Constraint Processing; The Graphical Models Perspective

Rina Dechter

Overview and Road Map

- **Introduction:**
  - The constraint network model
- Inference
- Search
- Hybrids of search and inference
- Relationships to Belief networks

Constraint Satisfaction

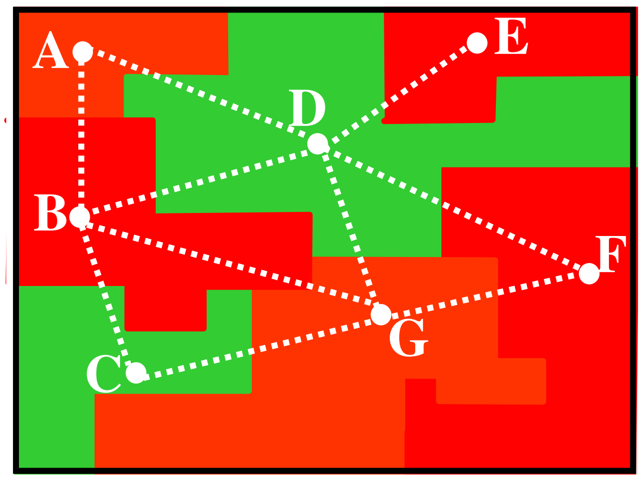
Example: map coloring

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

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

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

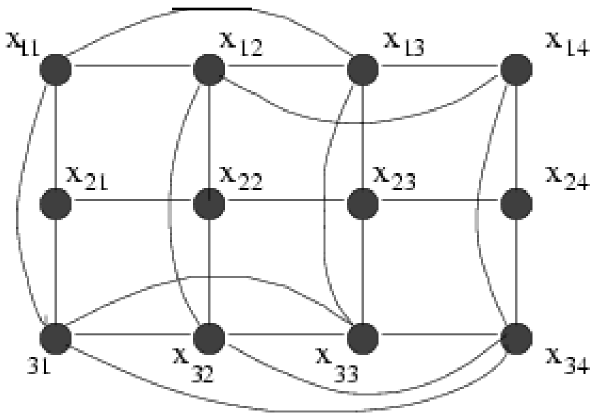
A	B
red	green
red	yellow
green	red
green	yellow
yellow	green
yellow	red



Task: consistency?
Find a solution, all solutions, counting

Constrained Optimization

Example: power plant scheduling



Unit #	Min Up Time	Min Down Time
1	3	2
2	2	1
3	4	1

Variables $= \{X_1, ..., X_n\}$, domain $= \{ON, OFF\}$.
Constraints : $X_1 \vee X_2, \neg X_3 \vee X_4$, min - up and min - down time,
power demand : $\sum \text{Power}(X_i) \geq Demand$

Objective : minimize $\text{TotalFuelCost}(X_1, ..., X_N)$

Applications of Constraint Satisfaction

- Planning and scheduling
- Configuration and design problems
- Circuit diagnosis
- Scene labeling
- Temporal reasoning
- Natural language processing

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
- ***Tasks:*** *consistency?, one or all solutions, counting, optimization*

Constraint's representations

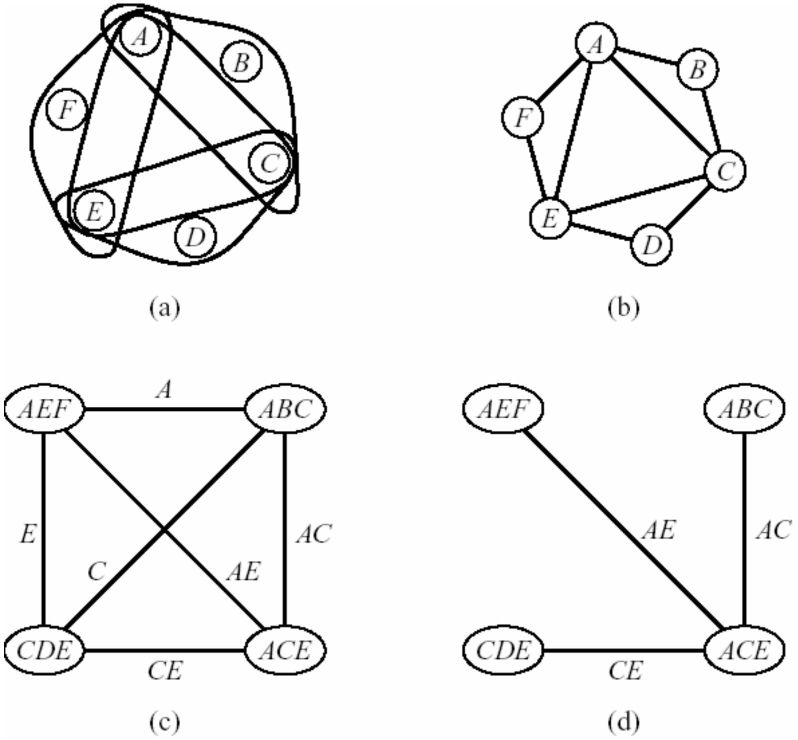
- Relation: allowed tuples

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

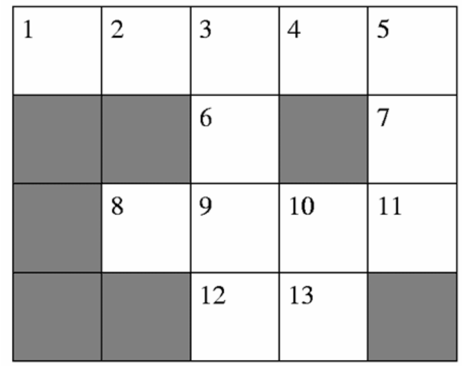
Graph Concepts Reviews:
Hyper Graphs and Dual Graphs

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

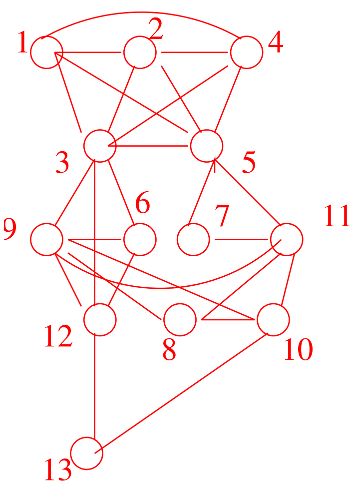


Constraint Graphs:

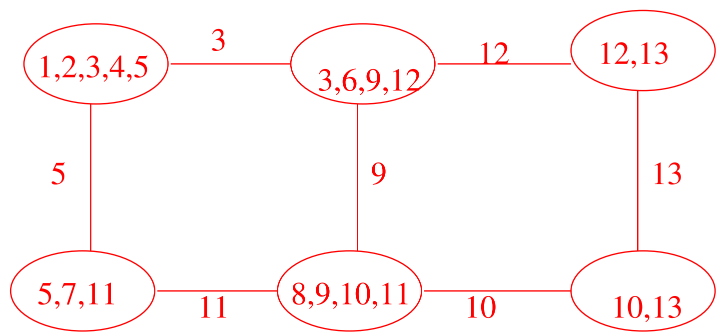
Primal, Dual and Hypergraphs



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



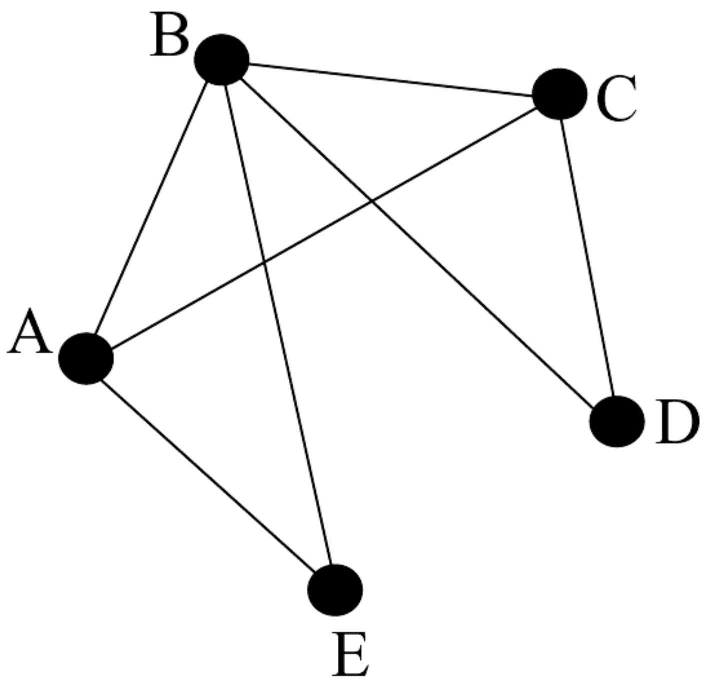
(a)



(b)

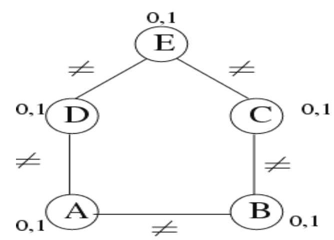
Propositional Satisfiability

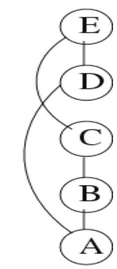
$\varphi = \{(\neg C), (A \lor B \lor C), (\neg A \lor B \lor E), (\neg B \lor C \lor D)\}$.

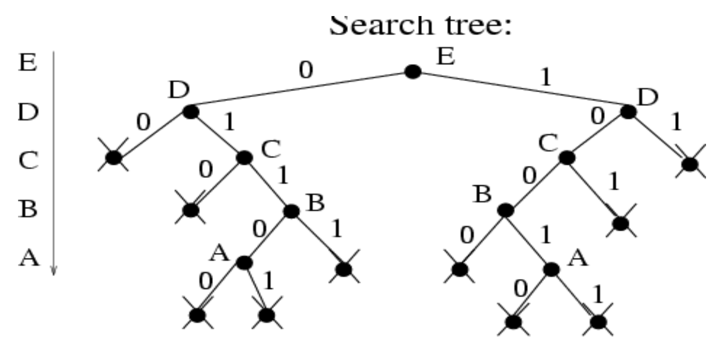


Two Primary Reasoning Methods

- Inference
  - Variable elimination
  - Tree-clustering
- Search
  - Backtracking (conditioning)
- Hybrids of search and inference





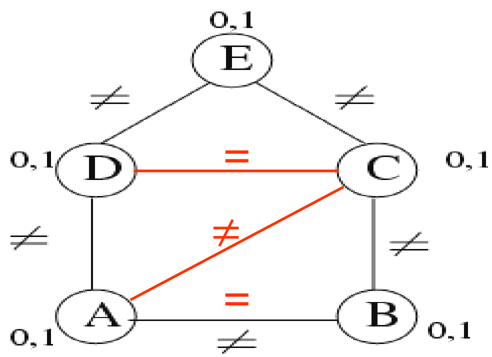


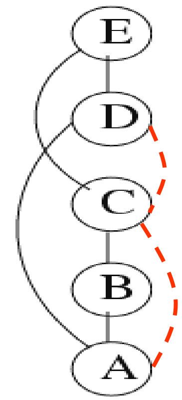
Road Map

- Introduction
- **Inference**:
  - **Variable elimination: Adaptive-consistency, bucket elimination**
  - **Constraint propagation: Arc, path and i-consistency**
- Search
- Hybrids of search and inference
- Relationships to Belief networks

Bucket Elimination

Adaptive Consistency (Dechter & Pearl, 1987)



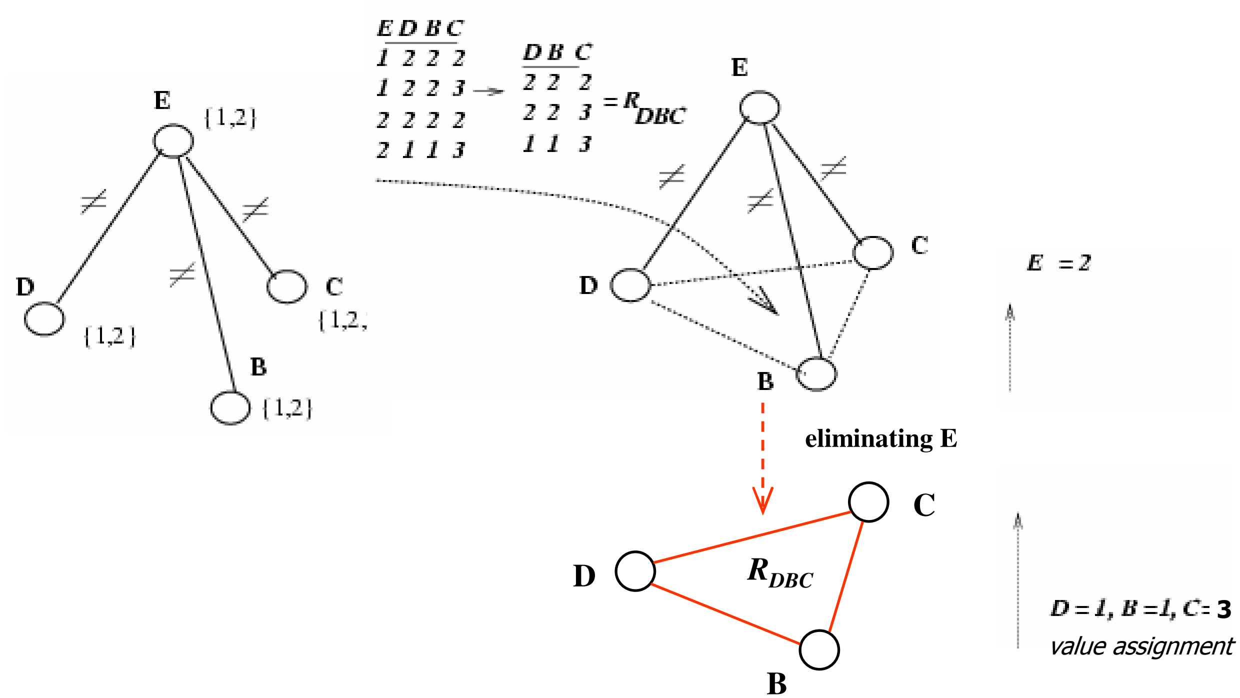


Bucket E:	$E \neq D,\ E \neq C$	
Bucket D:	$D \neq A$	$D = C$
Bucket C:	$C \neq B$	$A \neq C$
Bucket B:	$B \neq A$	$B = A$
Bucket A:		contradiction

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

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

The Idea of Elimination

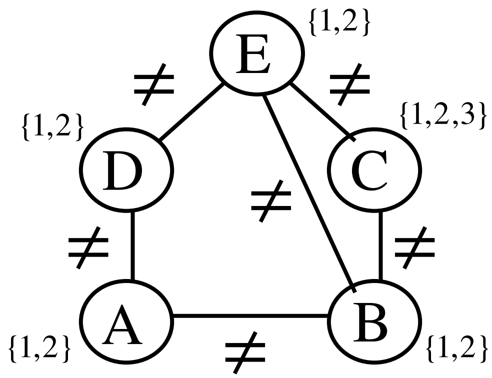


$$R_{DBC} = \prod_{DBC} R_{ED} \bowtie R_{EB} \bowtie R_{EC}$$

Eliminate variable E $\Leftrightarrow$ join and project

Bucket Elimination

Adaptive Consistency (Dechter & Pearl, 1987)



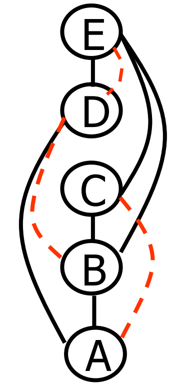
$Bucket(E):\ \mathrm{E} \neq \mathrm{D},\ \mathrm{E} \neq \mathrm{C},\ \mathrm{E} \neq \mathrm{B}$

$Bucket(D):\ \mathrm{D} \neq \mathrm{A}\ \ ||\ R_{DCB}$

$Bucket(C):\ \mathrm{C} \neq \mathrm{B}\ \ ||\ R_{ACB}$

$Bucket(B):\ \mathrm{B} \neq \mathrm{A}\ \ ||\ R_{AB}$

$Bucket(A):\ \ R_{A}$



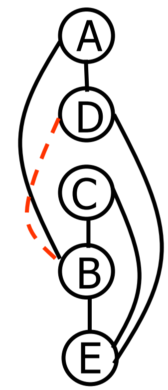
$Bucket(A):\ \mathrm{A} \neq \mathrm{D},\ \mathrm{A} \neq \mathrm{B}$

$Bucket(D):\ \mathrm{D} \neq \mathrm{E}\ \ ||\ R_{DB}$

$Bucket(C):\ \mathrm{C} \neq \mathrm{B},\ \mathrm{C} \neq \mathrm{E}$

$Bucket(B):\ \mathrm{B} \neq \mathrm{E}\ \ ||\ R^{D}_{BE},\ R^{C}_{BE}$

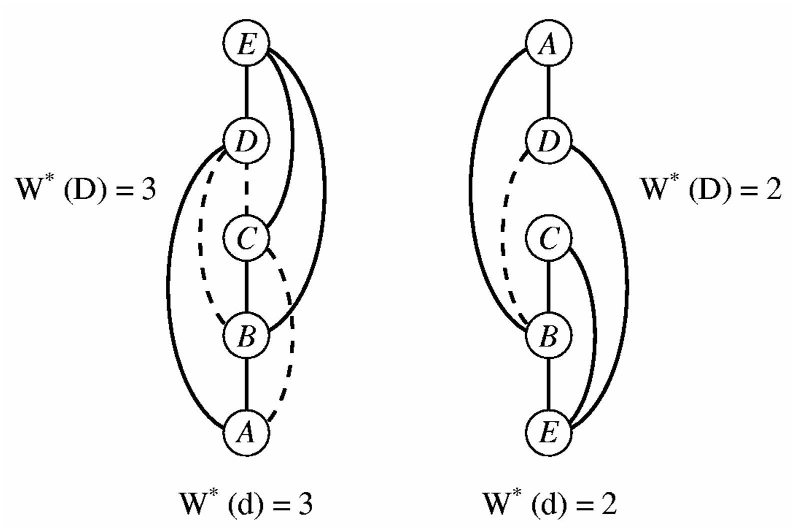
$Bucket(E):\ \ ||\ R_{E}$



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

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

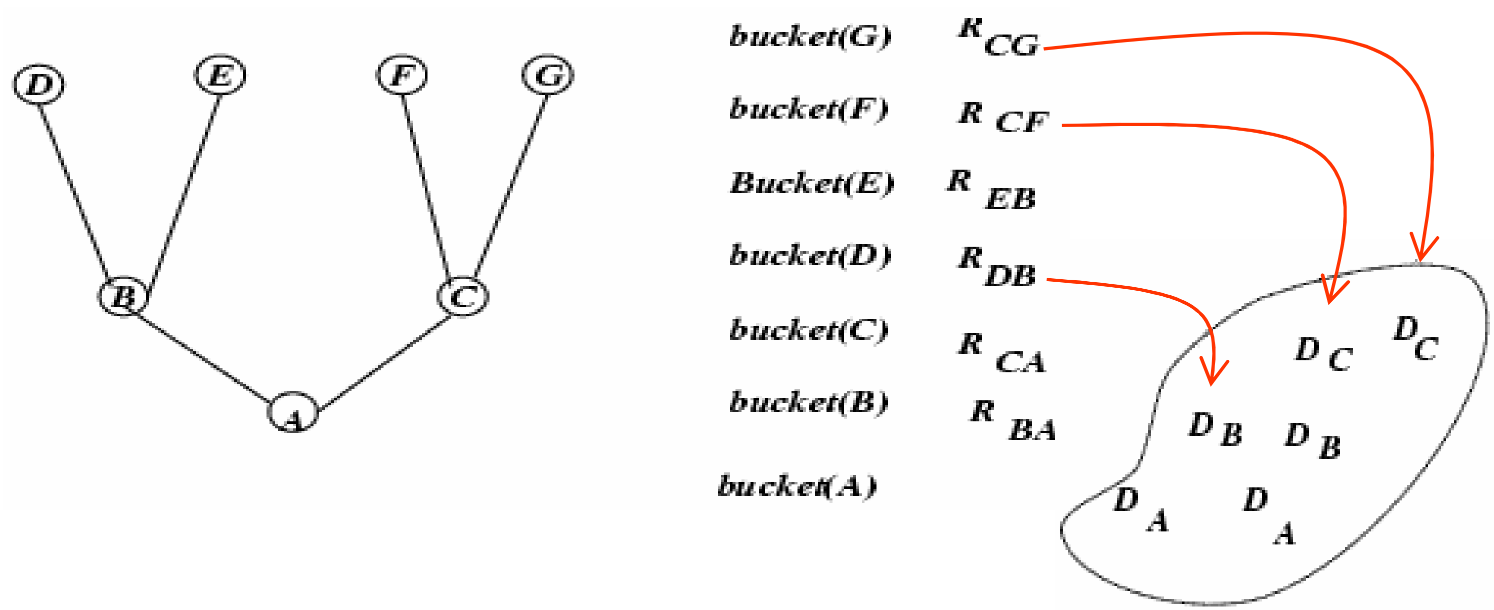
The induced-width



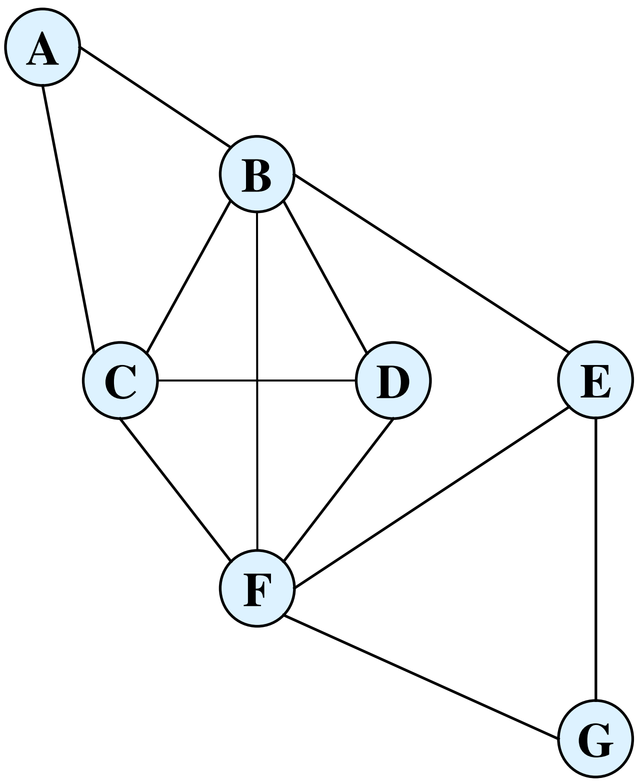
- **Width along ordering *d*, w(d):**
  - **max # of previous parents**
- Induced width w*(d):
  - **The width in the ordered *induced graph***
- Induced-width w*:
  - **Smallest induced-width over all orderings**
- Finding w*
  - **NP-complete** ***(Arnborg, 1985) but greedy heuristics (min-fill).***

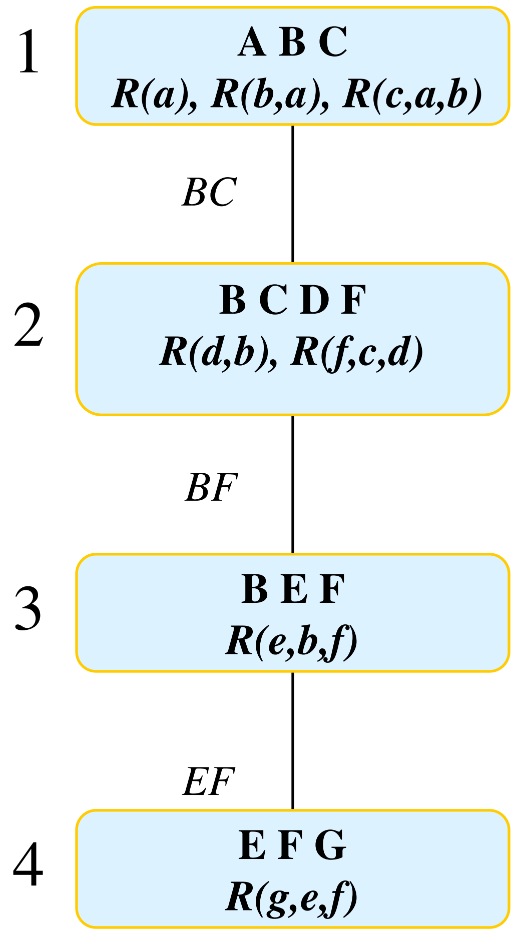
Solving Trees
(Mackworth and Freuder, 1985)

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



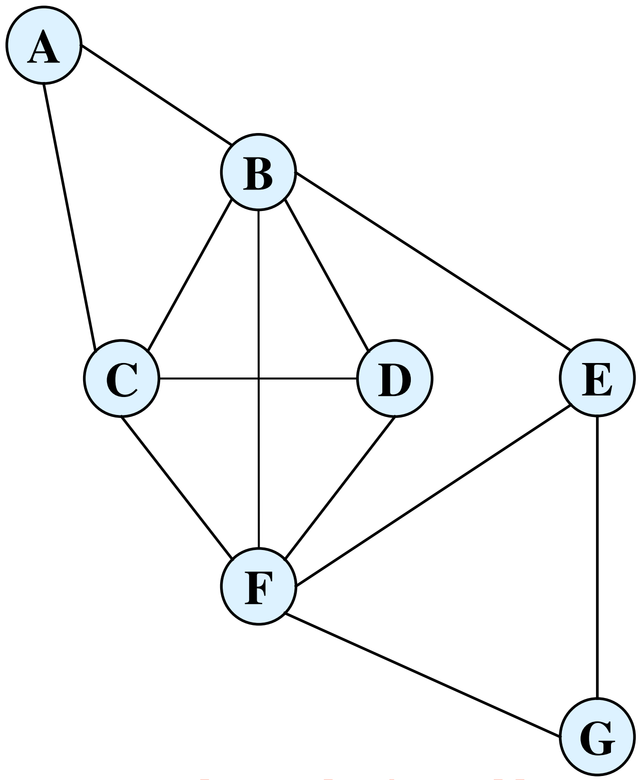
Tree Decomposition

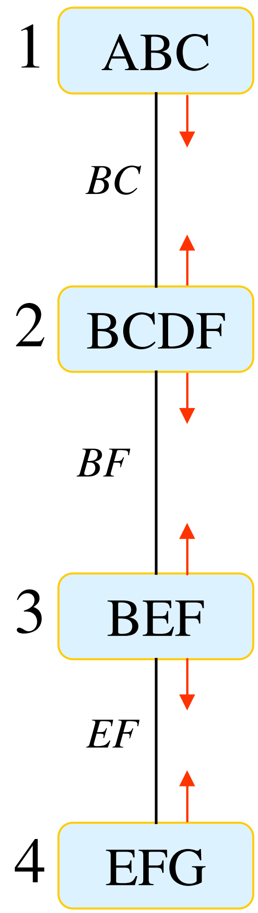




- Each function in a cluster
- Satisfy running intersection property

CTE: Cluster Tree Elimination





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

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

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

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

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

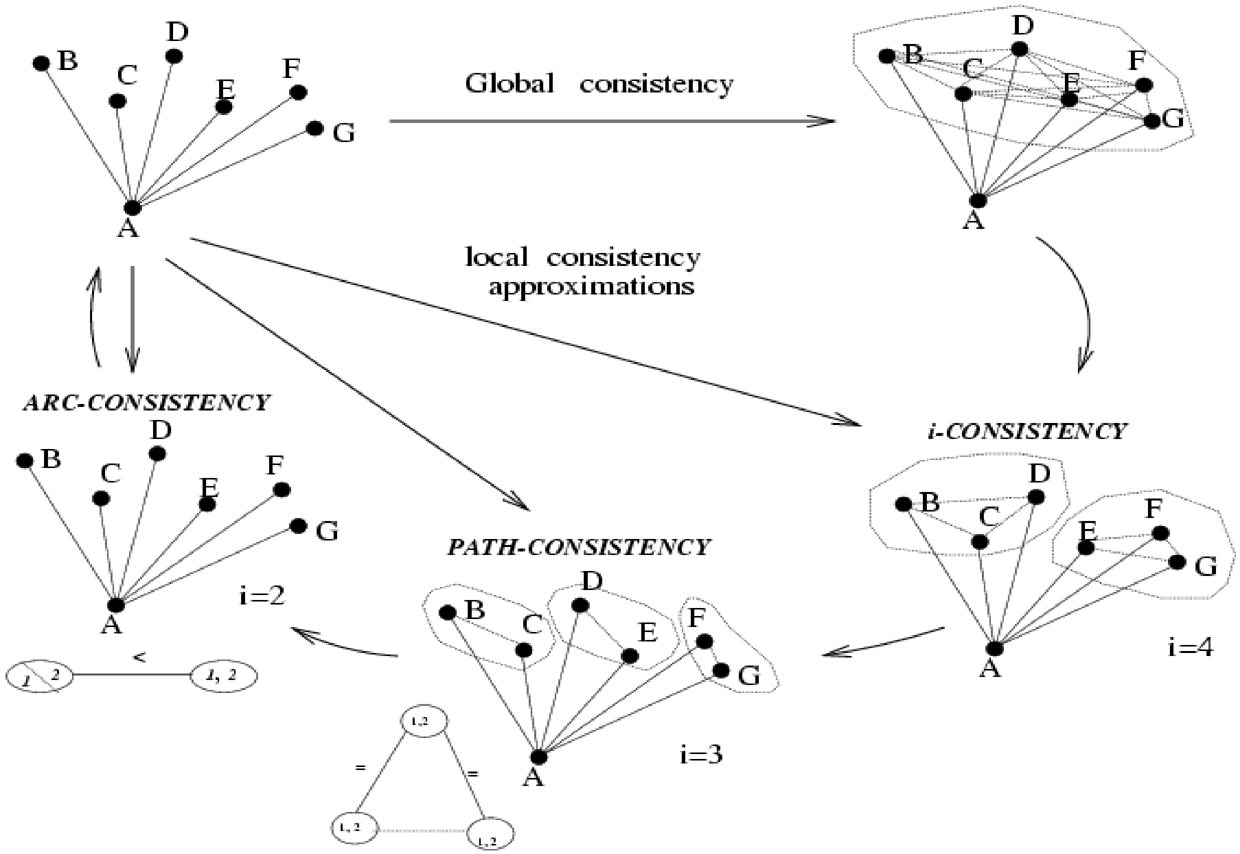
$$h_{(4,3)}(e,f) = \Downarrow_{g} R(g,e,f)$$

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

Road Map

- Introduction
- **Inference**:
  - **Variable elimination: Adaptive-consistency, bucket elimination**
  - **Constraint propagation**: **Arc, path and i-consistency**
- Search
- Hybrids of search and inference
- Relationships to Belief networks

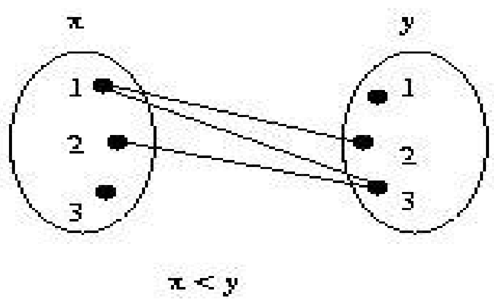
From Global to Local Consistency



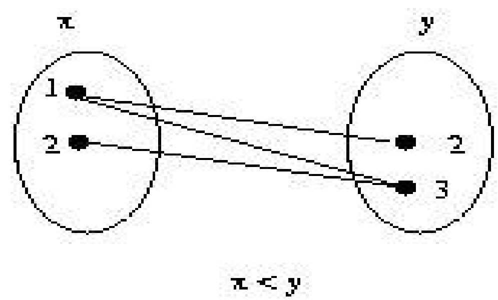
Arc-consistency

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

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



(a)

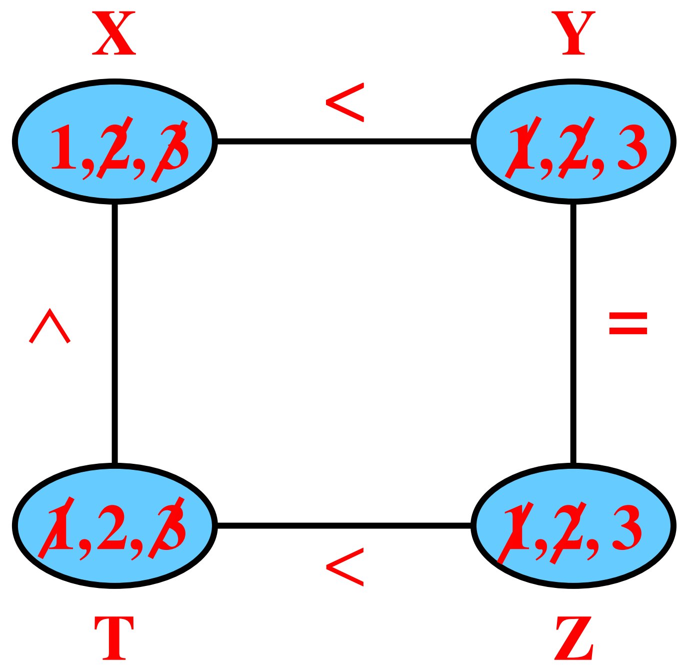


(b)

$$\text{Revise(x, y) reduces domain of } X \text{ to } R_X = \{1,2\}, O(k^2).$$

Arc-consistency

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



Arc-consistency

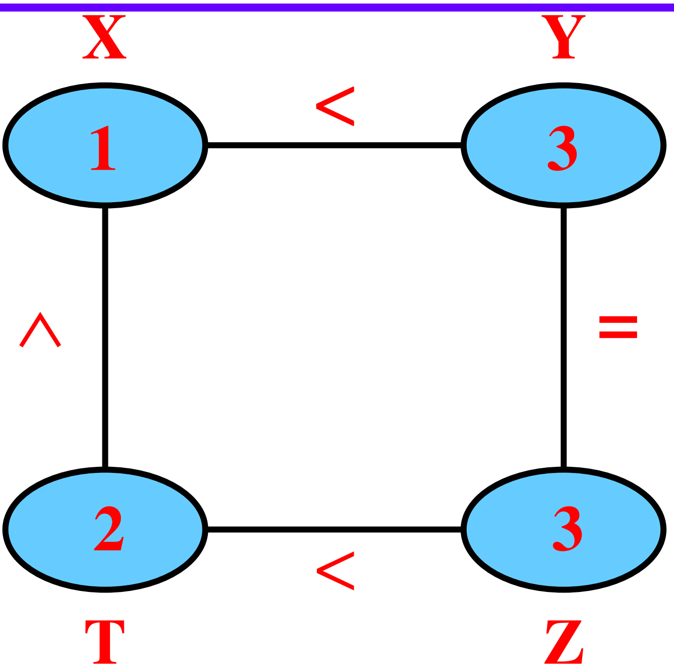
1 ≤ X, Y, Z, T ≤ 3

X < Y

Y = Z

T < Z

X ≤ T



- Only domains are reduced: $R_X \leftarrow \prod_X R_{XY} \bowtie D_Y$
- Incorporated into backtracking search

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)…

Examples:
Relational and Generalized Arc-consistency

- Linear inequalities

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

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

- Example of relational arc-consistency

$$A \wedge B \rightarrow G, \neg G, \Rightarrow \neg A \vee \neg B$$

Path-consistency

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

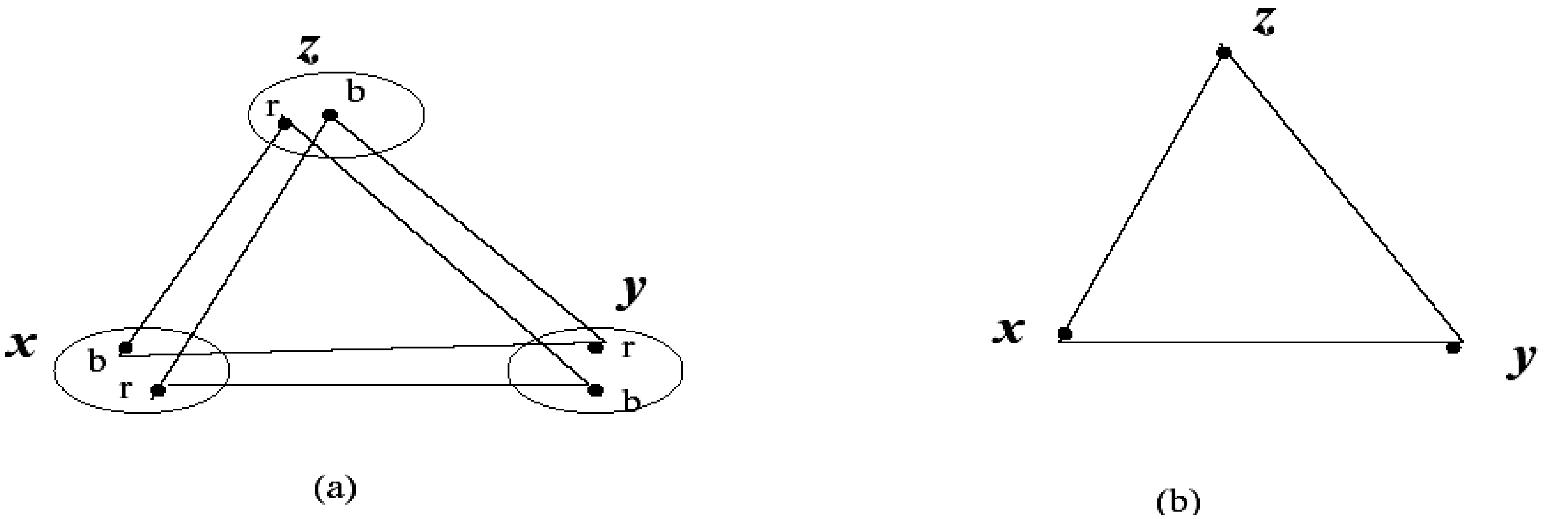


Figure 3.8: (a) The matching diagram of a 2-value graph coloring problem. (b) Graphical picture of path-consistency using the matching diagram.

Example: path-consistency

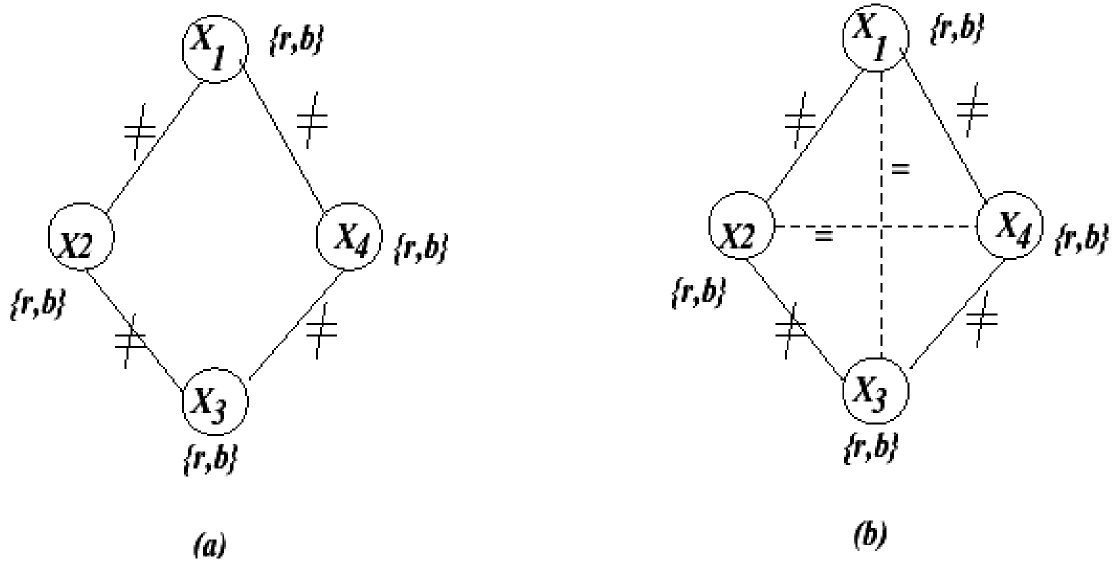


Figure 3.12: A graph-coloring graph (a) before path-consistency (b) after path-consistency

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**

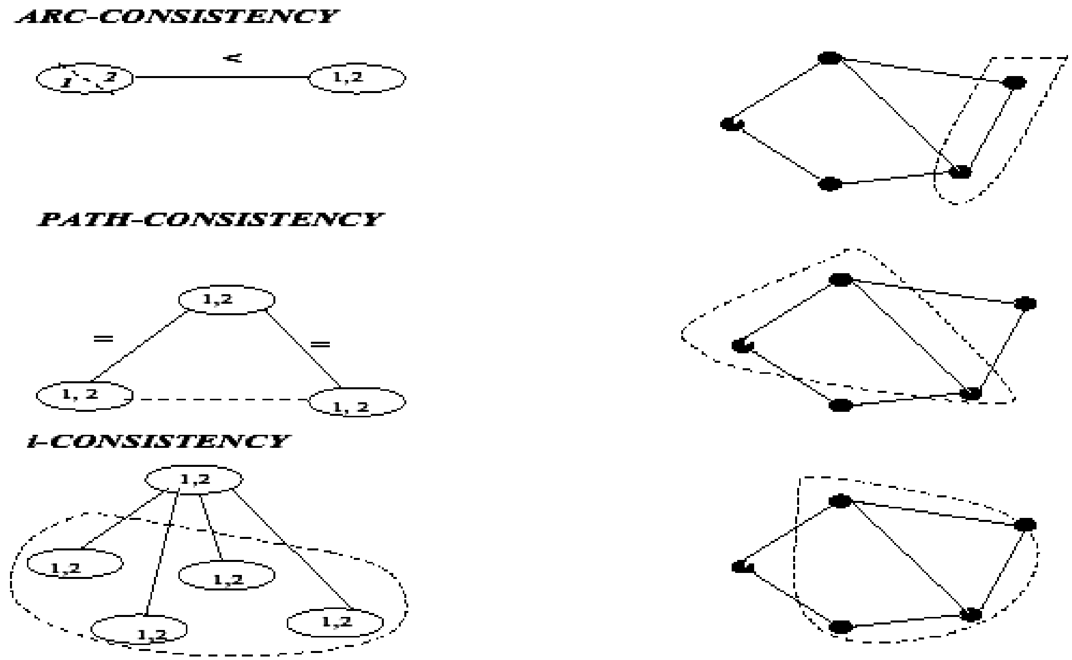


Figure 3.17: The scope of consistency enforcing: (a) arc-consistency, (b) path-consistency, (c) i-consistency

Distributed Relational Arc-consistency

The message that R2 sends to R1 is

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

R1 updates its relation and domains and sends messages to neighbors

$$D_i \leftarrow D_i \cap (\bowtie_{k\in ne(i)} D_k^i)$$

R_1

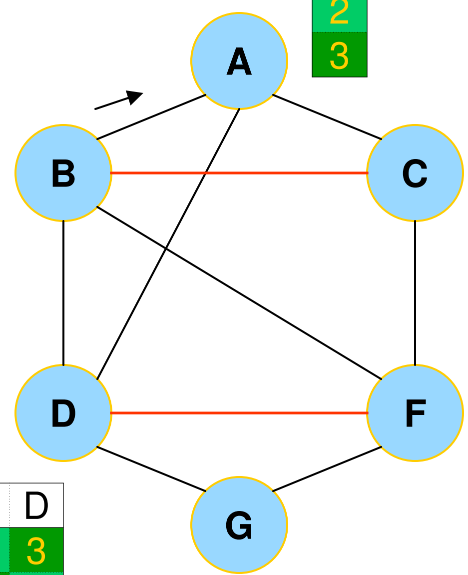
A
1
2
3

R_2

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

R_3

A	C
1	2
3	2



R_4

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

R_5

B	C	F
1	2	3
3	2	1

R_6

D	F	G
1	2	3
2	1	3

Distributed Relational Arc-consistency

The message that R2 sends to R1 is

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

R1 updates its relation and domains and sends messages to neighbors

$$D_i \leftarrow D_i \cap (\bowtie_{k \in ne(i)} D_k^i)$$

R_1

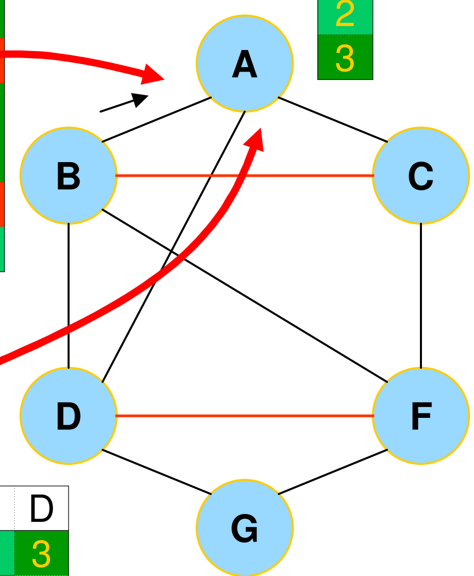
A
1
2
3

R_2

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

R_3

A	C
1	2
3	2



R_4

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

R_5

B	C	F
1	2	3
3	2	1

R_6

D	F	G
1	2	3
2	1	3

DR-AC on a Dual Join-Graph

R_1

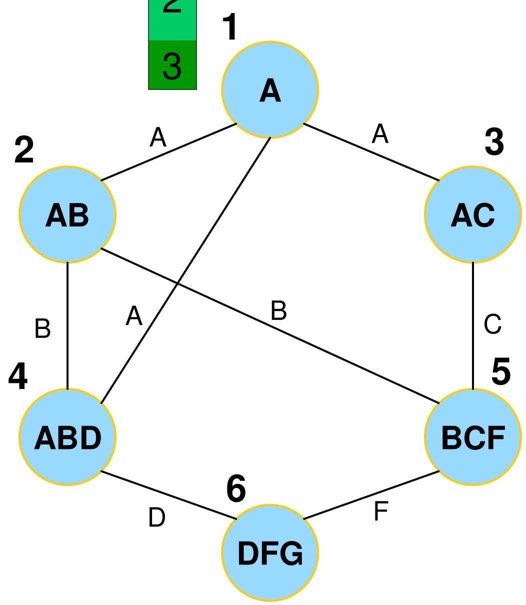
A
1
2
3

R_2

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

R_3

A	C
1	2
3	2



R_4

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

R_5

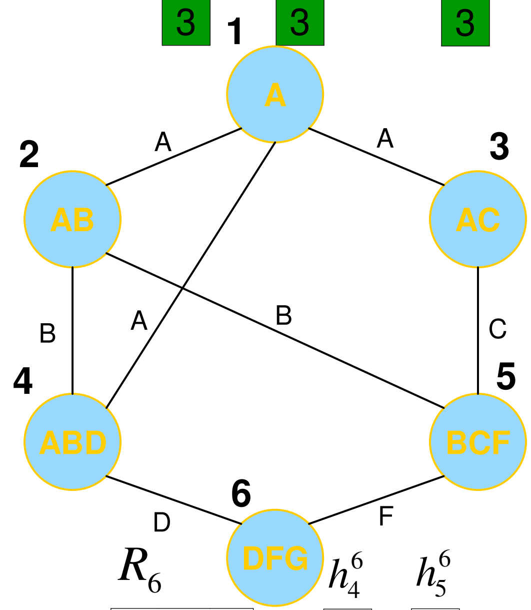
B	C	F
1	2	3
3	2	1

R_6

D	F	G
1	2	3
2	1	3

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

Iteration 1



R_1

A
1
2
3

h_2^1

A
1
2
3

h_3^1

A
1
3

h_4^1

A
1
2
3

h_5^2

B
1
3

h_4^2

B
1
2
3

h_1^2

A
1
2
3

R_2

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

R_3

A	C
1	2
3	2

h_1^3

A
1
2
3

h_5^3

C
2

h_6^4

D
1
2

h_2^4

B
1
2
3

h_1^4

A
1
2
3

R_4

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

R_5

B	C	F
1	2	3
3	2	1

h_2^5

B
1
2
3

h_3^5

C
2

h_6^5

F
1
3

R_6

D	F	G
1	2	3
2	1	3

h_4^6

D
1
2
3

h_5^6

F
1
3

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

Iteration 1

R_1

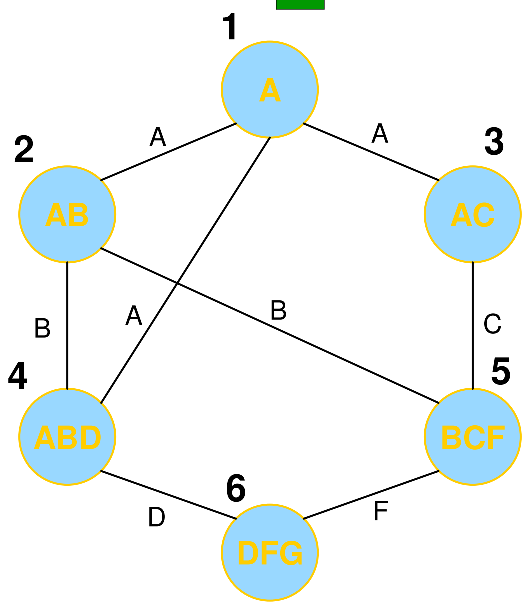
A
1
3

R_2

A	B
1	3
2	1
2	3
3	1

R_3

A	C
1	2
3	2



R_4

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

R_5

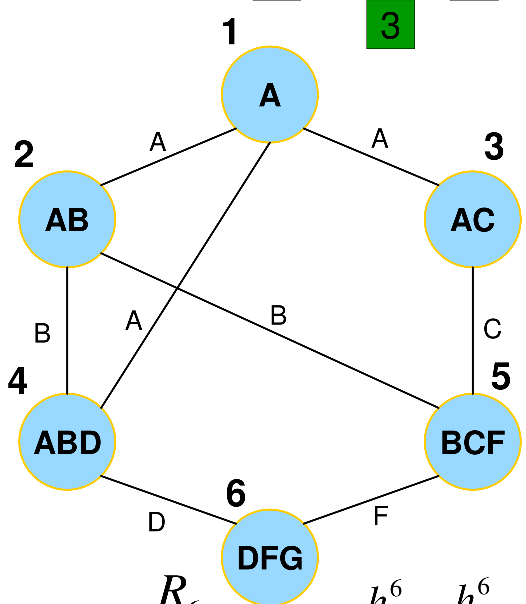
B	C	F
1	2	3
3	2	1

R_6

D	F	G
2	1	3

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

Iteration 2



R_1

A
1
3

h_2^1

A
1
2
3

h_3^1

A
1
3

h_4^1

A
1
2
3

h_5^2

B
1
3

h_4^2

B
1
2
3

h_1^2

A
1
3

R_2

A	B
1	3
2	1
2	3
3	1

R_3

A	C
1	2
3	2

h_1^3

A
1
3

h_5^3

C
2

h_6^4

D
2

h_2^4

B
1
3

h_1^4

A
1
3

R_4

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

R_5

B	C	F
1	2	3
3	2	1

h_2^5

B
1
3

h_3^5

C
2

h_6^5

F
1

R_6

D	F	G
2	1	3

h_4^6

D
1
2

h_5^6

F
1
3

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

Iteration 2

R_1

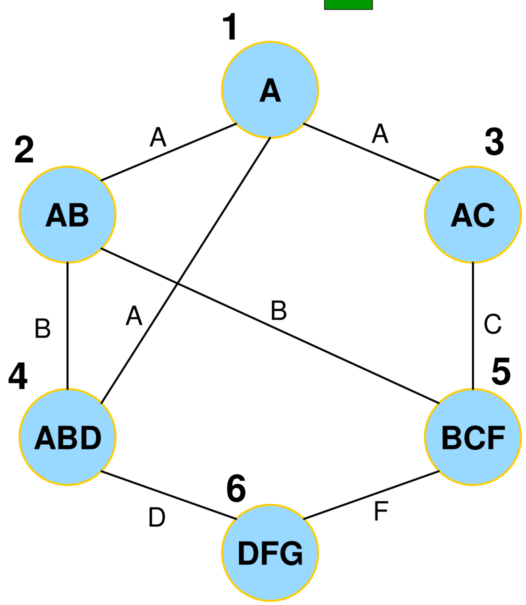
A
1
3

R_2

A	B
1	3
3	1

R_3

A	C
1	2
3	2



R_4

A	B	D
1	3	2
3	1	2

R_5

B	C	F
3	2	1

R_6

D	F	G
2	1	3

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

Iteration 3

R_1

A
1
3

h_2^1

A
1
3

h_3^1

A
1
3

h_4^1

A
1
3

h_5^2

B
3

h_4^2

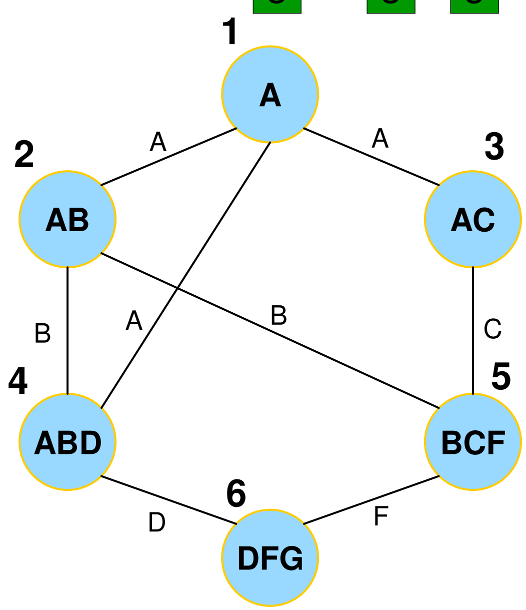
B
1
3

h_1^2

A
1
3

R_2

A	B
1	3
3	1



R_3

A	C
1	2
3	2

h_1^3

A
1
3

h_5^3

C
2

h_6^4

D
2

h_2^4

B
1
3

h_1^4

A
1
3

R_4

A	B	D
1	3	2
3	1	2

R_5

B	C	F
3	2	1

h_2^5

B
1
3

h_3^5

C
2

h_6^5

F
1

R_6

D	F	G
2	1	3

h_4^6

D
2

h_5^6

F
1

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

Iteration 3

R_1

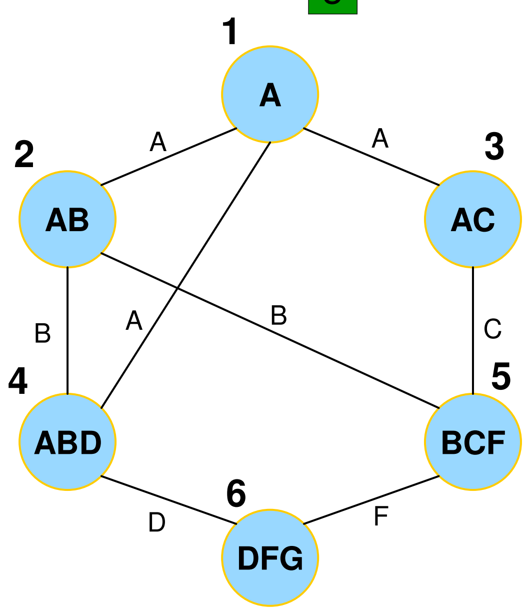
A
1
3

R_2

A	B
1	3

R_3

A	C
1	2
3	2



R_4

A	B	D
1	3	2
3	1	2

R_5

B	C	F
3	2	1

R_6

D	F	G
2	1	3

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

Iteration 4

R_1

A
1
3

h_2^1

A
1

h_3^1

A
1
3

h_4^1

A
1
3

h_5^2

B
3

h_4^2

B
1
3

h_1^2

A
1
3

R_2

A	B
1	3

R_3

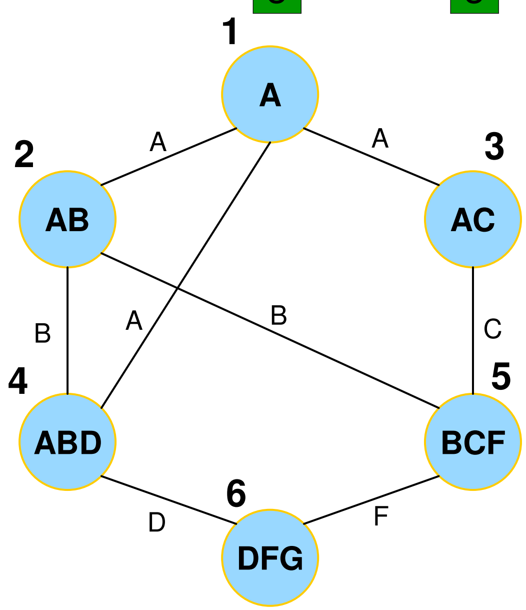
A	C
1	2
3	2

h_1^3

A
1
3

h_5^3

C
2



h_6^4

D
2

h_2^4

B
1
3

h_1^4

A
1
3

R_4

A	B	D
1	3	2
3	1	2

R_5

B	C	F
3	2	1

h_2^5

B
3

h_3^5

C
2

h_6^5

F
1

R_6

D	F	G
2	1	3

h_4^6

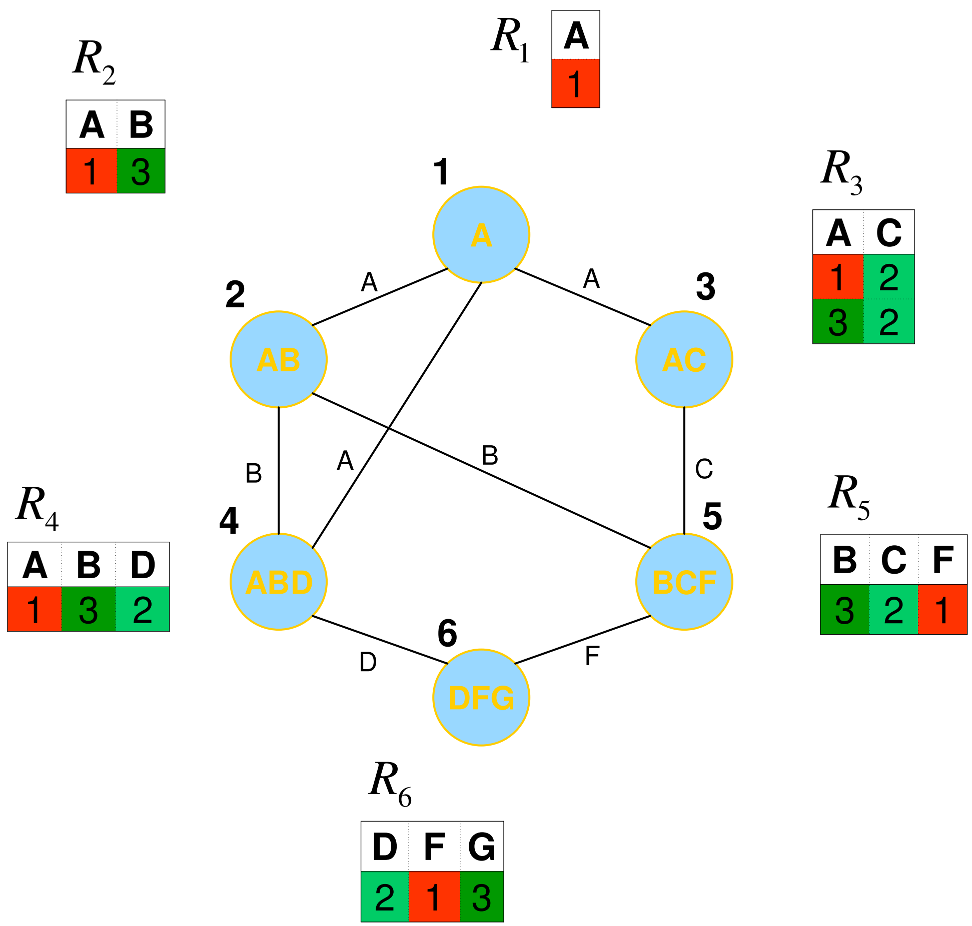
D
2

h_5^6

F
1

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

Iteration 4



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

Iteration 5

R_1

A
1

h_2^1

A
1

h_3^1

A
1

h_4^1

A
1

h_5^2

B
3

h_4^2

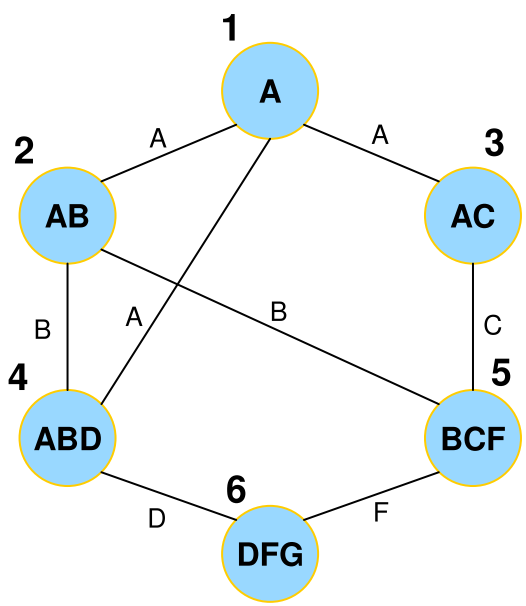
B
3

h_1^2

A
1

R_2

A	B
1	3



R_3

A	C
1	2
3	2

h_1^3

A
1

h_5^3

C
2

h_6^4

D
2

h_2^4

B
3

h_1^4

A
1

R_4

A	B	D
1	3	2

R_5

B	C	F
3	2	1

h_2^5

B
3

h_3^5

C
2

h_6^5

F
1

R_6

D	F	G
2	1	3

h_4^6

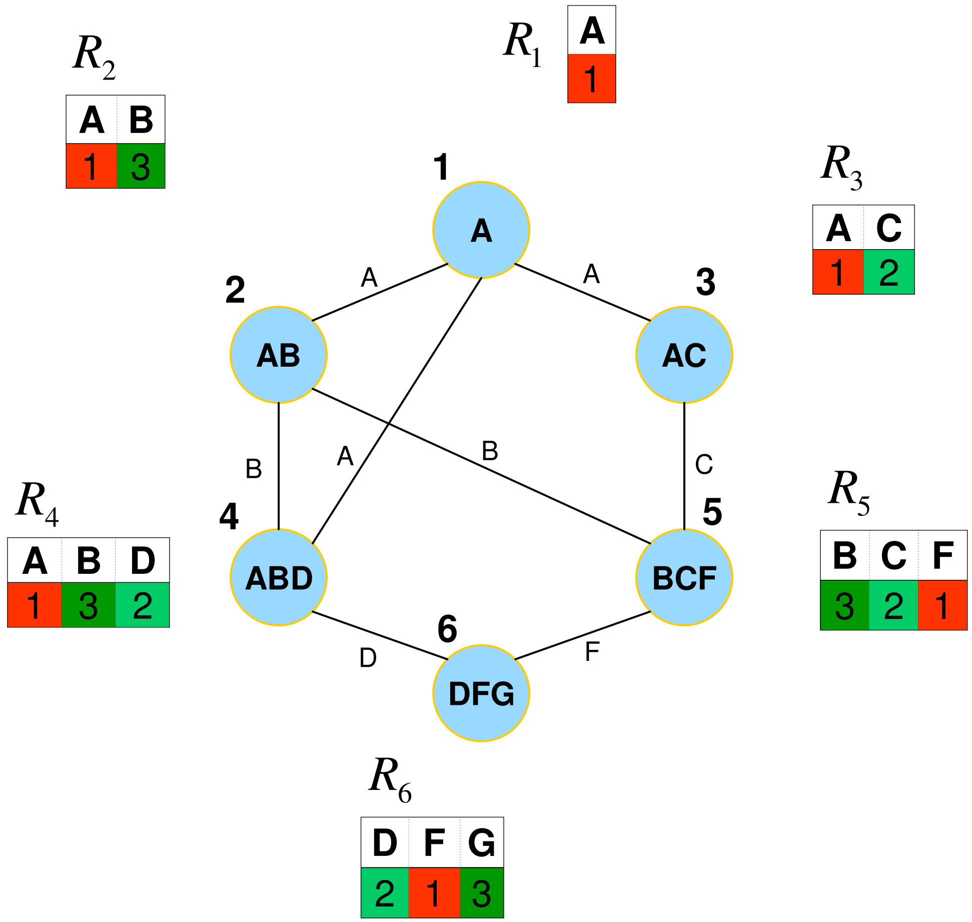
D
2

h_5^6

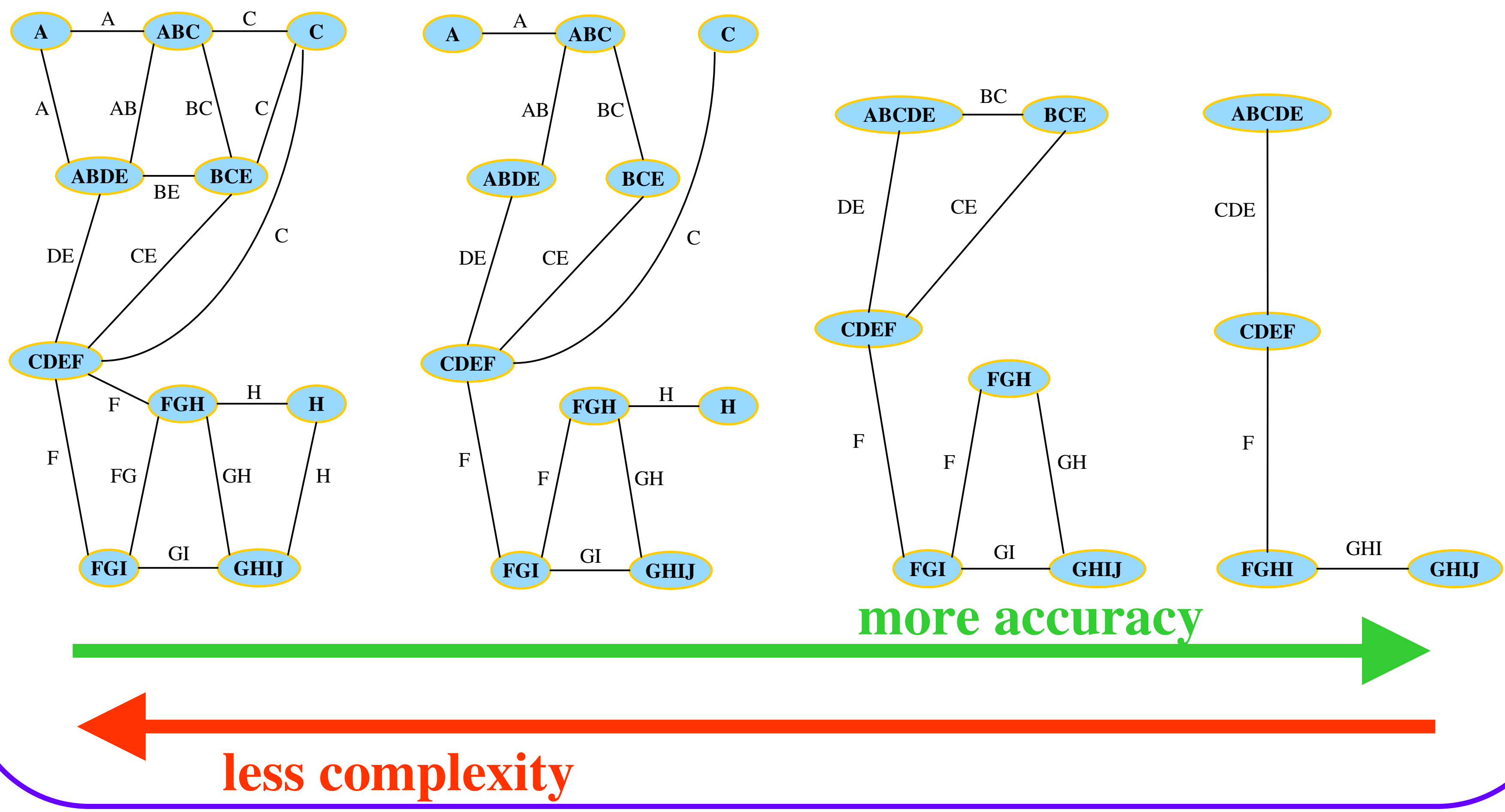
F
1

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

Iteration 5



Join-Graphs



A
ABC
C
ABDE
BCE
CDEF
FGH
H
FGI
GHIJ
ABCDE
FGHI
more accuracy
less complexity

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:

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

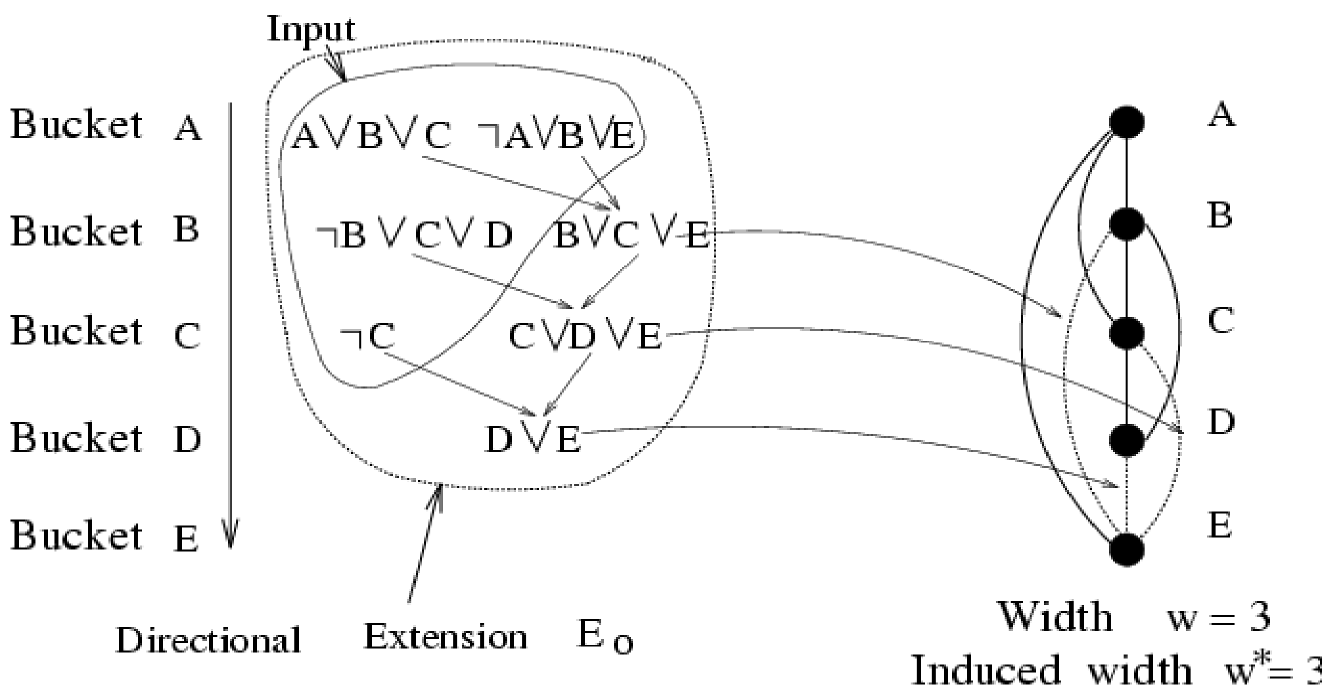
$\text{res}(\neg C \vee A, C) \Rightarrow A$

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

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



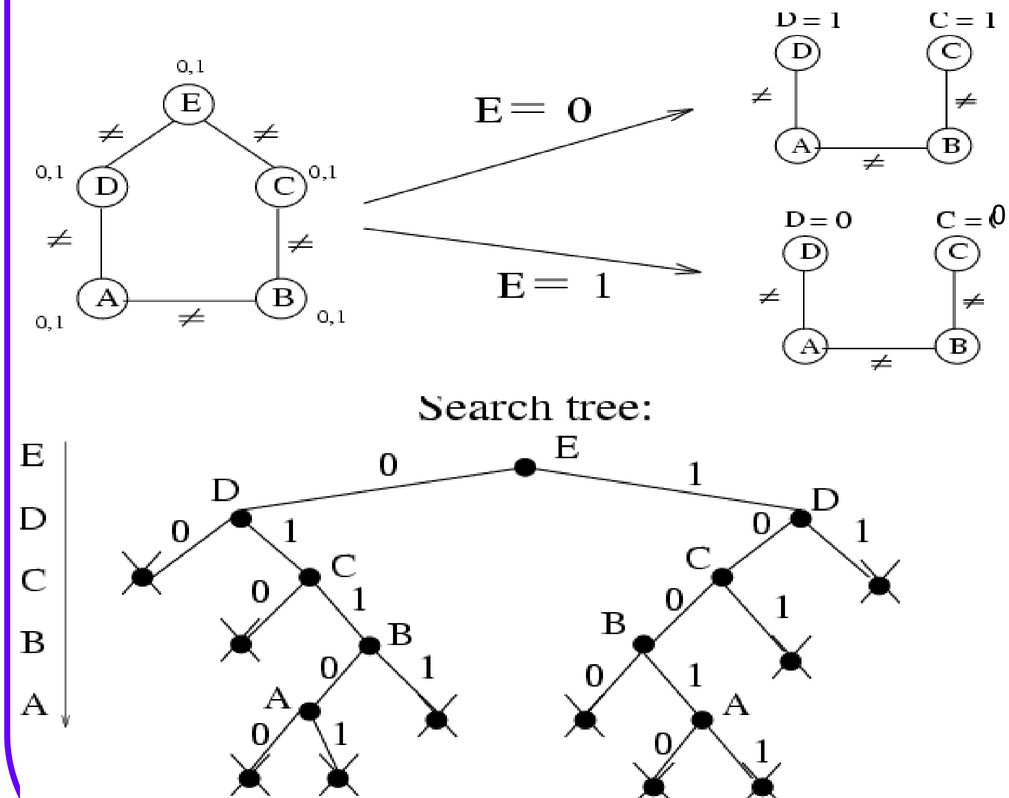
$$| bucket_i | = O(\exp(w^*))$$

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

Road Map

- Introduction
- Inference
- **Search:**
  - **Look-ahead schemes**
  - **Look-back schemes**
  - **The alternative AND/OR search space**
- Hybrids of search and inference
- Relationships to Belief networks

The Search Space



*The search-space:

A tree of all partial solutions

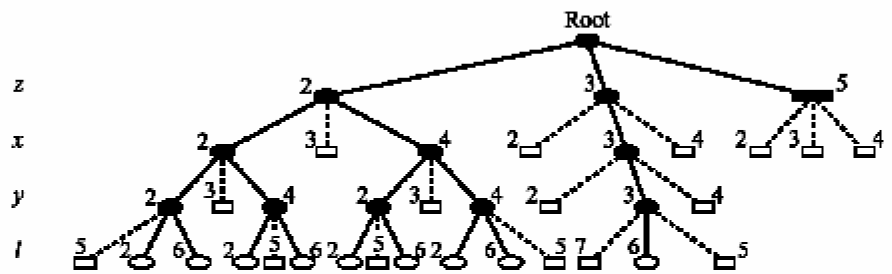
*A partial solution: (a1,…,aj) satisfying all relevant constraints

*Size depends on:

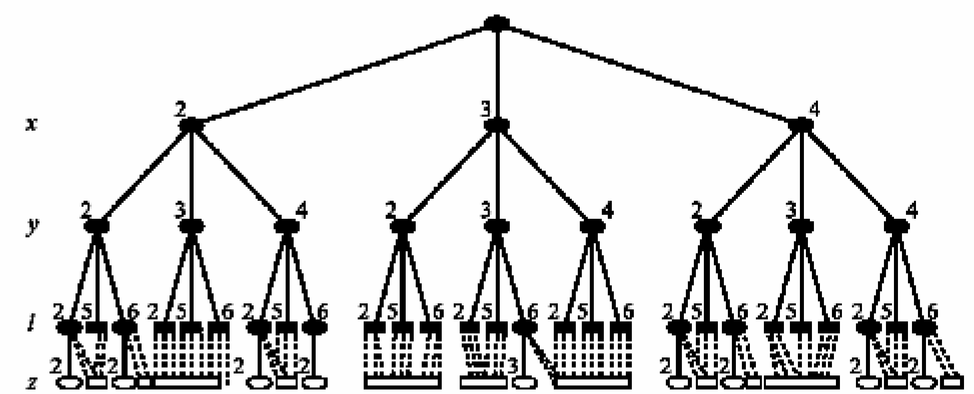
- Variable ordering
- Local-consistency

Complexity : *O(exp(n))*

The Effect of Variable Ordering

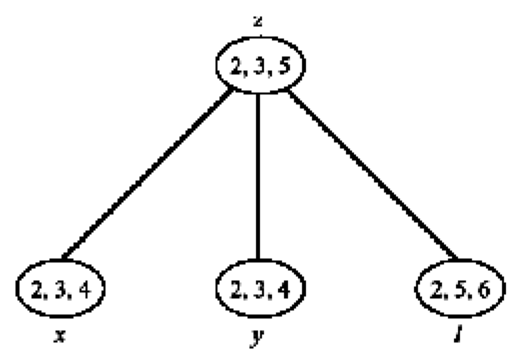


(b)



(c)

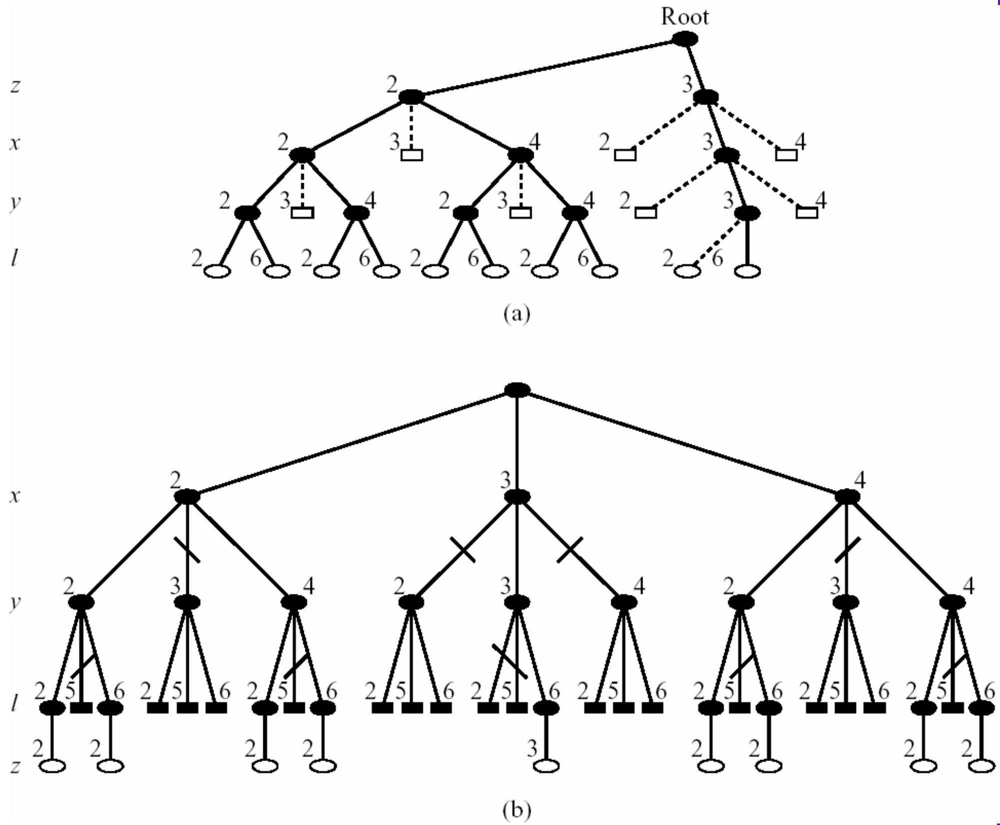
z divides x, y and t



(a)

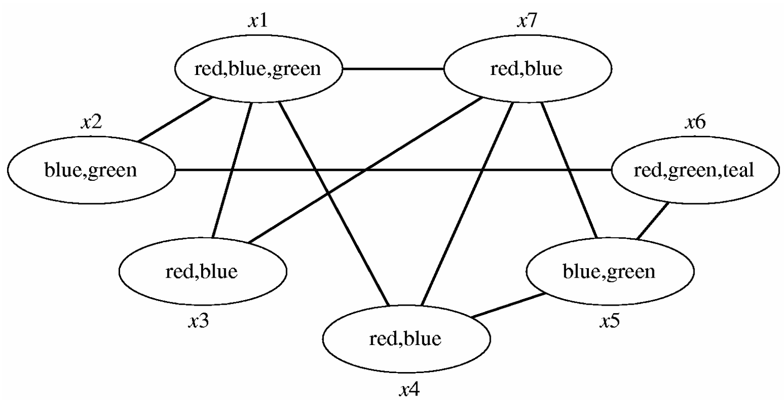
The Effect of Consistency Level

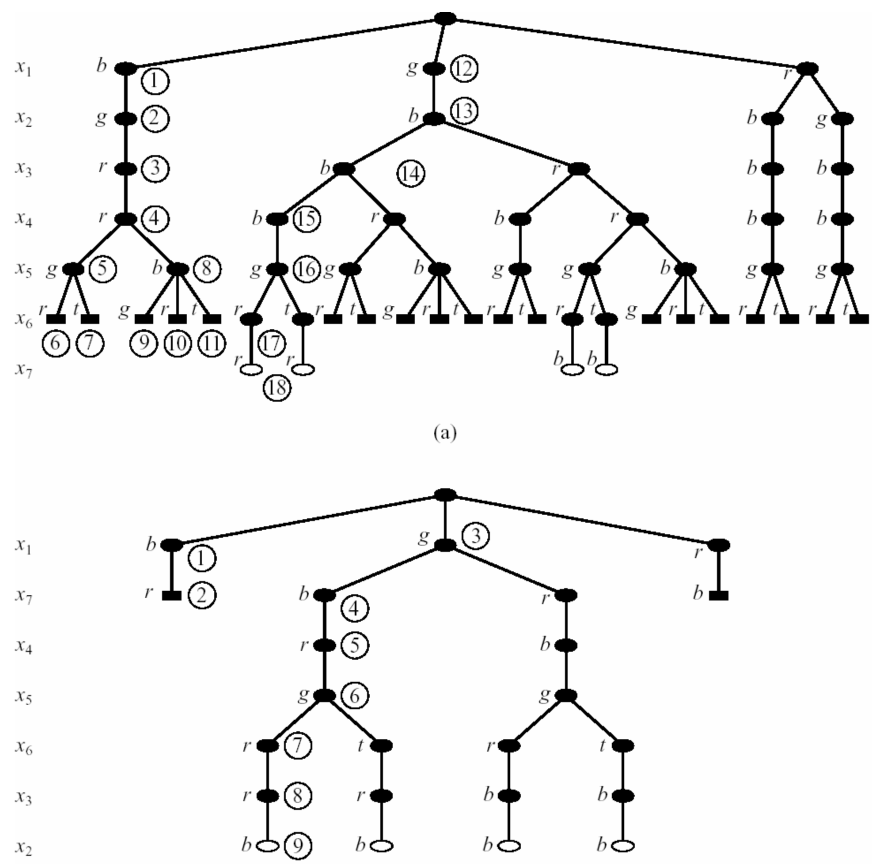
- After arc-consistency z=5 and l=5 are removed
- After path-consistency
  - R'_zx
  - R'_zy
  - R'_zl
  - R'_xy
  - R'_xl
  - R'_yl



Tighter networks yield smaller search spaces

Backtracking Search for a Solution

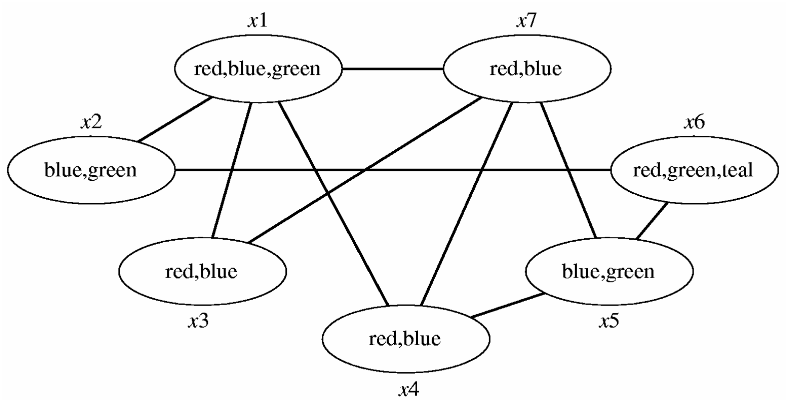


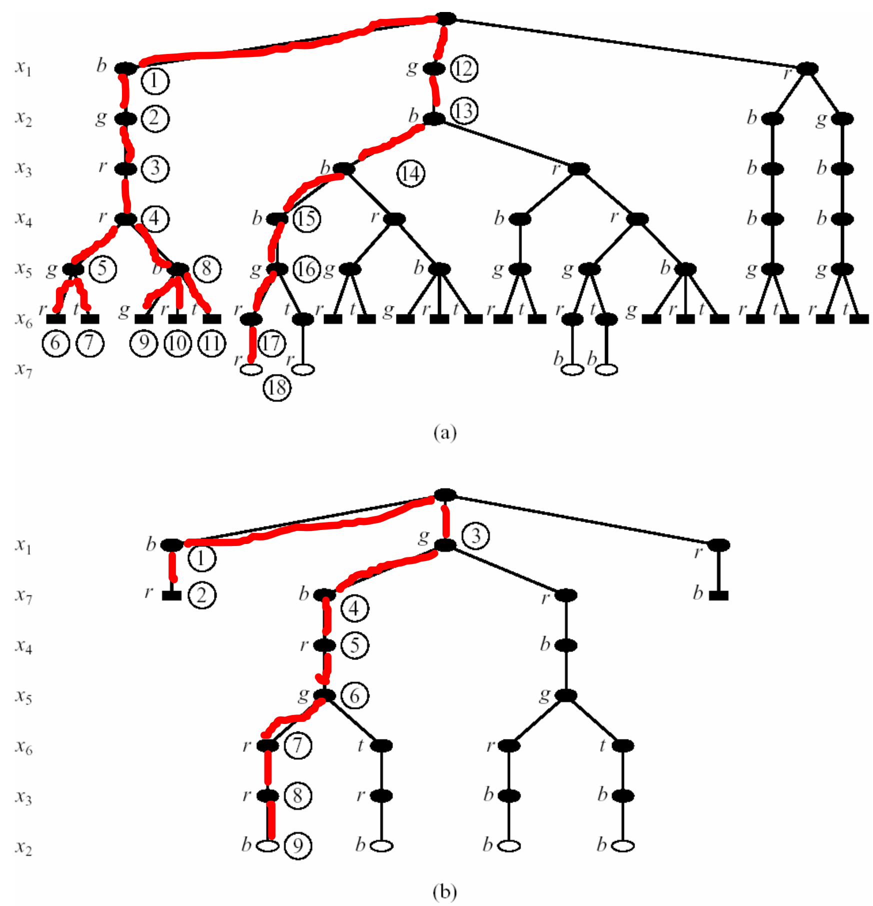


(a)

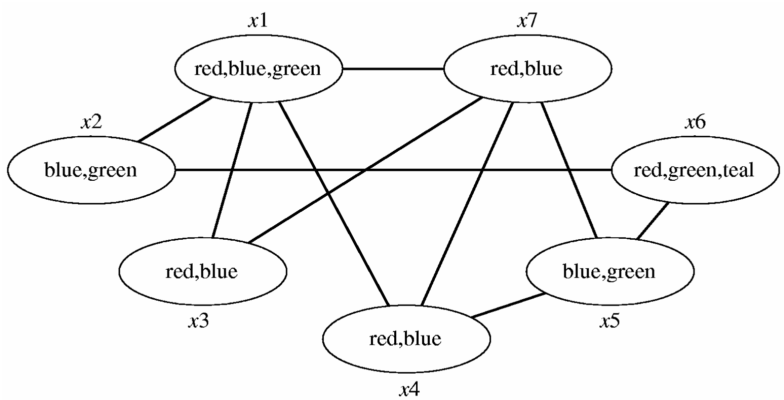
(b)

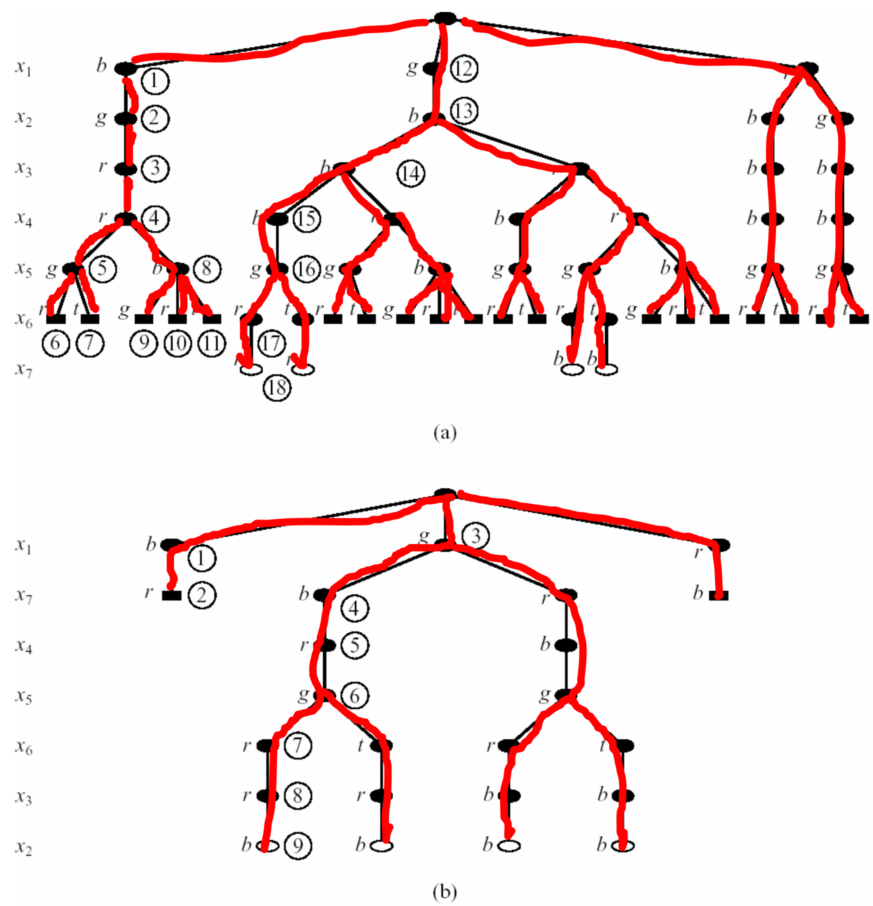
Backtracking Search for a Solution





Backtracking Search for All Solutions





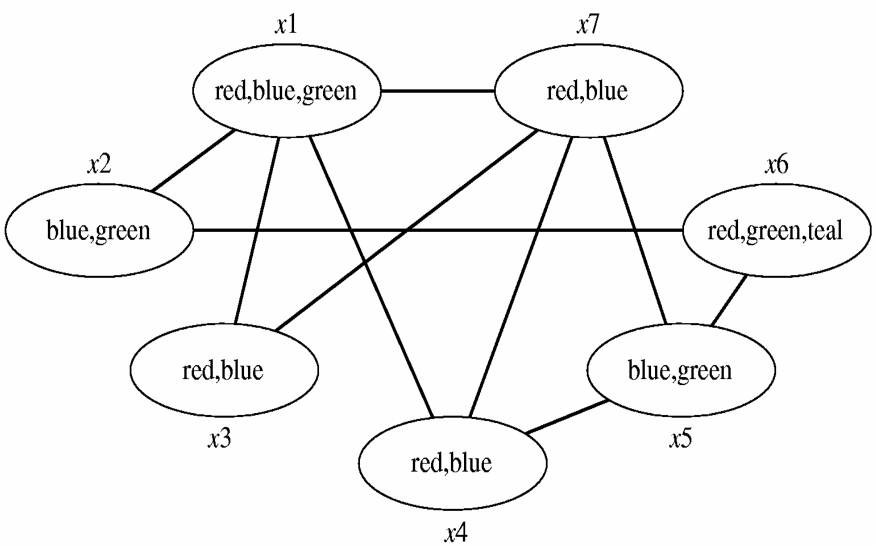
Improving Backtracking O(exp(n))

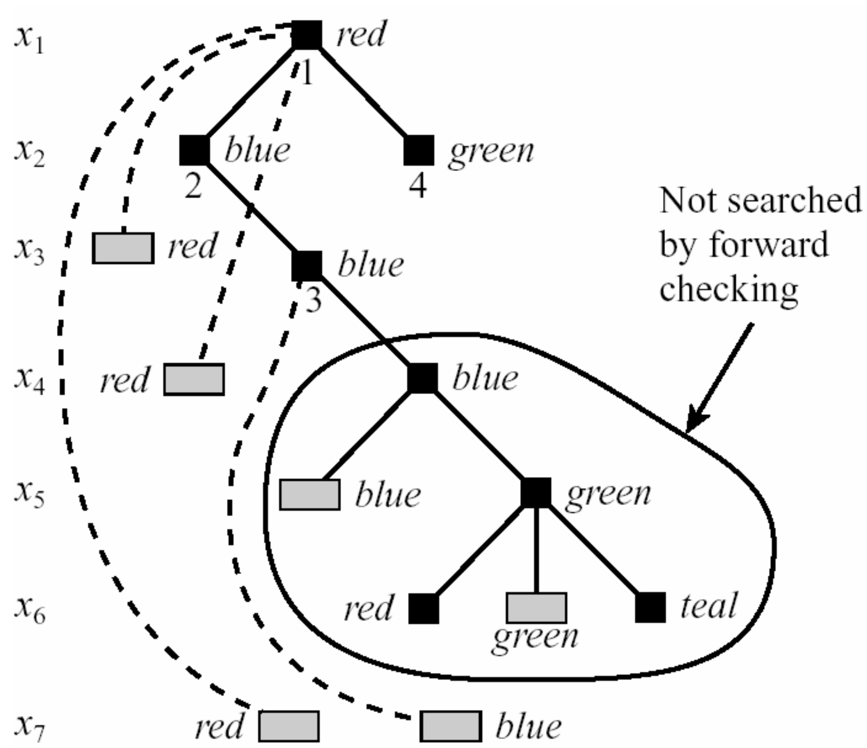
- Before search: (reducing the search space)
  - Arc-consistency, path-consistency, i-consistency
  - Variable ordering (fixed)
- During search:
  - **Look-ahead schemes**:
    - value ordering/pruning (*choose a least restricting value*),
    - variable ordering ***(Choose the most constraining variable)***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

Look-ahead:
Value Pruning Dynamically

- **Intuition:** apply constraint propagation at each node in the search tree and then choose value least likely to yield a dead-end.
- Forward-checking (FC)
  - (check each unassigned variable separately
- Maintaining arc-consistency (MAC)
  - (apply full arc-consistency)
- Full look-ahead
  - One pass of arc-consistency (AC-1)
- Partial look-ahead
  - directional-arc-consistency

Forward-checking on Graph-coloring





FW overhead: $O(ek^2)$

MAC overhead: $O(ek^3)$

Look-ahead:
Dynamic Value Ordering

Rank order the promise in non-rejected values

- Rank functions
  - MC (min conflict)
  - MD (min domain)
  - SC (expected solution counts)

- MC results (Frost and Dechter, 1996)
- SC – currently shows good performance using IJGP (Kask, Dechter and Gogate, 2004)

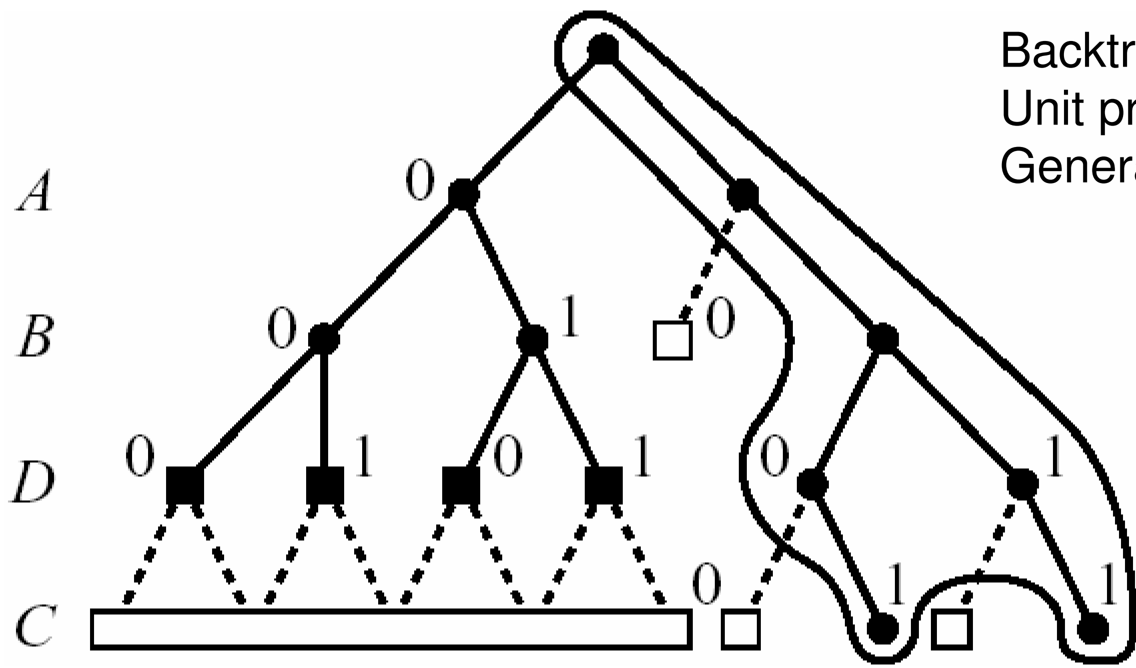
Look-ahead:
Dynamic Variable Ordering (DVO)

- Following constraint propagation, choose the most constrained variable
- **Intuition:** early discovery of dead-ends
- **Highly effective**: the single most important heuristic to cut down search space
- Most popular with FC
- Dynamic search rearrangement (Bitner and Reingold, 1975) (Purdon,1983)

Look-ahead for SAT: DPLL

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

(Davis-Putnam, Logeman and Laveland, 1962)



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

Only enclosed area will be explored with unit-propagation

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

Look-back:
Backjumping / Learning

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

Backjumping

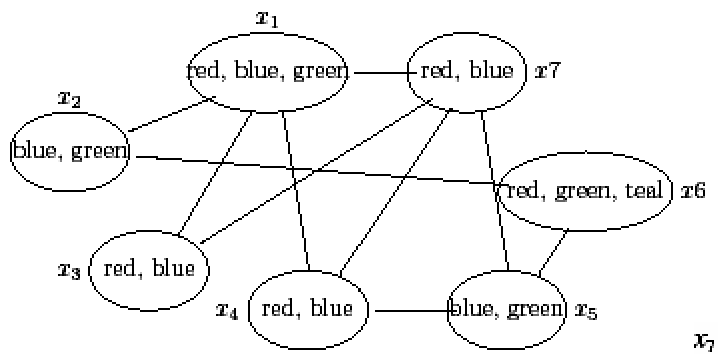
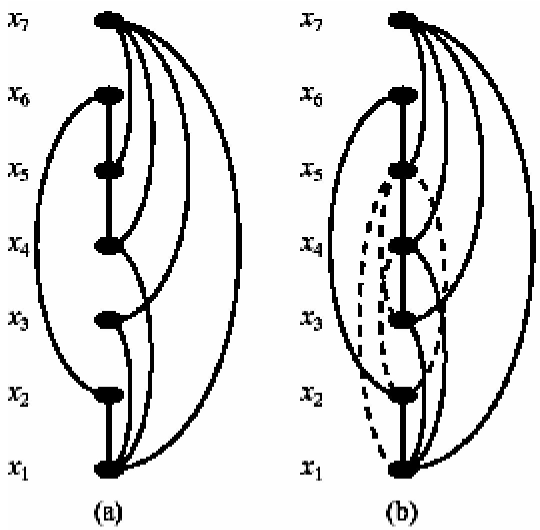


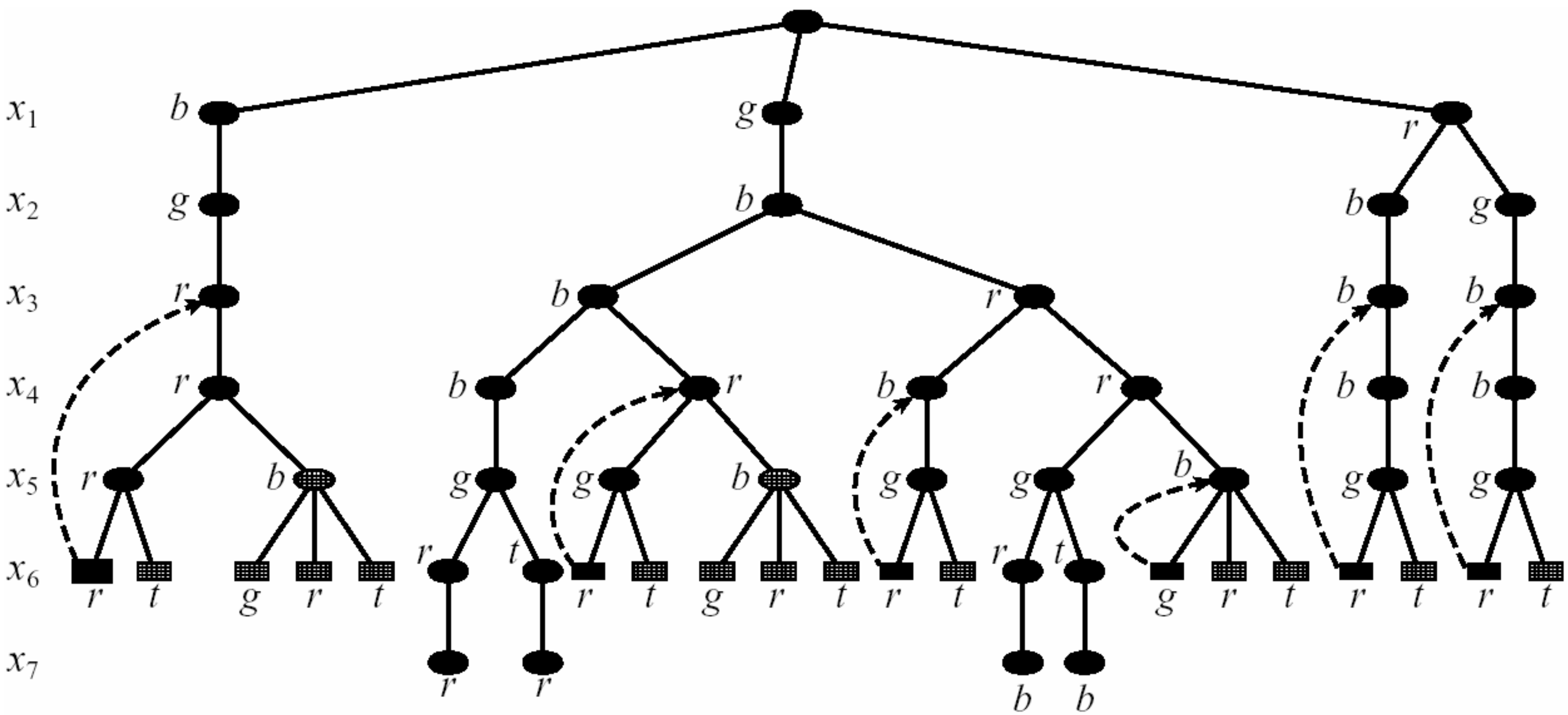
Figure 6.1: A modified coloring problem.

- (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**



Gaschnig jumps only at leaf-dead-ends

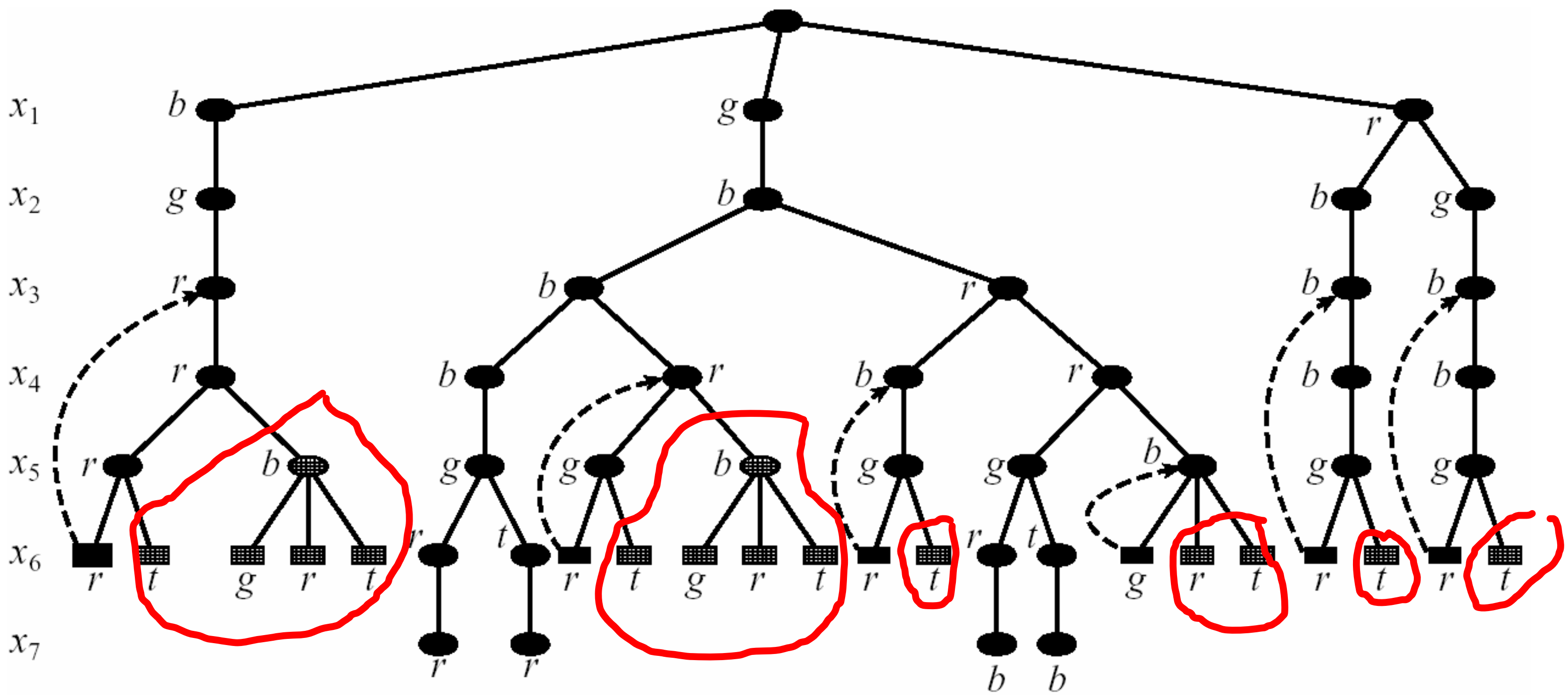
Internal dead-ends: dead-ends that are non-leaf



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

Gaschnig jumps only at leaf-dead-ends

Internal dead-ends: dead-ends that are non-leaf

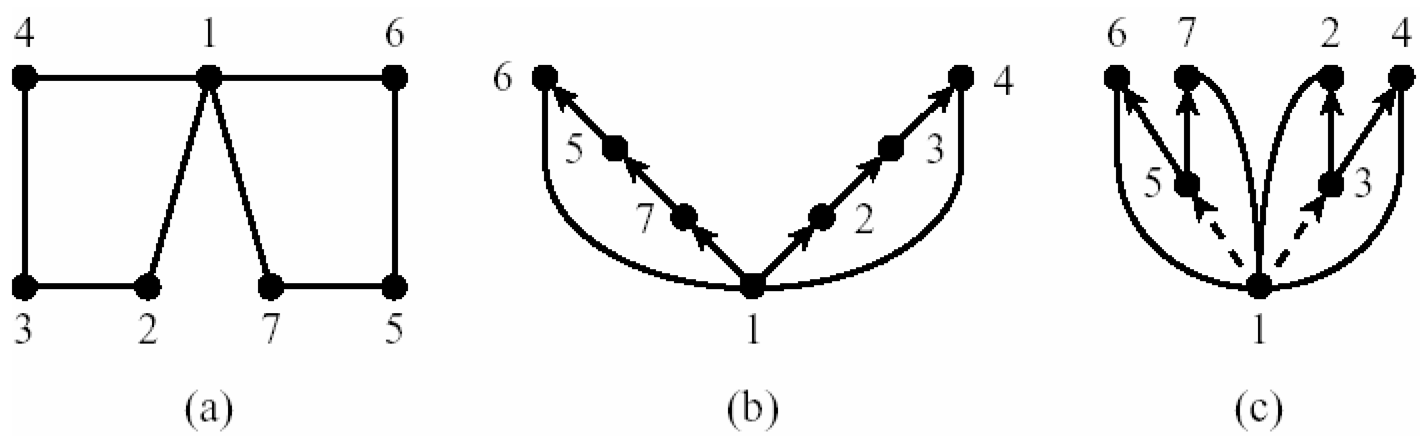


Example 6.3.1 In Figure 6.4, all of the backjumps illustrated lead to internal dead-ends, except for the jump back to $(\langle x_1, \mathit{green}\rangle, \langle x_2, \mathit{blue}\rangle, \langle x_3, \mathit{red}\rangle, \langle x_4, \mathit{blue}\rangle)$, because this is the only case where another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

Backjumping styles

- Jump at leaf only (Gaschnig 1977)
  - Context-based
- Graph-based (Dechter, 1990)
  - Jumps at leaf and internal dead-ends
- Conflict-directed (Prosser 1993)
  - Context-based, jumps at leaf and internal dead-ends

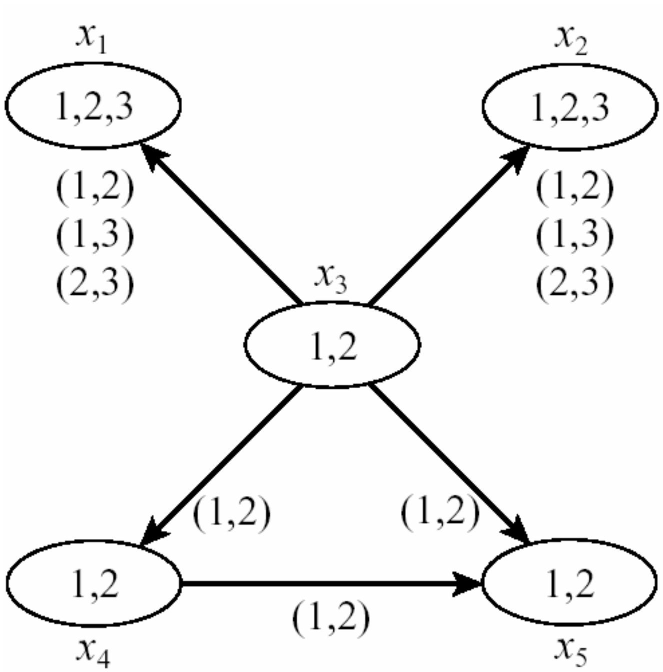
Backjumping on DFS Ordering



Simple: always jump back to parent in dfs tree
Complexity: exp(m)
m is depth of dfs/pseudo tree

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

Learning example

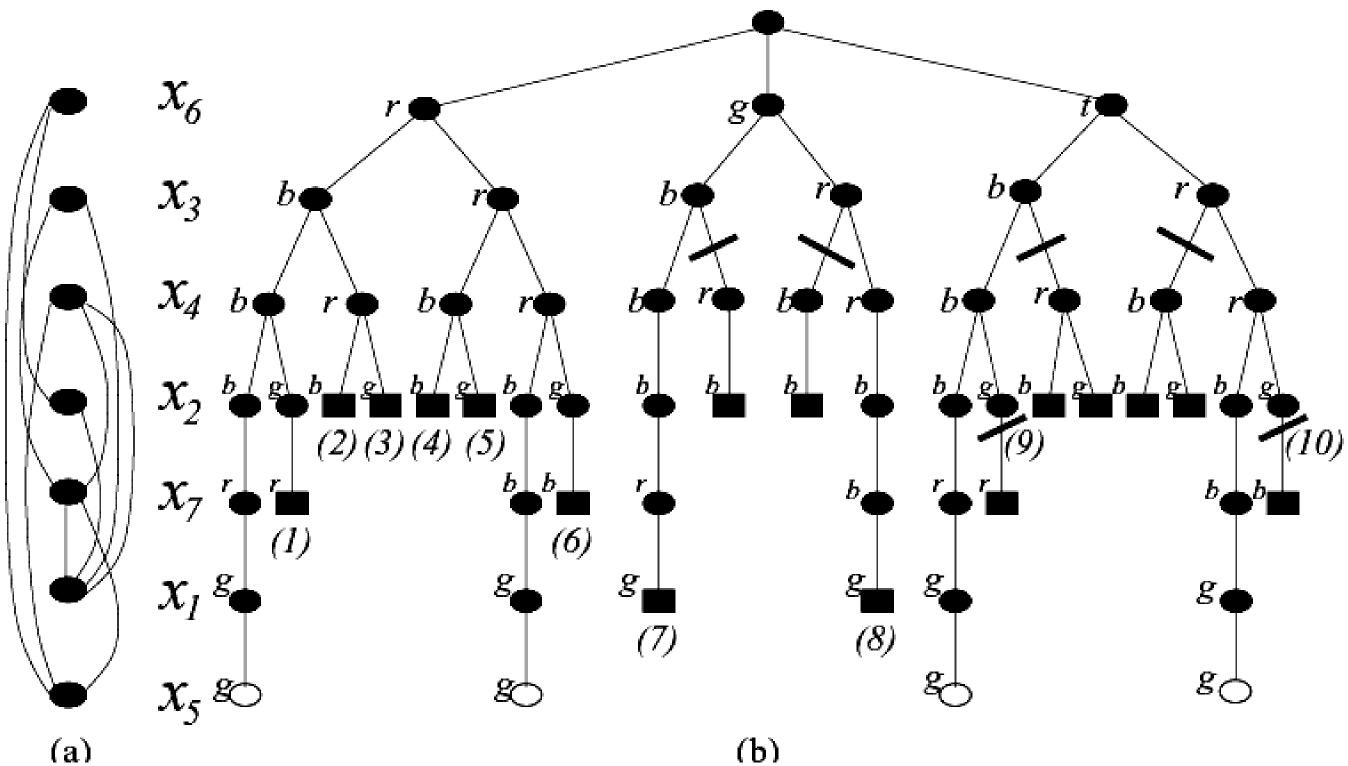


Figure 6.9: The search space explicated by backtracking on the CSP from Figure 6.1, using the variable ordering $(x_6, x_3, x_4, x_2, x_7, x_1, x_5)$ and the value ordering (*blue*, *red*, *green*, *teal*). Part (a) shows the ordered constraint graph, part (b) illustrates the search space. The cut lines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

Learning Issues

- Learning styles
  - Graph-based or context-based
  - Deep vs. shallow
  - i-bounded, scope-bounded
  - Relevance-based
- Non-systematic randomized learning
- Implies time and space overhead
- All these can be applied to SAT

Complexity of Backtrack-Learning

- The complexity of learning along d is time and space exponential in w*(d):

The number of dead-ends is bounded by $O(nk^{w^*(d)})$

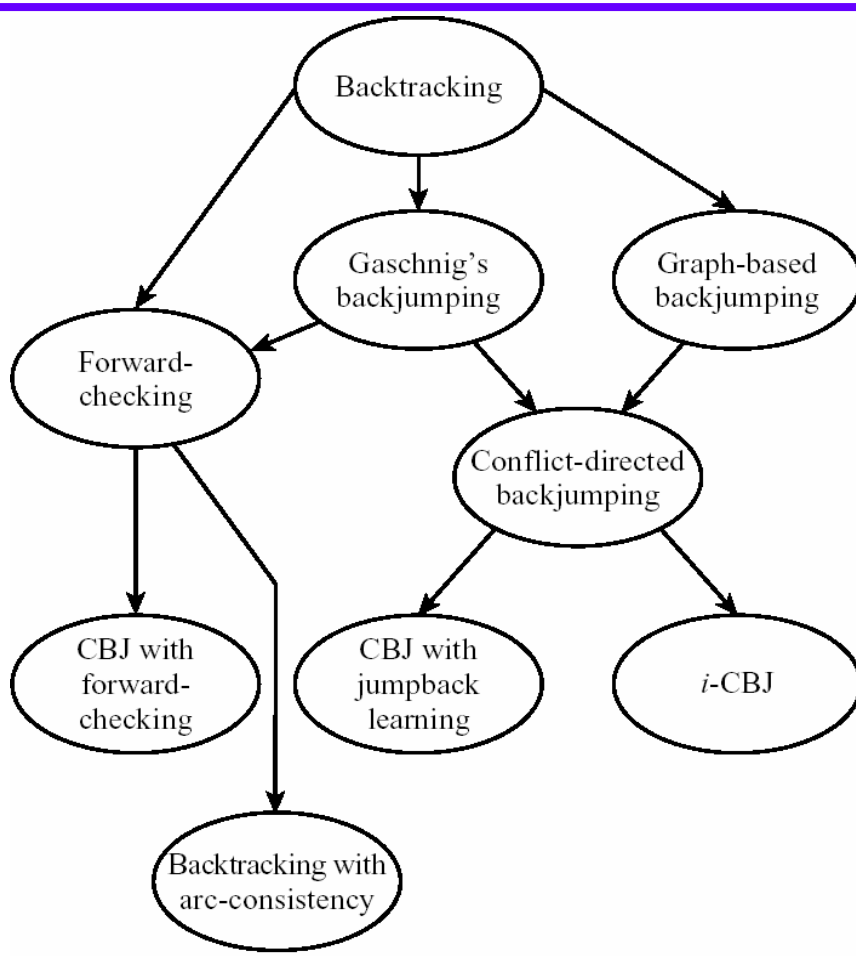
Number of constraint tests per dead-end are $O(2^{w^*(d)})$

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

Time complexity is $O(n^2(2k)^{w^*(d)})$

Relationships Between Various Backtracking Algorithms

Look-back and look-ahead can be integrated!



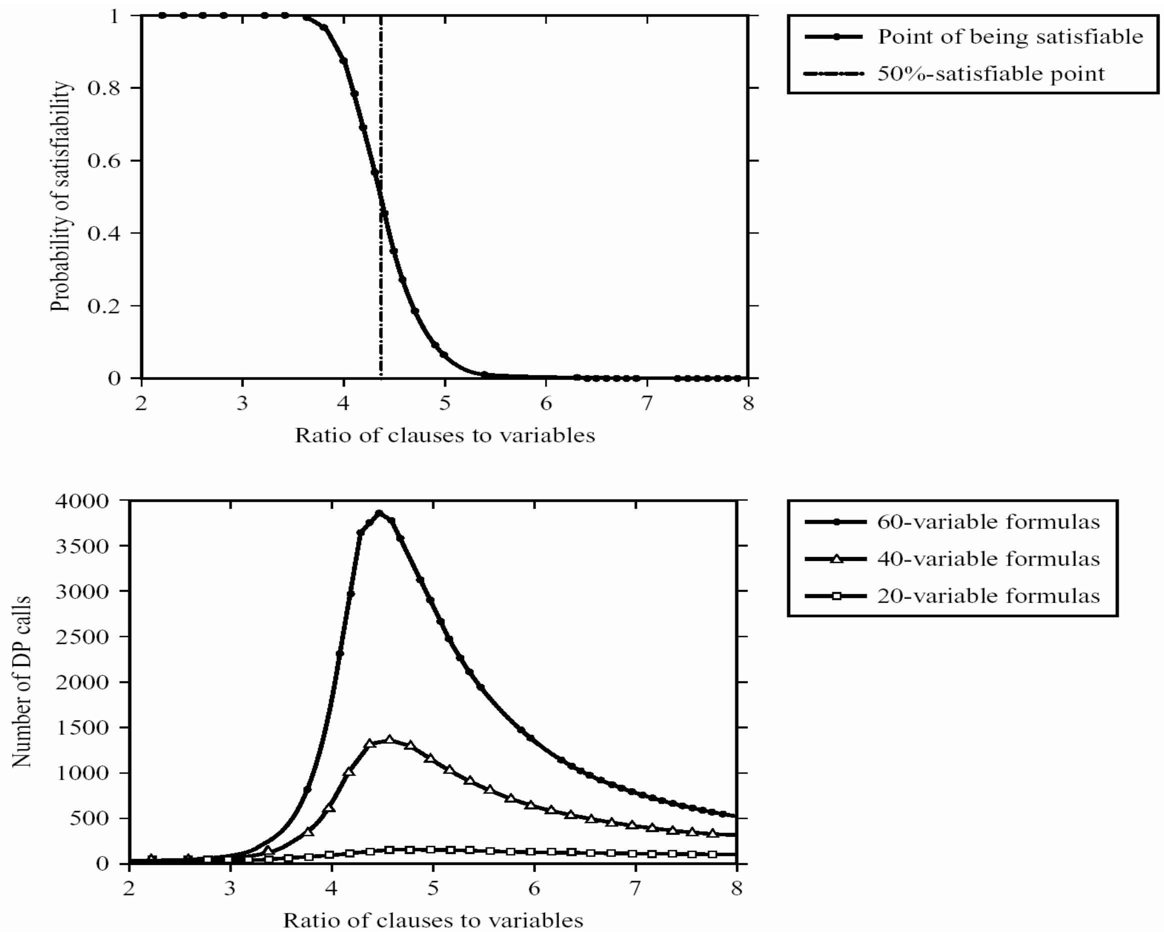
Exact Techniques: Complexity

	Search	Variable Elimination
Worst-case time	$O(\exp(n))$ $O(\exp(dfs-depth))$	$O(n\exp(w^*))$ $w^* \leq n$
Average time	Better than worst-case	Same as worst-case
Space	$O(n)$	$O(n\exp(w^*))$ $w^* \leq n$
Output	One solution	Knowledge compilation

Empirical Comparison

- Benchmark instances
- Random problems:
  - Generating fixed length random CSPs (N,K,T,C) or fixed length random k-sat (n, m) uniformly at random
- Application-based random problems

The Phase Transition (m/n) Important for empirical evaluation



Some Empirical Evaluation

- Sets 1-3 reports average over 2000 instances of random CSPs from 50% hardness. Set 1: 200 variables, set 2: 300, Set 3: 350. All had 3 values.:
- DIMACS problems

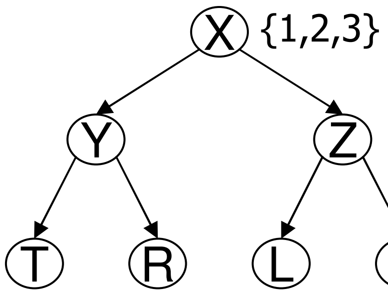
Algorithm	Set 1		Set 2		Set 3		ssa 038		ssa 158	
FC	207	68.5		-		-	46	14.5	52	20.0
FC+AC	40	55.4	1	0.6	1	0.4	4	3.5	18	8.2
FCr-CBJ	189	69.2	222	119.3	182	140.8	40	12.2	26	10.7
FC-CBJ+LVO	167	73.8	132	86.8	119	111.8	32	11.0	8	4.5
FC-CBJ+LRN	186	63.4	32	15.6	1	0.5	23	5.5	19	8.6
FC-CBJ+LRN+LVO	160	74.0	26	14.0	1	3.8	16	3.8	13	7.1

Figure 6.16: Empirical comparison of six selected CSP algorithms. See text for explanation. In each column of numbers, the first number indicates the number of nodes in the search tree, rounded to the nearest thousand, and final 000 omitted; the second number is CPU seconds.

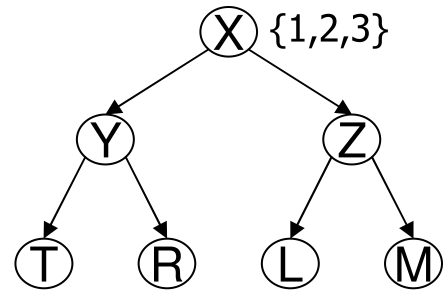
Road Map

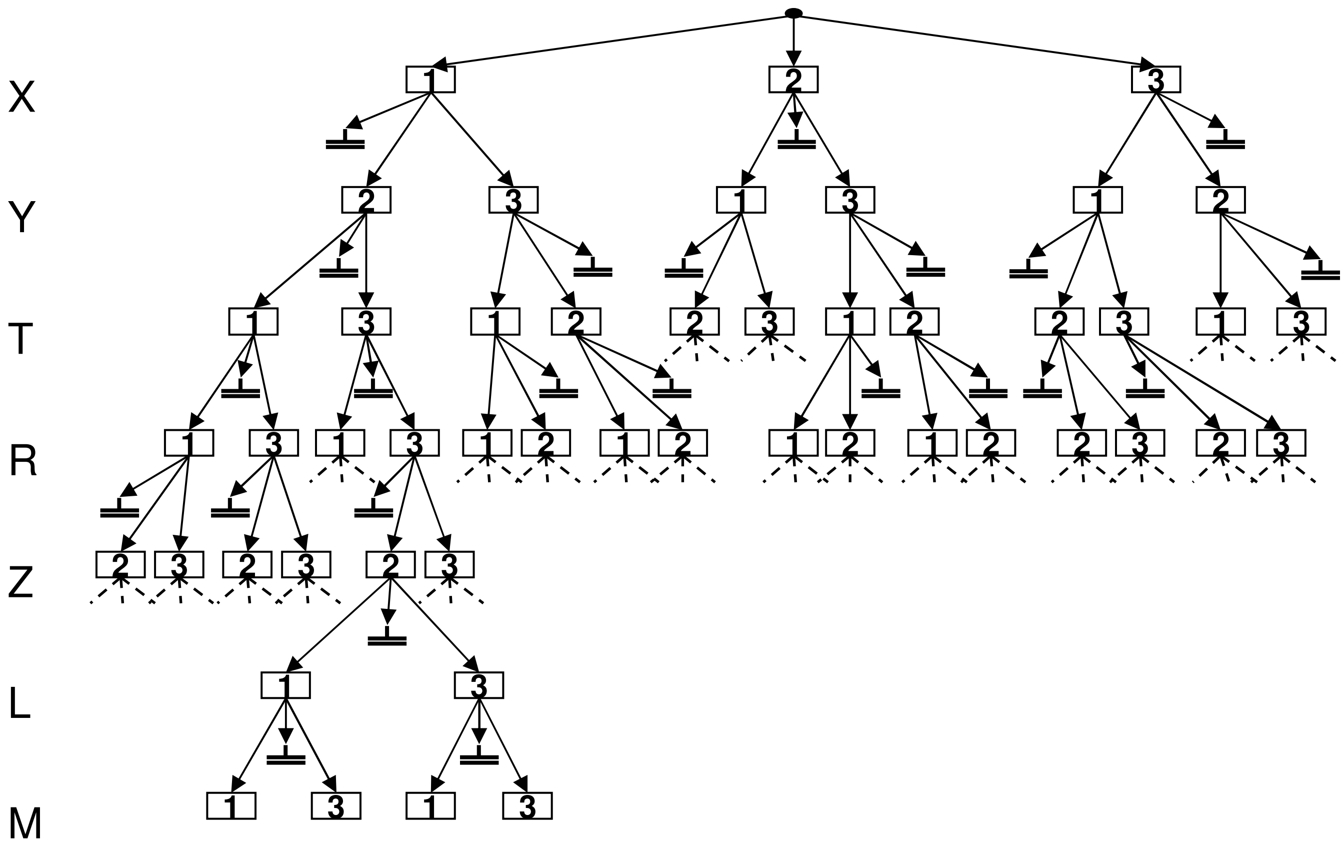
- Introduction
- Inference
- **Search:**
  - **Look-ahead schemes**
  - **Look-back schemes**
  - **The alternative AND/OR search space**
- Hybrids of search and inference
- Relationships to Belief networks

Search Space for CSP

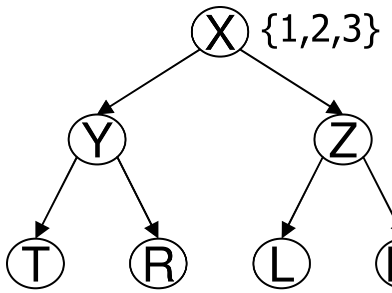


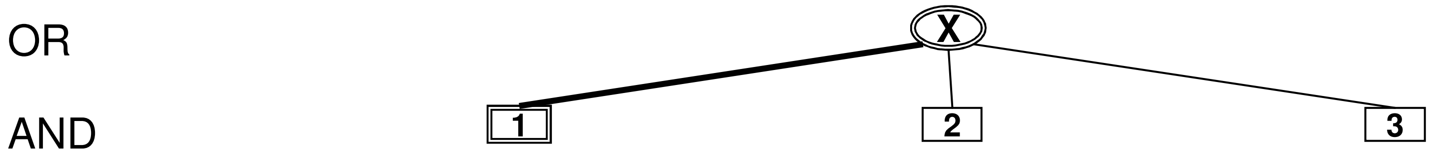
OR Search Space



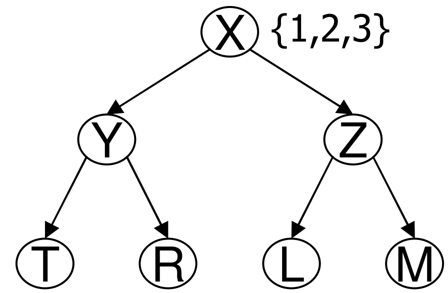


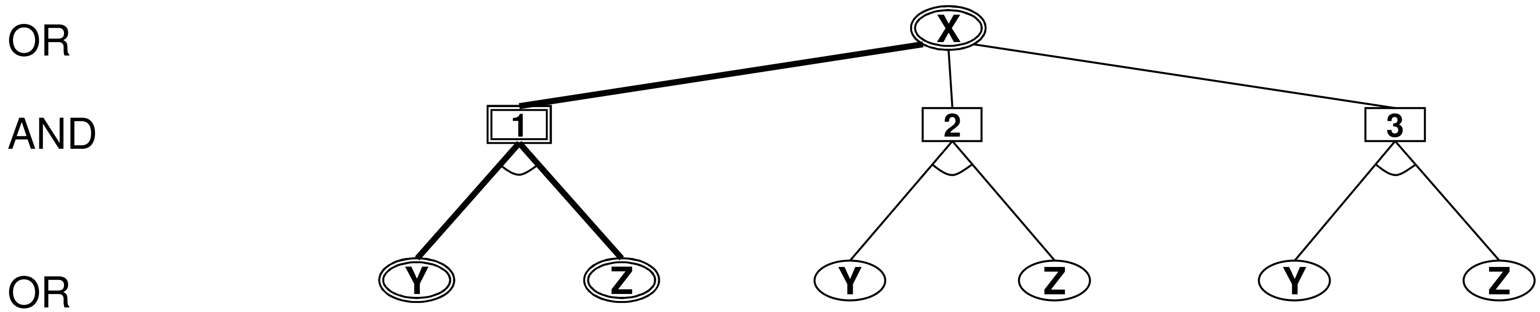
AND/OR Search Space



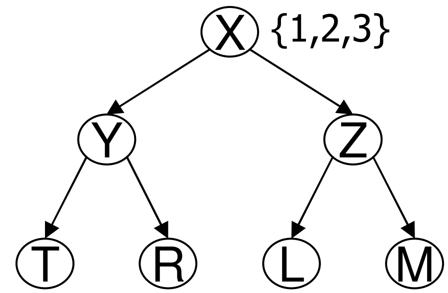


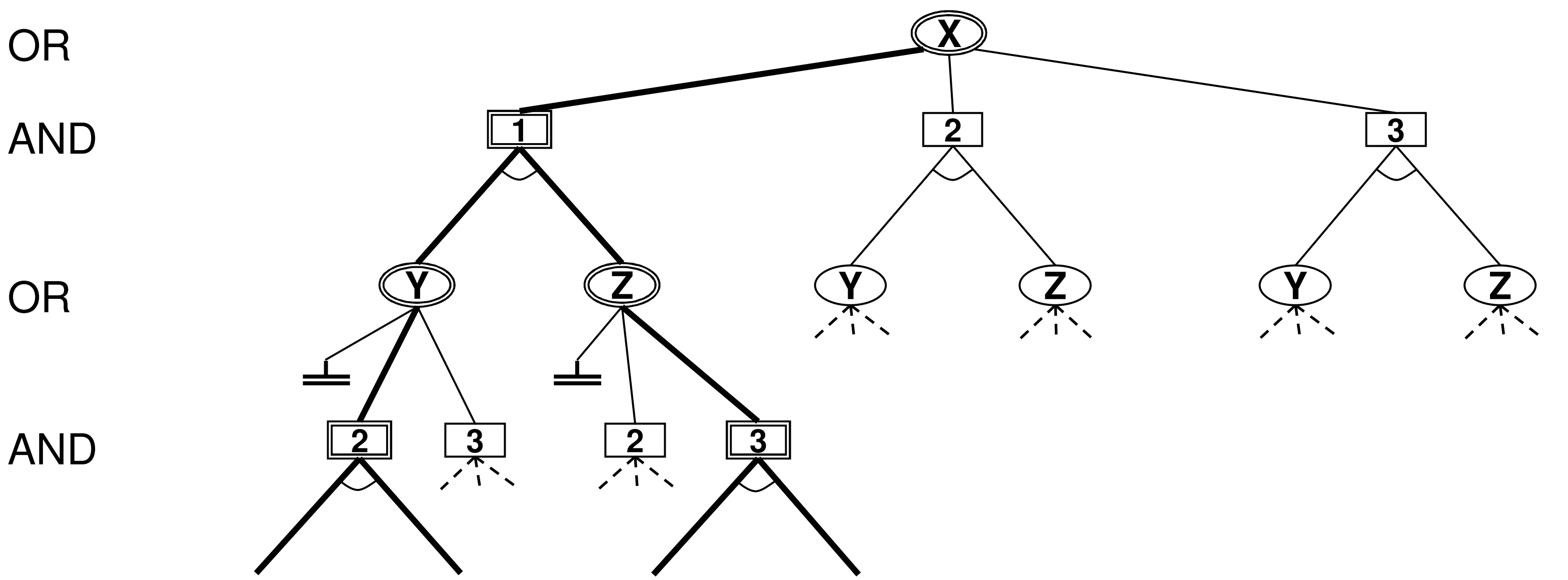
AND/OR Search Space



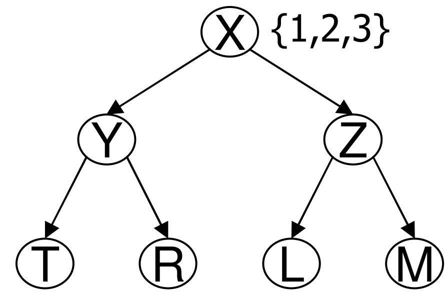


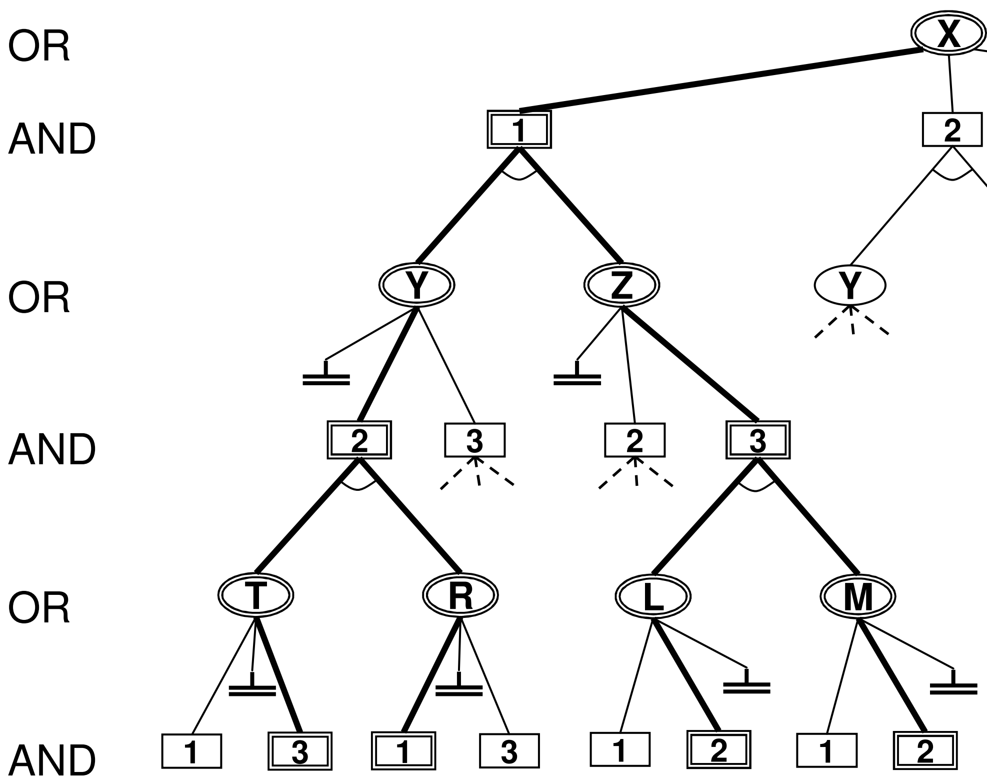
AND/OR Search Space



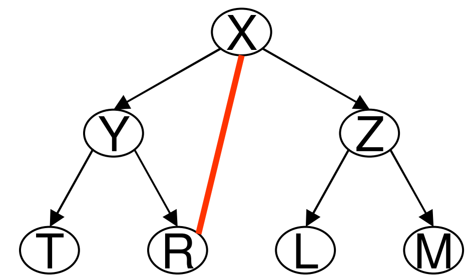


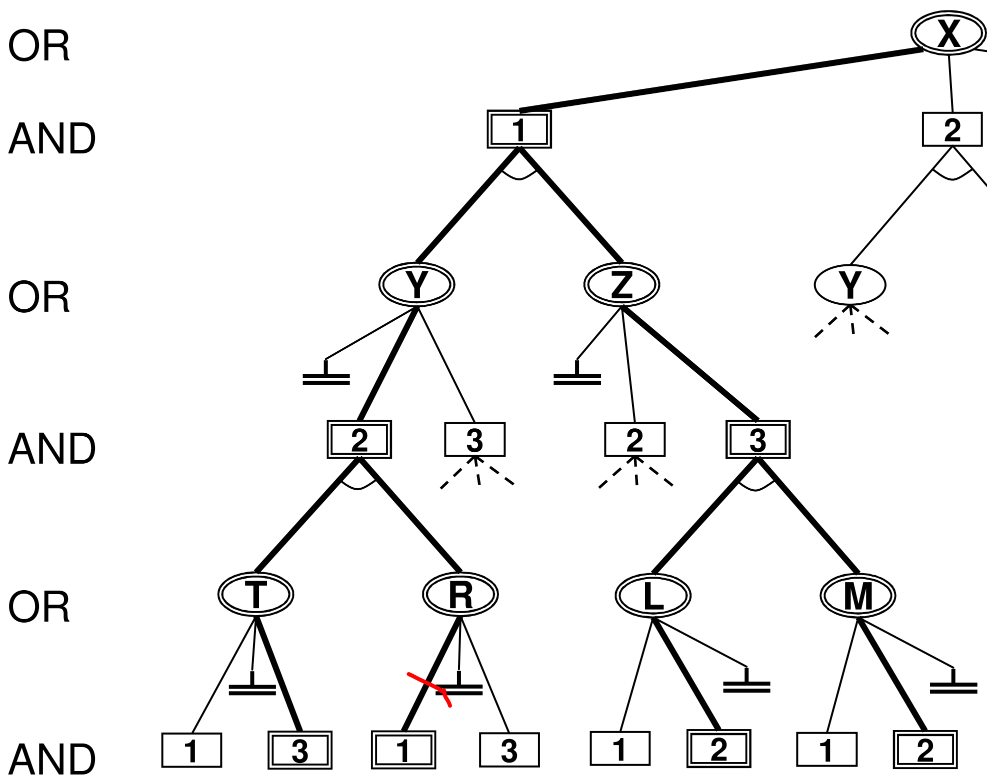
AND/OR Search Space



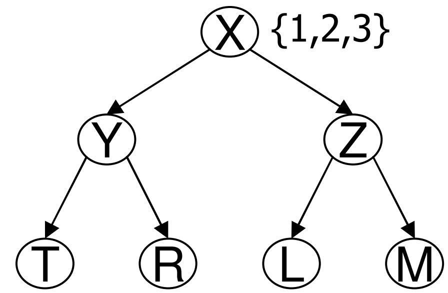


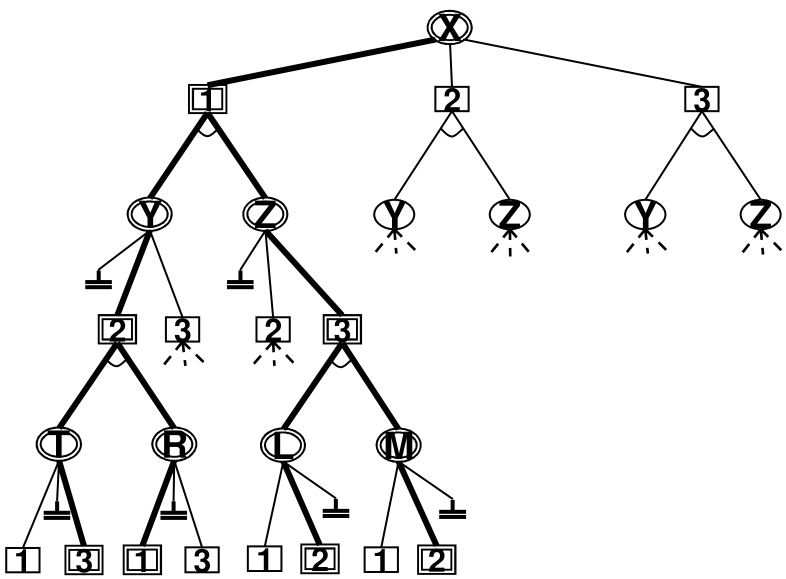
AND/OR Search Space

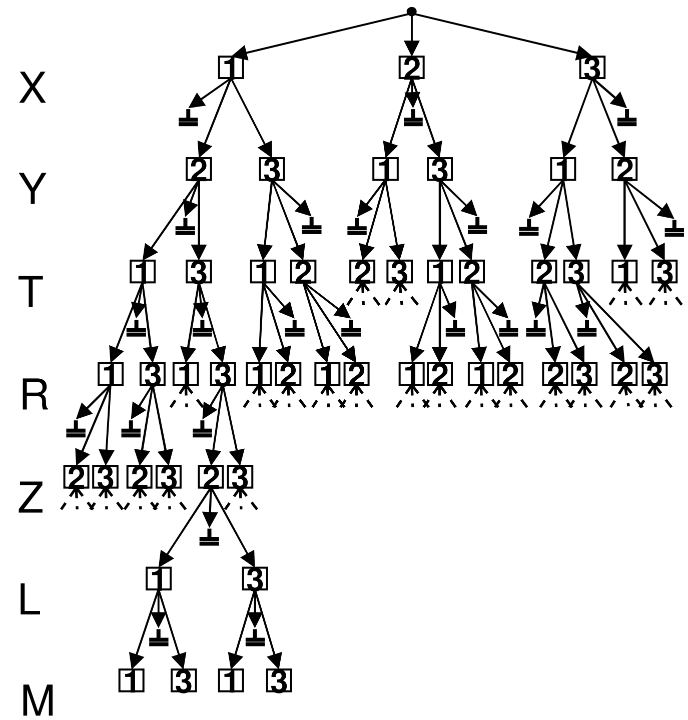




OR vs. AND/OR







OR space vs. AND/OR space

width	height	OR space			AND/OR space		
		Time (sec.)	Nodes	Backtracks	Time (sec.)	AND nodes	OR nodes
5	10	3.154	2,097,150	1,048,575	0.03	10,494	5,247
4	9	3.135	2,097,150	1,048,575	0.01	5,102	2,551
5	10	3.124	2,097,150	1,048,575	0.03	8,926	4,463
4	10	3.125	2,097,150	1,048,575	0.02	7,806	3,903
5	13	3.104	2,097,150	1,048,575	0.1	36,510	18,255

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

AND/OR Search-tree Complexity

- Theorem: The size of the AND/OR search tree is exponential in m (depth of legal tree) while the size of its OR search tree is exponential in n (number of variables)
- Given a tree-decomposition having w*, there is a legal tree T whose depth, satisfies:
- Conclusion: A graphical model has an AND/OR search tree bounded by O(exp(w* log n)).

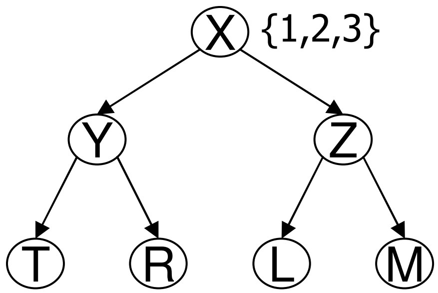
$$m \leq w^* \log n$$

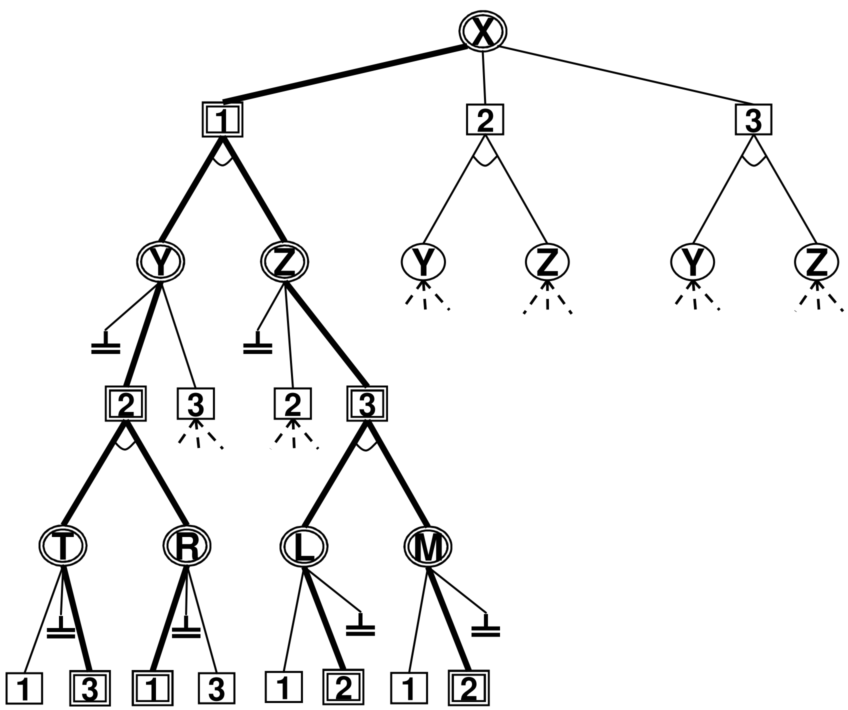
- Any DFS traversal of the AND/OR search tree is linear space and exponential time in m. AND-OR-counting,
  - AND-OR-solution
  - AND-OR belief
  - And-OR partition
  - AND-OR-MPE

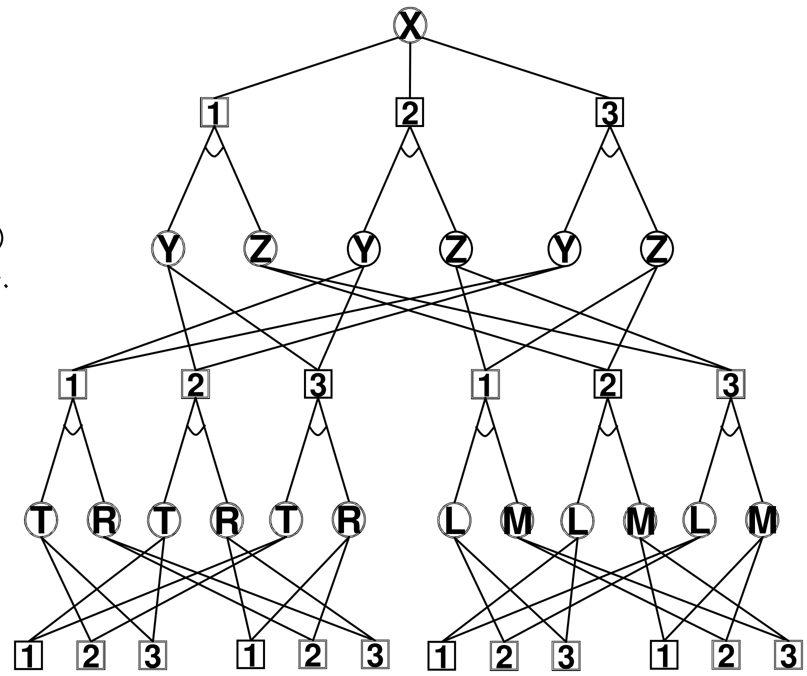
The Minimal AND/OR Search Graph

- Any two nodes <X,v> that root the same subtree in S_T can be merged.
- **Minimal AND/OR search graph**: The closure under merge of the AND/OR search tree is the unique, sound and complete **minimal** AND/OR search graph relative to T.
- Minimal AND/OR graph is related to tree-OBDD

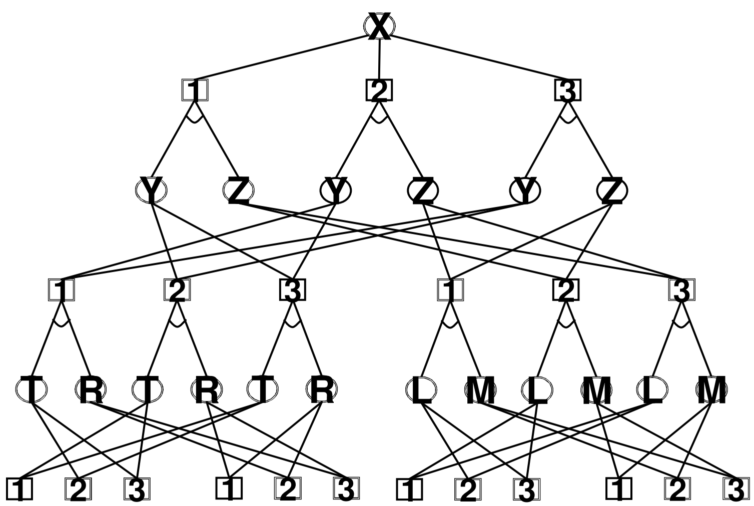
AND/OR tree vs Minimal AND/OR graph

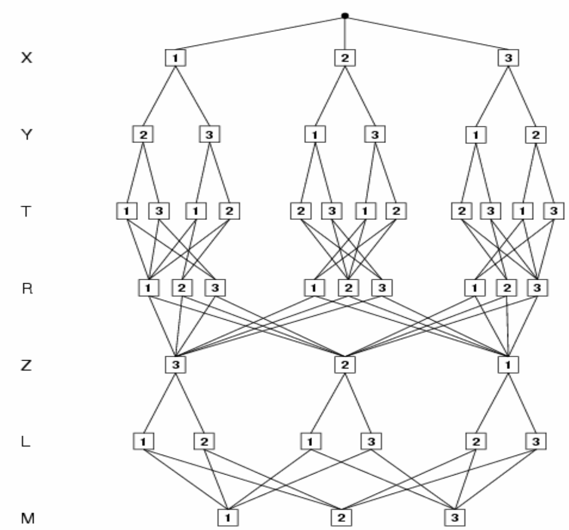






Minimal OR vs. Minimal AND/OR





- For a binary balanced tree-model of depth **r** , the tree-width is 1 while the path-width is **r**.
- For finding a solution minimal OR is as good as minimal AND/OR
- For counting, belief updating, optimization, the difference matters.

#CSP N40, K3, C50, P3, 20 inst, w*=13, h=20
#nodes / #deadends

	tightness	10\%		20\%		30\%		40\%		50\%		60\%	
		\# n	\# d	\# n	\# d	\# n	\# d	\# n	\# d	\# n	\# d	\# n	\# d
i=0	**A/O FC**	78	159	265	533	999	1994	4735	9229	60163	101135	1601674	1711947
	OR FC	78	159	265	533	1000	2003	4947	9897	273547	407350	384120807	324545908
i=3	**A/O FC**	78	159	265	533	986	1990	4525	9166	46763	98413	689154	1625075
	OR FC	78	159	265	533	1000	2003	4947	9897	224739	399210	228667363	287701079
i=6	**A/O FC**	78	159	265	533	981	1971	4467	8991	41876	85583	487320	917612
	OR FC	78	159	265	533	1000	2003	4947	9897	185422	329754	141610990	208159068
i=9	**A/O FC**	78	159	265	533	981	1958	4451	8866	37314	70337	362024	580325
	OR FC	78	159	265	533	1000	2003	4947	9897	147329	270446	102316417	135655353
i=12	**A/O FC**	78	159	265	533	981	1955	4422	8560	31669	54667	198505	197513
	OR FC	78	159	265	533	999	1994	4796	9358	116154	198177	53965244	56757351
i=13	**A/O FC**	78	159	265	533	981	1955	4415	8533	30610	50228	**170827**	**181157**
	OR FC	78	159	265	533	999	1994	4761	9283	99923	176630	**16210028**	**20018823**

Complexity of AND/OR-graph Search

- **Theorem:** If you search the AND/OR graph, complexity is time and space exponential in the induced width.

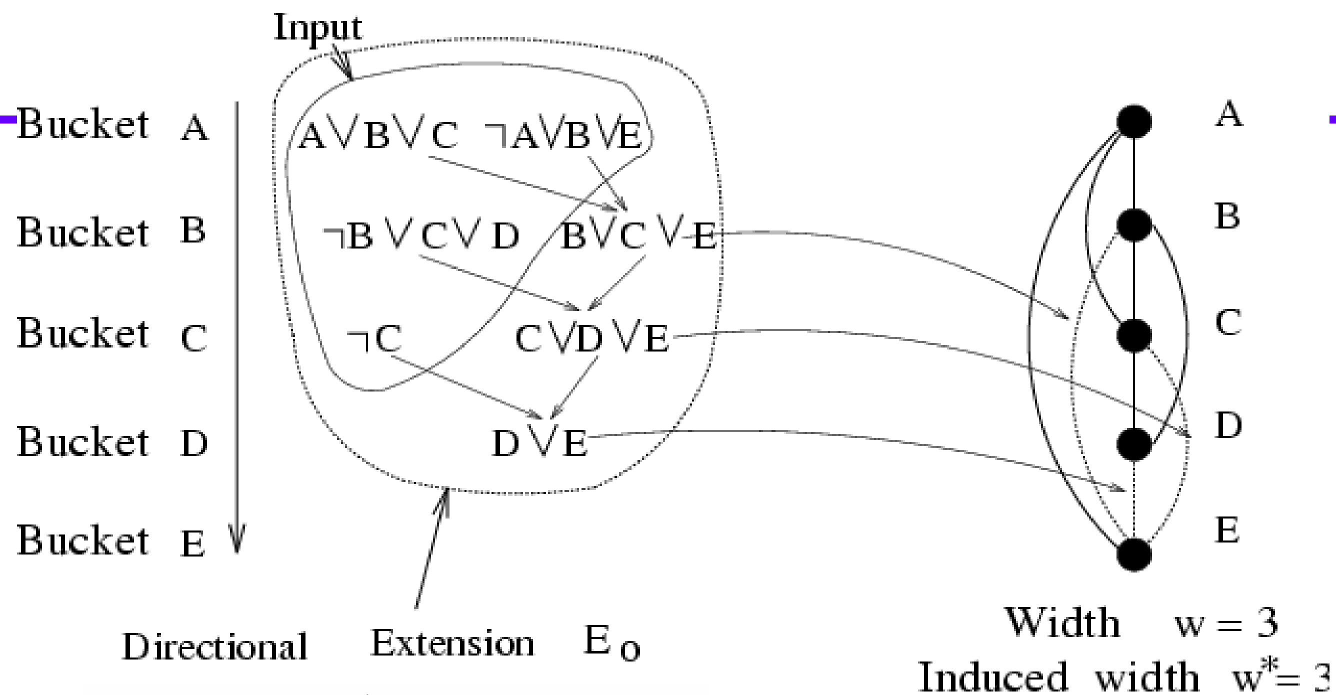
- If applied to the OR graph complexity is exponential in the path-width.

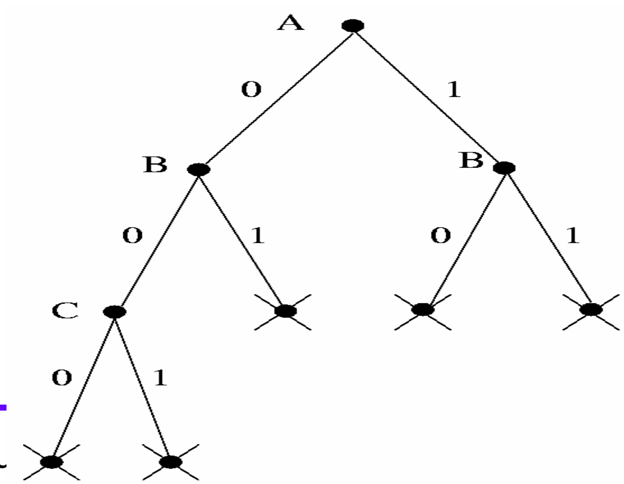
Road Map

- Introduction
- Inference
- Search
- **Hybrids of search and inference**
- Relationships to Belief networks

Satisfiability: Inference vs search

$\psi = (A \vee B \vee C) \wedge (\neg A \vee B \vee E) \wedge (\neg B \vee C \vee D) \wedge (\neg C)$





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

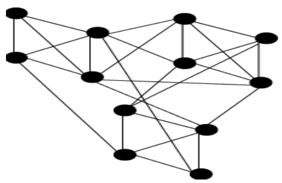
DR time and space : $O(n \exp(w^*))$

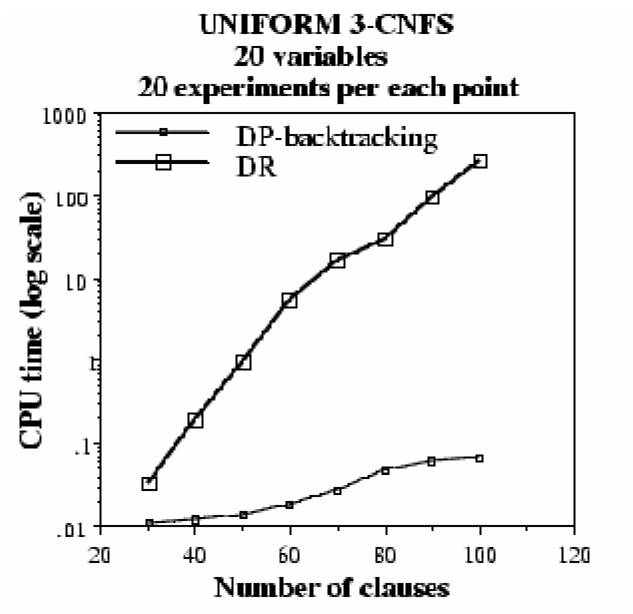
Search **= O(exp(n))**

Search = exp(dfs-height)

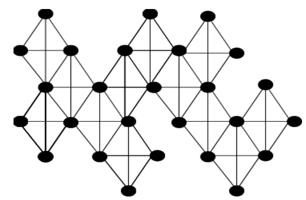
DR versus DPLL:
Complementary Properties

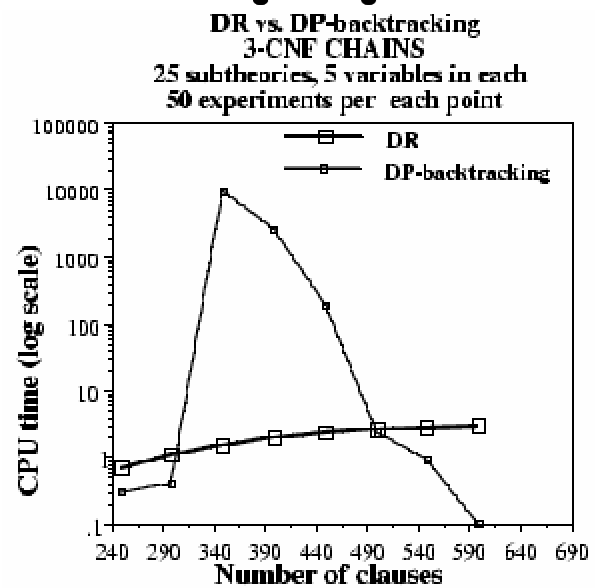
Uniform random 3-CNFs
(large induced width)





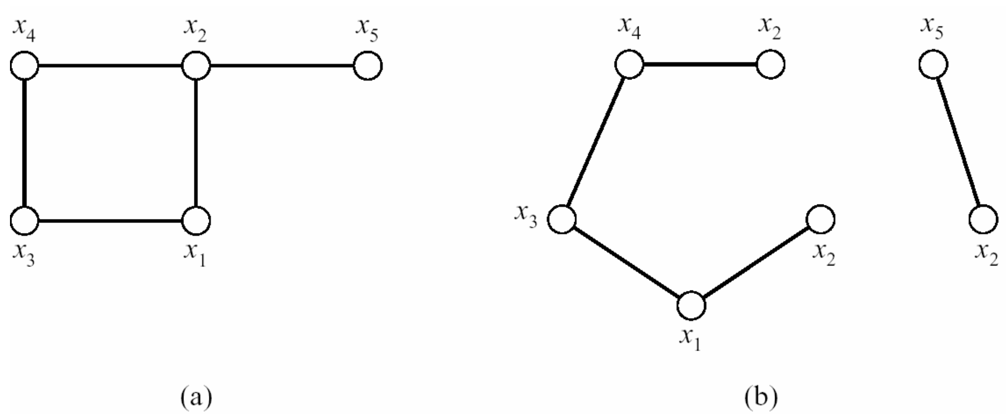
(k,m)-tree 3-CNFs
(bounded induced width)



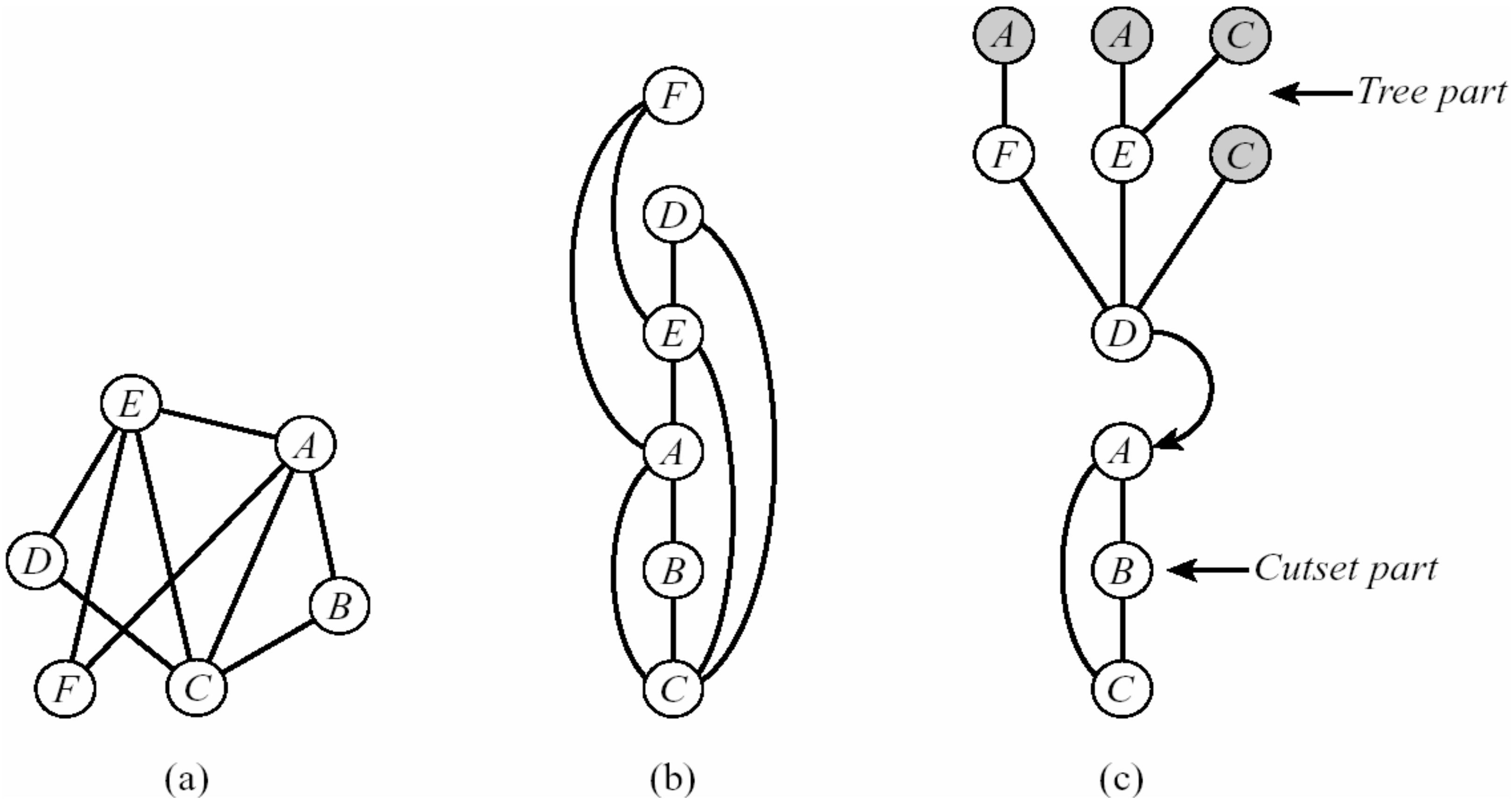


The Cycle-Cutset Effect

- A cycle-cutset is a subset of nodes in an undirected graph whose removal results in a graph with no cycles
- An instantiated variable cuts flow of information cycles
- If a cycle-cutset is instantiated the remaining problem is a tree and can be solved efficiently

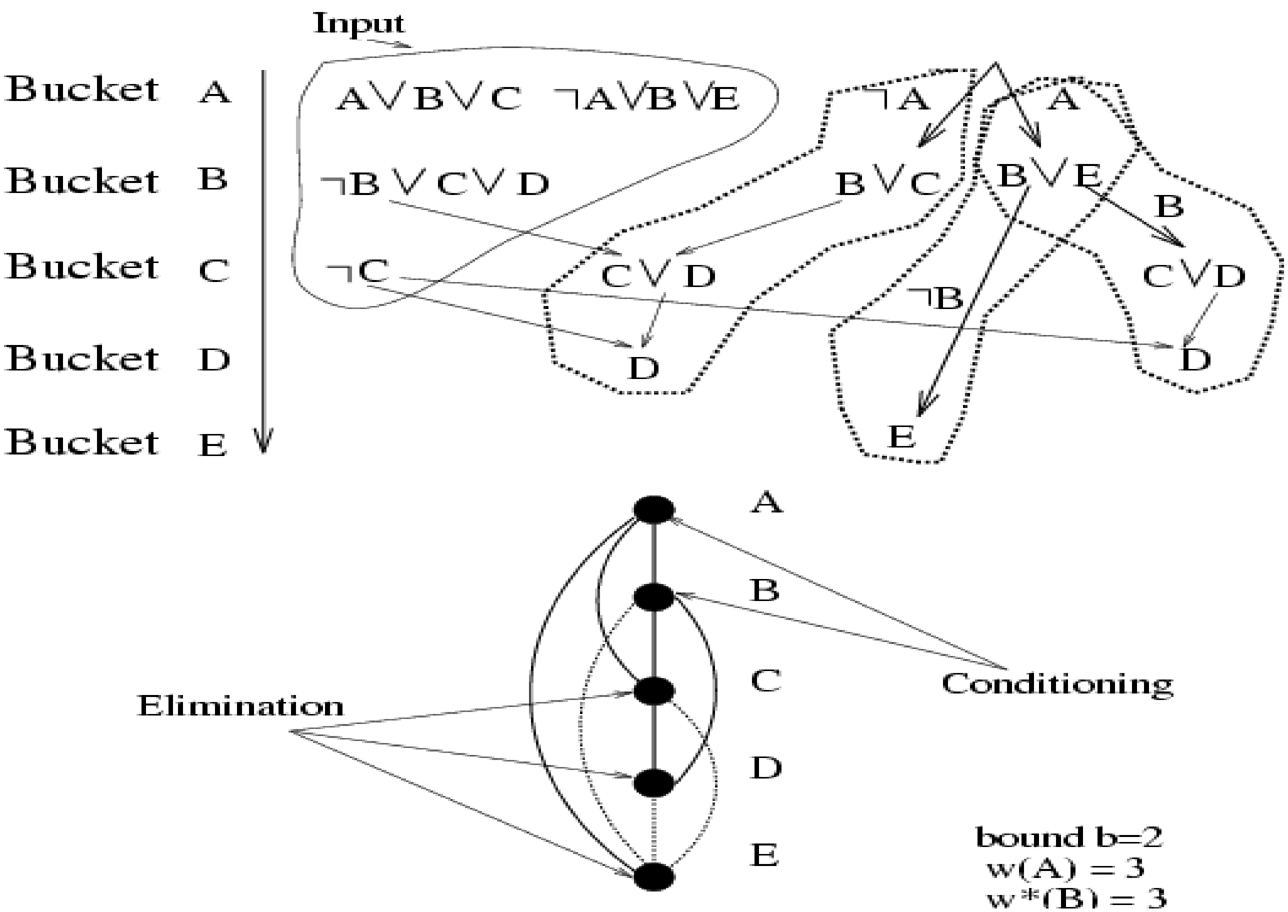


Example of the cycle-cutset scheme



W-cutset Example

(~C v E)(A v B v C v D)(~A v B v E v D)(B v C v D)



Time-space tradeoff

Theorem:

The w-cutset scheme yields space complexity exp(w) and time complexity exp(w+c_w), where c_w is the size of the w-cutset.

As w decreases, c_w increases.

The cycle-cutset decomposition is linear space and Has time complexity of $O((n-c)k^{(c+2)})$

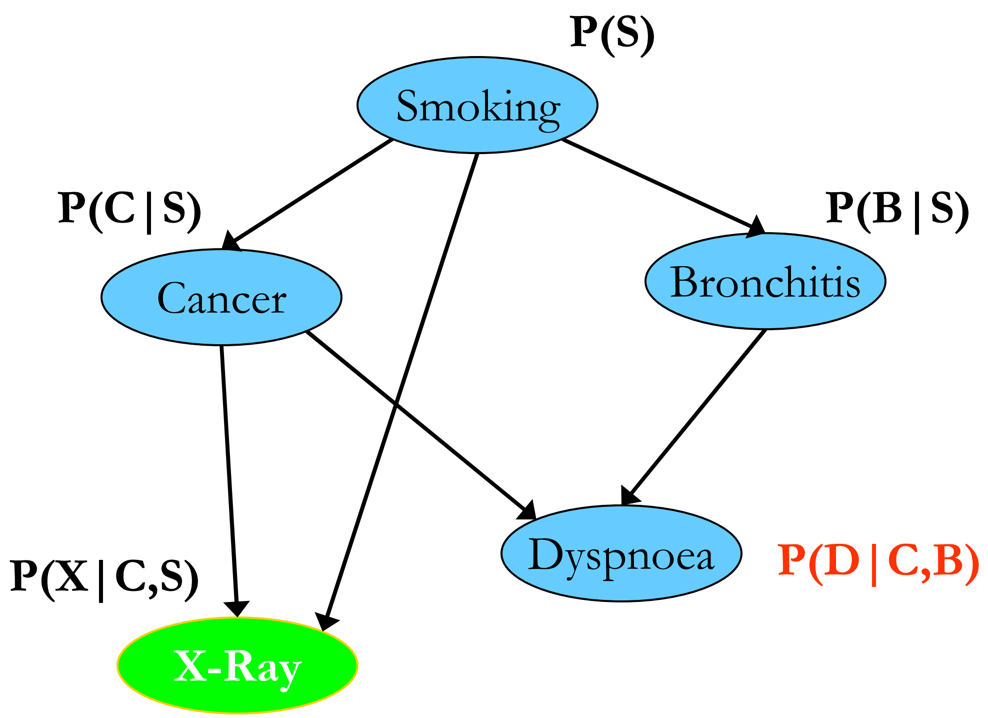
Exact Techniques: Complexity

	Search	Variable Elimination
Worst-case time	$O(\exp(n))$ $O(\exp(cutset))$ $O(\exp(dfs-depth))$	$O(n\exp(w^*))$ $w^* \leq n$
Average time	Better than worst-case	Same as worst-case
Space	$O(n)$	$O(n\exp(w^*))$ $w^* \leq n$
Output	One solution	Knowledge compilation

Road Map

- Introduction
- Inference:
- Search
- Hybrids of search and inference
- **Relationships to Belief networks**

Probabilistic Networks



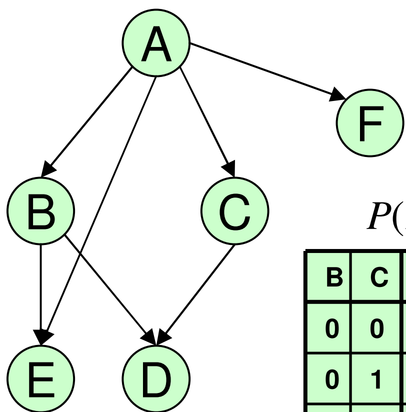
P(D|C,B)

C	B	D=0	D=1
0	0	0.1	0.9
0	1	0.7	0.3
1	0	0.8	0.2
1	1	0.9	0.1

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

Graphical models

Belief Networks



$P(D|B,C)$

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

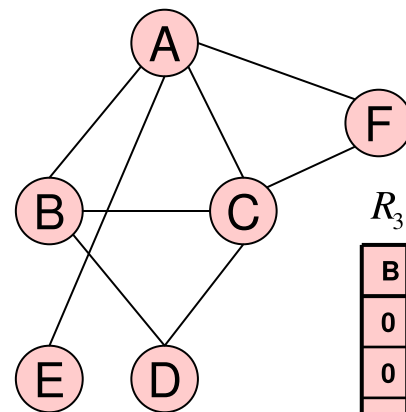
Variables: A, B, C, D, E, F

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

CPTS: $P(A), P(B|A), P(C|A), P(D|B,C)$

$P(E|A,B), P(F|A)$

Constraint Networks



$R_3(BCD)$

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: $\rho = R(ABCDEF)$

Inference: Bucket Elimination

$\textbf{sum}_B \prod$

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

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

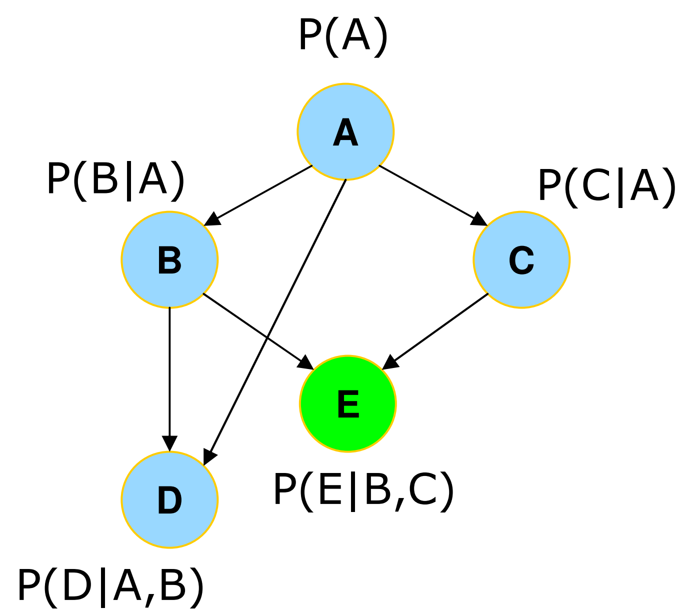
Bucket D h^C (A,D,E)

Bucket E   E = 0 h^D (A,E)

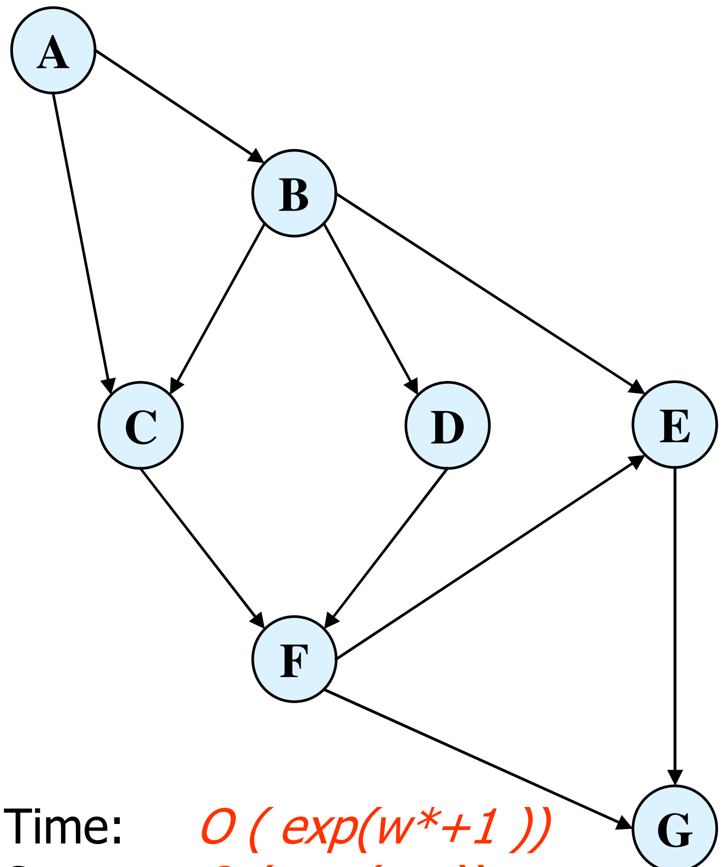
Bucket A   P(A) h^E (A)

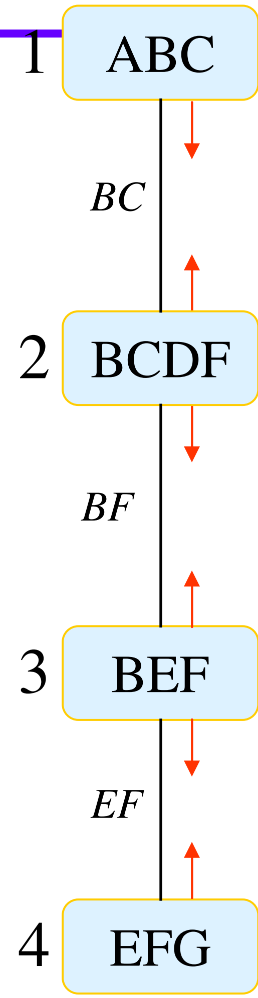
P(A,E=0)

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



Inference: Junction-Tree Propagation





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

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

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

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

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

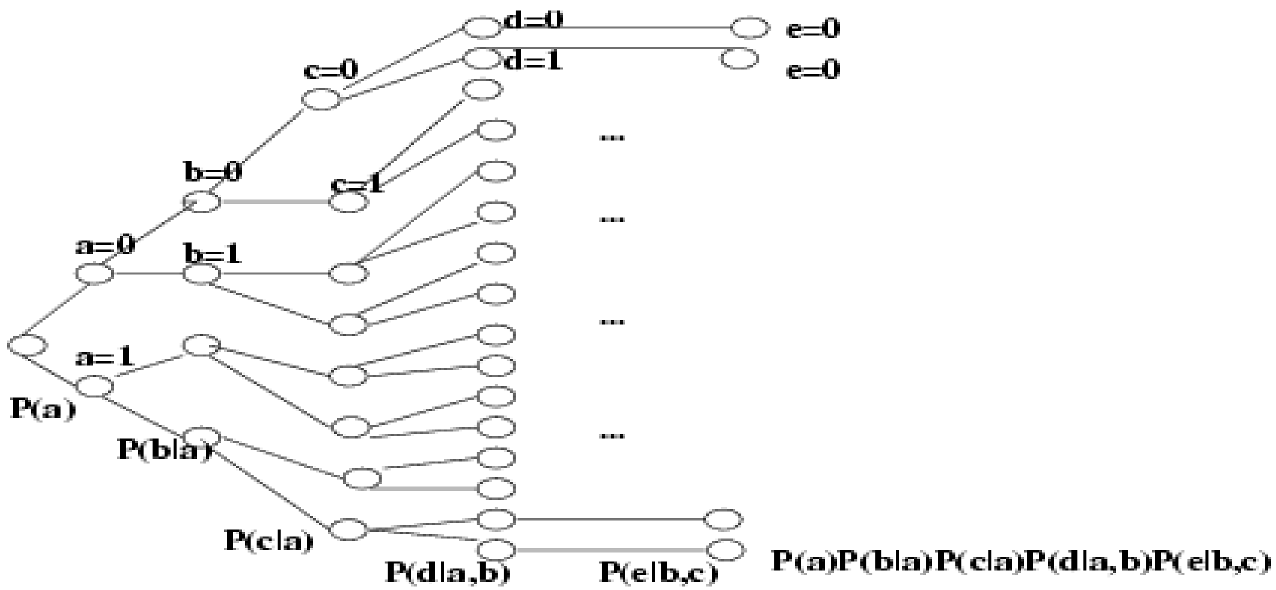
$$h_{(4,3)}(e,f) = p(G = g_e \mid e,f)$$

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

Space: *O (exp(sep))*

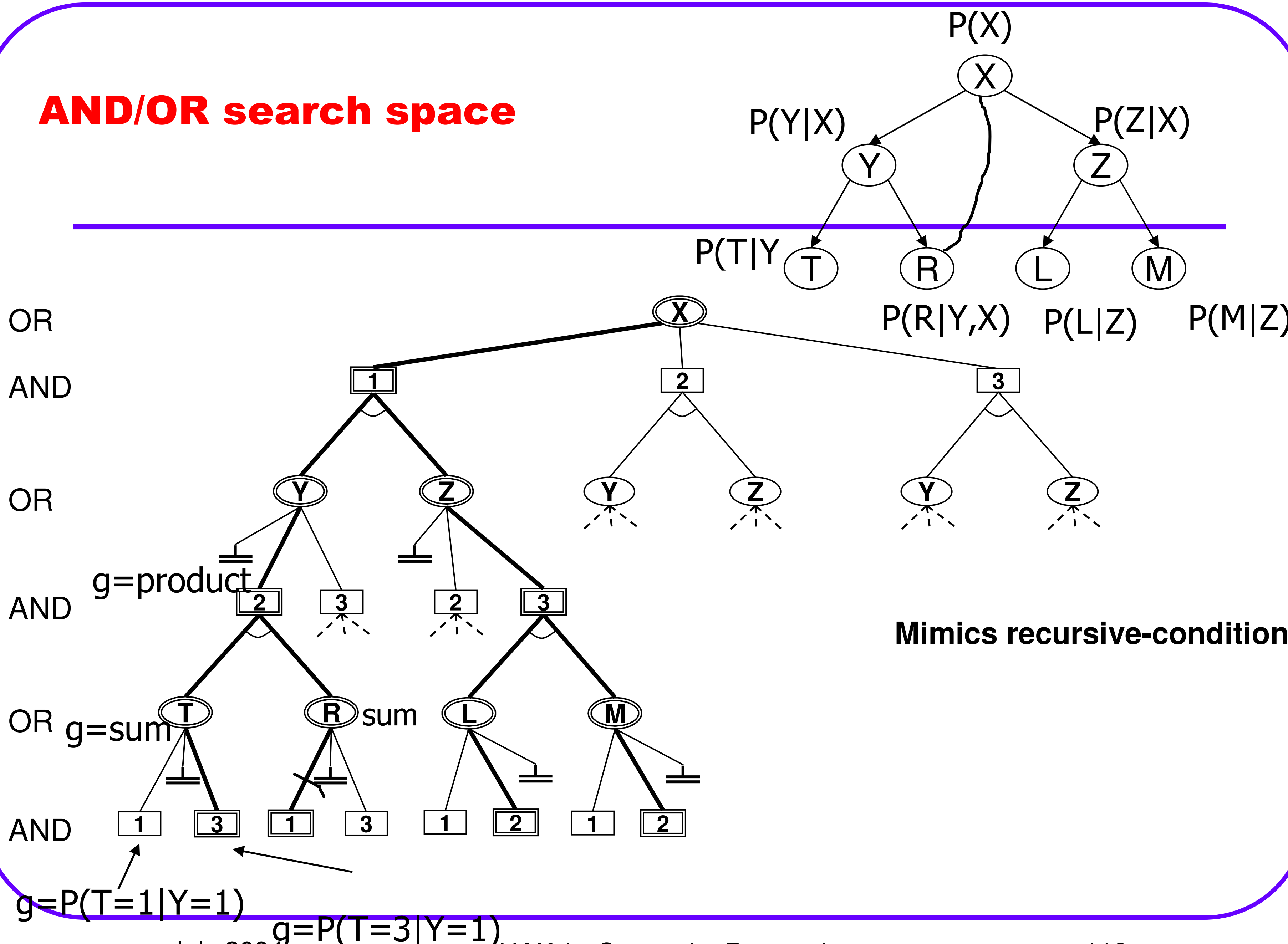
Conditioning generates the probability tree

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



Complexity: exponential time, linear space
Refined complexity: exponential in loop-cutest size,
Linear space.

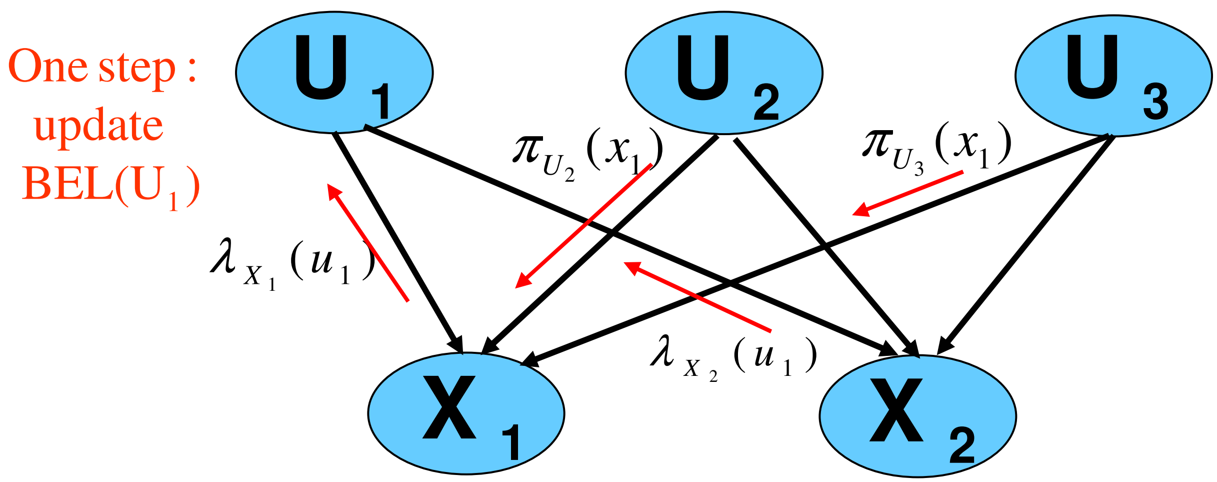
AND/OR search space



Mimics recursive-conditioning

Belief Propagation

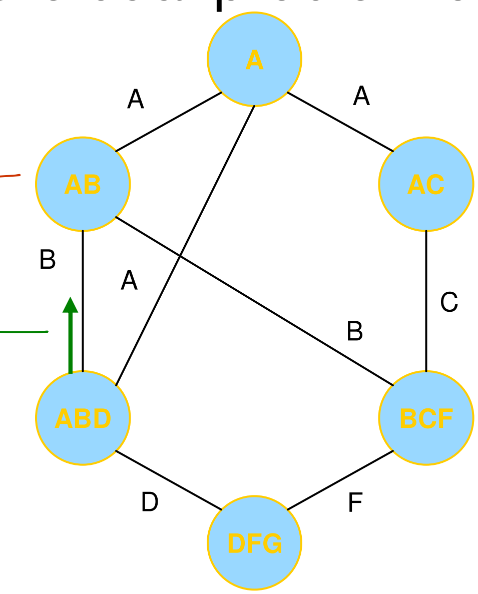
- Belief propagation is exact for poly-trees
- IBP - applying BP iteratively to cyclic networks



- No guarantees for convergence
- Works well for many coding networks

Belief Propagation on Dual Graph is Identical to Relational arc-Consistency

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



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

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

Belief Zero Propagation Equals Arc-Consistency

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

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

h_1^2, h_4^2, h_5^2

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

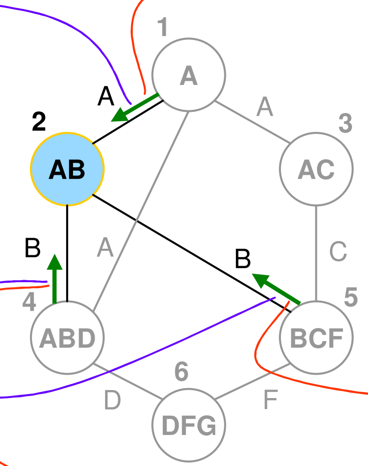
A	$h_1^2(A)$
1	>0
2	>0
3	>0
…	0

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:

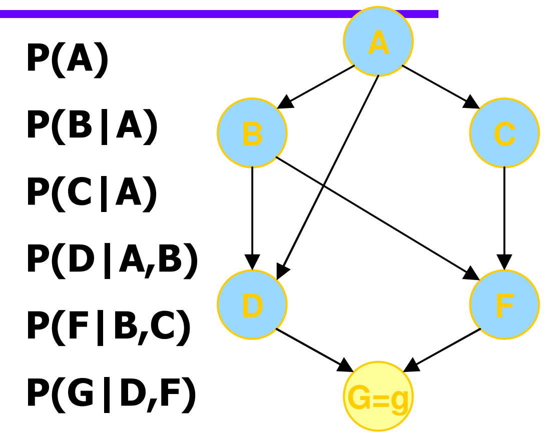
$$R(A,B) = R(A,B) \bowtie h_1^2 \bowtie h_4^2 \bowtie h_5^2 =$$

=

A	B
1	3
2	1
2	3
3	1

Properties of iterative algorithms

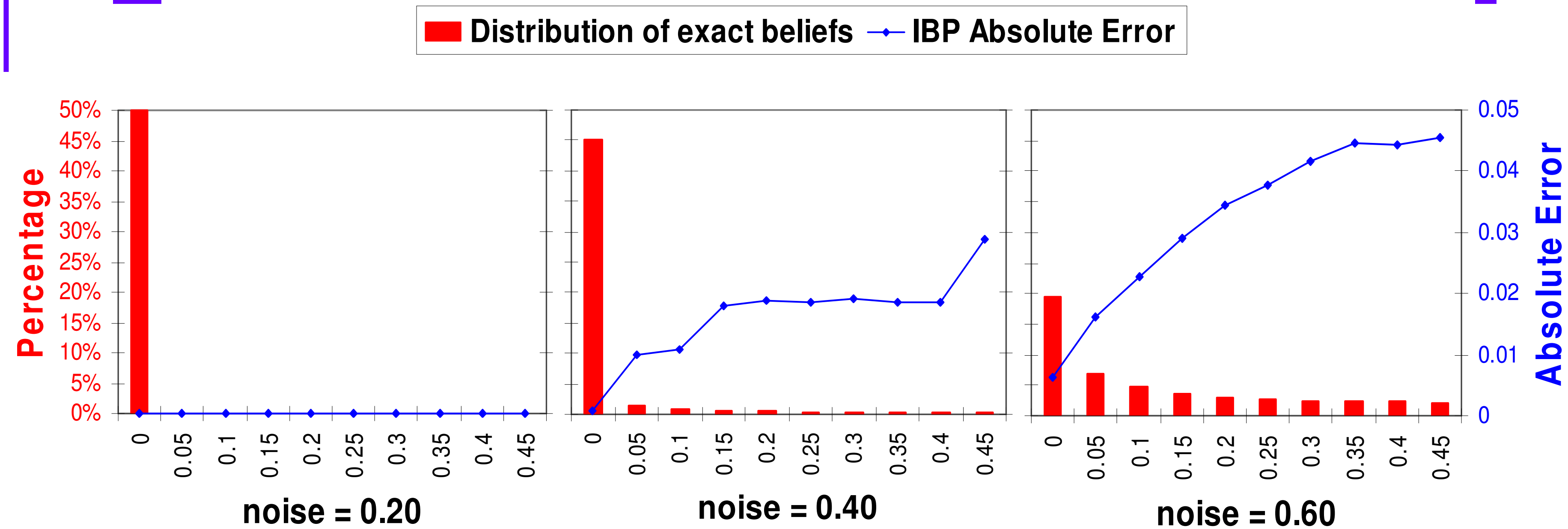
- IBP's inference of zero beliefs is identical to arc consistency's inference on the flat network
- Therefore:
  - **IBP is sound for zero beliefs**
  - **Weak/Strong when arc-consistency is**
- Empirical results suggest that these properties extend from zero beliefs to ε-small beliefs (ε > 0)



Belief updating:

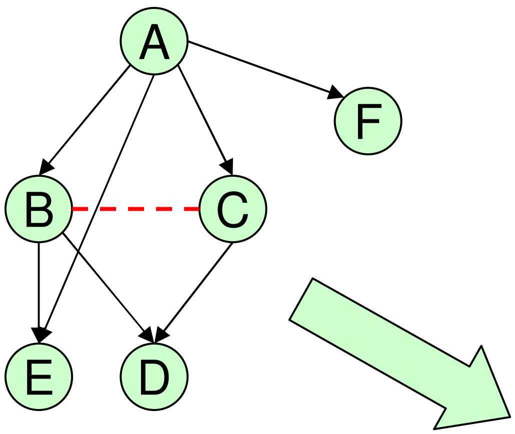
***Bel(B)*=?**

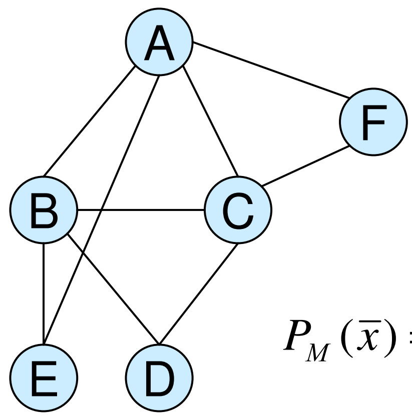
Coding networks

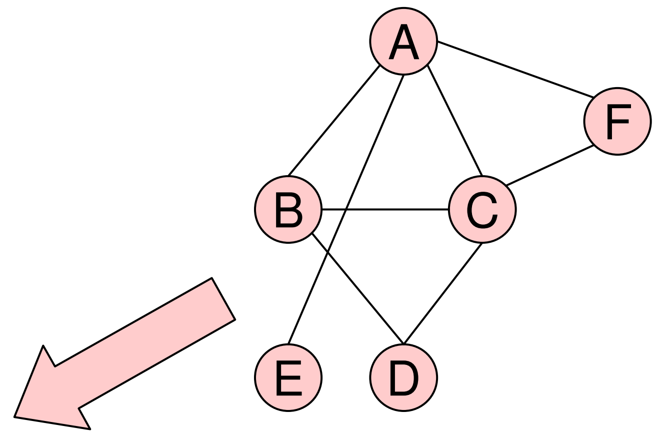


N=200, 1000 instances, w*=15

Mixed Networks







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

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

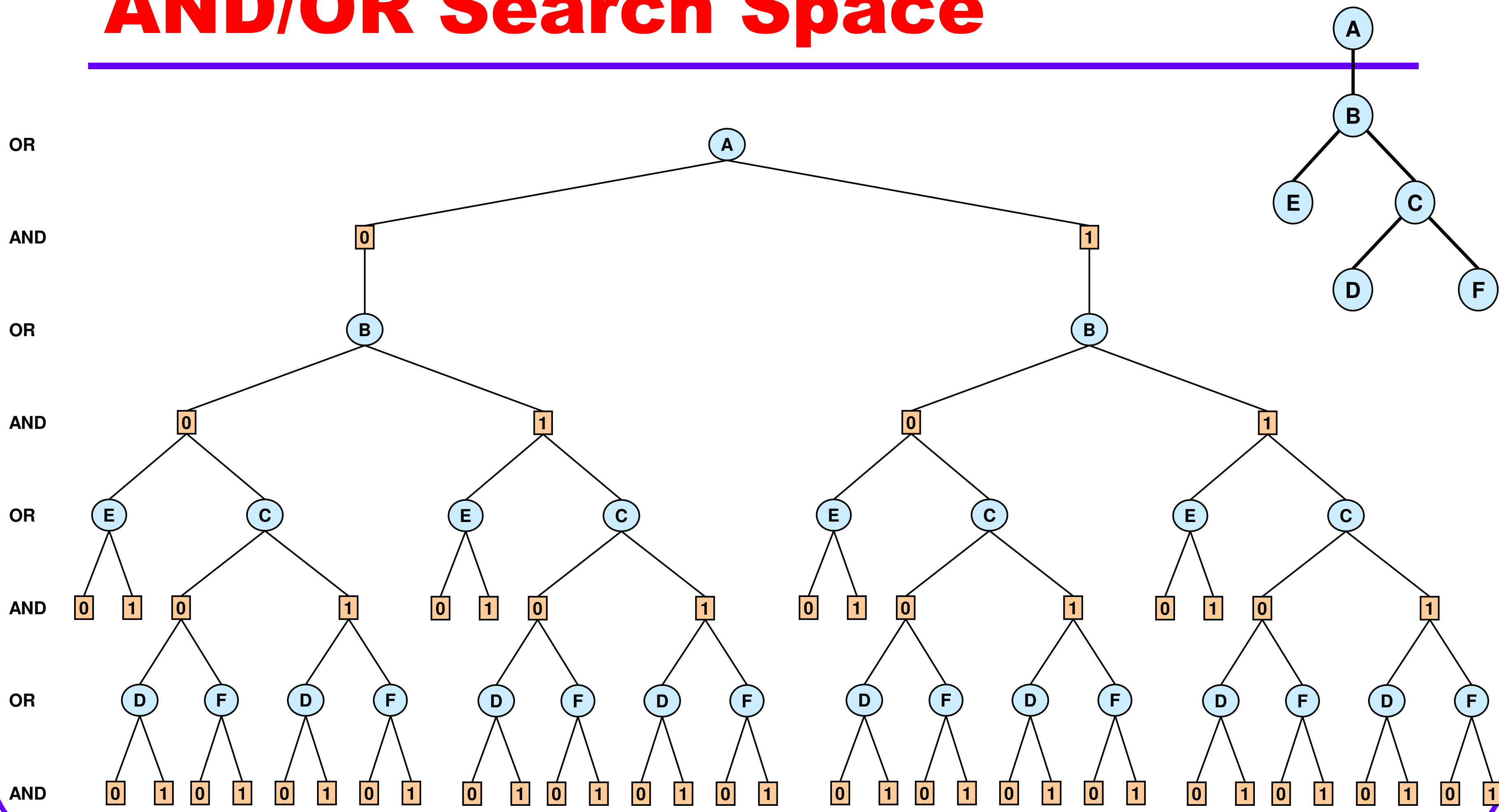
Domains: $D_A = D_B = D_C = D_D = D_E = D_F = \{0,1\}$

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

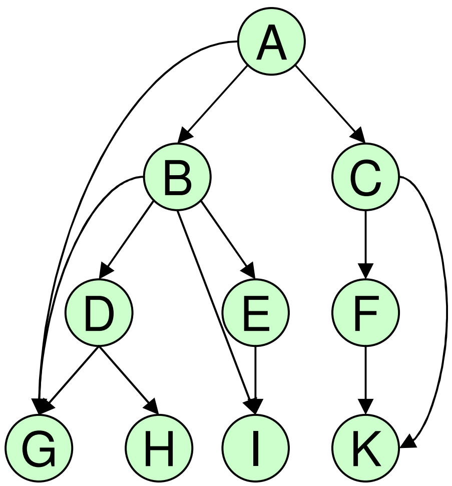
$P(E \mid A, B), P(F \mid A)$

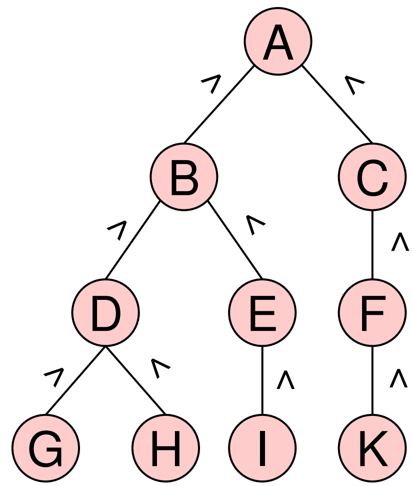
Relations: $R_1(ABC), R_2(ACF), R_3(BCD), R_4(A, E)$

AND/OR Search Space



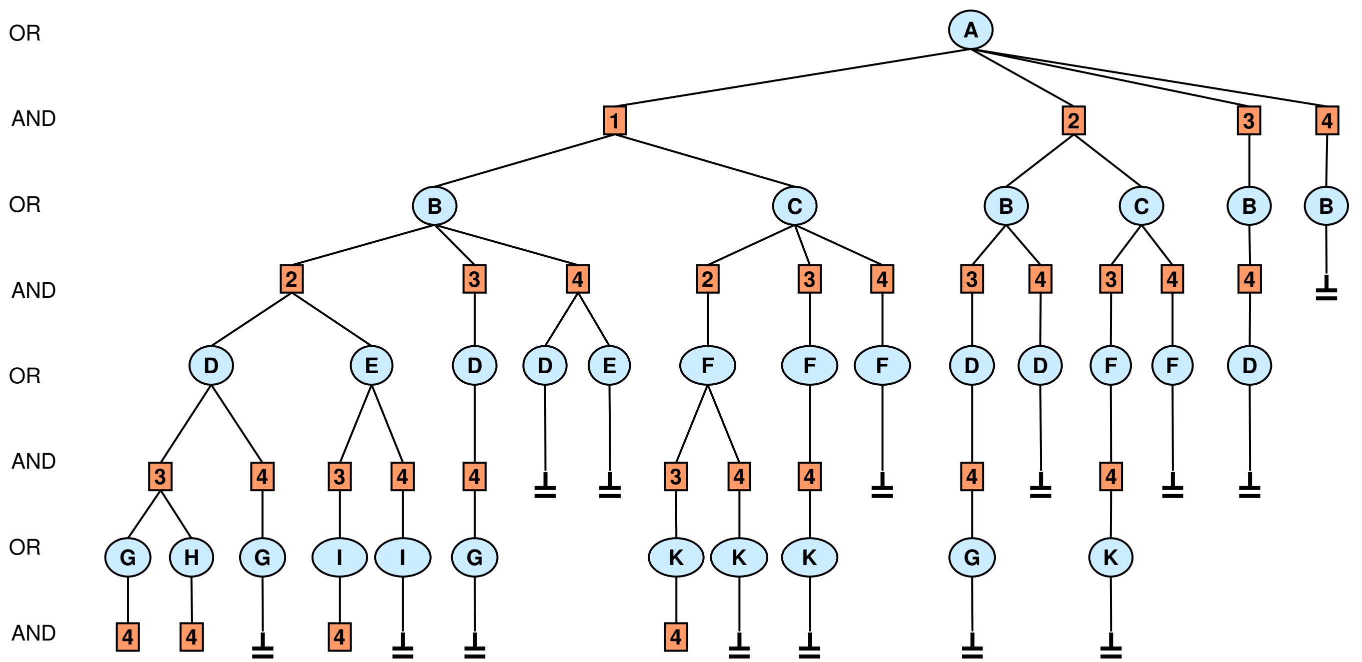
Constraint propagation

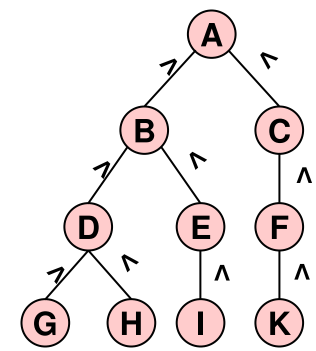




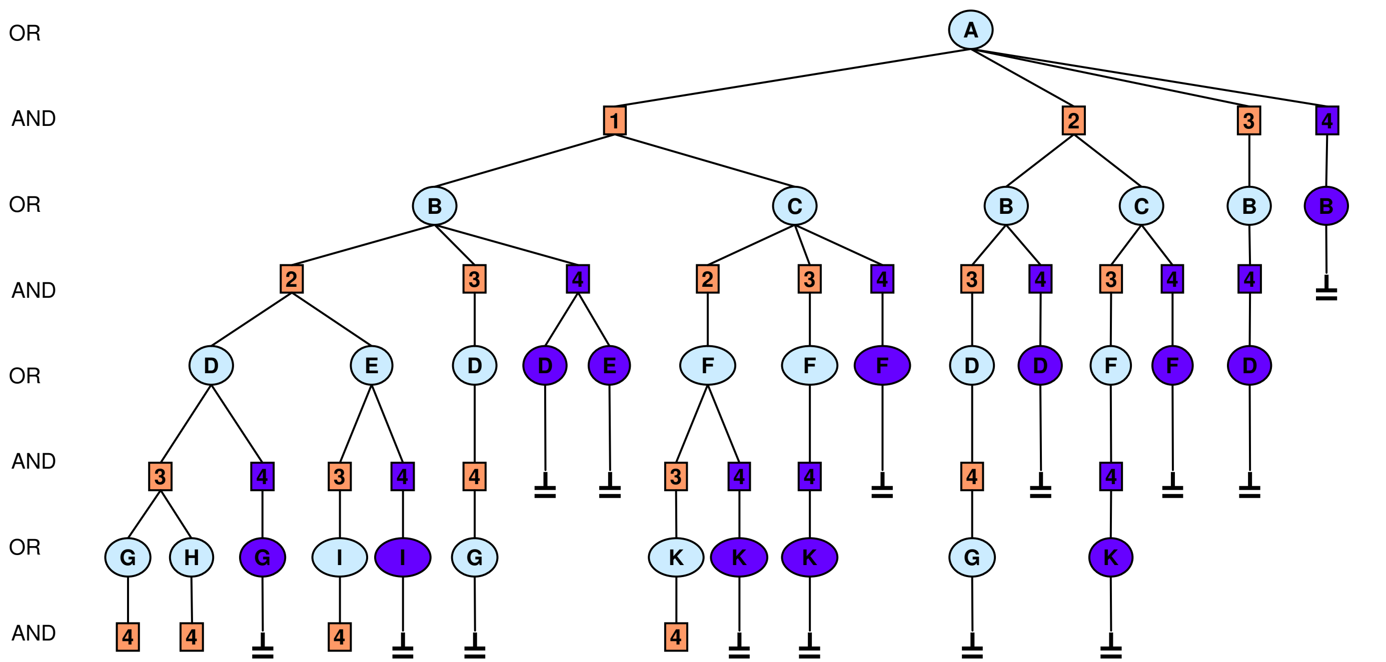
All domains are {1,2,3,4}

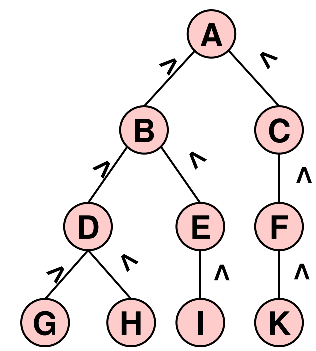
Constraint checking only



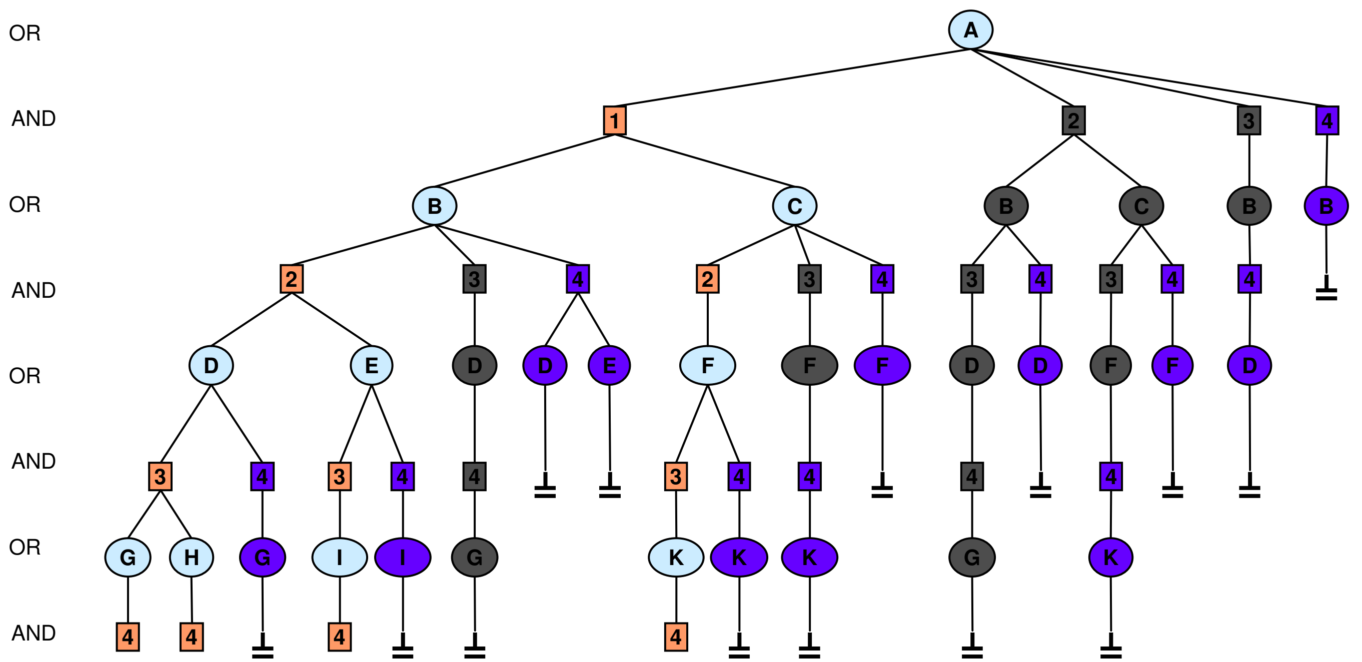


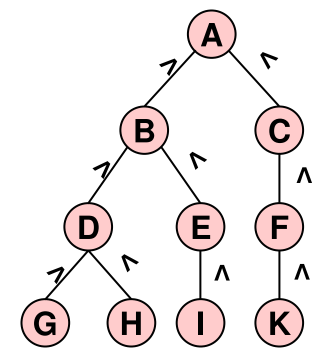
Forward checking





Maintaining arc-consistency





A recent Text book

Rina Dechter,

- **Constraint Processing,**
- Morgan Kaufmann