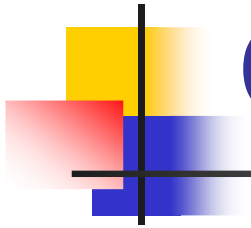
Constraint Processing and Probabilistic Reasoning from Graphical Models Perspective

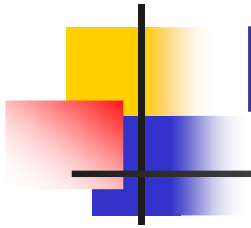
Rina Dechter

Information and Computer Science, UC-Irvine



Overview and Road Map

- The general graphical model
- Constraint networks
- Inference
- Search
- Probabilistic Networks



Road Map

- Graphical models
- Constraint networks Model
- Inference
- Search
- Probabilistic Networks

Constraint Networks

A

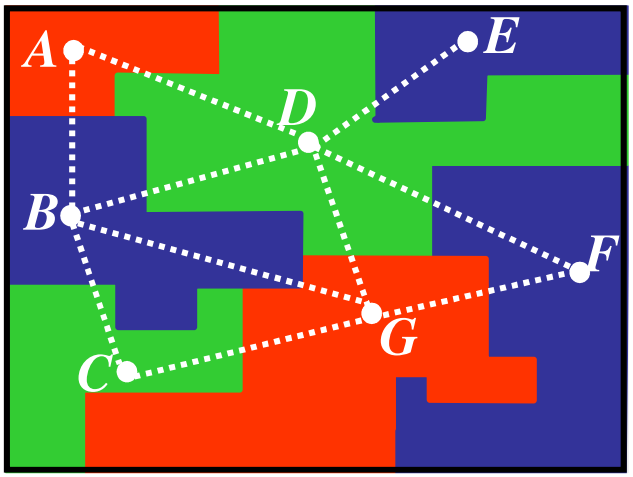
Example: map coloring

Variables - countries (A,B,C,etc.)

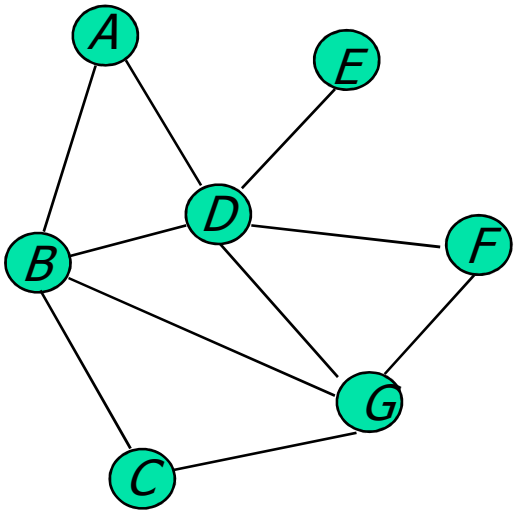
Values - colors (red, green, blue)

Constraints: **A ≠ B, A ≠ D, D ≠ E,** *etc.*

A	B
red	green
red	yellow
green	red
green	yellow
yellow	green
yellow	red



Constraint graph



Propositional Reasoning

Example: party problem

- If Alex goes, then Becky goes: $= A \rightarrow B$
- If Chris goes, then Alex goes: $= C \rightarrow A$

$$\mathbf{A \rightarrow B}$$

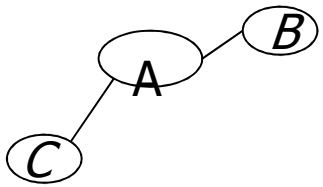
$$\mathbf{C \rightarrow A}$$

- **Question:**
  Is it possible that Chris goes to the party but Becky does not?

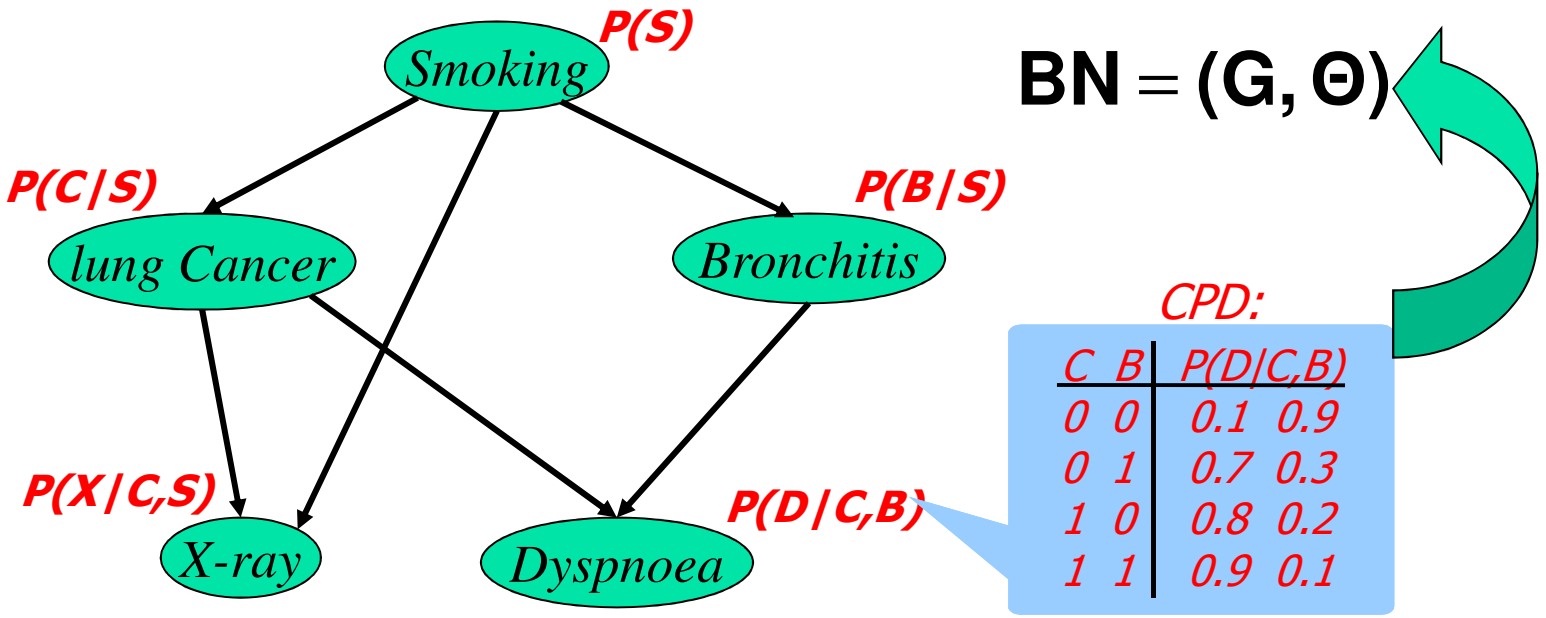
$$\Updownarrow$$

Is the *propositional theory*

$$\varphi = \{A \rightarrow B, C \rightarrow A, \ \neg \mathbf{B}, \ \mathbf{C}\} \text{ satisfiable?}$$



Bayesian Networks: Representation (Pearl, 1988)



$$P(S, C, B, X, D) = P(S)\,P(C|S)\,P(B|S)\,P(X|C,S)\,P(D|C,B)$$

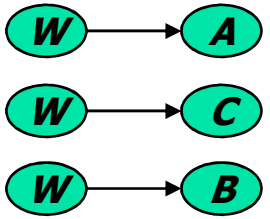
Belief Updating:

P (lung cancer=yes | smoking=no, dyspnoea=yes) = ?

Probabilistic Reasoning

Party example: the weather effect

- Alex is-likely-to-go in bad weather
- Chris rarely-goes in bad weather
- Becky is indifferent but unpredictable



P(A|W=bad)=.9

P(C|W=bad)=.1

P(B|W=bad)=.5

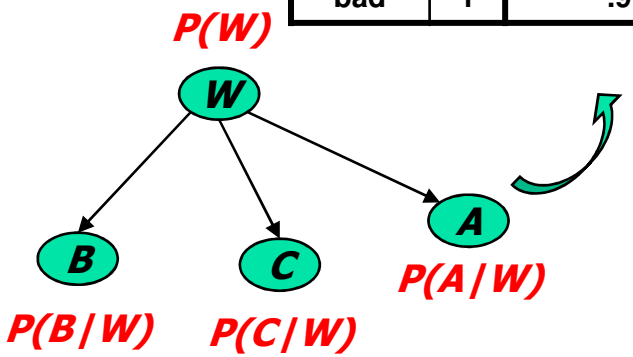
Questions:

- *Given bad weather, which group of individuals is most likely to show up at the party?*
- *What is the probability that Chris goes to the party but Becky does not?*

W	A	P(A|W)
good	0	.01
good	1	.99
bad	0	.1
bad	1	.9

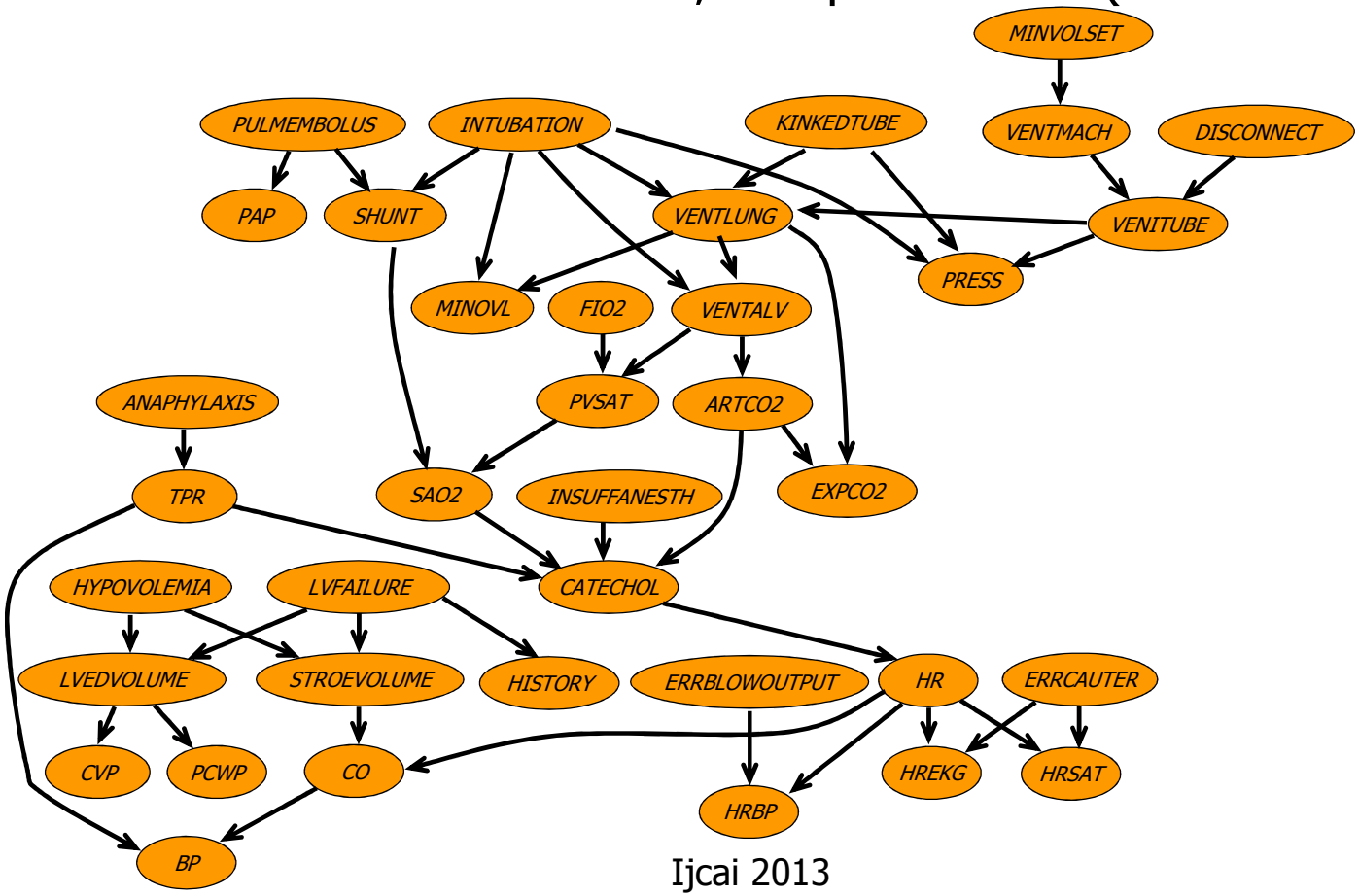
P(W,A,C,B) = P(B|W) · P(C|W) · P(A|W) · P(W)

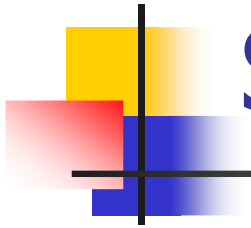
P(A,C,B|W=bad) = 0.9 · 0.1 · 0.5



Monitoring Intensive-Care Patients

The "alarm" network - 37 variables, 509 parameters (instead of 2^{37})



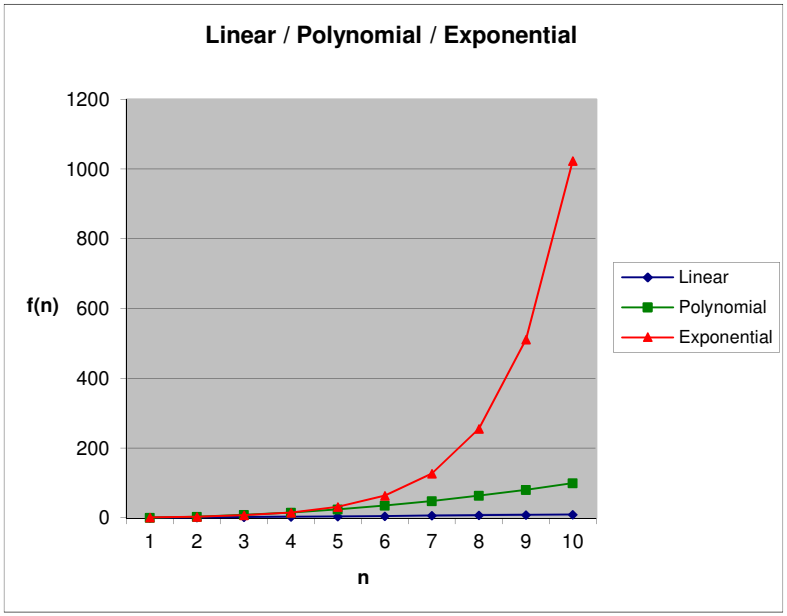


Sample Domains

- Web Pages and Link Analysis
- Battlespace Awareness
- Epidemiological Studies
- Citation Networks
- Communication Networks (Cell phone Fraud Detection)
- Intelligence Analysis (Terrorist Networks)
- Financial Transactions (Money Laundering)
- Computational Biology
- Object Recognition and Scene Analysis
- Natural Language Processing (e.g. Information Extraction and Semantic Parsing)

Complexity of Reasoning Tasks

- Constraint satisfaction
- Counting solutions
- Combinatorial optimization
- Belief updating
- Most probable explanation
- Decision-theoretic planning



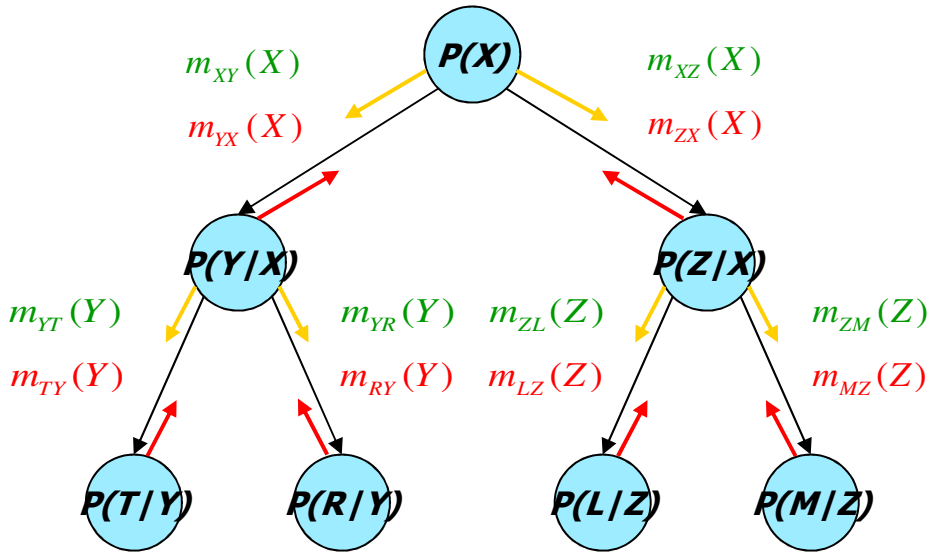
Reasoning is computationally hard

Complexity is Time and space(memory)

Tree-Solving is Easy

Belief updating (sum-prod)

CSP – consistency (projection-join)



MPE (max-prod)

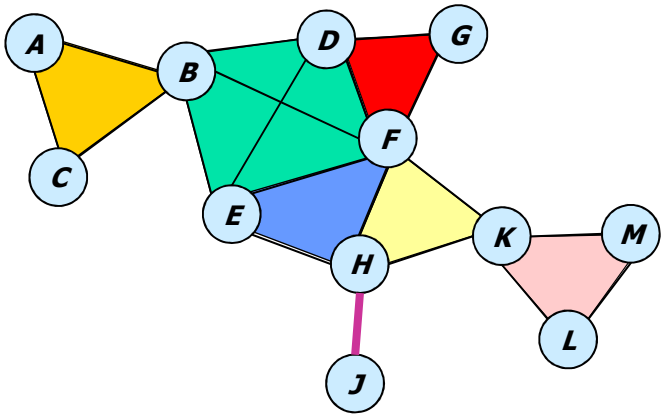
#CSP (sum-prod)

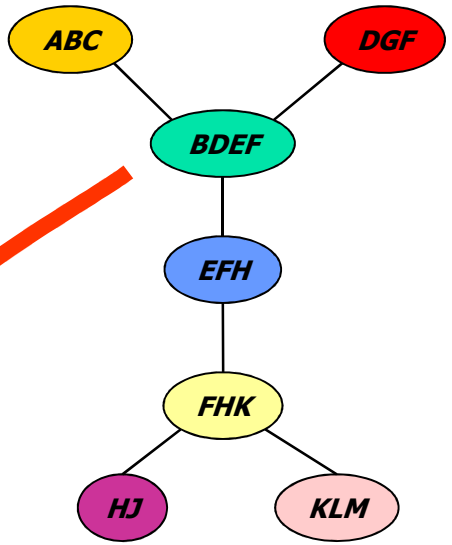
Trees are processed in linear time and memory

Transforming Into a Tree

- **By Inference (thinking)**
  - Transform into a single, equivalent tree of sub-problems
  - Atomic operation: inference
    - $f_1(\mathrm{X})\ combined\ with\ f_2\ (Y) \longrightarrow f_3(\mathrm{X,Y})$
- **By Conditioning (guessing)**
  - Transform into many tree-like sub-problems.
  - Atomic operation: assign a value $f_1(\mathrm{X=5})$

Inference and Treewidth





Inference algorithm:

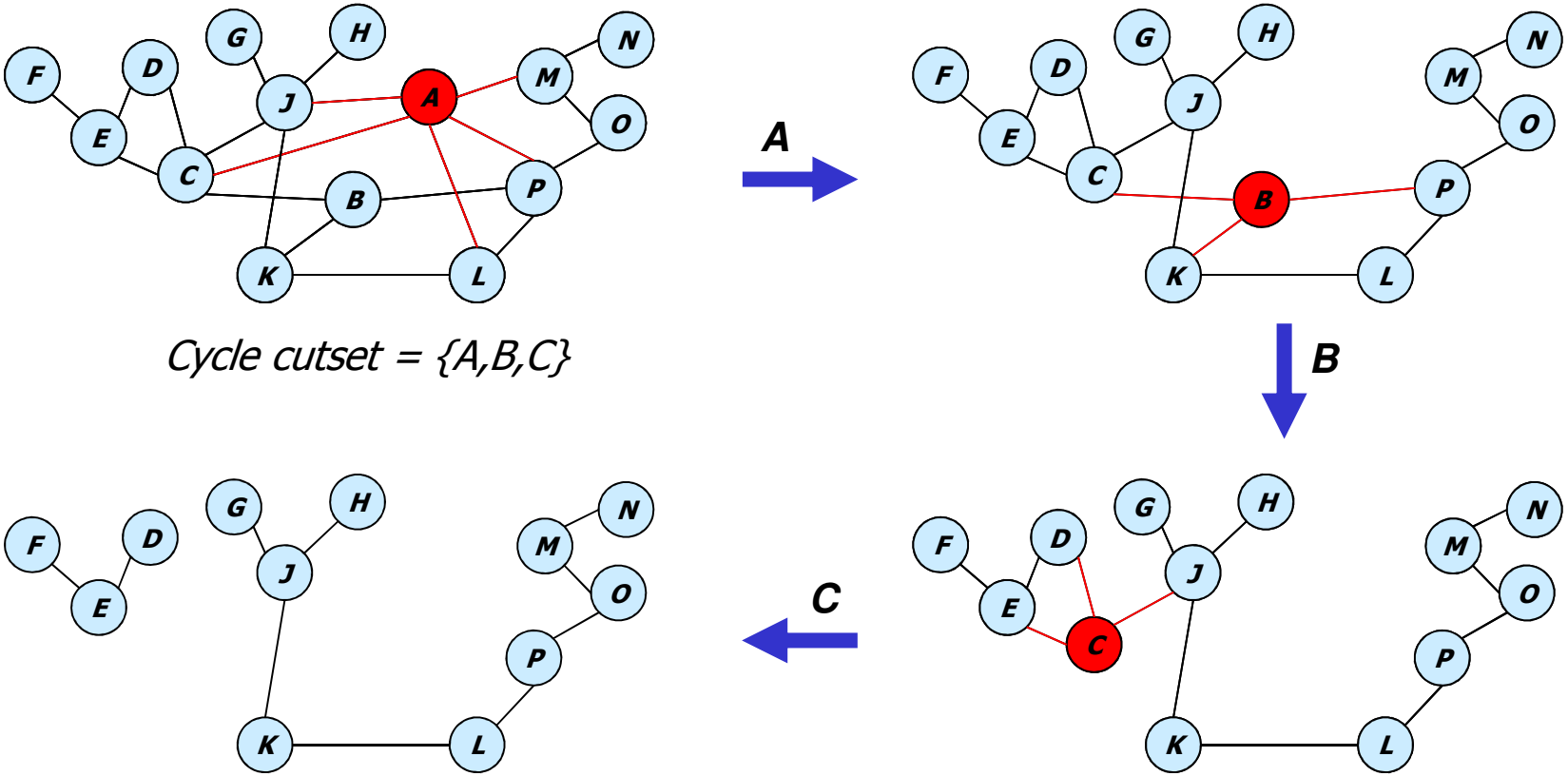
Time: exp(tree-width)

Space: exp(tree-width)

treewidth = 4 - 1 = 3

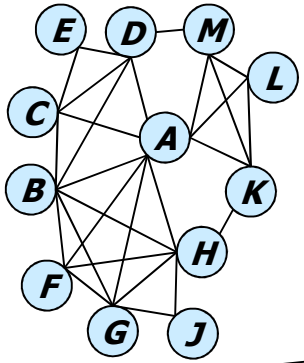
treewidth = (maximum cluster size) - 1

Conditioning and Cycle-Cutset

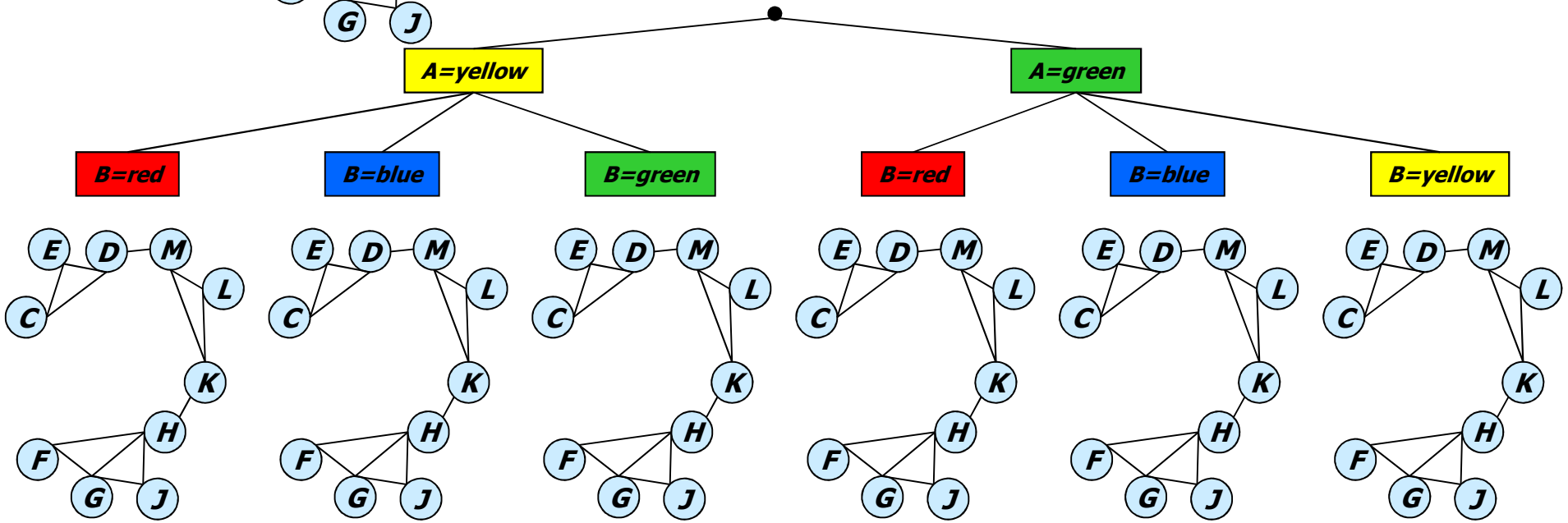


Search Over the Cutset (cont)

Graph Coloring problem

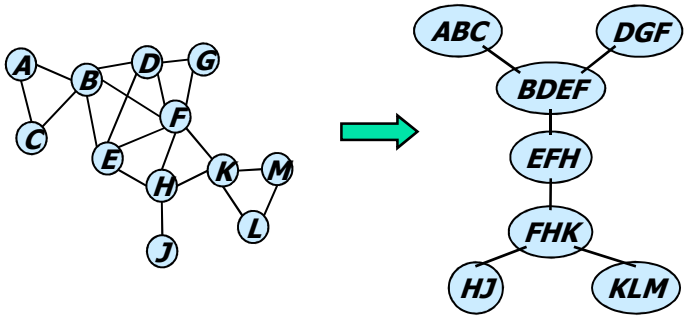


- *Inference may require too much memory*
- ***Condition** on some of the variables*



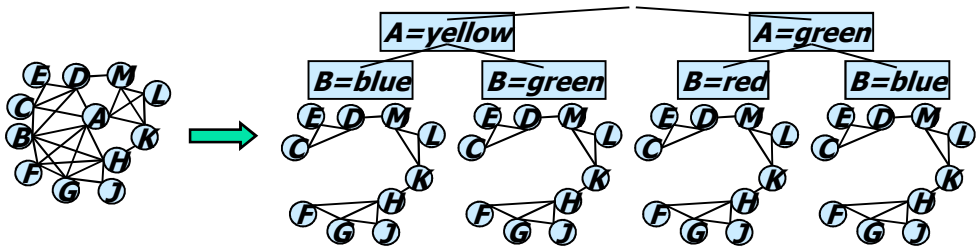
Inference vs. Conditioning

- **By Inference (thinking)**

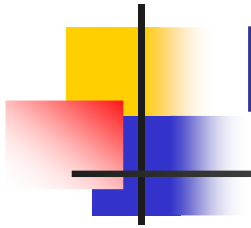


Exponential in treewidth
Time and memory

- ***By Conditioning (guessing)***



Exponential in cycle-cutset
Time-wise, linear memory



Road Map

- Graphical models
- Constraint networks
  - The model, Examples
  - The constraint graphs
- Inference
- Search
- Probabilistic Networks

Constraint-Networks and Tasks

Example: map coloring

Variables - countries (A,B,C,etc.)

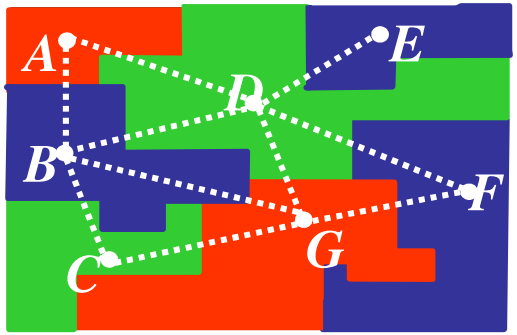
Values - colors (e.g., red, green, yellow)

Constraints:

$\mathbf{A \neq B,\ A \neq D,\ D \neq E,}$ *etc.*

A	B	C	D	E…
red	green	red	green	blue
red	blue	green	green	blue
…	…	…	…	green
…	…	…	…	red
red	blue	red	green	red

- Are the constraints consistent?
- Find a solution, find all solutions
- Count all solutions
- Find a good (optimal) solution



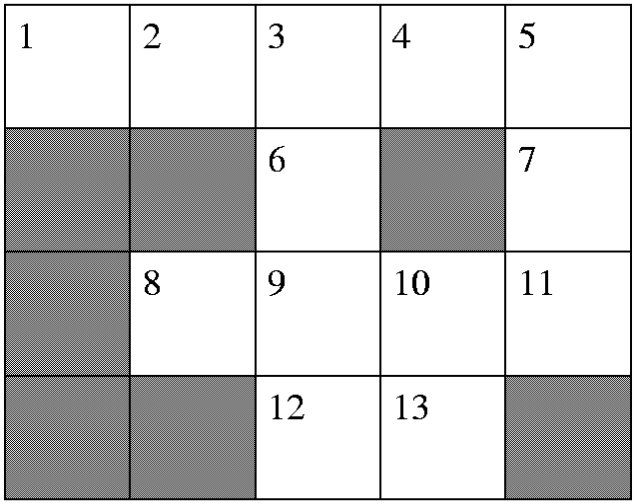
Constraint Networks (Formalisms)

- A constraint network is: *R=(X,D,C)*
  - X variables $X = \{X_1, \ldots, X_n\}$
  - D domain $D = \{D_1, \ldots, D_n\}, D_i = \{v_1, \ldots v_k\}$
  - C constraints $C = \{C_1, \ldots C_t\}$

    $C_i = (S_i, R_i)$
  - *Ri* expresses allowed tuples over scopes
- A solution is an assignment to all variables that satisfies all constraints (join of all relations).
- Tasks: consistency?, one or all solutions, counting, optimization

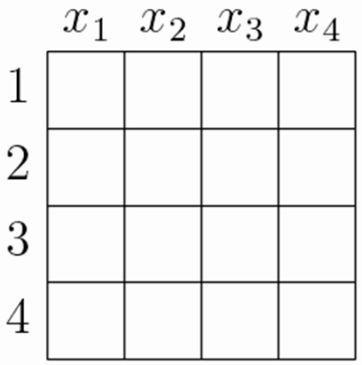
Example: Crossword Puzzle

- Variables: x_1, …, x_{13}
- Domains: letters
- Constraints: words from



{HOSES, LASER, SHEET, SNAIL, STEER, ALSO, EARN, HIKE, IRON, SAME, EAT, LET, RUN, SUN, TEN, YES, BE, IT, NO, US}

Example: The Queen Problem



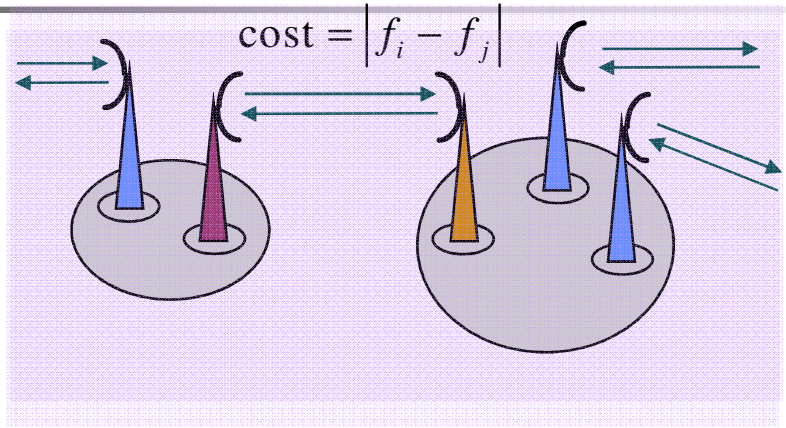
(a)

$$R_{12} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$$
$$R_{13} = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}$$
$$R_{14} = \{(1,2), (1,3), (2,1), (2,3), (2,4), (3,1), (3,2), (3,4)$$
$$(4,2), (4,3)\}$$
$$R_{23} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$$
$$R_{24} = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}$$
$$R_{34} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$$

(b)

The network has four variables, all with domains $D_i = \{1, 2, 3, 4\}$.
(a) The labeled chess board. (b) The constraints between variables.

Example: Radio Link Communications



Given a telecommunication network (where each communication link has various antenas) , assign a frequency to each antenna in such a way that all antennas may operate together without noticeable interference.

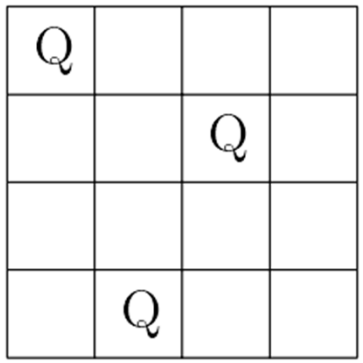
Encoding?

Variables: one for each antenna

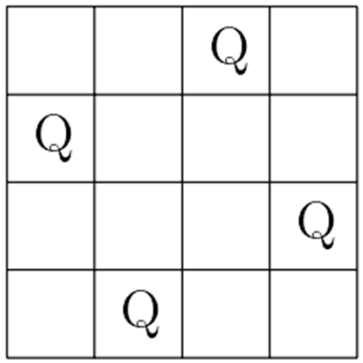
Domains: the set of available frequencies

Constraints: the ones referring to the antennas in the same communication link

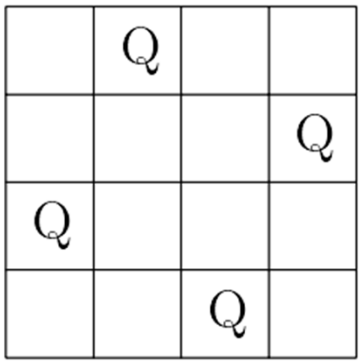
Partial Solutions



(a)

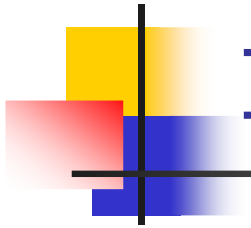


(b)



(c)

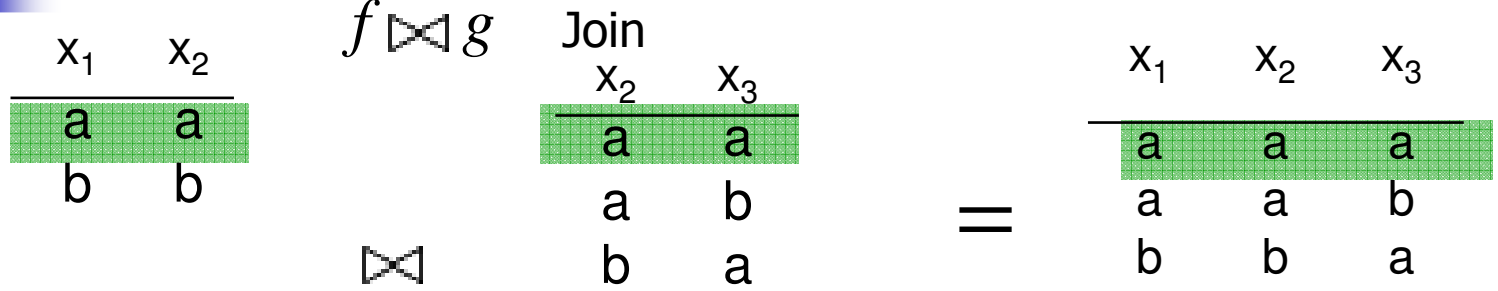
Not all consistent instantiations are part of a solution: (a) A consistent instantiation that is not part of a solution. (b) The placement of the queens corresponding to the solution (2, 4, 1, 3). (c) The placement of the queens corresponding to the solution (3, 1, 4, 2).



Inference: Using Relational Operators

- Intersection
- Union
- Difference
- Selection
- Projection
- Join
- Composition

Inference: Join of Relations



- Logical AND:

$$f \wedge g$$

x_1	x_2	f
a	a	true
a	b	false
b	a	false
b	b	true

$\wedge$

x_2	x_3	g
a	a	true
a	b	true
b	a	true
b	b	false

=

x_1	x_2	x_3	h
a	a	a	true
a	a	b	true
a	b	a	false
a	b	b	false
b	a	a	false
b	a	b	false
b	b	a	true
b	b	b	false

Inference: Join and Project

x_1	x_2	x_3
a	b	c
b	b	c
c	b	c
c	b	s

(a) Relation R

x_1	x_2	x_3
b	b	c
c	b	c
c	n	n

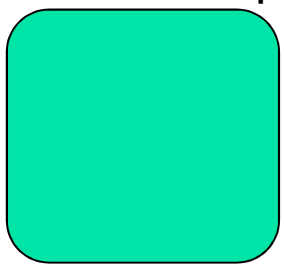
(b) Relation R'

x_2	x_3	x_4
a	a	1
b	c	2
b	c	3

(c) Relation R''

project

join



x_2	x_3
b	c
n	n

(b) $\pi_{\{x_2,x_3\}}(R')$

x_1	x_2	x_3	x_4
b	b	c	2
b	b	c	3
c	b	c	2
c	b	c	3

(c) $R' \bowtie R''$

Constraint's Representations

- Relation: allowed tuples

X	Y	Z
1	3	2
2	1	3

- Algebraic expression:

$$X + Y^2 \leq 10, X \neq Y$$

- Propositional formula:

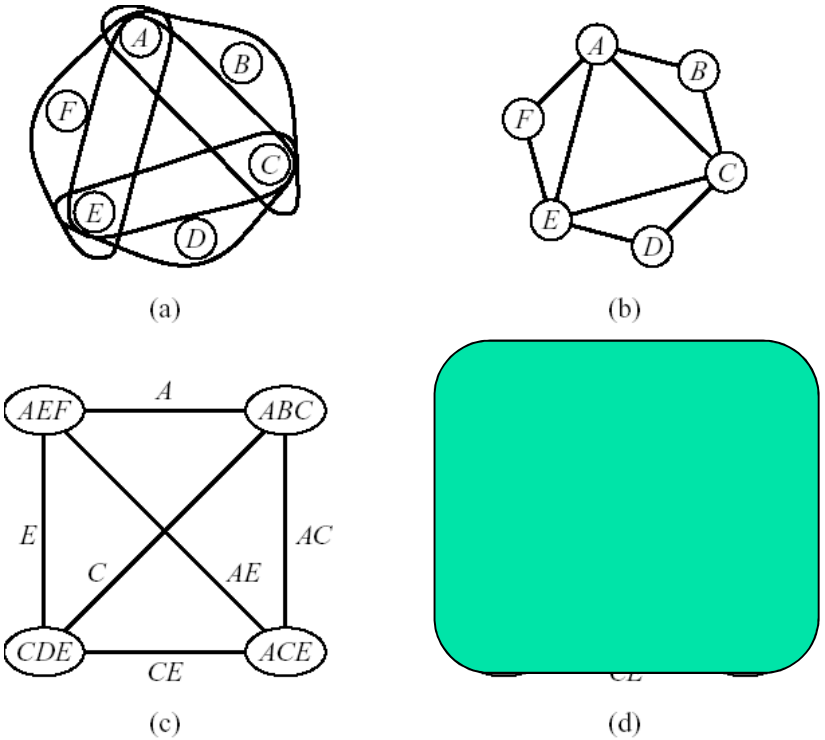
$$(a \vee b) \rightarrow \neg c$$

- Semantics: by a relation

Constraint Graphs:

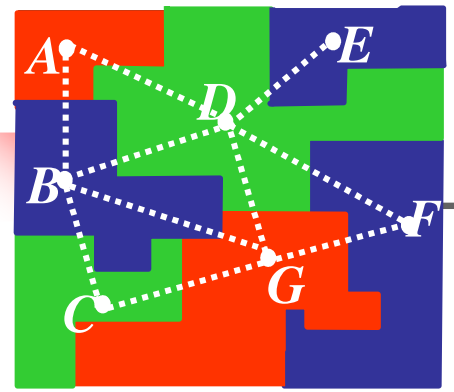
- Variables: A,B,C,D,E, F
- Realtions= R(AEF), R(ABC), R(CDE), R(ACE)

- Hypergraph
- Dual graphs
- Primal graphs
- Factor Graphs



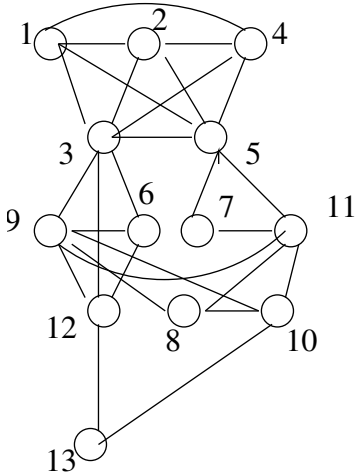
Constraint Graphs:

Primal, Dual and Hypergraphs

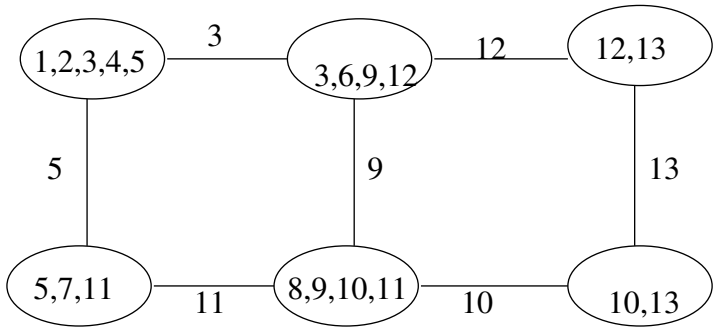


- ***A (primal) constraint graph: a node per variable, arcs connect constrained variables.***
- ***A dual constraint graph: a node per constraint's scope, an arc connect nodes sharing variables =hypergraph***

1	2	3	4	5
		6		7
	8	9	10	11
		12	13	



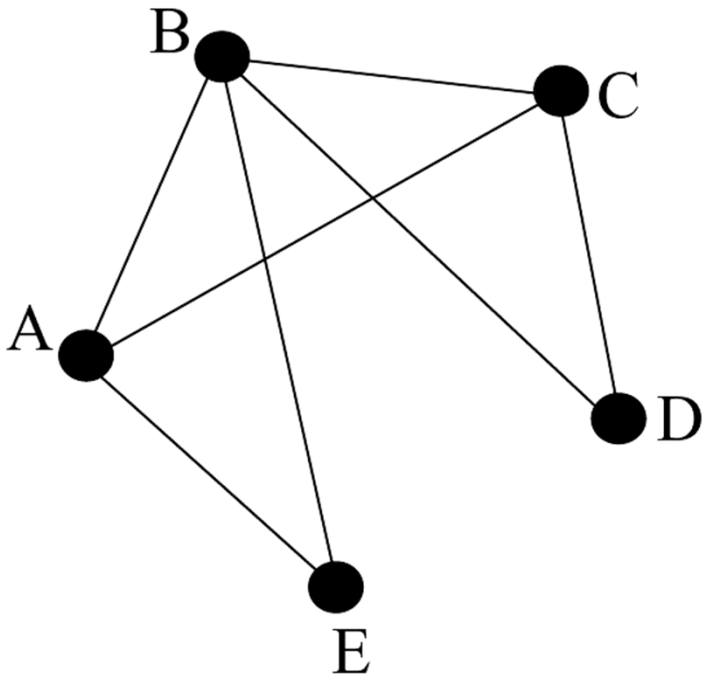
(a)



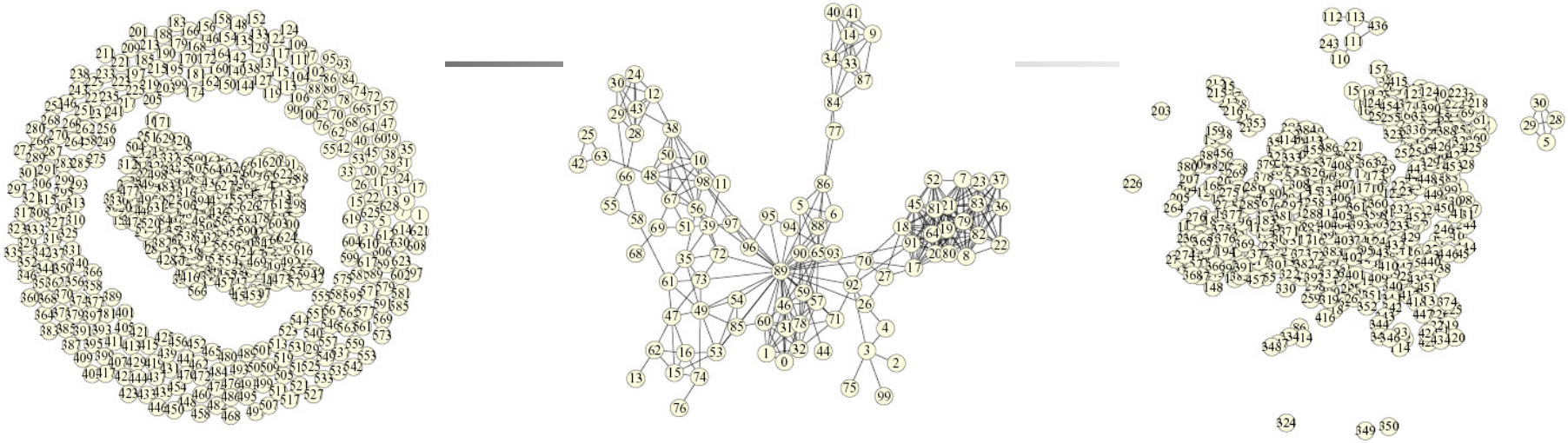
(b)

Propositional Satisfiability

$\varphi = \{(\neg C), (A \vee B \vee C), (\neg A \vee B \vee E), (\neg B \vee C \vee D)\}$.



Radio Link Assignment



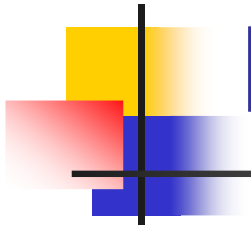
Given a telecommunication network (where each communication link has various antenas) , assign a frequency to each antenna in such a way that all antennas may operate together without noticeable interference.

Encoding?

Variables: one for each antenna

Domains: the set of available frequencies

Constraints: the ones referring to the antennas in the same communication link



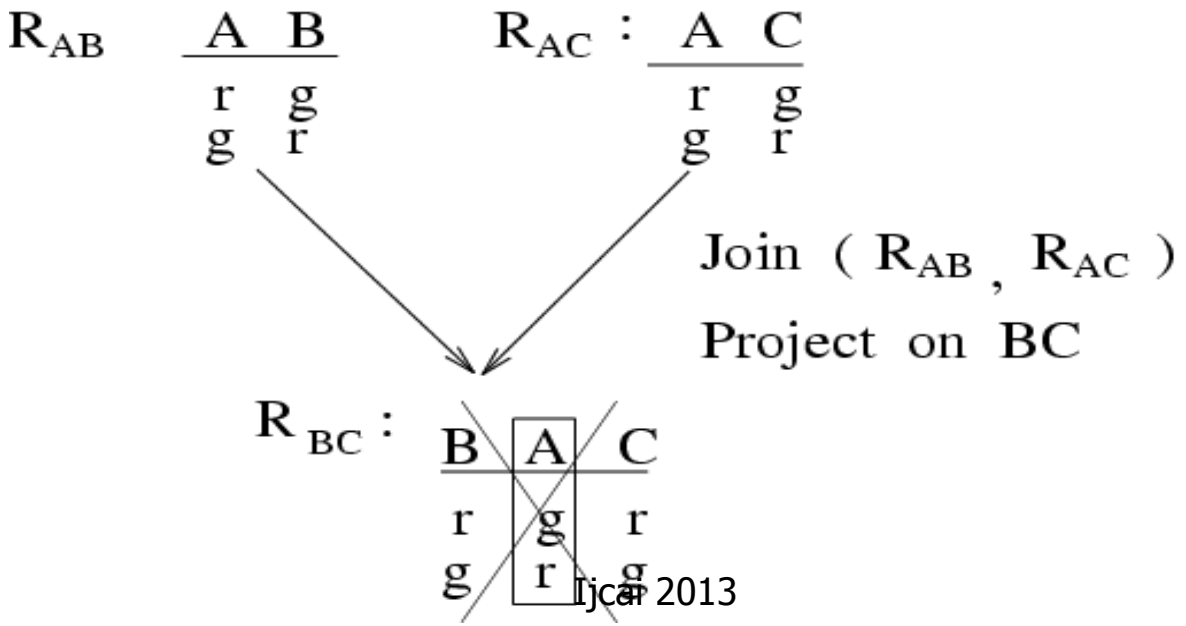
Road Map

- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination:
  - Tree-clustering
  - Constraint propagation
- Search
- Probabilistic Networks

Inference: Join and Project

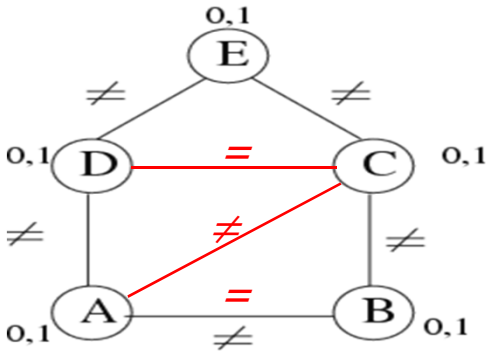
- Given 2 constraints we can deduce a new one by join and then project, via variable-elimination

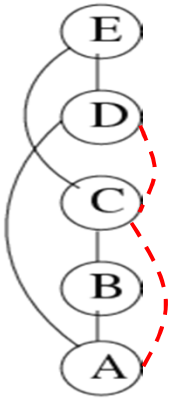
Join operation ⋈ *over A finds all solutions satisfying constraints that involve A*



Bucket Elimination

Adaptive Consistency (Dechter & Pearl, 1987)





Bucket E: $E \neq D,\ E \neq C$

Bucket D: $D \neq A$ → $D = C$

Bucket C: $C \neq B$ → $A \neq C$

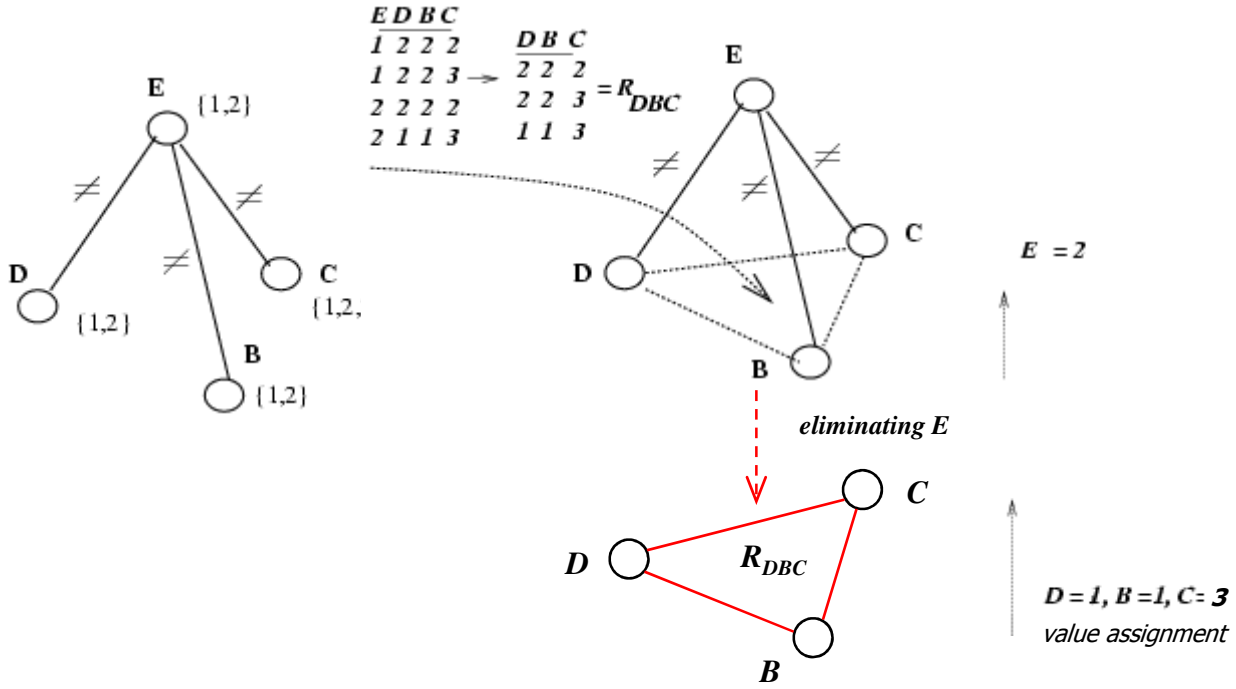
Bucket B: $B \neq A$ → $B = A$

Bucket A: → ***contradiction***

Complexity : $O(n \exp(w^*)+1)$

w^* - ***induced width***

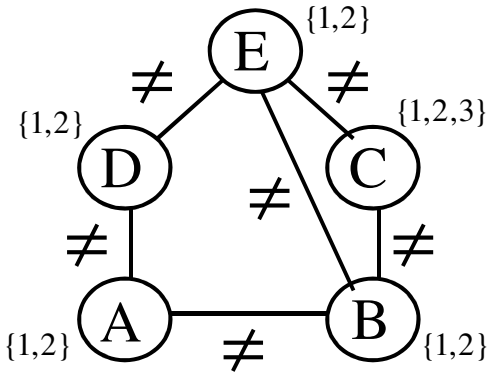
The Idea of Elimination



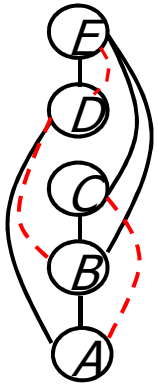
$$R_{DBC} = \prod_{DBC} R_{ED} \bowtie R_{EB} \bowtie R_{EC}$$

Eliminate variable E $\Leftrightarrow$ join and project

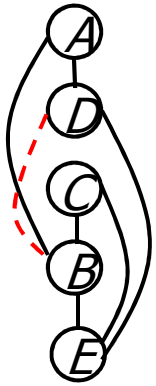
Bucket Elimination
Adaptive Consistency (Dechter & Pearl, 1987)



$Bucket(E)$: E ≠ D, E ≠ C, E ≠ B
$Bucket(D)$: D ≠ A || R_{DCB}
$Bucket(C)$: C ≠ B || R_{ACB}
$Bucket(B)$: B ≠ A || R_{AB}
$Bucket(A)$: R_A



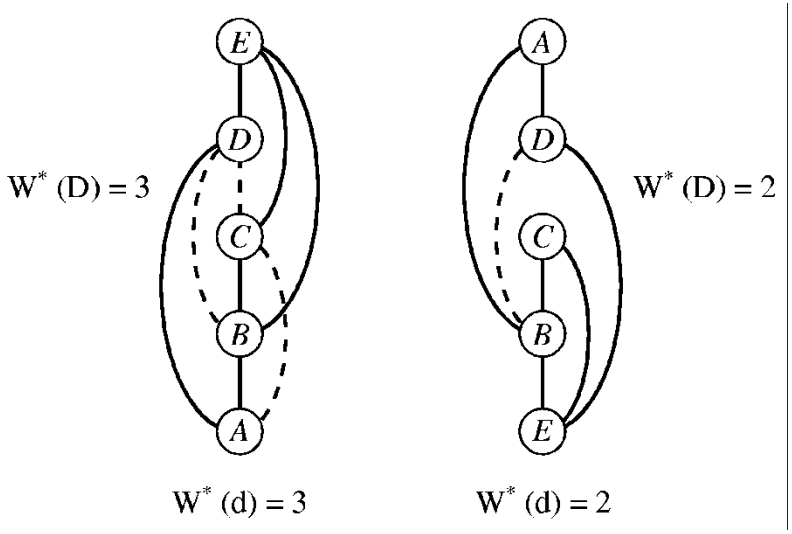
$Bucket(A)$: A ≠ D, A ≠ B
$Bucket(D)$: D ≠ E || R_{DB}
$Bucket(C)$: C ≠ B, C ≠ E
$Bucket(B)$: B ≠ E || R^D_{BE}, R^C_{BE}
$Bucket(E)$: || R_E



Complexity: $O(n \exp(w^*(d)+1))$,

$w^*(d)$ - ***induced width along ordering d***

The induced-width



- **Width along *d*, w(d):**
  - **max # of previous parents**
- **Induced width w*(d):**
  - **The width in the ordered *induced graph***
- **Induced-width w*:**
  - **Smallest induced-width over all orderings**
- **Finding w***
  - **NP-complete *(Arnborg, 1985) but greedy heuristics (min-fill).***

Adaptive Consistency, Bucket-Elimination

Initialize: partition constraints into $bucket_1,\ldots,bucket_n$

For *i=n* down to *1* along d // process in reverse order

for all relations $R_1,\ldots,R_m \in bucket_i$ **do**

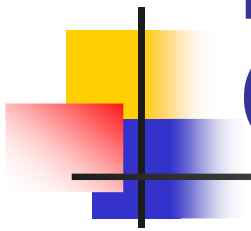
join and "project-out" X_i

$$R_{new} \leftarrow \prod_{(-X_i)} \left(\quad {}_j R_j \right)$$

If R_{new} *is not empty, add it to* $bucket_k, k < i,$
where k is the largest variable index in R_{new}
Else *problem is unsatisfiable*

Return *the set of all relations (old and new) in the buckets*

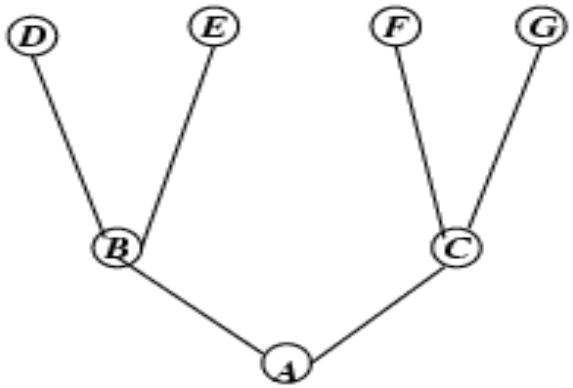
Properties of Bucket-Elimination (Adaptive Consistency)

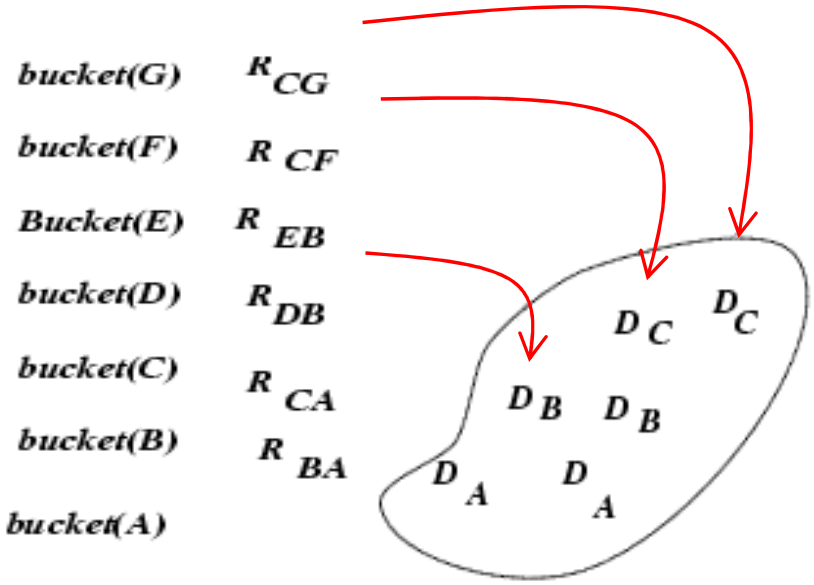


- Adaptive consistency generates a constraint network that is backtrack-free (can be solved without deadends).
- The time and space complexity of adaptive consistency along ordering *d* is exponential in w* +1, and w*, respectively. .
- Therefore, problems having bounded induced width are tractable (solved in polynomial time).
  - *trees* (w*=1),
  - *series-parallel networks* (w*=2),
  - and in general *k-trees* (w*=k).

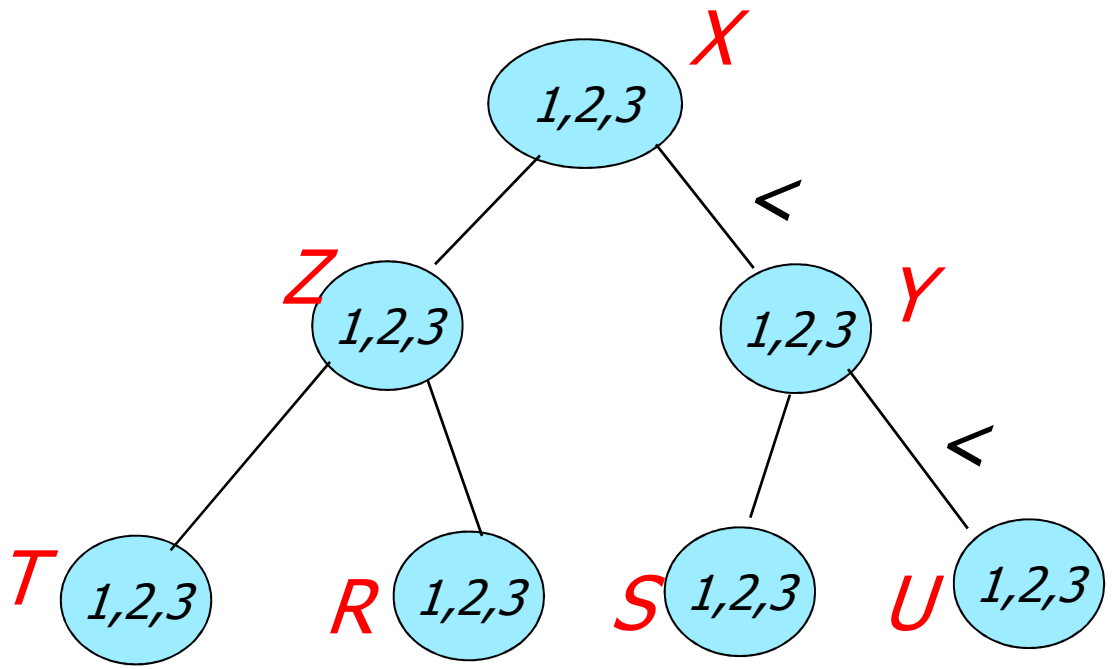
Solving Trees
(Mackworth and Freuder, 1985)

Adaptive consistency is linear for trees and equivalent to enforcing directional arc-consistency (recording only unary constraints)

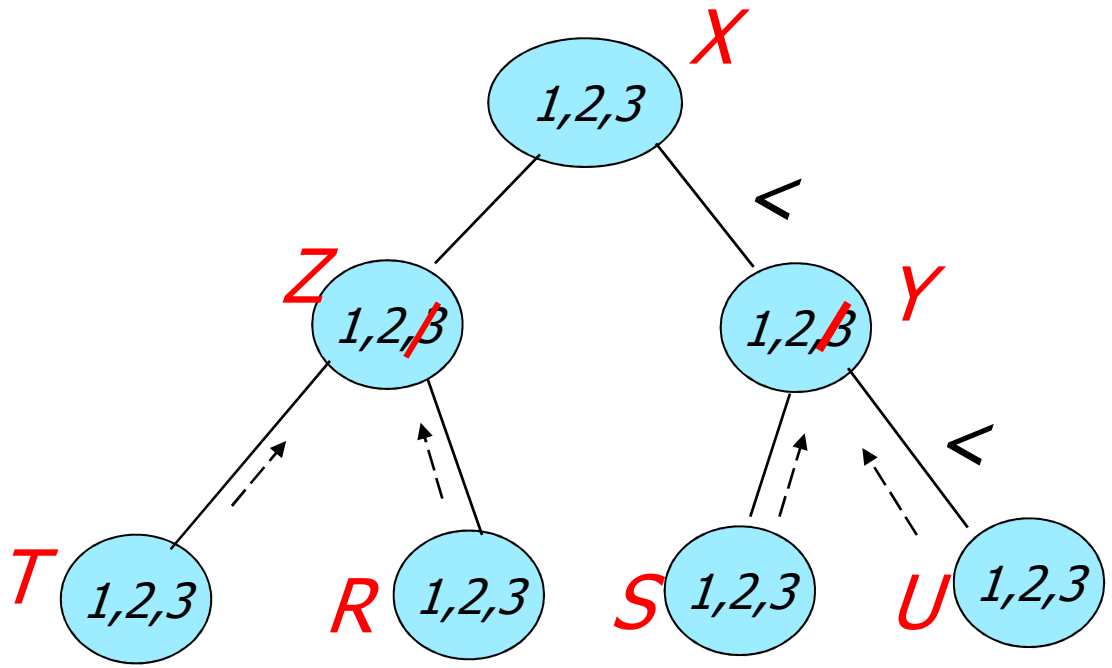




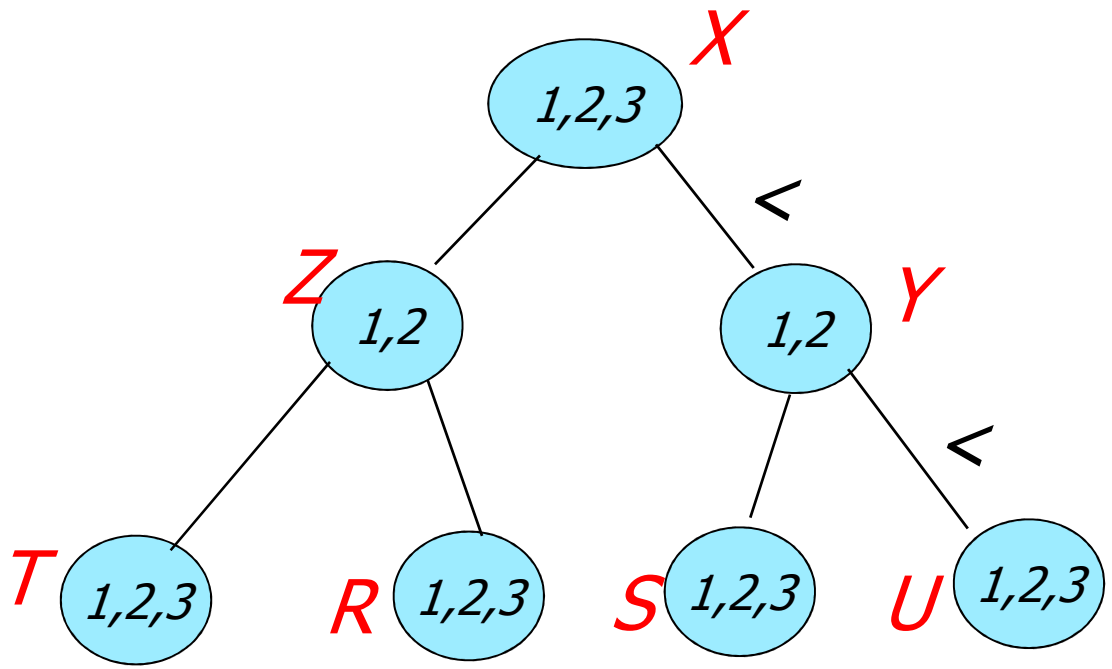
Tree Solving is Easy



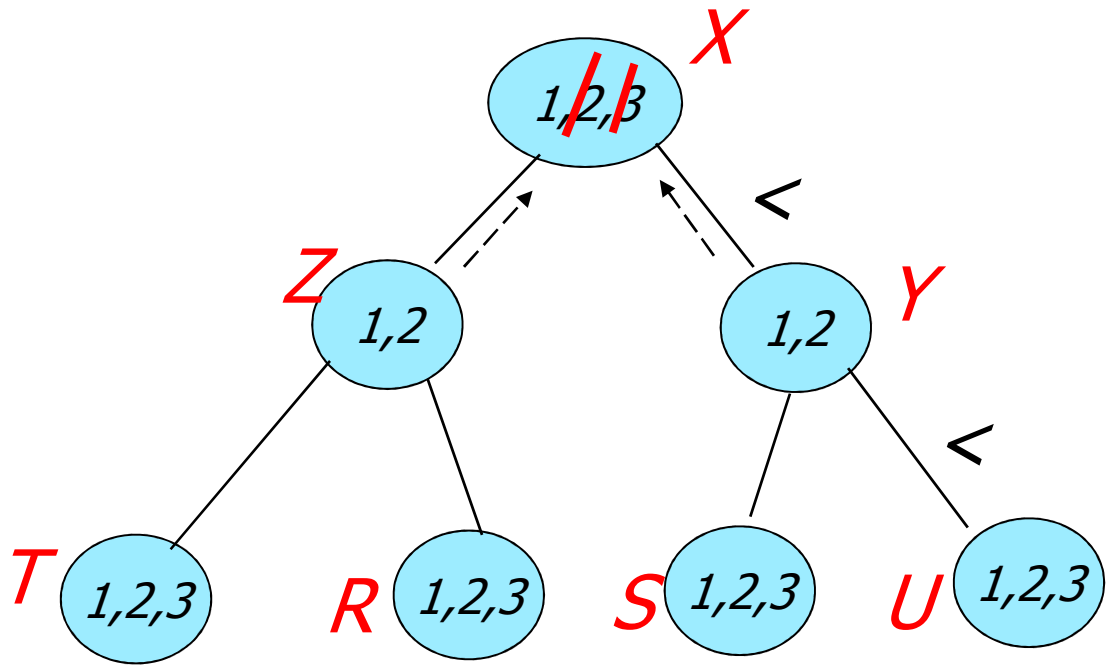
Tree Solving is Easy



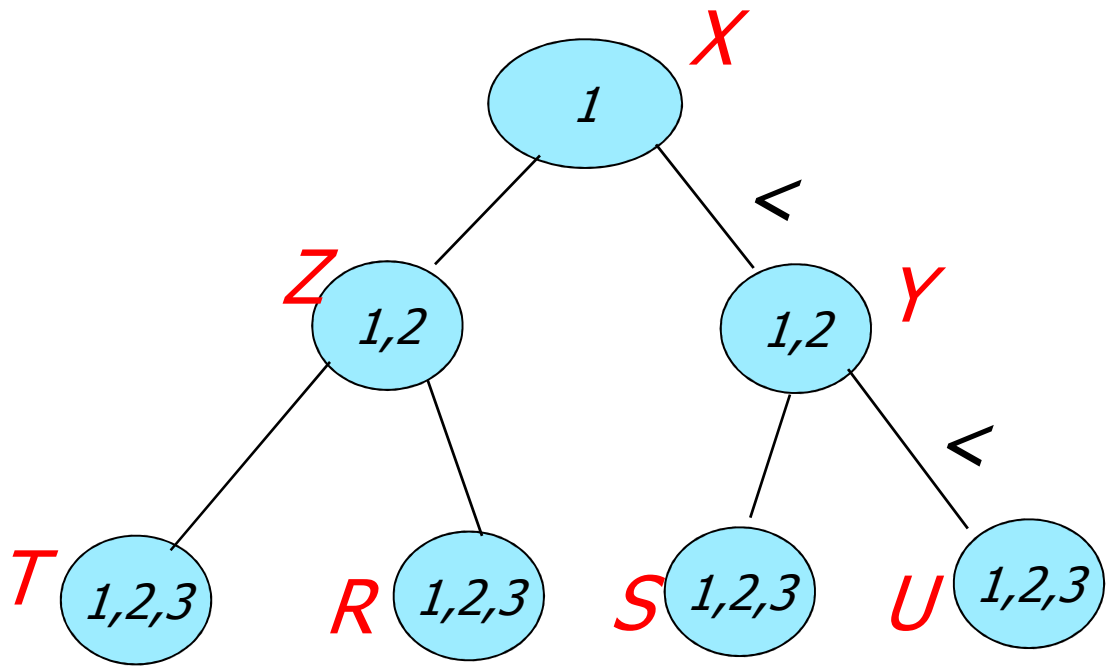
Tree Solving is Easy



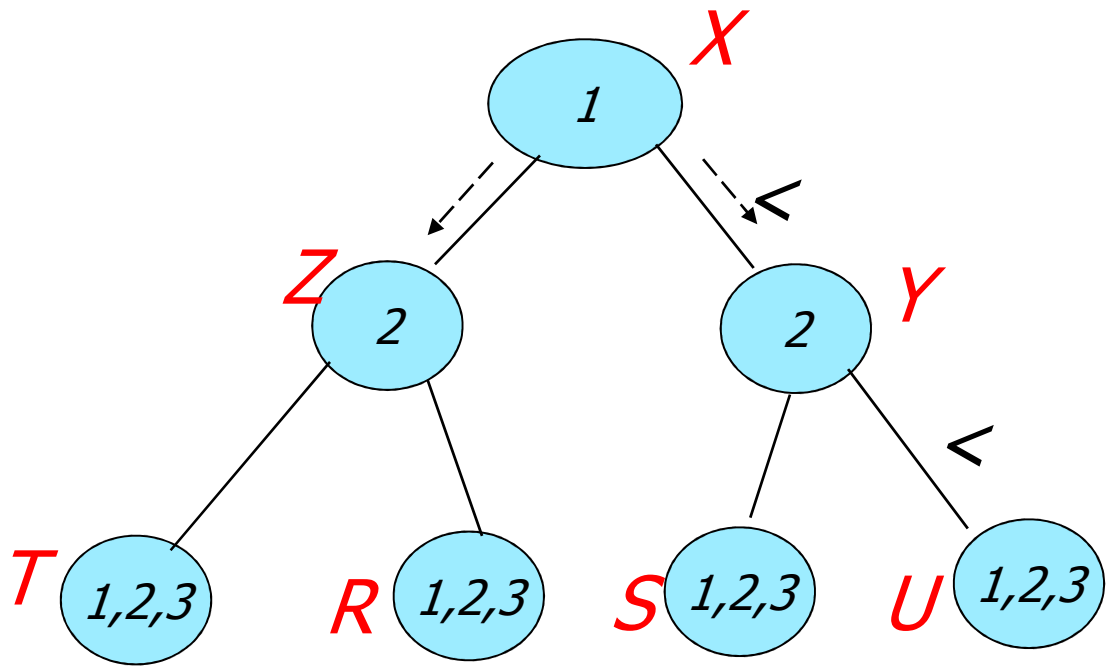
Tree Solving is Easy



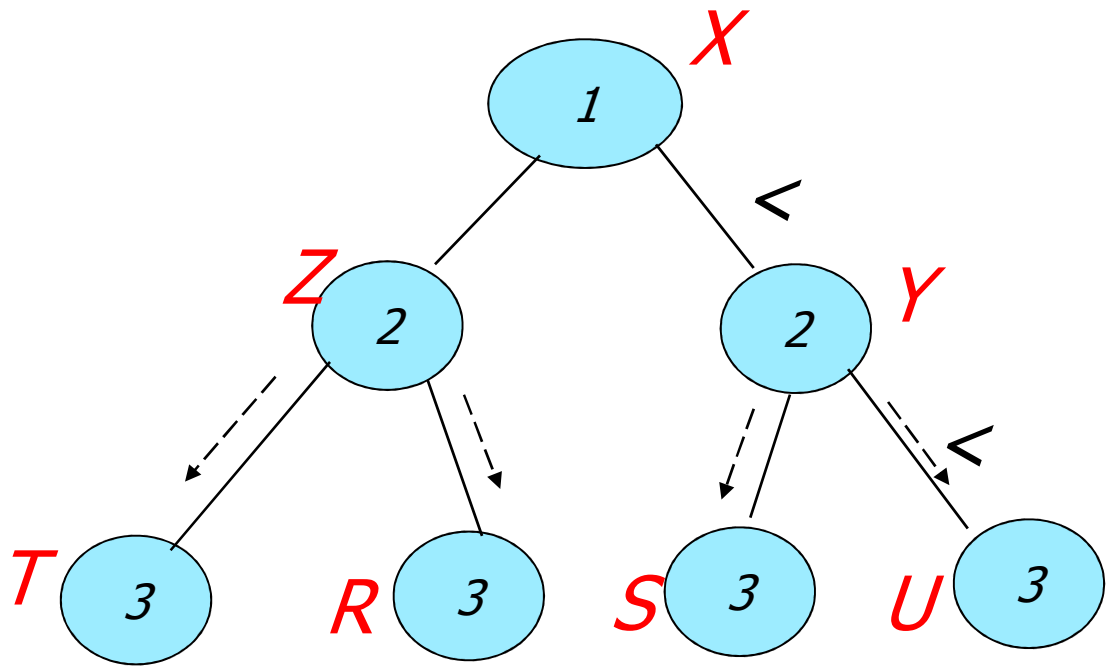
Tree Solving is Easy



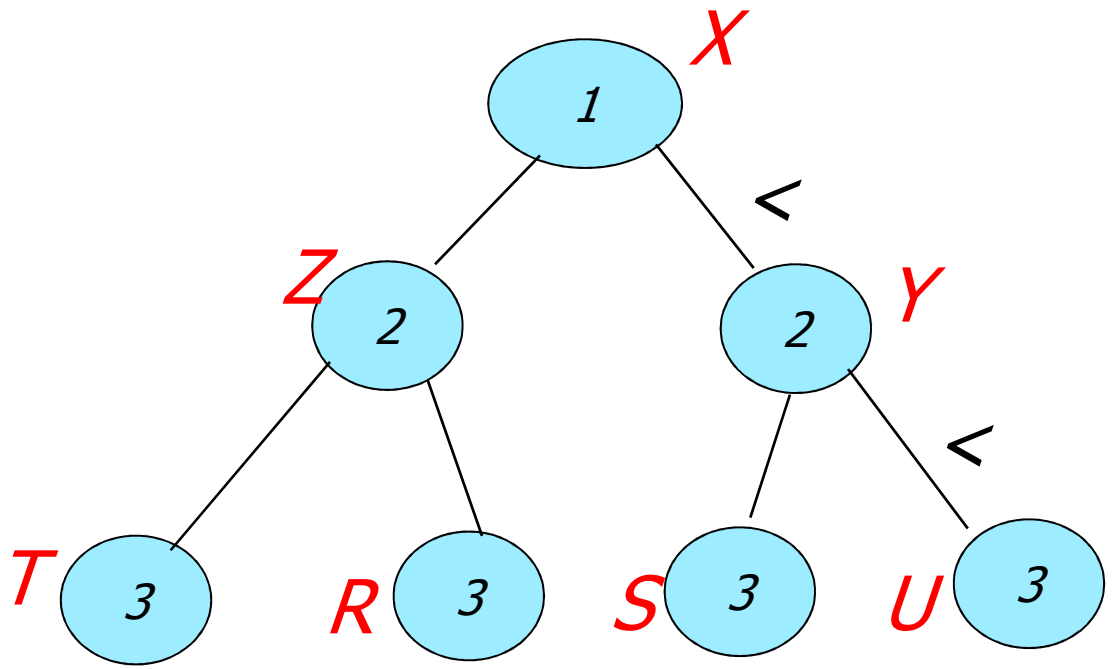
Tree Solving is Easy



Tree Solving is Easy

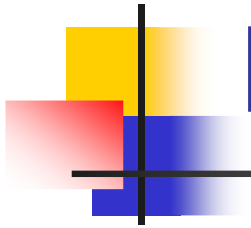


Tree Solving is Easy



Constraint propagation
Solves trees in linear time

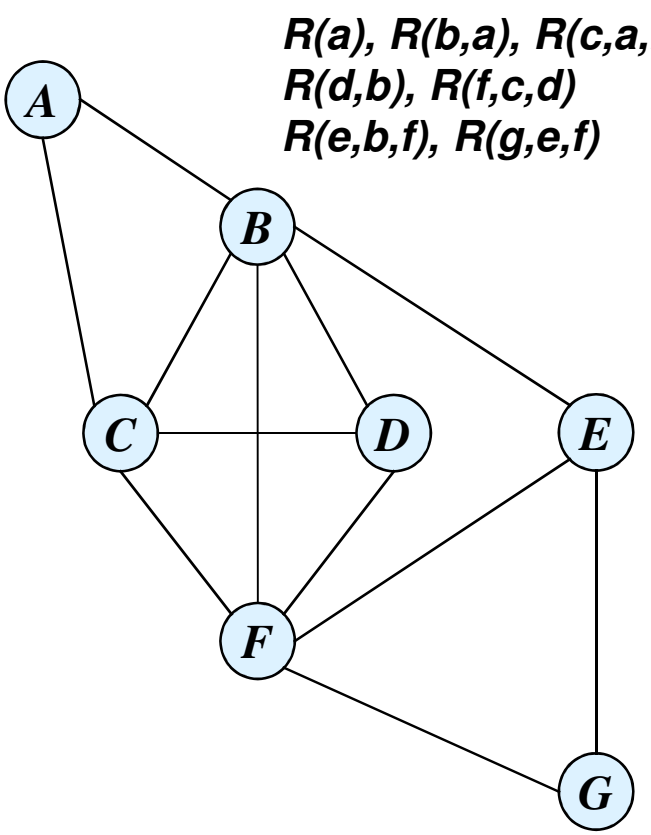
Inference makes global information local

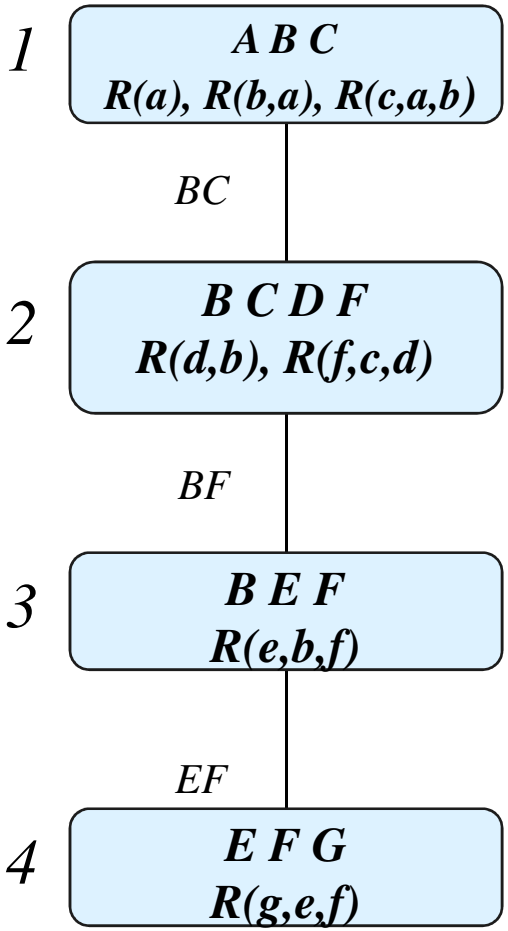


Road Map

- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination:
  - Tree-clustering
  - Constraint propagation
- Search
- Probabilistic Networks

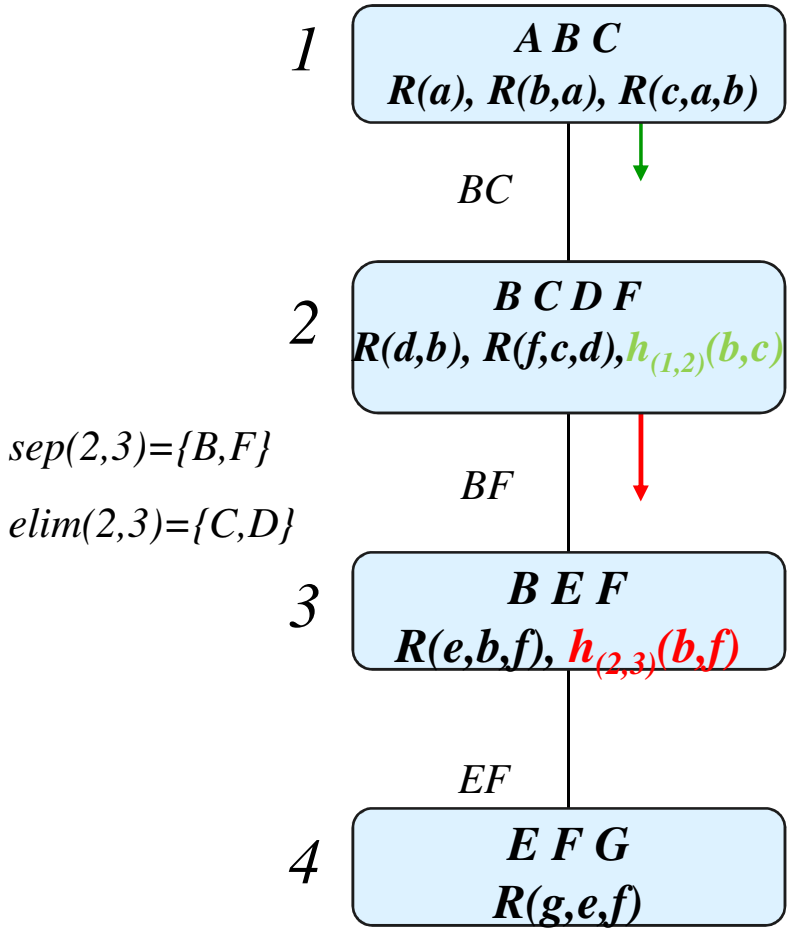
Tree Decomposition





- Each function in a cluster
- Satisfy running intersection property
- Then infer a function in a cluster and send to neighbors

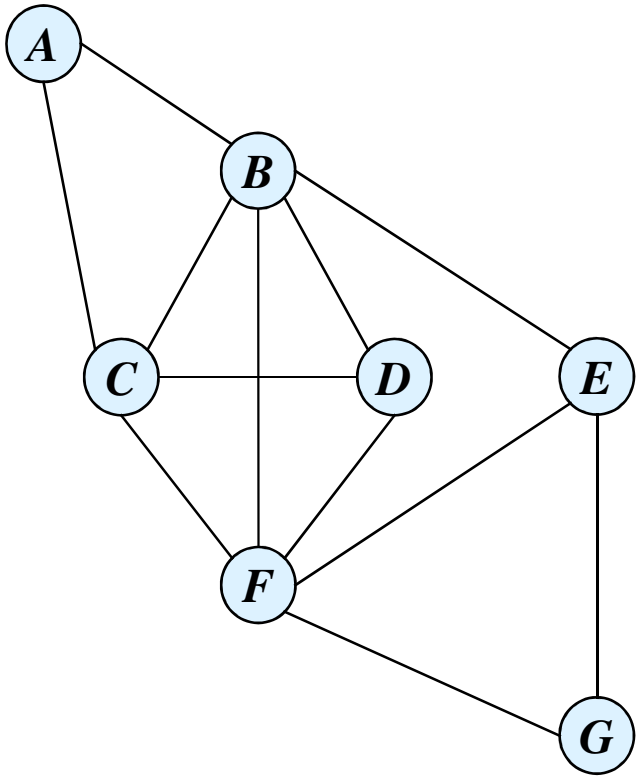
Cluster Tree Elimination

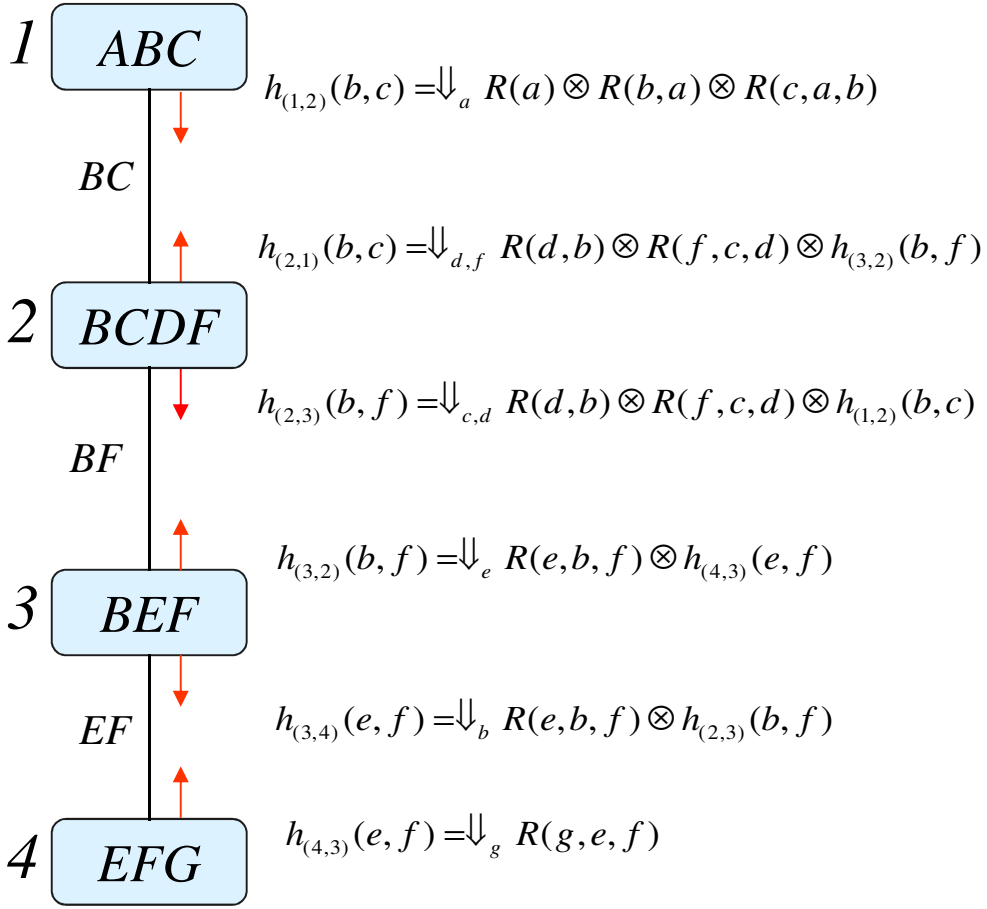


$$h_{(1,2)}(b,c) = \Downarrow_a R(a) \otimes R(b,a) \otimes R(c,a,b)$$

$$h_{(2,3)}(b,f) = \Downarrow_{c,d} R(d,b) \otimes R(f,c,d) \otimes h_{(1,2)}(b,c)$$

CTE: Cluster Tree Elimination





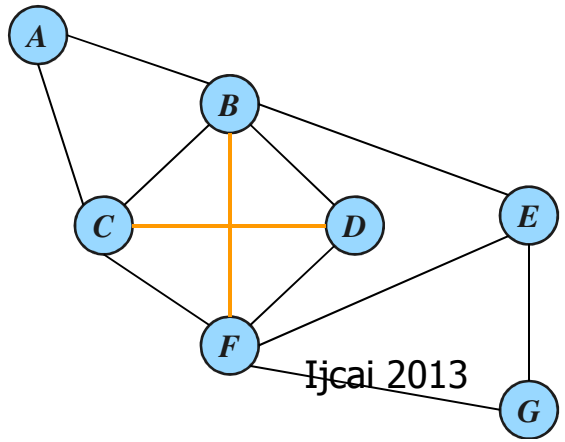
Time: ***O (exp(w*+1))***
Space: ***O (exp(sep))***

Local Information: Domains
Global Information: the minimal domains

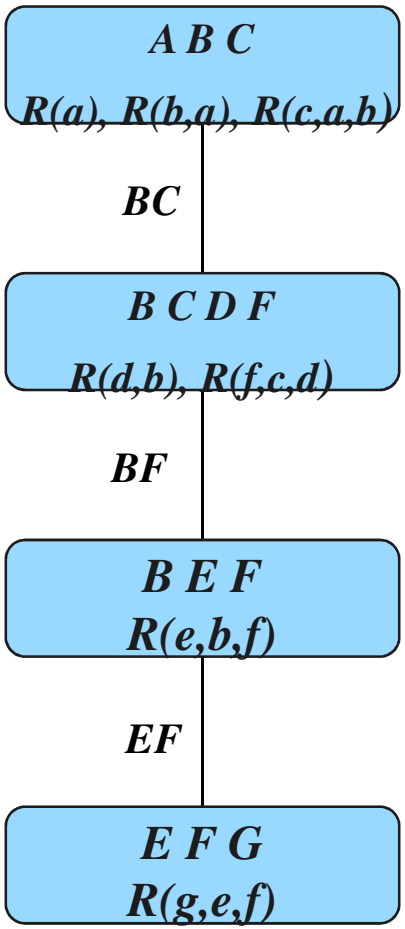
Tree decompositions

A *tree decomposition* for $R =< X,D,C >$ is a triple $< T, \chi, \psi >$, where $T = (V,E)$ is a tree and χ and ψ are labelings over vertex $v \in V$ $\chi(v) \subseteq X$ and $\psi(v) \subseteq C$ satisfying :

1. For each function $C_i \in C$ there is exactly one vertex such that $C_i \in \psi(v)$ and $scope(C_i) \subseteq \chi(v)$
2. For each variable $X_i \in X$ the set $\{v \in V | X_i \in \chi(v)\}$ forms a connected subtree (running intersection property)



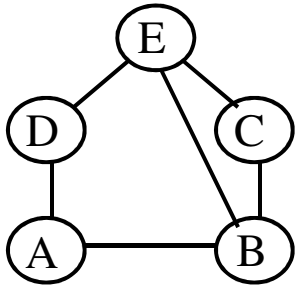
Belief network

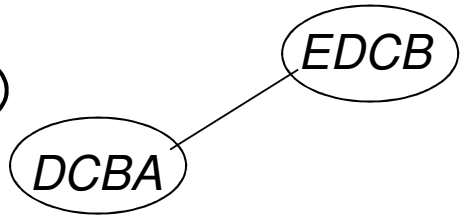


Tree decomposition

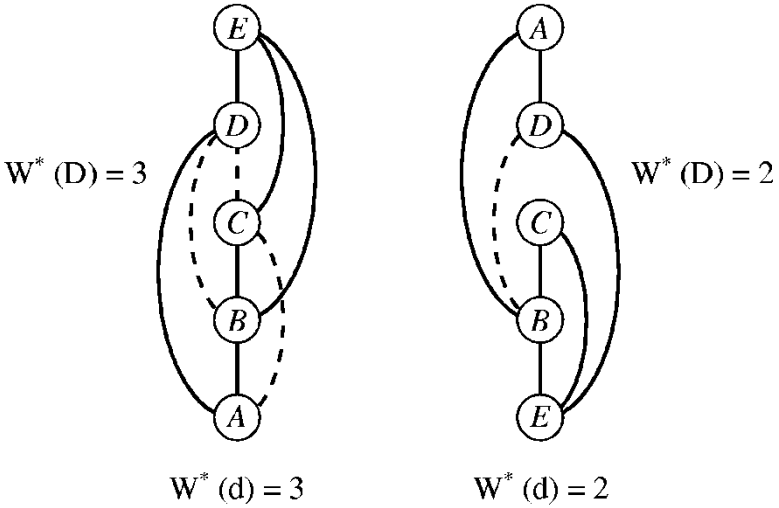
Induced-width and Tree-width

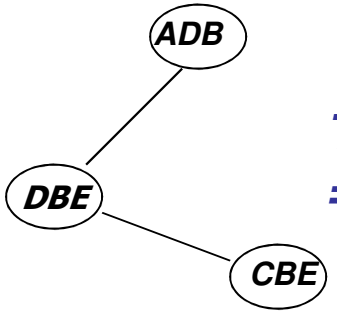
Induced-width Of ordering





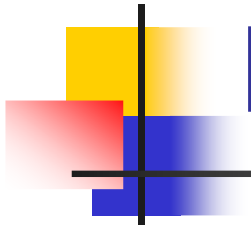
Tree-width =3





Tree-width =2

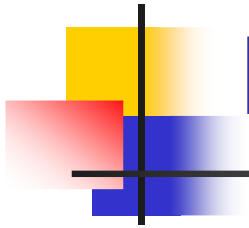
Tree-width of a graph = smallest cluster in a cluster-tree
Path-width of a graph = smallest cluster in a cluster-path



Road Map

- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination:
  - Tree-clustering
  - Constraint propagation
- Search
- Probabilistic Networks

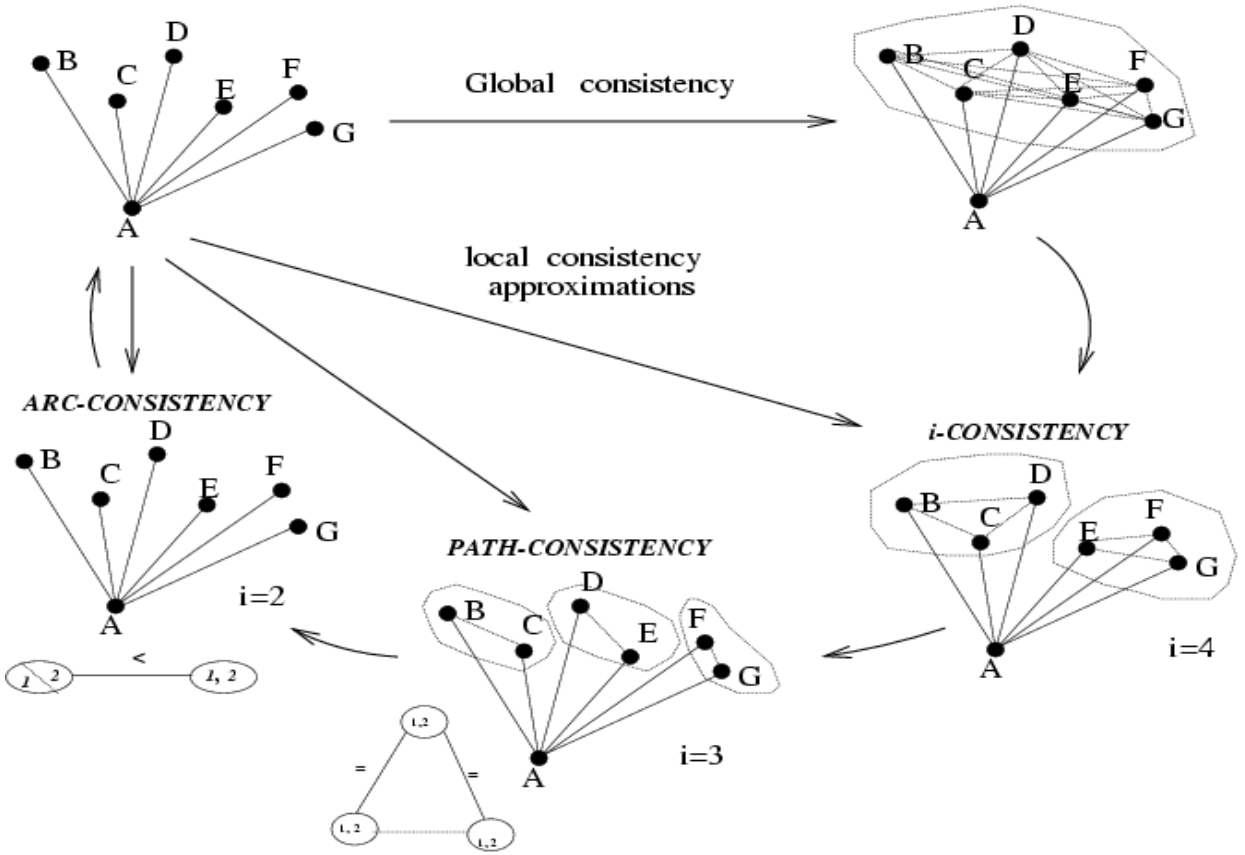
Approximating Inference: Local Constraint Propagation



- Problem: bucket-elimination/tree-clustering are intractable when *induced width* is large.

- Approximation: Do bounded inference: bound the size of recorded relations, i.e. perform *local constraint propagation (local inference)*

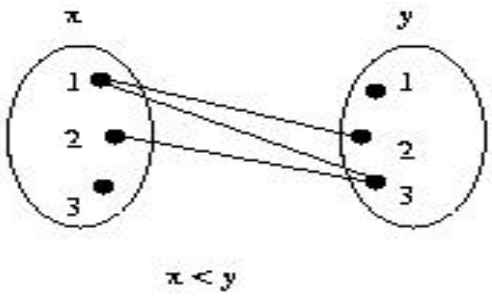
From Global to Local Consistency



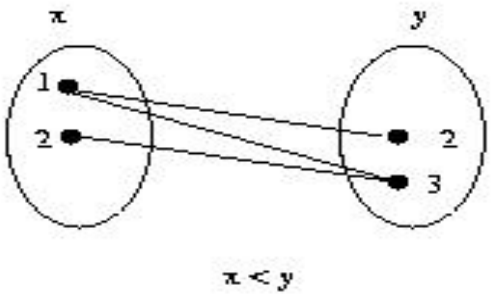
Arc-consistency

A binary constraint R(X,Y) is arc-consistent w.r.t. X is every value In x's domain has a match in y's domain.

$$R_X = \{1,2,3\},\ R_Y = \{1,2,3\},\ \text{constraint}\quad X < Y$$



(a)



(b)

Only domains are reduced:

$$R_X \leftarrow \prod_X R_{XY} \bowtie D_Y$$

Arc-consistency

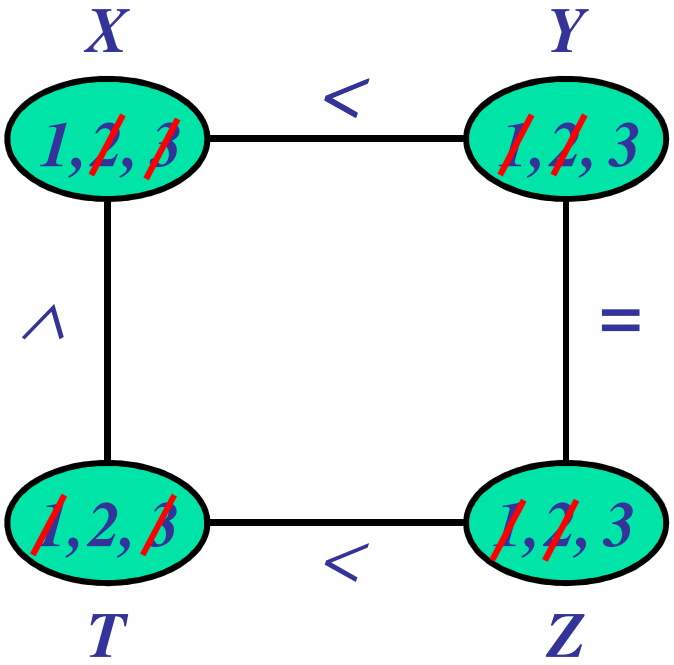
$1 \leq X, Y, Z, T \leq 3$

$X < Y$

$Y = Z$

$T < Z$

$X \leq T$



Arc-consistency

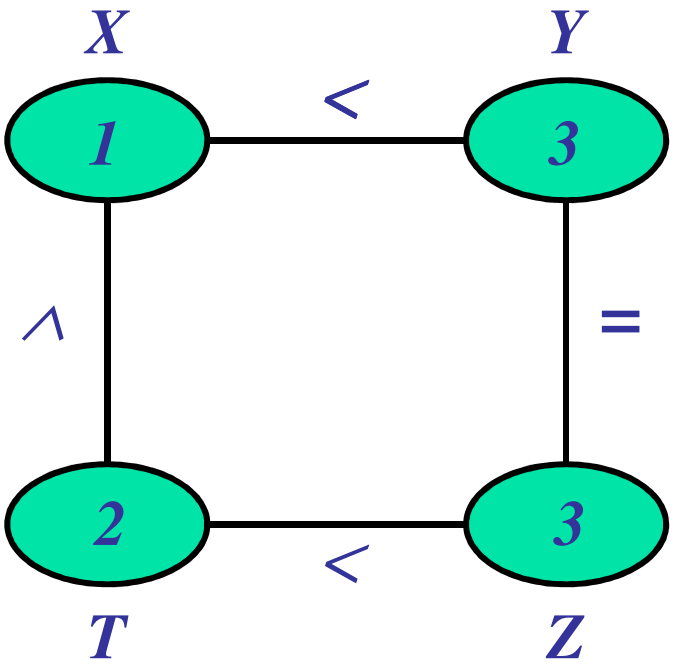
$1 \leq X, Y, Z, T \leq 3$

$X < Y$

$Y = Z$

$T < Z$

$X \leq T$



Inference of level 2

$$R_X \leftarrow \prod_X R_{XY} \bowtie D_Y$$

AC-3

AC-3($\mathcal{R}$)

input: a network of constraints $\mathcal{R} = (X, D, C)$
output: $\mathcal{R}'$ which is the largest arc-consistent network equivalent to $\mathcal{R}$

1. **for** every pair $\{x_i, x_j\}$ that participates in a constraint $R_{ij} \in \mathcal{R}$
2. $\quad queue \leftarrow queue \cup \{(x_i, x_j), (x_j, x_i)\}$
3. **endfor**
4. **while** $queue \neq \{\}$
5. $\quad$ select and delete (x_i, x_j) from $queue$
6. $\quad Revise((x_i), x_j)$
7. $\quad$ **if** $Revise((x_i), x_j)$ causes a change in D_i
8. $\quad\quad$ **then** $queue \leftarrow queue \cup \{(x_k, x_i), i \neq k\}$
9. $\quad$ **endif**
10. **endwhile**

Figure 3.5: Arc-consistency-3 (AC-3)

- Complexity: $O(ek^3)$
- Best case O(ek), since each arc may be processed in O(2k)

Arc-consistency Algorithms

- **AC-1**: brute-force, distributed $O(nek^3)$
- **AC-3**, queue-based $O(ek^3)$
- **AC-4**, context-based, optimal $O(ek^2)$
- **AC-5,6,7**,.... Good in special cases
- **Important:** applied at every node of search
- (n =# of variables, e=#constraints, k=domain size)
- Mackworth and Freuder (1977,1983), Mohr and Anderson, (1985)…

Path-consistency

- A pair (x, y) is path-consistent relative to Z, if every consistent assignment (x, y) has a consistent extension to z.

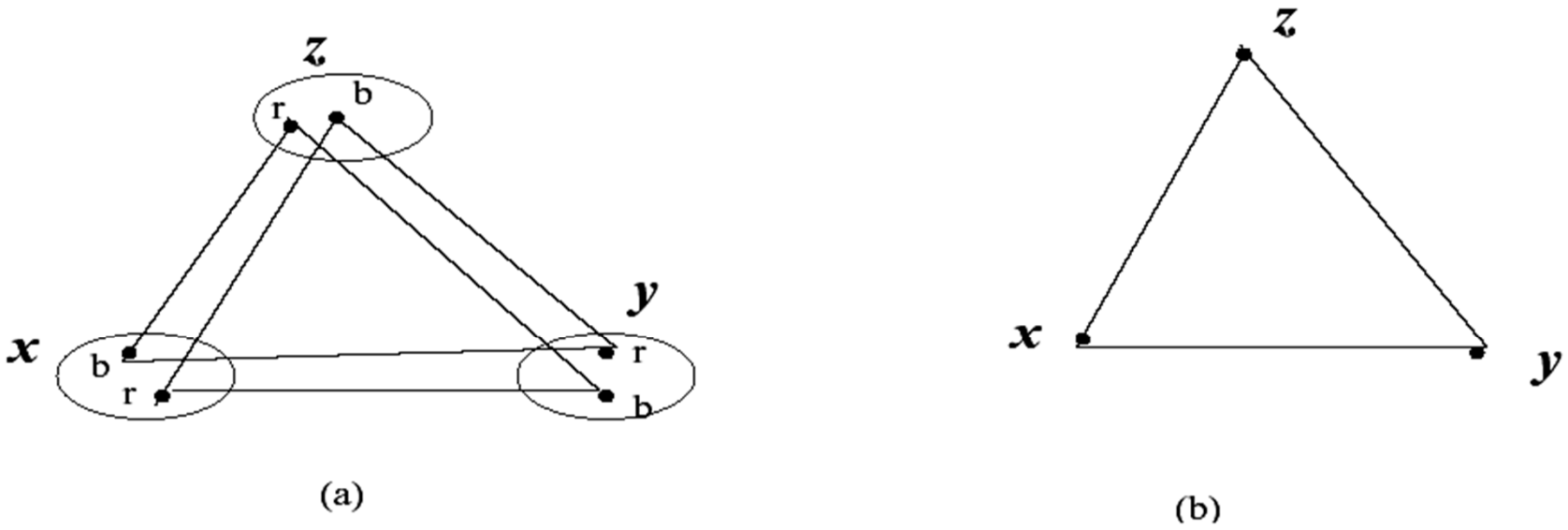


Figure 3.8: (a) The matching diagram of a 2-value graph coloring problem. (b) Graphical picture of path-consistency using the matching diagram.

Example: path-consistency

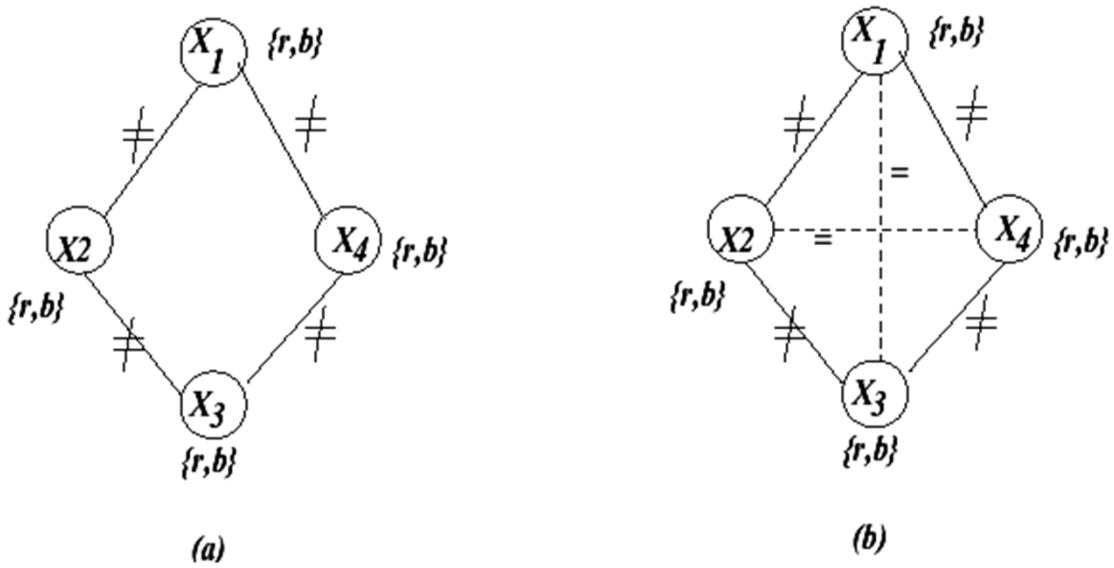


Figure 3.12: A graph-coloring graph (a) before path-consistency (b) after path-consistency

Path(3)-consistency Algorithms

- Apply Inference on 3 variables at a time (O(k^3)) until no change
$$R_{ij} \leftarrow R_{ij} \cap \pi_{ij}(R_{ik} \otimes D_k \otimes R_{kj})$$
- Path-consistency (3-consistency) adds binary constraints.
- PC-1: $O(n^5k^5)$
- PC-2: $O(n^3k^5)$
- PC-4 optimal: $O(n^3k^3)$

Local i-consistency, or Bounded Inference (i)

i-consistency: ***Any consistent assignment to any i-1 variables is consistent with at least one value of any i-th variable***

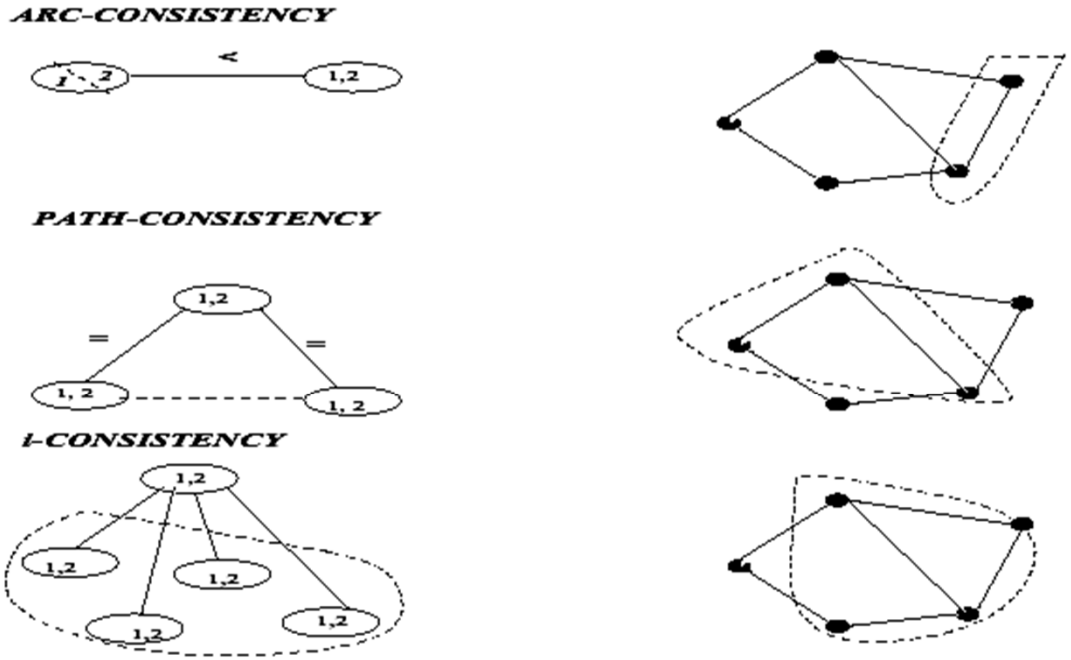


Figure 3.17: The scope of consistency enforcing: (a) arc-consistency, (b) path-consistency, (c) i-consistency

Gausian and Boolean Propagation, Resolution

- Linear inequalities

$$x + y + z \leq 15, z \geq 13 \Rightarrow$$

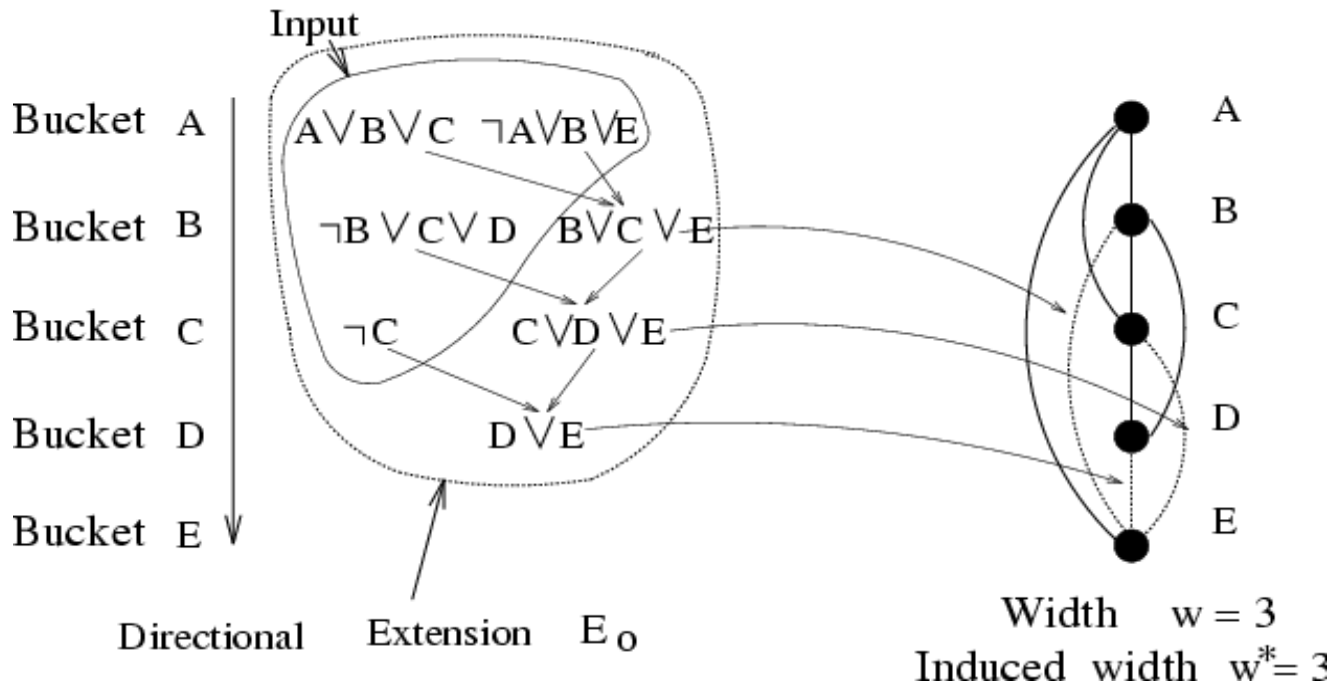
$$\boxed{x \leq 2, y \leq 2}$$

- Boolean constraint propagation, unit resolution

$$(A \vee B \vee \neg C), (\neg B) \Rightarrow$$

$$(A \vee \neg C)$$

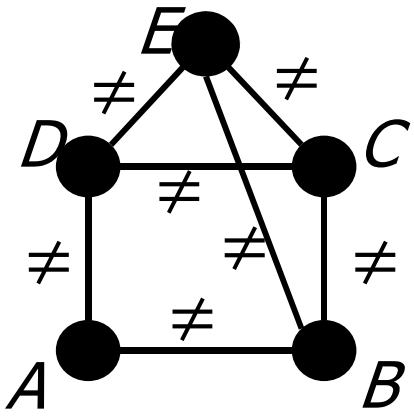
Directional Resolution ⇔ Adaptive Consistency



$|bucket_i| = O(\exp(w^*))$

DR time and space : $O(n \exp(w^*))$

Directional i-Consistency



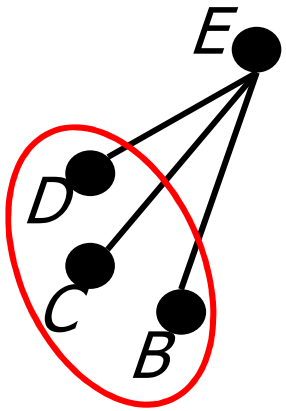
E : E ≠ D, E ≠ C, E ≠ B

D : D ≠ C, D ≠ A

C : C ≠ B

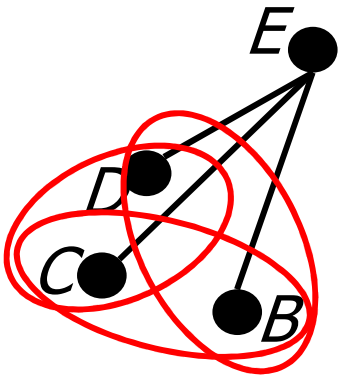
B : A ≠ B

A :



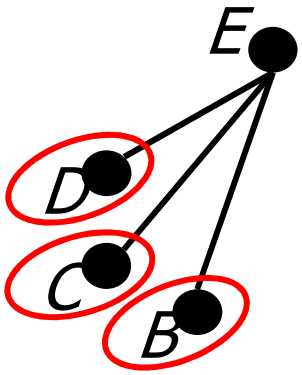
Adaptive

R_{DCB}



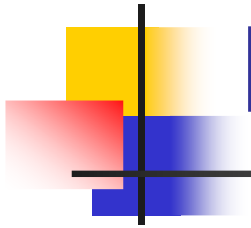
d-path

R_{DC}, R_{DB}
R_{CB}



d-arc

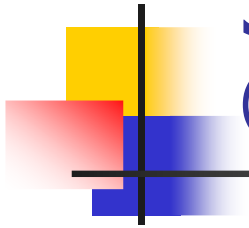
R_D
R_C
R_D



Road Map

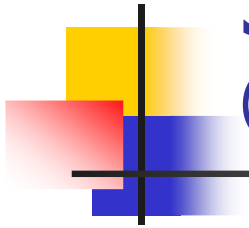
- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination:
  - Tree-clustering
  - Constraint propagation
- Search
- Probabilistic Networks

Search in Constraint Networks
(for a consistent solution, all solutions, counting)



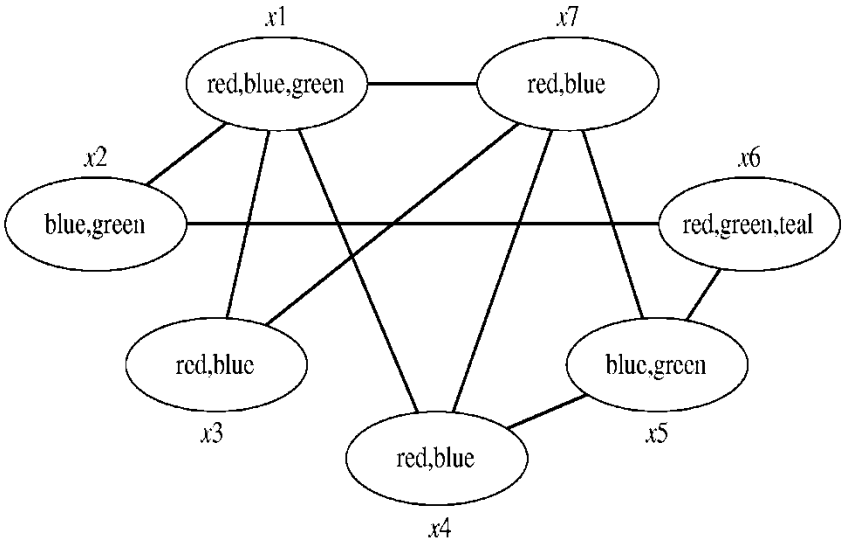
- Improving search by bounded-inference in branching ahead
- Improving search by looking-back
- The alternative AND/OR search space

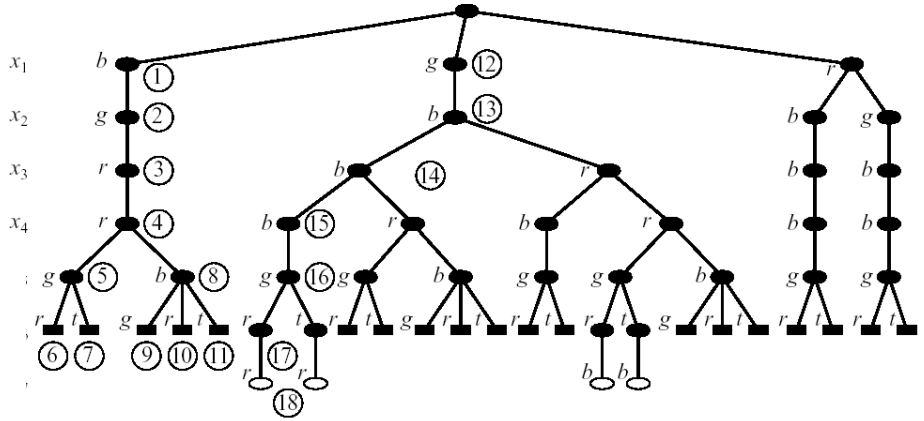
Search in Constraint Networks
(for a consistent solution, all solutions, counting)



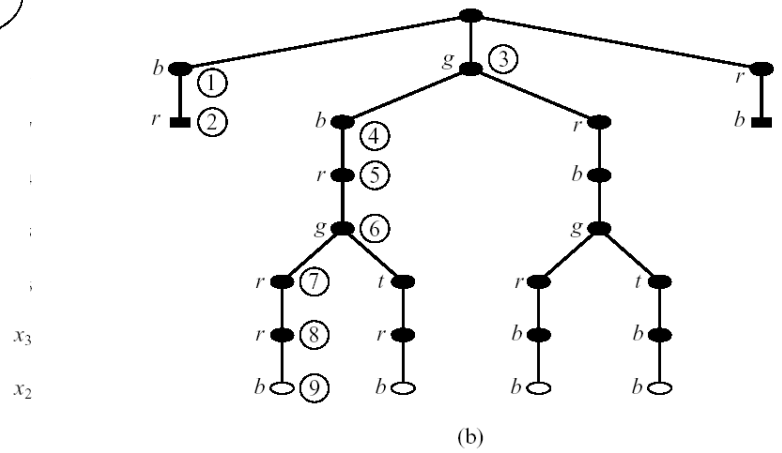
- Improving search by bounded-inference in branching ahead
- Improving search by looking-back
- The alternative AND/OR search space

Backtracking Search for a Solution



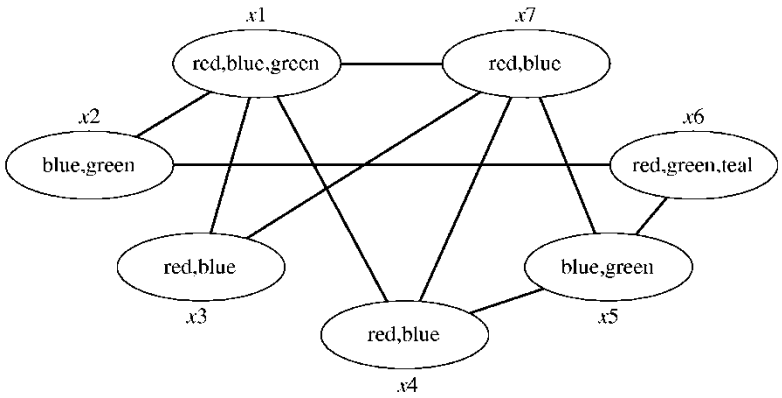


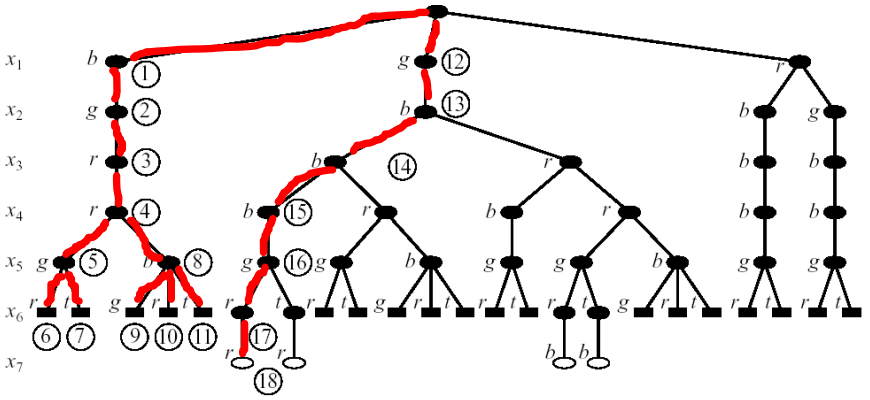
(a)

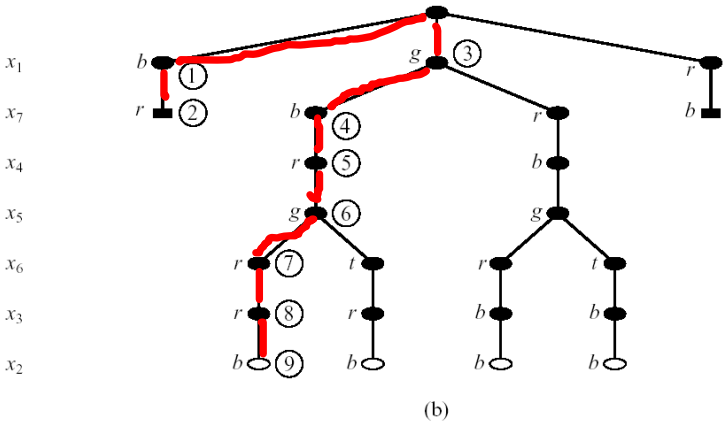


(b)

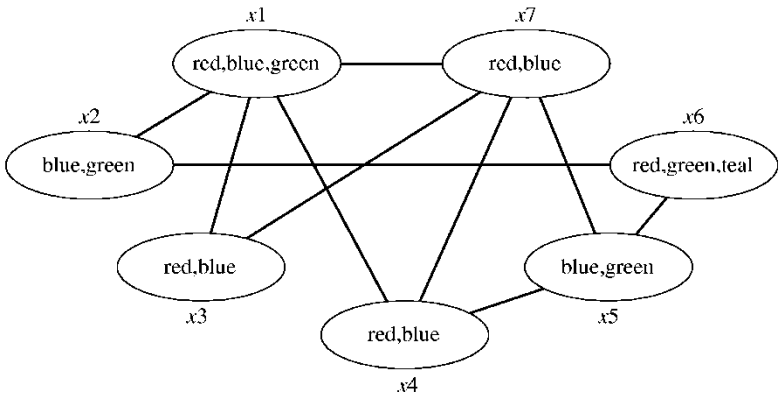
Backtracking Search for a Solution

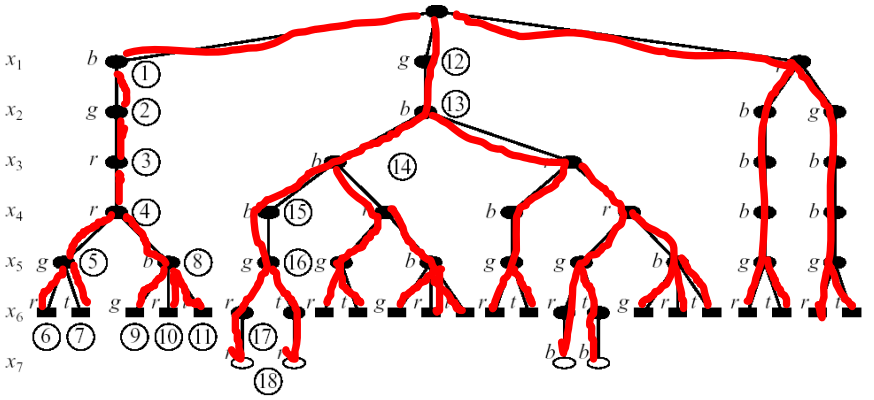




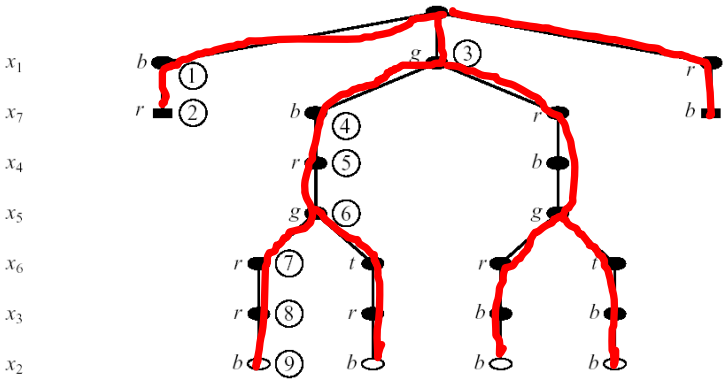


Backtracking Search for All Solutions



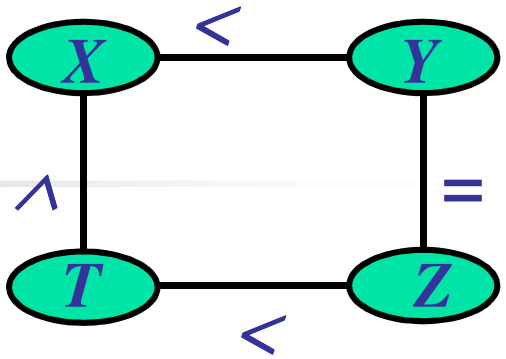


(a)

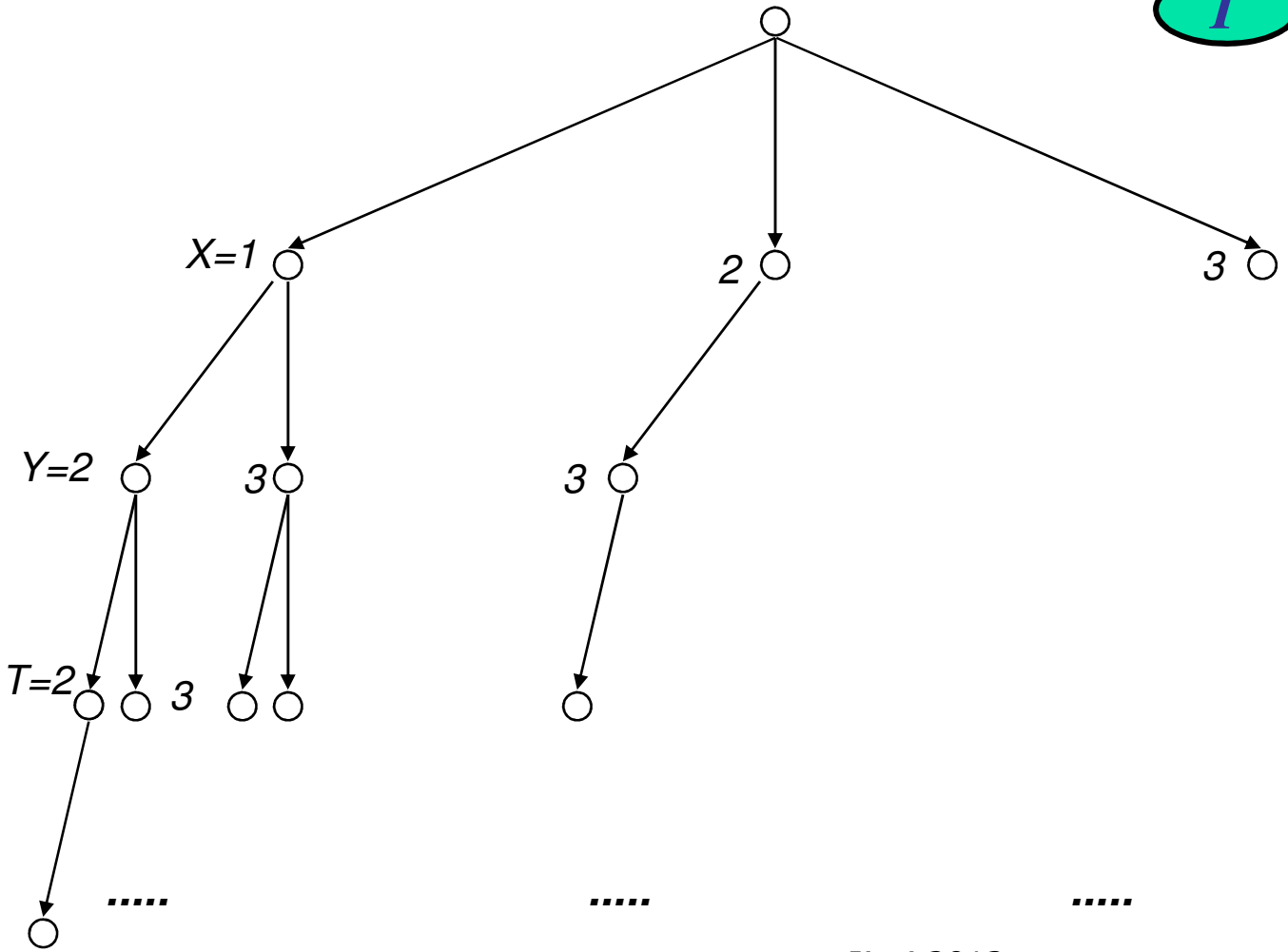


(b)

The Search Space Before Arc-Consistency



D={1,2,3}



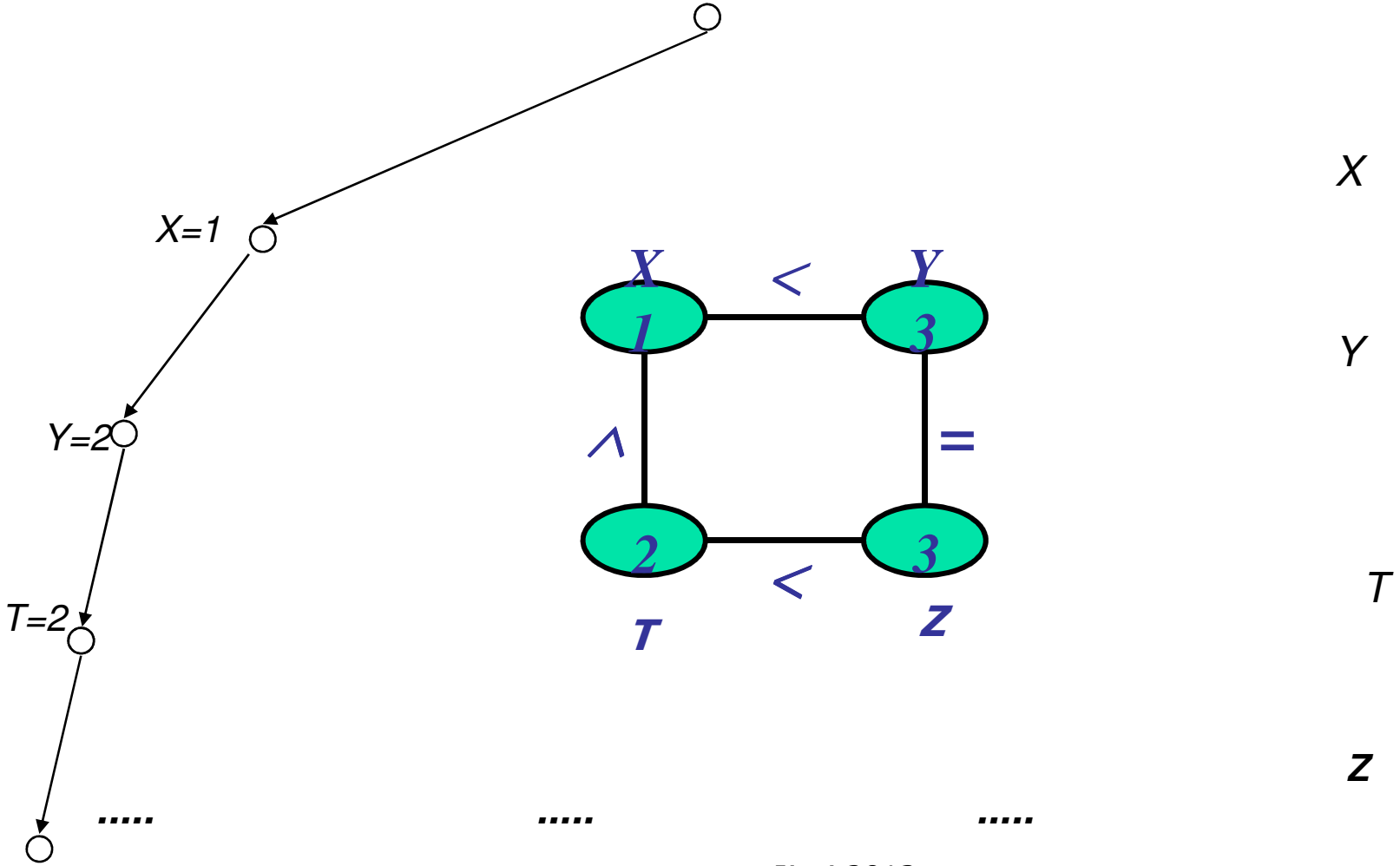
X

Y

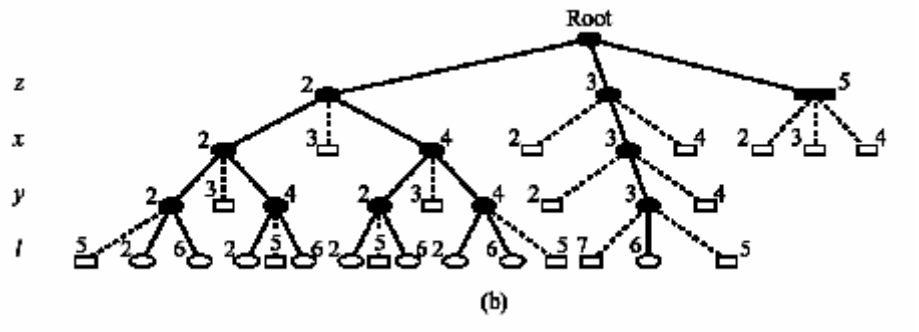
T

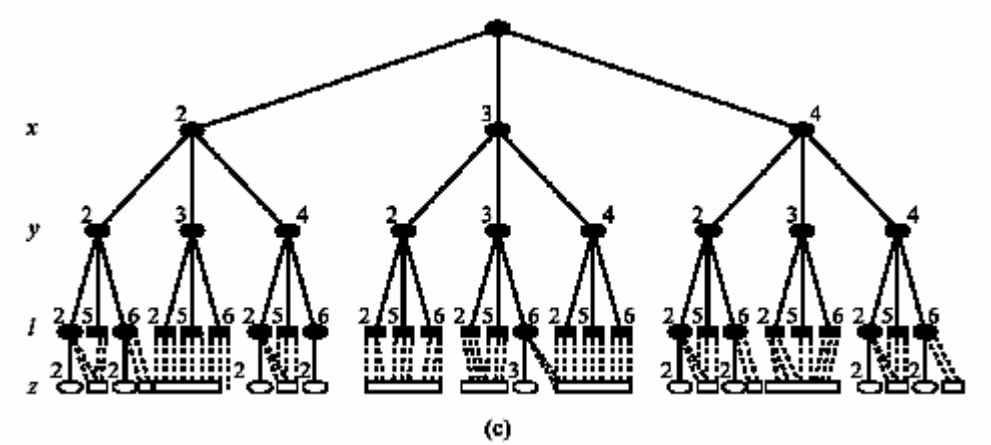
Z

The Search Space After Arc-Consistency

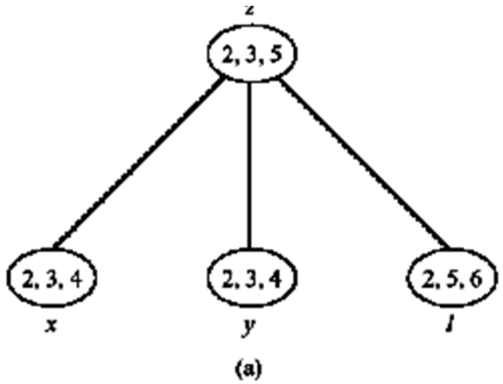


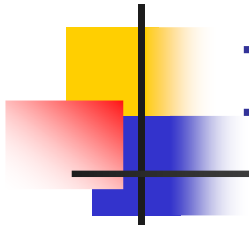
The Effect of Variable Ordering





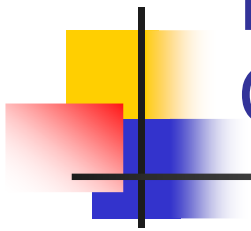
z divides x, y and t





Improving Backtracking O(exp(n))

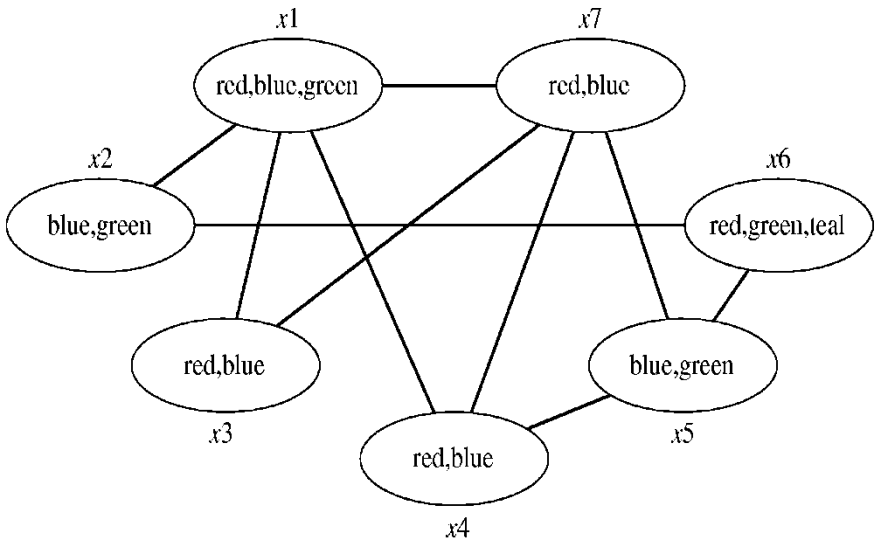
- Before search: (reducing the search space)
  - Arc-consistency, path-consistency, i-consistency
  - Variable ordering (fixed)
- During search:
  - Look-ahead schemes:
    - value ordering/pruning (*choose a least restricting value*),
    - variable ordering (*Choose the most constraining variable)*
  - Look-back schemes:
    - Backjumping
    - Constraint recording
    - Dependency-directed backtracking



Looking-Ahead: Constraint Propagation in Search

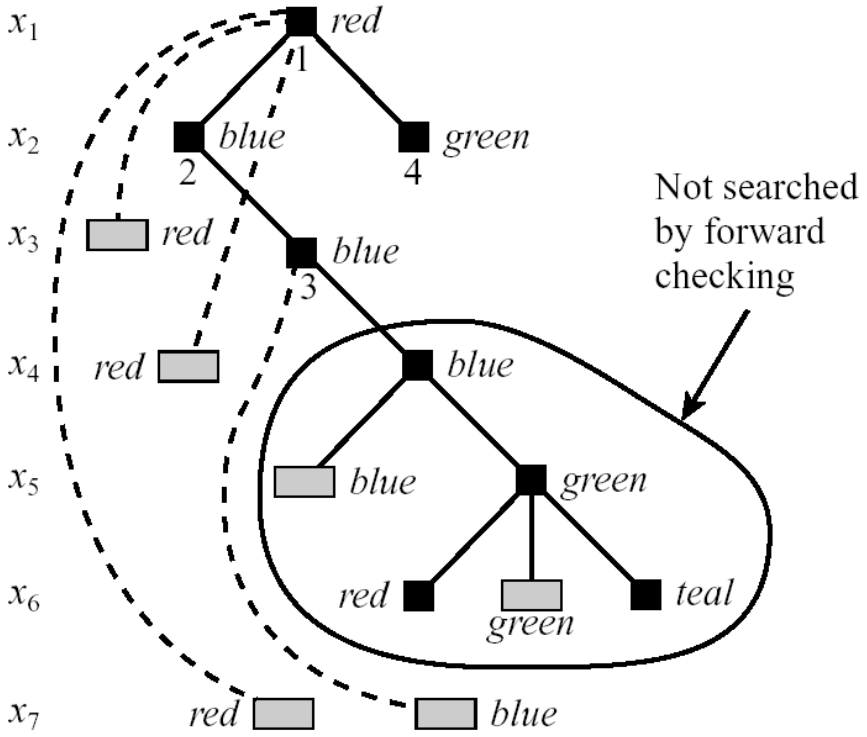
- Apply some level of constraint propagation at each node,
  - Forward-checking (FC)
  - Arc-consistency (MAC)
- Then:
  - Value pruning: prune values that lead to deadend
  - Variable ordering: choose a variable that leaves least options open

Forward-Checking for Value Ordering

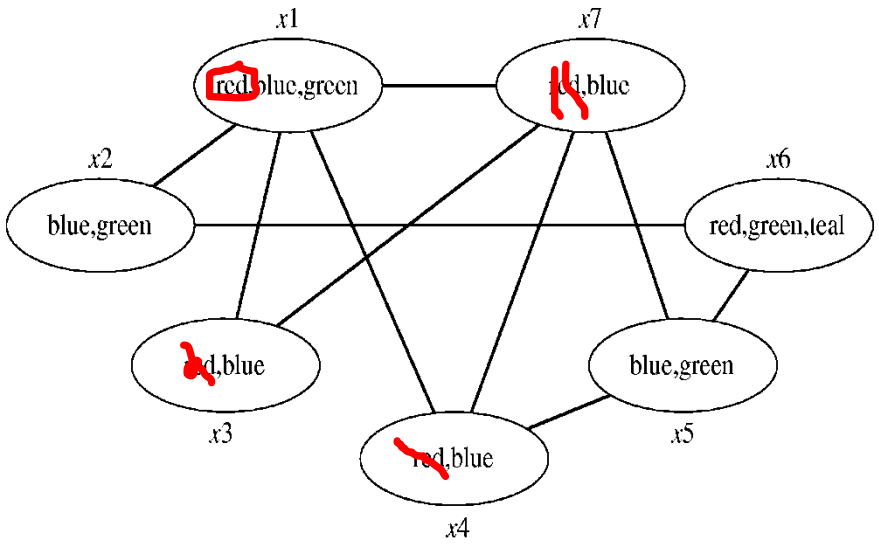


FC overhead: $O(ek^2)$

MAC overhead: $O(ek^3)$

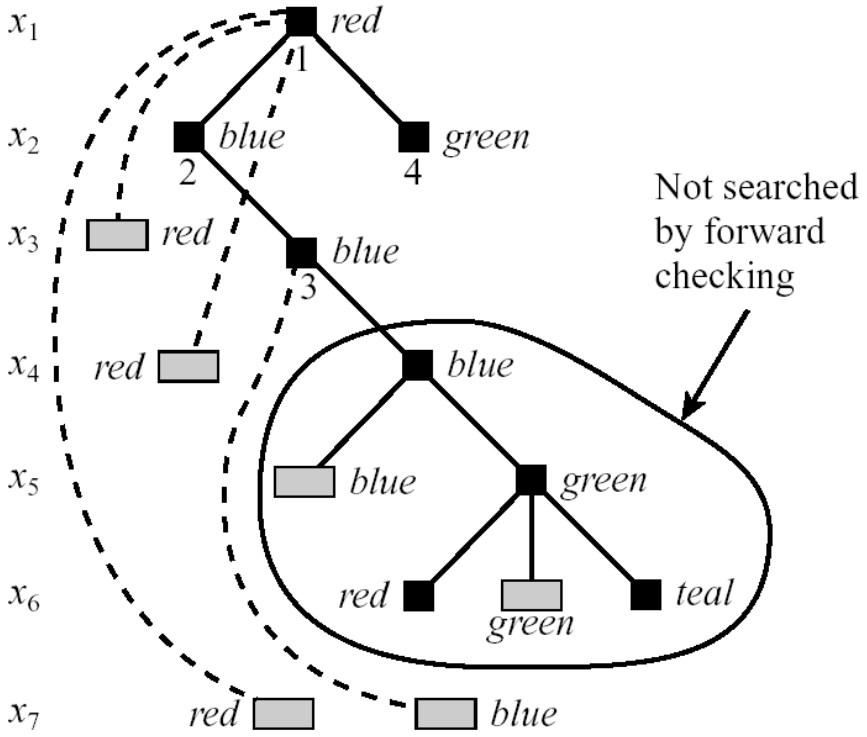


Forward-Checking for Value Ordering

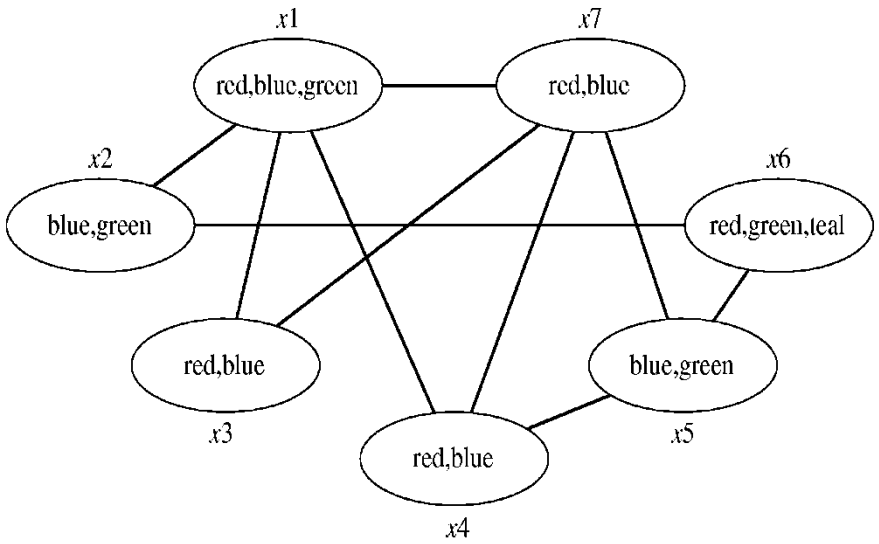


FW overhead: $O(ek^2)$

MAC overhead: $O(ek^3)$

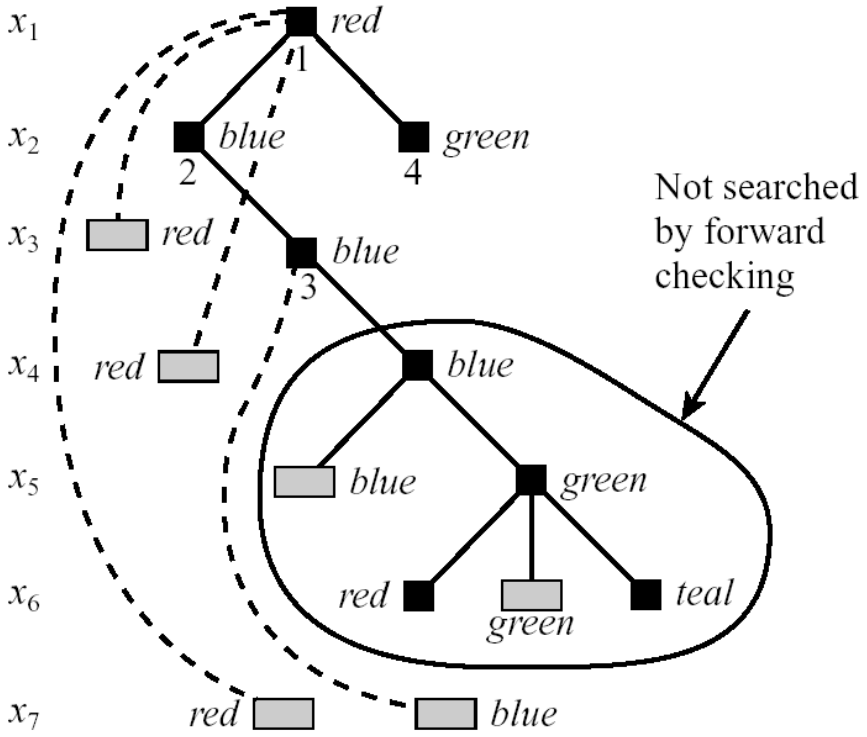


Forward-Checking, Variable Ordering



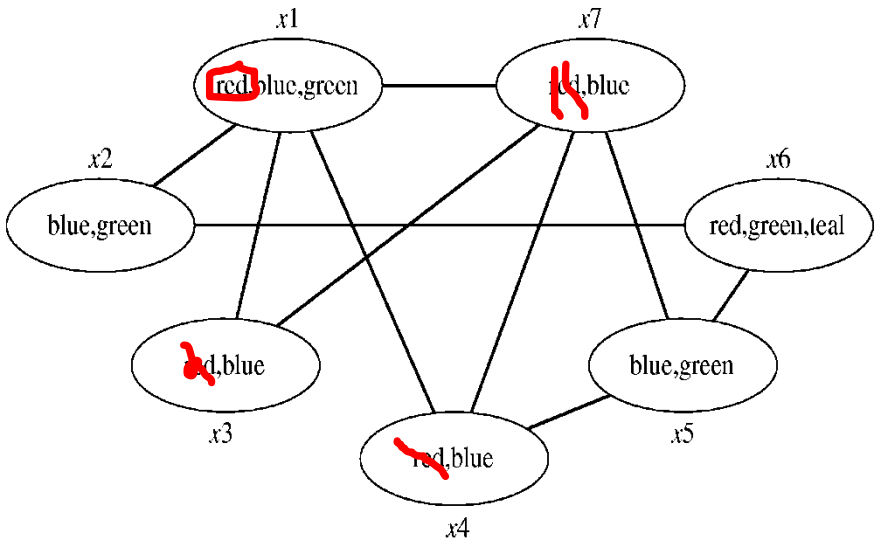
FW overhead: $O(ek^2)$

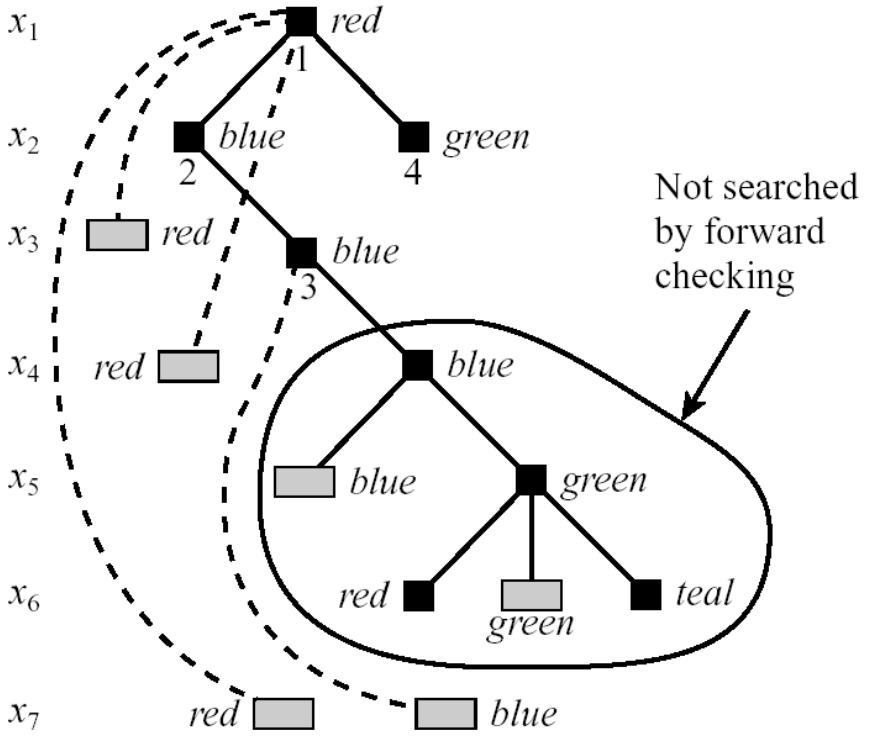
MAC overhead: $O(ek^3)$



Forward-Checking, Variable Ordering

After X1 = red choose X3 and not X2



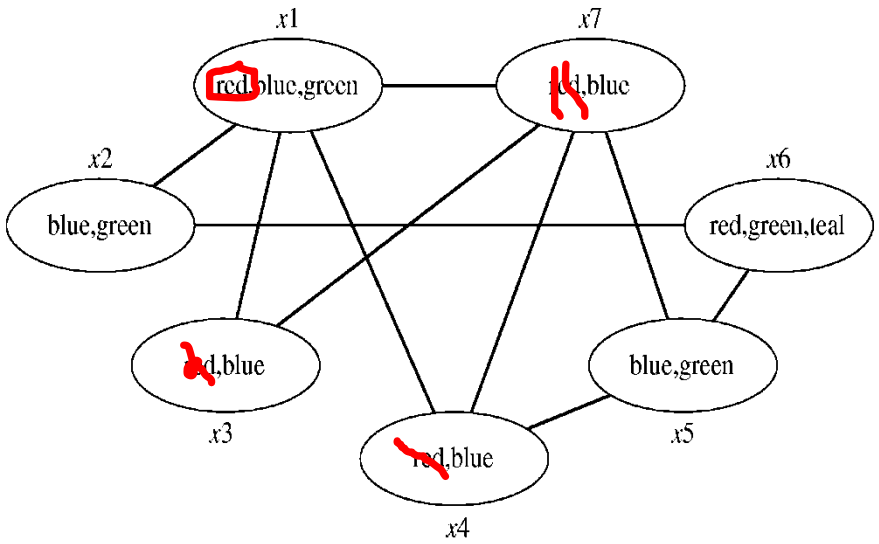


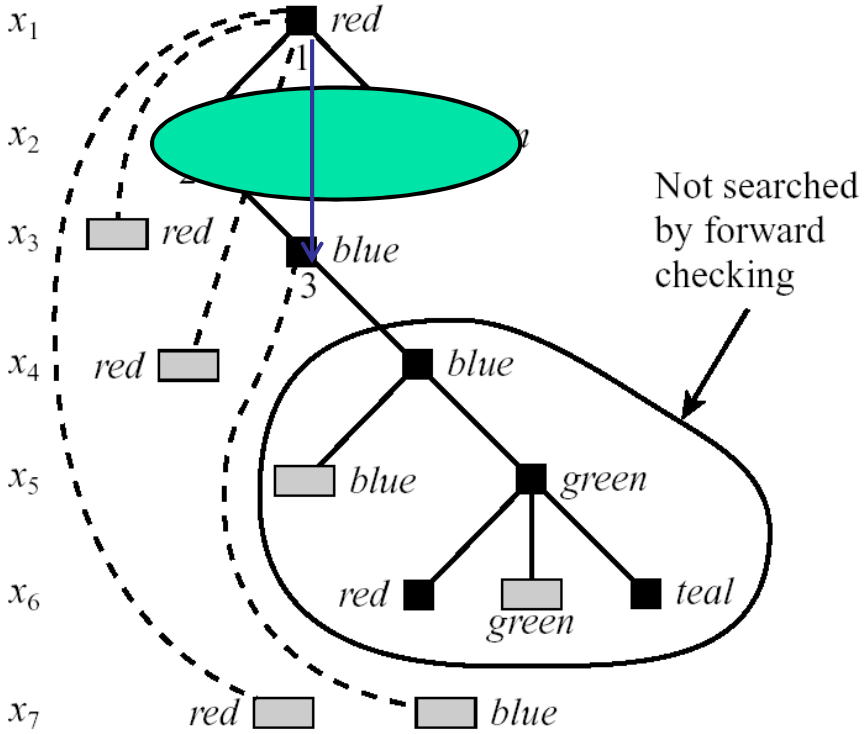
FW overhead: $O(ek^2)$

MAC overhead: $O(ek^3)$

Forward-Checking, Variable Ordering

After X1 = red choose X3 and not X2



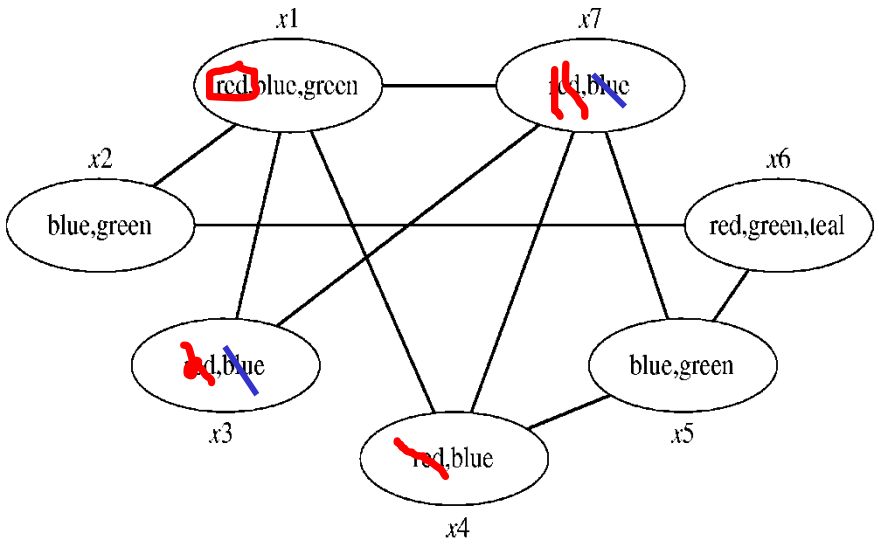


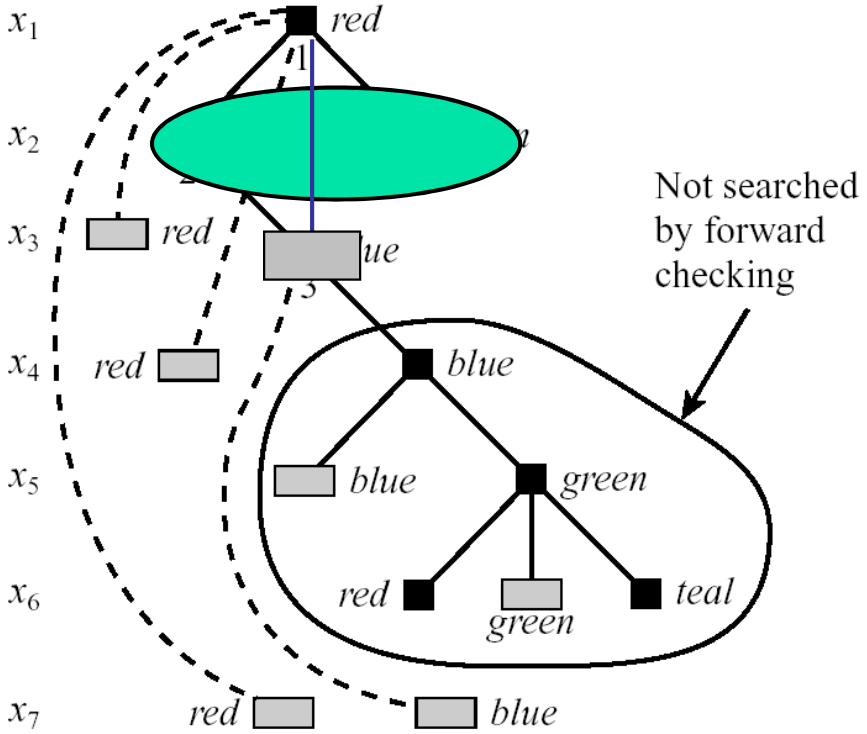
FW overhead: $O(ek^2)$

MAC overhead: $O(ek^3)$

Forward-Checking, Variable Ordering

After X1 = red choose X3 and not X2

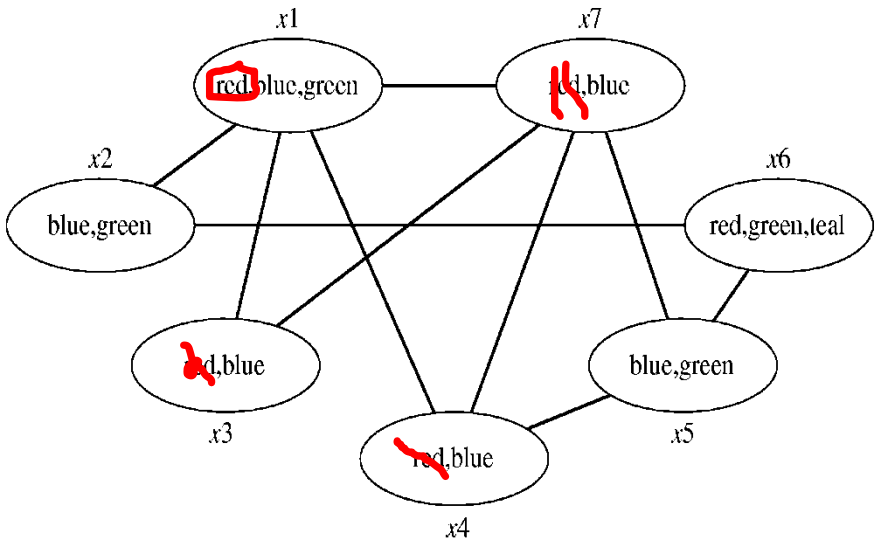




FW overhead: $O(ek^2)$

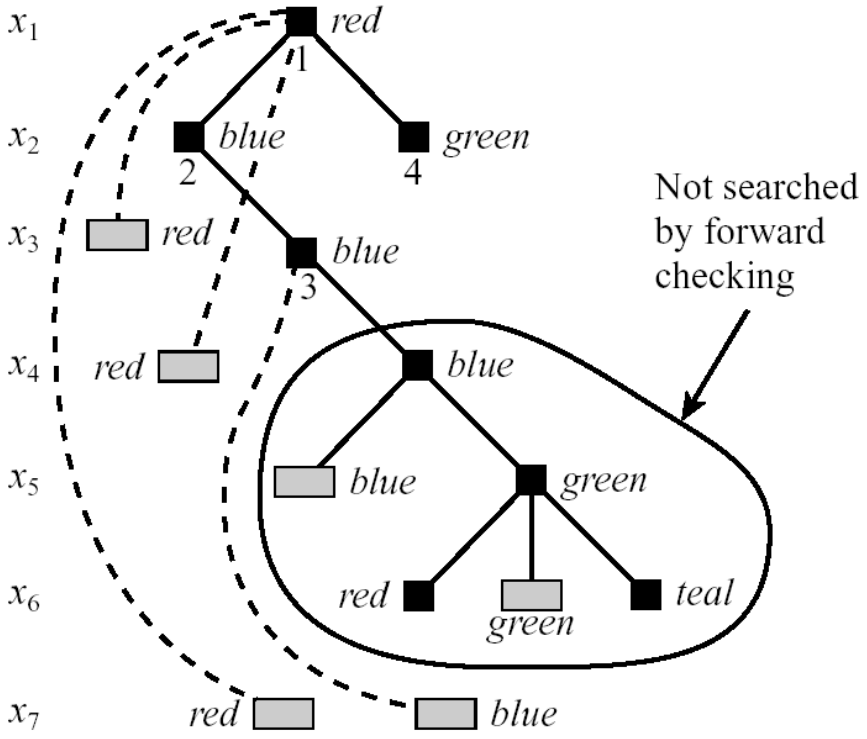
MAC overhead: $O(ek^3)$

Arc-consistency for Value Ordering



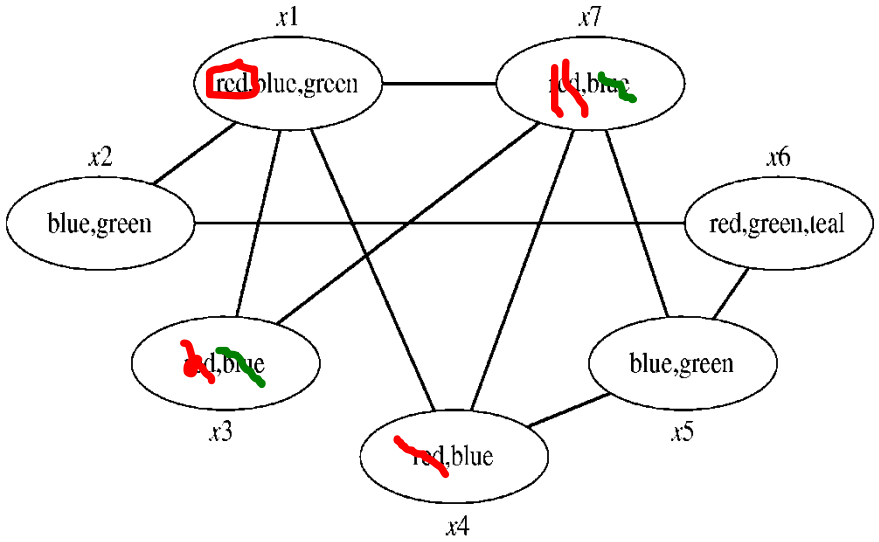
FW overhead: $O(ek^2)$

MAC overhead: $O(ek^3)$



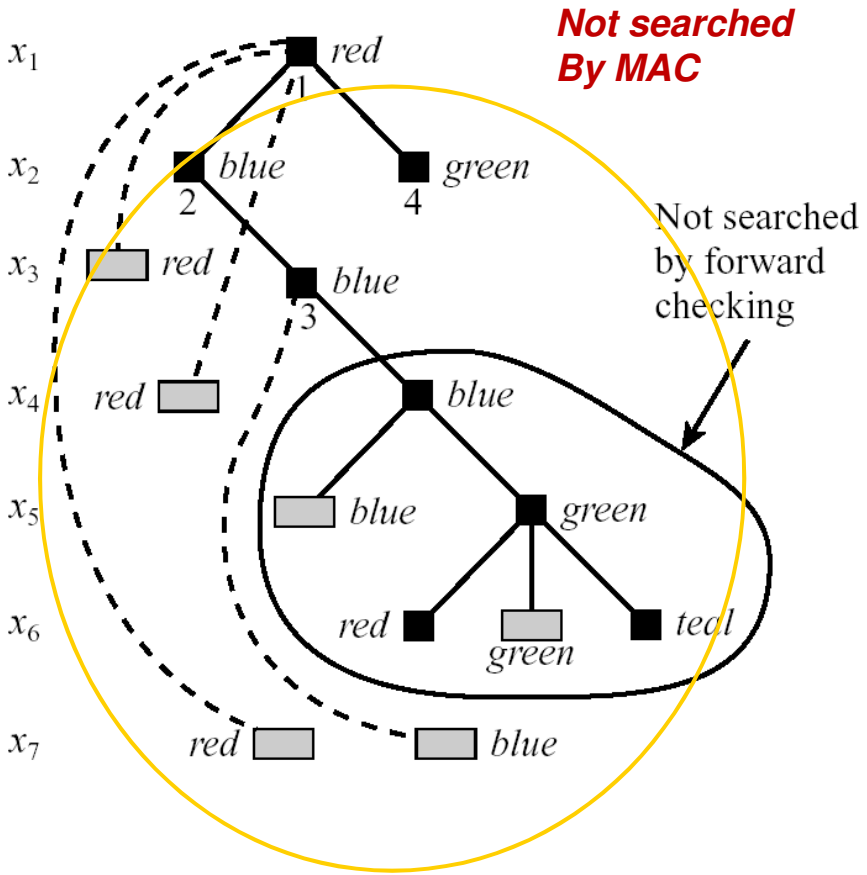
Arc-Consistency for Value Ordering

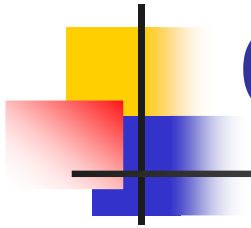
Arc-consistency prunes x1=red
Prunes the whole tree



FW overhead: $O(ek^2)$

MAC overhead: $O(ek^3)$





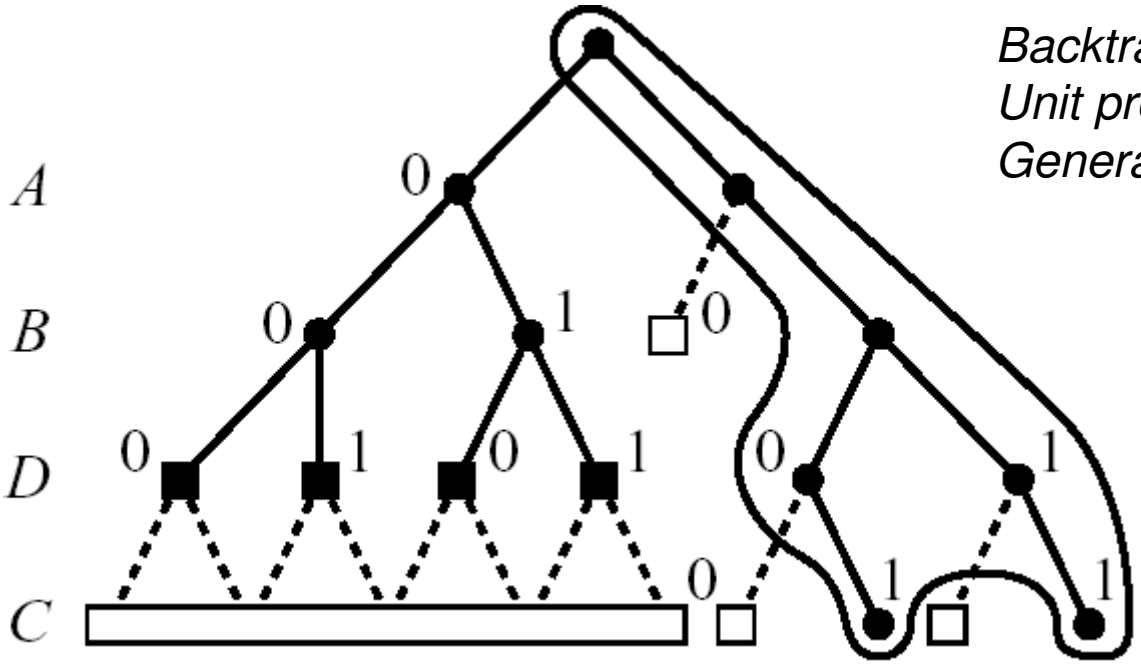
Constraint Programming

- Constraint solving embedded in programming languages
- Allows flexible modeling with algorithms
- Logic programs + forward checking
- Eclipse, ILog, OPL
- Using only look-ahead schemes

Looking-Ahead for SAT: DLL

example: (~AVB)(~CVA)(AVBVD)(C)

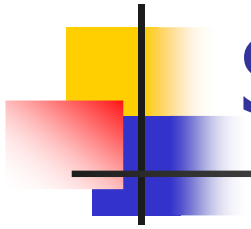
(Davis, Logeman and Laveland, 1962)



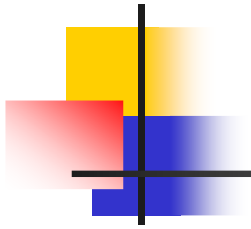
Backtracking look-ahead with Unit propagation= Generalized arc-consistency

Only enclosed area will be explored with unit-propagation

Road Map: Search in Constraint Networks



- Improving search by bounded-inference in branching ahead
- Improving search by looking-back
- The alternative AND/OR search space



Look-back: Backjumping / Learning

- Backjumping:
  - In deadends, go back to the most recent culprit.
- Learning:
  - constraint-recording, no-good recording.
  - good-recording

Look-Back: Backjumping

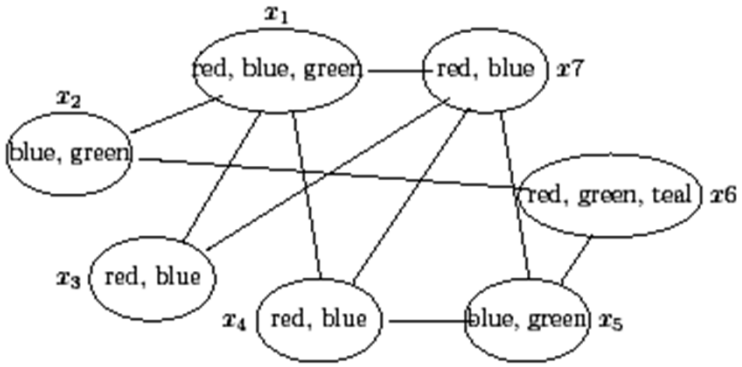
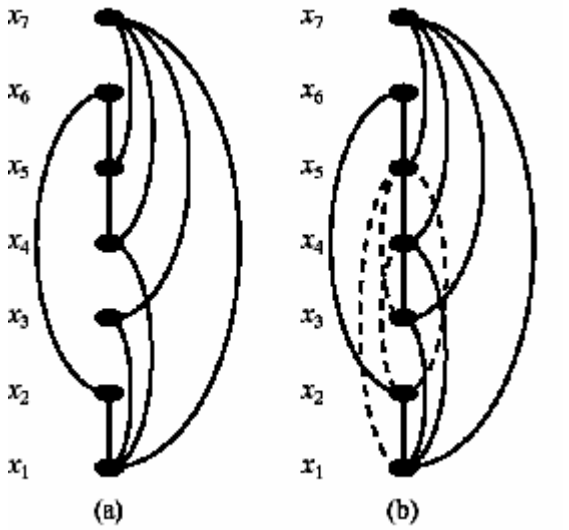
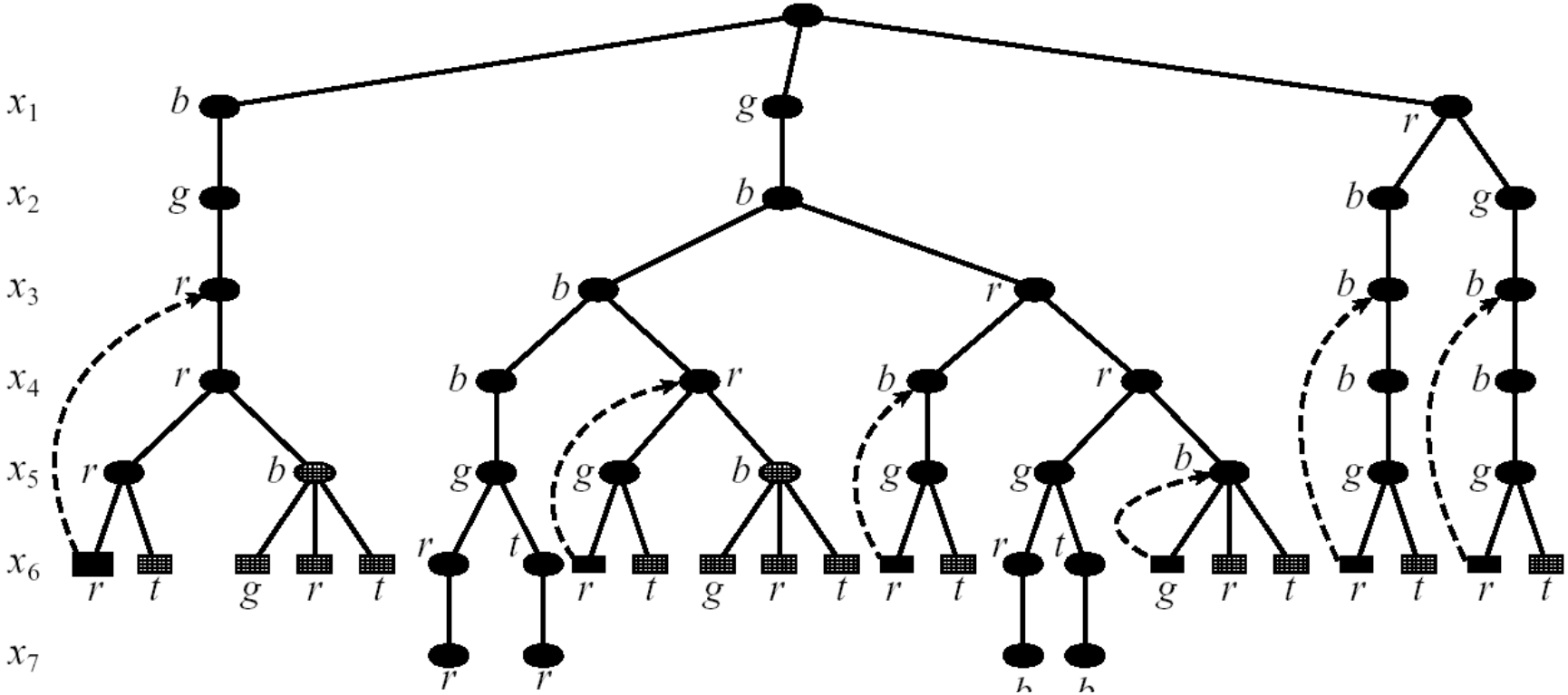


Figure 6.1: A modified coloring problem.

- (X1=r,x2=b,x3=b,x4=b,x5=g,x6=r,x7={r,b})
- (r,b,b,b,g,r) **conflict set** of x7
- (r,-,b,b,g,-) c.s. of x7
- (r,-,b,-,-,-,-) **minimal conflict-set**
- **Leaf deadend**: (r,b,b,b,g,r)
- Every conflict-set is a **no-good**

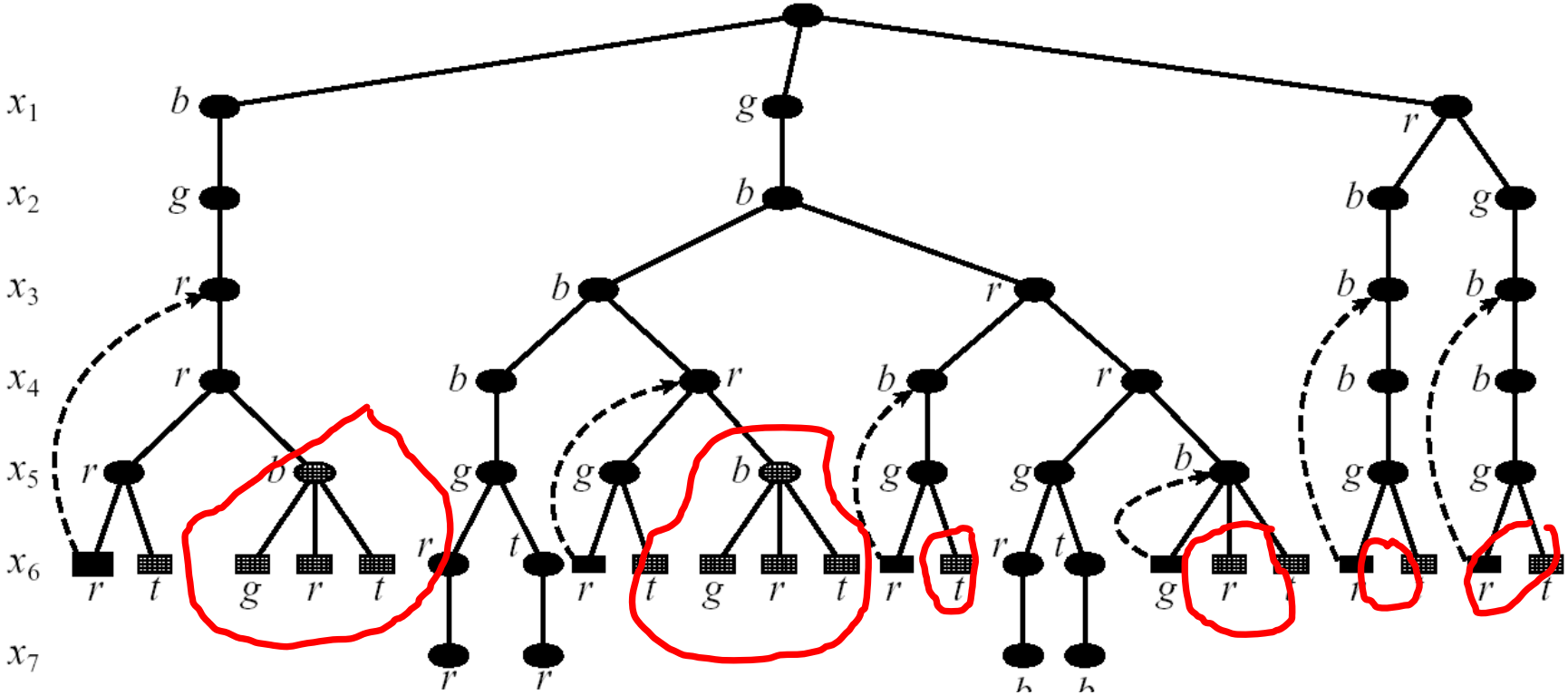


Jumps at dead-ends (Gascnnig-style 1977)



Example 6.3.1 In Figure 6.4, all of the backjumps illustrated lead to internal dead-ends, except for the jump back to $(\langle x_1, green\rangle, \langle x_2, blue\rangle, \langle x_3, red\rangle, \langle x_4, blue\rangle)$, because this is the only case where another value exists in the domain of the culprit variable. □

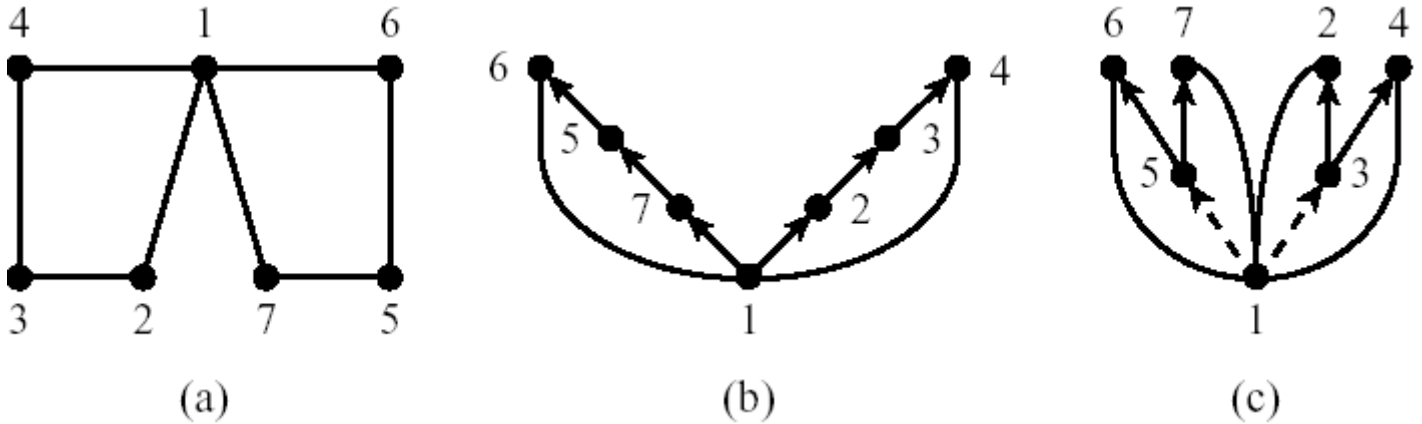
Jumps at Dead-Ends (Gascnnig 1977)



Example 6.3.1 In Figure 6.4, all of the backjumps illustrated lead to internal dead-ends, except for the jump back to $(\langle x_1, green\rangle, \langle x_2, blue\rangle, \langle x_3, red\rangle, \langle x_4, blue\rangle)$, because this is the only case where another value exists in the domain of the culprit variable. □

Complexity of Backjumping

Graph-based and conflict-based backjumpint



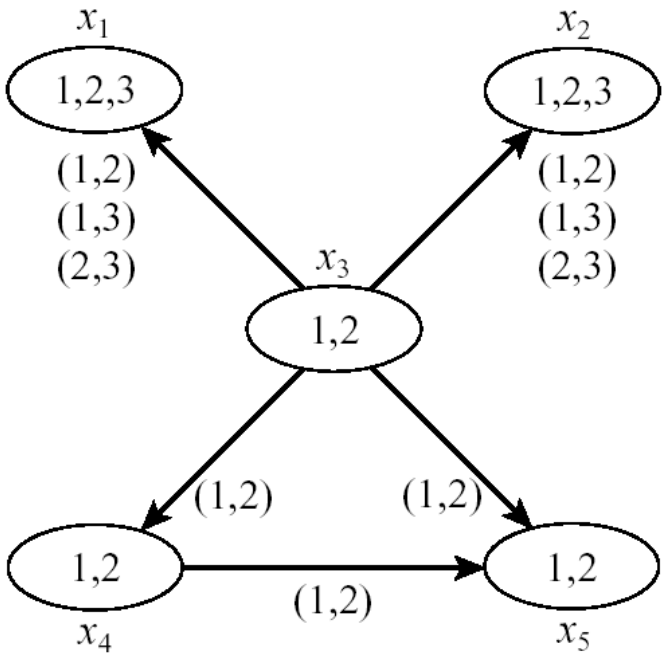
Simple: always jump back to parent in pseudo tree

*Complexity for csp: exp(w*log n)*

From exp(n) to exp(w*logn) while linear space

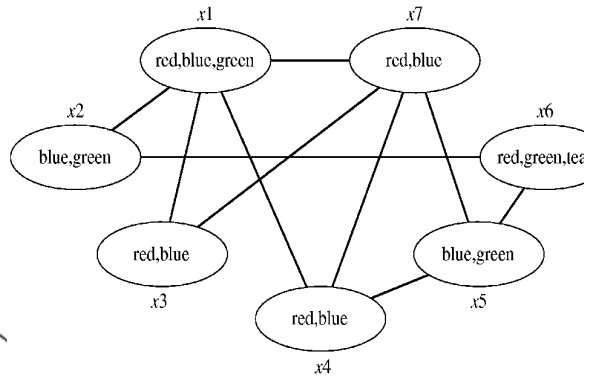
Look-back: No-good Learning

Learning means recording conflict sets used as constraints to prune future search space.



- (x1=2,x2=2,x3=1,x4=2) is a dead-end
- Conflicts to record:
  - (x1=2,x2=2,x3=1,x4=2) 4-ary
  - (x3=1,x4=2) binary
  - (x4=2) unary

No-good Learning Example



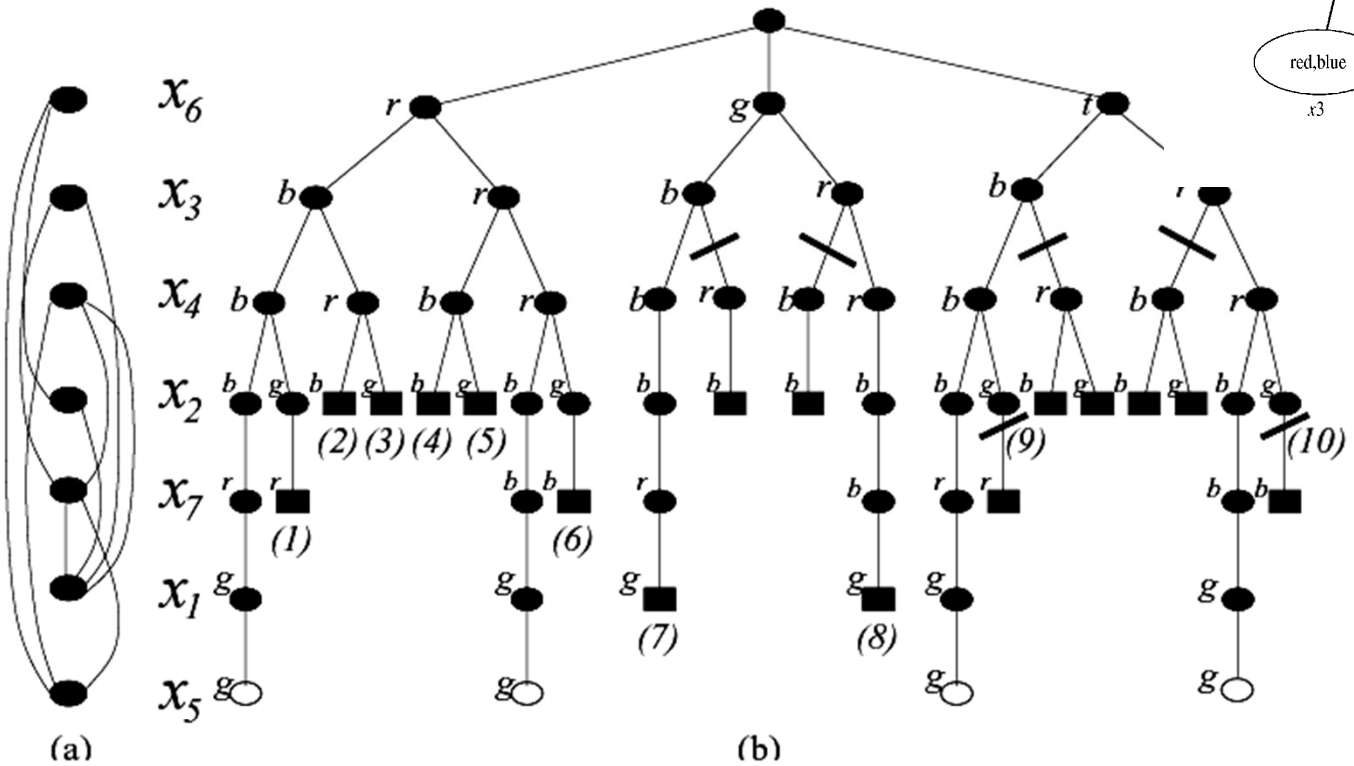


Figure 6.9: The search space explicated by backtracking on the CSP from Figure 6.1, using the variable ordering $(x_6, x_3, x_4, x_2, x_7, x_1, x_5)$ and the value ordering (*blue, red, green, teal*). Part (a) shows the ordered constraint graph, part (b) illustrates the search space. The cut lines in (b) indicate branches not explored when graph-based learning is used.

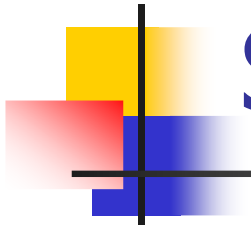
Complexity of Nogood-Learning for consistency

- The complexity of learning along d is time and space exponential in w*(d):
  - The number of dead-ends is bounded by $O(nk^{w^*(d)})$
  - Number of constraint tests per dead-end are $O(e)$

Space complexity is $O(nk^{w^*(d)})$

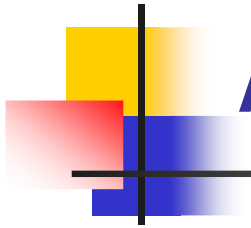
Time complexity is $O(n^2 e \cdot k^{w^*(d)+1})$

No-good Learning reduces time to O(exp(w*)+1) but requires O(exp(w*)) space.



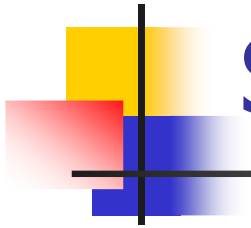
Summary: Search Principles

- Constraint propagation prunes search space
- Constraint propagation yields good advise for how to branch and where to go
- Backjumping and no-good learning helps prune search space and revise problem.



All Solutions and Counting

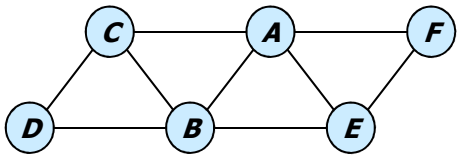
- For all solutions and counting we will see
  - The impact of problem decomposition
  - The additional impact of Good learning
  - → AND/OR search spaces



Search in Graphical Models

- Improving search by bounded-inference in branching ahead
- Improving search by looking-back
- The alternative AND/OR search space

#CSP - Tree DFS Traversal

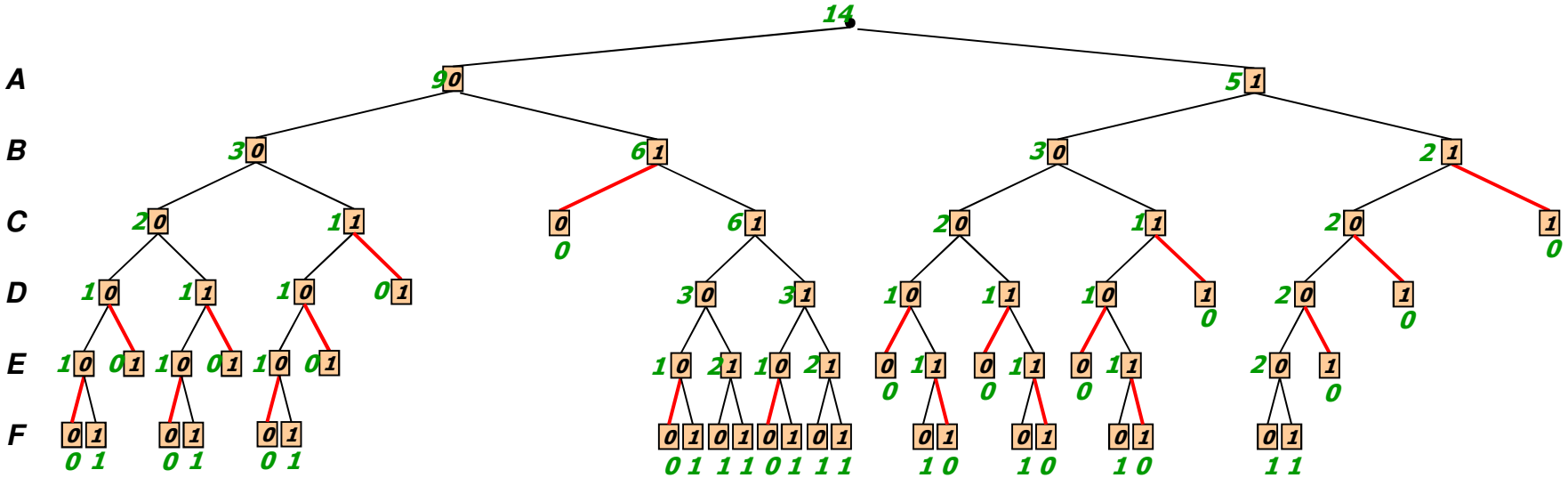


A	B	C	R_{ABC}
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

B	C	D	R_{BCD}
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

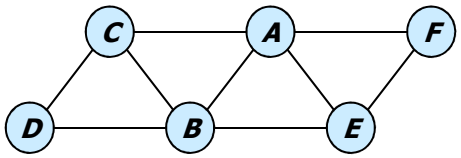
A	B	E	R_{ABE}
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

A	E	F	R_{AEF}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



Value of node = number of solutions below it

#CSP - OR Search Tree



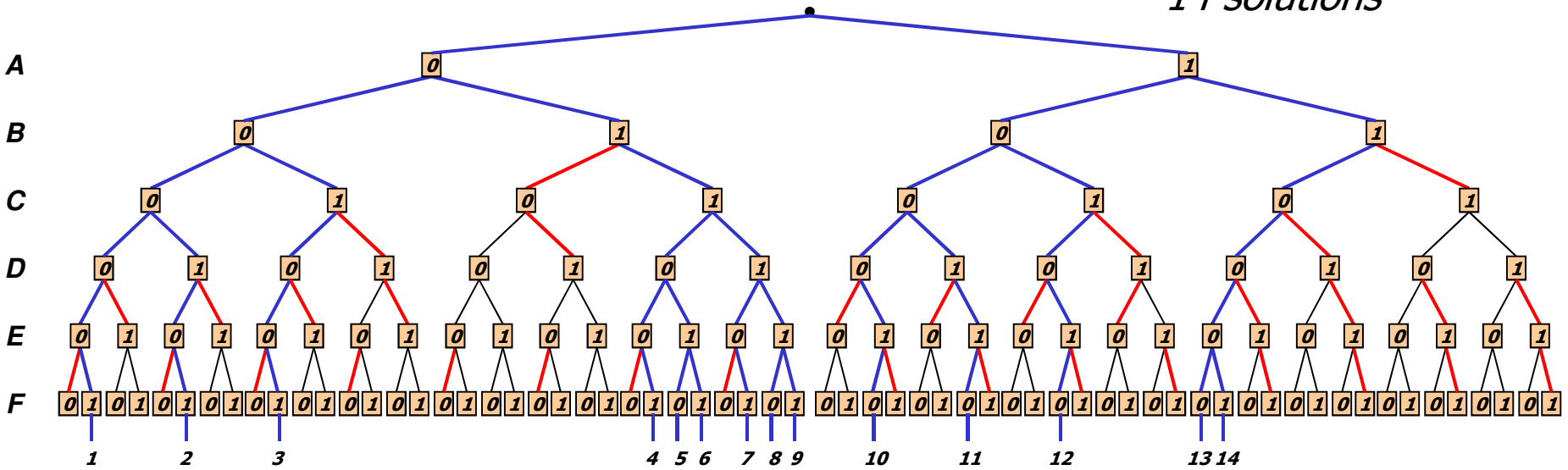
A	B	C	R_{ABC}
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

B	C	D	R_{BCD}
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

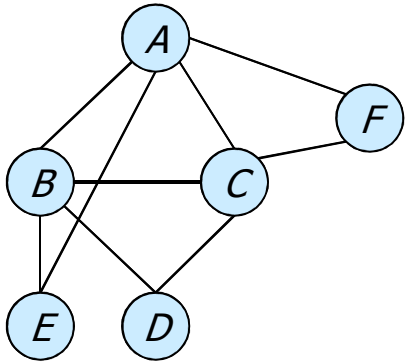
A	B	E	R_{ABE}
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

A	E	F	R_{AEF}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

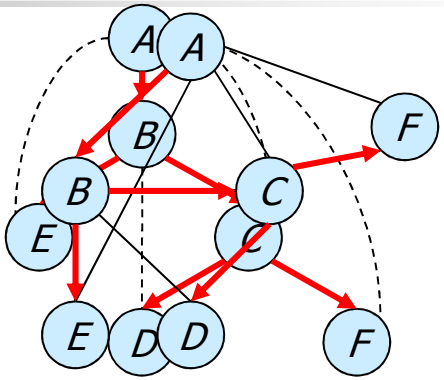
14 solutions



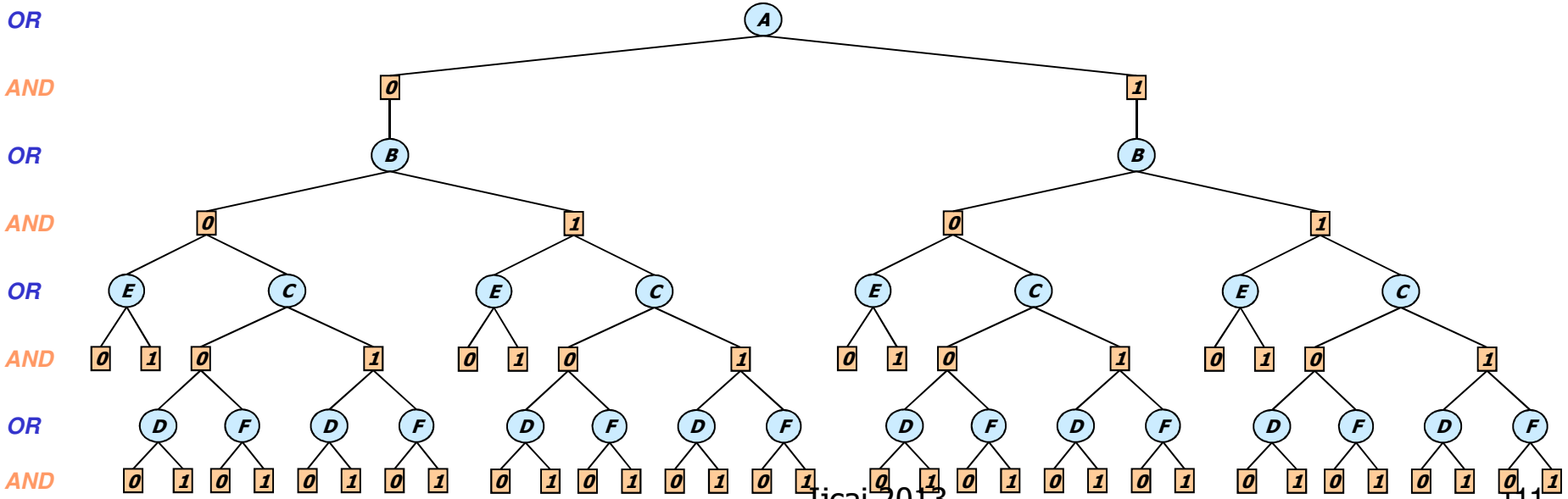
AND/OR Search Space



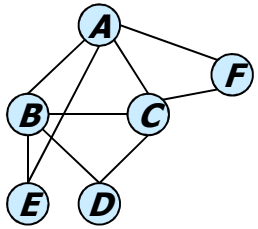
Primal graph

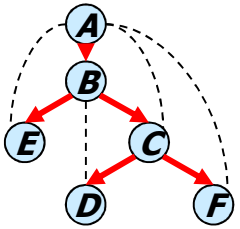


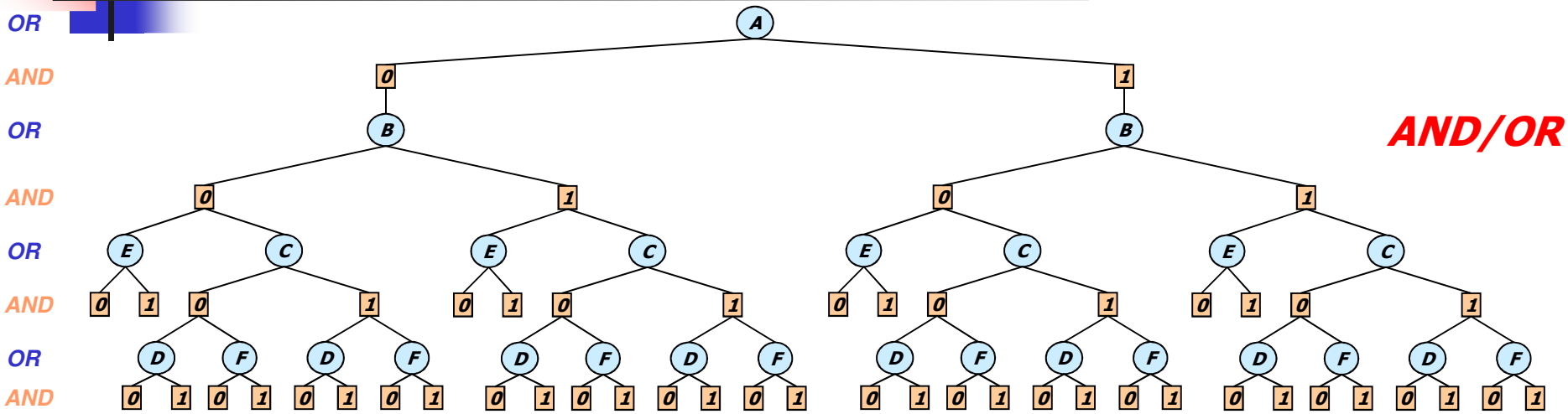
DFS tree



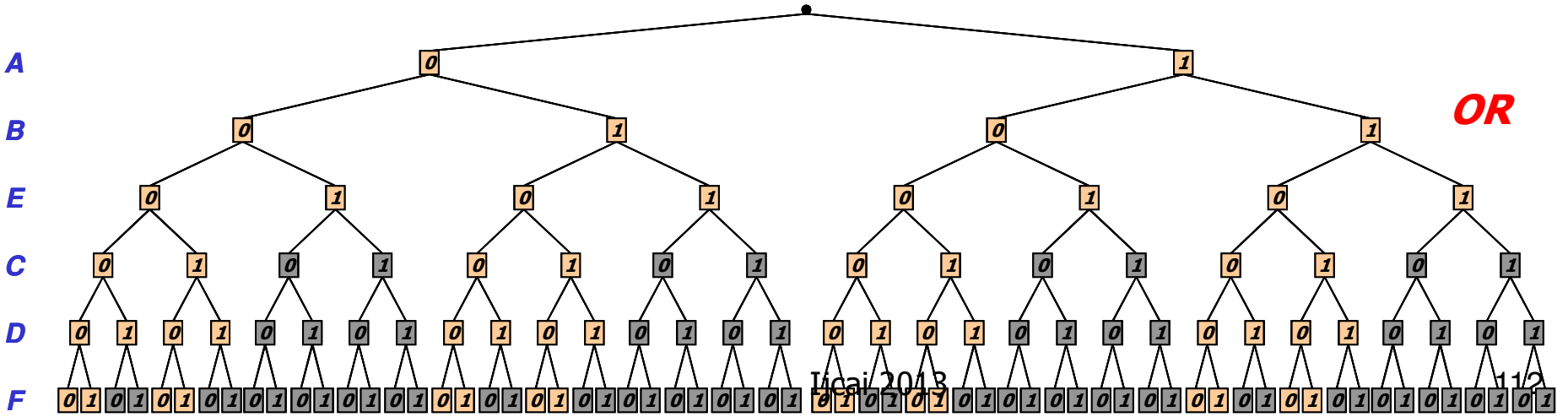
AND/OR vs. OR





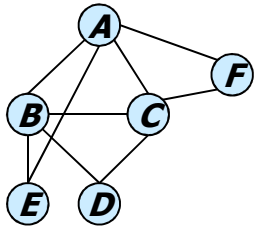


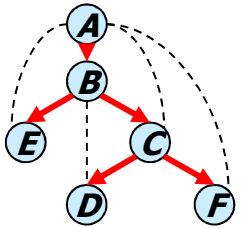
AND/OR size: exp(4), OR size exp(6)

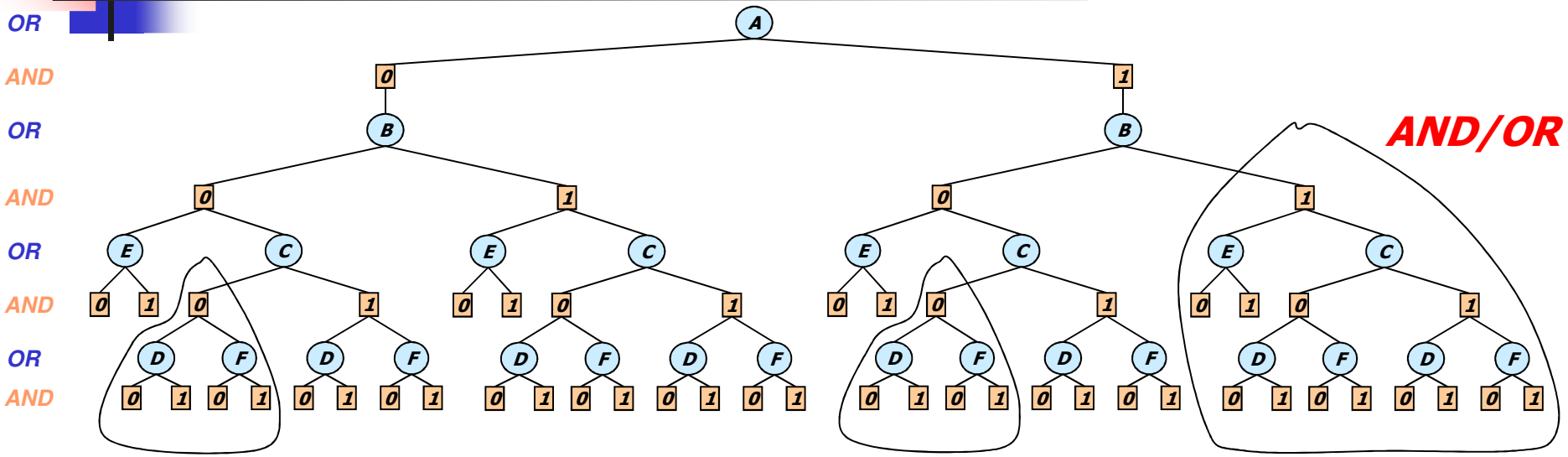


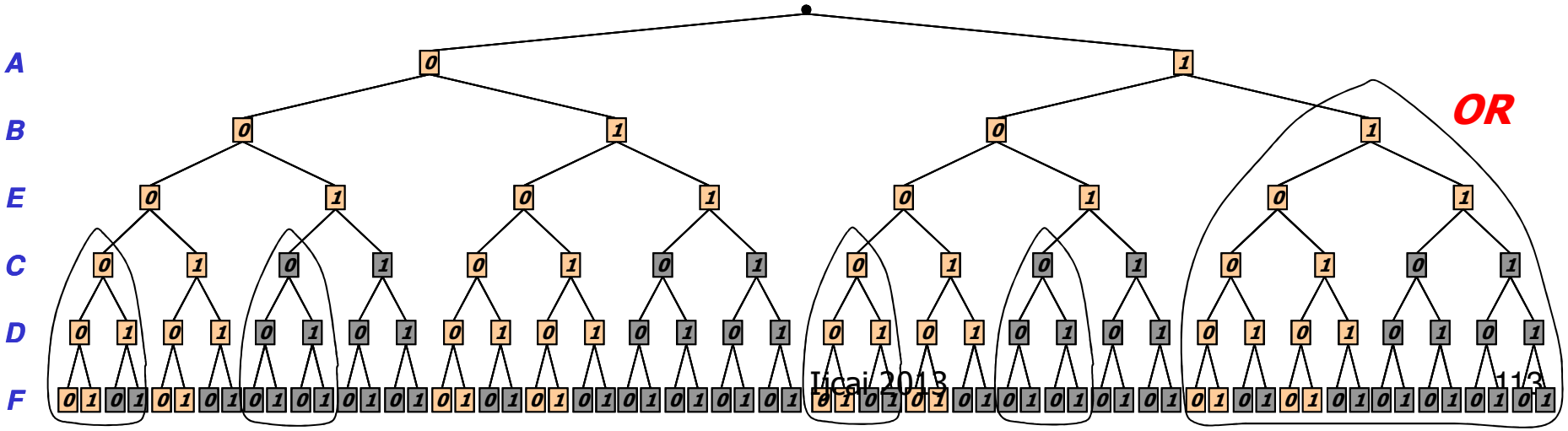
AND/OR vs. OR

No-goods
(A=1,B=1)
(B=0,C=0)



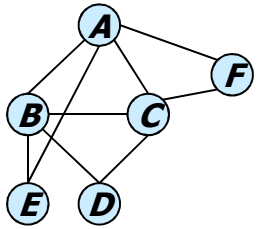


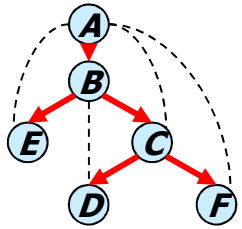


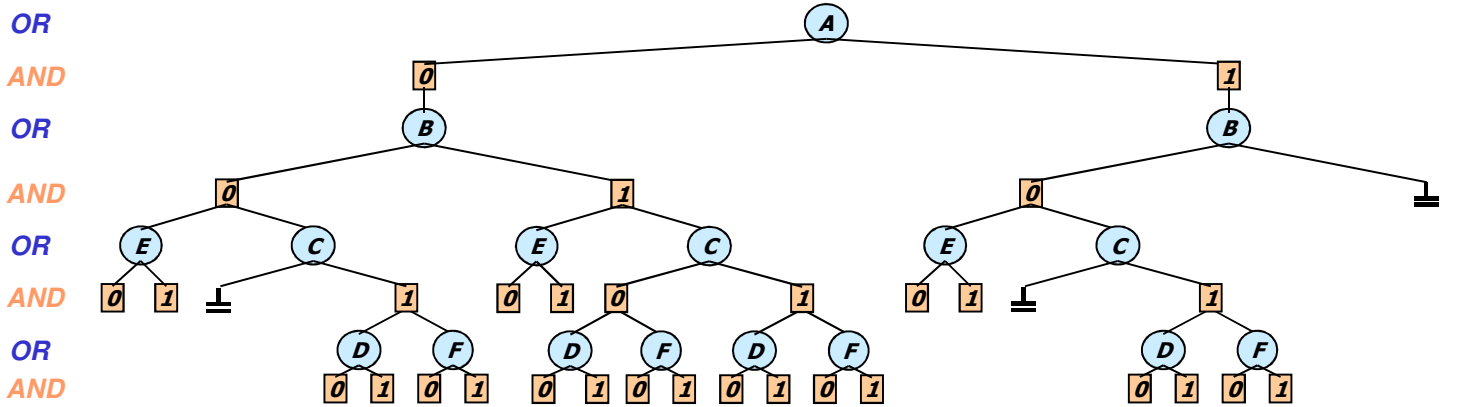


AND/OR vs. OR

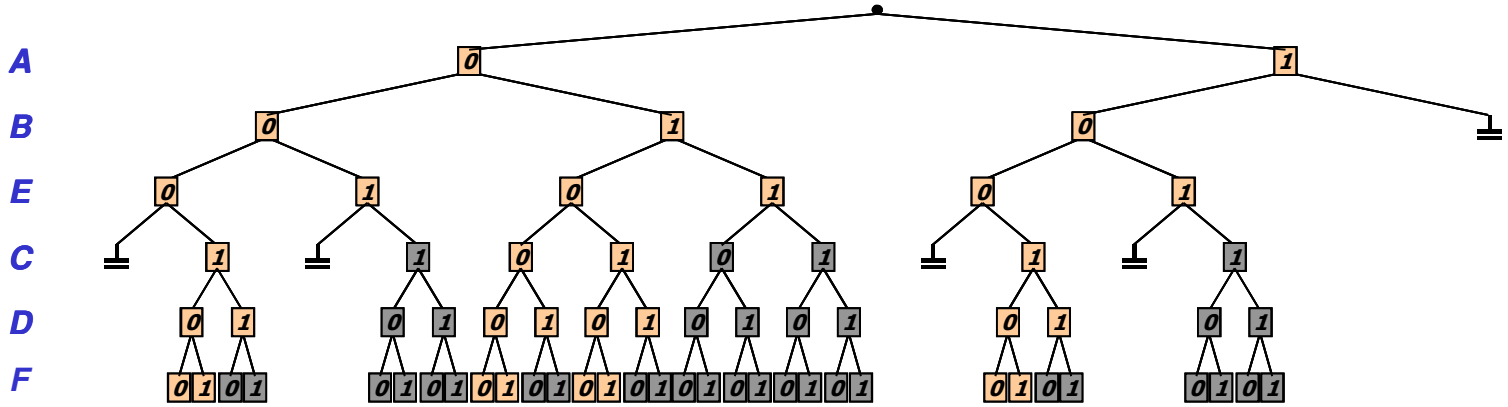
(A=1,B=1)
(B=0,C=0)







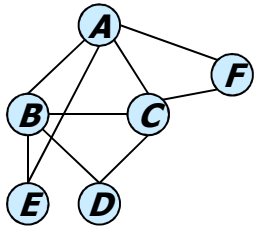
AND/OR

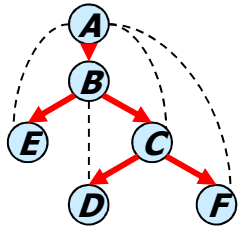


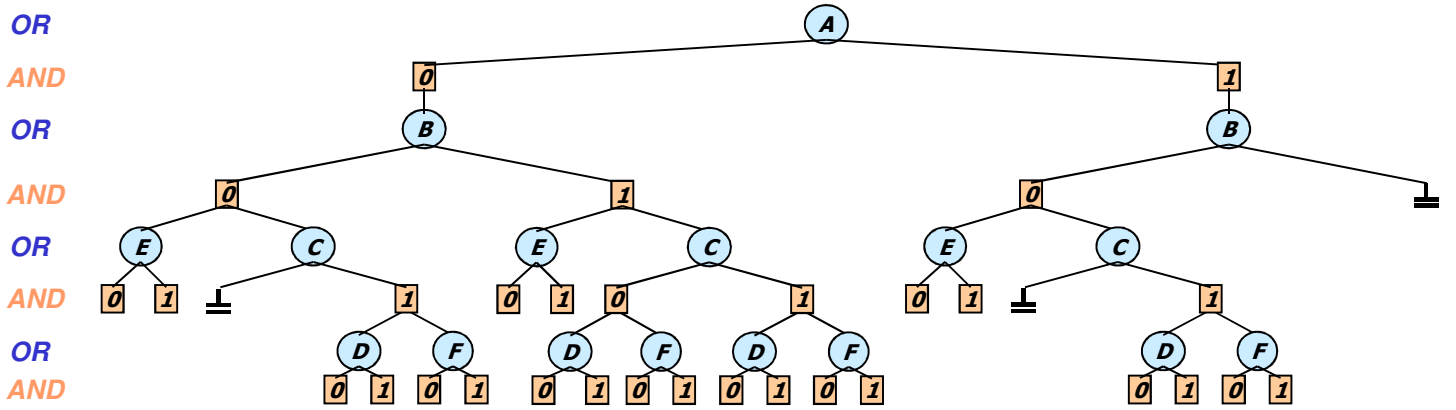
OR

AND/OR vs. OR

(A=1,B=1)
(B=0,C=0)

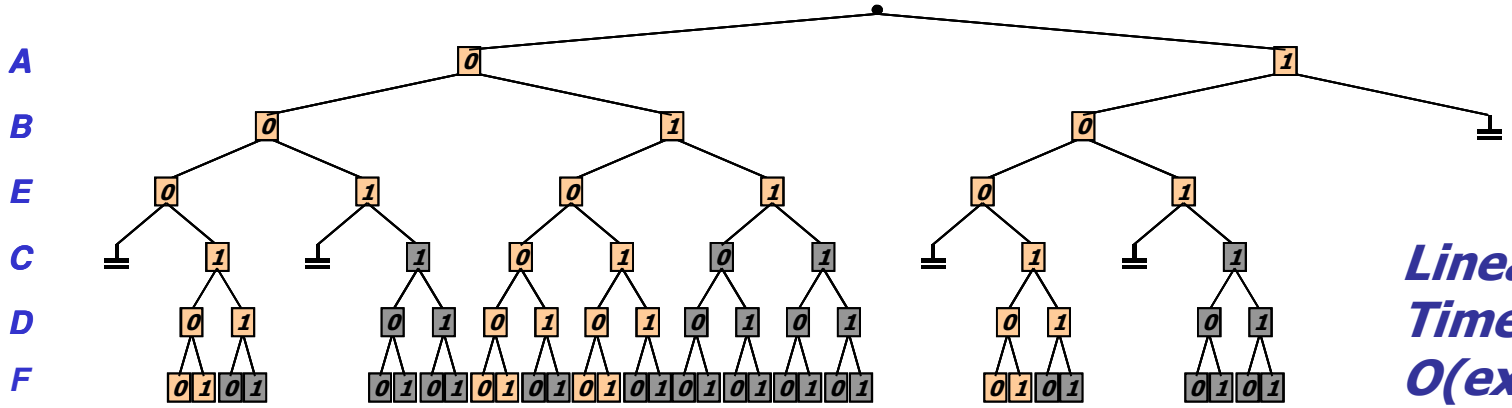






AND/OR

Space: linear
Time:
O(exp(m))
O(w* log n)



OR

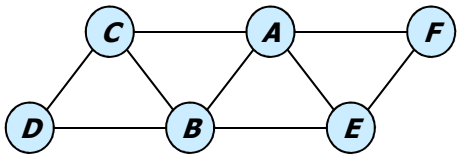
Linear space,
Time:
O(exp(n))

AND/OR vs. OR Spaces

width	depth	OR space		AND/OR space		
		Time (sec.)	Nodes	Time (sec.)	AND nodes	OR nodes
5	10	3.15	2,097,150	0.03	**10,494**	5,247
4	9	3.13	2,097,150	0.01	**5,102**	2,551
5	10	3.12	2,097,150	0.03	**8,926**	4,463
4	10	3.12	2,097,150	0.02	**7,806**	3,903
5	13	3.11	2,097,150	0.10	**36,510**	18,255

Random graphs with 20 nodes, 20 edges and 2 values per node

#CSP – AND/OR Search Tree

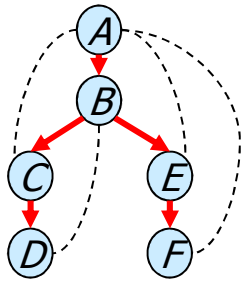


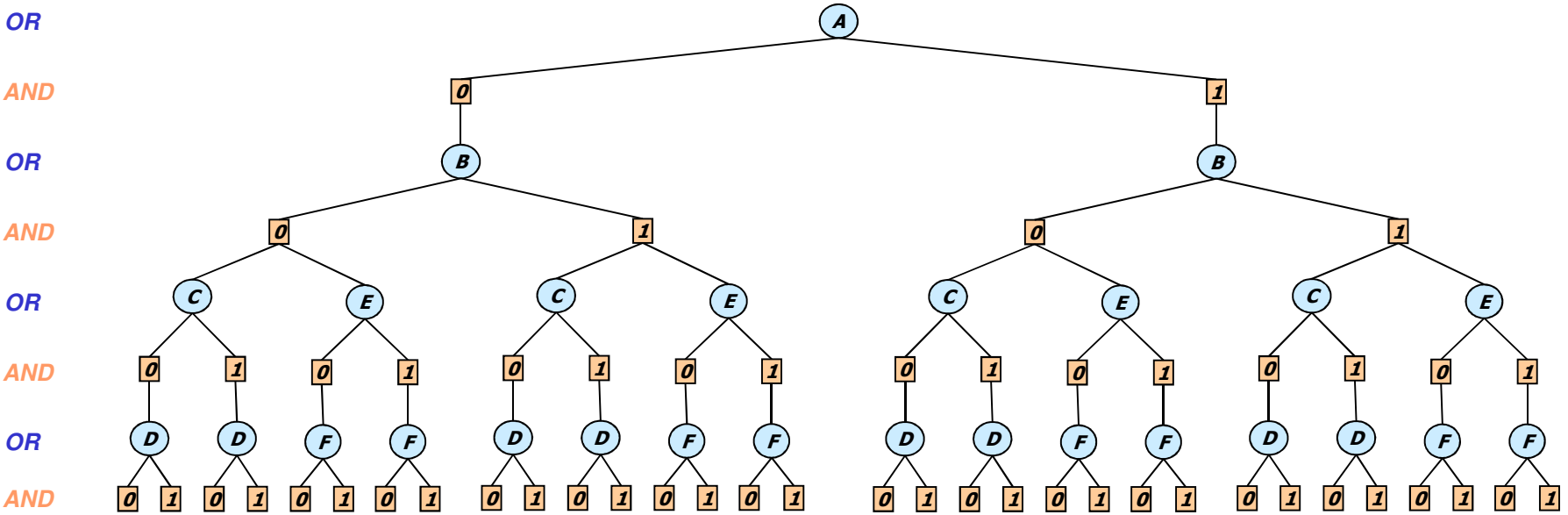
A	B	C	R_{ABC}
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

B	C	D	R_{BCD}
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

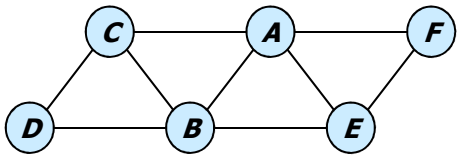
A	B	E	R_{ABE}
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

A	E	F	R_{AEF}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0





#CSP – AND/OR Tree DFS

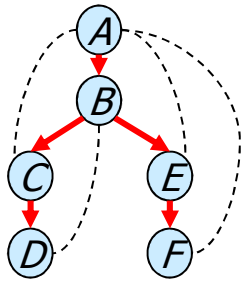


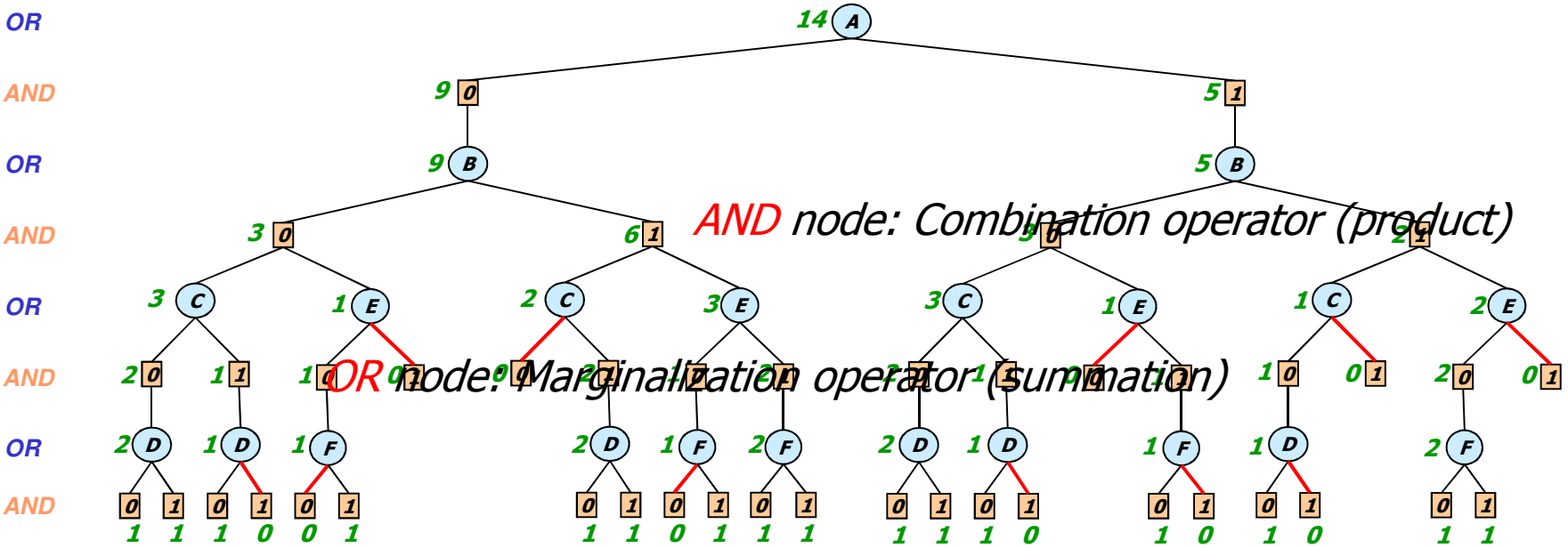
A	B	C	R_{ABC}
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

B	C	D	R_{BCD}
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

A	B	E	R_{ABE}
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

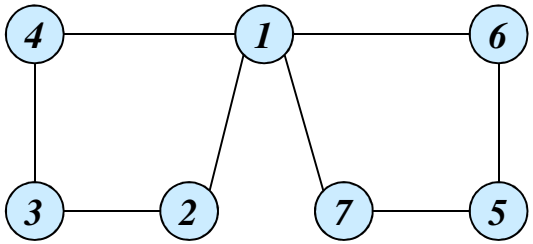
A	E	F	R_{AEF}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0





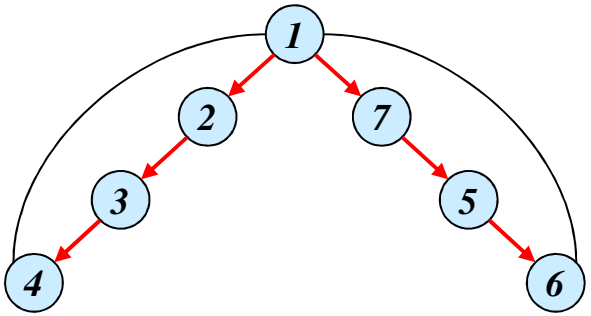
Pseudo-Trees

(Freuder 85, Bayardo 95, Bodlaender and Gilbert, 91)

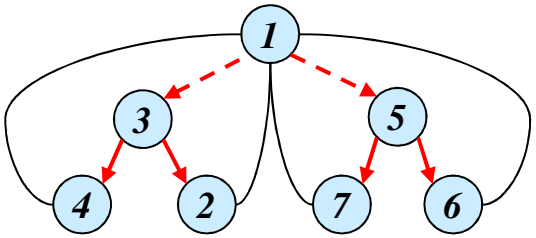


(a) Graph

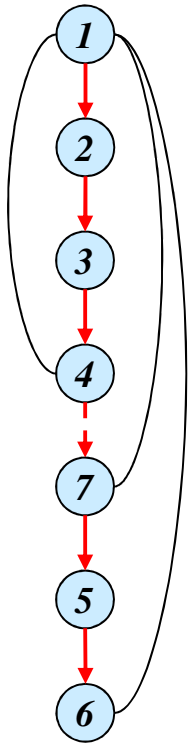
$$m <= w^* \log n$$



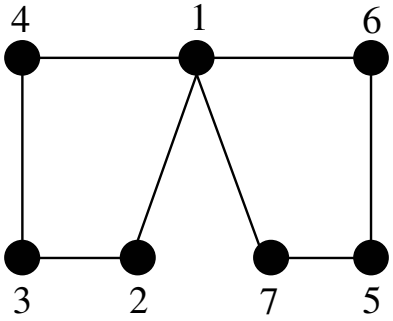
(b) DFS tree
depth=3



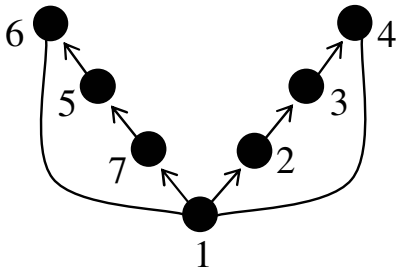
(c) pseudo- tree
depth=2



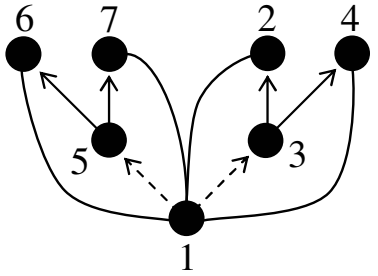
(d) Chain
depth=6



(a)



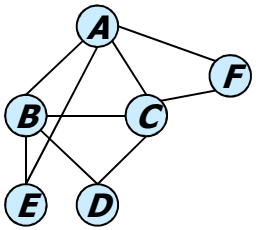
(b)



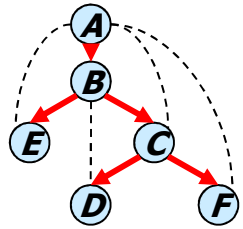
(c)

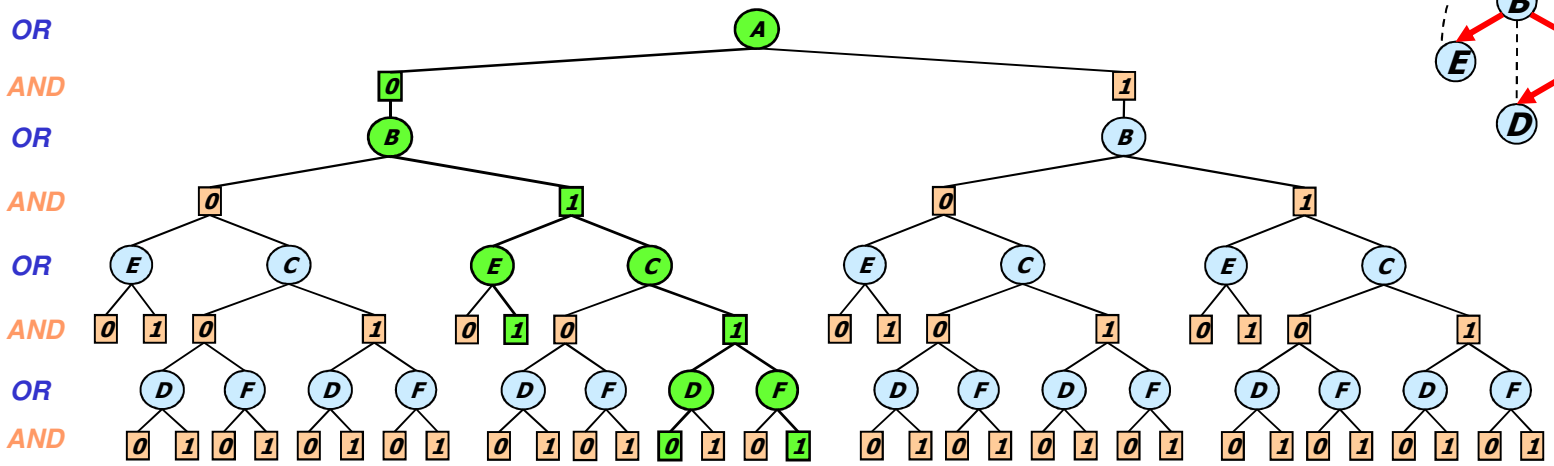
AND/OR Search Tree for Constraint Models

- **The AND/OR search tree of R relative to a tree, T, has:**
  - **Alternating levels of: OR nodes (variables) and AND nodes (values)**
- **Successor function:**
  - **The successors of OR nodes X are all its consistent values along its path**
  - **The successors of AND <X,v> are all X child variables in T**



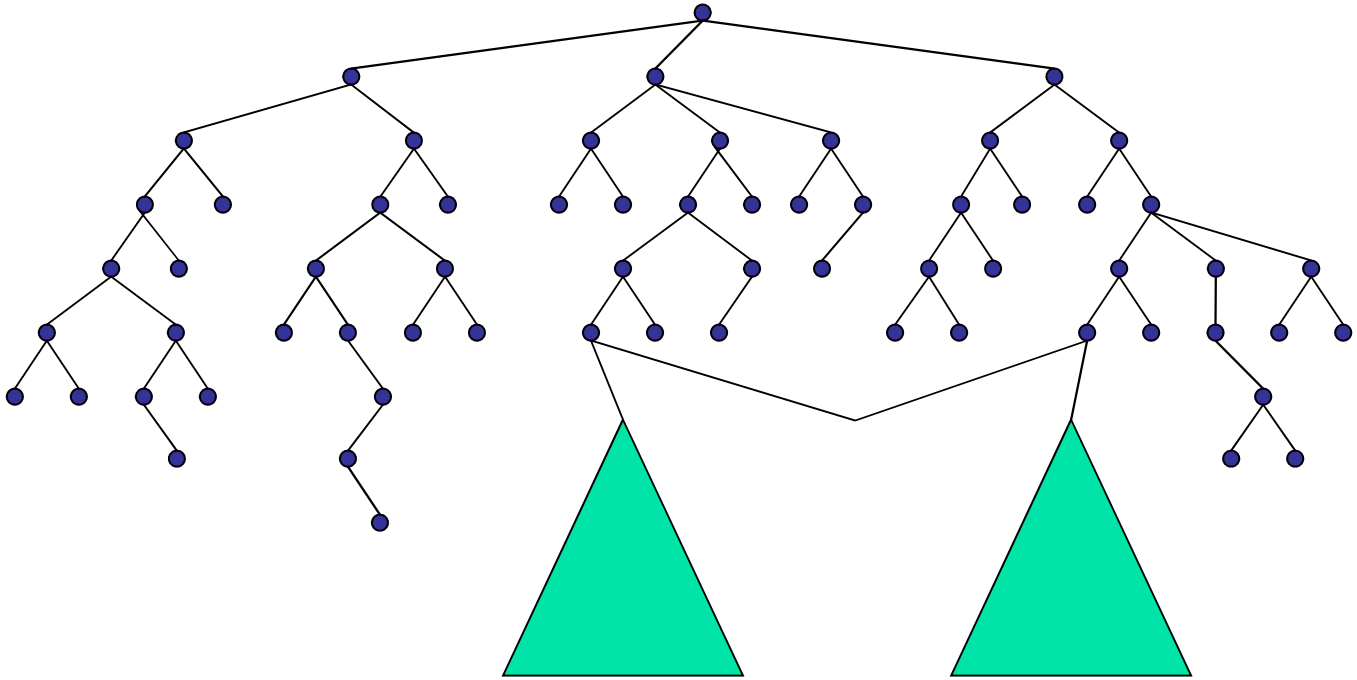
- **A solution is a consistent subtree**
- **Task: compute the value of the root node**



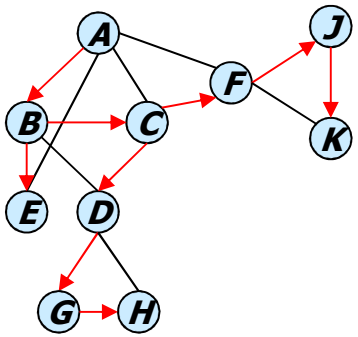


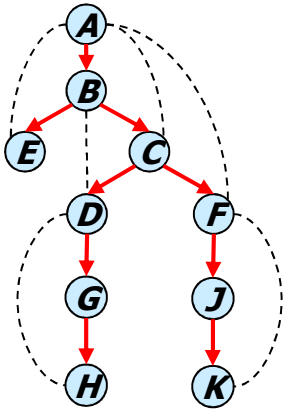
From Search Trees to Search Graphs

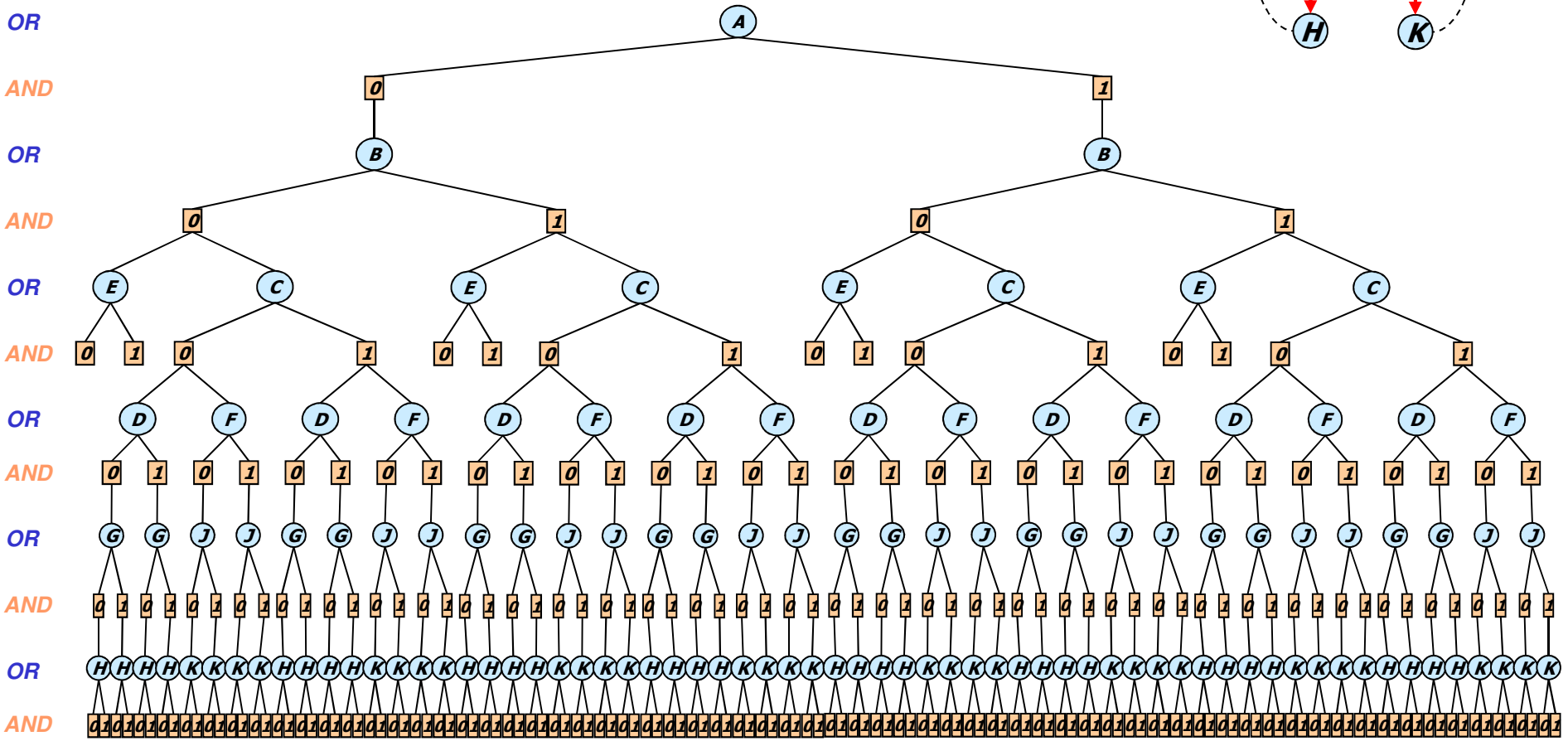
- Any two nodes that root identical subtrees (subgraphs) can be merged



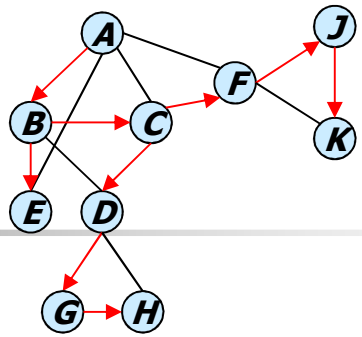
AND/OR Tree

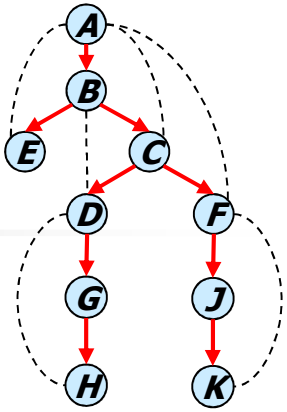

A
B
C
D
E
F
G
H
J
K

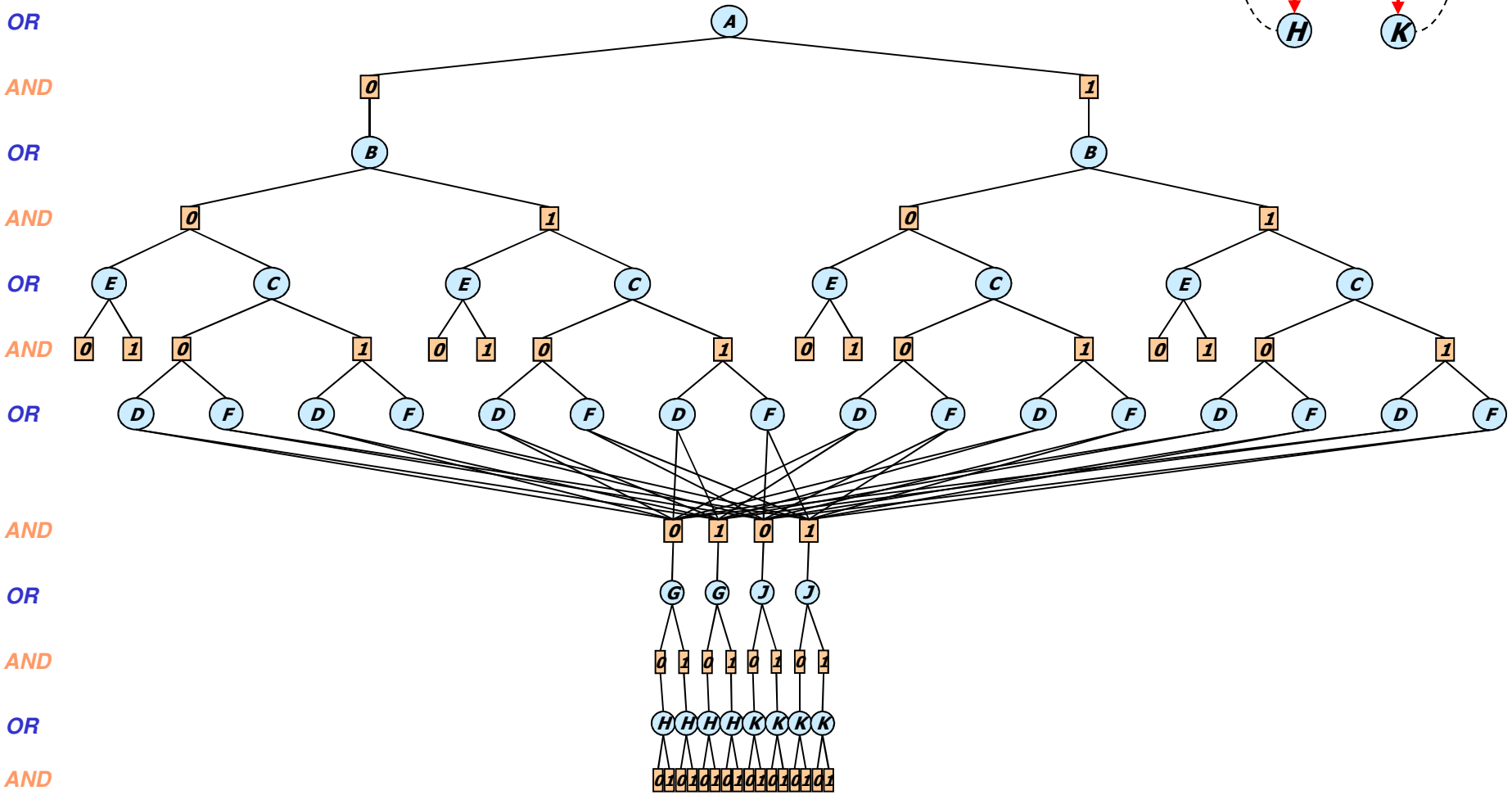

A
B
C
D
E
F
G
H
J
K


OR
AND
OR
AND
OR
AND
OR
AND
OR
AND
OR
AND

An AND/OR Graph: Caching Goods

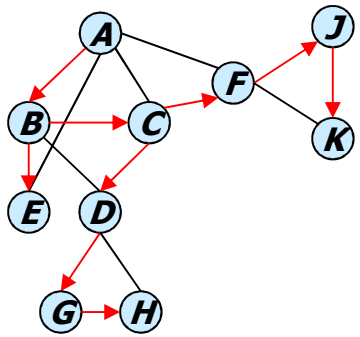


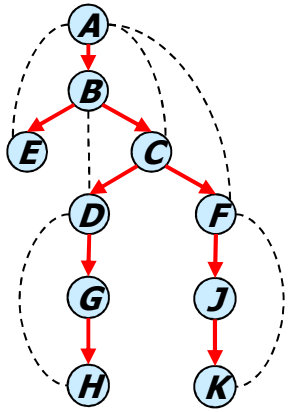




Context-based Caching

- Caching is possible when context is the same
- context = parent-separator set in induced pseudo-graph
  = current variable +
  parents connected to subtree below





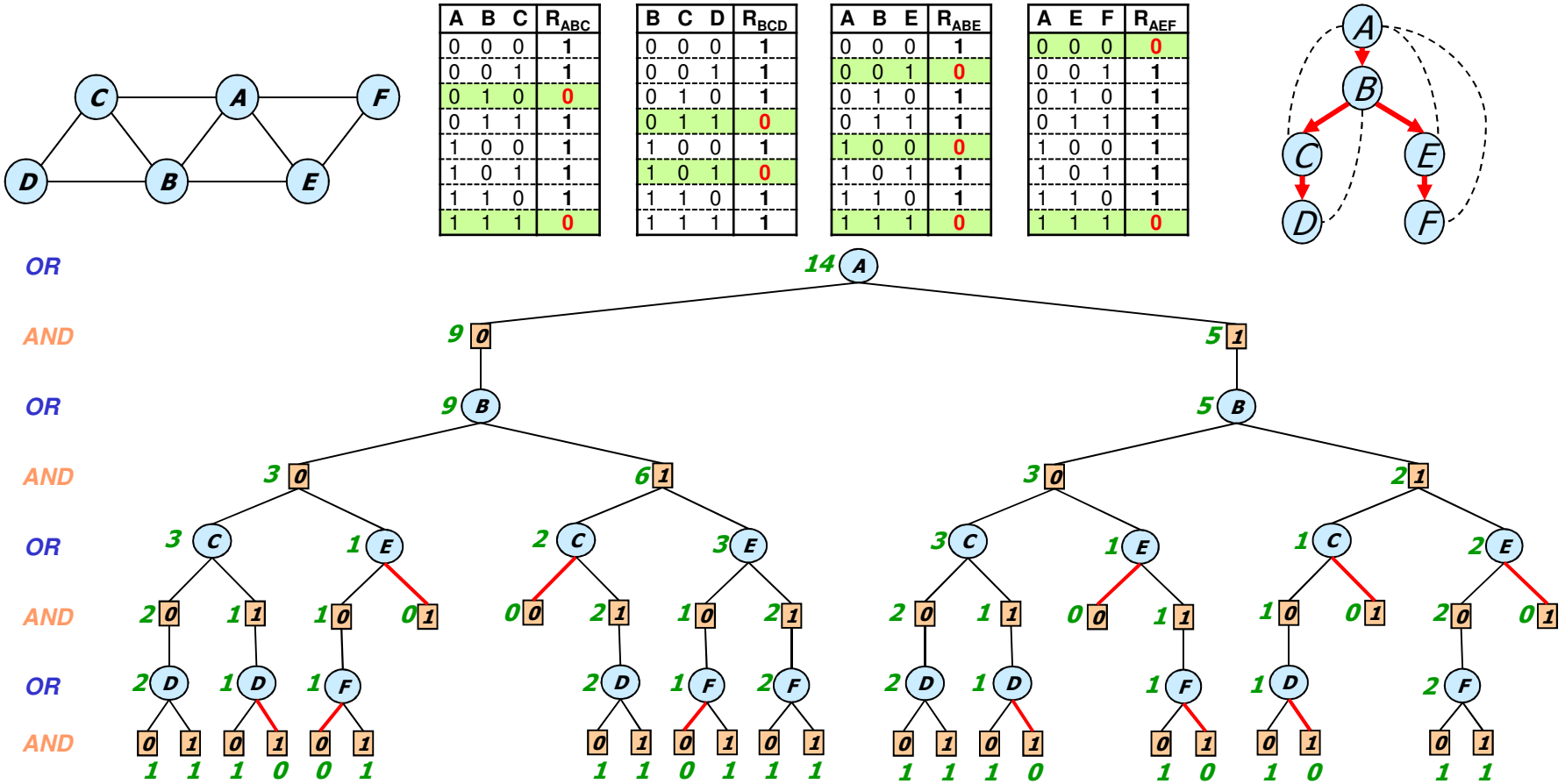
context(B) = {A, B}

context(c) = {A,B,C}

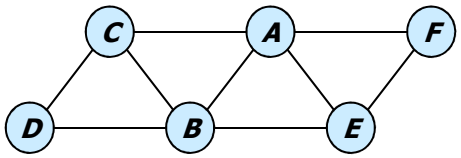
context(D) = {D}

context(F) = {F}

#CSP – AND/OR Tree DFS



#CSP – AND/OR Search Graph (Caching Goods)

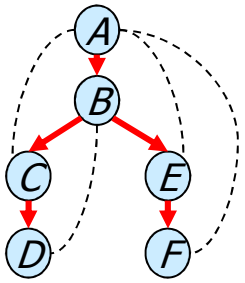


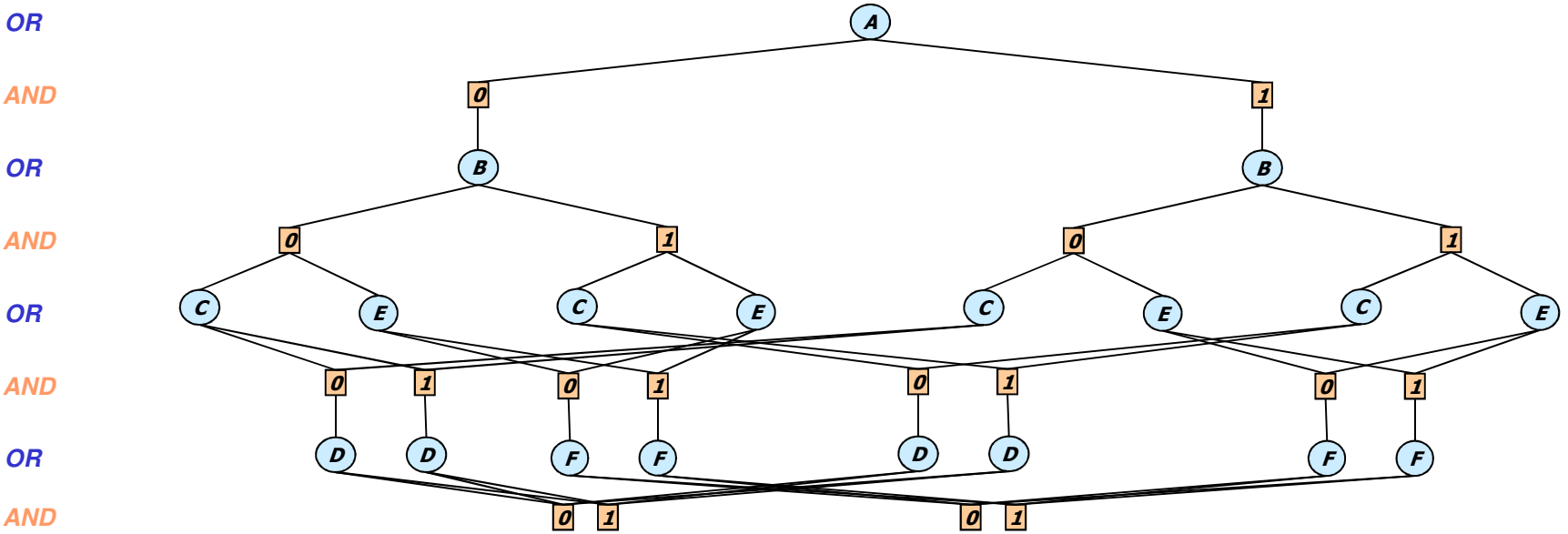
A	B	C	R_{ABC}
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

B	C	D	R_{BCD}
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

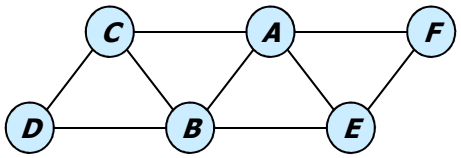
A	B	E	R_{ABE}
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

A	E	F	R_{AEF}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0





#CSP – AND/OR Search Graph (Caching Goods)

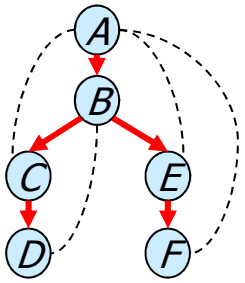


A	B	C	R_{ABC}
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

B	C	D	R_{BCD}
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

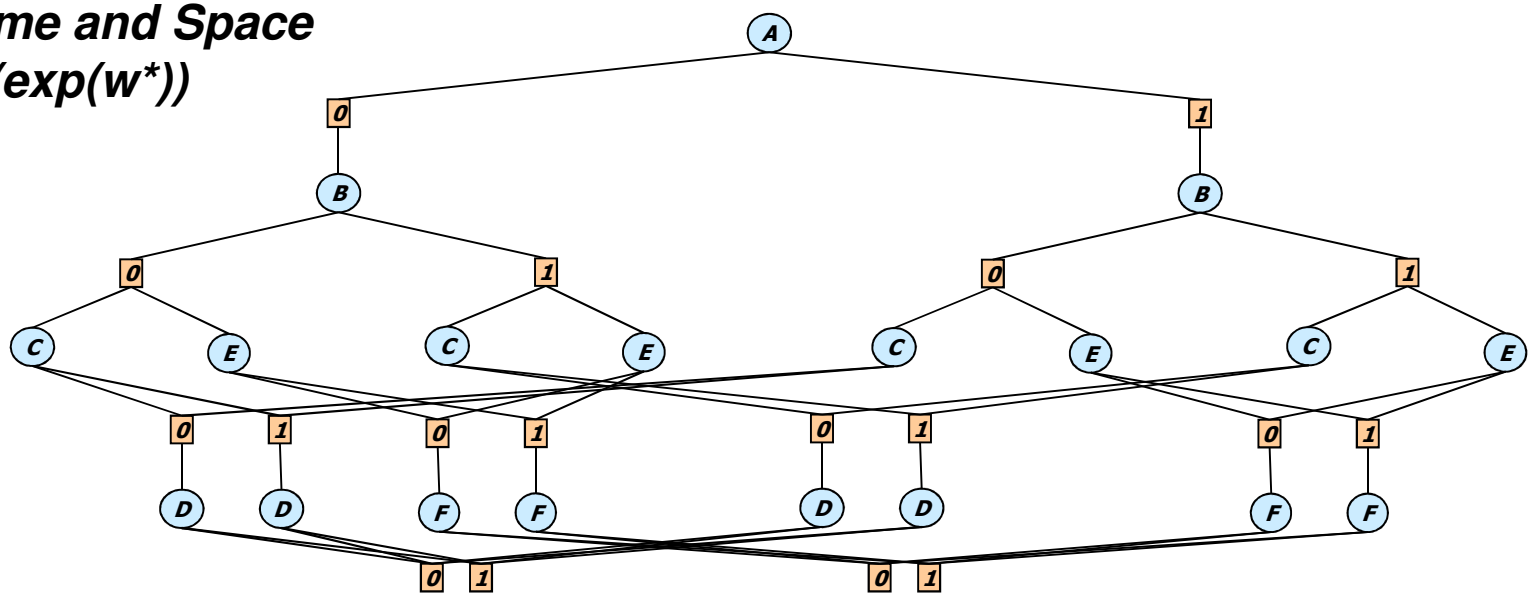
A	B	E	R_{ABE}
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

A	E	F	R_{AEF}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

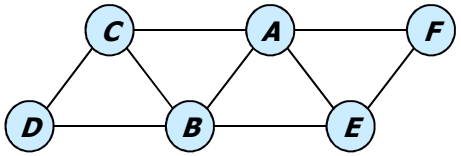


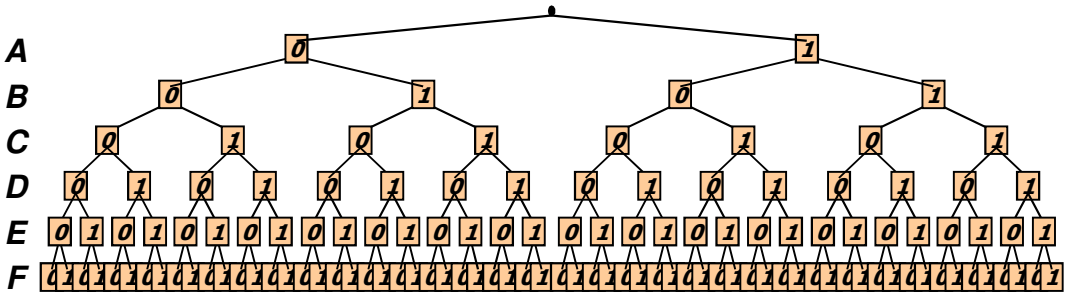
OR
AND
OR
AND
OR
AND
OR
AND

Time and Space
O(exp(w*))



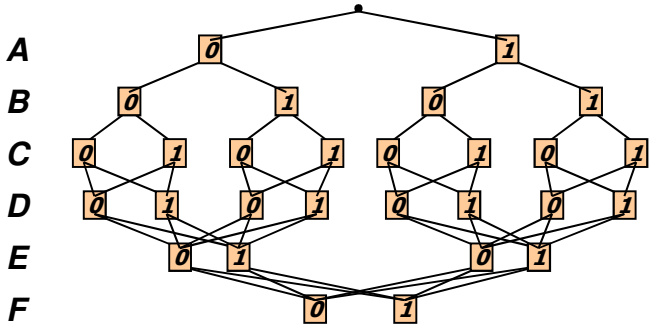
All Four Search Spaces





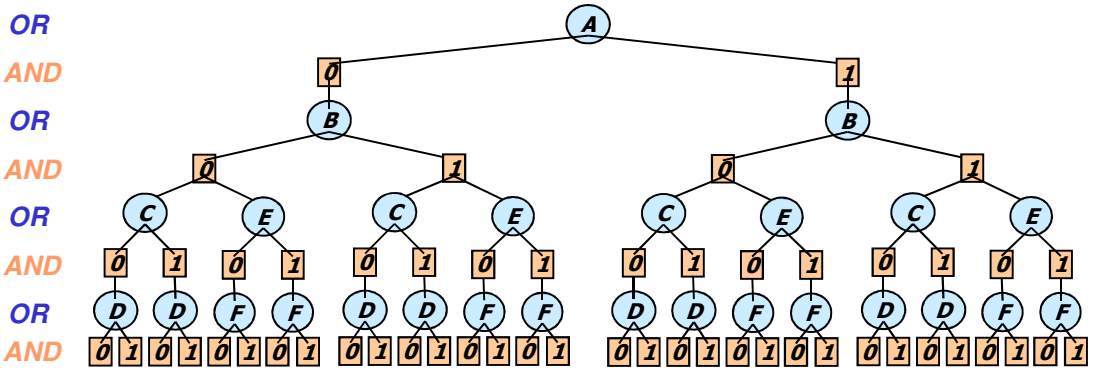
Full OR search tree

126 nodes



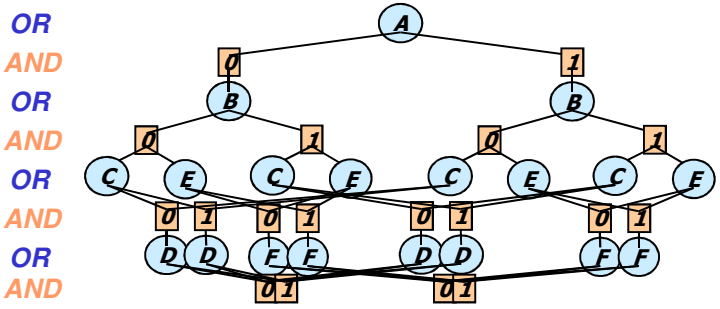
Context minimal OR search graph

28 nodes



Full AND/OR search tree

54 AND nodes



Context minimal AND/OR search graph

18 AND nodes

AND/OR vs. OR DFS Algorithms

k = domain size
m = tree depth
n = # of variables
w = induced width*
pw = path width*

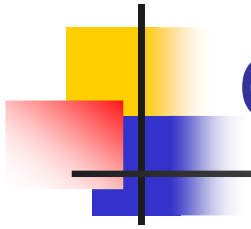
- **AND/OR tree**
  - **Space: O(n)**
  - **Time: $O(n\ k^m)$**
    $O(n\ k^{w^* \log n})$

  (Freuder85; Bayardo95; Darwiche01)

- **AND/OR graph**
  - **Space: $O(n\ k^{w^*})$**
  - **Time: $O(n\ k^{w^*})$**

- ***OR tree***
  - ***Space: $O(n)$***
  - ***Time: $O(k^n)$***

- ***OR graph***
  - ***Space: $O(n\ k^{pw^*})$***
  - ***Time: $O(n\ k^{pw^*})$***



Complexity of Searching AND/OR Graph

- **Theorem:** Traversing the AND/OR search graph is time and space exponential in the induced width/tree-width.

- If applied to the OR graph complexity is time and space exponential in the path-width.

Searching AND/OR Graphs

- AO(i): searches depth-first, cache i-context
  - i = the max size of a cache table (i.e. number of variables in a context)

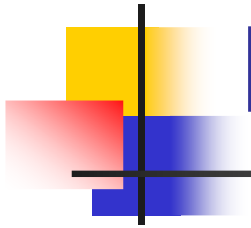
Space: ***O(n)***

Time: ***O(exp(w* log n))***

Space: ***O(exp w*)***

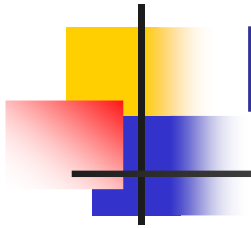
Time: ***O(exp w*)***

AO(i) time complexity?



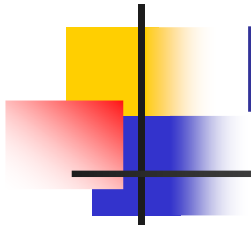
Road Map

- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination:
  - Tree-clustering
  - Constraint propagation
- Search
- Probabilistic Networks



Bayesian Networks

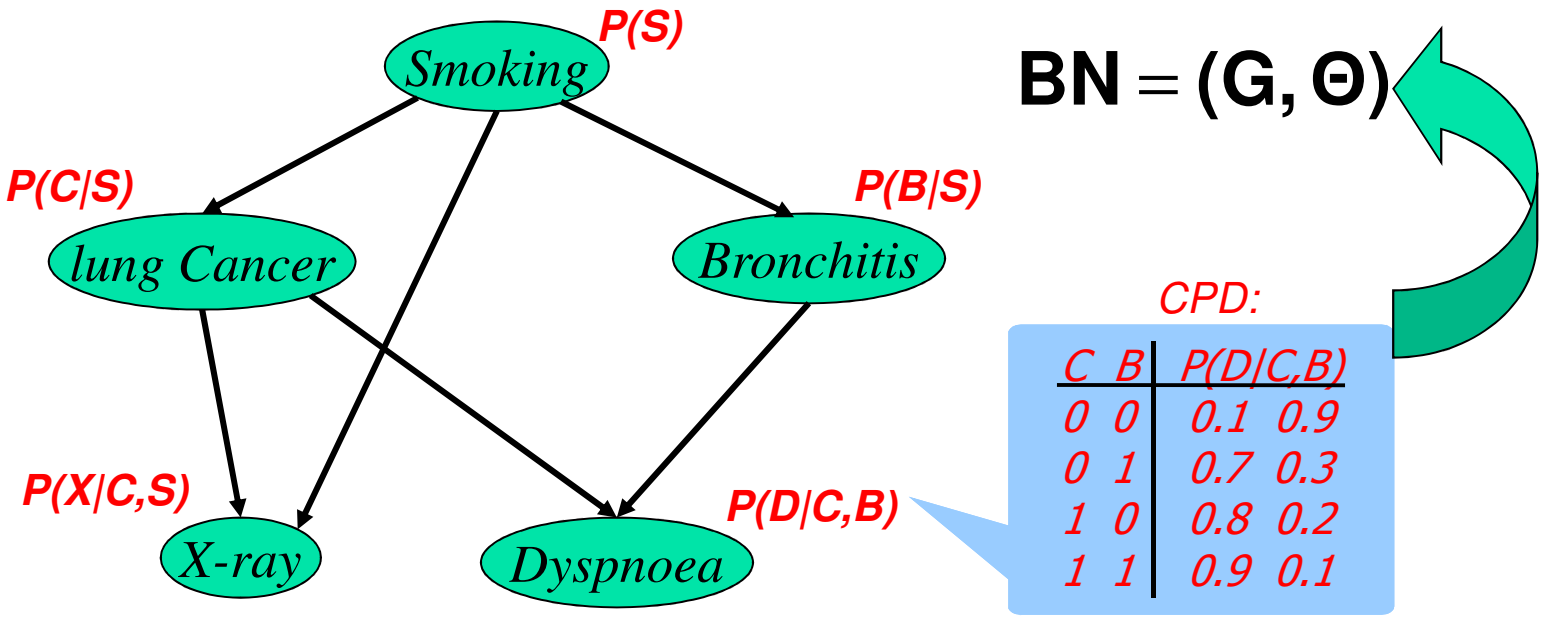
- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
  - Belief propagation
  - Mini-bucket, mini-clustering
  - Iterative join-graph propagation: a GBP scheme
- Search, conditioning



Bayesian Networks

- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
  - Belied propagation
  - Mini-bucket, mini-clustering
  - Iterative join-graph propagation: a GBP scheme
- Search, conditioning
- Sampling

Bayesian Networks: Representation
(Pearl, 1988)



$$P(S, C, B, X, D) = P(S)\,P(C|S)\,P(B|S)\,P(X|C,S)\,P(D|C,B)$$

Belief Updating:

P (lung cancer=yes | smoking=no, dyspnoea=yes) = ?

Sample Applications for Graphical Models

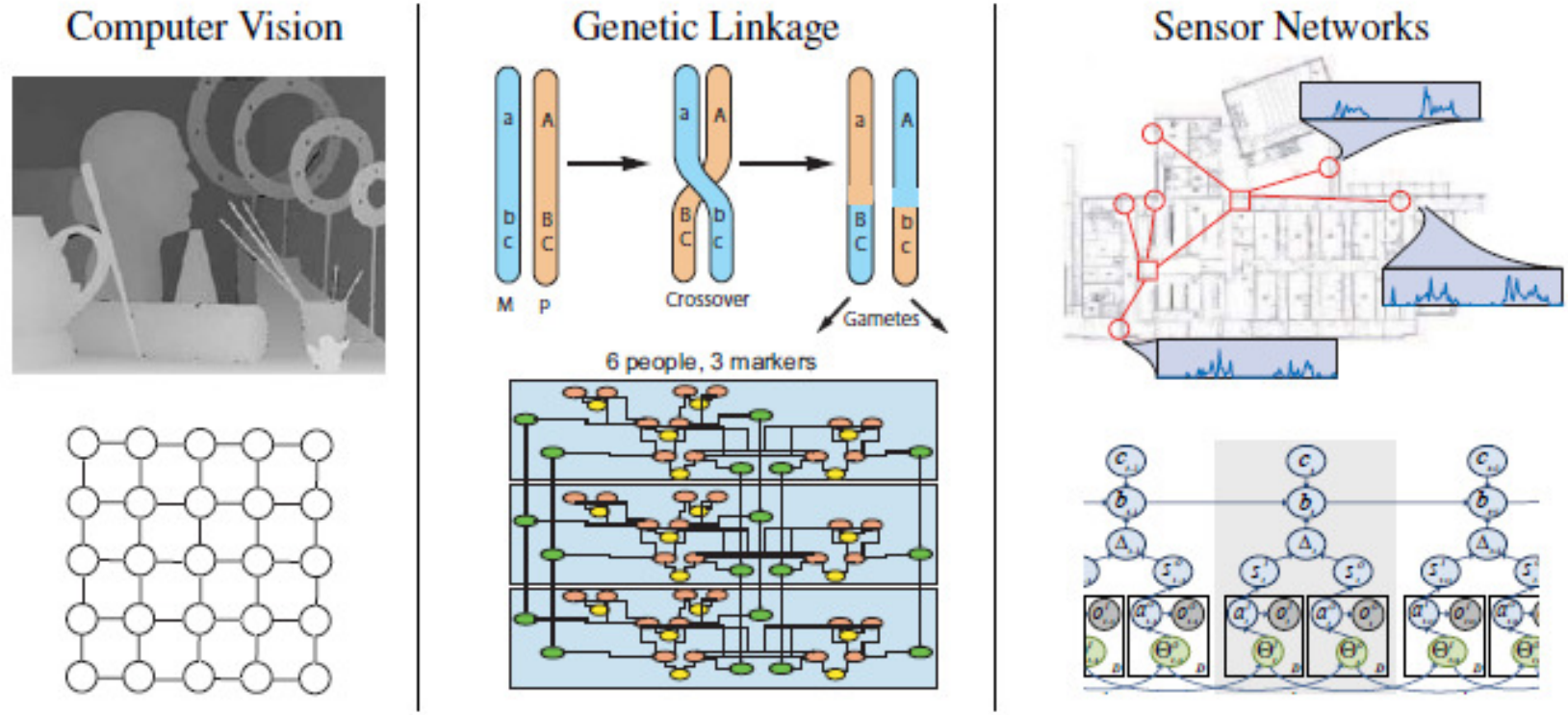
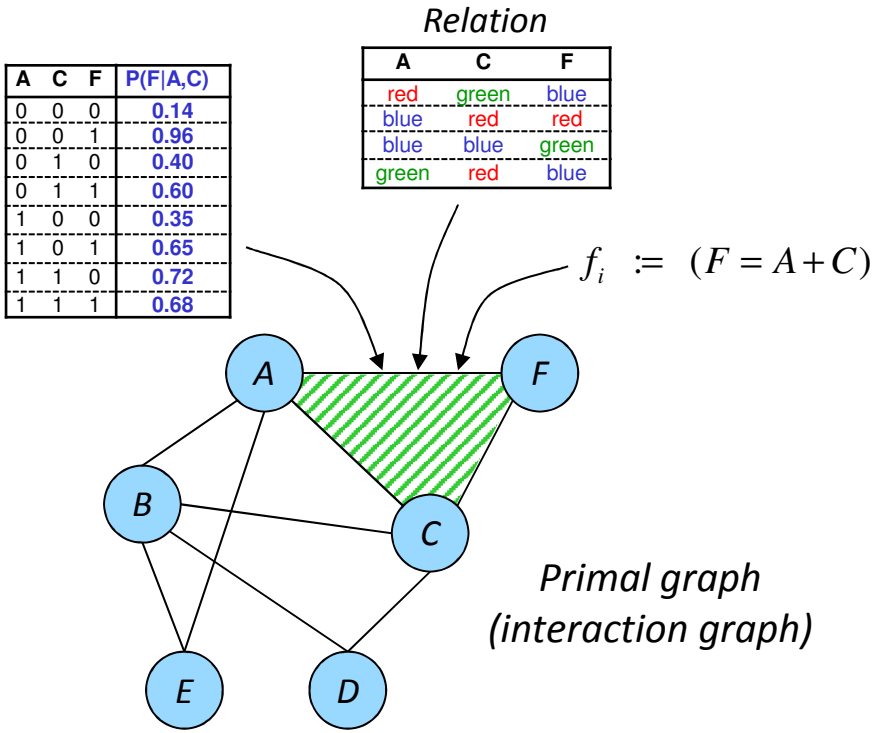


Figure 1: Application areas and graphical models used to represent their respective systems: (a) Finding correspondences between images, including depth estimation from stereo; (b) Genetic linkage analysis and pedigree data; (c) Understanding patterns of behavior in sensor measurements using spatio-temporal models.

Graphical Models

- A graphical model (**X**,**D**,**F**):
  - **X** = $\{X_1,\ldots X_n\}$ variables
  - **D** = $\{D_1, \ldots D_n\}$ domains
  - **F** = $\{f_1,\ldots,f_m\}$ functions
- Operators:
  - combination
  - elimination (projection)
- Tasks:
  - **Belief updating**: $\Sigma_{X-y} \prod_j P_i$
  - **MPE**: $\max_X \prod_j P_j$
  - **CSP**: $\prod_X \times_j C_j$
  - **Max-CSP**: $\min_X \Sigma_j f_j$



- *All these tasks are NP-hard*
  - *exploit problem structure*
  - *identify special cases*
  - *approximate*

Probabilistic Inference Tasks

- *Belief updating:*

$$\mathbf{BEL}(X_i) = \mathbf{P}(X_i = x_i \mid \mathbf{evidence})$$

- *Finding most probable explanation (MPE)*

$$\bar{x}^* = \arg\max_{\bar{x}} \mathbf{P}(\bar{x}, \mathbf{e})$$

- *Finding maximum a-posteriory hypothesis*

$$(a_1^*,\ldots,a_k^*) = \arg\max_{\bar{a}} \sum_{X/A} \mathbf{P}(\bar{x}, \mathbf{e})$$

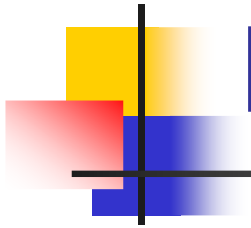
$A \subseteq X$: **hypothesis variables**

- *Finding maximum-expected-utility (MEU) decision*

$$(d_1^*,\ldots,d_k^*) = \arg\max_{\bar{d}} \sum_{X/D} \mathbf{P}(\bar{x}, \mathbf{e})\mathbf{U}(\bar{x})$$

$D \subseteq X$: **decision variables**

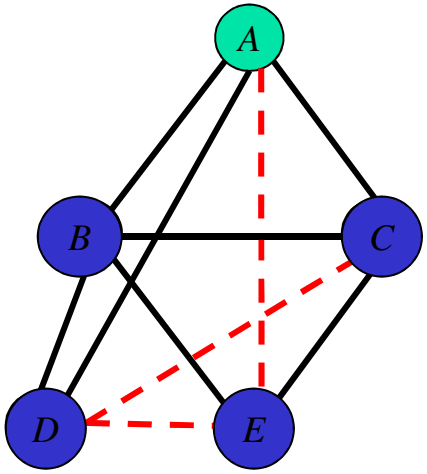
$U(\bar{x})$: **utility function**



Road Map

- Bayesian networks definition
- Inference:
  - Variable-elimination
  - tree-clustering
- Bounded-inference
  - Belied propagation
  - Mini-bucket, mini-clustering
  - Iterative join-graph propagation: a GBP scheme
- Search, conditioning
- Sampling

Query 1:
Belief updating: P(X|evidence)=?



"Moral" graph

$$P(a|e=0) \propto P(a,e=0)$$

$$= \sum_{e=0,d,c,b} P(a)P(b|a)P(c|a)P(d|b,a)P(e|b,c)$$

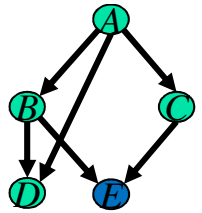
$$= P(a)\sum_{e=0}\sum_{d}\sum_{c} P(c|a)\sum_{b} P(b|a)P(d|b,a)P(e|b,c)$$

Variable Elimination

$h^B(a,d,c,e)$

Bucket elimination

Algorithm *BE-bel* (Dechter 1996)



$$P(A \mid E=0) = \alpha \sum_{E=0,D,C,B} P(A)\cdot P(B \mid A)\cdot P(C \mid A)\cdot P(D \mid A,B)\cdot P(E \mid B,C)$$

$\sum\prod_b$ ← *Elimination operator*

bucket B: $P(b|a)$ $P(d|b,a)$ $P(e|b,c)$

bucket C: $P(c|a)$ $\lambda^B(a, d, c, e)$

bucket D: $\lambda^C(a, d, e)$

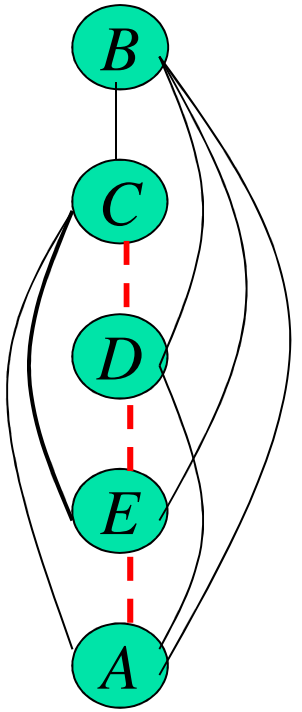
bucket E: $e=0$ $\lambda^D(a, e)$

bucket A: $P(a)$ $\lambda^E(a)$

$P(e=0)$

$P(a|e=0)$

W=4*
"induced width"
(max clique size)



$$P(a|e=0) = \frac{P(a,e=0)}{P(e=0)}$$

Combination of Cost Functions

A	B	f(A,B)
b	b	0.4
b	g	0.1
g	b	0
g	g	0.5

B	C	f(B,C)
b	b	0.2
b	g	0
g	b	0
g	g	0.8

A	B	C	f(A,B,C)
b	b	b	0.1
b	b	g	0
b	g	b	0
b	g	g	0.08
g	b	b	0
g	b	g	0
g	g	b	0
g	g	g	0.4

= 0.1 x 0.8

Factors: Sum-Out Operation

The result of summing out variable X from factor $f(\mathbf{X})$

is another factor over variables $\mathbf{Y} = \mathbf{X} \setminus \{X\}$:

$$\left(\sum_X f\right)(\mathbf{y}) \stackrel{def}{=} \sum_x f(x, \mathbf{y})$$

B	C	D	f_1
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

B	C	$\sum_D f_1$
true	true	1
true	false	1
false	true	1
false	false	1

	$\sum_B \sum_C \sum_D f_1$
$\top$	4

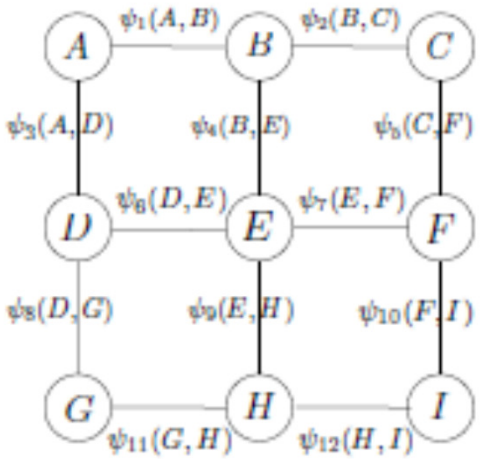
A sum-product algorithm

BE-BEL

Input: A belief network $\{P_1, ..., P_n\}$, d,e.
Output: belief of X_1 given e.

1. **Initialize:**
2. **Process buckets** from $p = n$ to 1
   for matrices $\lambda_1, \lambda_2, ..., \lambda_j$ in $bucket_p$ do
   - **If** (observed variable) $X_p = x_p$ assign $X_p = x_p$ to each λ_i.
   - **Else**, (multiply and sum) $\lambda_p = \sum_{X_p} \Pi_{i=1}^{j} \lambda_i$. Add λ_p to its bucket.
3. **Return** $Bel(x_1) = \alpha P(x_1) \cdot \Pi_i \lambda_i(x_1)$

BE for Markov networks queries



(a)

D	E	$\psi_6(D,E)$
0	0	20.2
0	1	12
1	0	23.4
1	1	11.7

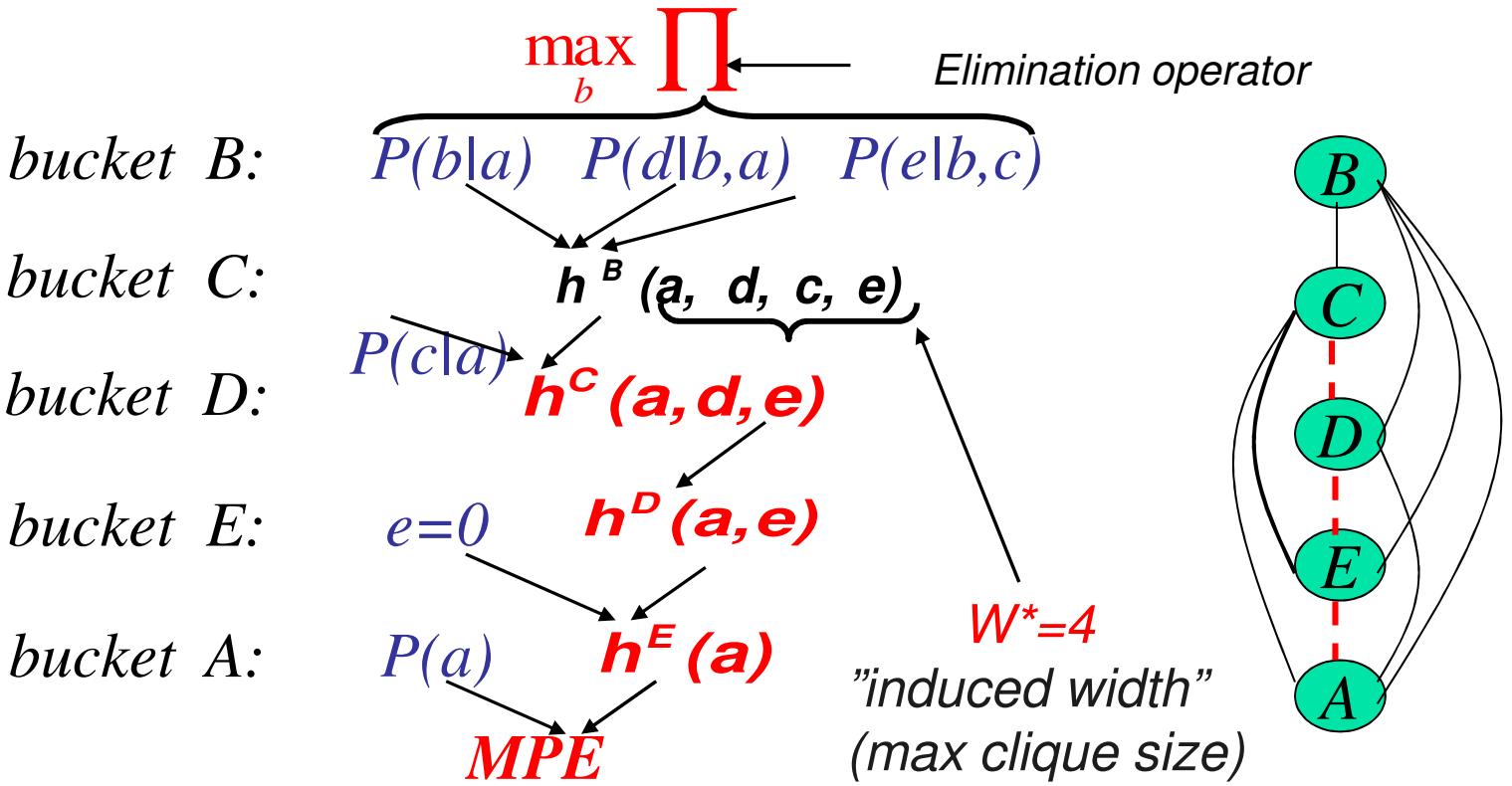
(b)

Query 2: Finding $\mathrm{MPE} = \max_{\bar{x}} P(\bar{x})$

Algorithm *BE-mpe* (Dechter 1996)

$\sum$ is replaced by ***max*** :

$$MPE = \max_{a,e,d,c,b} P(a)P(c\,|\,a)P(b\,|\,a)P(d\,|\,a,b)P(e\,|\,b,c)$$



Generating the MPE-tuple

5. $b' = \arg\max_b P(b \mid a') \times P(d' \mid b, a') \times P(e' \mid b, c')$

4. $c' = \arg\max_c P(c \mid a') \times h^B(a', d', c, e')$

3. $d' = \arg\max_d h^C(a', d, e')$

2. $e' = 0$

1. $a' = \arg\max_a P(a) \cdot h^E(a)$

B: $P(b|a)$ $P(d|b,a)$ $P(e|b,c)$

C: $P(c|a)$ $h^B(a,d,c,e)$

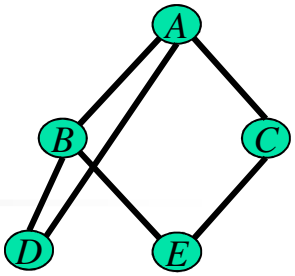
D: $h^C(a,d,e)$

E: $e=0$ $h^D(a,e)$

A: $P(a)$ $h^E(a)$

Return (a', b', c', d', e')

Query 3: Finding MAP



Algorithm *BE-map*

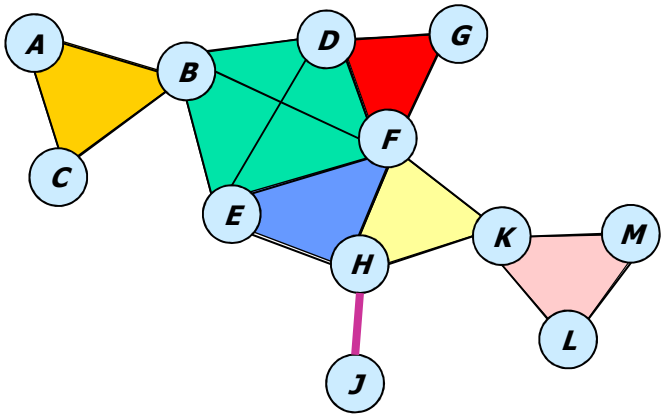
$$\sum \textbf{ and } \max:$$

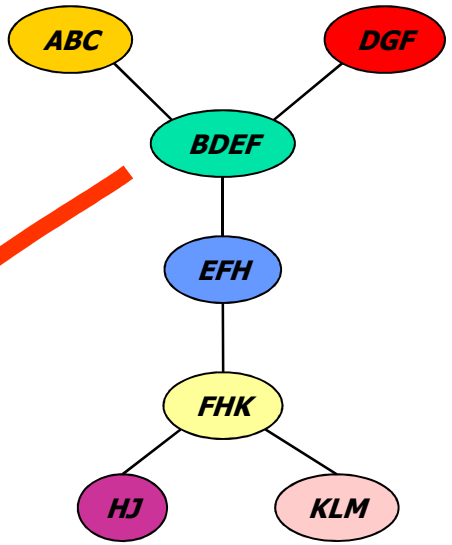
$$MPE = \max_{a,c} P(a)P(c \mid a) \sum_{e,d,b} P(b \mid a)P(d \mid a,b)P(e \mid b,c)$$

BE-MAP sum over regular buckets and max over hypothesis variables

Orderings are restricted so that max-variables come before sum variables
A mixed sum-product and max-product algorithm

Inference and Treewidth





Inference algorithm:

Time: ***exp(tree-width)***

Space: ***exp(tree-width)***

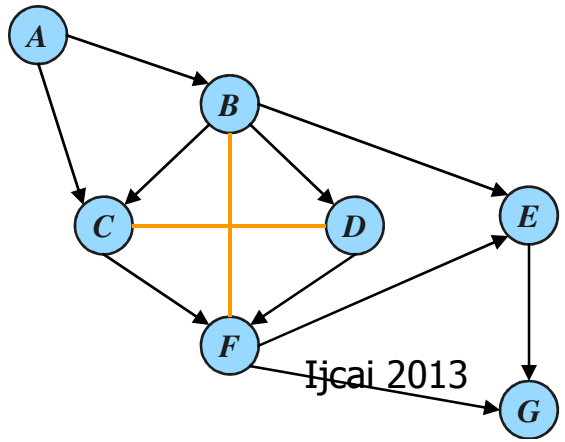
treewidth = 4 - 1 = 3

treewidth = (maximum cluster size) - 1

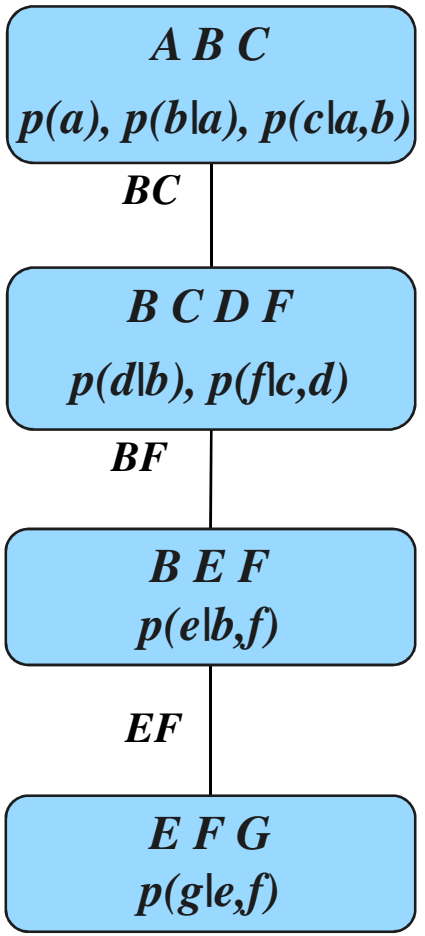
Tree decompositions

A *tree decomposition* for a belief network $BN =< X,D,G,P >$ is a triple $< T, \chi, \psi >$, where $T = (V,E)$ is a tree and χ and ψ are labeling functions, associating with each vertex $v \in V$ two sets, $\chi(v) \subseteq X$ and $\psi(v) \subseteq P$ satisfying :

1. For each function $p_i \in P$ there is exactly one vertex such that $p_i \in \psi(v)$ and $scope(p_i) \subseteq \chi(v)$
2. For each variable $X_i \in X$ the set $\{v \in V | X_i \in \chi(v)\}$ forms a connected subtree (running intersection property)

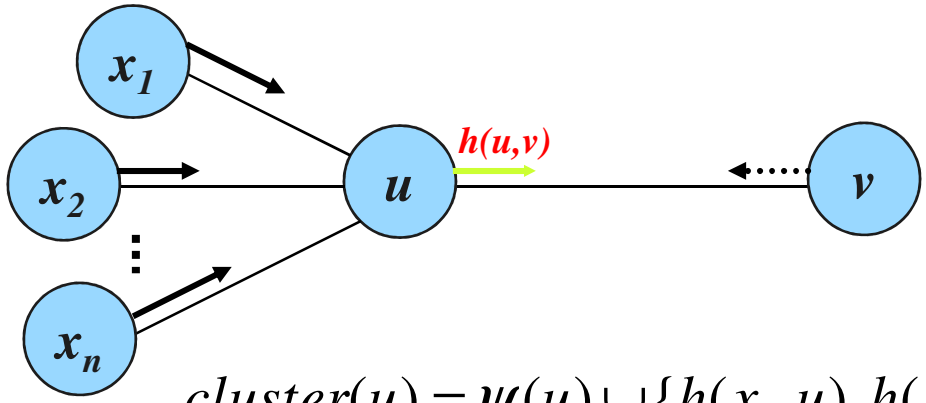


Belief network



Tree decomposition

Belief Propagation

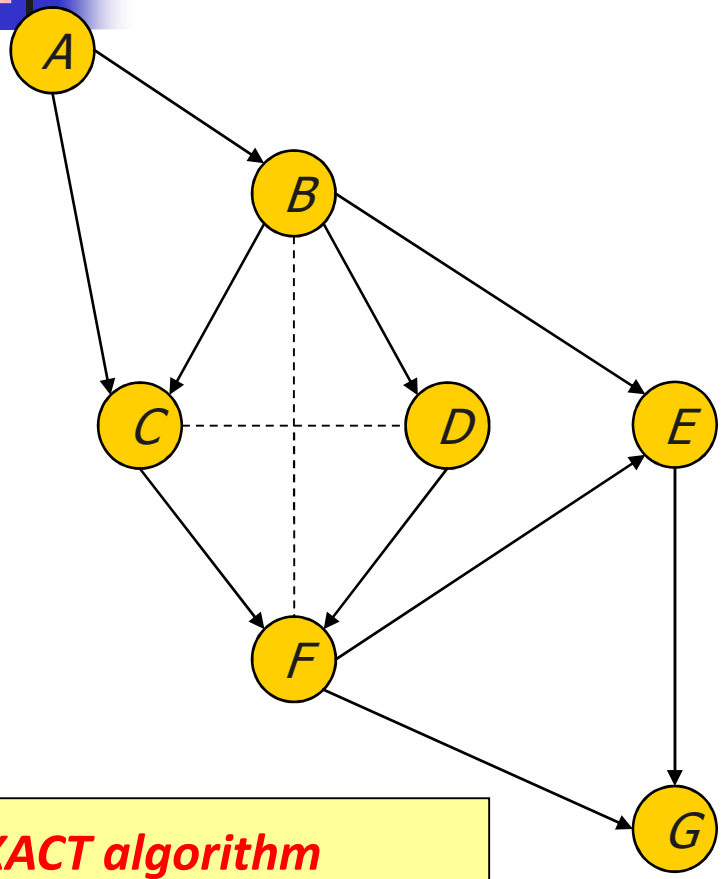


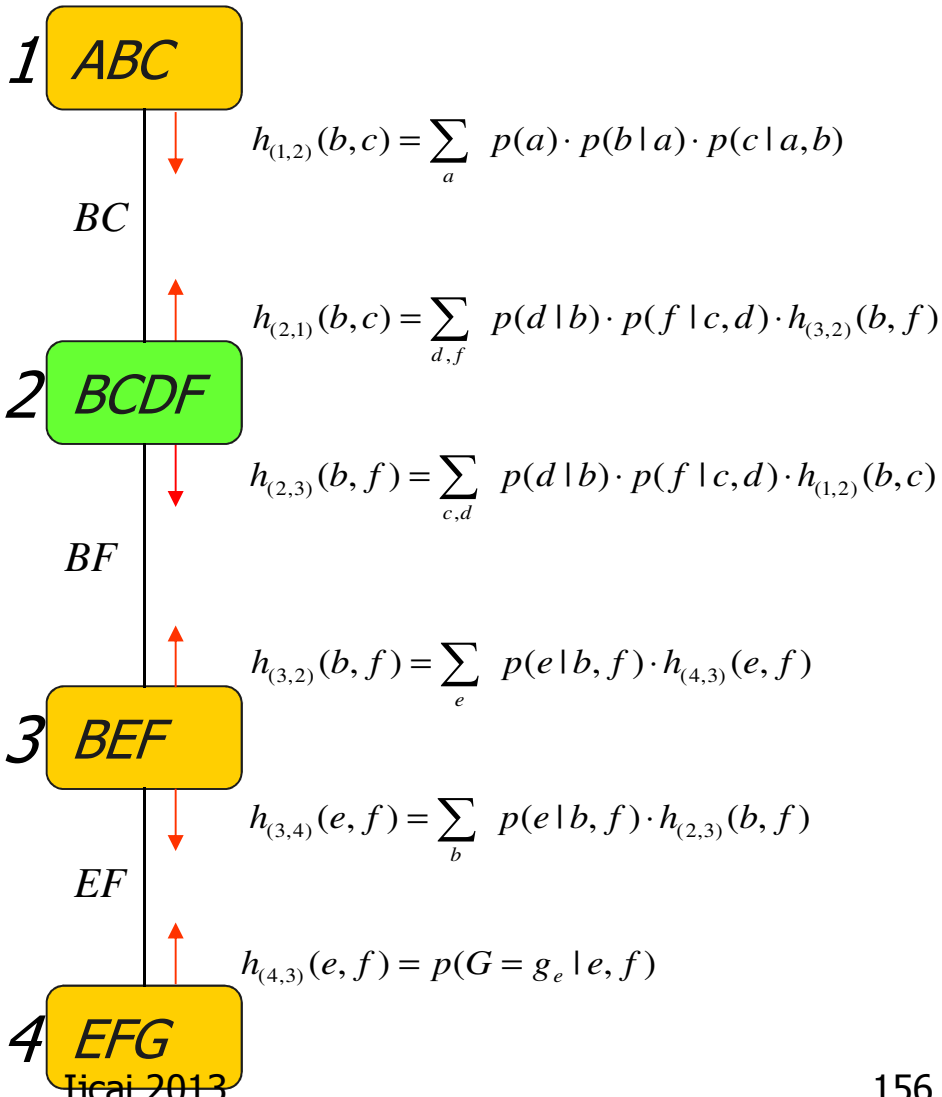
$$cluster(u) = \psi(u) \cup \{h(x_1,u), h(x_2,u), ..., h(x_n,u), h(v,u)\}$$

Compute the message :

$$h(u,v) = \sum_{elim(u,v)} \psi(u) \prod_{j \in neigh(u) - \{h(v,u)\}} h(j,u)$$

Join-Tree Clustering





EXACT algorithm

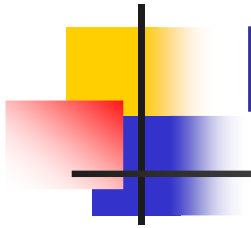
Time and space:
exp(cluster size)=
exp(treewidth)

Cluster Tree Elimination - properties

- Correctness and completeness: Algorithm CTE is correct, i.e. it computes the exact joint probability of a single variable and the evidence.
- Time complexity:
  - $O (deg \times (n+N) \times d^{w^*+1})$
- Space complexity: $O (N \times d^{sep})$

where
deg = the maximum degree of a node
n = number of variables (= number of CPTs)
N = number of nodes in the tree decomposition
d = the maximum domain size of a variable
*w** = the induced width
sep = the separator size

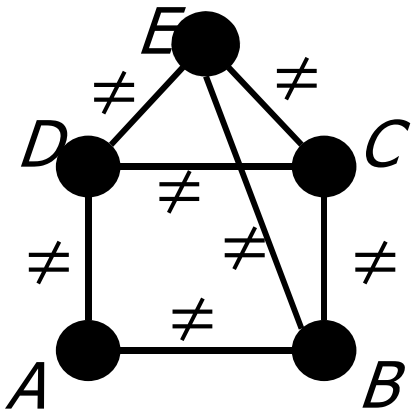
The algorithm makes local information global:
In each clutsre we have the marginal probability distribution



Road Map:Bayesian Networks

- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
  - Belied propagation
  - Mini-bucket, mini-clustering
  - Iterative join-graph propagation: a GBP scheme
- Search, conditioning

From Directional i-consistency to Mini-buckets



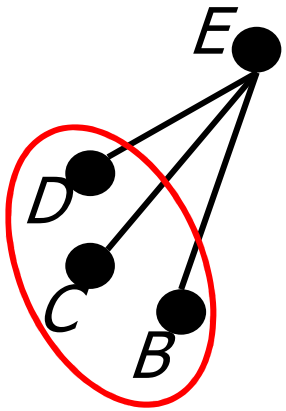
E : $E \neq D, E \neq C, E \neq B$

D : $D \neq C, D \neq A$

C : $C \neq B$

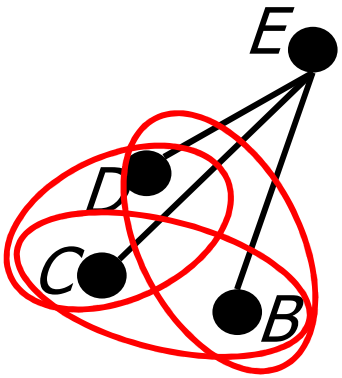
B : $A \neq B$

A :



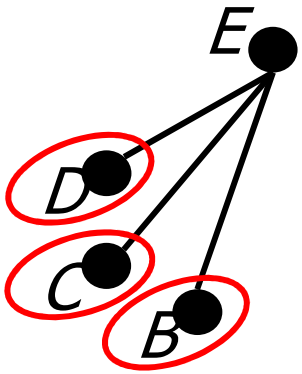
Adaptive

R_{DCB}



d-path

R_{DC}, R_{DB}
R_{CB}



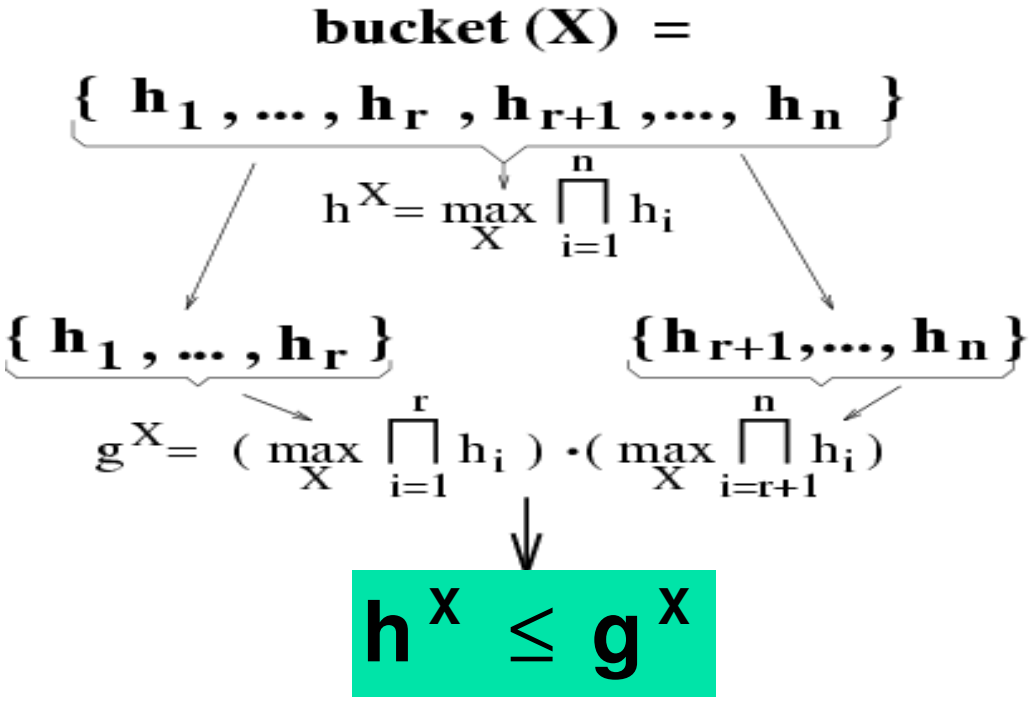
d-arc

R_D
R_C
R_D

The idea of Mini-bucket (Dechter and Rish 1997) (for optimization)

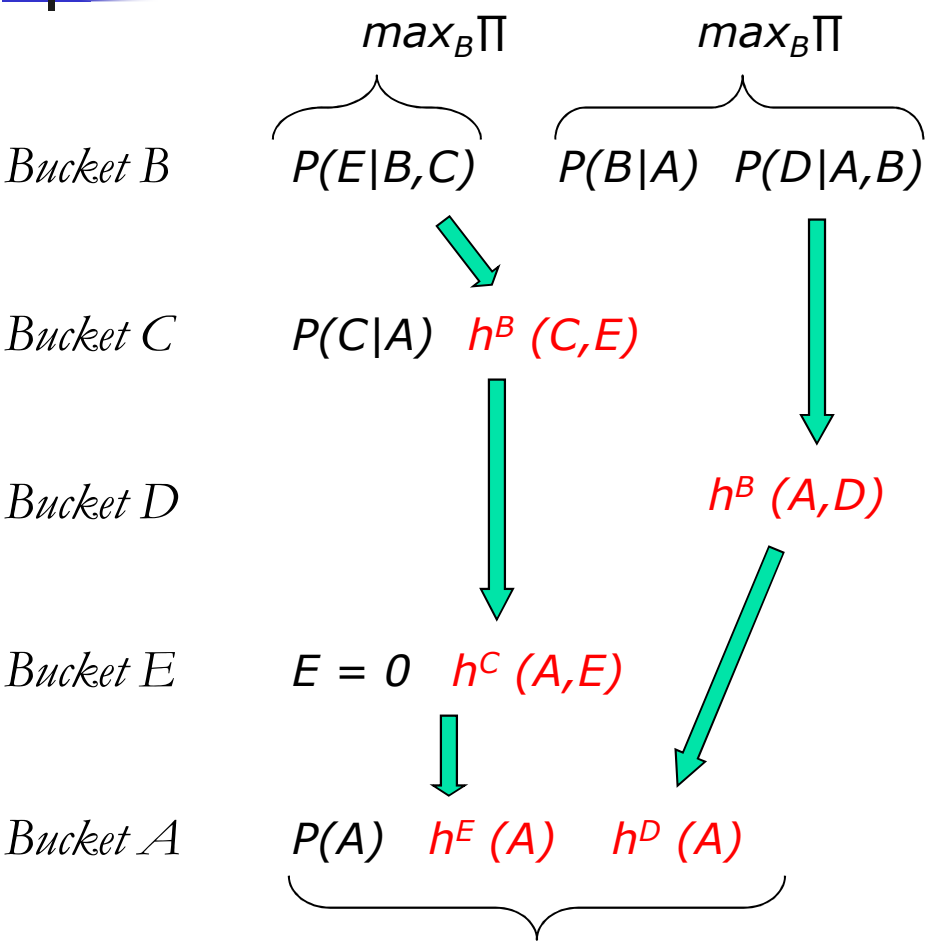
Local computation: bound the size of recorded dependencies

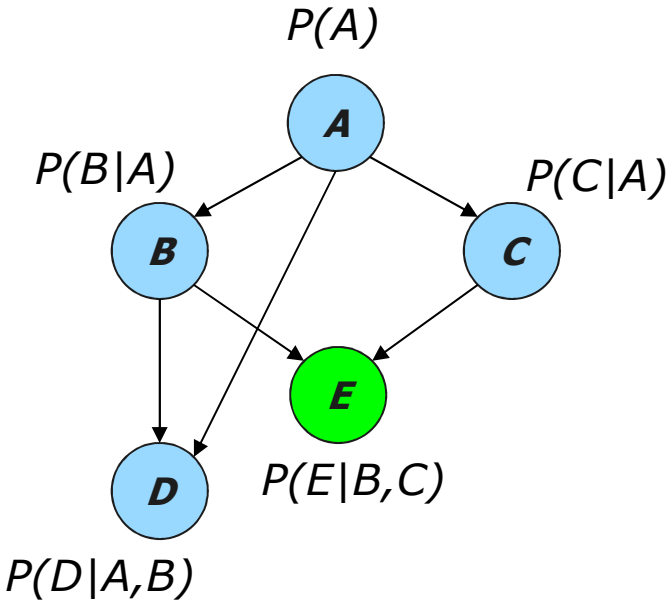
Split a bucket into mini-buckets =>bound complexity



$$\text{Exponential complexity decrease}: O(e^n) \rightarrow O(e^r) + O(e^{n-r})$$

Mini-Bucket Elimination



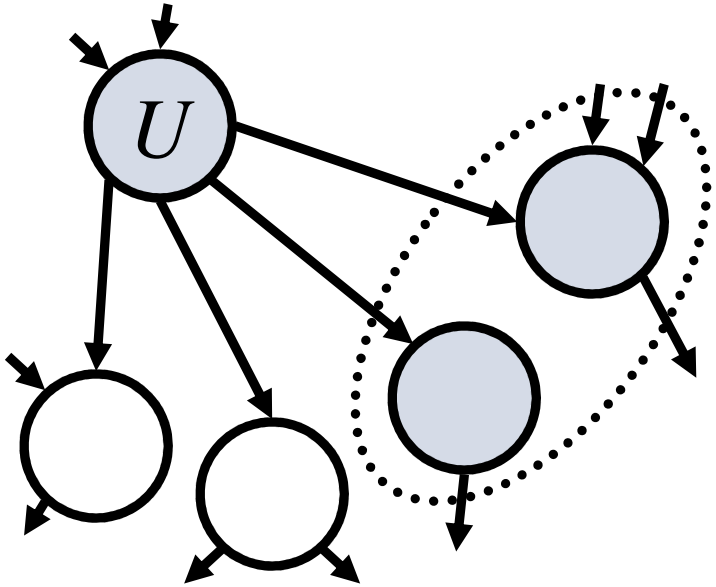


MPE* is an upper bound on MPE --U
Generating a solution yields a lower bound--L

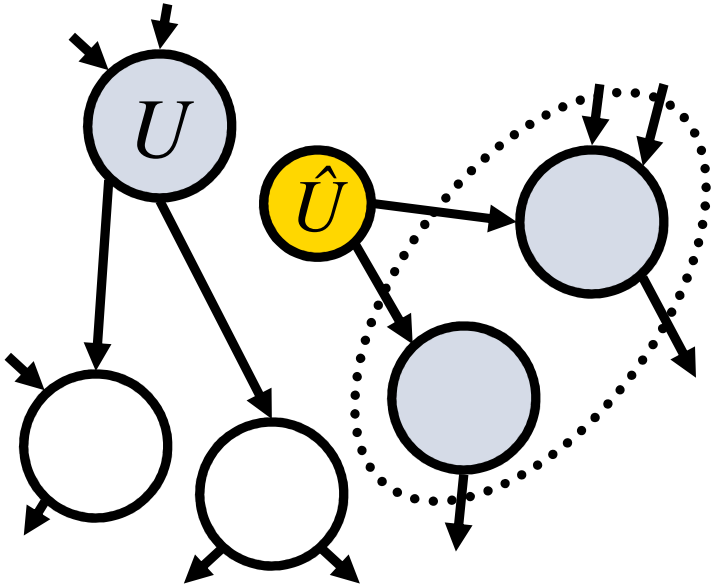
Semantics of Mini-Bucket: Splitting a Node

Variables in different buckets are renamed and duplicated
(Kask et. al., 2001), (Geffner et. al., 2007), (Choi, Chavira, Darwiche , 2007)

Before Splitting:
Network N



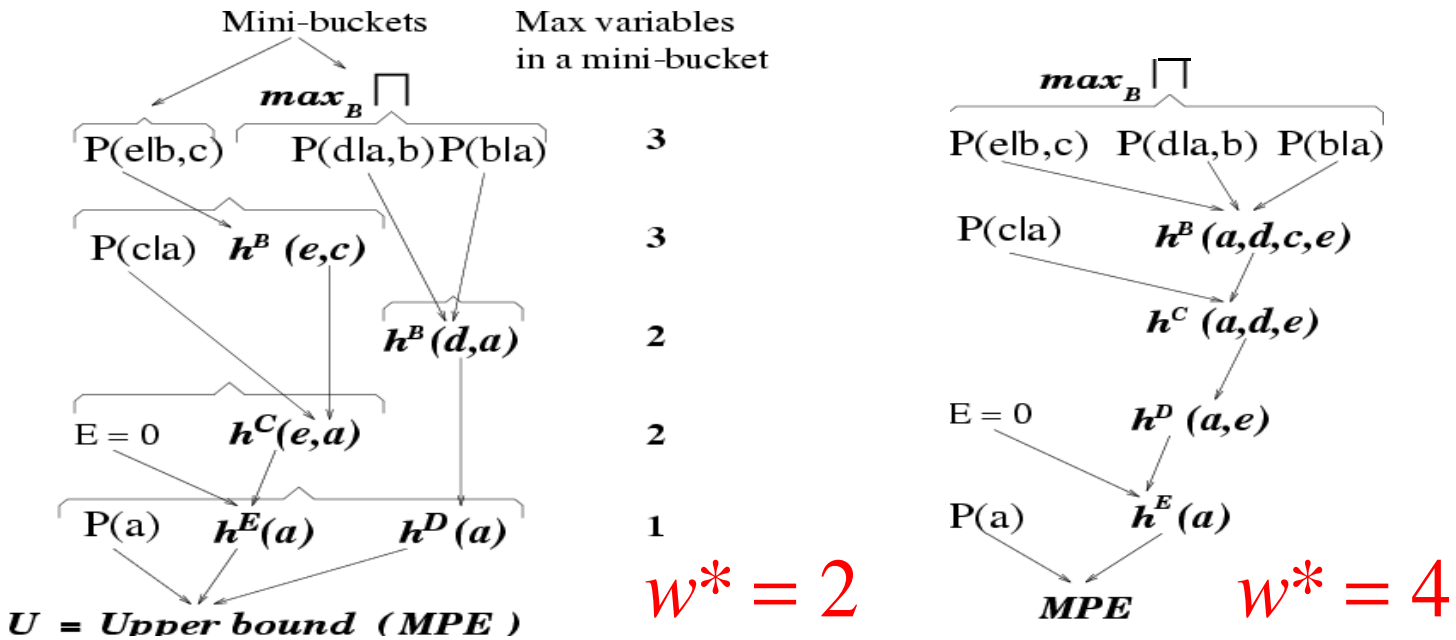
After Splitting:
Network N'



MBE(i) (Dechter and Rish 1997)

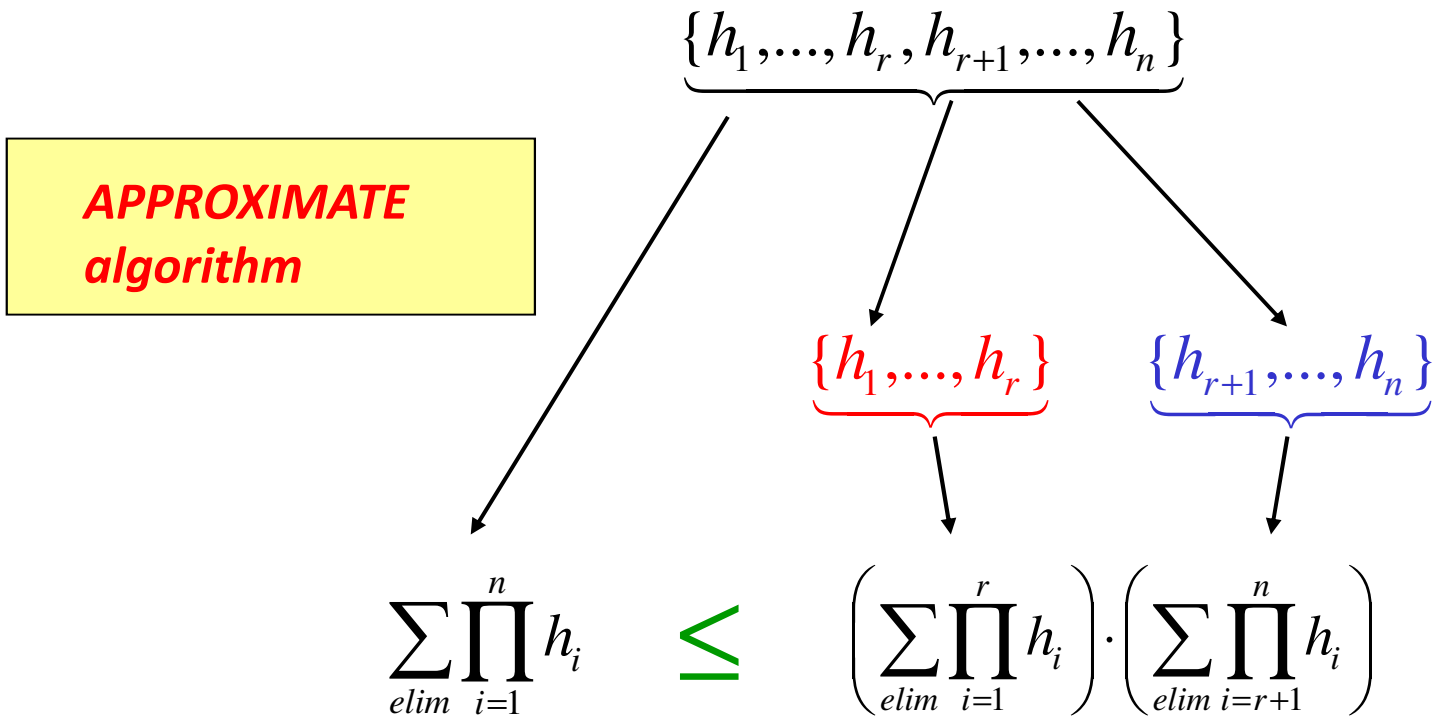
- **Input: i – max number of variables allowed in a mini-bucket**
- **Output: [lower bound (P of a sub-optimal solution), upper bound]**

Example: approx-mpe(3) versus elim-mpe



Mini-Clustering (for Sum-Product)

Split a cluster into mini-clusters => bound complexity



Exponential complexity decrease $O(e^n) \rightarrow O(e^{\mathrm{var}(r)}) + O(e^{\mathrm{var}(n-r)})$

MBE for Likelihood Computation

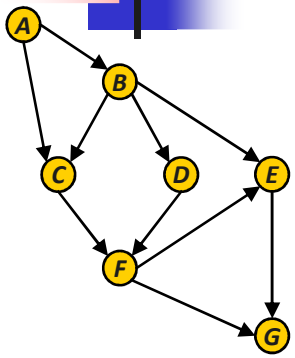
- Idea mini-bucket is the same:

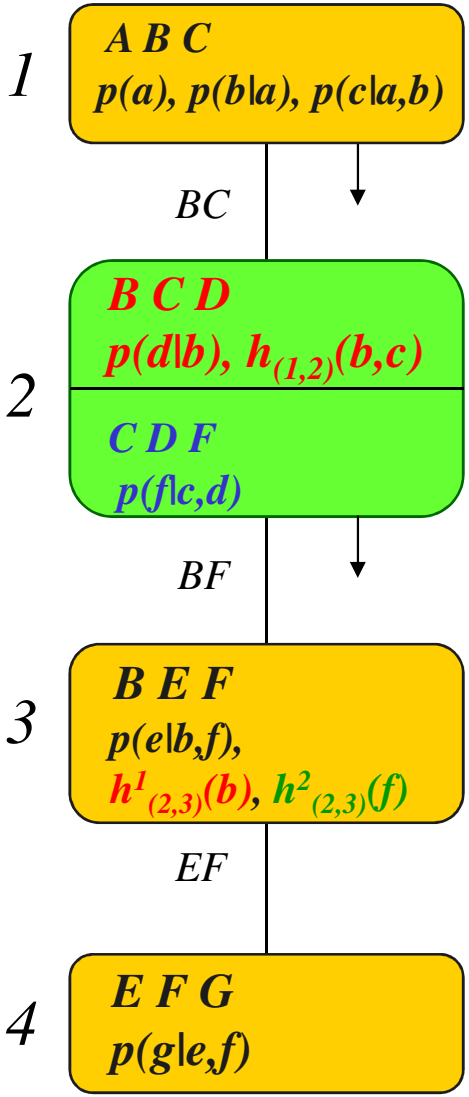
$$\sum_X f(x) \bullet g(x) \leq \sum_X f(x) \bullet \sum_X g(x)$$

$$\sum_X f(x) \bullet g(x) \leq \sum_X f(x) \bullet \max_X g(X)$$

- So we can apply a sum in each mini-bucket, or better, one sum and the rest max, or min (for lower-bound)
- MBE-bel-max(i), MBE-bel-min(i) generating upper and lower-bound on beliefs approximates BE-bel
- MBE-map(i): max buckets will be maximized, sum buckets will be sum-max. Approximates BE-map.

Mini-Clustering, i-bound=3





$$h^1_{(1,2)}(b,c) = \sum_a p(a) \cdot p(b \mid a) \cdot p(c \mid a,b)$$

$$h^1_{(2,3)}(b) = \sum_{c,d} p(d \mid b) \cdot h^1_{(1,2)}(b,c)$$

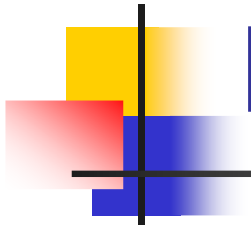
$$h^2_{(2,3)}(f) = \max_{c,d} p(f \mid c,d)$$

APPROXIMATE algorithm

Time and space:
exp(i-bound)

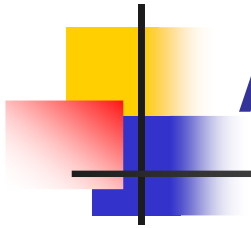
↓

Number of variables in a mini-cluster



Properties of MBE(i)/mc(I)

- **Complexity:** *O(r exp(i))* time and *O(exp(i)) space*.
- Yields an upper-bound and a lower-bound.
- **Accuracy:** determined by upper/lower (U/L) bound.
- As ***i*** increases, both accuracy and complexity increase.
- Possible use of mini-bucket approximations:
  - As anytime algorithms
  - As heuristics in search
- Other tasks: similar mini-bucket approximations for: belief updating, MAP and MEU (Dechter and Rish, 1997)



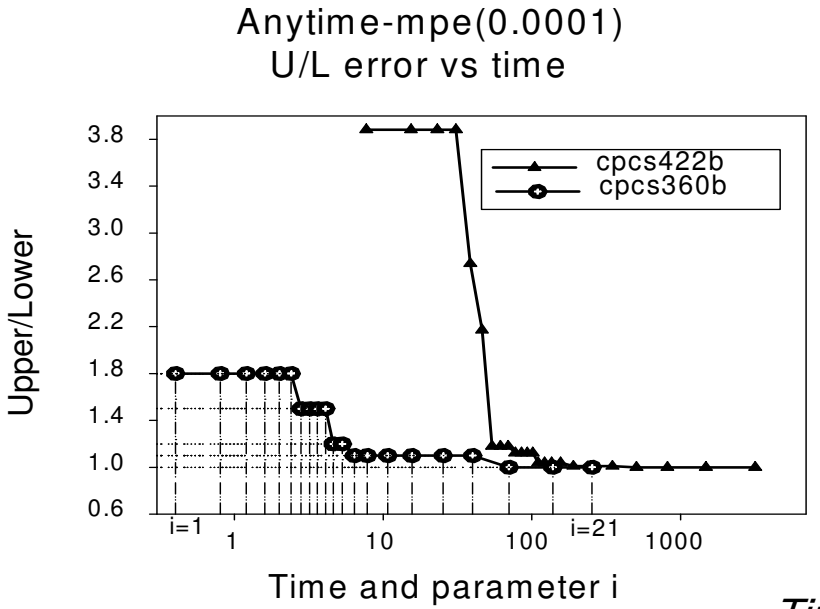
Anytime Approximation

anytime - mpe(ε)

Initialize : $i = i_0$
While time and space resources are available
$i \leftarrow i + i_{step}$
$U \leftarrow$ upper bound computed by *approx - mpe(i)*
$L \leftarrow$ lower bound computed by *approx - mpe(i)*
keep the best solution found so far
if $1 \leq \frac{U}{L} \leq 1 + \varepsilon$, return solution
end
return the largest L and the smallest U

CPCS networks – medical diagnosis (noisy-OR model)

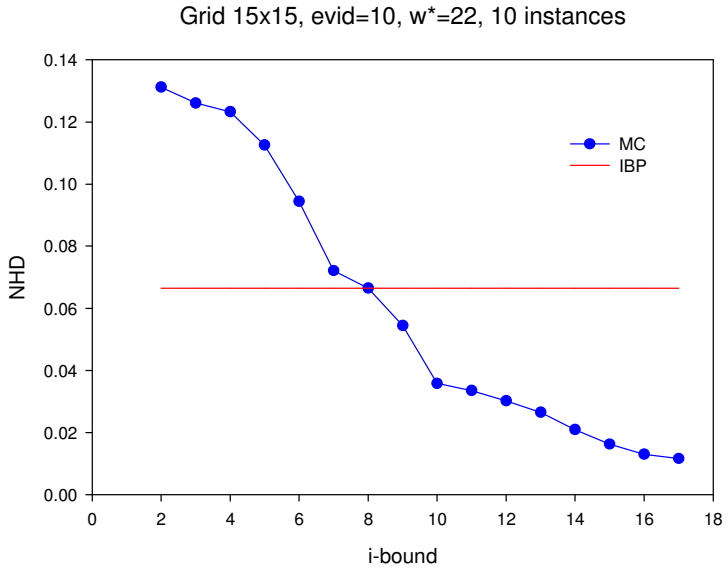
Test case: no evidence

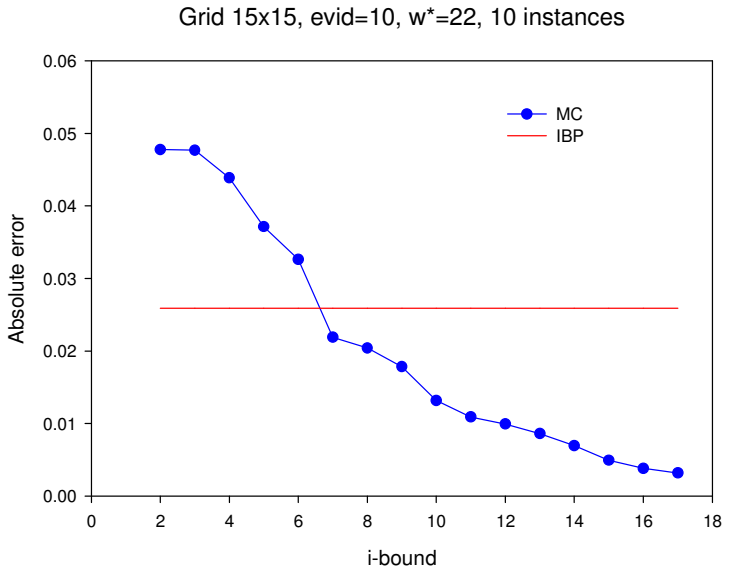


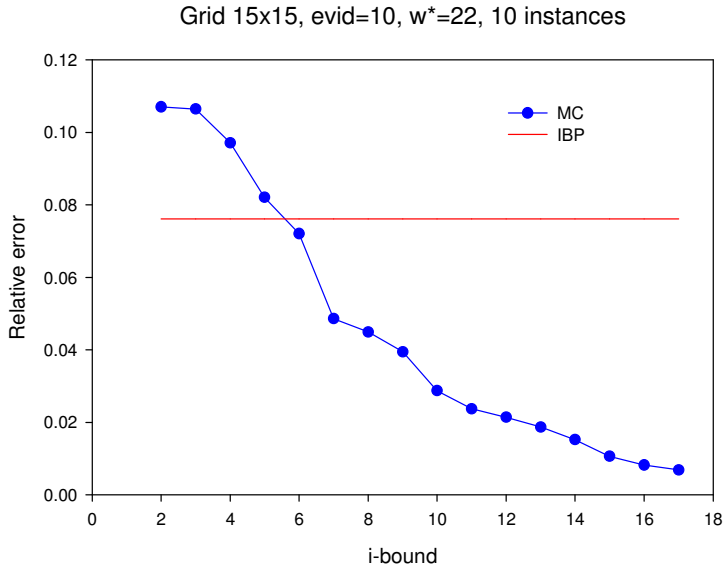
Time (sec)

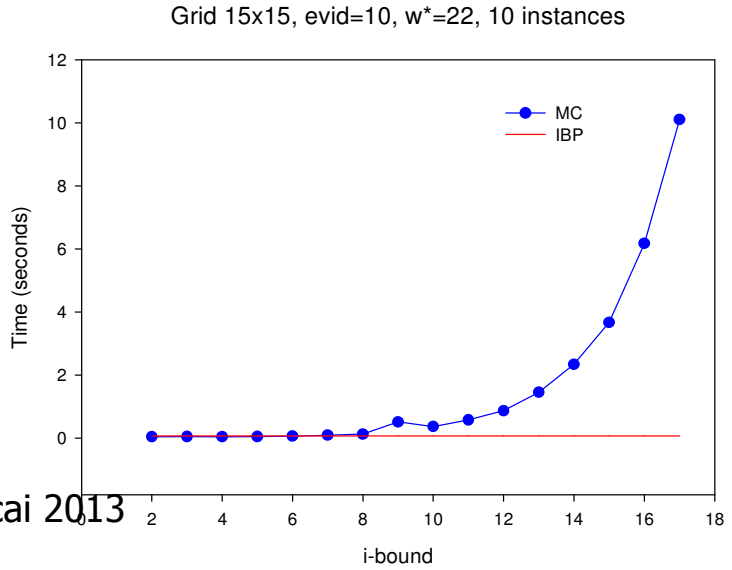
Algorithm	*cpcs360*	*cpcs422*
elim-mpe	*115.8*	*1697.6*
anytime-mpe(ε), $\varepsilon = 10^{-4}$	*70.3*	*505.2*
anytime-mpe(ε), $\varepsilon = 10^{-1}$	*70.3*	*110.5*

Grid 15x15 - 10 evidence





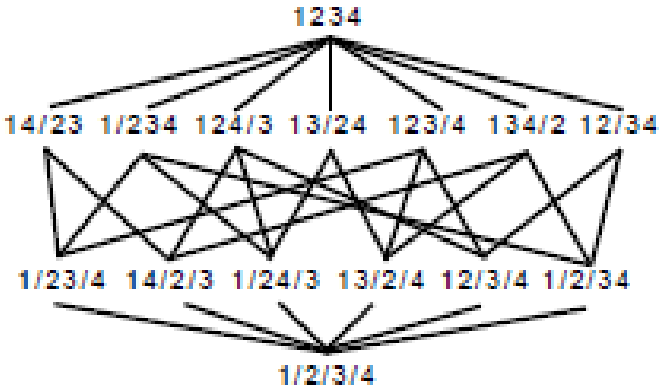




Heuristics for partitioning

(Dechter and Rish, 2003, Rollon and Dechter 2010, see also ijcai 2012)

Scope-based Partitioning Heuristic *(SCP) aims at minimizing the number of mini-buckets in the partition by including in each minibucket as many functions as respecting the i bound is satisfied*



Partitioning lattice of bucket $\{f_1, f_2, f_3, f_4\}$.

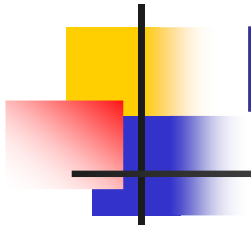
- *Log relative error*:

$$RE(f,h) = \sum_t (\log(f(t)) - \log(h(t)))$$

- *Max log relative error*:

$$MRE(f,h) = \max_t \{\log(f(t)) - \log(h(t))\}$$

Use greedy heuristic derived from a distance function to decide which functions go into a single mini-bucket

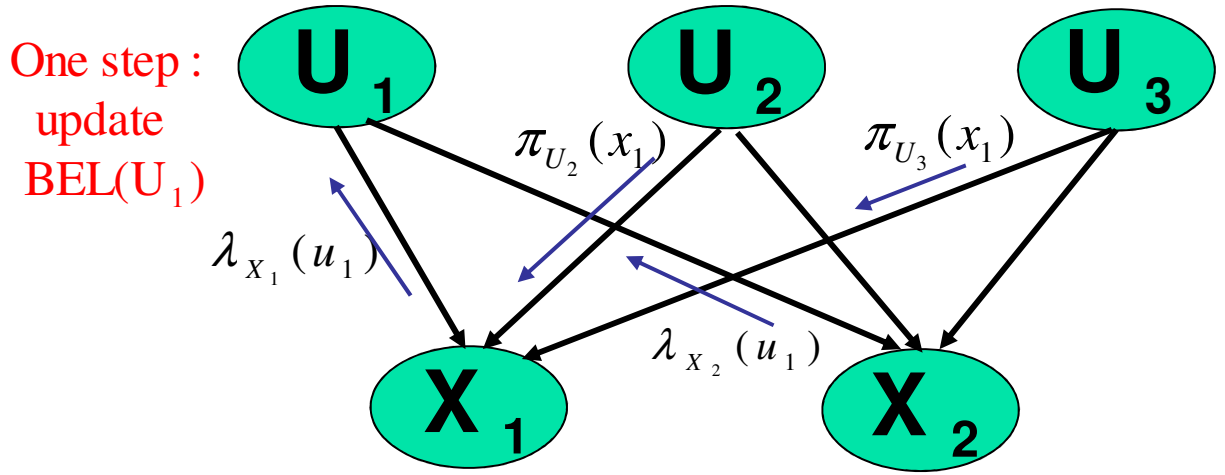


Road Map:Bayesian Networks

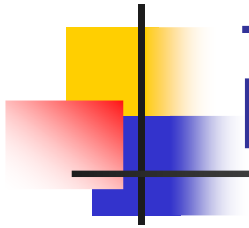
- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
  - Belied propagation
  - Mini-bucket, mini-clustering
  - Iterative join-graph propagation: a GBP scheme
- Search, conditioning

Iterative Belief Proapagation

- Belief propagation is exact for poly-trees
- IBP - applying BP iteratively to cyclic networks



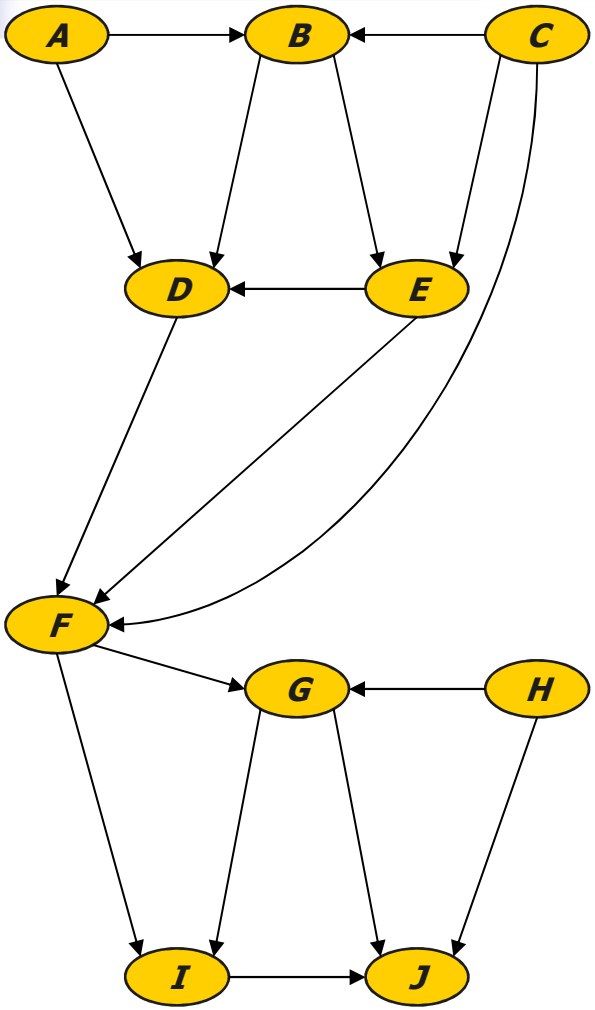
- No guarantees for convergence
- Works well for many coding networks
- Lets combine iterative-nature with anytime--IJGP



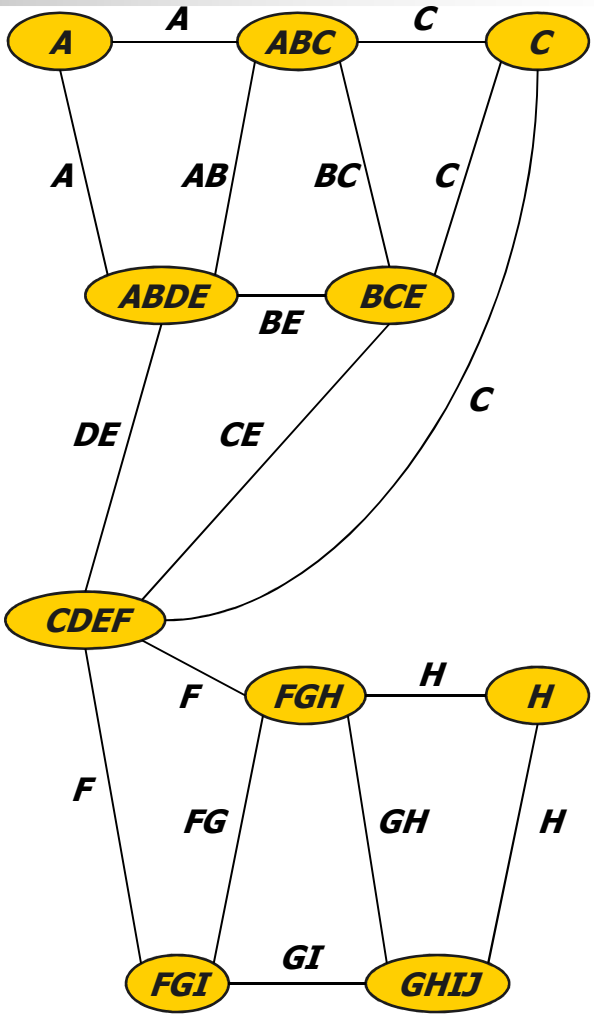
IJGP: a generalized Belief Propagation

- Apply belief propagation on a cluster-graph
- IJGP uses join-graph clustering which is both *anytime* and *iterative*
- IJGP applies message passing along a join-graph, rather than a join-tree
- Empirical evaluation shows that IJGP is almost always superior to other approximate schemes (IBP, MC)

IJGP - Example



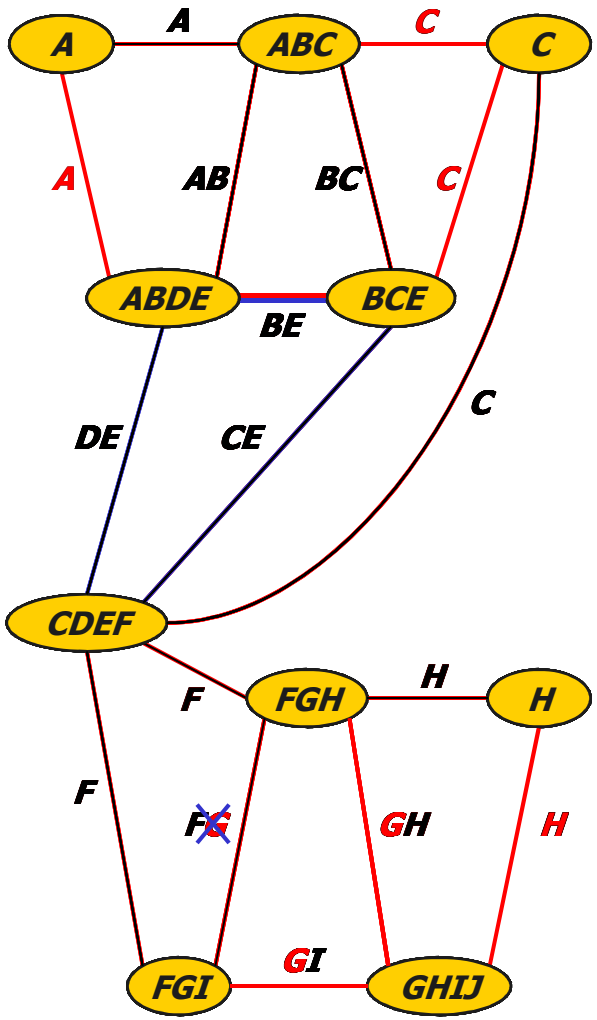
Belief network



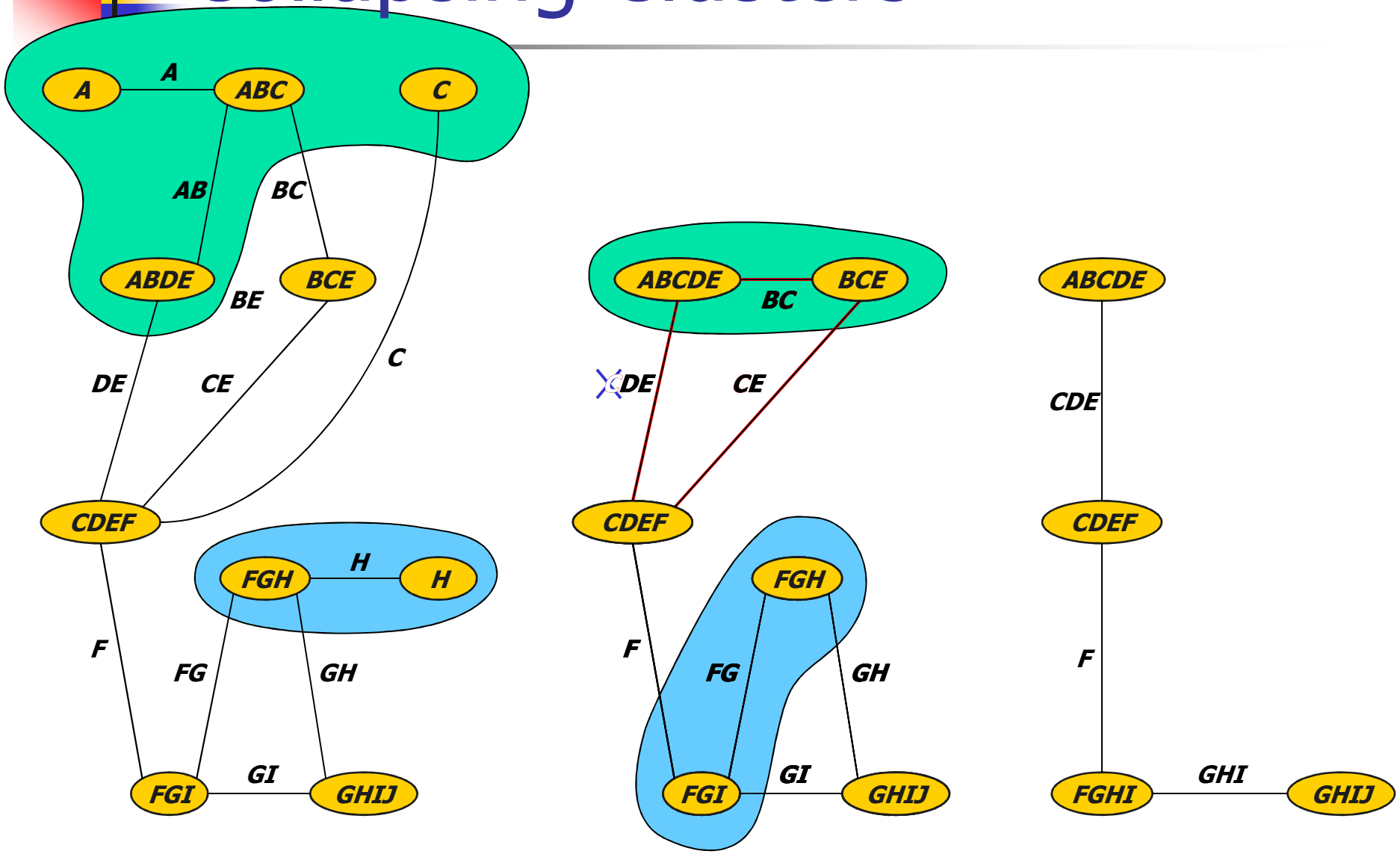
Loopy BP graph

Arc-Minimal Join-Graph

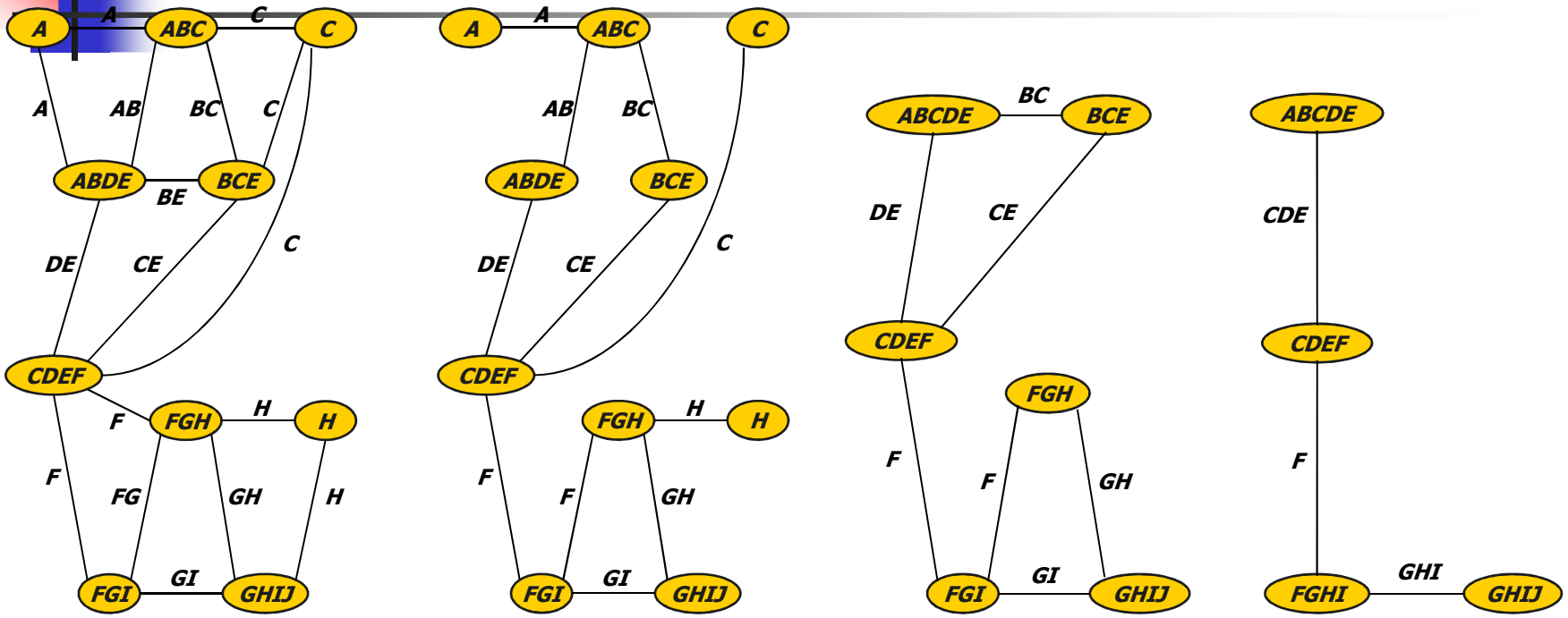
Arcs labeled with any single variable should form a TREE



Collapsing Clusters



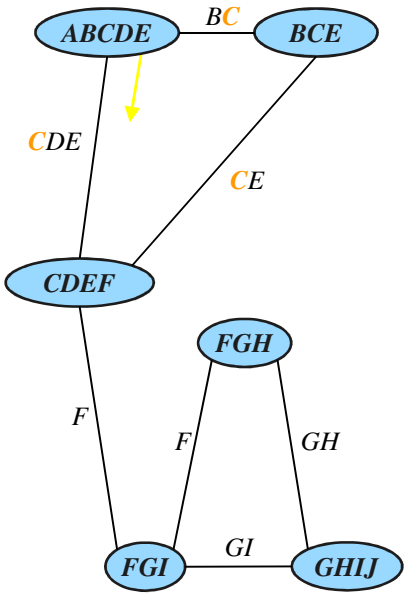
Join-Graphs

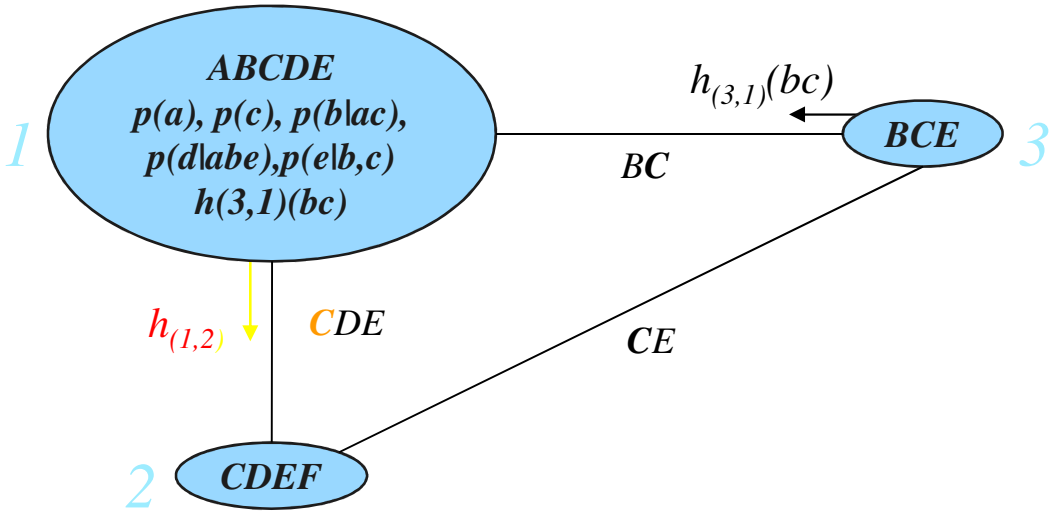


more accuracy

less complexity

Message propagation





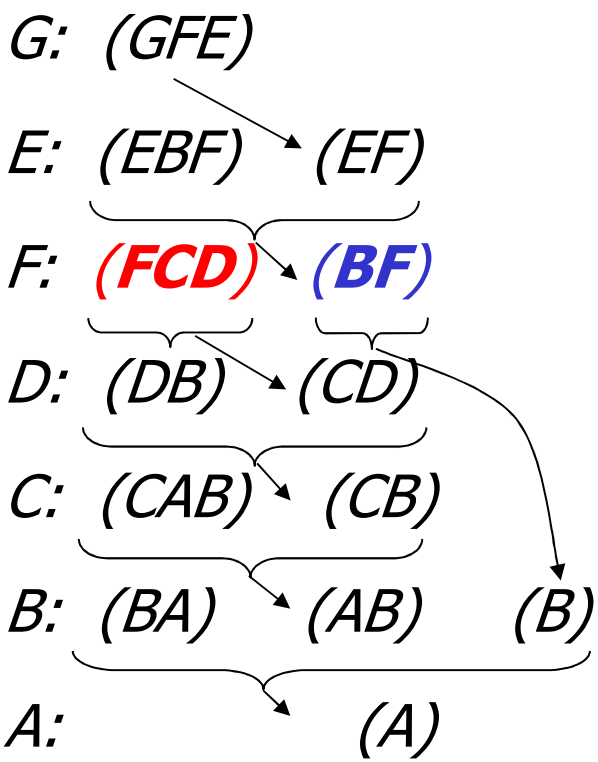
Minimal arc-labeled:
sep(1,2)={D,E}
elim(1,2)={A,B,C}

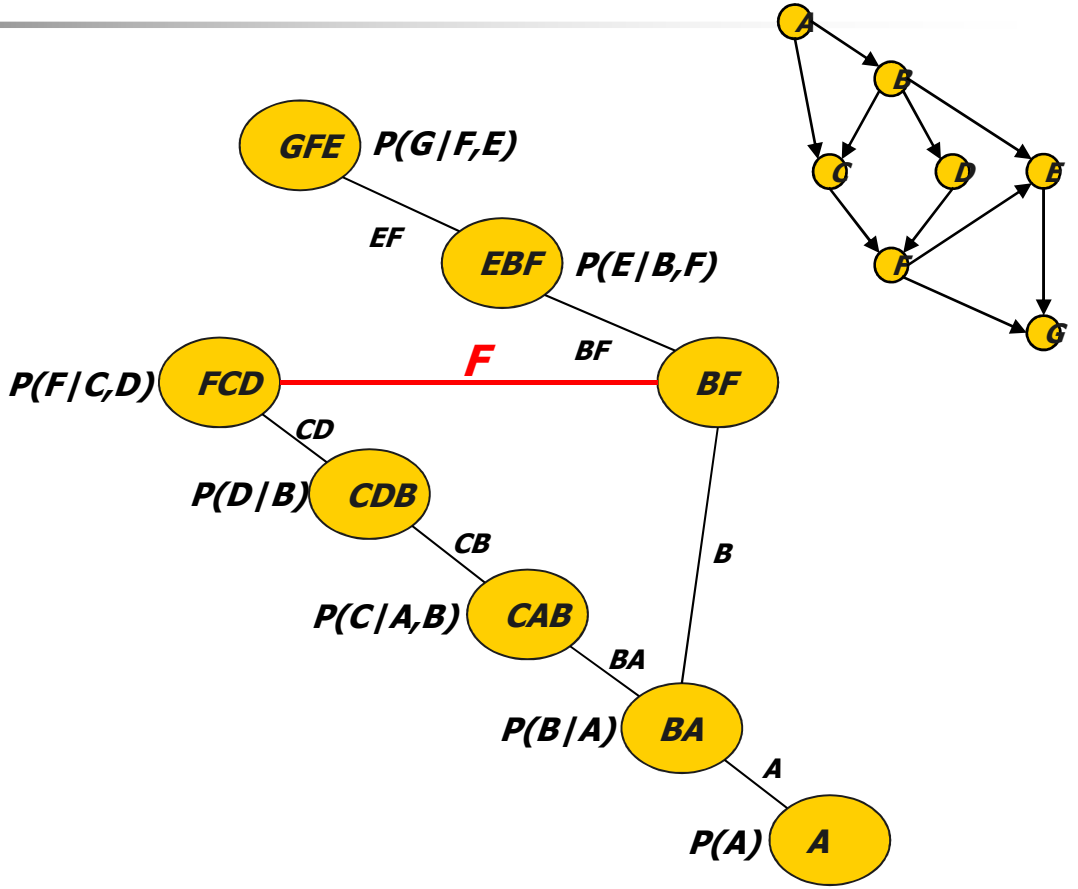
$$h_{(1,2)}(de) = \sum_{a,b,c} p(a)\,p(c)\,p(b \mid ac)\,p(d \mid abe)\,p(e \mid bc)\,h_{(3,1)}(bc)$$

Non-minimal arc-labeled:
sep(1,2)={C,D,E}
elim(1,2)={A,B}

$$h_{(1,2)}(cde) = \sum_{a,b} p(a)\,p(c)\,p(b \mid ac)\,p(d \mid abe)\,p(e \mid bc)\,h_{(3,1)}(bc)$$

Constructing Join-Graphs

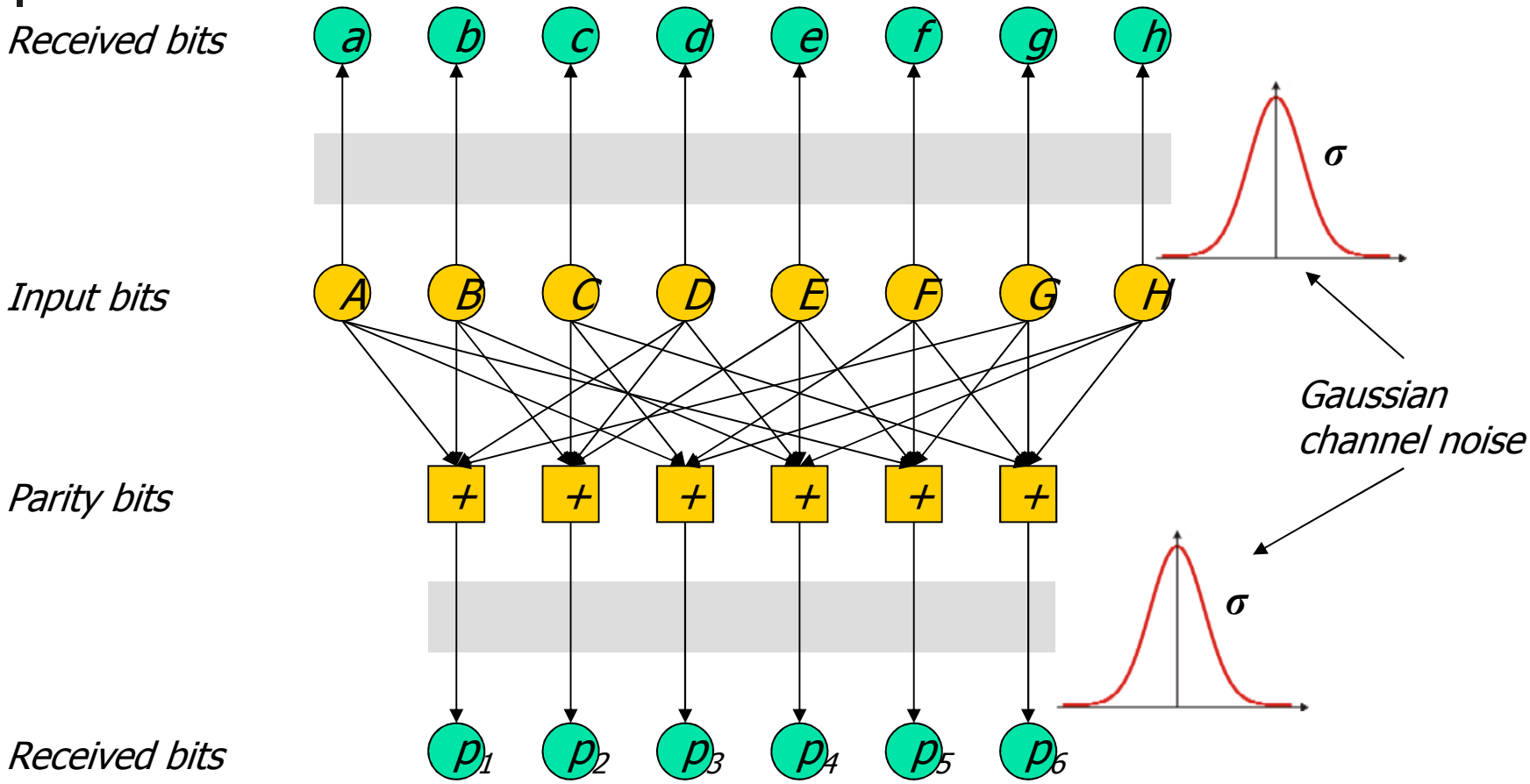




a) schematic mini-bucket(i), i=3

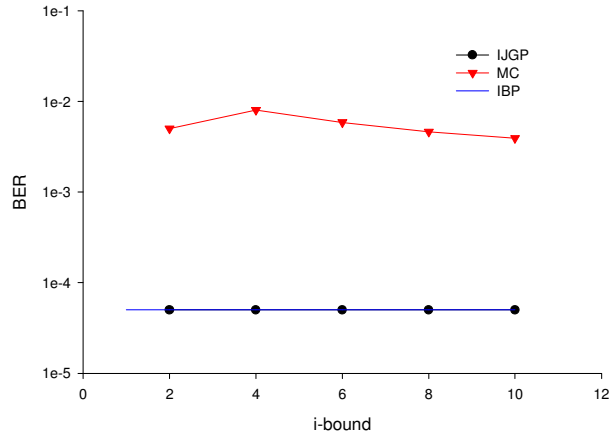
b) arc-labeled join-graph decomposition

Linear Block Codes

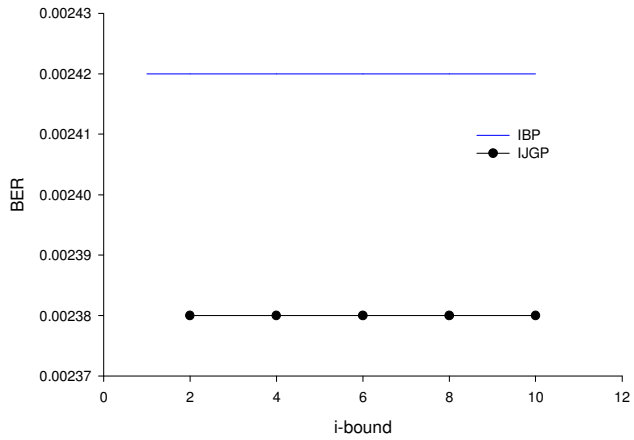


Coding Networks – Bit Error Rate

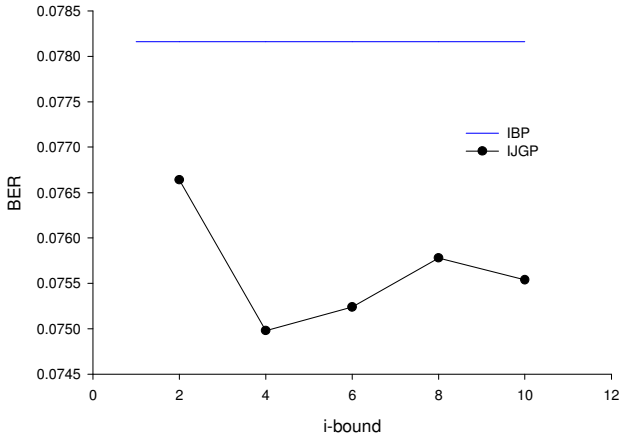
N=400, 1000 instances, 30 it, w*=43, $\sigma = .22$



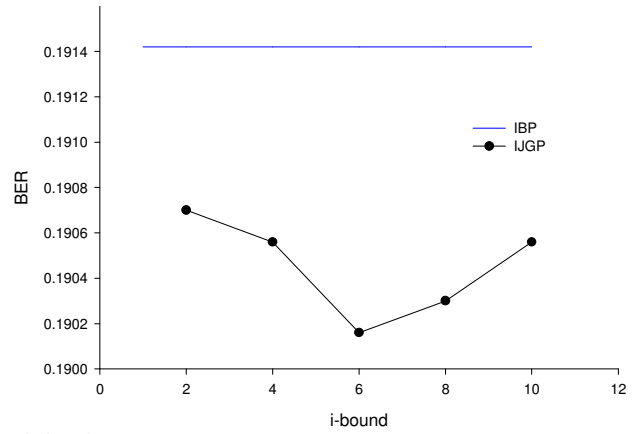
N=400, 500 instances, 30 it, w*=43, $\sigma = .32$



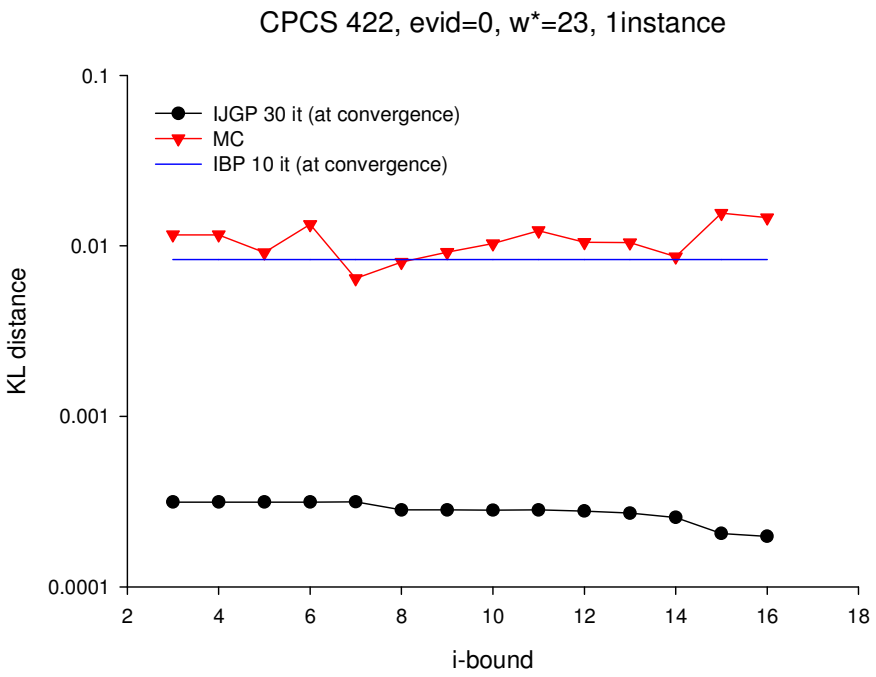
N=400, 500 instances, 30 it, w*=43, $\sigma = .51$



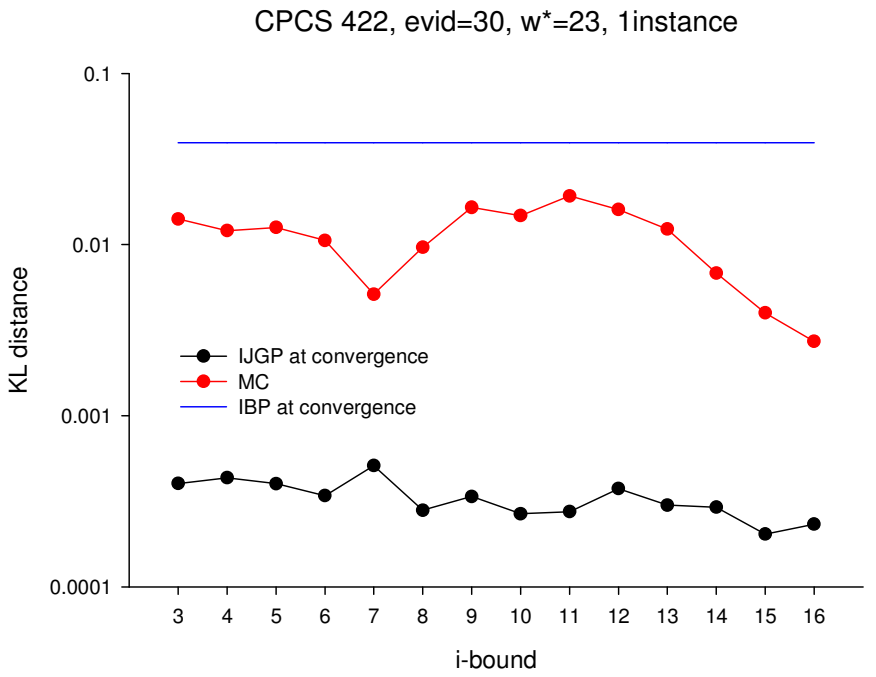
N=400, 500 instances, 30 it, w*=43, $\sigma = .65$



CPCS 422 – KL Distance

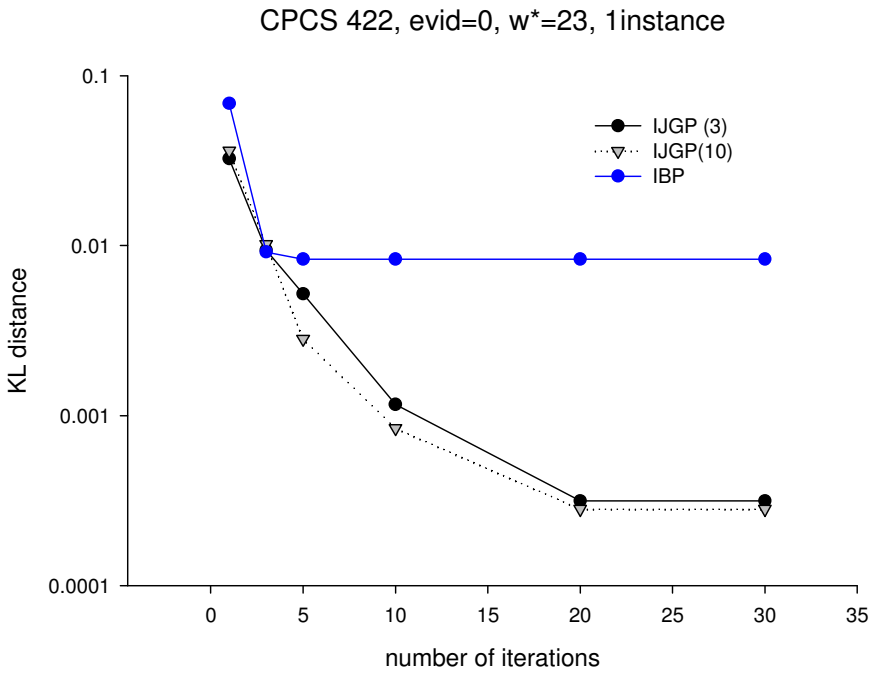


evidence=0

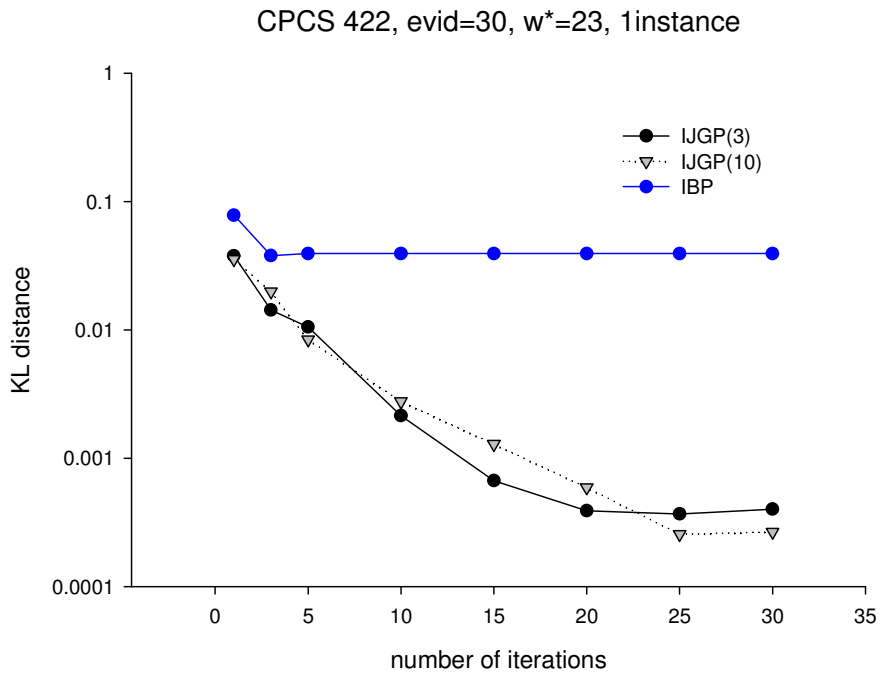


evidence=30

CPCS 422 – KL vs. Iterations



evidence=0



evidence=30

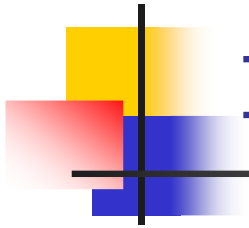
Summary of IJGP so far

A spectrum of approximations.

IBP: results from applying IJGP to the dual joingraph.

Jointree algorithm: results from applying IJGP to a jointree (as a joingraph).

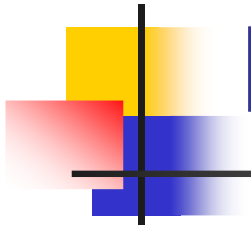
In between these two ends, we have a spectrum of joingraphs and corresponding factorizations, where IJGP seeks stationary points of the KL–divergence between these factorizations and the original distribution.



IBP – inference power for zero beliefs

- **IBP's trace for zero beliefs is identical to arc-consistency on the flat network**

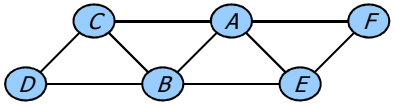
- **Consequently,**
  1. **Soundness:** The inference of zero beliefs by IBP converges in a finite number of iterations, and **all zero beliefs inferred are correct**
  2. **Incompleteness:** IBP may not infer all the true zero beliefs

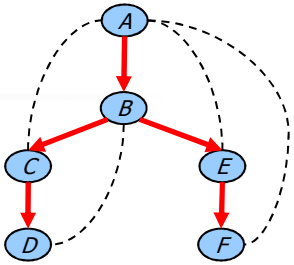


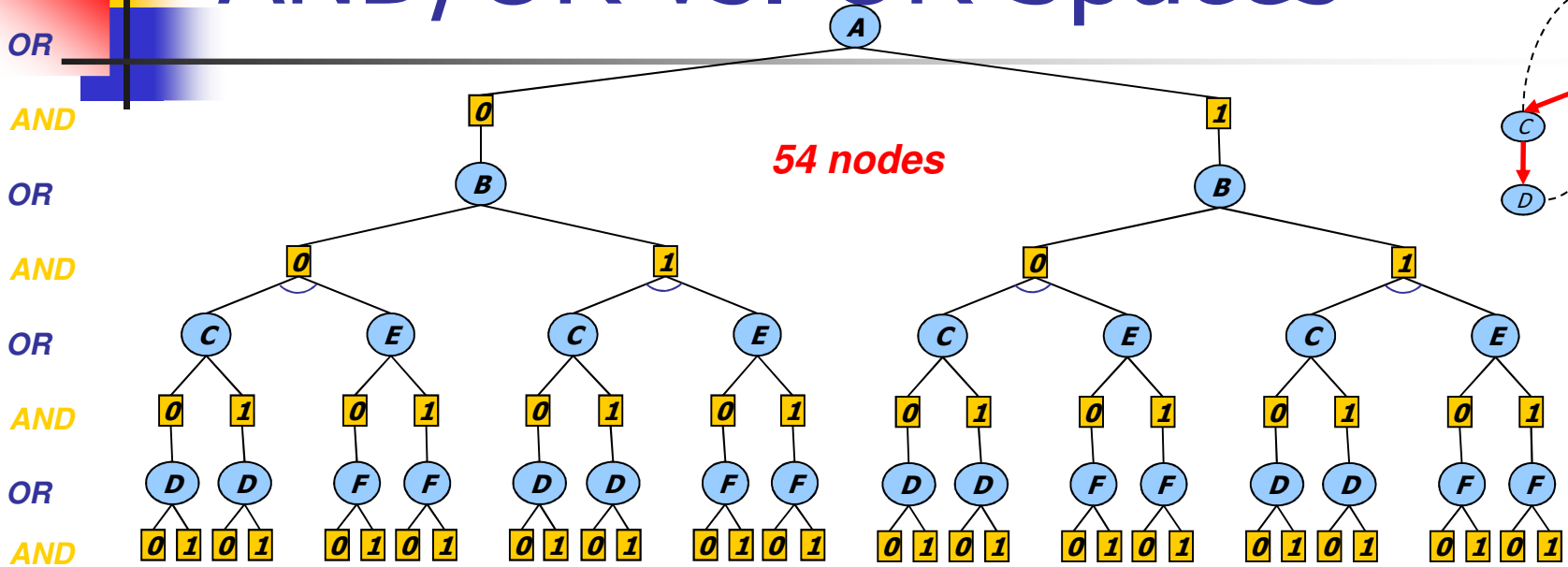
Road Map:Bayesian Networks

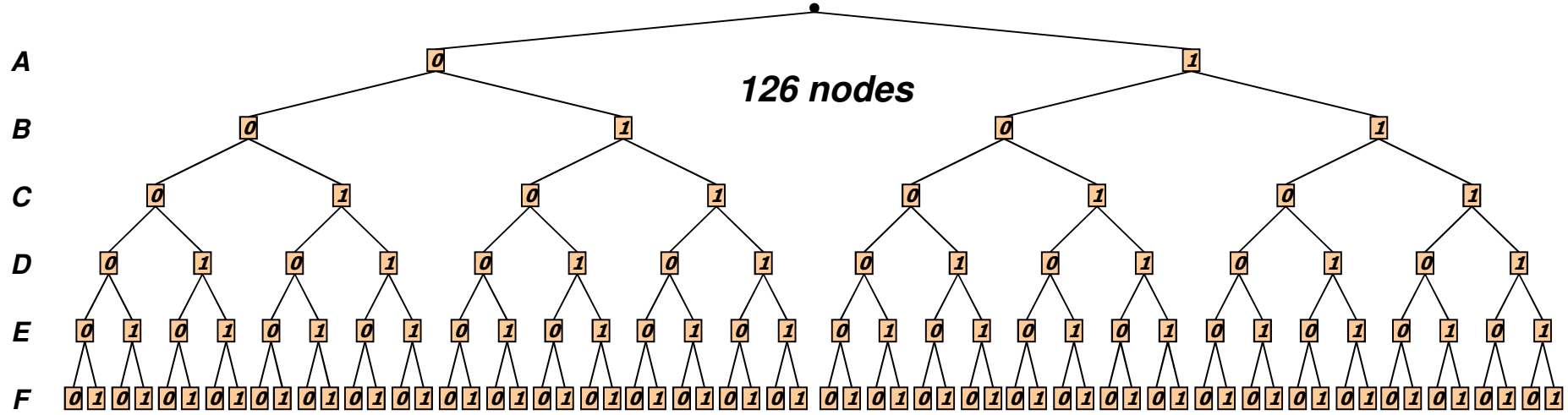
- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
  - Belied propagation
  - Mini-bucket, mini-clustering
  - Iterative join-graph propagation: a GBP scheme
- Search, conditioning
- Hybrid of Search and Inference

AND/OR vs. OR Spaces









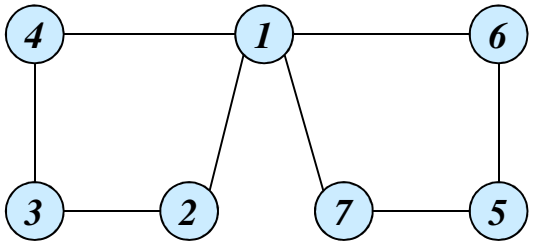
Complexity of AND/OR Tree Search

	AND/OR tree	**OR tree**
Space	O(n)	O(n)
Time	$O(n\ d^t)$ $O(n\ d^{w^* \log n})$ (Freuder & Quinn85), (Collin, Dechter & Katz91), (Bayardo & Miranker95), (Darwiche01)	$O(d^n)$

d = domain size
t = depth of pseudo-tree
n = number of variables
w = treewidth*

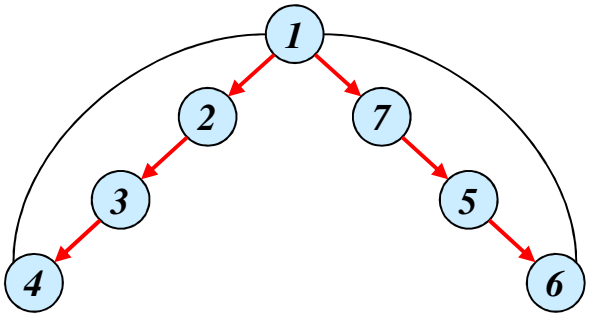
Pseudo-Trees

(Freuder 85, Bayardo 95, Bodlaender and Gilbert, 91)

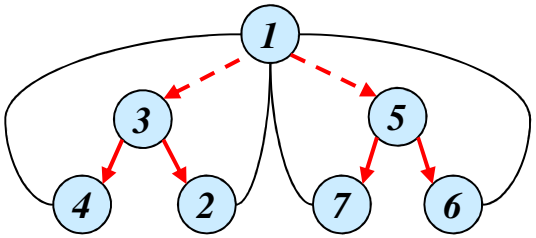


(a) Graph

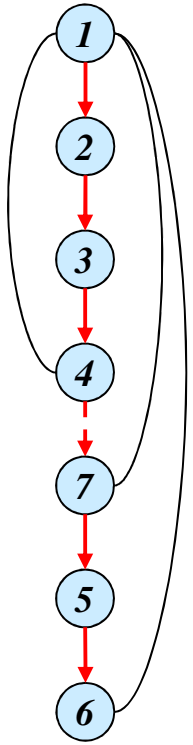
$$t <= w^* \log n$$



(b) DFS tree
depth=3

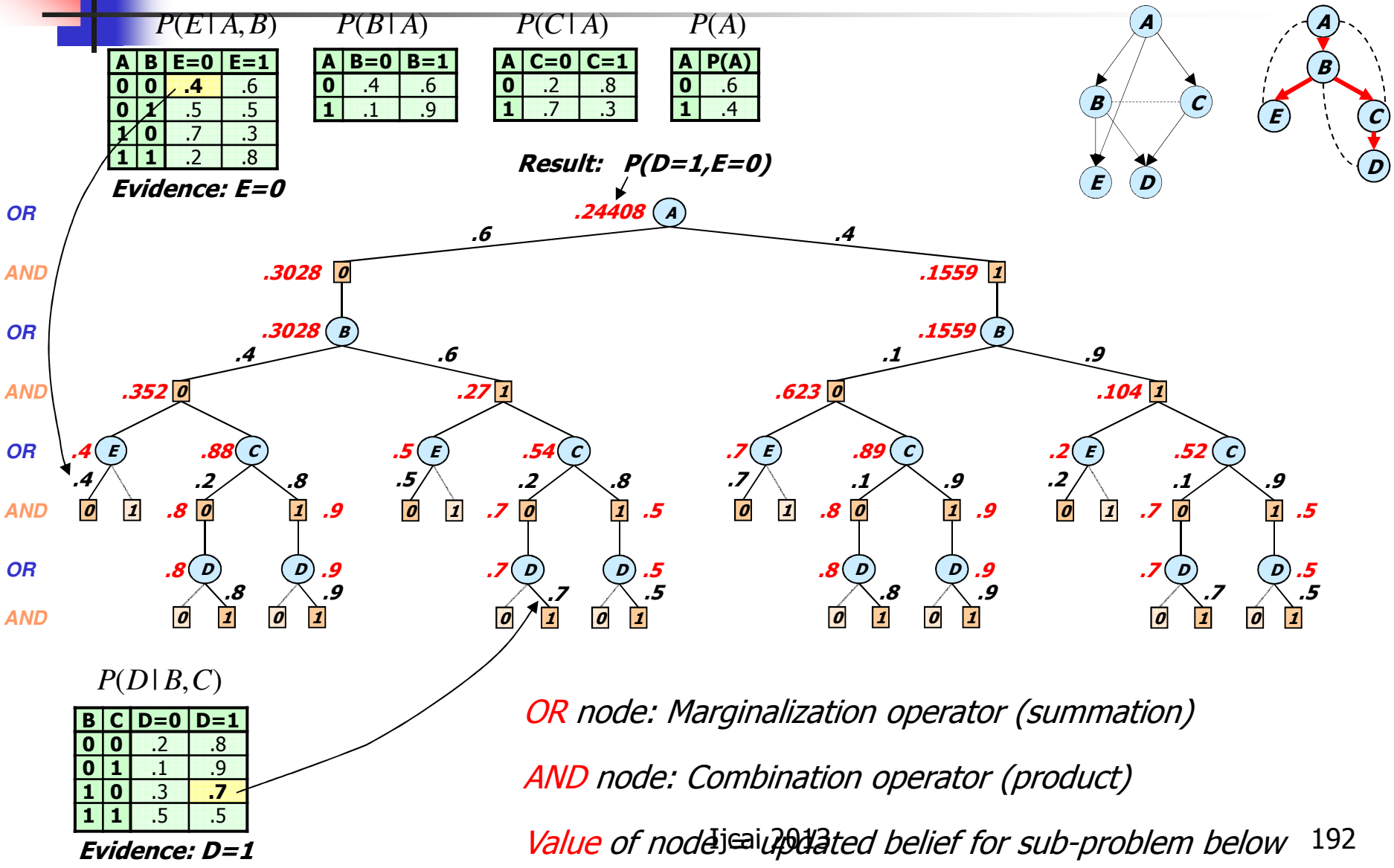


(c) pseudo- tree
depth=2



(d) Chain
depth=6

Weighted AND/OR Tree



A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

A	B=0	B=1
0	.4	.6
1	.1	.9

A	C=0	C=1
0	.2	.8
1	.7	.3

A	P(A)
0	.6
1	.4

B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

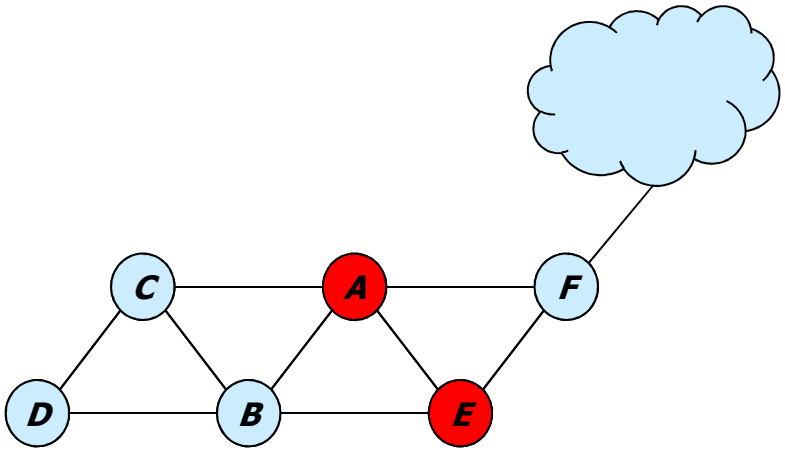
OR node: Marginalization operator (summation)

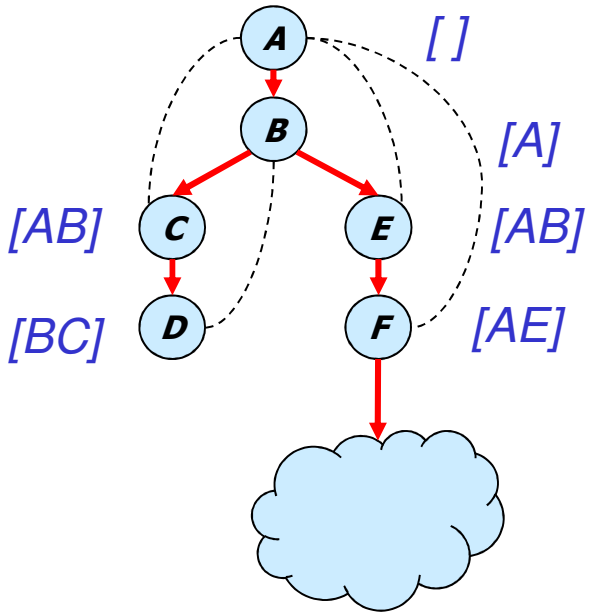
AND node: Combination operator (product)

Value of node = updated belief for sub-problem below

Merging based on context

context (X) = *ancestors of X connected to* → *X*, *descendants of X*



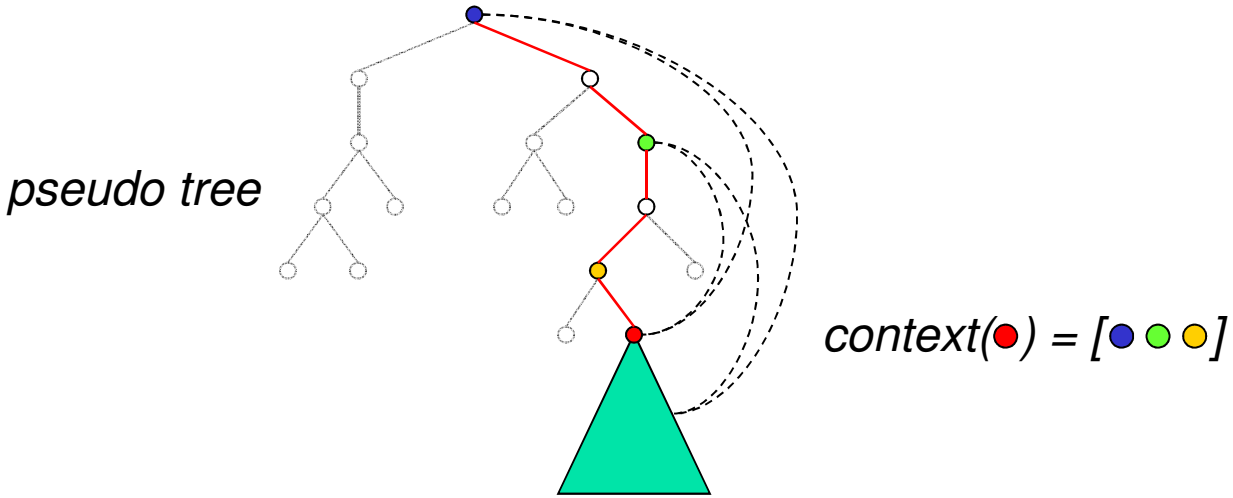


How big is the context?

context (X) = ancestors of X in pseudo tree that are connected to X, or to descendants of X

context (X) = parents in the induced graph

max |context| = induced width = treewidth



Weighted AND/OR Tree for Bayesian Network

$P(E \mid A,B)$

A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

$P(B \mid A)$

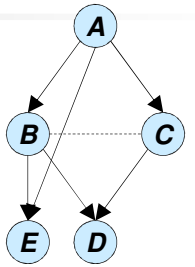
A	B=0	B=1
0	.4	.6
1	.1	.9

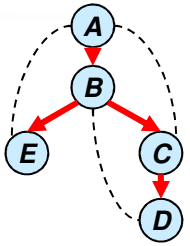
$P(C \mid A)$

A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4





Evidence: E=0

OR

AND

OR

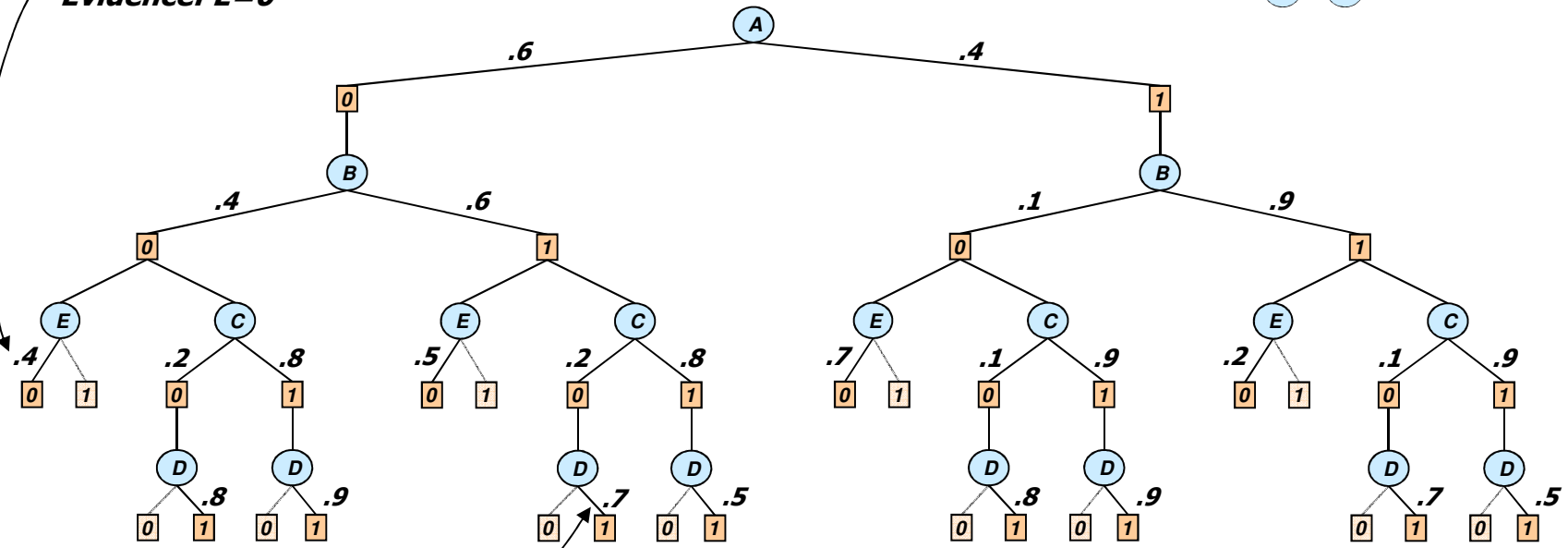
AND

OR

AND

OR

AND



$P(D \mid B,C)$

B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

Evidence: D=1

Weighted AND/OR Tree for Bayesian Network Query: Belief updating (Sum-Product Networks)

$P(E \mid A, B)$

A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

Evidence: E=0

$P(B \mid A)$

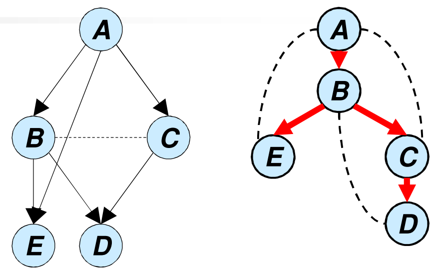
A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C \mid A)$

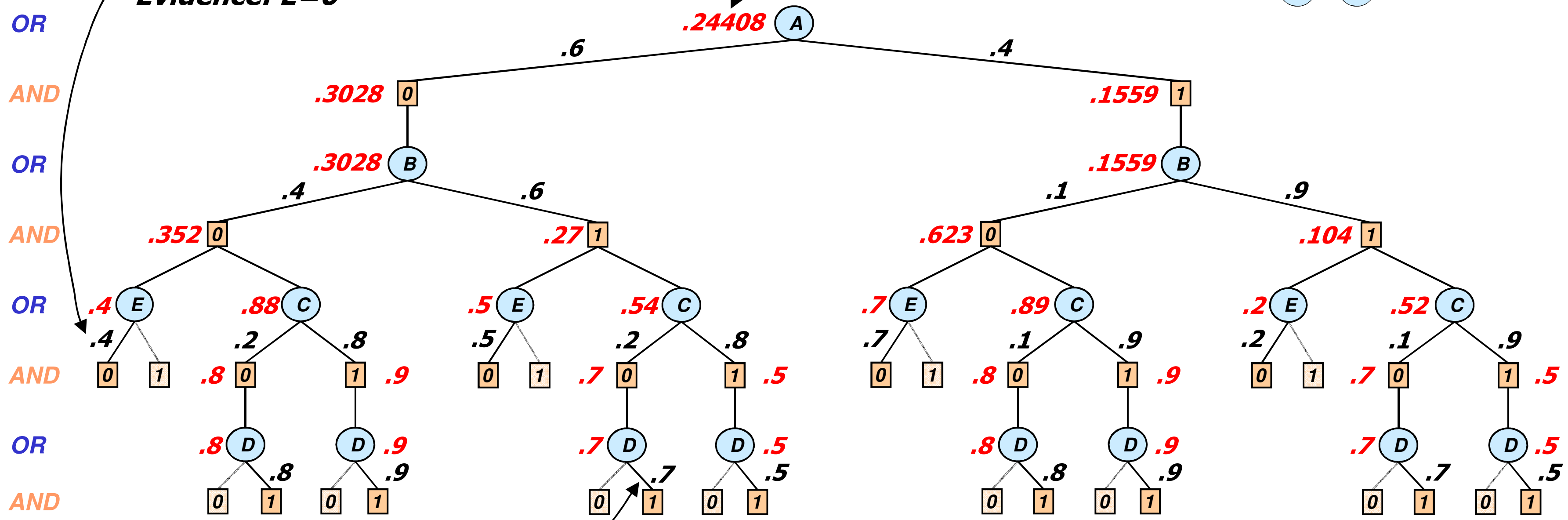
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4



Result: P(D=1,E=0)



$P(D \mid B, C)$

B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

Evidence: D=1

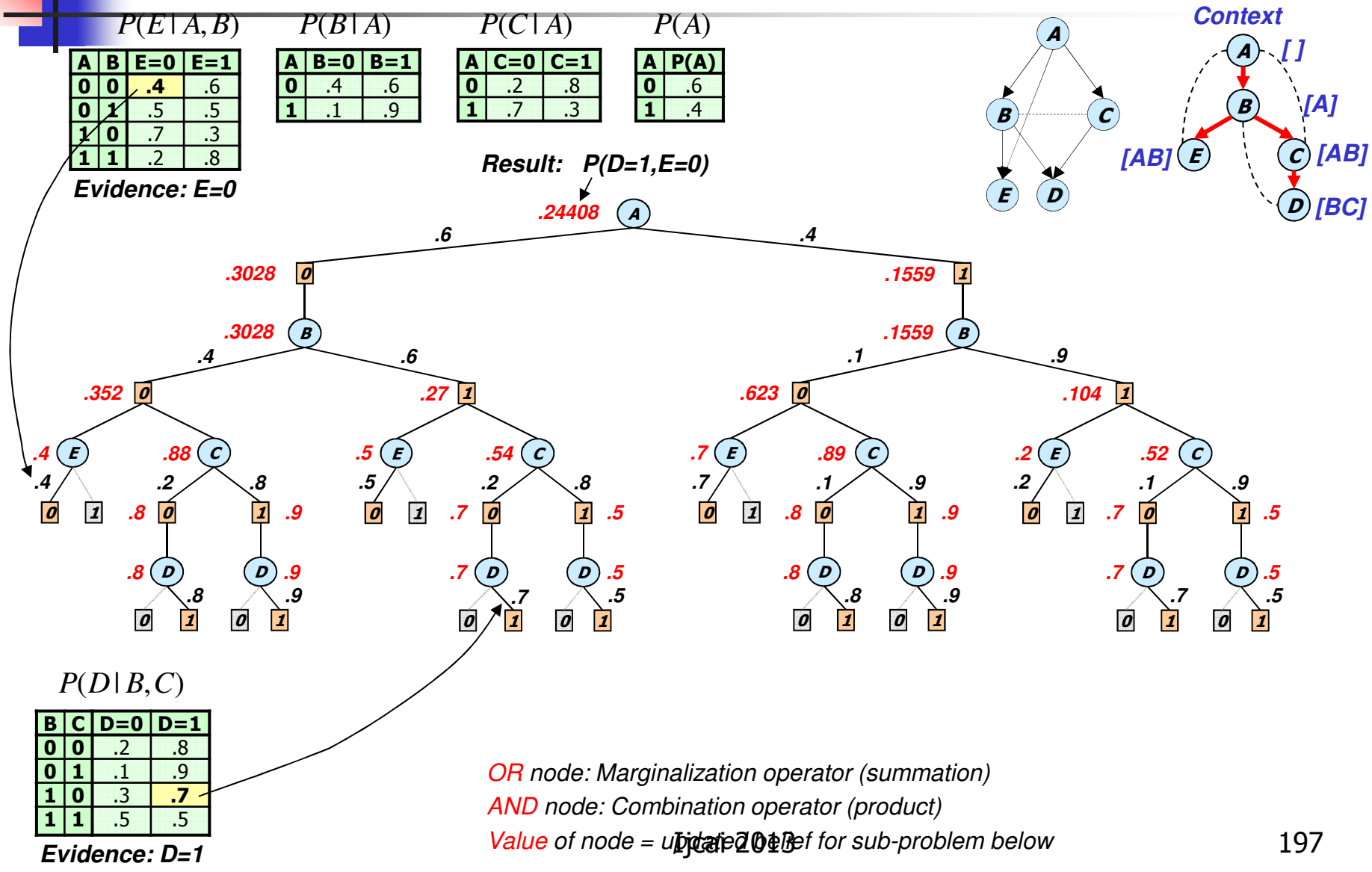
OR node: Marginalization operator (summation)

AND node: Combination operator (product)

Value of node = updated belief for sub-problem below

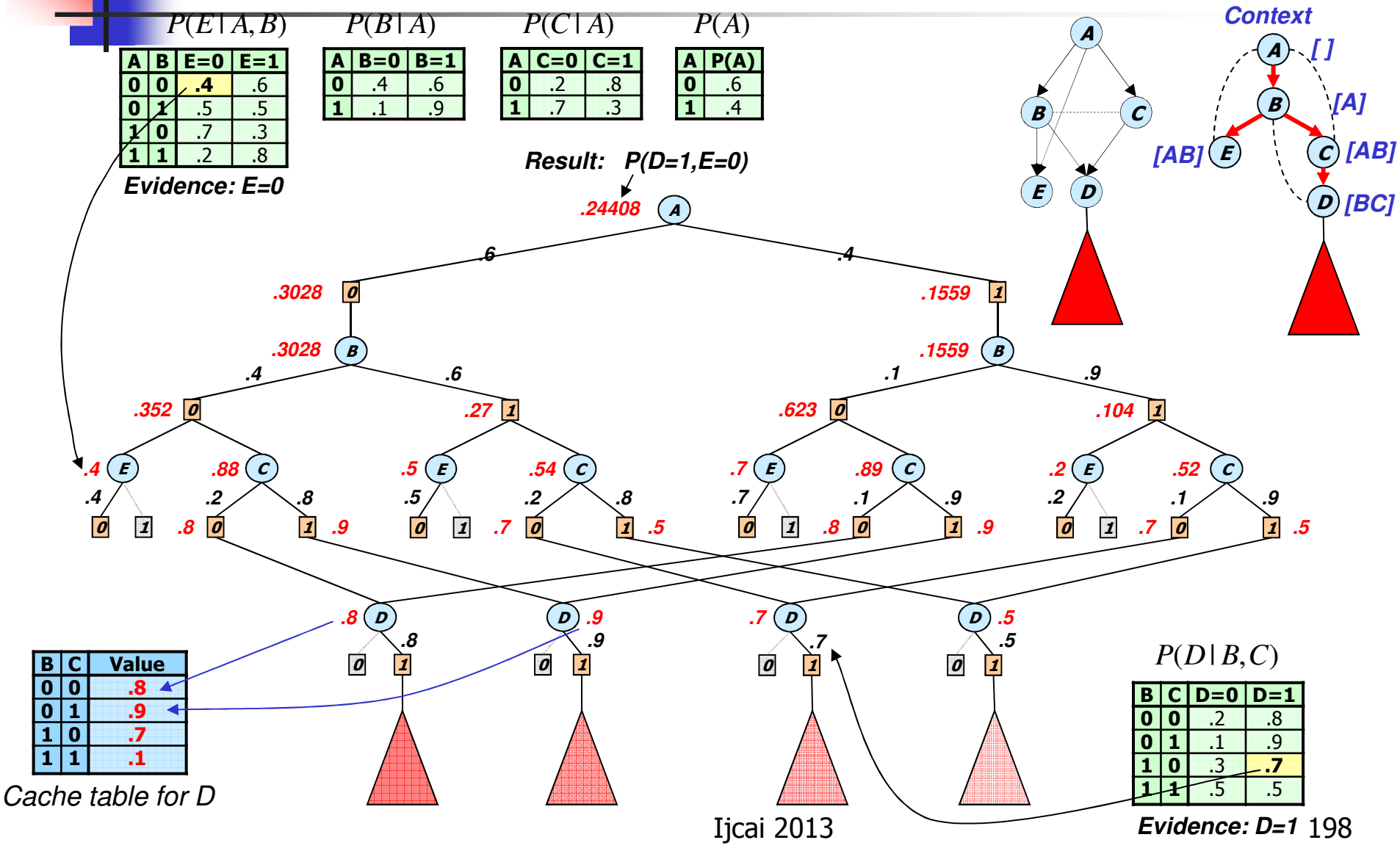
AND/OR Tree DFS Algorithm

(Belief Updating)



AND/OR Graph DFS Algorithm

(Belief Updating)

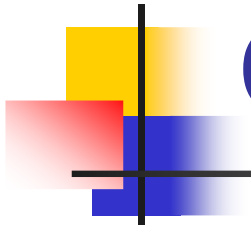


Complexity of AND/OR Graph Search

	AND/OR graph	OR graph
Space	$O(n\ d^{w^*})$	$O(n\ d^{pw^*})$
Time	$O(n\ d^{w^*})$	$O(n\ d^{pw^*})$

d = domain size
n = number of variables
w= treewidth*
pw= pathwidth*

$$w^* \leq pw^* \leq w^* \log n$$



Constructing Pseudo Trees

- AND/OR search algorithms are influenced by the quality of the pseudo tree
- Finding the minimal induced width / depth pseudo tree is NP-hard
- Heuristics
  - Min-Fill (min induced width)
  - Hypergraph partitioning (min depth)

Quality of the Pseudo Trees

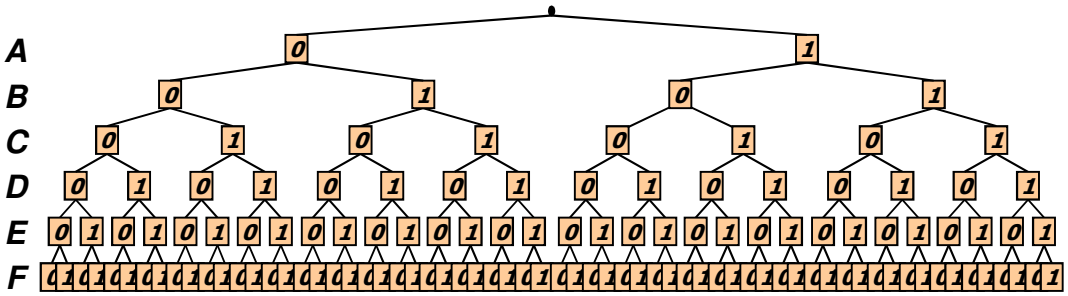
Network	**hypergraph**		**min-fill**	
	width	depth	width	depth
barley	7	**13**	7	23
diabetes	7	**16**	4	77
link	21	**40**	15	53
mildew	5	**9**	4	13
munin1	12	**17**	12	29
munin2	9	**16**	9	32
munin3	9	**15**	9	30
munin4	9	**18**	9	30
water	11	**16**	10	15
pigs	11	**20**	11	26

Bayesian Networks Repository

Network	**hypergraph**		**min-fill**	
	width	depth	width	depth
spot5	47	152	**39**	204
spot28	108	138	**79**	199
spot29	16	23	**14**	42
spot42	36	48	**33**	87
spot54	12	16	**11**	33
spot404	19	26	**19**	42
spot408	47	52	**35**	97
spot503	11	20	**9**	39
spot505	29	42	**23**	74
spot507	70	122	**59**	160

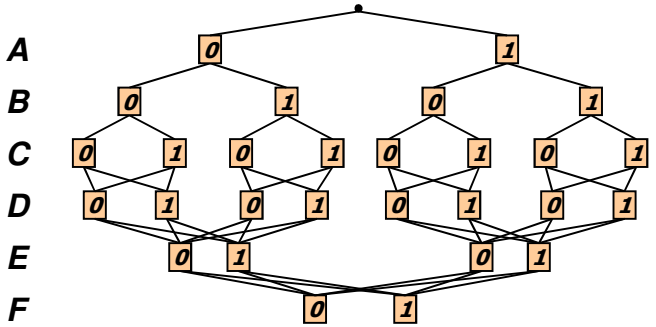
SPOT5 Benchmarks

All Four Search Spaces



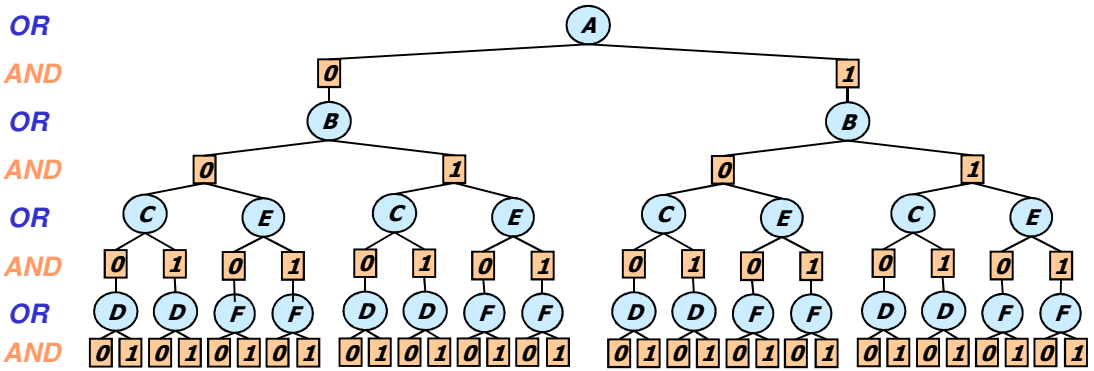
Full OR search tree

126 nodes



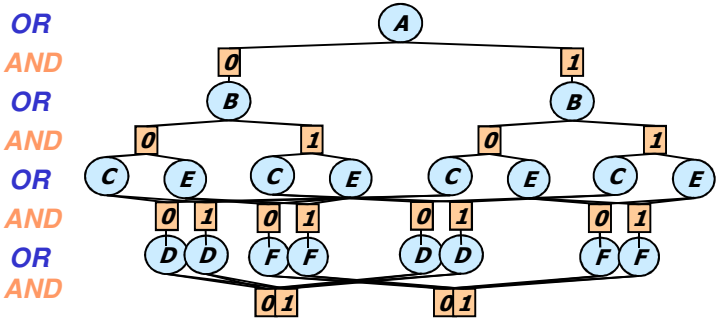
Context minimal OR search graph

28 nodes



Full AND/OR search tree

54 AND nodes

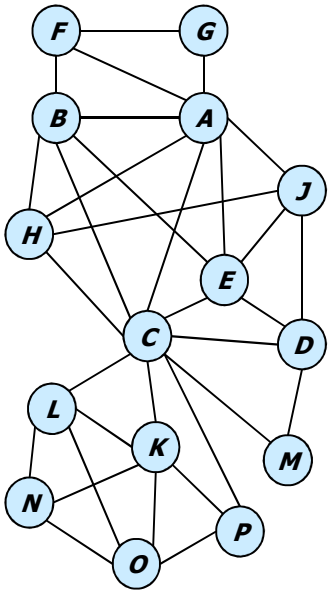


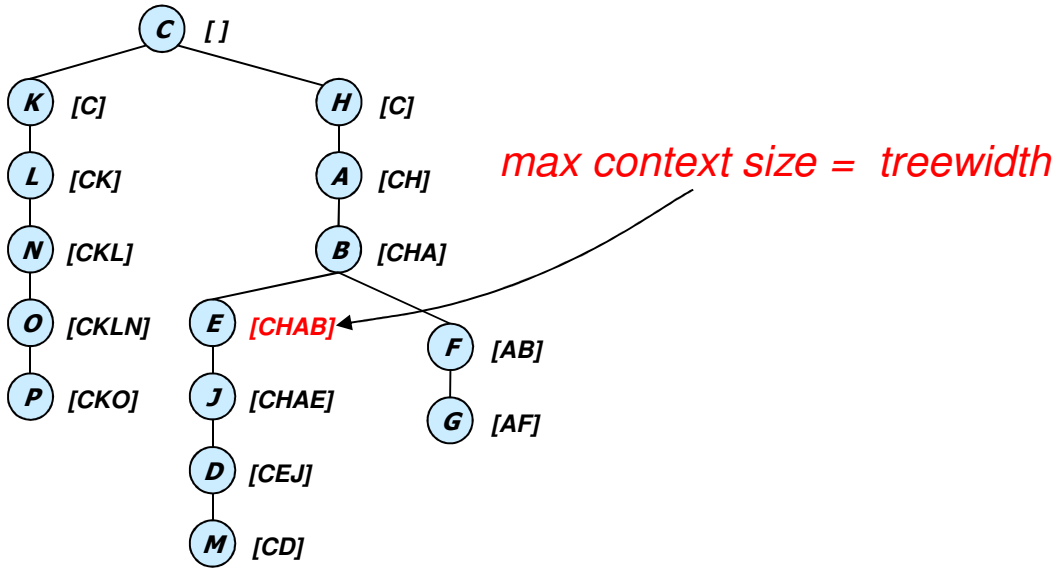
Context minimal AND/OR search graph

18 AND nodes

How Big Is The Context?

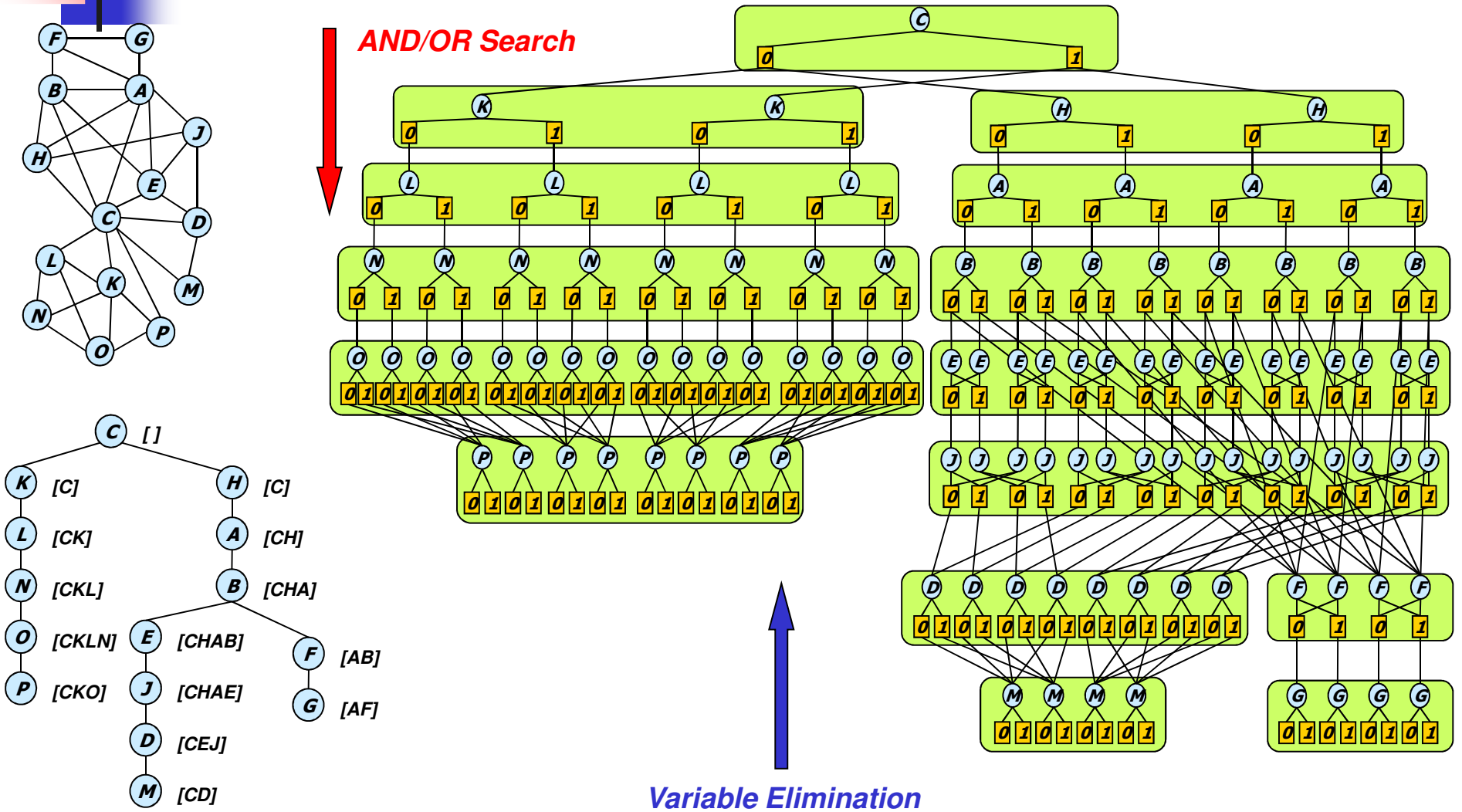
Theorem: The maximum context size for a pseudo tree is equal to the treewidth of the graph along the pseudo tree.





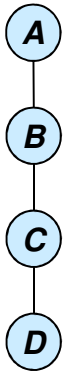
(C K H A B E J L N O D P M F G)

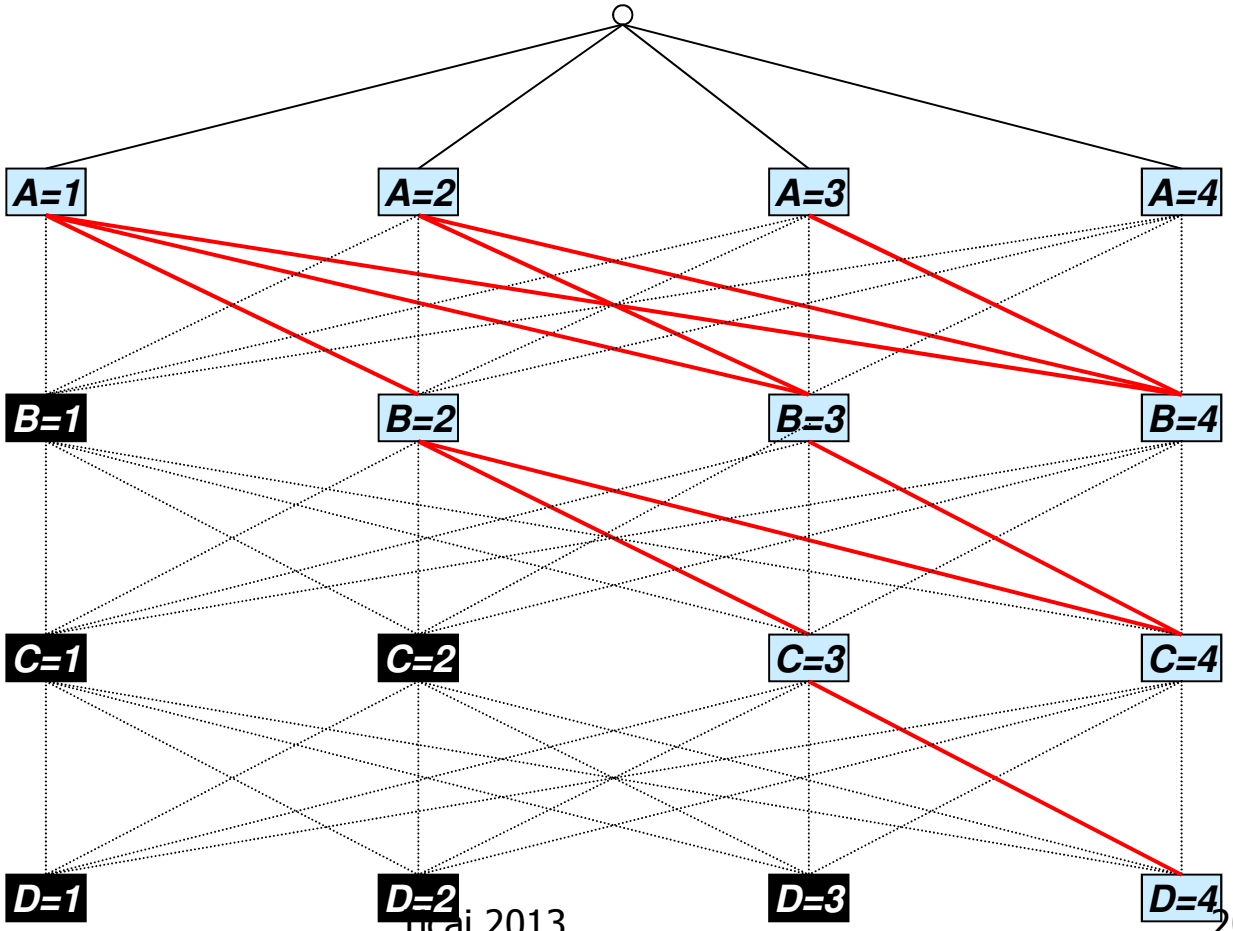
AND/OR Context Minimal Graph



AO vs. VE with determinism

Space traversed by AO



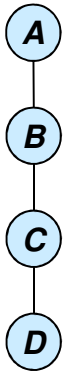


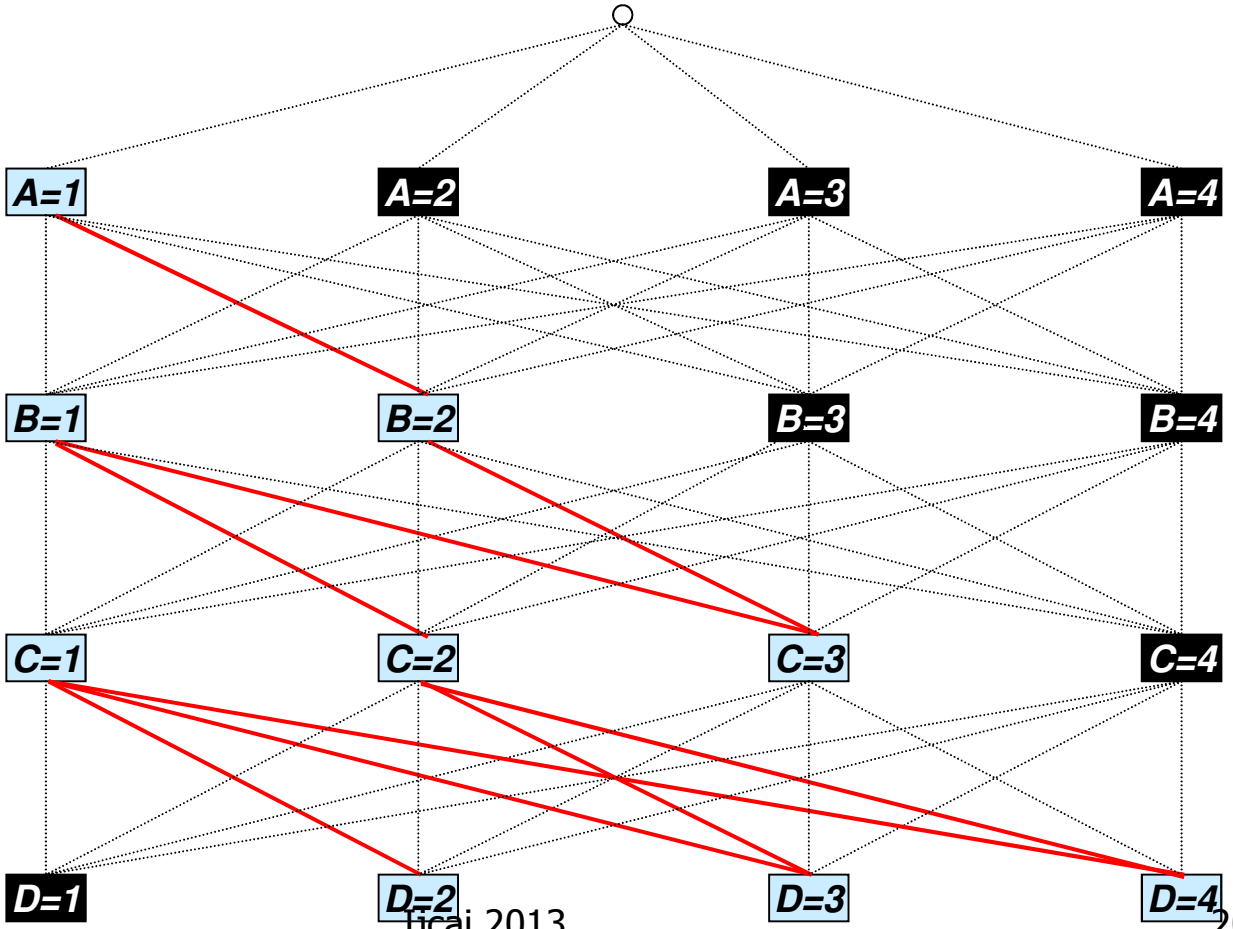
Domains: {1,2,3,4}

A < B; B < C; C < D

AO vs. VE with determinism

Space traversed by VE

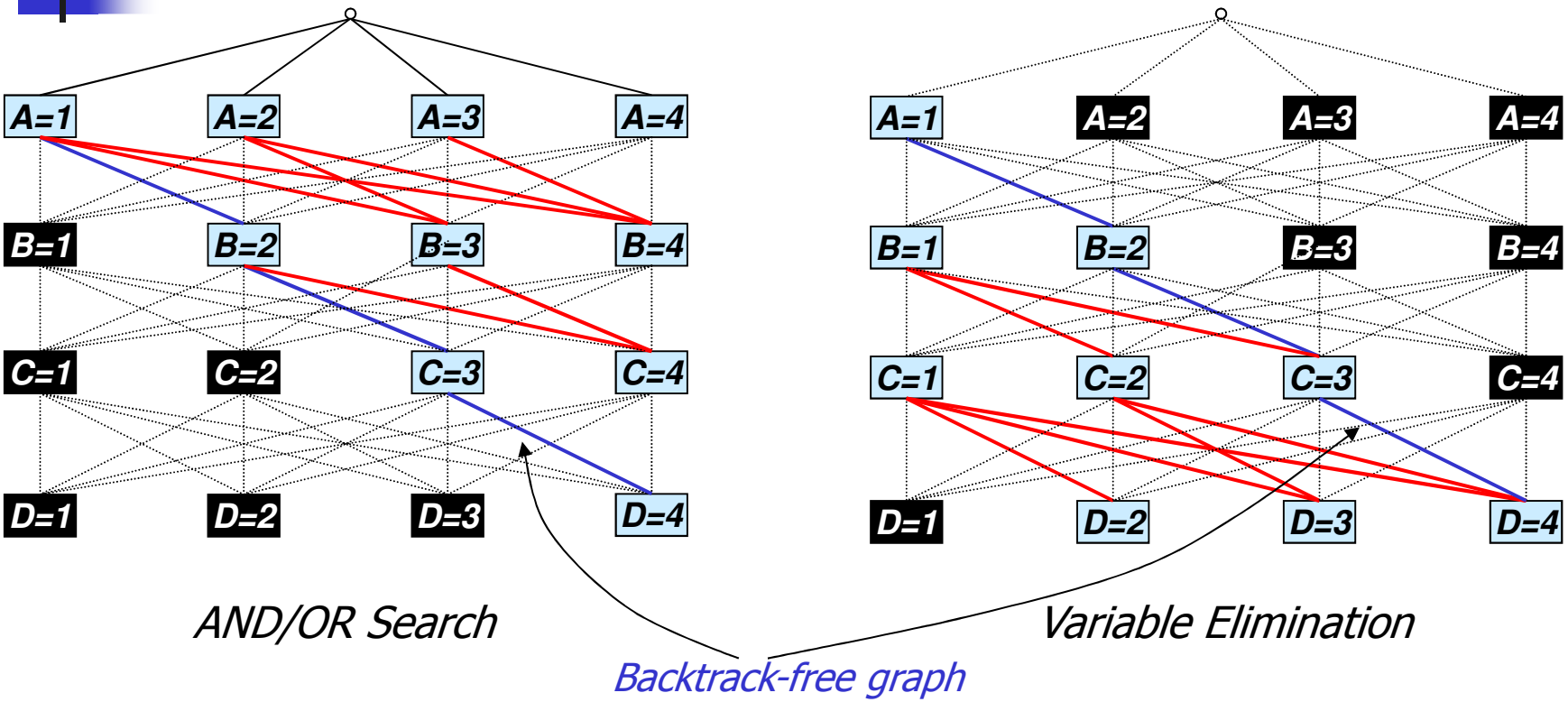




Domains: {1,2,3,4}

A < B; B < C; C < D

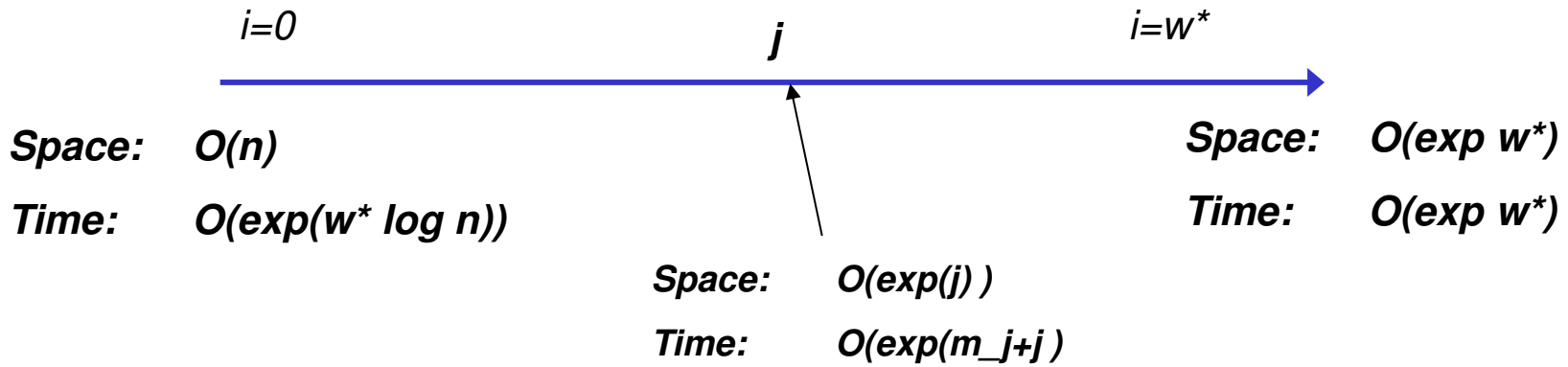
AO vs. VE with determinism



THEOREM Given a graphical model with determinism, AO and VE explore different portions of the full context-minimal graph which are not comparable.

Searching AND/OR Graphs

- AO(j): searches depth-first, cache i-context
  - j = the max size of a cache table (i.e. number of variables in a context)



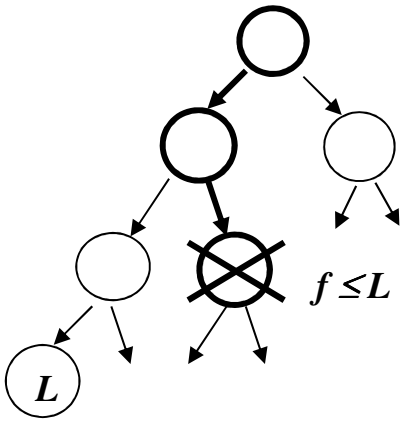
Optimization: Searching the AND/OR space for MPE/MAP

Heuristic function $f(x^p)$ *computes a lower bound on the best extension of* x^p *and can be used to guide a heuristic search algorithm. We focus on:*

1. DF Branch-and-Bound

Use heuristic function $f(x^p)$ *to prune the depth-first search tree*

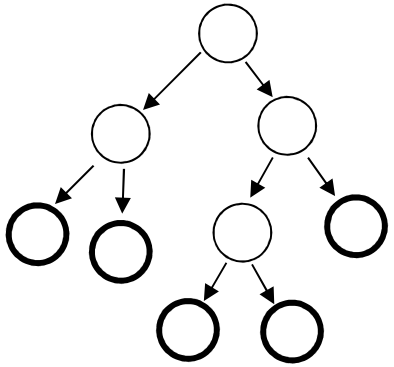
Linear space



2. Best-First Search

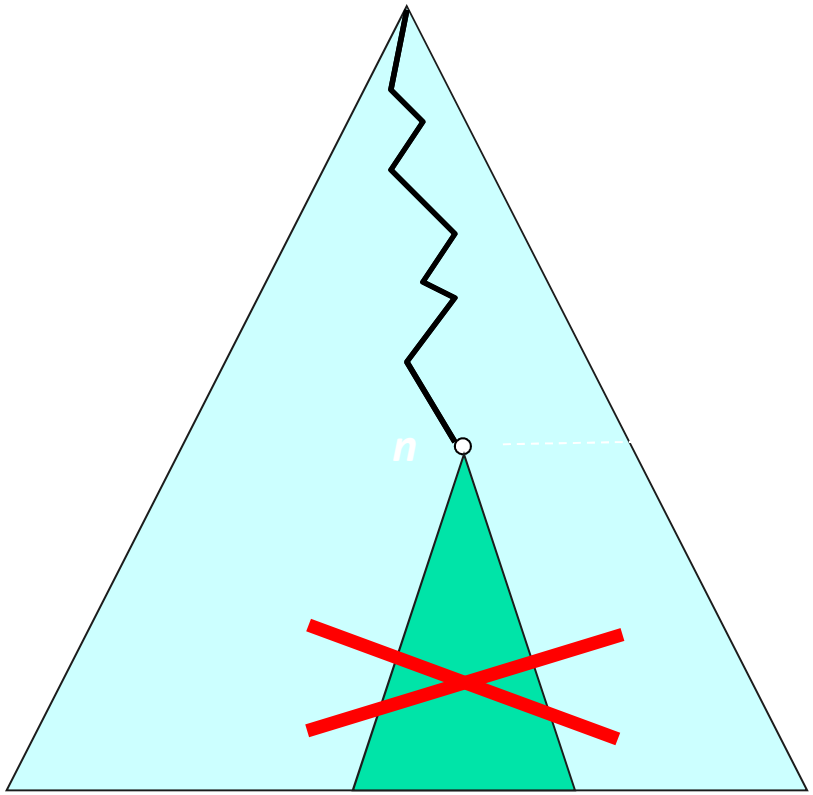
Always expand the node with the highest heuristic value $f(x^p)$

Needs lots of memory



Leads to AOBB with MBE heuristics

Branch-and-Bound Search



OR Search Tree

Upper Bound ***UB***

Lower Bound ***LB(n)***

*g(n)=****cost of the search path to n***

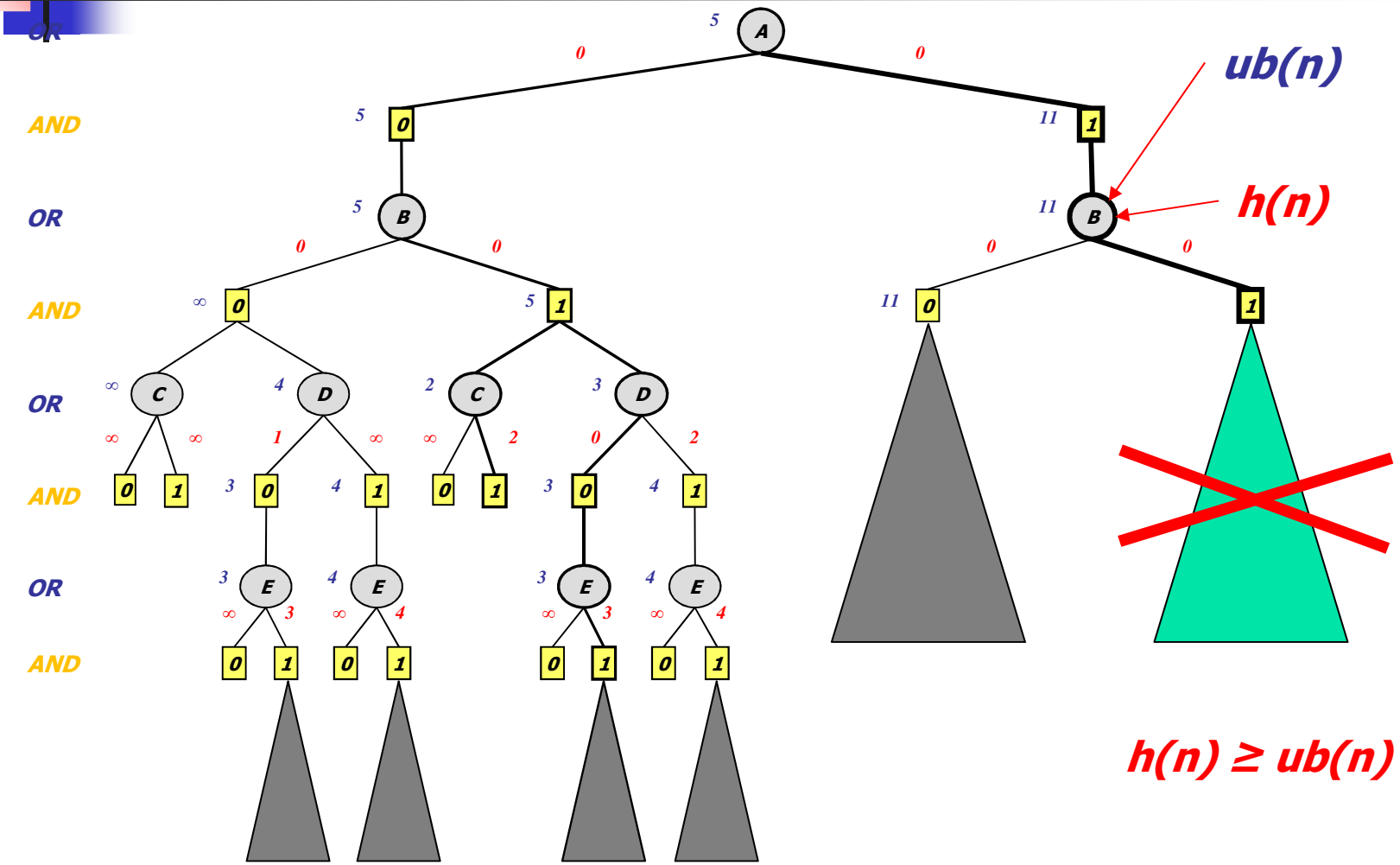
Prune if LB(n) ≥ UB

H(n) = ***estimates the optimal cost below n***

(Lawler & Wood66)

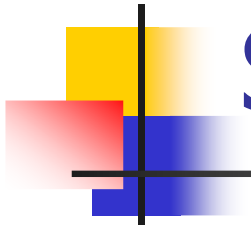
AND/OR Branch-and-Bound

Kask and Dechter 2001
Marinescue and Dechter, 2005-2009



PASCAL 2011 Probabilistic Inference Challenge

- http://www.cs.huji.ac.il/project/PASCAL/
- Evaluates solvers in three categories:
  - PR: Probability of evidence / partition function
  - MAR: Posterior node marginals
  - MPE: Most probable explanation (our entry)
- Three tracks each: 20 sec, 20 min, 1 hour.
- Variety of benchmark domains:
  - CSPs, Deep Belief Nets, Image Alignment

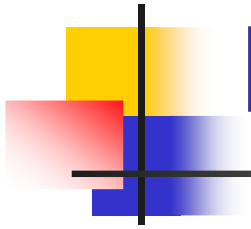


Software

- AND/OR search algorithms
- Bucket-tree elimination
- Generalized belief propagation
- Samplesearch sampling

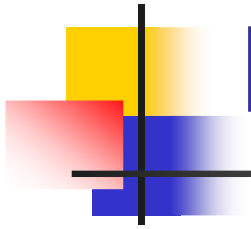
are available at:

- AOBB source code is freely available, under an open-source license.
  - http://graphmod.ics.uci.edu/group/Software



Road Map:Bayesian Networks

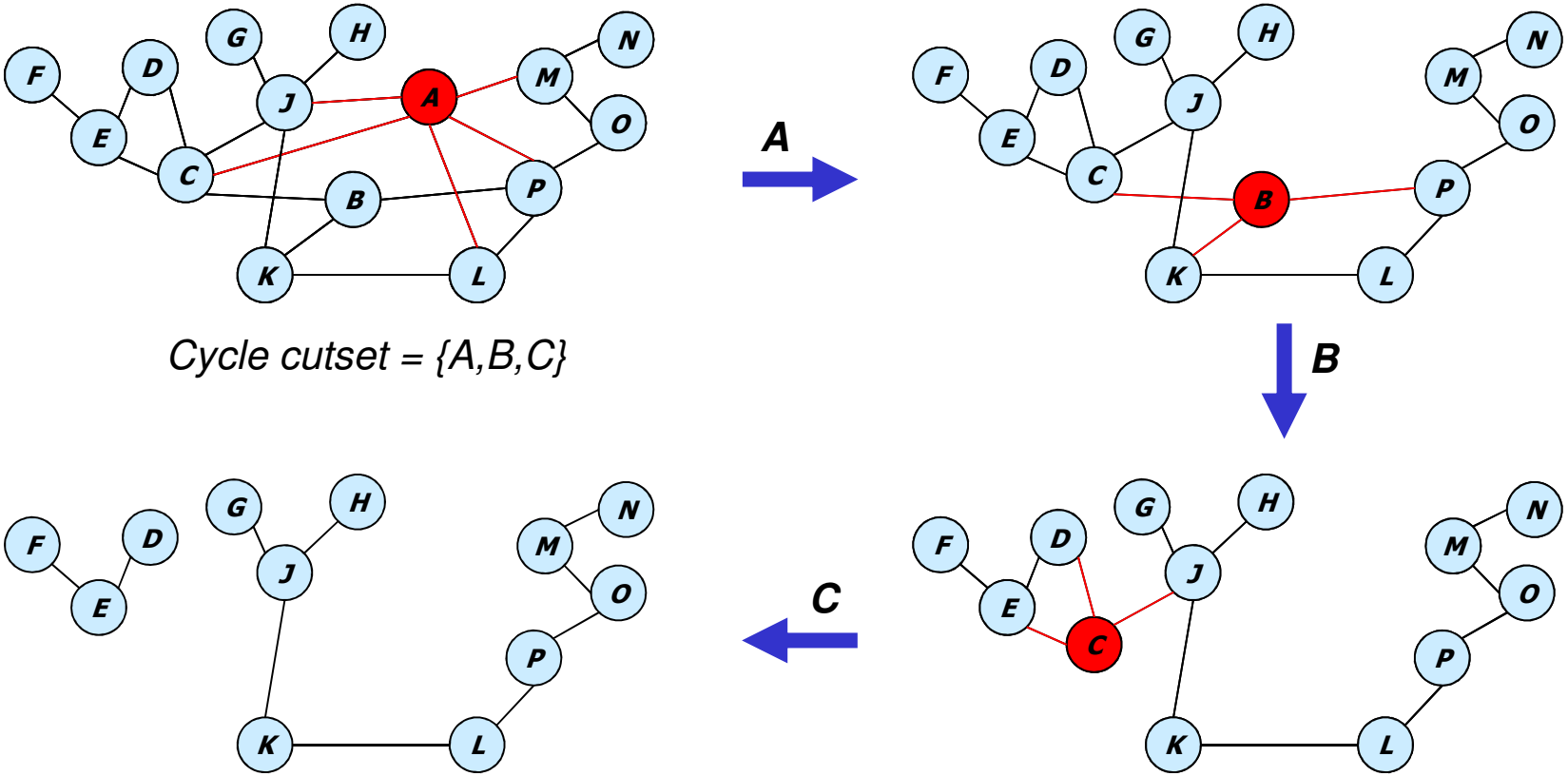
- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
  - Belied propagation
  - Mini-bucket, mini-clustering
  - Iterative join-graph propagation: a GBP scheme
- Search, conditioning
- Hybrid of Search and Inference (skiped)
- Reasoning with mixed networks



Road Map:Bayesian Networks

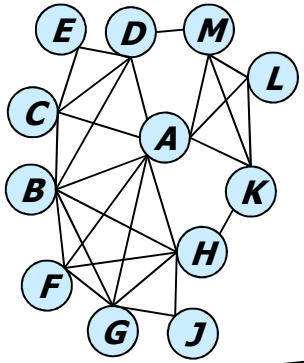
- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
  - Belied propagation
  - Mini-bucket, mini-clustering
  - Iterative join-graph propagation: a GBP scheme
- Search, conditioning
- Hybrid of Search and Inference (skiped)
- Reasoning with mixed networks

Conditioning and Cycle cutset

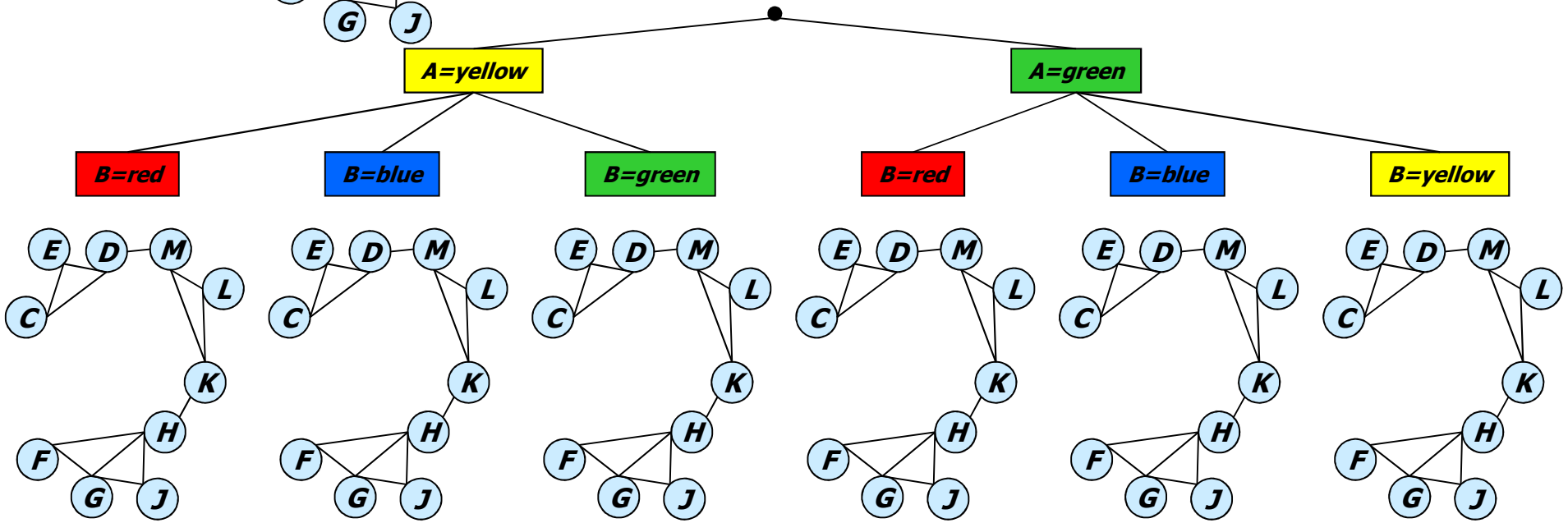


Search over the Cutset (cont)

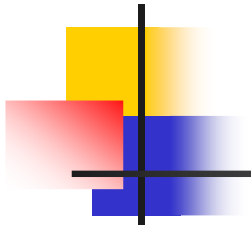
Graph Coloring problem



- *Inference may require too much memory*
- ***Condition** on some of the variables*



w-cutset algorithms

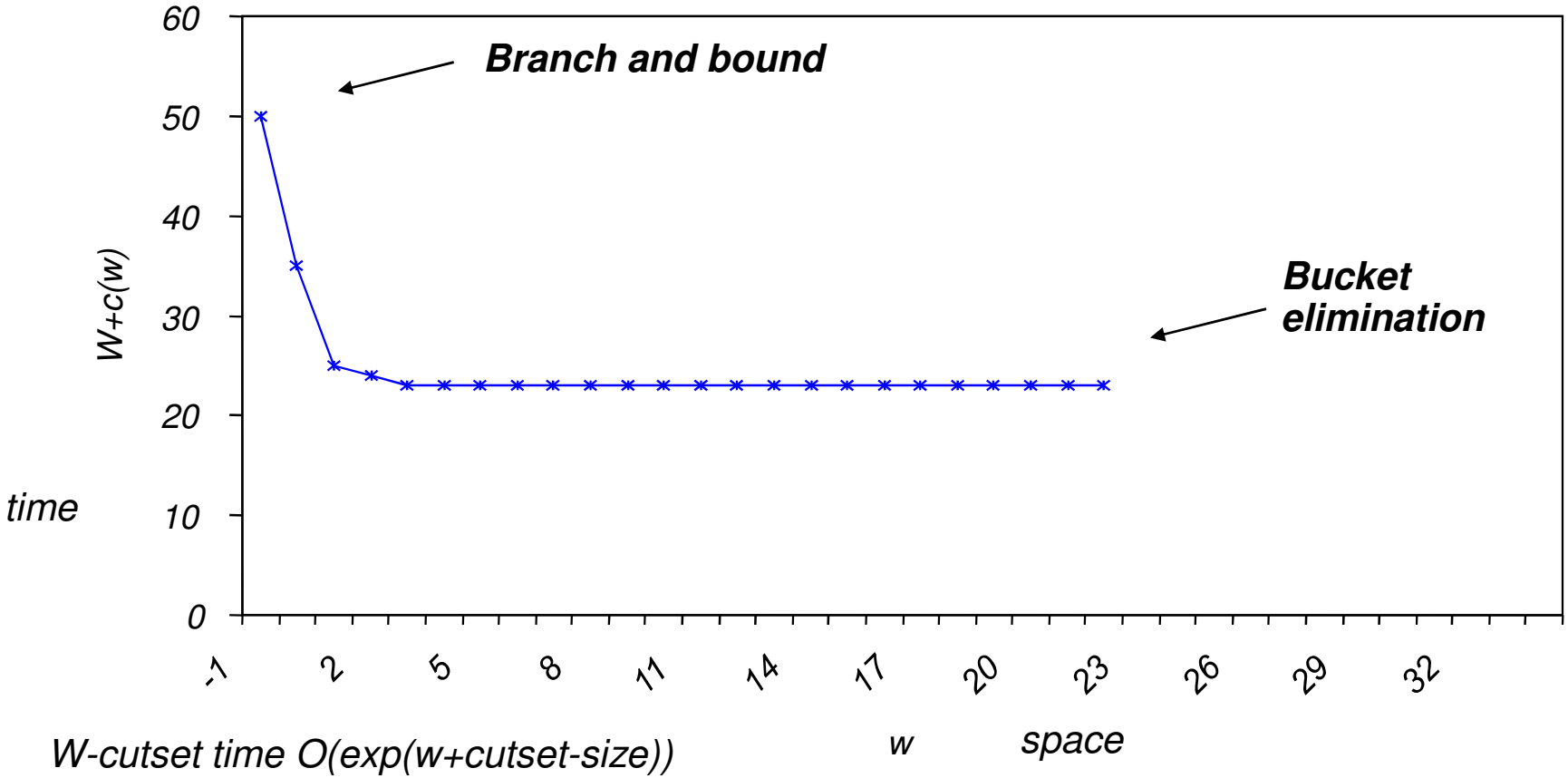


- Identify an w-cutset c_w of the network
- For each assignment to the cutset c_w solve the conditioned sub-problem by *CTE*
- Aggregate the solutions over all assignments.
- Time complexity: $O(k^{c_w+w})$
- Space complexity: $O(k^{\omega})$
- What w should we use?

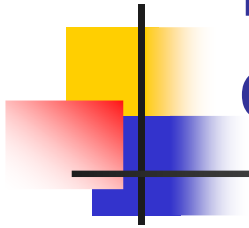
Time vs Space for w-cutset

(Dechter and El-Fatah, 2000)
(Larrosa and Dechter, 2001)
(Rish and Dechter 2000)

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



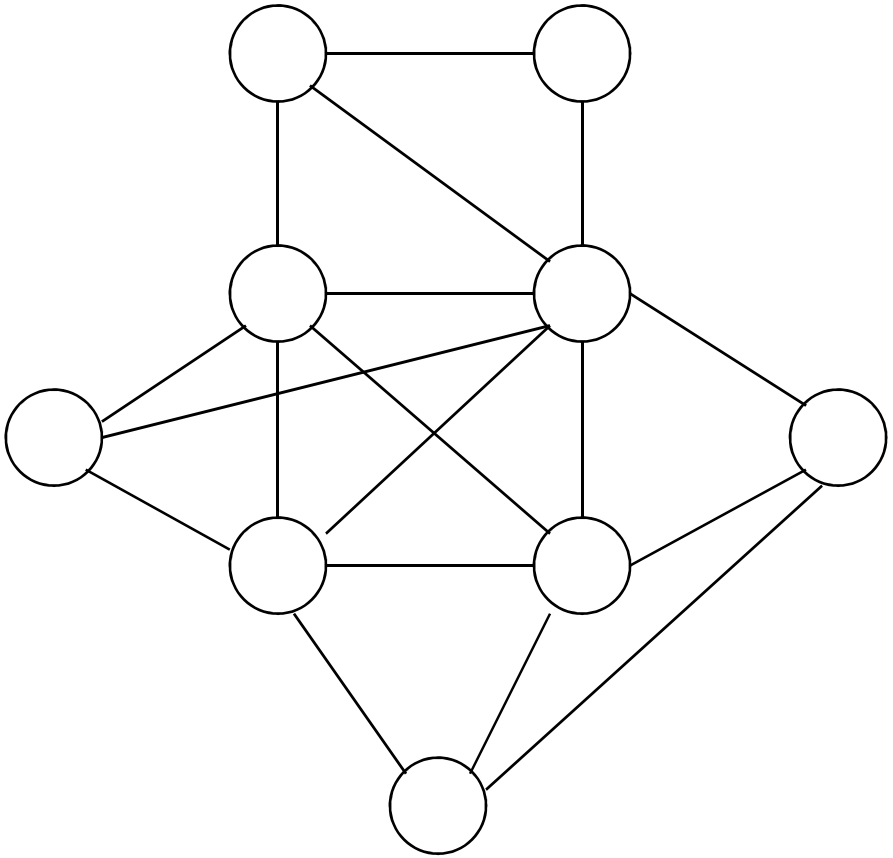
W-cutset time O(exp(w+cutset-size))
Space O(exp(w))



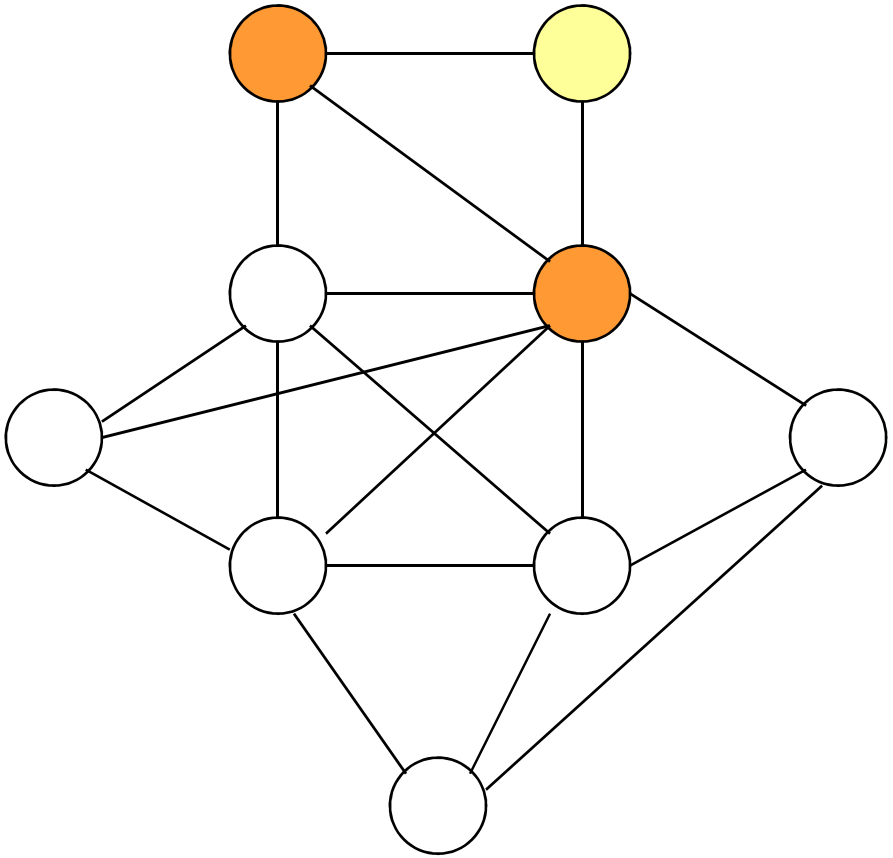
Hybrid of Variable-elimination and conditioning-Search

- Tradeoff space and time

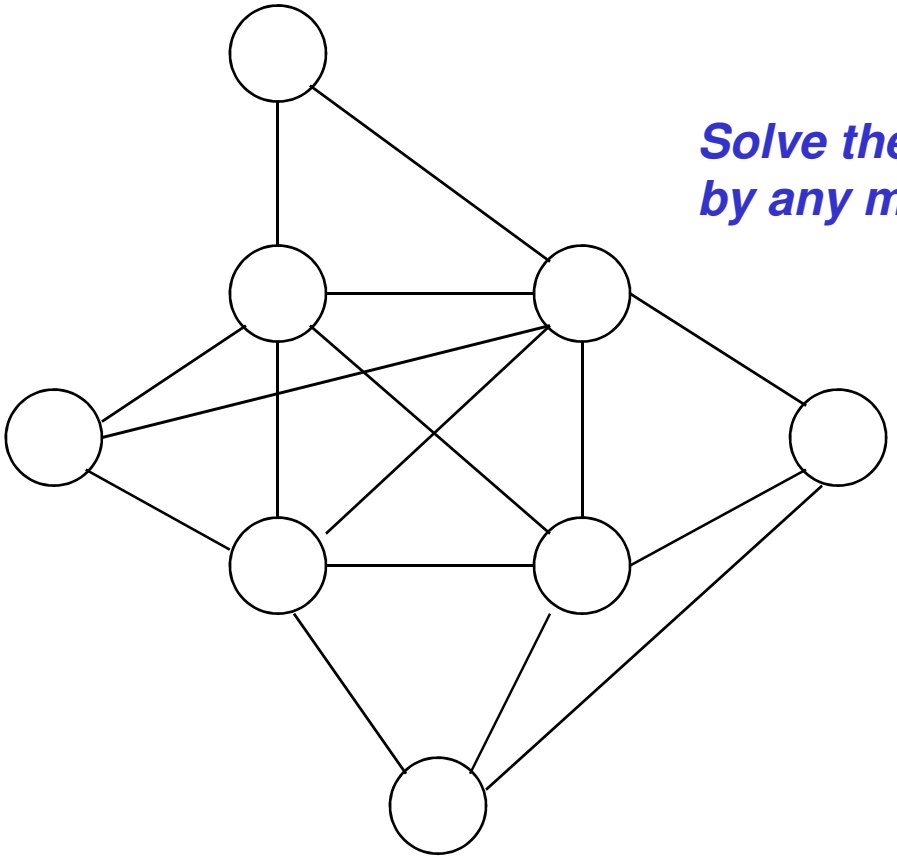
Eliminate First



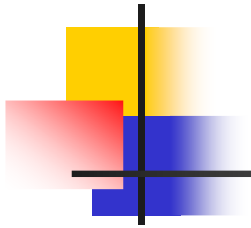
Eliminate First



Eliminate First



Solve the rest of the problem by any means

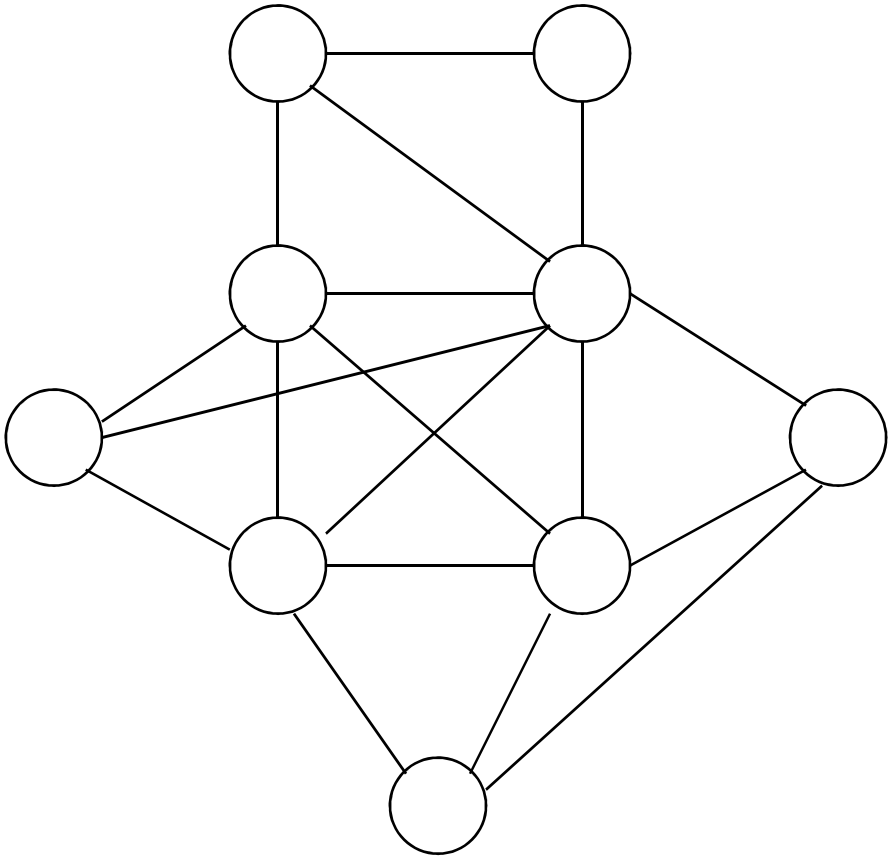


Hybrids Variants

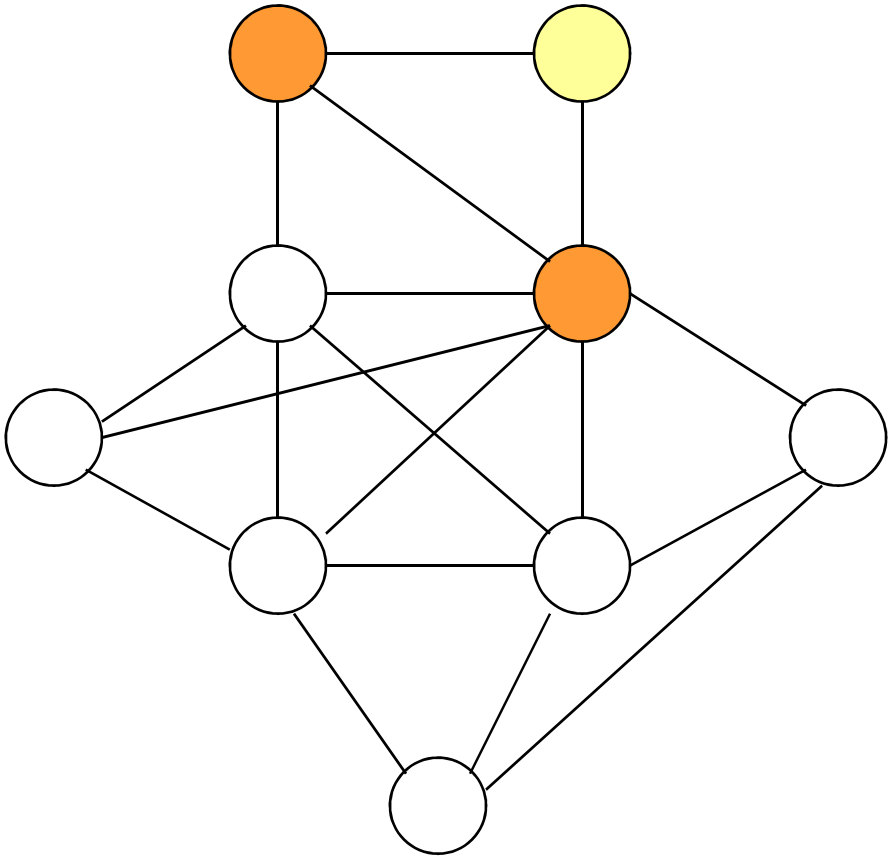
- Condition, condition, condition … and then only eliminate (w-cutset, cycle-cutset)
- Eliminate, eliminate, eliminate … and then only search
- Interleave conditioning and elimination (elim-cond(i), VE+C)

Interleaving Conditioning and Elimination

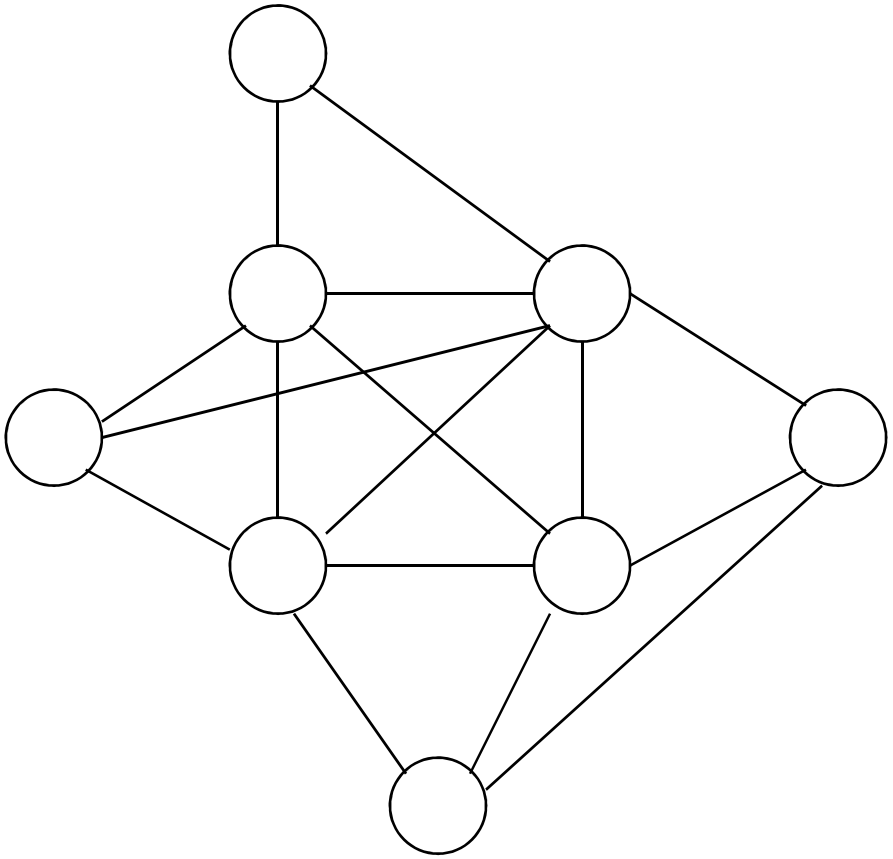
(Larrosa & Dechter, CP'02)



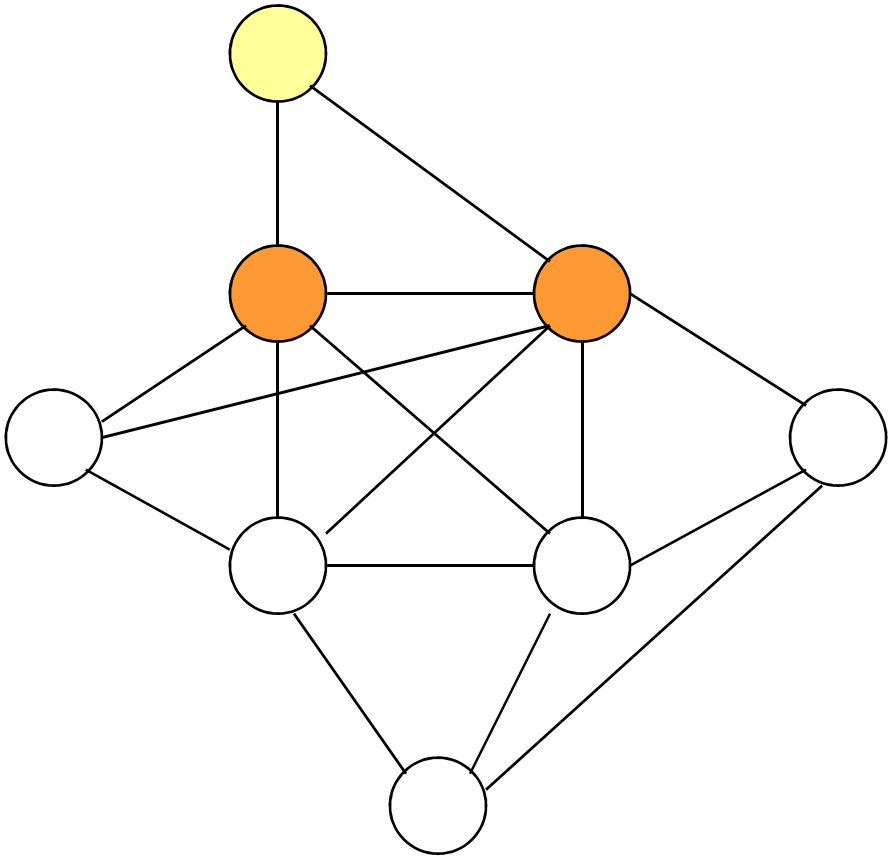
Interleaving Conditioning and Elimination



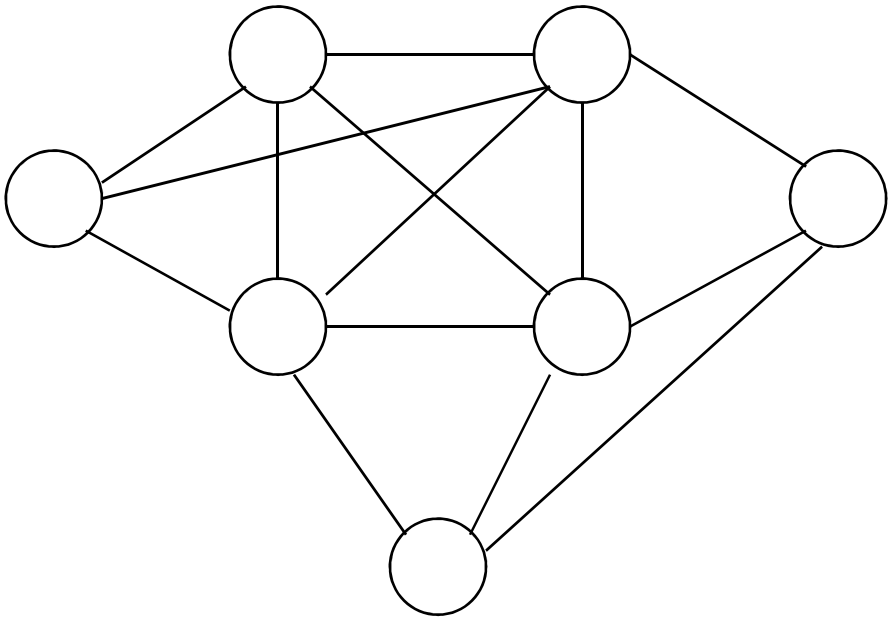
Interleaving Conditioning and Elimination



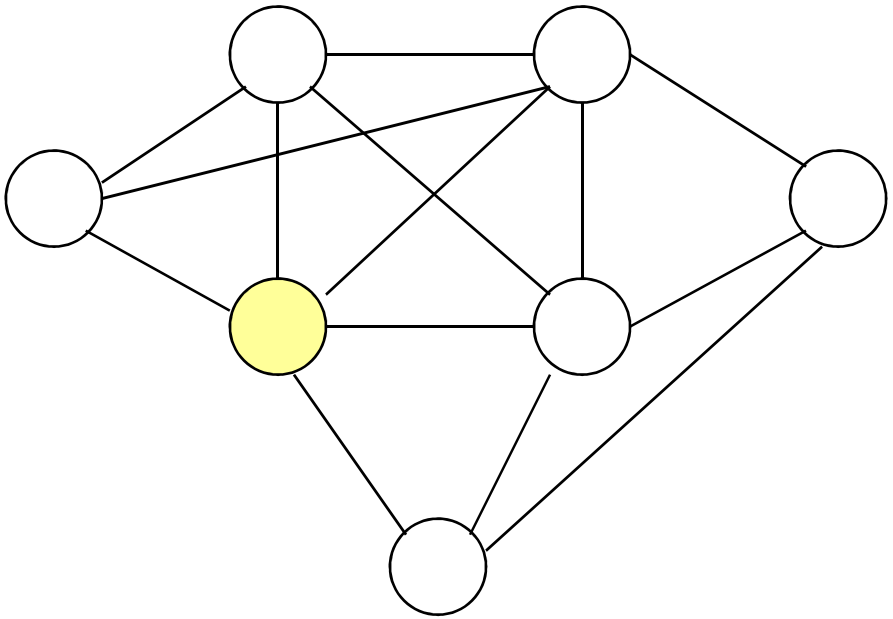
Interleaving Conditioning and Elimination



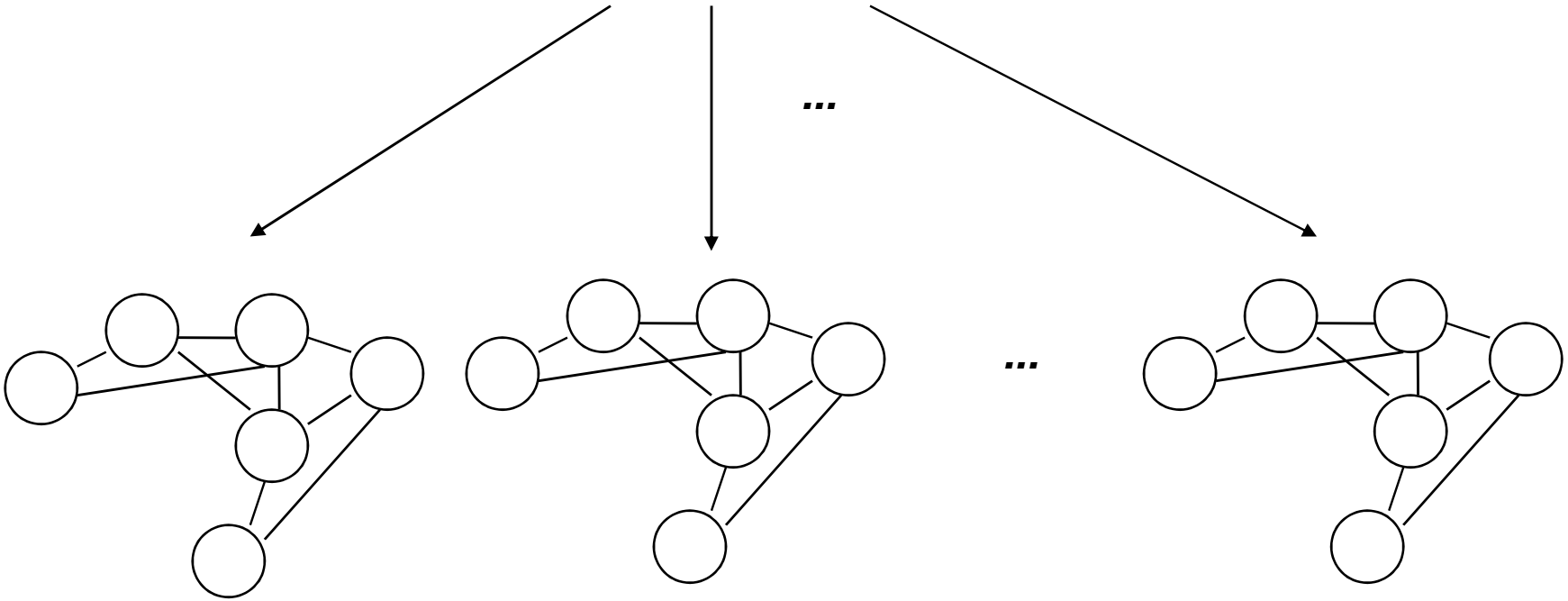
Interleaving Conditioning and Elimination

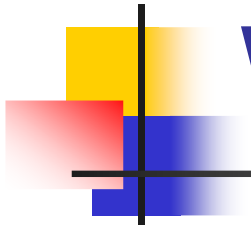


Interleaving Conditioning and Elimination



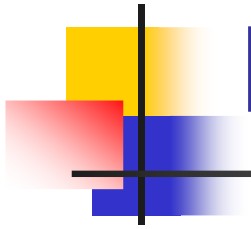
Interleaving Conditioning and Elimination





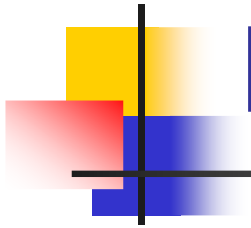
What hybrid should we use?

- w=1? (loop-cutset?)
- w=0? (Full search?)
- w=w* (Full inference)?
- w in between?
- depends… on the graph
- What is relation between cycle-cutset and the induced-width?



Road Map:Bayesian Networks

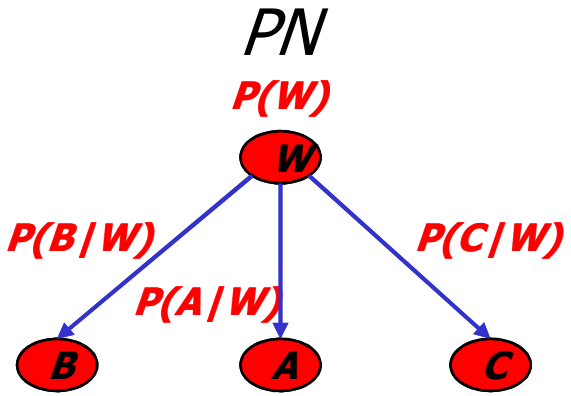
- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
- Search, conditioning
- Hybrid of Search and Inference
- Reasoning with mixed networks
  - Inference
  - Search
  - Sampling solutions

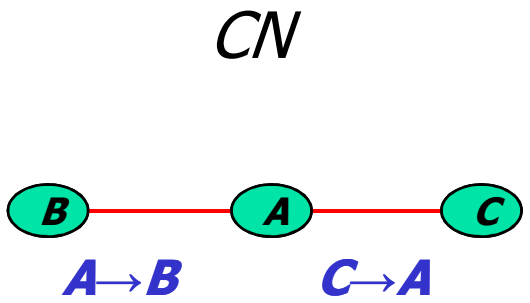


Road Map:Bayesian Networks

- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
- Search, conditioning
- Hybrid of Search and Inference
- Reasoning with mixed networks
  - Inference
  - Search
  - Sampling solutions

Mixed Probabilistic and Deterministic networks





Semantics?

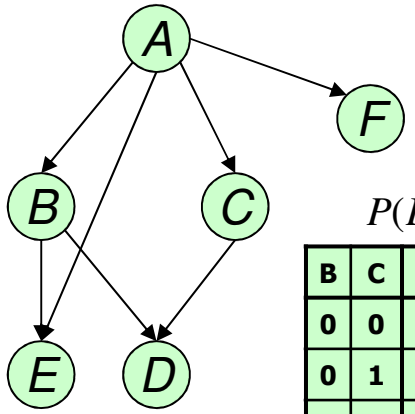
Algorithms?

Query:
Is it likely that Chris goes to the party if Becky does not but the weather is bad?

$$P(C, \neg B \mid w = bad, A \rightarrow B, C \rightarrow A)$$

Uncertainty and Determinism

Belief Network (B)



$P(D \mid B, C)$

B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

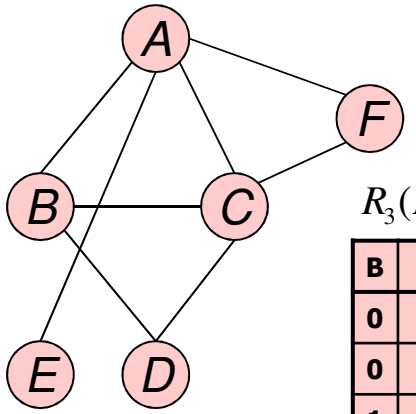
Variables: A, B, C, D, E, F

Domains: $D_A = D_B = D_C = D_D = D_E = D_F = \{0,1\}$

CPTS: $P(A), P(B \mid A), P(C \mid A), P(D \mid B, C)$

$P(E \mid A, B), P(F \mid A)$

Constraint Network (R)



$R_3(BCD)$

B	C	D
0	0	1
0	1	0
1	1	0

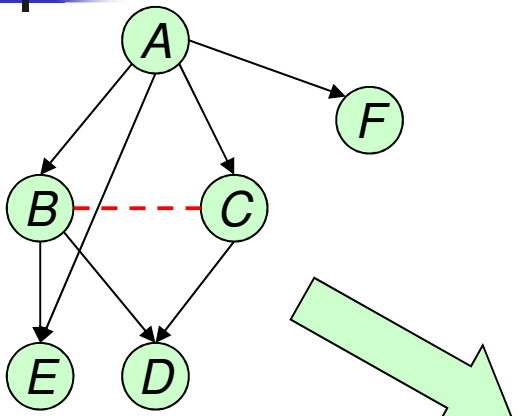
Variables: A, B, C, D, E, F

Domains: $D_A = D_B = D_C = D_D = D_E = D_F = \{0,1\}$

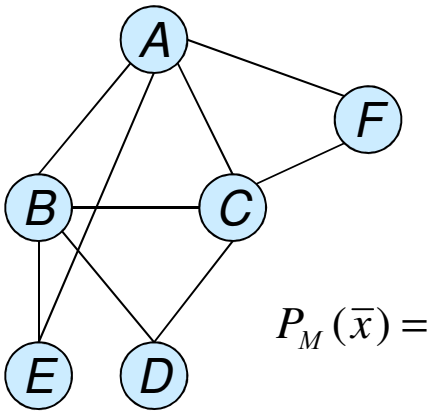
Relations: $R_1(ABC), R_2(ACF), R_3(BCD), R_4(A, E)$

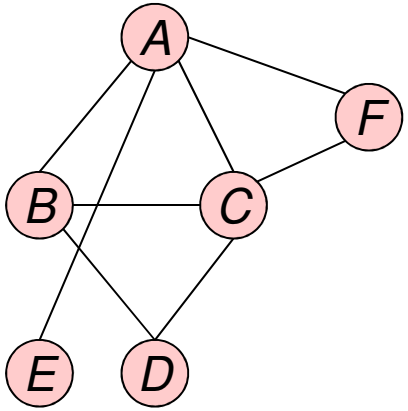
Expresses the set of solutions: $\rho = R(ABCDEF)$

Mixed Networks



Moral mixed graph

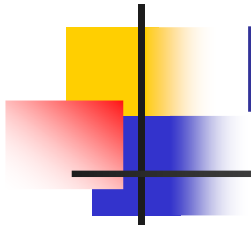




$$P_M(\bar{x}) = \begin{cases} P_B(\bar{x} \mid \bar{x} \in \rho) = \dfrac{P_B(\bar{x})}{P_B(\bar{x} \in \rho)}, \text{ if } \bar{x} \in \rho \\ 0, \text{ otherwise} \end{cases}$$

Tasks for Mixed Networks

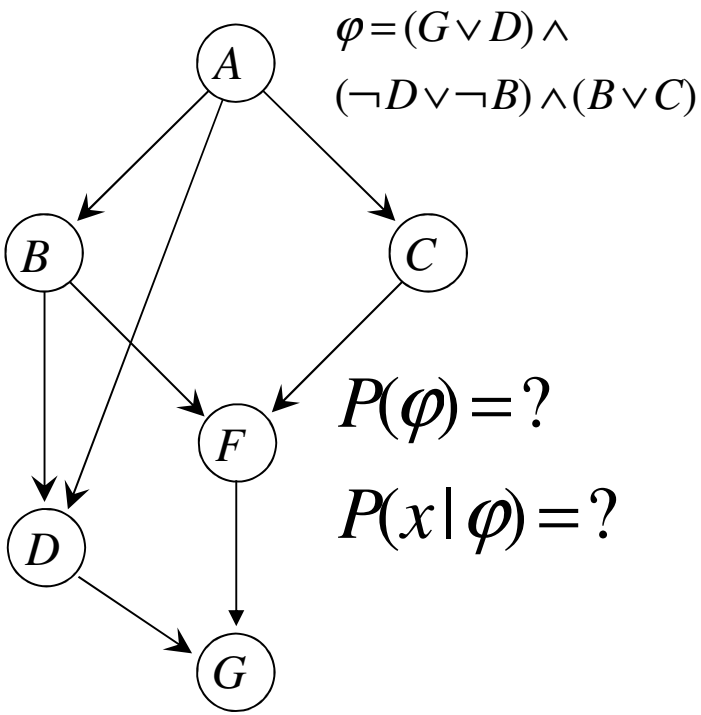
- Given M= (B,R):
  - *Belief updating: Find the liklihood of X given e and assuming consistency : (P(x|R,e)= ?)*
  - *MPE: find probability of most likely solution of R*
  - *MAP: Find the probability of most likely consistent assignment to a subset of variables*
  - *Constraint Probability Evaluation (CPE)*: Find the probability that an assignment is consistent,



Road Map:Bayesian Networks

- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
- Search, conditioning
- Hybrid of Search and Inference
- Reasoning with mixed networks
  - Inference
  - Search
  - Sampling solutions

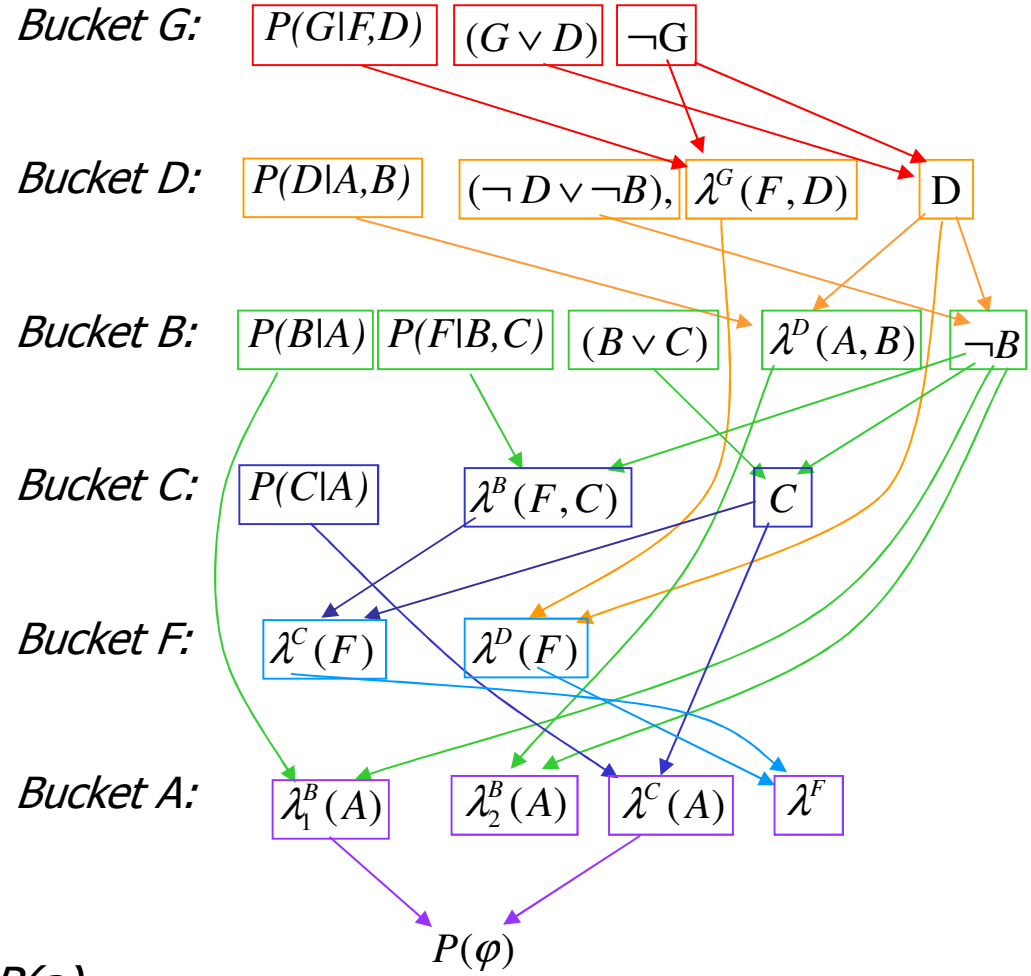
Bucket Elimination for Mixed Networks
Computing Probability of a cnf query



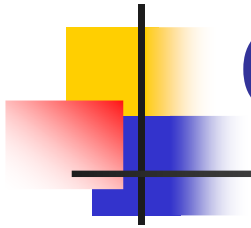
$\varphi = (G \vee D) \wedge (\neg D \vee \neg B) \wedge (B \vee C)$

$P(\varphi) = ?$

$P(x \mid \varphi) = ?$



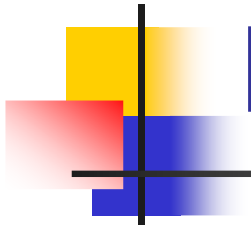
Belief network P(g,f,d,c,b,a)
=P(g|f,d)P(f|c,b)P(d|b,a)P(b|a)P(c|a)P(a)



Complexity

- Time and space exponential in the induced-width of the mixed graph, whose evidence node and unit literals are removed.

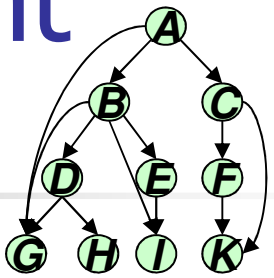
- Apply constraint propagation to the constraint portion and then solve the mixed network.

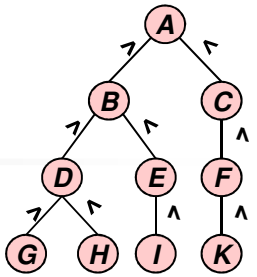


Road Map:Bayesian Networks

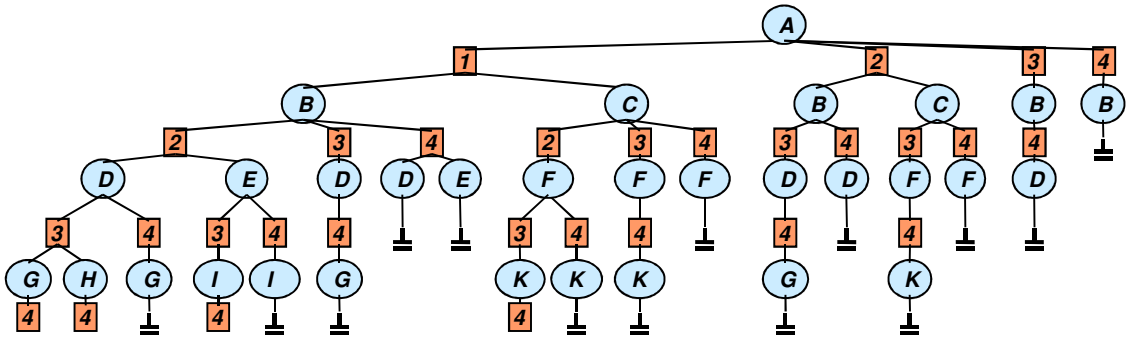
- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
- Search, conditioning
- Hybrid of Search and Inference
- Reasoning with mixed networks
  - Inference
  - Search
  - Sampling solutions

The Effect of Constraint Propagation

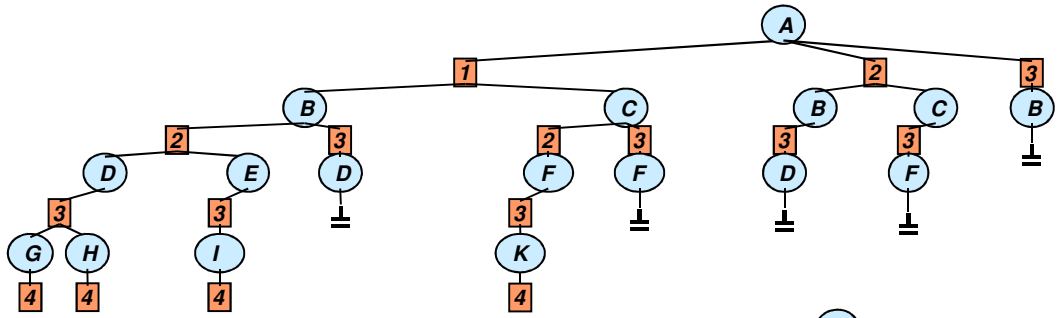




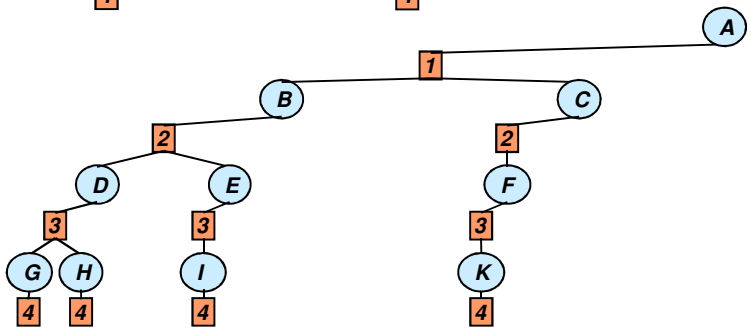
Domains are {1,2,3,4}



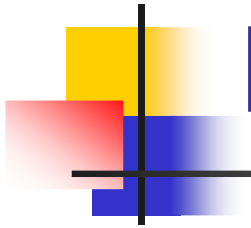
CONSTRAINTS ONLY



FORWARD CHECKING



MAINTAINING ARC CONSISTENCY

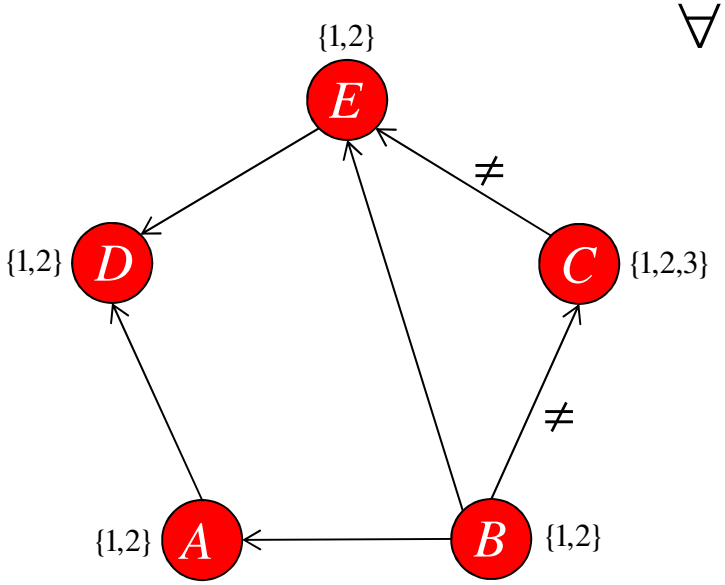


Road Map:Bayesian Networks

- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
- Search, conditioning
- Hybrid of Search and Inference
- Reasoning with mixed networks
  - Inference
  - Search
  - Sampling solutions

Generate Random Solutions

- *Motivation: generating tests for hardware verification*
- *Given a CSP, R = (X,D,C), generate solutions for R s.t. if ρ = sol(R):*



$$\forall\, t \in \rho, \mathrm{P}(t) = \frac{1}{|\rho|}$$

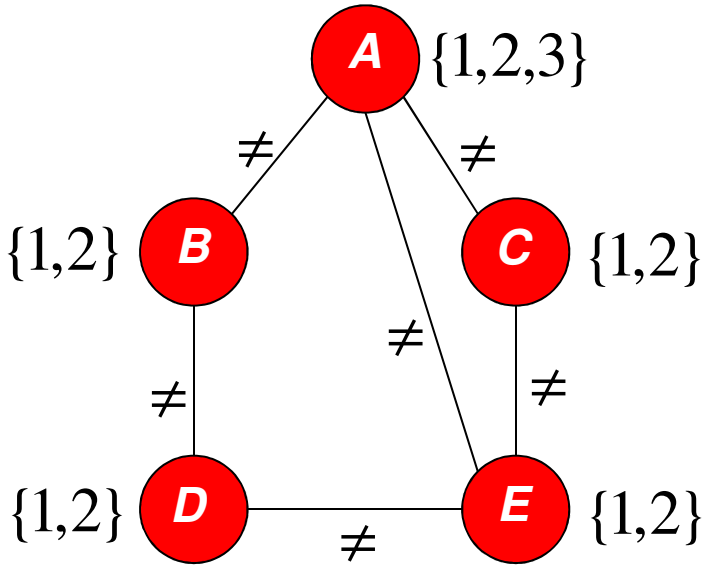
A	B	C	D	E	P
1	2	3	2	1	0.5
2	1	3	1	2	0.5

Brute-force: generate and list all solutions

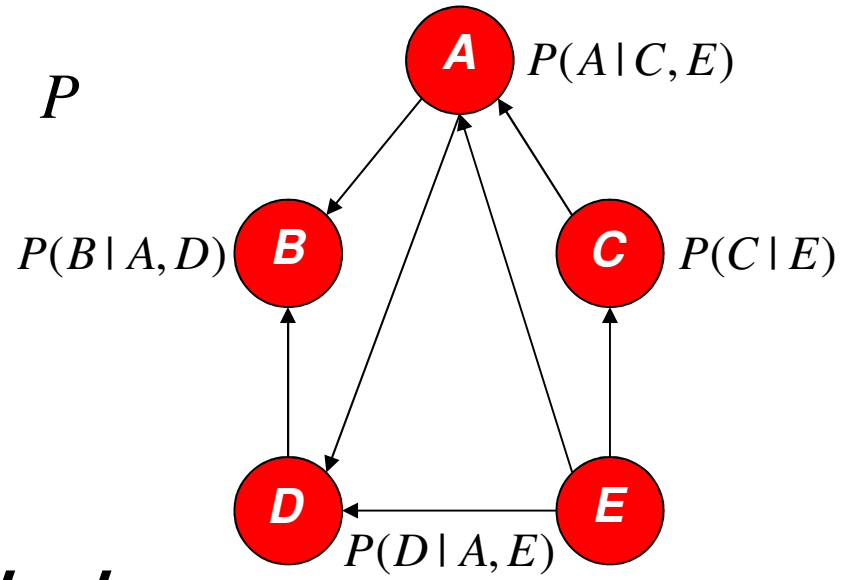
Modeling C*N* as B*N* on the same variables (Approach 2):

- Find a B*N* over same variables s.t.

$$P(x_1,\ldots,x_n) = \frac{1}{\#\text{sol}} \qquad if\ (x_1,\ldots,x_n)\ \text{is a solution}$$

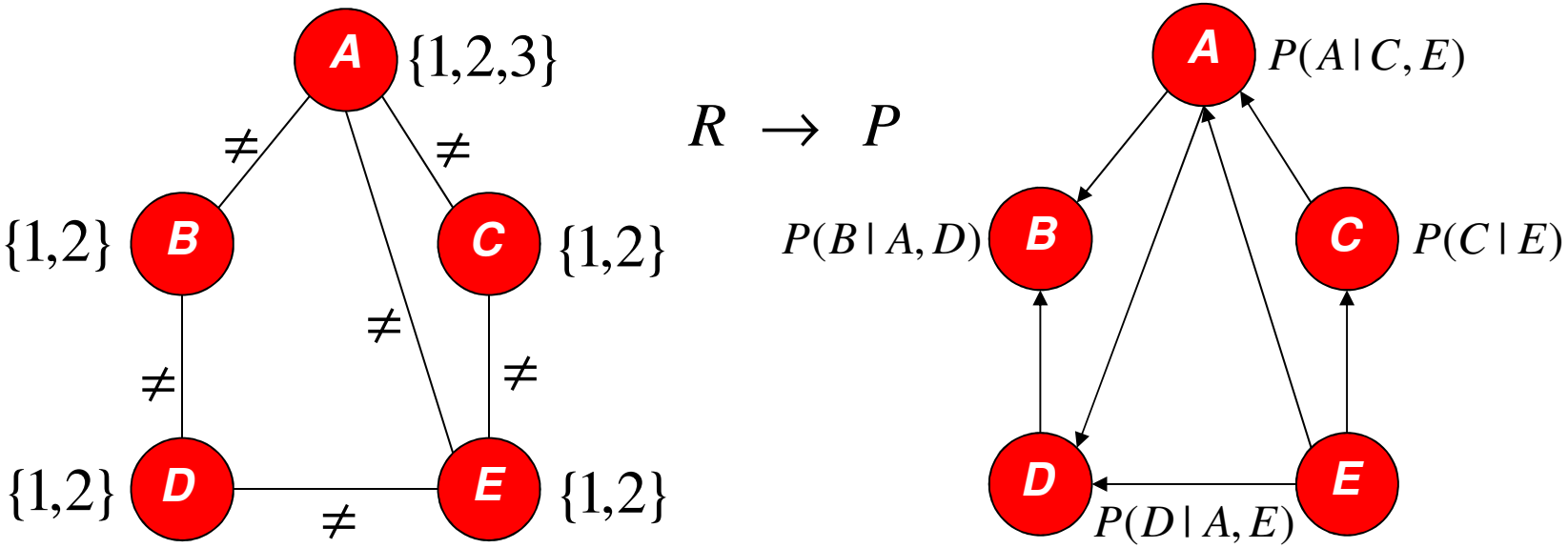


$R \rightarrow P$



Conversion solved problem

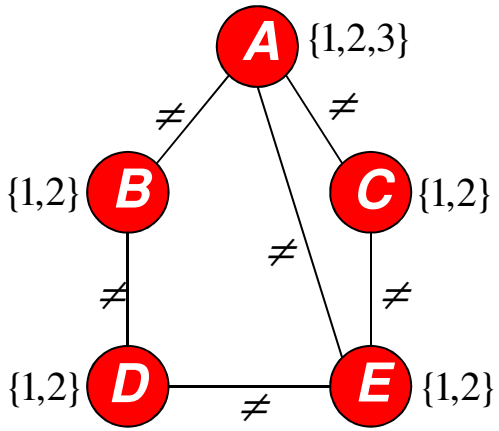
A variable-elmination-based conversion



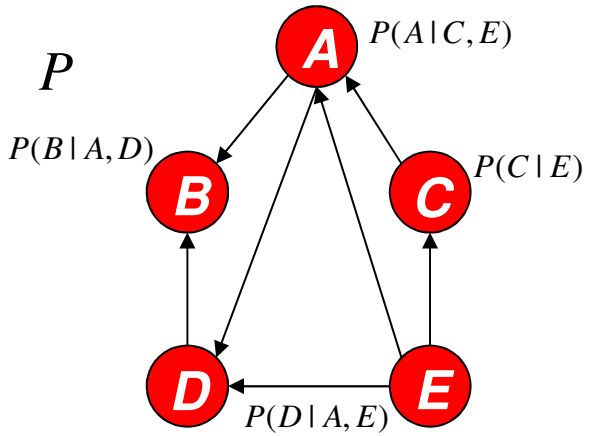
$$P(B \mid A, D) = \frac{R(A,B) * R(B,D)}{\sum_B R(A,B) * R(B,D)}$$

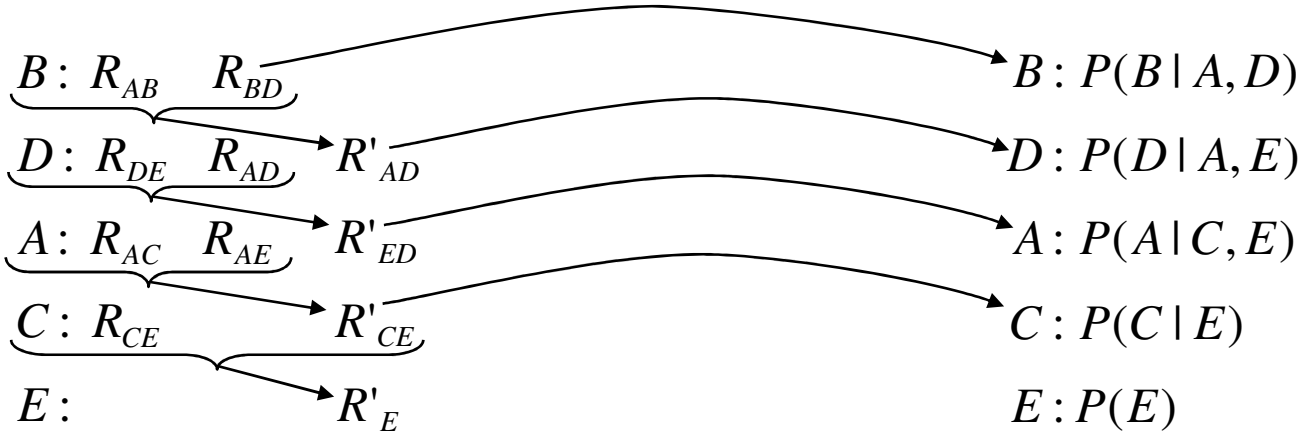
Complexity: exp(w)*
But the network is already easy

A conversion algorithm



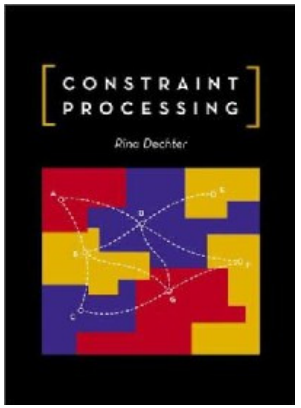
$R \rightarrow P$





$$P(B \mid A, D) = \frac{R(A,B) * R(B,D)}{\sum_B R(A,B) * R(B,D)} \qquad R(A,D) = \sum_B R(A,B) * R(B,D)$$

Thank you!



For publication see:
http://www.ics.uci.edu/~dechter/publications.html

Kalev Kask
Irina Rish
Bozhena Bidyuk
Robert Mateescu
Radu Marinescu
Vibhav Gogate
Emma Rollon
Natalia Flerova
Lars Otten
William Lam