Describing Constraints with Meta-Data and Application to some Learning Problems

N. Beldiceanu

TASC (CNRS/INRIA) Mines de Nantes

nicolas.beldiceanu@mines-nantes.fr

PGMO seminar (16 April 2013)

Overview

- **Defining constraints**
- Different views attached to a constraint
- Describing constraints
- Dedicated description languages
- Around the catalog
- Learning constraints models

Motto of the presentation:
separate definition from usage

Separate Definition from Usage

Because a same definition can have **different** usages [Pitrat 93]

DEFINING CONSTRAINTS

Context independent concept

Right level of abstraction

An **expressive** and **concise** condition

involving a **non-fixed** number of variables

Don’t go directly to the usage (**and skip the defintion**)

Examples of constraints (*without any usage*)

alldifferent(VARIABLES)

Enforce all variables of the collection VARIABLES to take distinct values.

$$(\langle 5, 1, 9, 3 \rangle)$$

nvalue(NVAL, VARIABLES)

NVAL is the number of distinct values taken by the variables of the collection VARIABLES.

$$(4, \langle 3, 1, 7, 1, 6 \rangle)$$

cycle(NCYCLE, NODES)

Consider a digraph G described by the NODES collection. NCYCLE is equal to the number of circuits for covering G in such a way that each vertex of G belongs to one single circuit. NCYCLE can also be interpreted as the number of cycles of the permutation associated with the successor variables of the NODES collection.

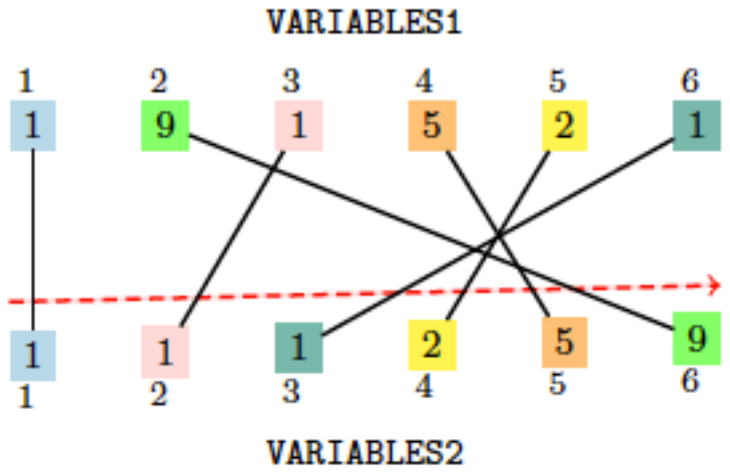
$$\left(2, \left\langle \begin{array}{ll} \mathrm{index}-1 & \mathrm{succ}-2, \\ \mathrm{index}-2 & \mathrm{succ}-1, \\ \mathrm{index}-3 & \mathrm{succ}-5, \\ \mathrm{index}-4 & \mathrm{succ}-3, \\ \mathrm{index}-5 & \mathrm{succ}-4 \end{array} \right\rangle \right)$$

Examples of constraints (*without any usage*)

`sort(VARIABLES1, VARIABLES2)`

The variables of the collection VARIABLES2 correspond to the variables of VARIABLES1 according to a permutation. The variables of VARIABLES2 are also sorted in increasing order.

$$\left(\begin{array}{c} \mathrm{var}-1, \\ \left\langle \begin{array}{c} \mathrm{var}-9, \\ \mathrm{var}-1, \\ \mathrm{var}-5, \end{array} \right\rangle, \\ \mathrm{var}-2, \\ \mathrm{var}-1 \\ \mathrm{var}-1, \\ \left\langle \begin{array}{c} \mathrm{var}-1, \\ \mathrm{var}-1, \\ \mathrm{var}-2, \\ \mathrm{var}-5, \end{array} \right\rangle \\ \mathrm{var}-9 \end{array}\right)$$



Examples of constraints (*without any usage*)

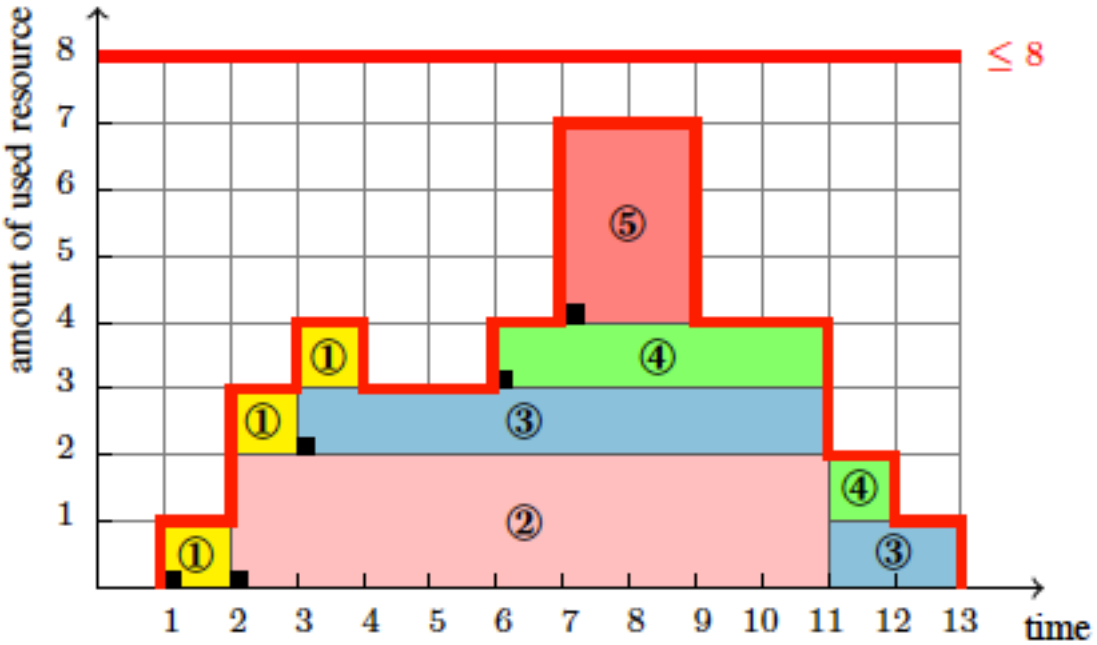
`cumulative(TASKS, LIMIT)`

Cumulative scheduling constraint or scheduling under resource constraints. Consider a set $\mathcal{T}$ of tasks described by the TASKS collection. The `cumulative` constraint enforces that at each point in time, the cumulated height of the set of tasks that overlap that point, does not exceed a given limit. A task overlaps a point i if and only if (1) its origin is less than or equal to i, and (2) its end is strictly greater than i. It also imposes for each task of $\mathcal{T}$ the constraint `origin` + `duration` = `end`.

TASKS

①	o − 1	d − 3	e − 4	h − 1
②	o − 2	d − 9	e − 11	h − 2
③	o − 3	d − 10	e − 13	h − 1
④	o − 6	d − 6	e − 12	h − 1
⑤	o − 7	d − 2	e − 9	h − 3

(o for origin, d for duration, e for end, h for height)



Declarative Aspect: Key Idea

- A constraint is **not an entire problem** !

  $\Rightarrow$ first difference from catalog of problems.

  (*allows to come up with **constraints** which can be **reused** across different problems*)

- There should be an **explicit description** of these constraints.

  $\Rightarrow$ second difference from catalog of problems.

  (*since different programs [that consider a **specific usage**] will exploit these structures, describing things with **natural language is useless***)

- Defining constraints
- **Different views attached to a constraint**
- Describing constraints
- Dedicated description languages
- Around the catalog
- Learning constraints models

Different usages of a constraint

- Checker view
- Feasibility view
- Filtering view
- Explanation view
- Degree of violation view
- Reification view
- Counting view
- Reformulation view
- Property view

will consider alldifferent
to illustrate these views

Checker view

- Testing whether a ground instance holds or not

  (*crucial for learning models*
  but useless for filtering !)

alldifferent

sort values and check that adjacent values are distinct

or alternatively

insert values into a hash table and check that no collision

Feasibility view

- Given a constraint where all variables are not yet fixed test whether the constraint has at least one solution or not.

alldifferent

1. construct variable-value graph (one vertex for each variable/value, one edge if val in domain of var)

2. cardinality of maximum matching is equal to the number of variables

Filtering view (*the most common view*)

- Given a constraint where all variables are not yet fixed, identify all variable-value pairs such that, if val is assigned to var, the constraint has no solution.

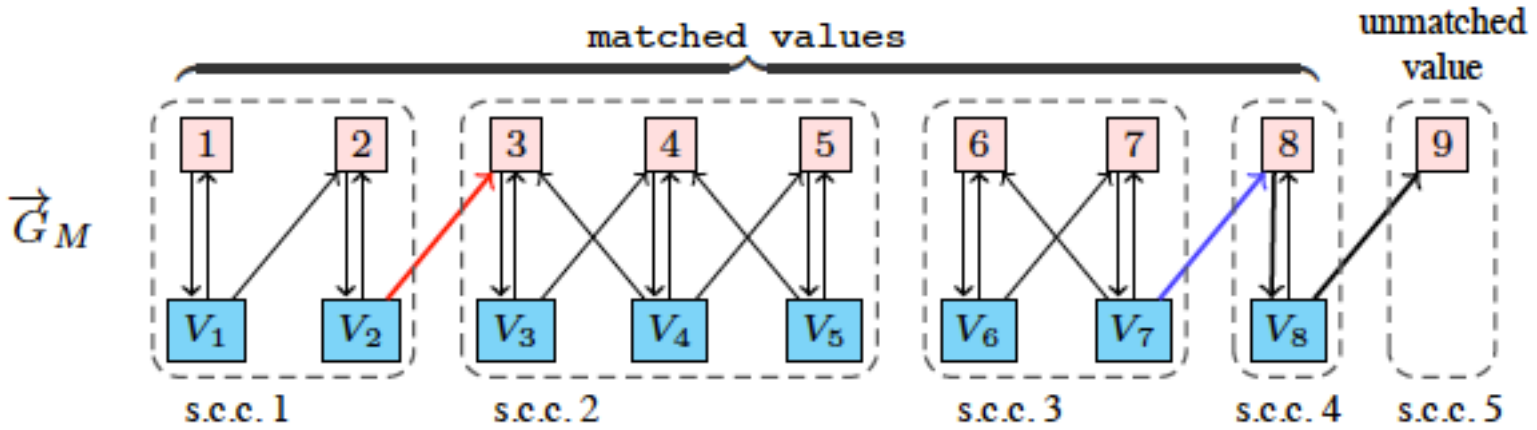
Filtering view

alldifferent

$\texttt{alldifferent}(\langle V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8\rangle)$

$V_1 \in [1,2]$, $V_2 \in [2,3]$, $V_3 \in [3,4]$, $V_4 \in [3,5]$, $V_5 \in [4,5]$, $V_6 \in [6,7]$, $V_7 \in [6,8]$, $V_8 \in [8,9]$

characterize edges that belong to at least one var-maximum matching



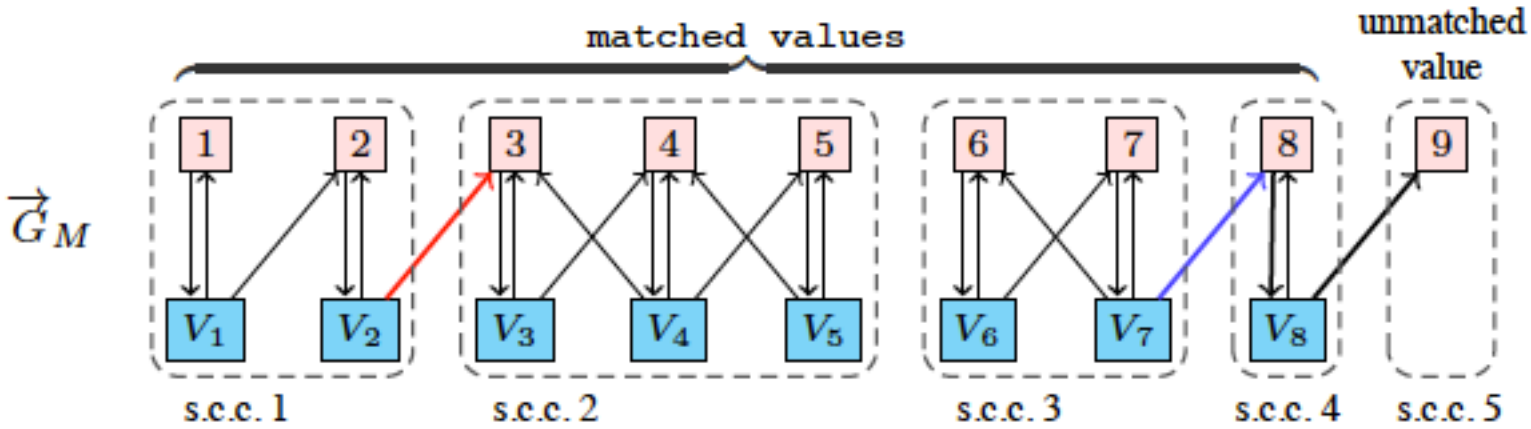
Explanation view (wrt filtering)

- Given a constraint where all variables are not yet fixed and an infeasible variable/value pair P, identify missing variable/values pairs such that if there are added back, P is no more infeasible

Explanation view (wrt filtering)

$\texttt{alldifferent}(\langle V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8 \rangle)$

$V_1 \in [1,2]$, $V_2 \in [2,3]$, $V_3 \in [3,4]$, $V_4 \in [3,5]$, $V_5 \in [4,5]$, $V_6 \in [6,7]$, $V_7 \in [6,8]$, $V_8 \in [8,9]$



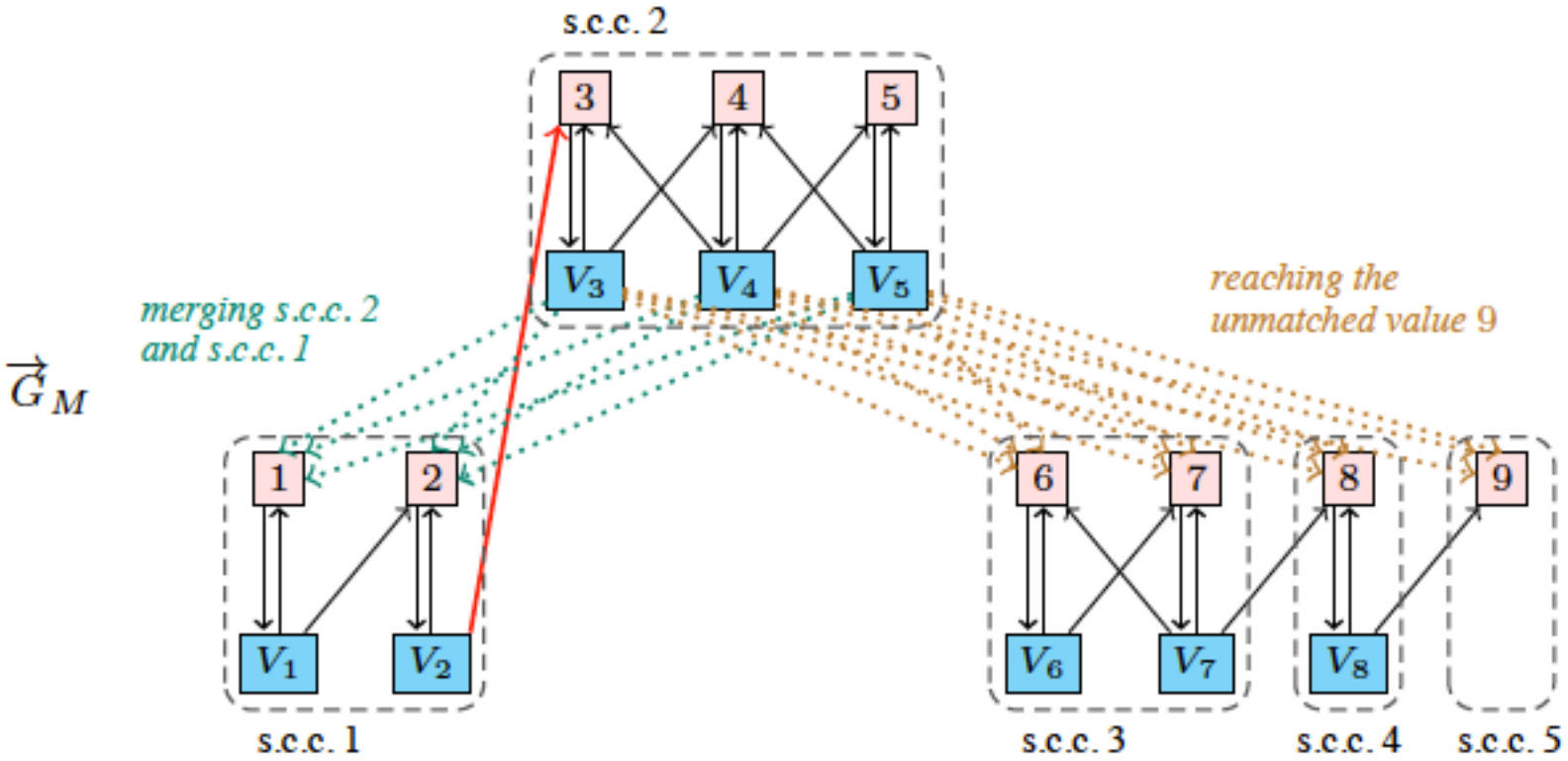
Explaining why (V_2=3) is not feasible

Explanation view (wrt filtering)

Explaining why (V_2=3) is not feasible:

(1) Arcs which merge s.c.c. 1 and s.c.c. 2

(2) Arcs which allow to reach an unmatched value



Degree of violation view

- Given a ground instance, evaluate its degree of violation, assuming that a satisfied constraint has 0 as degree of violation

variable-based degree of violation:

$\texttt{alldifferent}(\langle 2, 5, 2, 2, 5\rangle)$

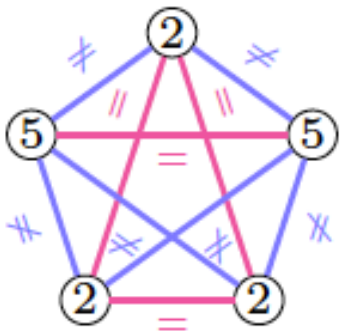
$\overset{1,3,4}{\texttt{alldifferent}(\langle 2, \overbrace{5, 2, 2}, 5\rangle)}$

$$\begin{aligned} \mathit{violation} &= (\#5 - 1) + (\#2 - 1) \\ &= (2 - 1) + (3 - 1) \\ &= 3 \end{aligned}$$

(three values need to be changed)

decomposition-based degree of violation:

$\texttt{alldifferent}(\langle 2, 5, 2, 2, 5\rangle)$



$\mathit{violation} = 4$

(four binary constraints are violated)

Reification view

- Associate a 0-1 variable *b* to a constraint *C* and maintain the equivalence between *b* and *C*.
- Done by reformulating C as a conjunction of pure functional dependency constraints with constraints that can be easily reified.

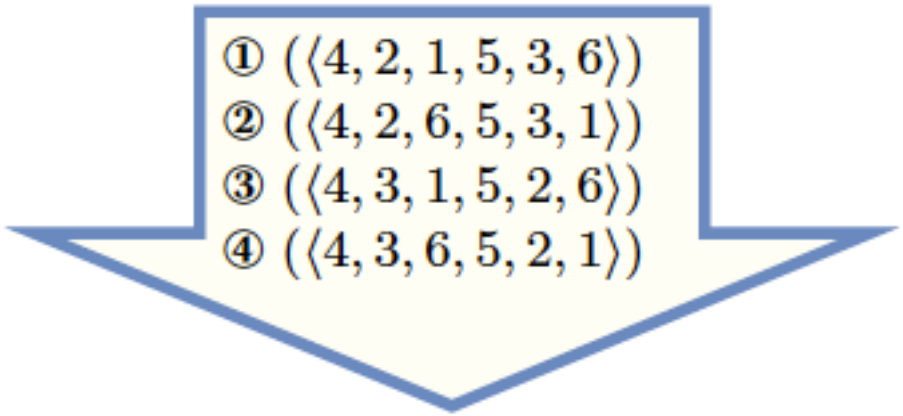
$$\textsc{alldifferent}(\langle v_1, \ldots, v_n \rangle)$$

$$\textsc{sort}(\langle v_1, \ldots, v_n \rangle, \langle w_1, \ldots, w_n \rangle) \wedge$$
$$(w_1 < w_2 \wedge \cdots \wedge w_{n-1} < w_n) \Leftrightarrow b$$

Counting view

- Given a constraint where all variables are not yet fixed, count (or estimate) its number of solutions (*useful for ranking constraints by their tightness*)

$$V_1 \in [2,4], V_2 \in [2,3], V_3 \in [1,6], V_4 \in [2,5], V_5 \in [2,3], V_6 \in [1,6],$$
$$\texttt{alldifferent}(\langle V_1, V_2, V_3, V_4, V_5, V_6 \rangle)$$



For alldifferent equivalent to computing number of maximum matching in a bipartite graph, #P-complete, so use approximations

Reformulation view

- Reformulation a constraint as a conjunction of other constraints (*since not available or for getting some filtering*)
- alldifferent: decomposition in disequalities
- alldifferent: 0/1 decomposition ensuring BC

For each potential interval $[l, u]$

0-1 variables $B_{1,l,u}, B_{2,l,u}, \ldots, B_{|\texttt{VARIABLES}|,l,u}$

$B_{i,l,u} \Leftrightarrow \texttt{VARIABLES}[i].\texttt{var} \in [l, u]$

$$B_{1,l,u} + B_{2,l,u} + \cdots + B_{|\texttt{VARIABLES}|,l,u} \leq u - l + 1$$

Property view

- E.g.: Contractible, Extensible, Functional dependency, Symmetries, Invariants, Compression, …
- Useful for ranking constraints and for proving that a conjunction of identical constraints is implied by another conjunction

- alldifferent

  given a ground satisfied instance we have:

Contractible:	**removing values** preserves satisfiability
Variable symmetry:	**sorting values** preserves satisfiability
Value symmetry:	**swapping two values** or **replacing a value by a new value** preserves satisfiability

- Defining constraints
- Different views attached to a constraint
- **Describing constraints**
- Dedicated description languages
- Around the catalog
- Learning constraints models

Meta data describing a constraint (*alldifferent*)

- Constraint
- Arguments
- Restrictions
- Typical

$$\texttt{alldifferent}(\texttt{VARIABLES})$$

$$\texttt{VARIABLES} \ : \ \texttt{collection}(\texttt{var}-\texttt{dvar})$$

$$\texttt{required}(\texttt{VARIABLES}, \texttt{var})$$

$$|\texttt{VARIABLES}| > 2$$

Typical used by constraint seeker to filter out constraints

Meta data describing a constraint (*alldifferent*)

- Symmetries
  - Items of VARIABLES are permutable.
  - Two distinct values of VARIABLES.var can be swapped; a value of VARIABLES.var can be renamed to any unused value.
- Arg Properties

Contractible wrt. VARIABLES.

Meta data (see also)

semantic links

assignment dimension added
assignment dimension removed
attached to cost variant
common keyword
comparison swapped
cost variant
generalisation
hard version
implied by
implies
implies (if swap arguments)
implies (items to collection)
negation
part of system of constraints
related
related to a common problem
root concept
shift of concept
soft variant
specialisation
system of constraints
used in graph description
used in reformulation
uses in its reformulation

alldifferent

implied by: `alldifferent_consecutive_values`, `circui`
`strictly_decreasing`, `strictly_increasing`.

implies: `alldifferent_except_0`, `not_all_equal`.

negation: `some_equal`.

- Defining constraints
- Different views attached to a constraint
- Describing constraints
- **Dedicated description languages**
- Around the catalog
- Learning constraints models

Currently 3 dedicated languages

- Describing constraints as graph properties
  (*search for a graph with certain characteristics*)

- Describing constraints by an automaton
  (*set of solutions equivalent to words accepted by the automaton*)

- Describing constraint as 1 order formulae
  (*restricted language for geometrical constraints: quantification over space dimensions and objects*)

Graph based description

nvalue(NVAL, VARIABLES)

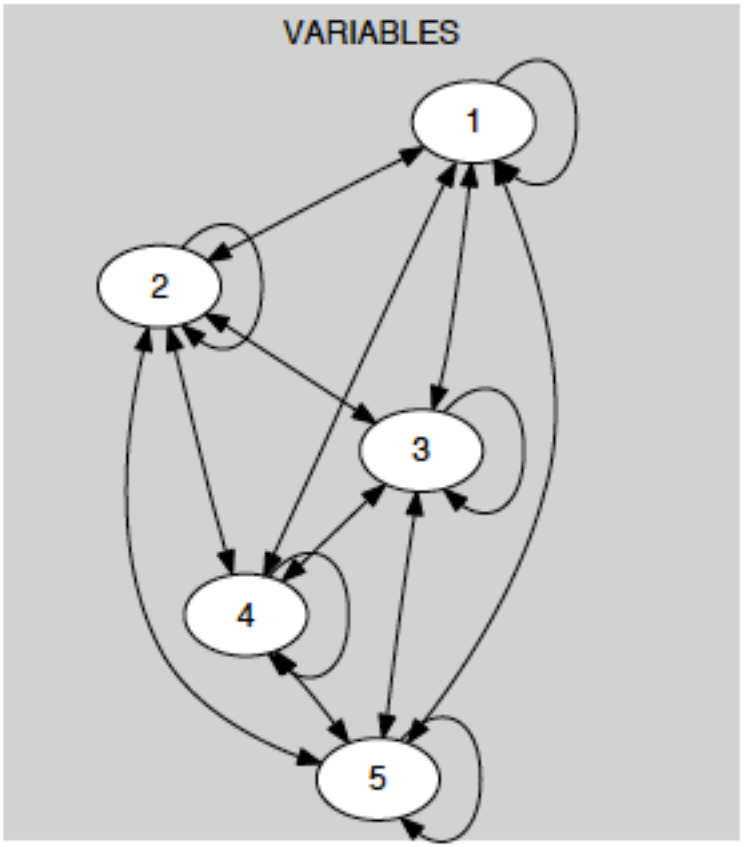
NVAL	:	dvar
VARIABLES	:	collection(var−dvar)

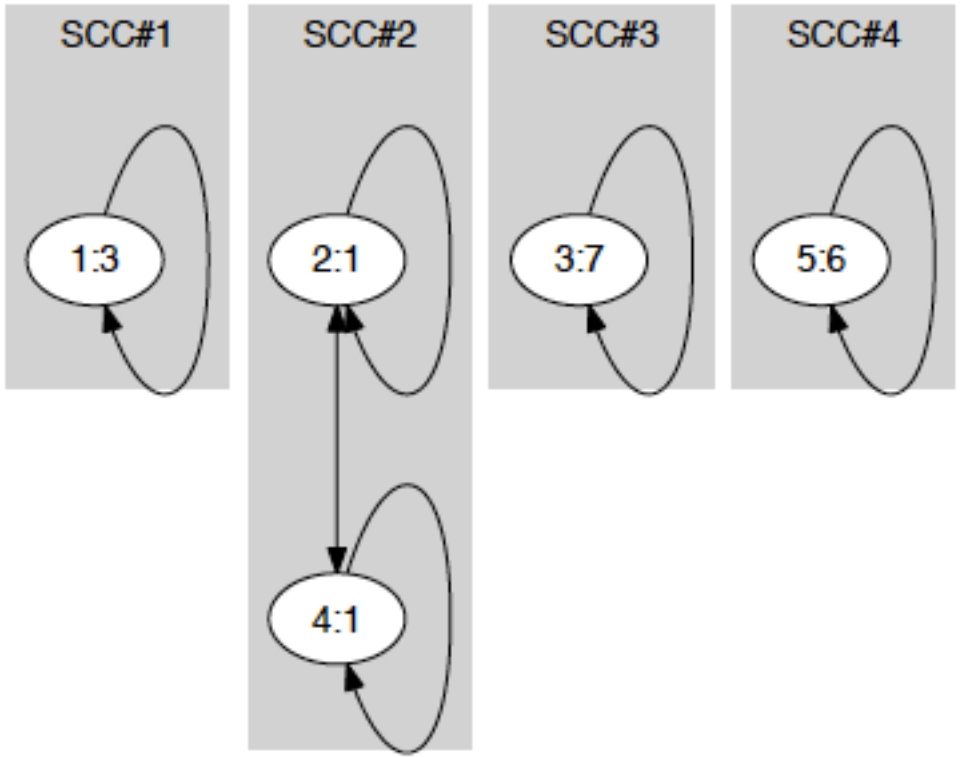
NVAL is the number of distinct values taken by the variables of the collection VARIABLES.

Arc input(s)	VARIABLES
Arc generator	$CLIQUE \mapsto$ collection(variables1, variables2)
Arc arity	2
Arc constraint(s)	variables1.var = variables2.var
Graph property(ies)	NSCC= NVAL

Arc input(s)	VARIABLES
Arc generator	$CLIQUE \mapsto$ collection(variables1, variables2)
Arc arity	2
Arc constraint(s)	variables1.var = variables2.var
Graph property(ies)	NSCC = NVAL

$(4, \langle 3, 1, 7, 1, 6 \rangle)$





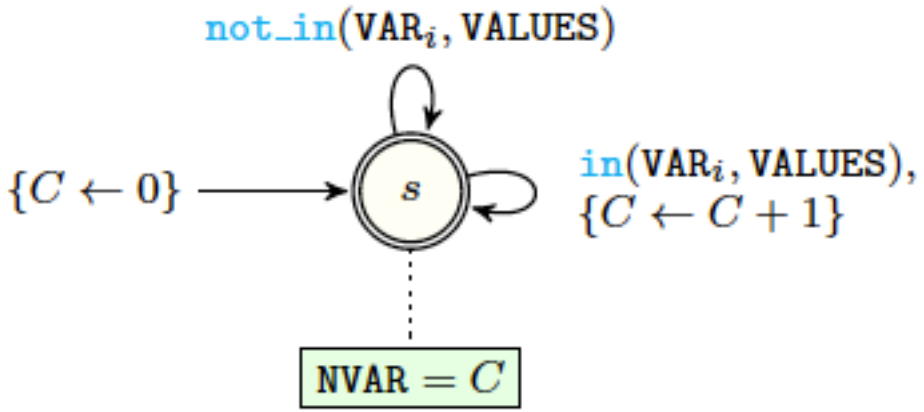
NSCC=4

Automaton based description

among(NVAR, VARIABLES, VALUES)

NVAR	:	dvar
VARIABLES	:	collection(var−dvar)
VALUES	:	collection(val−int)

NVAR is the number of variables of the collection VARIABLES that take their value in VALUES.

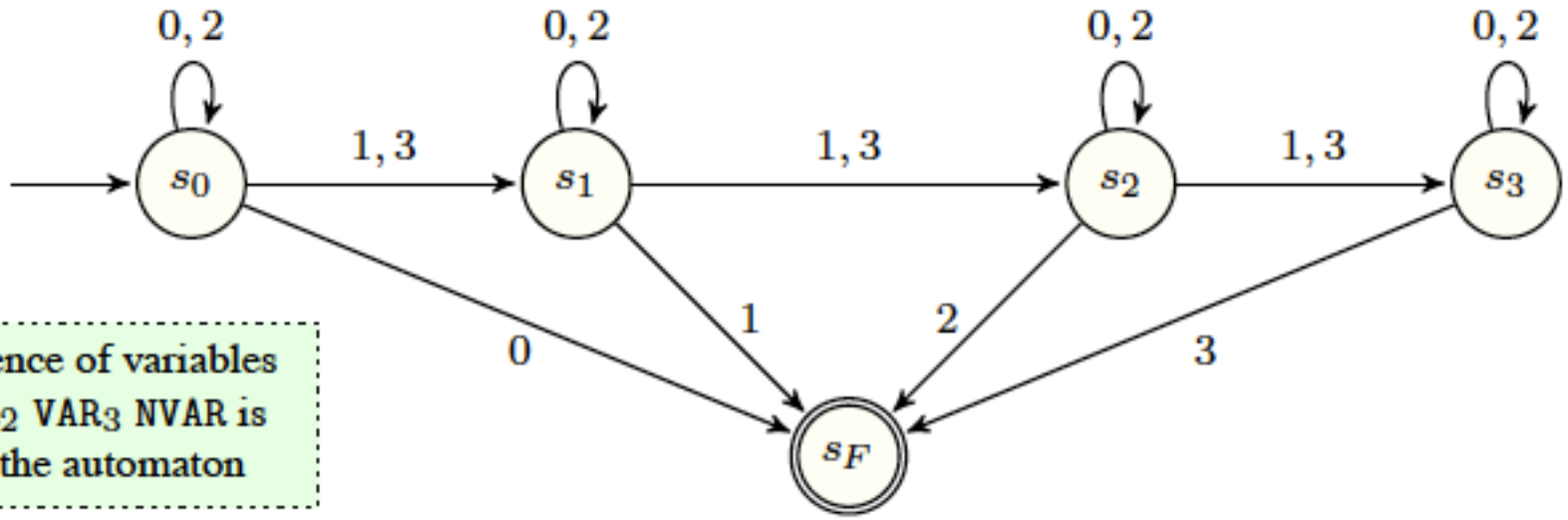


Automaton based description

among(NVAR, VARIABLES, VALUES)

```
NVAR      :  dvar
VARIABLES :  collection(var−dvar)
VALUES    :  collection(val−int)
```

NVAR is the number of variables of the collection VARIABLES that take their value in VALUES.



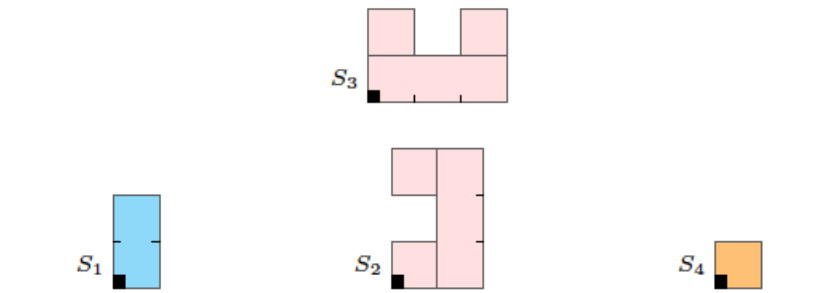
The sequence of variables VAR_1 VAR_2 VAR_3 NVAR is passed to the automaton

First order formulae based description

```
disjoint_sboxes(K, DIMS, OBJECTS, SBOXES)
```

```
K        :  int
DIMS     :  sint
OBJECTS  :  collection(oid−int, sid−dvar, x − VARIABLES)
SBOXES   :  collection(sid−int, t − INTEGERS, l − POSITIVES)
```

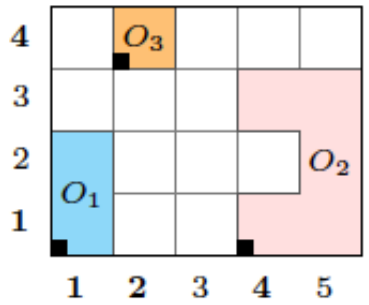
$$
\left(\begin{array}{l}
2, \{0,1\}, \\
\left\langle \begin{array}{llll} \texttt{oid}-1 & \texttt{sid}-1 & \texttt{x}-\langle 1,1\rangle, \\ \texttt{oid}-2 & \texttt{sid}-2 & \texttt{x}-\langle 4,1\rangle, \\ \texttt{oid}-3 & \texttt{sid}-4 & \texttt{x}-\langle 2,4\rangle \end{array} \right\rangle, \\
\left\langle \begin{array}{llll} \texttt{sid}-1 & \texttt{t}-\langle 0,0\rangle & \texttt{l}-\langle 1,2\rangle, \\ \texttt{sid}-2 & \texttt{t}-\langle 0,0\rangle & \texttt{l}-\langle 1,1\rangle, \\ \texttt{sid}-2 & \texttt{t}-\langle 1,0\rangle & \texttt{l}-\langle 1,3\rangle, \\ \texttt{sid}-2 & \texttt{t}-\langle 0,2\rangle & \texttt{l}-\langle 1,1\rangle, \\ \texttt{sid}-3 & \texttt{t}-\langle 0,0\rangle & \texttt{l}-\langle 3,1\rangle, \\ \texttt{sid}-3 & \texttt{t}-\langle 0,1\rangle & \texttt{l}-\langle 1,1\rangle, \\ \texttt{sid}-3 & \texttt{t}-\langle 2,1\rangle & \texttt{l}-\langle 1,1\rangle, \\ \texttt{sid}-4 & \texttt{t}-\langle 0,0\rangle & \texttt{l}-\langle 1,1\rangle \end{array} \right\rangle
\end{array}\right)
$$



(A) Shape of the first object

(B) Shapes of the second object

(C) Shape of the third object



First order formulae for disjoint_sboxes

- $\texttt{origin}(\texttt{O1}, \texttt{S1}, \texttt{D}) \stackrel{\text{def}}{=} \texttt{O1.x}(\texttt{D}) + \texttt{S1.t}(\texttt{D})$
- $\texttt{end}(\texttt{O1}, \texttt{S1}, \texttt{D}) \stackrel{\text{def}}{=} \texttt{O1.x}(\texttt{D}) + \texttt{S1.t}(\texttt{D}) + \texttt{S1.l}(\texttt{D})$
- $\texttt{disjoint\_sboxes}(\texttt{Dims}, \texttt{O1}, \texttt{S1}, \texttt{O2}, \texttt{S2}) \stackrel{\text{def}}{=}$
  $\exists \texttt{D} \in \texttt{Dims}$
  $$\bigvee \begin{pmatrix} \texttt{origin}(\texttt{O1}, \texttt{S1}, \texttt{D}) > \\ \texttt{end}(\texttt{O2}, \texttt{S2}, \texttt{D}) \\ , \\ \texttt{origin}(\texttt{O2}, \texttt{S2}, \texttt{D}) > \\ \texttt{end}(\texttt{O1}, \texttt{S1}, \texttt{D}) \end{pmatrix}$$
- $\texttt{disjoint\_objects}(\texttt{Dims}, \texttt{O1}, \texttt{O2}) \stackrel{\text{def}}{=}$
  $\forall \texttt{S1} \in \texttt{sboxes}([\texttt{O1.sid}])$
  $\forall \texttt{S2} \in \texttt{sboxes}\left(\left[\texttt{O2.sid} \right] \right)$
  $$\texttt{disjoint\_sboxes} \begin{pmatrix} \texttt{Dims}, \\ \texttt{O1}, \\ \texttt{S1}, \\ \texttt{O2}, \\ \texttt{S2} \end{pmatrix}$$

- $\texttt{all\_disjoint}(\texttt{Dims}, \texttt{OIDS}) \stackrel{\text{def}}{=}$
  $\forall \texttt{O1} \in \texttt{objects}(\texttt{OIDS})$
  $\forall \texttt{O2} \in \texttt{objects}(\texttt{OIDS})$
  $\texttt{O1.oid} < \Rightarrow$
  $\texttt{O2.oid}$
  $$\texttt{disjoint\_objects} \begin{pmatrix} \texttt{Dims}, \\ \texttt{O1}, \\ \texttt{O2} \end{pmatrix}$$
- $\texttt{all\_disjoint}(\texttt{DIMENSIONS}, \texttt{OIDS})$

- Defining constraints
- Different views attached to a constraint
- Describing constraints
- Dedicated description languages
- **Around the catalog**
- Learning constraints models

The constraint catalog

- 2000: Classification in term of graph properties (60 ctr.)
- 2003: Introduction of meta data and generation of catalogue from the meta-data
- 2004: Description of constraints with automata
- 2005: Filtering wrt to graph properties + data base of graph formulae (120 formulae)
- 2008-2009: Description of geometrical constraints
- 2010-2012: More meta data in the context of learning
- 2001-2013: Update constraints (currently 420 ctr.)

Software around the catalog

- 2003-2013: meta-data, checkers and constraints
- 2003-2013: Generating the catalog from the meta-data
- 2010: On line constraint seeker, searching and ranking constraints (*search for constraint seeker*)
- 2011-2013: Model seeker, learning constraints models from samples (plan for a version accessible on-line)

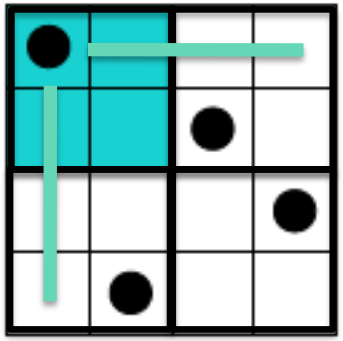
- Defining constraints
- Different views attached to a constraint
- Describing constraints
- Dedicated description languages
- Around the catalog
- **Learning constraints models**

Learning models: two cases

- Learning models for highly structured problems
- Learning models for semi-structured temporal series (supported by PGMO project)

De quoi s'agit t'il, un premier exemple

problème



C. Dürr (examen 2010 Polytechnique)

De l'exemple

3 0 2 1

APPRENTISSAGE

vers le modèle

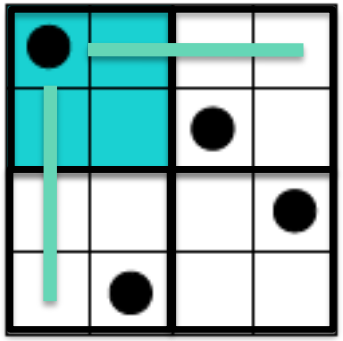
alldifferent(<V_1,V_2,V_3,V_4>)

alldifferent_interval(<V_1,V_2>, 2)

alldifferent_interval(<V_3,V_4>, 2)

De quoi s'agit t'il, un premier exemple

problème



C. Dürr (examen 2010 Polytechnique)

reformulation

$V_1=2*Q_1+R_1$ $(0 \leq R_1 <2)$
$V_2=2*Q_2+R_2$ $(0 \leq R_2 <2)$
alldifferent(<Q_1,Q_2>)

De l'exemple

3 0 2 1

APPRENTISSAGE

vers le modèle

alldifferent(<V_1,V_2,V_3,V_4>)

alldifferent_interval(<V_1,V_2>, 2)

alldifferent_interval(<V_3,V_4>, 2)

Un deuxième exemple

Description du problème (examen 2012 à l'EMN)

- Les élèves visitent un certain nombre d'entreprises sur plusieurs demi-journées consécutives.
- Chaque visite doit aborder un ensemble de thèmes.
- On répartit les thèmes entre les élèves lors de chacune des visites

Un deuxième exemple

Description du problème (examen 2012 à l'EMN)

- Les élèves visitent un certain nombre d'entreprises sur plusieurs demi-journées consécutives.
- Chaque visite doit aborder un ensemble de thèmes.
- On répartit les thèmes entre les élèves lors de chacune des visites

Règles à respecter

(A) Chaque élève doit se voir confier chacun des thèmes une fois et une seule.

(B) Deux élèves ne doivent pas être ensemble plus d'une fois.

(C) Lors de chaque visite, tous les thèmes doivent être couverts.

Un deuxième exemple

Description du problème (examen 2012 à l'EMN)

- Les élèves visitent un certain nombre d'entreprises sur plusieurs demi-journées consécutives.
- Chaque visite doit aborder un ensemble de thèmes.
- On répartit les thèmes entre les élèves lors de chacune des visites

Règles à respecter

(A) Chaque élève doit se voir confier chacun des thèmes une fois et une seule.

(B) Deux élèves ne doivent pas être ensemble plus d'une fois.

(C) Lors de chaque visite, tous les thèmes doivent être couverts.

Exemple de solution

0	1	2	3	4	5	6
0	2	6	1	3	4	5
1	0	2	4	6	3	5
1	2	3	6	0	5	4
2	3	4	5	6	0	1
2	0	6	3	5	1	4
3	1	4	2	5	6	0
3	4	0	5	1	2	6
4	5	1	6	3	2	0
6	5	3	0	2	4	1
5	6	1	4	2	0	3
4	6	5	0	1	3	2
5	3	0	1	4	6	2
6	4	5	2	0	1	3

Colonnes: demi-journées

Lignes: élèves

Valeurs: thèmes

Exemple (*suite et fin*)

Exemple(s) de solution

0	1	2	3	4	5	6	
0	2	6	1	3	4	5	
1	0	2	4	6	3	5	élèves
1	2	3	6	0	5	4	
2	3	4	5	6	0	1	
2	0	6	3	5	1	4	
3	1	4	2	5	6	0	
3	4	0	5	1	2	6	
4	5	1	6	3	2	0	
6	5	3	0	2	4	1	
5	6	1	4	2	0	3	
4	6	5	0	1	3	2	
5	3	0	1	4	6	2	
6	4	5	2	0	1	3	

demi-journées

Modèle appris

(A) Chaque élève traite une fois et une seule chaque thème

alldifferent_consecutive_values*14
alldifferent_from_at_least_k_pos(k=6)*1

(B) Deux élèves ne sont jamais plus d'une fois ensemble

0^{1}	1^{2}	2^{3}	3^{4}	4^{5}	5^{6}	6^{7}
0^{8}	2^{9}	6^{10}	1^{11}	3^{12}	4^{13}	5^{14}
1^{15}	0^{16}	2^{17}	4^{18}	6^{19}	3^{20}	5^{21}
1^{22}	2^{23}	3^{24}	6^{25}	0^{26}	5^{27}	4^{28}
2^{29}	3^{30}	4^{31}	5^{32}	6^{33}	0^{34}	1^{35}
2^{36}	0^{37}	6^{38}	3^{39}	5^{40}	1^{41}	4^{42}
3^{43}	1^{44}	4^{45}	2^{46}	5^{47}	6^{48}	0^{49}
3^{50}	4^{51}	0^{52}	5^{53}	1^{54}	2^{55}	6^{56}
4^{57}	5^{58}	1^{59}	6^{60}	3^{61}	2^{62}	0^{63}
6^{64}	5^{65}	3^{66}	0^{67}	2^{68}	4^{69}	1^{70}
5^{71}	6^{72}	1^{73}	4^{74}	2^{75}	0^{76}	3^{77}
4^{78}	6^{79}	5^{80}	0^{81}	1^{82}	3^{83}	2^{84}
5^{85}	3^{86}	0^{87}	1^{88}	4^{89}	6^{90}	2^{91}
6^{92}	4^{93}	5^{94}	2^{95}	0^{96}	1^{97}	3^{98}

(C) Chaque demi-journée on a 2 élèves sur chaque thème

global_cardinality([0..6-2])*7

0^{1}	1^{2}	2^{3}	3^{4}	4^{5}	5^{6}	6^{7}
0^{8}	2^{9}	6^{10}	1^{11}	3^{12}	4^{13}	5^{14}
1^{15}	0^{16}	2^{17}	4^{18}	6^{19}	3^{20}	5^{21}
1^{22}	2^{23}	3^{24}	6^{25}	0^{26}	5^{27}	4^{28}
2^{29}	3^{30}	4^{31}	5^{32}	6^{33}	0^{34}	1^{35}
2^{36}	0^{37}	6^{38}	3^{39}	5^{40}	1^{41}	4^{42}
3^{43}	1^{44}	4^{45}	2^{46}	5^{47}	6^{48}	0^{49}
3^{50}	4^{51}	0^{52}	5^{53}	1^{54}	2^{55}	6^{56}
4^{57}	5^{58}	1^{59}	6^{60}	3^{61}	2^{62}	0^{63}
6^{64}	5^{65}	3^{66}	0^{67}	2^{68}	4^{69}	1^{70}
5^{71}	6^{72}	1^{73}	4^{74}	2^{75}	0^{76}	3^{77}
4^{78}	6^{79}	5^{80}	0^{81}	1^{82}	3^{83}	2^{84}
5^{85}	3^{86}	0^{87}	1^{88}	4^{89}	6^{90}	2^{91}
6^{92}	4^{93}	5^{94}	2^{95}	0^{96}	1^{97}	3^{98}

En résumé

- **Apprend** un modèle à partir d'un **nombre réduit** d'exemples positifs (*de 1 à 3 exemples*)

- Un **modèle** est une **conjunction de conjunctions de contraintes identiques**

- Une **contrainte** est une **condition reliant des variables**

- On s'intéresse à des **problèmes structurés** (*i.e.* ***pouvant être décrit de manière concise***)

Construire un modèle est une tâche non triviale même pour des étudiants du domaine

Learning models for semi-structured temporal series

- Temporal series, electricity production profiles result from :

  - a demand (not completely structured) and
  - plant technological constraints (structured)

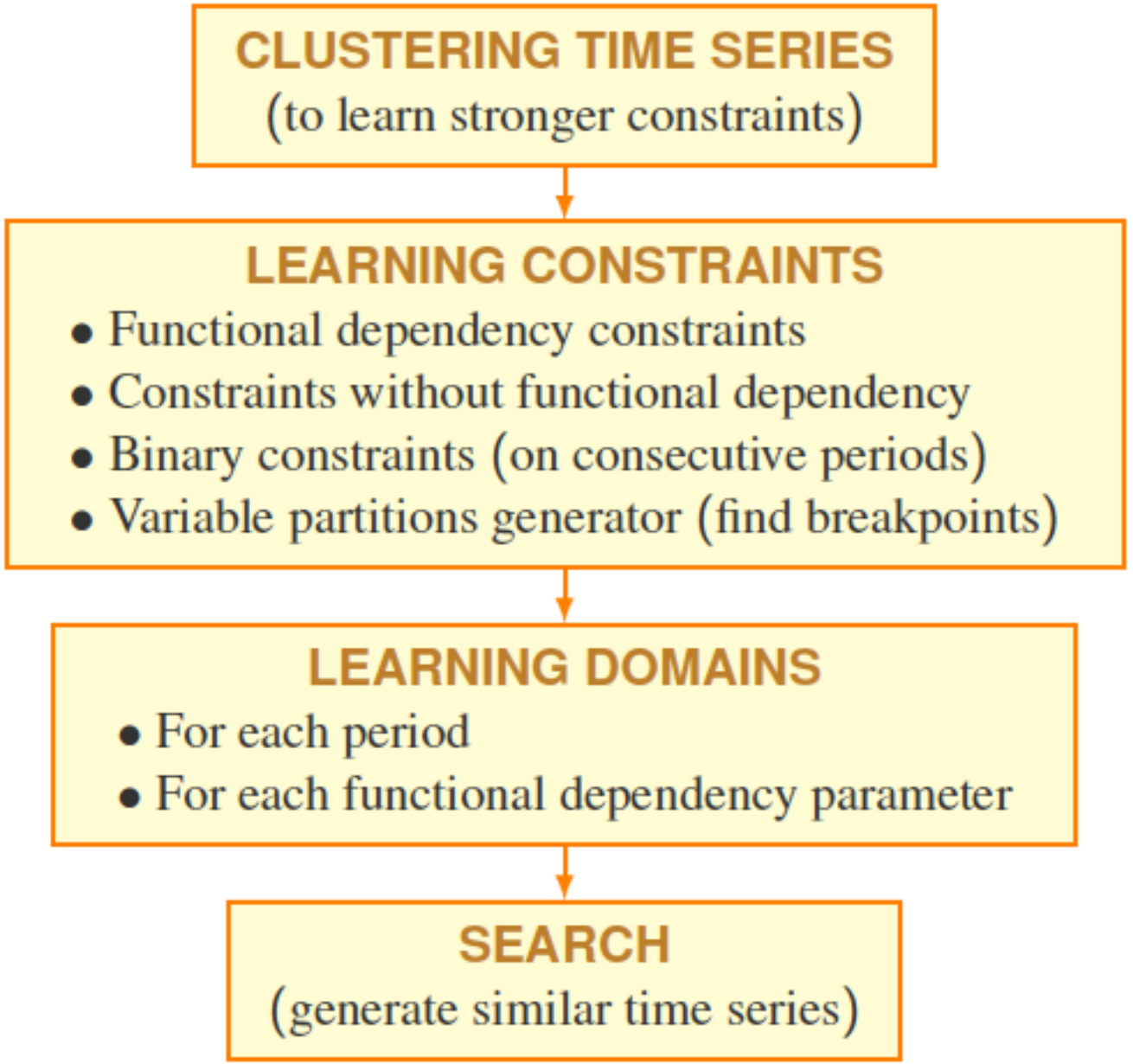
Overall Idea

- Apply ModelSeeker to problem from EDF
- Find constraint in Unit Commitment Problem
- Modelled with constraints having Functional Dependency
- Generate sample output similar to input data

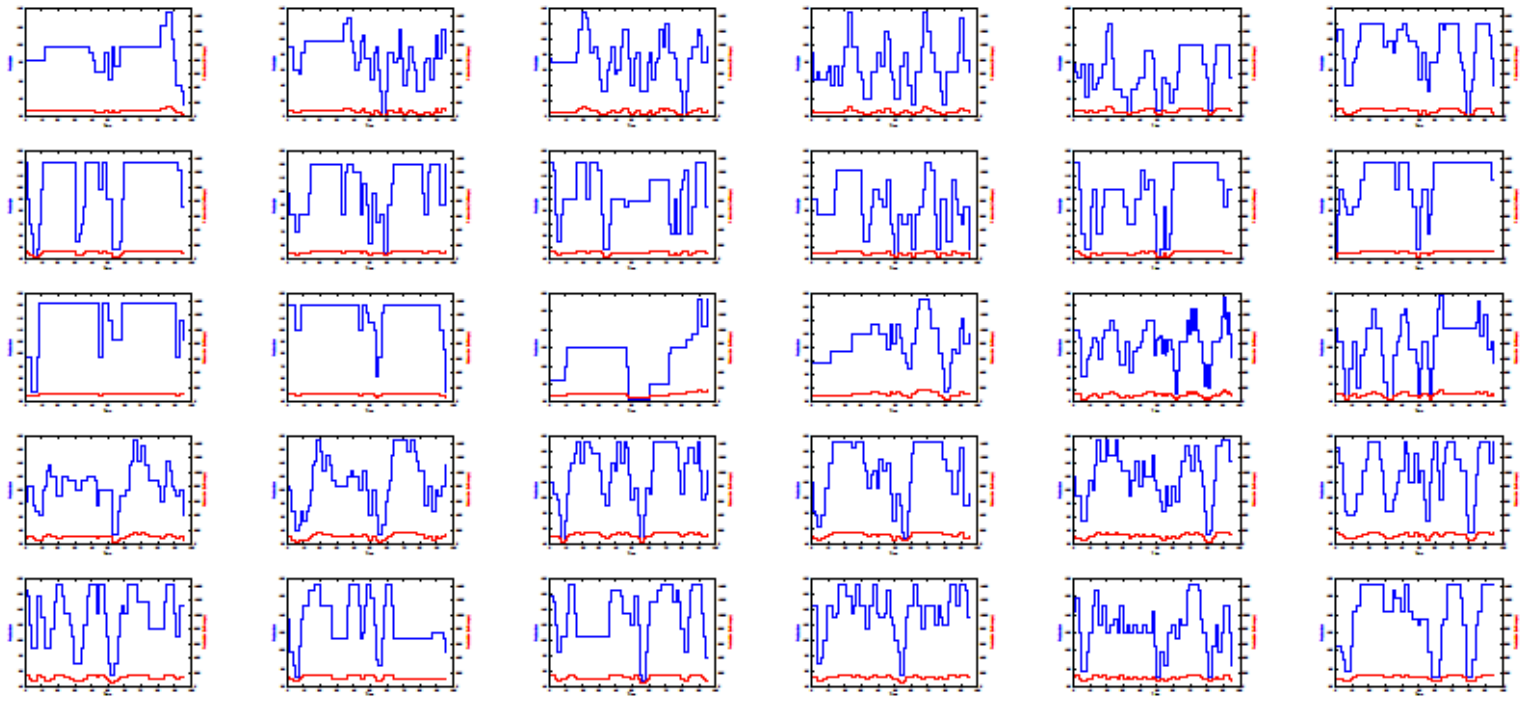
Problem: Identify Plant Specific Constraints

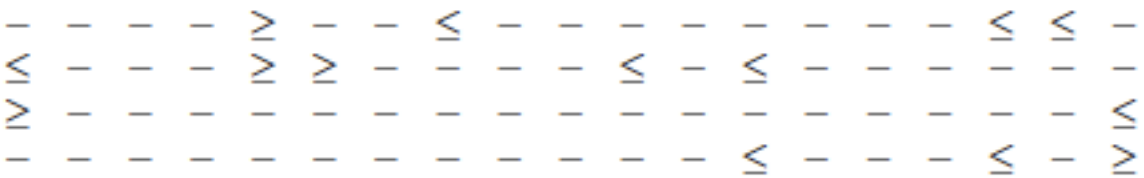
- Large part of model are plant specific constraints
- This determines how a plant can be scheduled
- Big differences between different types of plants (nuclear, thermal, hydro)
- Different parameter values for each plant
- There is a lot of data to analyze (766 000 x 3 times series)

Overview


CLUSTERING TIME SERIES
(to learn stronger constraints)
LEARNING CONSTRAINTS
• Functional dependency constraints
• Constraints without functional dependency
• Binary constraints (on consecutive periods)
• Variable partitions generator (find breakpoints)
LEARNING DOMAINS
• For each period
• For each functional dependency parameter
SEARCH
(generate similar time series)

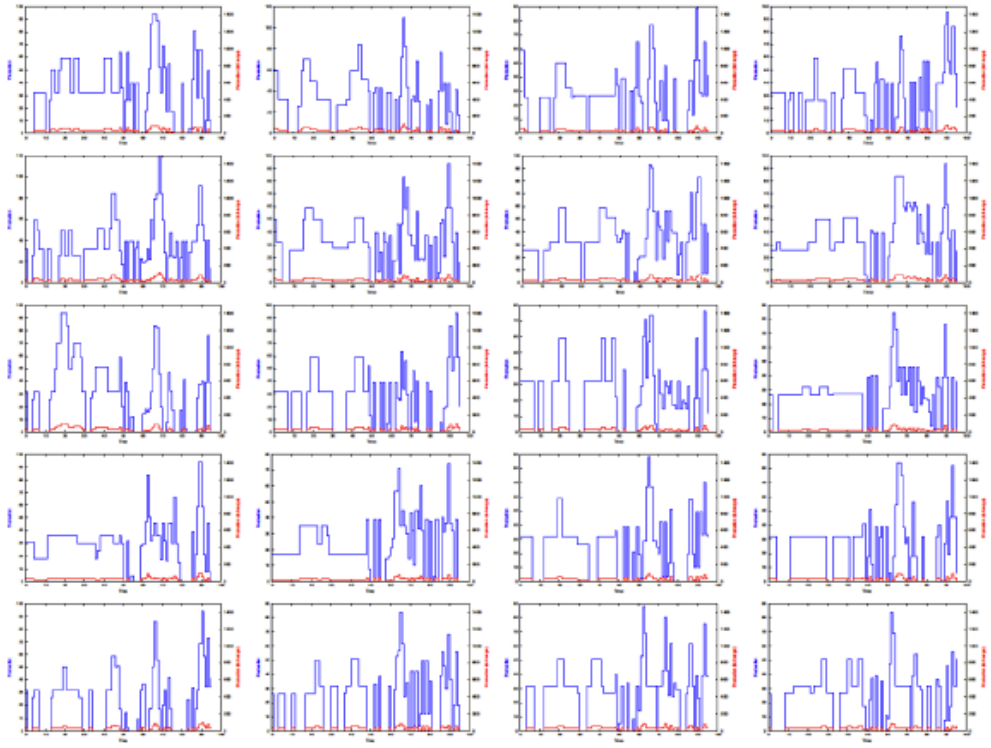
Binary ctr. learned on adjacent periods

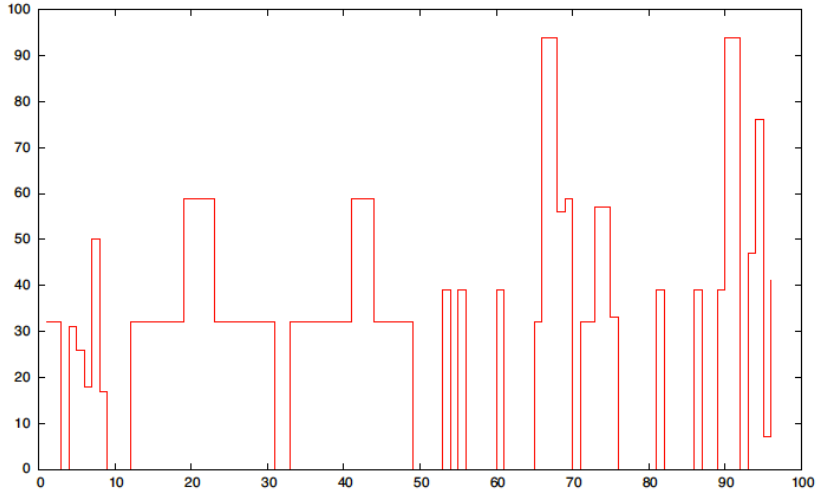




Constraint	Count	Percentage
none	5852	23.58
$=$	9326	37.21
$\geq$	4388	17.68
$\leq$	5129	20.66
$>$	45	0.18
$<$	59	0.24
$\neq$	21	0.08

Learned models for producing similar profiles





More information

- **Catalog**
  - http://soda.swedish-ict.se/5195/
  - http://link.springer.com/article/10.1007%2Fs10601-006-9010-8
  - http://www.emn.fr/z-info/sdemasse/gccat/
- **Constraint seeker**
  - http://link.springer.com/chapter/10.1007%2F978-3-642-23786-7_4
  - http://seeker.mines-nantes.fr/
- **Model seeker**
  - http://link.springer.com/chapter/10.1007%2F978-3-642-33558-7_13
  - http://4c.ucc.ie/~hsimonis/modelling/report.pdf