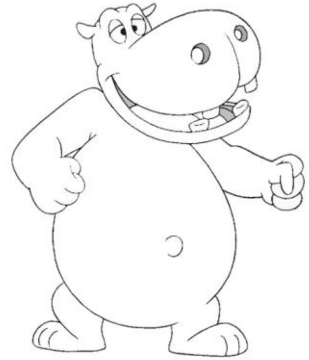


Filtering decomposable global cost functions

D. Allouche, C. Bessière, P. Boizumault, S. de Givry,
P. Gutierrez, S. Loudni, J-P. Métivier, **T. Schiex**

INRA-UBIA Toulouse, U. Montpellier,
GREYC-CNRS Caen, IIIA-CSIC Barcelone

ANR -10-BLA-0214

Optimizing in Cost Function Networks (aka CFN or WCSP)

(Shapiro, Haralick, IEEE PAMI 81)

- *n* variables
  - finite domains
- *e* scoped cost functions
  - scope, cost function
- Costs $\in$ {0,…,k}

$X = \{X_1, \ldots, X_n\}$

$X_i \in D_i, |D_i| \leq d$

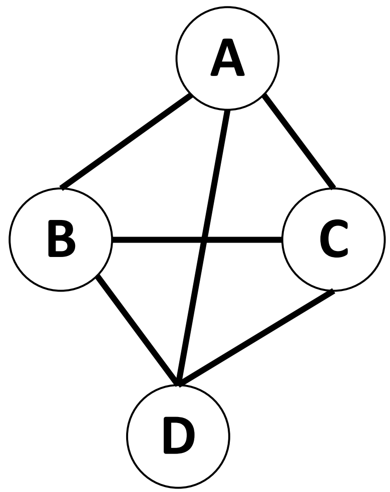
$F = \{f_1, \ldots, f_e\}$

k used for forbidden combinations

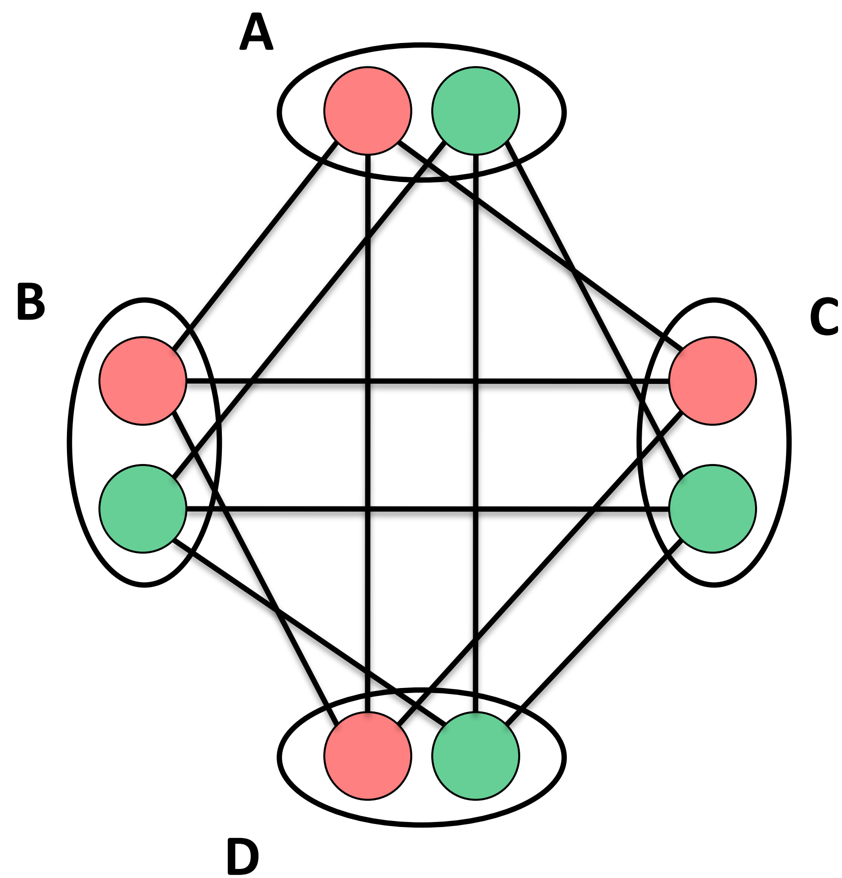
Minimize $\sum_F f_i(X)$

NP-hard

Example Min-2coloring



CFN graph
X={A,B,C,D},
F={f(A,B), f(A,C), f(A,D),
f(B,C), f(B,D), f(C,D)})



micro-structure
(each edge has cost 1)

Connections with MRF

- Cost functions are similar to energies
- Always positive (but wlog for optimization)

- CFN Inherits from Constraint networks
  - Emphasis on constraints (0/1 probabilities)
  - Optimization: tree search + local inference (filtering)
  - Global constraints

Filtering **Global cost functions**

Filtering a CFN

- Transforms a network into an equivalent network (same cost distribution).

Incremental.

- Using local transformations in the scope of one (arc consistency) cost function.

Efficient (bounded arity).

- Makes the network more « explicit » until a given property is satisfied.

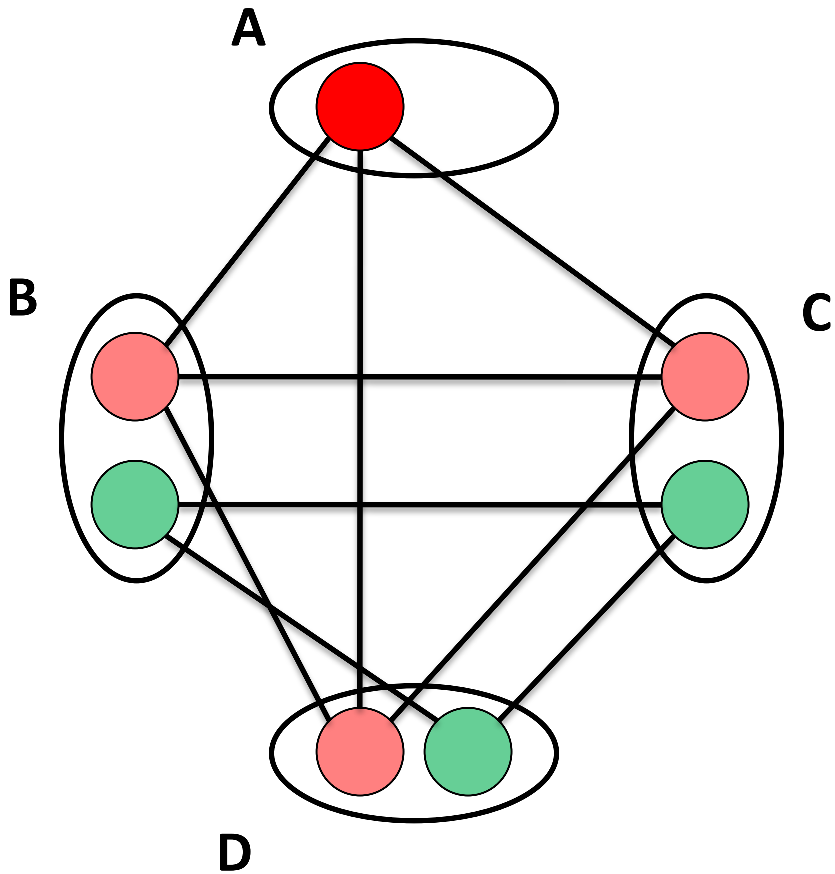
Well characterized (Converges)

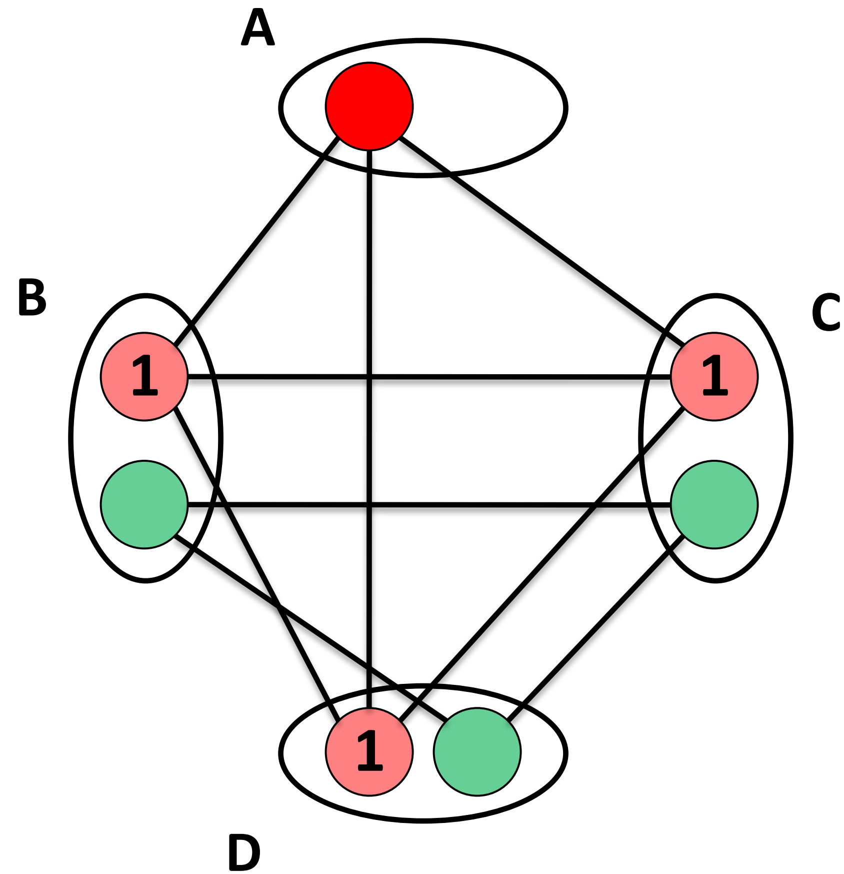
Filtering a CFN

- Applies equivalence preserving transformations (EPTs) to move costs to smaller arities :
  - $f(X_i)$, $\forall X_i \in X$
  - $f_\varnothing$ lower bound on the optimum cost

- Two families of algorithms
  - Chaotic EPTs applications: AC, DAC, *FDAC*, *EDAC*
  - Planified EPTs application: *OSAC*, *VAC*

Equivalence preserving transformation

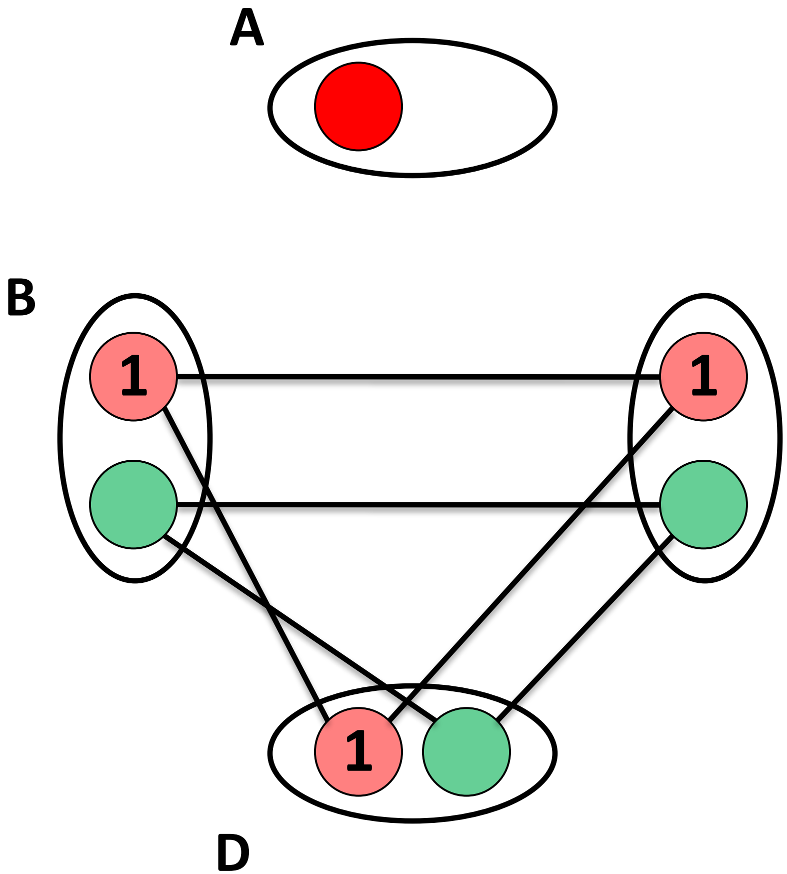




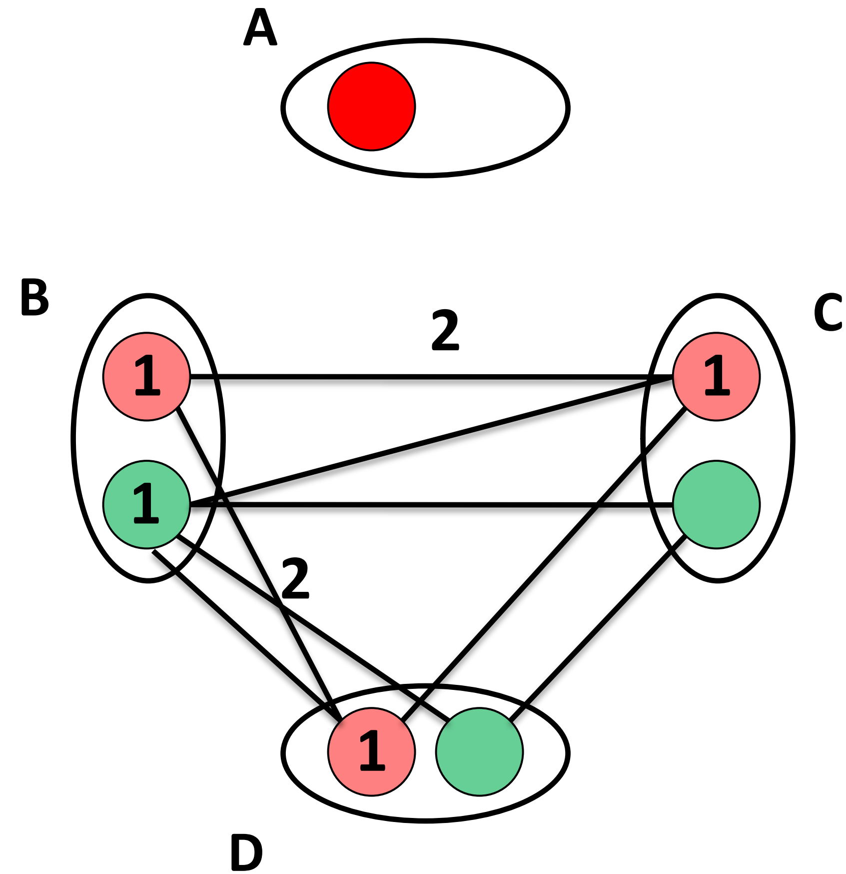
Arc consistent problem (AC)

(Schiex, CP 2000)

Chaotic application of EPTs



AC problem



$f_{\varnothing} = 1$ (Cooper, FSS 03)

Directional Arc Consistent problem (DAC)

$O(e\ d^r)$

A < B < C < D

Global constraints

- A constraint c(T) over any scope T
- No fixed arity
- Associated efficient« filtering » algorithms

`AllDifferent(X1,…Xm)`

captures permutations, assignment (Régin 1994)

Represented as a matching in a bipartite graph

Global cost functions

- A cost function f(T) over any scope T
- No fixed arity
- Associated efficient« filtering » algorithms

`SoftAllDifferent(X1,…Xm)`

(captures approximate permutations, assignment)

Represented as min cost flow in a transportation network

Filtering global cost functions

- Need to efficiently detect which costs can be moved to smaller scopes, preserving internal representation.
  - Monolithic approach
    - Uses flow based algorithms (softAllDifferent, softGCC, softRegular) (Lee, Leung, IJCAI 2009, AAAI 2010, JAIR 2012)
  - Decomposition based-approach
    - Rewrite the global cost function as a sum of smaller bounded scope cost functions (a sub network)

Decomposable cost function

A global cost function with a polynomial transformation δ_p

$$f(T) \xrightarrow{\delta_p} (T \cup E, F)$$ a cost function network

Such that

- $\forall f'(S) \in F,\ |S| \leq p$ arity bounded by p
- $\forall t \in D^T,\ f(t) = \min_{t' \in D^{T \cup E},\ t'[T]=t} \sum_{f'(S) \in F} f'(t'[S])$

Preserves marginal cost distribution

★ *Works also with constraints*

Relaxing decomposable global constraints

- Let c(T) $\Longrightarrow$ $(T \cup E, C)$ Constraint network

- Let g such that

  $\forall c'(S) \in C$, $\forall$ t' $(g \bullet c')(t') \leq c'(t')$ (constraint relaxation)

Then

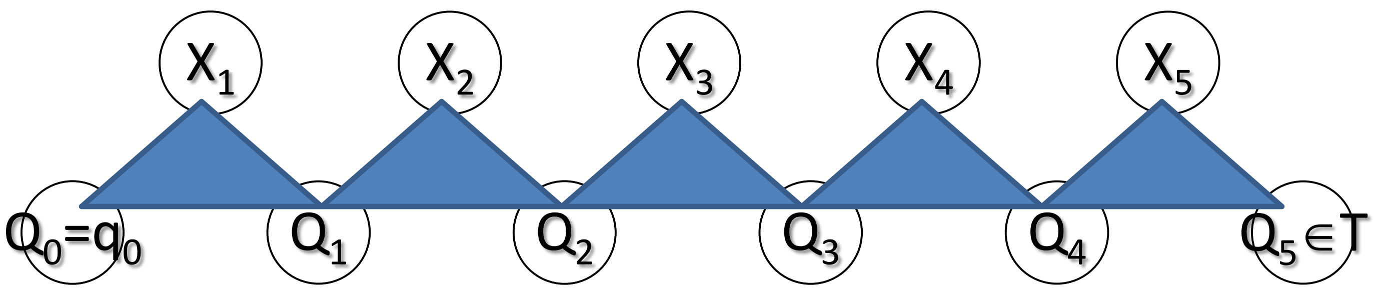
$(T \cup E, g \bullet C)$ is a decomposition of a global cost function which is a specific relaxation of c(T)

★ *Scopes are preserved by g => same hyper-graph*

Regular Global constraint

Regular(X_1, X_2, X_3, X_4, $X_{5,}$ (Q, Σ, β, q_0, S))

$$\Sigma = \text{U } D_{Xi}$$



$D_{Qi} = Q$

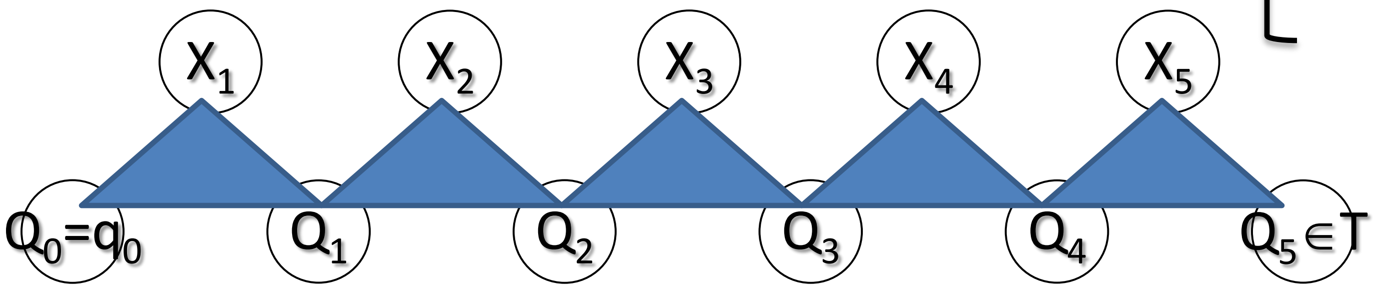
Polynomial transform (p=3) with extra variables (Q_i representing states)

➔ *AC filtering solves the decomposed Berge-acyclic network*

softRegular

softRegular(X_1, X_2, X_3, X_4, X_5, $(Q, \Sigma, \delta, q_0, T)$)

$$g\bullet\delta(Q_{i-1}, X_i, Q_i) = \begin{cases} 0 \text{ si } (q_{i-1}, x_i, q_i)\in\delta \\ 1 \text{ si } (q_{i-1}, v, q_i)\in\delta, v\neq x_i \\ +\infty \text{ sinon} \end{cases}$$

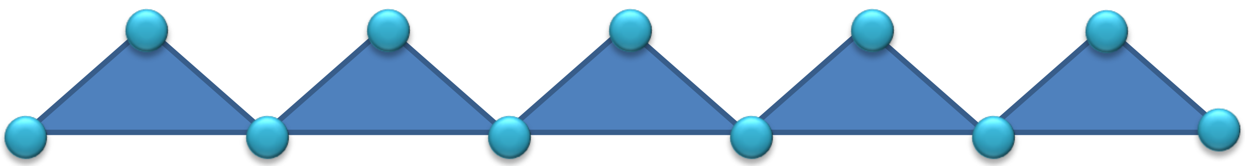


Polynomial transform (p=3) with extra variables (Q_i representing states) with Hamming distance relaxation.

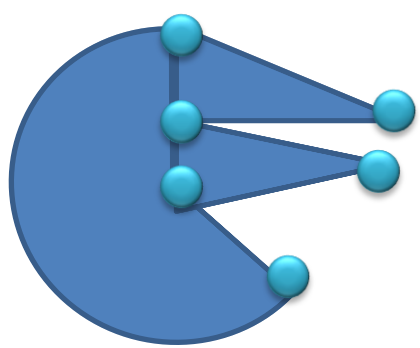
➔ *DAC filtering solves the decomposed Berge-acyclic network*

Berge-acyclic decomposition

- Example of Berge-acyclic decomposition



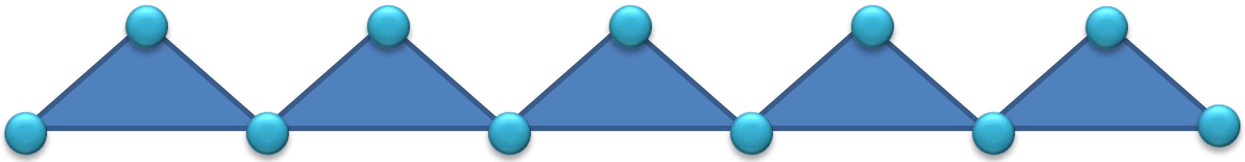
- Counter-example



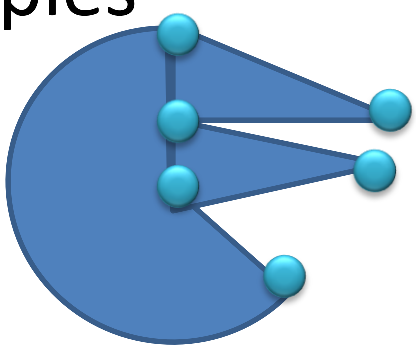
(X,F) Berge-acyclic iff the incidence graph $(X \cup F, E_F)$ acyclic $(\{X_i, f(T)\} \in E_F$ si $X_i \in T)$

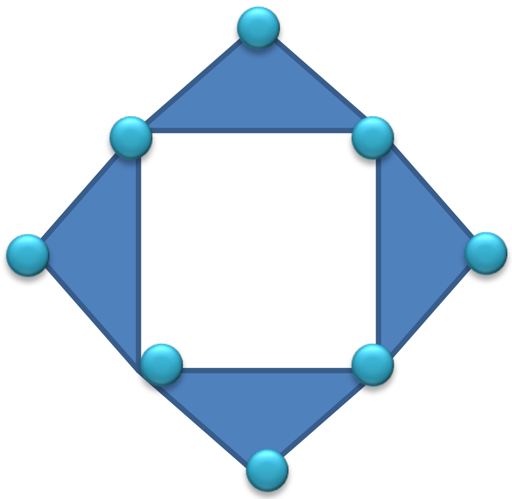
Berge-acyclic decomposition

- Example of Berge-acyclic decomposition



- Counter-examples



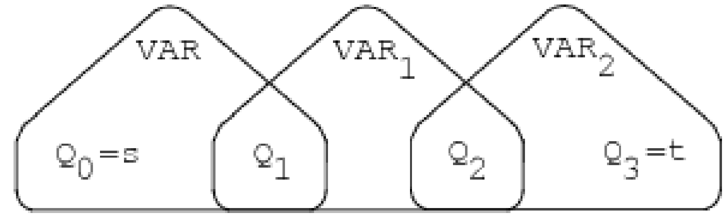


(X,F) Berge-acyclic iff the incidence graph $(X \cup F, E_F)$ acyclic $(\{X_i, f(T)\} \in E_F$ si $X_i \in T)$

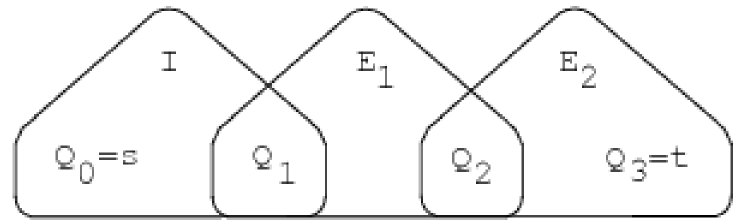
Berge-acyclic decomposable global constraints

max(VAR) :
max(VAR[i], max(VAR[i+1],…))

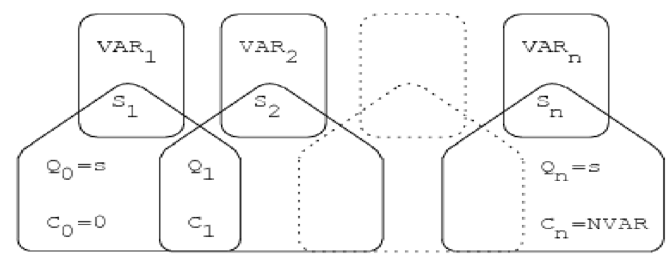
max, min, and, or, xor, …



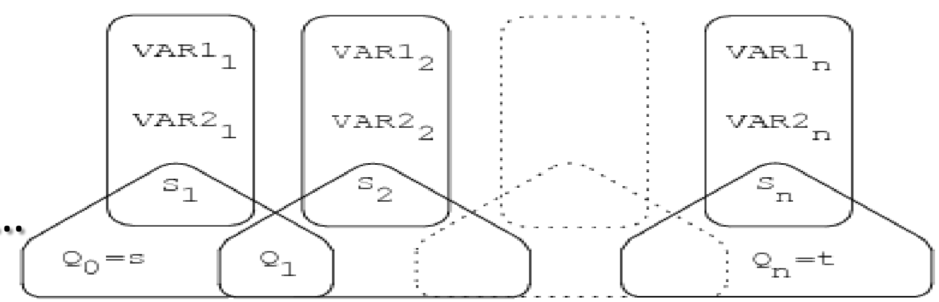
elementn(I, Table, E) :
E_1=Table[I], E_2=Table[I+1]…



among(NVAR, VAR, Valeurs) :
#{ i |VAR[i] $\in$ Valeurs } = NVAR



lex_less(VAR1, VAR2) :
VAR1[i] < VAR2[i] ou
VAR1[i]=VAR2[i] et VAR1[i+1]<VAR2[i+1] …



stretch, global_contiguity, …

**Global Constraint Catalog*

Berge-acyclic decomposition & directional arc consistency

- DAC solves Berge-acyclic CFN

$f(T) \Rightarrow (T \cup E, F)$ Berge-acyclic

$\exists$ order$(X_1,\ldots,X_m)$ on $T \cup E$ such that

$X_1 \in T$,

$f(X_1)$ after filtering by DAC$(T, f(T) \cup \{f(X_i) \mid X_i \in T\})$

$=$

$f(X_1)$ after filtering by DAC$(T \cup E, F \cup \{f(X_i) \mid X_i \in T\})$

★ *Extends to several decomposable global cost functions whose overall Decomposition is Berge-acyclic*

Berge-acyclic decomposition & virtual arc consistency

- VAC solves Berge-acyclic decompositions

$f(T) \Rightarrow (T \cup E, F)$ Berge-acyclic network

$f_{\varnothing}$ after filtering by $VAC(T, f(T) \cup \{f(X_i) \mid X_i \in T\})$

$=$

$f_{\varnothing}$ after filtering by $VAC(T \cup E, F \cup \{f(X_i) \mid X_i \in T\})$

★ *Extends to several decomposable global cost functions whose overall Decomposition is Berge-acyclic*

Experiments

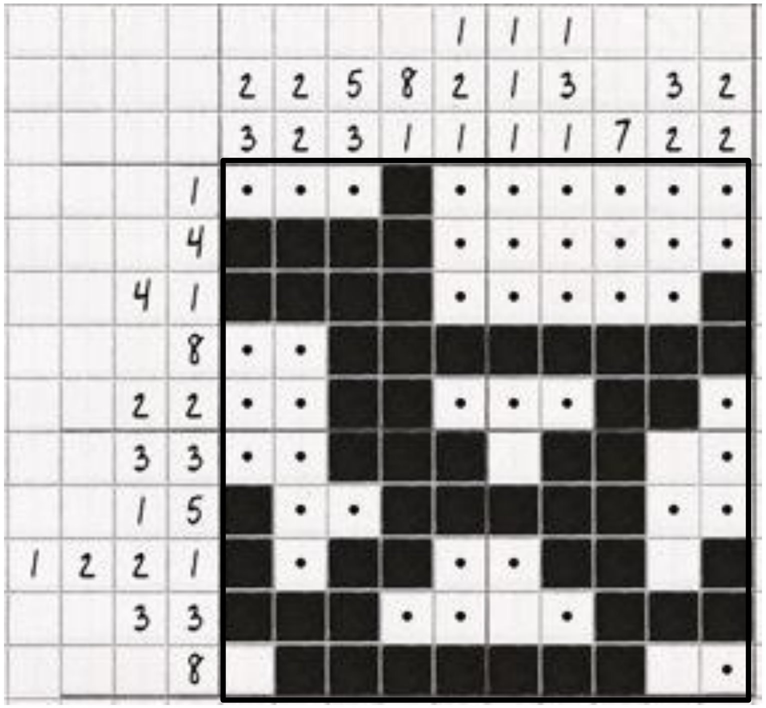
- Benchmarks
  - 1-softRegular (Pesant, CP 2004)
    - 30% of authorized transitions, 50% of terminal states
    - Random unary cost functions in [0,9]
  - Random nonograms (CSPLib #12)
    - relaxed (2n-softRegular) or with white noise (2n-Regular, scalar)
  - Knapsack with linear constraints (*Market Split*)
    - non polynomial decomposition (max. domain size<*1000)*
- Comparison of monolithic (flow based, EDGAC) vs. decomposed (EDAC) in toulbar2* solver.

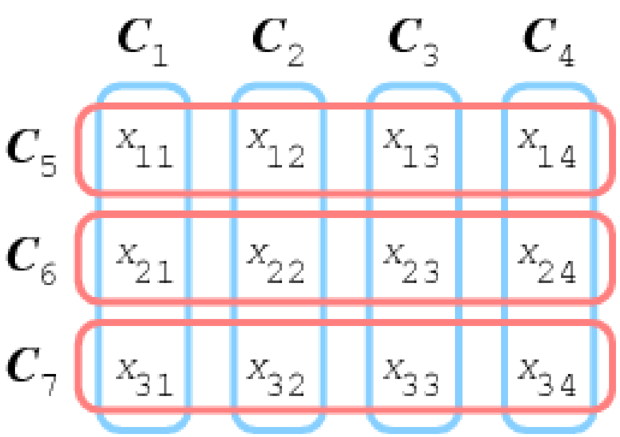
**toulbar2* version 0.9.5 https://mulcyber.toulouse.inra.fr/projects/toulbar2/*
With no preprocessing option and a static DAC compatible variable order

1-softRegular

n	$\|\Sigma\|$	$\|Q\|$	Monolithic		Decomposed	
			filter	solve	filter	solve
25	5	10	0.12	0.51	0.00	0.00
		80	2.03	9.10	0.08	0.08
25	10	10	0.64	2.56	0.01	0.01
		80	10.64	43.52	0.54	0.56
25	20	10	3.60	13.06	0.03	0.03
		80	45.94	177.5	1.51	1.55
50	5	10	0.45	3.54	0.00	0.00
		80	11.85	101.2	0.17	0.17
50	10	10	3.22	20.97	0.02	0.02
		80	51.07	380.5	1.27	1.31
50	20	10	15.91	100.7	0.06	0.07
		80	186.2	1,339	3.38	3.47

n × n Nonograms





- Each cell is black or white
- The lengths of successive black segments is fixed on every row and column (*NP-hard)*
- One boolean variable per cell
- Length specifications can be described by a regular language (□*■■□*■■■ □*)
- One **Regular** per row/column

★ *A DAC order compatible with all Berge-acyclic regulars exists*

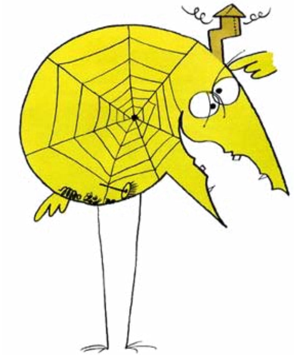
Relaxed Nonograms

2n-softRegular (*Hamming distance*)

Size	Monolithic		Decomposed	
	Solved	Time	Solved	Time
6×6	100%	1.98	100%	0.00
8×8	96%	358	100%	0.52
10×10	44%	2,941	100%	30.2
12×12	2%	3,556	82%	1,228
14×14	0%	3,600	14%	3,316

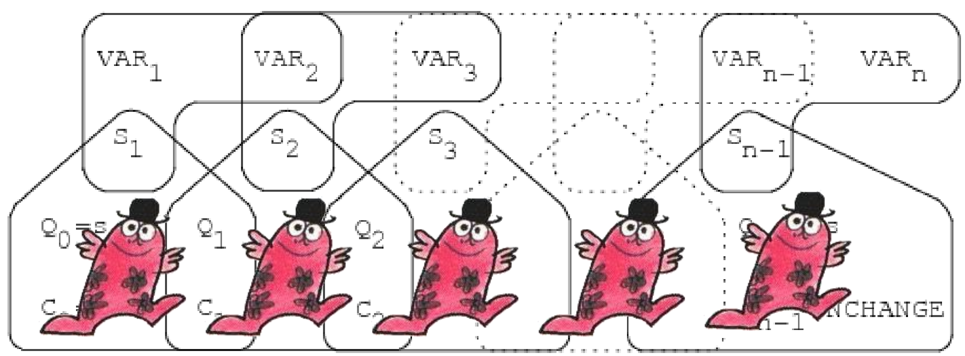
CPU limit one hour

Conclusion



- Relaxation of Berge-acyclic global constraints (SoftRegular, softAmong,…)
- DAC/VAC solves Berge-acyclic decompositions
- Incrementality for free, but additional variables
- Possible extension to other decompositions

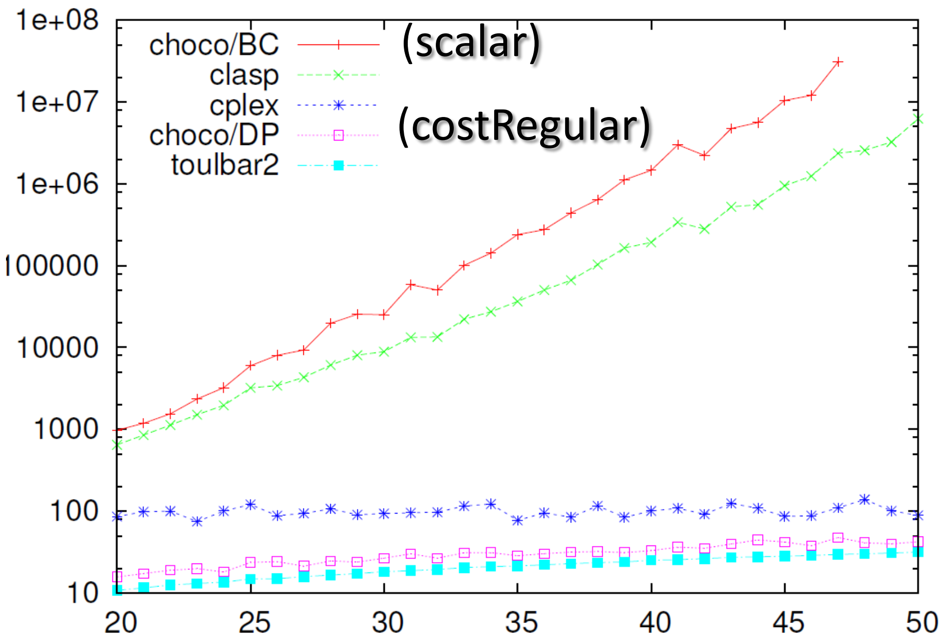
change(NVAR, VAR, CTR)



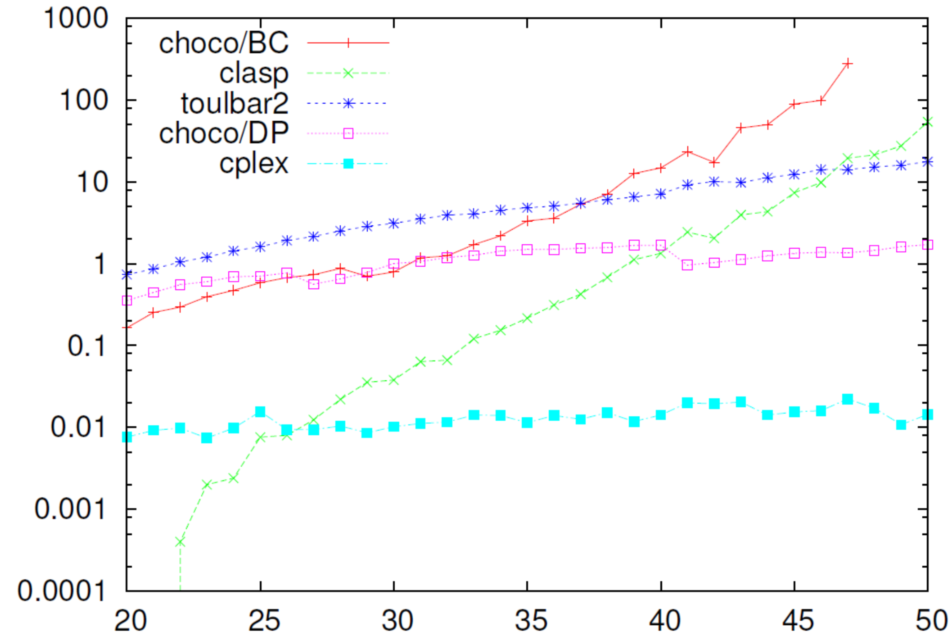
Market Split

1-linearEquation

Number of nodes



Cpu time (seconds)

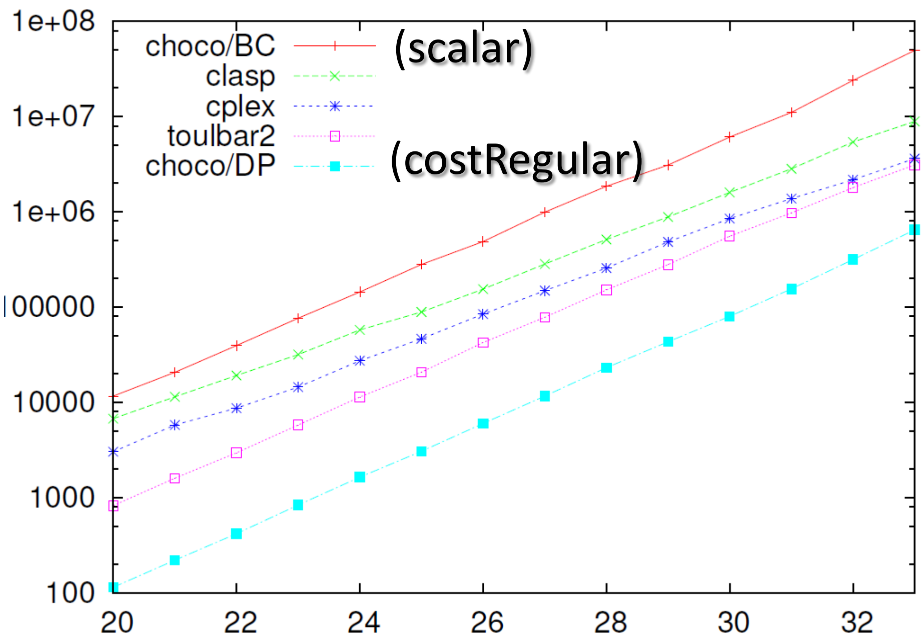


Number of boolean variables

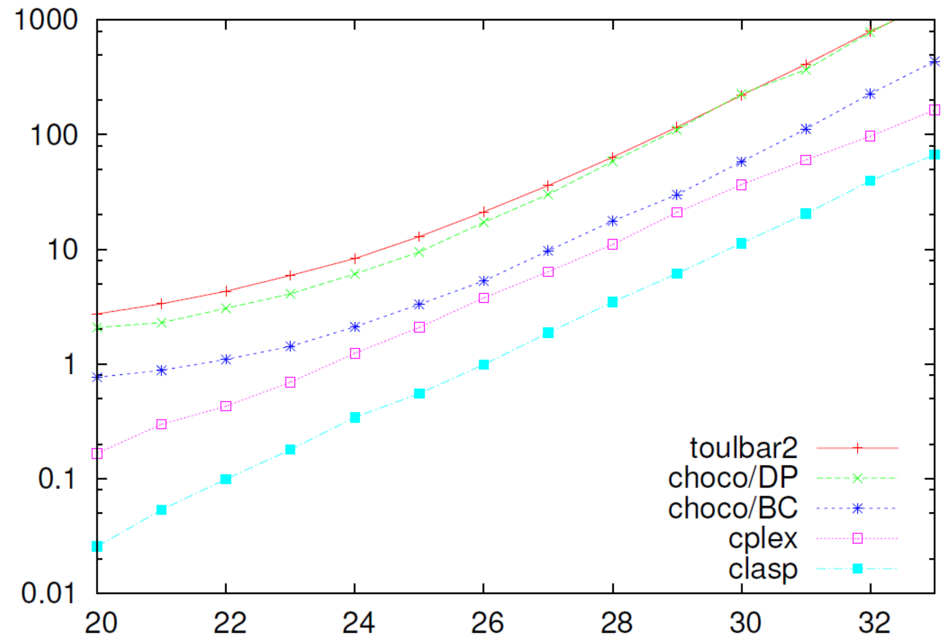
Market Split

4-linearEquation

Number of nodes

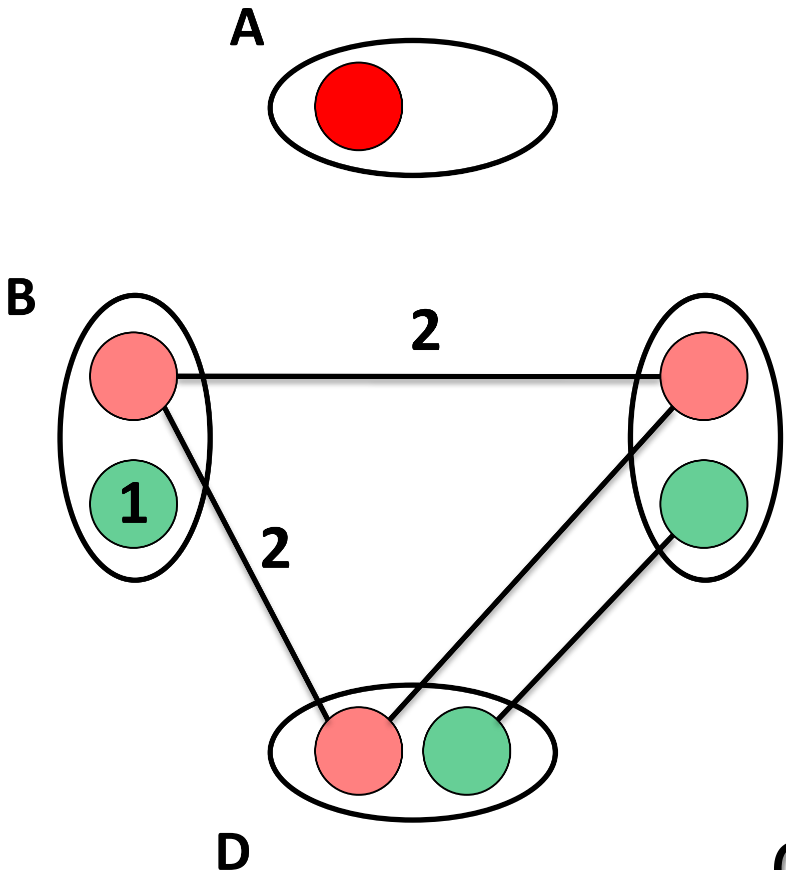


Cpu time (seconds)



Number of boolean variables

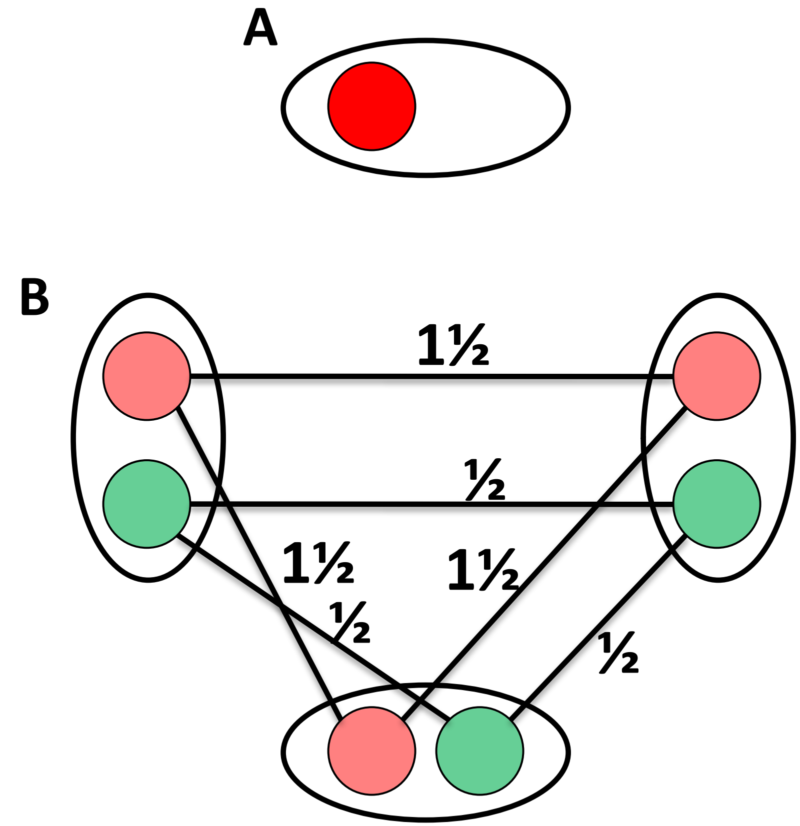
Planified reformulation



$f_\varnothing = 1$

EDAC problem

$O(e\ k\ d^r/\varepsilon)$



$f_\varnothing = 1\frac{1}{2}$

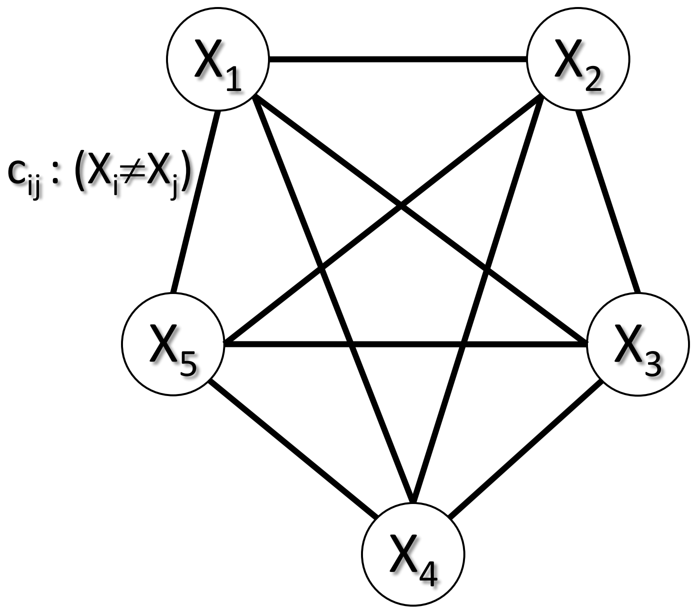
Virtual Arc Consistent (VAC)

(Cooper et al, AAAI 2008) (Cooper et al, AIJ 2010)

★ *Time exponential in the function arity*

softAllDifferent

softAllDiff(X_1, X_2, X_3, X_4, X_5)



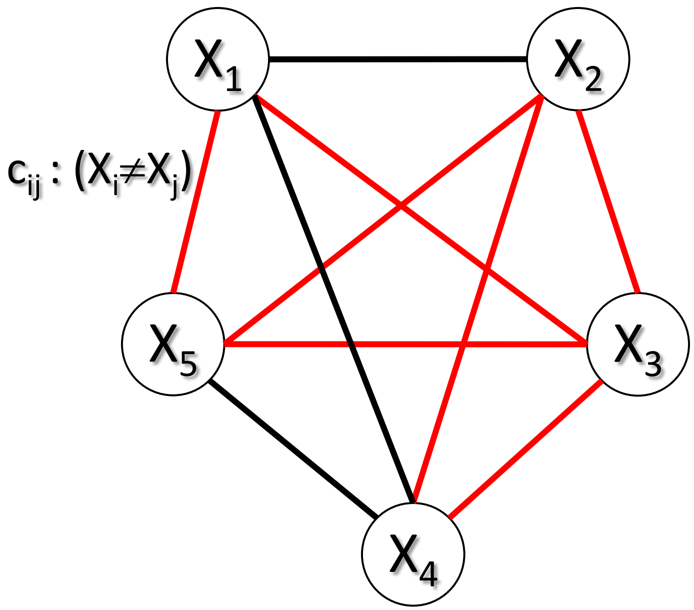
$$g \bullet c_{ij}(X_i, X_j) = \begin{cases} 0 \text{ si } X_i \neq X_j \\ 1 \text{ si } X_i = X_j \end{cases}$$

Polynomial transform(p=2) with a relaxation semantics defined by the number of pairs if variables with the same value.

★ *AC/DAC on the decomposed problem not equivalent to a direct filtering of the global cost function. Pol. Time using dedicated flow algorithms.*

soft AllDifferent

softAllDiff(X_1, X_2, X_3, X_4, X_5)



$$g \bullet c_{ij}(X_i, X_j) = \begin{cases} 0 & \text{if } X_i \neq X_j \\ & \text{or } (X_i, X_j) \notin G \\ 1 & \text{else} \end{cases}$$

★ *Finding the minimum of the cost function can be NP-hard Depending on g (graph coloring)*