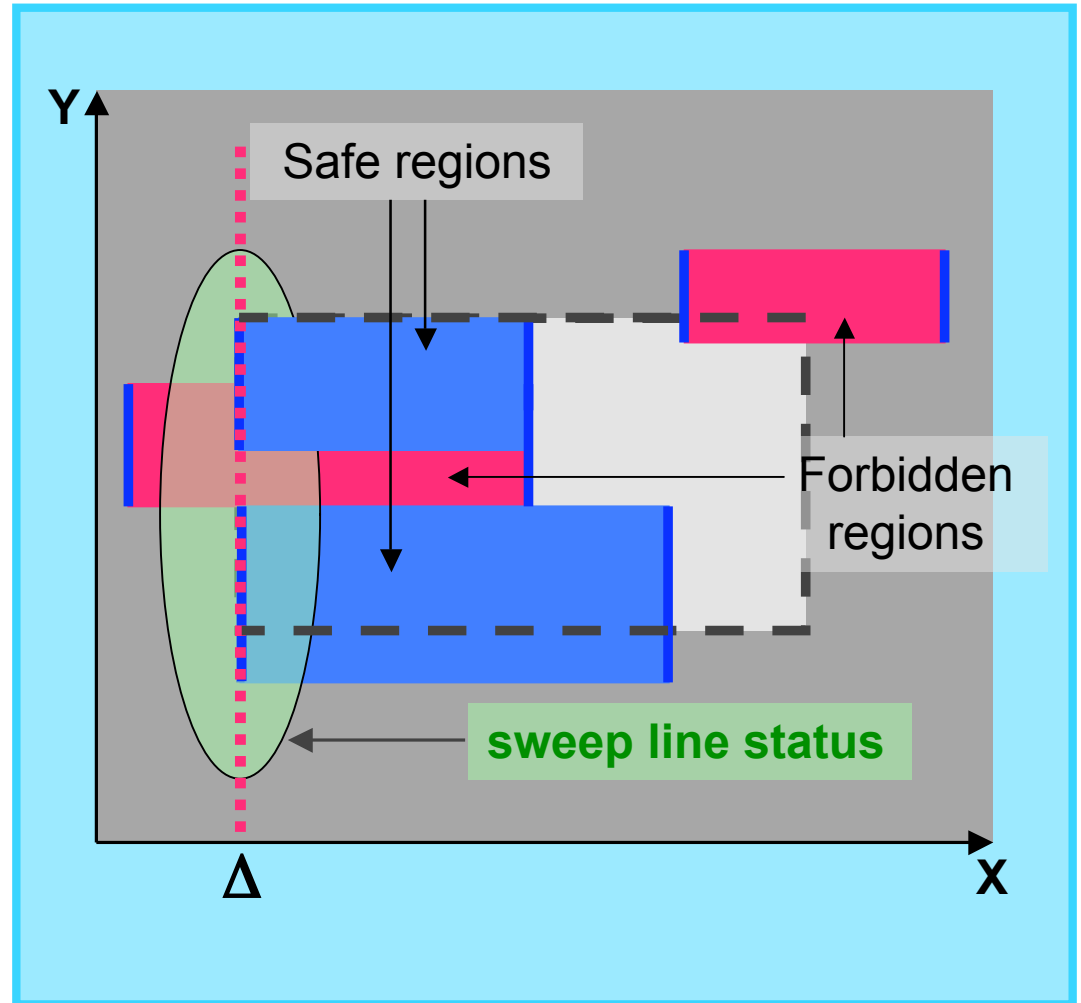
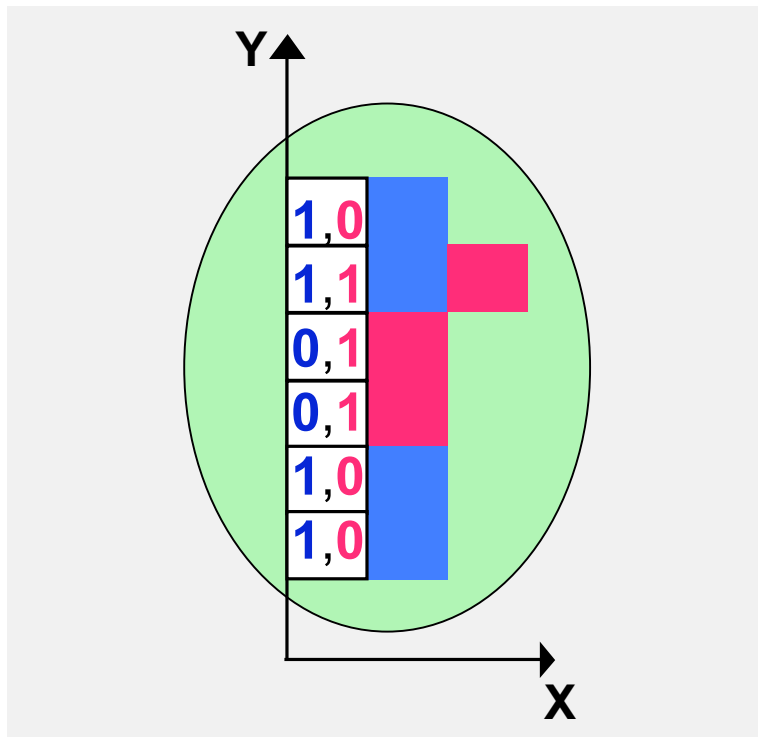
$$0 \leq Y \leq 4$$

$$1 \leq S \leq 6$$

$$X + 2Y \leq S$$



# Sweep-Line Status



For each  $y \in \text{dom}(Y)$ :

- number of **safe** regions
- number of **forbidden** regions containing the point  $(\Delta, y)$

Remove a value  $\Delta \in \text{dom}(X)$  if for all  $y \in \text{dom}(Y)$ :

$$n_{\text{safe}}[y]..n_{\text{ctr}} - n_{\text{forbidden}}[y] \cap C = \emptyset$$

# An Example

PROBLEM:

Adjust minimum of **X** according to **Y** and to the fact that **4 or 5 constraints** should **hold**:

$$0 \leq X \leq 4 \quad 0 \leq Y \leq 4 \quad 2 \leq Z \leq 3$$

$$1 \leq S \leq 6 \quad 0 \leq T \leq 0 \quad 1 \leq U \leq 2$$

alldifferent({**X**,**Y**,**R**})

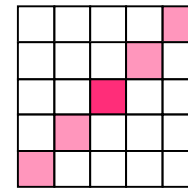
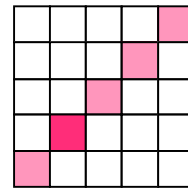
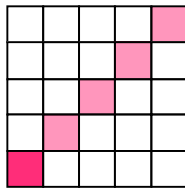
$$|X - Y| > Z$$

$$X + 2Y \leq S$$

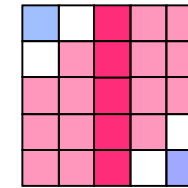
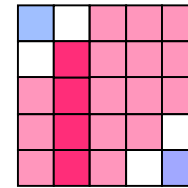
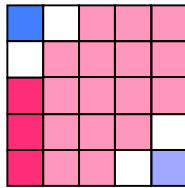
$$X + 3 \leq T \vee T + 1 \leq X \vee$$

$$Y + 3 \leq U \vee U + 4 \leq Y$$

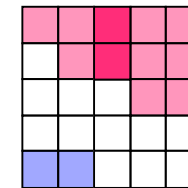
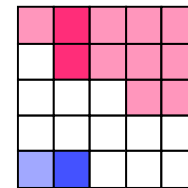
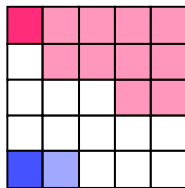
$$X + Y \equiv 0 \pmod{2}$$



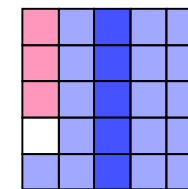
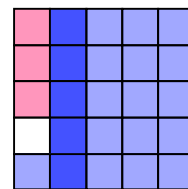
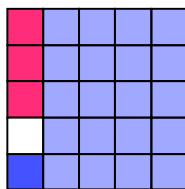
alldifferent({**X**,**Y**,**R**})



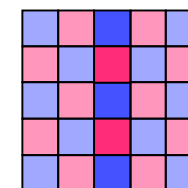
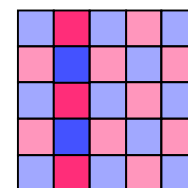
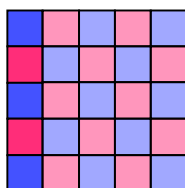
$|X - Y| > Z$



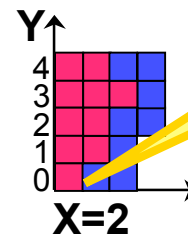
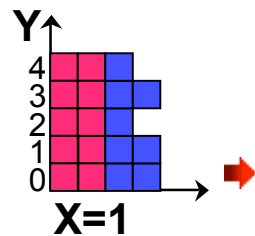
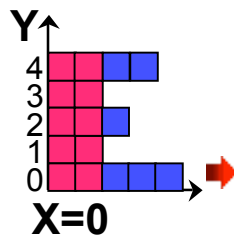
$X + 2Y \leq S$



$X + 1 \leq T \vee T + 1 \leq X \vee$   
 $Y + 1 \leq U \vee U + 4 \leq Y$



$X + Y \equiv 0 \pmod{2}$



**Deduction:**  
**X > 1**

## A SHORT OVERVIEW

## A BRIEF HISTORY

## DECLARATIVE ASPECTS

- Classification of Global Constraints
- Common Pitfalls when Creating Global Constraints

## PROCEDURAL ASPECTS OF FILTERING ALGORITHMS

- Common Pitfalls when Creating Filtering Algorithms
- Incremental Algorithms
- Geometrical Filtering Algorithms



- **Hypothetical Propagation**

## CONCLUSION

## REFERENCES

# Hypothetical Propagation

A filtering algorithm which:

- makes **assumptions** and posts constraints,
- **observes** the domain reductions and **undoes** propagation,
- **deduces** something.

Typical examples are:

- constructive disjunction,
- constructive cardinality,
- cardinality-path,
- soft constraints, probing.

Generic and useful but **difficult to characterize** the complexity and the amount of pruning !

## A SHORT OVERVIEW

## A BRIEF HISTORY

## DECLARATIVE ASPECTS

- Classification of Global Constraints
- Common Pitfalls when Creating Global Constraints

## PROCEDURAL ASPECTS OF FILTERING ALGORITHMS

- Common Pitfalls when Creating Filtering Algorithms
- Incremental Algorithms
- Geometrical Filtering Algorithms
- Hypothetical Propagation



## CONCLUSION

## REFERENCES

# Conclusion

- **Short Term**

**Explicit** description of constraints

**Efficient** and **generic** filtering algorithms

- **Long Term**

**Synthesize**

{ checkers  
filtering algorithms  
incremental moves  
visualization  
specialized heuristics }

from the description of constraints

**Understand** global constraints in terms of the **properties** of their basic **constituents**

## A SHORT OVERVIEW

## A BRIEF HISTORY

## DECLARATIVE ASPECTS

- Classification of Global Constraints
- Common Pitfalls when Creating Global Constraints

## PROCEDURAL ASPECTS OF FILTERING ALGORITHMS

- Common Pitfalls when Creating Filtering Algorithms
- Incremental Algorithms
- Geometrical Filtering Algorithms
- Hypothetical Propagation

## CONCLUSION



## REFERENCES

## References

Next week, you can get this presentation from:

<http://www.sics.se/isl/cps/>

At the end there is  
a list of references.

## **CLASSIFICATION AND CATALOG OF GLOBAL CONSTRAINTS**

### **Classification of problems that can be tackle by constraint programming:**

H. Simonis. A Problem Classification Scheme for Finite Domain Constraint Solving. In *Proc. Workshop on constraint applications, CP96*, Boston, (August 1996).

### **Catalog of global constraints:**

N. Beldiceanu. Global Constraints as Graph Properties on Structured Network of Elementary Constraints of the Same Type. SICS Technical Report T2000/01, (2000).

N. Beldiceanu. Global Constraints as Graph Properties on a Structured Network of Elementary Constraints of the Same Type. *Principles and Practice of Constraint Programming - CP 2000, 6th International Conference*, Singapore, September 18-21, 2000, Proceedings. Lecture Notes in Computer Science, (1894), 52--66, Springer , 2000.

## SYSTEMS CONTAINING GLOBAL CONSTRAINTS

### ALICE:

J-L. Laurière. *Un langage et un programme pour énoncer et résoudre des problèmes combinatoires*. Thèse de Doctorat d'État de l'Université Paris 6, (May 1976). In French.

J-L. Laurière. A Language and a Program for Stating and Solving Combinatorial Problems. *Artificial Intelligence* 10, 29-127, (1978).

The chapter 12 of the following book mentions a declarative reimplement of ALICE:

J. Pitrat. *Penser autrement l'informatique*. Ed. Hermès, Paris, (1993). In French.

### CHIP:

M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf and F. Berthier. The constraint logic programming language CHIP, In *Proc. Int. Conf. On Fifth Generation Computer Systems FGCS-88*, Tokyo, 693-702, (1988).

<http://www.cosytec.com/>

### Ilog Solver:

<http://www.ilog.com/>

### Mozart:

S. Haridi, P. Van Roy, P. Brand, and C. Schulte. Programming Languages for Distributed Applications. *New Generation Computing*, n3 v16, (1998), Omsa, Ltd. and Springer-Verlag.

G. Smolka. Constraints in Oz. In *ACM Computing Surveys*, Vol 28, No 4, (December 1996).

<http://www.mozart-oz.org/>

### ECLAIR:

<http://www.lcr.thomson-csf.fr/projects/ECLAIR/eclair1.6.html>

### ECLiPSe:

<http://www.icparc.doc.ic.ac.uk/eclipse/>

### FaCile (constraints programming library in OCaml):

<http://www.recherche.enac.fr/opti/facile/>

### ICEBERG(a library of global constraints):

<http://www.choco-constraints.net/download/ICEBERGuserguide0-5.pdf>

### IF/PROLOG:

[http://www.ifcomputer.com/IFProlog/Constraints/home\\_en.html](http://www.ifcomputer.com/IFProlog/Constraints/home_en.html)

### SICStus:

M. Carlsson, G. Ottosson, B. Carlsson. An Open-Ended Finite Domain Constraint Solver. *Proc. Programming Languages: Implementations, Logics, and Programs*, (1997).

<http://www.sics.se/isl/sicstus/>

## GLOBAL CONSTRAINTS

Element constraint.

P. Van Hentenryck and J-P. Carillon. Generality versus Specificity: an Experience with AI and OR Techniques. In *American Association for Artificial Intelligence (AAAI-88)*, St. Paul, Mi, (August 1988).

Alldifferent on pairs of variables:

N. Beldiceanu. An Example of Introduction of Global Constraints in CHIP: Application to Block Theory Problems. ECRC Technical Report TR-LP-49, (15 May 1990).

A constraint which count how many constraints holds.

P. Van Hentenryck and Y. Deville. The Cardinality Operator: A New Logical Connective for Constraint Logic Programming. *ICLP 1991*, 745-759, (1991).

Cumulative (a resource constraint):

A. Aggoun and N. Beldiceanu. Extending CHIP in order to solve complex scheduling and placement problems, *Mathl. Comput. Modelling* 17 (7), 57-73 (1993).

Among, diffn and cycle (a counting, a placement and a routing constraint):

N. Beldiceanu and E. Contejean. Introducing global constraint in CHIP. *Mathl. Comput. Modelling* Vol. 20, No. 12, 97-123 (1994).

Alldifferent (the first complete filtering algorithm for the alldifferent constraint):

J-C. Régim. A filtering algorithm for constraints of difference in CSP. In *Proc. of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, 362-367, (1994).

Filtering algorithm for the cumulative constraint:

Y. Caseau and F. Laburthe. Cumulative Scheduling with Task Intervals, Proceedings of the Joint International Conference and Symposium on Logic Programming, MIT Press, (1996).

Global cardinality constraint (a counting constraint):

J-C. Régim. Generalized Arc Consistency for Global Cardinality Constraint. In *Proc. of the Fourteenth National Conference on Artificial Intelligence (AAAI-96)*, (1996).

Sorting constraint:

N. Bleuzen Guernalec and A. Colmerauer. Narrowing a  $2n$ -Block of Sortings in  $O(n \log n)$ . In *Principles and Practice of Constraint Programming - CP97, Third International Conference*, Linz, Austria, (October 29 - November 1, 1997), Proceedings. Lecture Notes in Computer Science, 2-16, Vol. 1330, Springer, (1997).

Sorting constraint:

J. Zhou. A permutation-based approach for solving the job-shop problem. *Constraints*, 2(2), 185-213, (1997).

Covering the edges with cycles:

G. Pesant and P. Soriano. An Optimal Strategy for the Constrained Cycle Cover Problem. CRT Pub. no 98-65, (14 pages), (December 1998).

Symmetric alldifferent constraint (a permutation where all cycles have a length of two):

J-C. Régin. The symmetric alldiff constraint. In *Proceedings IJCAI'99*, Stockholm, Sweden, 420-425, (1999).

Cycle (routing constraint with a lot of variants):

E. Bourreau. *Traitement de contraintes sur les graphes en programmation par contraintes*. PhD thesis of University Paris 13, (March 30, 1999). In French.

K.Mehlhorn, S. Thiel. Faster Algorithms for Bound-Consistency of the Sortedness and the Alldifferent Constraint.

In *Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming CP'2000*, LNCS 1894, Springer, 2000.

Non-overlapping rectangles constraint:

N. Beldiceanu, M. Carlsson. Sweep as a Generic Pruning Technique Applied to the Non-Overlapping Rectangles Constraint. *Principles and Practice of Constraint Programming – CP 2001, 7<sup>th</sup> International Conference*, Cyprus, Nov 26-Dec 1, Springer, (2001).

Generalized non-overlapping constraint:

N. Beldiceanu, Q. Guo, S. Thiel. Non-overlapping Constraint between Convex Polytopes. *Principles and Practice of Constraint Programming – CP 2001, 7<sup>th</sup> International Conference*, Cyprus, Nov 26-Dec 1, Springer, (2001).

Timetabling constraint:

G. Pesant. A Filtering Algorithm for the Strech Constraint. In *Principles and Practice of Constraint Programming - CP'2001, 7th International Conference*, Paphos, Cyprus, (November 26 – December 1, 2001). Lecture Notes in Computer Science, Vol. 2239, Springer, 183-195, (2001).

The minimum and number of distinct values (yet another counting constraint) constraints.

N. Beldiceanu. Pruning for the minimum Constraint Family and for the number of distinct values Constraint Family. In *Principles and Practice of Constraint Programming - CP'2001, 7th International Conference*, Paphos, Cyprus, (November 26 – December 1, 2001). Proceedings. Lecture Notes in Computer Science, Vol. 2239, Springer, 211-224, (2001).

M. Marte. *A Global Constraint for Parallelizing the Execution of Task Sets in Non-Preemptive Scheduling*. In Electronic Proceedings of the CP-2001 Doctoral Programme, (2001).

Resource scheduling problem:

N. Beldiceanu and M. Carlsson A New Multi-Resource cumulatives Constraint with Negative Heights. In *Principles and Practice of Constraint Programming - CP'2002, 8th International Conference*, Cornell University, Ithaca, NY, USA, (September 8-13, 2002).

Ordering constraint between two tuples of domain variables:

A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel and T. Walsh. Global constraints for lexicographic orderings. In *Principles and Practice of Constraint Programming - CP'2002, 8th International Conference*, Cornell University, Ithaca, NY, USA, (September 8-13, 2002).

## GLOBAL CONSTRAINTS WITH COST

- Y. Caseau and F. Laburthe. Solving Various Weighted Matching Problems with Constraints. In *Principles and Practice of Constraint Programming - CP'97*, 3rd International Conference, Schloss Hagenberg, Austria, (October 29 - November 1, 1997), Proceedings. Lecture Notes in Computer Science, Vol. 1330, Springer, 17-31, (1997).
- Y. Caseau and F. Laburthe. Solving Small TSPs with Constraints. In *Fourteenth International Conference on Logic Programming - ICLP'97*, Leuven, Belgium, (July 8-11, 1997), Proceedings. MIT Press, 316-330, (1997).
- F. Focacci, A. Lodi and M. Milano. Cost-Based Domain Filtering. In *Principles and Practice of Constraint Programming - CP'99*, 5th International Conference, Alexandria, Virginia, USA, (October 11-14, 1999), Proceedings. Lecture Notes in Computer Science, Vol. 1713, Springer, 189-203, (1999).
- J.-C. Régin. Arc Consistency for Global Cardinality Constraints with Costs. In *Principles and Practice of Constraint Programming - CP'99*, 5th International Conference, Alexandria, Virginia, USA, (October 11-14, 1999), Proceedings. Lecture Notes in Computer Science, Vol. 1713, Springer, 390-404, (1999).
- T. Fahle. Cost Based Filtering vs. Upper Bounds for Maximum Clique. In *Fourth International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimisation Problems (CP-AI-OR'02)*, 93-107, Le Croisic, France (March 25-27, 2002).
- M. Milano and W.J. van Hoes. Reduced cost-based ranking for generating promising subproblems. In *Principles and Practice of Constraint Programming - CP'2002*, 8th International Conference, Cornell University, Ithaca, NY, USA, (September 8-13, 2002).
- M. Sellmann. An Arc Consistency Algorithm for the Minimum Weight All Different Constraint. In *Principles and Practice of Constraint Programming - CP'2002*, 8th International Conference, Cornell University, Ithaca, NY, USA, (September 8-13, 2002).
- N. Beldiceanu, M. Carlsson, S. Thiel. Cost-Filtering Algorithms for the two Sides of the Sum of Weights of Distinct Values Constraints. SICS Technical Report T2002/14, (2002).

## GLOBAL CONSTRAINTS AND LINEAR PROGRAMMING

G. Ottosson, E. Thorsteinsson and J. N. Hooker, Mixed global constraints and inference in hybrid IP-CLP solvers, CP99 Post-Conference Workshop on Large-Scale Combinatorial Optimization and Constraints, <http://www.dash.co.uk/wscp99> (1999) 57-78.

G. Ottosson, E. Thorsteinsson and J. N. Hooker. Mixed global constraints and inference in hybrid IP-CLP solvers. In *CP99 Post-Conference Workshop on Large-Scale Combinatorial Optimization and Constraints*, 57-78, (1999).

P. Refalo. Linear Formulation of Constraint Programming Models and Hybrid Solvers. In *Principles and Practice of Constraint Programming - CP'2000, 6th International Conference*, Singapoure, (September 18-22, 2000), Proceedings. Lecture Notes in Computer Science, Vol. 1894, Springer, (2000).

J. N. Hooker and Hong Yan, A relaxation for the cumulative constraint, October 2001, Revised May 2002.

Available at <http://ba.gsia.cmu.edu/jnh/papers.html>

T.H.Yunes: On the Sum Constraint: Relaxation and Applications. In *Principles and Practice of Constraint Programming - CP'2002*, 8th International Conference, Cornell University, Ithaca, NY, USA, (September 8-13, 2002).

## RELAXATION OF GLOBAL CONSTRAINTS

Relaxation within the domain of scheduling:

P. Baptiste, C. Le Pape, L. Peridy. Global Constraints for Partial CSPs: A Case-Study of Resource and Due Date Constraints. In *Principles and Practice of Constraint Programming - CP'98*, 4th International Conference, Pisa, Italy, (October 26-30, 1998), Proceedings. Lecture Notes in Computer Science, Vol. 1520, Springer, 87-101, (1998).

N. Beldiceanu and M. Carlsson. Revisiting the *cardinality* Operator and Introducing the *cardinality-path* Constraint Family. *Seventeenth International Conference on Logic Programming ICLP'01*, November 2001.

T. Petit, J-C. Régin and C. Bessière. Specific Filtering Algorithms for Over-Constrained Problems. In *Principles and Practice of Constraint Programming - CP'2001*, 7th International Conference, Paphos, Cyprus, (November 26 – December 1, 2001). Lecture Notes in Computer Science, Vol. 2239, Springer, 451-463, (2001).

T. Petit, J-C. Régin and C. Bessière. Range-Based Algorithm for Max-CSP. In *Principles and Practice of Constraint Programming - CP'2002*, 8th International Conference, Cornell University, Ithaca, NY, USA, (September 8-13, 2002).

## COMBINATION OF GLOBAL CONSTRAINTS

(Combines several cumulative constraints with a precedence graph)

N. Beldiceanu, E. Bourreau, D. Rivreau and H. Simonis. Solving Resource-constrained Project Scheduling Problems with CHIP. In *proc. of Fifth International Workshop on Project Management and Scheduling (PMS'96)*, Poznan, 35-38, (1996).

J-C. Régin and M. Rueher. A global constraint combining a sum constraint and binary inequalities. In *Sixteenth International Joint Conference on Artificial Intelligence (IJCAI 99)*, Stockholm, Workshop on Non Binary Constraints, (August 2 1999).