Constraint Processing from the Graphical Model Perspective

Rina Dechter

Information and Computer Science, UC-Irvine

Overview and Road Map

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

Road Map

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

Propositional Reasoning

Example: party problem

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

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

$$\Updownarrow$$

Is the *propositional theory*

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

Sudoku –
Constraint Satisfaction

- ***Constraint***
- ***Propagation***

- ***Inference***

		2	4		6			
8	6	5	1			2		
	1				8	6		9
9				4		8	6	
	4	7				1	9	
	5	8		6				3
4		6	9				7	2,~~3~~ ~~4~~,~~6~~ 2
		9			4	5	8	1
			3		2	9		

- ***Variables*: *empty slots***
- ***Domains* = *{1,2,3,4,5,6,7,8,9}***
- ***Constraints*:**
  - ***27 all-different***

Each row, column and major block must be alldifferent

"Well posed" if it has unique solution: 27 constraints

Constraint Networks

A

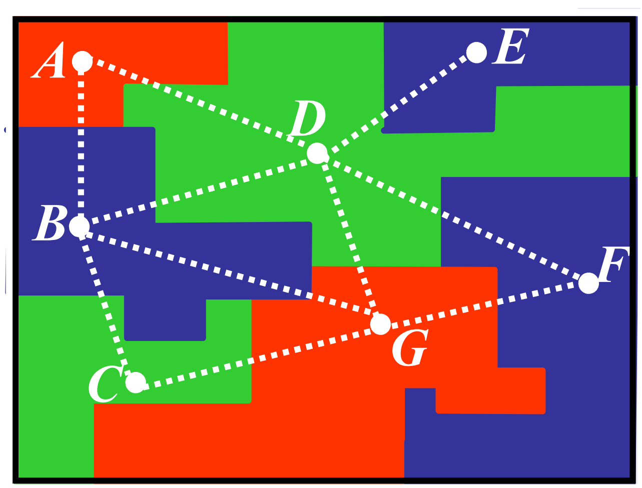
Example: map coloring

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

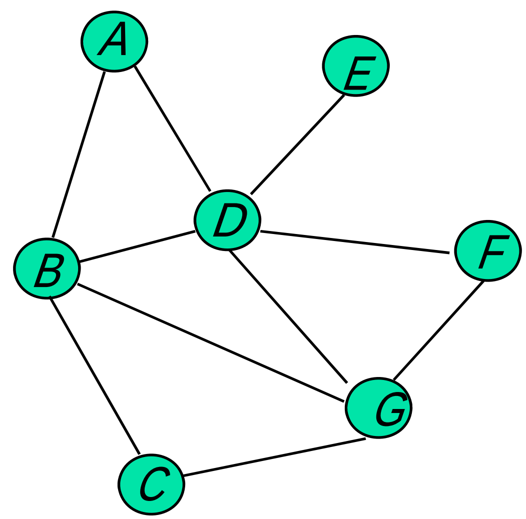
Values - colors (red, green, blue)

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

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



Constraint graph



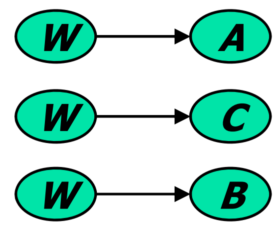
Applications

- Planning and scheduling
  - Transportation scheduling, factory scheduling
- Configuration and design problems
  - floorplans
- Circuit diagnosis
- Scene labeling
- Spreadsheets
- Temporal reasoning, Timetabling
- Natural language processing
- Puzzles: crosswords, sudoku, cryptarithmetic

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

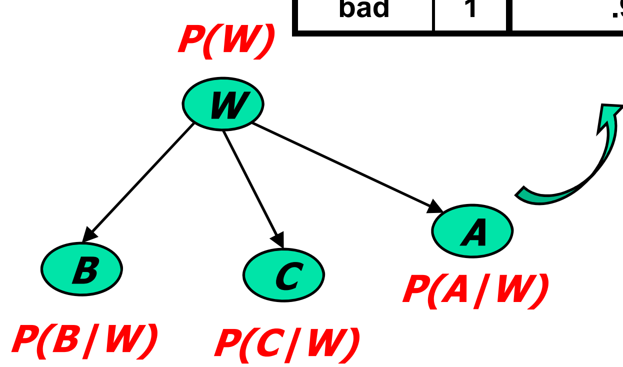
P(C|W=bad)=.1

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?*

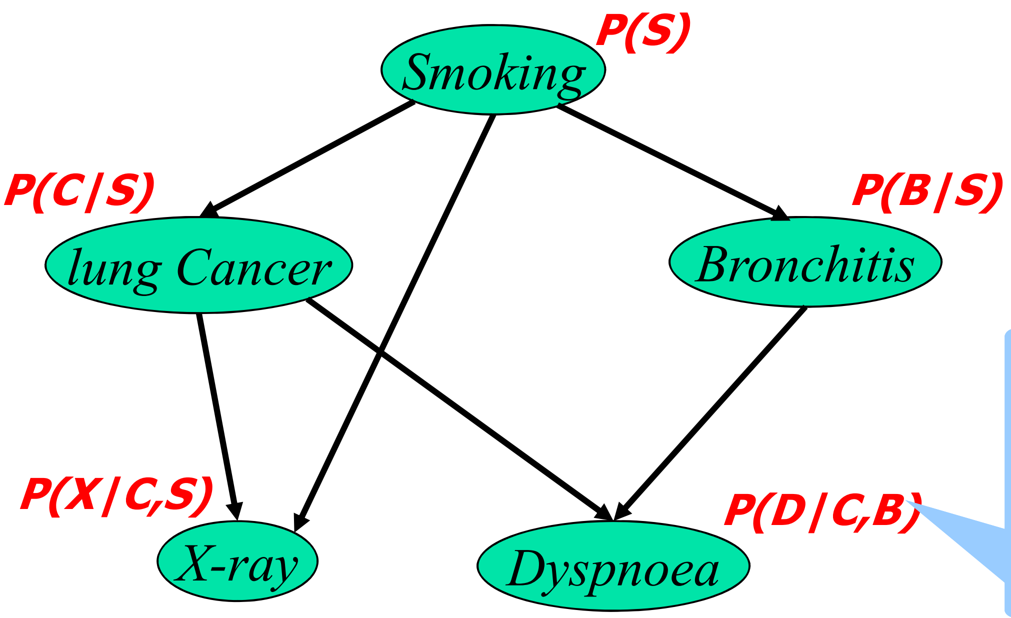
W	A	P(A|W)
good	0	.01
good	1	.99
bad	0	.1
bad	1	.9



$$P(W,A,C,B) = P(B|W) \cdot P(C|W) \cdot P(A|W) \cdot P(W)$$

$$P(A,C,B|W=bad) = 0.9 \cdot 0.1 \cdot 0.5$$

Bayesian Networks: Representation
(Pearl, 1988)



$\mathbf{BN} = (\mathbf{G}, \mathbf{\Theta})$

CPD:

C	B	P(D|C,B)	
0	0	0.1	0.9
0	1	0.7	0.3
1	0	0.8	0.2
1	1	0.9	0.1

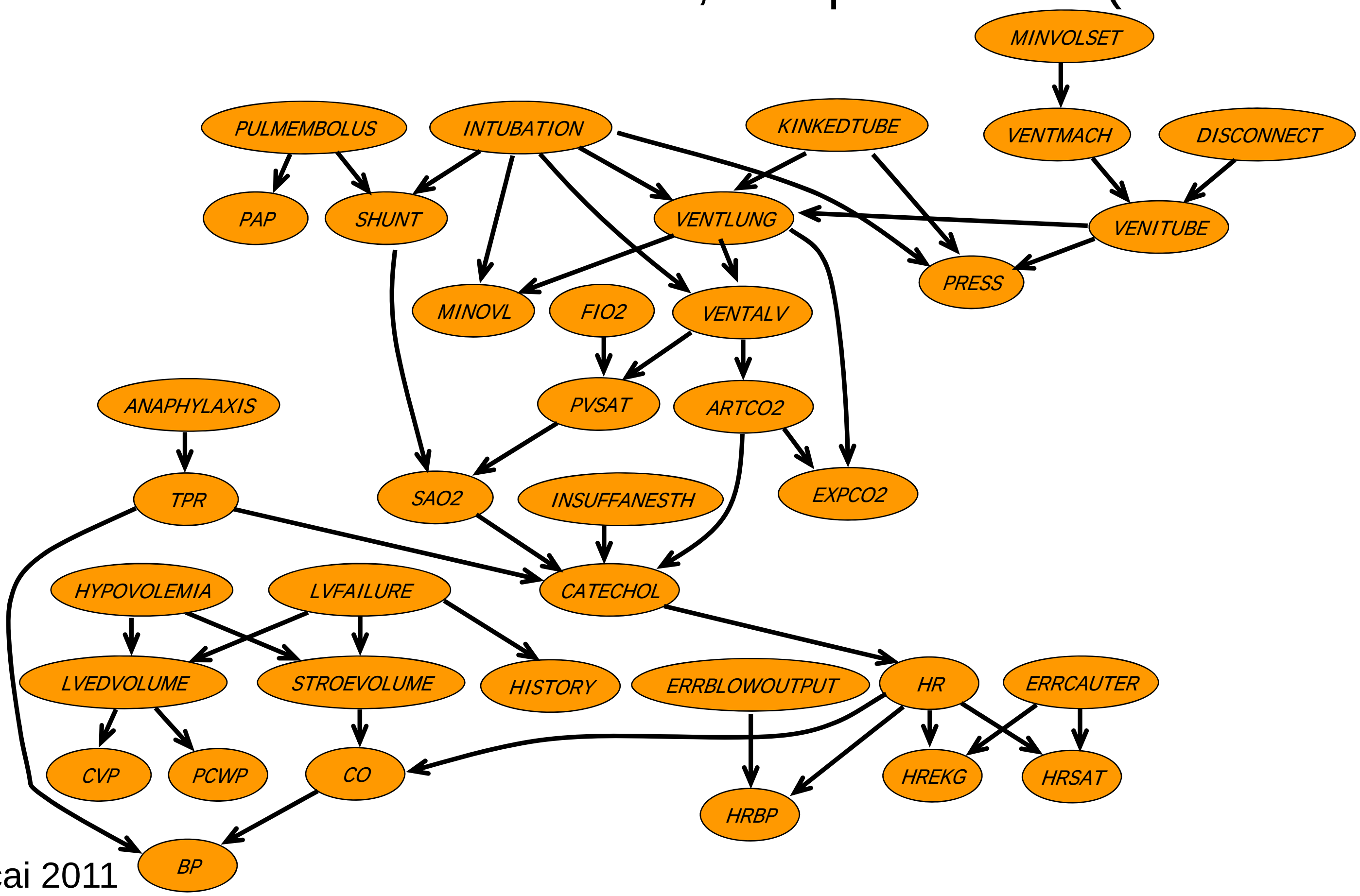
$P(S, C, B, X, D) = P(S)\, P(C|S)\, P(B|S)\, P(X|C,S)\, P(D|C,B)$

Belief Updating:

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

Monitoring Intensive-Care Patients

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

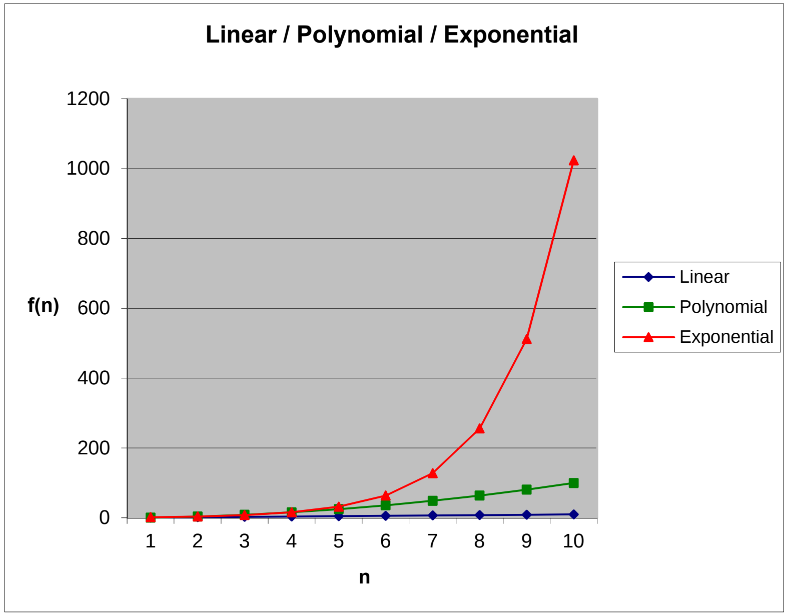


Sample Domains

- 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



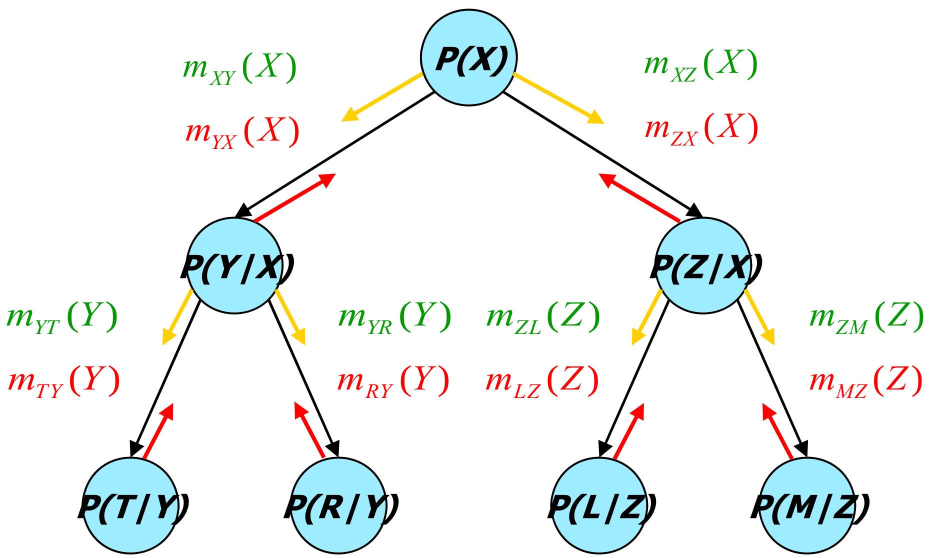
Reasoning is computationally hard

Complexity is Time and space(memory)

Tree-solving is easy

Belief updating (sum-prod)

CSP – consistency (projection-join)



MPE (max-prod)

#CSP (sum-prod)

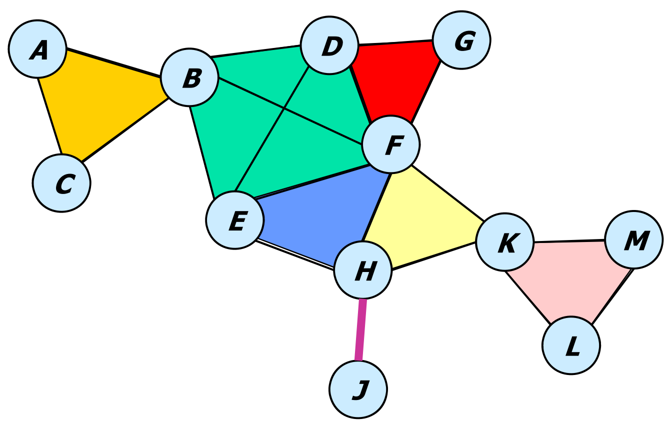
Trees are processed in linear time and memory

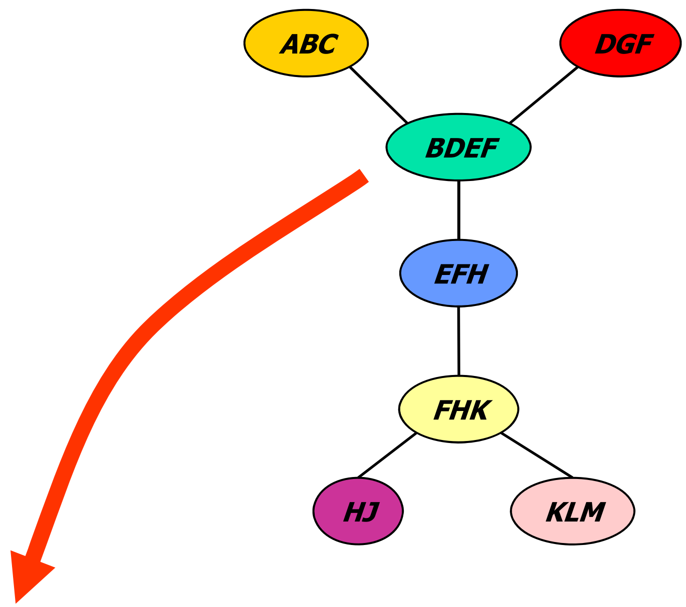
Transforming into a Tree

- **By Inference (thinking)**
  - Transform into a single, equivalent tree of sub-problems

- **By Conditioning (guessing)**
  - Transform into many tree-like sub-problems.

Inference and Treewidth

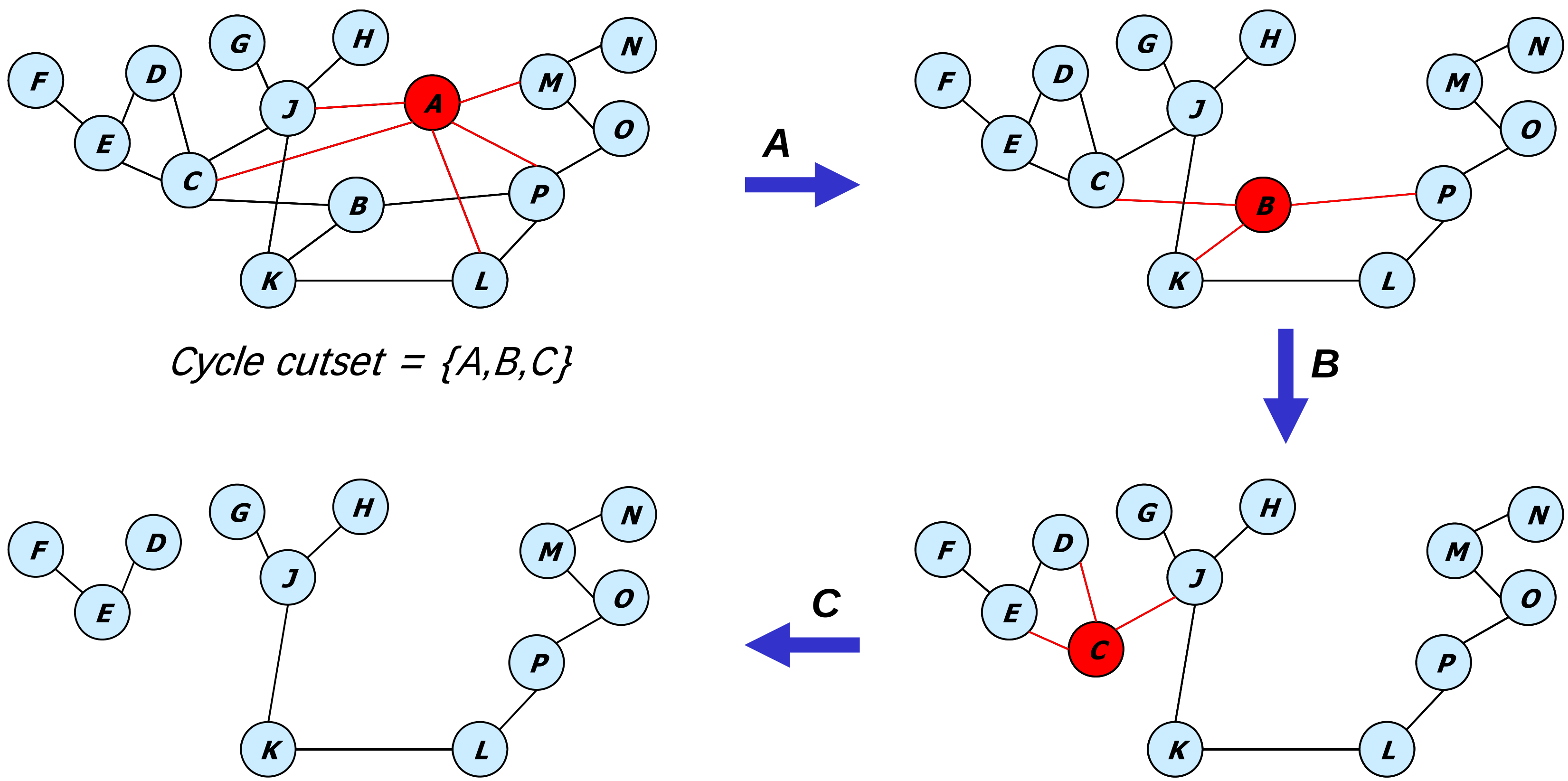




Inference algorithm:
Time: exp(tree-width)
Space: exp(tree-width)

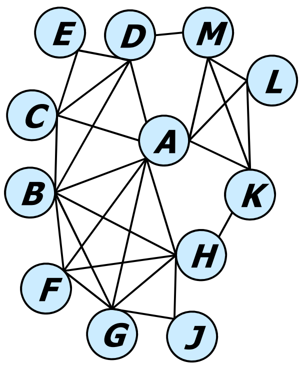
treewidth = 4 - 1 = 3
treewidth = (maximum cluster size) - 1

Conditioning and Cycle cutset

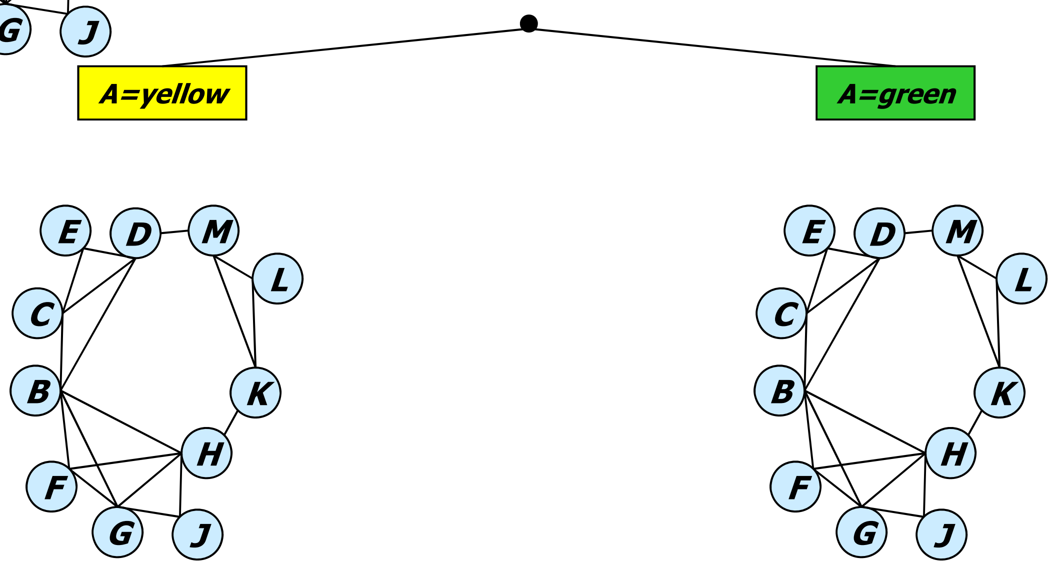


Search over the Cutset

Graph Coloring problem

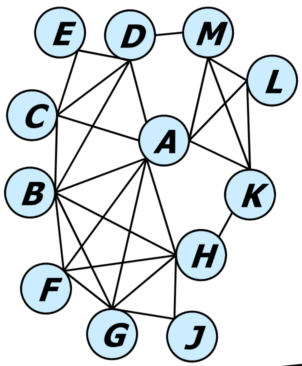


- *Inference may require too much memory*
- ***Condition (guessing)*** *on some of the variables*

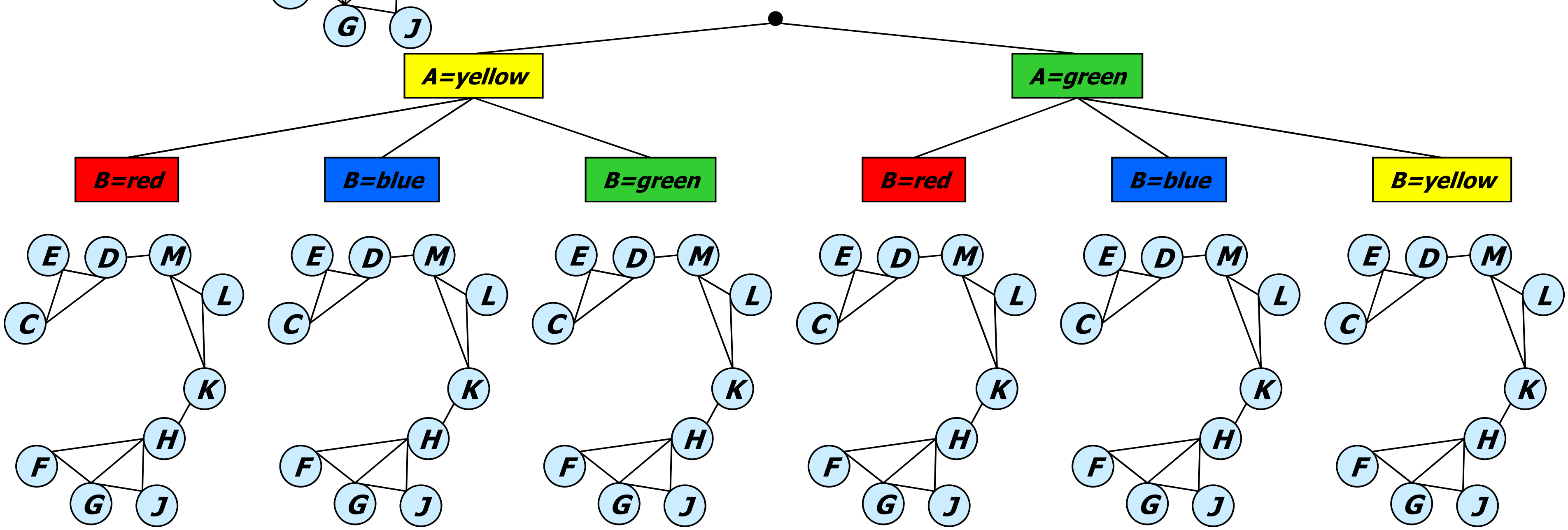


Search over the Cutset (cont)

Graph Coloring problem

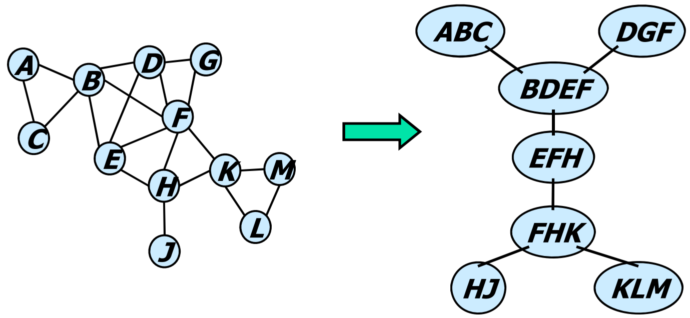


- *Inference may require too much memory*
- ***Condition*** *on some of the variables*



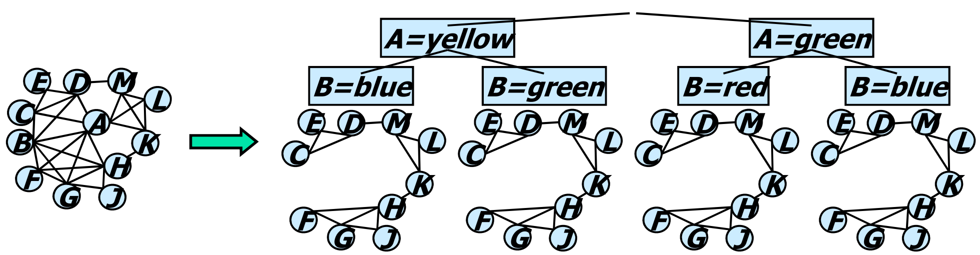
Inference vs. Conditioning

- **By Inference (thinking)**



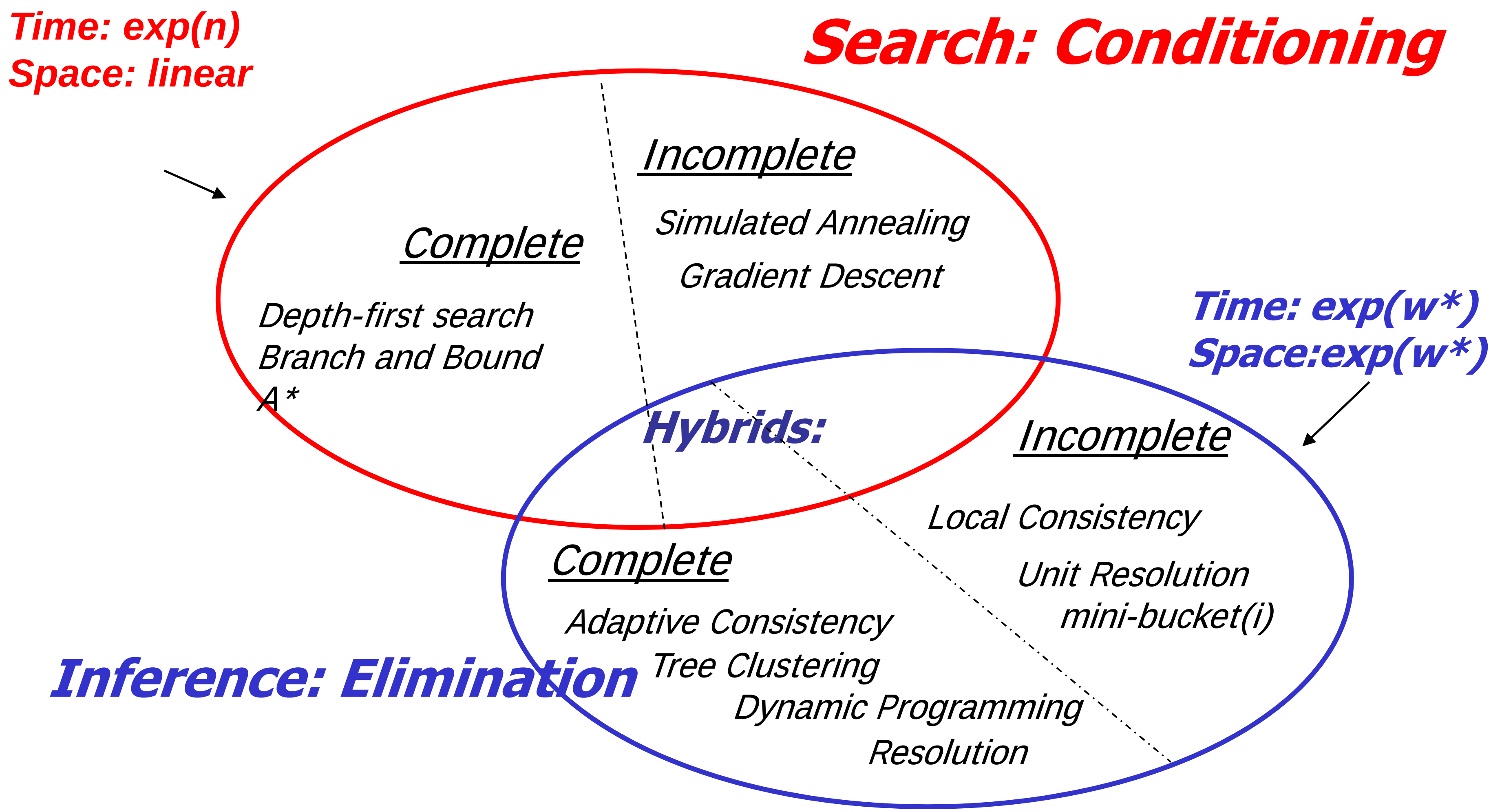
Exponential in treewidth
Time and memory

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



Exponential in cycle-cutset
Time-wise, linear memory

Graphical Models Reasoning



Road Map

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

Constraint Networks

A

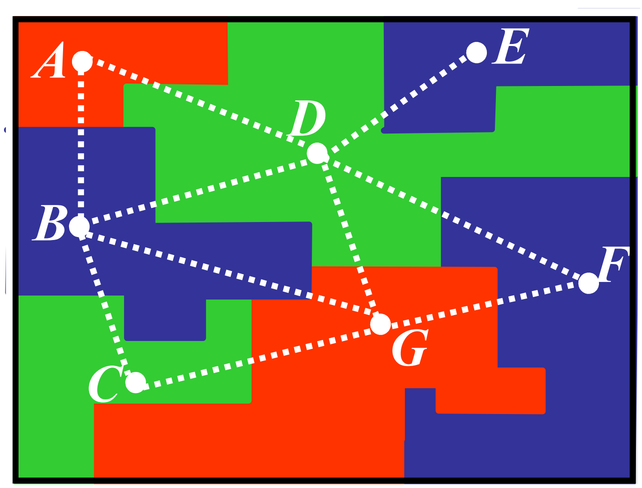
Example: map coloring

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

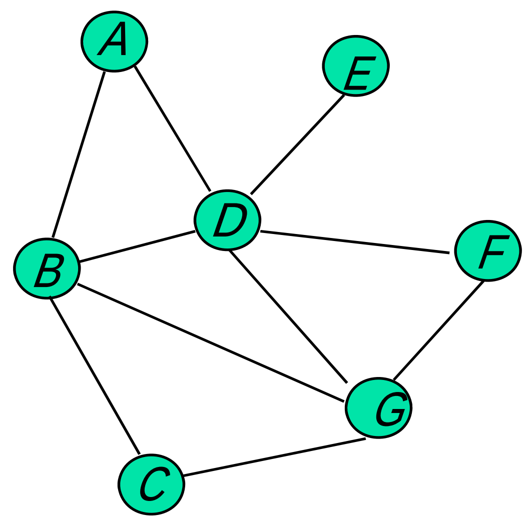
Values - colors (red, green, blue)

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

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



Constraint graph



Constraint Satisfaction Tasks

Example: map coloring

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

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

Constraints:

$A \neq B$, $A \neq D$, $D \neq E$, *etc.*

A	B	C	D	E…
red	green	red	green	blue
red	blu	gre	green	blue
…	…	…	…	gre en
…	…	…	…	red
red	blu e	red	green	red

Are the constraints consistent?

Find a solution, find all solutions

Count all solutions

Find a good solution

Information as Constraints

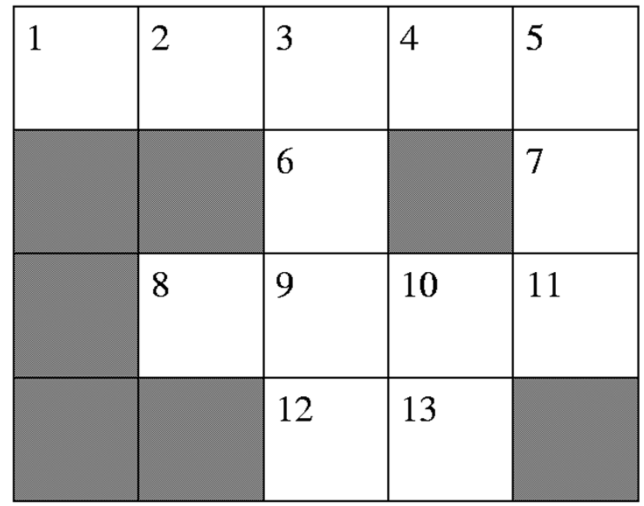
- I have to finish my talk in 30 minutes
- 180 degrees in a triangle
- Memory in our computer is limited
- The four nucleotides that makes up a DNA only combine in a particular sequence
- Sentences in English must obey the rules of syntax
- Susan cannot be married to both John and Bill
- Alexander the Great died in 333 B.C.

Constraint Network

- 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** $C = \{C_1, \ldots C_t\},,, C_i = (S_i, R_i)$
  - ***R*** **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*

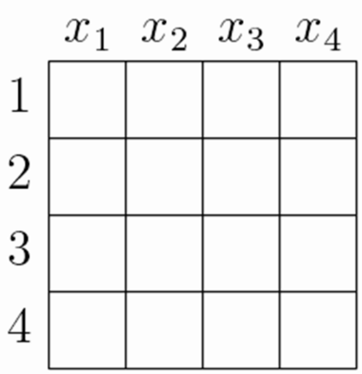
Crossword puzzle

- Variables: $x_1, \ldots, x_{13}$
- 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}

Figure 2.1: The 4-queens constraint network. The network has four variables, all with domains $D_i = \{1, 2, 3, 4\}$. (a) The labeled chess board. (b) The constraints between variables.



(a)

$$
\begin{aligned}
R_{12} &= \{(1,3),\ (1,4),\ (2,4),\ (3,1),\ (4,1),\ (4,2)\}\\
R_{13} &= \{(1,2),\ (1,4),\ (2,1),\ (2,3),\ (3,2),\ (3,4),\ (4,1),\ (4,3)\}\\
R_{14} &= \{(1,2),\ (1,3),\ (2,1),\ (2,3),\ (2,4),\ (3,1),\ (3,2),\ (3,4)\\
&\quad (4,2),\ (4,3)\}\\
R_{23} &= \{(1,3),\ (1,4),\ (2,4),\ (3,1),\ (4,1),\ (4,2)\}\\
R_{24} &= \{(1,2),\ (1,4),\ (2,1),\ (2,3),\ (3,2),\ (3,4),\ (4,1),\ (4,3)\}\\
R_{34} &= \{(1,3),\ (1,4),\ (2,4),\ (3,1),\ (4,1),\ (4,2)\}
\end{aligned}
$$

(b)

Figure 2.2: 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).

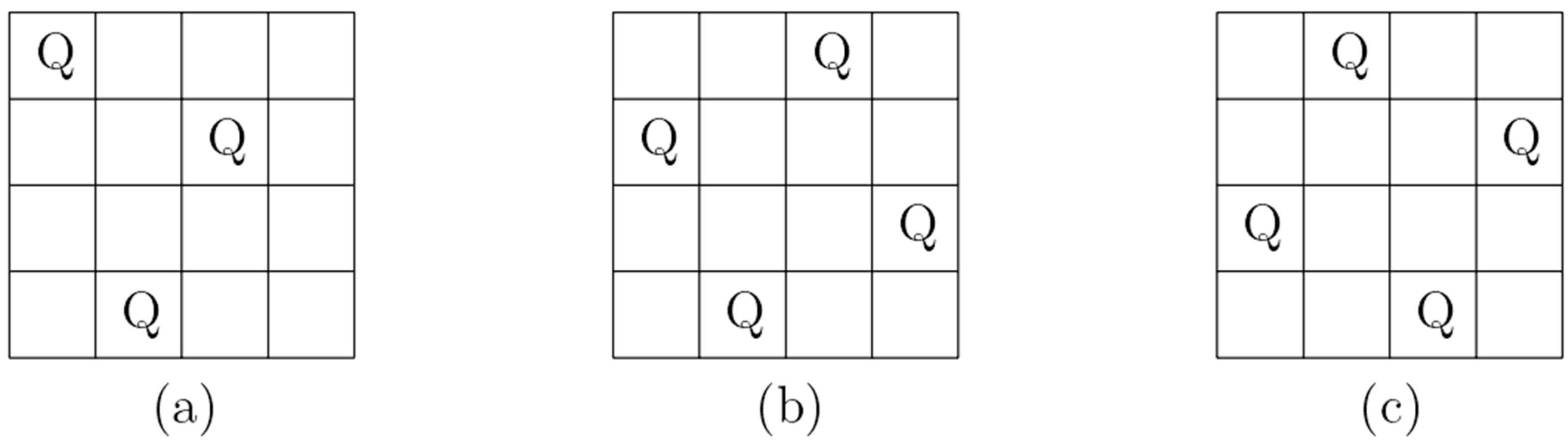
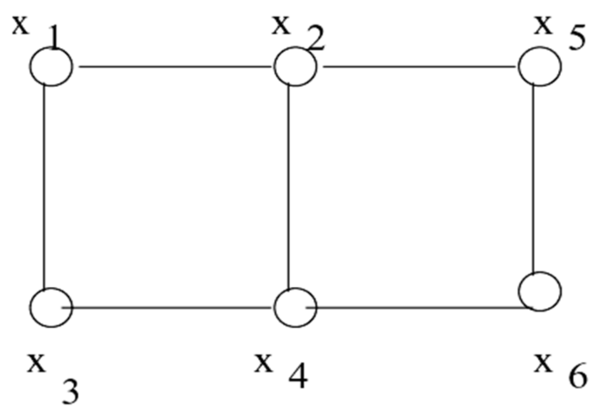
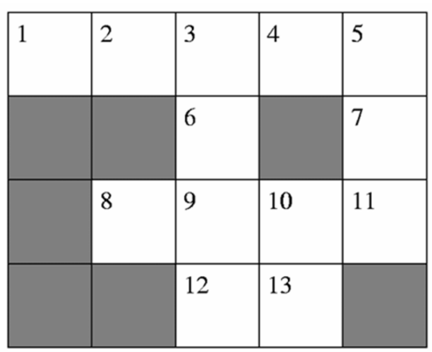
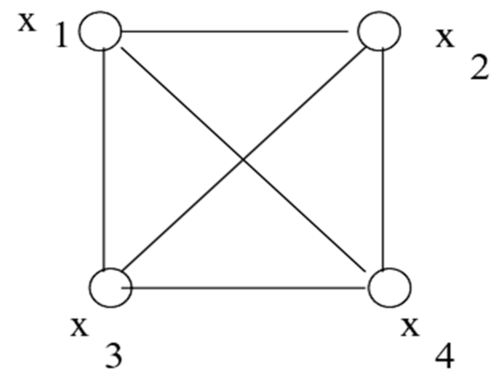


Figure 2.3: Constraint graphs of (a) the crossword puzzle and (b) the 4-queens problem.



(a)





(b)

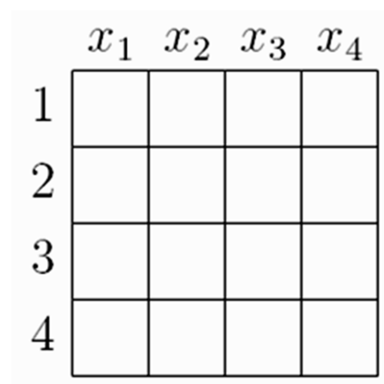


Figure 1.9: Example of selection, projection, and join operations on relations.

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''

x_1	x_2	x_3
b	b	c
c	b	c

(a) $\sigma_{x_3=c}(R')$

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

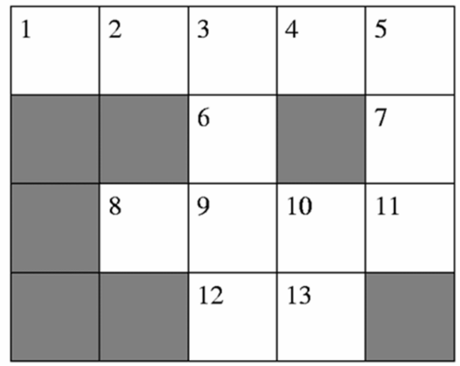
- Relation: allowed tuples

X	Y	Z
1	3	2
2	1	3

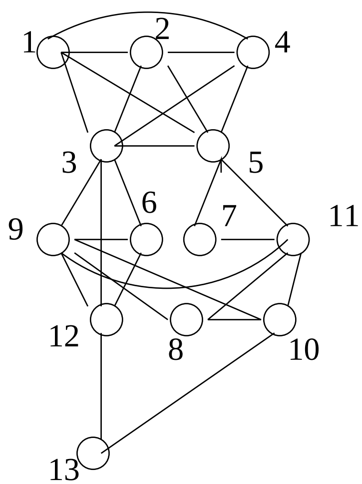
- Algebraic expression: $X + Y^2 \leq 10, X \neq Y$
- Propositional formula: $(a \vee b) \rightarrow \neg c$
- Semantics: by a relation

Constraint Graphs:

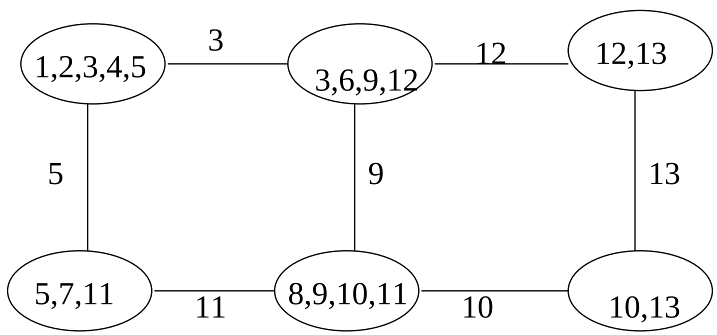
Primal, Dual and Hypergraphs



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



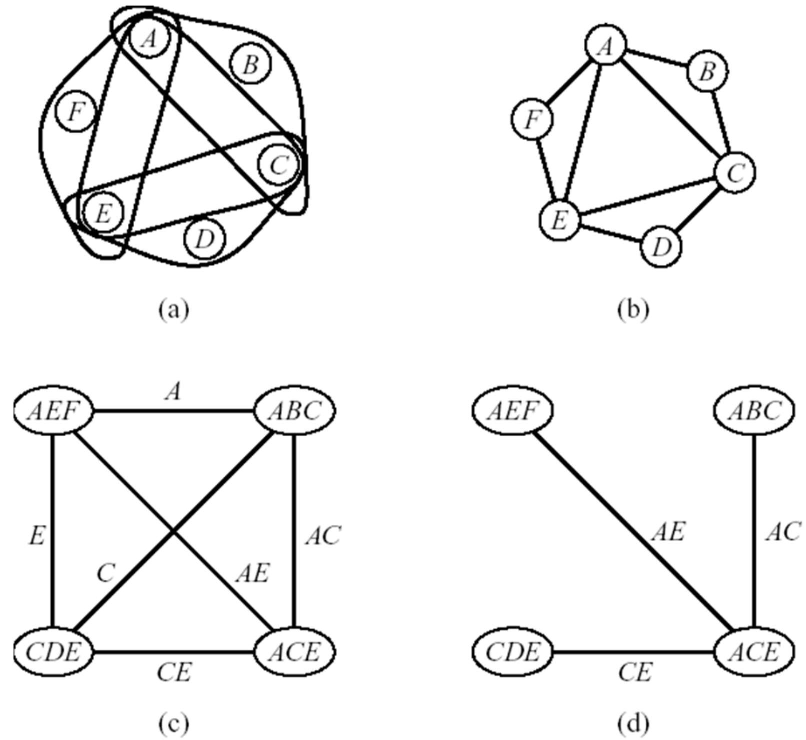
(a)



(b)

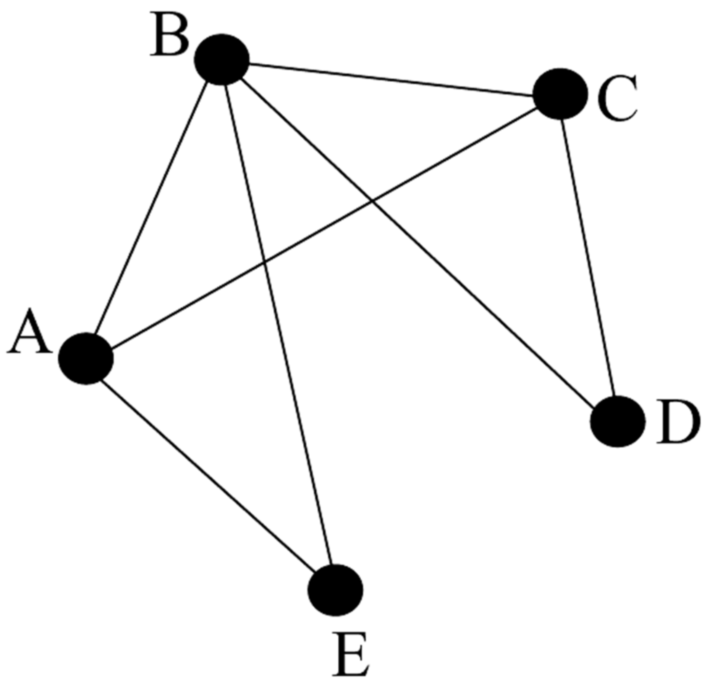
Graph Concepts Reviews:
Hyper Graphs and Dual Graphs

- **A hypergraph**
- **Dual graphs**
- **A primal graph**

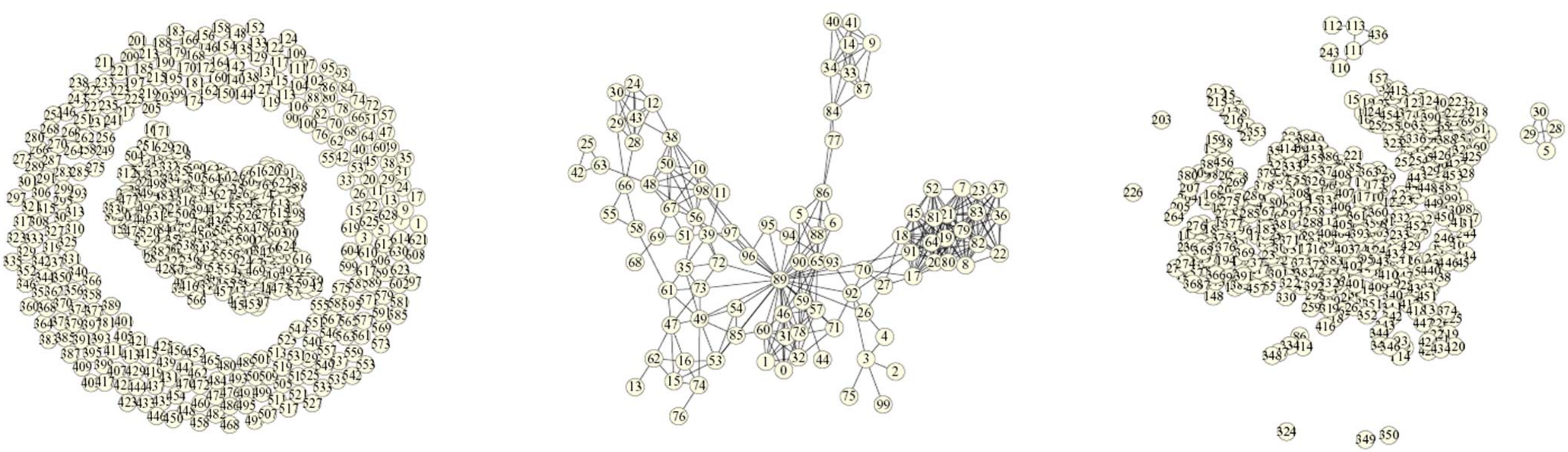


Propositional Satisfiability

φ = {(¬C), (A v B v C), (¬A v B v E), (¬B v C v D)}.



Constraint graphs of 3 instances of the Radio frequency assignment problem in CELAR's benchmark

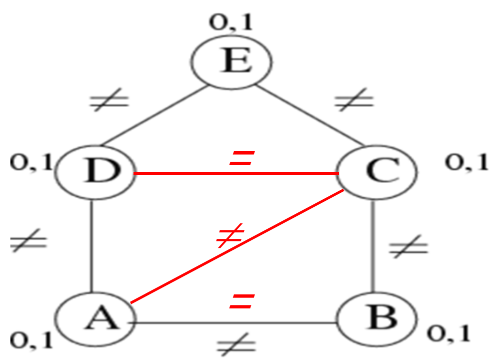


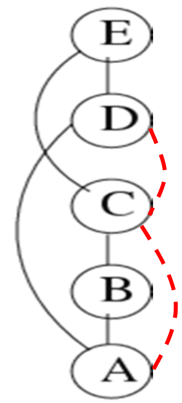
Road Map

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

Bucket Elimination

Adaptive Consistency (Dechter & Pearl, 1987)





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

Bucket D: $D \neq A$ → $A \neq C$

Bucket C: $C \neq B$ → $B = A$

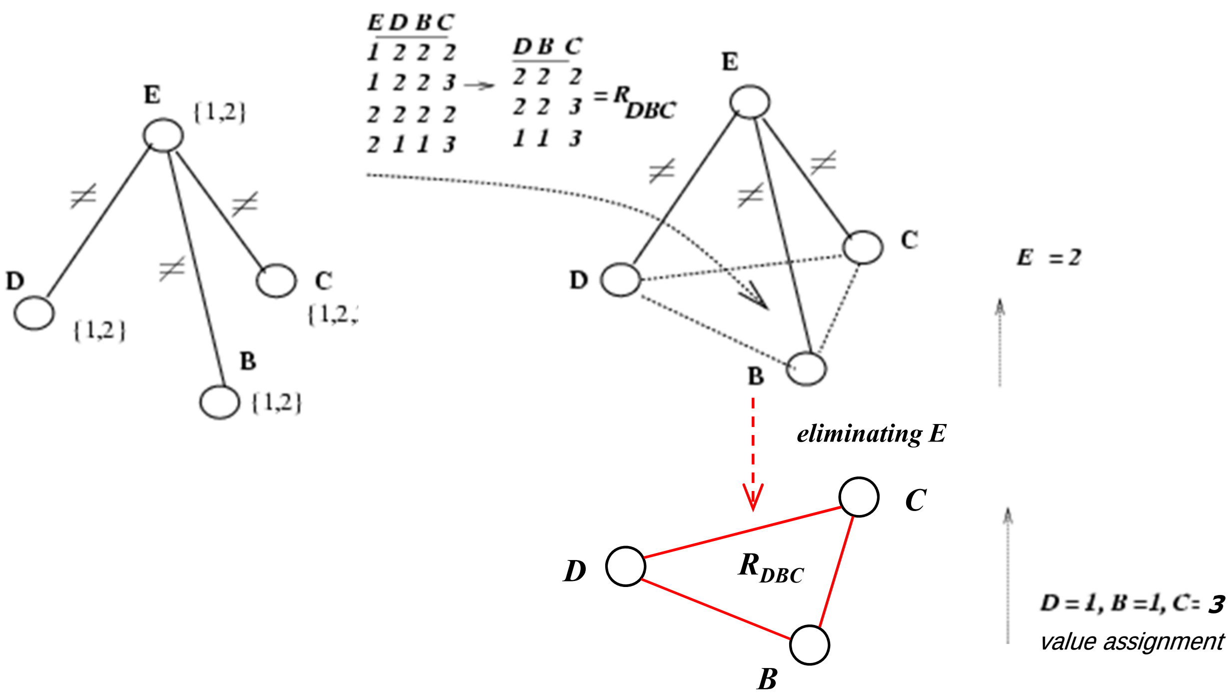
Bucket B: $B \neq A$ → *contradiction*

Bucket A:

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

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

The Idea of Elimination

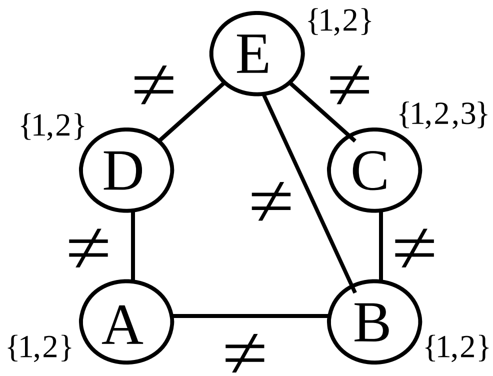


$$R_{DBC} = \prod_{DBC} R_{ED} \bowtie R_{EB} \bowtie R_{EC}$$

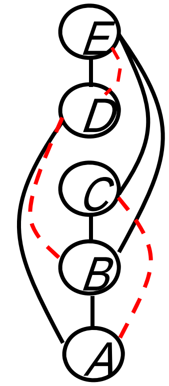
Eliminate variable E $\Leftrightarrow$ join and project

Bucket Elimination

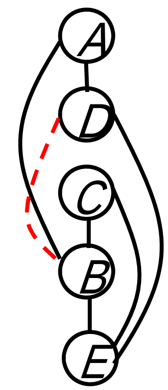
Adaptive Consistency (Dechter & Pearl, 1987)



$$
\begin{aligned}
&Bucket(E):\ \mathrm{E} \neq \mathrm{D},\ \mathrm{E} \neq \mathrm{C},\ \mathrm{E} \neq \mathrm{B} \\
&Bucket(D):\ \mathrm{D} \neq \mathrm{A}\ \|\ R_{DCB} \\
&Bucket(C):\ \mathrm{C} \neq \mathrm{B}\ \|\ R_{ACB} \\
&Bucket(B):\ \mathrm{B} \neq \mathrm{A}\ \|\ R_{AB} \\
&Bucket(A):\ R_{A}
\end{aligned}
$$

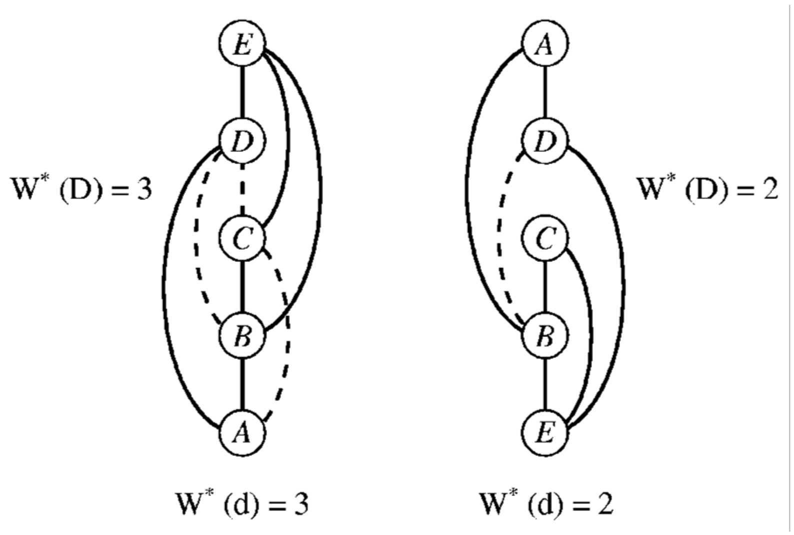


$$
\begin{aligned}
&Bucket(A):\ \mathrm{A} \neq \mathrm{D},\ \mathrm{A} \neq \mathrm{B} \\
&Bucket(D):\ \mathrm{D} \neq \mathrm{E}\ \|\ R_{DB} \\
&Bucket(C):\ \mathrm{C} \neq \mathrm{B},\ \mathrm{C} \neq \mathrm{E} \\
&Bucket(B):\ \mathrm{B} \neq \mathrm{E}\ \|\ R^{D}_{BE},\ R^{C}_{BE} \\
&Bucket(E):\ \|\ R_{E}
\end{aligned}
$$



Complexity : $O(n \exp(w^*(d)))$,
$w^*(\mathrm{d})$ - ***induced width along ordering d***

The induced-width



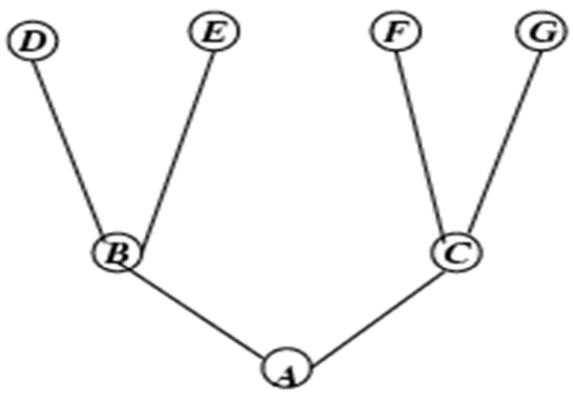
- **Width along *d*, w(d):**
  - **max # of previous parents**
- **Induced width w*(d):**
  - **The width in the ordered *induced graph***
- **Induced-width w*:**
  - **Smallest induced-width over all orderings**
- **Finding w***
  - **NP-complete *(Arnborg, 1985) but greedy heuristics (min-fill).***

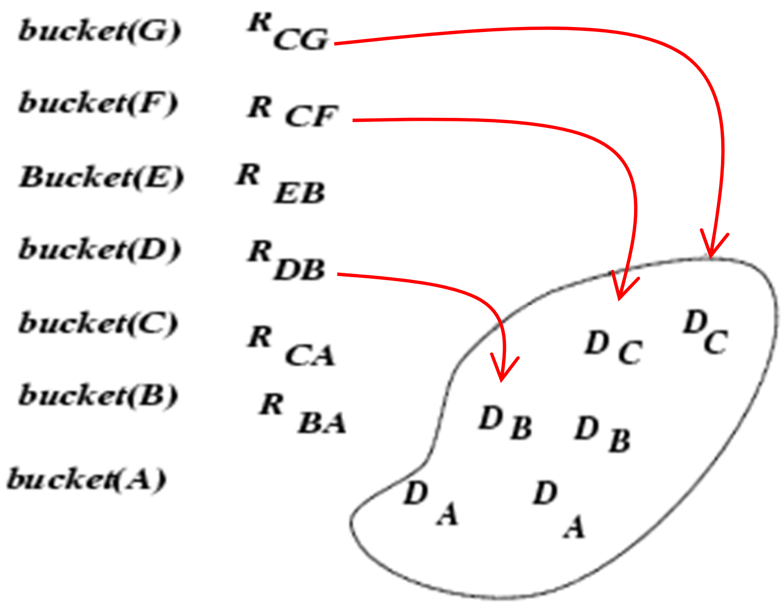
Properties of bucket-elimination (Adaptive consistency)

- Adaptive consistency generates a **backtrack-free** representation.
- The time and space complexity of adaptive consistency along ordering *d* is exp(w*(d)). Therefore, problems having **bounded induced width** are tractable (solved in polynomial time).
- Examples :
  - ***trees (w*=1),***
  - ***series-parallel networks (w*=2)***
  - partial ***k-trees (w*=k).***

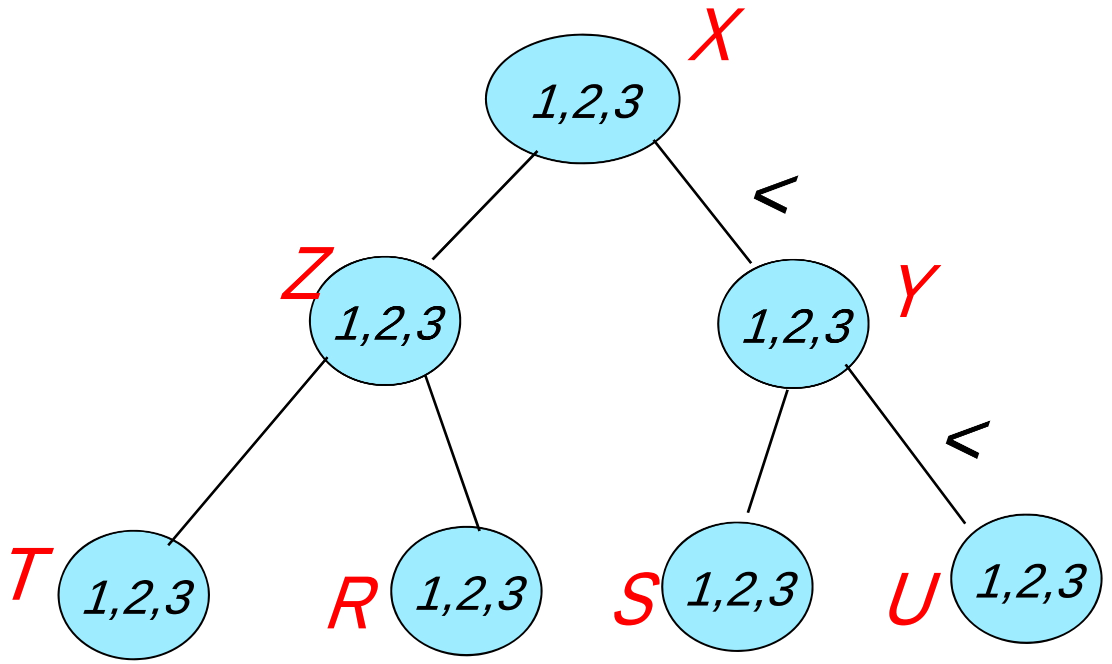
Solving Trees
(Mackworth and Freuder, 1985)

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

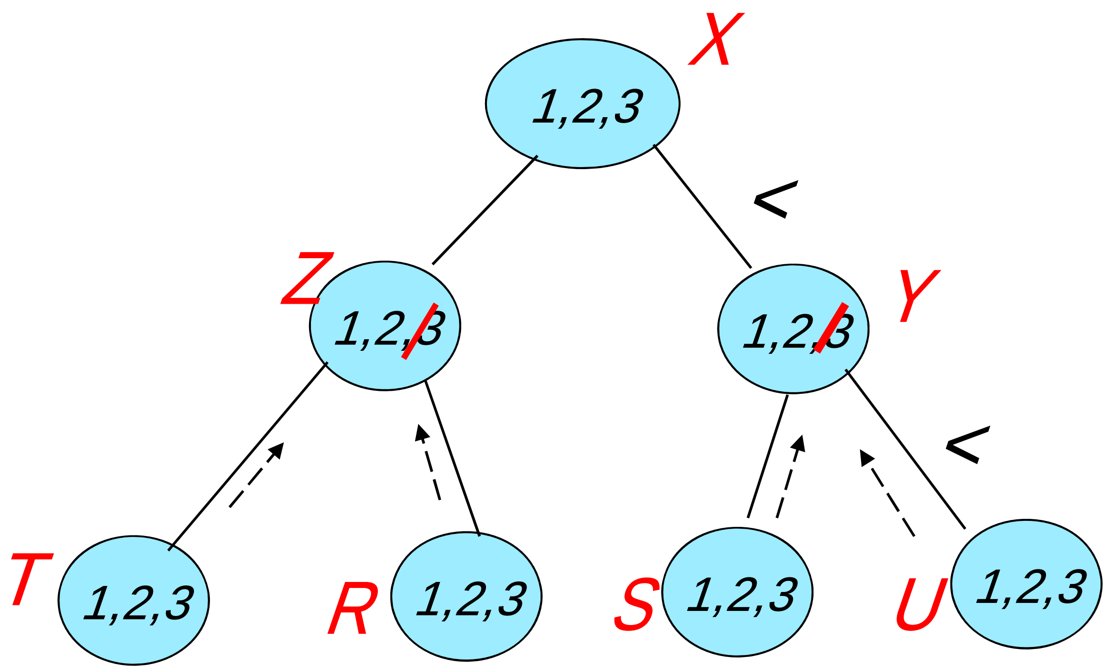




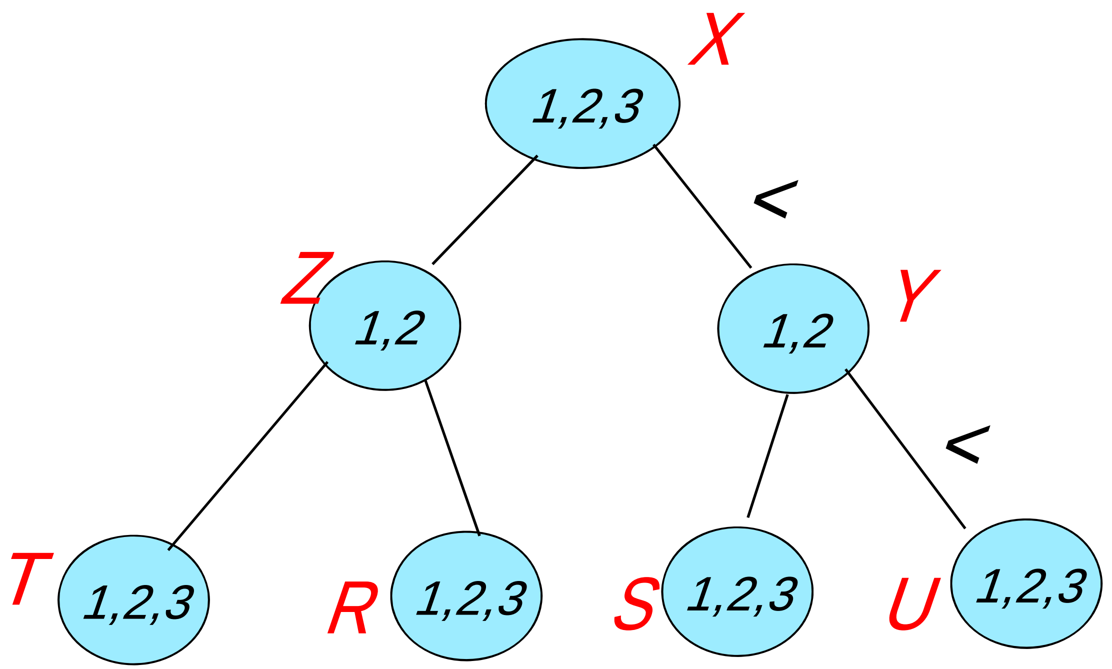
Tree Solving is Easy



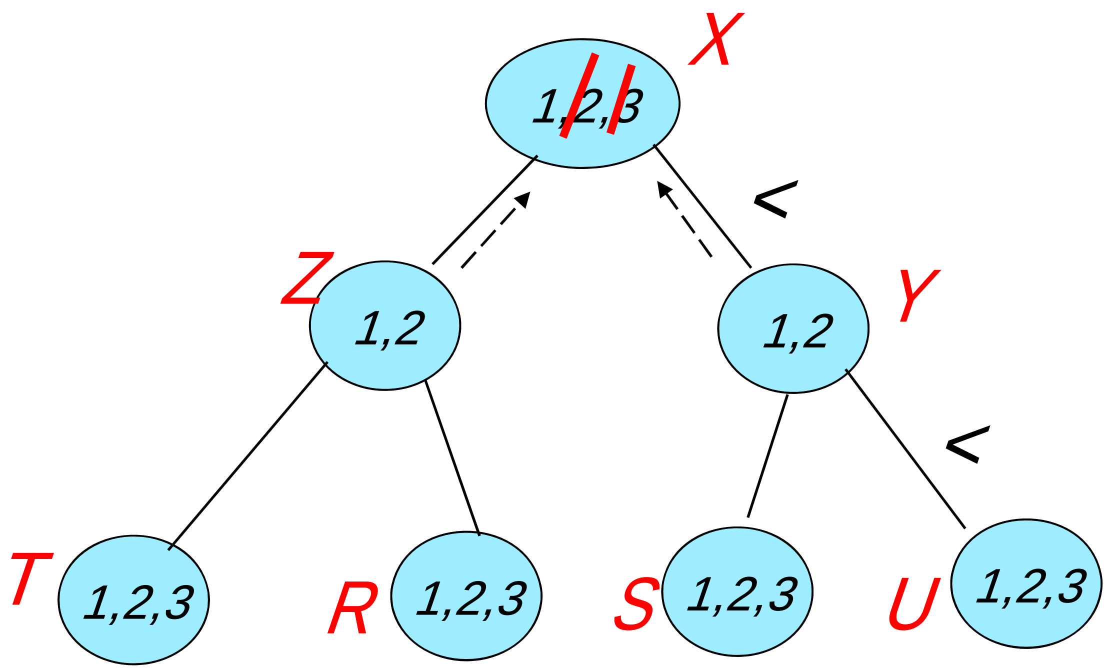
Tree Solving is Easy



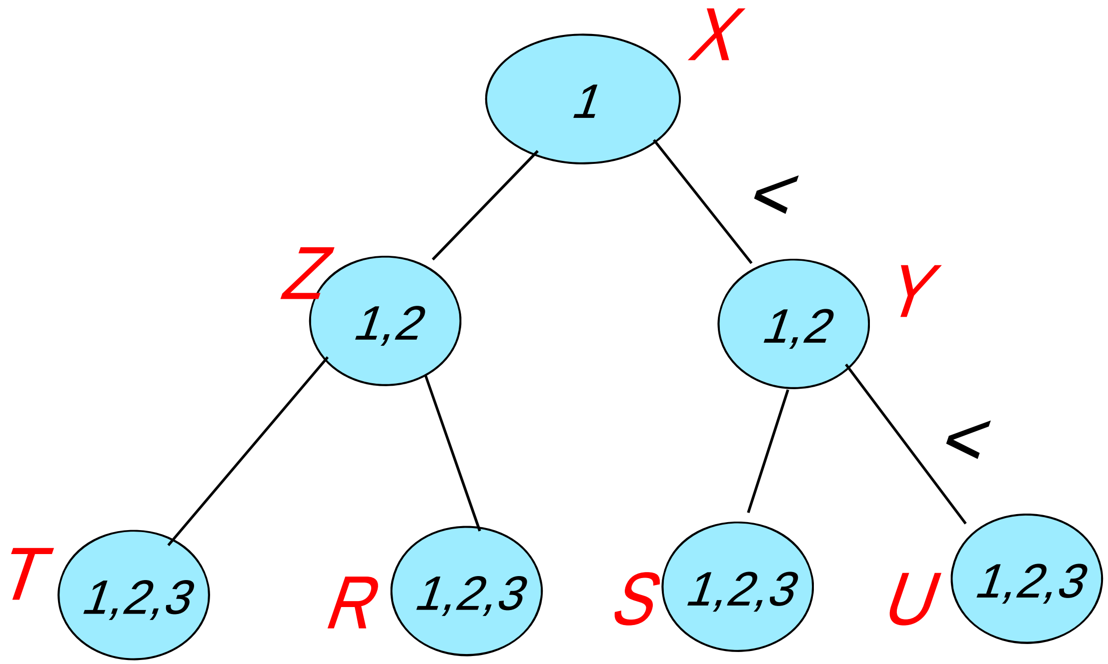
Tree Solving is Easy



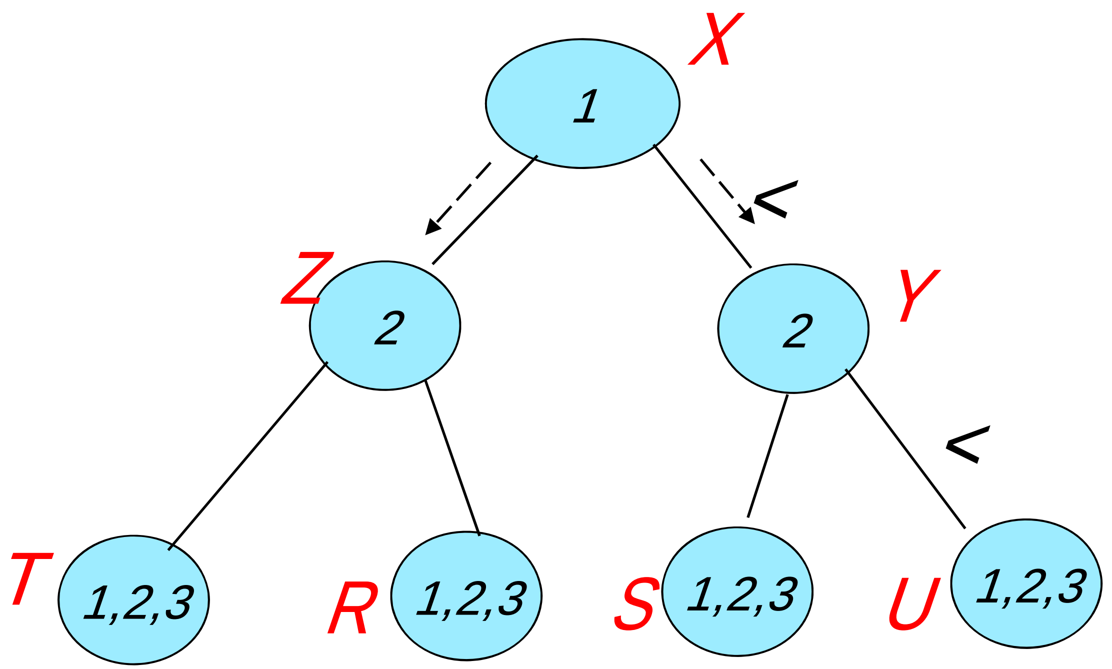
Tree Solving is Easy



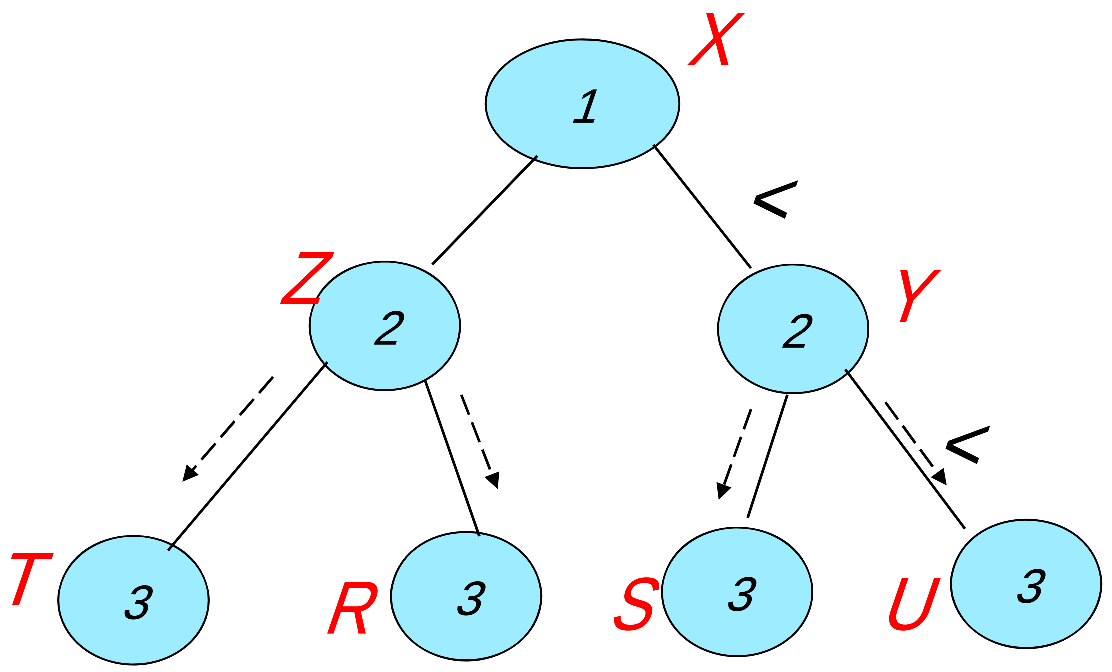
Tree Solving is Easy



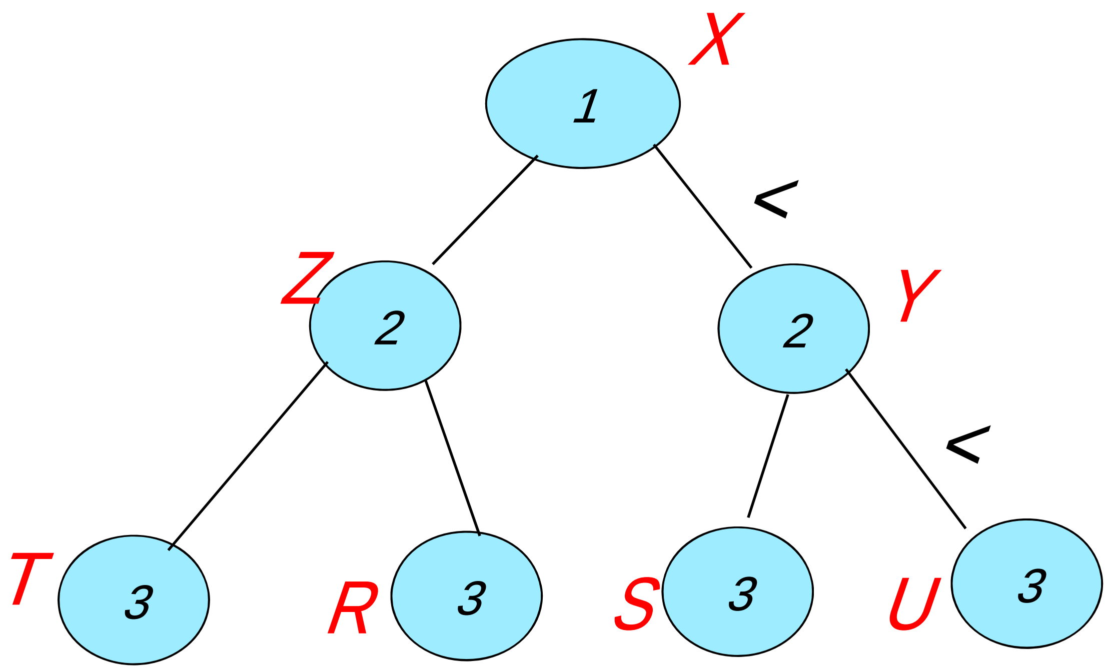
Tree Solving is Easy



Tree Solving is Easy

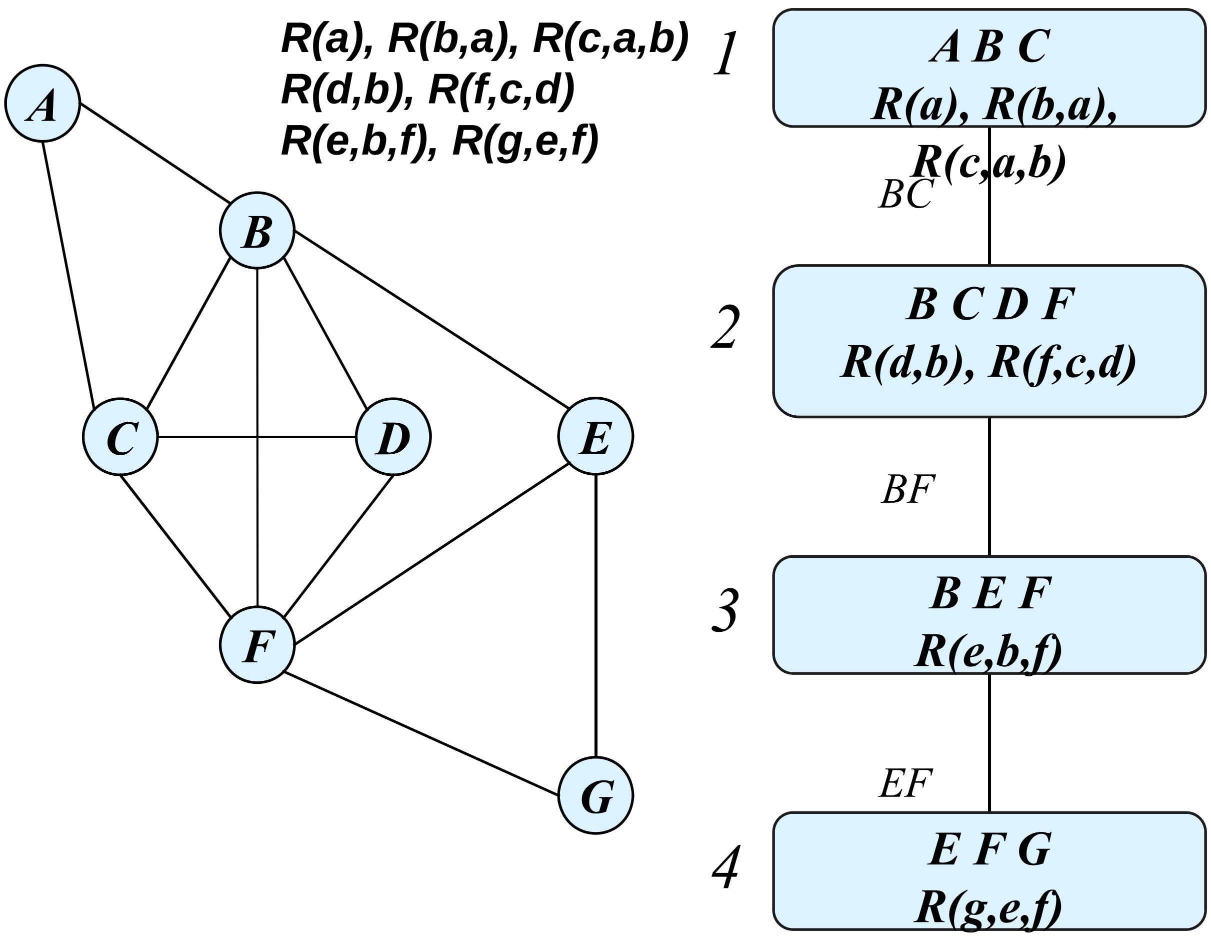


Tree Solving is Easy



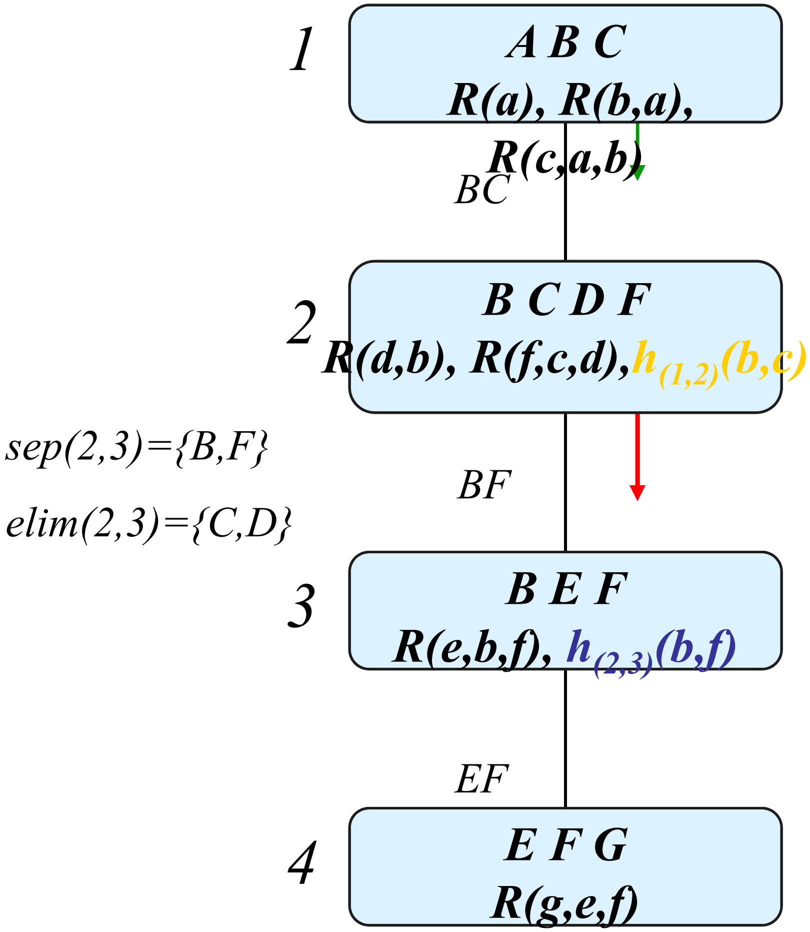
Constraint propagation
Solves trees in linear time

Tree Decomposition



- Each function in a cluster
- Satisfy running intersection property

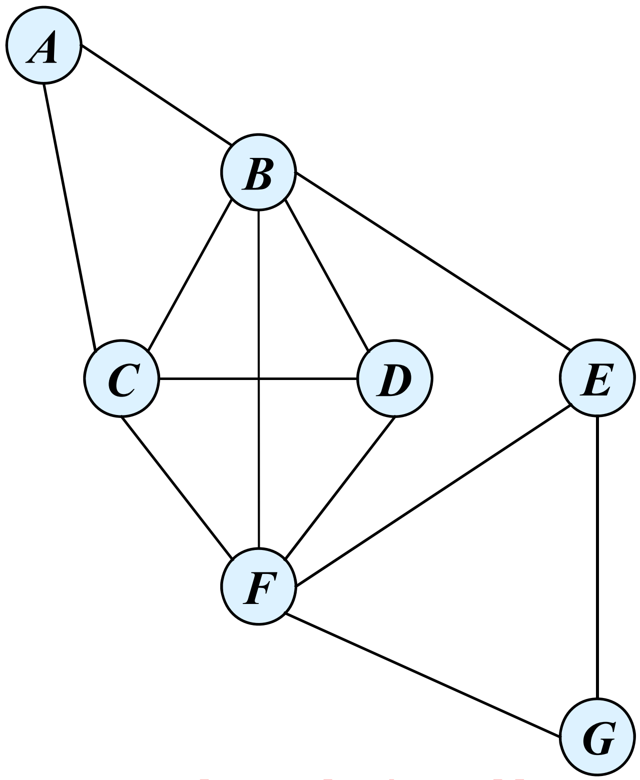
Cluster Tree Elimination



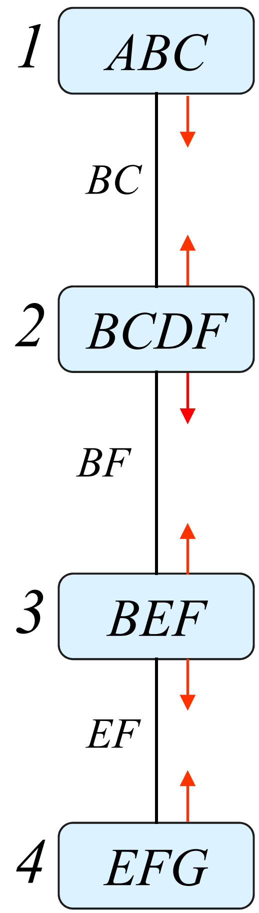
$$h_{(1,2)}(b,c) = \Downarrow_a R(a) \otimes R(b,a) \otimes R(c,a,b)$$

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

CTE: Cluster Tree Elimination



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



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

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

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

$$h_{(3,2)}(b,f) = \Downarrow_e R(e,b,f) \otimes h_{(4,3)}(e,f)$$

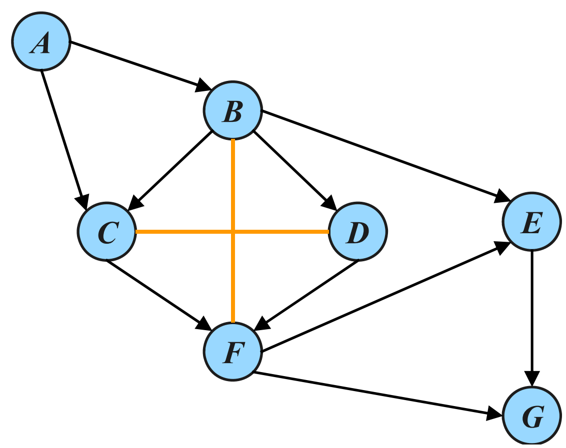
$$h_{(3,4)}(e,f) = \Downarrow_b R(e,b,f) \otimes h_{(2,3)}(b,f)$$

$$h_{(4,3)}(e,f) = \Downarrow_g R(g,e,f)$$

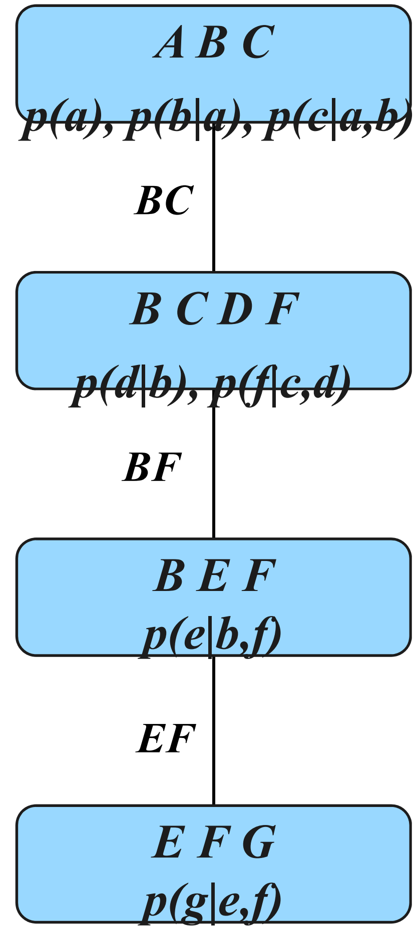
Tree decompositions

A *tree decomposition* for a belief network $BN = \langle X, D, G, P \rangle$ is a triple $\langle T, \chi, \psi \rangle$, 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)

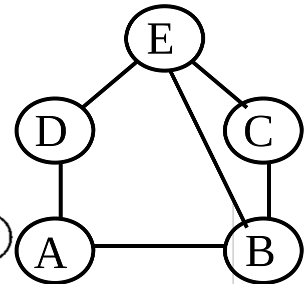


Belief network

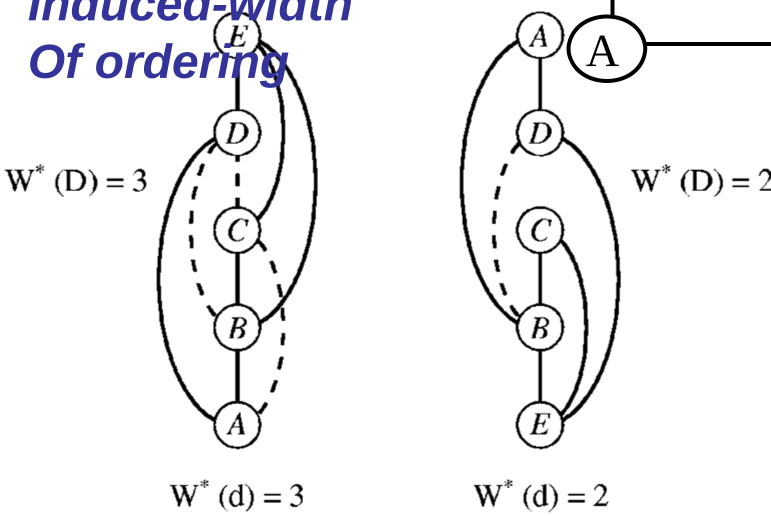


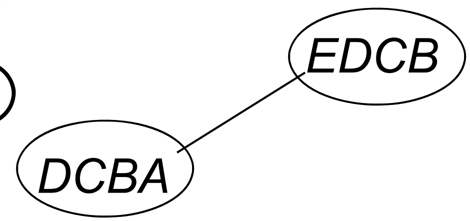
Tree decomposition

Induced-width and Tree-width

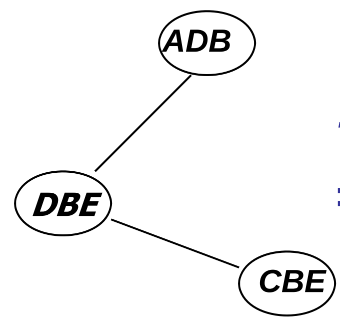


Induced-width Of ordering





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**

Adaptive Consistency, Bucket-elimination

Initialize: partition constraints into $bucket_1, ..., bucket_n$

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

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

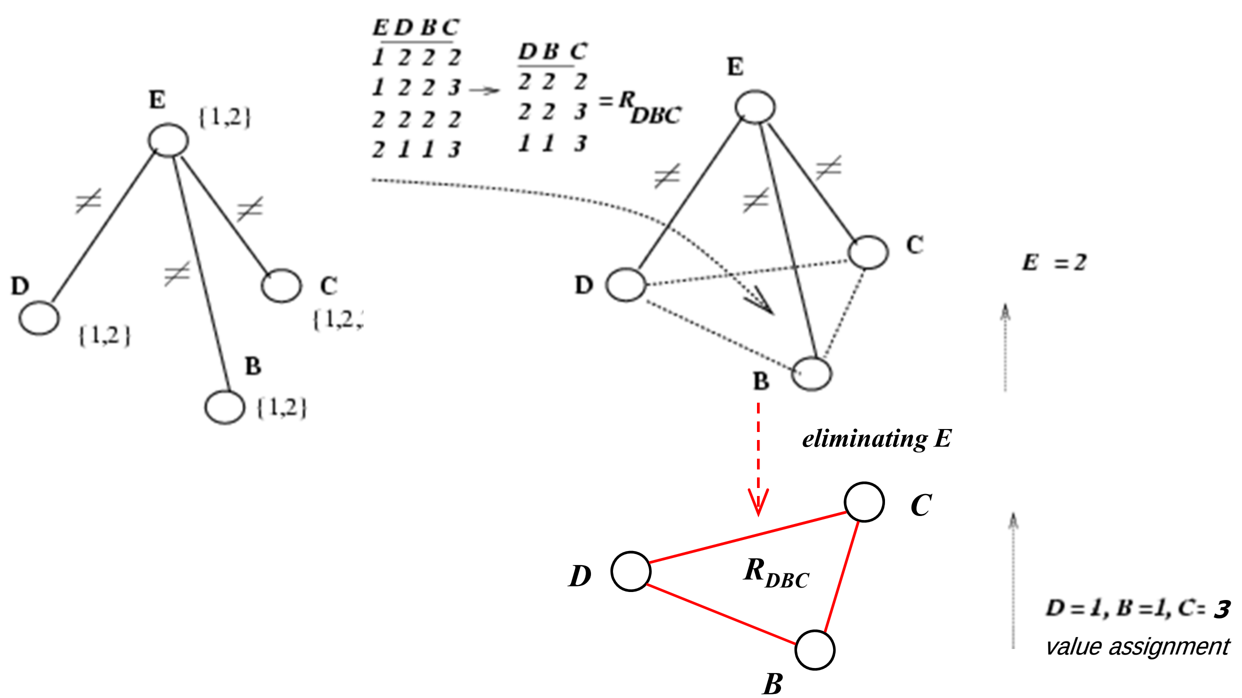
// join all relations and "project-out" X_i

$$R_{new} \leftarrow \prod_{(-X_i)} (\bowtie_j R_j)$$

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*

The Idea of Elimination



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

Eliminate variable E $\Leftrightarrow$ join and project

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 .
- Therefore, problems having bounded induced width are tractable (solved in polynomial time).
- Examples of tractable problem classes: *trees* (w*=1), *series-parallel networks* (w*=2), and in general *k-trees* (w*=k).

Road Map

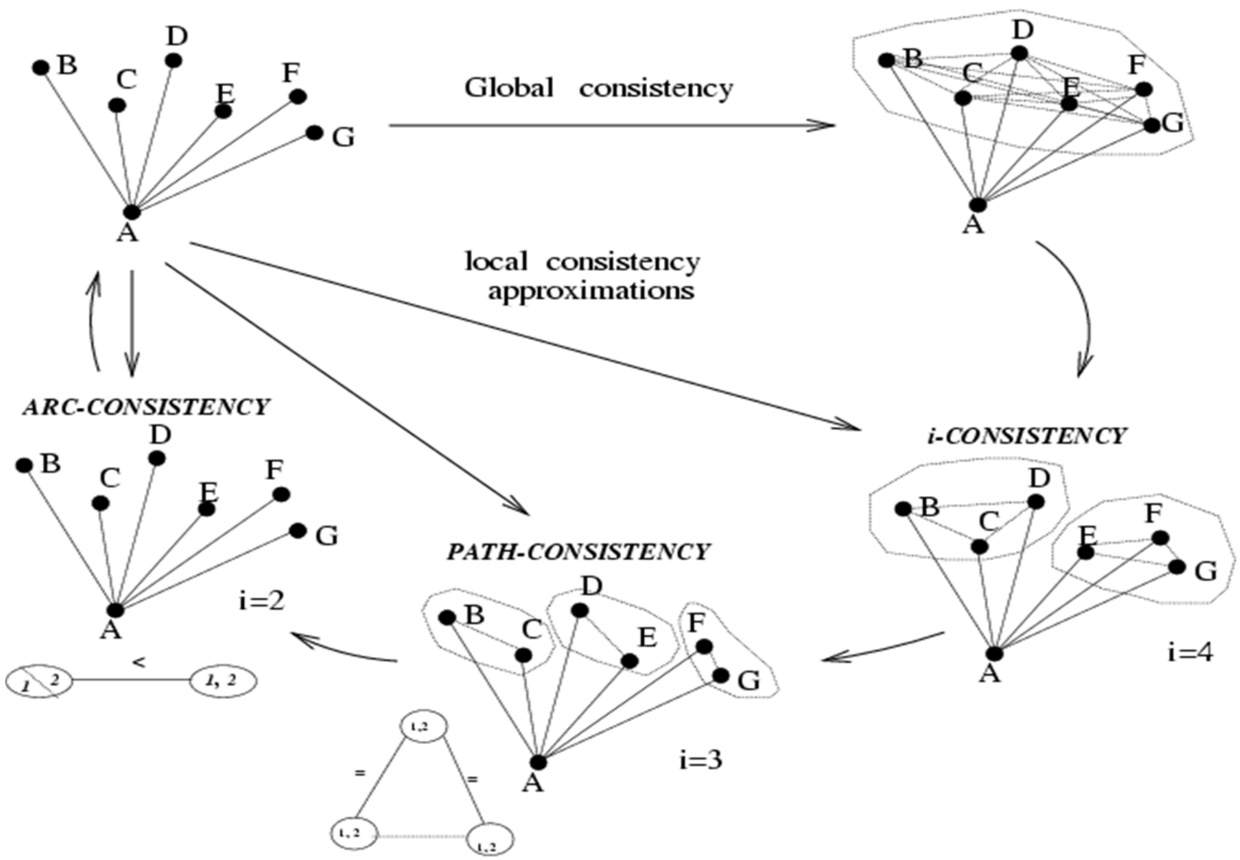
- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination:
  - Tree-clustering
  - Constraint propagation
- Search
- Probabilistic Networks

Approximating Inference: Local Constraint Propagation

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

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

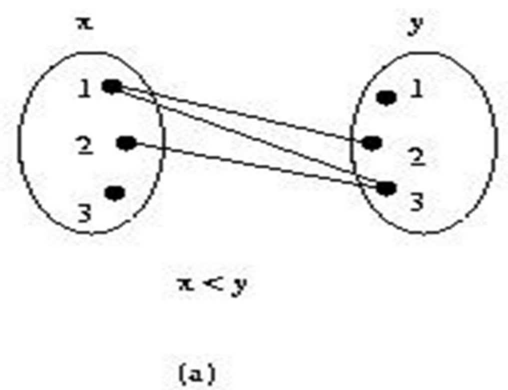
From Global to Local Consistency



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.

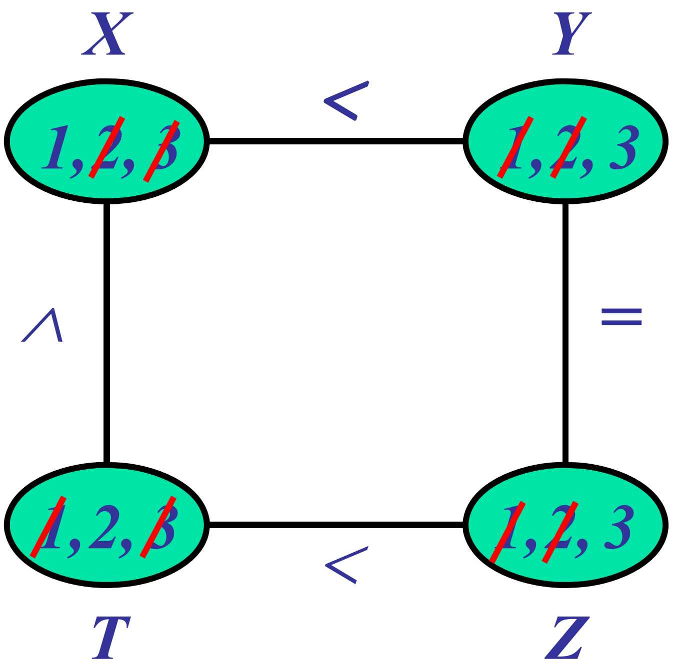
$$R_X = \{1,2,3\},\ R_Y = \{1,2,3\},\ \text{constraint}\quad X < Y$$



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

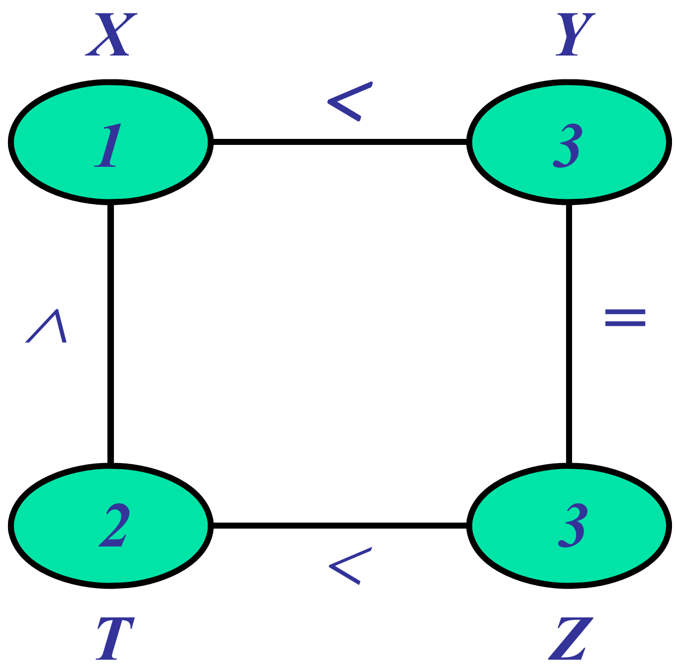
$1 \leq X, Y, Z, T \leq 3$

$X < Y$

$Y = Z$

$T < Z$

$X \leq T$



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

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 $\mathcal{R}$

for every pair $\{x_i, x_j\}$ that participates in a constraint $R_{ij} \in \mathcal{R}$
    $queue \leftarrow queue \cup \{(x_i, x_j), (x_j, x_i)\}$
endfor
while $queue \neq \{\}$
    select and delete (x_i, x_j) from *queue*
    $Revise((x_i), x_j)$
    if $Revise((x_i), x_j)$ causes a change in D_i
        then $queue \leftarrow queue \cup \{(x_k, x_i), i \neq k\}$
    endif
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 number 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.

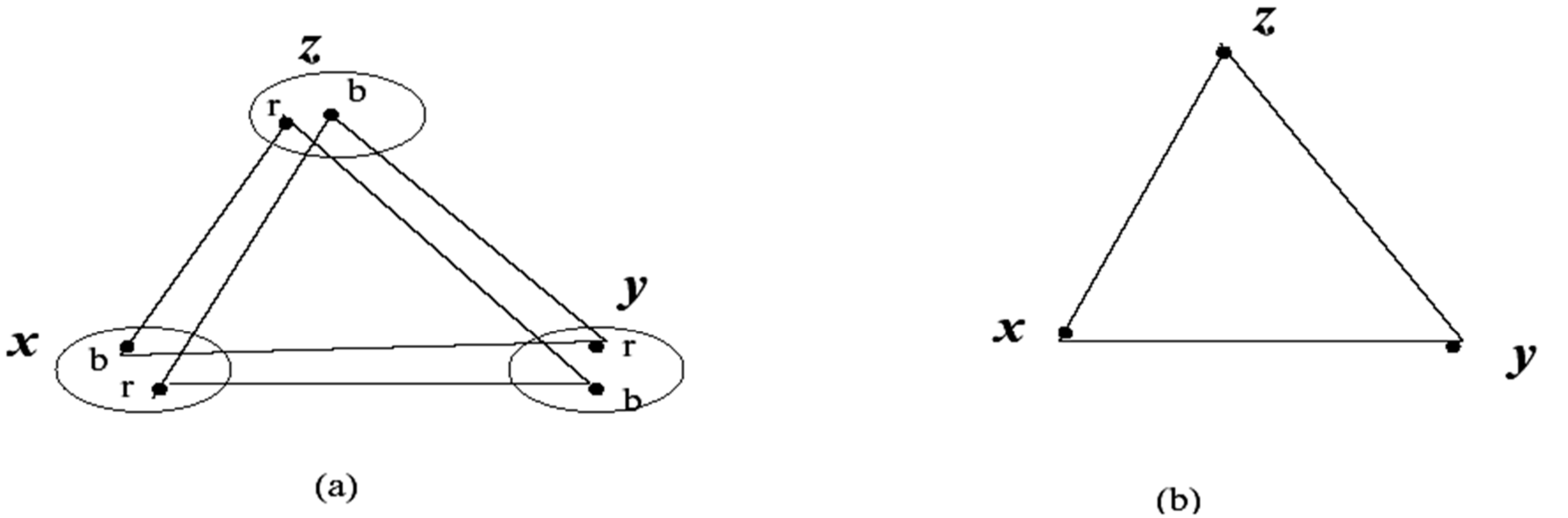


Figure 3.8: (a) The matching diagram of a 2-value graph coloring problem. (b) Graphical picture of path-consistency using the matching diagram.

Example: path-consistency

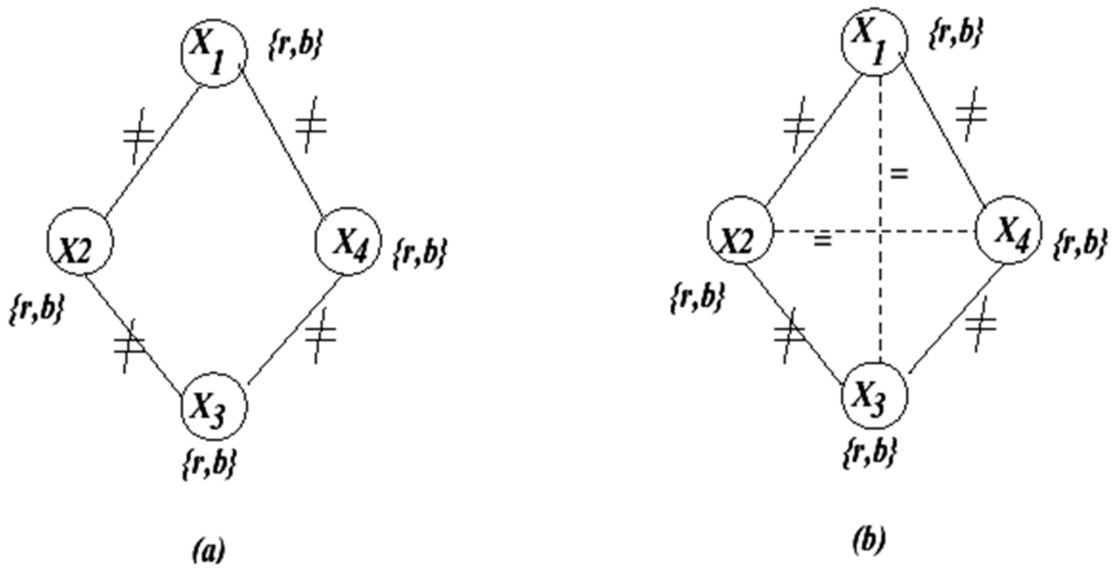


Figure 3.12: A graph-coloring graph (a) before path-consistency (b) after path-consistency

Path-consistency Algorithms

- Apply Revise-3 (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

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

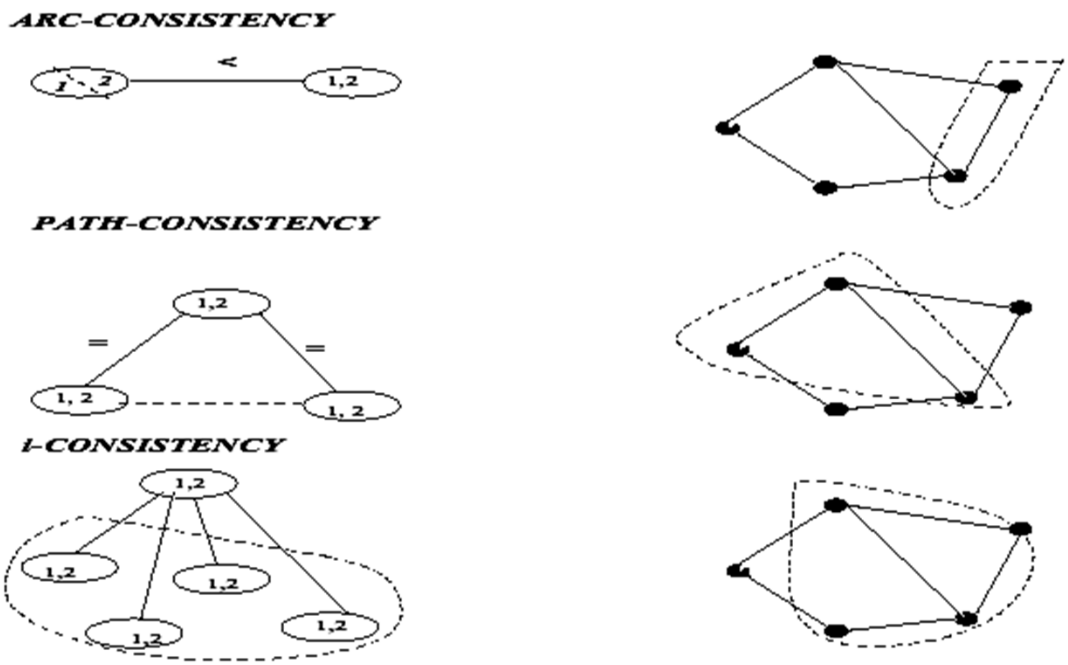


Figure 3.17: The scope of consistency enforcing: (a) arc-consistency, (b) path-consistency, (c) i-consistency

Gausian and Boolean Propagation, Resolution

- Linear inequalities

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

$$x \leq 2, y \leq 2$$

- Boolean constraint propagation, unit resolution

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

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

Boolean Constraint Propagation

Is *propositional theory*

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

A is not arc-consistent relative to B

Enforce arc-consistency by resolution:

$\text{res}(\neg A \vee B, \neg B) \Rightarrow \neg A$

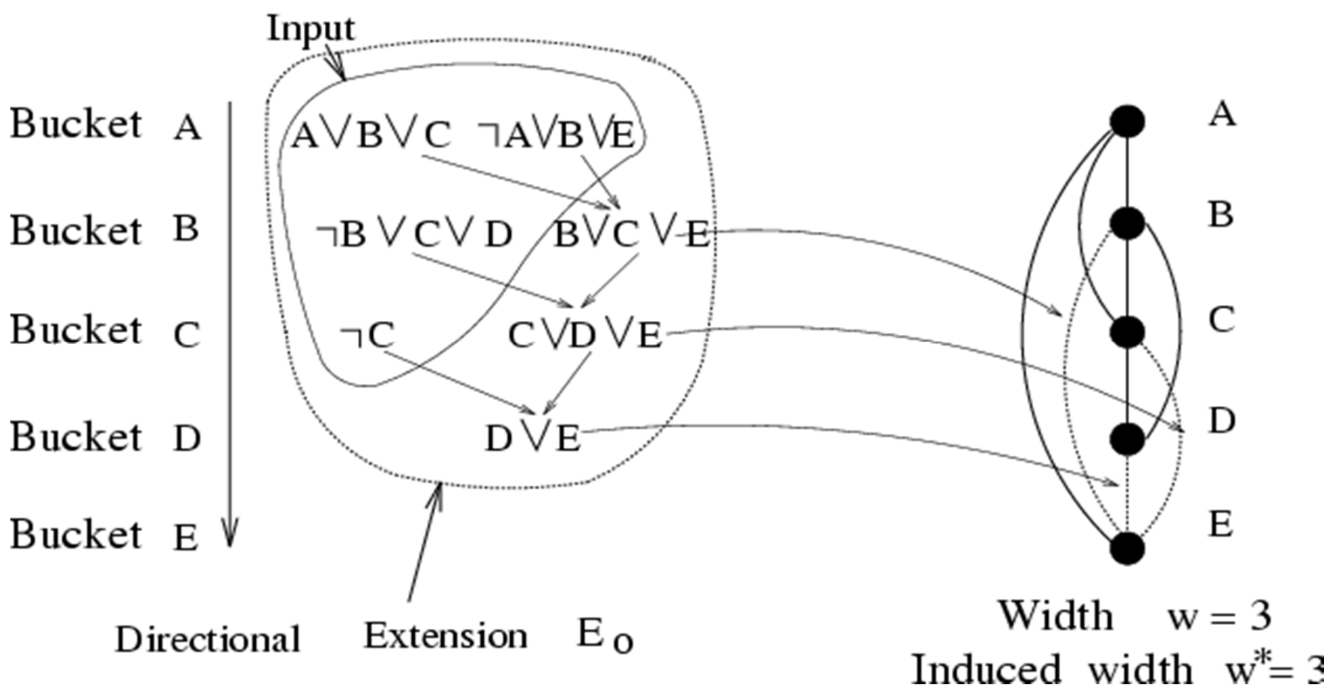
$\text{res}(\neg C \vee A, C) \Rightarrow A$

$\text{res}\,(A, \neg A) \Rightarrow \Phi$

Given also (B V C), path-consistency:
Res((A V ~B),(B V C)) = (A V C)

Relational arc-consistency rule = unit-resolution

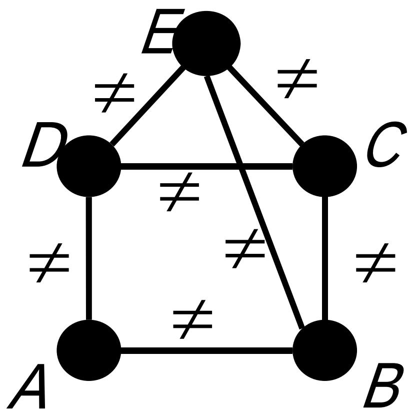
Directional Resolution ⇔ Adaptive Consistency

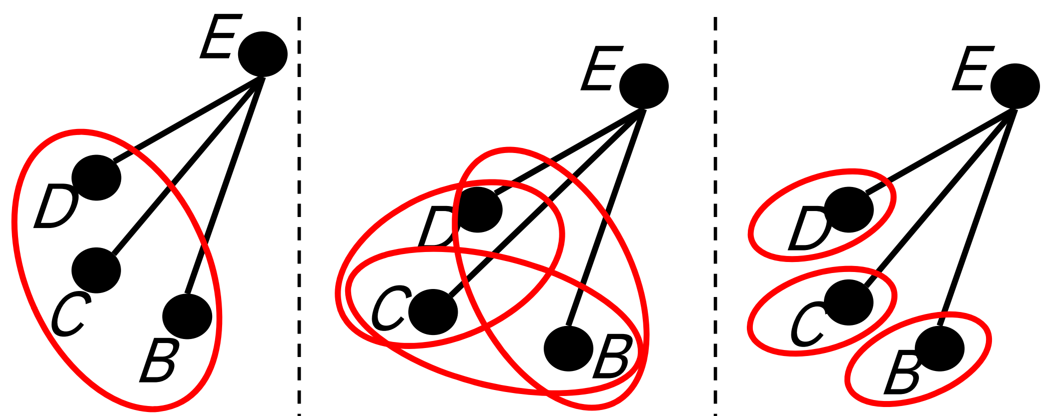


$$|\, bucket_i \,| = O(\exp(w^*))$$

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

Directional i-consistency

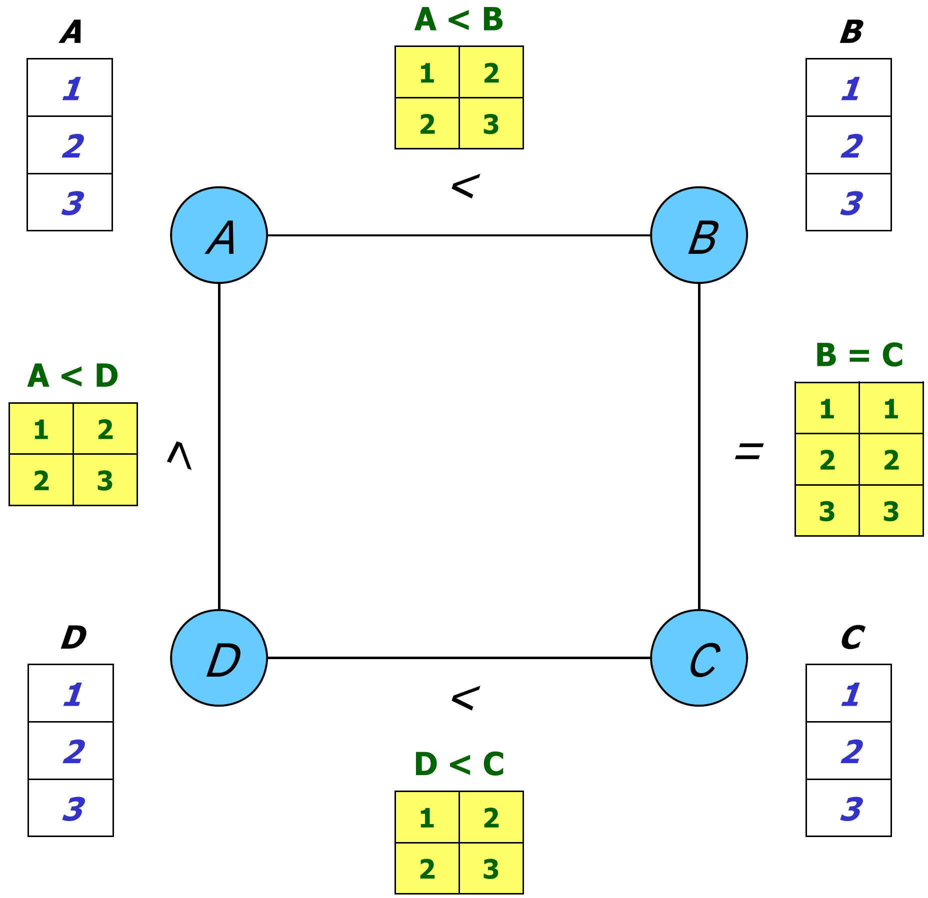




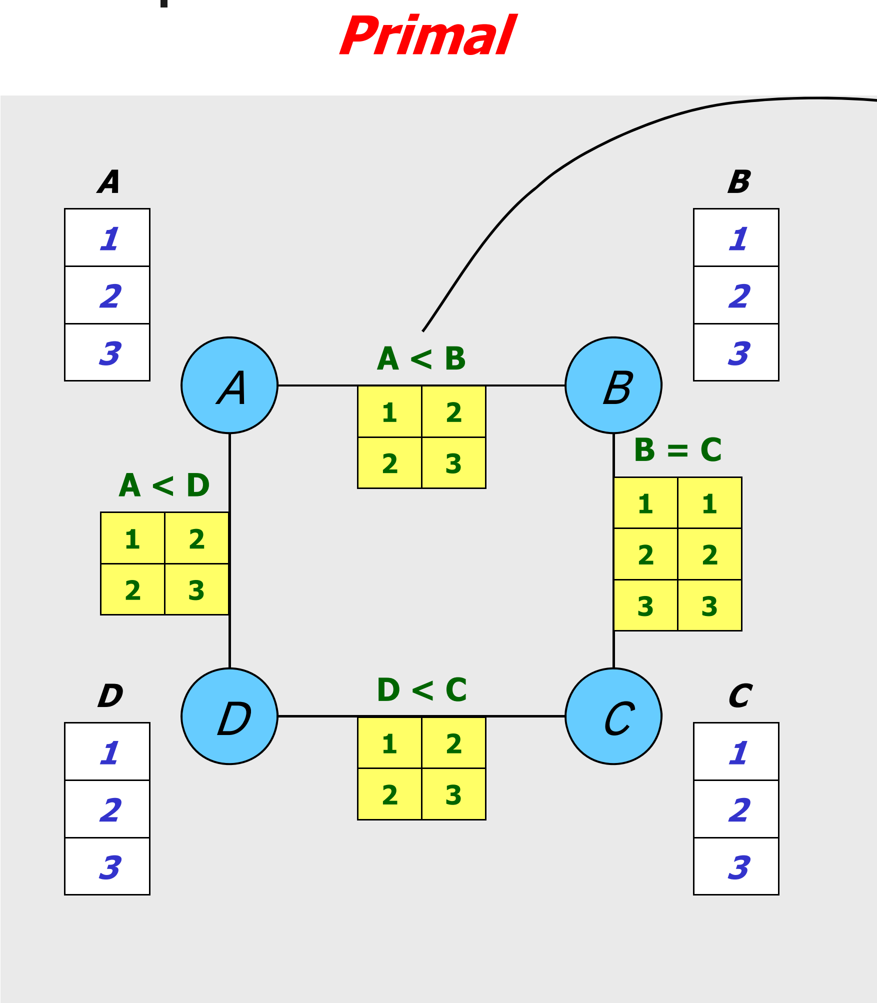
	Adaptive	*d-path*	*d-arc*
E : E $\neq$ D, E $\neq$ C, E $\neq$ B	R_{DCB}	R_{DC}, R_{DB}	R_D
D : D $\neq$ C, D $\neq$ A		R_{CB}	R_C
C : C $\neq$ B			R_D
B : A $\neq$ B			
A :			

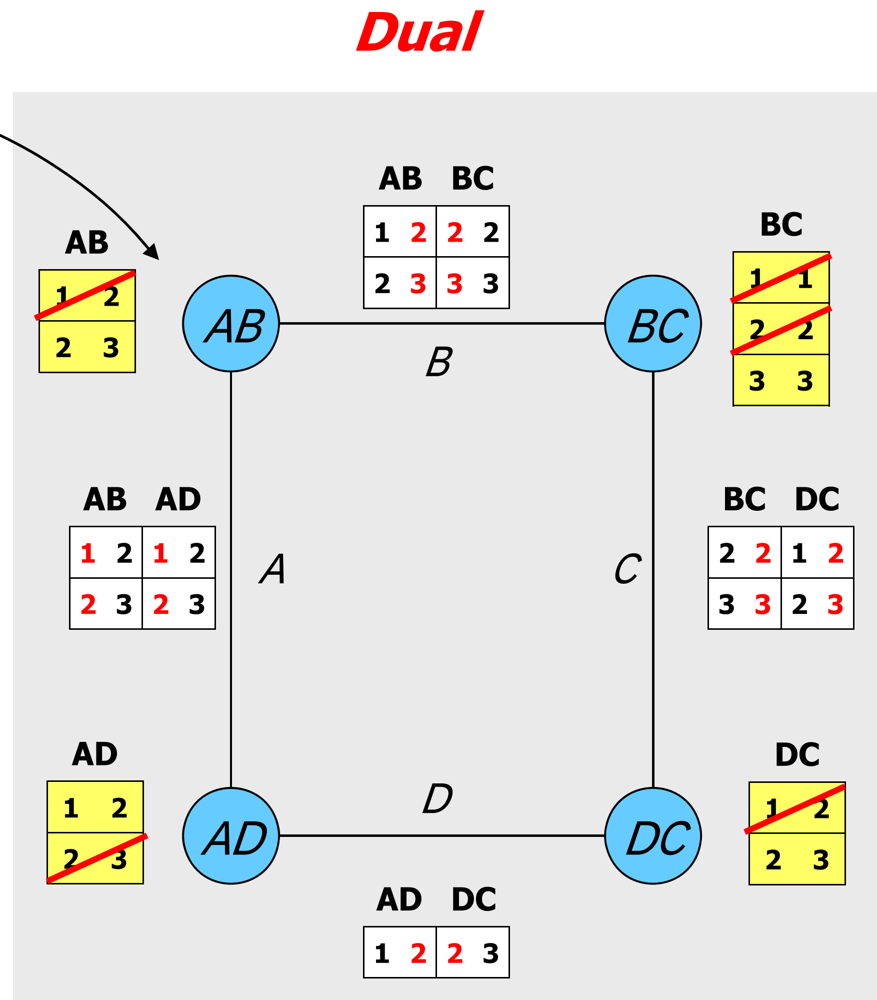
Arc-consistency

- Sound
- Incomplete
- Always converges (polynomial)



Relational Distributed Arc-Consistency





Road Map

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

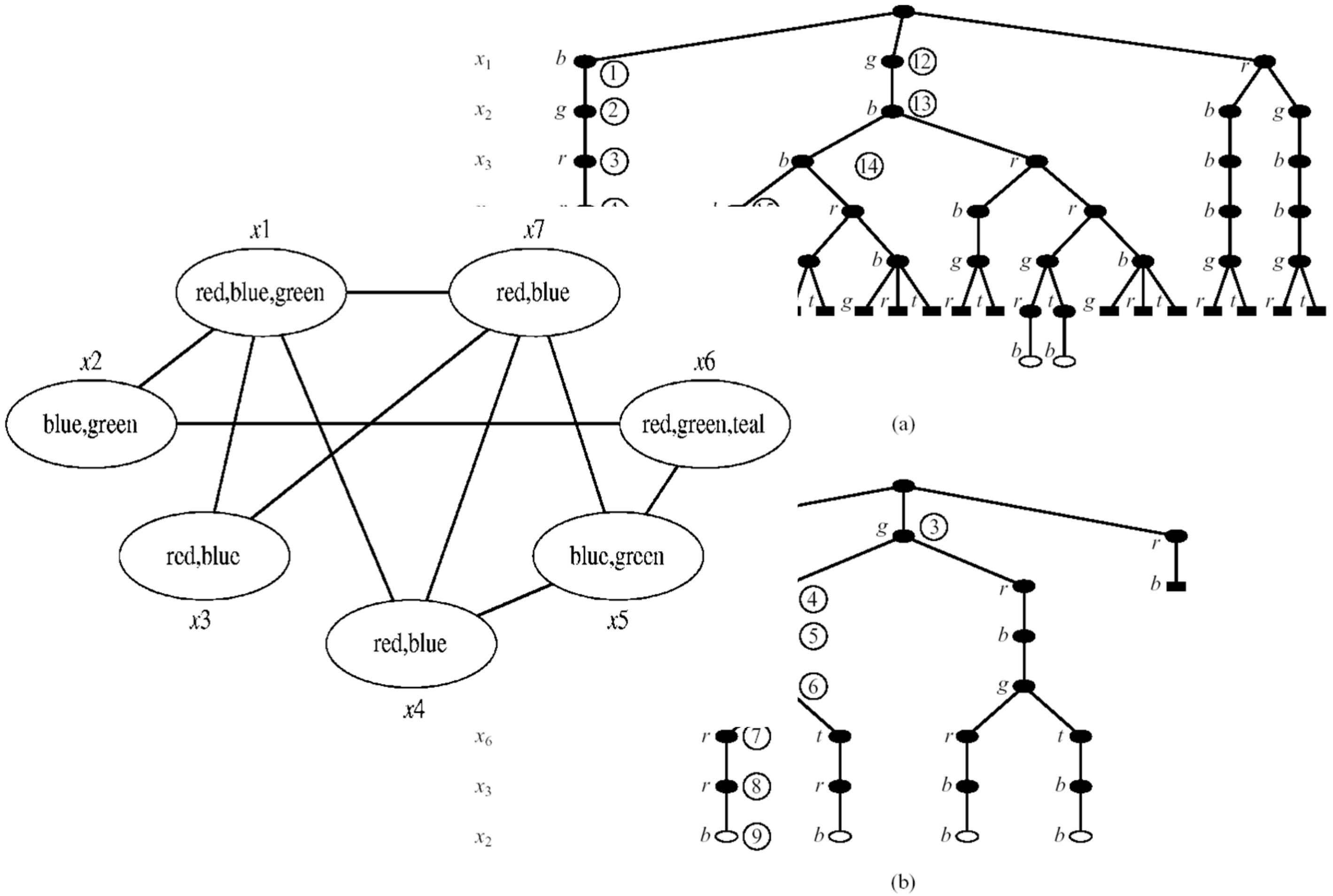
Road Map:
Search in Graphical Models

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

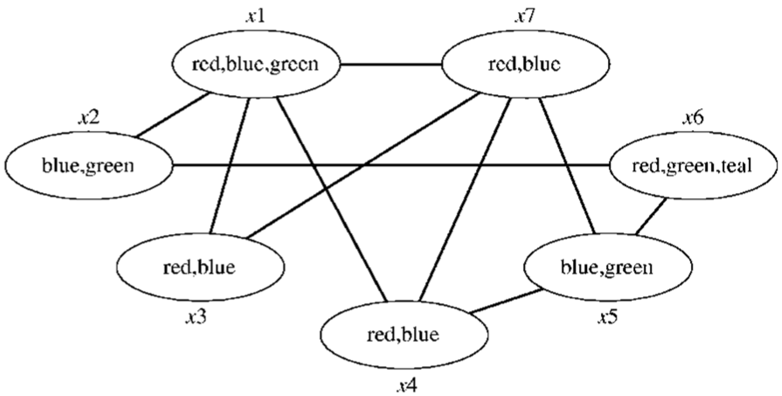
Road Map: Search in Graphical Models

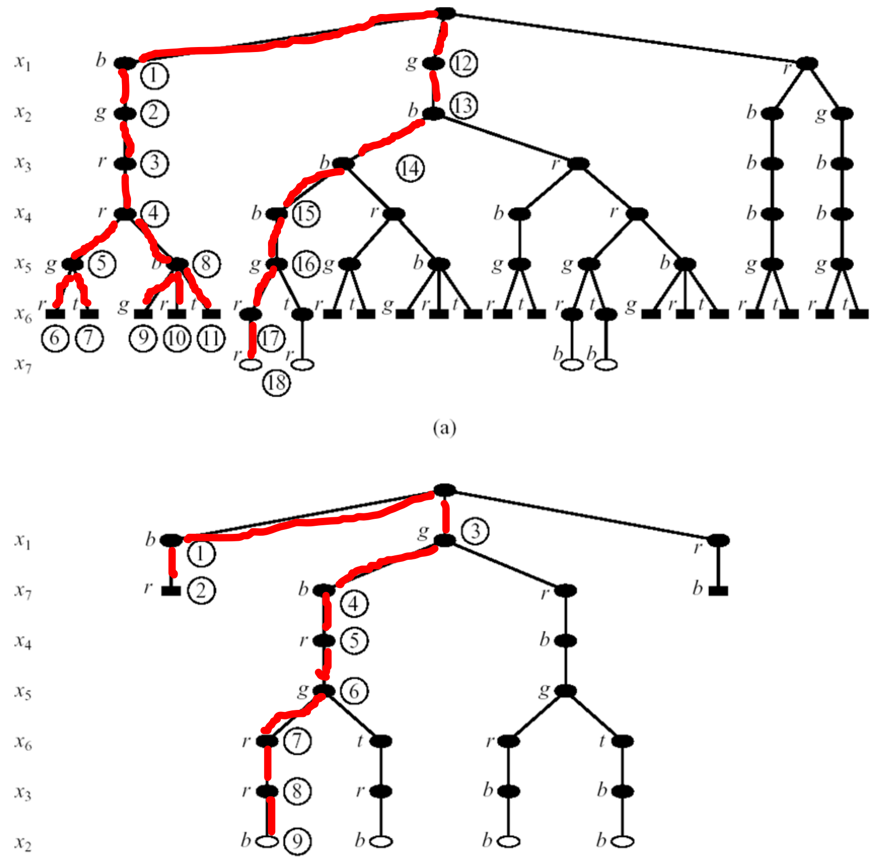
- Improving search by bounded-inference in branching ahead
- Improving search by looking-back
- The alternative AND/OR search space

Backtracking Search for a Solution



Backtracking Search for a Solution

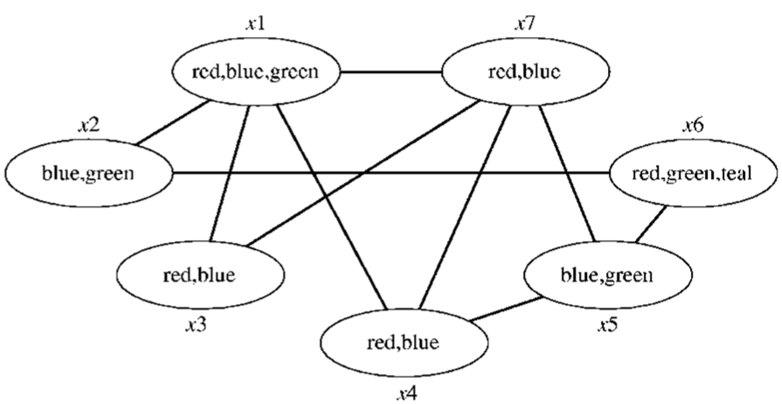


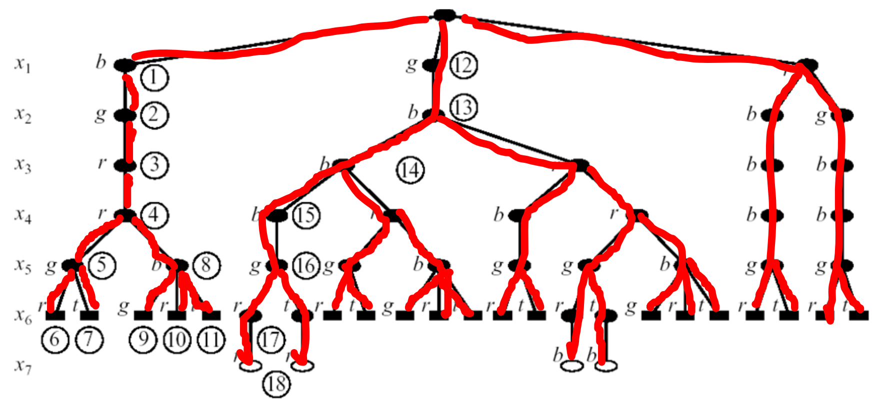


(a)

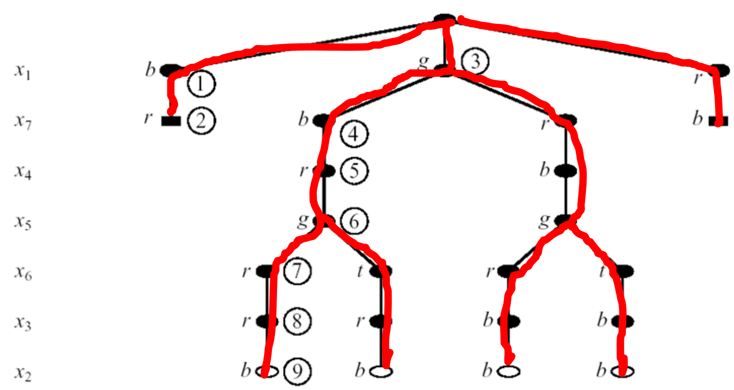
(b)

Backtracking Search for All Solutions



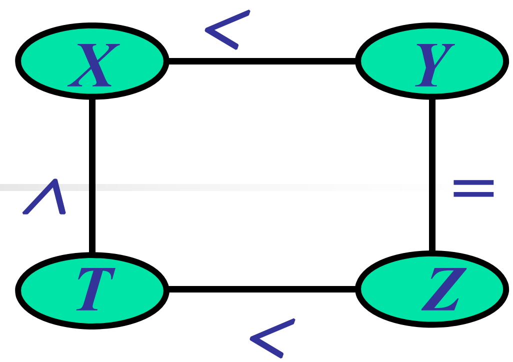


(a)

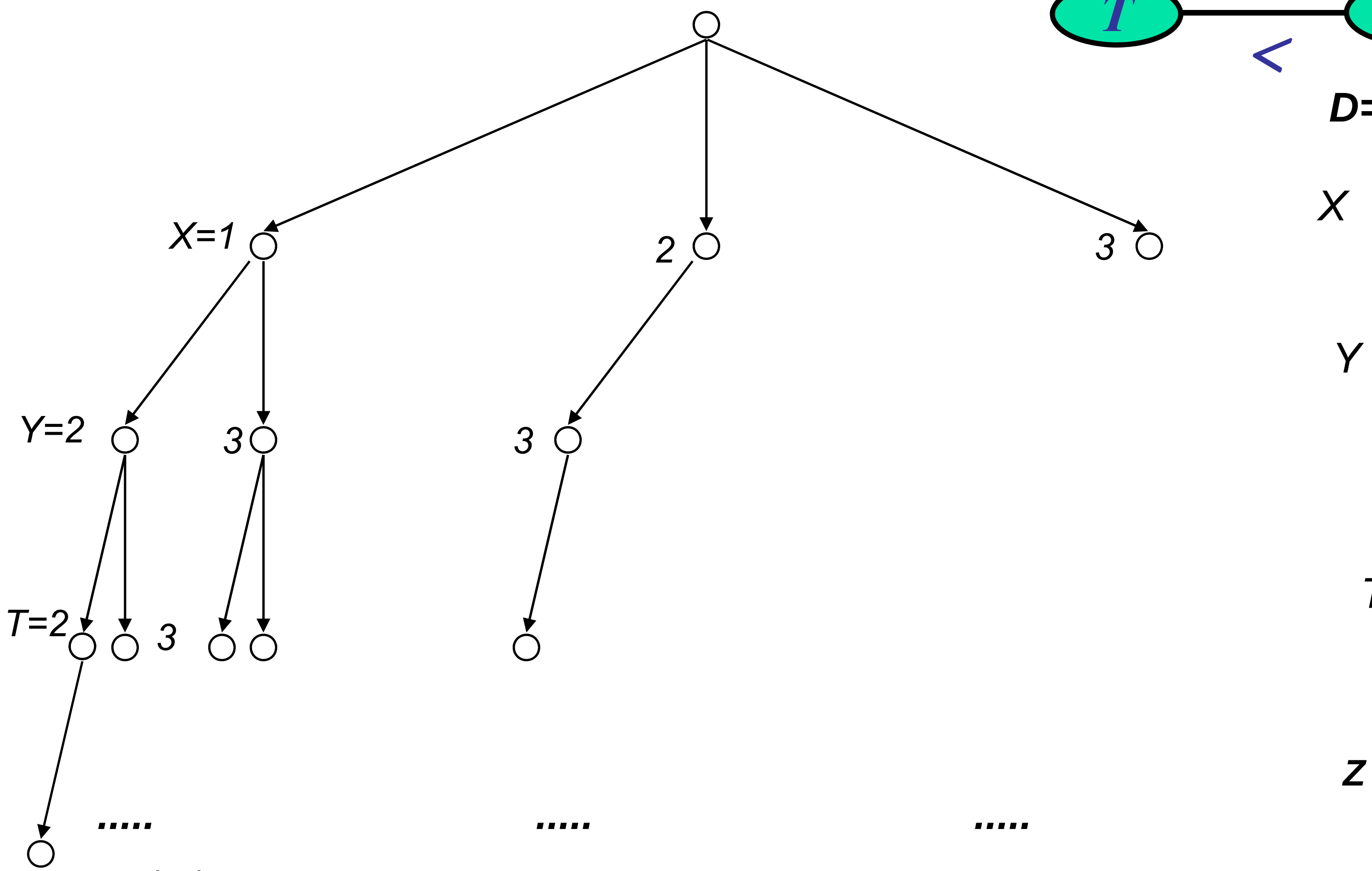


(b)

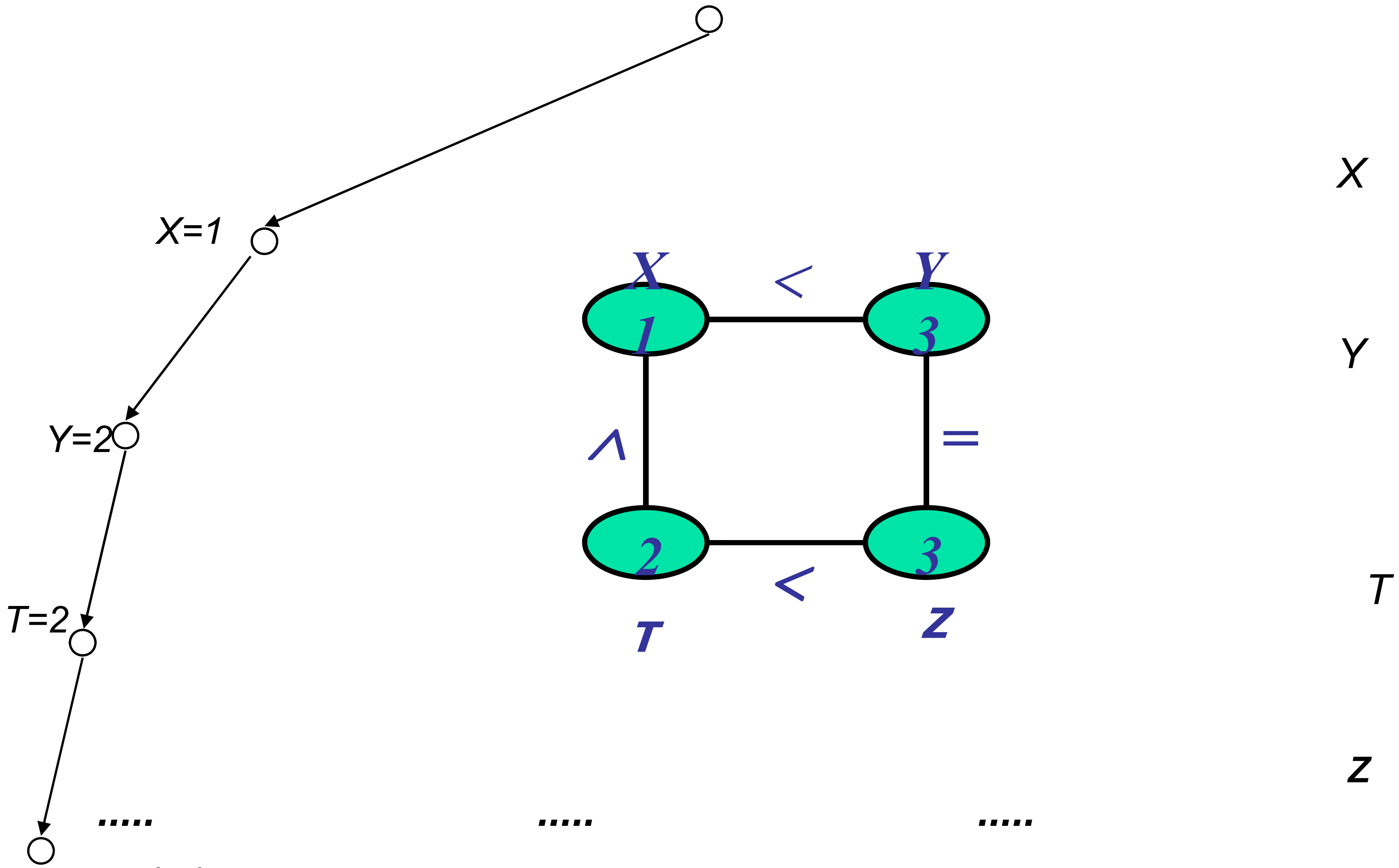
The Search Space Before Arc-Consistency



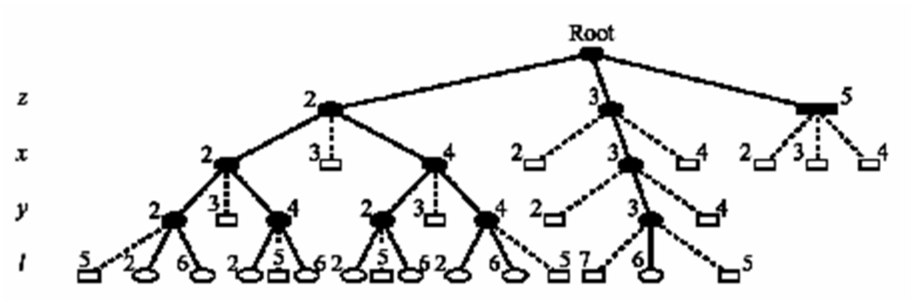
D={1,2,3}



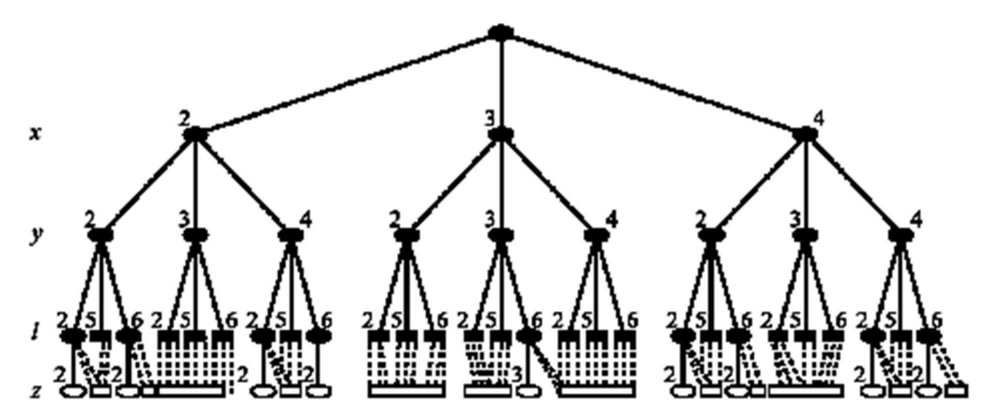
The Search Space After Arc-Consistency



The Effect of Variable Ordering

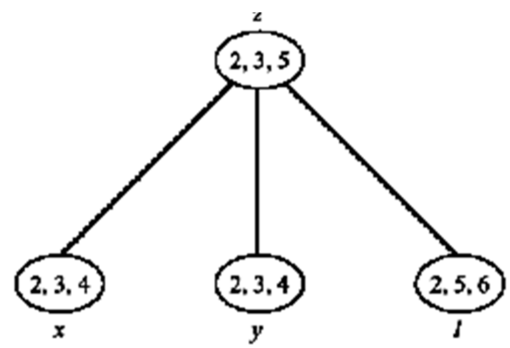


(b)



(c)

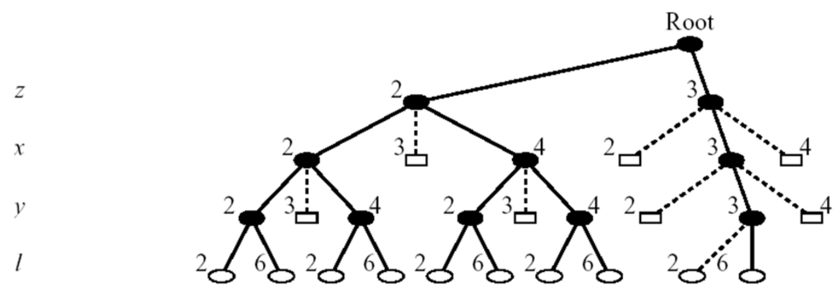
z divides x, y and t



(a)

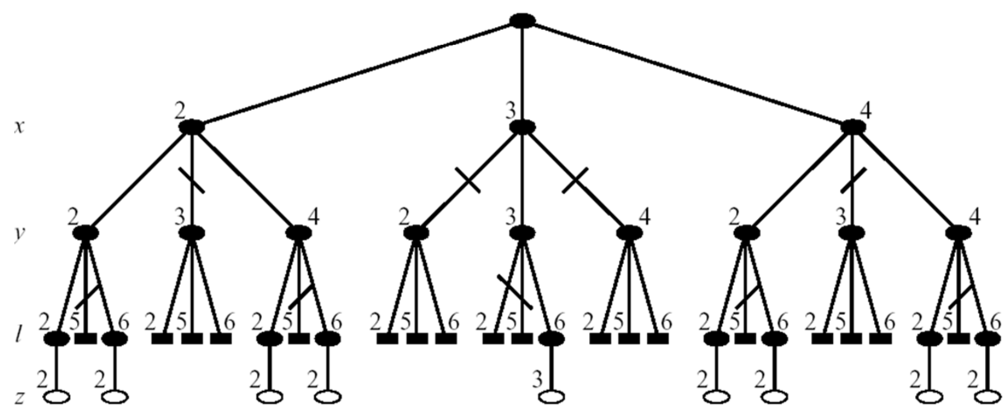
The Effect of Consistency Level

- After arc-consistency z=5 and l=5 are removed



(a)

- After path-consistency
  - R'_zx
  - R'_zy
  - R'_zl
  - R'_xy
  - R'_xl
  - R'_yl



(b)

Tighter networks yield smaller search spaces

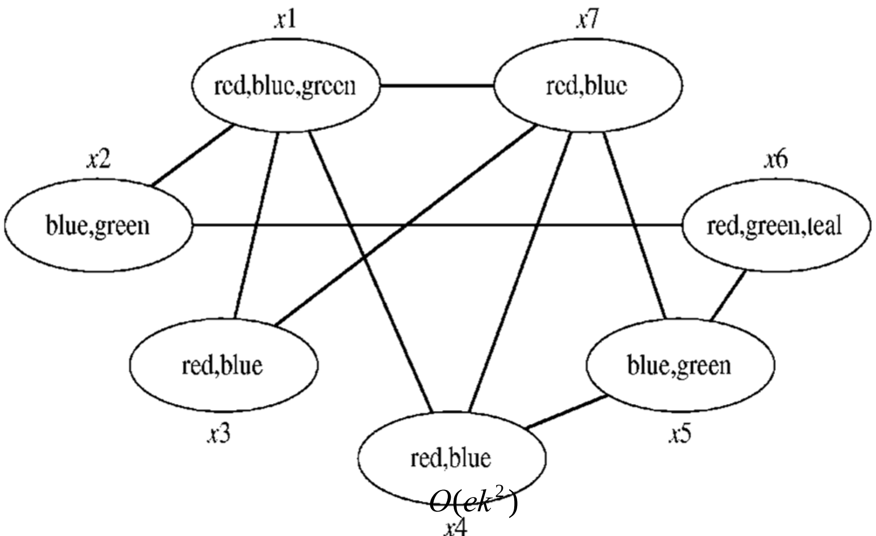
Improving Backtracking O(exp(n))

- 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

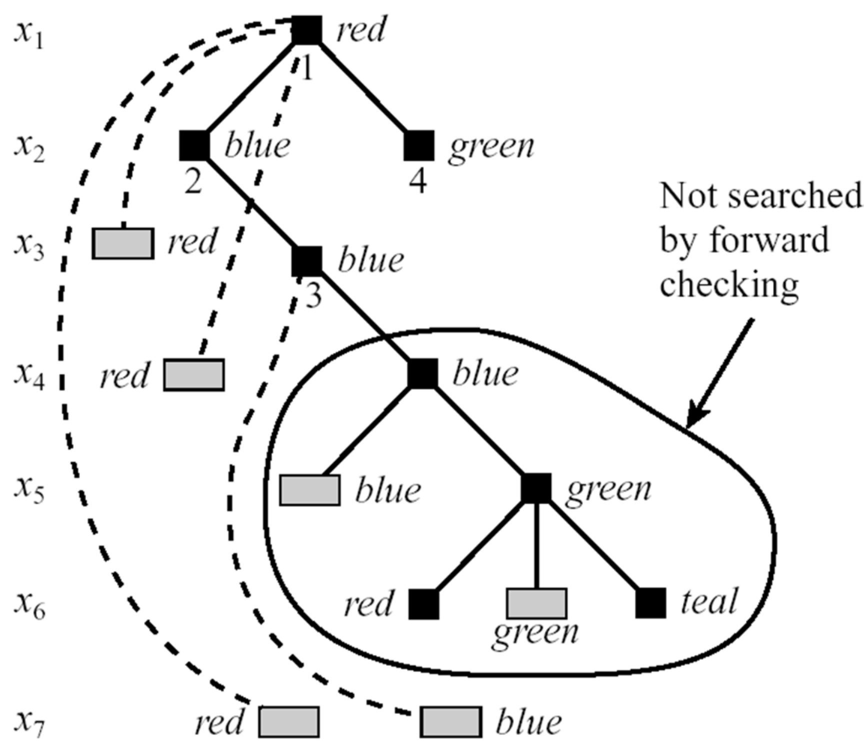
- Apply some level of constraint propagation at each node,
  - Forward-checking (FC)
  - Arc-consistency (MAC)
- Then:
  - Value pruning or ordering : prune values that lead to deadend
  - Variable ordering: choose a variable that leaves **least** options open

Forward-Checking for Value Ordering

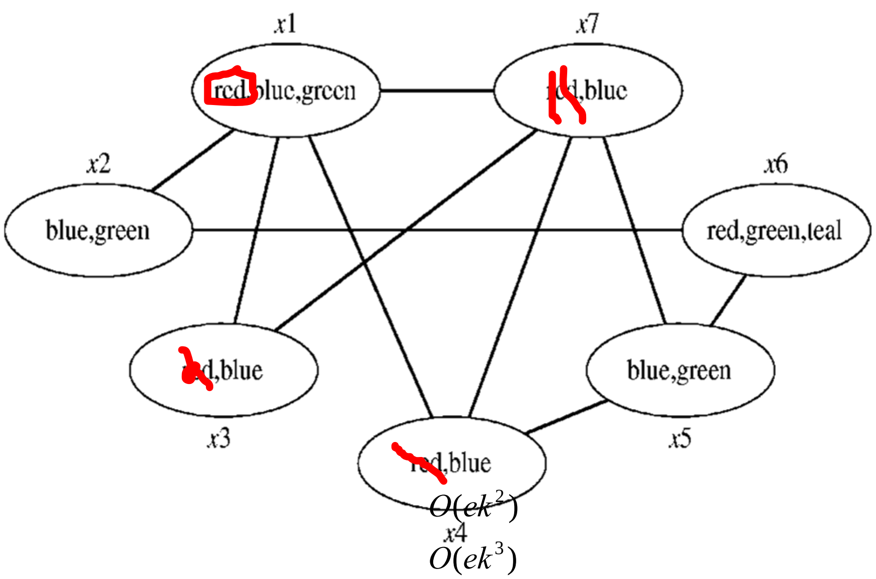


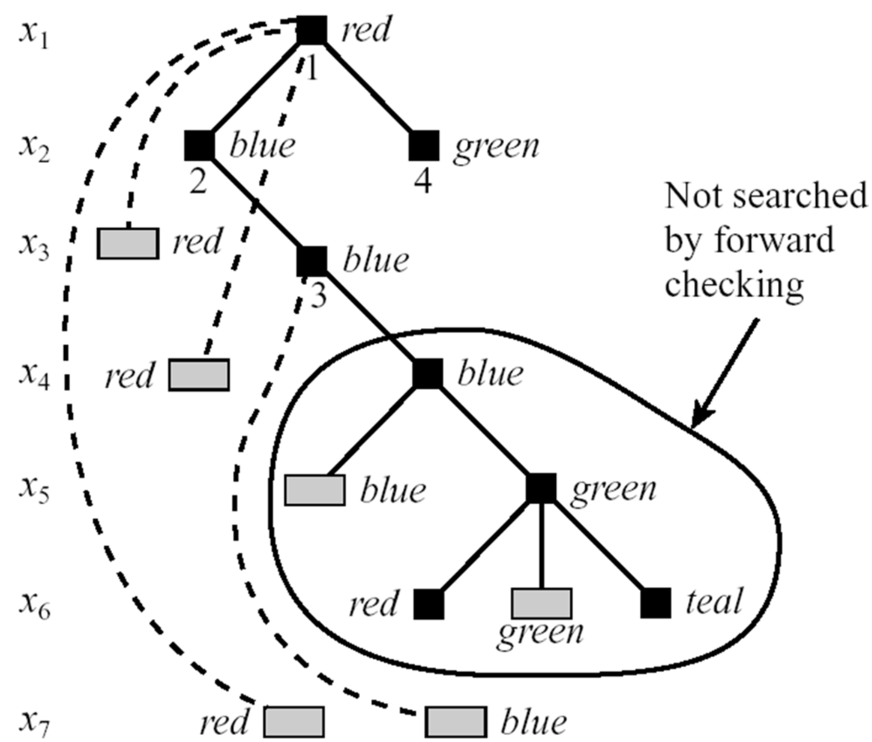
FC overhead: $O(ek^2)$

MAC overhead: $O(ek^3)$



Forward-Checking for Value Ordering

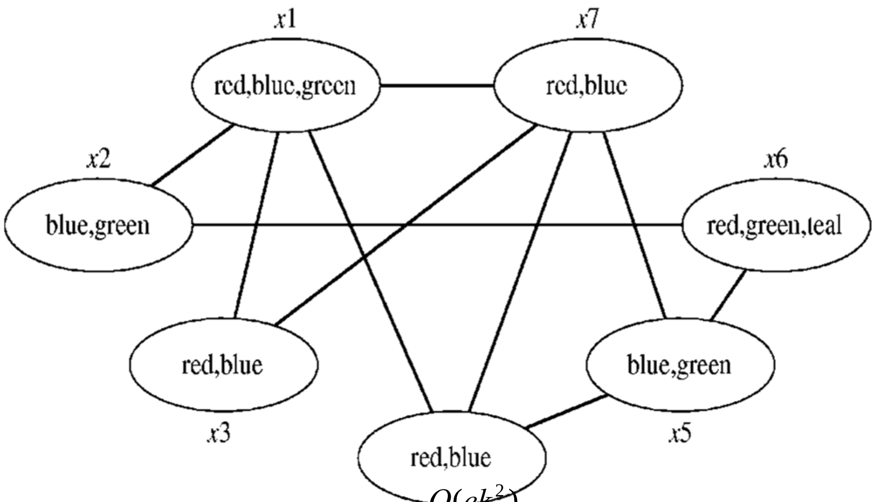




FW overhead:

MAC overhead:

Forward-Checking, Variable Ordering

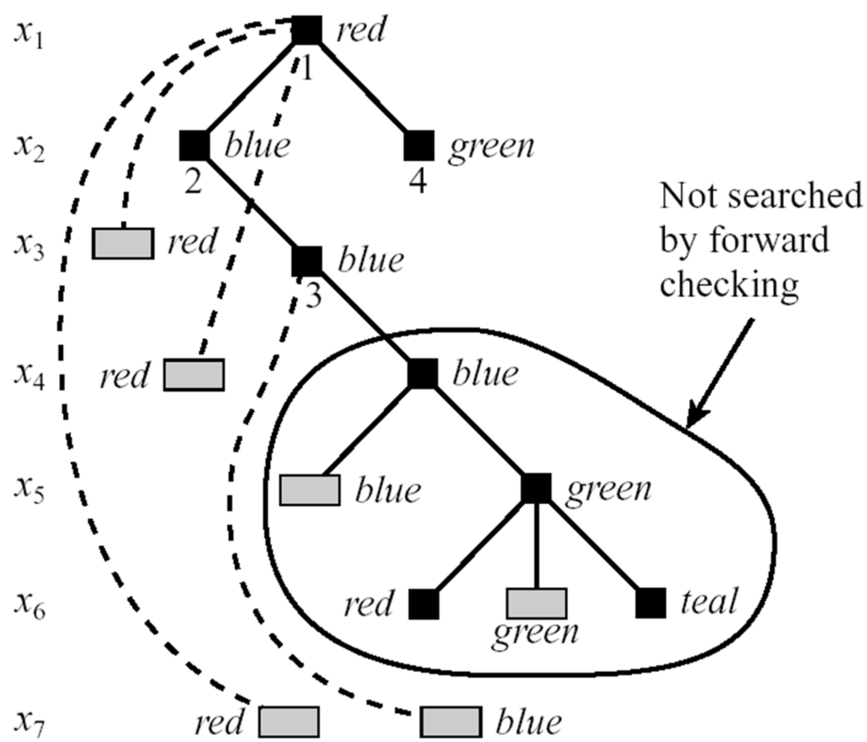


$O(ek^2)$

$O(ek^3)$

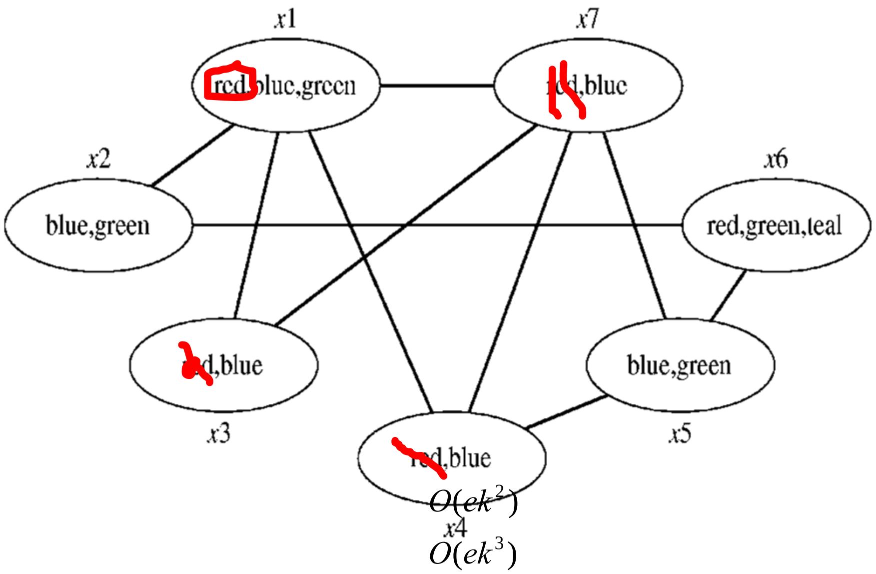
FW overhead:

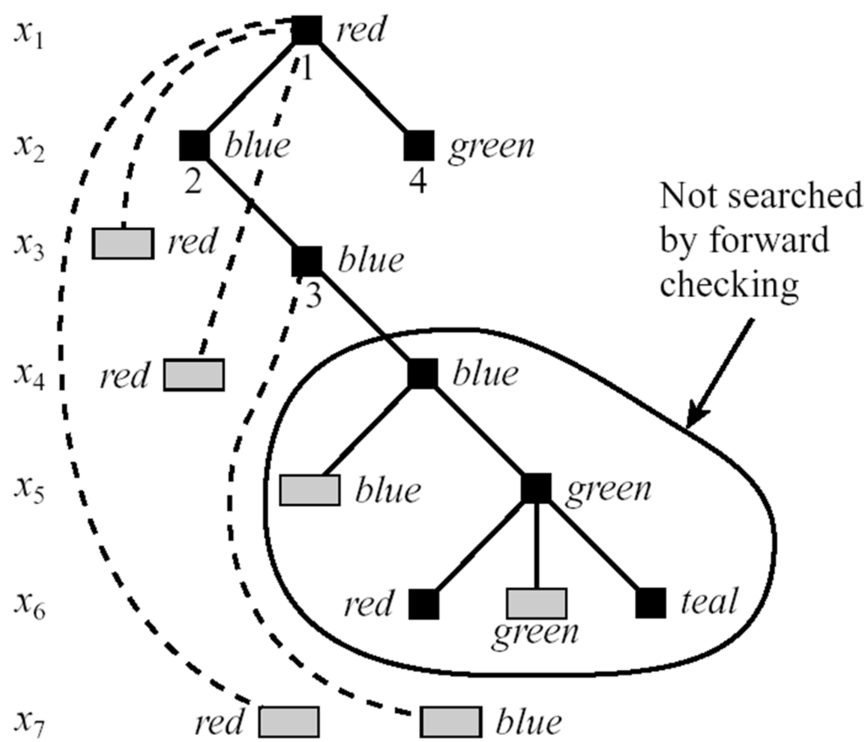
MAC overhead:



Forward-Checking, Variable Ordering

After X1 = red choose X3 and not X2



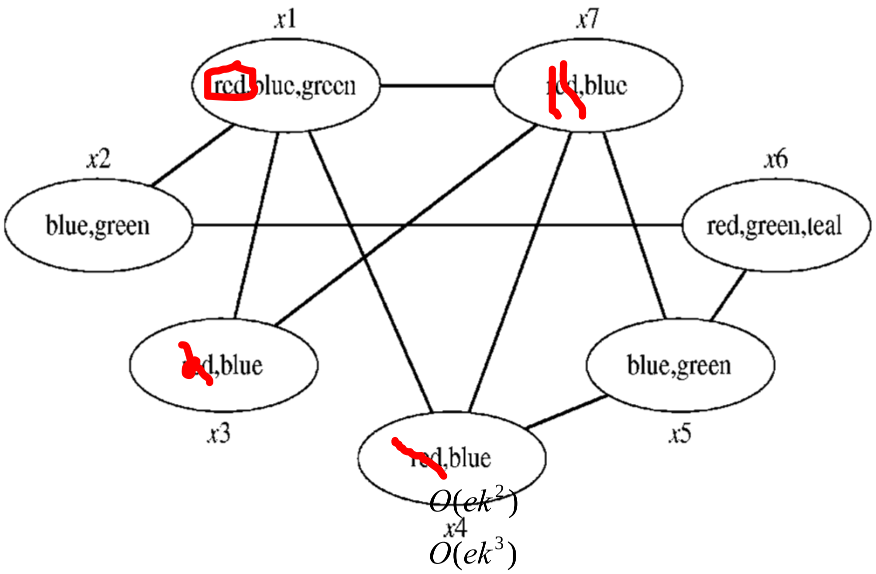


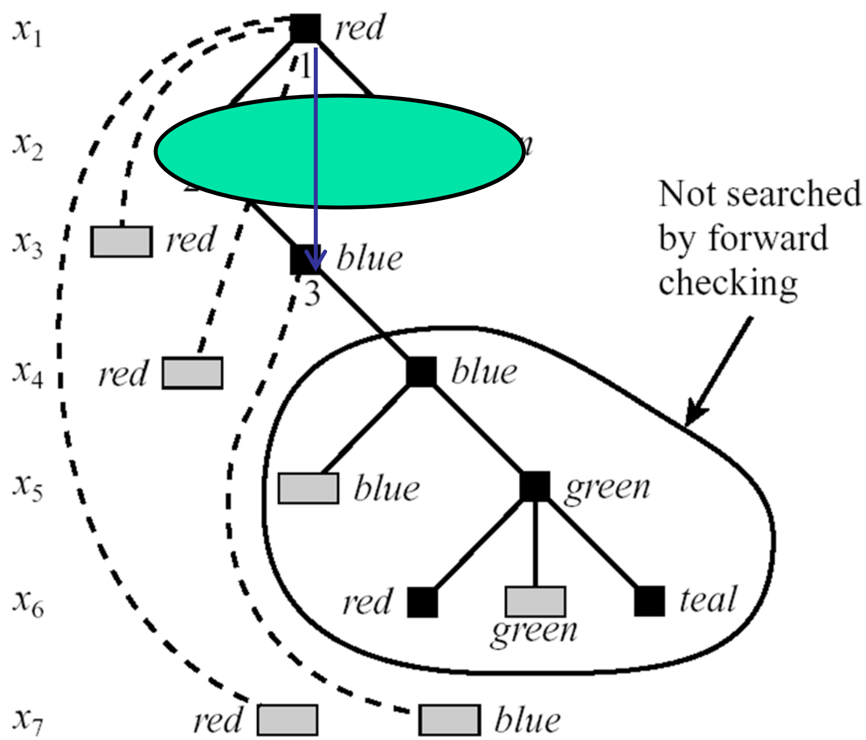
FW overhead:

MAC overhead:

Forward-Checking, Variable Ordering

After X1 = red choose X3 and not X2



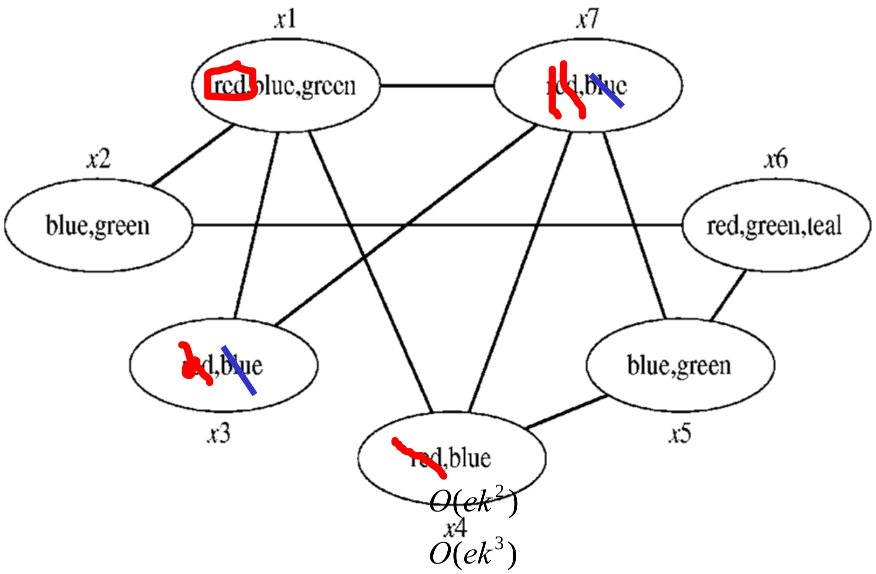


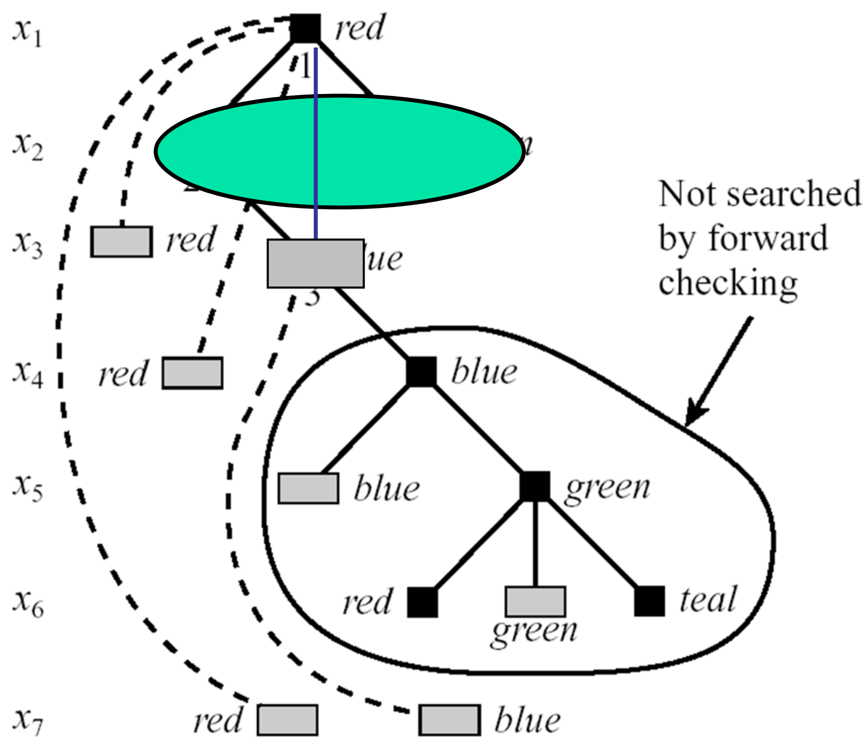
FW overhead:

MAC overhead:

Forward-Checking, Variable Ordering

After X1 = red choose X3 and not X2

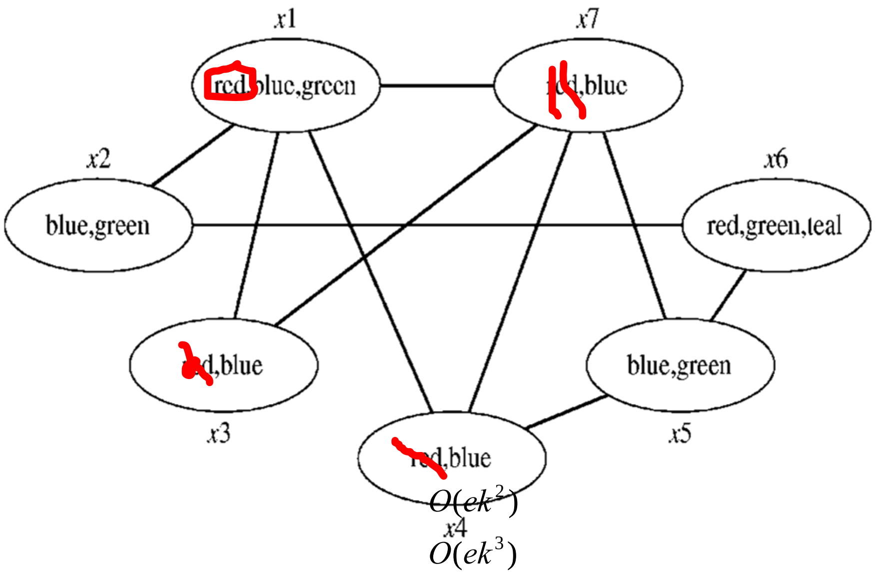


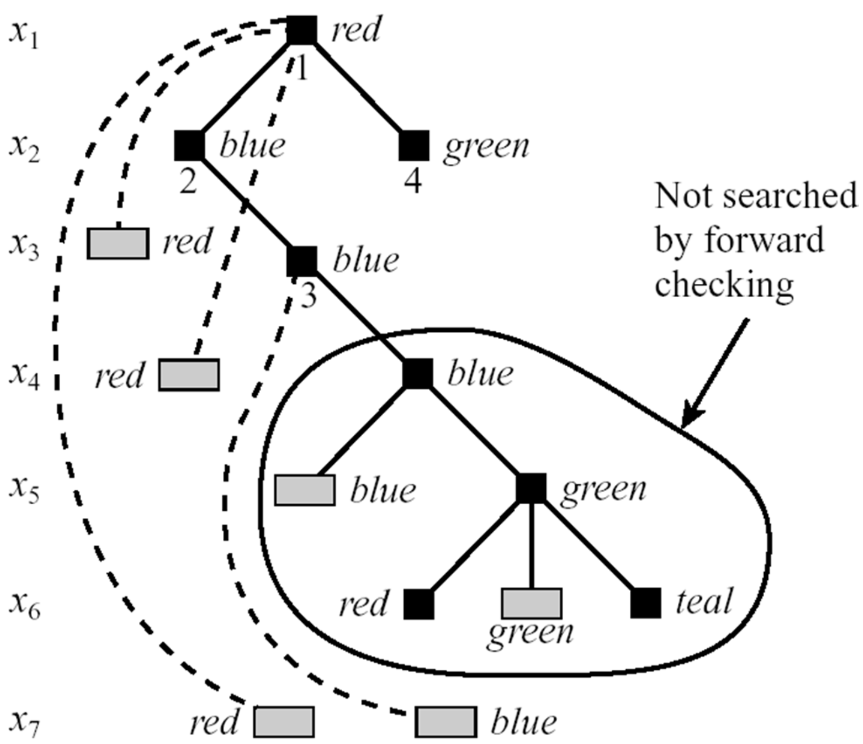


FW overhead:

MAC overhead:

Arc-consistency for Value Ordering



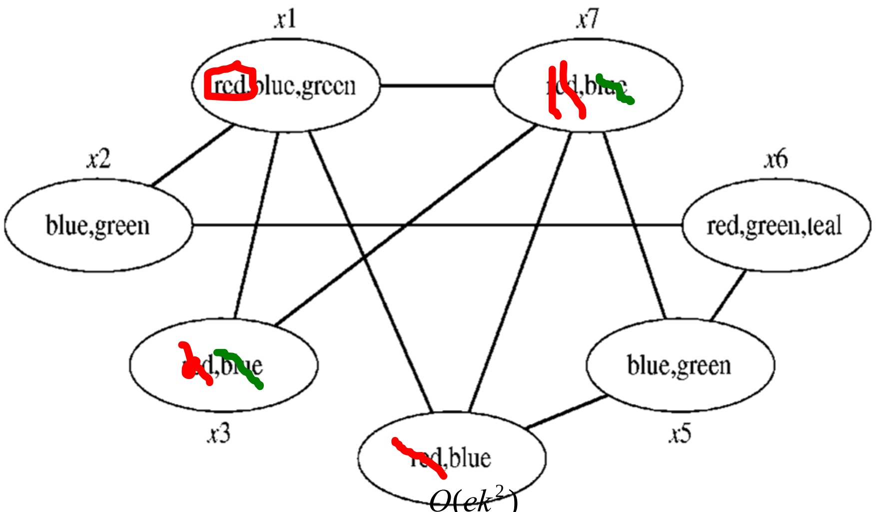


FW overhead:

MAC overhead:

Arc-Consistency for Value Ordering

Arc-consistency prunes x1=red
Prunes the whole tree

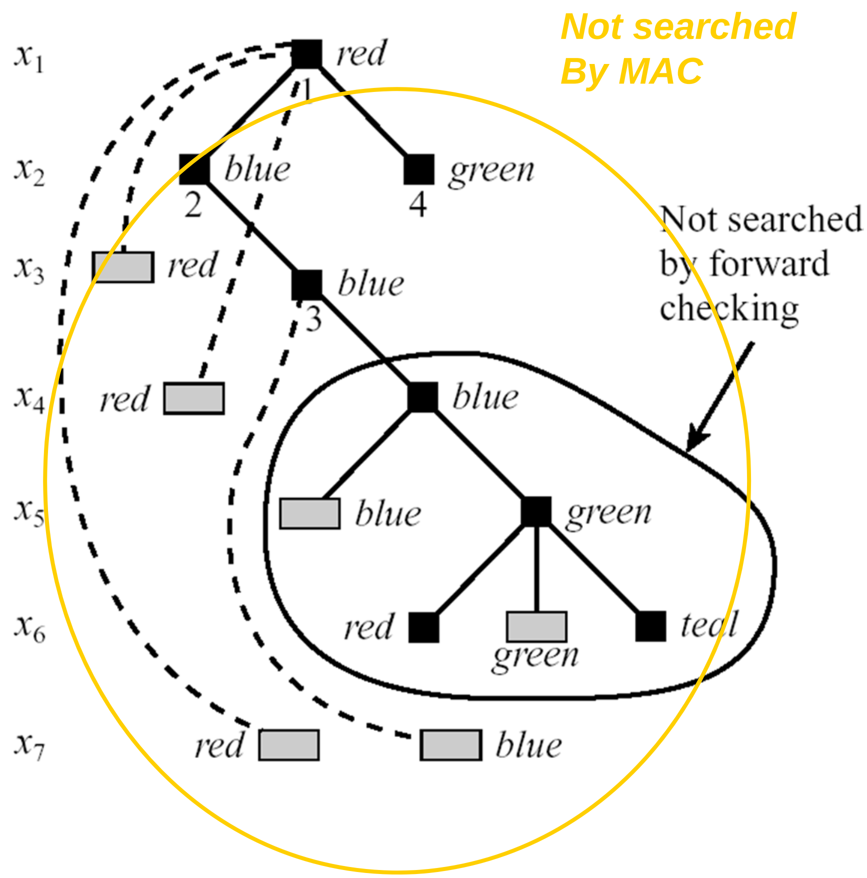


$O(ek^2)$

$O(ek^3)$

FW overhead:

MAC overhead:



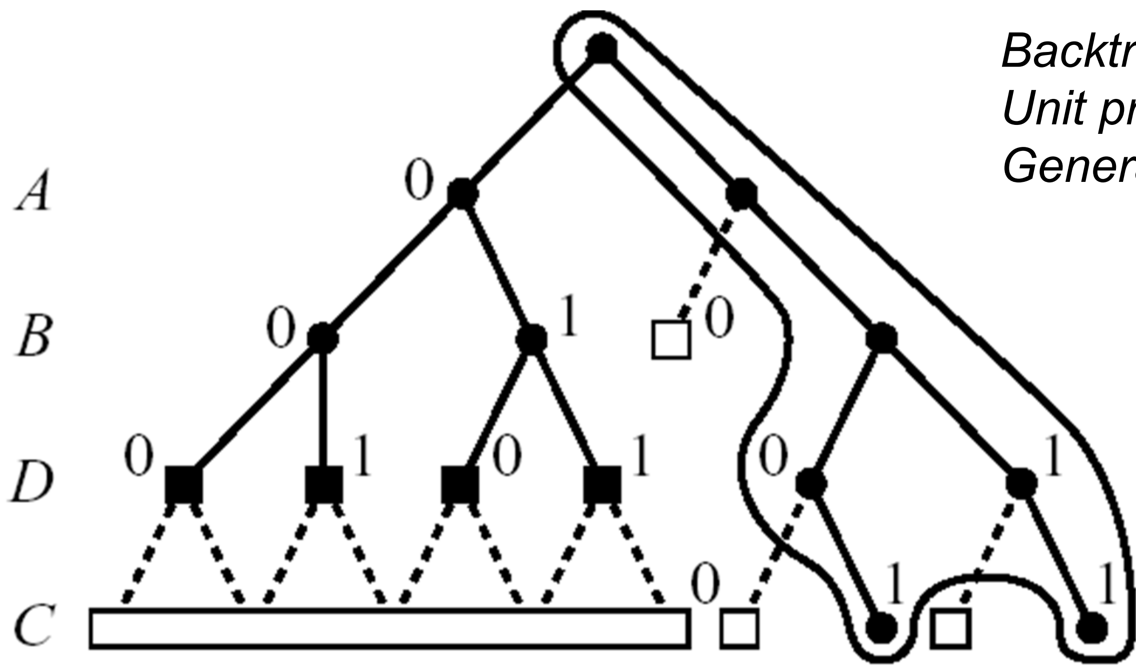
Constraint Programming

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

Branching-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 Graphical Models

- 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

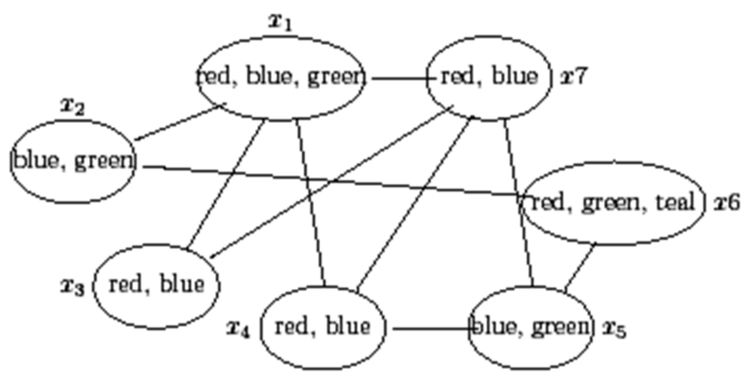
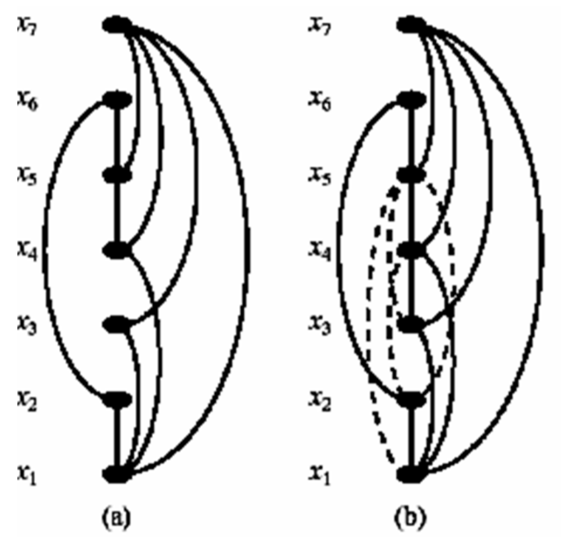
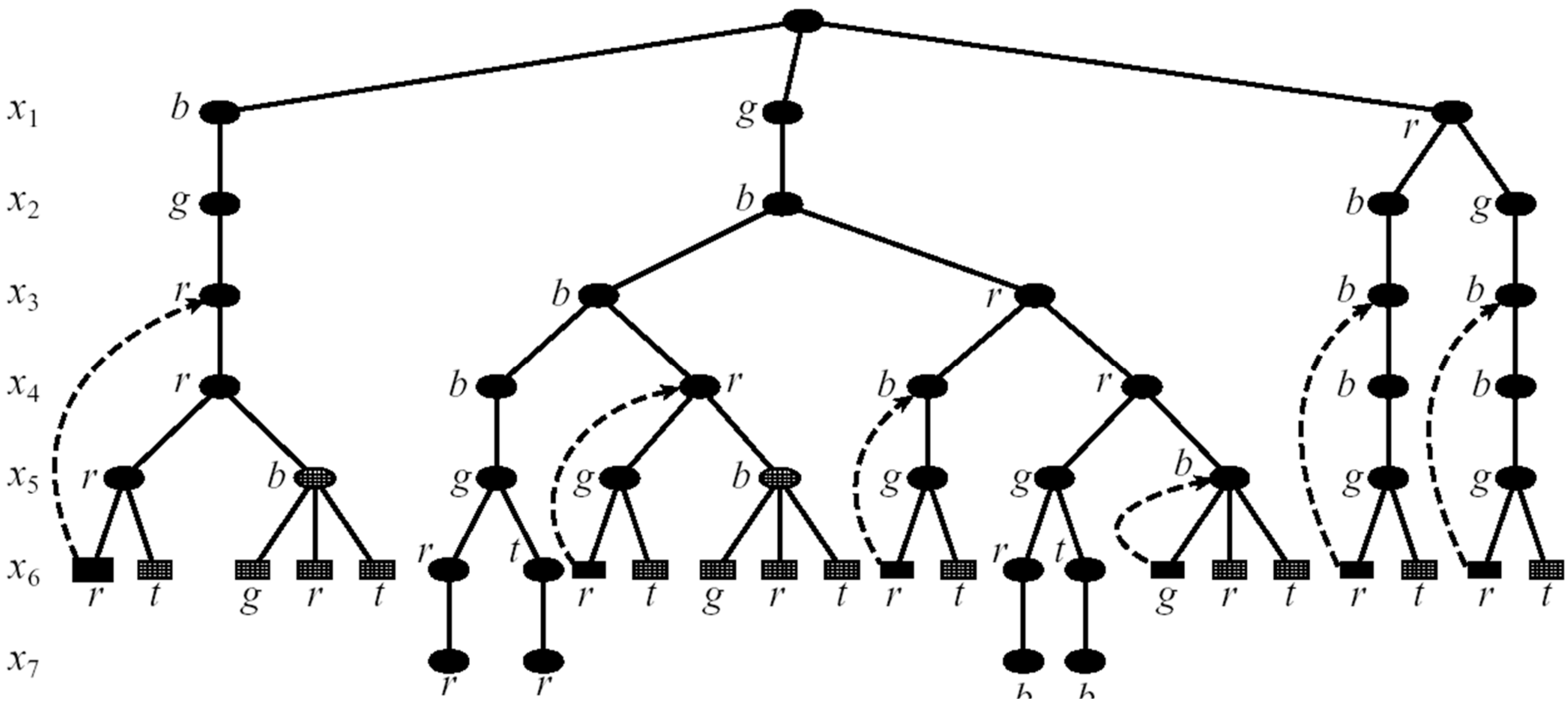


Figure 6.1: A modified coloring problem.

- (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**

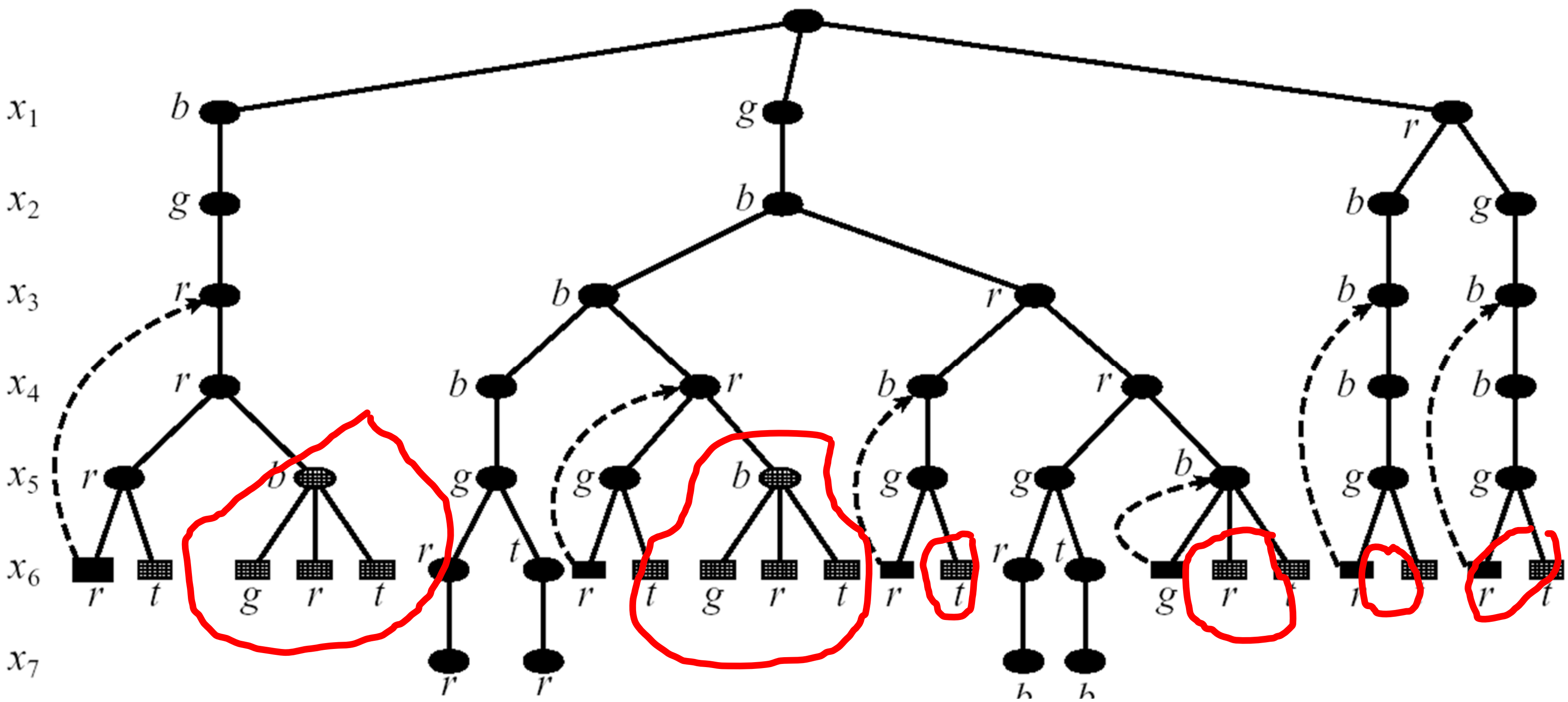


Jumps at dead-ends (Gascnnig-style 1977)



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

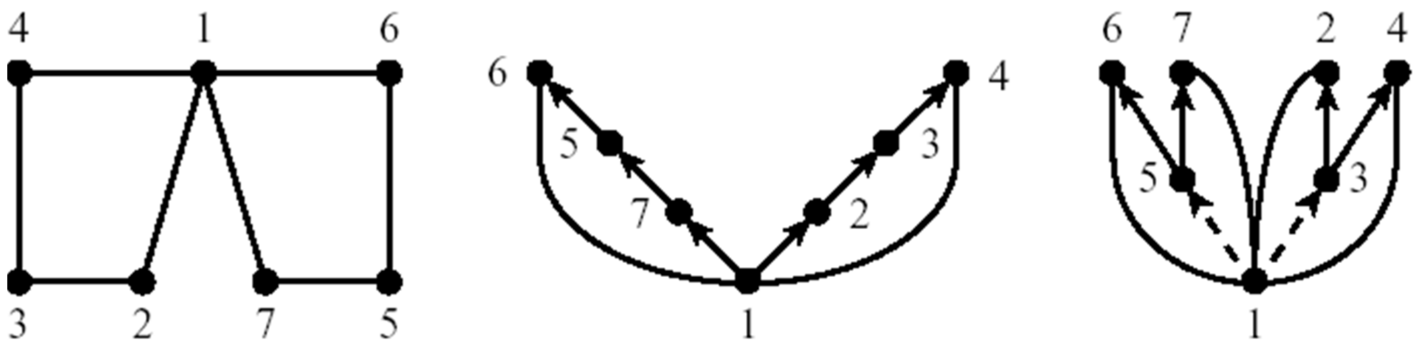
Jumps at Dead-Ends (Gascnnig 1977)



Example 6.3.1 In Figure 6.4, all of the backjumps illustrated lead to internal dead-ends, except for the jump back to $(\langle x_1, \textit{green}\rangle, \langle x_2, \textit{blue}\rangle, \langle x_3, \textit{red}\rangle, \langle x_4, \textit{blue}\rangle)$, 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



(a) (b) (c)

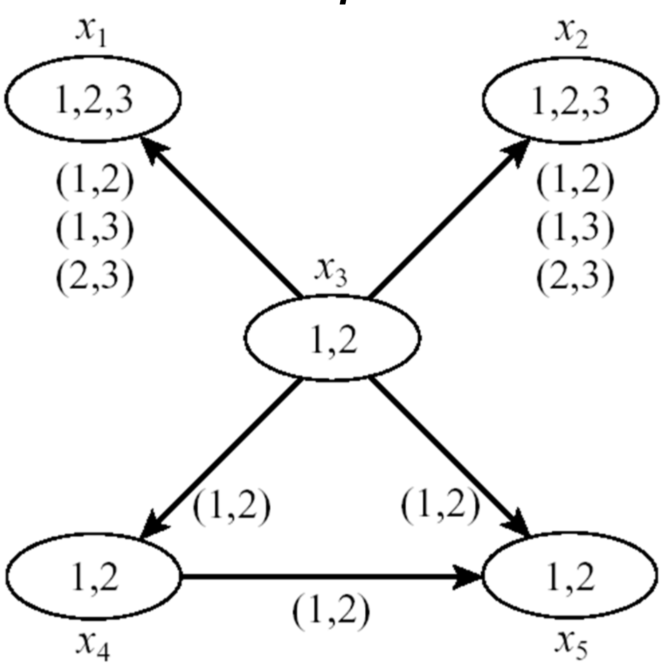
Simple: always jump back to parent in pseudo tree

*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

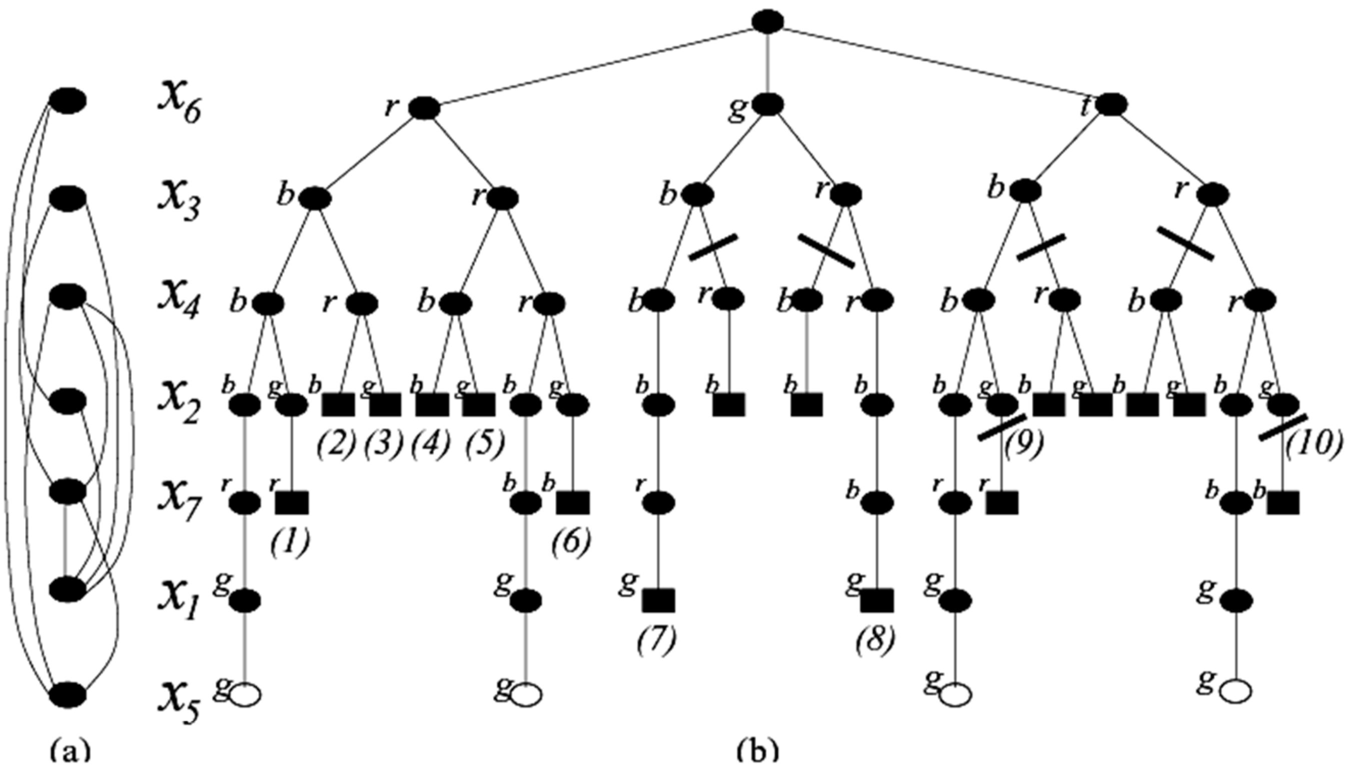


Figure 6.9: The search space explicated by backtracking on the CSP from Figure 6.1, using the variable ordering $(x_6, x_3, x_4, x_2, x_7, x_1, x_5)$ and the value ordering (*blue, red, green, teal*). Part (a) shows the ordered constraint graph, part (b) illustrates the search space. The cut lines in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

Complexity of Nogood-Learning for consistency

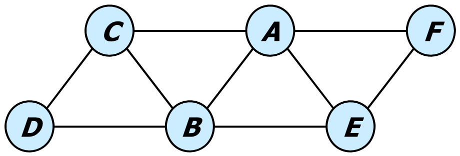
- The complexity of learning along d is time and space exponential in w*(d):
  - The number of dead-ends is bounded by $O(nk^{w^*(d)})$
  - Number of constraint tests per dead-end are $O(e)$

Space complexity is $O(nk^{w^*(d)})$
Time complexity is $O(n^2 e \cdot k^{w^*(d)})$
No-good Learning reduces time to O(exp(w*)) but requires O(exp(w*)) space.

All Solutions and Counting

- For all solutions and counting we will see
  - The additional impact of Good learning
  - BFS makes sense with good learning
  - BFS and DFS time and space exp(path-width)
  - Good-learning doesn't help consistency task

#CSP - Tree DFS Traversal

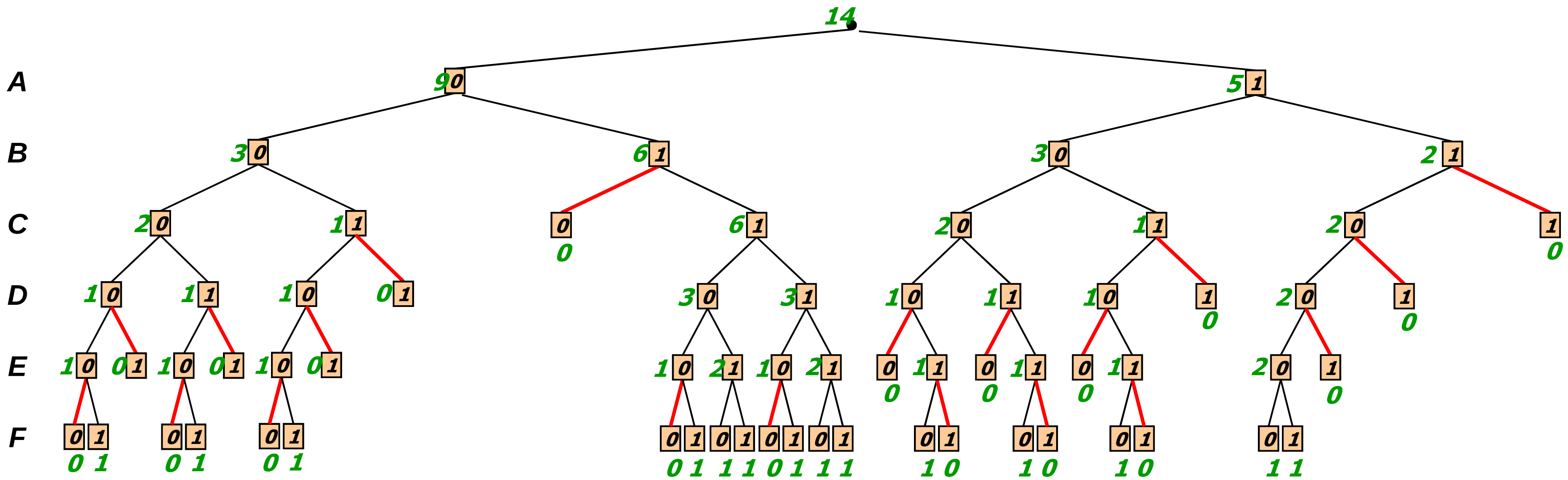


A	B	C	R_{ABC}
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

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

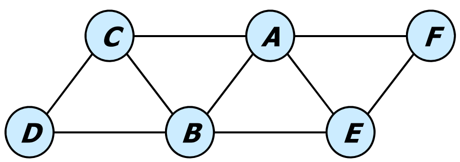
A	B	E	R_{ABE}
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

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



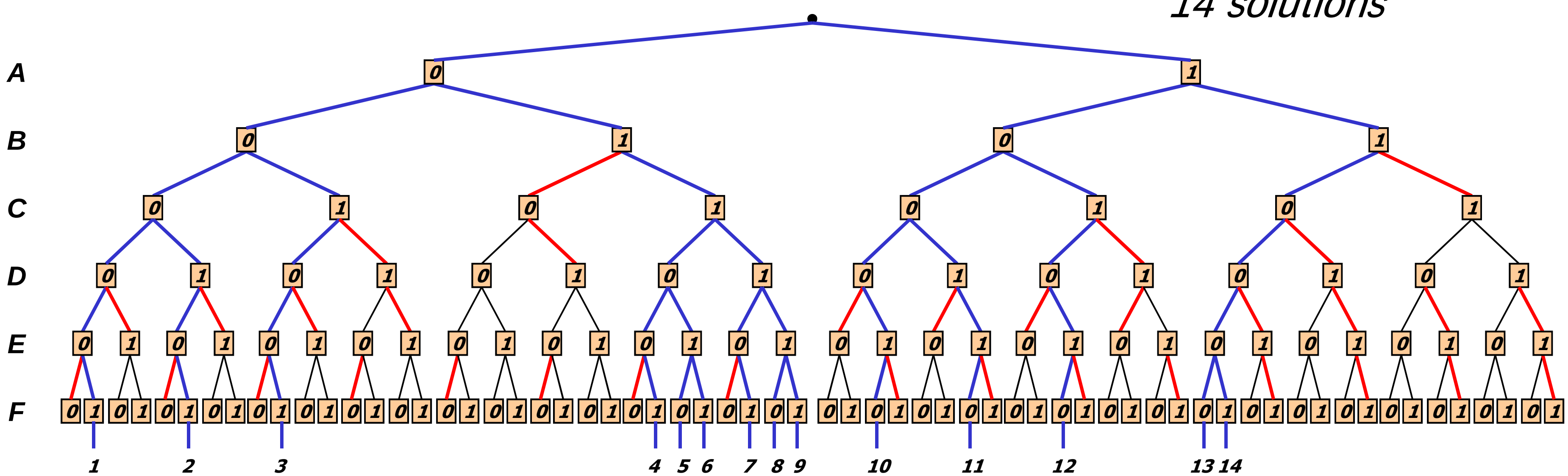
A	B	C	R_{ABC}
0	0	0	**1**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

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**

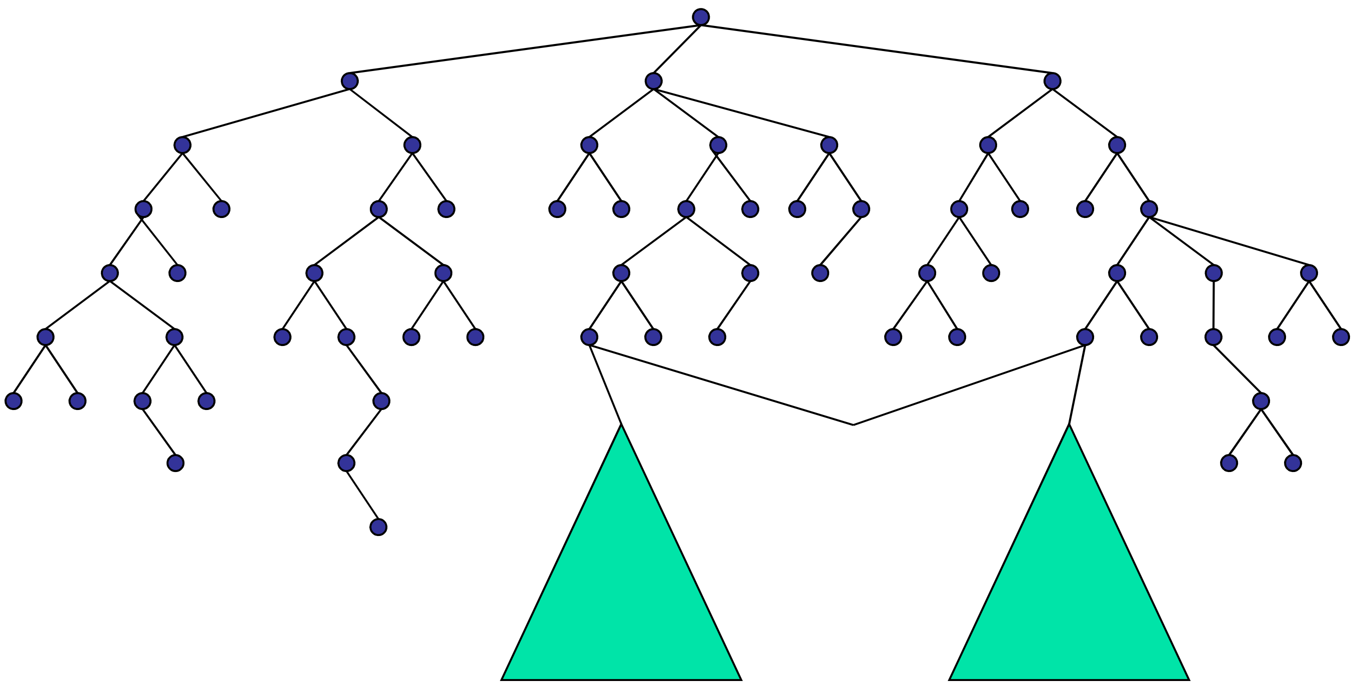
A	E	F	R_{AEF}
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

14 solutions



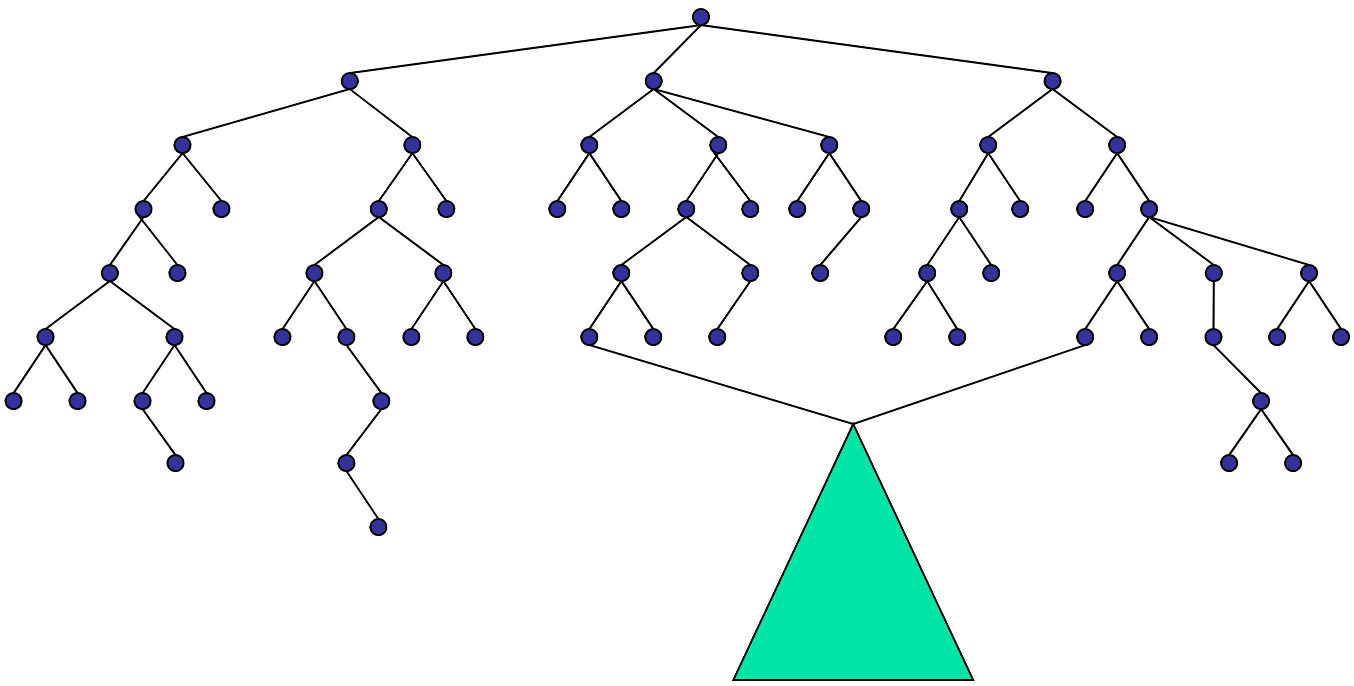
From search trees to search graphs

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

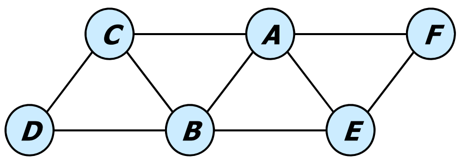


From search trees to search graphs

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



#CSP - Searching the Graph by Good Caching



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

A ***context(A) = [A]***

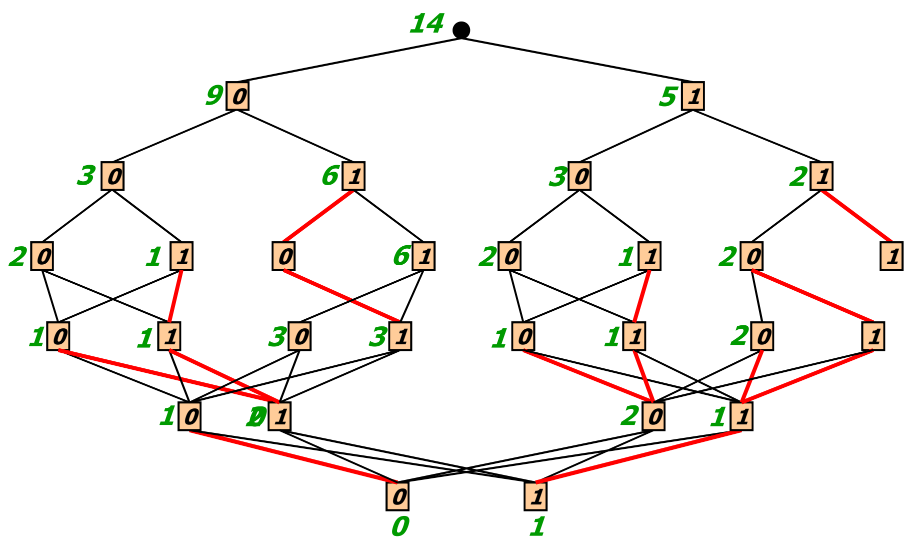
B ***context(B) = [AB]***

C ***context(C) = [ABC]***

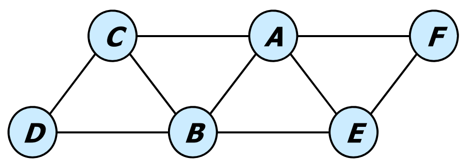
D ***context(D) = [ABD]***

E ***context(E) = [AE]***

F ***context(F) = [F]***



#CSP - Searching the Graph by Good Caching



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

A ***context(A) = [A]***

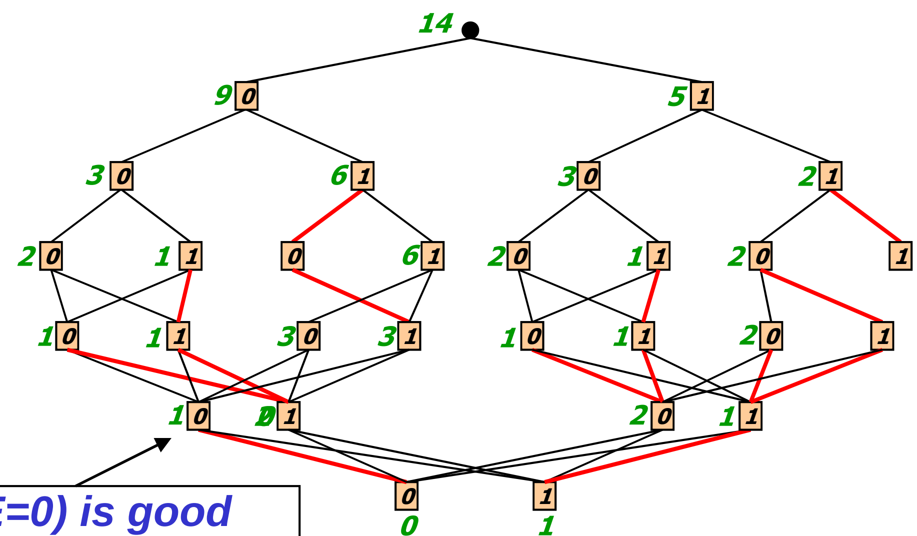
B ***context(B) = [AB]***

C ***context(C) = [ABC]***

D ***context(D) = [ABD]***

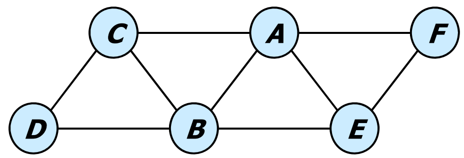
E ***context(E) = [AE]***

F ***context(F) = [F]***



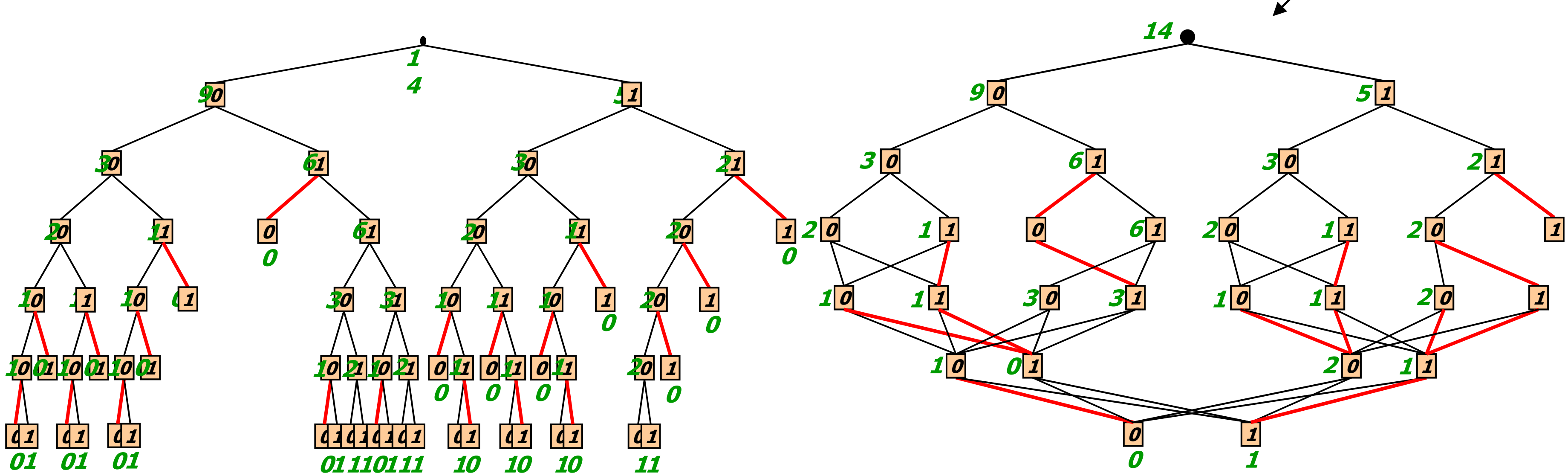
(A=0,E=0) is good
V(A=0,E=0)=1

#CSP - Searching the Graph by Good Caching



No caching:
O(exp(n))

Good-caching:
O(exp(pw))



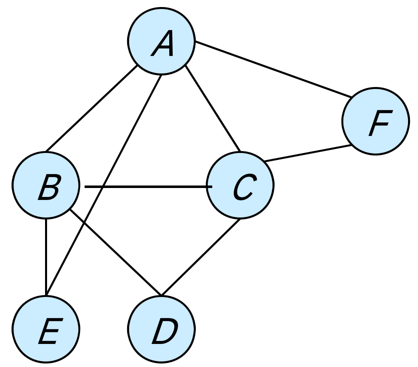
Summary: Search Principles

- 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.
- Good learning cache results but helps only counting, enumeration

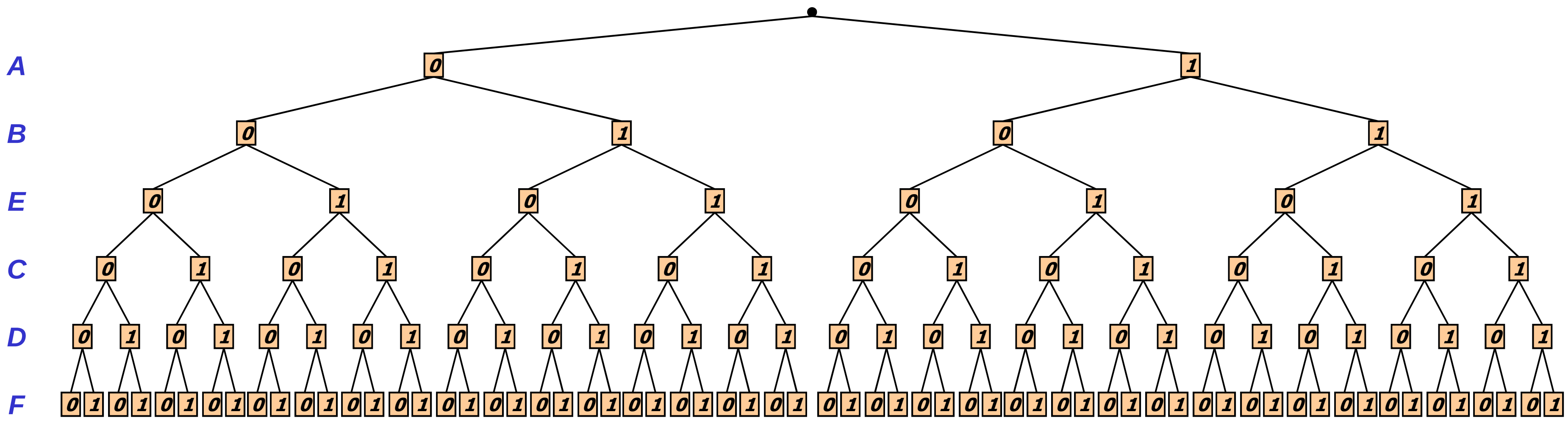
Road Map:
Search in Graphical Models

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

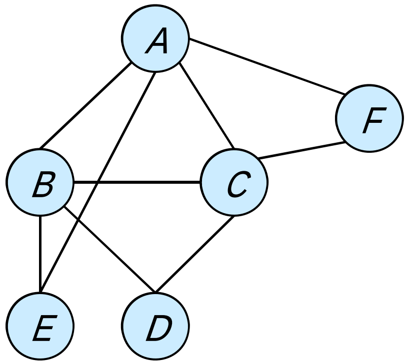
OR Search Space



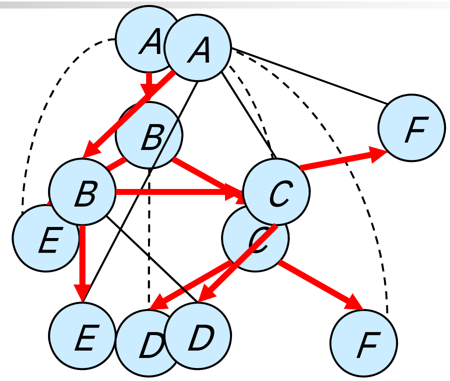
Ordering: A B E C D F



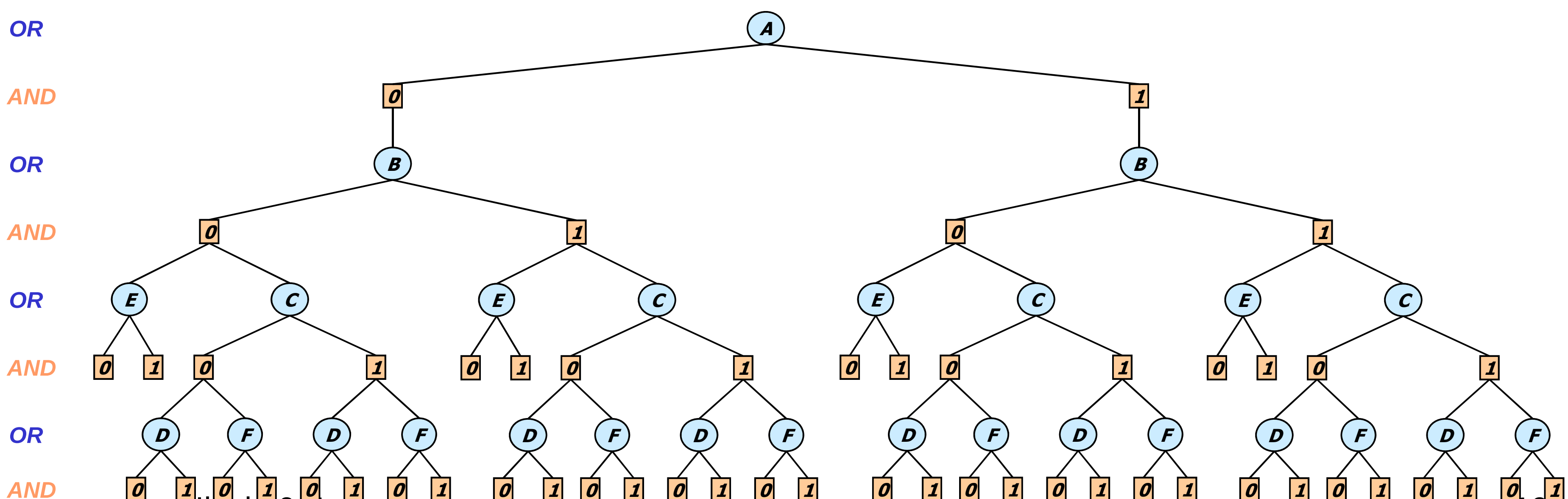
AND/OR Search Space



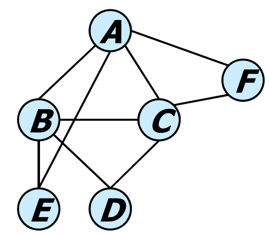
Primal graph

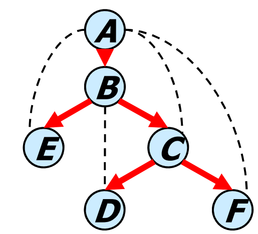


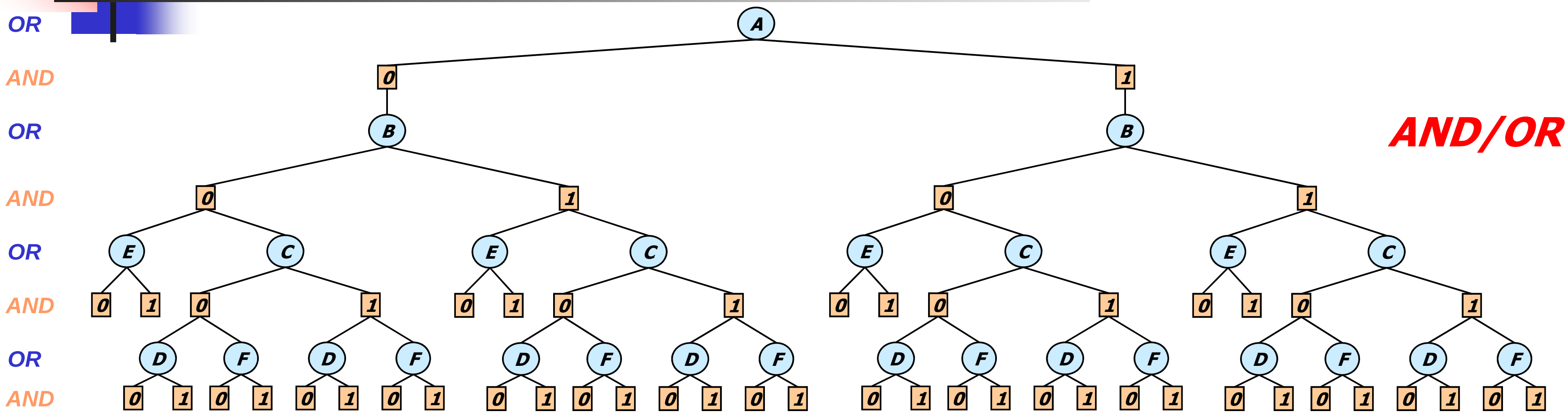
DFS tree



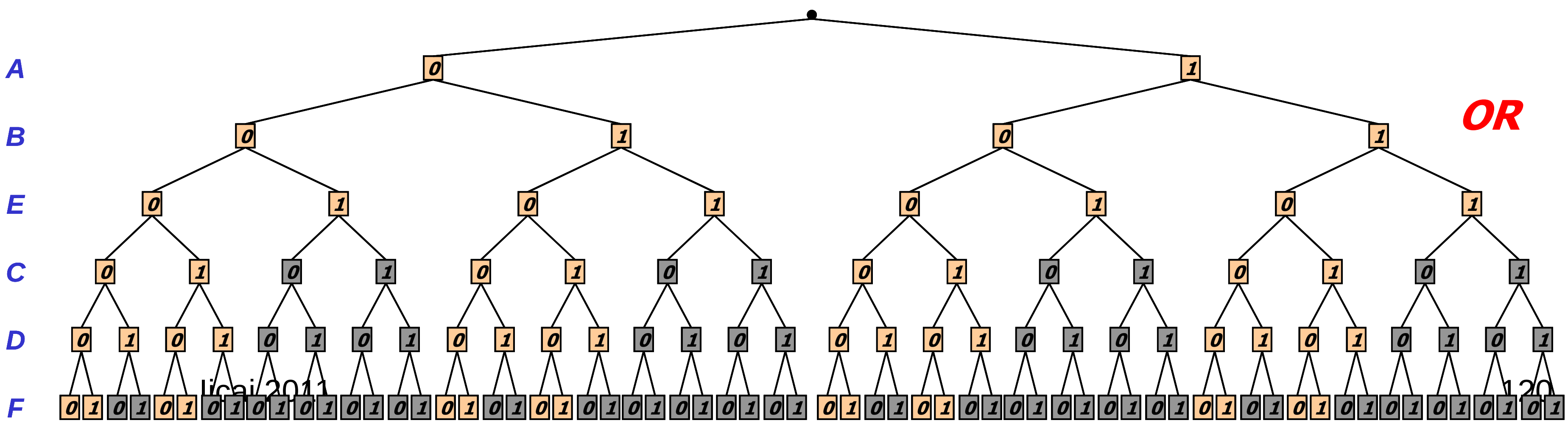
AND/OR vs. OR





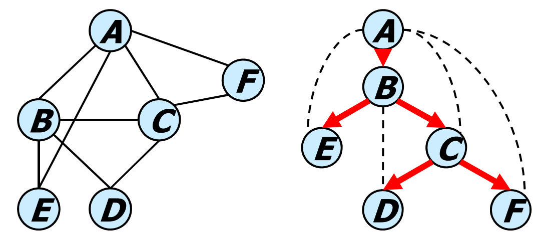


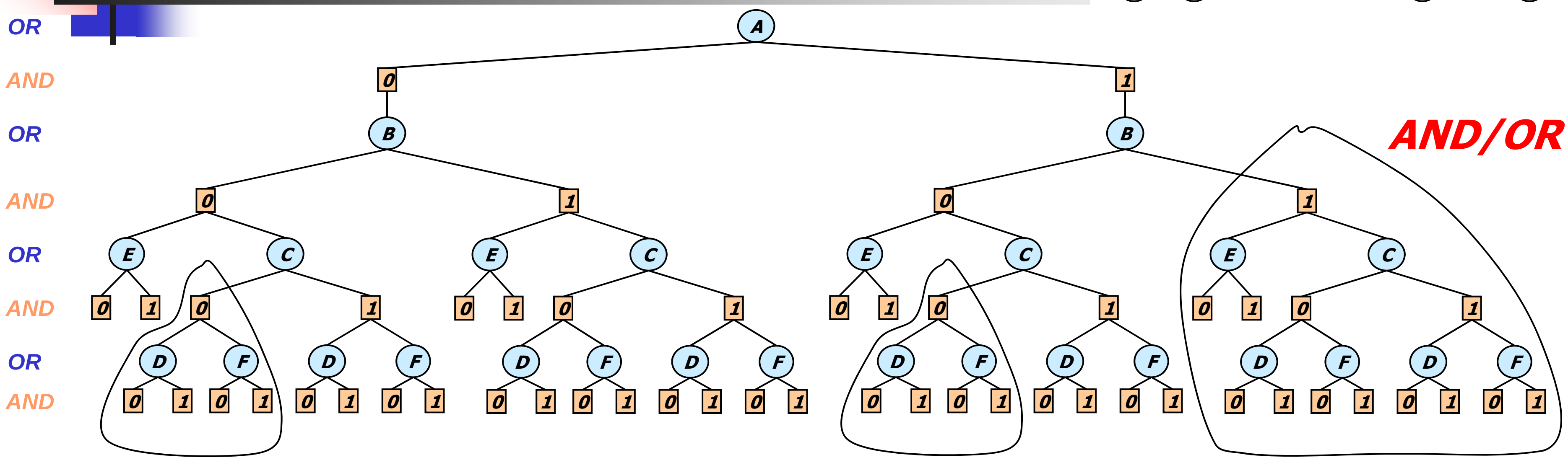
AND/OR size: exp(4), OR size exp(6)

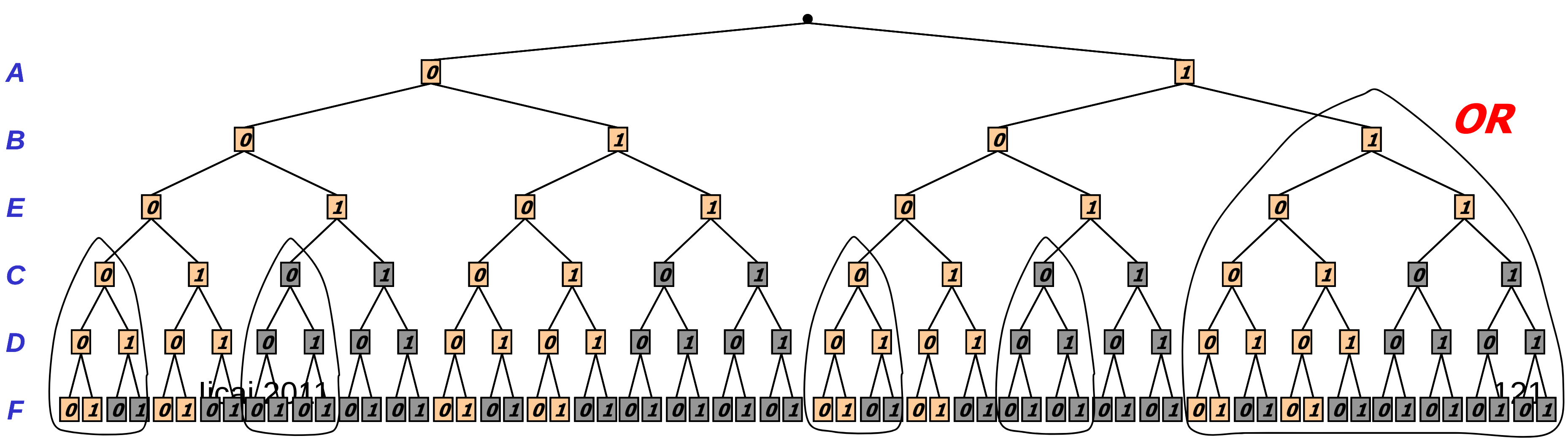


AND/OR vs. OR

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

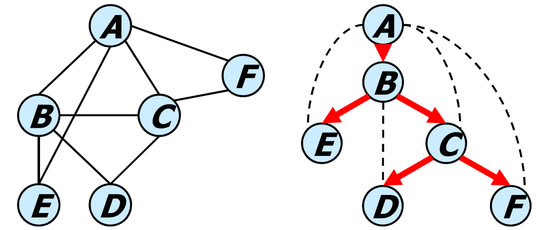


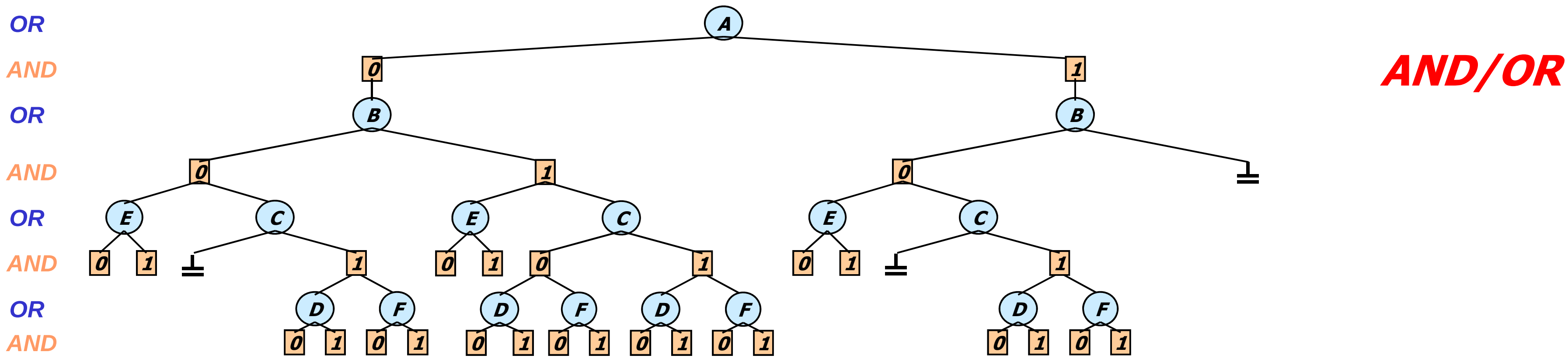


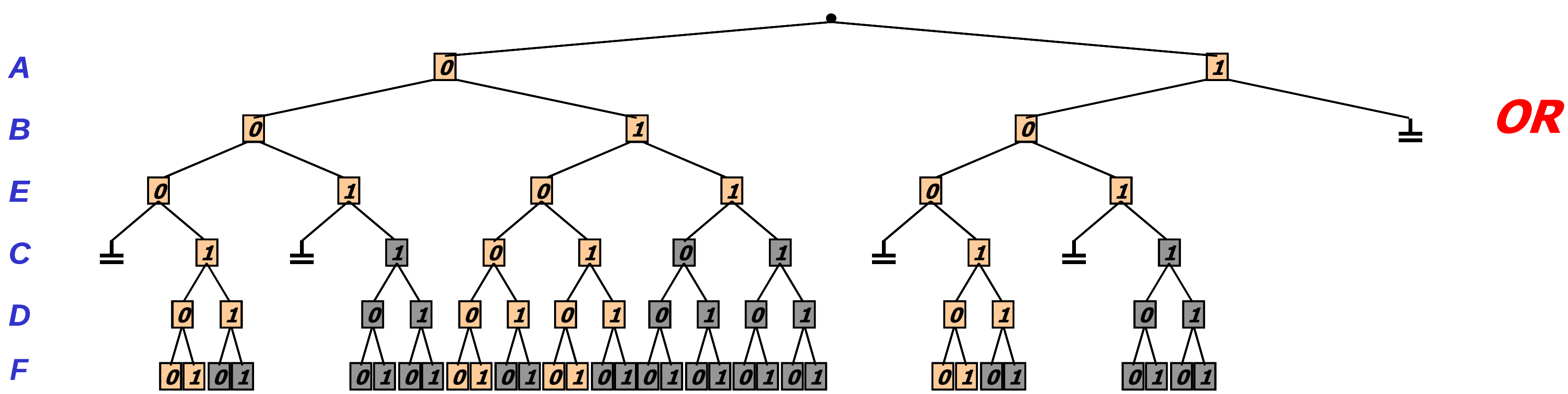


AND/OR vs. OR

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

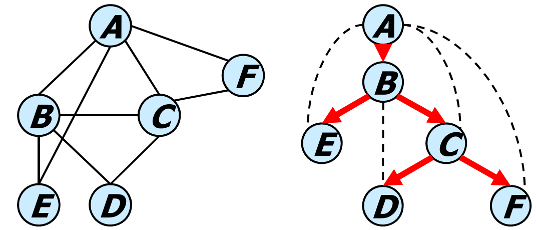


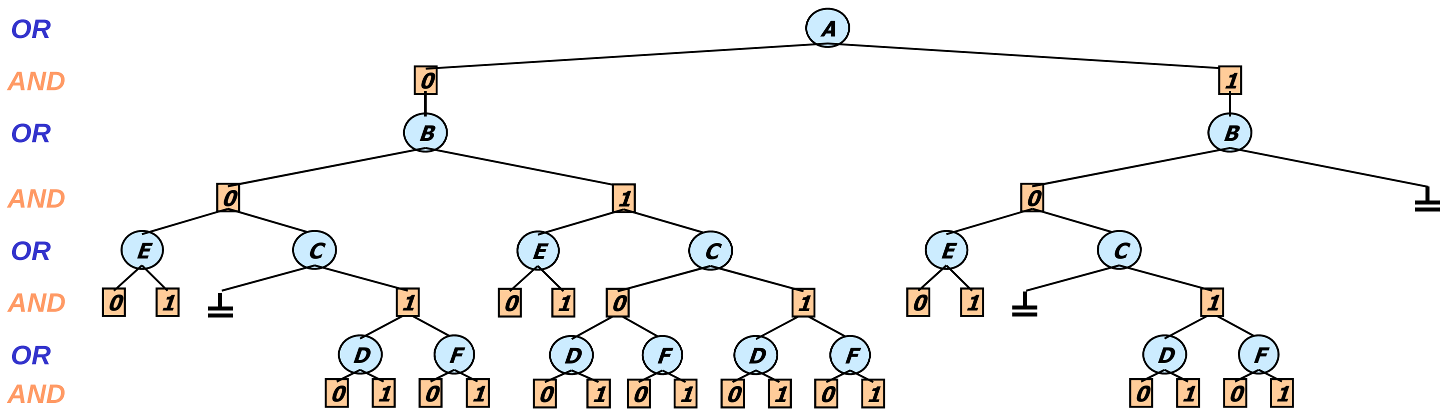




AND/OR vs. OR

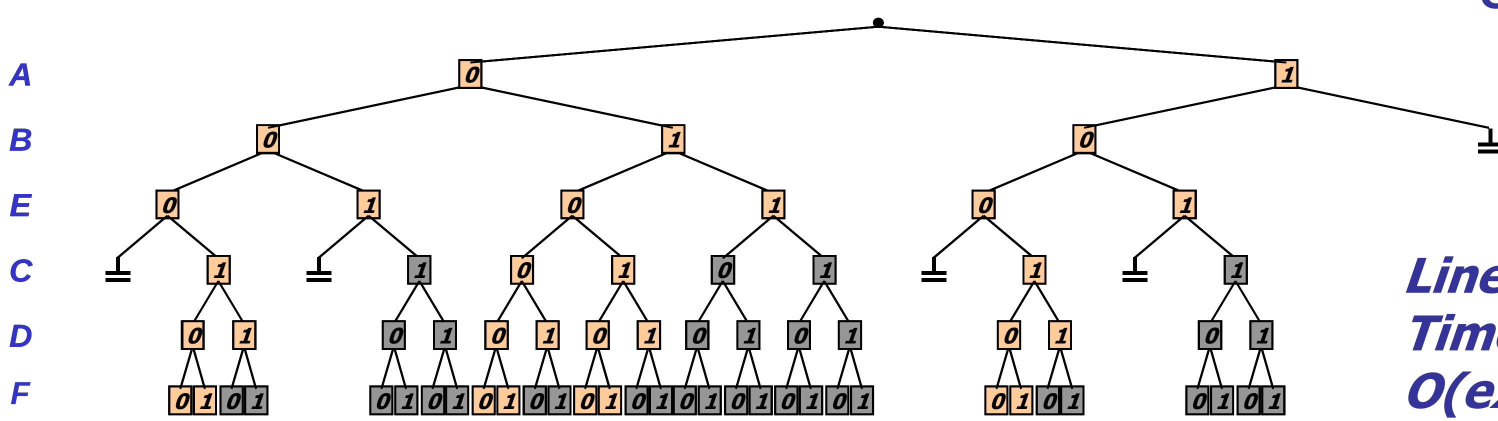
(A=1,B=1)
(B=0,C=0)





AND/OR

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



OR

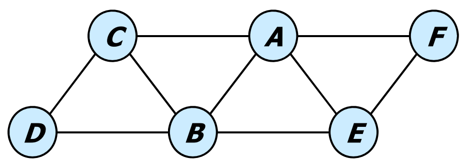
Linear space,
Time:
O(exp(n))

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

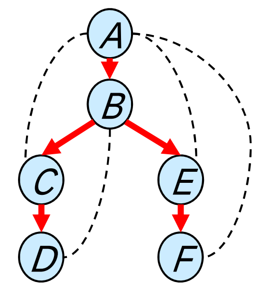


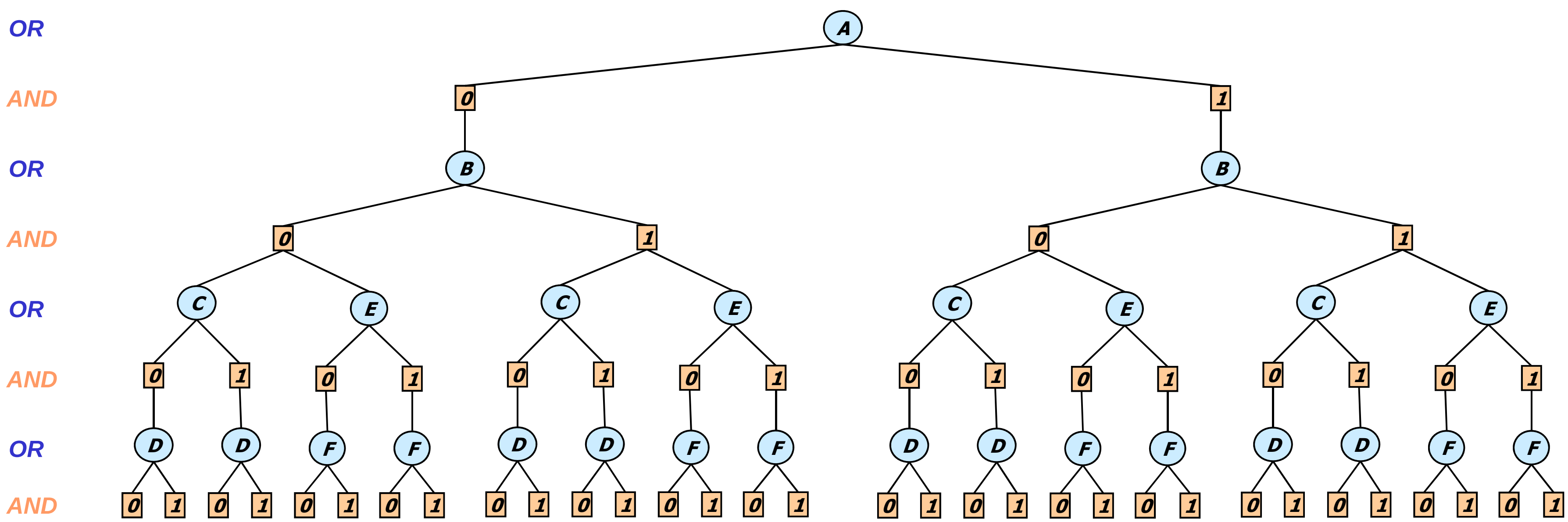
A	B	C	R_{ABC}
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

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

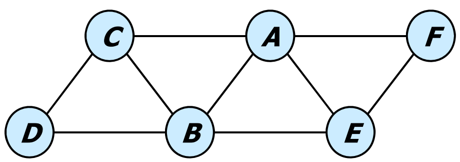
A	B	E	R_{ABE}
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

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

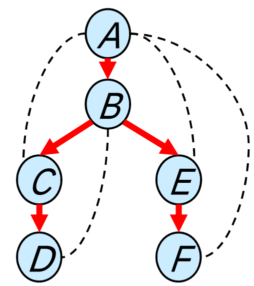


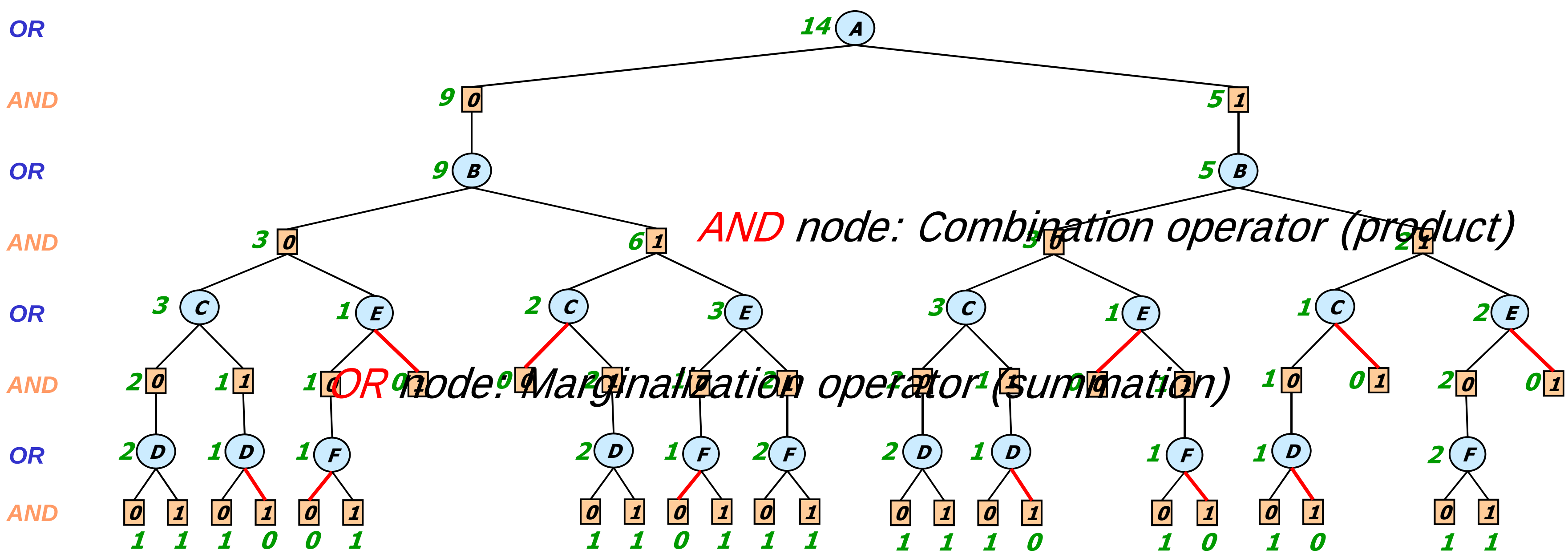
A	B	C	R_{ABC}
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

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

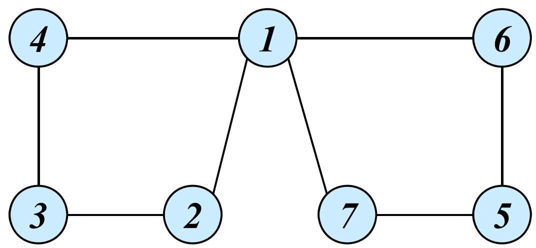
A	E	F	R_{AEF}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0





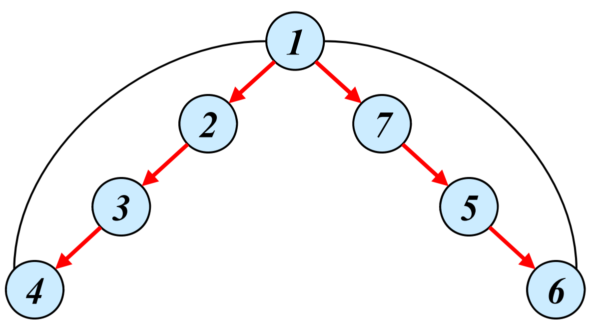
Pseudo-Trees

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

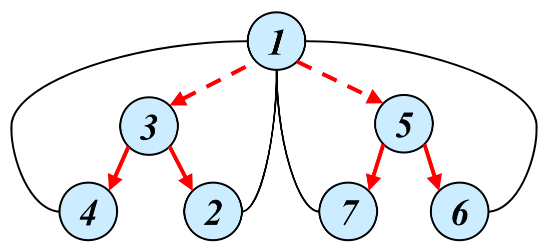


(a) Graph

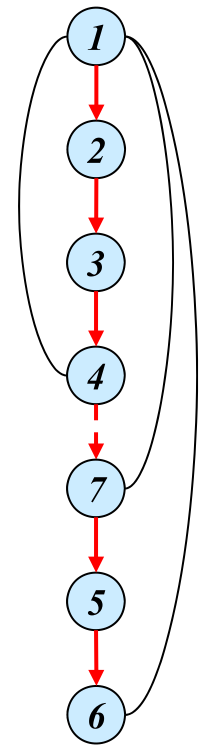
m <= w* log n



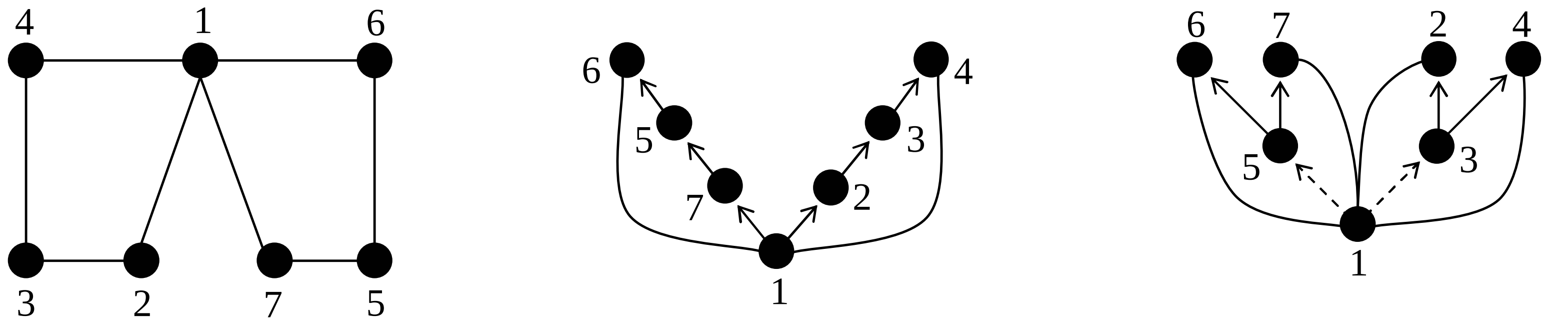
(b) DFS tree
depth=3



(c) pseudo- tree
depth=2



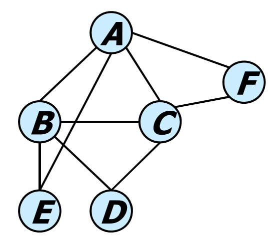
(d) Chain
depth=6



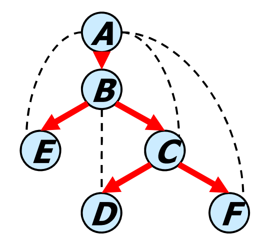
(a) (b) (c)

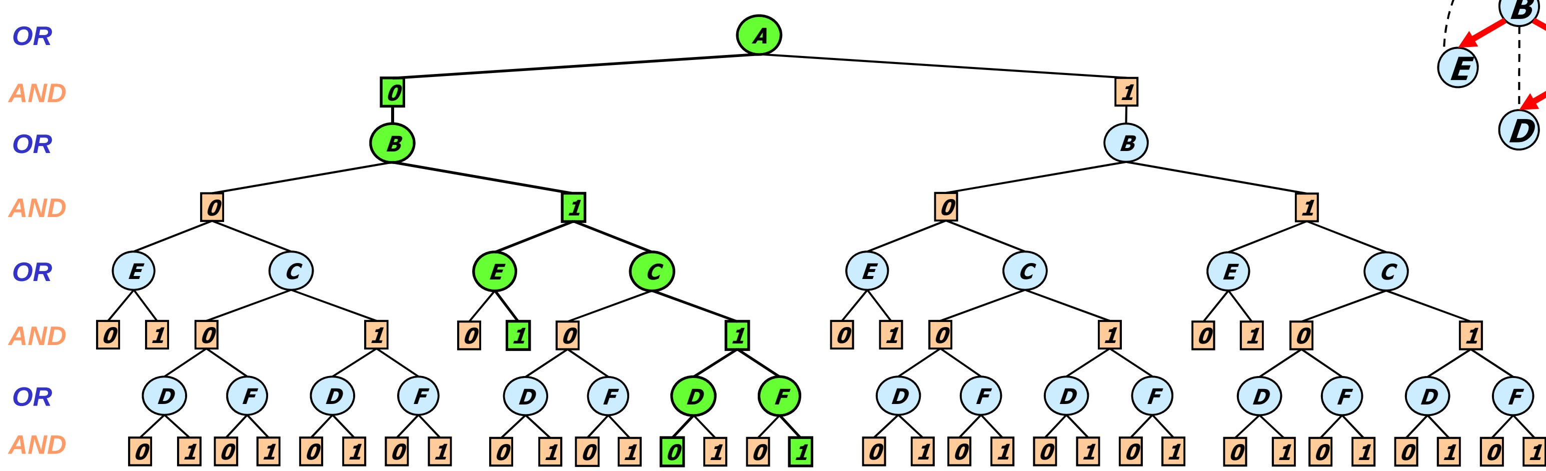
AND/OR search tree for graphical models

- **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 A/O Trees to Search A/O Graphs

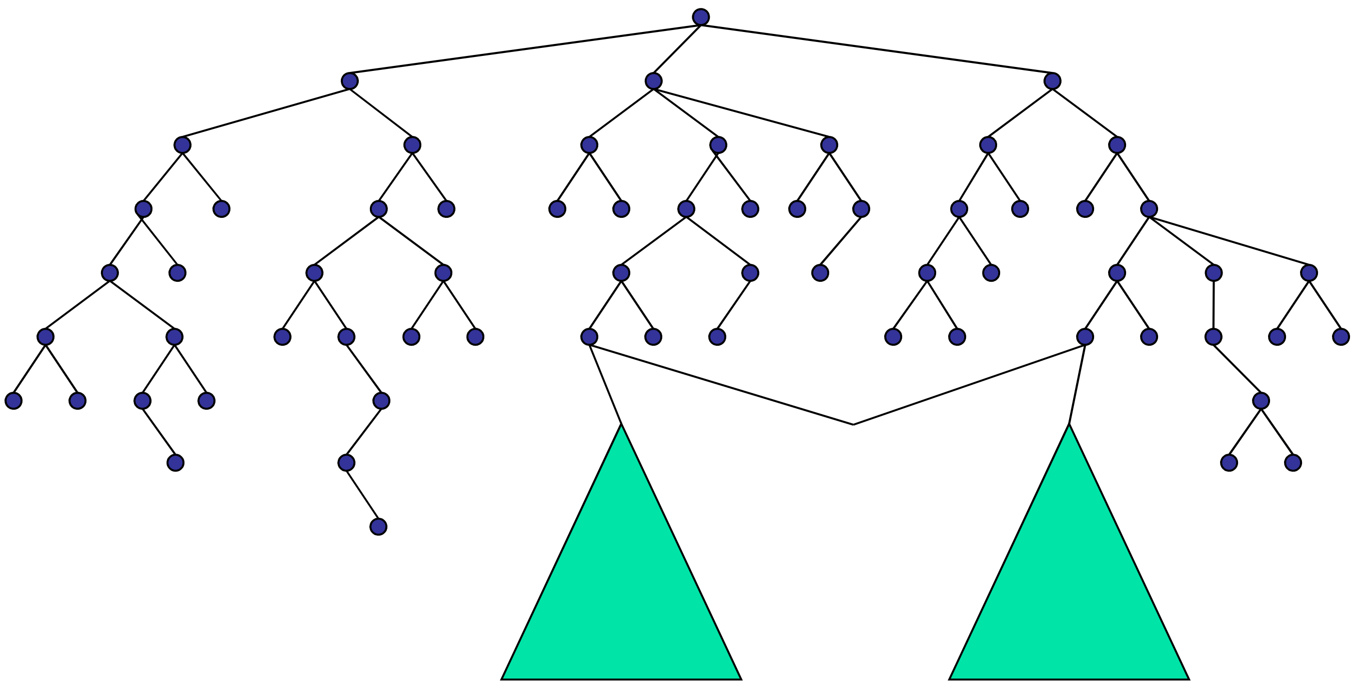
- Any two nodes that root identical subtrees/subgraphs can be merged

- Minimal AND/OR search graph: closure under merge of the AND/OR search tree

  - Inconsistent sub-trees can be pruned too.
  - Some portions can be collapsed or reduced.

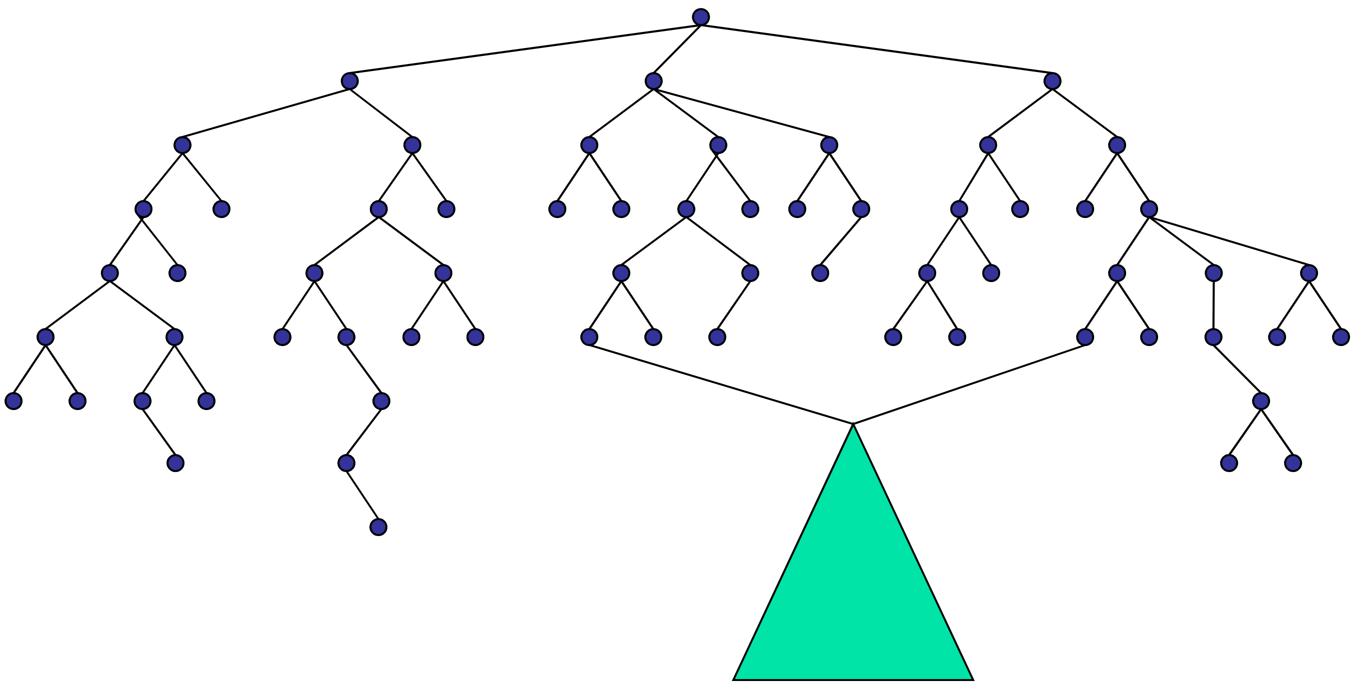
From search trees to search graphs

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

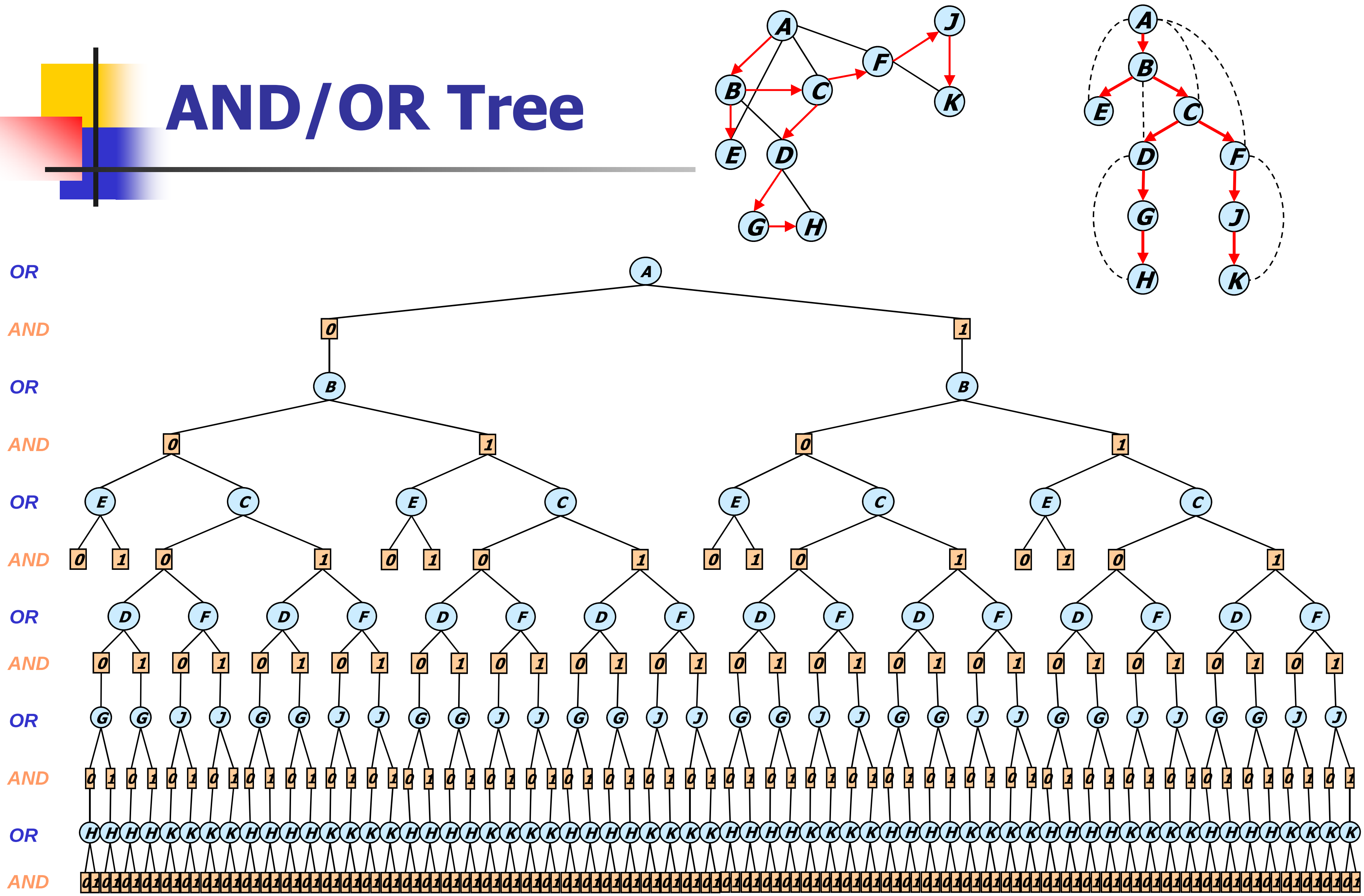


From search trees to search graphs

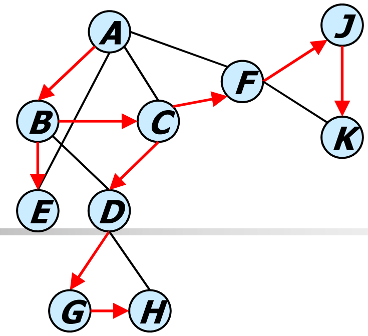
- Any two nodes that root identical subtrees (subgraphs) can be merged

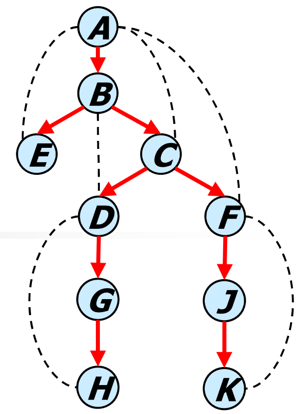


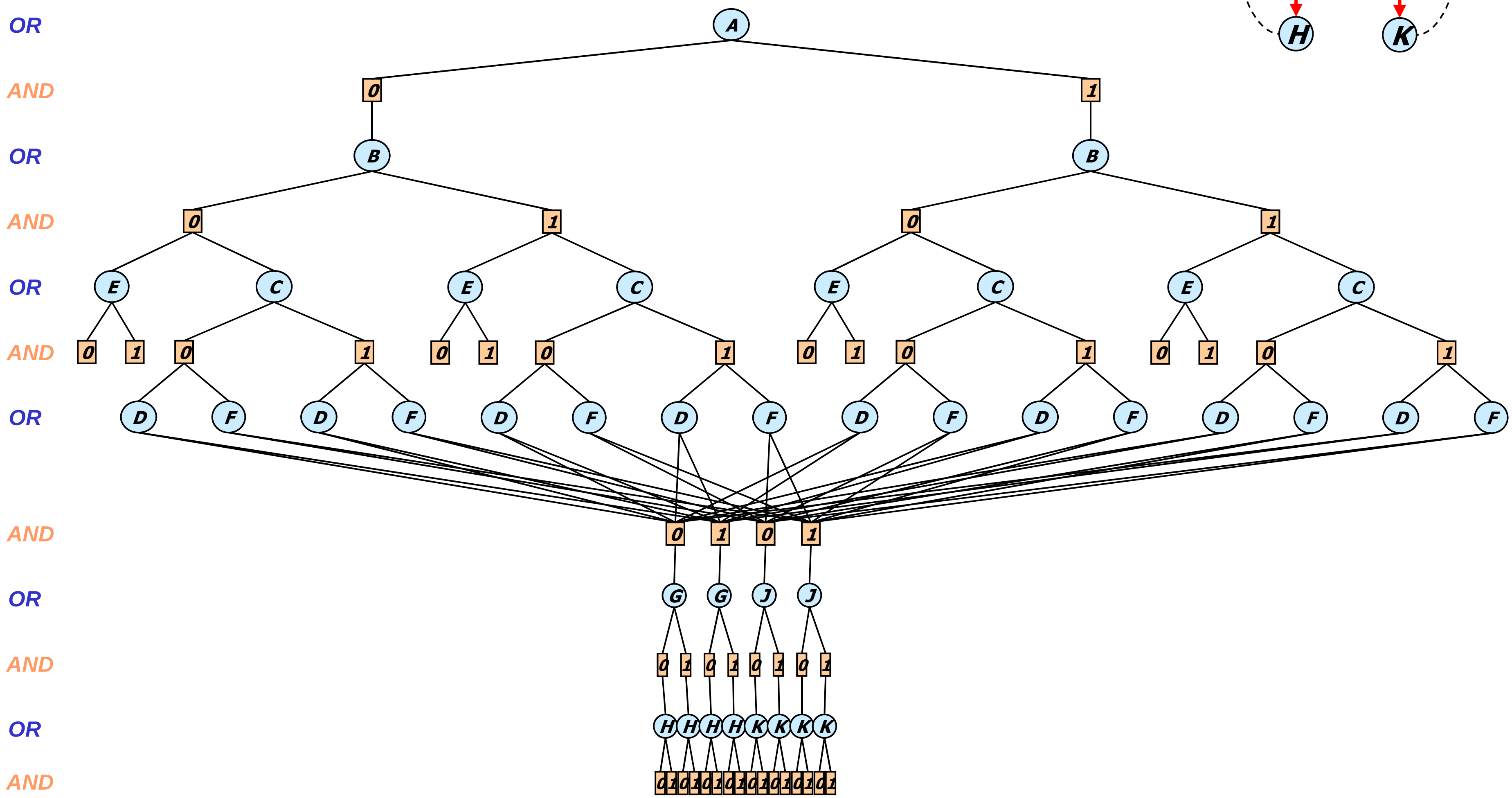
AND/OR Tree



An AND/OR Graph: Caching Goods



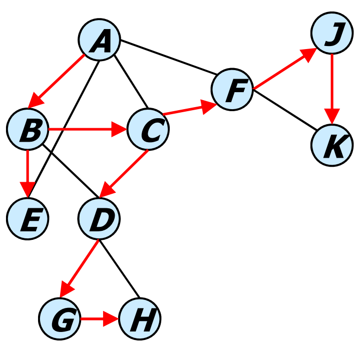


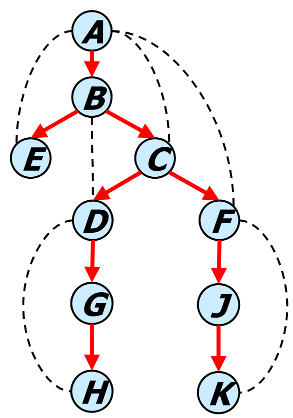


Context-based Caching

- 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}

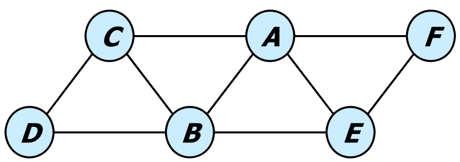
context(F) = {F}

Complexity of 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.

#CSP – AND/OR Tree DFS

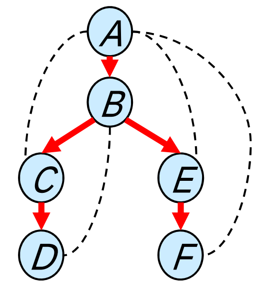


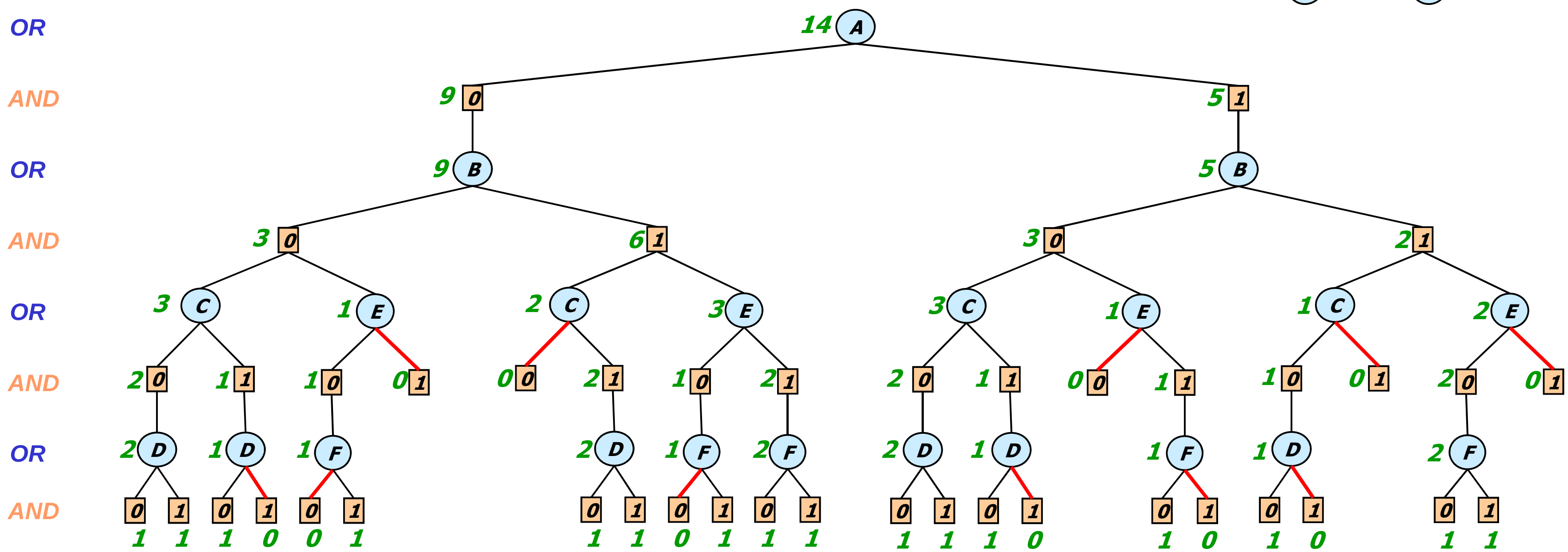
A	B	C	R_{ABC}
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

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

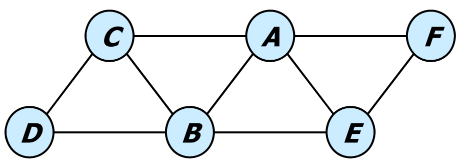
A	B	E	R_{ABE}
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

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





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

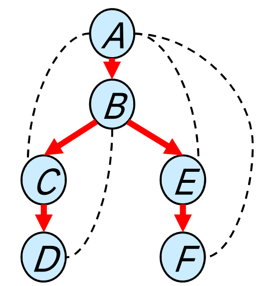


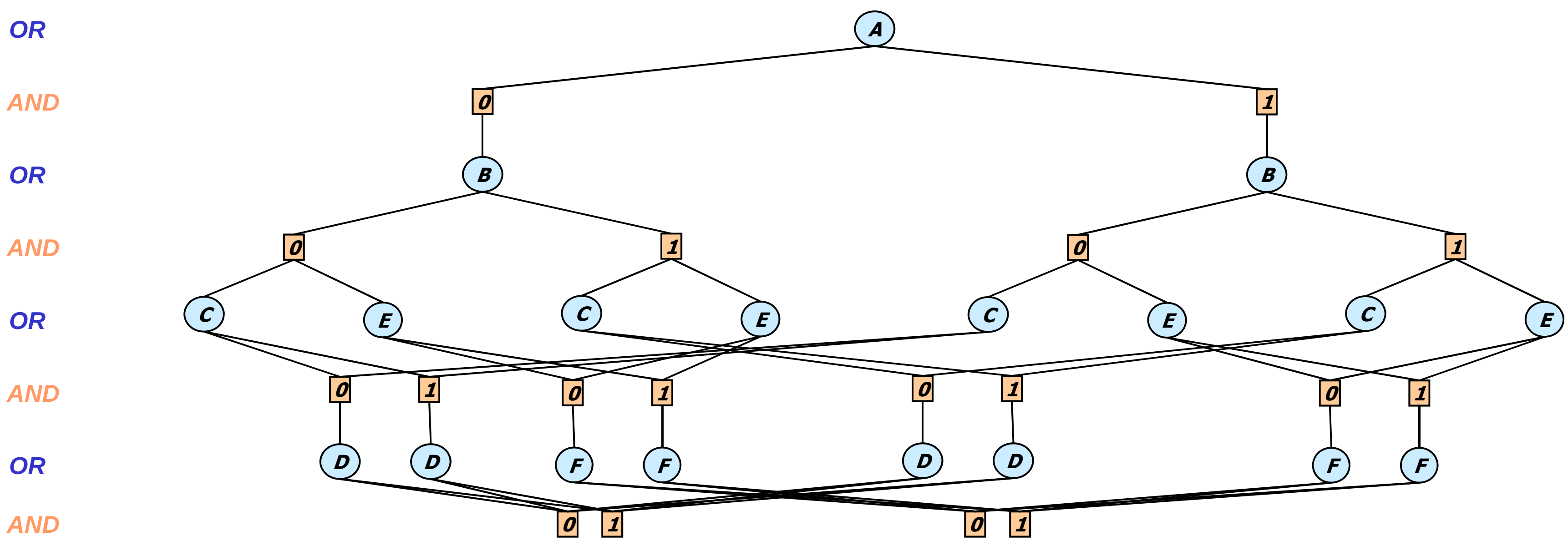
A	B	C	R_{ABC}
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

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

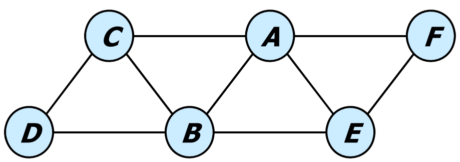
A	B	E	R_{ABE}
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

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)

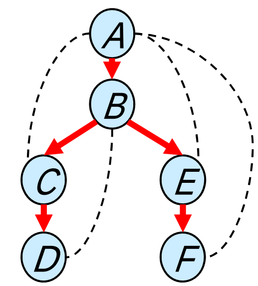


A	B	C	R_{ABC}
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

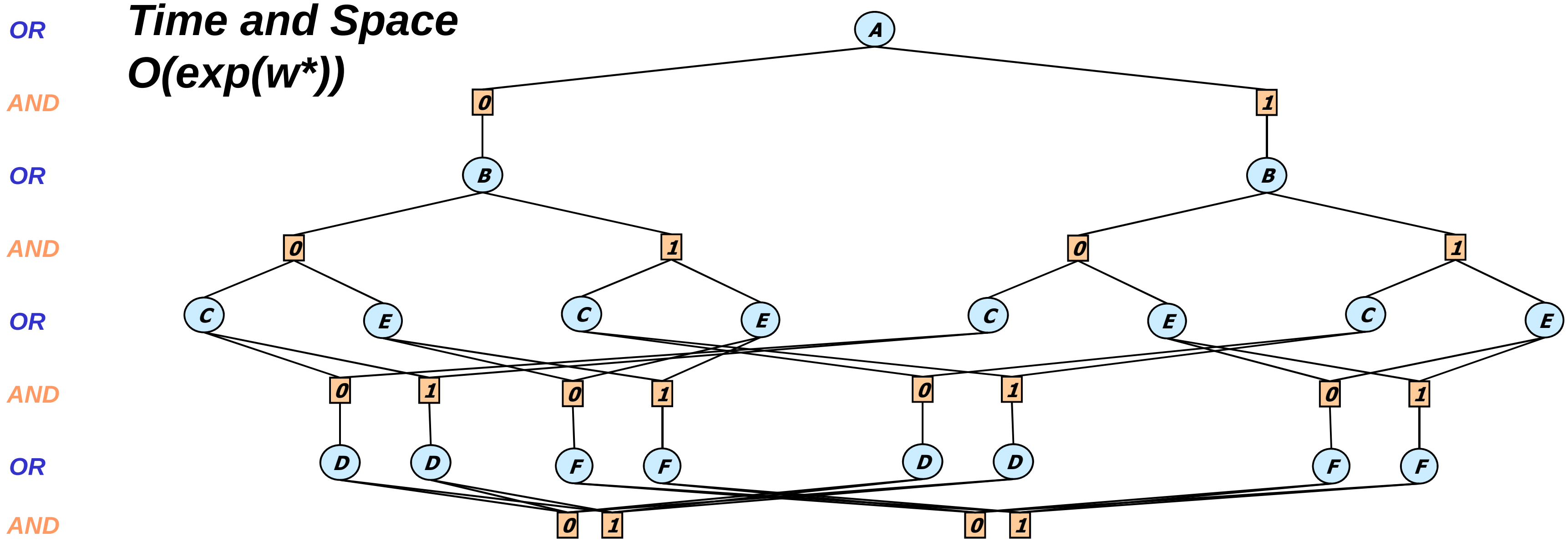
B	C	D	R_{BCD}
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

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

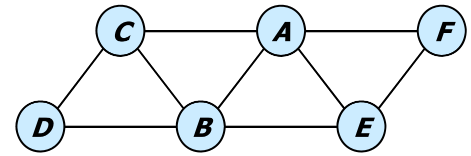
A	E	F	R_{AEF}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

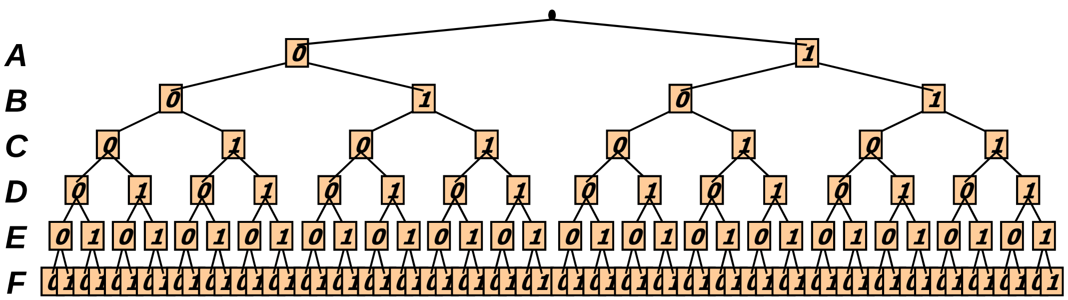


Time and Space O(exp(w*))



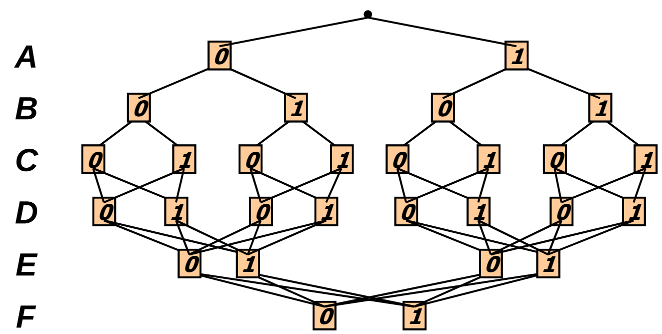
All Four Search Spaces





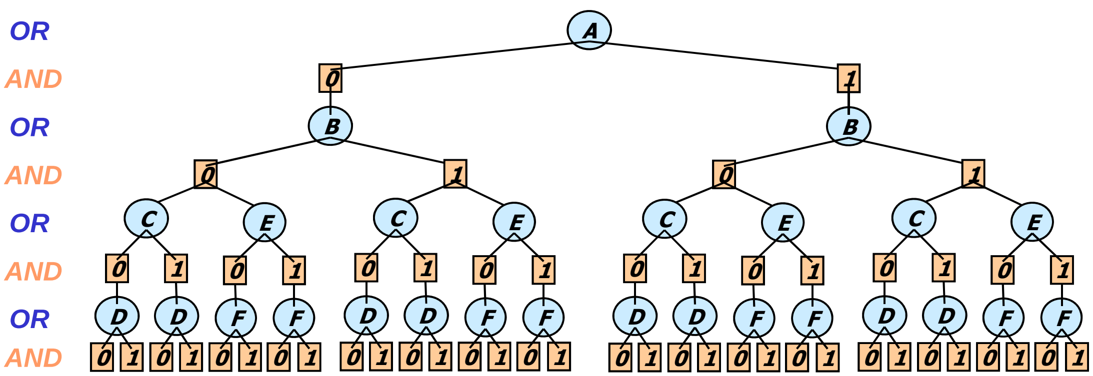
Full OR search tree

126 nodes



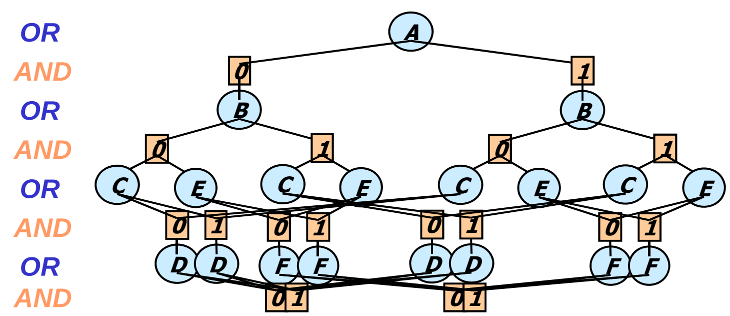
Context minimal OR search graph

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)

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

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

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

Properties of minimal AND/OR graphs

Theorem (Complexity):

- **Minimal AND/OR context graph is bounded exponentially by the tree-width w* of the graphical model along its pseudo-tree**
- **Minimal OR search graph is bounded exponentially by its path-width pw* (w* ≤ pw*)**

Theorem (Uniqueness):

- **Given a pseudo-tree, the minimal AND/OR search graph is unique (canonical) for all equivalent graph-models that are consistent with that pseudo tree.**

- **Related to compilation schemes:**
  - **Minimal OR – related to OBDDs (Bryant, McMillan)**
  - **Minimal AND/OR – related to tree-OBDDs (McMillan 94),**
  - **d-DNNF (Darwiche et. Al. 2002)**
  - **Case-factor diagrams (Mcallester, Collins, Pereira, 2005)**

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)

i=0 → *i=w**

Space: O(n)

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

Space: O(exp w*)

Time: O(exp w*)

AO(i) time complexity?

Impact of AND/OR for Constraint Processing

- Minor impact for Constraint-satisfaction
  - Search with backjumping or without backjumping
    - Space: linear, Time: O(exp(logn w*))
  - Search with lno-goods learning
    - time and space: O(exp(w*))
  - Variable-elimination
    - time and space: O(exp(w*))
- Counting, enumeration
  - Search with backjumping
    - Space: linear, Time: O(exp(n))
    - Space: linear, Time: O(exp(log n w*))
  - Search with no-goods caching only
    - space: O(exp(w*)) Time: O(exp(n))
    - space: O(exp(w*)) Time: O(exp(log n w*))
  - Search with goods and no-goods learning
    - Time and space: O(exp(path-width), O(exp(log n w*))
    - Time and space: O(exp(tree-width), O(exp(w*))
  - Variable-elimination
    - Time and space: O(exp(w*))

Road Map

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

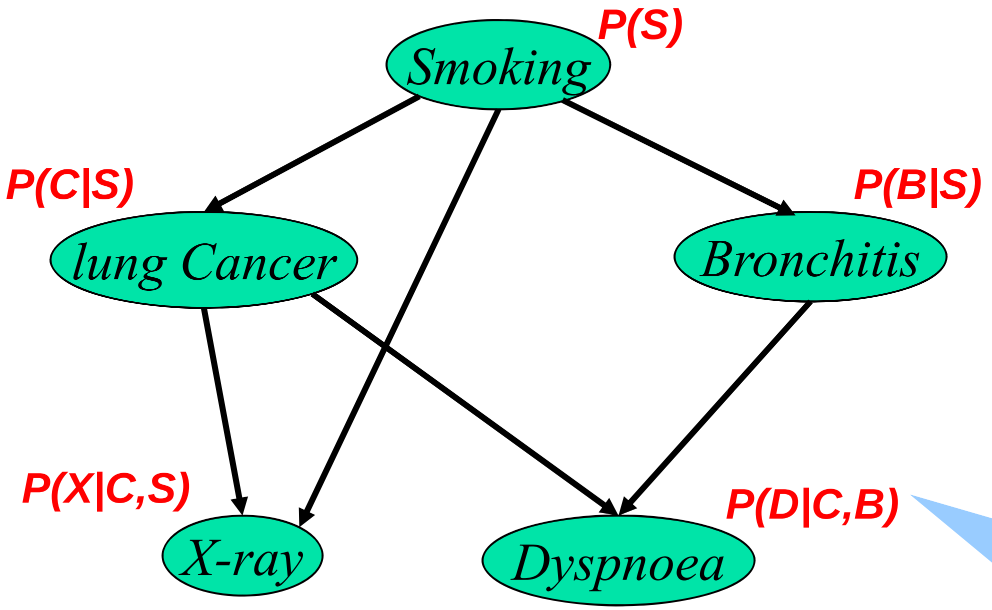
Road Map:Bayesian Networks

- 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

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
- Sampling

Bayesian Networks: Representation
(Pearl, 1988)



BN = (G, Θ)

CPD:

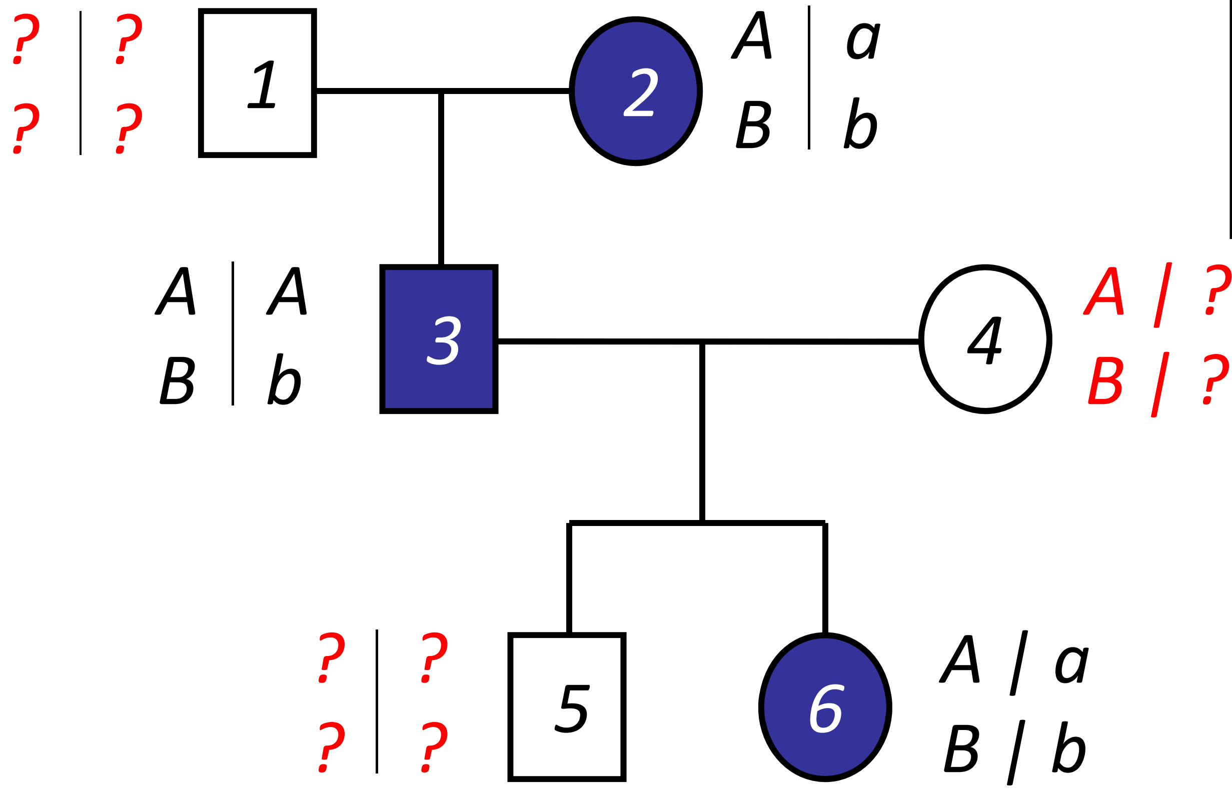
C	B	P(D|C,B)	
0	0	0.1	0.9
0	1	0.7	0.3
1	0	0.8	0.2
1	1	0.9	0.1

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

Belief Updating:

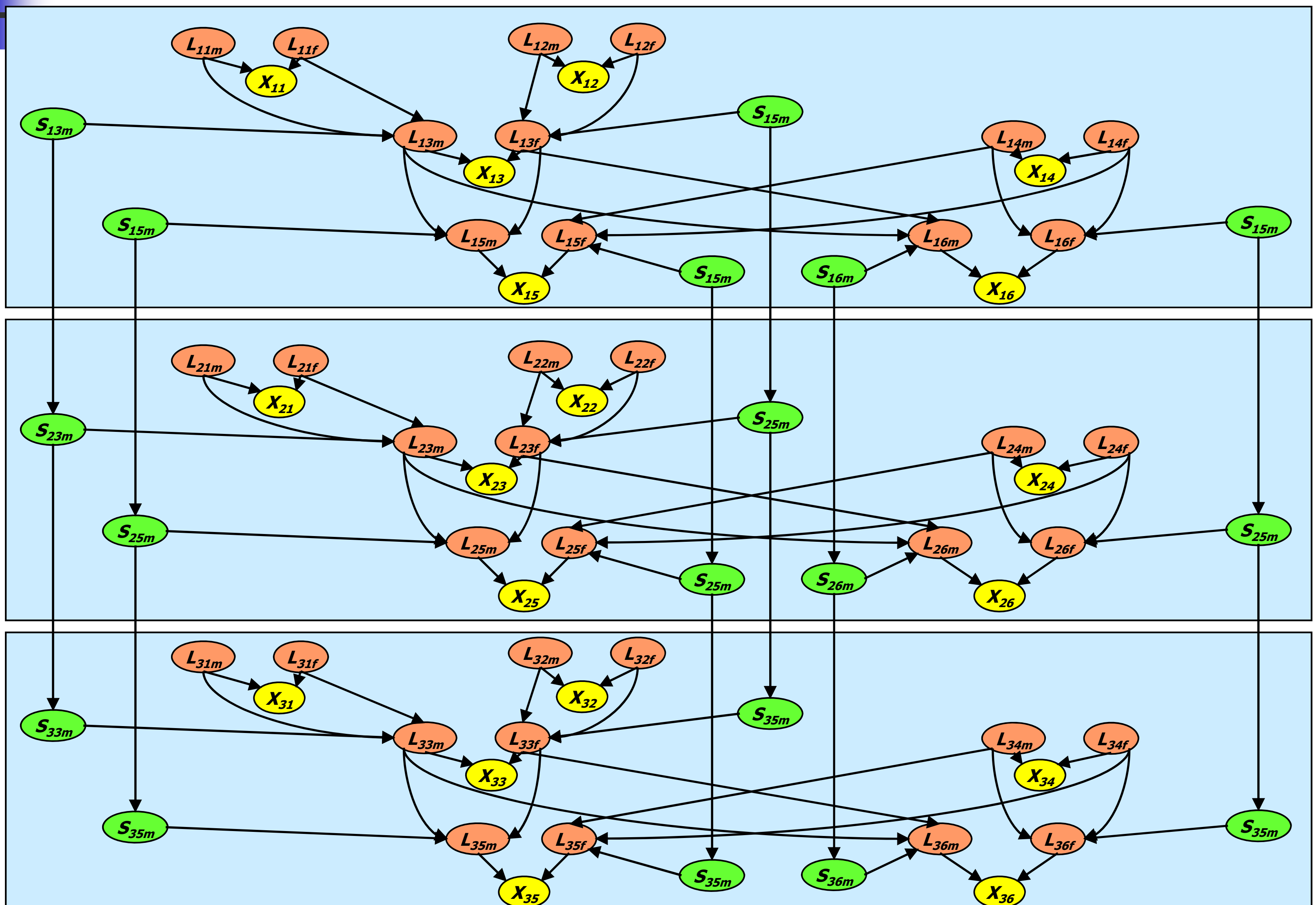
P (lung cancer=yes | smoking=no, dyspnoea=yes) = ?

Linkage Analysis



- *6 individuals*
- *Haplotype: {2, 3}*
- *Genotype: {6}*
- *Unknown*

Pedigree: 6 people, 3 markers



Graphical Models

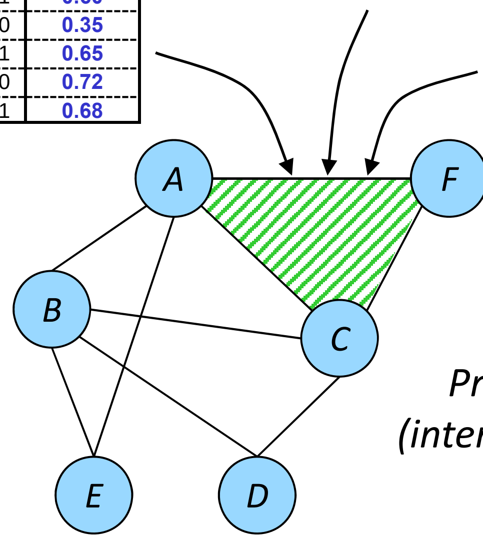
- A graphical model **(X,D,F)**:
  - **X** = $\{X_1,\ldots X_n\}$ variables
  - **D** = $\{D_1, \ldots D_n\}$ domains
  - **F** = $\{f_1,\ldots,f_m\}$ functions
- Operators:
  - combination
  - elimination (projection)
- Tasks:
  - **Belief updating**: $\Sigma_{X-y} \prod_j P_i$
  - **MPE**: $\max_X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

A	C	F	P(F|A,C)
0	0	0	0.14
0	0	1	0.96
0	1	0	0.40
0	1	1	0.60
1	0	0	0.35
1	0	1	0.65
1	1	0	0.72
1	1	1	0.68

Relation

A	C	F
red	green	blue
blue	red	red
blue	blue	green
green	red	blue

$f_i := (F = A + C)$



Primal graph (interaction graph)

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

Probabilistic Inference Tasks

- *Belief updating:*

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

- *Finding most probable explanation (MPE)*

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

- *Finding maximum a-posteriory hypothesis*

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

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

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

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

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

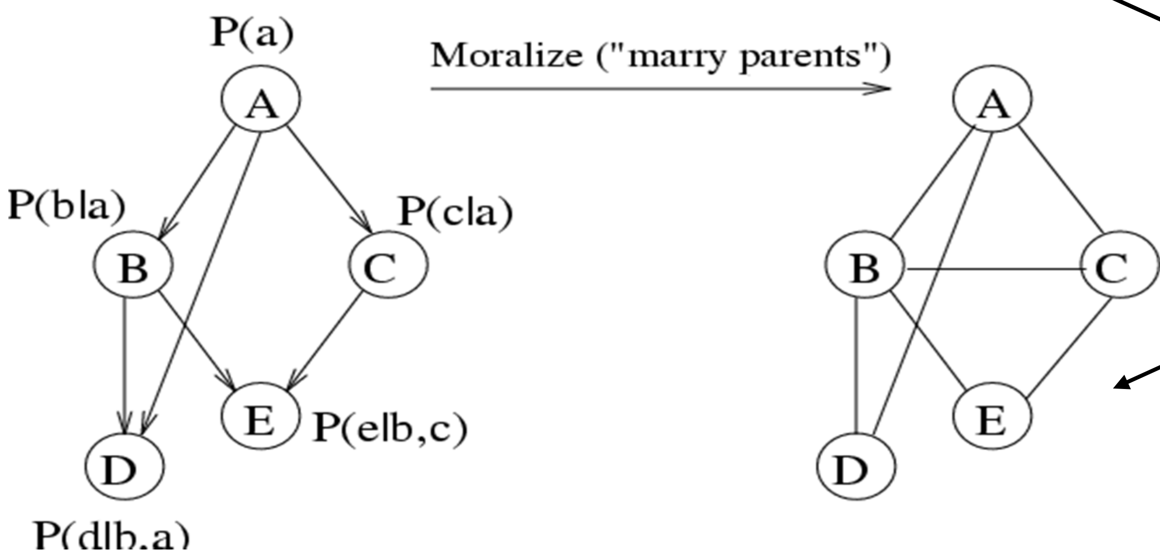
$U(\bar{\mathbf{x}})$: **utility function**

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

"Moral" Graph

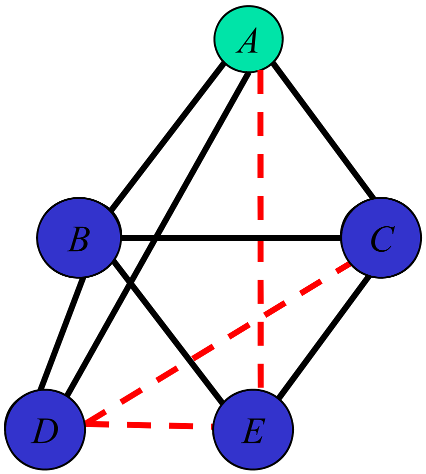
$$P(X_1,\dots,X_n) = \underbrace{\prod_{i=1}^{n} P(X_i \mid parents(X_i))}$$



Conditional Probability Distribution (CPD)

Clique in moral graph ("family")

Belief updating: P(X|evidence)=?



"Moral" graph

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

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

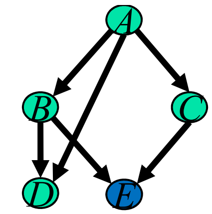
$$= P(a)\sum_{e=0}\sum_{d}\sum_{c}P(c|a)\sum_{b}P(b|a)P(d|b,a)P(e|b,c)$$

Variable Elimination

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

Bucket elimination

Algorithm *BE-bel* (Dechter 1996)



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

$\sum_b \prod$ ← *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: $e=0$ $\lambda^D(a, e)$

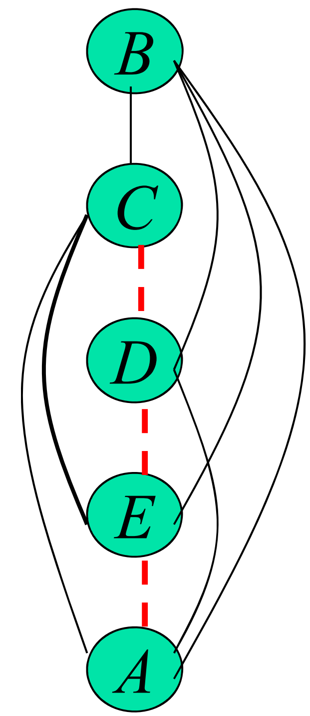
bucket A: $P(a)$ $\lambda^E(a)$

$P(e=0)$

$P(a|e=0)$

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

Thanks to Darwiche

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** $Bel(x_1) = \alpha P(x_1) \cdot \Pi_i \lambda_i(x_1)$

Finding $\text{MPE} = \max_{\bar{x}} P(\bar{x})$

Algorithm *BE-mpe* (Dechter 1996)

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

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

$\max_b \prod$ ← Elimination operator

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

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

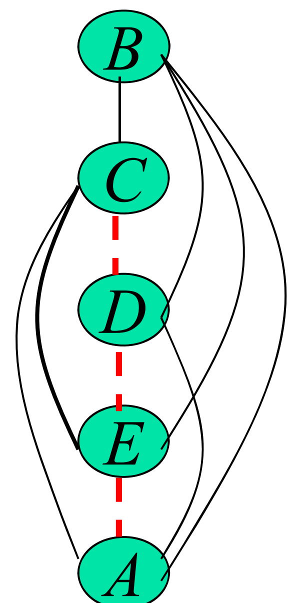
bucket D: $P(c|a)$ $h^C(a,d,e)$

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

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

MPE

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



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

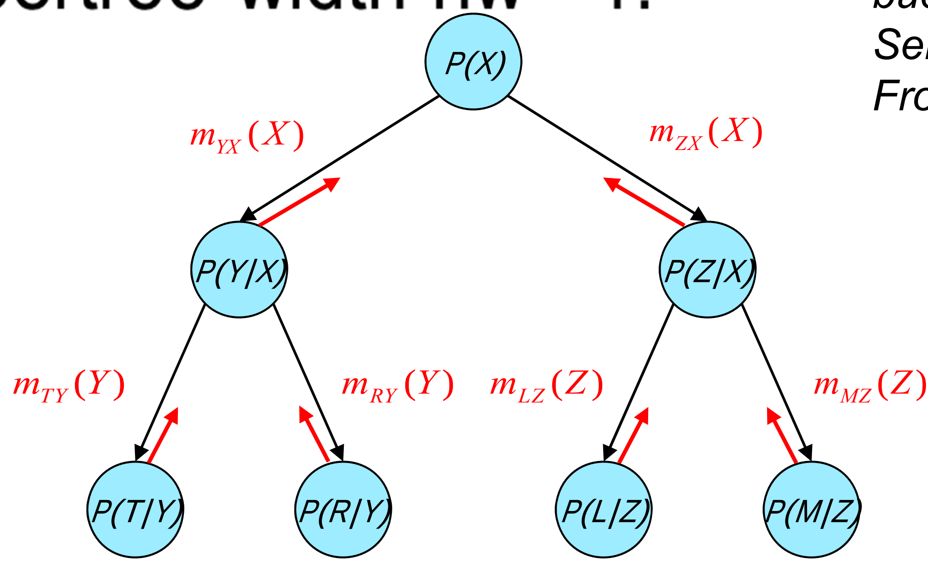
A: $P(a)$ $h^E(a)$

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

Complexity of Bucket-elimination

- Theorem: Bucket-elimination is O(r•k^{w*+1}) time and O(nk^{w*}) space.
- When w=1 then w*=1 → trees
- When we have a tree of functions w=w* and the hypertree width hw =1.

bucket-elimination
Sends messages
From leaves to root



Belief Updating Example

SUM-PROD operators
POLY-TREE structure

H	P(H)
0	.9
1	.1

F	P(F)
0	.99
1	.01

*

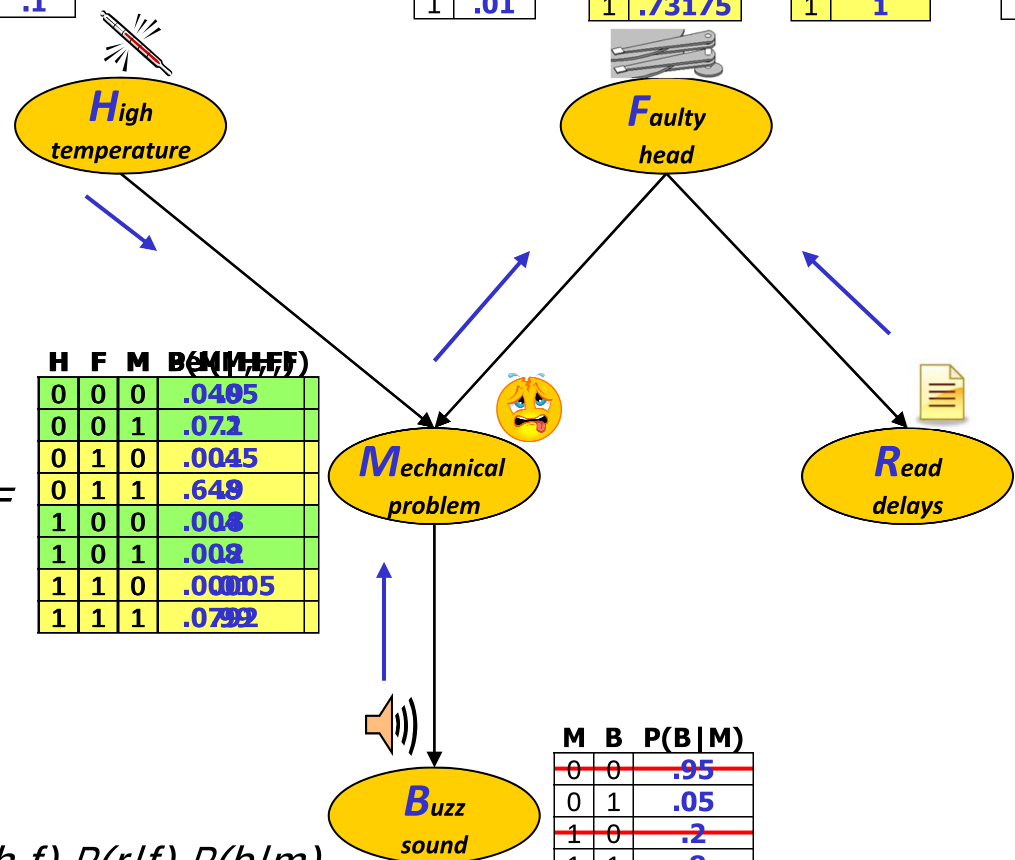
F	h_3(F)
0	.1245
1	.73175

*

F	h_4(F)
0	1
1	1

=

F	P(F,B=1)
0	.123255
1	.073175



H	F	M	P(M|H,F)
0	0	0	.9
0	0	1	.1
0	1	0	.1
0	1	1	.9
1	0	0	.8
1	0	1	.2
1	1	0	.01
1	1	1	.99

*

M	h_1(M)
0	.05
1	.8

*

H	h_2(H)
0	.9
1	.1

=

H	F	M	Bel(M,H,F)
0	0	0	.0405
0	0	1	.072
0	1	0	.0045
0	1	1	.648
1	0	0	.004
1	0	1	.008
1	1	0	.00005
1	1	1	.0792

F	R	P(R|F)
0	0	.8
0	1	.2
1	0	.3
0	1	.7

M	B	P(B|M)
~~0~~	~~0~~	~~.95~~
0	1	.05
~~1~~	~~0~~	~~.2~~
1	1	.8

$P(h,f,r,m,b) = P(h)\,P(f)\,P(m|h,f)\,P(r|f)\,P(b|m)$

$P(F \mid B=1) = ?$

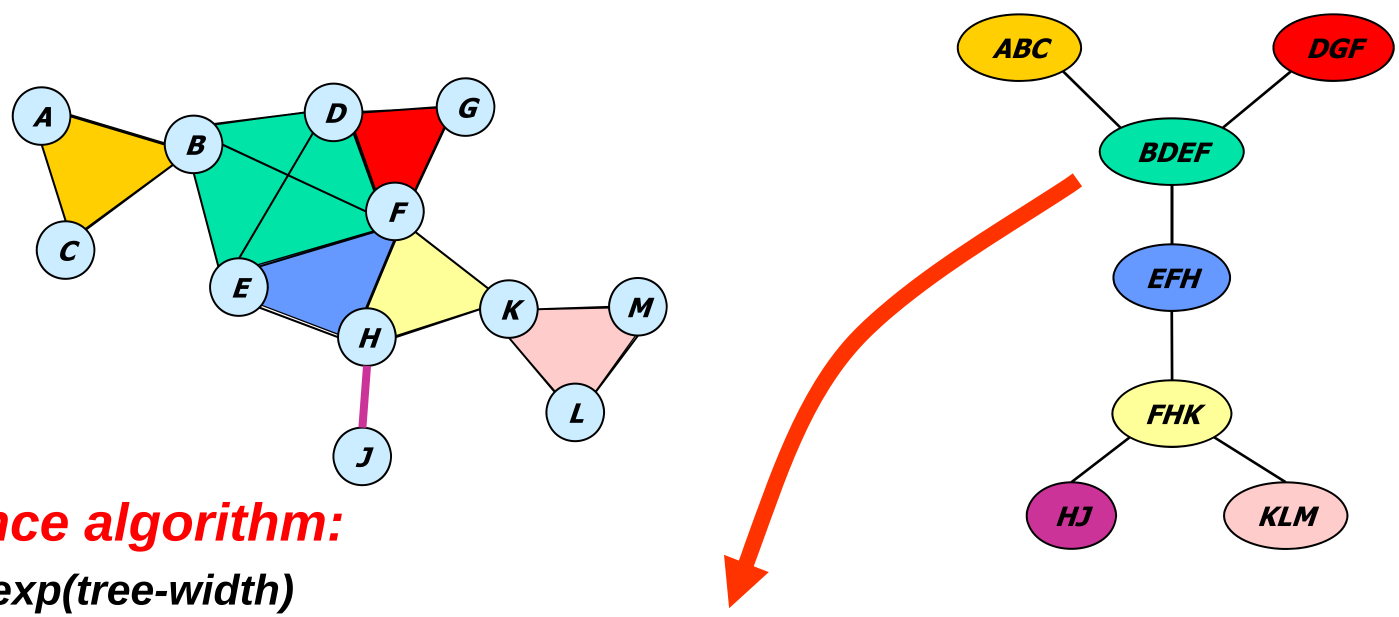
$P(B=1) = .19643$

Probability of evidence

$P(F=1|B=1) = .3725$

Updated belief

Inference and Treewidth



Inference algorithm:

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

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

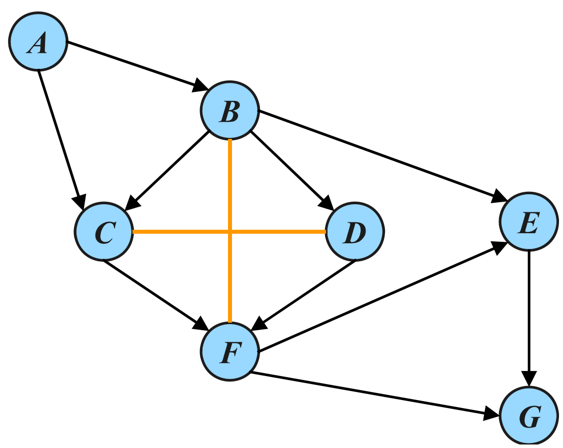
treewidth = 4 - 1 = 3

treewidth = (maximum cluster size) - 1

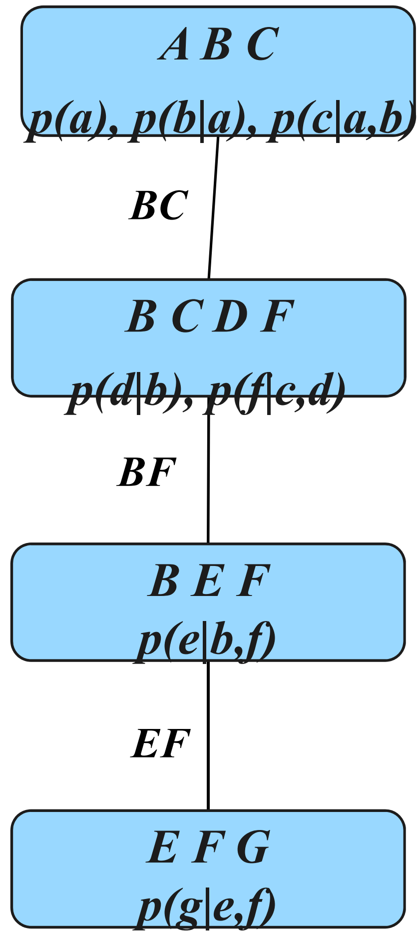
Tree decompositions

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)

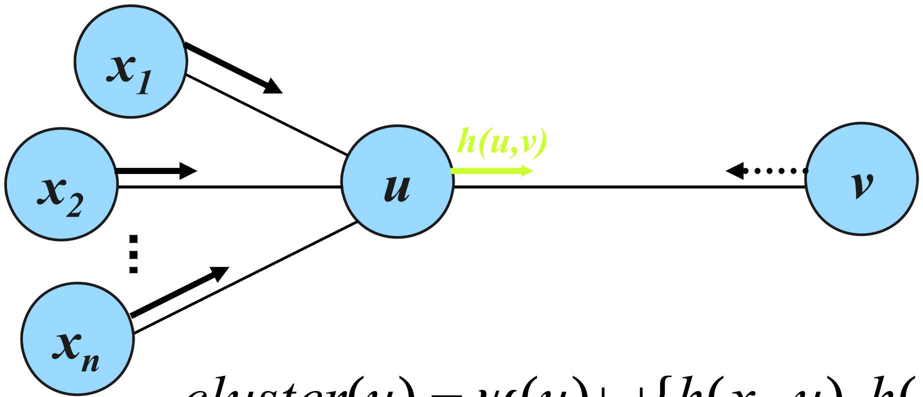


Belief network



Tree decomposition

Belief Propagation

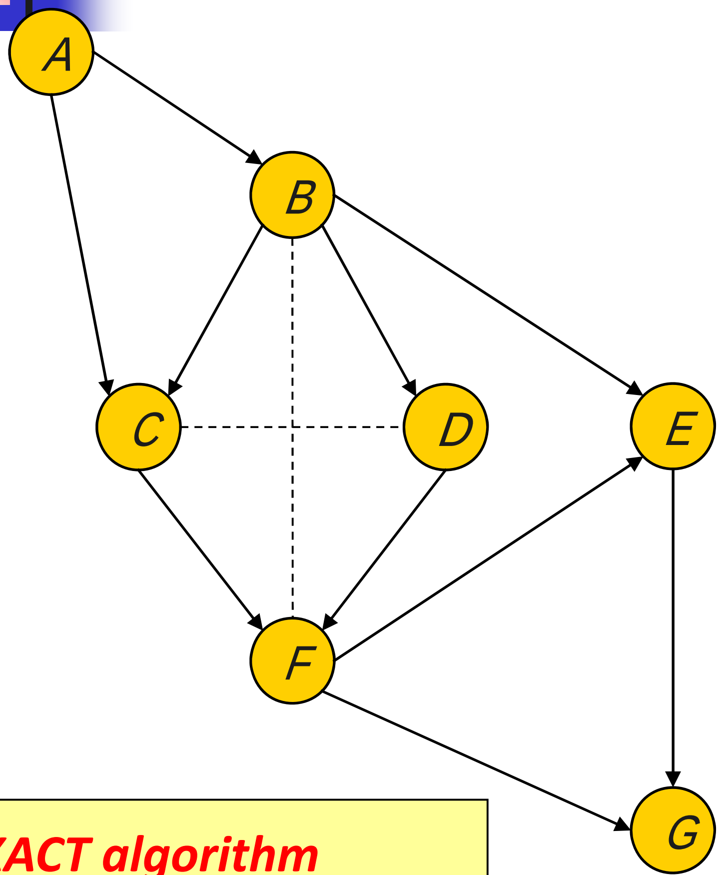


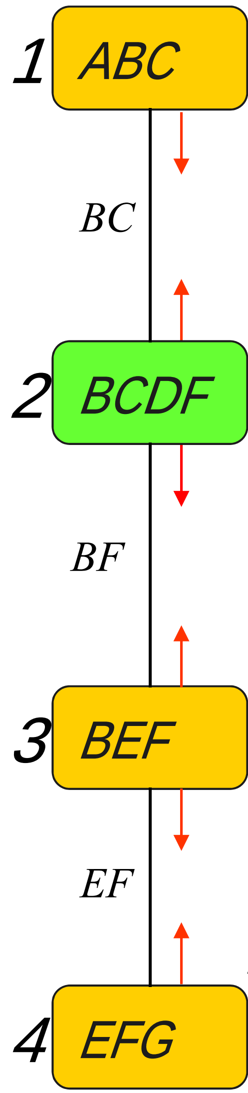
$$cluster(u) = \psi(u) \cup \{h(x_1,u), h(x_2,u), \ldots, h(x_n,u), h(v,u)\}$$

Compute the message :

$$h(u,v) = \sum\nolimits_{elim(u,v)} \prod\nolimits_{f \in cluster(u) - \{h(v,u)\}} f$$

Join-Tree Clustering





$$h_{(1,2)}(b,c) = \sum_{a} p(a) \cdot p(b \mid a) \cdot p(c \mid a,b)$$

$$h_{(2,1)}(b,c) = \sum_{d,f} p(d \mid b) \cdot p(f \mid c,d) \cdot h_{(3,2)}(b,f)$$

$$h_{(2,3)}(b,f) = \sum_{c,d} p(d \mid b) \cdot p(f \mid c,d) \cdot h_{(1,2)}(b,c)$$

$$h_{(3,2)}(b,f) = \sum_{e} p(e \mid b,f) \cdot h_{(4,3)}(e,f)$$

$$h_{(3,4)}(e,f) = \sum_{b} p(e \mid b,f) \cdot h_{(2,3)}(b,f)$$

$$h_{(4,3)}(e,f) = p(G = g_e \mid e,f)$$

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:
  - $O(\, deg \times (n+N) \times d^{\,w^*+1})$

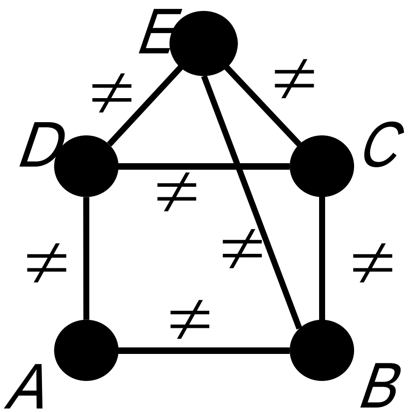
- Space complexity: $O(\, N \times d^{\,sep})$

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

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



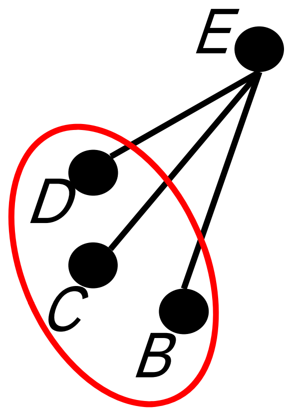
E : E ≠ D, E ≠ C, E ≠ B

D : D ≠ C, D ≠ A

C : C ≠ B

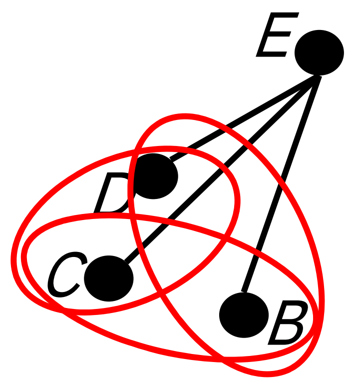
B : A ≠ B

A :



Adaptive

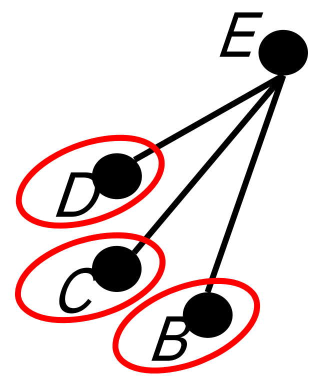
R_{DCB}



d-path

R_{DC}, R_{DB}

R_{CB}



d-arc

R_D

R_C

R_D

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

Local computation: bound the size of recorded dependencies

Split a bucket into mini-buckets =>bound complexity

$$\text{bucket}(X) = \{h_1, \ldots, h_r, h_{r+1}, \ldots, h_n\}$$

$$h^X = \max_X \prod_{i=1}^{n} h_i$$

$$\{h_1, \ldots, h_r\} \qquad \{h_{r+1}, \ldots, h_n\}$$

$$g^X = \left(\max_X \prod_{i=1}^{r} h_i\right) \cdot \left(\max_X \prod_{i=r+1}^{n} h_i\right)$$

$$h^X \le g^X$$

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

Mini-Bucket Elimination

$max_B \prod$ (over P(E|B,C)) $max_B \prod$ (over P(B|A) P(D|A,B))

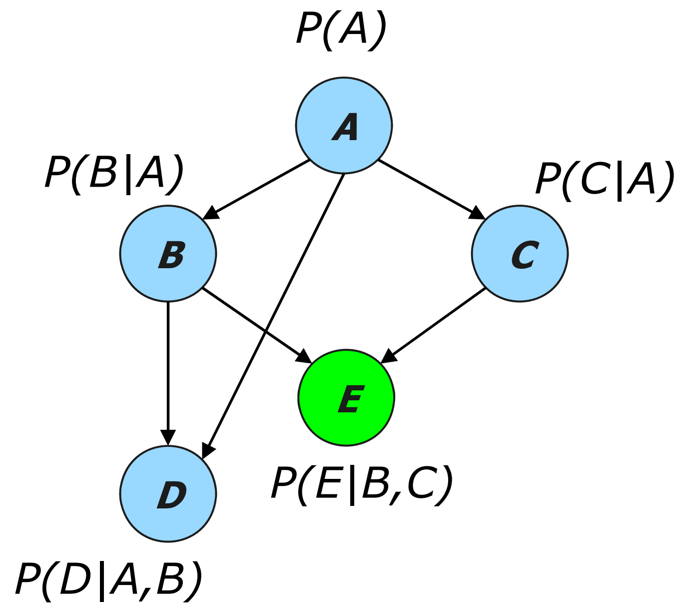
Bucket B: P(E|B,C) P(B|A) P(D|A,B)

Bucket C: P(C|A) $h^B(C,E)$

Bucket D: $h^B(A,D)$

Bucket E: E = 0 $h^C(A,E)$

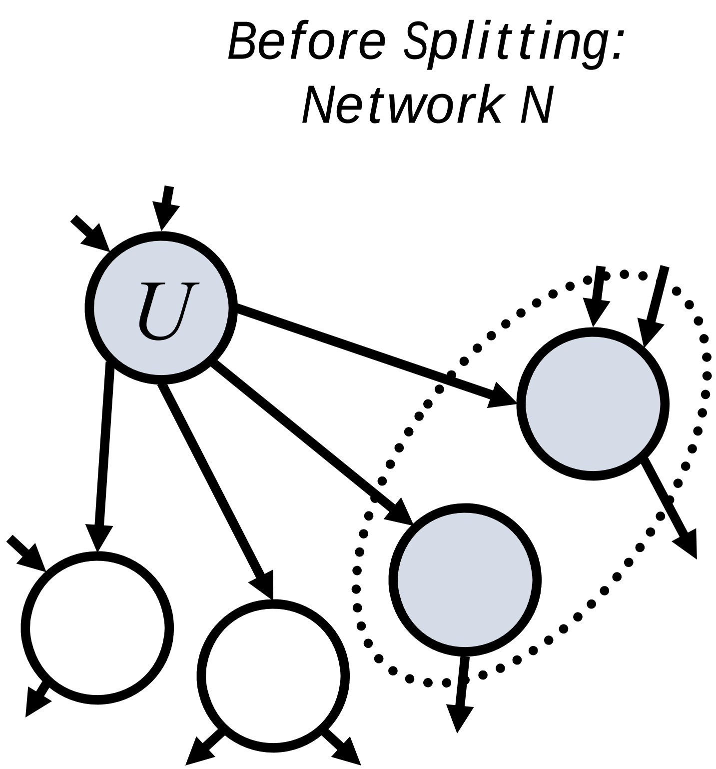
Bucket A: P(A) $h^E(A)$ $h^D(A)$

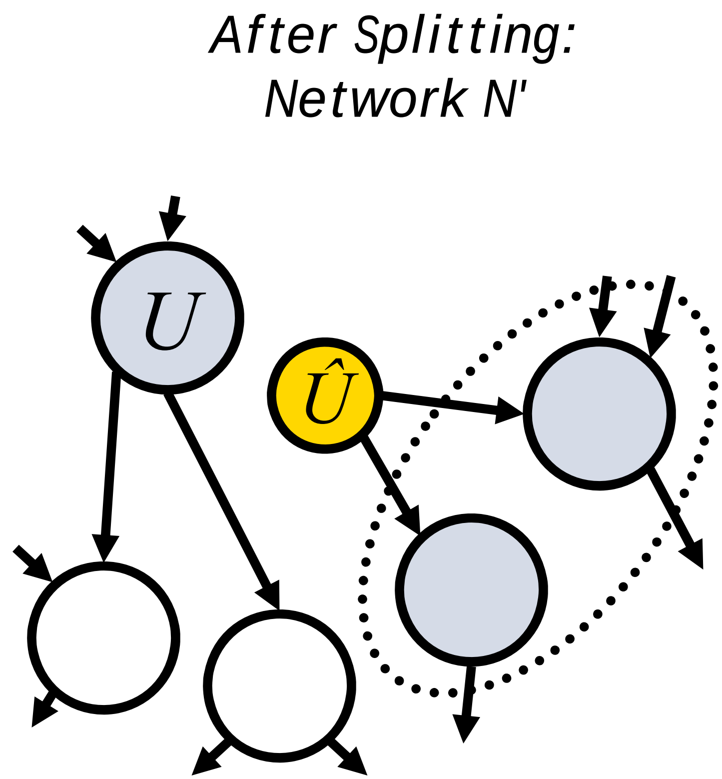


MPE* is an upper bound on MPE --U
Generating a solution yields a lower bound--L

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)

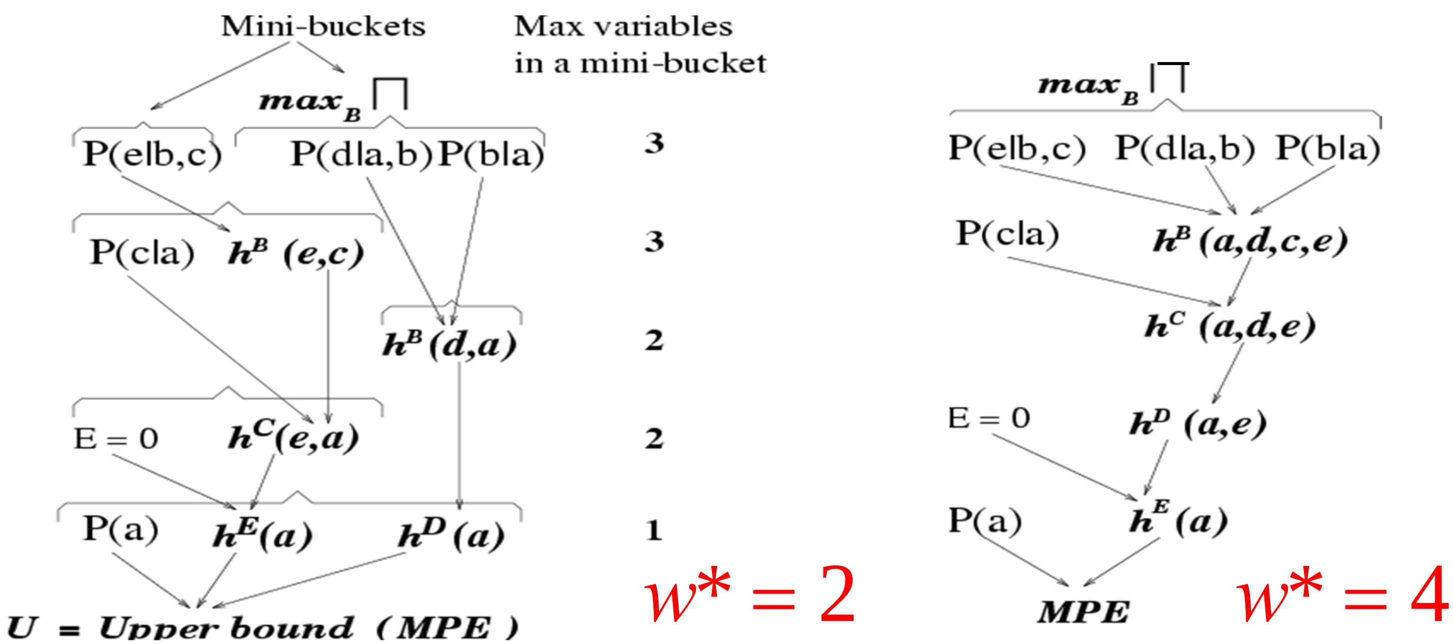




MBE(i) (Dechter and Rish 1997)

- **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

APPROXIMATE algorithm

$$\underbrace{\{h_1,\ldots,h_r,h_{r+1},\ldots,h_n\}}$$

$$\underbrace{\{h_1,\ldots,h_r\}} \qquad \underbrace{\{h_{r+1},\ldots,h_n\}}$$

$$\sum_{elim}\prod_{i=1}^{n} h_i \leq \left(\sum_{elim}\prod_{i=1}^{r} h_i\right)\cdot\left(\sum_{elim}\prod_{i=r+1}^{n} h_i\right)$$

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:

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

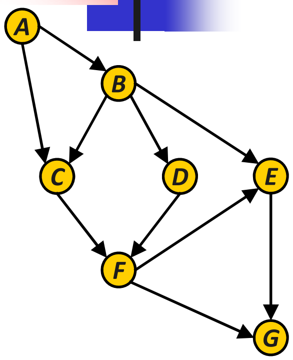
$$\sum_X f(x) \bullet g(x) \le \sum_X f(x) \bullet \max_X g(X)$$

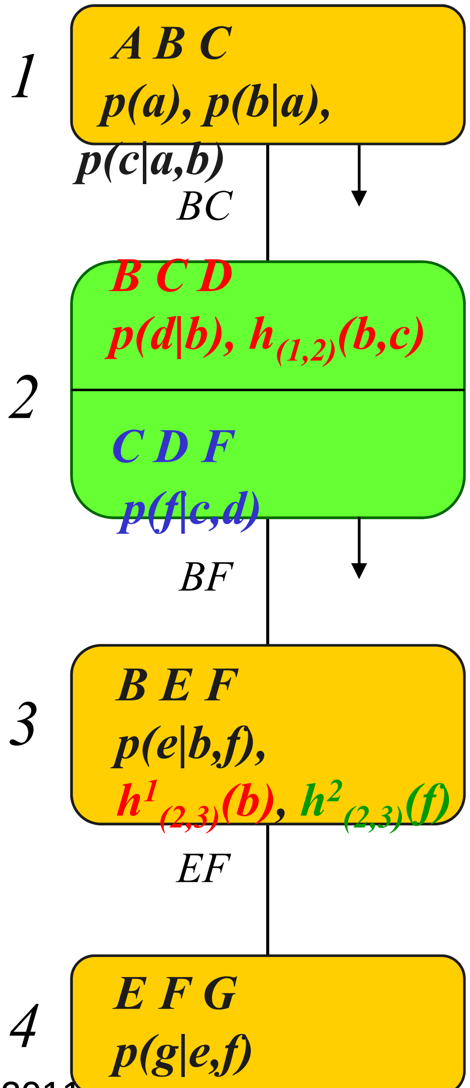
- So we can apply a sum in each mini-bucket, or better, one sum and the rest max, or min (for lower-bound)

- MBE-bel-max(i,m), MBE-bel-min(i,m) generating upper and lower-bound on beliefs approximates BE-bel

- MBE-map(i,m): 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

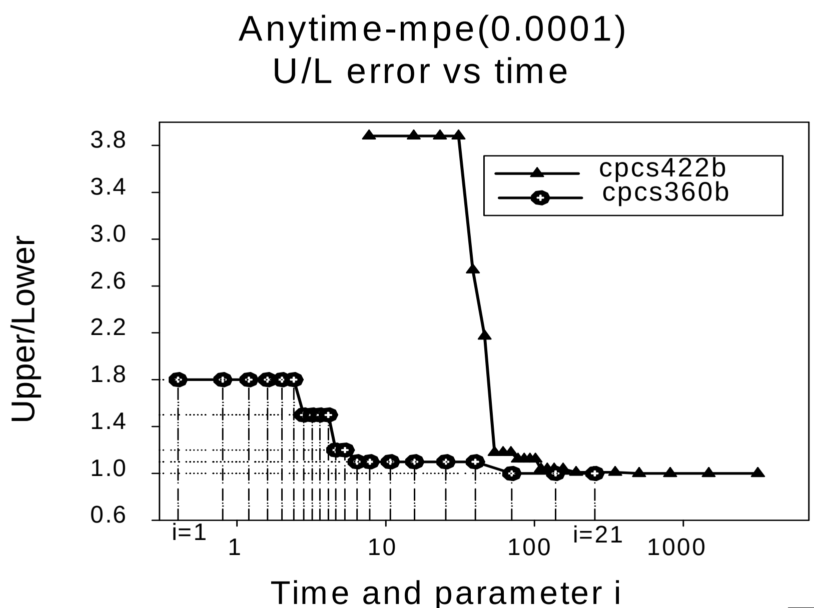
if $1 \leq \frac{U}{L} \leq 1 + \varepsilon$, return solution

end

return the largest L and the smallest U

CPCS networks – medical diagnosis (noisy-OR CPD's)

Test case: no evidence



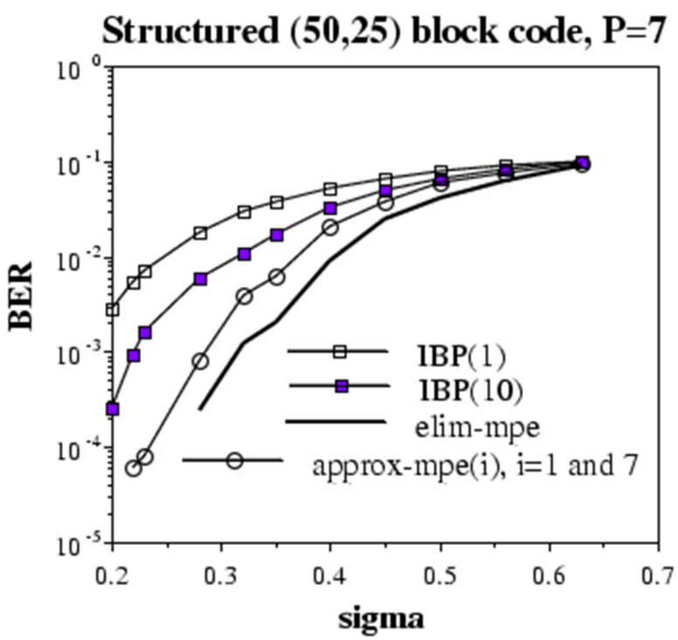
Time (sec)

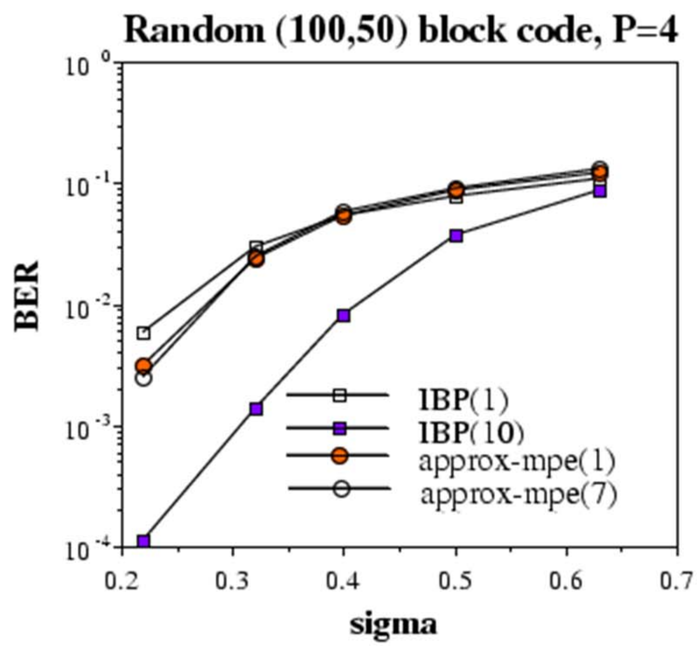
Algorithm	*cpcs360*	*cpcs422*
elim-mpe	*115.8*	*1697.6*
anytime-mpe(ε), $\epsilon = 10^{-4}$	*70.3*	*505.2*
anytime-mpe(ε), $\epsilon = 10^{-1}$	*70.3*	*110.5*

MBE-mpe vs. IBP

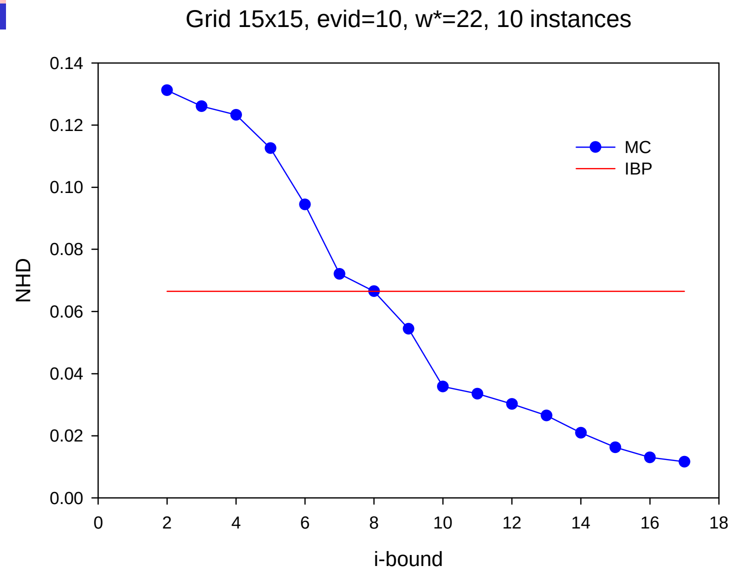
approx - mpe is better on low - w * codes
IBP is better on randomly generated (high - w*) codes

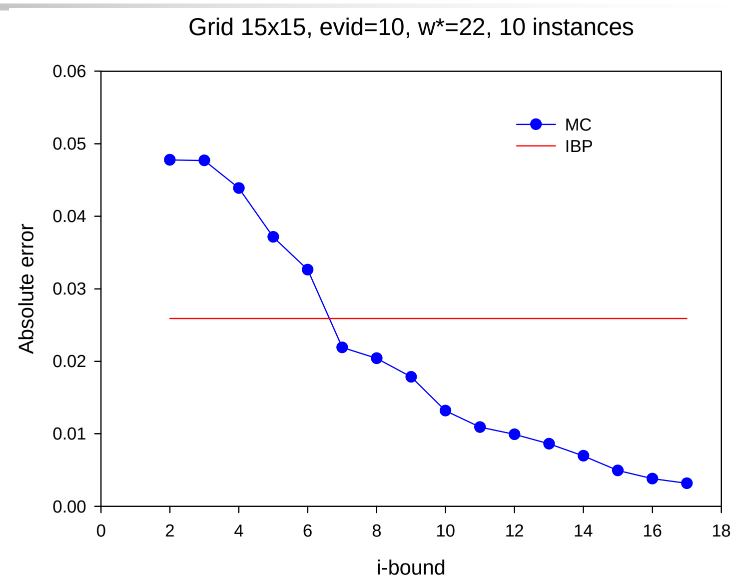
Bit error rate (BER) as a function of noise (sigma):

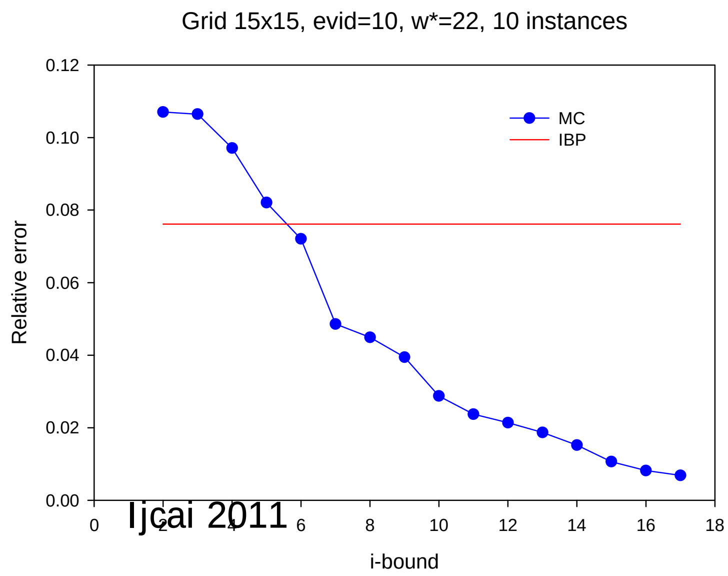


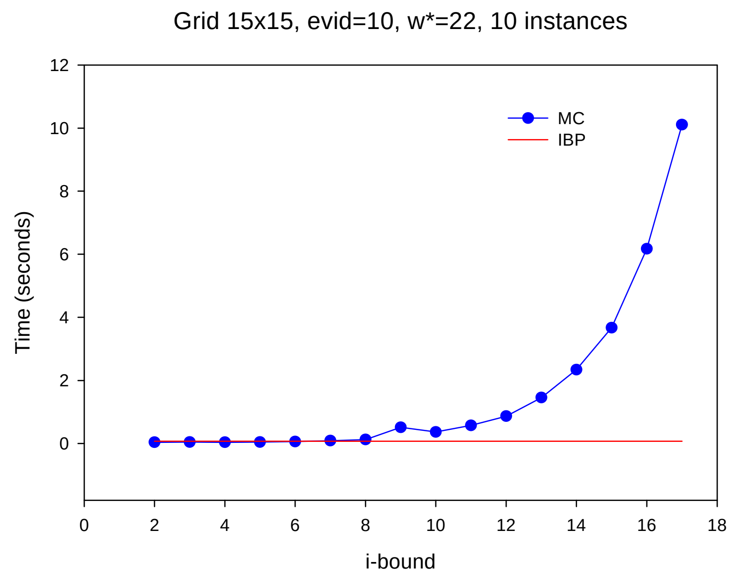


Grid 15x15 - 10 evidence





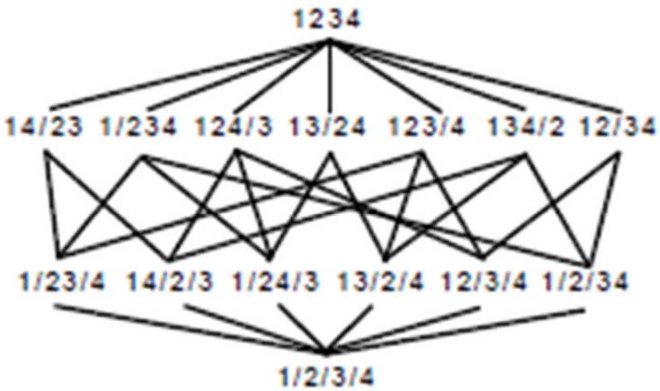




Heuristics for partitioning

(Dechter and Rish, 2003, Rollon and Dechter 2010)

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*:

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

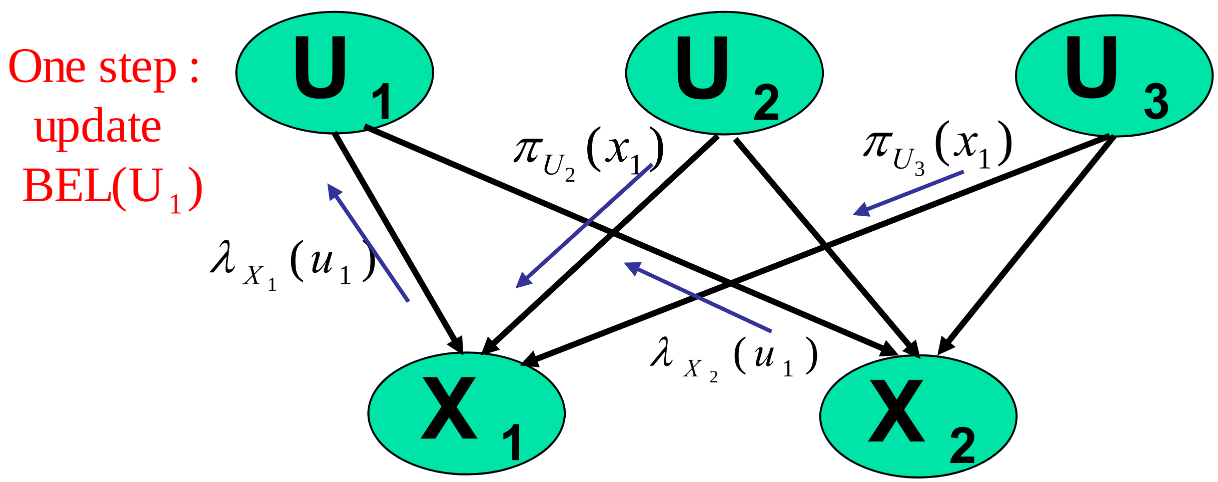
Use greedy heuristic derived from a distance function to decide which functions go into a single mini-bucket

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

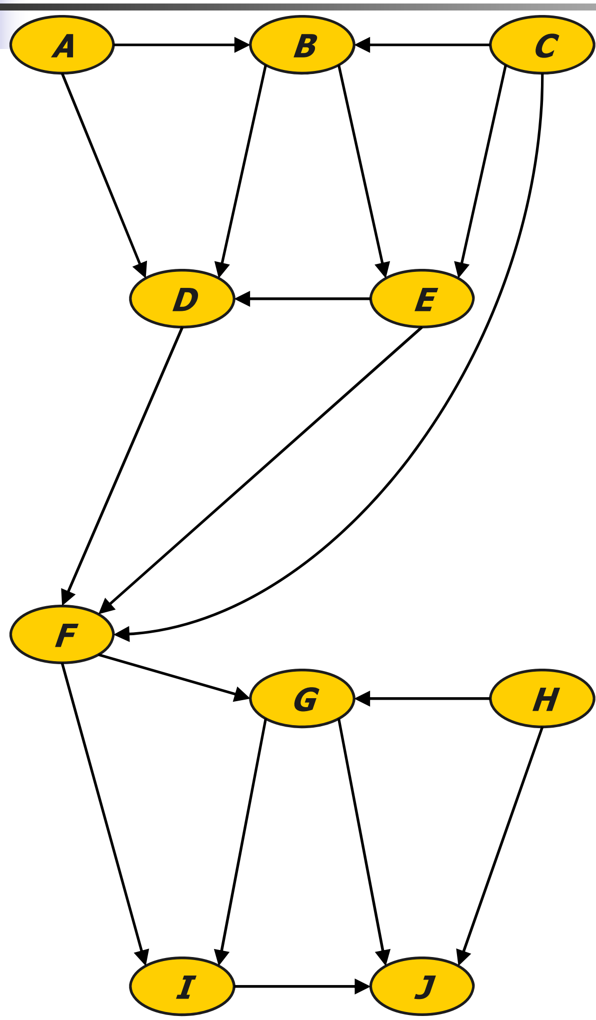


- No guarantees for convergence
- Works well for many coding networks
- Lets combine iterative-nature with anytime--IJGP

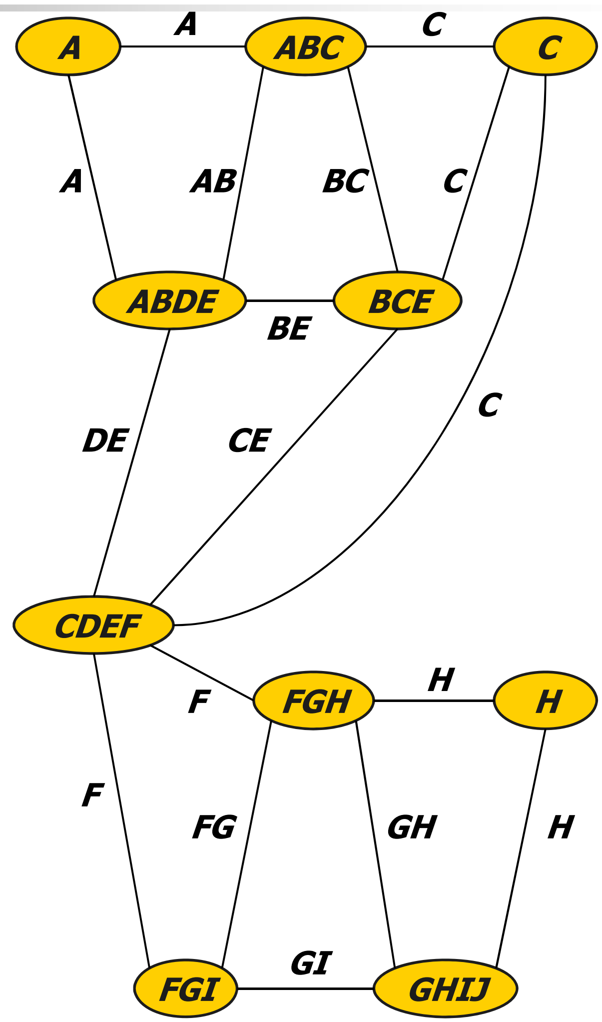
IJGP: a generalized Belief Propagation

- 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



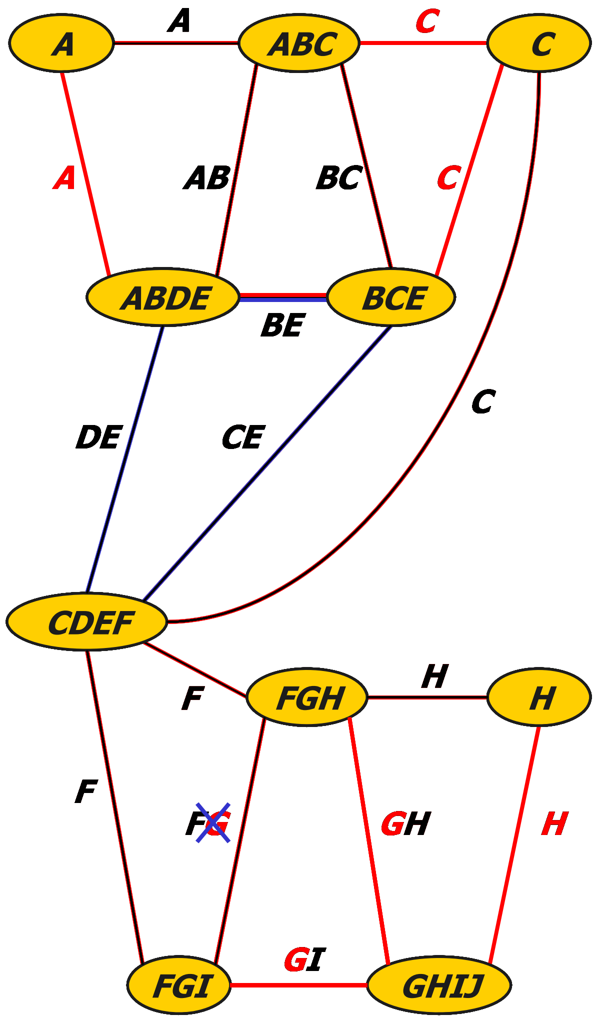
Belief network



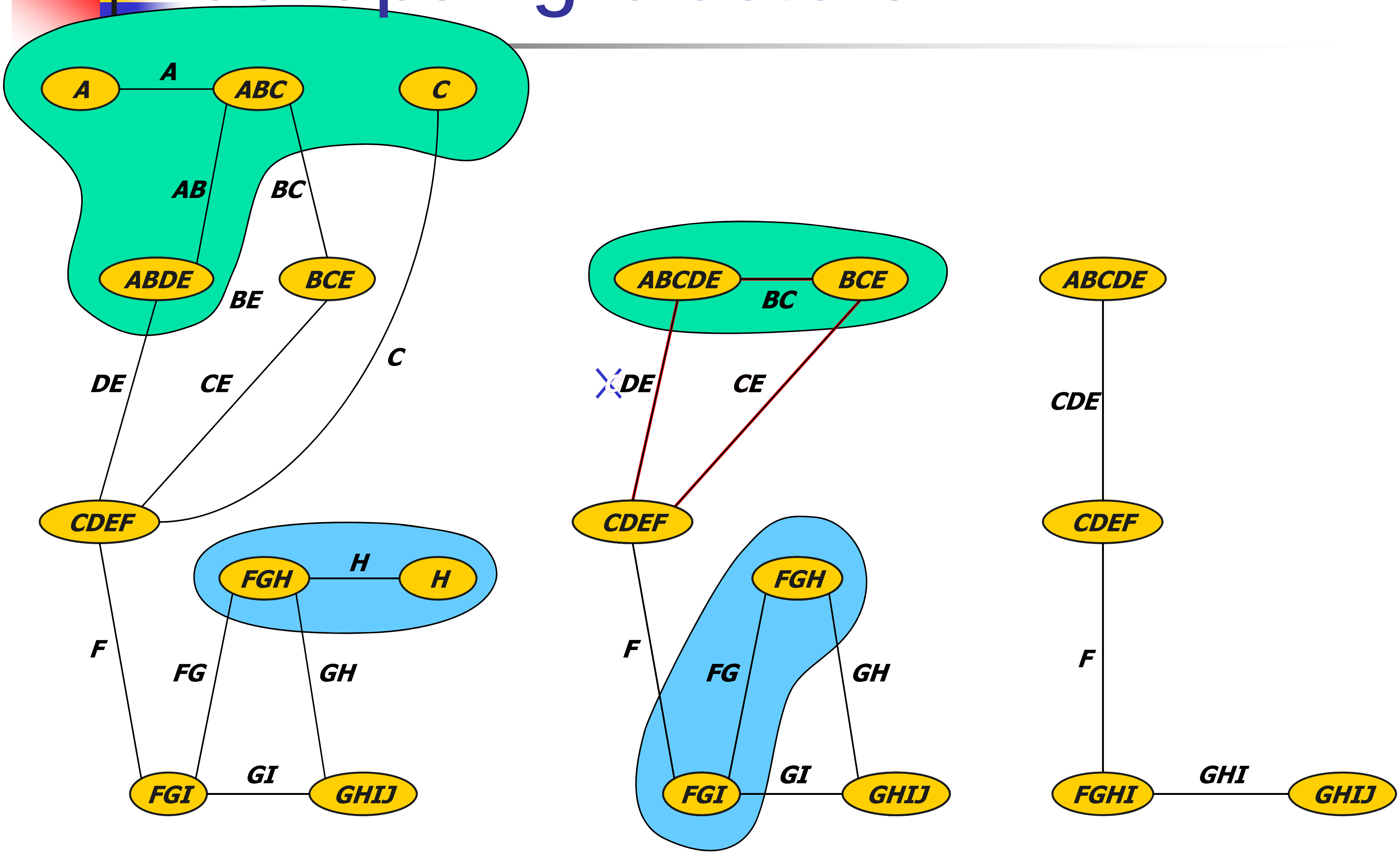
Loopy BP graph

Arc-Minimal Join-Graph

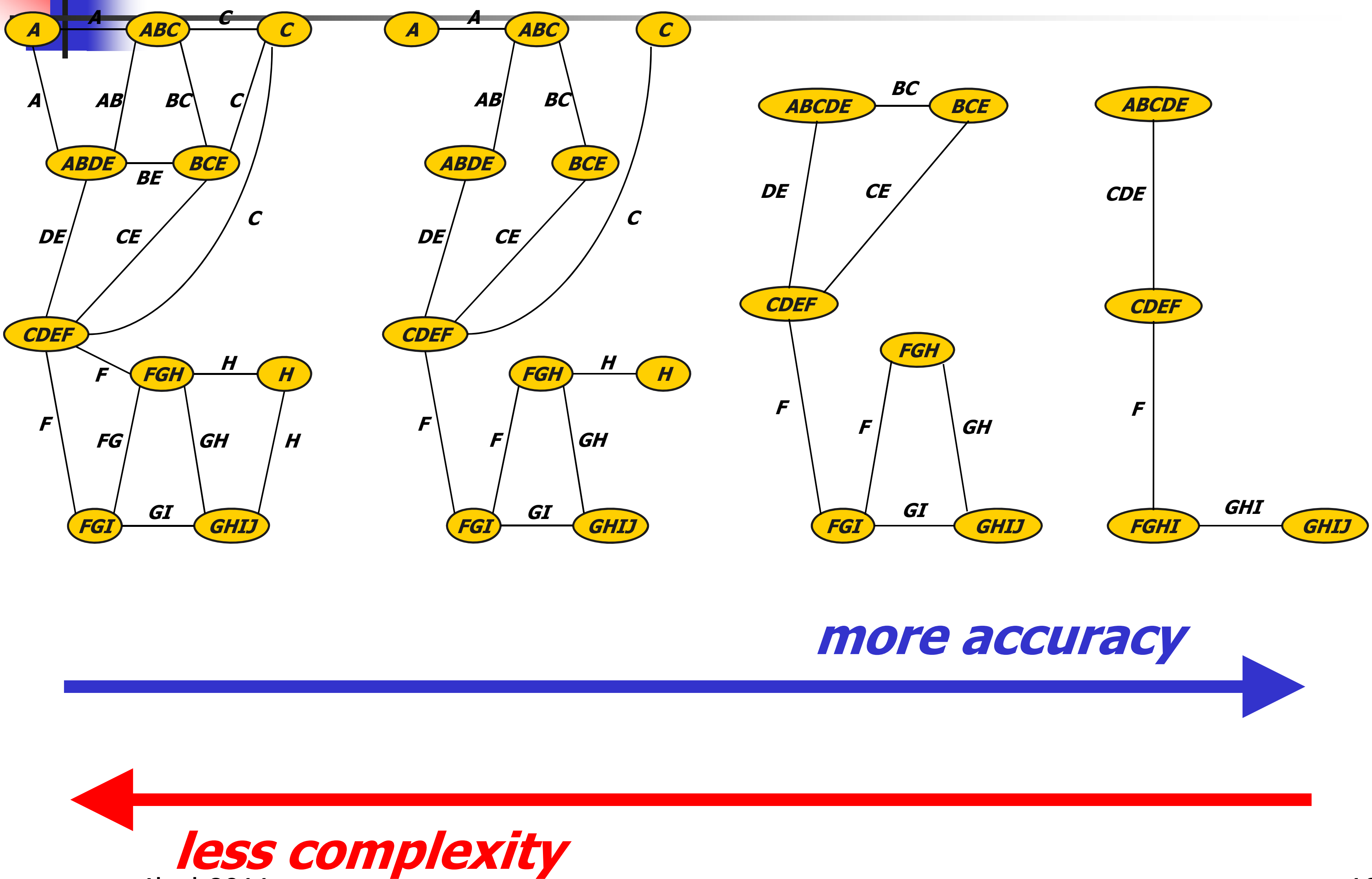
Arcs labeled with any single variable should form a TREE



Collapsing Clusters

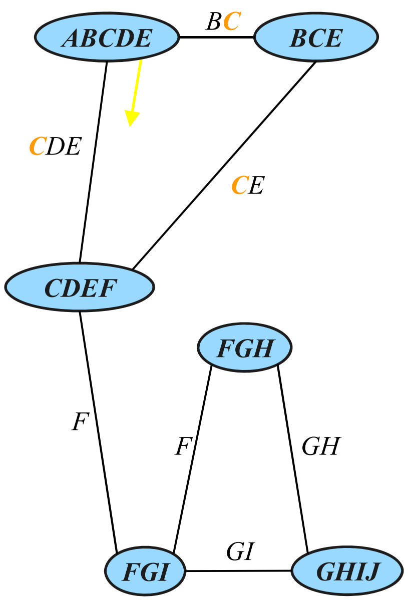


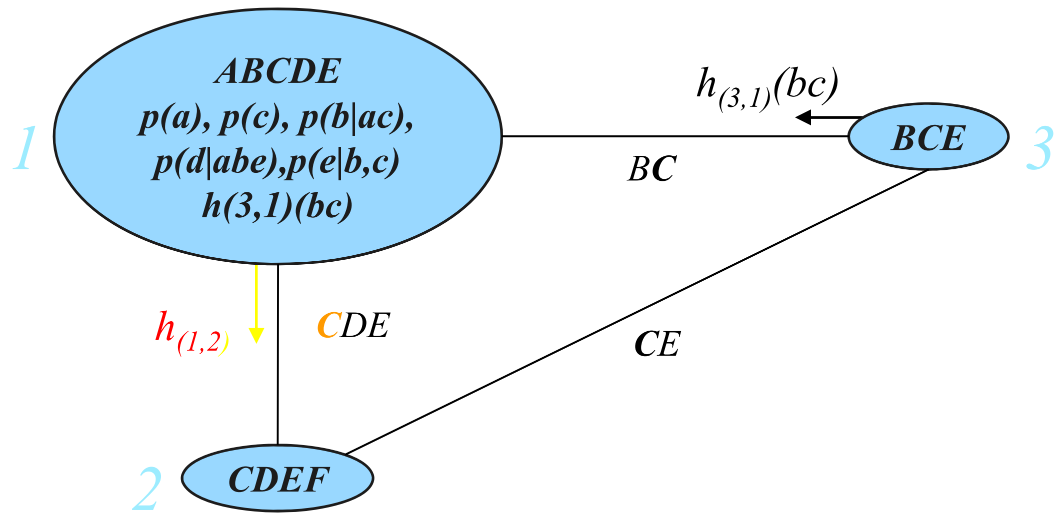
Join-Graphs



A
ABC
C
ABDE
BCE
CDEF
FGH
H
FGI
GHIJ
ABCDE
FGHI
A
AB
BC
C
BE
DE
CE
F
FG
GH
GI
H
CDE
GHI
more accuracy
less complexity

Message propagation





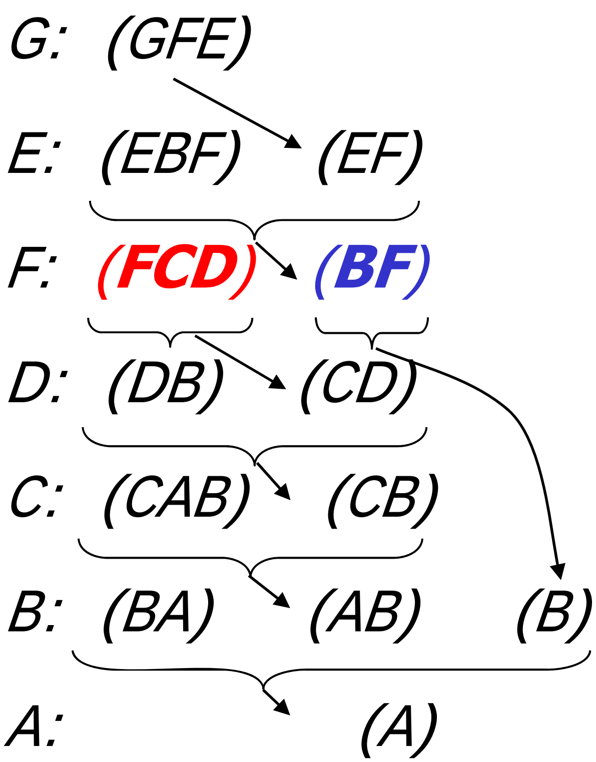
Minimal arc-labeled:
sep(1,2)={D,E}
elim(1,2)={A,B,C}

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

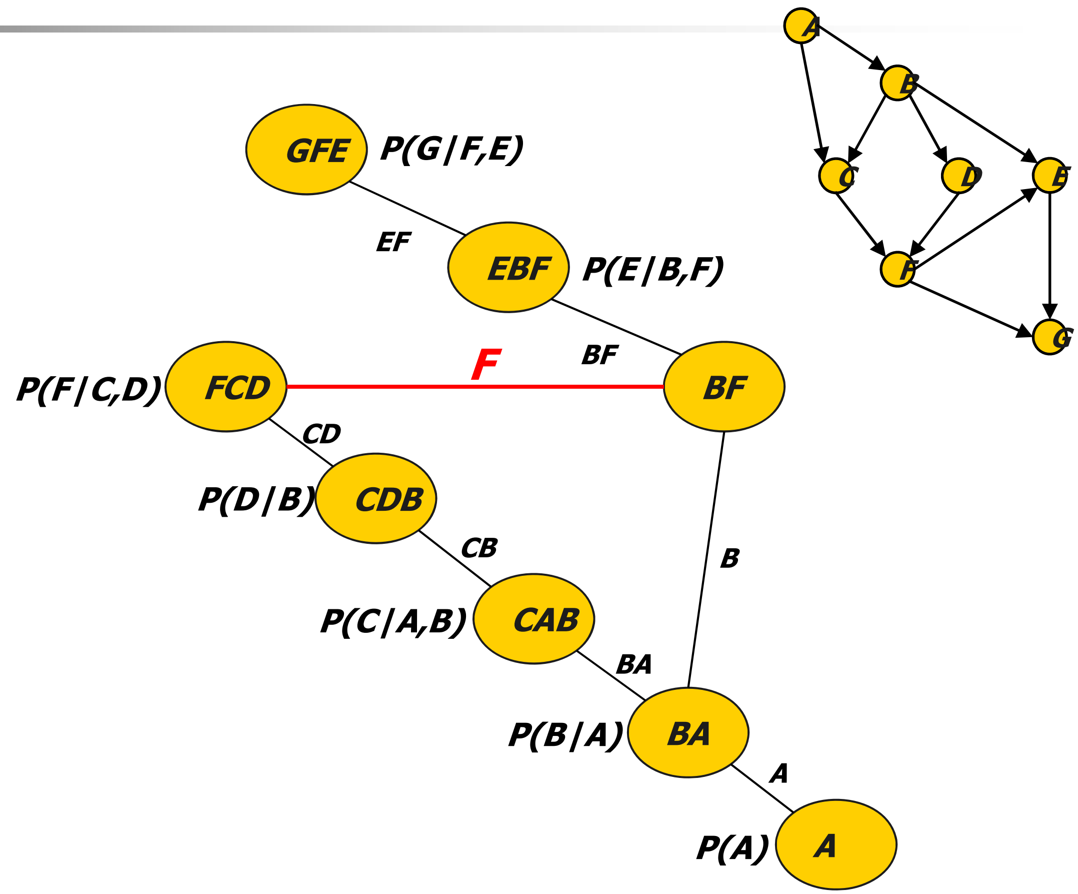
Non-minimal arc-labeled:
sep(1,2)={C,D,E}
elim(1,2)={A,B}

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

Constructing Join-Graphs

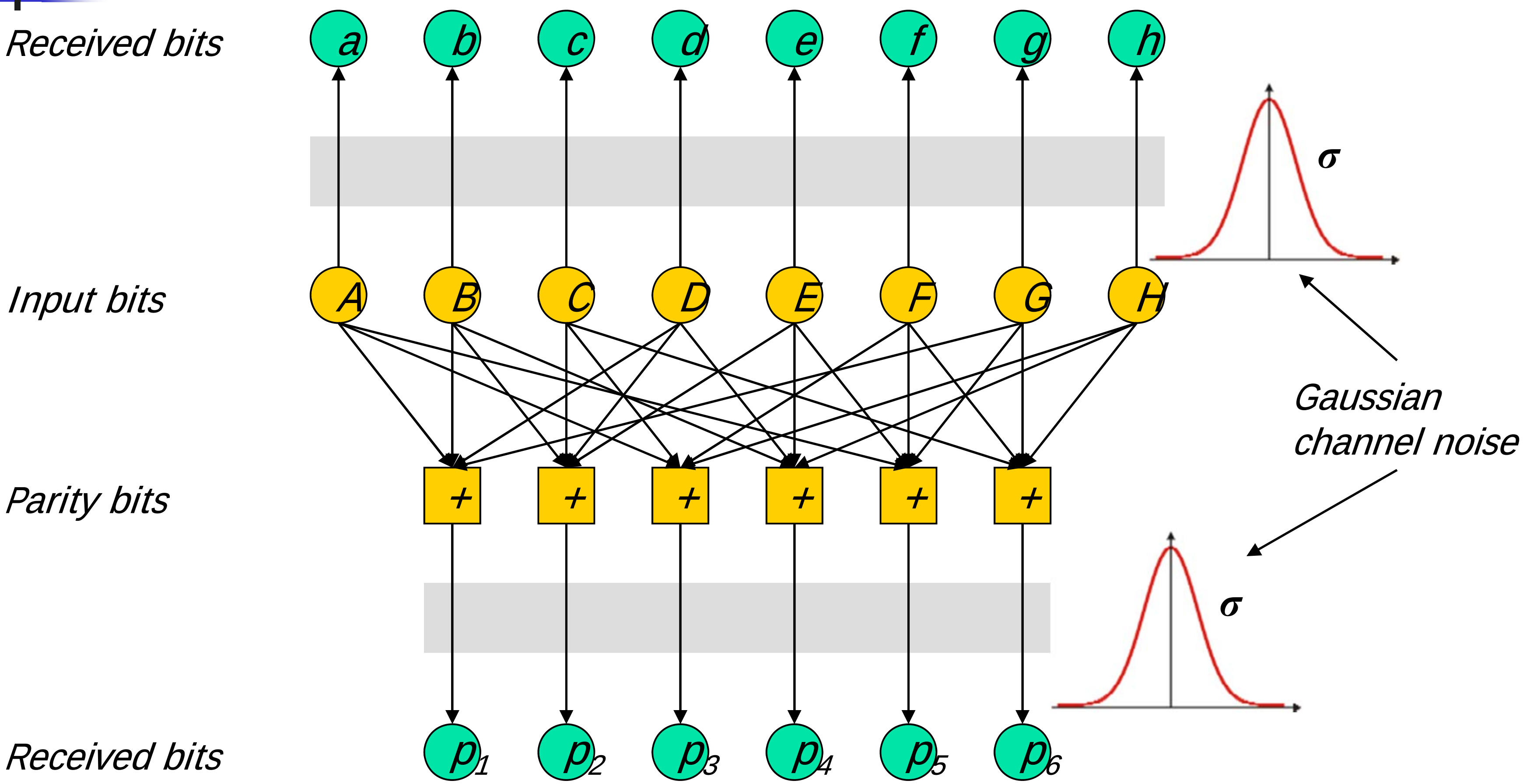


a) schematic mini-bucket(i), i=3



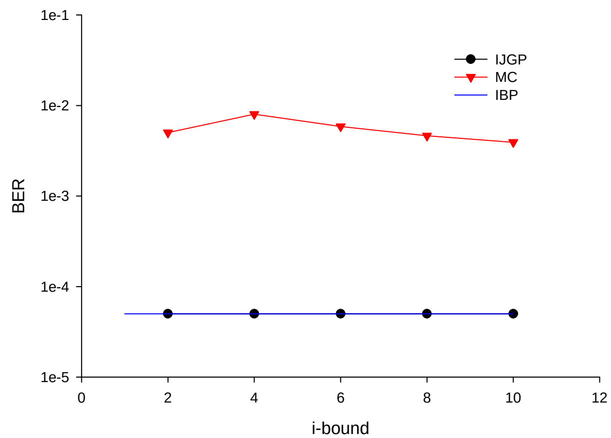
b) arc-labeled join-graph decomposition

Linear Block Codes

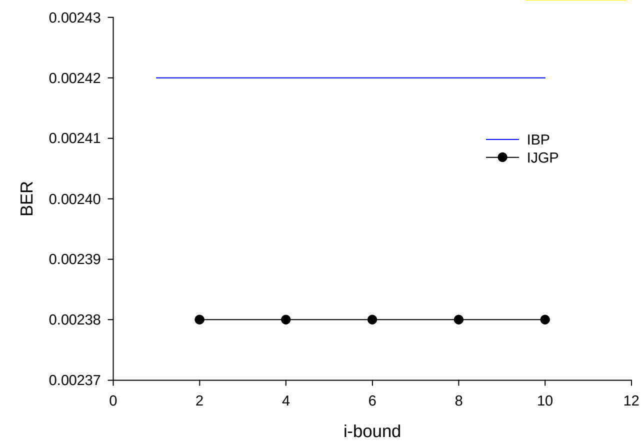


Coding Networks – Bit Error Rate

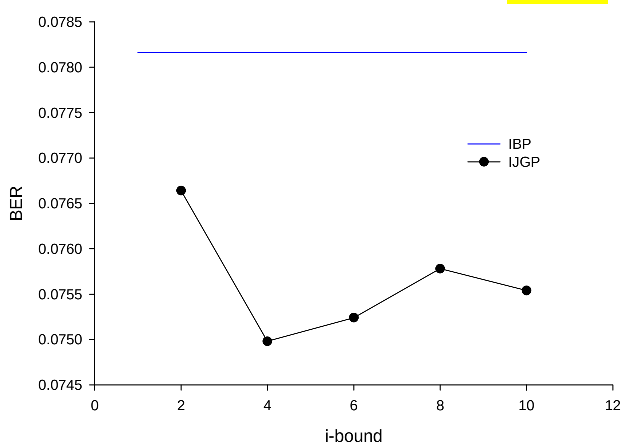
N=400, 1000 instances, 30 it, w*=43, $\sigma = .22$



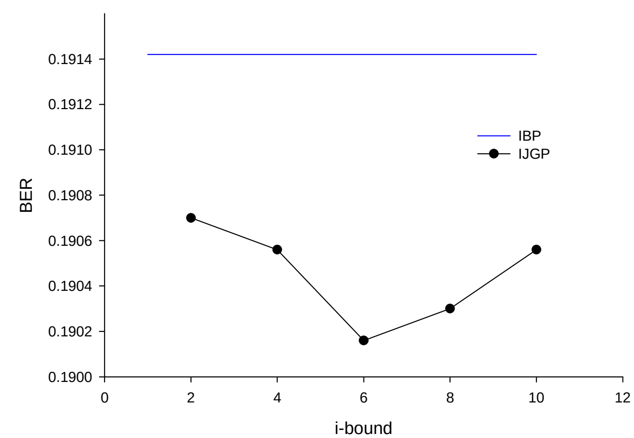
N=400, 500 instances, 30 it, w*=43, $\sigma = .32$



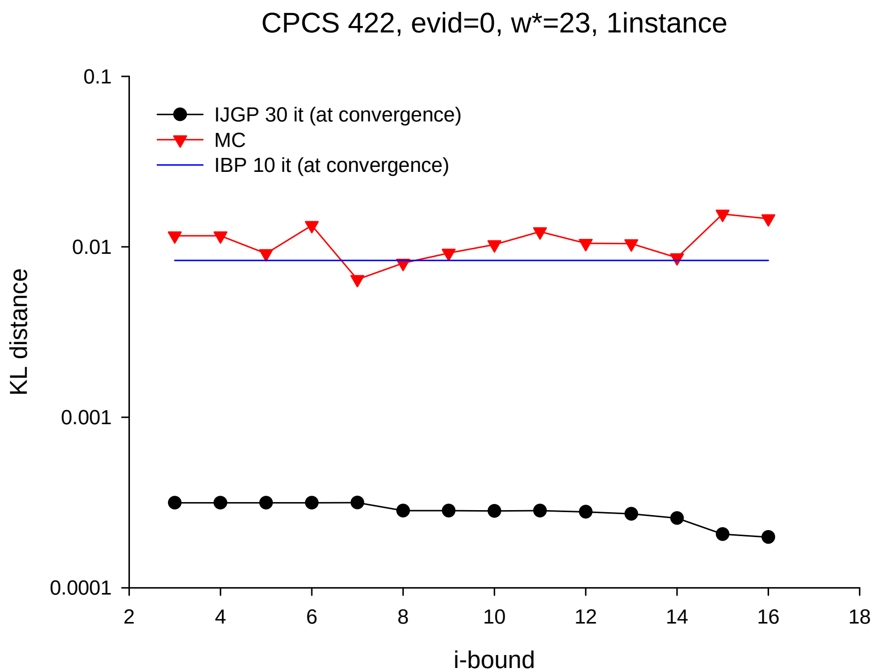
N=400, 500 instances, 30 it, w*=43, $\sigma = .51$

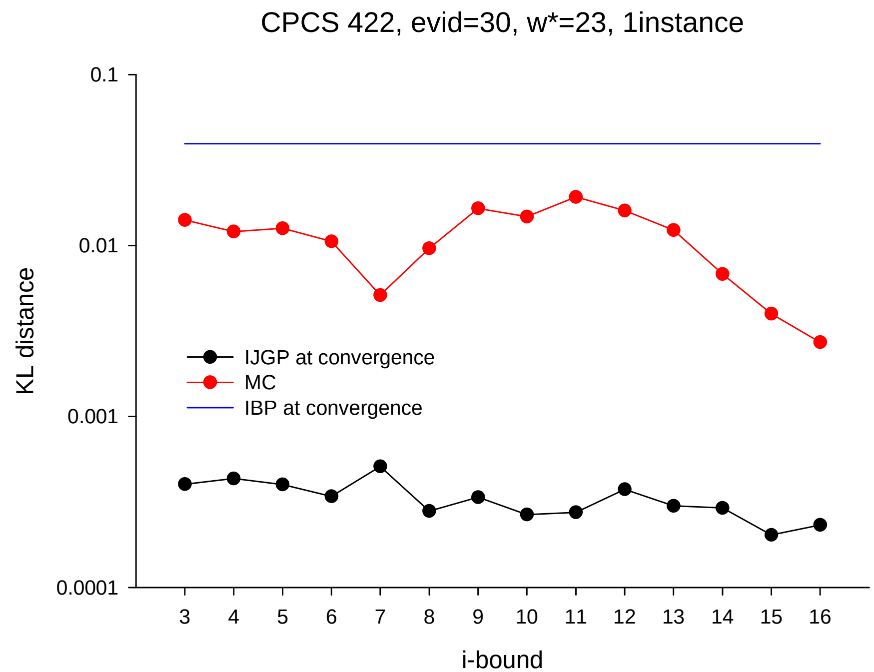


N=400, 500 instances, 30 it, w*=43, $\sigma = .65$



CPCS 422 – KL Distance

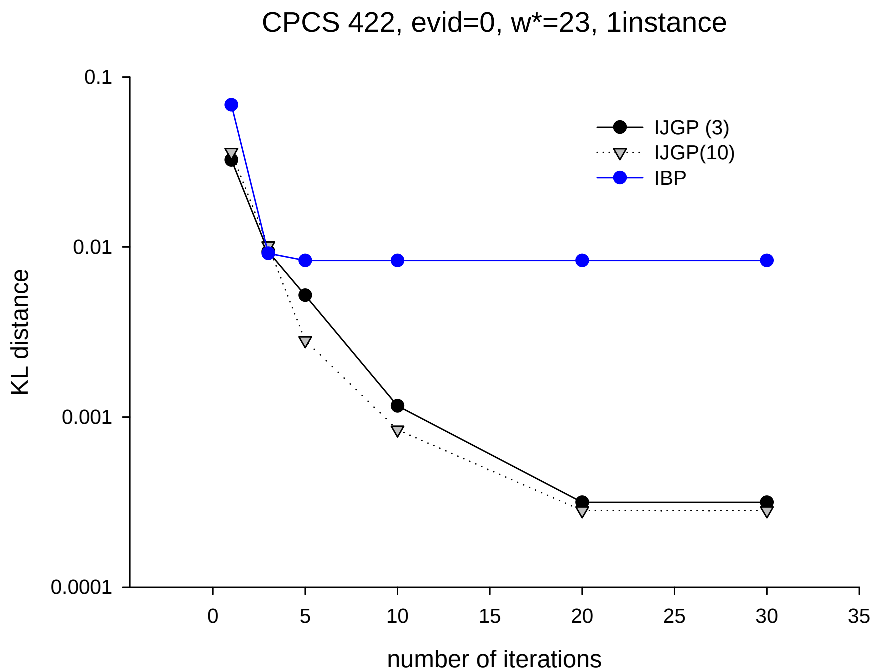




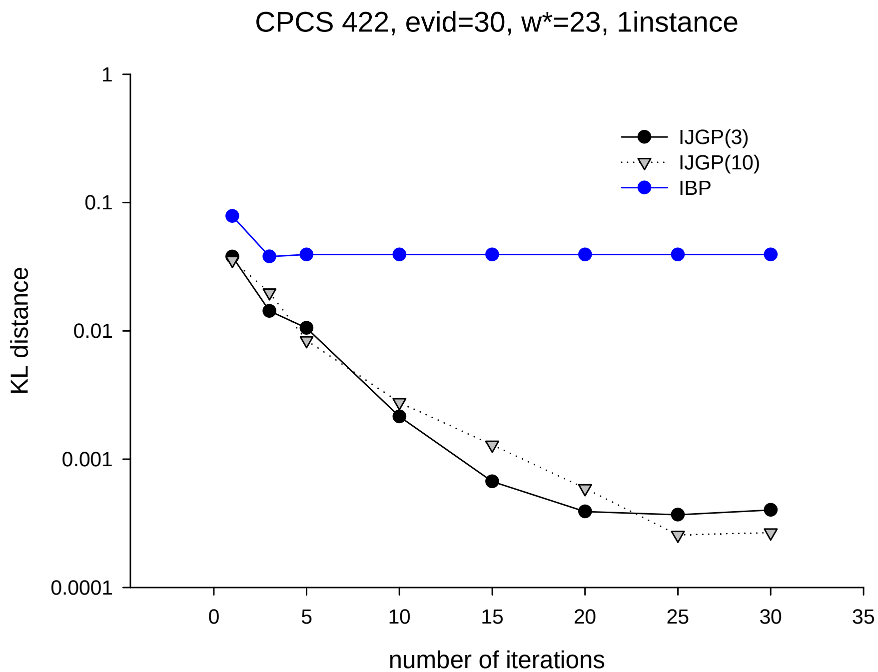
evidence=0

evidence=30

CPCS 422 – KL vs. Iterations



evidence=0



evidence=30

More On the Power of Belief Propagation

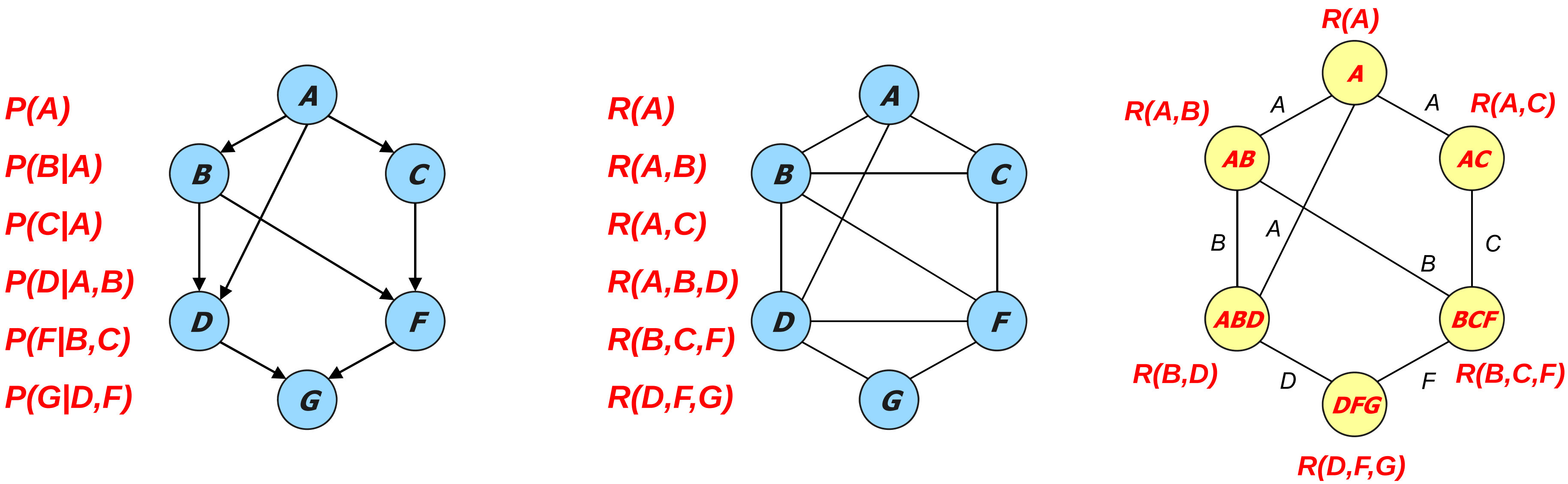
- BP as local minima of KL distance
- BP's power from constraint propagation perspective.

Optimizing the KL-Divergence

- IBP fixed points are stationary points of the KL–divergence: they may only be local minima, or they may not be minima.
- When IBP performs well, it will often have fixed points that are indeed minima of the KL–divergence.
- For problems where IBP does not behave as well, we will next seek approximations $\Pr'$ whose factorizations are more expressive than that of the polytree-based factorization.

Therse results also extend to generalizzed BP

Belief and constraint networks



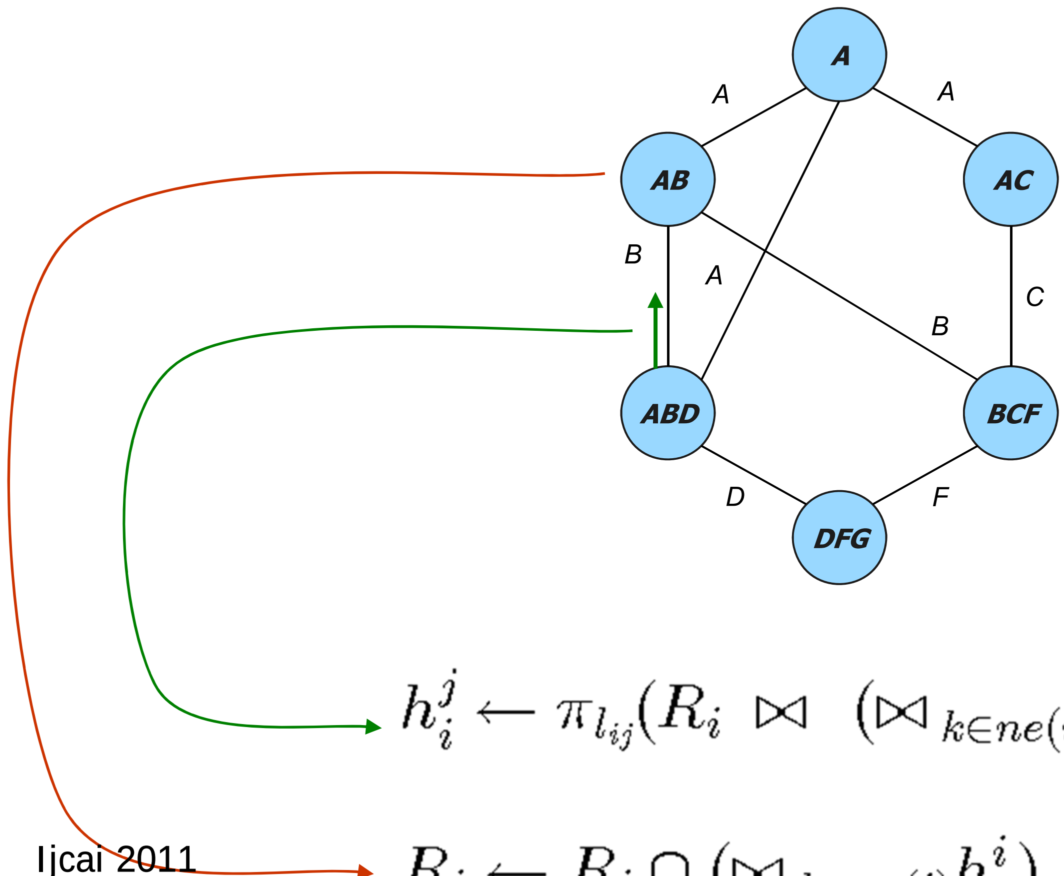
a) Belief network

b) Constraint network

c) Singleton dual join-graph

Distributed Relational Arc-Consistency

- Can be applied to the dual problem of any constraint network:



$$h_i^j \leftarrow \pi_{l_{ij}}(R_i \bowtie (\bowtie_{k \in ne(i)} h_k^i)) \qquad (1)$$

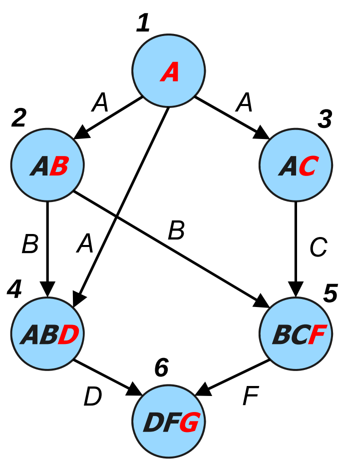
$$R_i \leftarrow R_i \cap (\bowtie_{k \in ne(i)} h_k^i) \qquad (2)$$

Flattening the Bayesian network

A	P(A)
1	.2
2	.5
3	.3
…	0

A	B	P(B|A)
1	2	.3
1	3	.7
2	1	.4
2	3	.6
3	1	.1
3	2	.9
…	…	0

A	C	P(C|A)
1	2	1
3	2	1
…	…	0



A	B	D	P(D|A,B)
1	2	3	1
1	3	2	1
2	1	3	1
2	3	1	1
3	1	2	1
3	2	1	1
…	…	…	0

B	C	F	P(F|B,C)
1	2	3	1
3	2	1	1
…	…	…	0

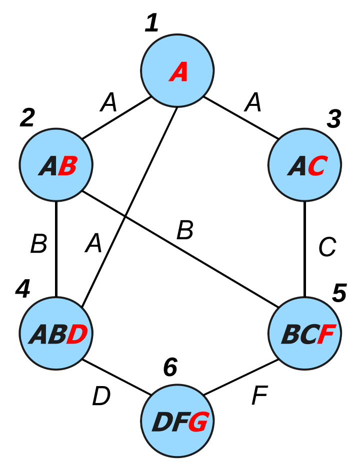
D	F	G	P(G|D,F)
1	2	3	1
2	1	3	1
…	…	…	0

Belief network

A
1
2
3

A	B
1	2
1	3
2	1
2	3
3	1
3	2

A	C
1	2
3	2



A	B	D
1	2	3
1	3	2
2	1	3
2	3	1
3	1	2
3	2	1

B	C	F
1	2	3
3	2	1

D	F	G
1	2	3
2	1	3

Flat constraint network

Belief zero Propagation equals Arc-Consistency

$$h_i^j = \sum_{elim(i,j)} (p_i \cdot (\prod_{\{k \in ne_j(i)\}} h_k^i))$$

$$h_i^j = \pi_{l_{ij}} (R_i \bowtie (\bowtie_{k \in ne(i)} h_k^i))$$

A	B	P(B|A)
1	2	>0
1	3	>0
2	1	>0
2	3	>0
3	1	>0
3	2	>0
…	…	0

h_1^2

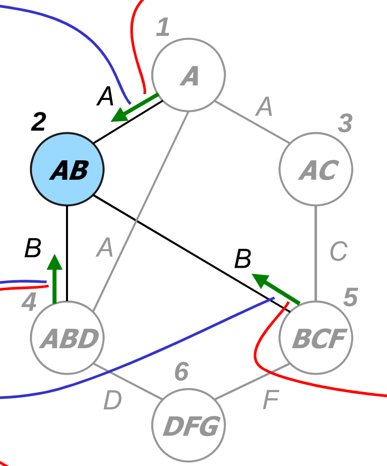
A	$h_1^2(A)$
1	>0
2	>0
3	>0
…	0

h_4^2

B	$h_1^2(B)$
1	>0
2	>0
3	>0
…	0

h_5^2

B	$h_1^2(B)$
1	>0
3	>0
…	0



A	B
1	2
1	3
2	1
2	3
3	1
3	2

h_1^2

A
1
2
3

h_4^2

B
1
2
3

h_5^2

B
1
3

Updated belief:

$$Bel(A,B) = P(B \mid A) \cdot h_1^2 \cdot h_4^2 \cdot h_5^2 =$$

$=$

A	B	*Bel* (A,B)
1	3	>0
2	1	>0
2	3	>0
3	1	>0
…	…	0

Updated relation:

$$R(A,B) = R(A,B) \bowtie h_1^2 \bowtie h_4^2 \bowtie h_5^2 =$$

$=$

A	B
1	3
2	1
2	3
3	1

IBP – inference power for zero beliefs

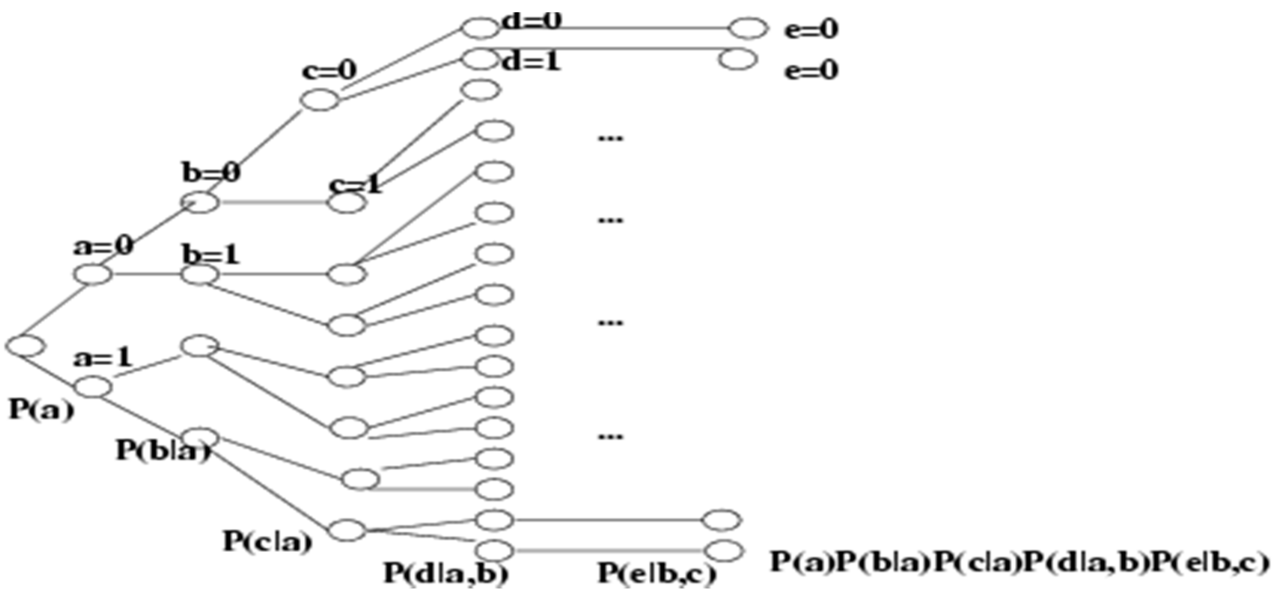
- **Main theorem: IBP's trace for zero beliefs is identical to arc-consistency on the flat network**
- **Consequently,**
  1. **Soundness:** The inference of zero beliefs by IBP converges in a finite number of iterations, and **all zero beliefs inferred are correct**
  2. **Incompleteness:** IBP may 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

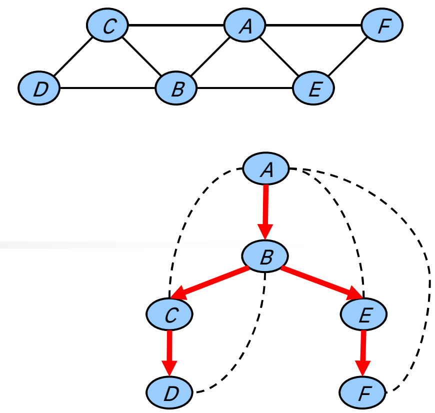
Conditioning generates the probability tree

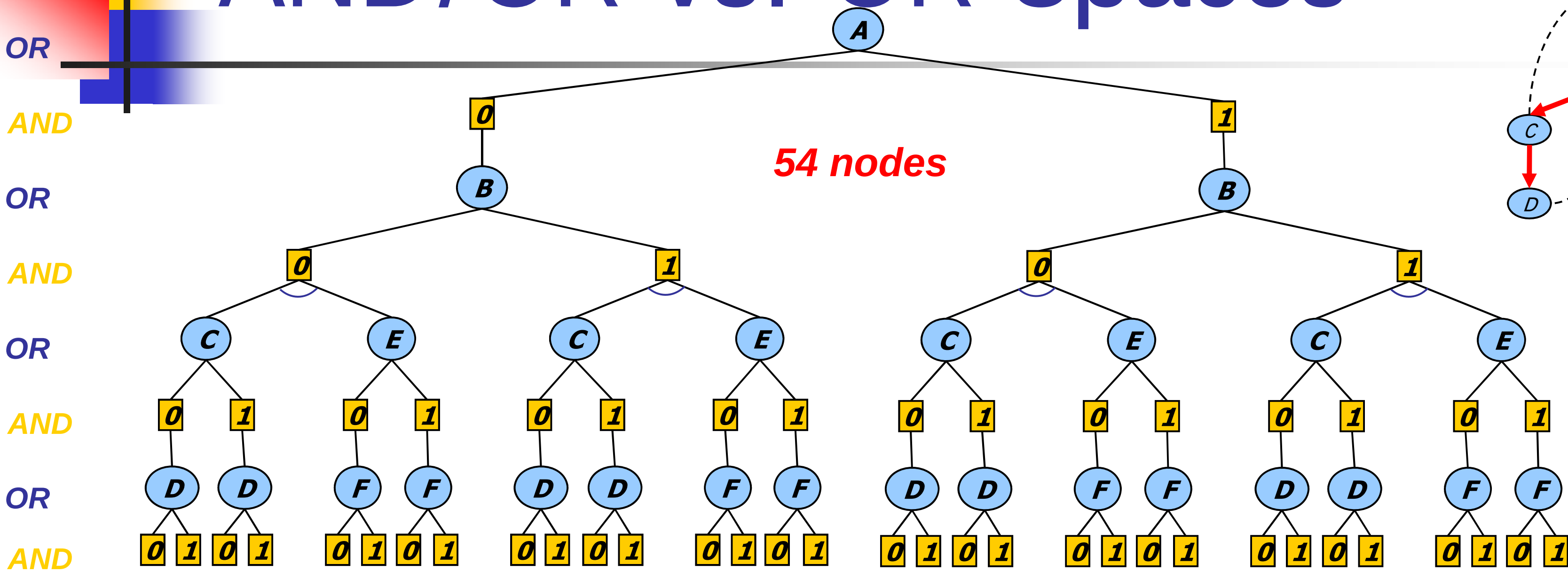
$$P(a, e=0) = P(a)\sum_b P(b \mid a)\sum_c P(c \mid a)\sum_b P(d \mid a, b)\sum_{e=0} P(e \mid b, c)$$

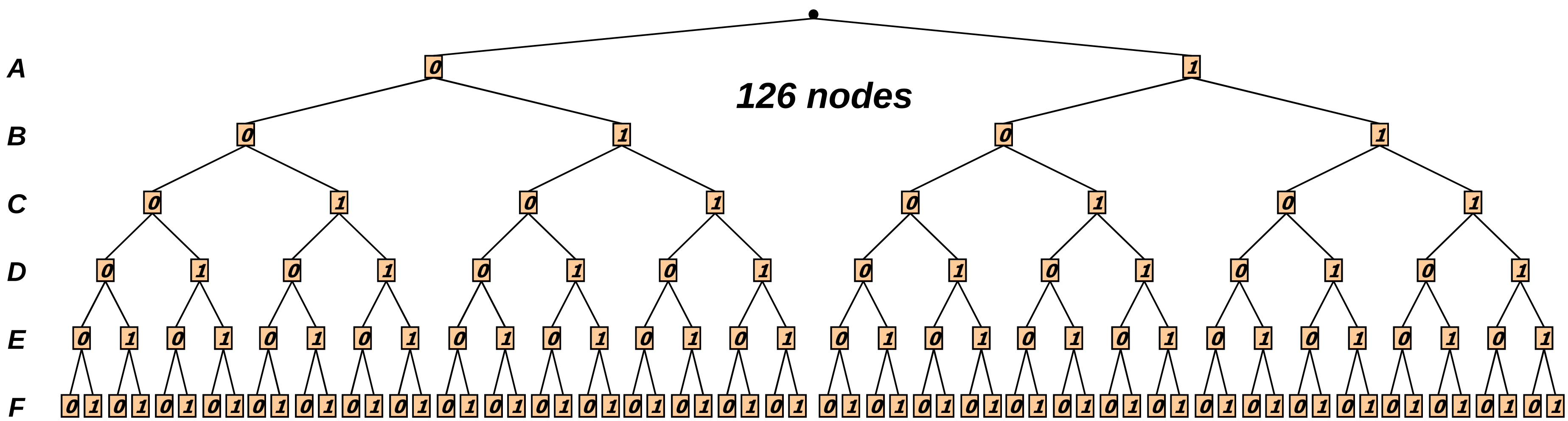


Complexity of conditioning: exponential time, linear space

AND/OR vs. OR Spaces







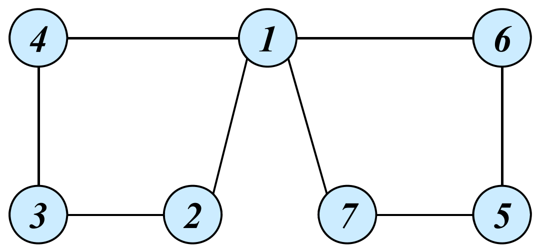
Complexity of AND/OR Tree Search

	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

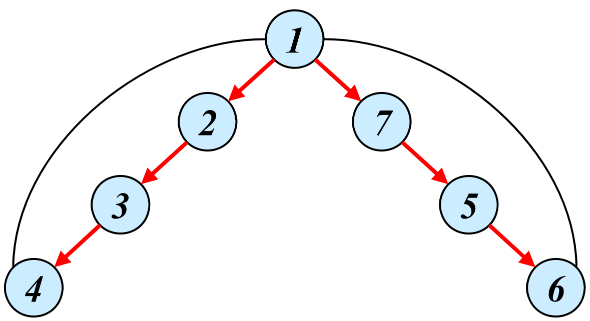
Pseudo-Trees

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

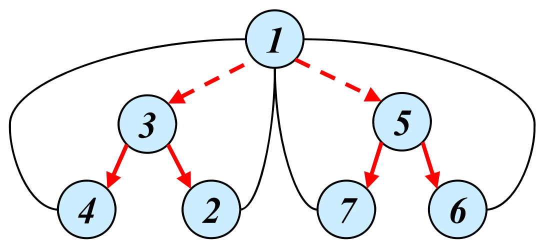


(a) Graph

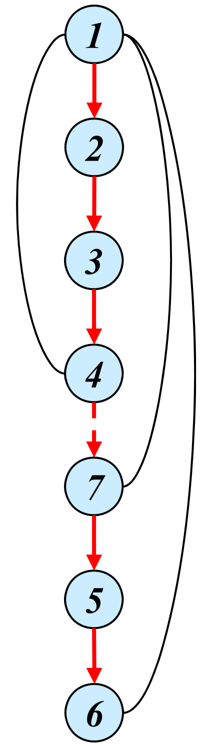
$t <= w^* \log n$



(b) DFS tree
depth=3



(c) pseudo- tree
depth=2



(d) Chain
depth=6

Weighted AND/OR Tree

$P(E \mid A,B)$

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

$P(B \mid A)$

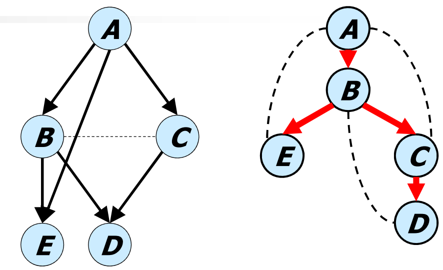
A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C \mid A)$

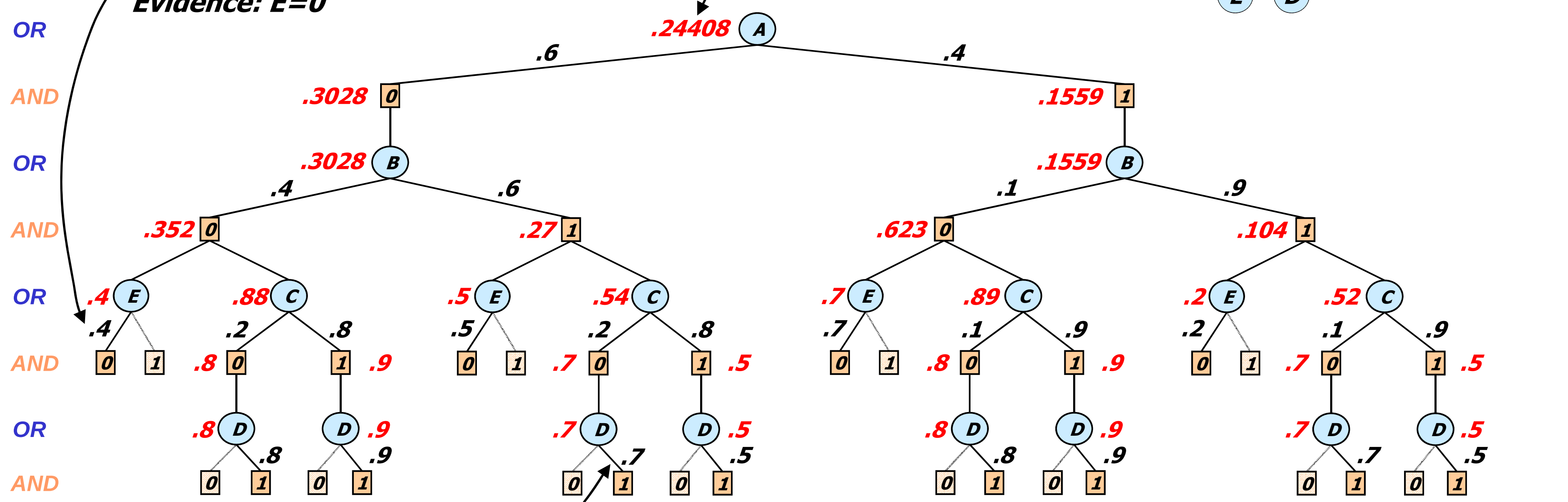
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

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

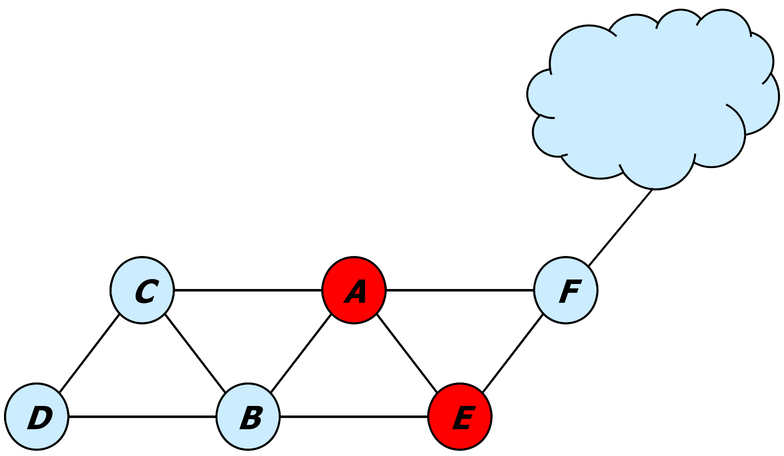
Evidence: D=1

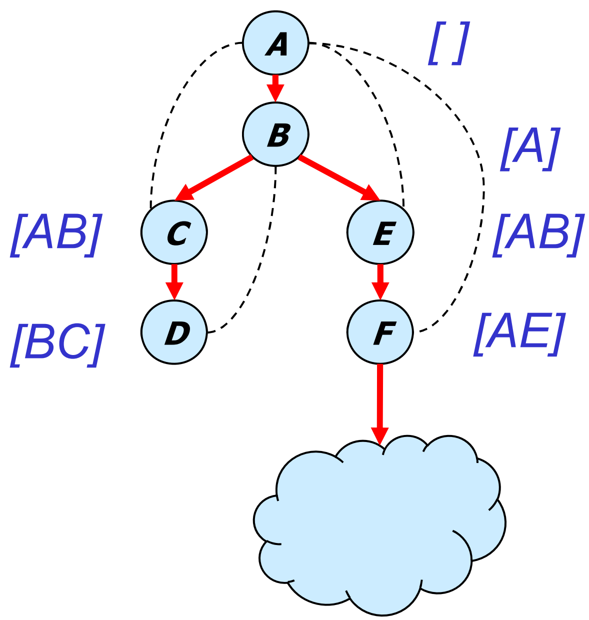
OR node: Marginalization operator (summation)

AND node: Combination operator (product)

Value of node = updated belief for sub-problem below

Merging based on context



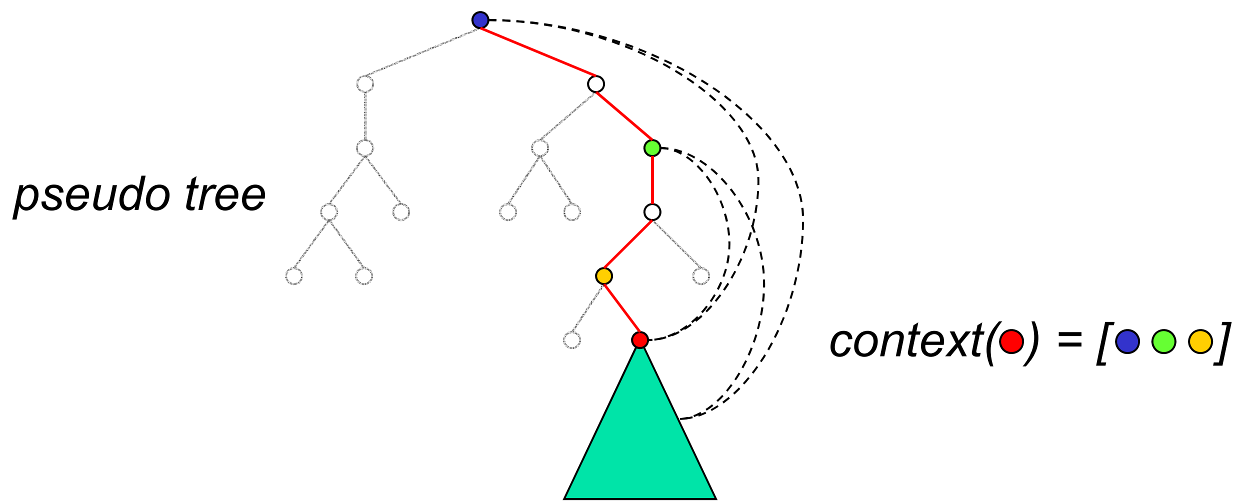


How big is the context?

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



AND/OR Tree DFS Algorithm

(Belief Updating)

$P(E \mid A,B)$

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

$P(B \mid A)$

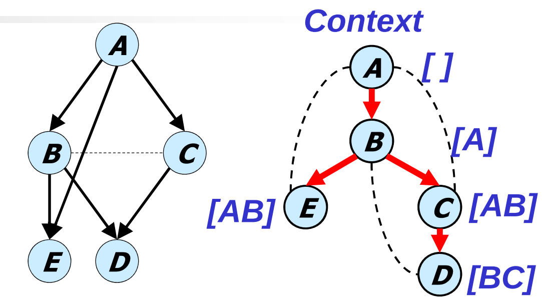
A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C \mid A)$

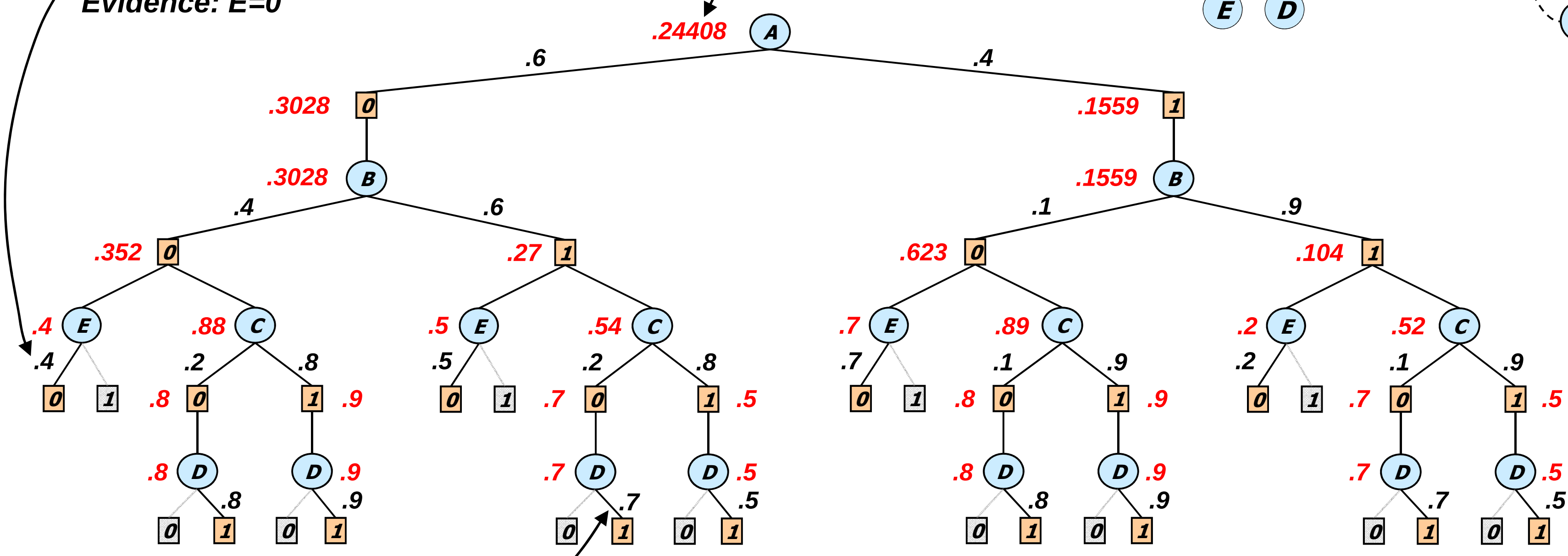
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

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 Graph DFS Algorithm

(Belief Updating)

$P(E \mid A,B)$

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

$P(B \mid A)$

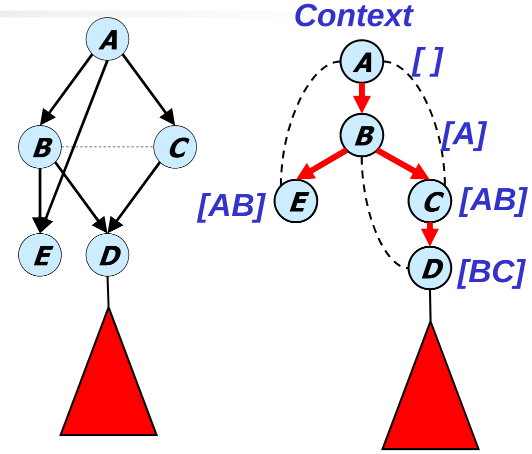
A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C \mid A)$

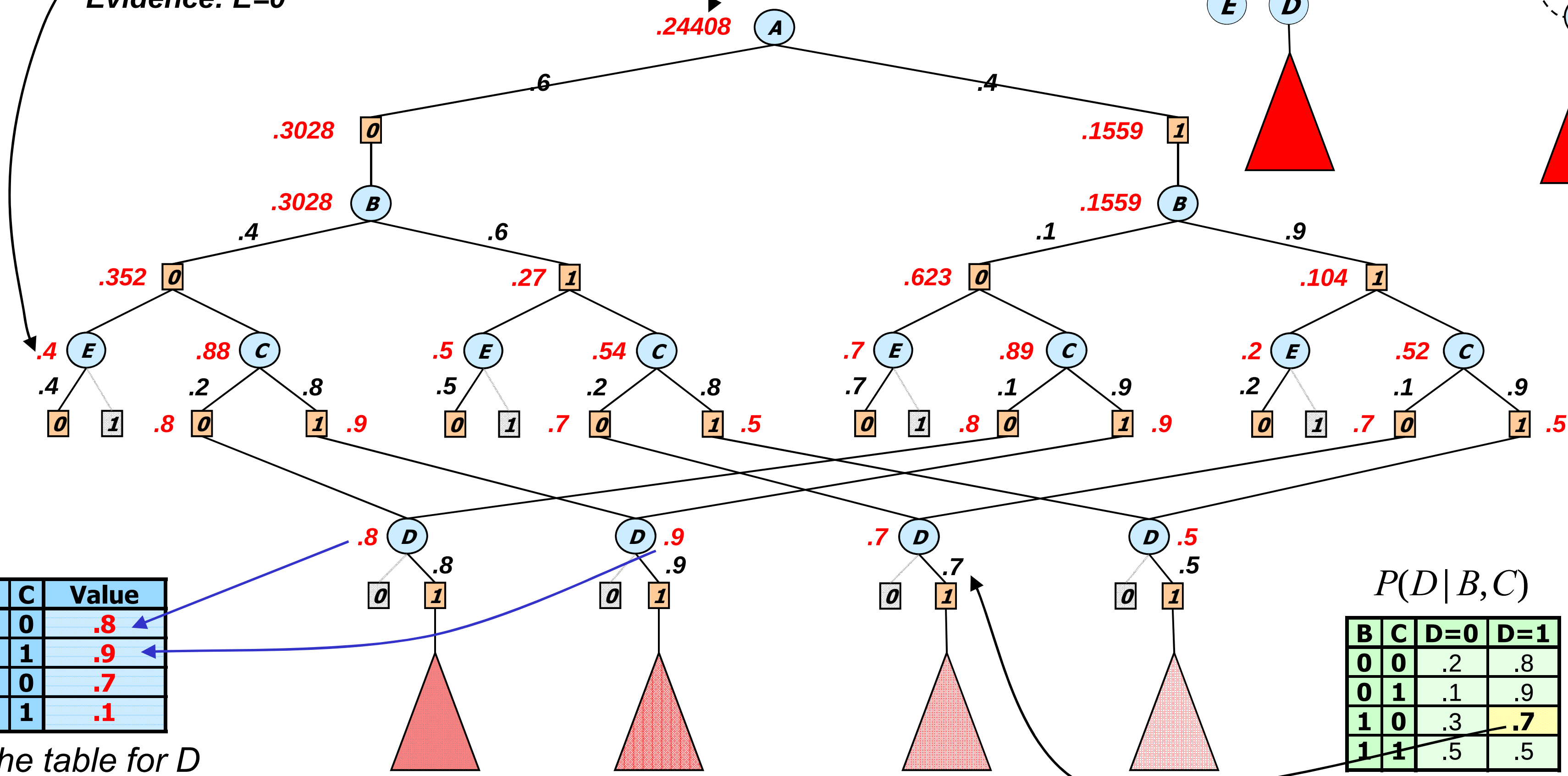
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4



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



B	C	Value
0	0	.8
0	1	.9
1	0	.7
1	1	.1

Cache table for D

$P(D \mid B,C)$

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

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*

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

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

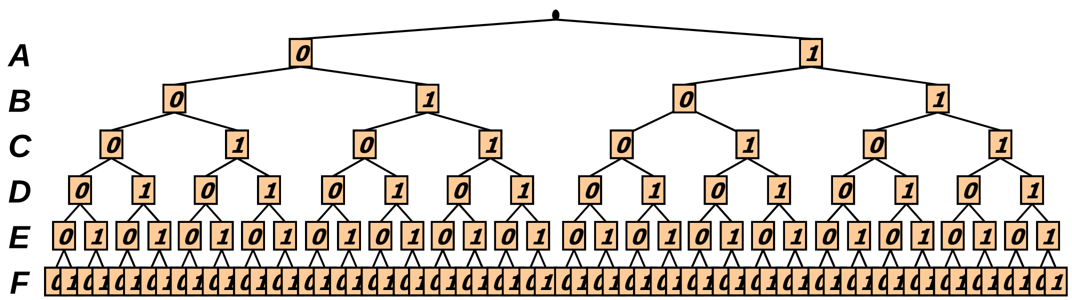
Network	hypergraph		min-fill	
	width	depth	width	depth
barley	7	**13**	7	23
diabetes	7	**16**	4	77
link	21	**40**	15	53
mildew	5	**9**	4	13
munin1	12	**17**	12	29
munin2	9	**16**	9	32
munin3	9	**15**	9	30
munin4	9	**18**	9	30
water	11	**16**	10	15
pigs	11	**20**	11	26

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

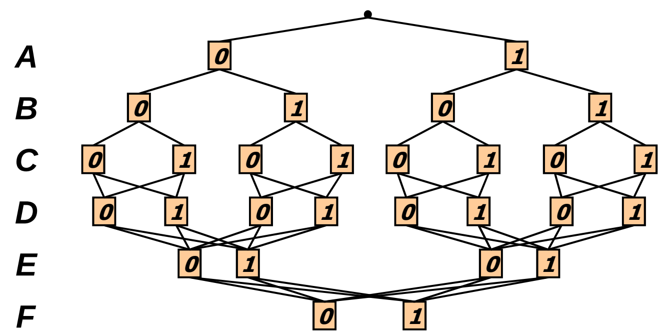
SPOT5 Benchmarks

All Four Search Spaces



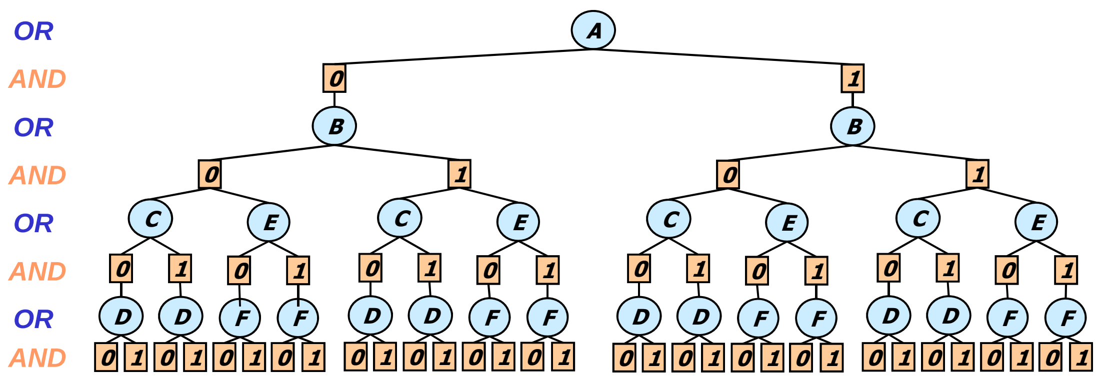
Full OR search tree

126 nodes



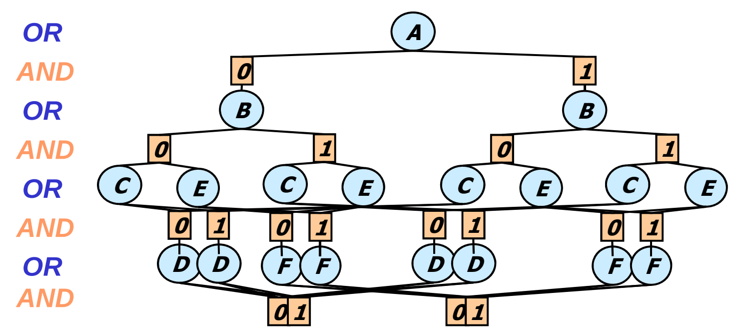
Context minimal OR search graph

28 nodes



Full AND/OR search tree

54 AND nodes

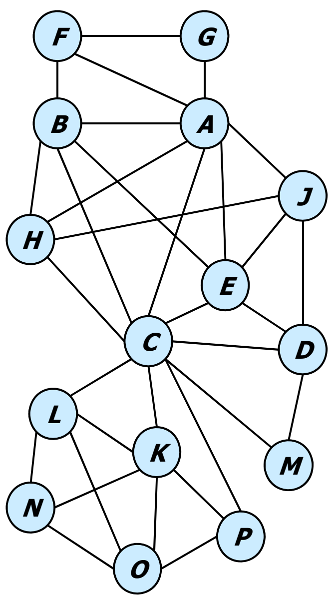


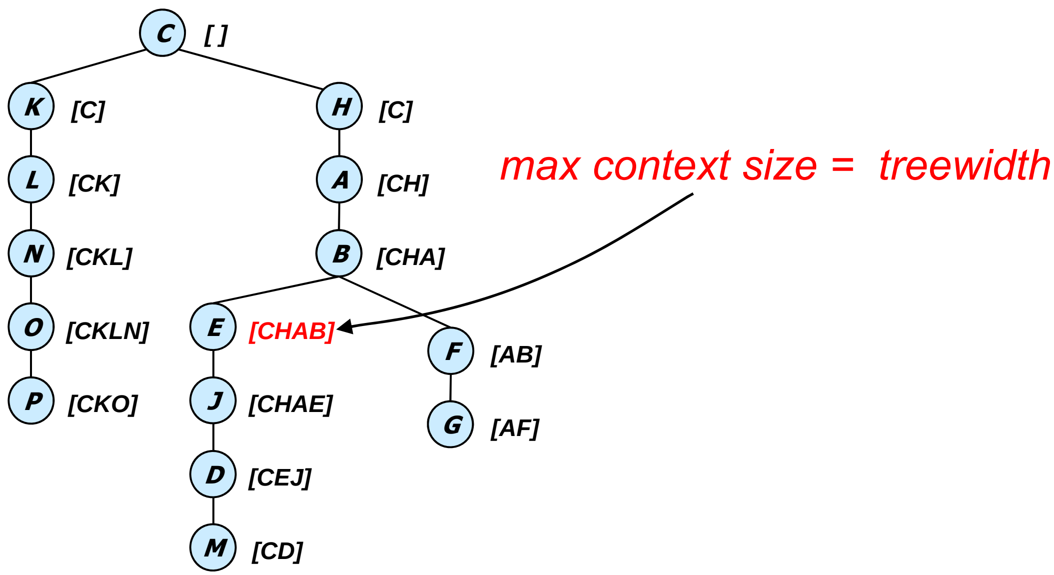
Context minimal AND/OR search graph

18 AND nodes

How Big Is The Context?

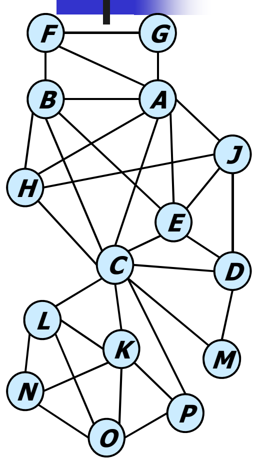
Theorem: The maximum context size for a pseudo tree is equal to the treewidth of the graph along the pseudo tree.

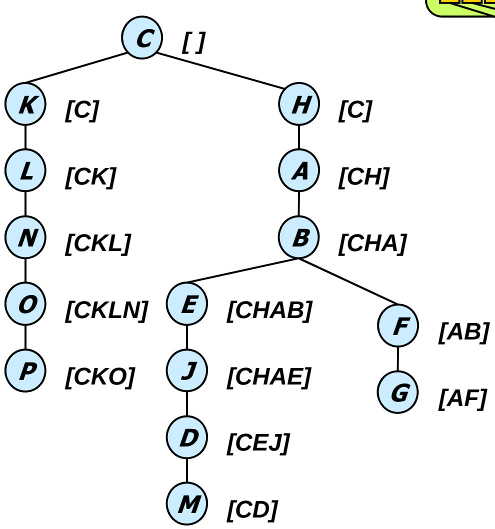


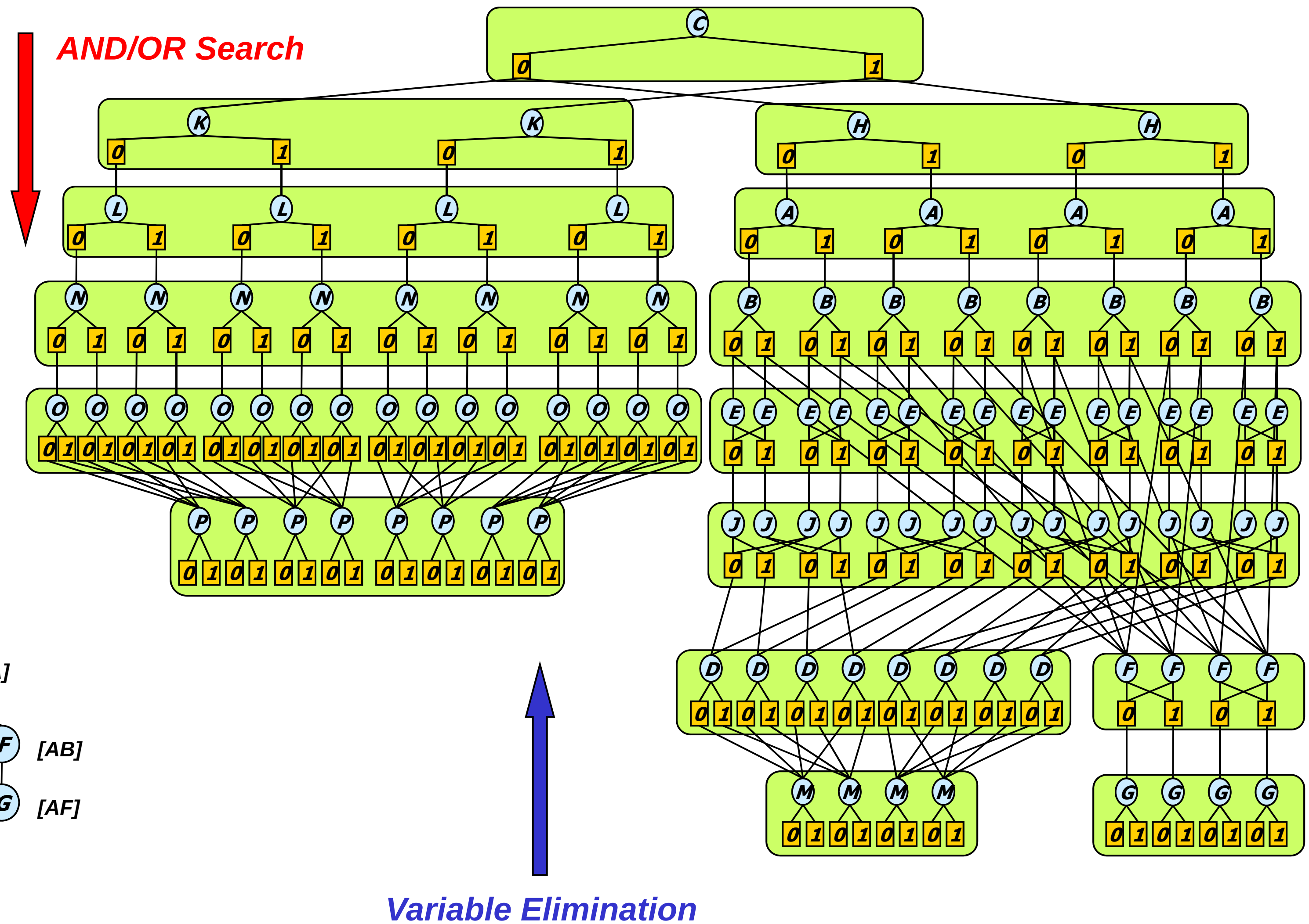


(C K H A B E J L N O D P M F G)

AND/OR Context Minimal Graph



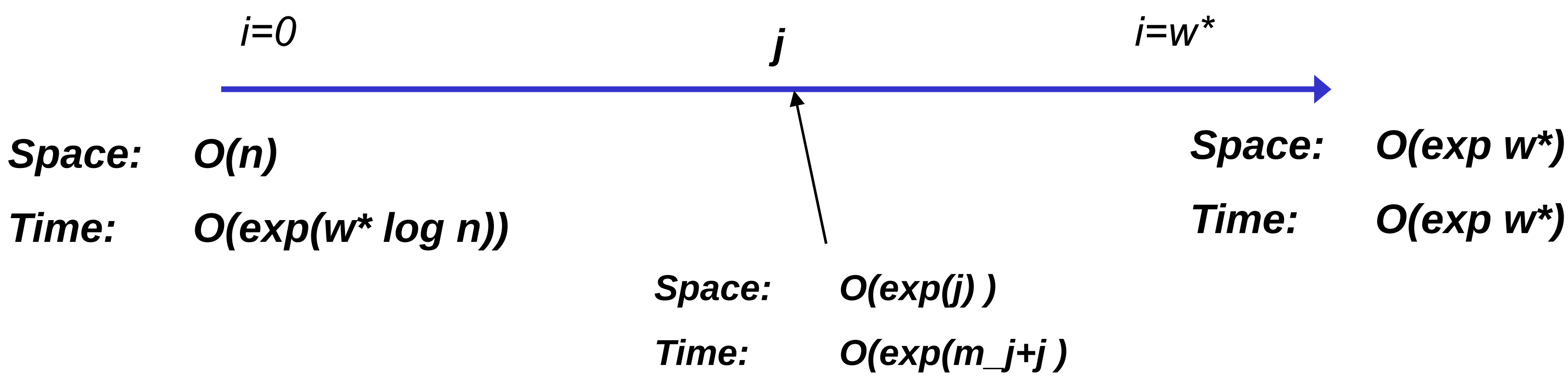




(C K H A B E J L N O P M F G)

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)



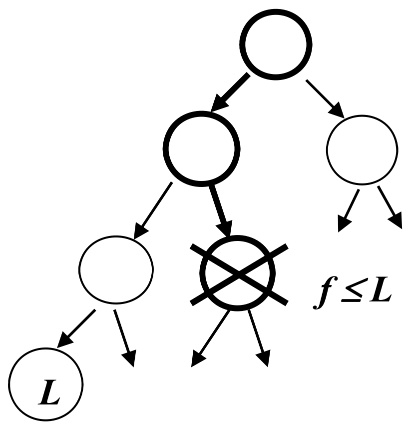
Searching the AND/OR space for MPE/MAP

*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*

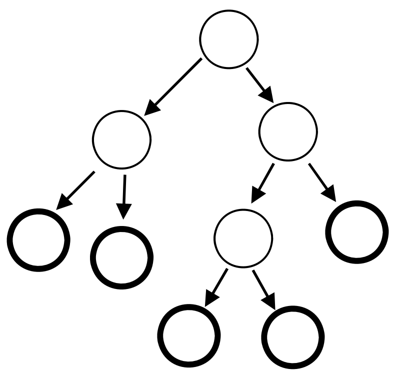
Linear space



2. Best-First Search

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

Needs lots of memory

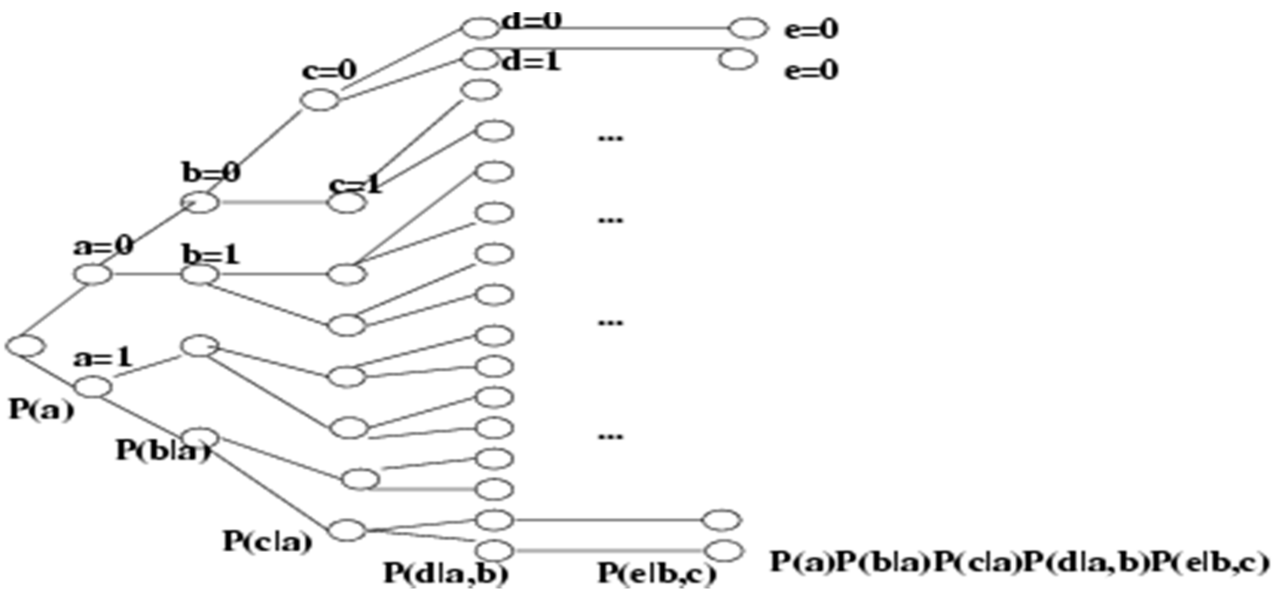


Road Map:Bayesian Networks

- 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
- Reasoning with mixed networks

Conditioning generates the probability tree

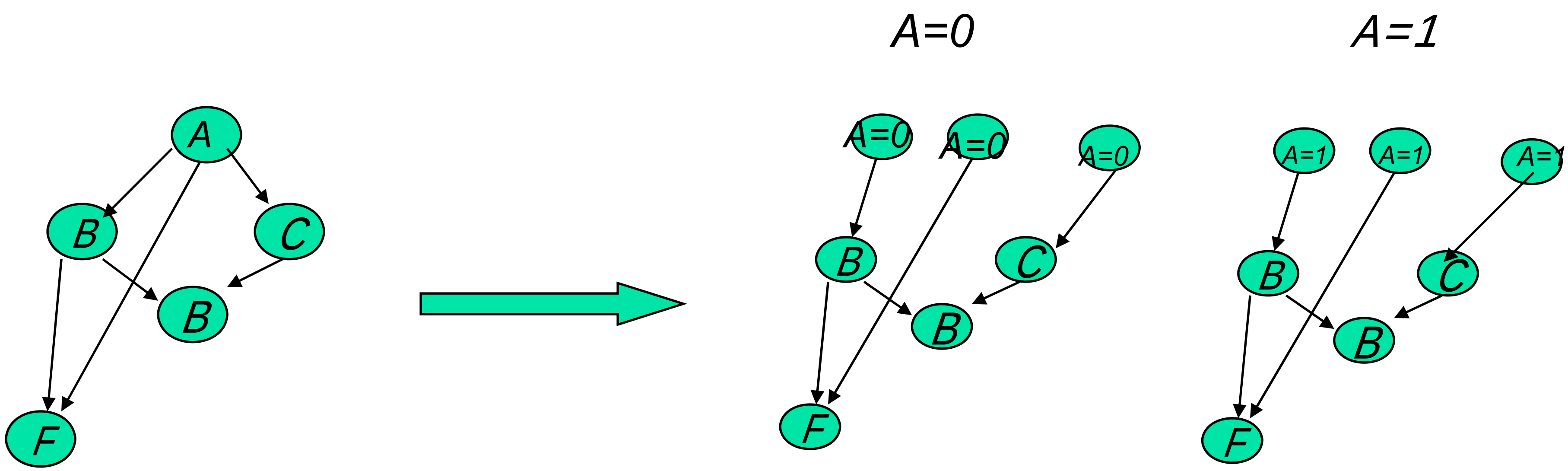
$$P(a,e=0)=P(a)\sum_{b}P(b\mid a)\sum_{c}P(c\mid a)\sum_{b}P(d\mid a,b)\sum_{e=0}P(e\mid b,c)$$



Complexity of conditioning: exponential time, linear space

Loop-cutset decomposition

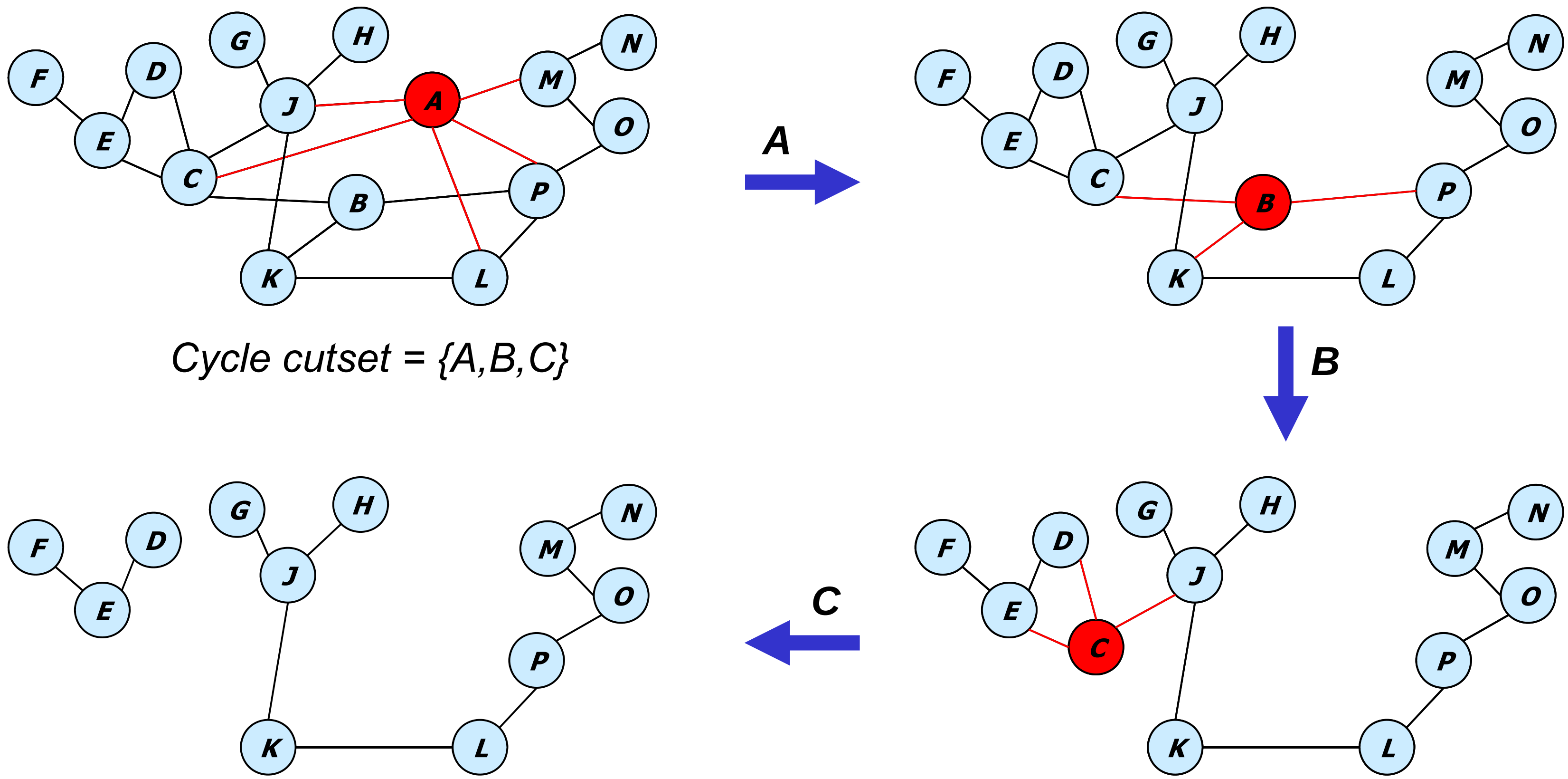
- You condition until you get a polytree



P(B|F=0) = P(B, A=0|F=0)+P(B,A=1|F=0)

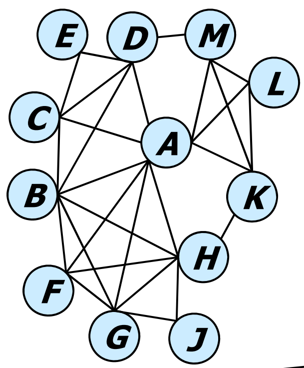
Loop-cutset method is time exp in loop-cutset size and linear space. For each cutset we can do BP

Conditioning and Cycle cutset

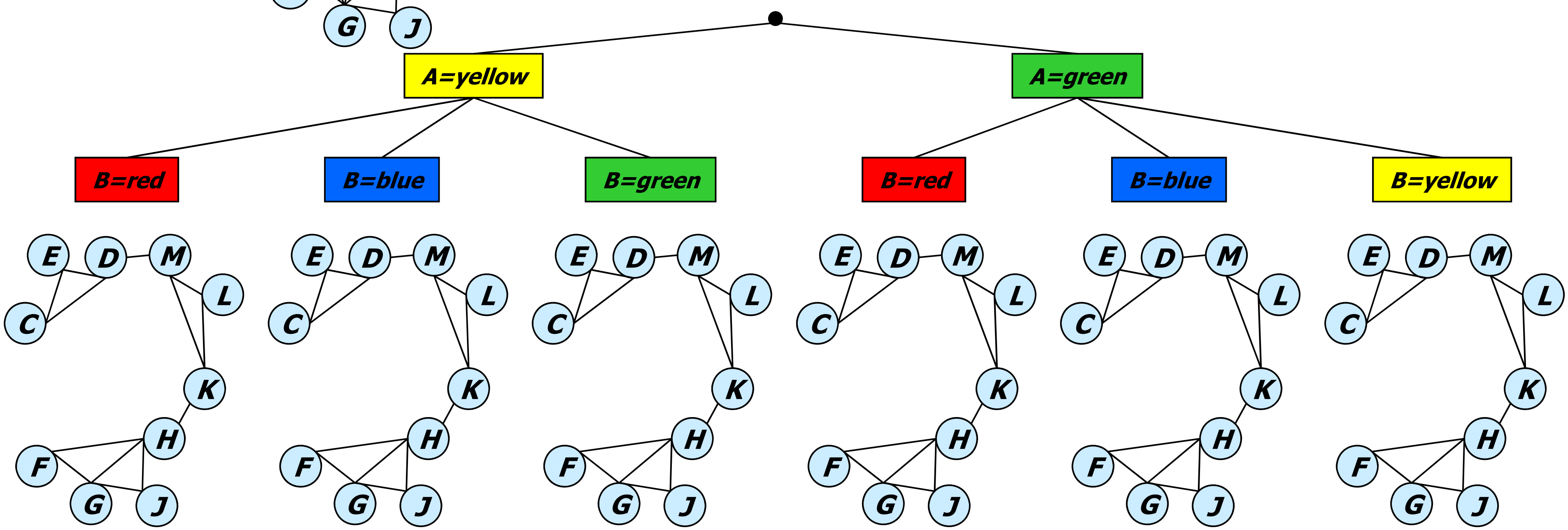


Search over the Cutset (cont)

Graph Coloring problem



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



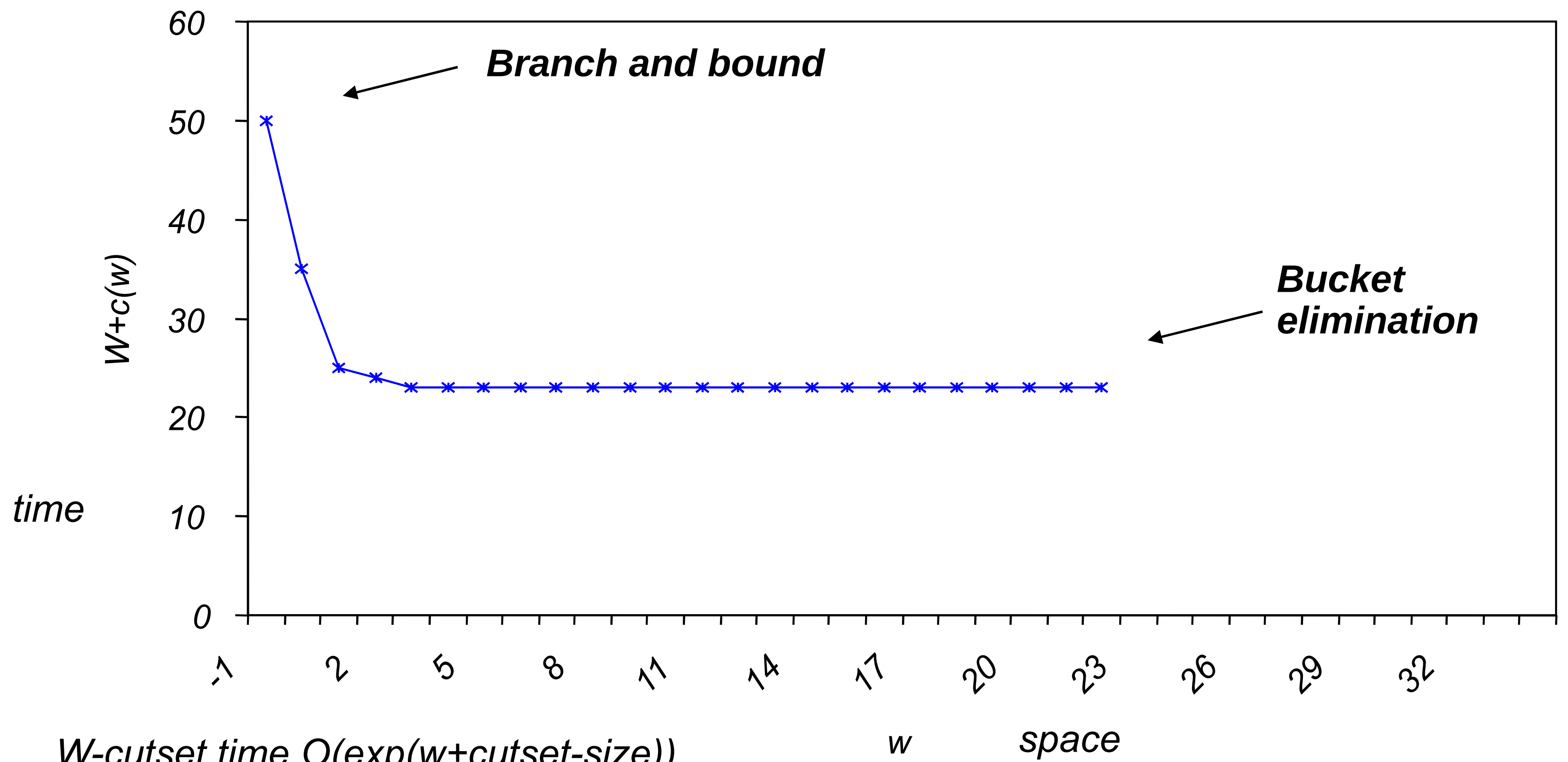
Variable elimination with conditioning; w-cutset algorithms

- 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))

Road Map:Bayesian Networks

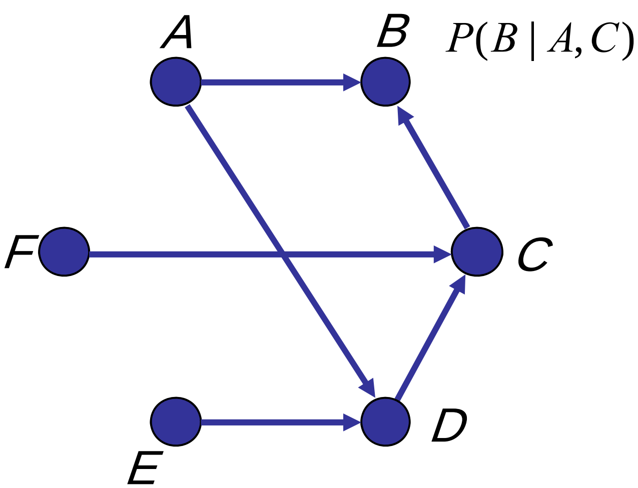
- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
- Search, conditioning
- Hybrid of Search and Inference
- Reasoning with mixed networks
  - Inference
  - Search
  - Sampling solutions

Road Map:Bayesian Networks

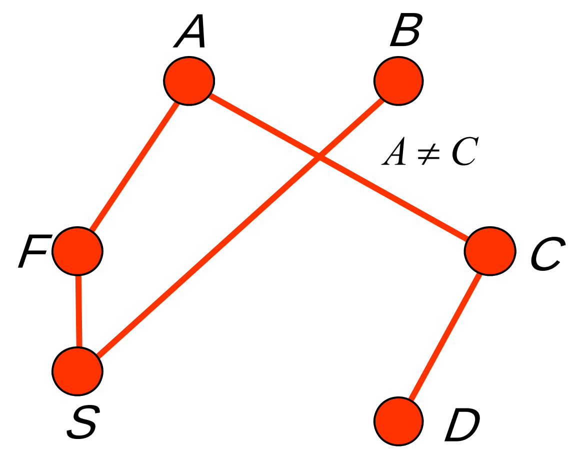
- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
- Search, conditioning
- Hybrid of Search and Inference
- Reasoning with mixed networks
  - Inference
  - Search
  - Sampling solutions

Mixed Networks

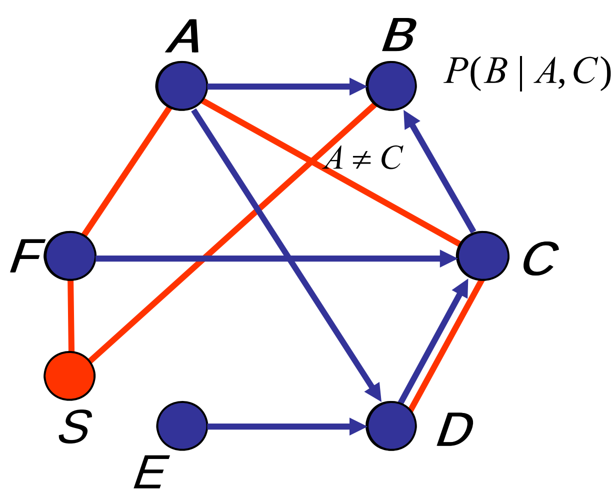
Bayesian Network



Constraint Network



Mix: Combine? Subsume?

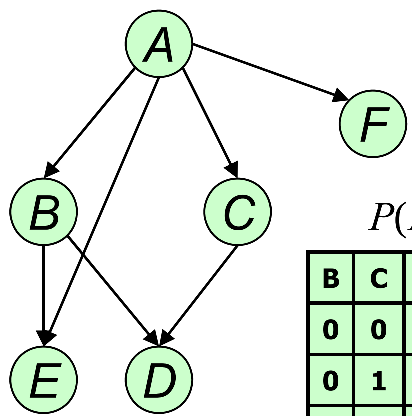


Semantics?

Algorithms?

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

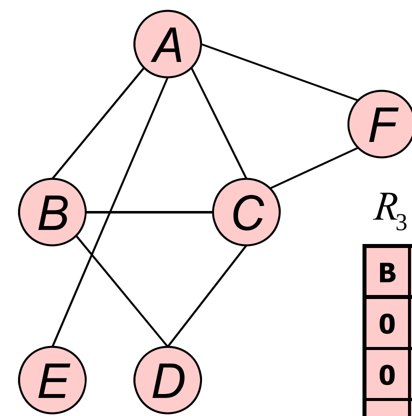
Variables: A, B, C, D, E, F

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

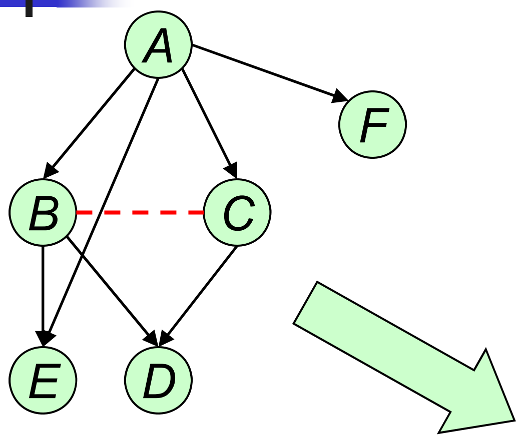
Variables: A, B, C, D, E, F

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

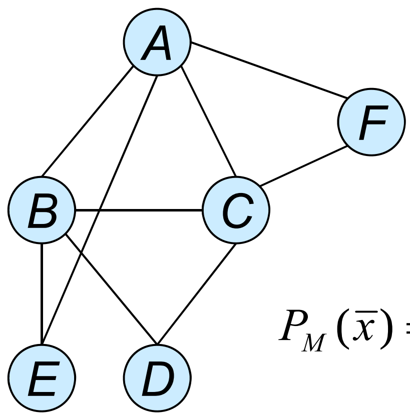
Relations: $R_1(ABC), R_2(ACF), R_3(BCD), R_4(A, E)$

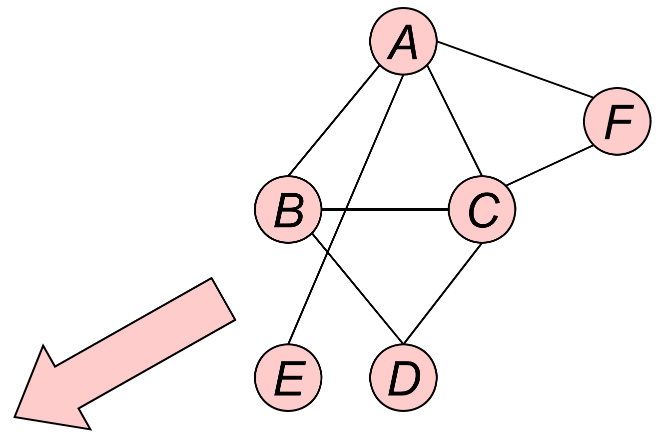
Expresses the set of solutions: $\rho = R(ABCDEF)$

Mixed Networks



Moral mixed graph





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

Theorem: *The mixed graph is a minimal I-map (independency map) relative to dm-separation.*

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,

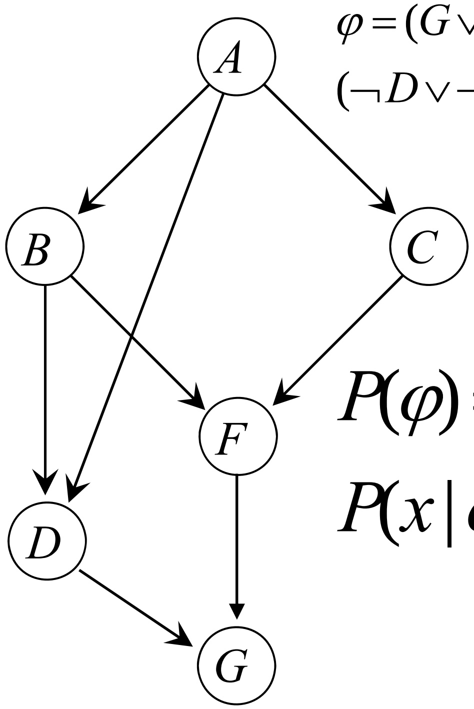
- All these can be extended to influence diagrams, and
  - We can add constraints between control variables
  - The relevant probability distribution is conditional of R

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

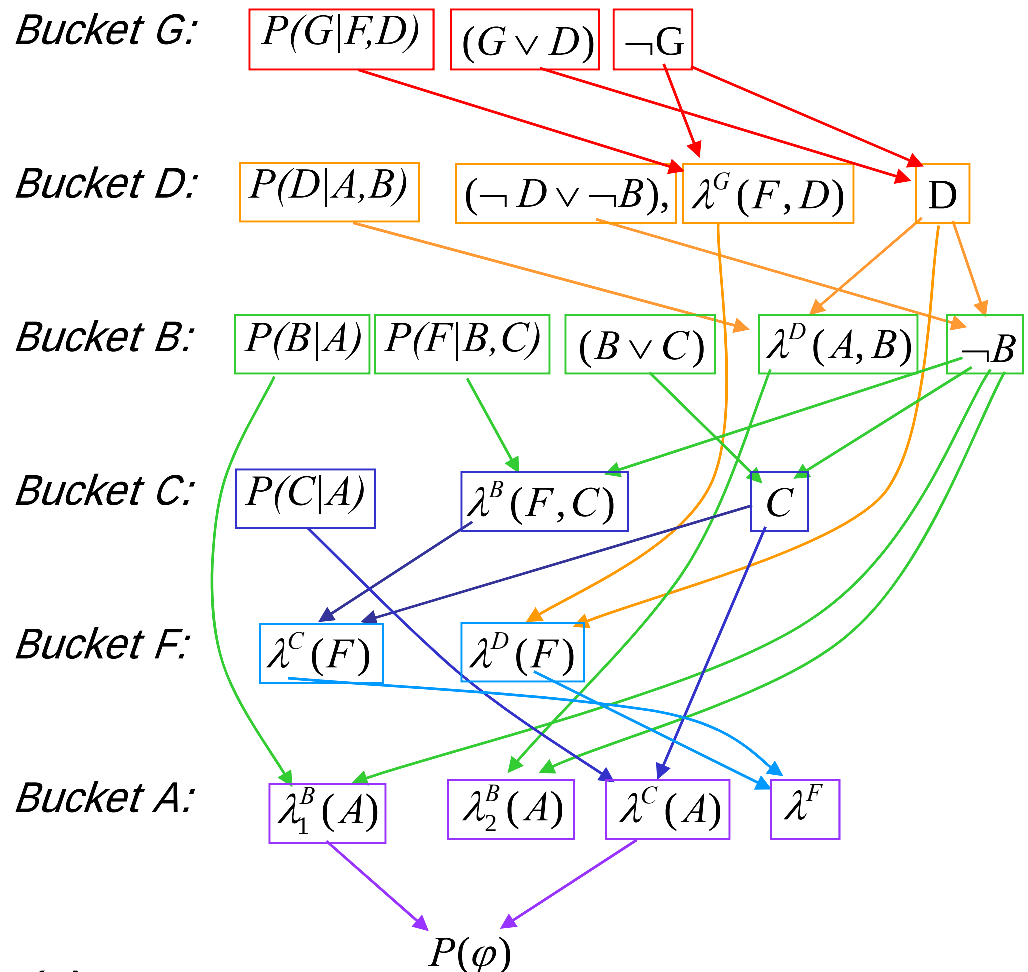
Computing Probability of a cnf query



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

$$P(\varphi) = ?$$

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



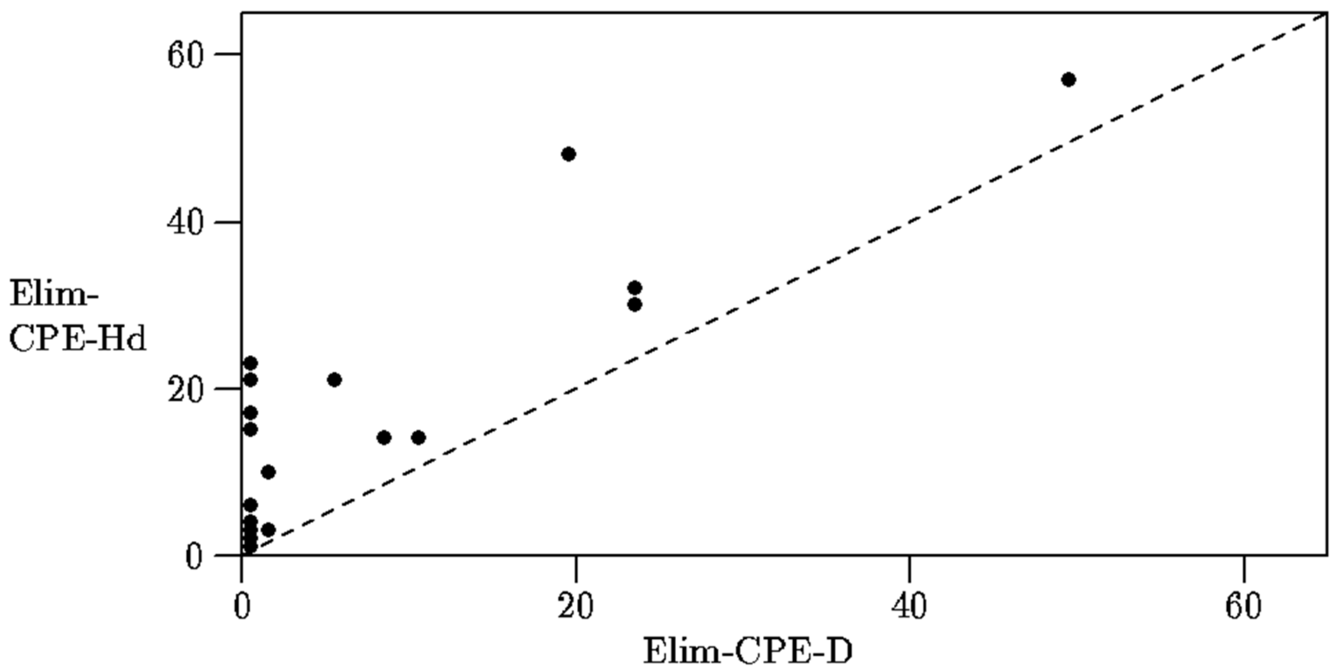
Belief network P(g,f,d,c,b,a)
=P(g|f,d)P(f|c,b)P(d|b,a)P(b|a)P(c|a)P(a)

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.

Elim-CPE-D on Insurance network
(Dechter and Larkin, UAI2001)

19 instances with Insurance network. 20 relations, arity 3, tightness 25 %, 5 evidence nodes.

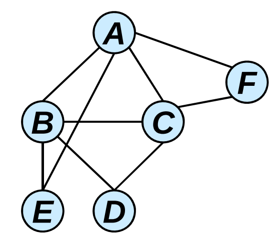


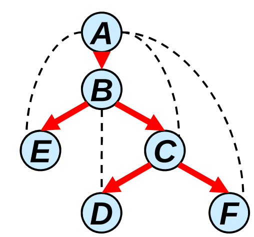
Road Map:Bayesian Networks

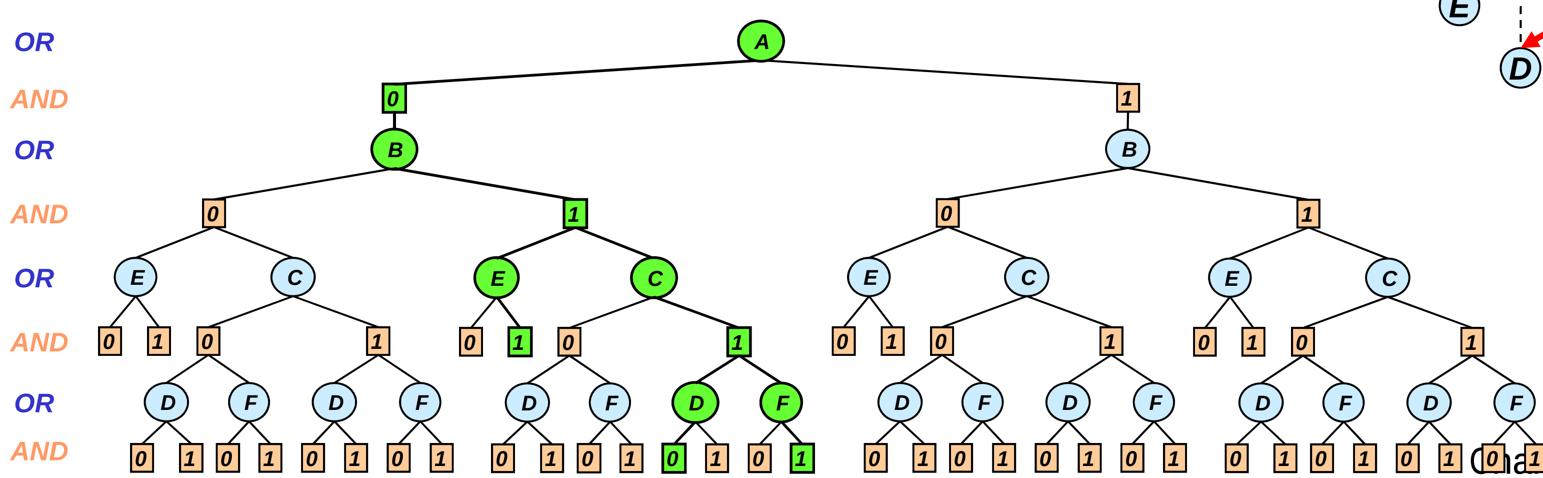
- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
- Search, conditioning
- Hybrid of Search and Inference
- Reasoning with mixed networks
  - Inference
  - Search
  - Sampling solutions

AND/OR search tree

- The AND/OR search tree of a constraint network R relative to a pseudo-tree, T, has alternating levels of: AND nodes (variables) and OR nodes (values)
- The root is the root of T (OR node)
- Successor function:
  - The successors of an OR node X are all its consistent values along its path
  - The successors of an AND node <X,v> are all the children of X in T
  - AND nodes have labels
- A solution is a subtree







AND/OR search tree properties

- Theorem: Any AND/OR search tree based on a pseudo-tree is sound and complete (expresses all and only solutions)

- Theorem:

  Size of AND/OR search tree is $O(n\ k^m)$

  Size of OR search tree is $O(k^n)$

k = *domain size*
m = *pseudo-tree depth*
n = *number of variables*

- Theorem: A constraint network that has a tree-width w* has an AND/OR search tree whose size is bounded by O(exp(w* log n))
(similar to RC, Darwiche 01; Bacchus et.al 03; Freuder 85; Bayardo 95)

- Result: AND/OR search tree algorithms are
  - Space: O(n)
  - Time: O(exp(w* log n))

AND/OR tree DFS algorithm (belief updating)

$P(E|A,B)$

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

$P(B|A)$

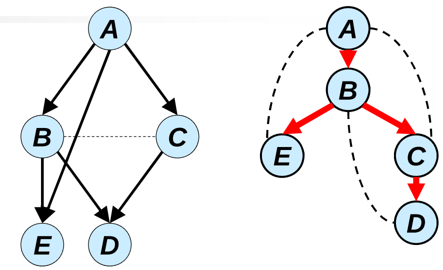
A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C|A)$

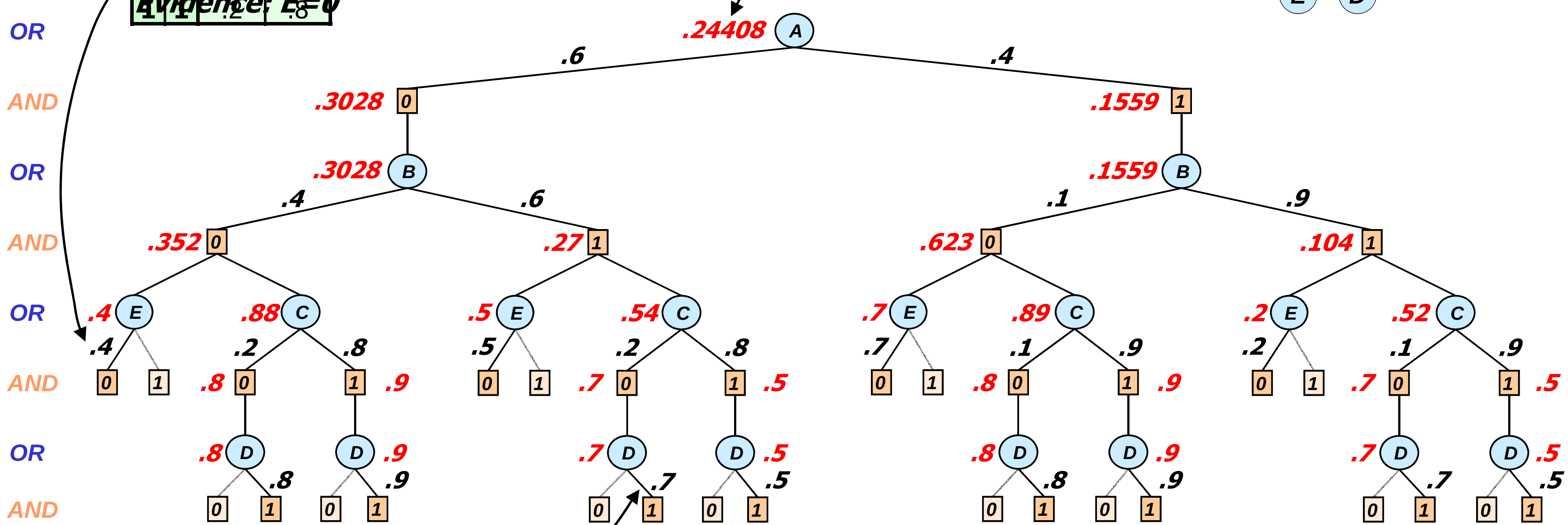
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4



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



$P(D|B,C)$

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

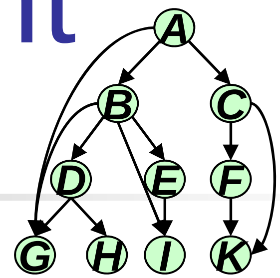
OR node: Marginalization operator (summation)

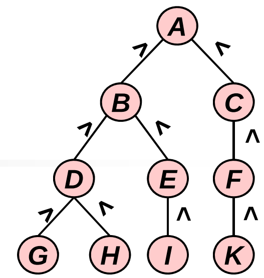
AND node: Combination operator (product)

Value of node = updated belief for subproblem below

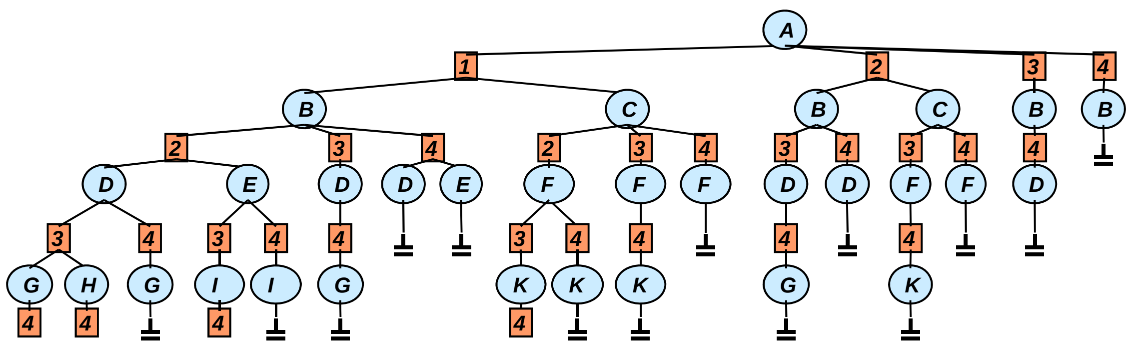
Changes'05

The Effect of Constraint Propagation

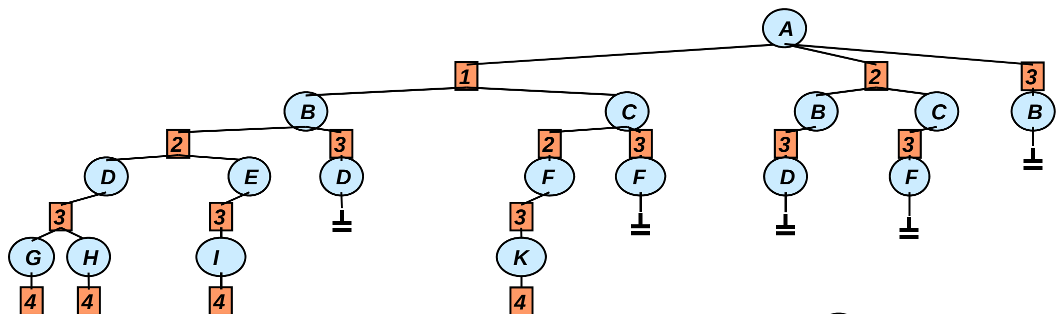




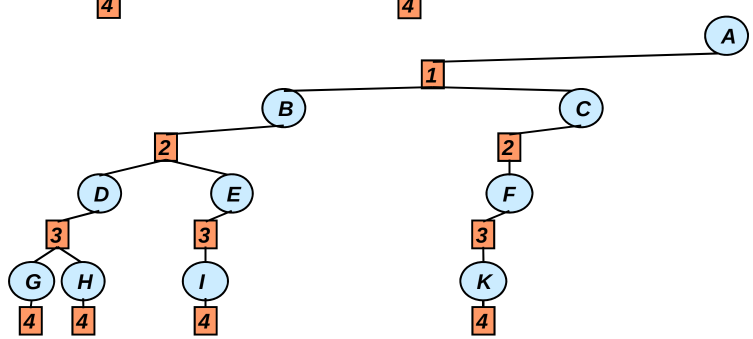
Domains are {1,2,3,4}



CONSTRAINTS ONLY



FORWARD CHECKING



MAINTAINING ARC CONSISTENCY

Experimental results

- **Parameters of the mixed networks:**
  - N = number of variables
  - K = number of values per variable

- *Belief network - (N,K,R,P):*
  - *R = number of root nodes*
  - *P = number of parents*
- *Measures:*
  - *Time*
  - *Number of nodes*
  - *Number of dead-ends*

- *Constraint network - (N,K,C,S,t):*
  - *C = number of constraints*
  - *S = scope size of the constraints*
  - *t = tightness (no. of allowed tuples)*
- *Algorithms:*
  - *AO-C constraint checking only*
  - *AO-FC forward checking*
  - *AO-RFC relational forward checking*

OR vs. AND/OR Spaces

N=25, K=2, R=2, P=2, C=10, S=3, t=70%, 20 instances, w*=9, h=14				
	Time	Nodes	Dead-ends	Full space
AO-C	0.15	44,895	9,095	152,858
OR-C	11.81	3,147,577	266,215	67,108,862

AND/OR Search Algorithms (1)

N=40, K=2, R=2, P=2, C=10, S=4, 20 instances, w*=12, h=19											
t	**i**	**Time**			**Nodes**			**Dead-ends**			**#sol**
		AO-C	**AO-FC**	**AO-RFC**	**AO-C**	**AO-FC**	**AO-RFC**	**AO-C**	**AO-FC**	**AO-RFC**	
20%	0	0.671	0.056	0.022	153,073	4,388	1,066	95,197	3,299	962	1.6E+05
	3	0.619	0.053	0.019	101,268	3,476	950	95,197	3,299	962	
	6	0.479	0.055	0.022	75,397	3,213	936	57,306	3,168	940	
	9	0.297	0.053	0.019	51,746	3,152	926	10,414	2,933	751	
	12	0.103	0.044	**0.016**	16,579	2,273	683	2,638	1,537	398	
40%	0	2.877	0.791	1.094	774,697	167,921	158,007	239,991	40,069	36,119	7.7E+07
	3	2.426	0.663	0.894	329,686	57,023	52,197	239,991	40,069	36,119	
	6	1.409	0.445	0.544	183,286	35,325	31,607	107,362	27,575	24,153	
	9	0.739	0.301	0.338	119,125	23,655	20,832	19,635	11,929	10,144	
	12	0.189	**0.142**	0.149	27,848	9,148	7,357	3,343	3,997	3,048	
60%	0	6.827	4.717	7.427	1,974,952	1,158,544	1,148,044	362,279	162,781	158,968	6.2E+09
	3	5.560	3.908	6.018	673,117	351,022	345,763	362,279	162,781	158,968	
	6	2.809	2.219	3.149	346,842	183,895	180,463	150,864	88,822	85,522	
	9	1.334	1.196	1.535	204,414	105,527	102,270	18,961	24,571	22,830	
	12	**0.255**	**0.331**	0.425	36,262	23,160	22,293	2,825	5,083	4,658	
80%	0	14.181	14.199	21.791	4,282,678	3,703,920	3,702,692	370,314	278,479	277,250	1.1E+11
	3	11.334	11.797	17.916	1,319,599	1,108,561	1,107,332	370,314	278,479	277,250	
	6	5.305	6.286	9.061	626,405	519,258	518,029	127,683	98,100	96,872	
	9	2.204	2.890	3.725	336,146	274,375	273,147	16,726	20,900	19,671	
	12	**0.318**	0.543	0.714	44,340	39,550	39,524	1,431	2,647	2,659	
100%	0	23.595	27.129	41.744	7,450,537	7,450,537	7,450,537	0	0	0	1.1E+12
	3	19.050	22.842	34.800	2,161,401	2,161,401	2,161,401	0	0	0	
	6	8.325	11.528	16.636	956,965	956,965	956,965	0	0	0	
	9	3.153	4.863	6.255	483,921	483,921	483,921	0	0	0	
	12	**0.366**	0.681	0.884	50,616	50,616	50,616	0	0	0	

AND/OR Search Algorithms (2)

N=100, K=2, R=10, P=2, C=30, S=3, 20 instances, w*=28, h=38								
t	i	Time		Nodes		Dead-ends		#sol
		AO-FC	AO-RFC	AO-FC	AO-RFC	AO-FC	AO-RFC	
10%	0	**1.743**	**1.743**	15,466	15,408	15,468	15,410	0
	10	1.748	1.746	15,466	15,408	15,468	15,410	
	20	1.773	1.784	15,466	15,408	15,468	15,410	
20%	0	**3.193**	3.201	27,840	27,617	27,842	27,619	0
	10	3.195	3.200	27,840	27,617	27,842	27,619	
	20	3.276	3.273	27,840	27,617	27,842	27,619	
30%	0	69.585	62.911	804,527	659,305	804,529	659,307	0
	10	69.803	**62.908**	804,527	659,305	804,529	659,307	
	20	69.275	63.055	804,527	659,305	686,769	659,307	

N=100, K=2, R=5, P=2, C=40, S=3, 20 instances, w*=41, h=51								
t	i	Time		Nodes		Dead-ends		#sol
		AO-FC	AO-RFC	AO-FC	AO-RFC	AO-FC	AO-RFC	
10%	0	1.251	0.382	7,036	2,253	7,038	2,255	0
	10	1.249	**0.379**	7,036	2,253	7,038	2,255	
	20	1.265	0.386	7,036	2,253	7,038	2,255	
20%	0	22.992	15.955	164,491	112,794	162,854	111,158	0
	10	22.994	**15.978**	162,137	110,441	162,345	110,648	
	20	22.999	16.047	161,958	110,262	162,140	110,444	
30%	0	253.289	43.255	2,093,151	350,692	2,046,342	303,883	0
	10	254.250	**42.858**	2,025,869	283,410	2,031,725	289,266	
	20	253.439	43.228	2,020,310	277,851	2,025,878	283,419	

AND/OR Search vs. Bucket Elimination

N=70, K=2, R=5, P=2, C=30, S=3, 20 instances, w*=23, h=31									
t	i	Time			Nodes		Dead-ends		#sol
		BE	AO-FC	AO-RFC	AO-FC	AO-RFC	AO-FC	AO-RFC	
40%	0	**26.400**	1.956	1.263	48,768	21,176	34,582	18,980	0
	10		1.872	**1.231**	30,299	17,954	29,406	17,675	
	20		1.878	1.291	26,335	17,467	21,370	16,384	
50%	0		30.652	35.570	2,883,284	2,707,820	1,095,867	1,032,176	1.64E+12
	10		18.583	18.872	557,478	511,931	341,554	302,322	
	20		12.444	**12.110**	244,635	215,801	146,241	130,219	
60%	0		396.754	511.434	51,223,471	50,089,187	13,199,632	12,844,925	7.02E+14
	10		167.852	182.451	5,881,086	5,707,665	2,318,591	2,241,029	
	20		**80.484**	83.601	1,722,900	1,655,420	718,363	697,237	

N=60, K=2, R=5, P=2, C=40, S=3, 20 instances, w*=23, h=31									
t	i	Time			Nodes		Dead-ends		#sol
		BE	AO-FC	AO-RFC	AO-FC	AO-RFC	AO-FC	AO-RFC	
40%	0	**66.871**	0.655	0.603	9,126	8,510	7,982	7,367	0
	10		0.630	0.568	5,732	5,117	5,282	4,667	
	20		0.632	**0.566**	5,175	4,560	4,461	3,856	
50%	0		3.178	3.021	57,769	54,802	41,149	38,222	6.24E+04
	10		2.993	2.794	30,991	28,121	27,946	25,055	
	20		2.731	**2.578**	24,668	22,522	19,925	17,882	
60%	0		65.171	70.242	2,302,068	2,291,538	1,205,751	1,195,221	7.51E+08
	10		54.101	56.419	791,391	780,861	659,694	649,165	
	20		**39.606**	40.718	459,131	448,659	319,196	308,774	

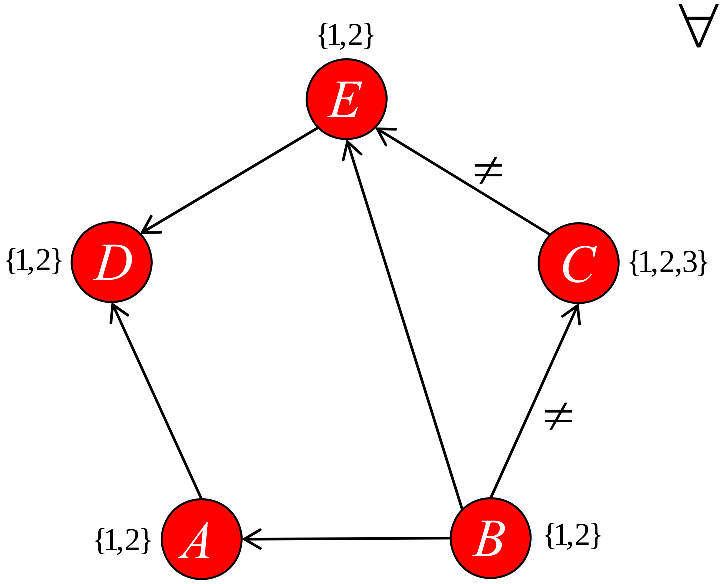
Road Map:Bayesian Networks

- 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):*

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



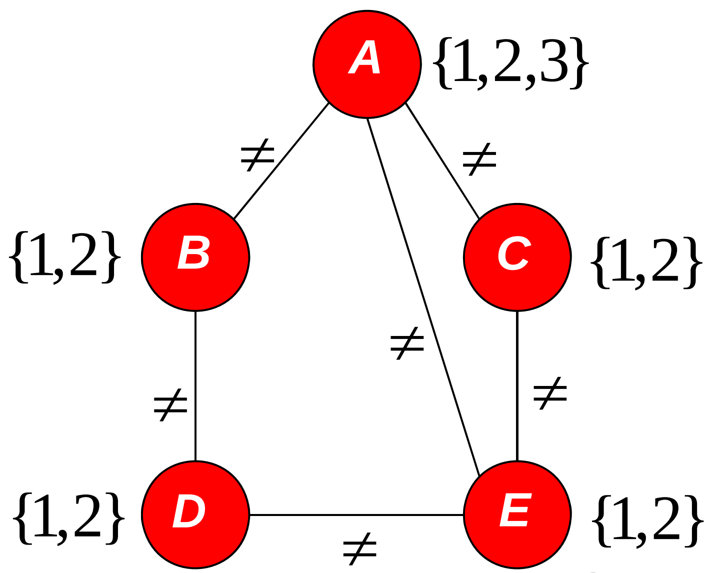
A	B	C	D	E	P
1	2	3	2	1	0.5
2	1	3	1	2	0.5

Brute-force: generate and list all solutions

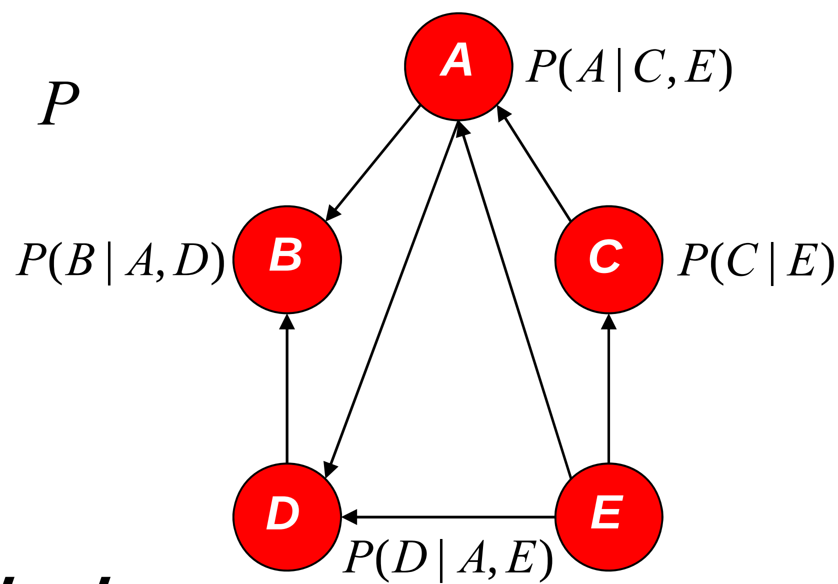
Modeling C*N* as B*N* on the same variables (Approach 2):

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

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



$$R \rightarrow P$$

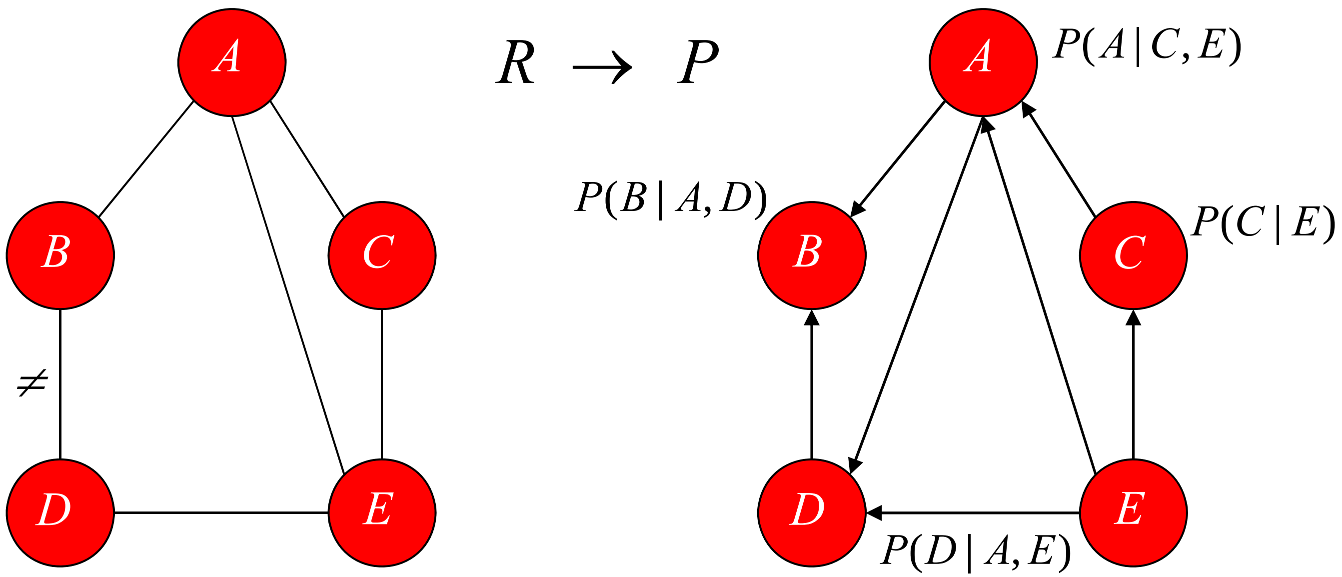


Conversion solved problem

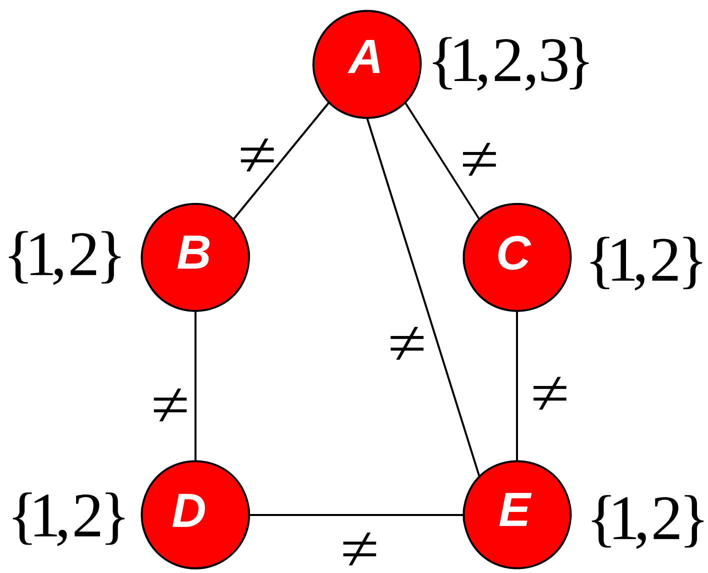
Modeling C*N* as B*N* on the same variables (Approach 2):

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

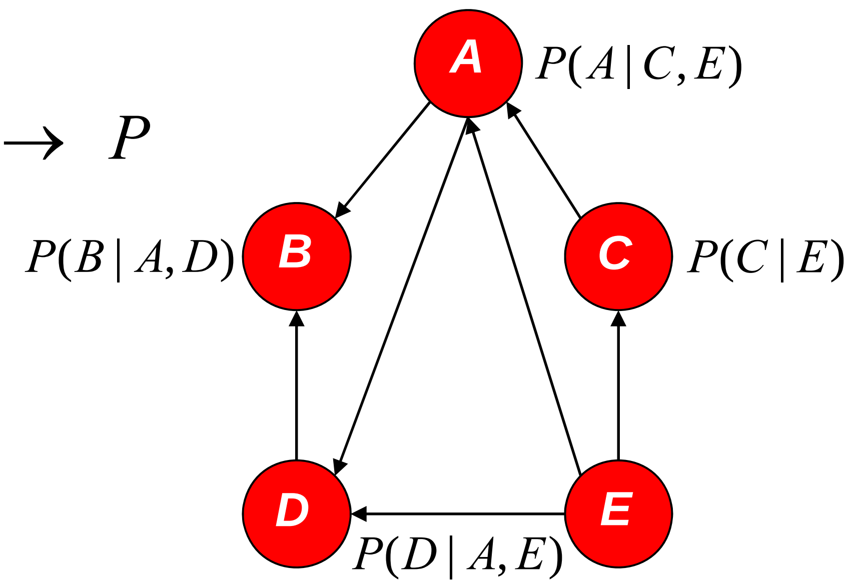
$$P(x_1,\ldots,x_n) = \frac{1}{\#\text{sol}} \qquad if\ (x_1,\ldots,x_n) \text{ is a solution}$$



A variable-elmination-based conversion



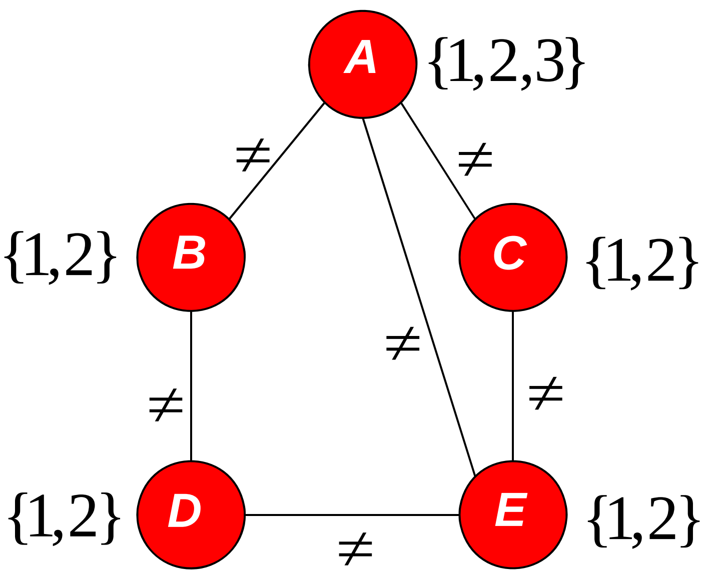
$$R \rightarrow P$$



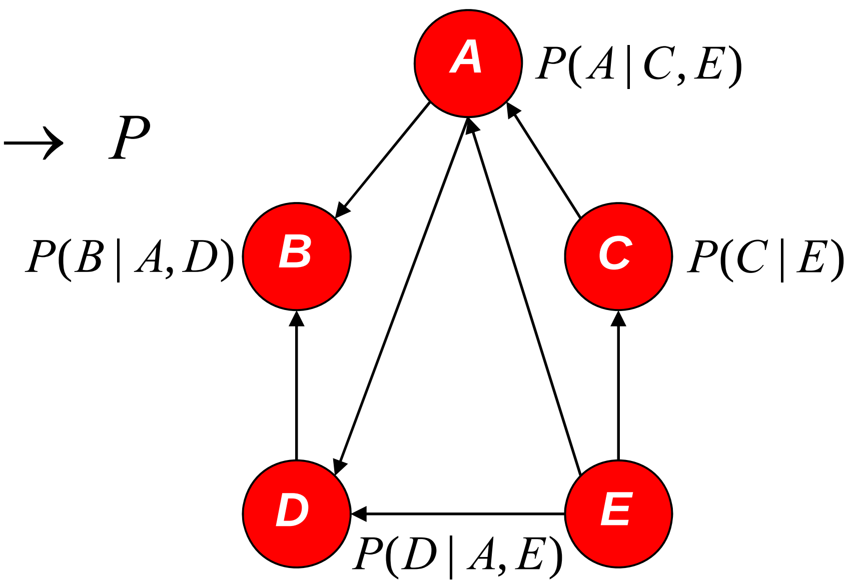
$$P(B \mid A, D) = \frac{R(A,B) * R(B,D)}{\sum_B R(A,B) * R(B,D)}$$

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

A variable-elmination-based conversion



$R \rightarrow P$

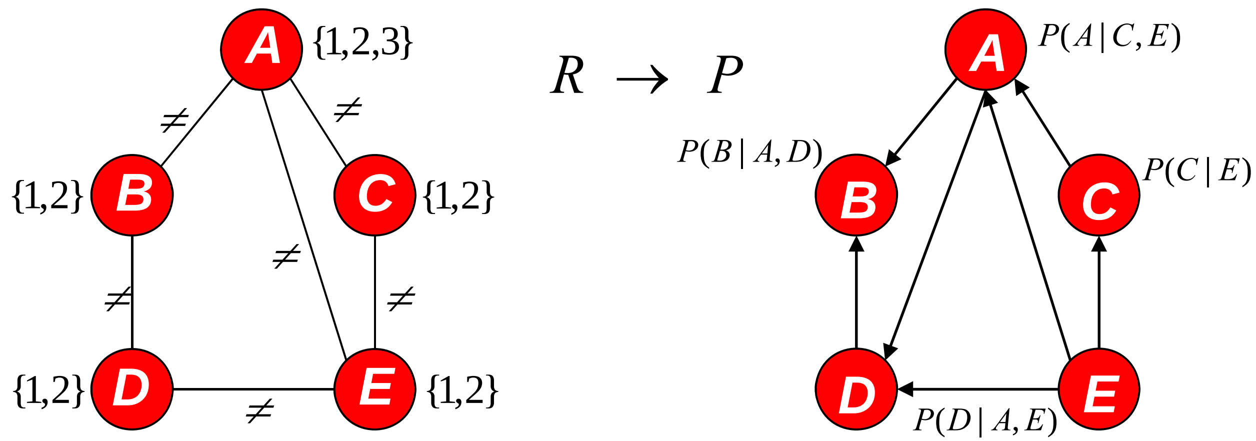


$$P(B \mid A, D) = 1 / \tilde{\prod}_{AB} (R_{AB} \bowtie R_{DB})$$

A	*D*	*B*	*P(B|AD)*
0	1	0	0.5
0	1	1	0.5
0	0	0	1
1	1	0	0.3
1	1	1	0.3
1	1	2	0.3

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

A conversion algorithm



$B:\ R_{AB} \quad R_{BD}$ → $B: P(B \mid A, D)$

$D:\ R_{DE} \quad R_{AD} \quad R'_{AD}$ → $D: P(D \mid A, E)$

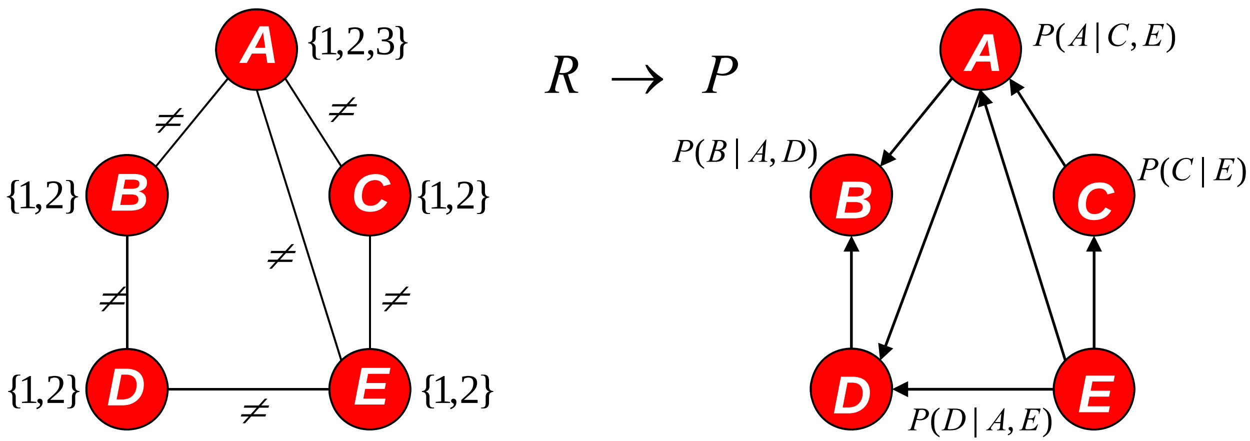
$A:\ R_{AC} \quad R_{AE} \quad R'_{ED}$ → $A: P(A \mid C, E)$

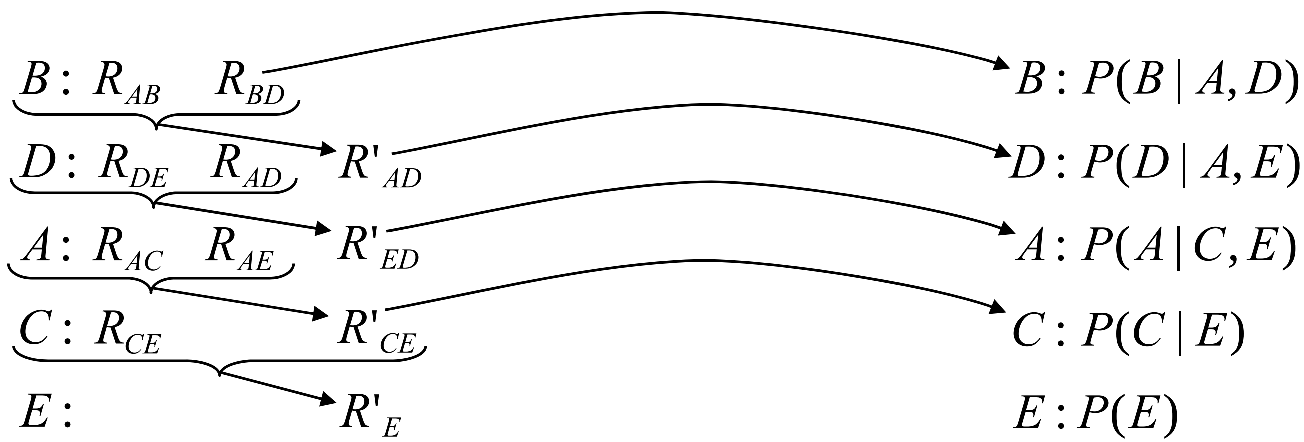
$C:\ R_{CE} \quad R'_{CE}$ → $C: P(C \mid E)$

$E: \quad R'_{E}$ → $E: P(E)$

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

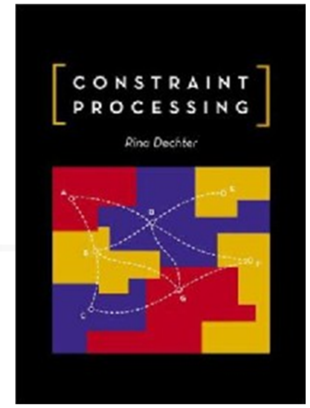
A conversion algorithm





$$P(B \mid A, D) = 1 / \tilde{\prod}_{AB} (R_{AB} \bowtie R_{DB})$$

Thank You



- "Constraint Processing", Morgan Kaufmann, 2003
- Probabilistic networks: Transferring these ideas to Probabilistic network, helping unifying the principles.
- Current work: Mixing probabilistic and deterministic network

Questions?