

# Graph Based Filtering

Nicolas Beldiceanu

**LINA, FRE CNRS 2729**

École des Mines de Nantes

Nicolas.Beldiceanu@emn.fr

## Constraint & Graphs/OR: Peoples (companies)

- **J.-C. Régim** (Ilog, France - *CPLEX & Solver*), **M. Minoux** (Paris 6)
- **Artelys** (France), **Dash Optimization** (England)
- **T. Walsh** (NICTA Australia), **C. G. Quimper** (Waterloo)
- **G. Pesant**, **M. Gendreau** (Polytechnique Montréal)
- **N. Beldiceanu** (Nantes), **I. Katriel** (Max-Plank, Saarbrücken)
- **P. Van Henteryck** (Brown), **I. Katriel** (Max-Plank, Saarbrücken)
- **P. Baptiste** (École Polytechnique & Thalès, France)

# Overview of the Presentation

- ➔ • **Getting into the Topic**
- Describing Global Constraints
- Graph Based Filtering
- Conclusion

## A First Example of Global Constraint: *circuit*

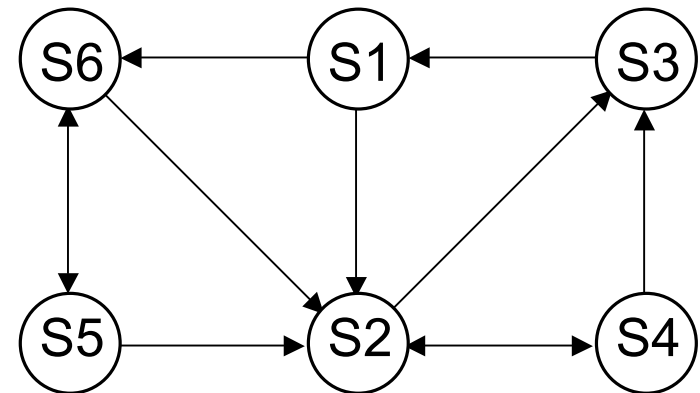
### DEFINITION

Cover a directed graph by a set of circuits in such a way that each vertex belongs to exactly one circuit.

Other interpretation: number of cycles of a permutation.

### CONSTRAINT (routing)

`circuit(Ncircuits, Successors)`



`circuit(Ncircuits, {S1,S2,S3,S4,S5,S6})`

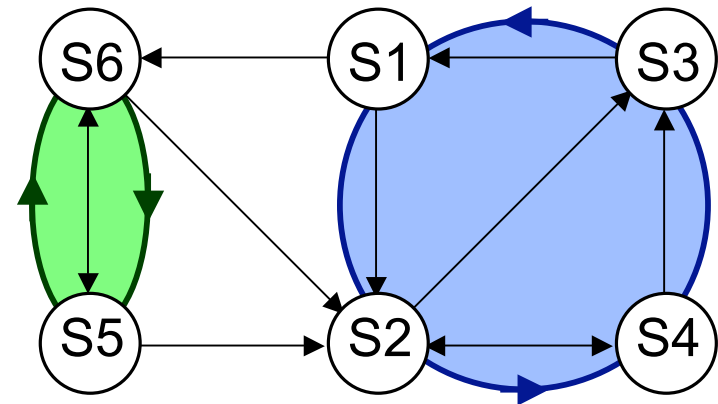
## Example of Global Constraint: *circuit*

### DEFINITION

Cover a directed graph by a set of circuits in such a way that each vertex belongs to exactly one circuit.

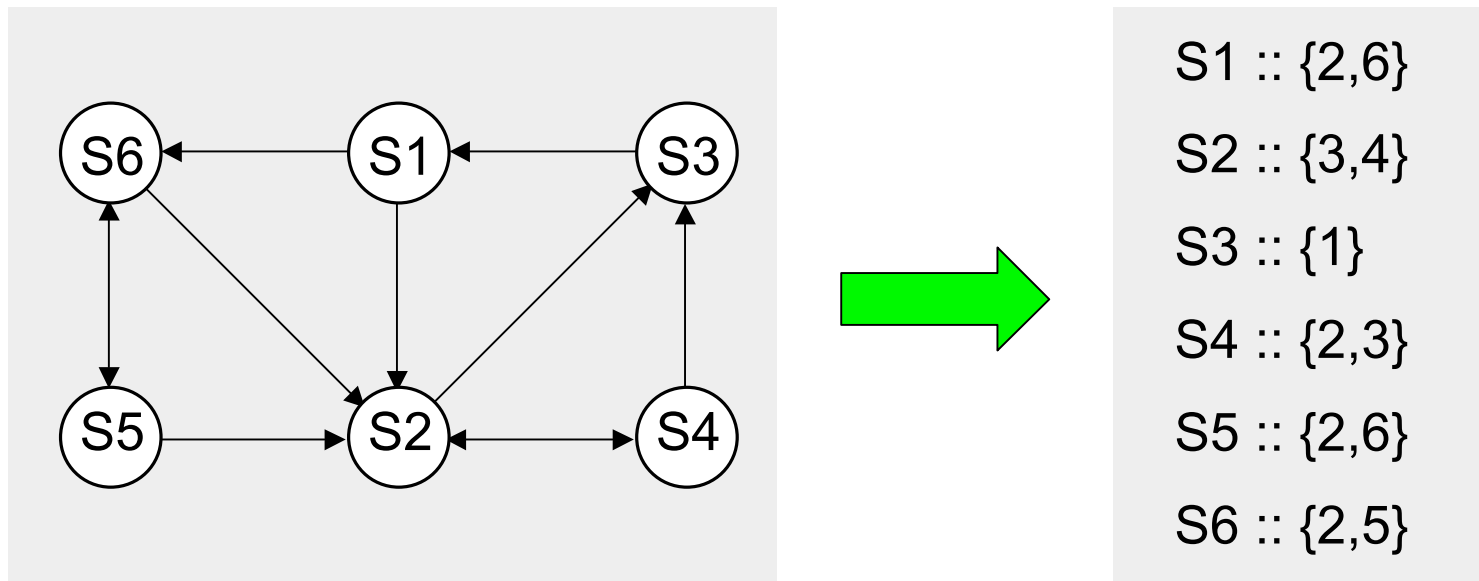
### CONSTRAINT

`circuit(Ncircuits, Successors)`



`circuit(2, {2,4,1,3,6,5})`

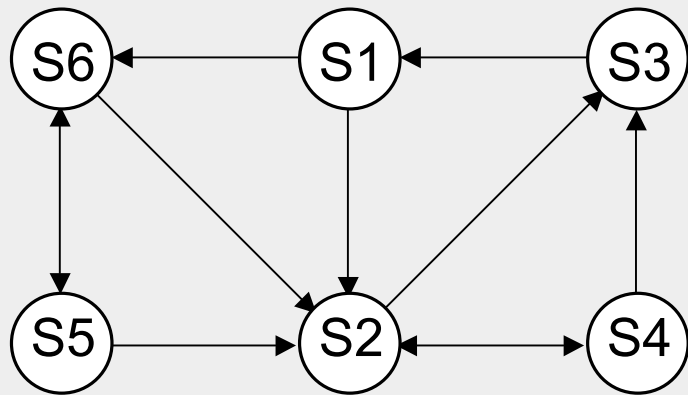
## Representation of the Graph



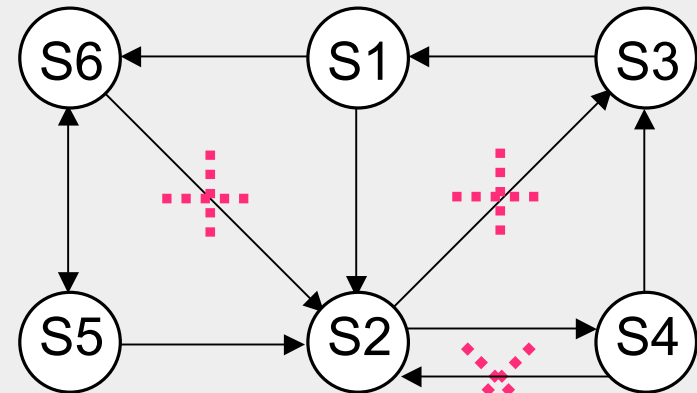
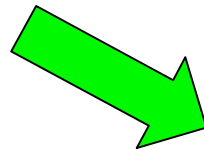
One **variable** for each node  $i$

**Values** are the potential successors of  $i$

# Constraint Propagation



circuit(N, {S1,S2,S3,S4,S5,S6})

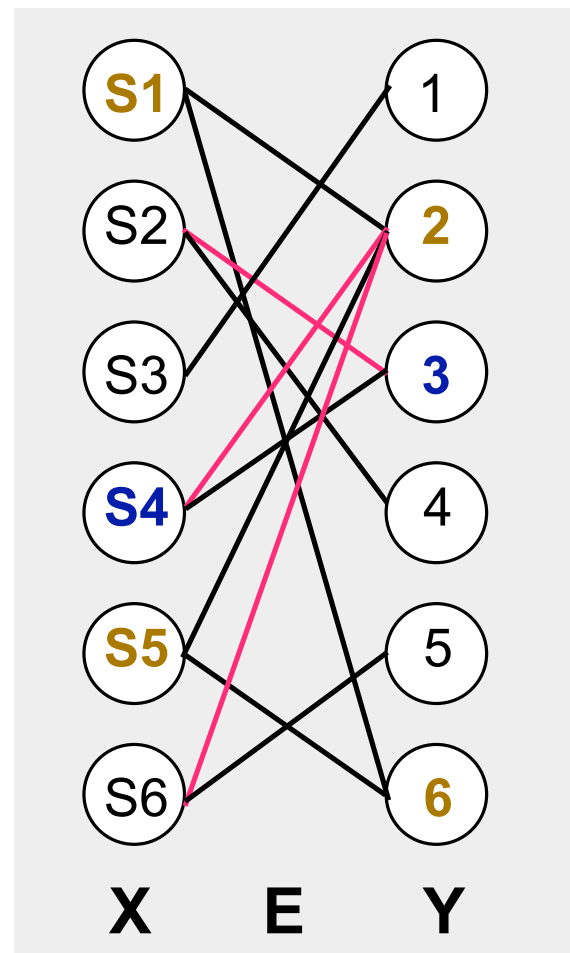
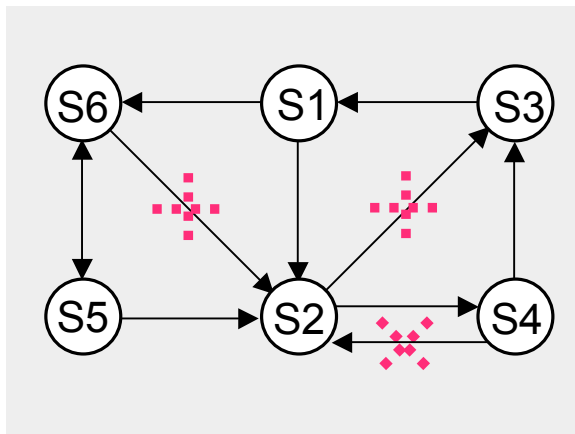


$N \leq 2$

$S6 \neq 2$     $S2 \neq 3$     $S4 \neq 2$

# Constraint Propagation as Maintaining Graph Invariants

Invariant : *maintain a maximum cardinality matching  $\mu$  of size 6*



$$\mu(G((X,Y), E) \geq 6$$

$$S1, S5: \{2, 6\} \Rightarrow S4 \neq 2$$

$$S4: \{3\} \Rightarrow S2 \neq 3$$

## Constraint Propagation as Maintaining Graph Invariants

One other invariant for the *circuit* constraint is:

*The number of circuits  $N$  is greater than or equal to the **number of strongly connected components** of the graph  $G$  to cover.*

$$N \geq \text{nsc}(G)$$

## The Thesis

Constraint Propagation  
=  
Maintaining Graph Invariants

Focus first on formula (invariants)  
and then on how to interpret these formula (algorithm)

## The Whole Idea

- (1) Describe global constraints in terms of graph properties
- (2) Maintain graph invariants involving those graph properties
- (3) Build a data base of graph invariants
- (4) Extract relevant invariants and interpret them

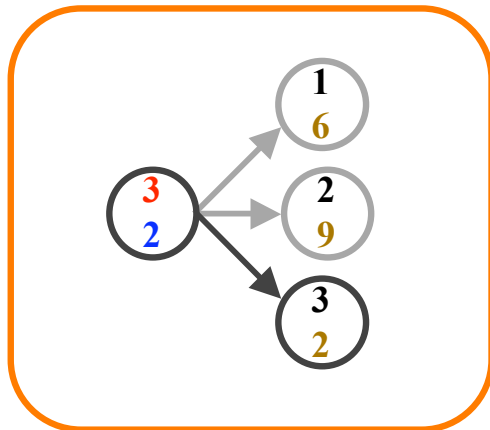
- Getting into the Topic

 • **Describing Global Constraints**

- Graph Based Filtering
- Conclusion

# Global Constraints

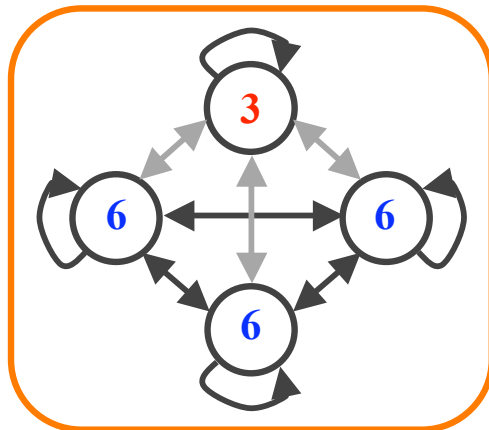
Common **structures** in discrete combinatorial problems.



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

`element(X,L,Y)`

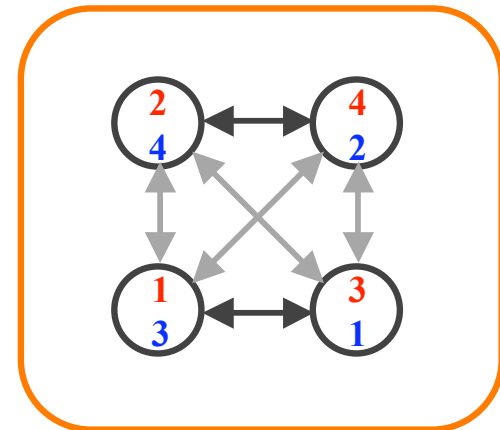
$L[X]=Y$



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

`nvalue(N,L)`

$N=|L|$



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

`symmetric_alldiff(L)`

$L$  in  $[1,n]$ , `alldifferent(L)`,  
 $X_i=j \Leftrightarrow X_j=i$

(*matching of card.  $n/2$* )

## Main Idea of the Classification

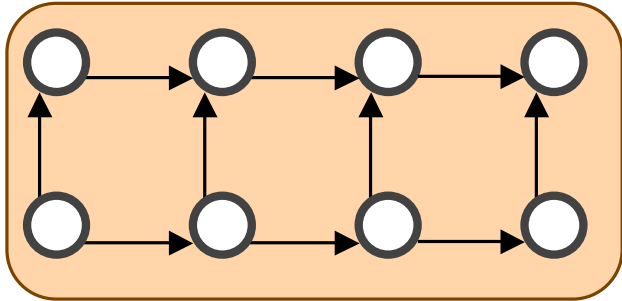
Global Constraints as:

**Graph Properties**

on **Structured Network**

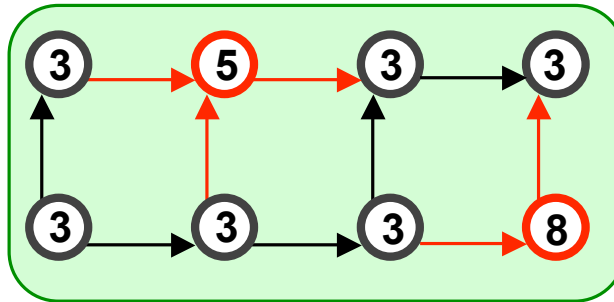
of **Elementary Constraints**

of the **Same Type**



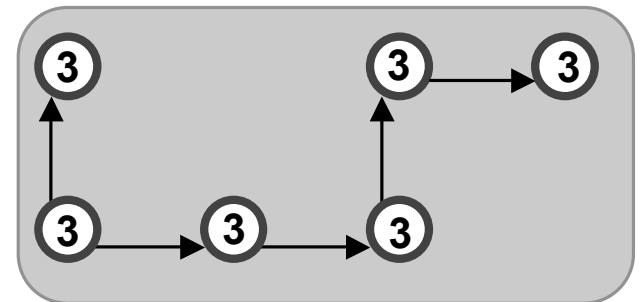
**Initial network**

- vertex: variable
- arc: equality ctr



**Variables get fixed:**

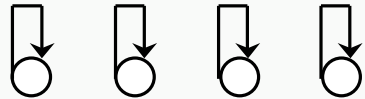
- some equalities hold,
- some equalities **do not hold**



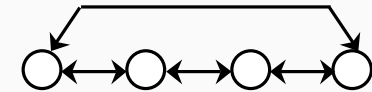
Ask propertie on sub-graph  
of constraints that still hold  
(i.e., one single connected component)

# Examples of Graph Generator

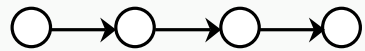
- **LOOP**



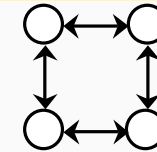
- **CYCLE**



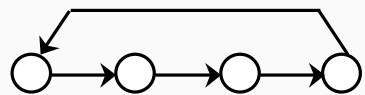
- **PATH**



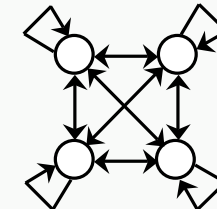
- **GRID**



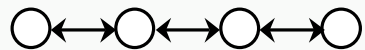
- **CIRCUIT**



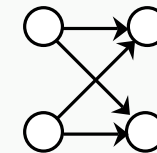
- **CLIQUE**



- **CHAIN**



- **PRODUCT**



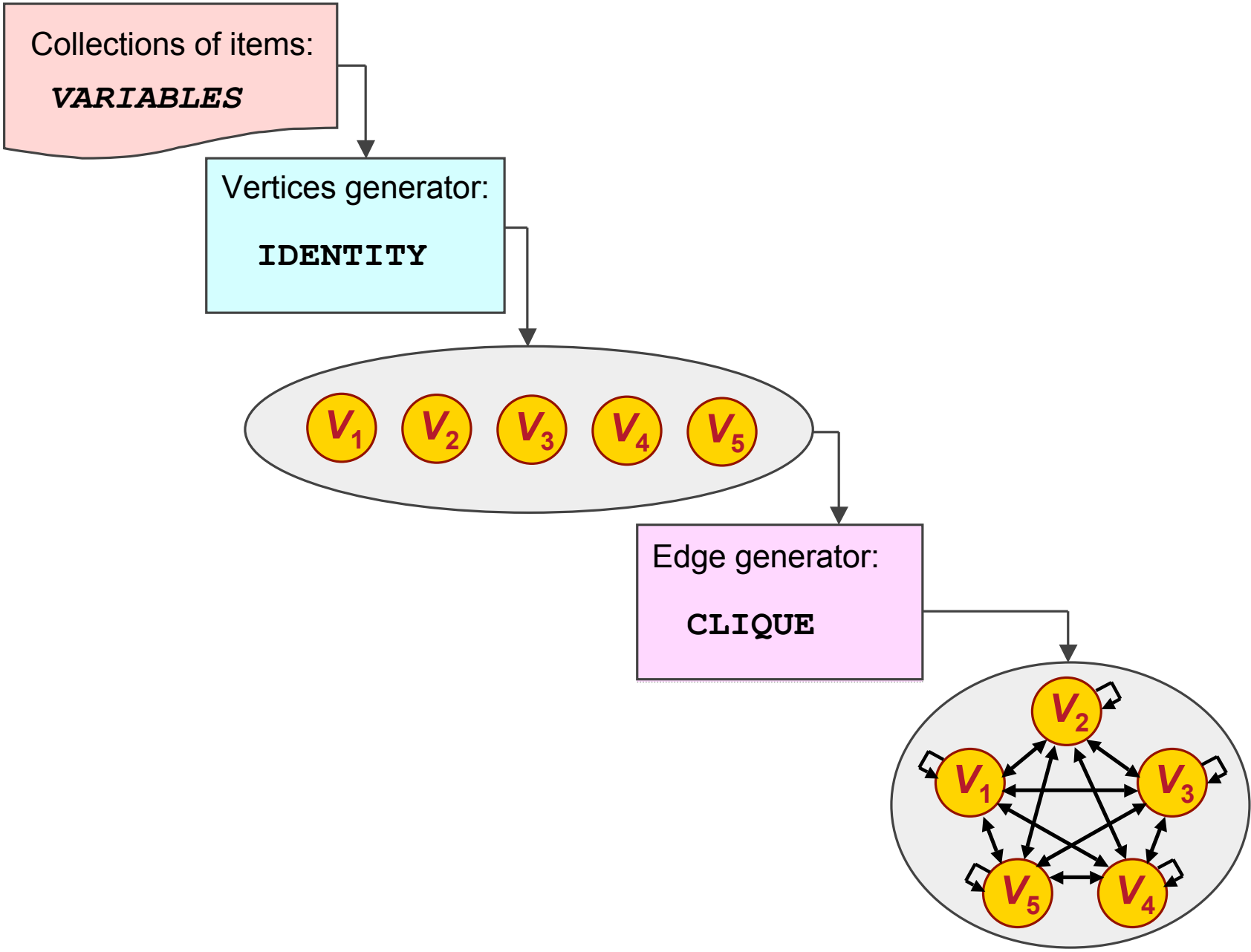
## Examples of Graph Properties

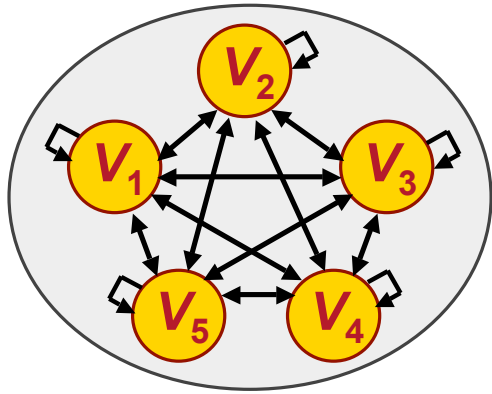
- **NVERTEX** : cardinality of  $V(G)$
- **NEDGE** : cardinality of  $E(G)$
- **NSOURCE** : number of vertices without any predecessor
- **NSINK** : number of vertices without any successor
- **NCC** : number of connected components of graph  $G$
- **MIN\_NCC** : number of vertices of the smallest c.c. of graph  $G$
- **MAX\_NCC** : number of vertices of the largest c.c. of graph  $G$
- **NSCC** : number of strongly connected components of graph  $G$
- **MIN\_NSCC** : number of vertices of the smallest s.c.c. of graph  $G$
- **MAX\_NSCC** : number of vertices of the largest s.c.c. of graph  $G$
- **MAX\_IN\_DEGREE** : number of predecessors of the vertex of  $G$  that has the maximum number of predecessors (without considering loops)

## nvalue(NVAL, VARIABLES )

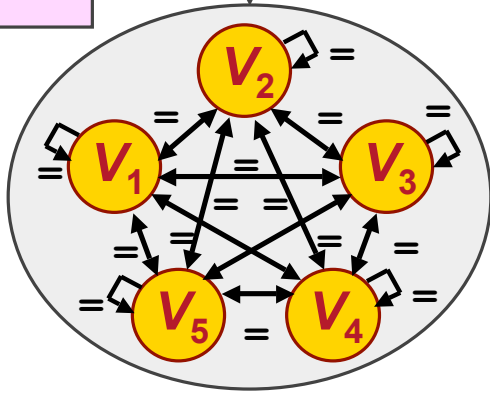
• ARGUMENT	:	NVAL	:	dvar
				VARIABLES: collection(var-dvar)
• RESTRICTION(S)	:	NVAL ≥ 0		
				NVAL ≤  VARIABLES
				required(VARIABLES.var)
• VERTEX GENERATOR	:	VARIABLES		
• EDGE GENERATOR	:	CLIQUE		
• EDGE ARITY	:	2		
• EDGE CONSTRAINT	:	VARIABLES.var[1] = VARIABLES.var[2]		
• GRAPH PROPERTY	:	NSCC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

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

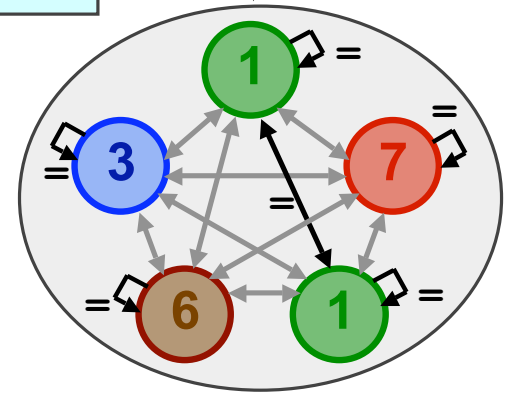




Edge constraint:  
=



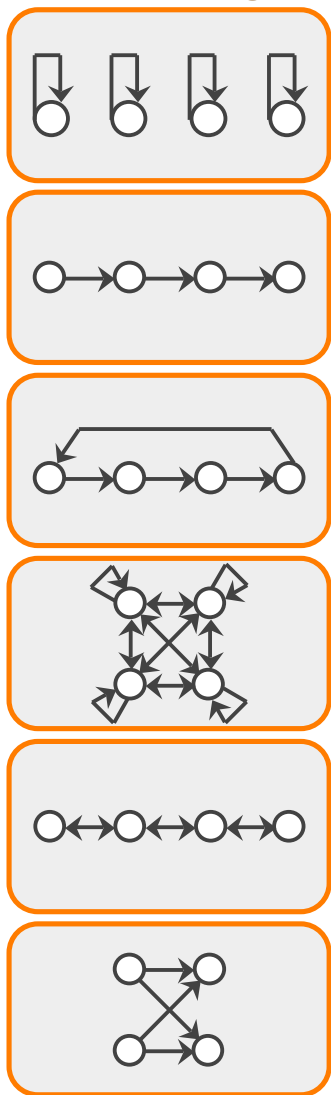
Graph property:  
 $NSCC=NVAL$



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

# Global Constraint "Space"

Structure of the initial graph



loop

path

circuit

clique

chain

product

Property of the final graph

asymmetric

equivalence

transitive

reflexive

symetric

`nvalue(4, {3, 1, 7, 1, 6`

`}}`

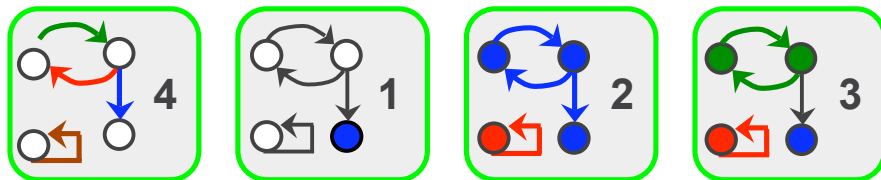
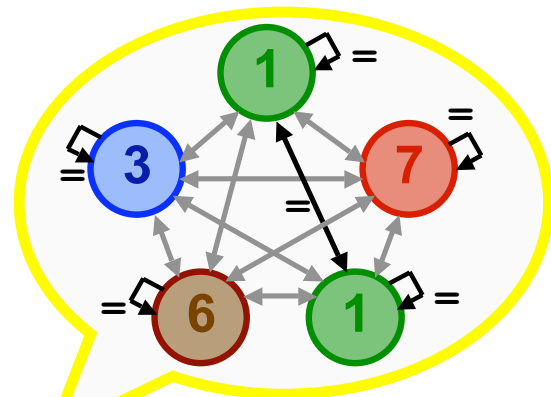
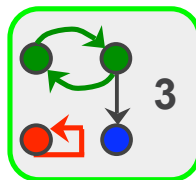
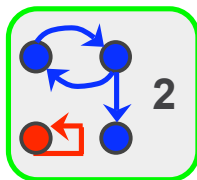
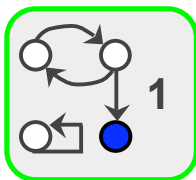
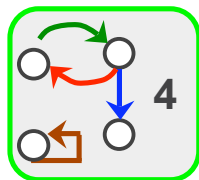
number of arcs

number of sinks

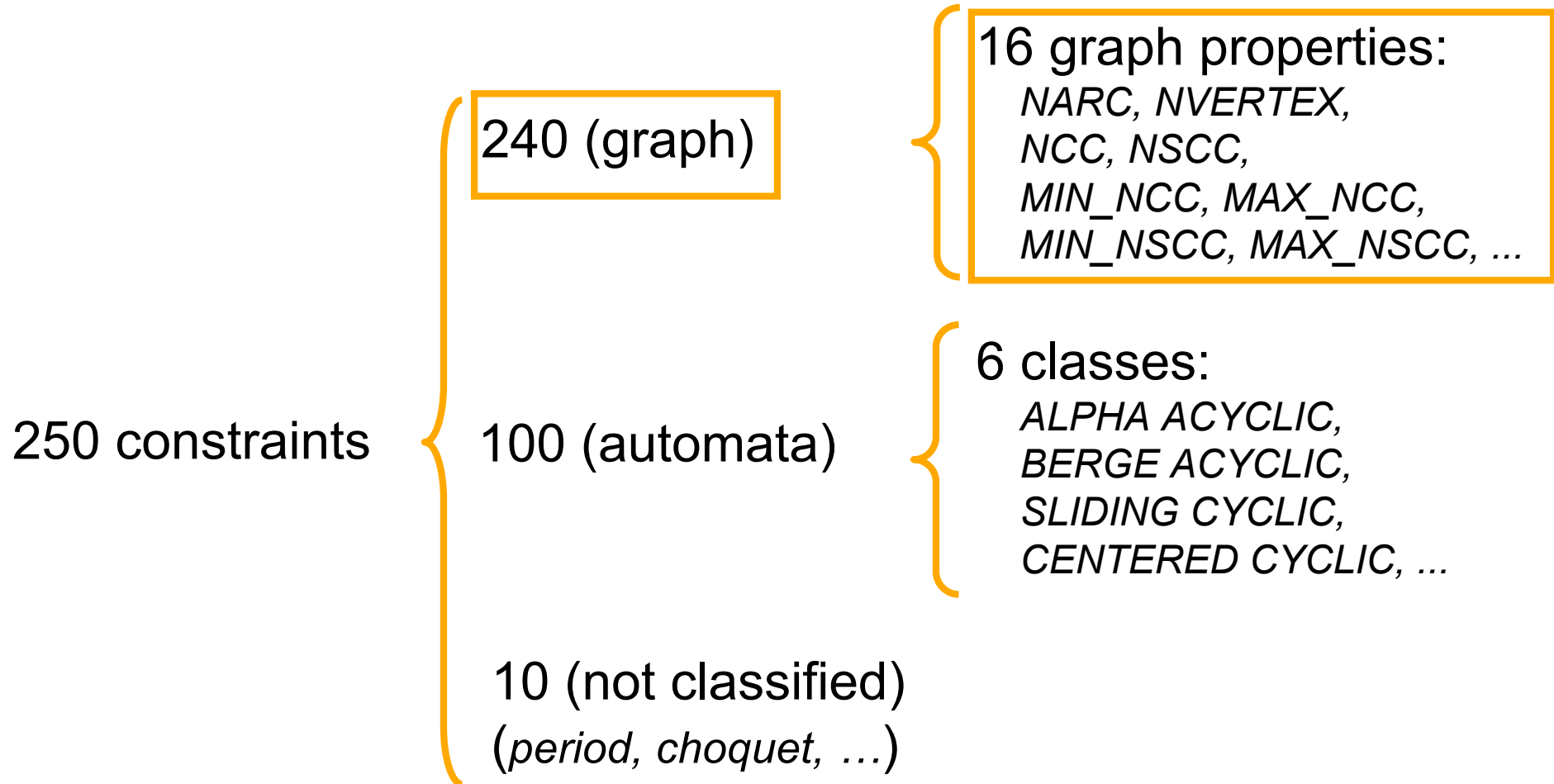
number of connected components

number of strongly connected components

Graph property



# A Catalog of Global Constraints



- Getting into the Topic
- Describing Global Constraints

- **Graph Based Filtering**

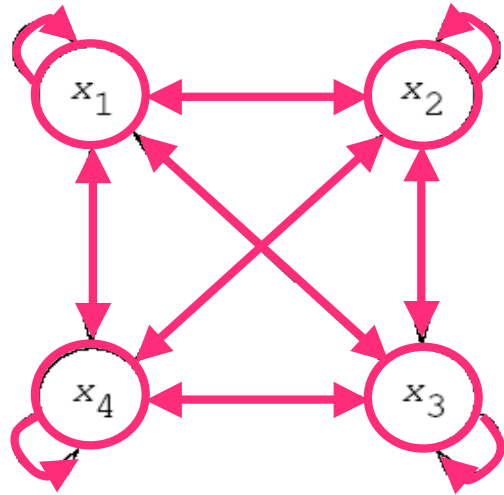


- **Introduction**

- Bounds of graph properties
    - Filtering back from the bounds to the arcs
    - Invariants linking several graph characteristics
    - Taking advantage from the graph structure

- Conclusion

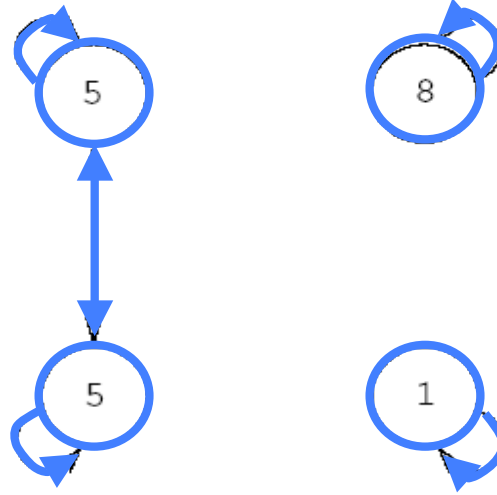
# Intermediate Graph



(A)

**Initial graph**

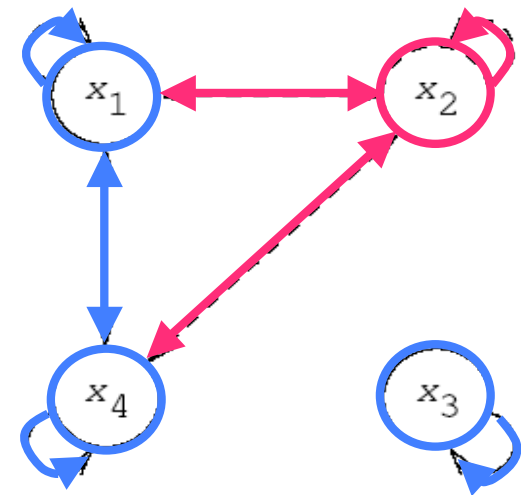
$nvalue(N, \{x_1, x_2, x_3, x_4\})$



(B)

**Final graph**

$nvalue(3, \{5, 8, 1, 5\})$



(C)

**Intermediate graph**

$nvalue(3, \{x_1, x_2, x_3, x_4\})$

$x_1 = x_4$

$x_1 \neq x_3 \quad x_2 \neq x_3 \quad x_3 \neq x_4$

Some vertices/arcs **belong for sure** to the final graph

(**T**-vertices, **T**-arcs)

Some vertices/arcs **may belong** to the final graph

(**U**-vertices, **U**-arcs)

# Graph Based Filtering

Reasoning on **one** single graph property:

- (1) From the status of the vertices/arcs **get bounds on that property**
- (2) From limits on the bounds of that property **propagate back to status**

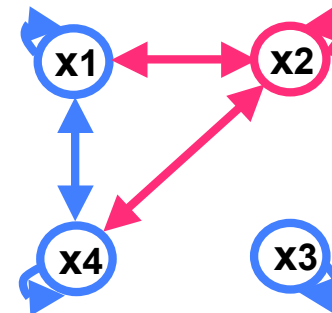
Reasoning on **several** graph properties on the same graph:

- (3) Use invariants linking several properties

$$(1) \min(\mathbf{NSCC}) \geq 2 \quad \max(\mathbf{NSCC}) \leq 3$$

$$(2) \mathbf{NSCC} \geq 3 \Rightarrow \mathbf{x}_1 \neq \mathbf{x}_2, \quad \mathbf{x}_2 \neq \mathbf{x}_4$$

$$(3) \mathbf{NSCC} \geq \left\lceil \frac{\mathbf{NVERTEX}^2}{\mathbf{NARC}} \right\rceil \quad (\text{Turán, 1941})$$



- Getting into the Topic
- Describing Global Constraints
- Graph Based Filtering
  - Introduction
  - **– Bounds of graph properties**
  - Filtering back from the bounds to the arcs
  - Invariants linking several graph characteristics
  - Taking advantage from the graph structure
- Conclusion

## Bounds ( no isolated vertices in the final graph )

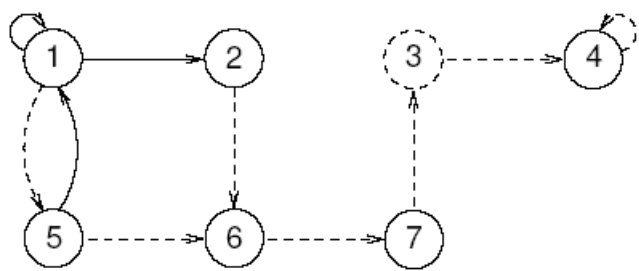
Graph characteristics	Bound
<u>NARC</u>	$ E_T  +  X_{T,-T}  - \mu(\vec{G}(X_{T,-T}, E_U))$
<u>NARC</u>	$ E_{TU} $
<u>NVERTEX</u>	$ X_T  + h(\vec{G}((X_{T,-T,-T}, X_{U,-T,T}), E_{U,T}))$
<u>NVERTEX</u>	$ X_{TU} $
<u>NCC</u>	$ cc_{[ X_T  \geq 1]}(\vec{G}(X_{TU}, E_{TU})) $
<u>NCC</u>	$ cc_{[ E_T  \geq 1]}(\vec{G}(X_T, E_T))  + \mu_l(\vec{G}_{rem})$
<u>NSCC</u>	$ scc_{[ X_T  \geq 1]}(\vec{G}(X_{TU}, E_{TU}))  + h(G_{\underline{NSCC}}((Y, Z), E))$
<u>NSCC</u>	$ scc(\vec{G}(X_{TU}, E_T)) $
<u>NSINK</u>	$ sink_{[ X_T =1]}(\vec{G}(X_{TU}, E_{TU}))  + h(G'_r((Y, Z), E))$
<u>NSINK</u>	$ sink(\vec{G}(X_T, E_T))  +  X_U  -  source_{[ X_U =1]}(\vec{G}(X_{TU}, E_{TU}))  -  XP $

# Minimum Number of Arcs

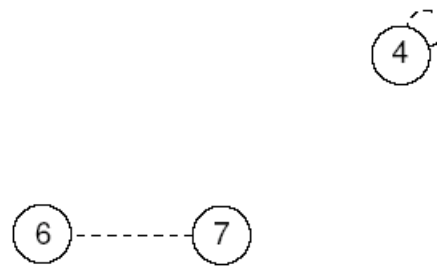
cardinality of a maximum matching

$$\underline{\text{NARC}} \geq |E_T| + |X_{T,-T}| - \mu(\overleftrightarrow{G}(X_{T,-T}, E_U))$$

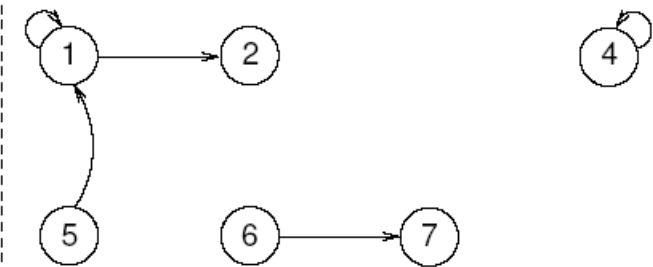
3
3
1



Intermediate digraph



$\overleftrightarrow{G}(X_{T,-T}, E_U)$



A solution reaching 5 arcs

$E_U$  : Arcs to be undetermined

$E_T$  : Arcs to be true

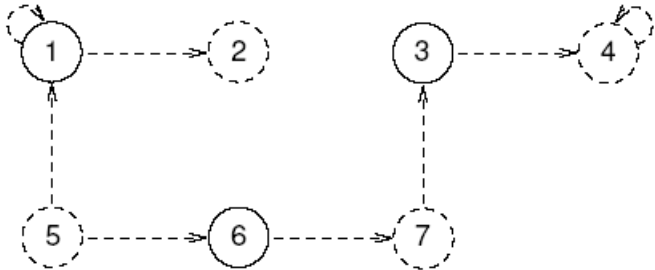
$X_{T,-T}$  : Vertices to be true with all adjacent arcs to be undetermined

# Minimum Number of Vertices

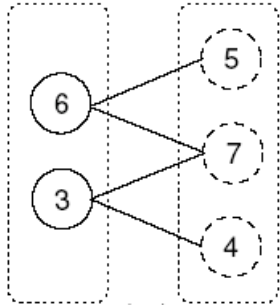
cardinality of a minimum hitting set

$$\underline{\text{NVERTEX}} \geq \underbrace{|X_T|}_3 + \underbrace{h(\overleftrightarrow{G}((X_{T,-T,-T}, X_{U,-T,T}), E_{U,T}))}_1$$

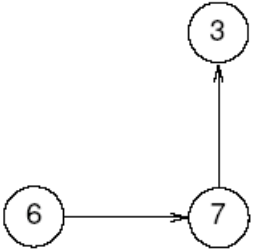
$X_{T,-T,-T}$     $X_{U,-T,T}$



Intermediate digraph



$\overleftrightarrow{G}((X_{T,-T,-T}, X_{U,-T,T}), E_{U,T})$



A solution reaching 4 vertices

$X_{T,-T,-T}$  : True vertices that are not the extremity of any true arcs and that have not a true vertex as neighbour

$X_{U,-T,T}$  : Undetermined vertices that has at least one true vertex as neighbour

$E_{U,T}$  : Undetermined arcs such that at least one of their extremities is a true vertex

- Getting into the Topic
- Describing Global Constraints
- Graph Based Filtering
  - Introduction
  - Bounds of graph properties
  - **– Filtering back from the bounds to the arcs**
  - Invariants linking several graph characteristics
  - Taking advantage from the graph structure
- Conclusion

## Filtering Back From the Maximum Number of Arcs

If  $\overline{\text{NARC}} = |E_T| + |X_{T,-T}| - \mu(\overleftrightarrow{G}(X_{T,-T}, E_U))$  then:

1. All arcs  $(u, v) \in E_U$  such that  $u$  or  $v$  belongs to two, not necessarily distinct, connected components in  $cc_{[|E_T| \geq 1]}(\overleftrightarrow{G}(X_T, E_T))$  can be turned into  $F$ -arcs.
2. All arcs in  $E_{U,U,U}$  can be turned into  $F$ -arcs.
3. All  $U$ -vertices in  $X_{U,-T,-T}$  (i.e., not connected to any  $T$ -vertex) can be turned into  $F$ -vertices.
4. For all edges  $e = (u, v)$ ,  $u \neq v$ , of  $\overleftrightarrow{G}(X_{T,-T}, E_U)$  such that  $u, v \in X_{T,-T}$  and  $e$  does not belong to any maximum matching of  $\overleftrightarrow{G}(X_{T,-T}, E_U)$ , the corresponding arcs  $(u, v)$  and  $(v, u)$  of  $\overleftrightarrow{G}(X_{T,-T}, E_U)$  can be turned into  $F$ -arcs.
5. For all edges  $e = (u, v)$ ,  $u \neq v$ , of  $\overleftrightarrow{G}(X_{T,-T}, E_U)$  such that  $u, v \in X_{T,-T}$  and  $e$  belongs to all maximum matchings of  $\overleftrightarrow{G}(X_{T,-T}, E_U)$ , if a unique arc  $(u, v)$  or  $(v, u)$  in  $\overleftrightarrow{G}(X_{T,-T}, E_U)$  corresponds to  $e$  then this arc can be turned to a  $T$ -arc.
6. All arcs  $e = (u, v) \in E_U$  such that  $u \in X_{T,-T}$  is saturated in all maximum matchings of  $\overleftrightarrow{G}(X_{T,-T}, E_U)$  and  $v \in X_U$  can be turned into  $F$ -arcs.
7. All loops  $e = (u, u) \in E_U$  such that  $u \in X_{T,-T}$  is saturated in all maximum matchings of  $\overleftrightarrow{G}(X_{T,-T}, E_U)$  can be turned into  $F$ -arcs.

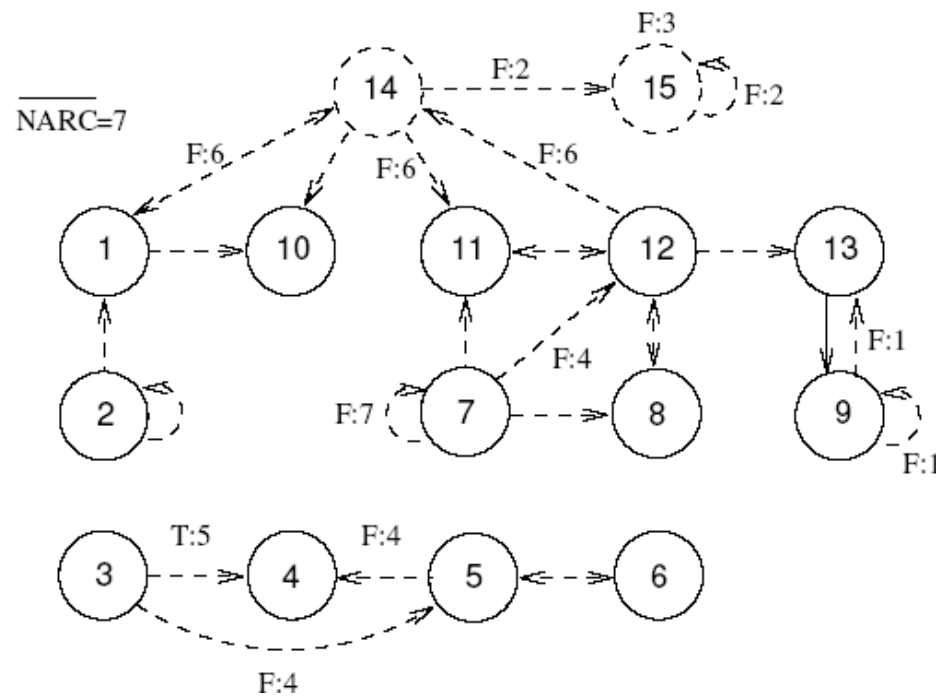
# Filtering Back From the Maximum Number of Arcs

If  $\overline{\text{NARC}} = |E_T| + |X_{T,-T}| - \mu(\overleftarrow{G}(X_{T,-T}, E_U))$  then:

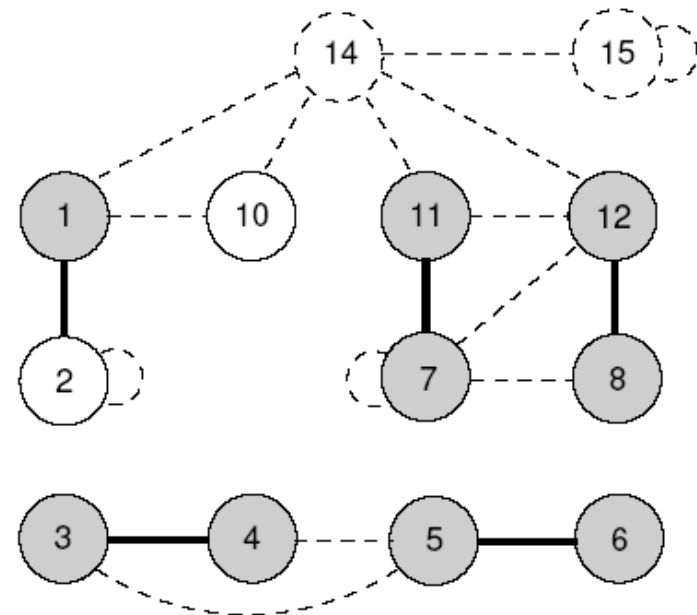
1

11

5



Intermediate digraph



$\overleftarrow{G}(X_{T,-T}, E_U)$

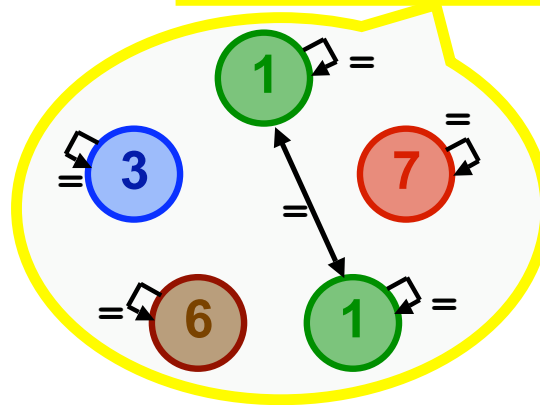
- Getting into the Topic
- Describing Global Constraints
- Graph Based Filtering
  - Introduction
  - Bounds of graph properties
  - Filtering back from the bounds to the arcs
  - **– Invariants linking several graph characteristics**
  - Taking advantage from the graph structure
- Conclusion

## Motivation

A lot of global constraints use **more than one** graph characteristics.

**Example:** consider the *nvalue* constraint and suppose we want a *balanced assignment* by restricting the min. and max. numbers of occurrences of any value effectively used.

`balanced_nvalue(4, 1, 2, {3, 1, 7, 1, 6})`



MIN\_NSCC=1  
MAX\_NSCC=2

This involves constraining the `MIN_NSCC` and `MAX_NSCC` graph characteristics.

## Idea

A lot of global constraints use more than one graph characteristics.

But graph characteristics are not independent, they are related by graph invariants.

Graph invariants have been collected in a database.

Given a global constraint  $C$  specified in terms of graph characteristics, the relevant graph invariants form necessary conditions for  $C$ .

---

**Example:** in the context of *balanced\_nvalue* you have:

$$NVERTEX \leq \max(NSCC - 1, 0) * MAX\_NSCC + MIN\_NSCC$$

$$NVERTEX \geq \max(NSCC - 1, 0) * MIN\_NSCC + MAX\_NSCC$$

## A data base of graph invariants

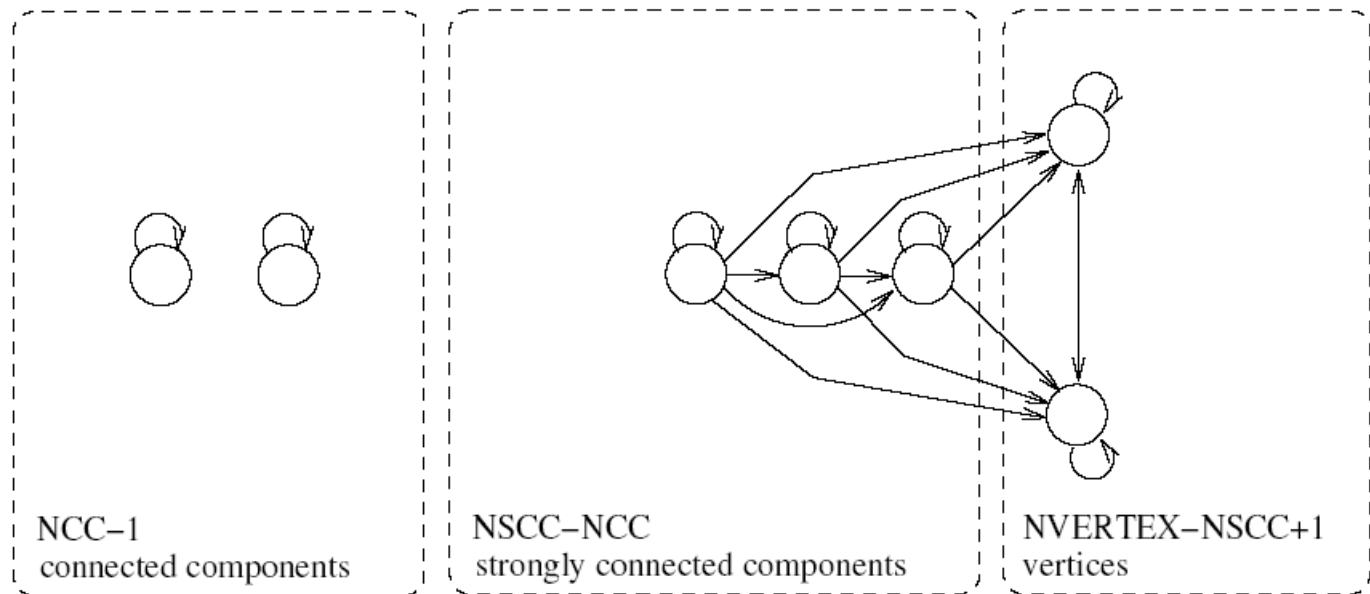
Indexed on: graph class (see later) and graph characteristics mentioned by the constraint of interest.

#graphs	#GC	#invariants
1	1	13
1	2	50
1	3	34
1	4	12
1	5	2
2	2	10
2	3	10
2	4	6
2	5	16
2	6	4

## Example of Graph Invariant

$$\text{NARC} \leq \text{NCC} - 1 + (\text{NVERTEX} - \text{NSCC} + 1) \cdot (\text{NVERTEX} - \text{NCC} + 1) + \frac{(\text{NSCC} - \text{NCC} + 1) \cdot (\text{NSCC} - \text{NCC})}{2}$$

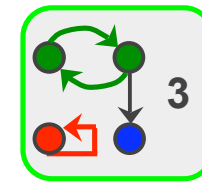
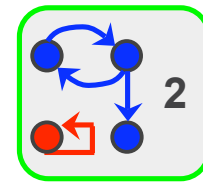
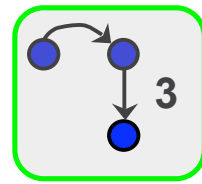
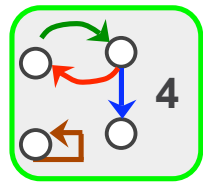
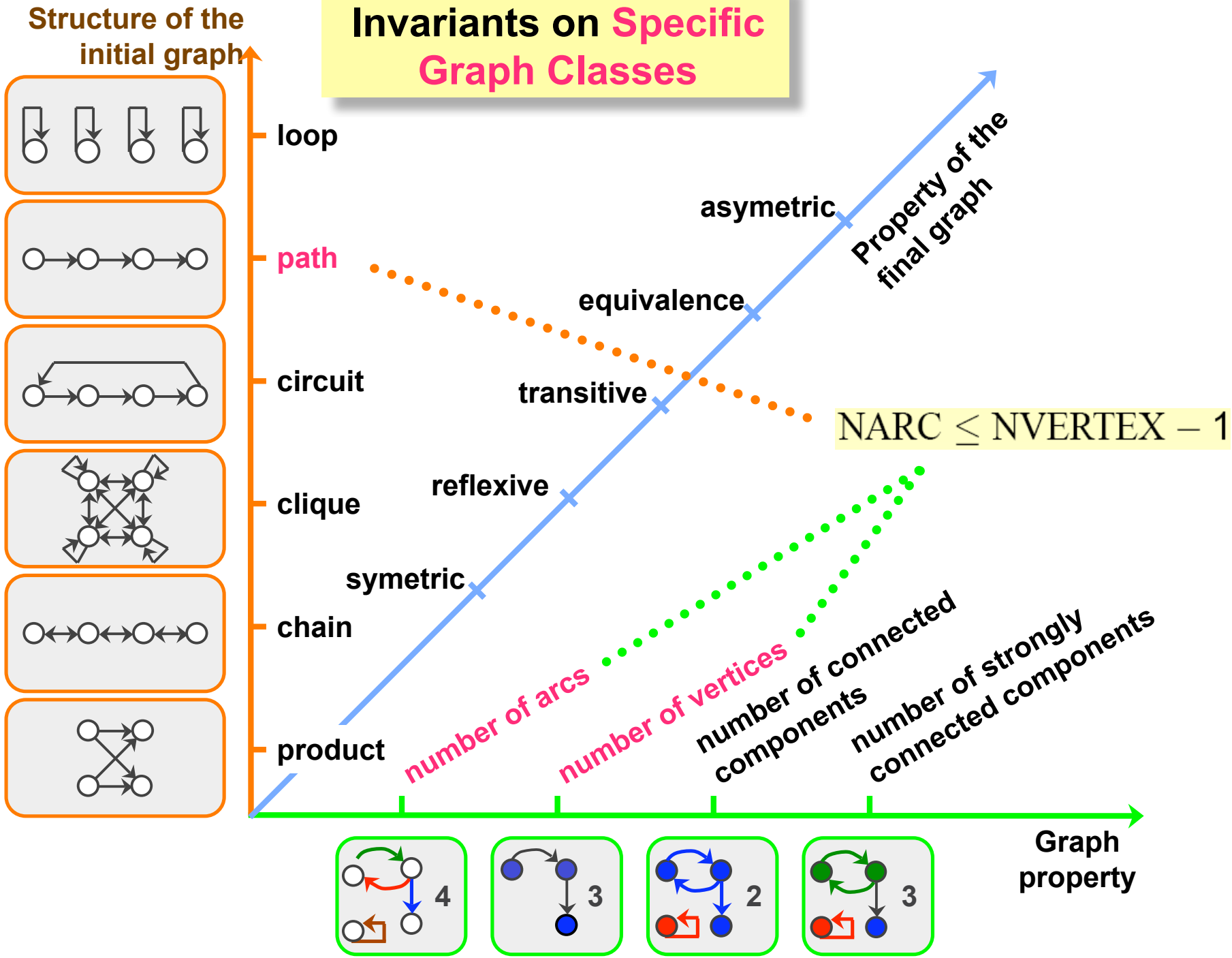
Pattern that achieves the maximum number of **arcs** for a fixed number of **vertices**, a fixed number of **connected components** and a fixed number of **strongly connected components**



- Getting into the Topic
- Describing Global Constraints
- Graph Based Filtering
  - Bounds of graph properties
  - Filtering back from the bounds to the arcs
  - Invariants linking several graph characteristics
  - **Taking advantage from the graph structure**
- Conclusion



# Invariants on Specific Graph Classes



## Idea

By considering the **structure of the final graph** we can get **tighter** bounds as well as **tighter** graph invariants.

- ▶ A general graph invariant:

$$\text{NARC} \leq \text{NVERTEX}^2$$

- ▶ A tighter graph invariant that holds for arc generator PATH:

$$\text{NARC} \leq \text{NVERTEX} - 1$$

# Graph Class

binary constraint	arc generator
<ul style="list-style-type: none"> <li>• acyclic</li> <li>• bipartite</li> <li>• <b>equivalence</b></li> <li>• no_loop</li> <li>• one_succ</li> </ul>	<ul style="list-style-type: none"> <li>• CHAIN</li> <li>• CIRCUIT</li> <li>• PATH</li> <li>• PRODUCT</li> <li>• SYMMETRIC_PRODUCT</li> </ul>

## nvalue

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

FINAL GRAPH  
of nvalue :

**a set of cliques**

## Examples of Tighter Graph Invariants

$$\text{MIN\_NCC} > 0 \Rightarrow \text{NARC} \geq \max(1, \text{MIN\_NCC} - 1)$$

$$\text{symmetric}: \text{MIN\_NCC} > 0 \Rightarrow \text{NARC} \geq \max(1, 2 \cdot \text{MIN\_NCC} - 2)$$

$$\text{equivalence}: \text{NARC} \geq \text{MIN\_NCC}^2$$

$$\text{arc\_gen} = \text{PATH}: \text{NARC} \geq \text{MIN\_NCC} - 1$$

# Specializing bounds

## General bounds

Graph characteristics	Bound
$\underline{\text{NARC}}$	$ E_T  +  X_{T,-T}  - \mu(\overrightarrow{G}(X_{T,-T}, E_U))$
$\overline{\text{NARC}}$	$ E_{TU} $
$\underline{\text{NVERTEX}}$	$ X_T  + h(\overrightarrow{G}((X_{T,-T,-T}, X_{U,-T,T}), E_{U,T}))$
$\overline{\text{NVERTEX}}$	$ X_{TU} $
$\underline{\text{NCC}}$	$ cc_{[ X_T  \geq 1]}(\overrightarrow{G}(X_{TU}, E_{TU})) $
$\overline{\text{NCC}}$	$ cc_{[ E_T  \geq 1]}(\overrightarrow{G}(X_T, E_T))  + \mu_l(\overrightarrow{G}_{rem})$
$\underline{\text{NSCC}}$	$ scc_{[ X_T  \geq 1]}(\overrightarrow{G}(X_{TU}, E_{TU}))  + h(G_{\text{NSCC}}((Y, Z), E))$
$\overline{\text{NSCC}}$	$ scc(\overrightarrow{G}(X_{TU}, E_T)) $
$\underline{\text{NSINK}}$	$ sink_{[ X_T =1]}(\overrightarrow{G}(X_{TU}, E_{TU}))  + h(G'_r((Y, Z), E))$
$\overline{\text{NSINK}}$	$ sink(\overrightarrow{G}(X_T, E_T))  +  X_U  -  source_{[ X_U =1]}(\overrightarrow{G}(X_{TU}, E_{TU}))  -  XP $

According to the initial graph: current bounds remain tight but can simplify them.

According to the final graph (symmetric, equivalence): current bounds may not be tight any more.

# Sketch of a Generic Graph Based Filtering Algorithm

**INPUT:** A global constraint  $C$  defined by:  $P_1 = V_1 \wedge P_2 = V_2 \wedge \dots \wedge P_n = V_n$

01. Creates the **intermediate graph  $G$**  of  $C$  and **evaluates the status of each arc** of  $C$
02. **Normalises** it according to the graph class of  $C$
03. **For** each graph characteristics  $P_i$  ( $1 \leq i \leq n$ ) **do**
04.  $P_{min} =$  **evaluate lower bound of  $P_i$**  on  $G$  according to the graph class of  $C$
05. adjust minimum value of  $V_i$  to  $P_{min}$
06.  $P_{max} =$  **evaluate upper bound of  $P_i$**  on  $G$  according to the graph class of  $C$
07. adjust maximum value of  $V_i$  to  $P_{max}$
08. **If**  $\min(V_i) = P_{max}$  **then**
09. Turn the status of some arc-constraints to True or False to **ensure  $\min(V_i) = P_{max}$**  and propagate the corresponding binary constraints
10. **If**  $\max(V_i) = P_{min}$  **then**
11. Turn the status of some arc-constraints to True or False to **ensure  $\max(V_i) = P_{min}$**  and propagate the corresponding binary constraints
12. **Propagate all graph invariants** (associated to the graph class  $C$ ) mentioning the graph characteristics  $P_1, P_2, \dots, P_n$

## Conclusion

There is still a lot of bounds and graph invariants to be found for **specific graph class** (necessary if want to be efficient)

But this is a task that can be done in an incremental way (much more easy than building a huge library of algorithms)

**Synthesize** { checkers  
filtering algorithms  
incremental moves  
visualization  
specialized heuristics } from the description of constraints

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

## CC(FD) for graphs

A. Design a graph-based language that allow to express formula

$$\underline{\text{NVERTEX}} \geq |X_T| + h(\overleftrightarrow{G}((X_{T,-T,-T}, X_{U,-T,T}), E_{U,T}))$$

The language should be **compositional**  
(i.e., you **build** intermediate graphs and **compute** something on them)  
see **Comet** (P.Van Hentenryck)

B. Design an efficient evaluator for this language

# Formula and family of extreme graphs

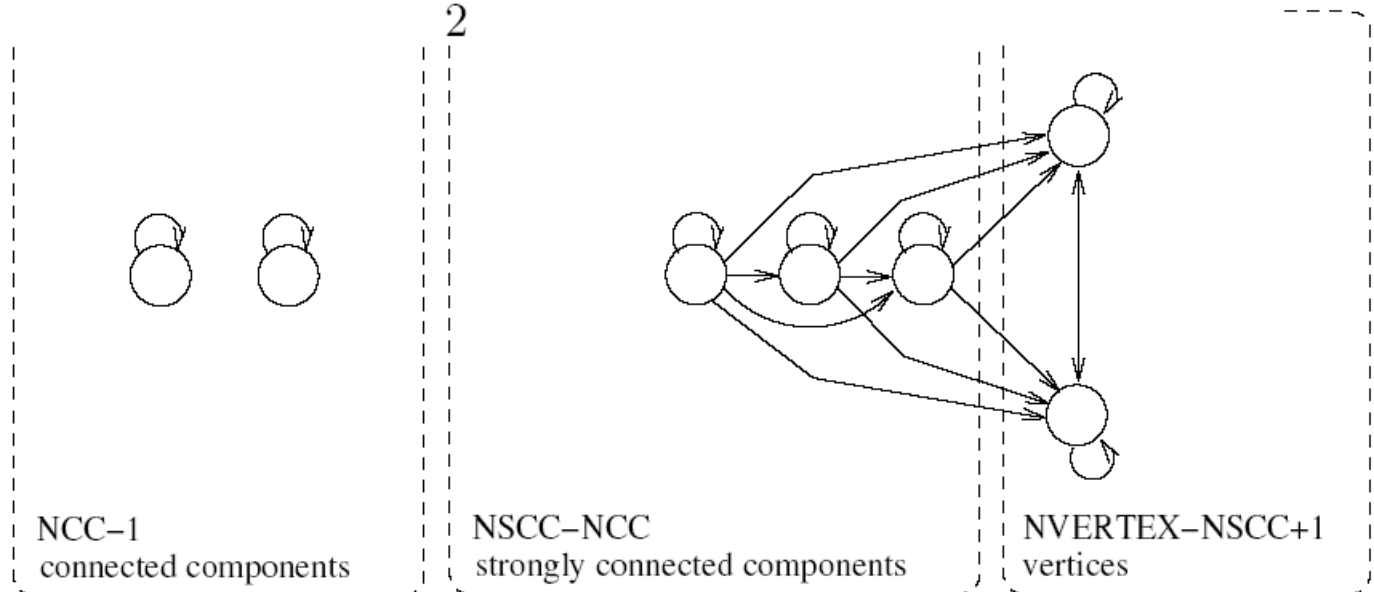
Graph invariants are nice since:

- they express a clean and universal piece of knowledge,
- they are somehow much more declarative than an algorithm.

But still the proof of a formula contains more information than the formula itself (very often the proof characterises a family of extreme graphs that would be useful for performing some constraint propagation)

$$\text{NARC} \leq \text{NCC} - 1 + (\text{NVERTEX} - \text{NSCC} + 1) \cdot (\text{NVERTEX} - \text{NCC} + 1) + \frac{(\text{NSCC} - \text{NCC} + 1) \cdot (\text{NSCC} - \text{NCC})}{2}$$

**QUESTION:**  
Would it be possible to get an explicit description of families of extreme graphs that are behind some formula?



## Remark

**Coming up with tight formula involving a lot of graph characteristics seems to be more and more difficult as the number of graph characteristics involved in the formula increases.**

**Perhaps one needs sometime to replace a formula by a greedy algorithm.**

## Question 1: Graph Invariants

1) Given a set of graph characteristics  $C_1, \dots, C_n$  how to produce in a systematic way graph invariants (for specific graph classes) of the form :

$$\text{NARC} \leq \text{formula}(C_1, \dots, C_n)$$

$$\text{NARC} \geq \text{formula}(C_1, \dots, C_n)$$

Is it true that :

$$\text{NARC} \leq (\text{NCC} - 1) \cdot \max(1, (\text{MIN\_NCC} - 1)) + \\ (\text{NVERTEX} - \text{NSCC} + 1) \cdot (\text{NVERTEX} - \text{NCC} + 1) + \\ \frac{(\text{NSCC} - \text{NCC} + 1) \cdot (\text{NSCC} - \text{NCC})}{2}$$

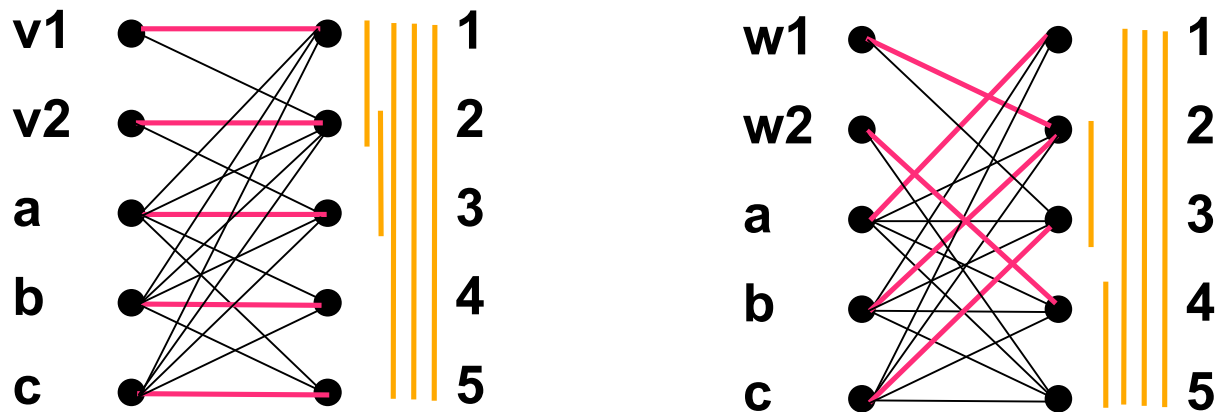
2) Are there some general proof methods (induction hypothesis) ?

3) How difficult it get when n increases (a formula linking 10 graph characteristics) ?

4) Is it possible to characterise in a systematic way the family of graphs associated to the extreme case (i.e., the equality) ?

## Question 2: Coinciding matchings on two convex bipartite graphs

Given two convex bipartite graphs that completely coincide on a given subset of vertices of the first class of vertices does there exist two maximum cardinality matching that coincide on these common vertices ? (Is this decision problem NP-hard)



Two convex bipartite graphs coinciding on a,b,c for which no maximum cardinality matching coinciding on a,b,c exists

If the problem is not NP-hard then give a polynomial algorithm for deciding if there exist a coinciding matching and characterise those edges which do not belong to any coinciding matching.

### Question 3: Identifying edges which do not belong to any maximum matching

Given an undirected graph  $G$  (not necessarily bipartite) and one maximum cardinality matching on that graph, is it possible to identify all edges of  $G$  which do not belong to any maximum cardinality matching in  $O(m)$ , where  $m$  is the number of edges of  $G$  ?

This problem occurs in several global constraints, e.g. :

- (1) *symmetric\_alldifferent* constraint (or *one\_factor*) of J.-C. Régin
- (2) some graph properties (like for instance NARC)

## Question 4: Generalized Global Cardinality Constraint

### INPUT :

- A set of integer variables  $V_1, \dots, V_n$ , each variable taking its value from a finite predefined set of values.
- Sets of values  $W_1, \dots, W_m$ , where two distinct set of values may intersect and integers  $low_1, \dots, low_m$  and  $up_1, \dots, up_m$  ( $low_i \leq up_i$ ).

### QUESTION :

- Does there exists a polynomial time algorithm for checking if there is at least one assignment for variables  $V_1, \dots, V_n$  that is both
  - ◆ compatible with each predefined set of values,
  - ◆ compatible with the lower and upper bounds on each set of values  $W_1, \dots, W_m$  (for each set of values  $W_i$  there should be between  $low_i$  and  $up_i$  variables of  $V_1, \dots, V_n$  that take their value in  $W_i$ ).

## Question 5: Strong articulation points

### QUESTION :

Given a directed graph that is strongly connected (with  $n$  vertices and  $m$  arcs), is there an efficient algorithm (i.e., with a worst-case complexity less than  $O(n.m)$ ) for identifying all vertices for which the removal creates more than one strongly connected component in the remaining graph.

