Soft constraint processing

Thomas Schiex
INRA – Toulouse
France

Pedro Meseguer
CSIC – IIIA – Barcelona
Spain

Special thanks to:
Javier Larrosa
UPC – Barcelona
Spain

Overview

- **Introduction and definitions**
  - Why soft constraints and dedicated algorithms?
  - Generic and specific models
- **Solving soft constraint problems**
  - By search (systematic and local)
  - By complete inference... and search
  - By incomplete inference... and search
- **Existing implementations**

Why soft constraints?

- CSP framework: for *decision* problems
- Many problems are *optimization* problems

- Economics (combinatorial auctions)
  - Given a set G of goods and a set B of bids...
    - Bid (B_i , V_i), B_i requested goods, V_i value
  - ... find the best subset of compatible bids

  - Best = maximize revenue (sum)

Why soft constraints?

- Satellite scheduling
  - Spot 5 is an earth observation satellite
  - It has 3 on-board cameras
  - Given a set of requested pictures (of different importance)...
    - Resources, data-bus bandwidth, setup-times, orbiting
  - ...select a subset of compatible pictures with max. importance (sum)

Why soft constraints

- Multiple sequence alignment (DNA,AA)
  - Given *k* homologous sequences...
    - AATAATGTTATTGGTGGATCGATGA
    - ATGTTGTTCGCGAAGGATCGATAA
  - ... find the best alignment (sum)
    - AATAATGTTATTGGTG---GATCGATGATTA
    - ----ATGTTGTTCGCGAAGGATCGATAA---

Why soft constraints?

- Probabilistic inference (bayesian nets)
  - Given a probability distribution defined by a DAG of conditional probability tables
  - And some evidence
  - ...find the *most probable* explanation for the evidence (product)

Why soft constraints?

- Ressource allocation (frequency assignment)
  - Given a telecommunication network
  - ...find the best frequency for each communication link
  - Best can be:
    - Minimize the maximum frequency (max)
    - Minimize the global interference (sum)

Why soft constraints

- Even in decision problems, the user may have *preferences* among solutions.

- It happens in most real problems.

Experiment: give users a few solutions and they will find reasons to prefer some of them.

Overview

- Introduction and definitions
  - Why soft constraints and dedicated algorithms?
  - Generic and specific models
- Solving soft constraint problems
  - By search (systematic and local)
  - By complete inference... and search
  - By incomplete inference... and search
- Existing implementations

Soft constraints

- Two new difficulties:
  - How to express preferences in the model?
  - How to solve the resulting optimization problem?

Solving soft constraints as a CSP

- Using a global criteria constraint
  - New constraint S_k on all X
  - $S_k(X)$ = criteria value must be less than k
    - Number of variables having value *blue*

n-ary constraint
Inefficient!!

- Apply:

$k := n$
Repeat
    solve the original problem $\cup$ $S_k(X)$
    $k := k-1$;
Until solution

Combined local preferences

- Constraints are local *cost functions*
- *Costs* combine with a dedicated *operator*
  - *max*: priorities — Fuzzy/min-max CSP
  - +: additive costs — Weighted CSP
  - *: factorized probabilities... — Probabilistic CSP, BN

- Goal: find an assignment with the optimal combined cost

Soft constraint network

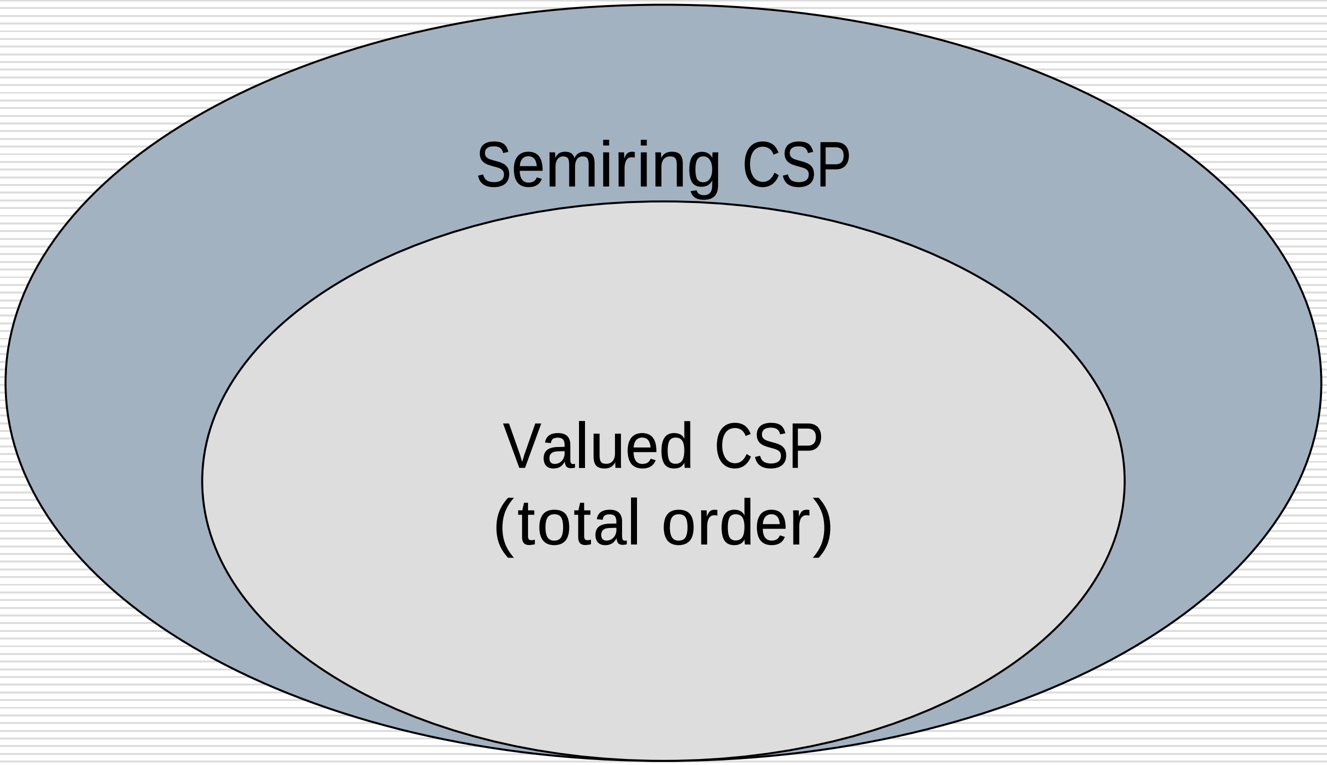
- (X,D,C)
  - $X=\{x_1,\ldots, x_n\}$ variables
  - $D=\{D_1,\ldots, D_n\}$ finite domains
  - $C=\{c_1,\ldots, c_e\}$ cost functions
    - $\mathrm{var}(c_i)$ scope
    - $c_i(t)$: $\rightarrow$ E (ordered cost domain, T, $\perp$)

annihilator

identity

- **Obj. Function**: $F(X)= \oplus c_i\,(X)$
  - commutative
  - associative
  - monotonic
- **Solution**: $F(t) \neq$ T
- **Soft CN**: find minimal cost solution

Valued CSP and Semiring CSP

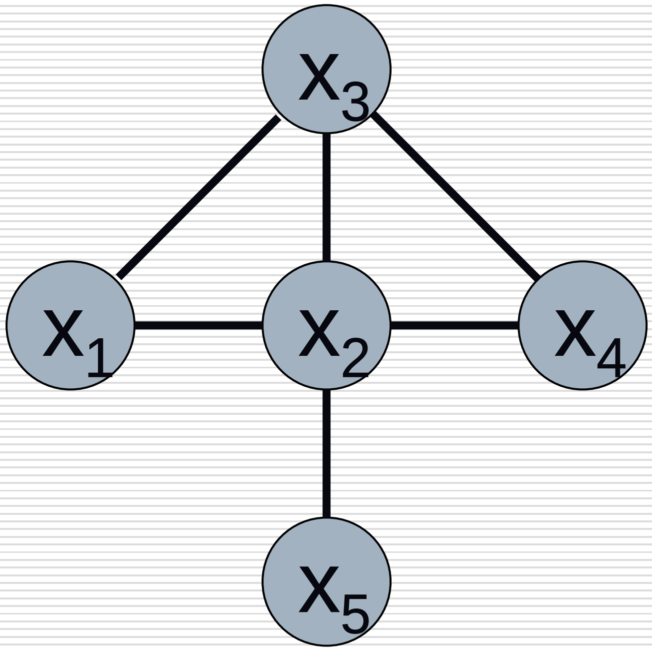


Specific frameworks

- $E = \{t, f\}$ $\oplus$ = and classical CSP
- $E = [0,1]$ $\oplus$ = max $\approx$ fuzzy CSP
- $E = \mathrm{N} \cup \{\infty\}$ $\oplus$ = + weighted CSP
- $E = [0,1]$ $\oplus$ = * bayesian net

Lexicographic CSP, probabilistic CSP...

Weighted CSP example ($\oplus$ = +)



For each vertex

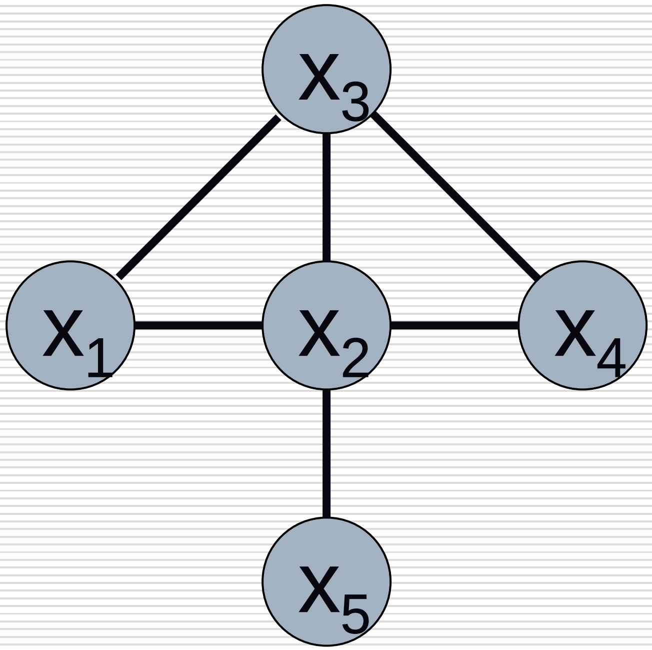
x_i	$c(x_i)$
b	0
g	1
r	1

For each edge:

x_i	x_j	$c(x_i,x_j)$
b	b	**T**
b	g	0
b	r	0
g	b	0
g	g	**T**
g	r	0
r	b	0
r	g	0
r	r	**T**

F(*X*): number of non blue vertices

Fuzzy constraint network (⊕=max)



For each vertex

x_i	$c(x_i)$
b	0.2
g	0.4
r	0.6
y	0.8

Connected vertices must have different colors

F(*X*): highest color used (b<g<r<y)

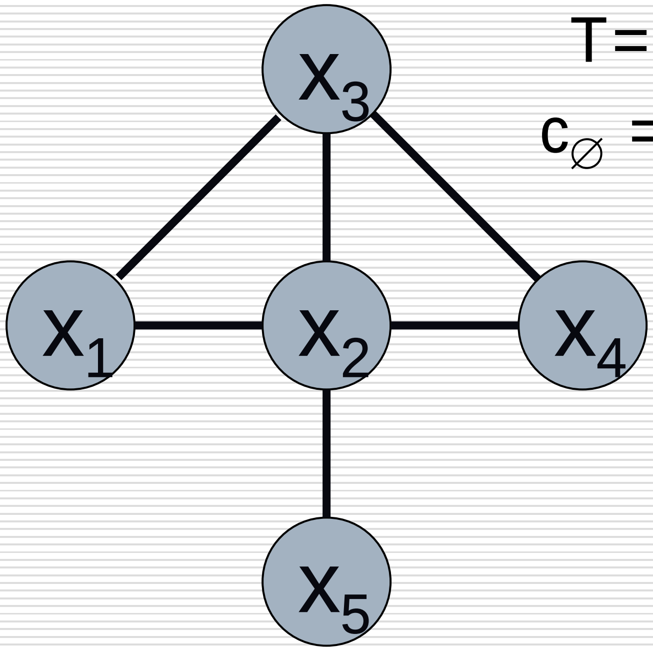
Some insight in unusual items

- T = maximum violation.
  - Can be set to a bounded max. violation k
  - Solution: $F(t) < k = Top$

- Empty scope soft constraint $c_\varnothing$ (a constant)
  - Gives an obvious lower bound on the optimum
  - If you do not like it: $c_\varnothing = \perp$

Additional expression power

Weighted CSP example (⊕ = +)



T=3

$c_\varnothing = 0$

For each vertex

x_i	$c(x_i)$
b	0
g	1
r	1

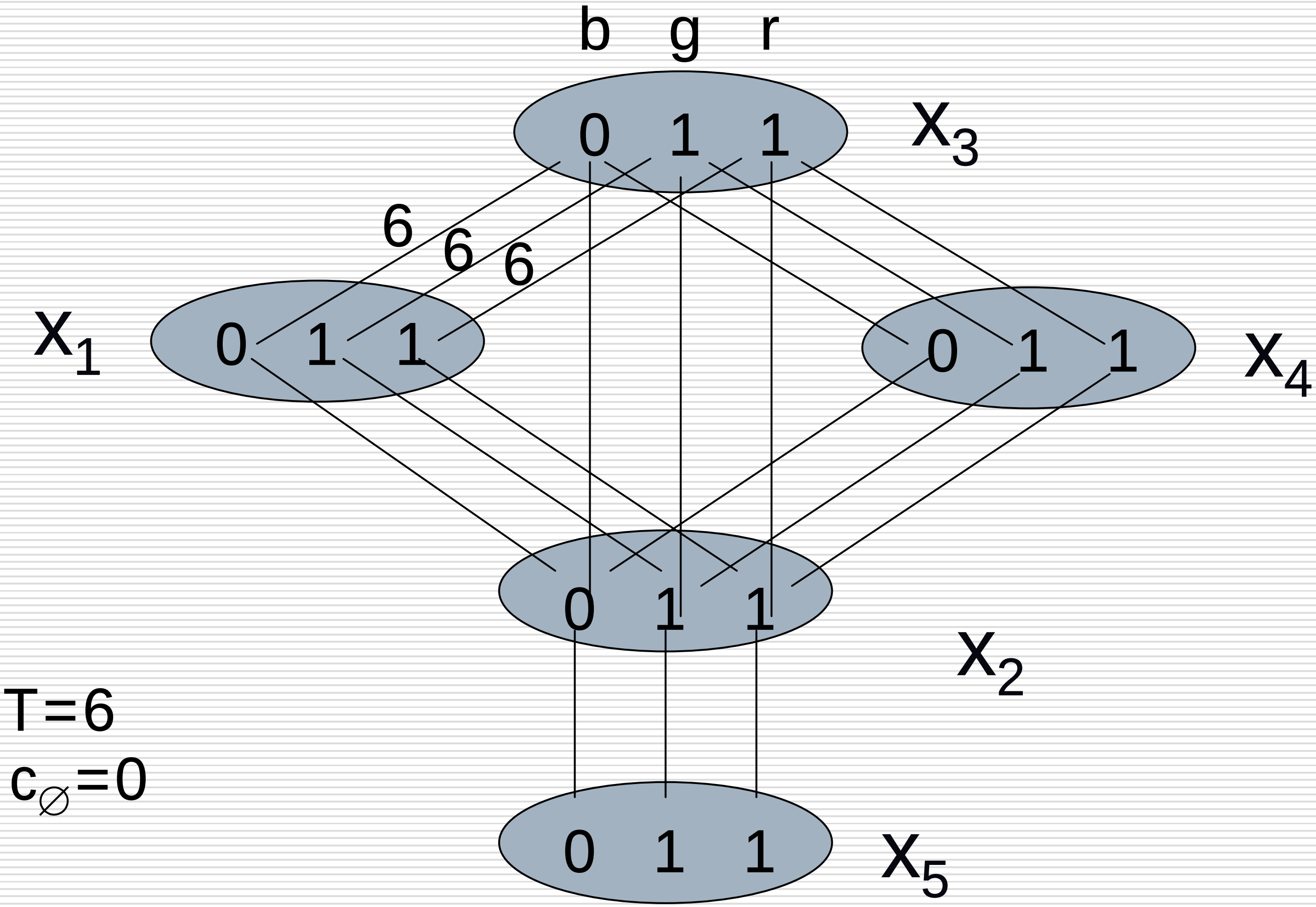
For each edge:

x_i	x_j	$c(x_i,x_j)$
b	b	**T**
b	g	0
b	r	0
g	b	0
g	g	**T**
g	r	0
r	b	0
r	g	0
r	r	**T**

F(*X*): number of non blue vertices

Optimal coloration with less then 3 non-blue

Microstructural graph



T=6

$c_\varnothing=0$

Basic operations on cost functions

I. Assignment (conditioning)

II. Combination (join)

III. Projection (elimination)

Assignment (conditioning)

x_i	x_j	$c(x_i,x_j)$
b	b	**T**=6
b	g	0
b	r	0
g	b	0
g	g	**T**=6
g	r	0
r	b	0
r	g	0
r	r	**T**=6

Assign($\mathbf{c}$,$\mathbf{x_j}$,b)

x_j	$f(x_j)$
b	**T**=6
g	0
r	0

Assign($\mathbf{f}$,$\mathbf{x_j}$,g)

$h_\varnothing$
0

Combination (join with ⊕)

x_i	x_j	$c(x_i,x_j)$
b	b	6
[illegible]	[illegible]	0
g	b	0
g	g	6

⊕

x_j	x_j	$c(x_j,x_k)$
b	b	6
b	g	0
g	b	0
[illegible]	[illegible]	6

x_i	x_j	x_k	$f(x_i,x_j,x_k)$
b	b	b	12
b	b	g	6
b	g	b	0
[illegible]	[illegible]	[illegible]	6
g	b	b	6
g	b	g	0
g	g	b	6
g	g	g	12

= 0 ⊕ 6

Projection (elimination)

x_i	x_j	$c(x_i,x_j)$
b	b	4
b	g	6
b	r	0
g	b	2
g	g	6
g	r	3
r	b	1
r	g	0
r	r	6

Min

Elim($\mathbf{c}$,$\mathbf{x_j}$)

x_i	$f(x_i)$
b	0
g	2
r	0

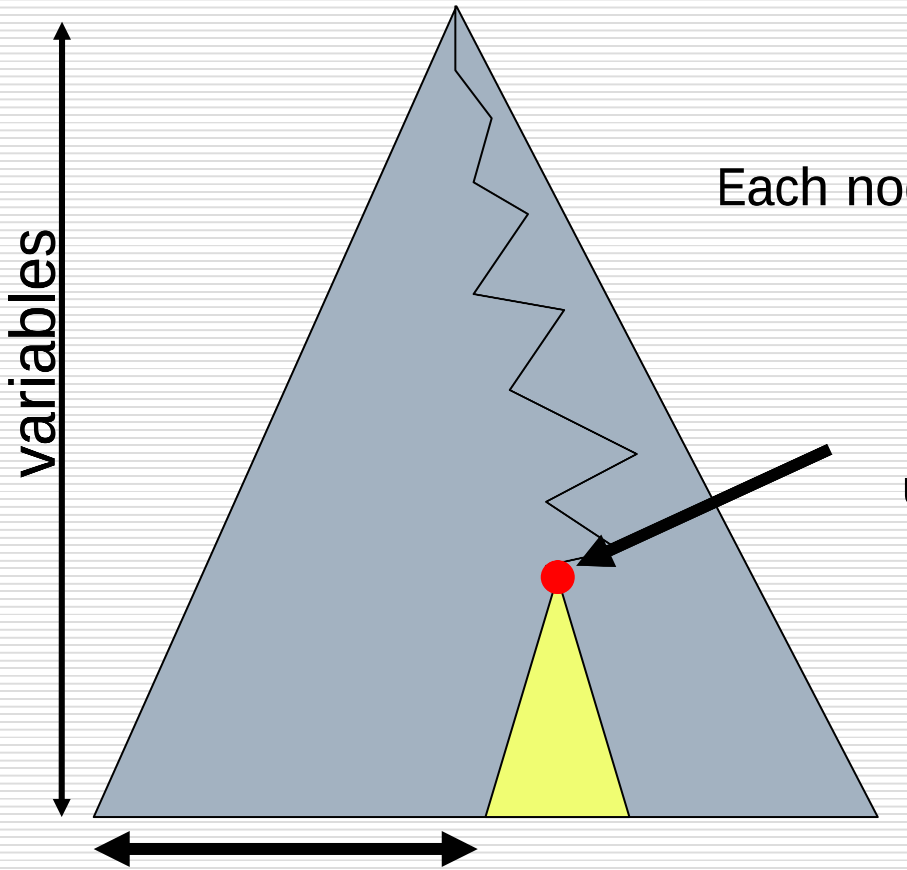
Elim($\mathbf{f}$,$\mathbf{x_j}$)

$h_{\varnothing}$
0

Overview

- Introduction and definitions
  - Why soft constraints and dedicated algorithms?
  - Generic and specific models
- **Solving soft constraint problems**
  - By search (systematic and local)
  - By complete inference... and search
  - By incomplete inference... and search
- Existing implementations

Systematic search



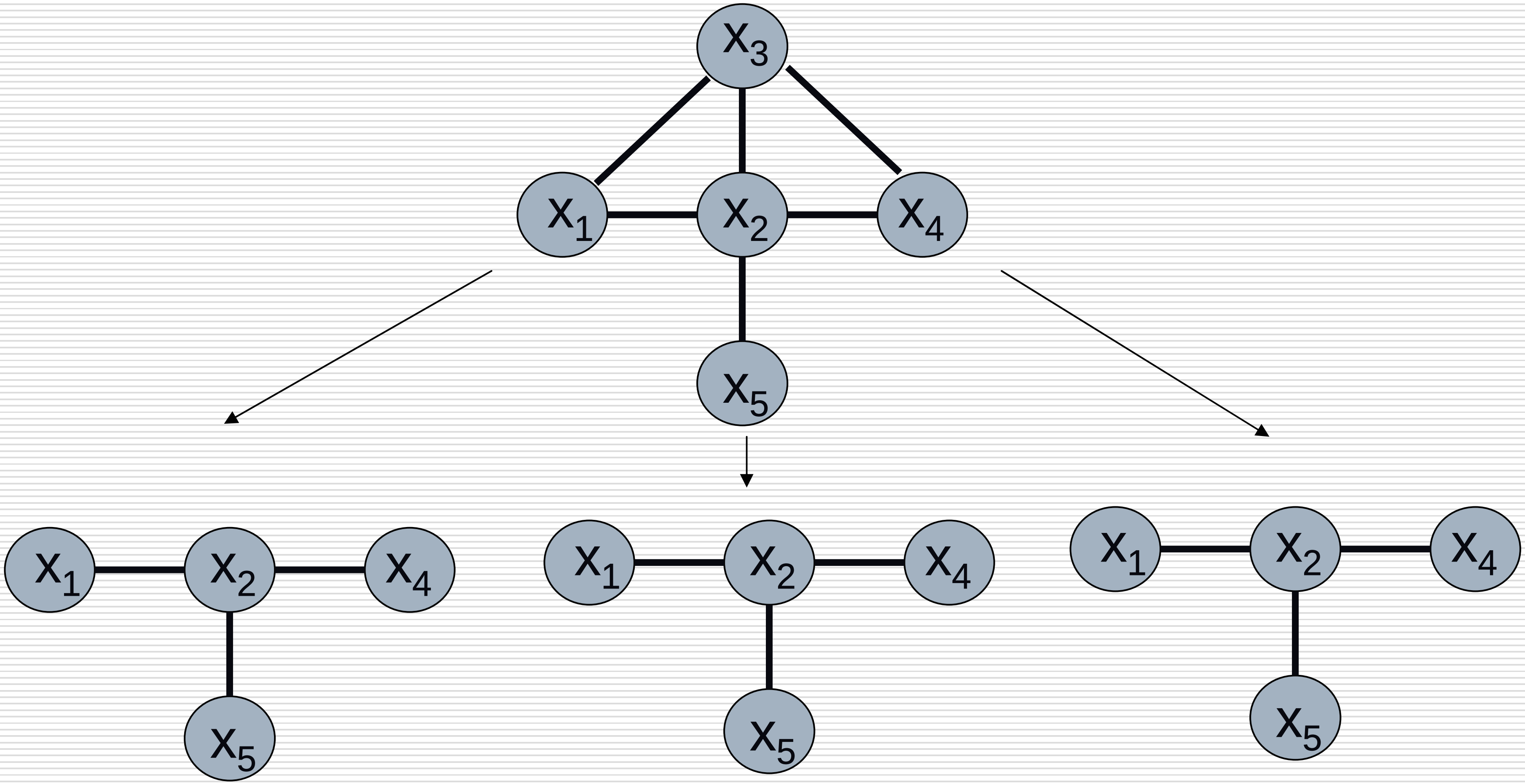
Each node is a soft constraint subproblem

(LB) Lower Bound = $c_{\varnothing}$
under estimation of the best
solution in the sub-tree

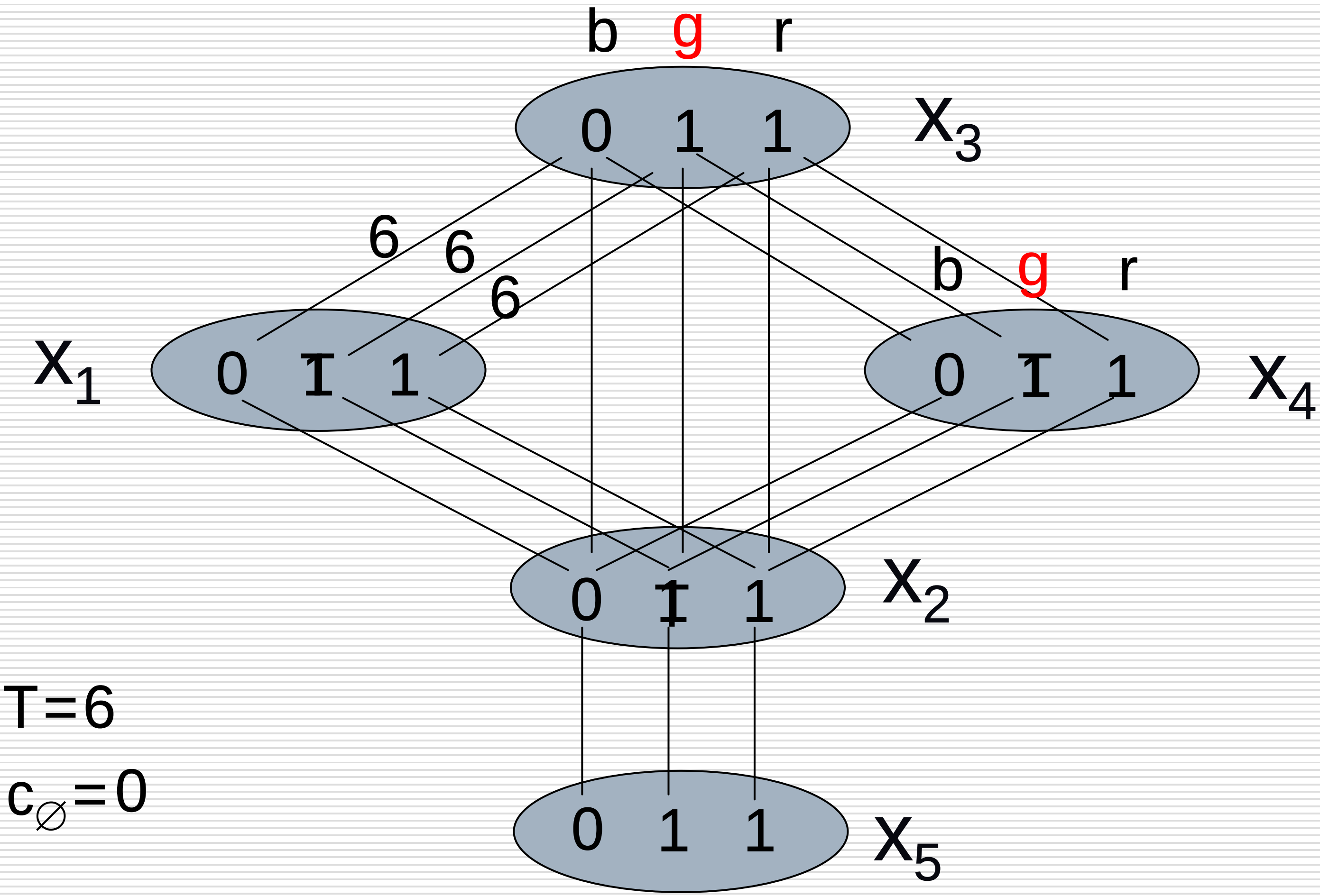
If LB ≥ UB then prune

(UB) Upper Bound = best solution so far = **T**

Assigning x_3



Assign g to X_4



$T=6$

$c_{\varnothing}=0$

Depth First Search (DFS)

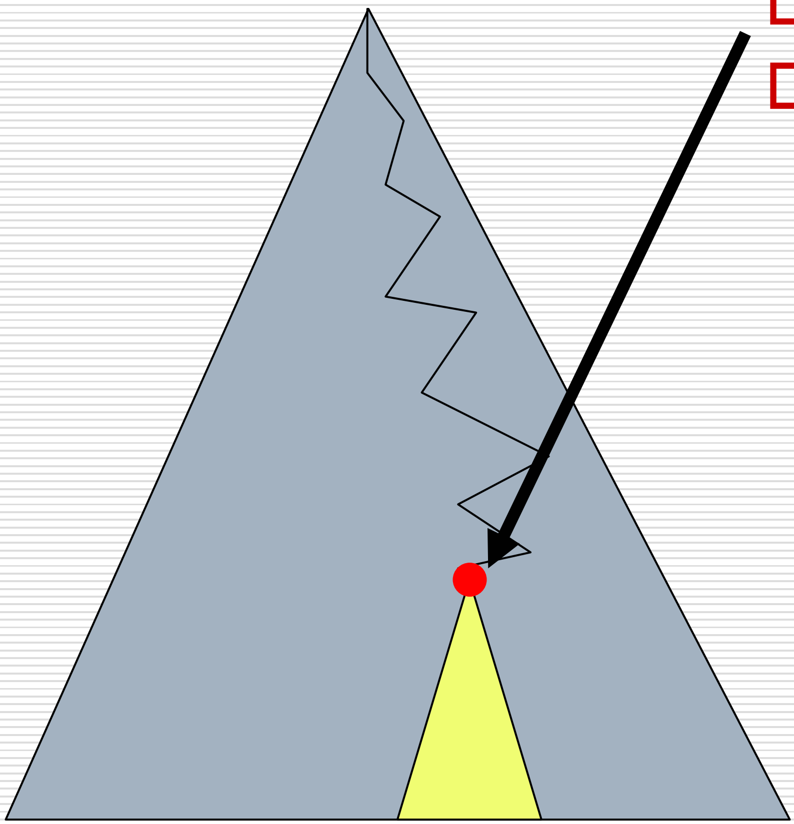
```
BT(X,D,C)
  if (X=∅) then Top := c∅
  else
    xj := selectVar(X)
    forall a∈Dj do
      ∀ c∈C s.t. xj ∈var(c)  c:=Assign(c, xj, a)
      c∅:= Σ c∈C s.t. var(c)= ∅  c
      if (LB<Top) then BT(X-{xj},D-{Dj},C)
```

Heuristics

Improve LB

Good UB ASAP

Improving the LB



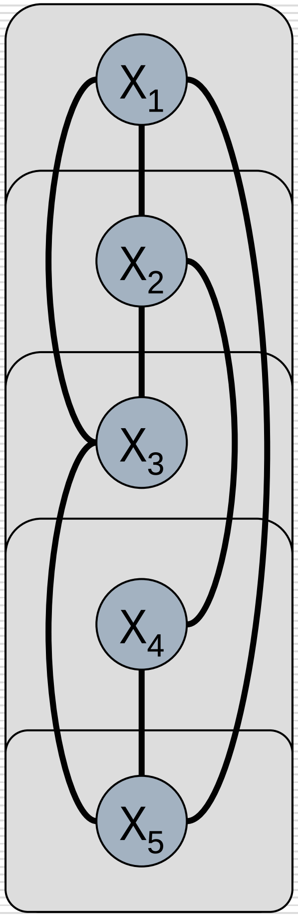
- Assigned constraints ($c_\varnothing$)
- Original constraints

Solve

Optimal cost $\oplus\ c_\varnothing$

- Gives a stronger LB
- Can be solved beforehand

Russian Doll Search



- static variable ordering
- solves increasingly large subproblems
- uses previous LB recursively
- May speedup search by several orders of magnitude

Local search

Based on perturbation of solutions in a local neighborhood

- Simulated annealing
- Tabu search
- Variable neighborhood search
- Greedy rand. adapt. search (GRASP)
- Evolutionary computation (GA)
- Ant colony optimization...

- See: Blum & Roli, ACM comp. surveys, 35(3), 2003

Boosting Systematic Search with Local Search

- Do local search prior systematic search
- Use best cost found as initial T
  - If optimal, we just prove optimality
  - In all cases, we may improve pruning

Boosting Systematic Search with Local Search

- Ex: Frequency assignment problem
  - Instance: CELAR6-sub4
    - #var: 22 , #val: 44 , Optimum: 3230
  - Solver: toolbar with default options
  - UB initialized to 100000 ➔ 10 hours
  - UB initialized to 3230 ➔ 3 hours
    - Local search can find the optimum in a few minutes

Overview

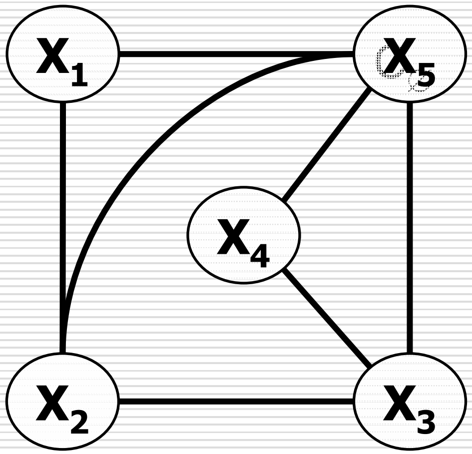
- Introduction and definitions
  - Why soft constraints and dedicated algorithms?
  - Generic and specific models
- **Solving soft constraint problems**
  - By search (systematic and local)
  - By complete inference... and search
  - By incomplete inference... and search
- Existing implementations

Variable elimination (aka bucket elimination)

- Solves the problem by a sequence of **problem transformations**
- Each step yields an **equivalent** problem with **one less variable**
- When all variables have been eliminated, the **problem is solved**
- Optimal solutions of the original problem can be recomputed

Variable elimination

- Select a variable $\mathbf{X_i}$
- Compute the set K_i of constraints that involves this variable
- Add Elim$(\bigoplus_{c \in K_i} c, X_1)$
- Remove variable and K_i
- Time: $\Theta(\exp(deg_i+1))$
- Space: $\Theta(\exp(deg_i))$



Variable elimination

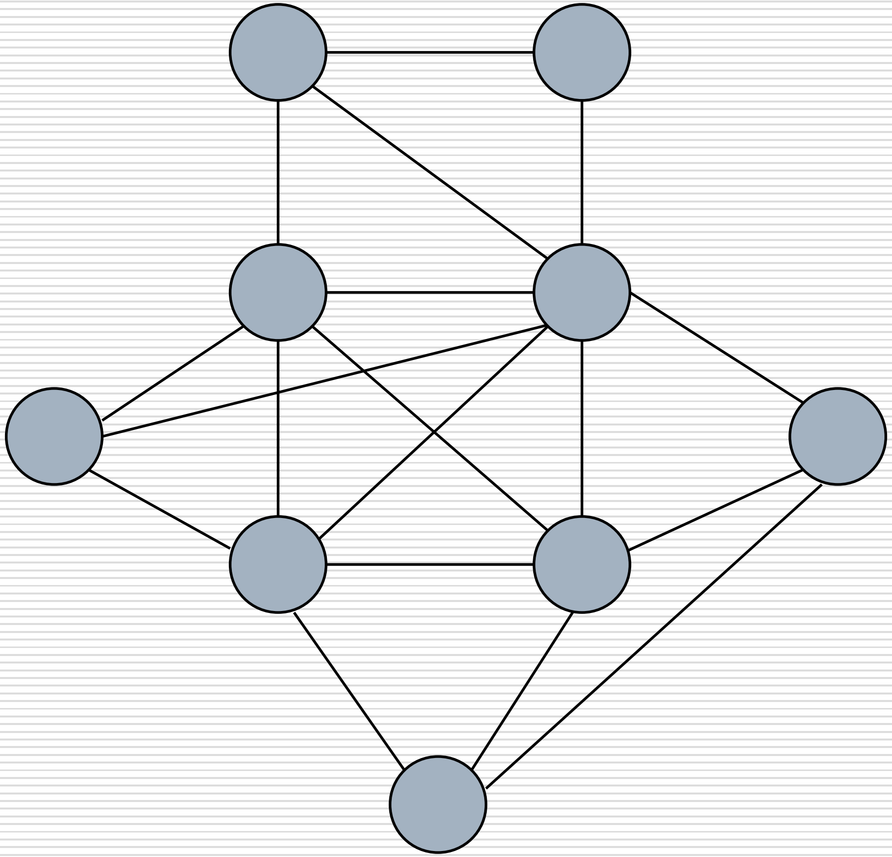
- Time: exponential in the size of the largest combination performed (1+induced width)
- Space: similar !
- Infeasible as a general method...

Boosting search with variable elimination: BB-VE(k)

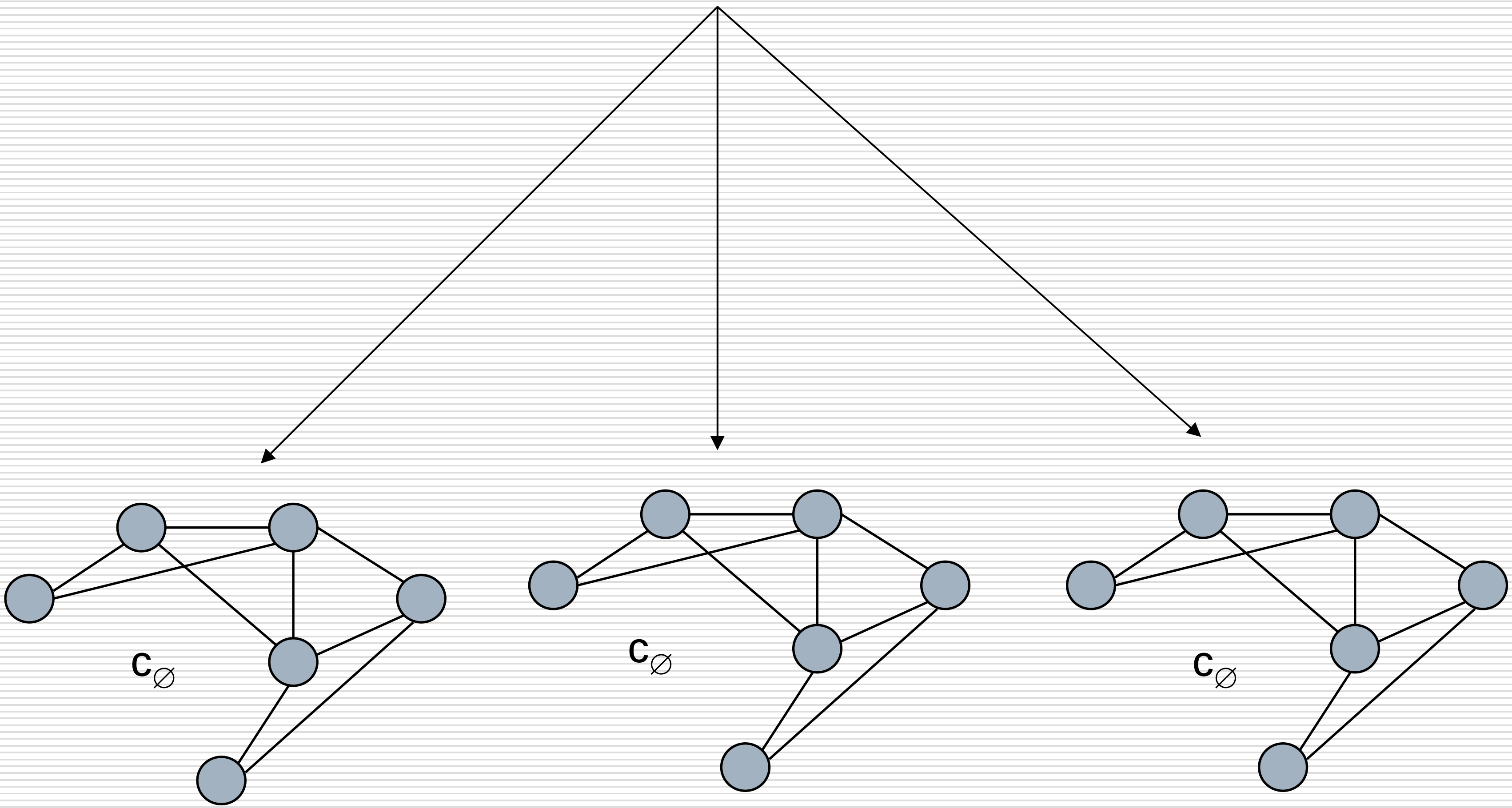
- At each node
  - **Select an unassigned variable X_i**
  - If **$deg_i \leq k$** then **eliminate X_i**
  - Else **branch on the values of X_i**

- Properties
  - BE-VE(-1) is BB
  - BE-VE(w*) is VE
  - BE-VE(1) is like cycle-cutset

Example BB-VE(2)



Example BB-VE(2)



Boosting search with variable elimination

- Ex: still-life (academic problem)
  - Instance: n=14
    - #var:196 , #val:2
  - Ilog Solver → 5 days
  - Variable Elimination → 1 day
  - BB-VE(18) → 2 seconds

Overview

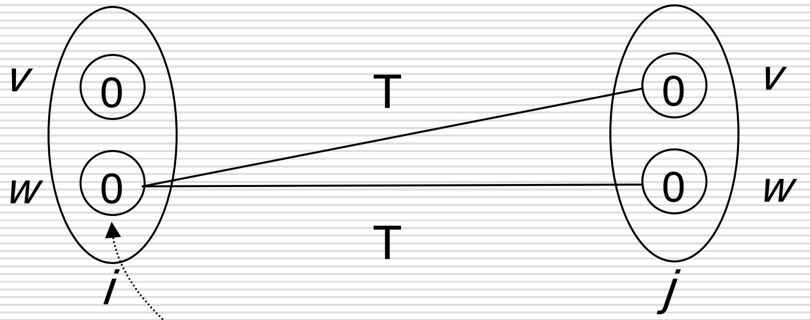
- **Introduction and definitions**
  - Why soft constraints and dedicated algorithms?
  - Generic and specific models
- **Solving soft constraint problems**
  - By search (systematic and local)
  - By complete inference... and search
  - By incomplete inference... and search
- **Existing implementations**

Local consistency

- **Local** property enforceable in **polynomial time** yielding an **equivalent** and **more explicit** problem
- Massive local (incomplete) inference
- Very useful in classical CSP
  - Node consistency
  - Arc consistency...

Classical arc consistency (binary CSP)

- for any X_i and c_{ij}
  - $f = Elim(c_{ij} \oplus c_i \oplus c_j, X_j)$ brings no new information on X_i



$c_{ij} \oplus c_i \oplus c_j$

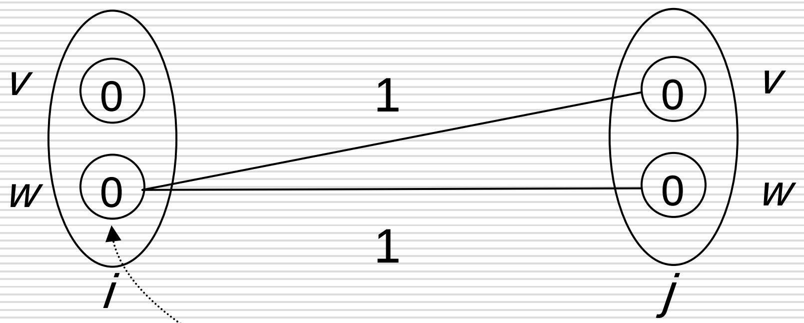
X_i	X_j	c_{ij}
v	v	0
v	w	0
w	v	T
w	w	T

Elim x_j $\Rightarrow$

X_i	$f(X_i)$
v	0
w	T

Arc consistency and soft constraints

- for any X_i and c_{ij}
  - $f = Elim(c_{ij} \oplus c_i \oplus c_j, X_j)$ brings no new information on X_i



$c_{ij} \oplus c_i \oplus c_j$

X_i	X_j	c_{ij}
v	v	0
v	w	0
w	v	1
w	w	1

Elim x_j ⇒

X_i	$f(X_i)$
v	0
w	1

EQUIVALENCE LOST ...unless $\oplus$ idempotent (fuzzy CSP)

A parenthesis…
Why min-max (fuzzy) CSP is easy

- A new operation on soft constraints:
  - **The α-cut** of a soft constraint c is the hard constraint c_α that forbids t iff $c(t) > \alpha$

x_i	x_j	$c(x_i,x_j)$
b	b	0.6
b	g	0.3
g	b	0.1
g	g	0.5

Cut(c, 0.4) →

x_i	x_j	$c(x_i,x_j)$
b	b	T
b	g	0
g	b	0
g	g	T

- Can be applied to a whole soft CN

Solving a min-max CSP by slicing

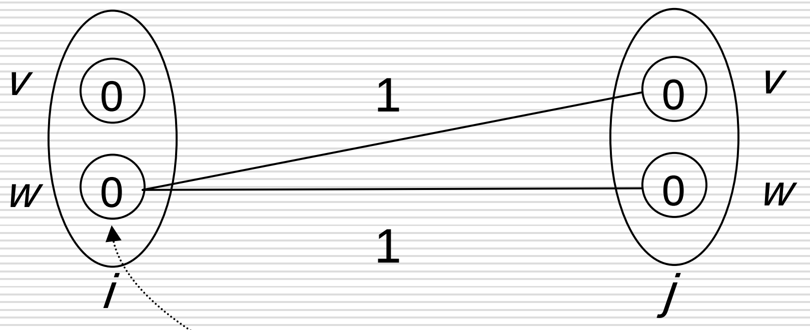
- Let S be the set of all optimal solutions of a min-max CN
- Let β be the maximum α s.t. the α-cut is consistent. Then S is the set of solutions of the β-cut.

Binary search

- A min-max (or fuzzy) CN can be solved in O(log(# different costs)) CSP.
- Can be used to lift polynomial classes

Back to Weighted CSP and Arc consistency

- for any X_i and c_{ij}
  - $f = Elim(c_{ij} \oplus c_i \oplus c_j, X_j)$ brings no new information on X_i



$c_{ij} \oplus c_i \oplus c_j$

X_i	X_j	c_{ij}
v	v	0
v	w	0
w	v	1
w	w	1

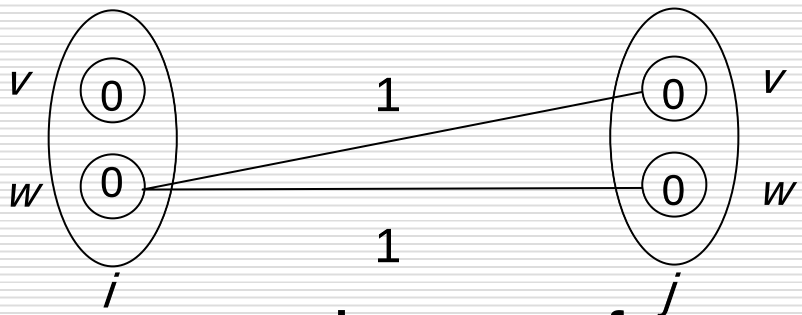
Elim x_j

X_i	$f(X_i)$
v	0
w	1

EQUIVALENCE LOST ...unless $\oplus$ idempotent (fuzzy CSP)

A new operation on soft networks

- Projection of c_{ij} on X_i with compensation



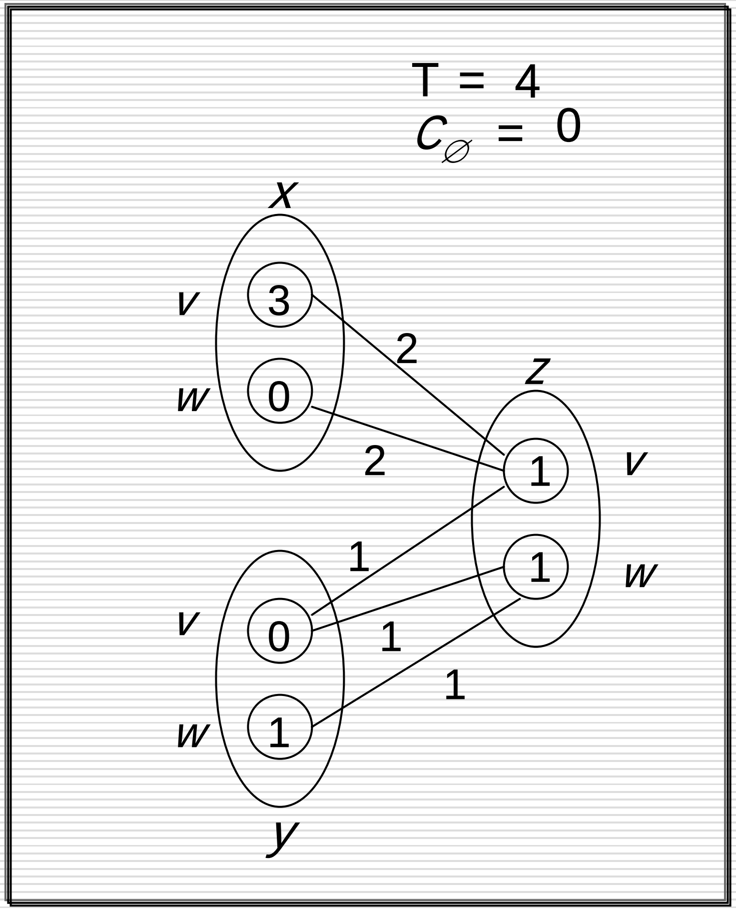
- Equivalence preserving transformation
- Requires the existence of a pseudo difference $(a \ominus b) \oplus b = a$
- Can be reversed

Node Consistency (NC*)

- For all variable i
  - $\forall a,\ C_\varnothing + C_i(a) < T$
  - $\exists\ a,\ C_i(a) = 0$

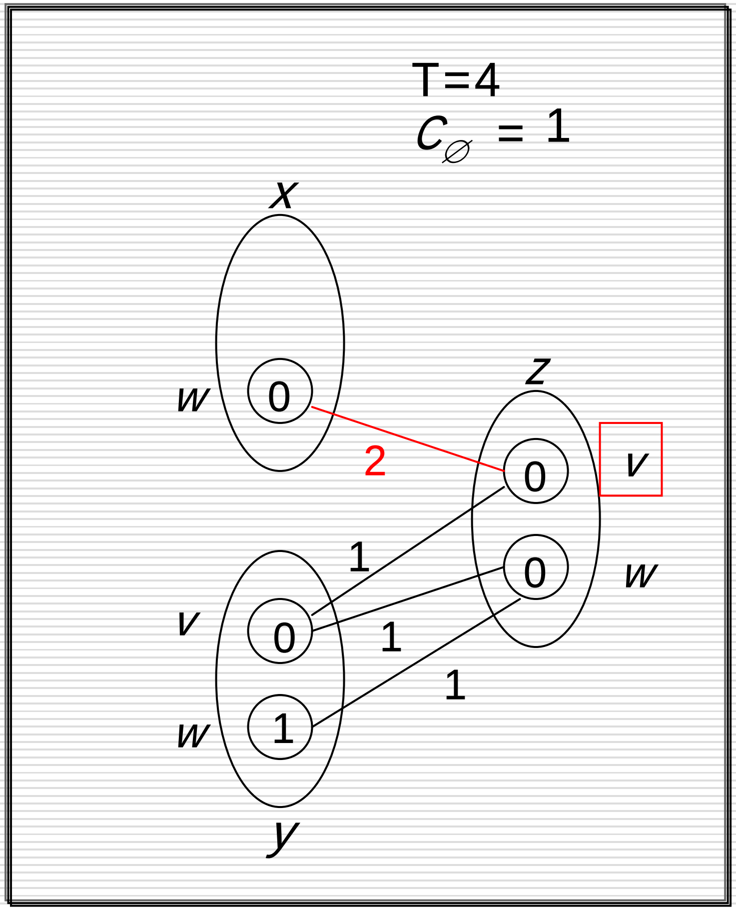
- Complexity:

  $\mathbf{O}(nd)$



Arc Consistency (AC*)

- NC*
- For all C_{ij}
  - $\forall a \exists b$
    $C_{ij}(a,b) = 0$
- b is a *support*
- complexity:
  $\mathbf{O}(n^2 d^3)$

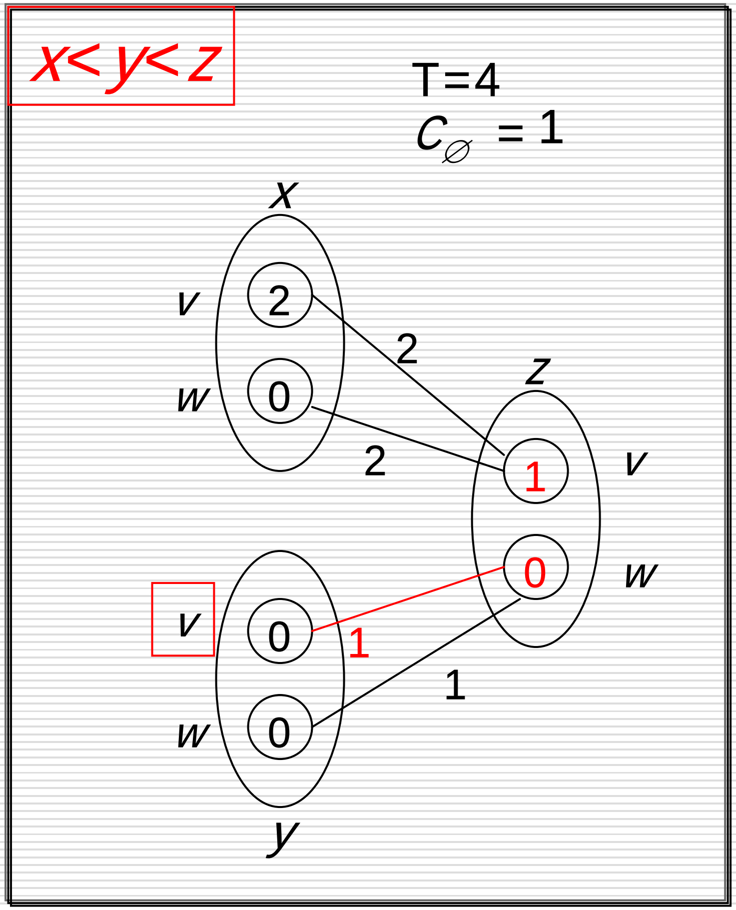


Directional AC (DAC*)

- NC*
- For all C_{ij} $(i<j)$
  - $\forall a \, \exists b$

    $C_{ij}(a,b) + C_j(b) = 0$
- *b* is a *full-support*
- complexity:

  O(ed^2)

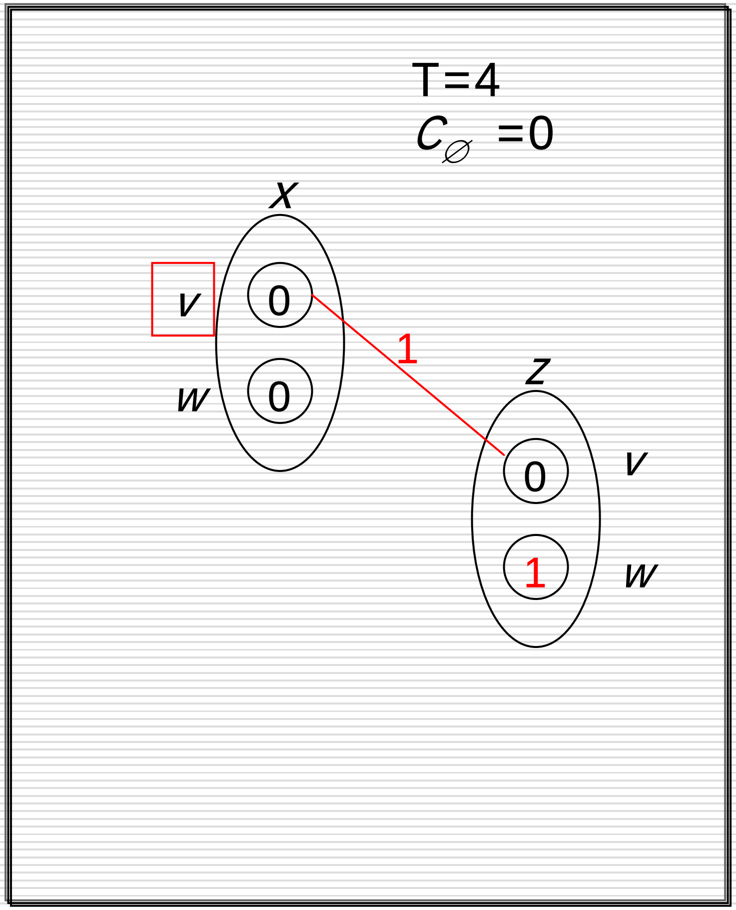


Full AC (FAC*)

- NC*
- For all C_{ij}
  - $\forall a \exists b$

    $C_{ij}(a,b) + C_j(b) = 0$

    (full support)



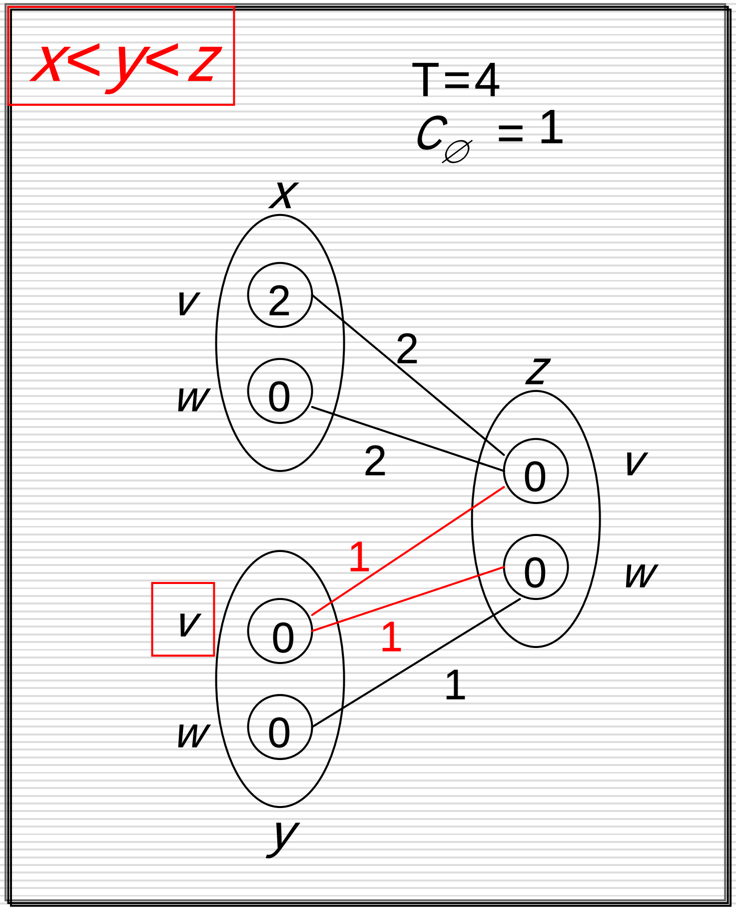
Full DAC (FDAC*)

- NC*
- For all C_{ij} ($i<j$)
  - $\forall a \exists b$

    $C_{ij}(a,b) + C_j(b) = 0$

    (full support)
- For all C_{ij} ($i>j$)
  - $\forall a \exists b$, $C_{ij}(a,b) = 0$

    (support)
- Complexity: O(end³)



Hierarchy

Special case: CSP (Top=1)

$$\begin{tikzcd}[column sep=small]
 & \text{NC}^* \ \mathbf{O}(nd) \arrow[dl] \arrow[dr] & \\
\text{AC}^* \ \mathbf{O}(n^2 d^3) \arrow[dr] & & \text{DAC}^* \ \mathbf{O}(ed^2) \arrow[dl] \\
 & \text{FDAC}^* \ \mathbf{O}(end^3) &
\end{tikzcd}$$

NC

DAC

AC

Boosting search with LC

BT(X,D,C)

  if ($X=\emptyset$) then Top $:= c_\emptyset$

  else

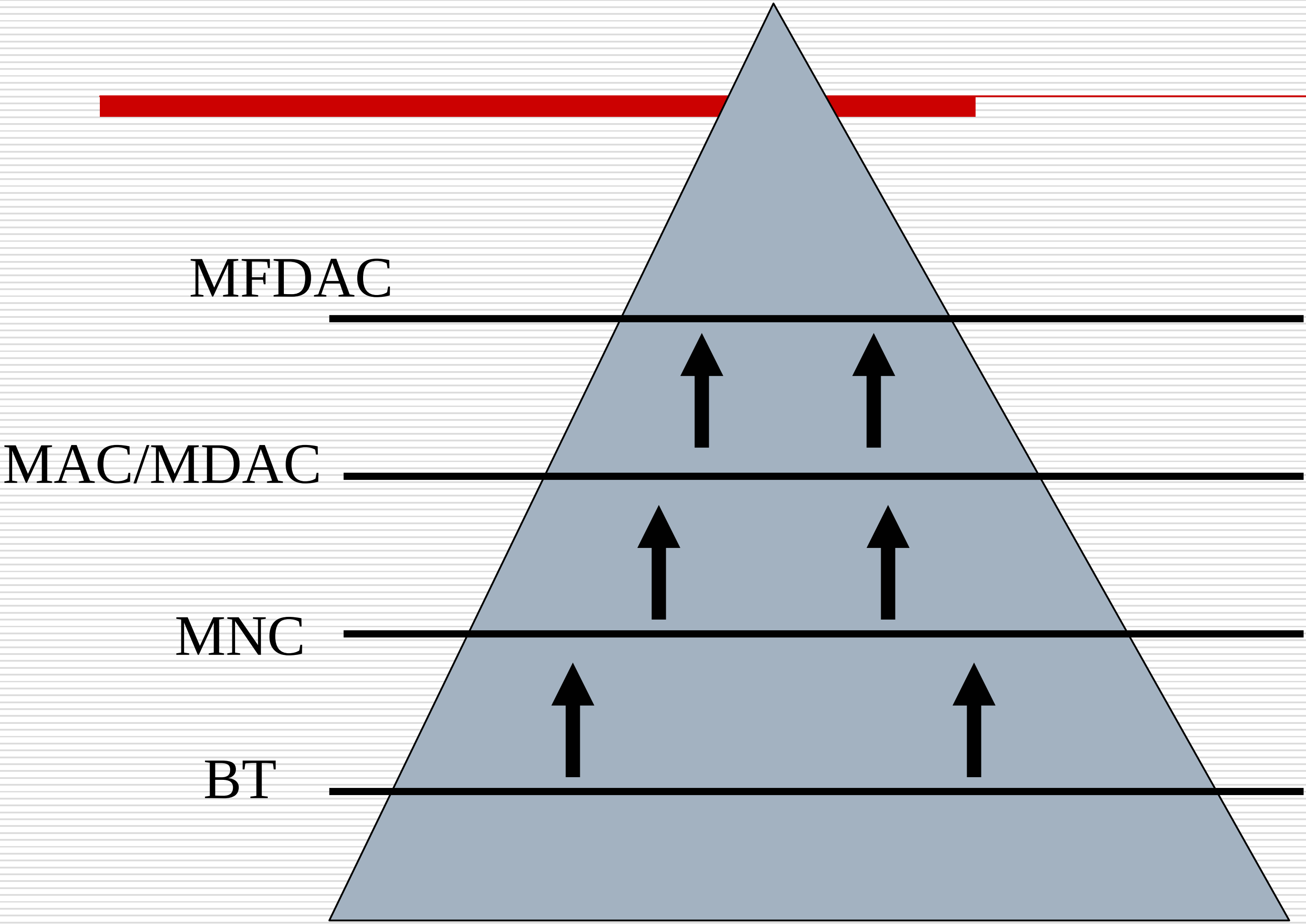
    $x_j :=$ selectVar(X)

    forall $a \in D_j$ do

      $\forall_{c \in C \text{ s.t. } xj \in \text{var}(c)}\ c := \text{Assgn}(c, x_j, a)$

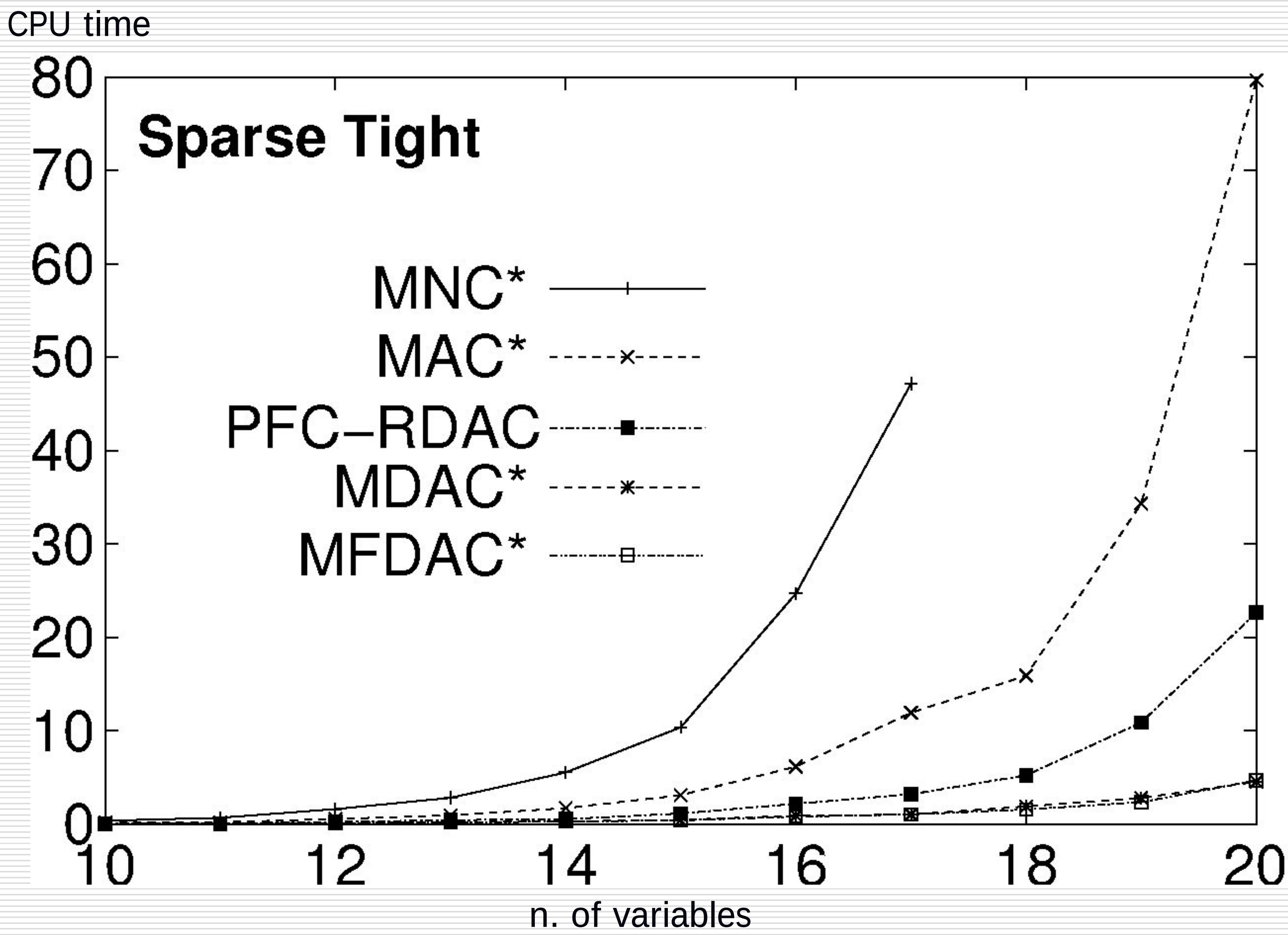
      $c_\emptyset := \Sigma_{c \in C \text{ s.t. var}(c) = \emptyset}\ c$

      if (**LC**) then BT(X-{x_j}, D-{D_j}, C)


MFDAC
MAC/MDAC
MNC
BT

Boosting Systematic Search with Local consistency

- Ex: Frequency assignment problem
  - Instance: CELAR6-sub4
    - #var: 22 , #val: 44 , Optimum: 3230
  - Solver: toolbar
  - MNC*→ 1 year (estimated)
  - MFDAC* → 1 hour


CPU time
Sparse Tight
80
70
60
50
40
30
20
10
0
MNC*
MAC*
PFC–RDAC
MDAC*
MFDAC*
10
12
14
16
18
20
n. of variables

Overview

- Introduction and definitions
  - Why soft constraints and dedicated algorithms?
  - Generic and specific models
- Solving soft constraint problems
  - By search (systematic and local)
  - By complete inference... and search
  - By incomplete inference... and search
- **Existing implementations**

Known implementations

- **Con'Flex:** fuzzy CSP system with integer, symbolic and numerical constraints (*www.inra.fr/bia/T/conflex*).
- **clp(FD,S):** semi-ring CLP (*pauillac.inria.fr/.georget/clp_fds/clp_fds.html*).
- **LVCSP**: Common-Lisp library for Valued CSP with an emphasis on strictly monotonic operators (*ftp.cert.fr/pub/lemaitre/LVCSP*).
- **Choco**: a claire library for CSP. Existing layers above Choco implements some WCSP algorithms (*www.choco-constraints.net*).
- **toolbar**: C library for MaxCSP and related problems (*carlit.toulouse.inra.fr/cgi-bin/awki.cgi/ToolBarIntro*).

carlit.toulouse.inra.fr/cgi-bin/awki.cgi/SoftCSP