2nd. Part

- Modeling
  - Primality/Duality
  - Global Constraints
- Constraint programming
  - examples in CHOCO
- Soft Constraints
  - Models
  - Algorithms

Modeling

- Any CSP can be formulated in different (equivalent) ways
- The efficiency of the solving algorithms can vary dramatically
- No strong results are known
- Active line of research
- Alternative formulations:
  - Primal/Dual
  - Primitive/Global constraints

Primal/Dual

Primal CSP: (X, D, C)

- $X = \{x_1, x_2, \ldots, x_n\}$, $D = \{d_1, d_2, \ldots, d_n\}$, $C = \{c_1, c_2, \ldots, c_r\}$

  $c \in C$ $\quad var(c) = \{x_i, x_j, \ldots, x_k\}$ $\quad$ scope

  $\quad rel(c) \subseteq d_i \times d_j \times .. \times d_k$ $\quad$ permitted tuples

Dual CSP: (X', D', C')

- $X' = \{x'_1, x'_2, \ldots, x'_r\}$,
- $D' = \{d'_1, d'_2, \ldots, d'_r\}$, where $\quad d'_i = rel(c_i)$
- $C' = \{c'_{ij}\}$, binary constraints

  $var(c'_{ij}) = \{x'_i, x'_j\}$

  $\forall c'_{ij} \in C' \Leftrightarrow \mathrm{var}(c_i) \cap \mathrm{var}(c_j) \neq \varnothing$

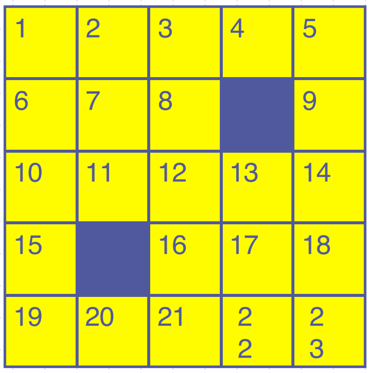
  $rel(c'_{ij})$ = consistent pairs of tuples

Example: Crossword puzzles

1	2	3	4	5
6	7	8	■	9
10	11	12	13	14
15	■	16	17	18
19	20	21	22	23

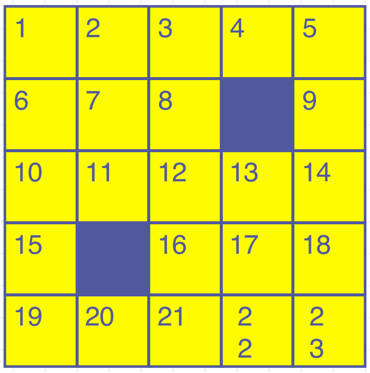
a	monarch
aardvark	monarchy
aback	monarda
abacus	…
abaft	zymurgy
abalone	zyrian
abandon	zythum
…	

Primal model (Non-binary)



- variables: cells
- domains:
  'a', …, 'z'
- constraints: contiguous letters must form words in dictionary

Dual model (binary)



- variables: words across and down
- domains: words from dictionary
- constraints: intersecting words must agree on common letter

Global Constraints

c is global iff:

- arity(c)=$r > 2$
- c is logically equivalent to $\{c_1,c_2,\dots,c_k\}$ binary
- **AC**(c) *prunes more* than **AC**($c_1,c_2,\dots,c_k$)

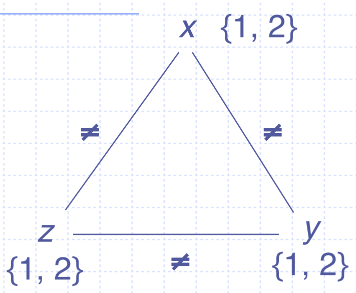
Propagation:

- There is a specialized efficient algorithm (exploits the semantics)

Catalog:

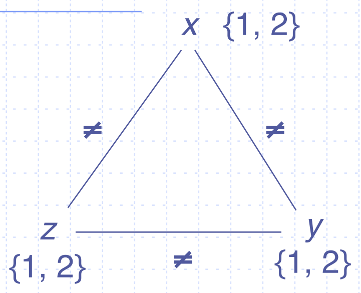
- set of global constraints
- best known algorithms for propagation

Example: all-different

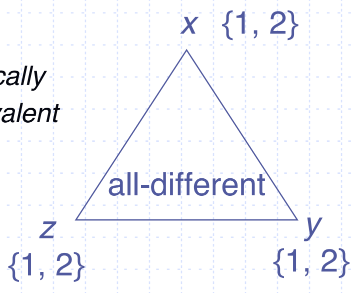


3 binary constraints,
they are AC,
no pruning

Example: all-different



logically equivalent

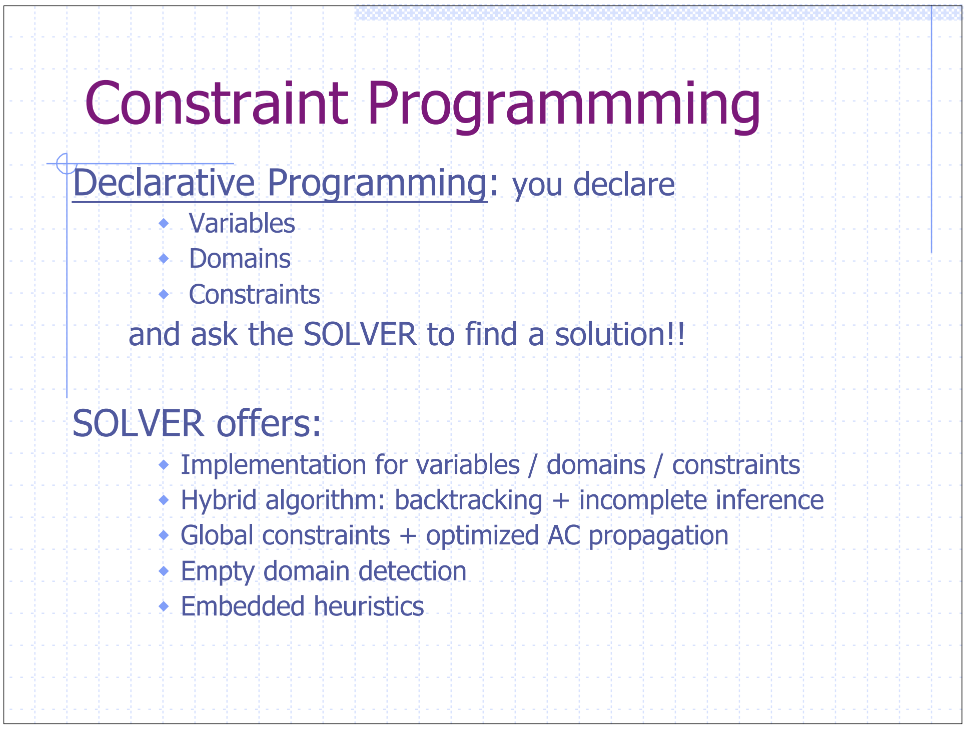


3 binary constraints,
they are AC,
no pruning

1 ternary constraint,
it is not AC,
AC pruning □ empty domain
no solution!!

Example: all-different

- Enforcing arc-consistency:
  - *n* variables, *d* values
  - *n(n-1)/2* binary constraints : $O(n^2 d^2)$
  - 1 *n*-ary constraint:
    - general purpose algorithm $O(d^n)$
    - specialized algorithm $O(n^2 d^2)$



Constraint Programmming

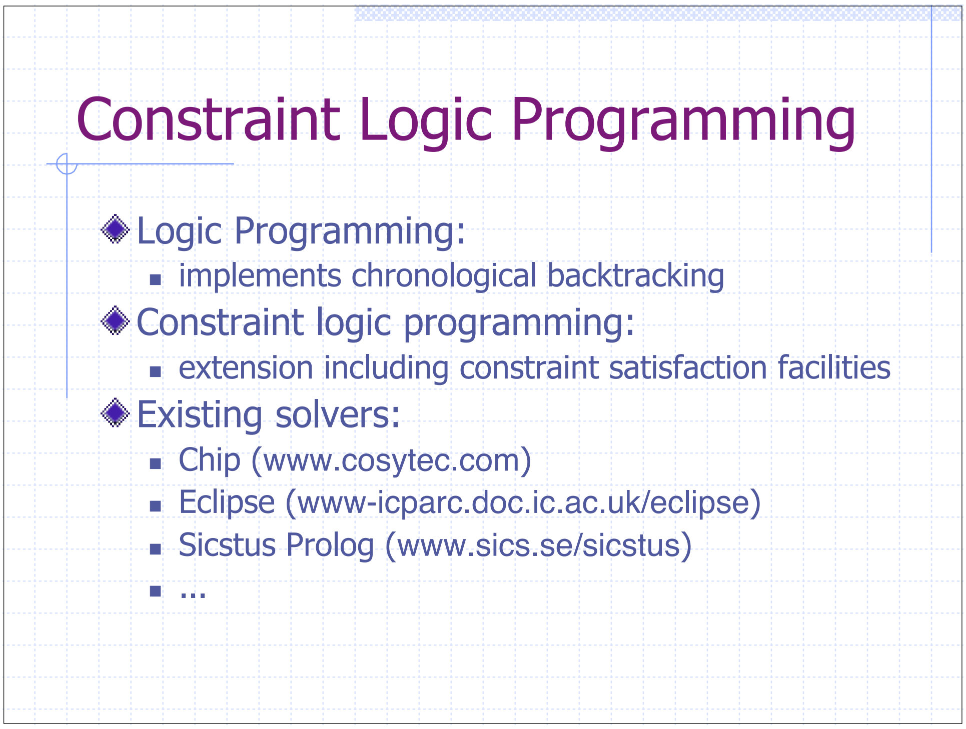
Declarative Programming: you declare

- Variables
- Domains
- Constraints

and ask the SOLVER to find a solution!!

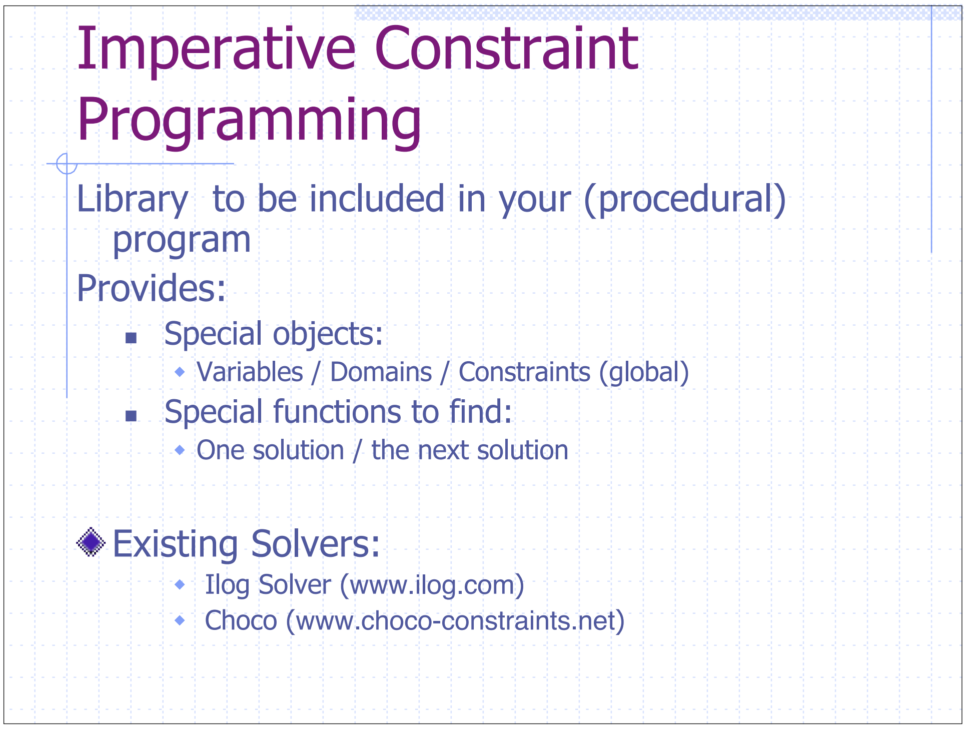
SOLVER offers:

- Implementation for variables / domains / constraints
- Hybrid algorithm: backtracking + incomplete inference
- Global constraints + optimized AC propagation
- Empty domain detection
- Embedded heuristics



Constraint Logic Programming

- Logic Programming:
  - implements chronological backtracking
- Constraint logic programming:
  - extension including constraint satisfaction facilities
- Existing solvers:
  - Chip (www.cosytec.com)
  - Eclipse (www-icparc.doc.ic.ac.uk/eclipse)
  - Sicstus Prolog (www.sics.se/sicstus)
  - ...



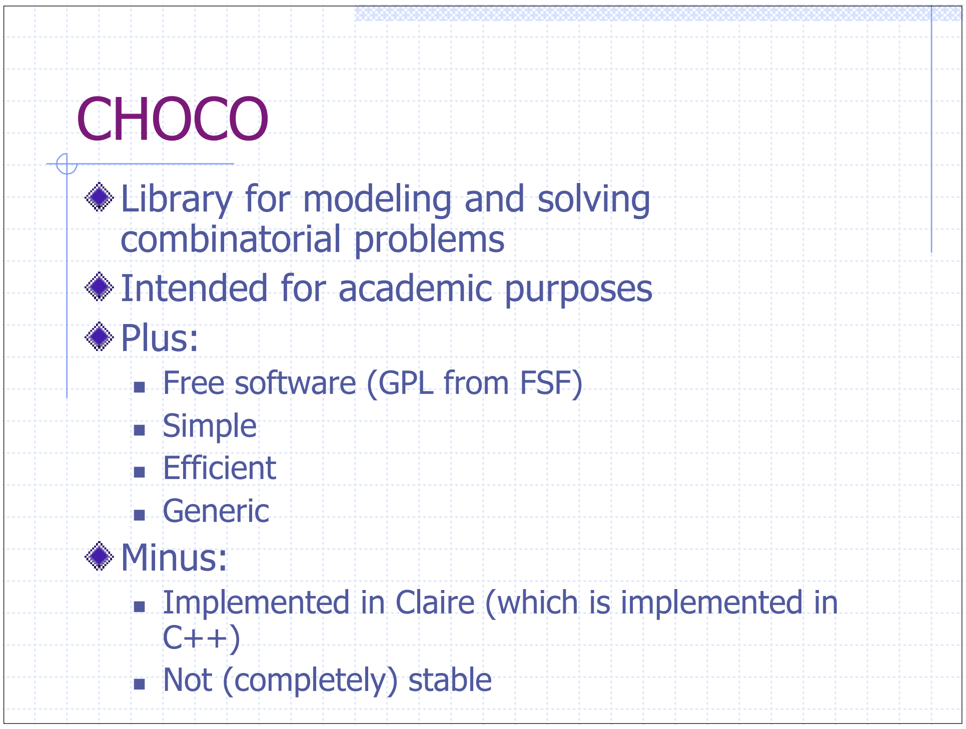
Imperative Constraint Programming

Library to be included in your (procedural) program

Provides:

- Special objects:
  - Variables / Domains / Constraints (global)
- Special functions to find:
  - One solution / the next solution

- Existing Solvers:
  - Ilog Solver (www.ilog.com)
  - Choco (www.choco-constraints.net)



CHOCO

- Library for modeling and solving combinatorial problems
- Intended for academic purposes
- Plus:
  - Free software (GPL from FSF)
  - Simple
  - Efficient
  - Generic
- Minus:
  - Implemented in Claire (which is implemented in C++)
  - Not (completely) stable

Choco: 1st example

```
[sillyCSP() : void
 -> let pb := choco/makeProblem("Silly CSP",3),
        x := choco/makeIntVar(pb, "x", 1, 3),
        y := choco/makeIntVar(pb, "y", 1, 3),
        z := choco/makeIntVar(pb, "z", 1, 3) in
        (choco/post(pb, x + y == z),
       choco/post(pb, x > y),
        choco/solve(pb,false),
        printf("~S ~S ~S\n",x,y,z) )]
```

Choco: 2nd example

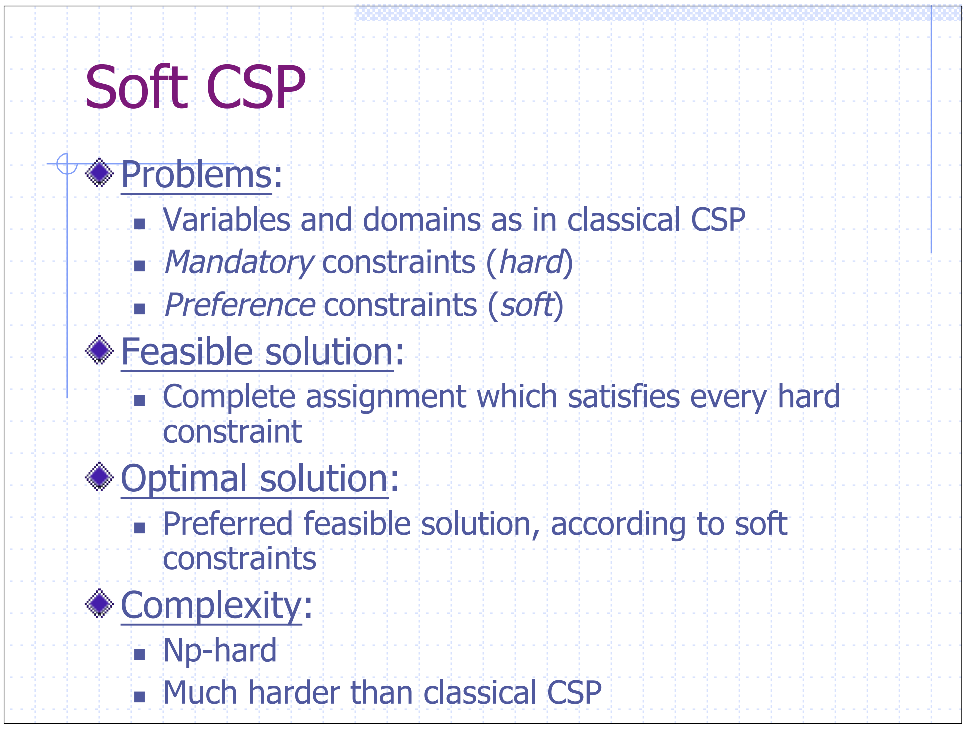
```
[queens(n:integer, all:boolean)
 -> let pb := choco/makeProblem(" n queens",n),
      queens := list{choco/makeIntVar(pb,"Q" /+ string!(i), 1, n) | i in (1 .. n) }
    in
      (for i in (1 .. n)
        for j in (i + 1 .. n)
          let k := j - i in
            ( choco/post(pb, queens[i] !== queens[j]),
              choco/post(pb, queens[i] !== queens[j] + k),
              choco/post(pb, queens[j] !== queens[i] + k) ),
      choco/solve(pb,all) )]
```

Soft Constraints (2nd. Part)

- Motivation (10’)
- Models (20’)
- Algorithms (60’)

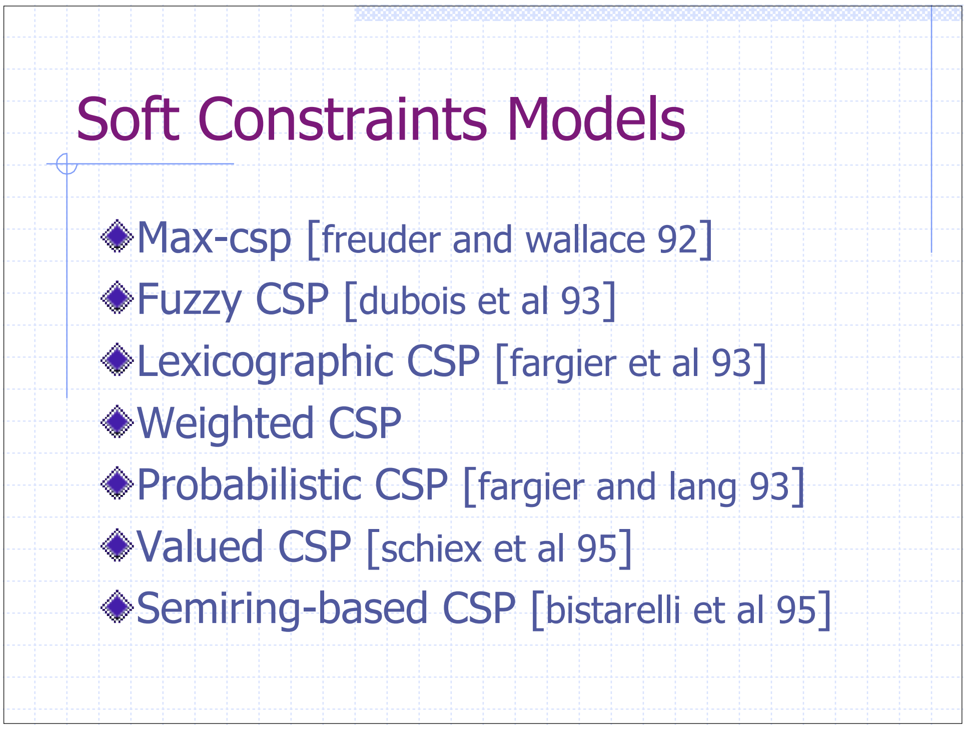
Motivation

- Using the classical CSP framework:
  - Many problems have **many** solutions
    - Algorithms either give the first one they find or all of them
    - Typically, the user likes some solutions more than others
  - Many problems **do not** have any solution
    - Algorithms just report failure
    - Typically, the user can identify some non critical constraint



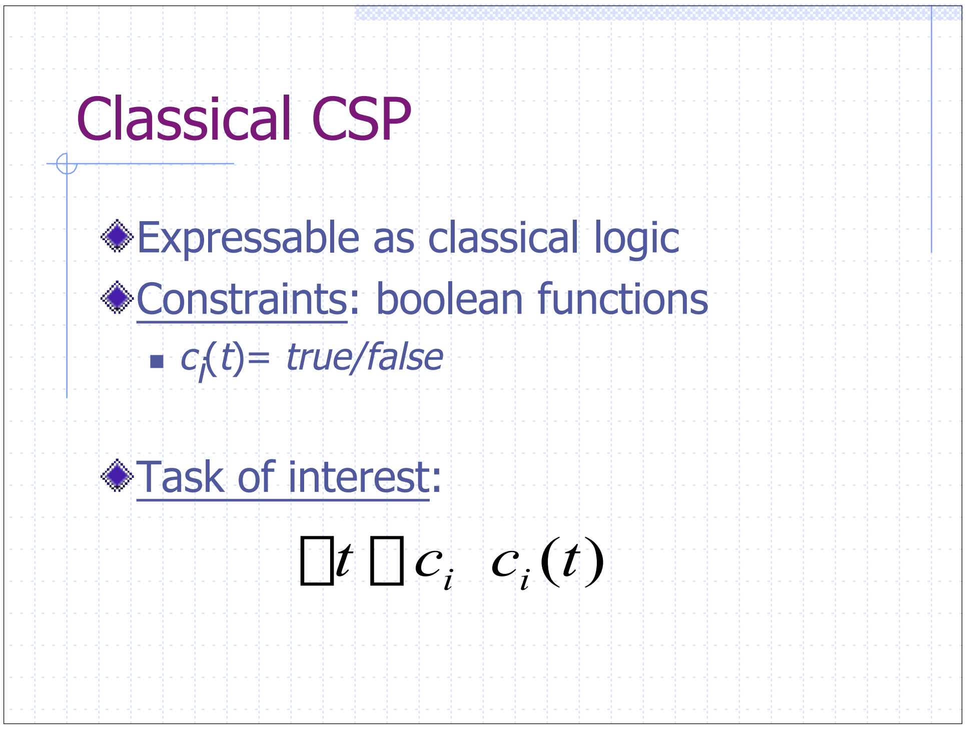
Soft CSP

- Problems:
  - Variables and domains as in classical CSP
  - *Mandatory* constraints (*hard*)
  - *Preference* constraints (*soft*)
- Feasible solution:
  - Complete assignment which satisfies every hard constraint
- Optimal solution:
  - Preferred feasible solution, according to soft constraints
- Complexity:
  - Np-hard
  - Much harder than classical CSP



Soft Constraints Models

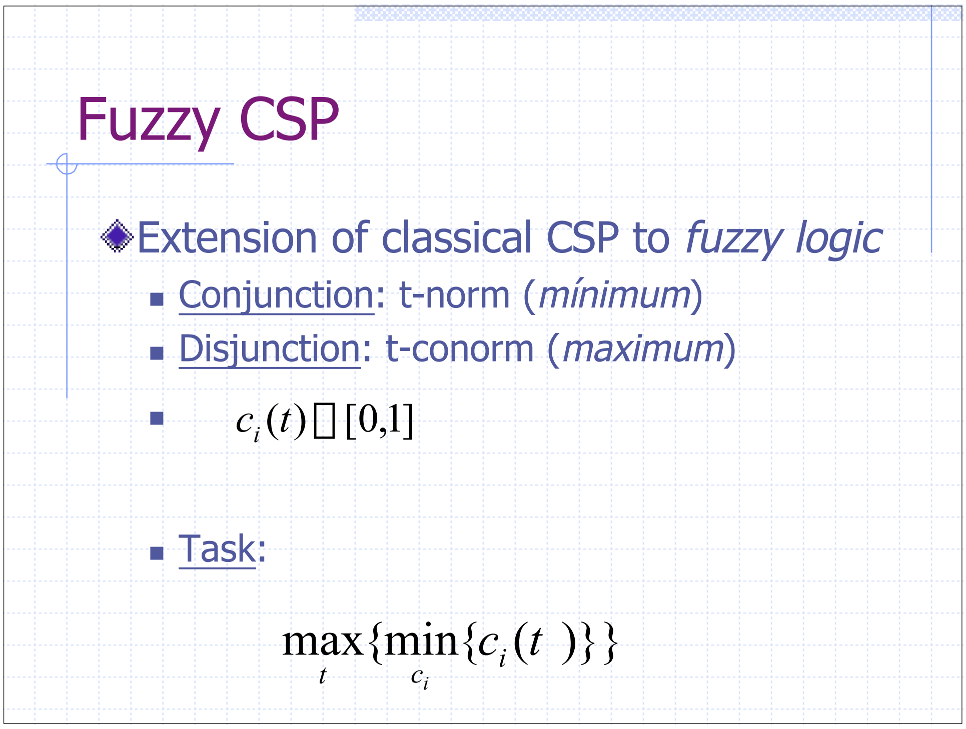
- Max-csp [freuder and wallace 92]
- Fuzzy CSP [dubois et al 93]
- Lexicographic CSP [fargier et al 93]
- Weighted CSP
- Probabilistic CSP [fargier and lang 93]
- Valued CSP [schiex et al 95]
- Semiring-based CSP [bistarelli et al 95]



Classical CSP

- Expressable as classical logic
- Constraints: boolean functions
  - $c_i(t)$= *true/false*
- Task of interest:

$$\exists t \; \forall c_i \; c_i(t)$$



Fuzzy CSP

- Extension of classical CSP to *fuzzy logic*
  - Conjunction: t-norm (*mínimum*)
  - Disjunction: t-conorm (*maximum*)
  - $c_i(t) \in [0,1]$
  - Task:

$$\max_{t}\{\min_{c_i}\{c_i(t)\}\}$$

Weighted CSP

- Preferences are expressed as *costs*
  - Constraints: cost functions

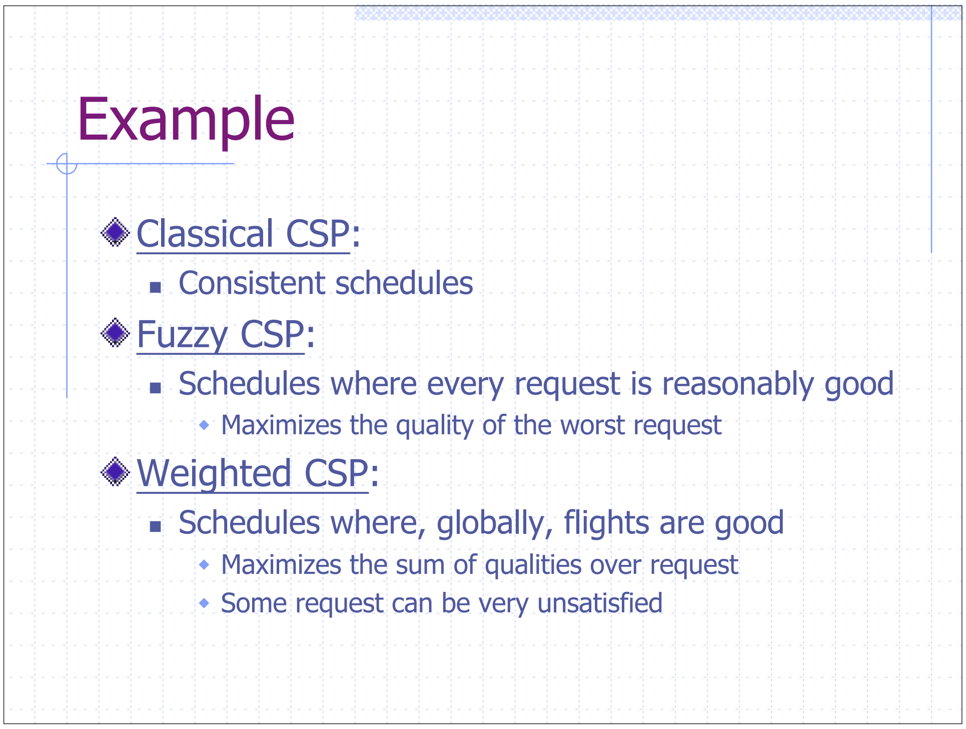
$$c_i(t) \in \{0,1,\ldots,\infty\}$$

  - Task:

$$\min_t \{\sum_{c_i} \{c_i(t)\}\}$$

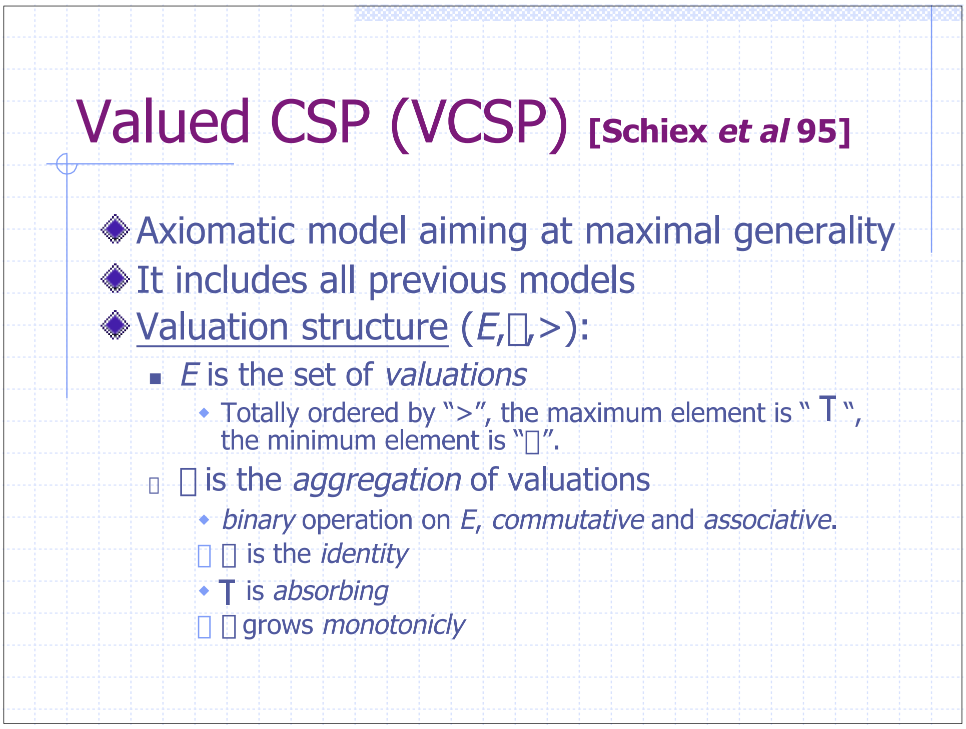
Example

- Airlines flight scheduling:
  - Input:
    - Aircrafts, airports
    - Flights: (origin, destination, frequency)
    - Requirements:
      - From origin to destination on the corresponding date
      - ...
    - Requests:
      - No more than four legs per flight
      - 1 hour < transfer time < 5 hours
      - ...
  - Output:
    - Schedule: each flight is a sequence of scheduled legs



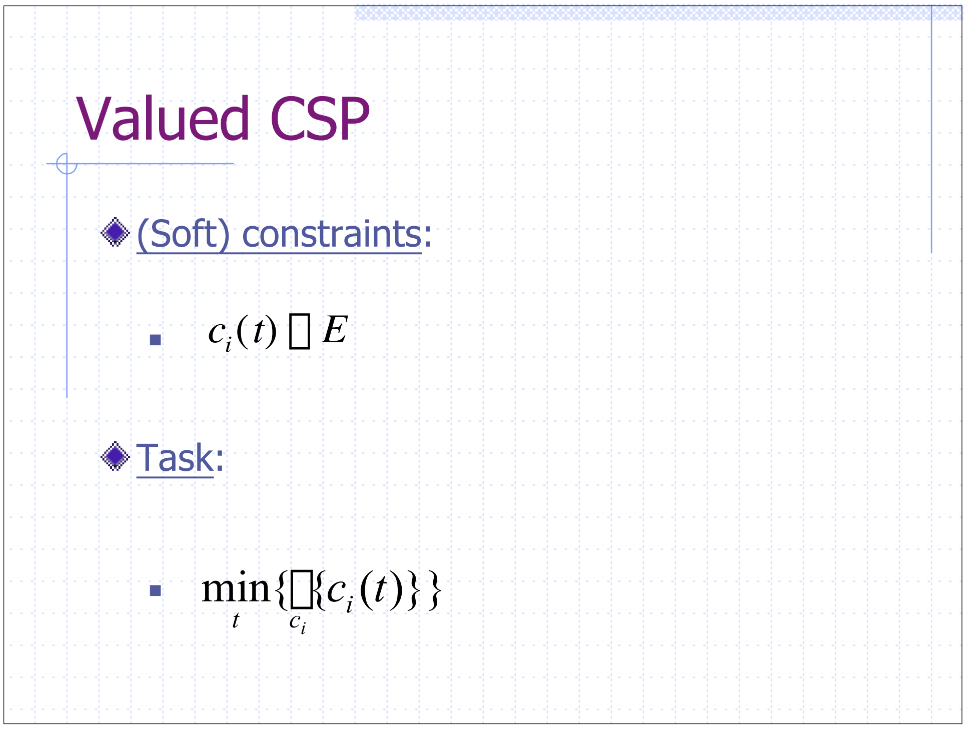
Example

- **Classical CSP:**
  - Consistent schedules
- **Fuzzy CSP:**
  - Schedules where every request is reasonably good
    - Maximizes the quality of the worst request
- **Weighted CSP:**
  - Schedules where, globally, flights are good
    - Maximizes the sum of qualities over request
    - Some request can be very unsatisfied



Valued CSP (VCSP) **[Schiex *et al* 95]**

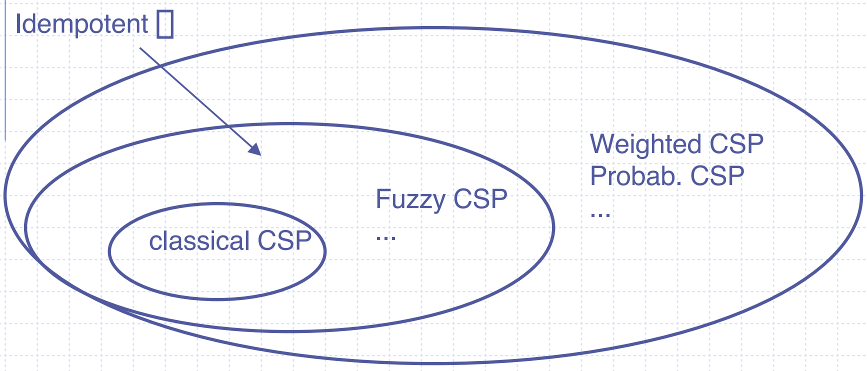
- Axiomatic model aiming at maximal generality
- It includes all previous models
- Valuation structure $(E, \oplus, >)$:
  - *E* is the set of *valuations*
    - Totally ordered by ">", the maximum element is "$\top$", the minimum element is "$\bot$".
  - $\oplus$ is the *aggregation* of valuations
    - *binary* operation on *E*, *commutative* and *associative*.
    - $\bot$ is the *identity*
    - $\top$ is *absorbing*
    - $\oplus$ grows *monotonicly*



Valued CSP

- (Soft) constraints:
  - $c_i(t) \in E$
- Task:
  - $\min_t \{ \circledast_{c_i} \{ c_i(t) \} \}$

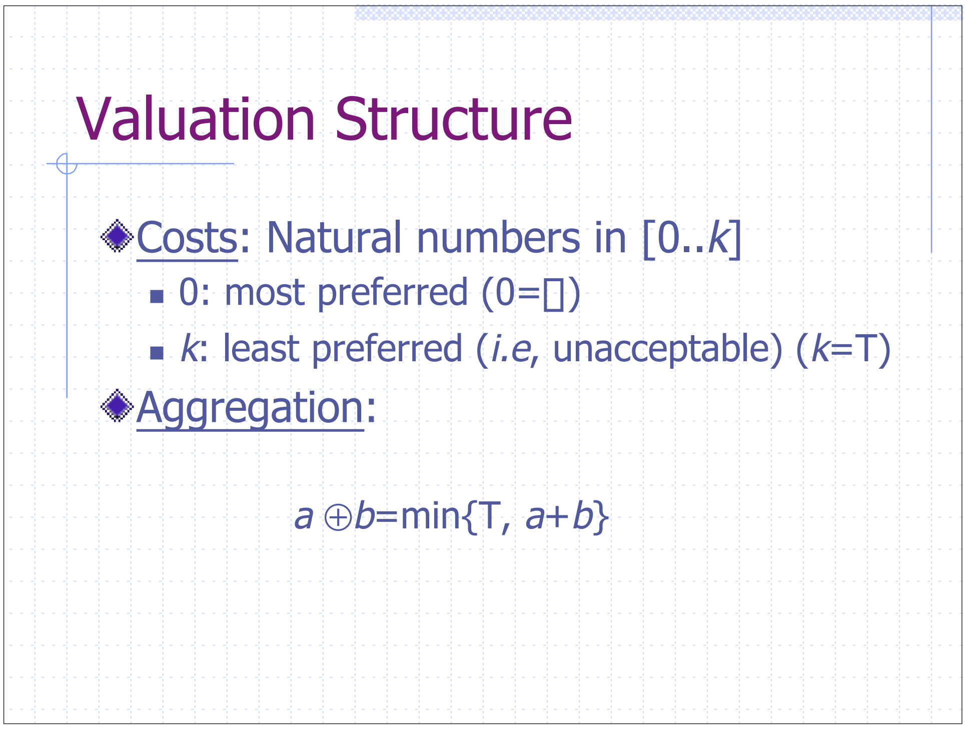
Valued CSP



Solving Valued CSP (solving Weighted CSP)

Binary Weighted CSPs

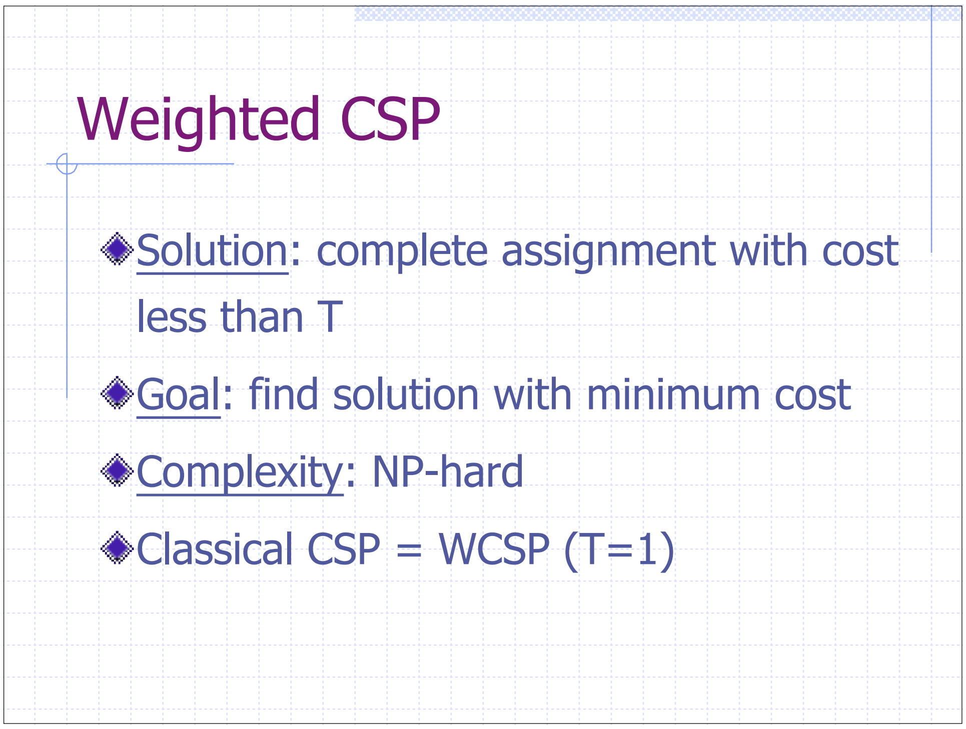
- $P=(X,D,C)$
  - $X=\{x_1,\dots, x_n\}$ variables
  - $D=\{D_1,\dots, D_n\}$ finite domains
  - $C=\{C_\varnothing, C_i, C_{ij}\}$ soft constraints
    - $C_{ij} : D_i \times D_j \rightarrow$ Cost
    - $C_i : D_i \rightarrow$ Cost
    - $C_\varnothing$: Cost (it is a constant)



Valuation Structure

- Costs: Natural numbers in [0..*k*]
  - 0: most preferred (0=⊥)
  - *k*: least preferred (*i.e*, unacceptable) (*k*=T)
- Aggregation:

$$a \oplus b = \min\{T, a+b\}$$



Weighted CSP

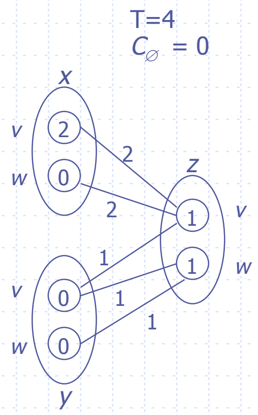
- Solution: complete assignment with cost less than T
- Goal: find solution with minimum cost
- Complexity: NP-hard
- Classical CSP = WCSP (T=1)

WCSP: Example

$X=\{x\ y\ z\}$

$D_i=\{v\ w\}$

$C=\{C_{xz}\ \ C_{yz}\ \ C_x$
$\quad C_y\ \ C_z\ \ C_{\varnothing}\}$

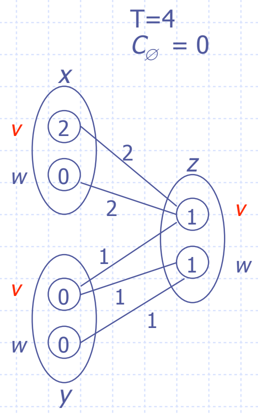


WCSP: Example

$X=\{x\ y\ z\}$

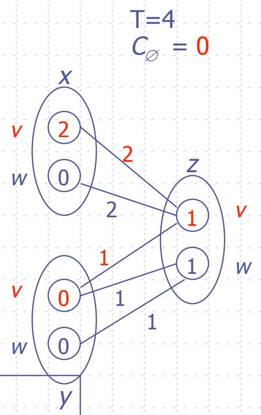
$D_i=\{v\ w\}$

$C=\{C_{xz}\ \ C_{yz}\ \ C_x$
$\quad C_y\ \ C_z\ \ C_{\varnothing}\}$



WCSP: Example

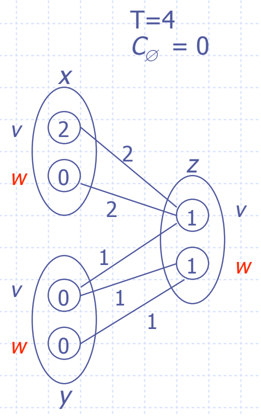
$X=\{x\ y\ z\}$
$D_i=\{v\ w\}$
$C=\{C_{xz}\ \ C_{yz}\ \ C_x$
$\quad C_y\ \ C_z\ \ C_\varnothing\}$



Valuation:
$2 \oplus 1 \oplus 2 \oplus 1 \oplus 0 \oplus 0 = T$
Not a solution

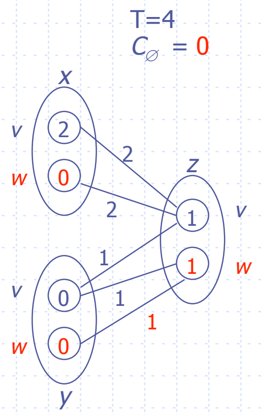
WCSP: Example

$X=\{x\ y\ z\}$
$D_i=\{v\ w\}$
$C=\{C_{xz}\ \ C_{yz}\ \ C_x$
$\quad C_y\ \ C_z\ \ C_\varnothing\}$



WCSP: Example

$X=\{x\ y\ z\}$
$D_i=\{v\ w\}$
$C=\{C_{xz}\ \ C_{yz}\ \ C_x$
$\quad C_y\ \ C_z\ \ C_{\varnothing}\}$



Valuation:
$0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 = 2$
(optimal) solution

Algorithms

- Search
  - Local search
  - Systematic search
- Inference
  - Complete inference
  - Incomplete inference
- Hybrid approaches

Local search (metaheuristics)

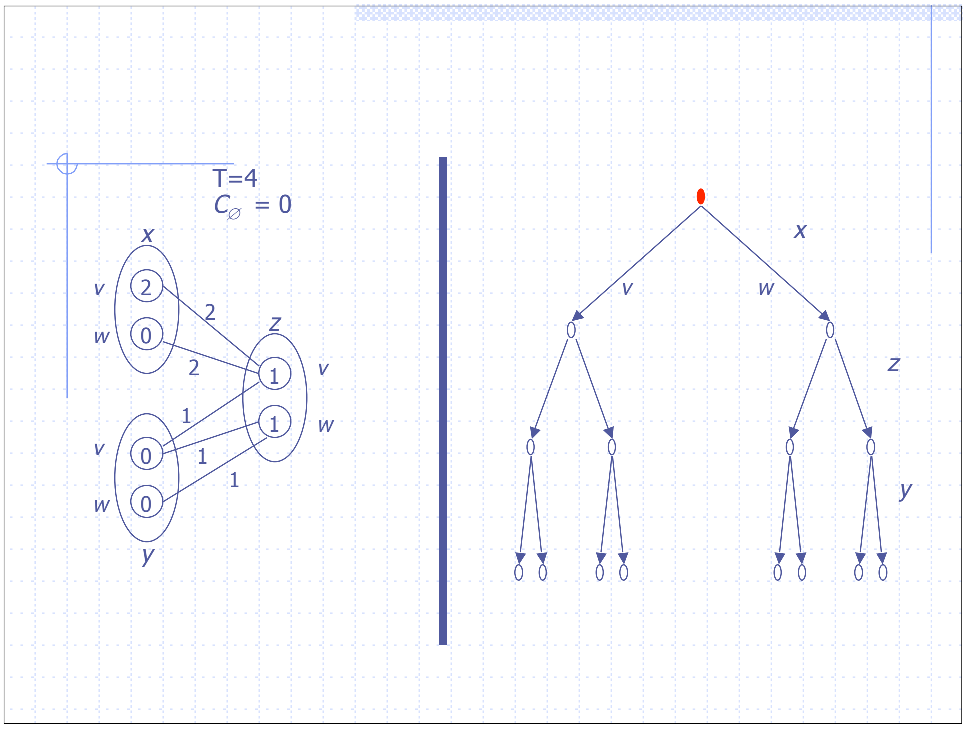
- Simulated annealing
- Tabu search
- Variable neighborhood search
- Greedy rand. adapt. search (GRASP)
- Evolutionary Computation
- Ant colony optimization

- Excellent survey: Blum & Roli, ACM computing surveys, 35(3), 2003

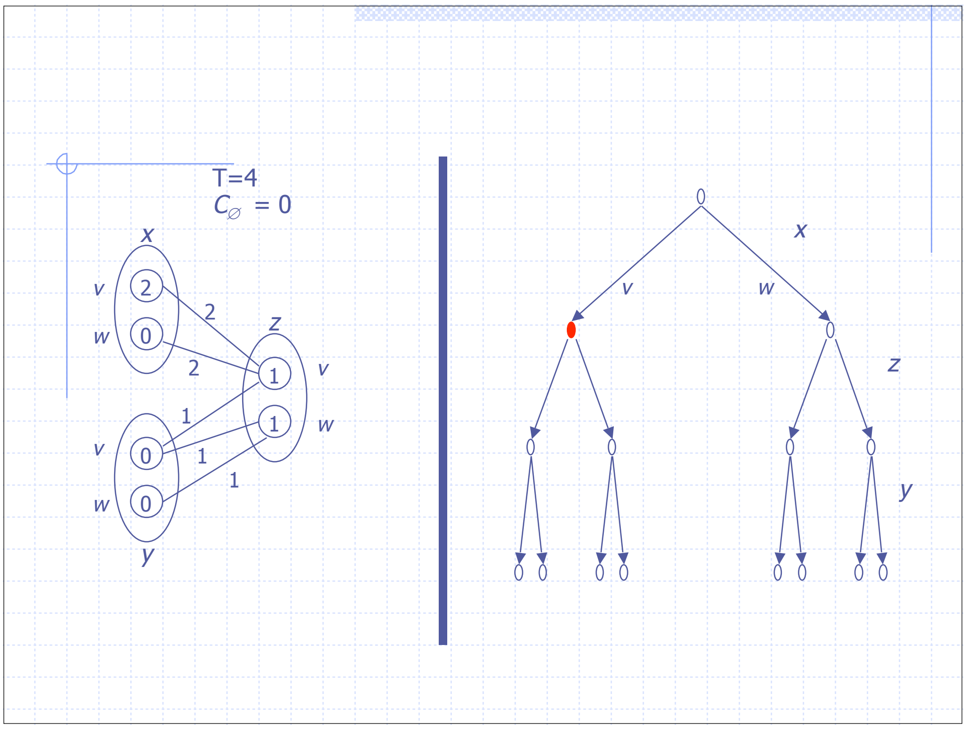
Systematic search

- Depth-first tree search:
  - *Internal node*: partial assignment
  - *Leaf*: total assignment
- At each node:
  - *Upper bound (UB)*:
    cost of the current best solution
  - *Lower bound (LB)*:
    underestimation of minimum cost among leaves below current node
- *Pruning*: *UB <= LB*

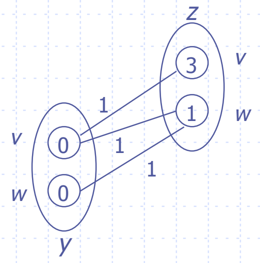
T=4
$C_{\varnothing} = 0$

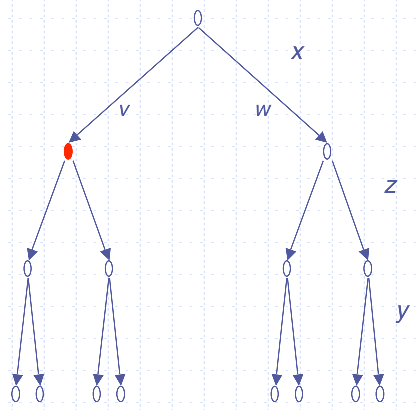


T=4
$C_{\varnothing} = 0$

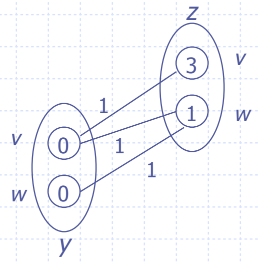


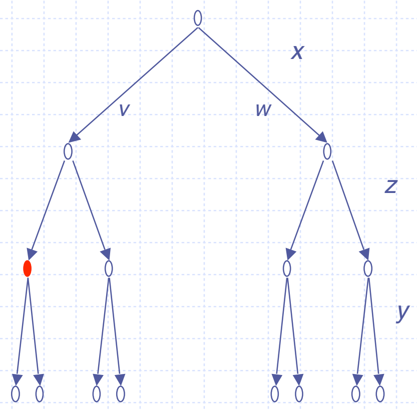
T=4
$C_{\varnothing} = 2$



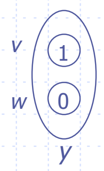


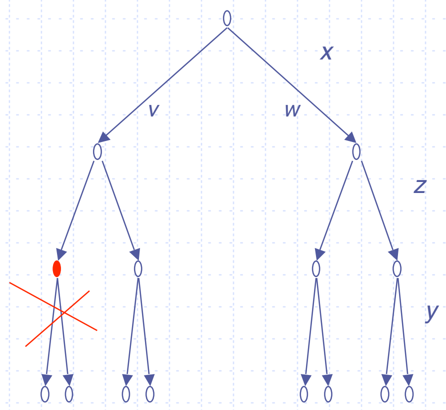
T=4
$C_{\varnothing} = 2$



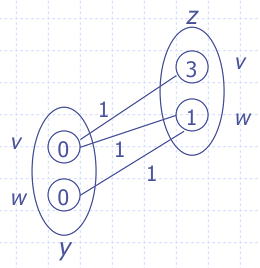


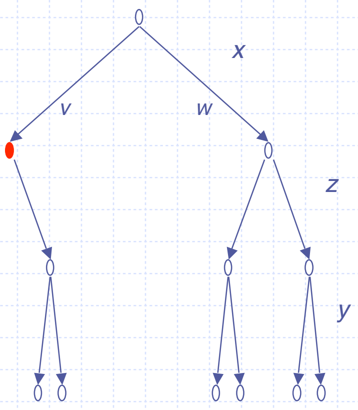
T=4
$C_\emptyset = 4$



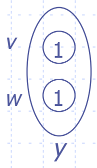


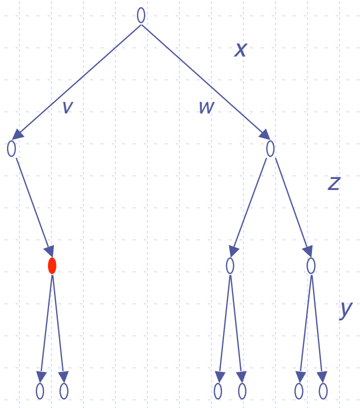
T=4
$C_\emptyset = 2$



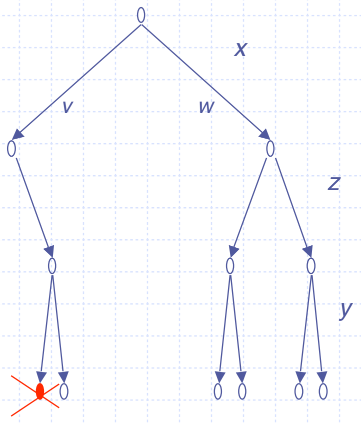


T=4
$C_{\varnothing} = 3$



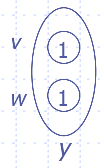


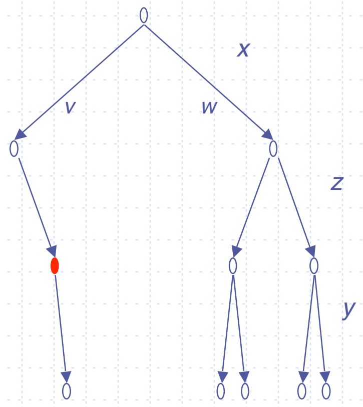
T=4
$C_{\varnothing} = 4$



T=4

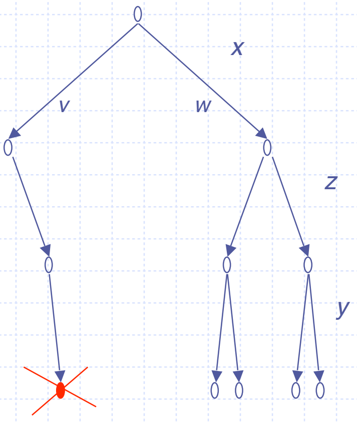
$C_{\emptyset} = 3$



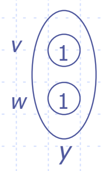


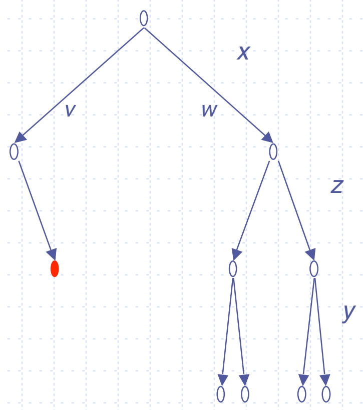
T=4

$C_{\emptyset} = 4$

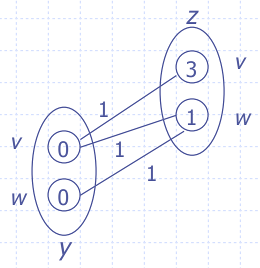


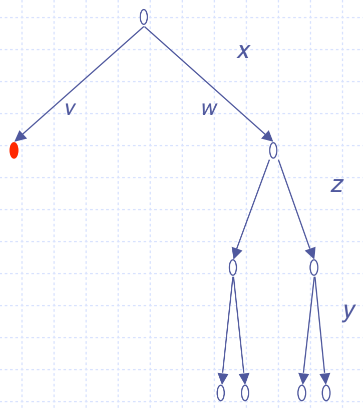
T=4
$C_\varnothing = 3$



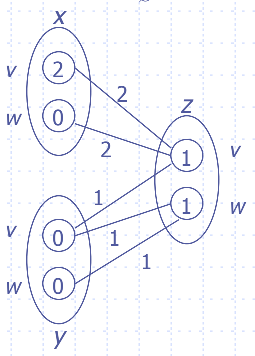


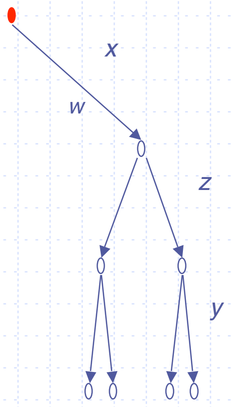
T=4
$C_\varnothing = 2$



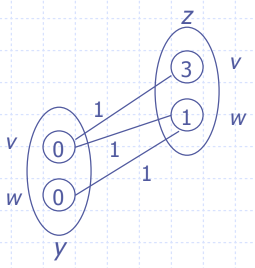


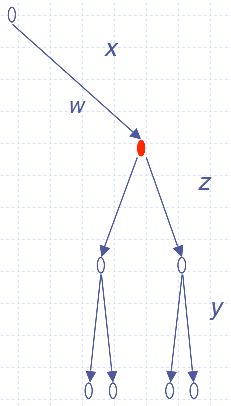
T=4
$C_{\varnothing} = 0$



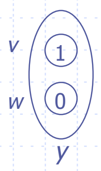


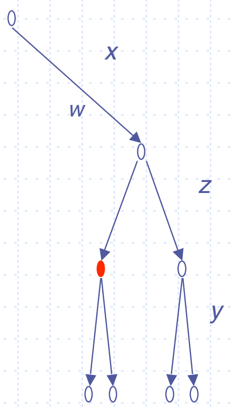
T=4
$C_{\varnothing} = 0$



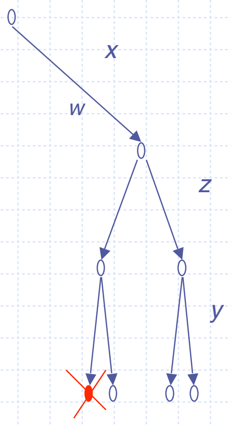


T=4
$C_{\varnothing} = 3$

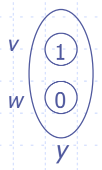


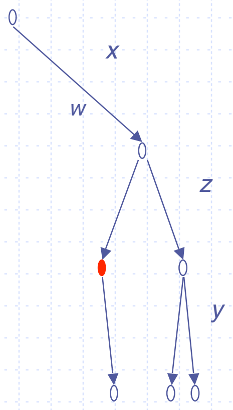


T=4
$C_{\varnothing} = 4$

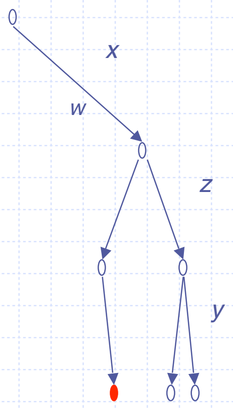


T=4
$C_{\emptyset} = 3$

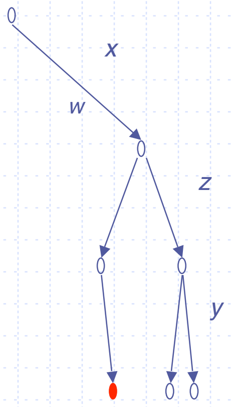




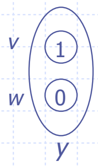
T=4
$C_{\emptyset} = 3$

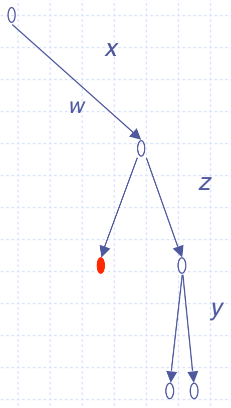


T=3
$C_{\varnothing}$ = 3



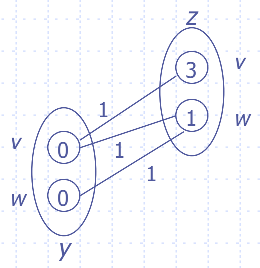
T=3
$C_{\varnothing}$ = 3

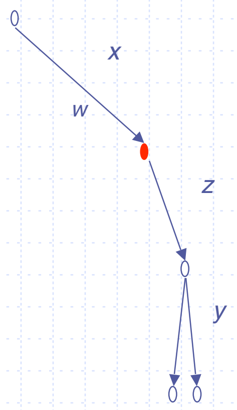




T=3

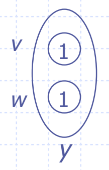
$C_{\varnothing} = 0$

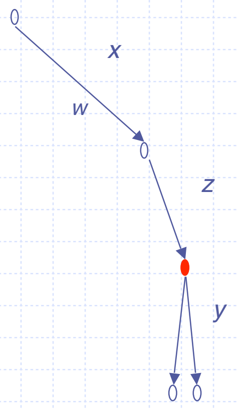




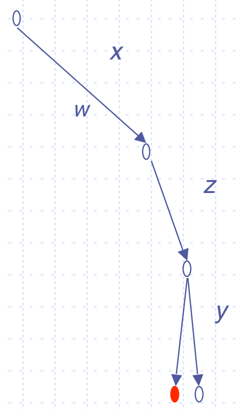
T=3

$C_{\varnothing} = 1$

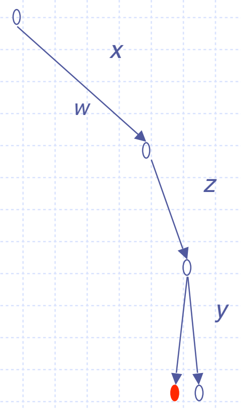


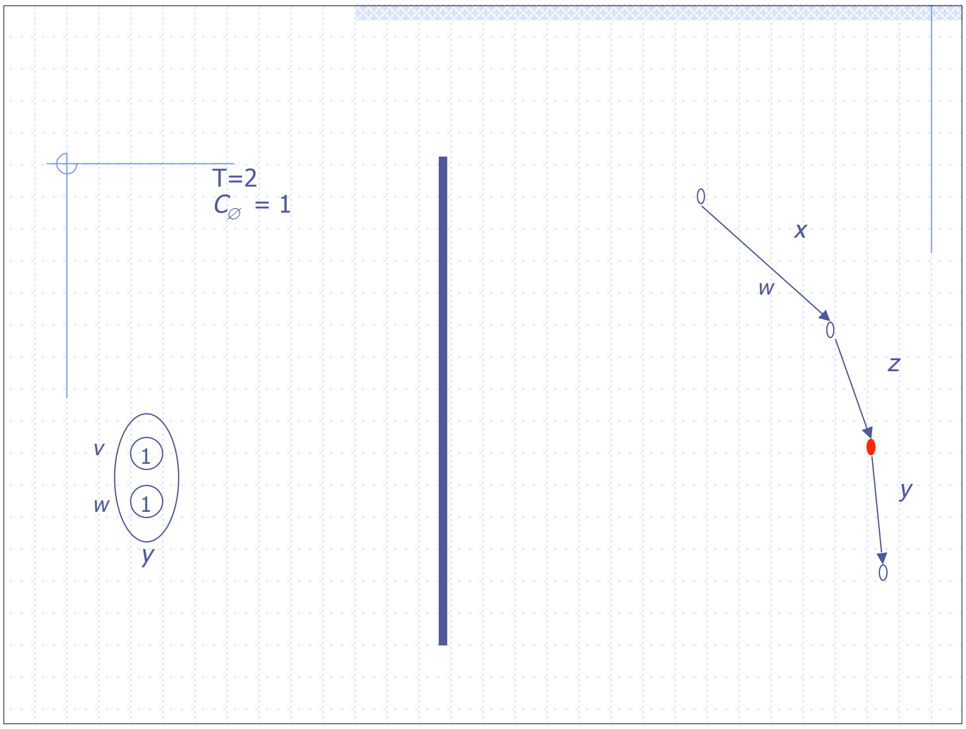


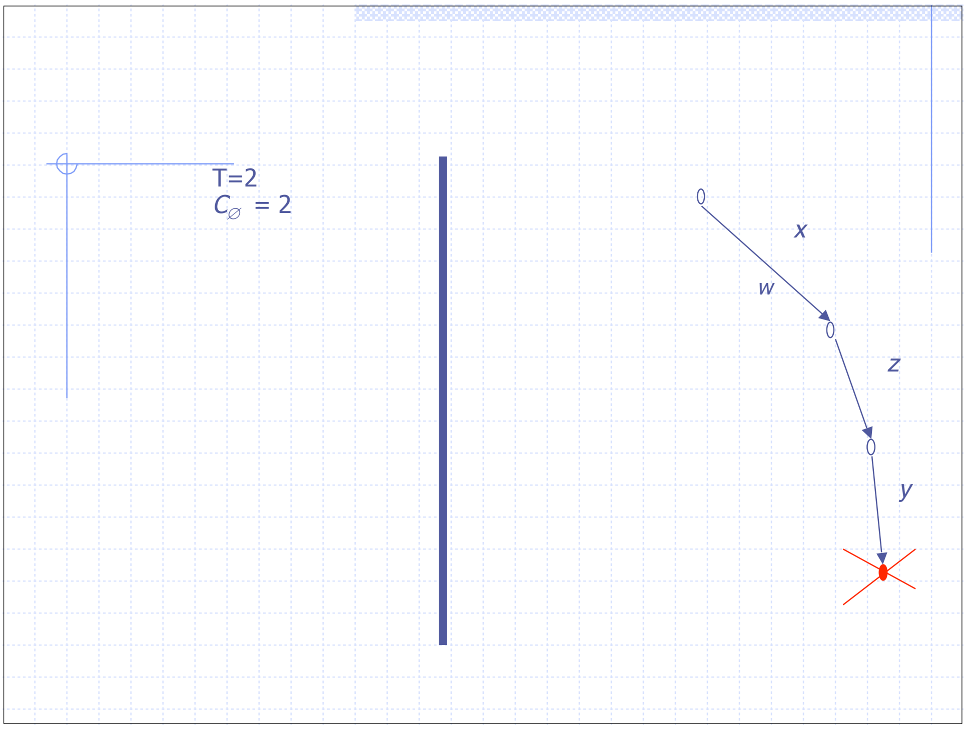
T=3
$C_{\varnothing} = 2$

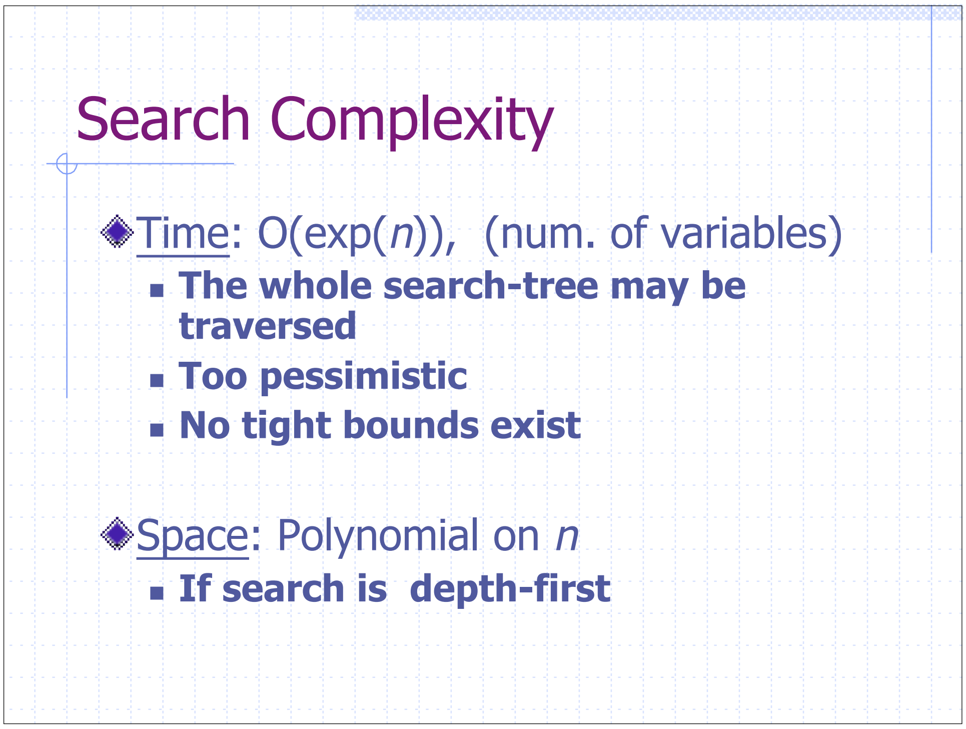


T=2
$C_{\varnothing} = 2$



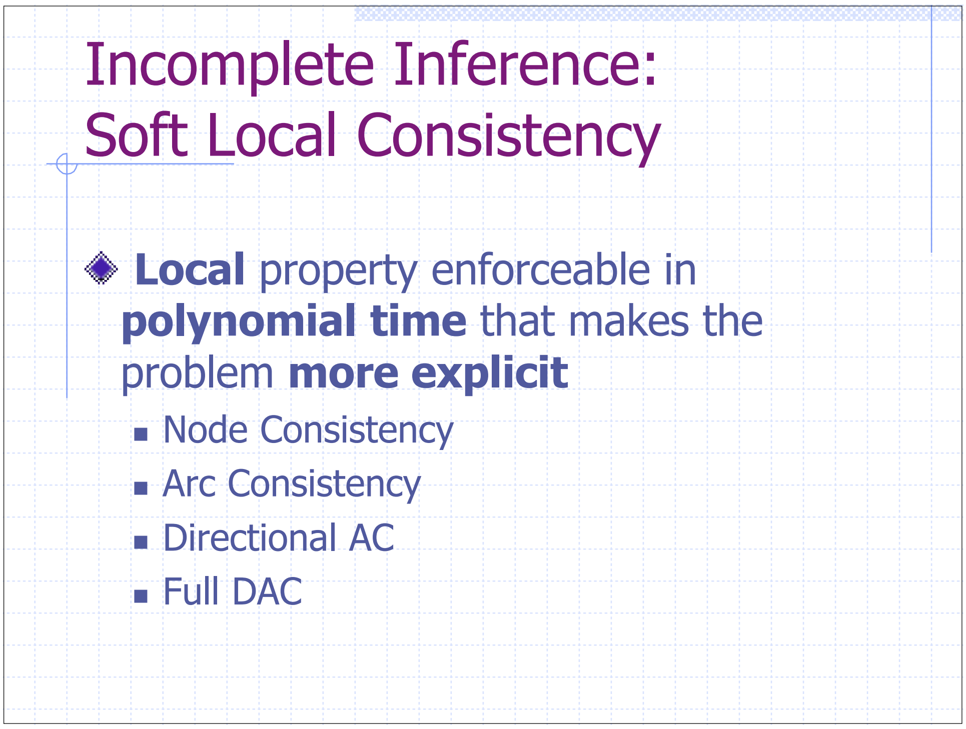

T=2
$C_\varnothing = 1$
v
w
1
1
y
0
x
w
0
z
y
0


T=2
$C_\varnothing = 2$
0
x
w
0
z
0
y



Search Complexity

- Time: O(exp(*n*)), (num. of variables)
  - **The whole search-tree may be traversed**
  - **Too pessimistic**
  - **No tight bounds exist**
- Space: Polynomial on *n*
  - **If search is depth-first**

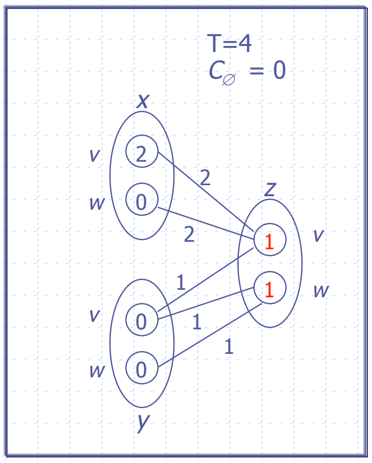


Incomplete Inference: Soft Local Consistency

- **Local** property enforceable in **polynomial time** that makes the problem **more explicit**
  - Node Consistency
  - Arc Consistency
  - Directional AC
  - Full DAC

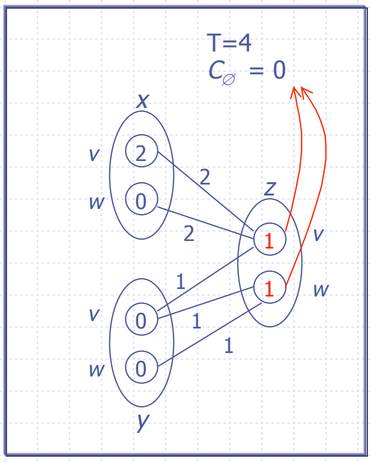
Node Consistency (NC*)

- For all variable *i*
  - ▯▯*a*, $C_\varnothing \oplus C_i\,(a)<T$
  - ▯▯ a, $C_i\,(a)= 0$



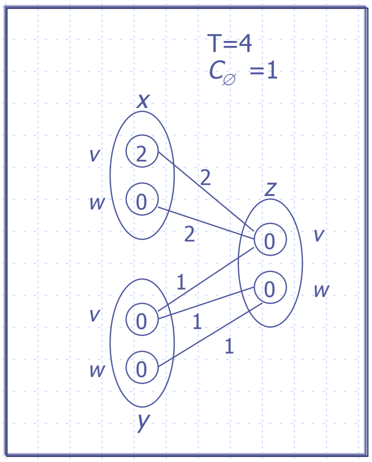
Node Consistency (NC*)

- For all variable *i*
  - ▯▯*a*, $C_\varnothing \oplus C_i\,(a)<T$
  - ▯▯ a, $C_i\,(a)= 0$



Node Consistency (NC*)

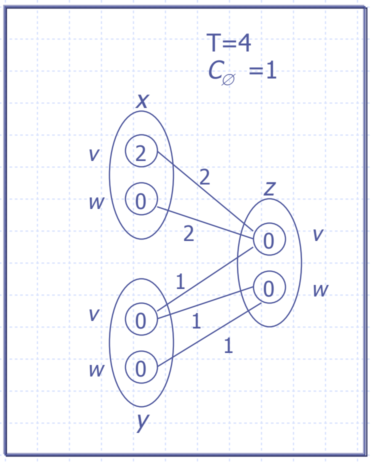
- For all variable *i*
  - $\forall a, C_\varnothing \oplus C_i(a) < T$
  - $\exists a, C_i(a) = 0$



Node Consistency (NC*)

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

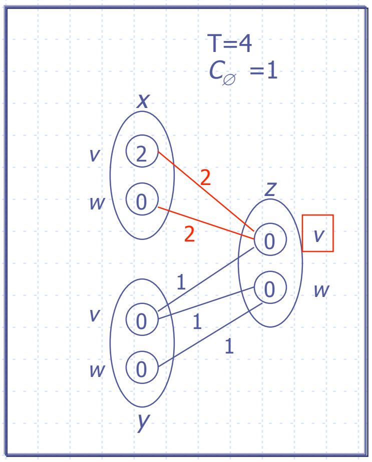
  O(*nd*)



Arc Consistency (AC*)

- NC*
- For all C_{ij}
  - $\forall a \exists b$
    $C_{ij}(a,b)=0$

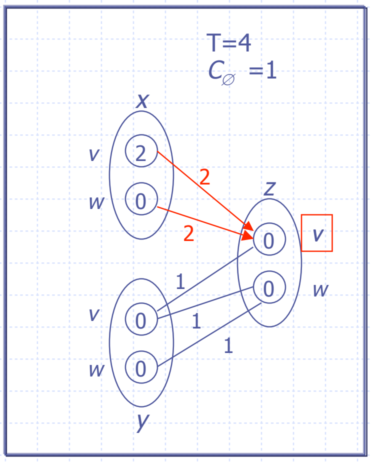
▯ *b* is a *support*



Arc Consistency (AC*)

- NC*
- For all C_{ij}
  - $\forall a \exists b$
    $C_{ij}(a,b)=0$

▯ *b* is a *support*



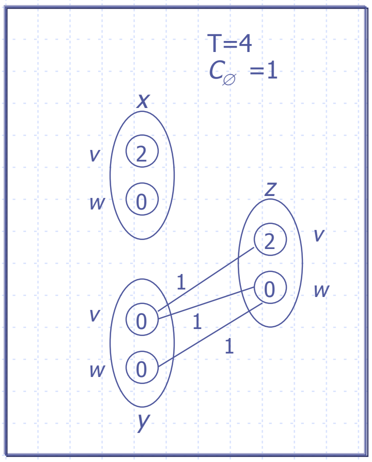
Arc Consistency (AC*)

- NC*
- For all C_{ij}

  □□*a* □ *b*

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

□ *b* is a *support*



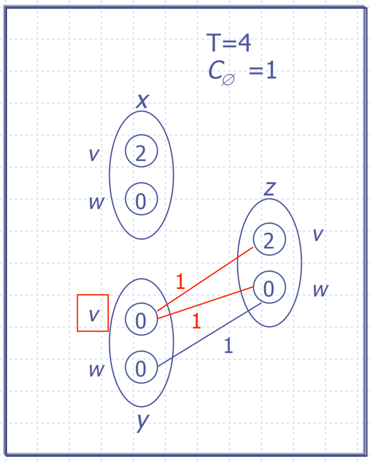
Arc Consistency (AC*)

- NC*
- For all C_{ij}

  □□*a* □ *b*

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

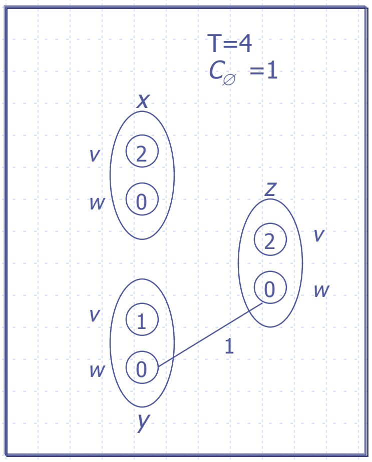
□ *b* is a *support*



Arc Consistency (AC*)

- NC*
- For all C_{ij}
  - $\forall a \; \exists b$
    - $C_{ij}(a,b) = 0$

b is a *support*

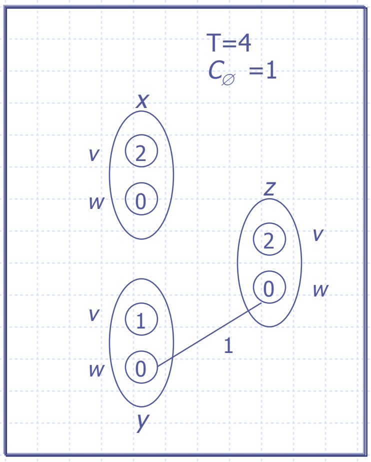


Arc Consistency (AC*)

- NC*
- For all C_{ij}
  - $\forall a \; \exists b$
    - $C_{ij}(a,b) = 0$

b is a *support*

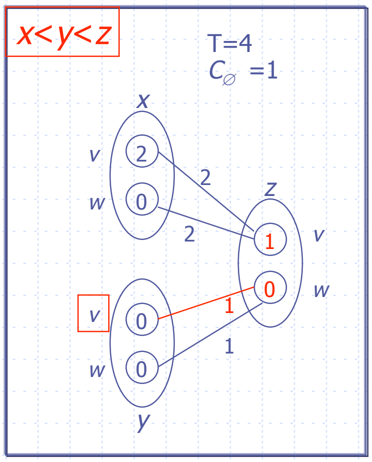
complexity:
$\mathbf{O}(n^2 d^3)$



Directional AC (DAC*)

- NC*
- For all C_{ij} $(i<j)$
  - $\forall a \exists b$
    $C_{ij}(a,b) \oplus C_j(b) = 0$

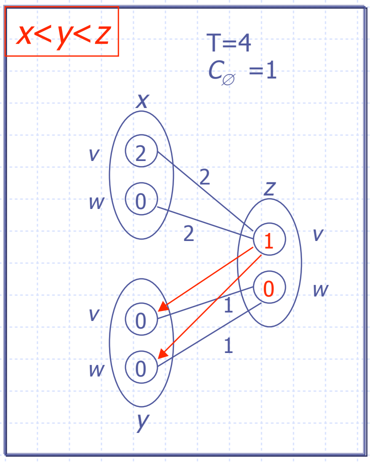
- b is a *full-support*



Directional AC (DAC*)

- NC*
- For all C_{ij} $(i<j)$
  - $\forall a \exists b$
    $C_{ij}(a,b) \oplus C_j(b) = 0$

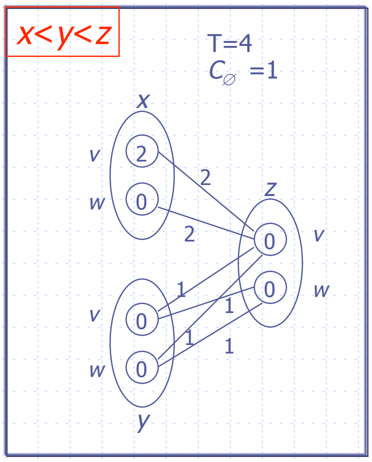
- b is a *full-support*



Directional AC (DAC*)

- NC*
- For all C_{ij} $(i<j)$
  - $\forall a \; \exists b$
    $C_{ij}(a,b) \oplus C_j(b) = 0$

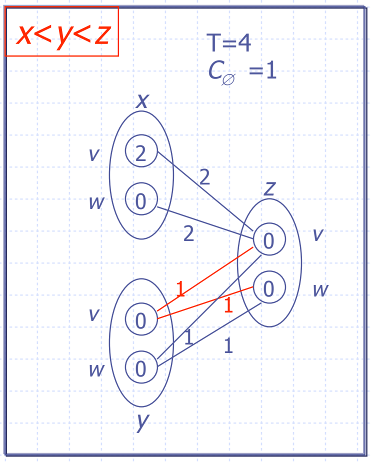
- *b* is a *full-support*



Directional AC (DAC*)

- NC*
- For all C_{ij} $(i<j)$
  - $\forall a \; \exists b$
    $C_{ij}(a,b) \oplus C_j(b) = 0$

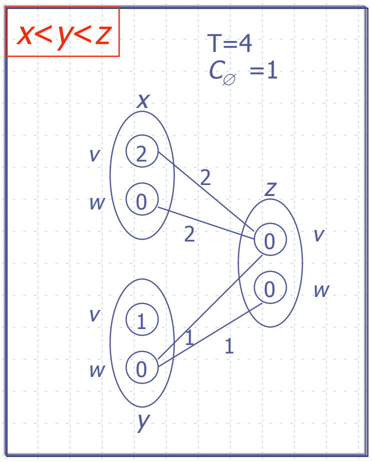
- *b* is a *full-support*



Directional AC (DAC*)

- NC*
- For all C_{ij} $(i<j)$
  - $\forall a \; \exists b$
    $C_{ij}(a,b) \oplus C_j(b) = 0$

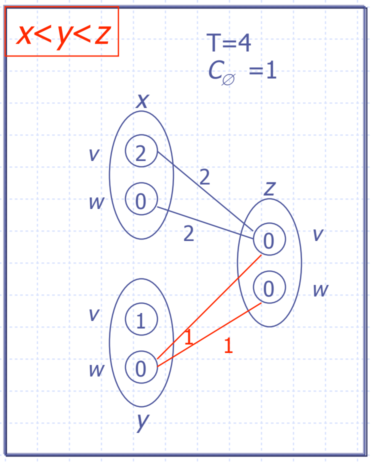
⇒ *b* is a *full-support*



Directional AC (DAC*)

- NC*
- For all C_{ij} $(i<j)$
  - $\forall a \; \exists b$
    $C_{ij}(a,b) \oplus C_j(b) = 0$

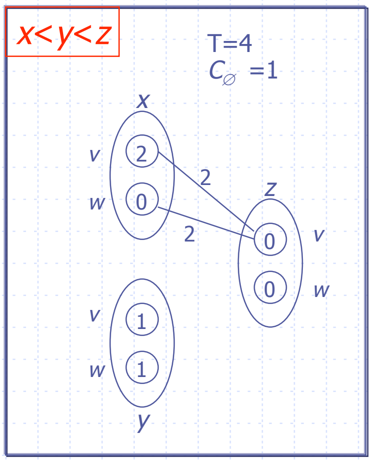
⇒ *b* is a *full-support*



Directional AC (DAC*)

- NC*
- For all C_{ij} ($i<j$)
  - $\forall a \exists b$ $C_{ij}(a,b) \oplus C_j(b) = 0$

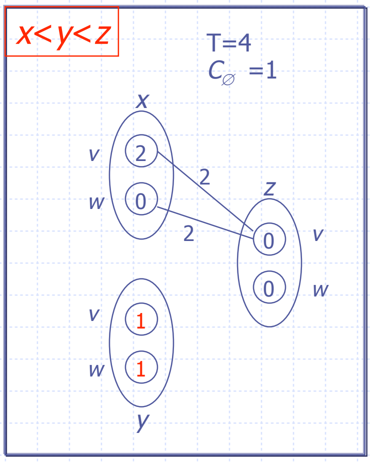
b is a *full-support*



Directional AC (DAC*)

- NC*
- For all C_{ij} ($i<j$)
  - $\forall a \exists b$ $C_{ij}(a,b) \oplus C_j(b) = 0$

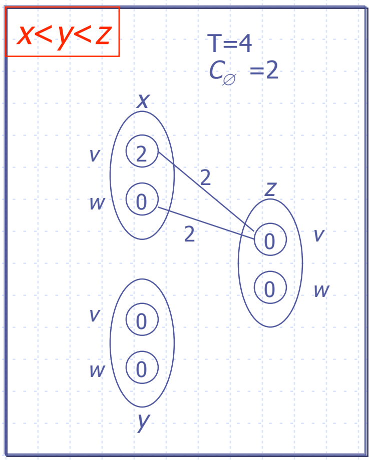
b is a *full-support*



Directional AC (DAC*)

- NC*
- For all C_{ij} $(i<j)$
  $\forall a \exists b$
  $C_{ij}(a,b) \oplus C_j(b) = 0$

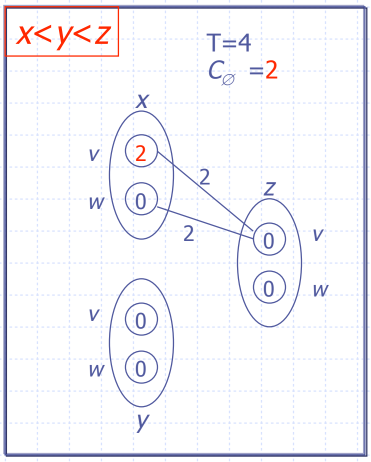
$\Rightarrow$ *b* is a *full-support*



Directional AC (DAC*)

- NC*
- For all C_{ij} $(i<j)$
  $\forall a \exists b$
  $C_{ij}(a,b) \oplus C_j(b) = 0$

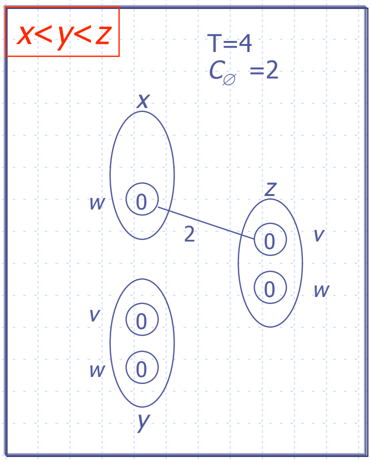
$\Rightarrow$ *b* is a *full-support*



Directional AC (DAC*)

- NC*
- For all C_{ij} $(i<j)$
  ▯▯a ▯ b
  $C_{ij}(a,b) \oplus C_j(b) = 0$

▯ *b* is a *full-support*

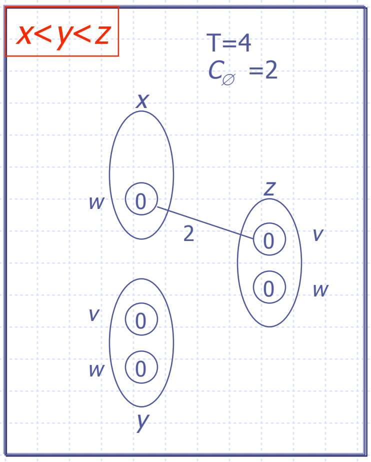


Directional AC (DAC*)

- NC*
- For all C_{ij} $(i<j)$
  ▯▯a ▯ b
  $C_{ij}(a,b) \oplus C_j(b) = 0$

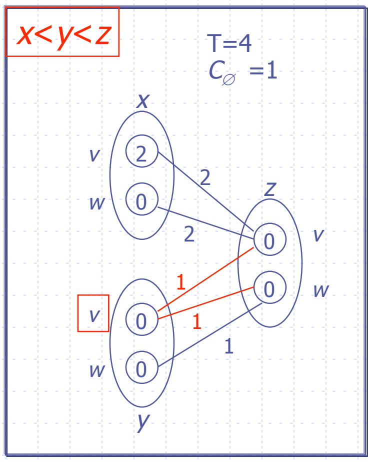
▯ *b* is a *full-support*

▯ complexity:
O(ed^2)



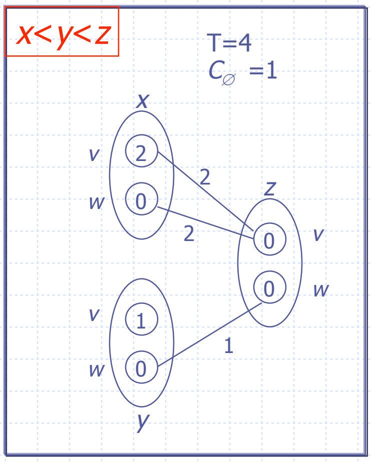
Full DAC (FDAC*)

- NC*
- For all C_{ij} $(i<j)$
  - ∀a ∃b
    $C_{ij}(a,b) \oplus C_j(b) = 0$
    (full support)
- For all C_{ij} $(i>j)$
  - ∀a ∃b,
    $C_{ij}(a,b) = 0$
    (support)



Full DAC (FDAC*)

- NC*
- For all C_{ij} $(i<j)$
  - ∀a ∃b
    $C_{ij}(a,b) \oplus C_j(b) = 0$
    (full support)
- For all C_{ij} $(i>j)$
  - ∀a ∃b
    $C_{ij}(a,b) = 0$
    (support)



Full DAC (FDAC*)

- NC*
- For all C_{ij} ($i<j$)

  □ □a □ b

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

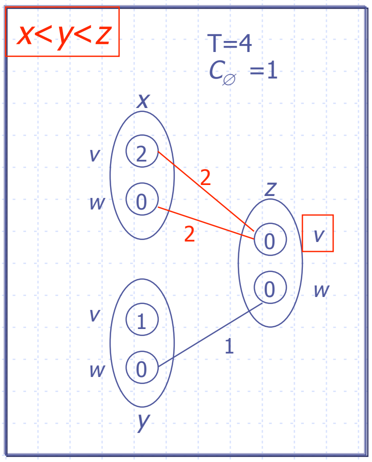
  (full support)

- For all C_{ij} ($i>j$)

  □ □a □ b

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

  (support)



Full DAC (FDAC*)

- NC*
- For all C_{ij} ($i<j$)

  □ □a □ b

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

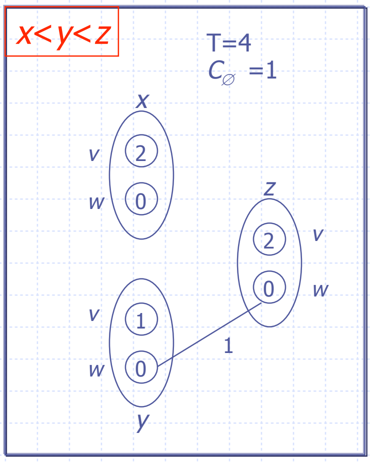
  (full support)

- For all C_{ij} ($i>j$)

  □ □a □ b

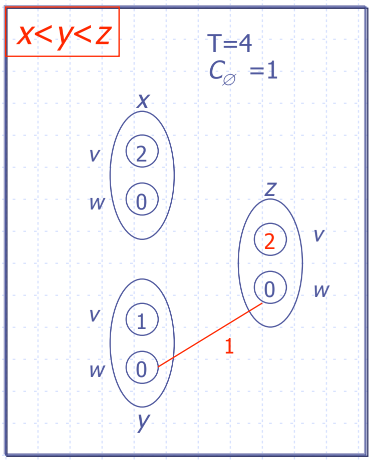
  $C_{ij}(a,b) = 0$

  (support)



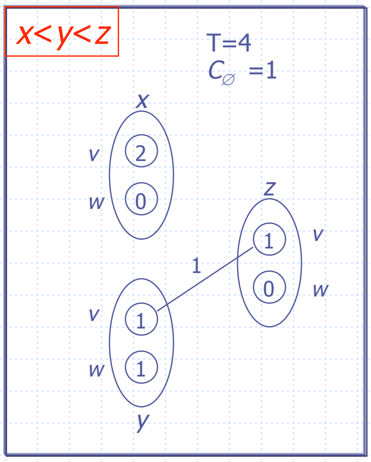
Full DAC (FDAC*)

- NC*
- For all C_{ij} $(i<j)$
   a b
  $C_{ij}(a,b) \oplus C_j(b) = 0$
  (full support)
- For all C_{ij} $(i>j)$
   a b
  $C_{ij}(a,b) = 0$
  (support)



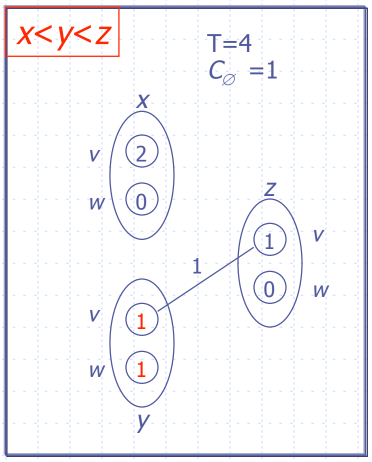
Full DAC (FDAC*)

- NC*
- For all C_{ij} $(i<j)$
   a b
  $C_{ij}(a,b) \oplus C_j(b) = 0$
  (full support)
- For all C_{ij} $(i>j)$
   a b
  $C_{ij}(a,b) = 0$
  (support)



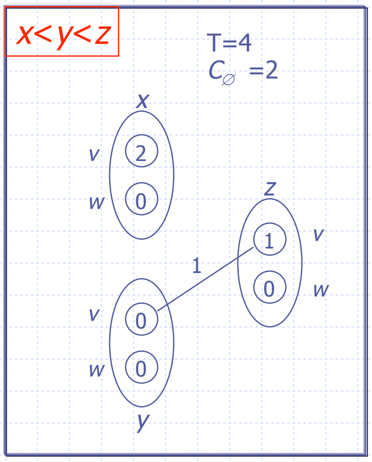
Full DAC (FDAC*)

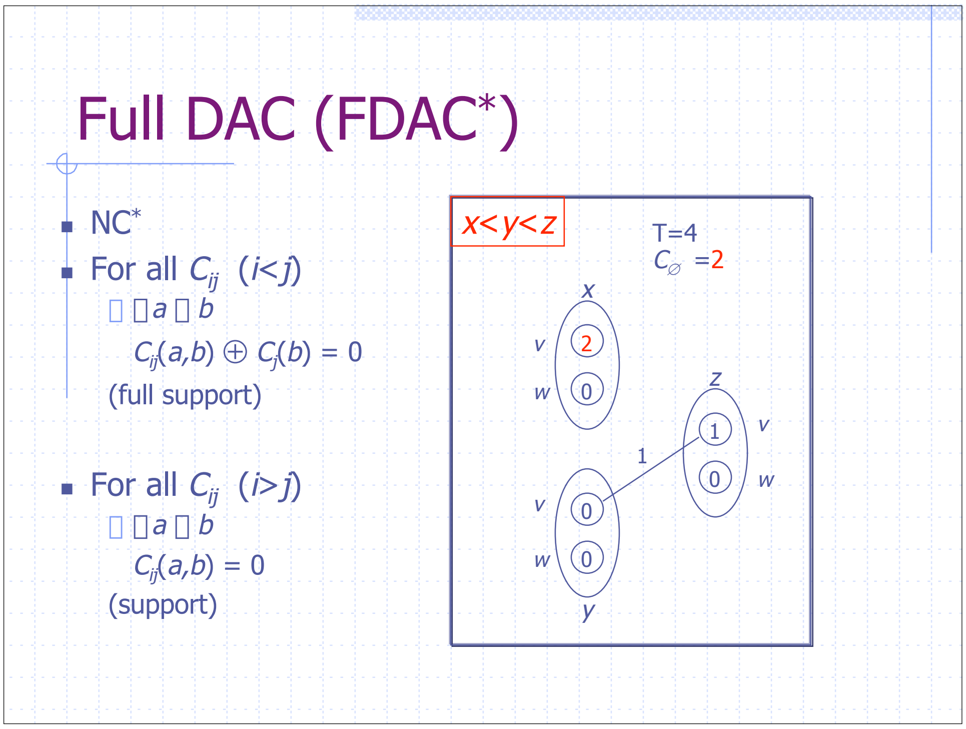
- NC*
- For all C_{ij} $(i<j)$
  - $\forall a \exists b$
    $C_{ij}(a,b) \oplus C_j(b) = 0$
    (full support)
- For all C_{ij} $(i>j)$
  - $\forall a \exists b$
    $C_{ij}(a,b) = 0$
    (support)



Full DAC (FDAC*)

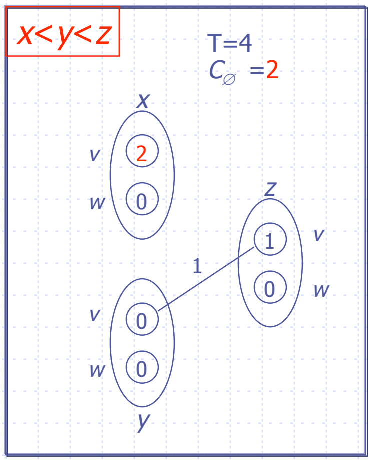
- NC*
- For all C_{ij} $(i<j)$
  - $\forall a \exists b$
    $C_{ij}(a,b) \oplus C_j(b) = 0$
    (full support)
- For all C_{ij} $(i>j)$
  - $\forall a \exists b$
    $C_{ij}(a,b) = 0$
    (support)

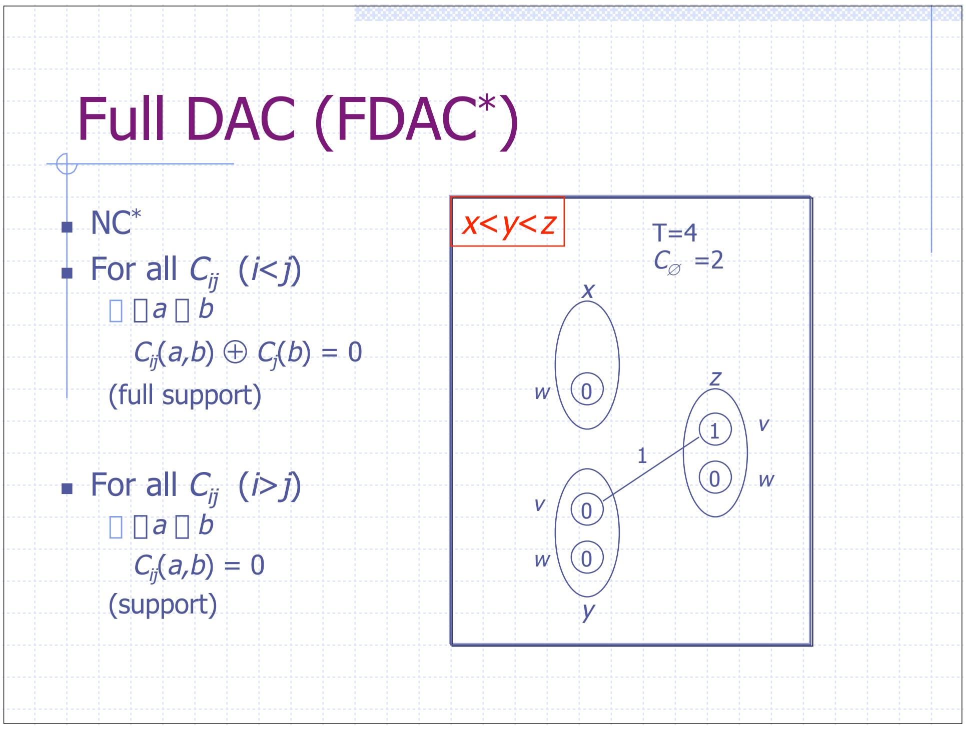




Full DAC (FDAC*)

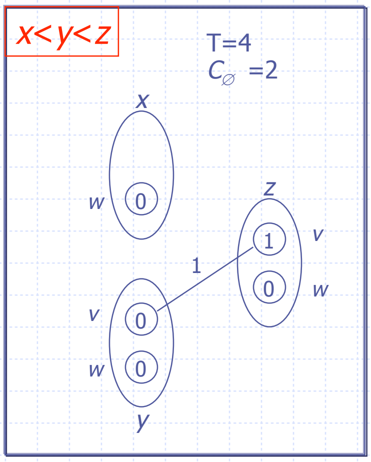
- NC*
- For all C_{ij} $(i<j)$
  - $\forall a \exists b$
    $C_{ij}(a,b) \oplus C_j(b) = 0$
    (full support)
- For all C_{ij} $(i>j)$
  - $\forall a \exists b$
    $C_{ij}(a,b) = 0$
    (support)





Full DAC (FDAC*)

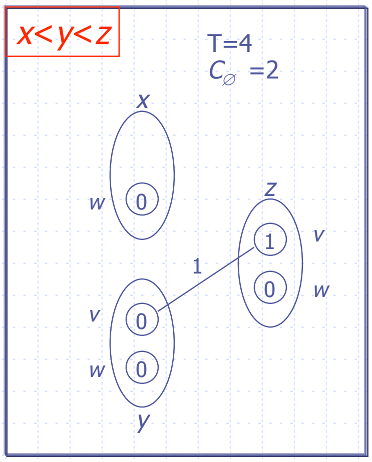
- NC*
- For all C_{ij} $(i<j)$
  - $\forall a \exists b$
    $C_{ij}(a,b) \oplus C_j(b) = 0$
    (full support)
- For all C_{ij} $(i>j)$
  - $\forall a \exists b$
    $C_{ij}(a,b) = 0$
    (support)



Full DAC (FDAC*)

- NC*
- For all C_{ij} ($i<j$)
  - $\forall a \exists b$
    $C_{ij}(a,b) \oplus C_j(b) = 0$
    (full support)

- For all C_{ij} ($i>j$)
  - $\forall a \exists b$
    $C_{ij}(a,b) = 0$
    (support)
- complexity:
  $\mathbf{O}(ed^3)$



Hierarchy

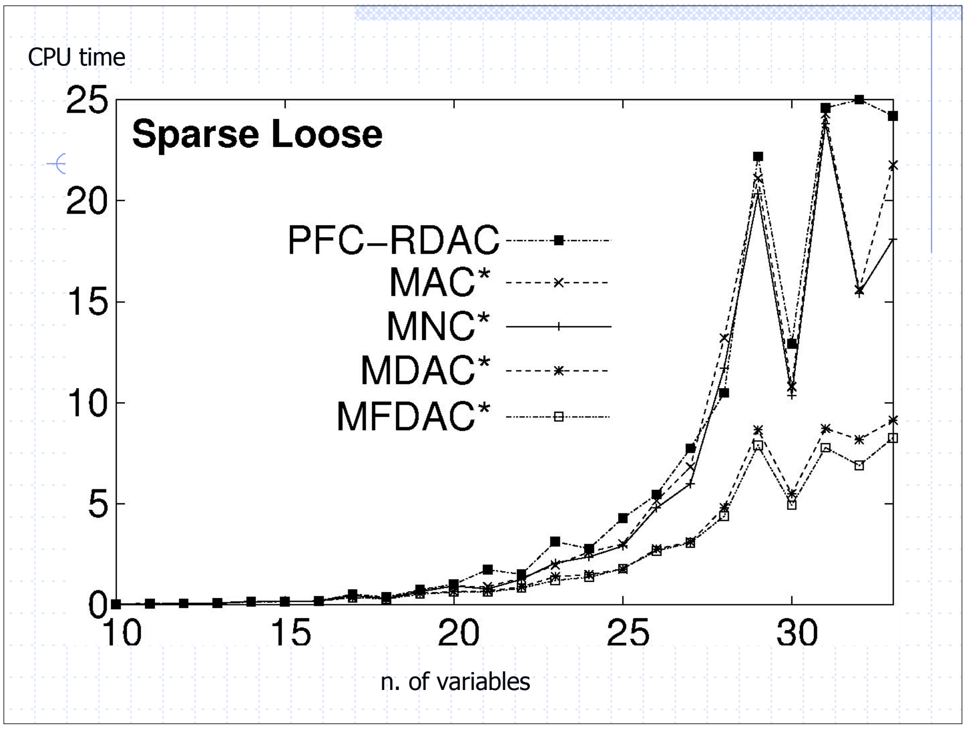
$$\begin{tikzcd}
 & \text{NC}^* \ \mathbf{O}(nd) \arrow[dl] \arrow[dr] & \\
\text{AC}^* \ \mathbf{O}(n^2d^3) \arrow[dr] & & \text{DAC}^* \ \mathbf{O}(ed^2) \arrow[dl] \\
 & \text{FDAC}^* \ \mathbf{O}(ed^3) &
\end{tikzcd}$$

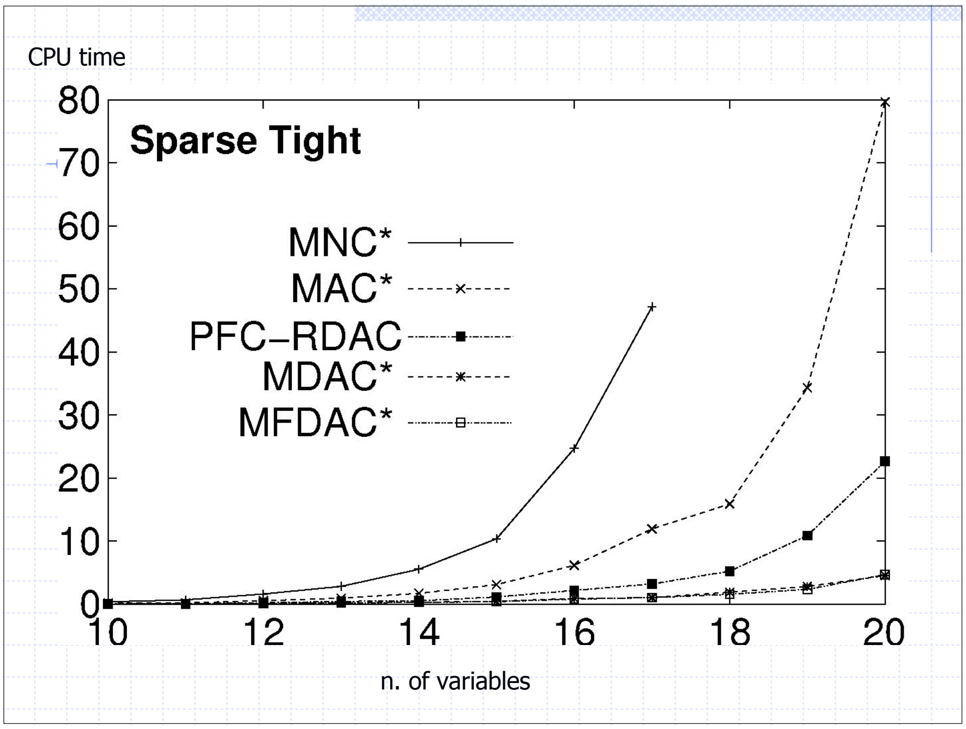
Hybrid: search+local consist.

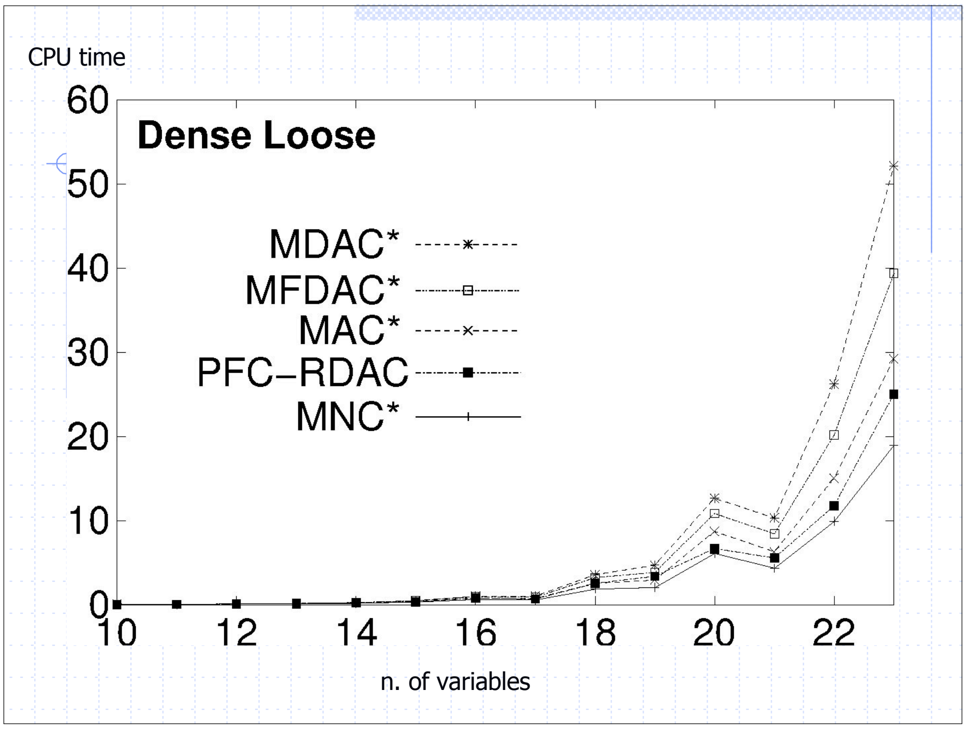
- WCSPs are solved with search:
  - Lower Bound $\geq$ Upperbound => Backtrack
- Each node is a WCSP subproblem
  - T : Upper Bound (best known solution)
  - $C_{\varnothing}$: Lower Bound
- Algorithm: maintain local consistency during search
  - MNC, MAC, MDAC, MFDAC

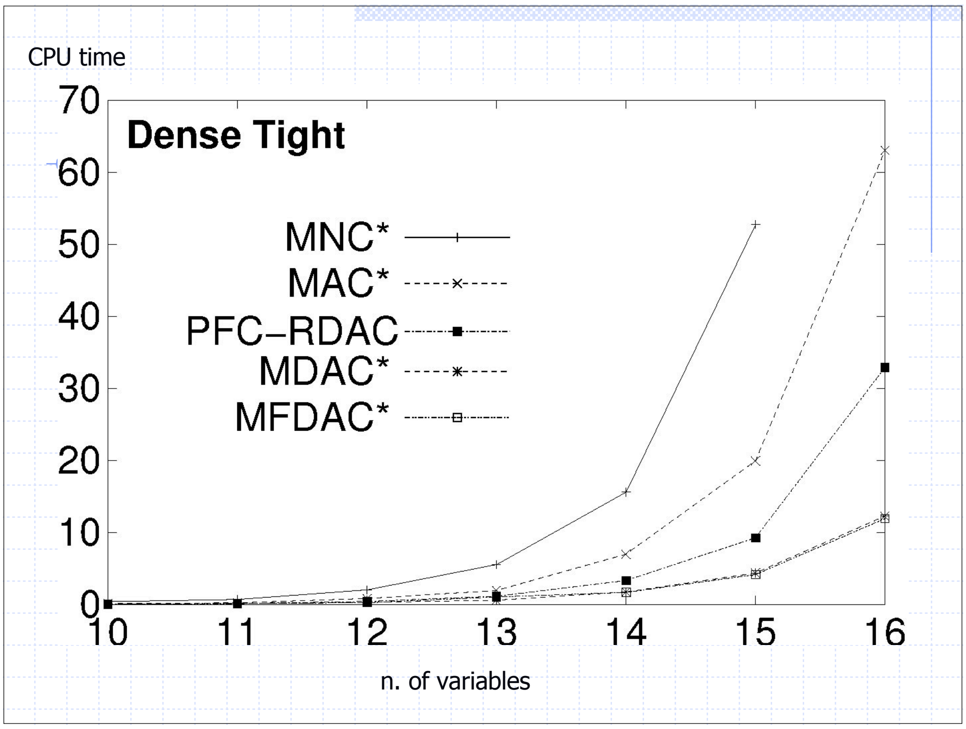
Experiments

- Overconstrained Random CSPs


CPU time
Sparse Loose
PFC−RDAC
MAC*
MNC*
MDAC*
MFDAC*
25
20
15
10
5
0
10
15
20
25
30
n. of variables


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


CPU time
Dense Loose
MDAC*
MFDAC*
MAC*
PFC−RDAC
MNC*
0
10
20
30
40
50
60
10
12
14
16
18
20
22
n. of variables


CPU time
Dense Tight
MNC*
MAC*
PFC−RDAC
MDAC*
MFDAC*
0
10
20
30
40
50
60
70
10
11
12
13
14
15
16
n. of variables

Complete Inference: Bucket Elimination

- Backtracking-free approach
- Sequence of problem reductions that preserve the best solution
- *Bucket Elimination* (BE) **[Dechter 99]**
  - **Variables are eliminated one at a time**
  - **When no variable remains, the problem is trivially solved**
- This approach has been rediscovered once and again **[Bertele and Brioschi 72]**

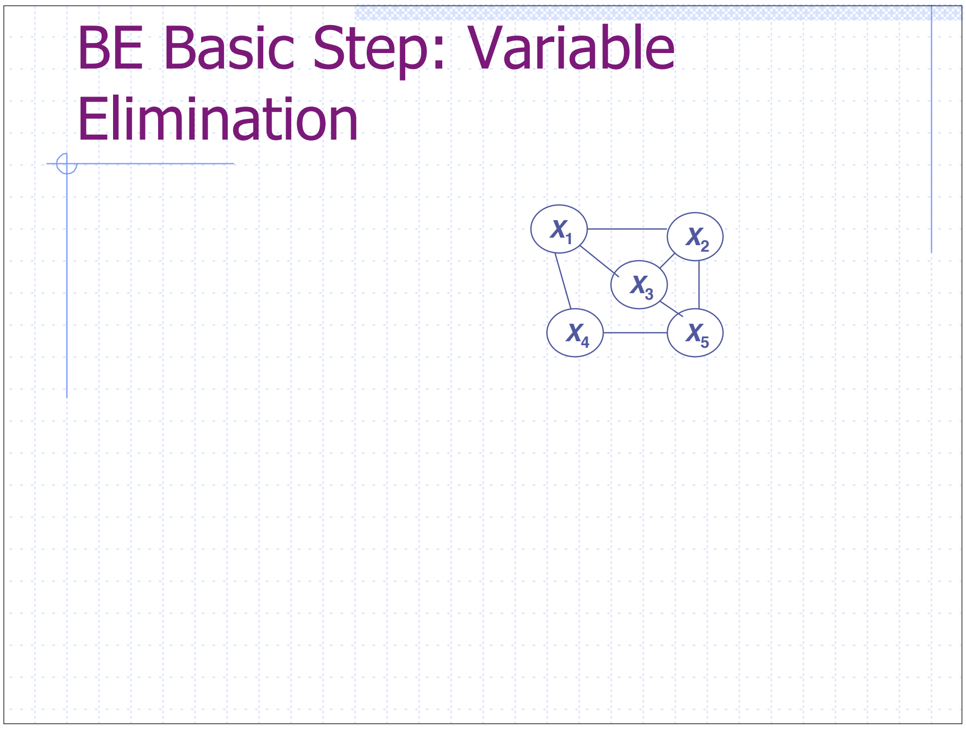
Bucket Elimination (BE)

- Two primitive operators:
  - **Sum of functions** $(f+g)$
  - **Elimination of a variable** $elim_i(f)$

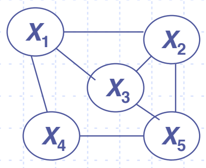
$$f(x_1,x_2)=x_1+x_2,\quad g(x_2,x_3)=x_2x_3$$

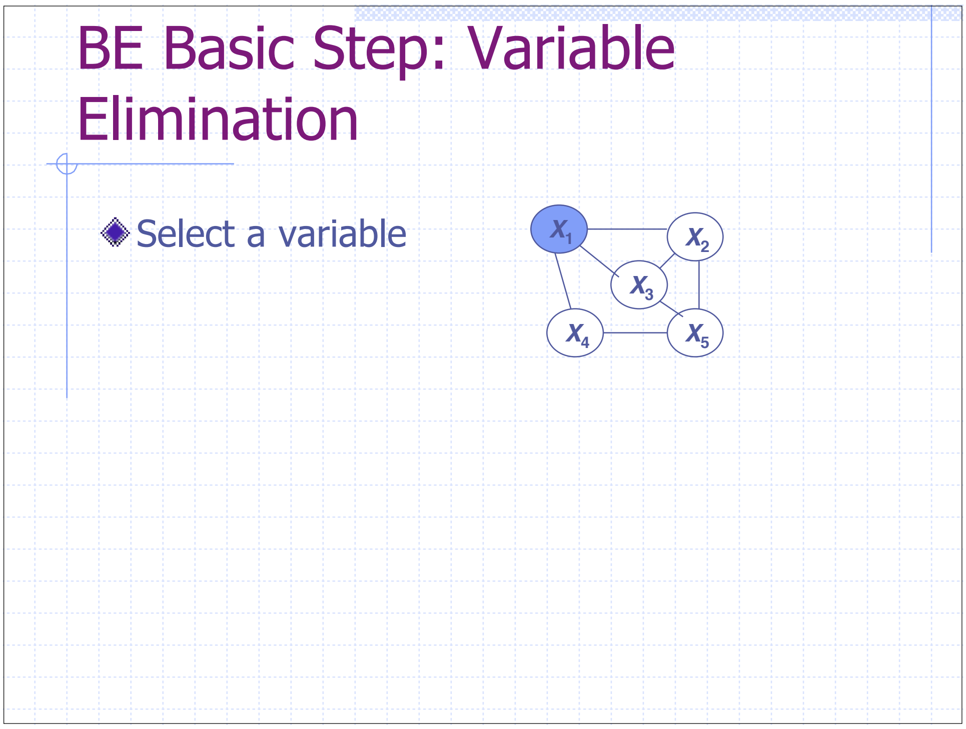
- e.g.: $(f+g)(x_1,x_2,x_3)=x_1+x_2+x_2x_3$

$$elim_1(f)(x_2)=\min_{a\in D_1}\{f(a,x_2)\}$$



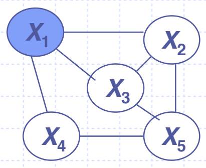
BE Basic Step: Variable Elimination





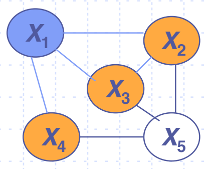
BE Basic Step: Variable Elimination

- Select a variable



BE Basic Step: Variable Elimination

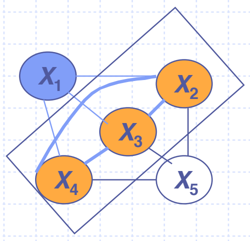
- Compute its *bucket*
- Bucket: set of functions that *mention* the variable

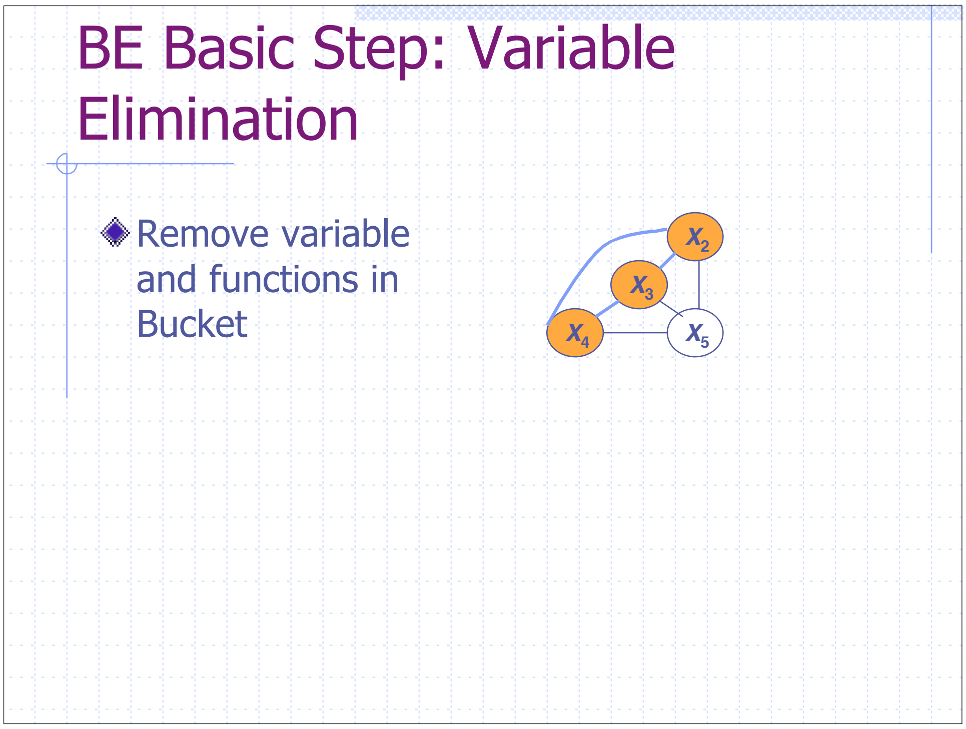


BE Basic Step: Variable Elimination

- Compute new function

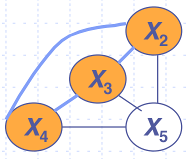
$$g = \mathrm{elim}_1\left(\prod_{f \in Bucket} f\right)$$

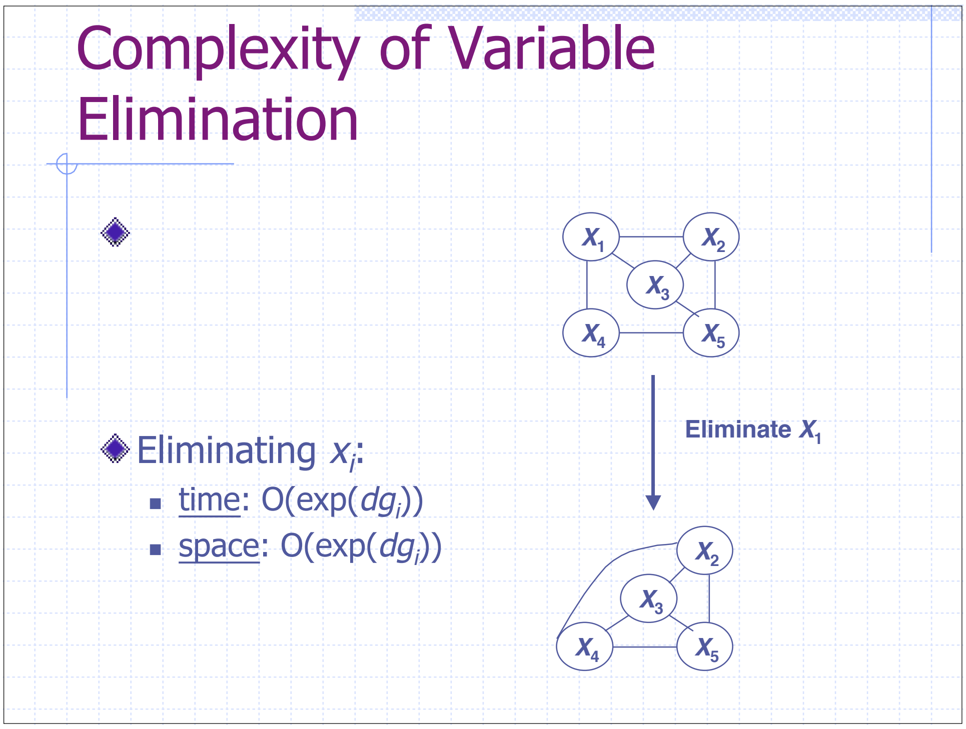




BE Basic Step: Variable Elimination

- Remove variable and functions in Bucket

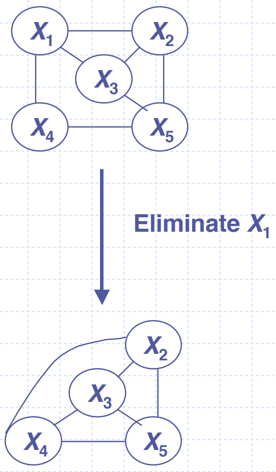




Complexity of Variable Elimination

-

- Eliminating x_i:
  - time: O(exp(dg_i))
  - space: O(exp(dg_i))

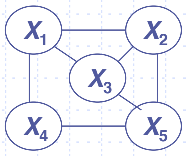


BE: complexity

- time: O(exp(w^*))
- space: O(exp(w^*))
- $w^* \le n$
- these bounds are tight
- the space complexity renders BE infeasible as a general method

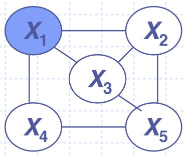
Hybrid: search + complete inference

Search Basic Step: Variable Branching

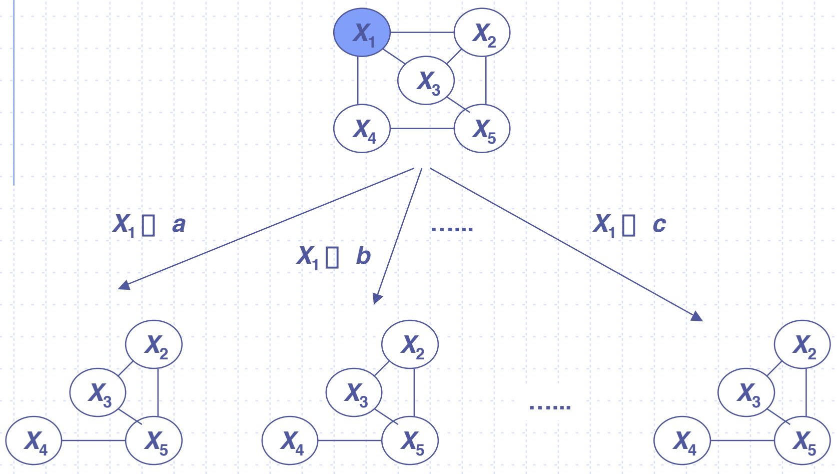


Search Basic Step: Variable Branching

- Select a variable



Search Basic Step: Variable Branching



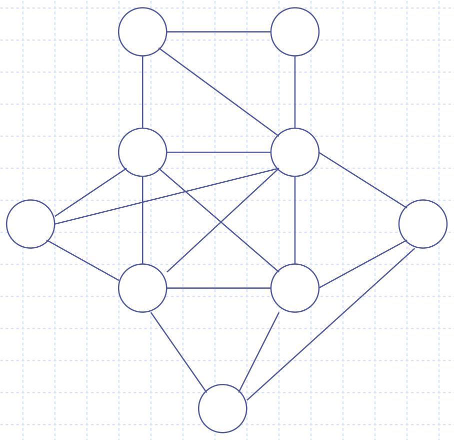
Hybrid: search + complete inference

- Idea:
  - **Select a variable**
  - **If it is not too costly, then eliminate it**
  - **Else let search take care of it**
- Two examples:
  - **BE-BB(*k*) [Larrosa and Dechter, 2001]**
  - **SBE(*k*) [Dechter and El Fattah 2000, Kask *et al* 2001]**
- *k* is a control parameter
  - ***k* small, more search**
  - ***k* large, more variable elimination**

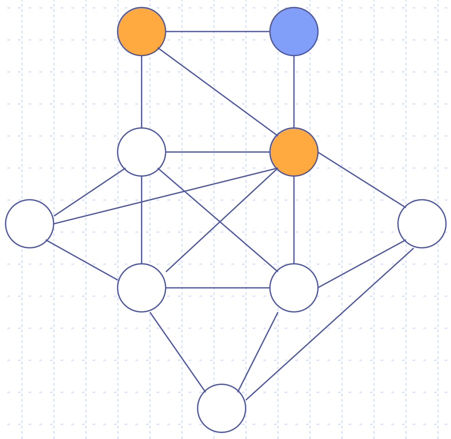
BE-BB(k)

- At each node:
  - **x_i □ select a future variable**
  - if **dg(x_i)** □ ***k*** then **eliminate x_i**
  - else **branch on the values of x_i**
- Property:
  - **BE-BB(-1) is BB**
  - **BE-BB(w*) is BE**

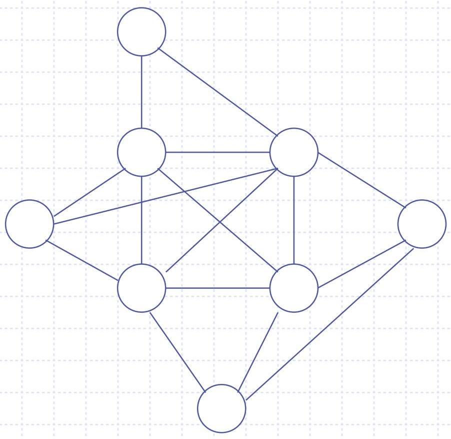
BE-BB(2): example



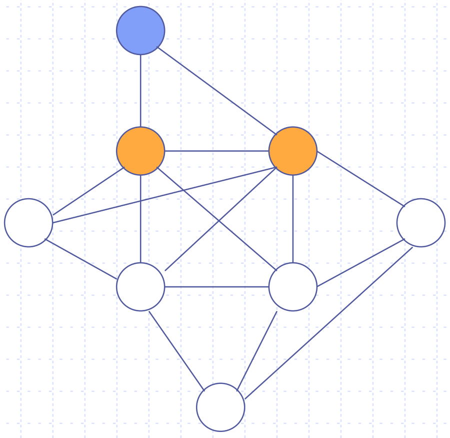
BE-BB(2): example



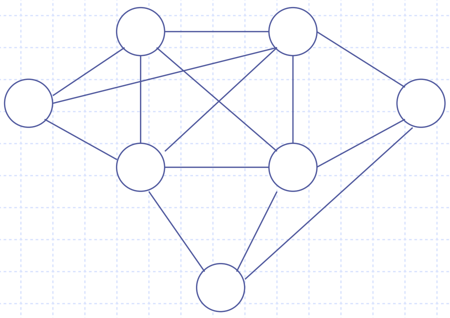
BE-BB(2): example



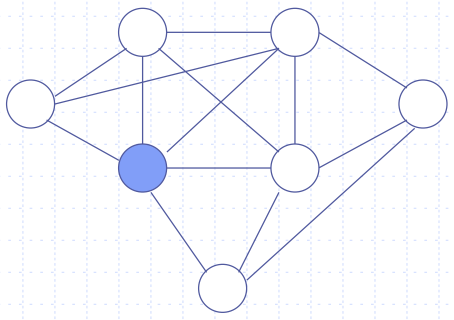
BE-BB(2): example



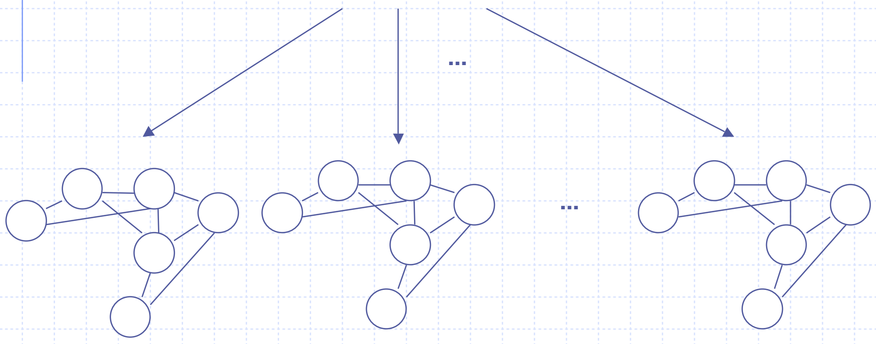
BE-BB(2): example



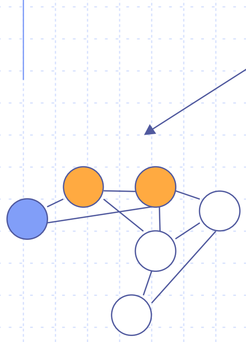
BE-BB(2): example



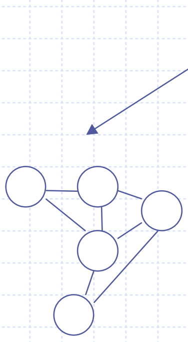
BE-BB(2): example



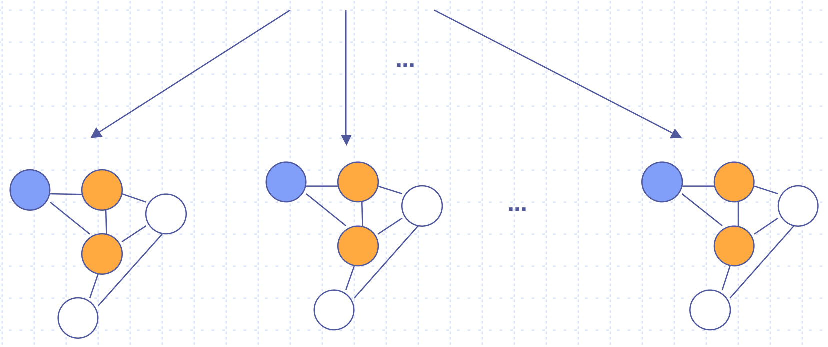
BE-BB(2): example



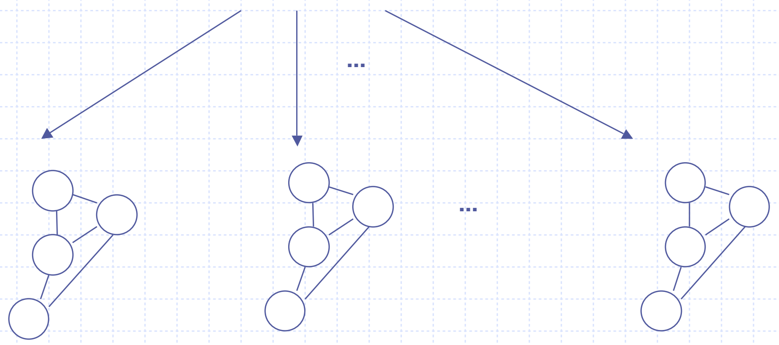
BE-BB(2): example



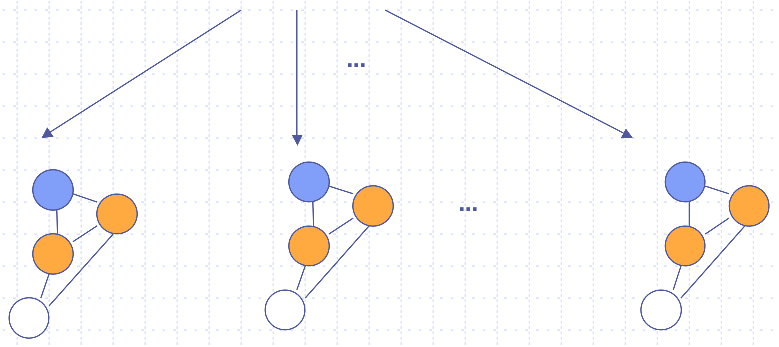
BE-BB(2): example



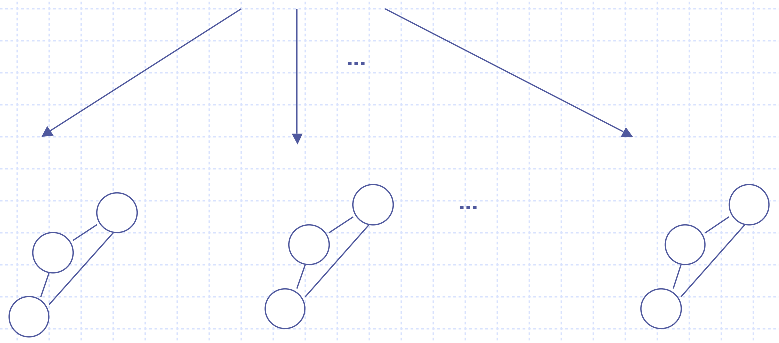
BE-BB(2): example



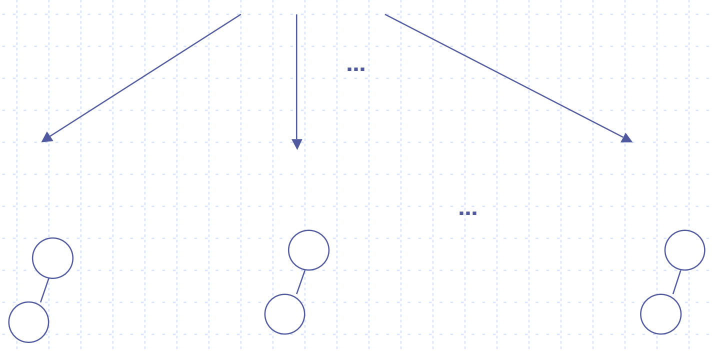
BE-BB(2): example



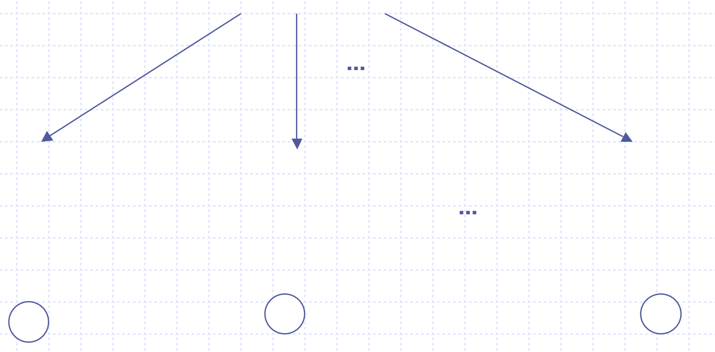
BE-BB(2): example



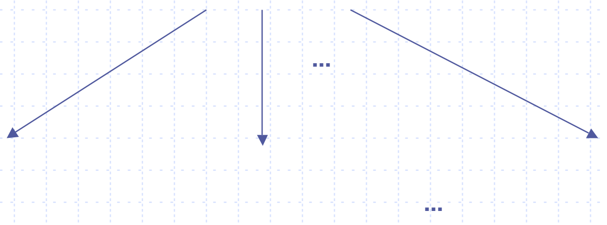
BE-BB(2): example



BE-BB(2): example



BE-BB(2): example

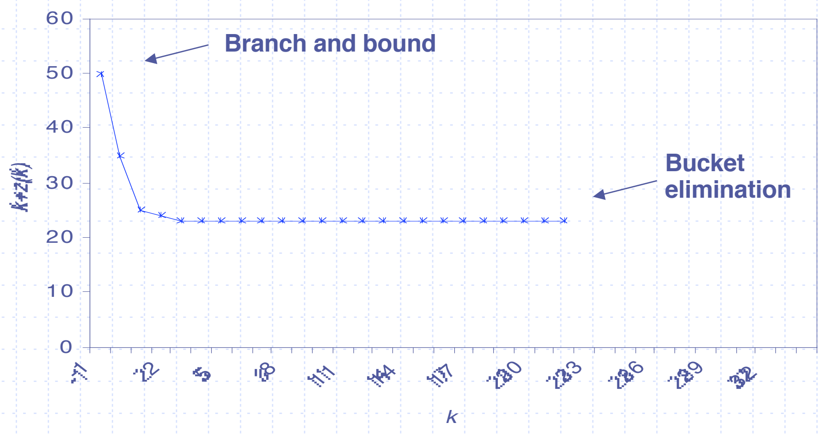


BE-BB(k): complexity

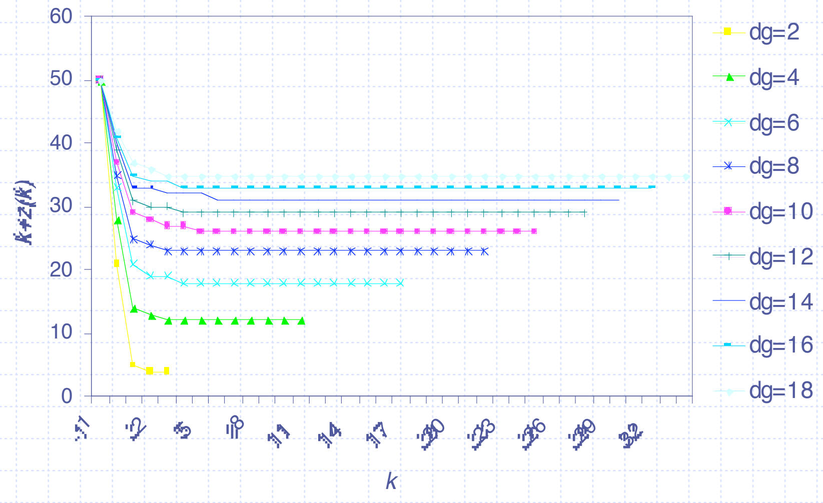
- Space: $O(\exp(k))$
- Time: $O(\exp(k+z(k)))$
  - ***z*(*k*): number of branched variables**
  - ***z*(*k*): it can be computed out of the *k*-restricted induced graph G*(k,o)**

Empirical Evaluation (time bounds)

• Random Graphs (50 nodes, 200 edges, average degree 8, *w**≈23)



Empirical Evaluation (time bounds)



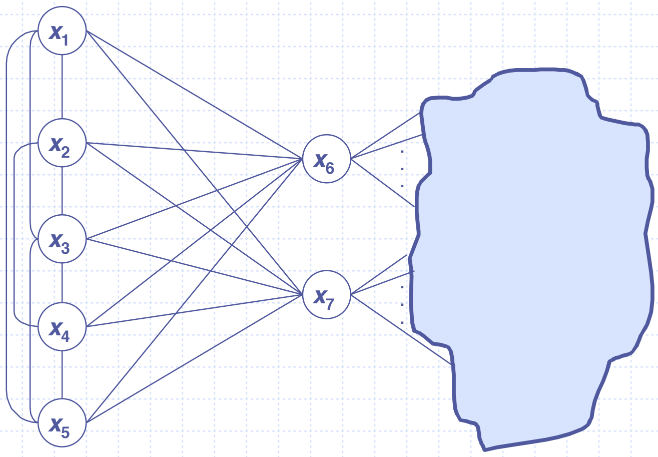
Empirical Evaluation (CPU time)

k	*n=30, r=5, dg=6*	*n=35, r=5, dg=6*	*n=20, r=5, dg=7*	*n=40, r=2, dg=4*
-1	49.0	107.5	45.3	84.9
0	6.1	27.5	38.8	63.2
1	2.5	11.2	31.1	26.5
2	1.6	4.3	15.9	6.8
3	.9	3.7	8.8	6.0
4	.5	2.1	11.5	8.7
5	2.6	6.2	46.3	29.6
6	3.2	9.6	89.8	131.3

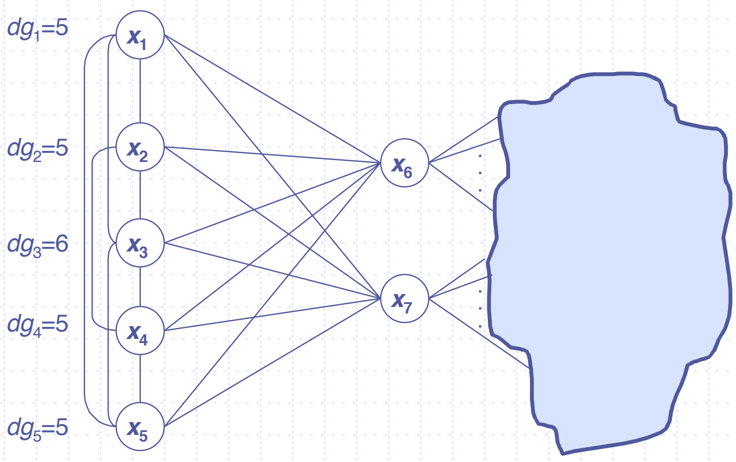
Super-Bucket Elimination, SBE(k)

- Eliminate *sets of variables* such that:
  - **individual eliminations are too costly in space (namely, each variable in the set has degree larger than *k*)**
  - **the join degree is lower than *k***

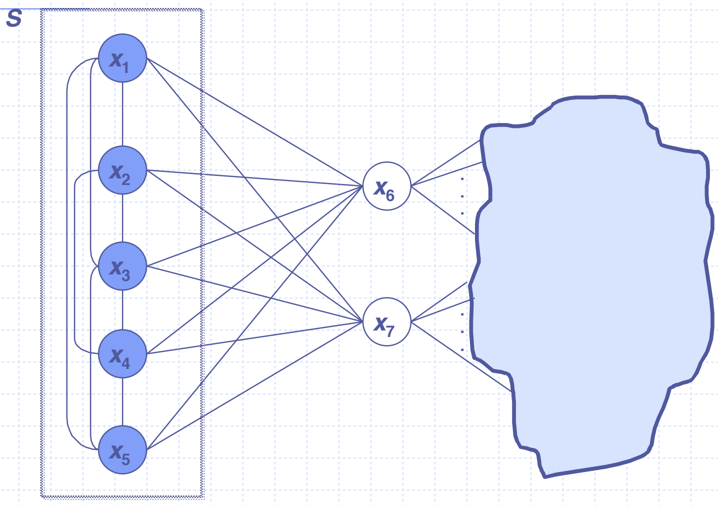
SBE(2): example



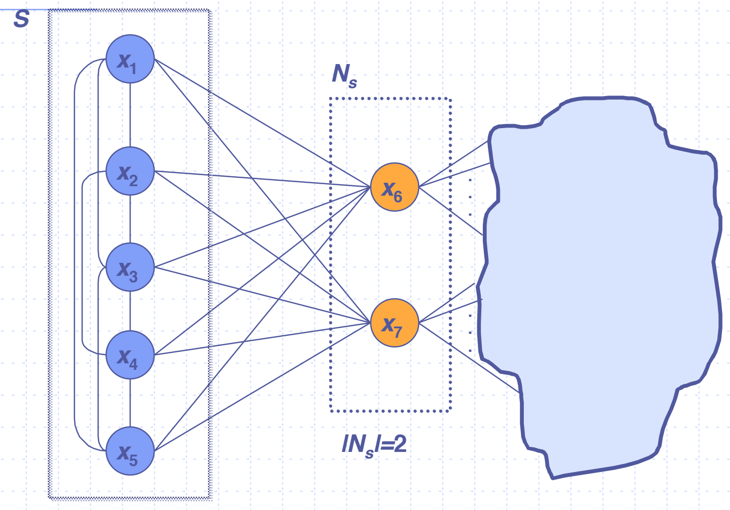
SBE(2): example



SBE(2): example

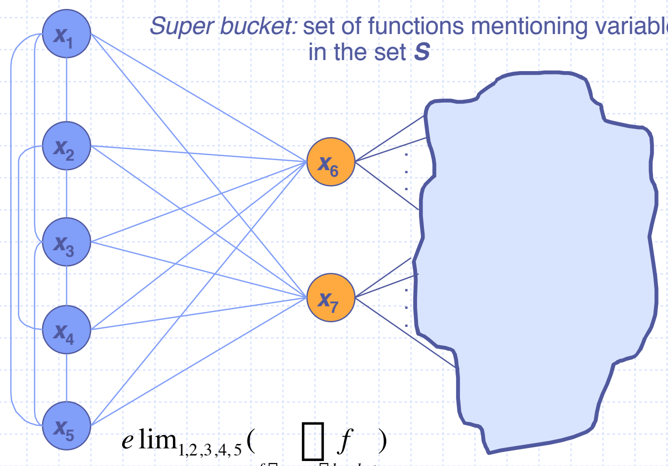


SBE(2): example



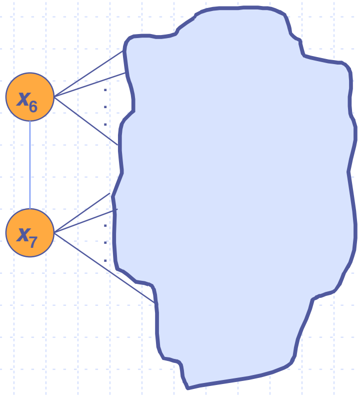
SBE(2): example

Super bucket: set of functions mentioning variables in the set ***S***



$$elim_{1,2,3,4,5}\left(\prod_{f \in super\_bucket} f\right)$$

SBE(2): example



SBE(*k*)

- Each super-bucket elimination is a set of COP instances that can be solved with BB!!
- e.g.:

$$f(x_1,x_3,x_4), g(x_3,x_4,x_5), h(x_2,x_3,x_5),$$

$$elim_{1,2,3}(f+g+h)(x_4,x_5)$$

$D_4 \times D_5$ optimization problems

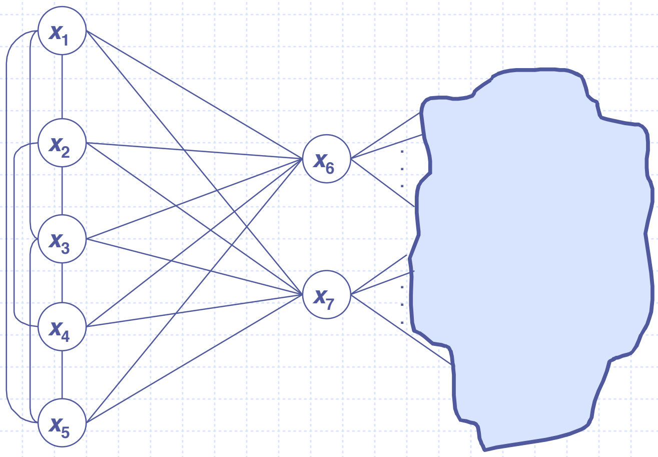
SBE(k)

- Repeat:

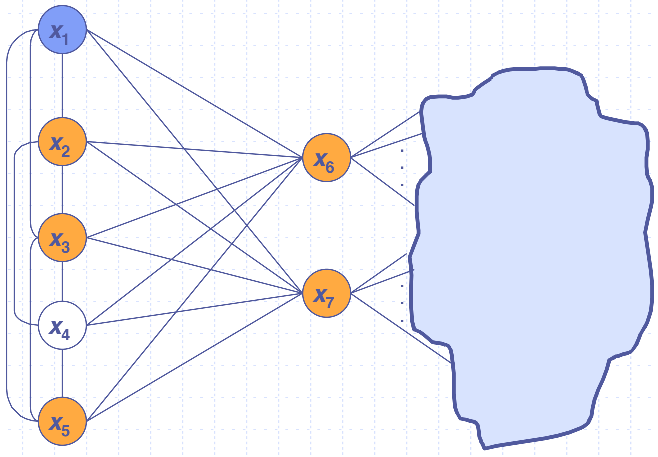
```
S ← {x_i}, future variable
while |N_S| > k do
    S ← S ∪ {x_j}, future variable
endwhile
eliminate S from the super-bucket (Branch
  and Bound)
```

- Property:
  - **SBE(0) is BB**
  - **SBE(w*) is BE**

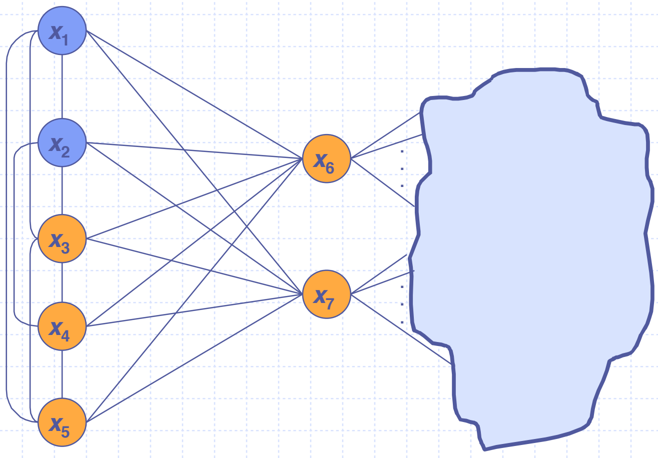
SBE(2): example



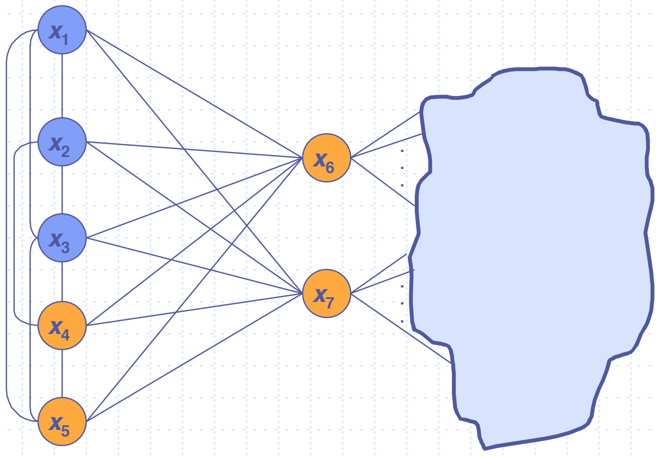
SBE(2): example



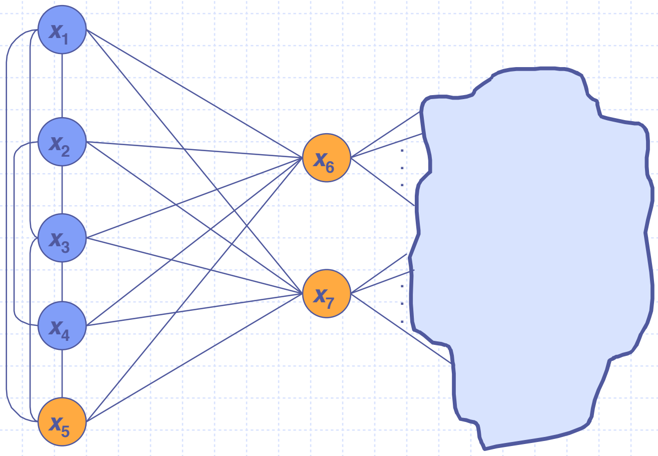
SBE(2): example

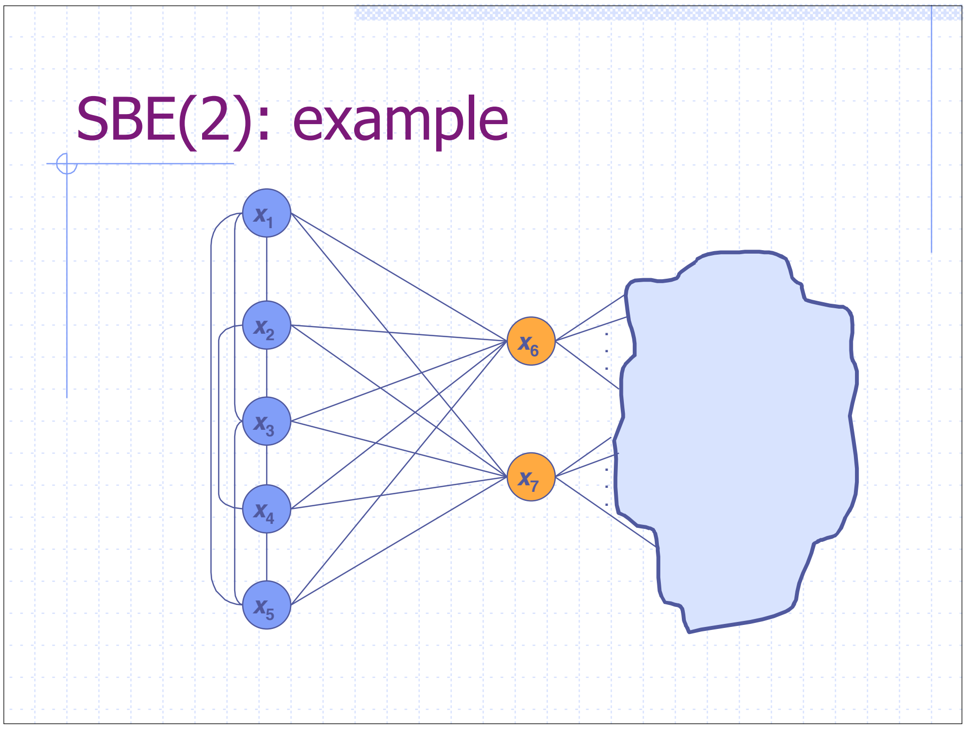


SBE(2): example

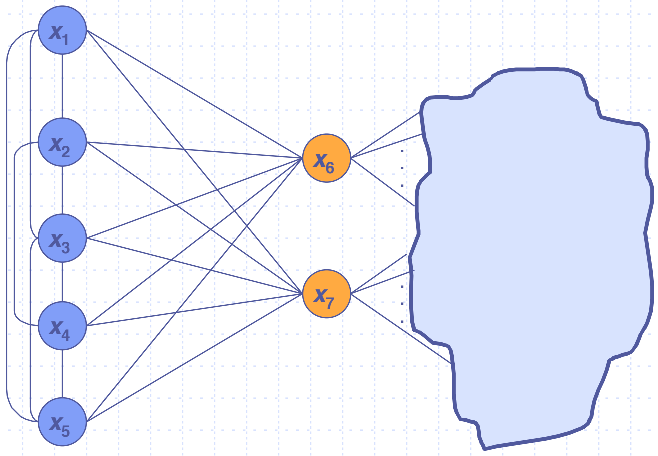


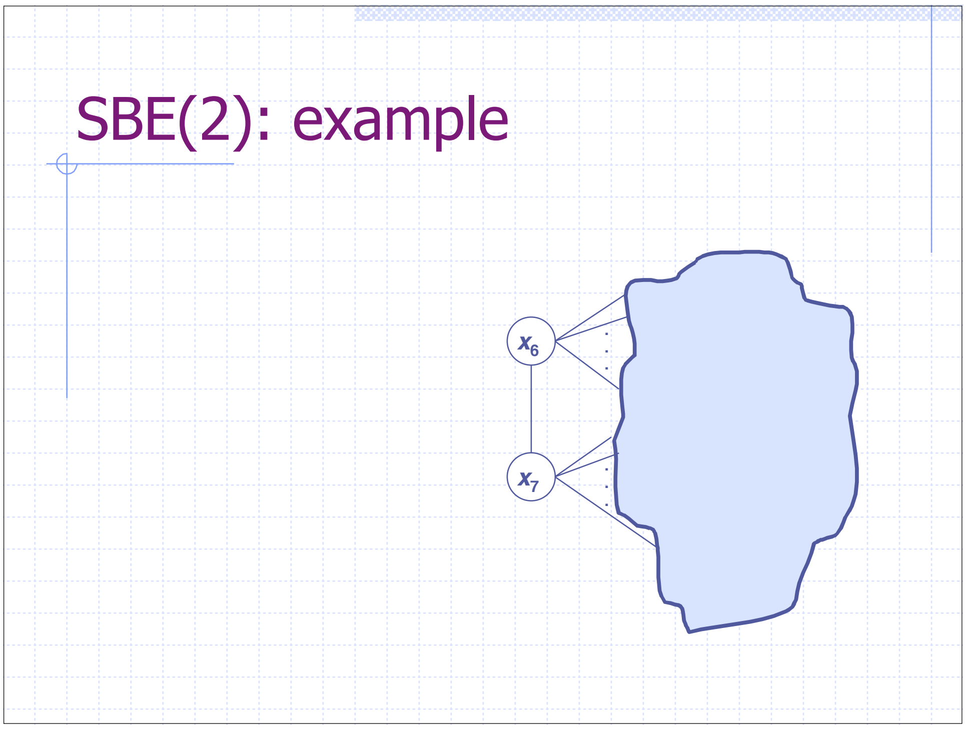
SBE(2): example



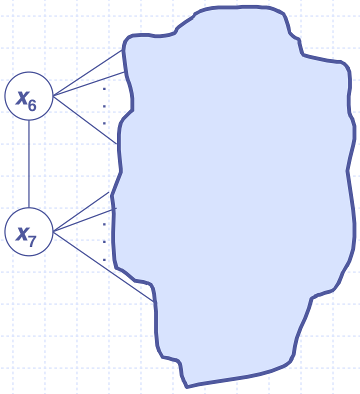


SBE(2): example





SBE(2): example



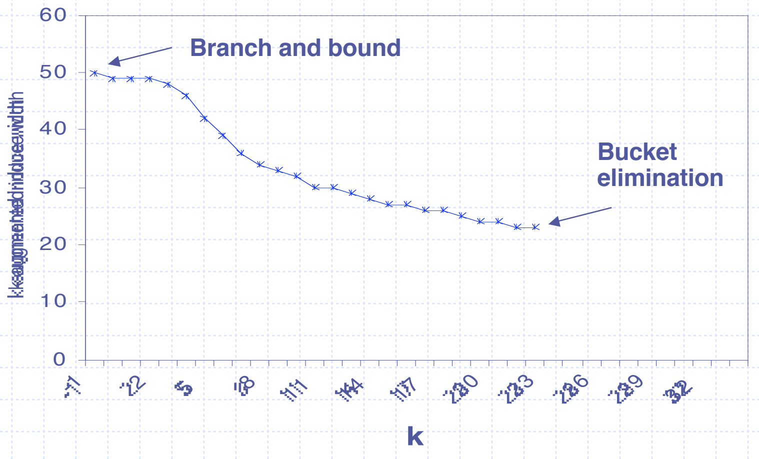
SBE(k)

- Complexity:
  - **space: O(exp(k))**
  - **time: O(exp(w_k*))**

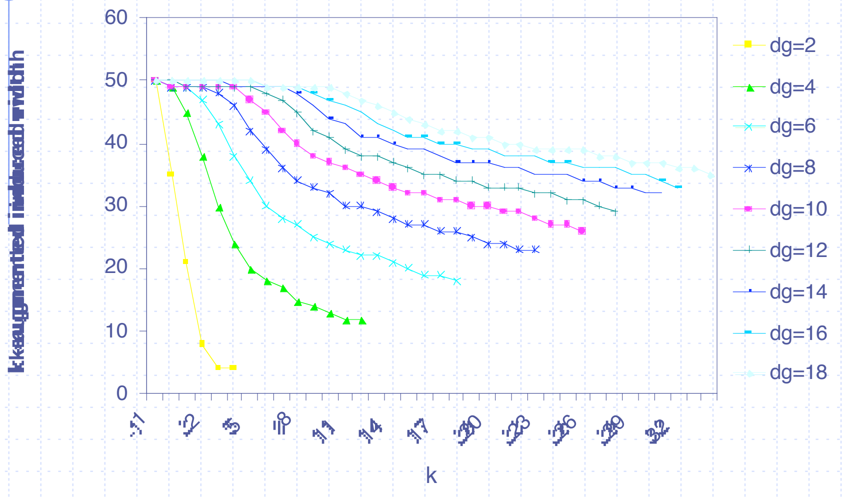
k-augmented induced width

Empirical Evaluation (time bounds)

- **Random Graphs (50 nodes, 200 edges, average degree 8, $w^* \approx 23$)**



Empirical Evaluation (time bounds)



Summary

- Soft constraints:
  - augment the CSP framework
  - find *best* solution
- Valued CSP:
  - general axiomatic framework
- Solving techniques:
  - generalization of CSP techniques

That’s all !

-
- Slides available next week at:
  - www.lsi.upc.es/~larrosa
  - www.iiia.csic.es/~pedro