Advanced algorithms for Graphical Models

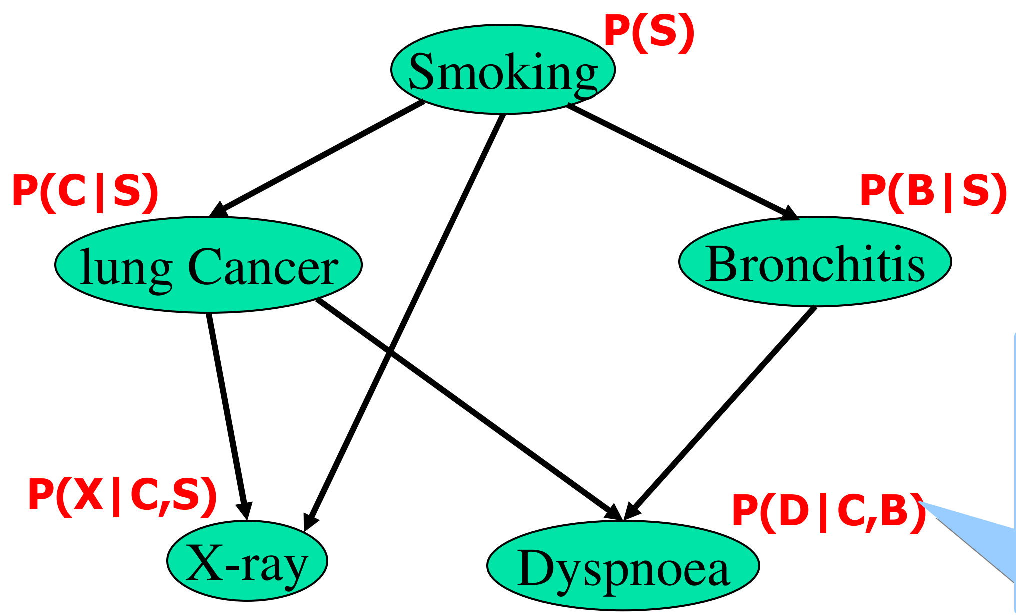
Rina Dechter
University of California
Irvine

Collaborators:
Kalev Kask
Radu Marinescu
Robert Mateescu

Overview

- **Introduction and background for graphical models: inference and search**
- **Bounded inference**: mini-bucket and mini-clustering, Generalized belief propagation
- **Hybrid of inference and search:** Heuristic generation and Brunch and Bound
- **AND/OR search spaces for graphical models:** tree spaces, graph spaces, empirical evaluation.

Probabilistic Networks



BN = (G, Θ)

CPD:

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

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

$P(S|d) = ?$

MPE: argmax $P(S)\,P(C|S)\,P(B|S)\,P(X|C,S)\,P(D|C,B)$

Constraint Satisfaction

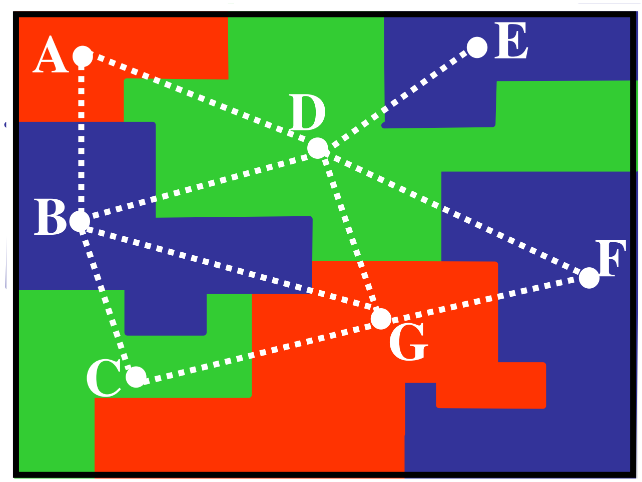
Example: map coloring

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

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

Constraints (C): **A ≠ B, A ≠ D, D ≠ E,** *etc.*

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

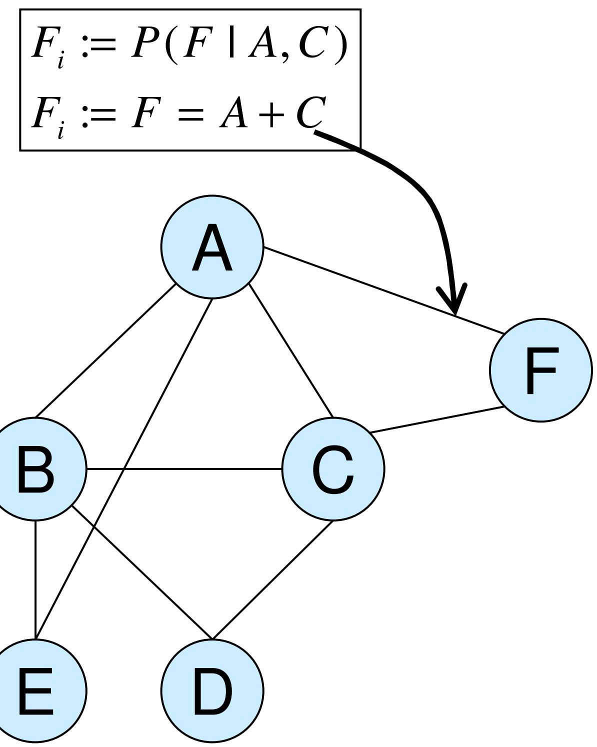


Semantics: set of all solutions

Primary task: find a solution

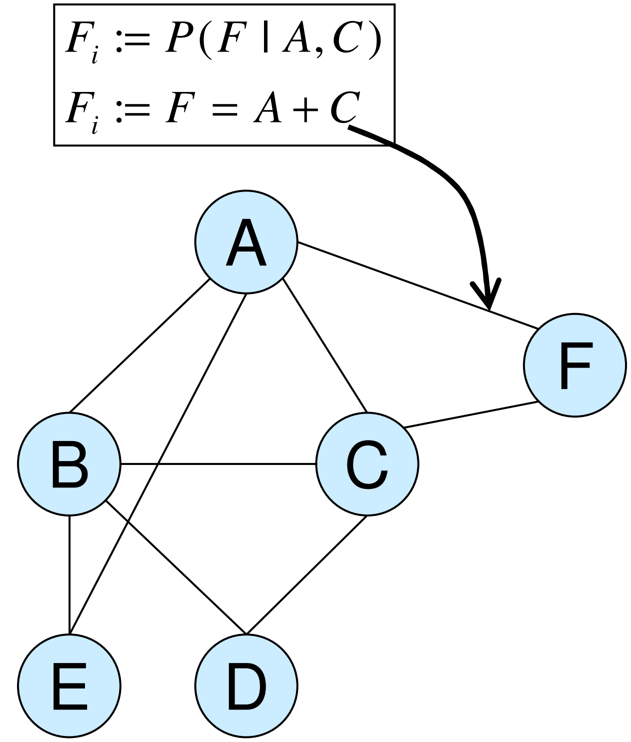
Graphical models

- A graphical model (X,D,C):
  - $X = \{X_1, \ldots X_n\}$ variables
  - $D = \{D_1, \ldots D_n\}$ domains
  - $C = \{F_1, \ldots, F_t\}$ functions (constraints, CPTS, cnfs)
- Primal graph G (constraint graph, moral graph)
- Depth-first search (DFS) spanning trees
- Induced-width, tree-width, path-width



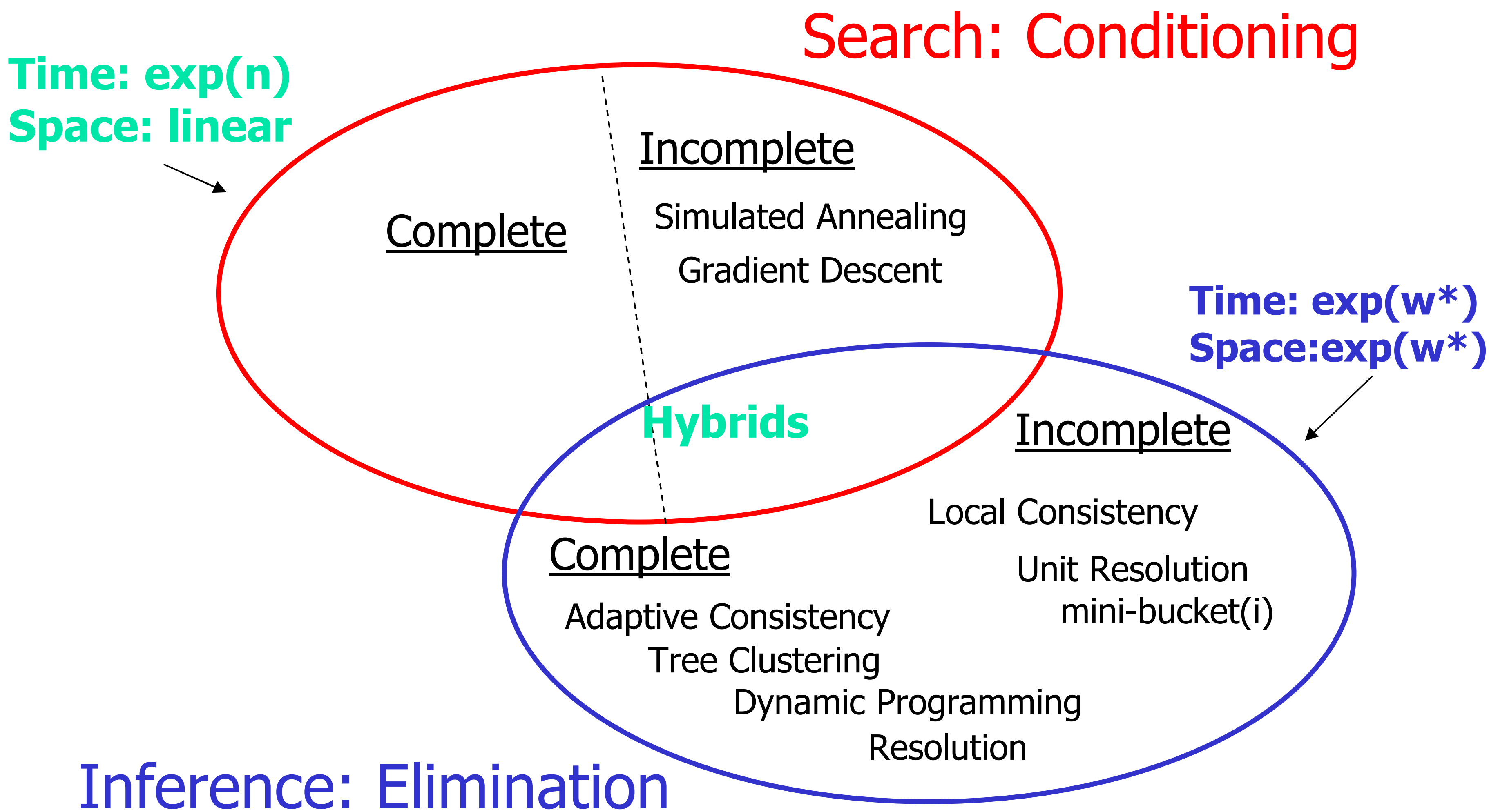
Graphical models

- A graphical model (X,D,C):
  - $X = \{X_1, \ldots X_n\}$ variables
  - $D = \{D_1, \ldots D_n\}$ domains
  - $C = \{F_1, \ldots, F_t\}$ functions (constraints, CPTS, cnfs)
- Primal graph G (constraint graph, moral graph)
- Depth-first search (DFS) spanning trees
- Induced-width, tree-width, path-width

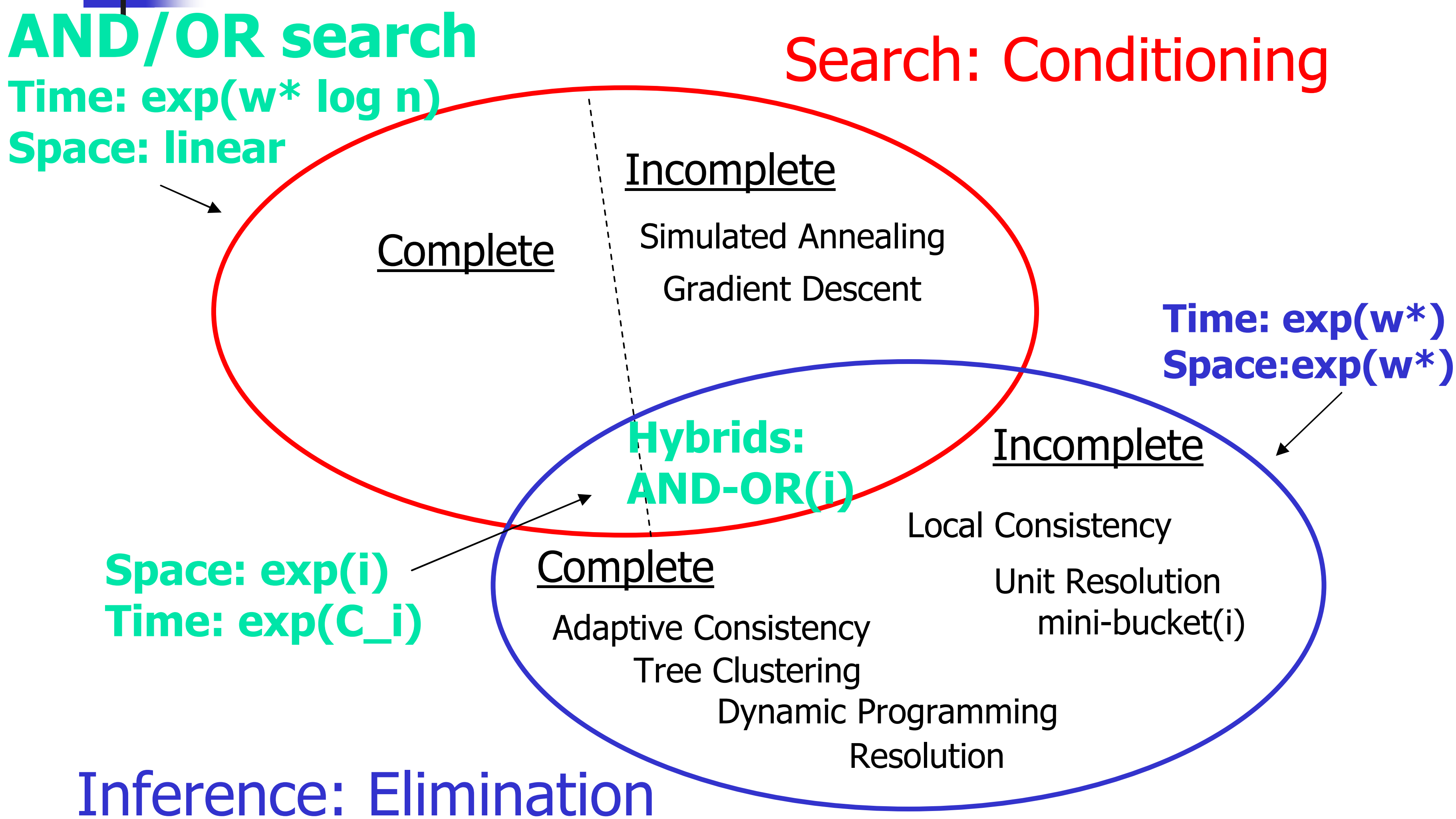


- **Belief updating:** $\Sigma_{X-y} \prod_j P_j$
- **MPE:** $\max_X \prod_j P_j$
- **CSP:** $\prod_X \times_j C_j$
- **Max-CSP:** $\min_X \Sigma_j F_j$

Solution Techniques



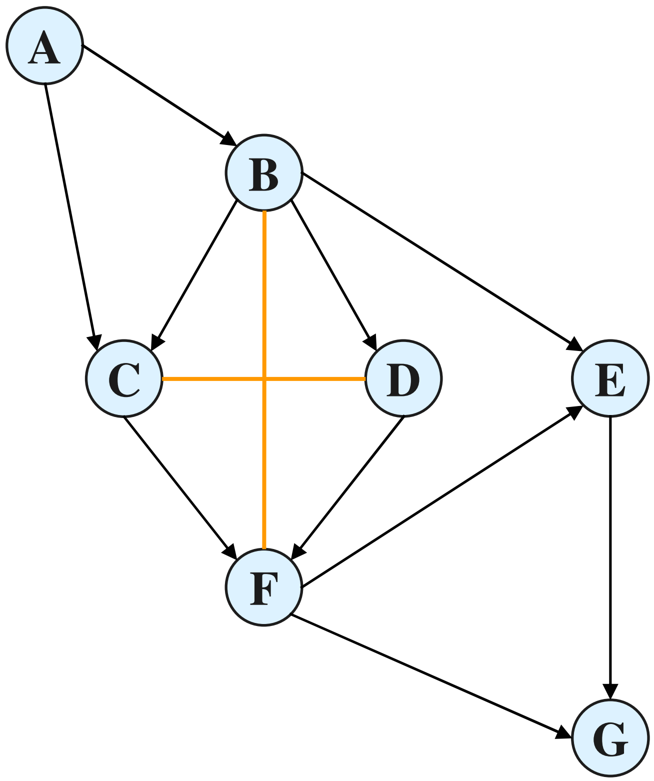
Solution Techniques

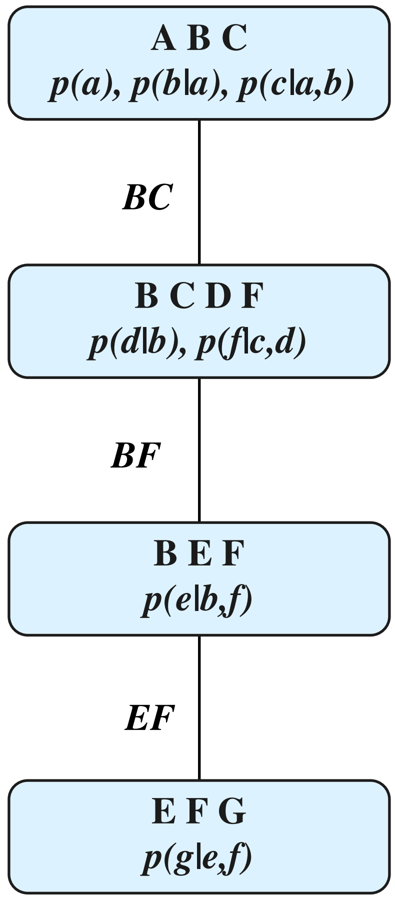


Overview

- Introduction and background for graphical models: inference and search
- **Inference:**
  - **Tree-clustering, variable elimination,**
  - **mini-bucket and mini-clustering,**
  - **Generalized belief propagation**
- **Hybrid of inference and search:** Heuristic generation and Brunch and Bound
- AND/OR search spaces for graphical models

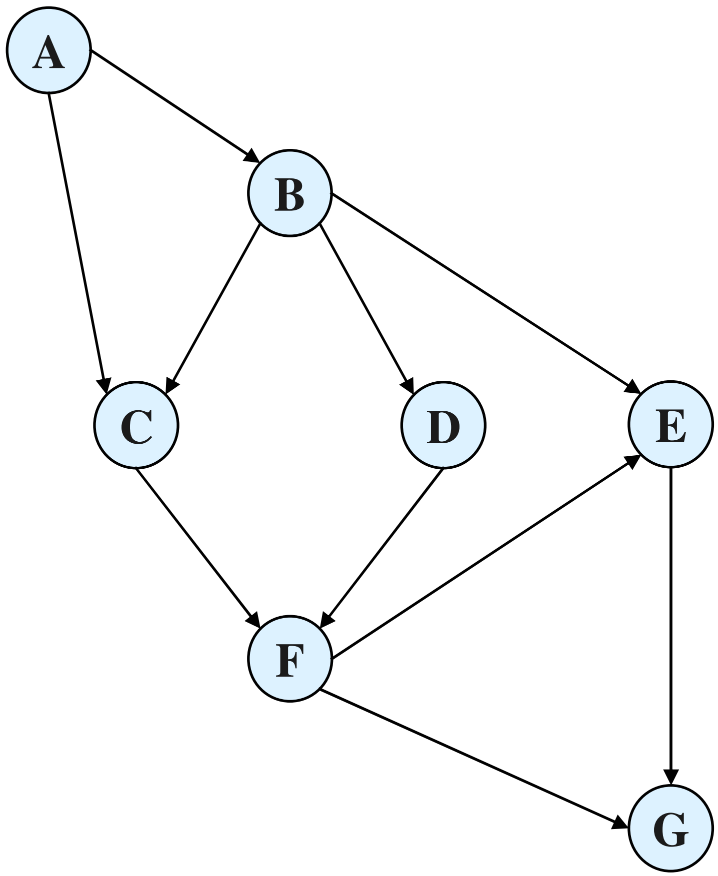
Tree decomposition

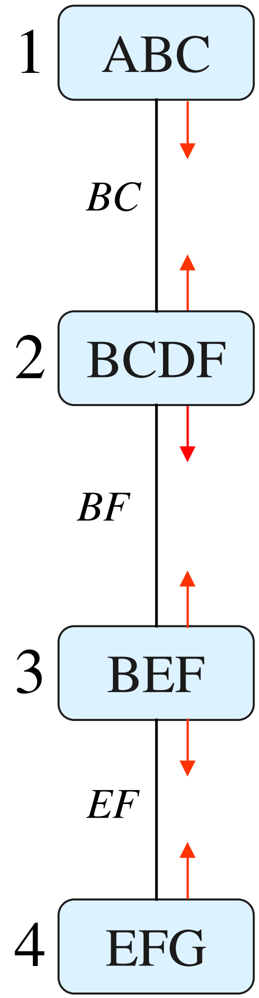




- Each function in a cluster
- Satisfy running intersection property

CTE: Cluster Tree Elimination





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

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

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

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

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

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

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

For each cluster P(X|e) is computed

Bucket Elimination
(dechter 1996)

$\max_B \prod$

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

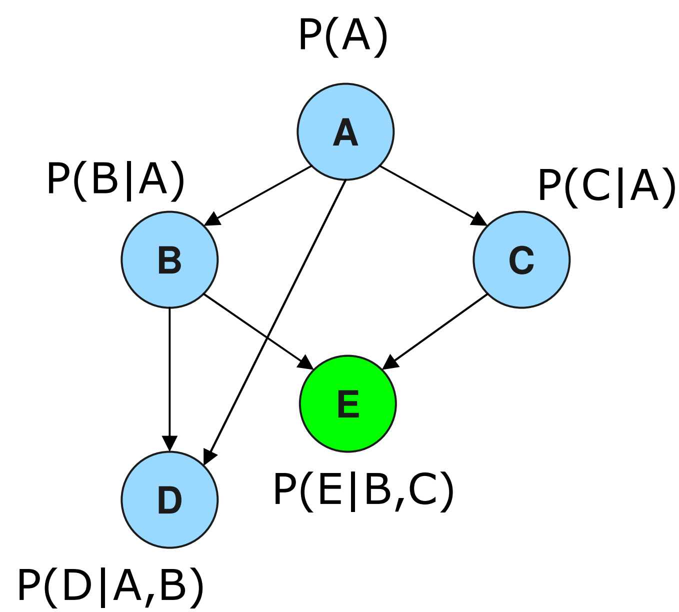
Bucket C P(C|A) h^B (A,D,C,E)

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

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

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

MPE



Bucket elimination

Algorithm Elim-*MPE* (Dechter 1996)

$\max_b \prod$ ← Elimination operator

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

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

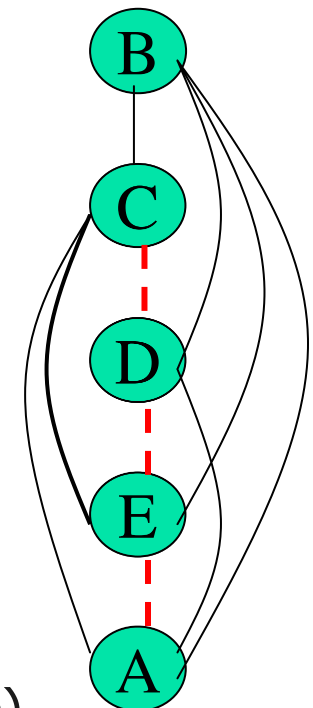
bucket D: $h^C(a, d, e)$

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

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

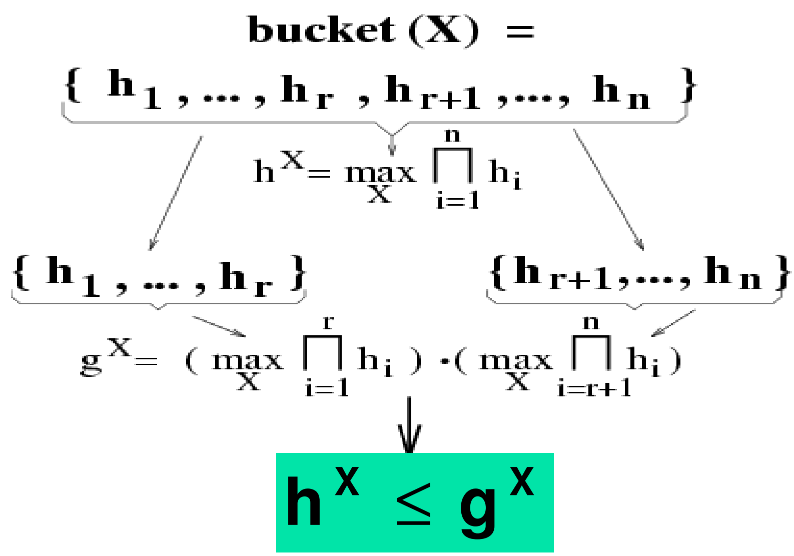
mpe

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



Two Principles for Bounded Inference

- **Bounded-Partitioning**
  - **mini-bucket(i), MC(i)**
  - **Computes a bound**
  - **Exp(i) time space**

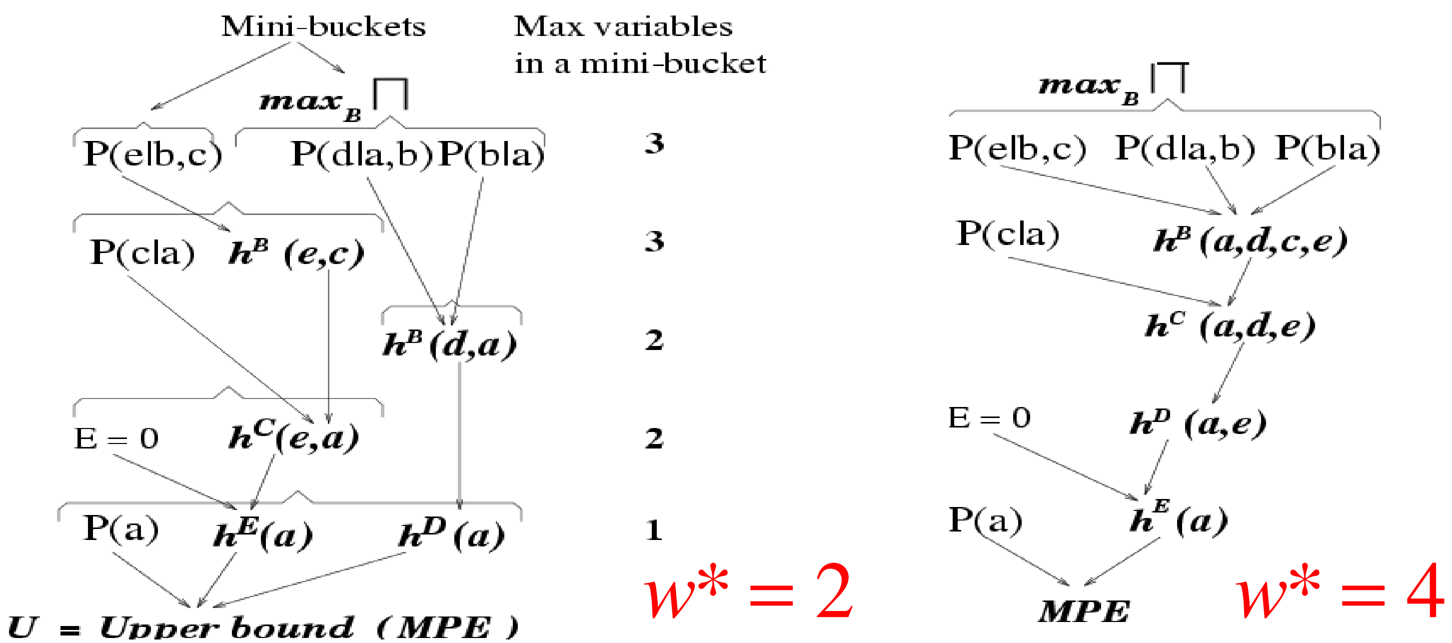


Approx-mpe(i)

(Dechter&Rish 1997)

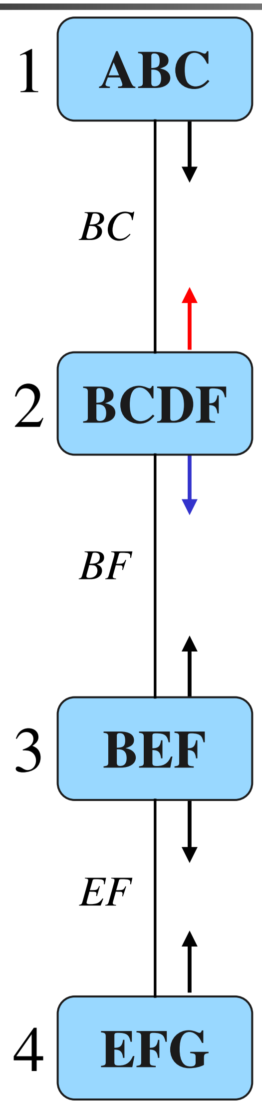
- **Input: i – max number of variables allowed in a mini-bucket**
- **Output: [lower bound (P of a sub-optimal solution), upper bound]**

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



Mini-Clustering – MCTE(i)

(Kask, Dechter, mateescu,2002)



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

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

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

$H_{(3,2)}$: $$h^1_{(3,2)}(b,f) := \sum_e p(e|b,f)\cdot h^1_{(4,3)}(e,f)$$

$H_{(3,4)}$: $$h^1_{(3,4)}(e,f) := \sum_b p(e|b,f)\cdot h^1_{(2,3)}(b)\cdot h^2_{(2,3)}(f)$$

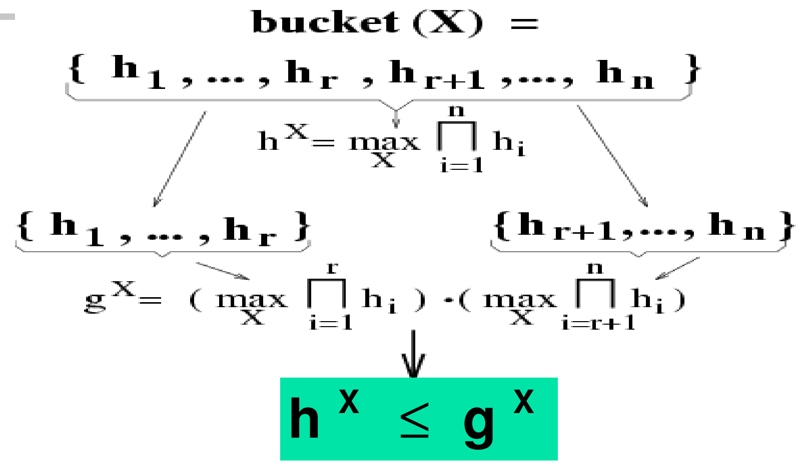
$H_{(4,3)}$: $$h^1_{(4,3)}(e,f) := p(G=g_e|e,f)$$

Properties of MC(i)

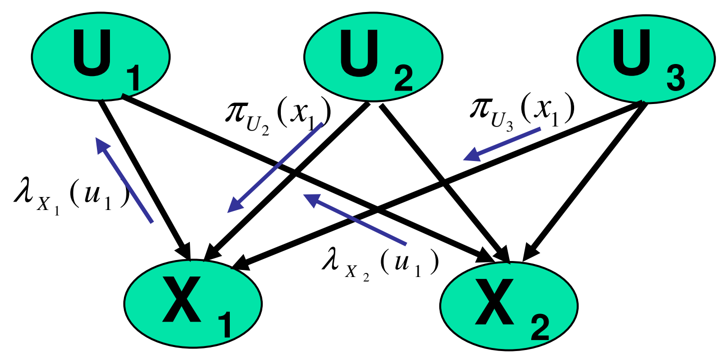
- MC (i) computes a bound on the exact value : approximating the exact query
- Time & space complexity: O(n exp(i))
- Approximation improves with i but takes more time

Two Principles for Bounded Inference

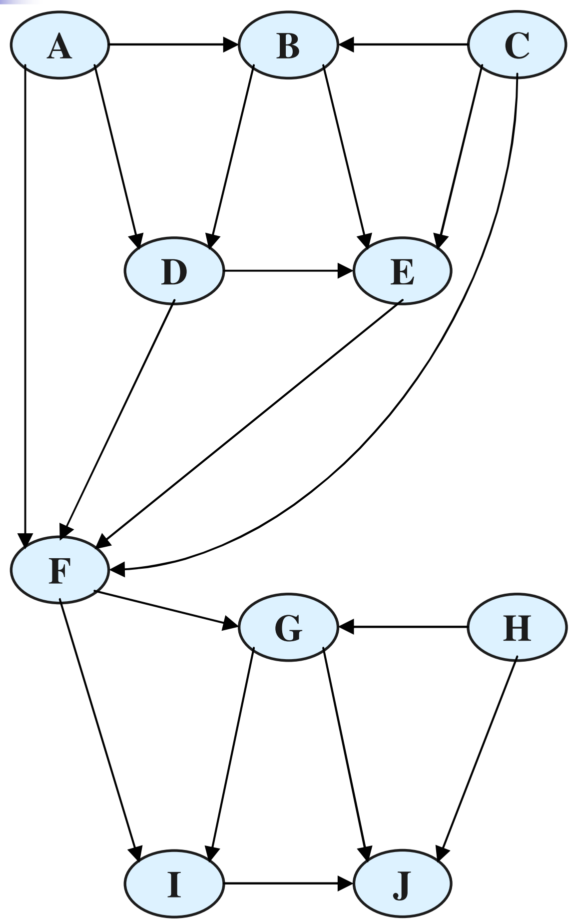
- **Bounded-Partitioning**
  - **mini-bucket(i), MC(i)**
  - **Computes a bound**
  - **Exp(i) time space**



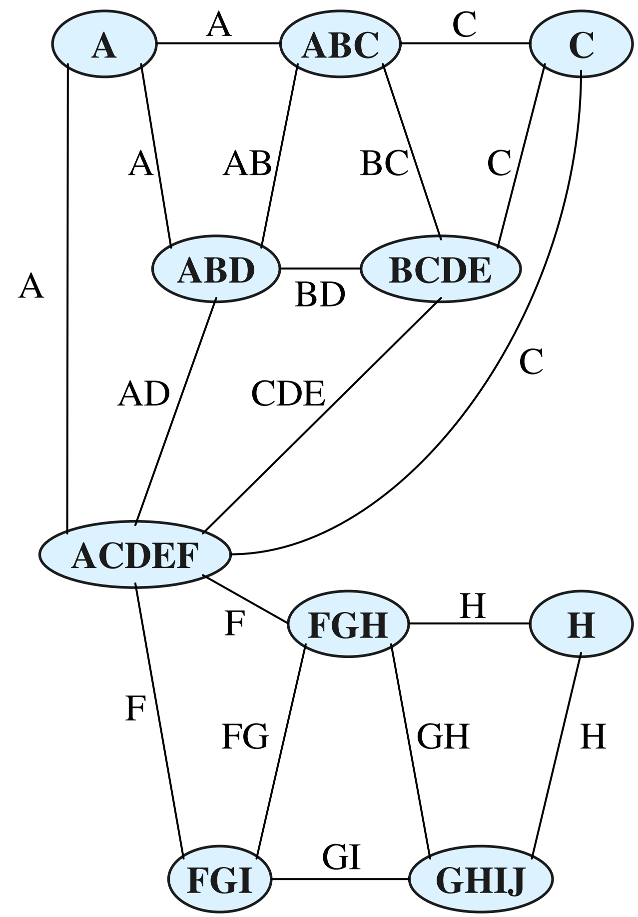
- **Belief propagation on join-graphs**
  - **IBP, IJGP(i)**
  - **No guuarantees**
  - **Each iteration is exp(i)**



Iterative Join-Graph Propagation

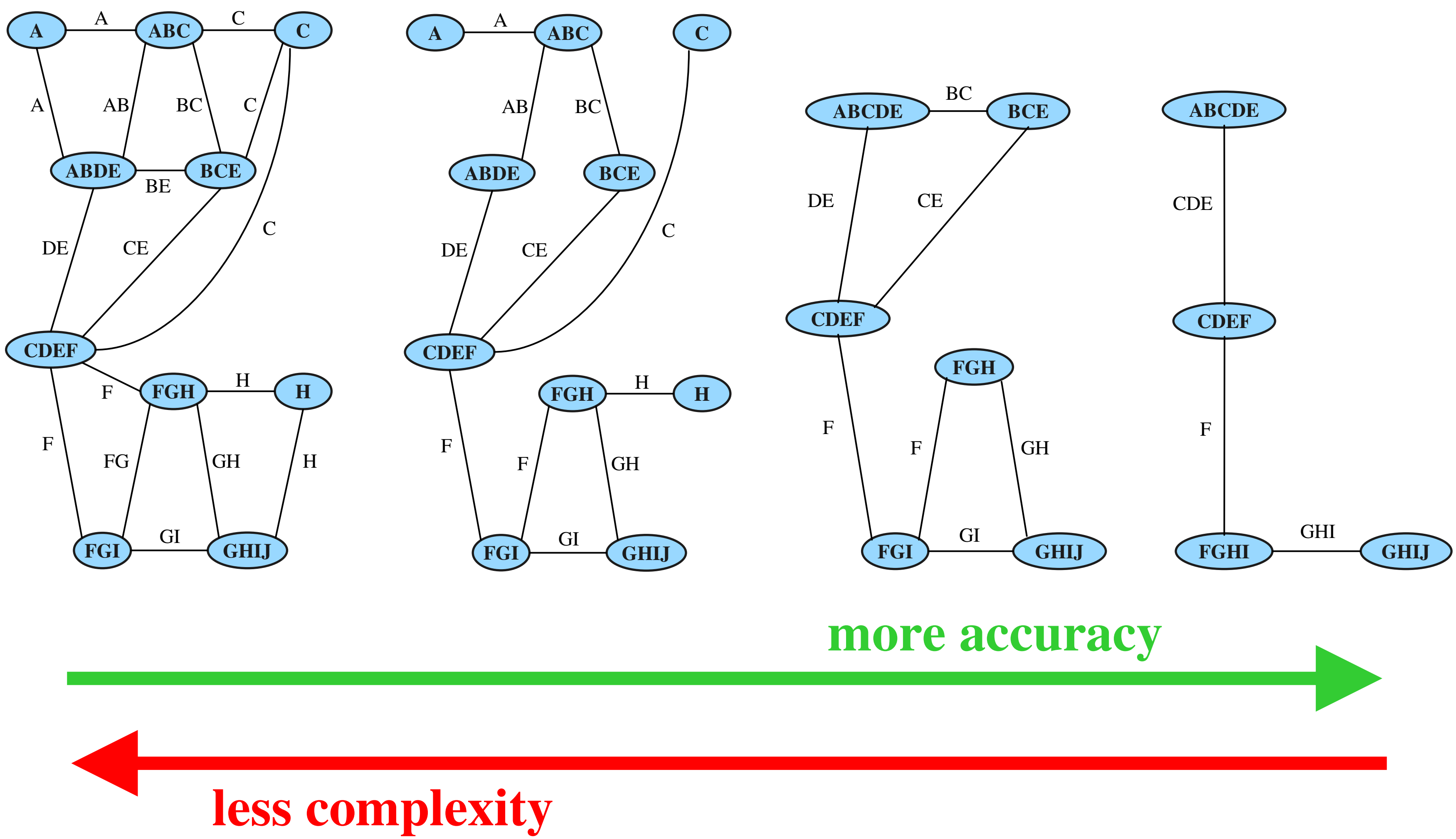


a) Belief network

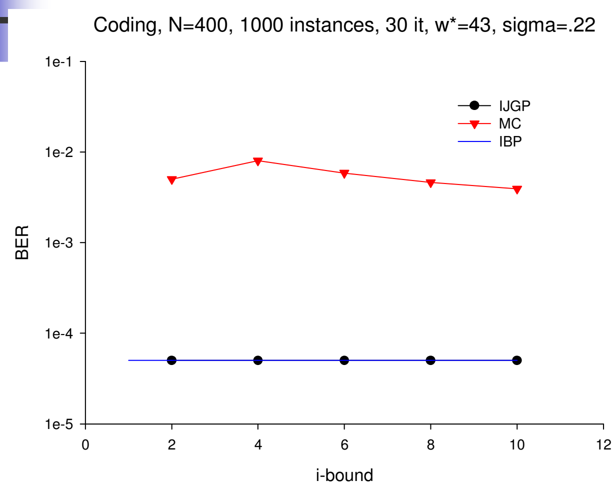


a) The graph IBP works on

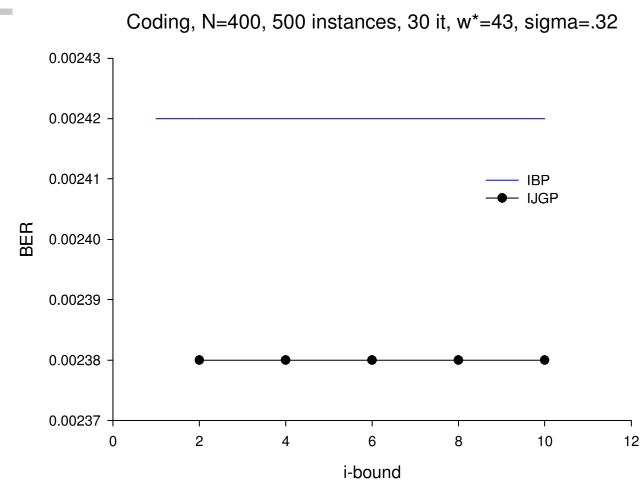
Join-graphs


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

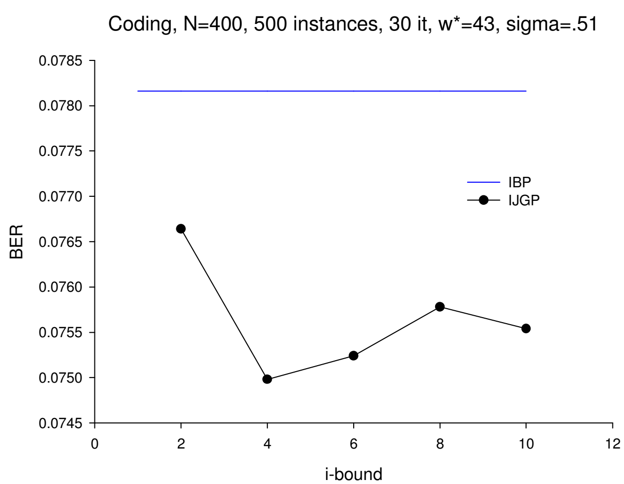
Coding networks - BER



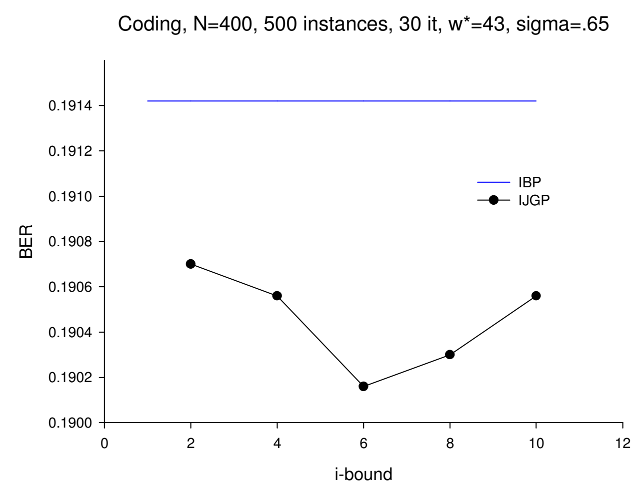
sigma=.22



sigma=.32

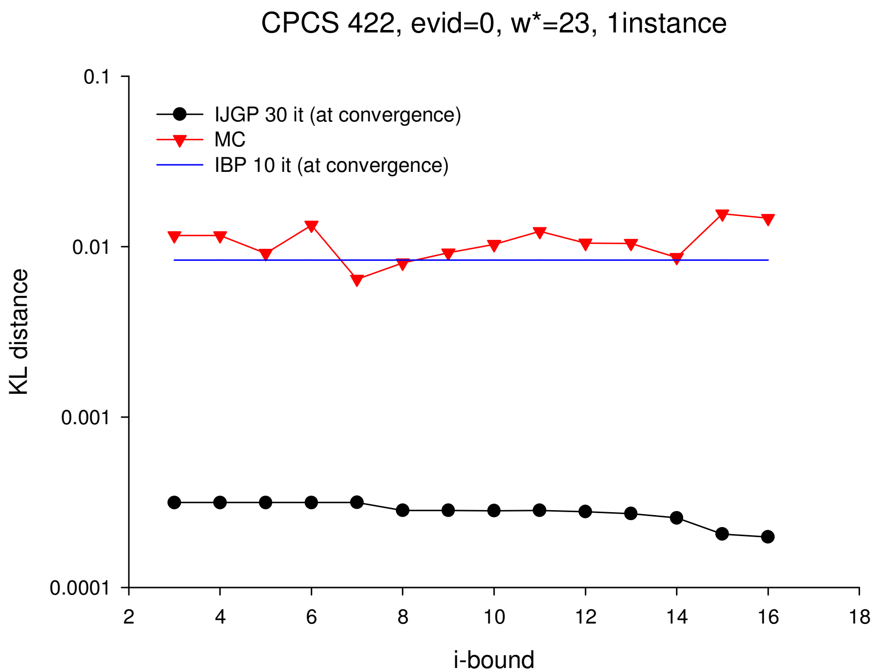


sigma=.51

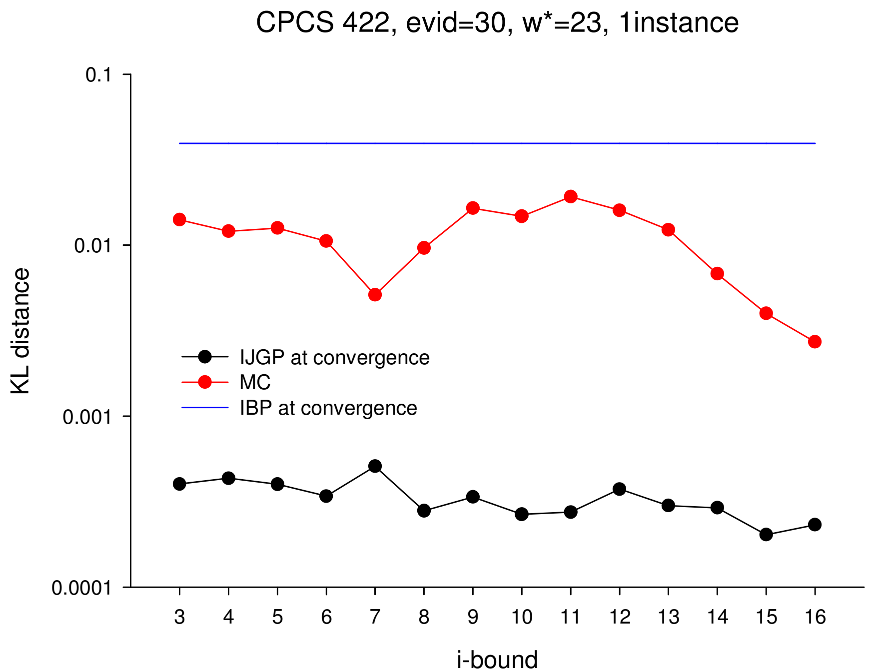


sigma=.65

CPCS 422 – KL distance



evidence=0



evidence=30

Empirical results showed:

- **Mini-bucket(i) and MC(i)**
  - Accuracy/time increase with i-bound
  - Compute upper/lower bounds.
  - demonstrate impressive performance for many problem classes for both optimization and belief updating.

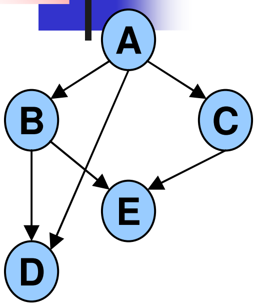
- **IJGP(i) is generally superior to MC for belief updating. But no bound.**

Overview

- Introduction and background for graphical models: inference and search
- **Inference:**
  - **Tree-clustering, variable elimination,**
  - **mini-bucket and mini-clustering,**
  - **Generalized belief propagation**
- **Hybrid of inference and search:** Heuristic generation and Branch and Bound
- AND/OR search spaces for graphical models

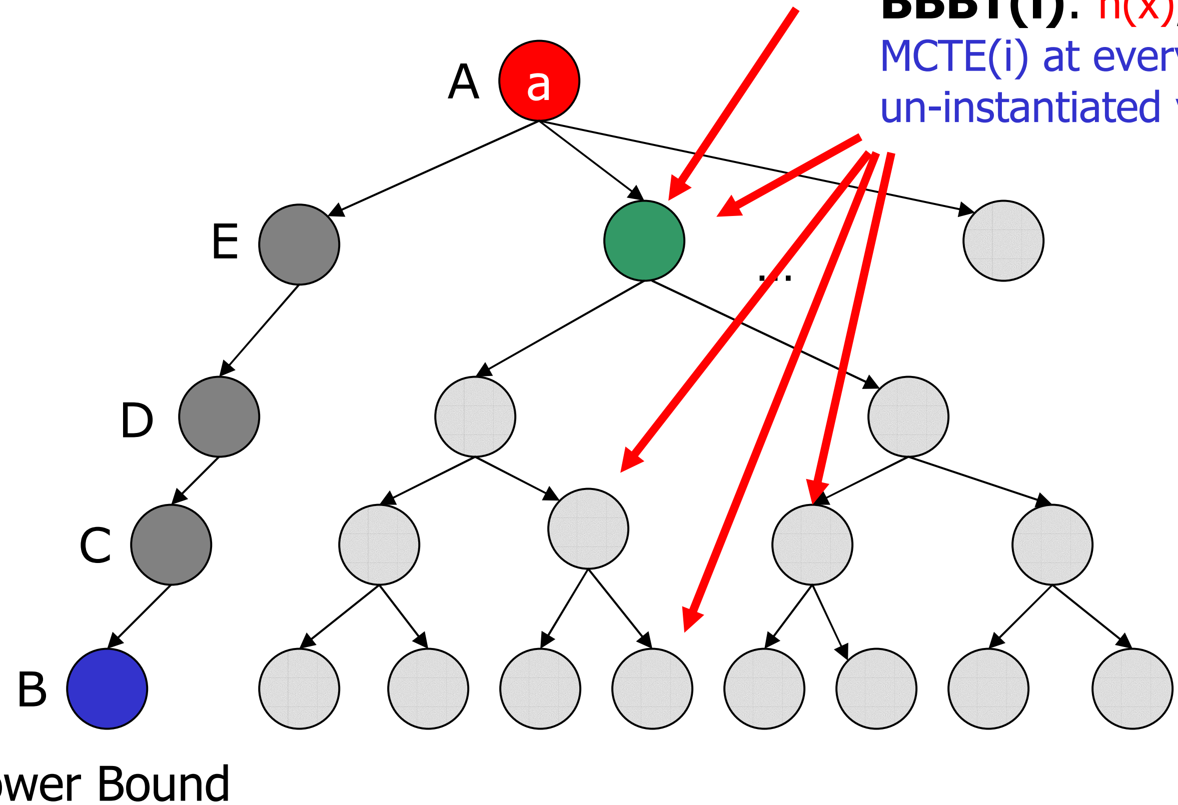
Two BnB schemes

(Kask & Dechter, 1999) (Kask, Dechter and Marinescu, 2003)

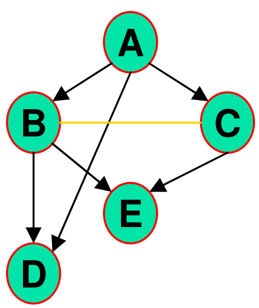


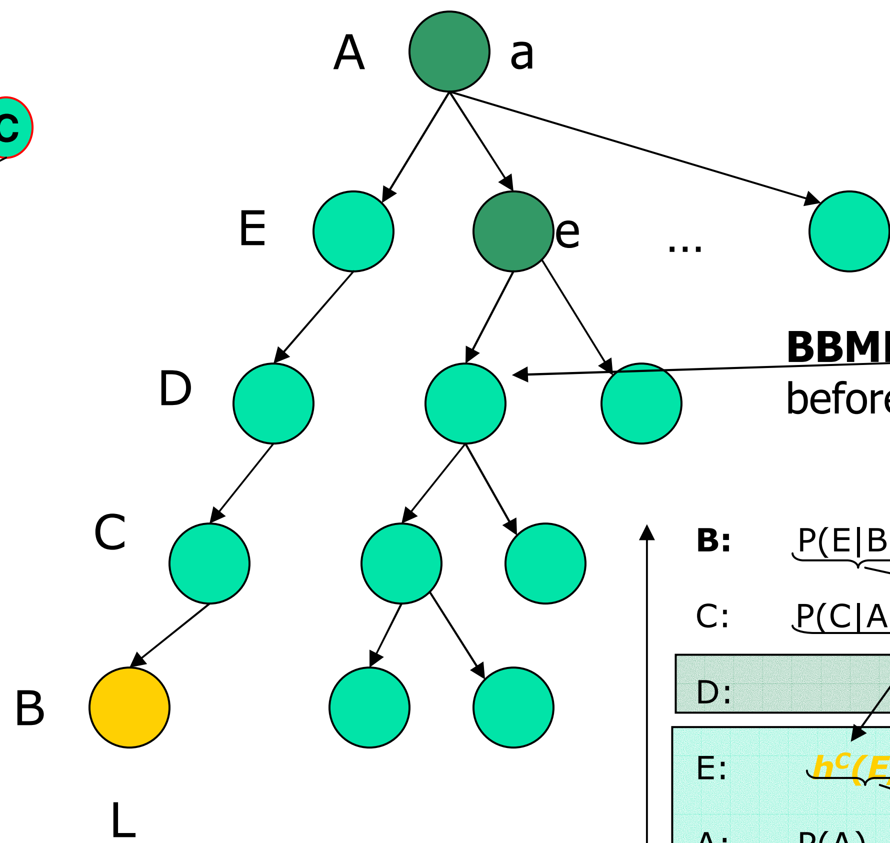
BBMB(i): h(x) computed by MB(i), before search, static ordering

BBBT(i): h(x), computed via MCTE(i) at every node for every un-instantiated variable

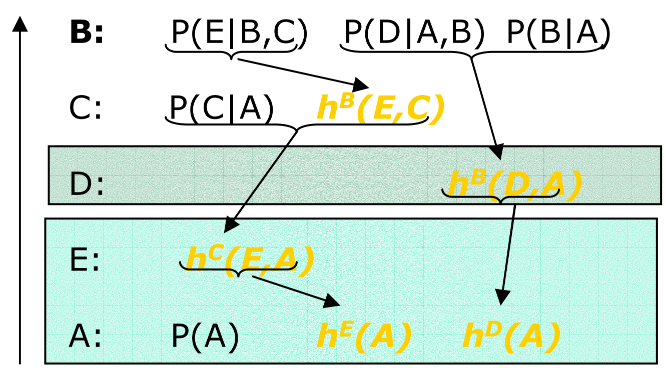


BBMB



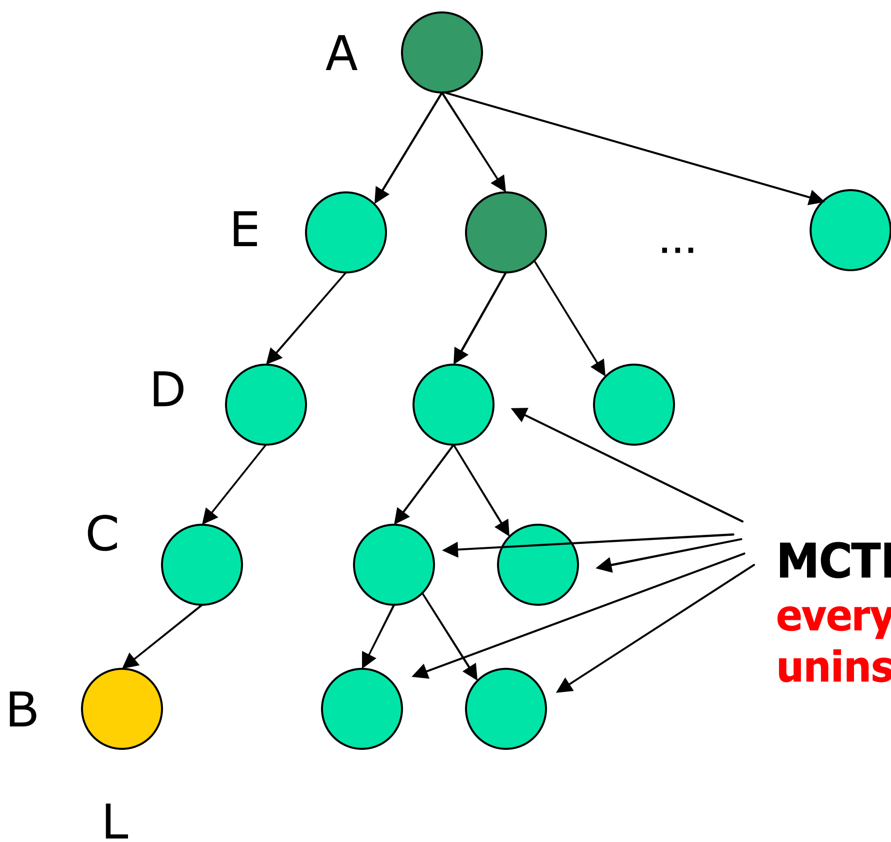


BBMB(i): h(x) computed by MB(i), before search, static ordering



$$f(a,e,D) = P(a) \cdot h^B(D,a) \cdot h^C(e,a)$$

BBBT(i) – Search Space

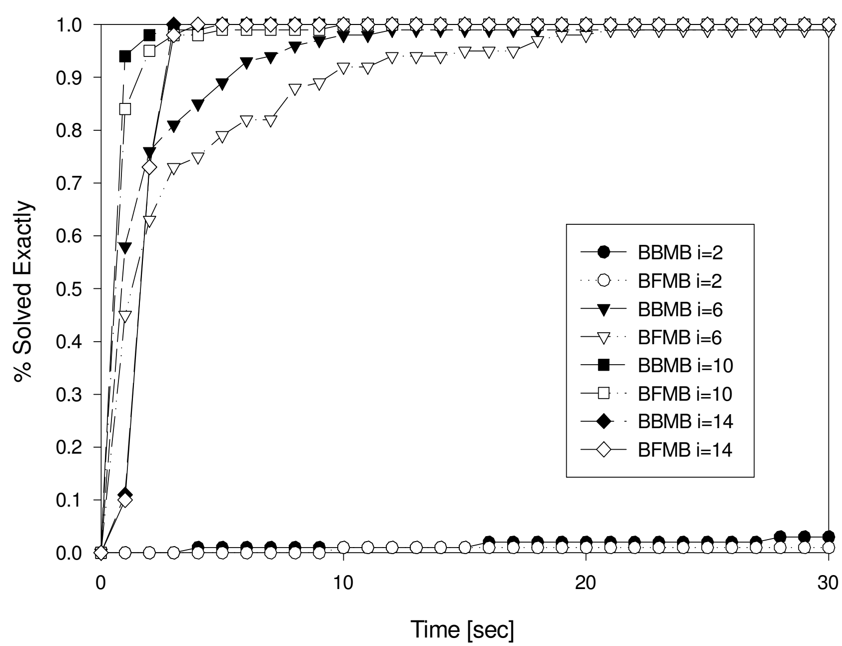


MCTE(i) computes h(x), at every Node for every uninstantiated variable.

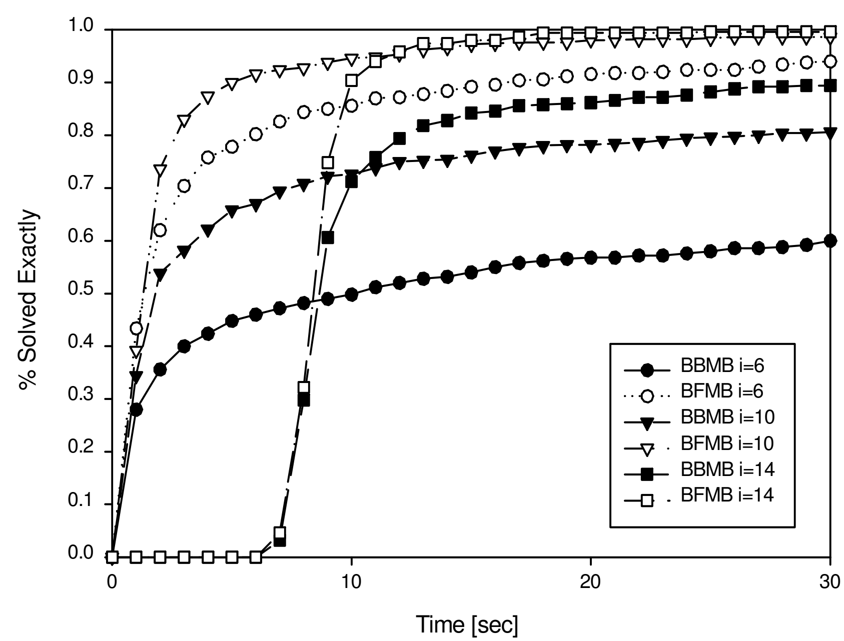
- Branch-and-Bound search where MCTE(i) is executed at each visited node
  - Domain pruning
  - Dynamic variable ordering
  - Dynamic value ordering

Empirical Evaluation of mini-bucket heuristics

Random Coding, K=100, noise=0.28



Random Coding, K=100, noise=0.32



Real World Benchmarks

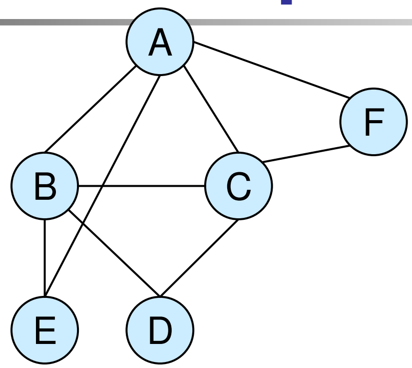
Network	# vars	avg. dom.	max dom.	BBBT/ BBMB/ IJGP i=2 %[time]	BBBT/ BBMB/ IJGP i=4 %[time]	BBBT/ BBMB/ IJGP i=6 %[time]	BBBT/ BBMB/ IJGP i=8 %[time]	GLS % [time]	DLM % [time]	SLS % [time]
Mildew	35	17	100	**100[0.28]** 30[10.5] 90[3.59]	**100[0.56]** 95[0.18] 97[33.3]	- - -	- - -	15 [30.02]	0 [30.02]	90 [30.02]
Munin2	1003	5	21	95[1.65] 95[30.3] 95[2.44]	95[1.65] 95[30.5] 95[5.17]	95[2.32] 95[31.3] 95[64.9]	**100[1.97]** **100[1.84]** -	0 [30.01]	0 [30.01]	0 [30.01]
Pigs	441	3	3	**90[15.2]** 0[30.01] 80[0.31]	**100[3.73]** 60[4.85] 77[0.53]	**100[2.36]** 80[0.02] 80[1.43]	**100[0.58]** 95[0.04] 83[6.27]	10 [30.02]	0 [30.02]	0 [30.02]
CPCS360b	360	2	2	100[0.17] **100[0.04]** 100[10.6]	100[0.27] **100[0.03]** 100[10.5]	100[0.21] **100[0.03]** 100[9.82]	100[0.19] **100[0.03]** 100[8.59]	100 [30.02]	100 [30.02]	100 [30.02]

Average Accuracy and Time. 30 samples, 10 observations, 30 seconds

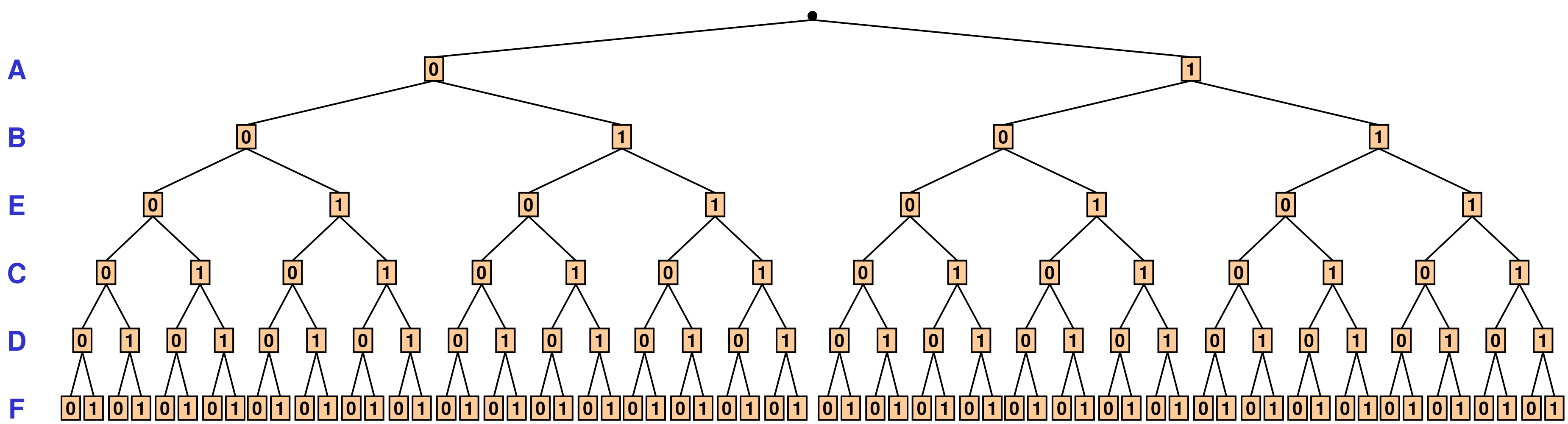
Overview

- **Introduction and background for graphical models: inference and search**
- **Inference:**
  - **Tree-clustering, variable elimination,**
  - **mini-bucket and mini-clustering,**
  - **Generalized belief propagation**
- **Hybrid of inference and search:** Heuristic generation and Brunch and Bound
- **AND/OR search spaces for graphical models**

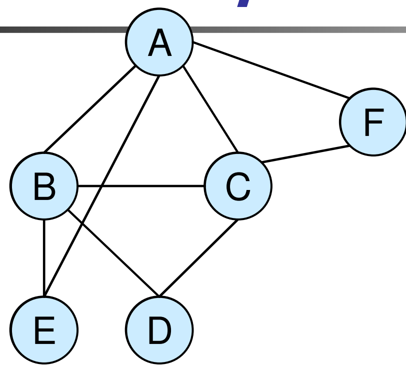
OR search space



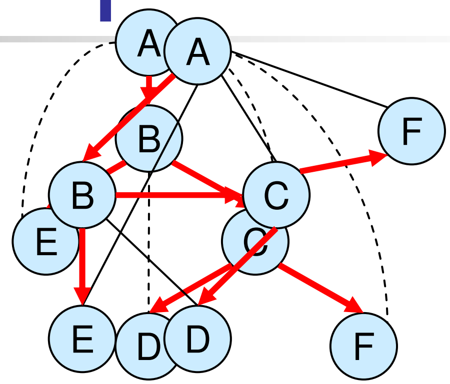
Ordering: A B E C D F



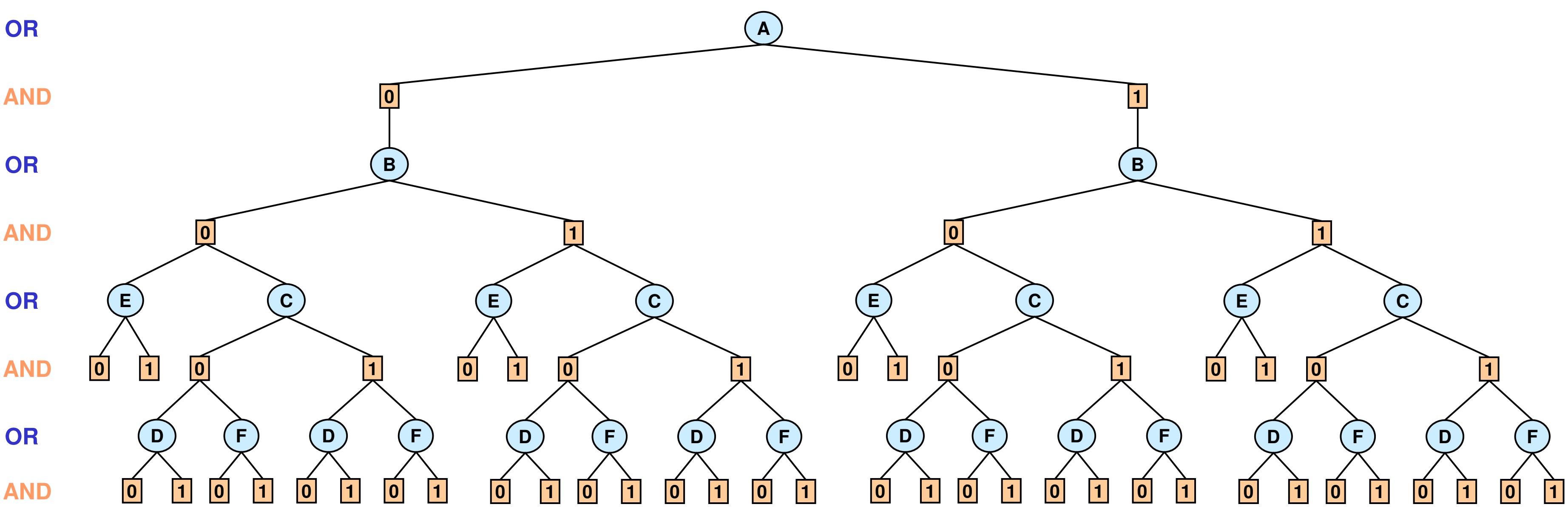
AND/OR search space



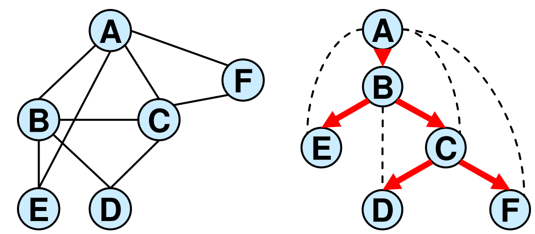
Primal graph

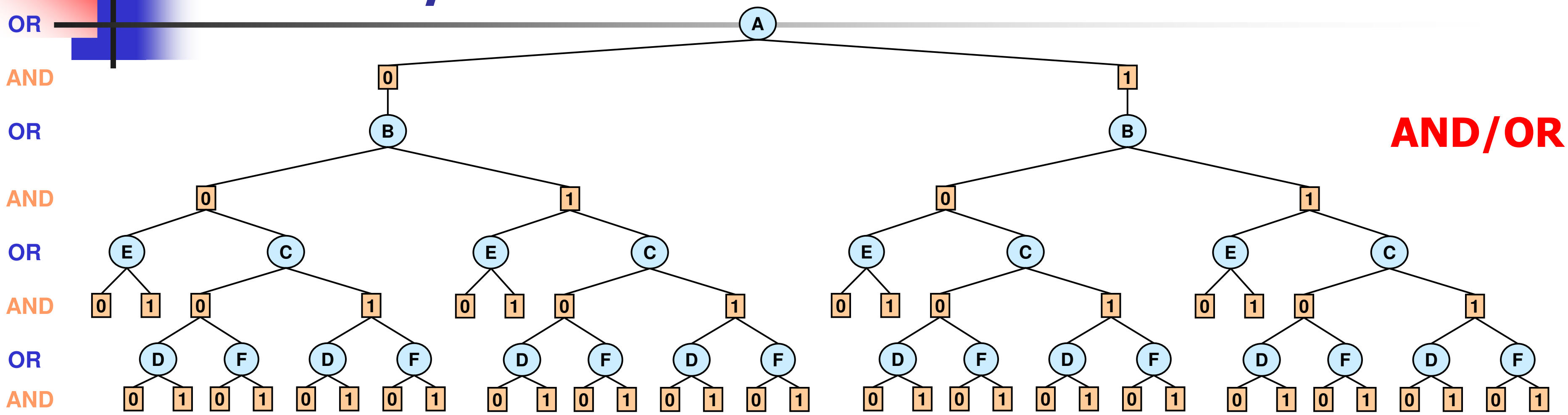


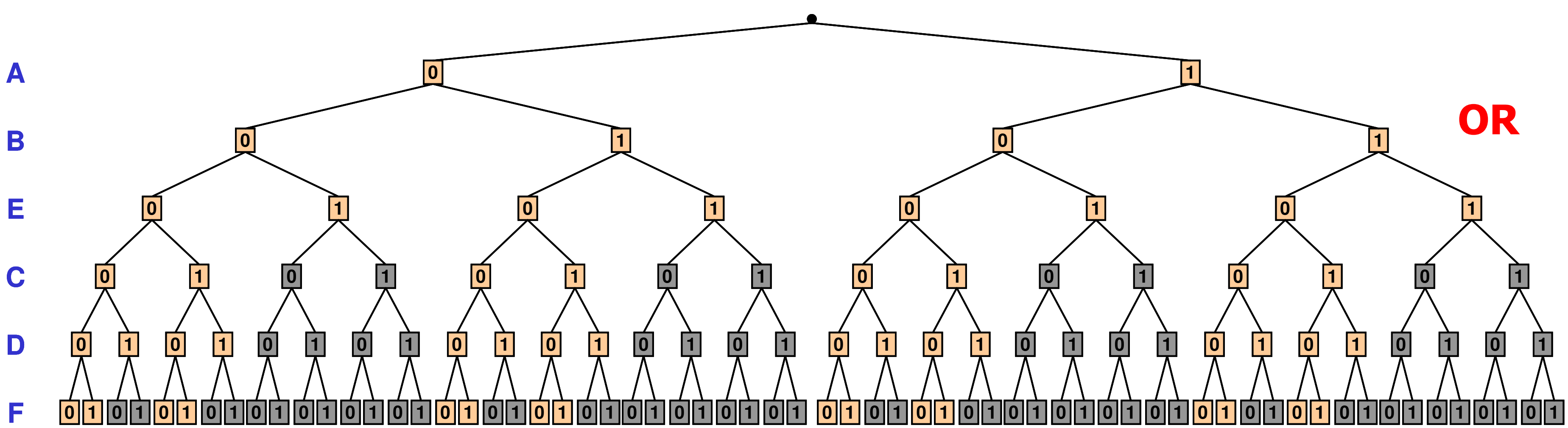
DFS tree



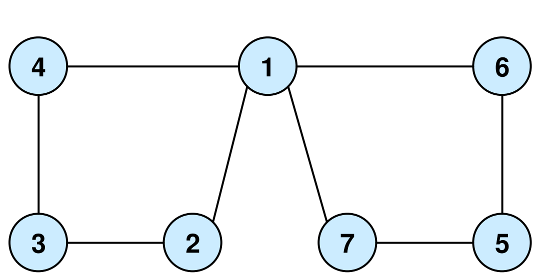
AND/OR vs. OR



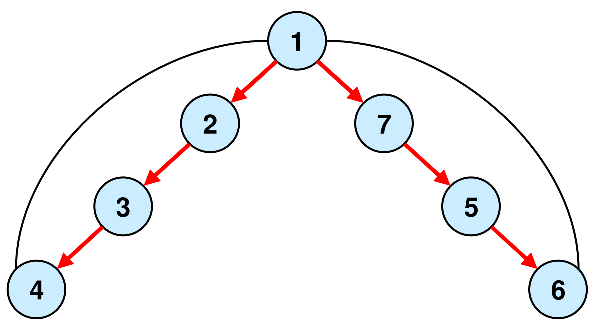




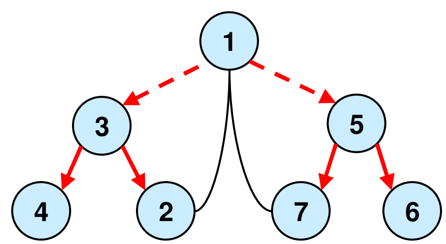
From DFS trees to pseudo-trees (Freuder 85, Bayardo 95)



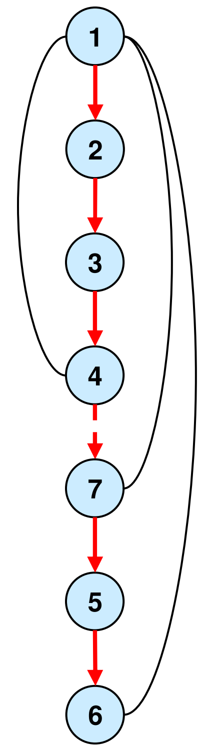
(a) Graph



(b) DFS tree
depth=3

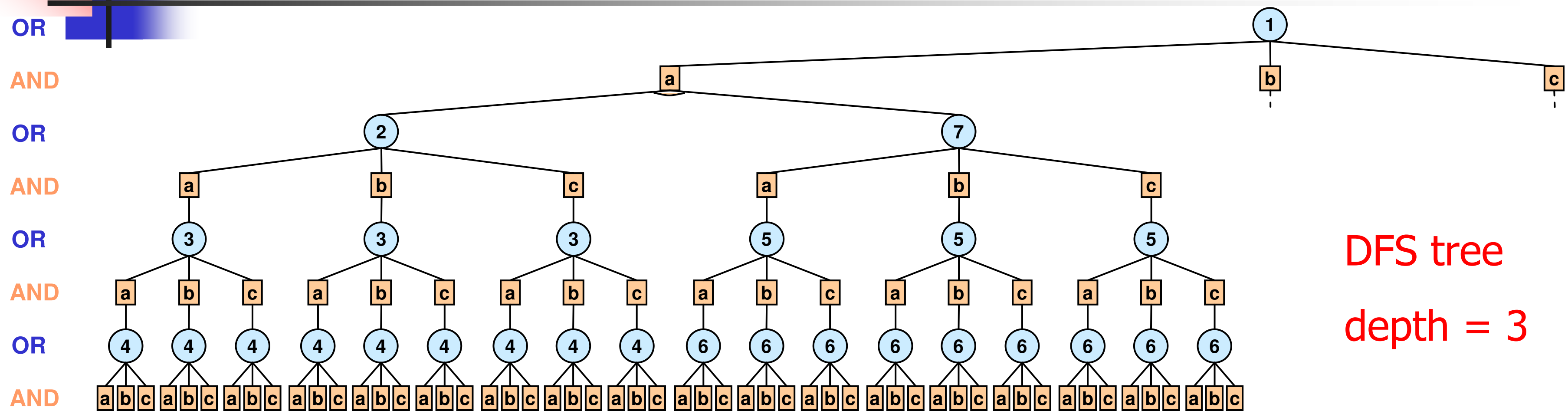


(c) pseudo- tree
depth=2



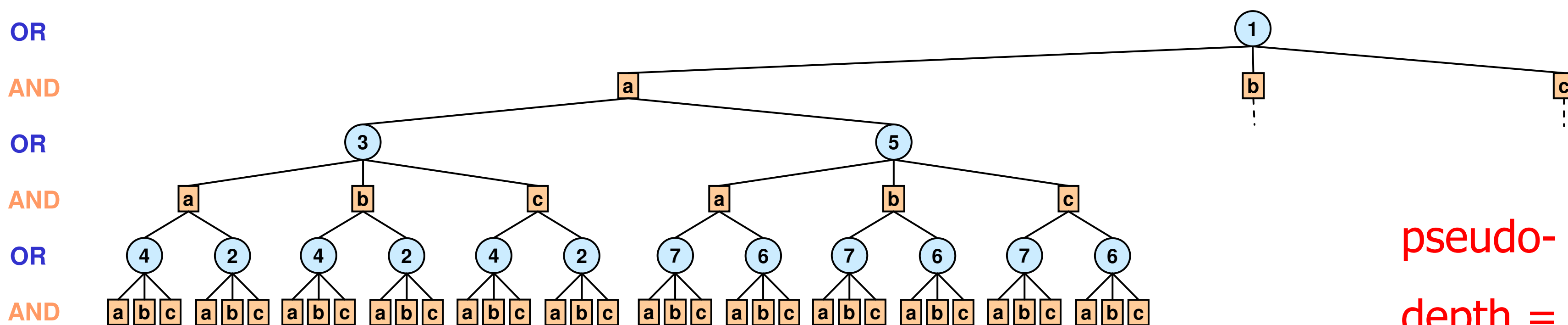
(d) Chain
depth=6

From DFS trees to Pseudo-trees



DFS tree

depth = 3

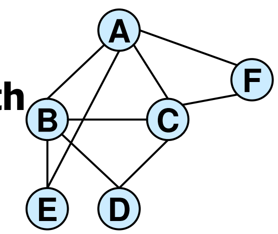


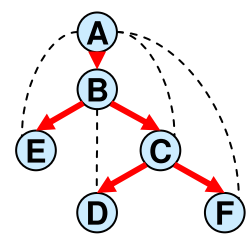
pseudo- tree

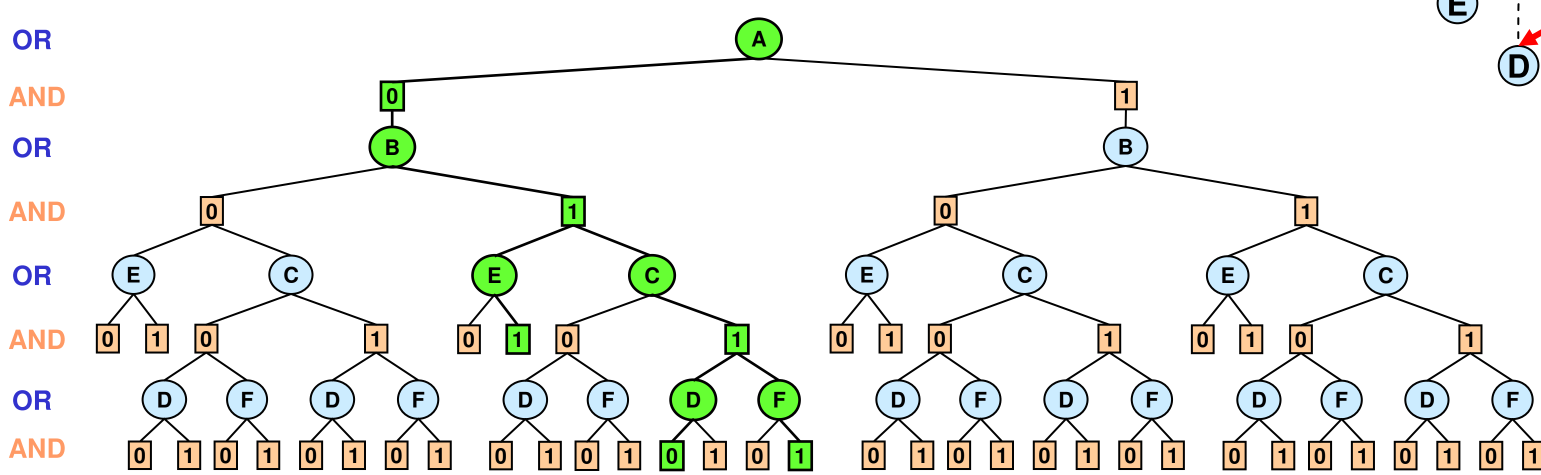
depth = 2

AND/OR search tree for graphical models

- **The AND/OR search tree of 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 (AND node)**
- **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**
  - **AND nodes have labels based on graphical model**
- **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 graphical model 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; Freuder 85; Bayardo 95)

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

Tasks

- **Consistency**: find a solved solution subtree
- **Counting**: find the number of solution subtrees
- **Optimization**: Find a solution subtree with min cost, max-csp
- **Belief updating**: find the belief of the root node (or evidence)
- **Query probability** (e.g. CNF) evaluation in belief or mixed networks
- **Partition function** computation
- **MPE:** find the solution tree with max product cost
- **Maximum expected utility** in influence diagrams

Tasks and value of nodes

- **The task defines a value function on nodes:**
  - **Given node n, that roots tree T_n**

  - **Consistency: value of n is 0 if T_n inconsistent, 1 othewise.**
  - **Counting: value of n is number of solutions in T_n**
  - **Optimization: Value of n is the optimal solution in T_n**
  - **Belief updating: value of n is the probability of evidence restricted to T_n, etc.**

- Task is accmplished when the value of the root node and/or is known.

- Value function can be computed recursively from leaves to root .

Minimal AND/OR search graph

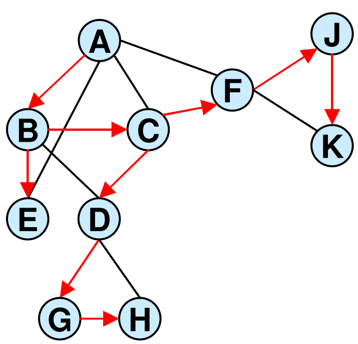
- Any two nodes that root identical subtrees/subgraphs can be merged
- Identical subgraphs can be identified based on context
- Minimal AND/OR search graph: closure under merge of the AND/OR search tree

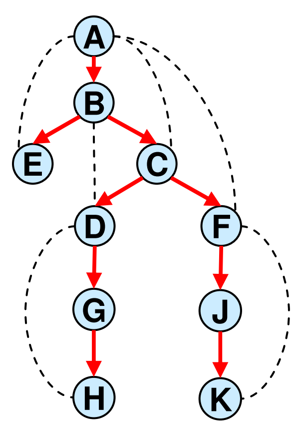
Context based caching

- Caching is possible when context is the same

- context = parent separator set
  = current variable +
  parents connected to subtree below

- context is bounded by tree-width w*





context(B) = {A, B}

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

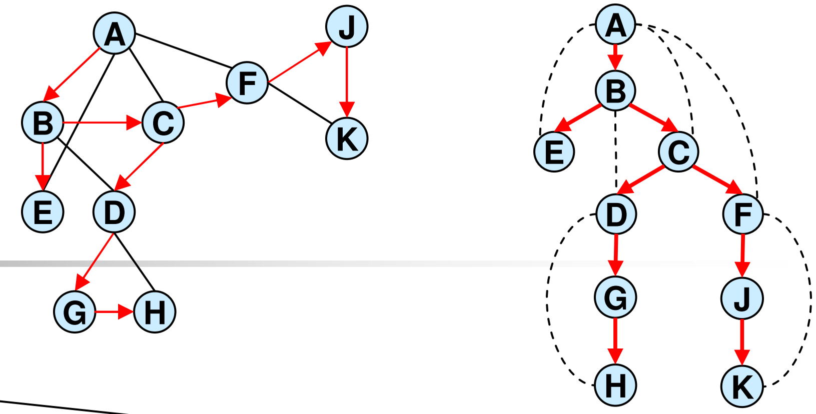
context(D) = {D}

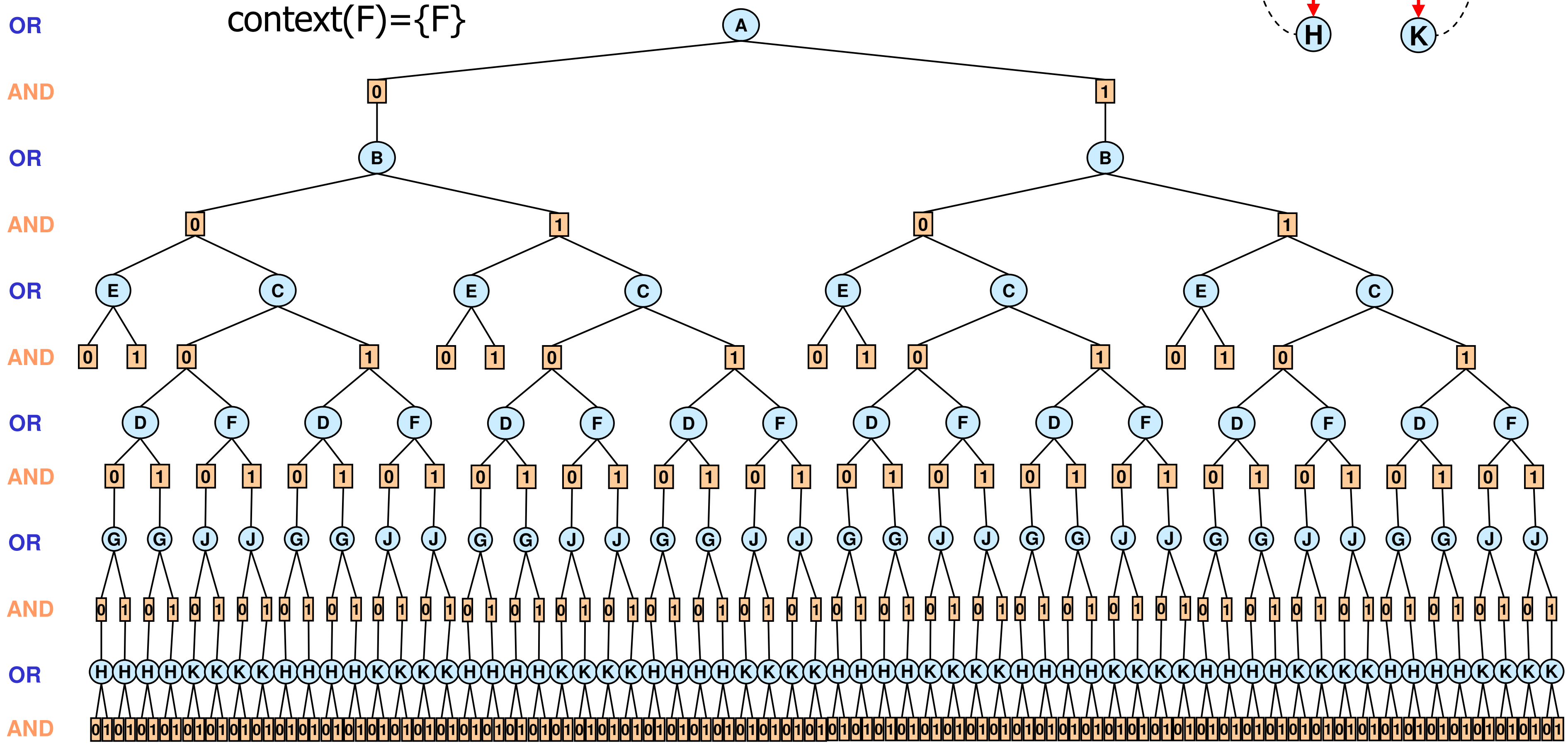
context(F) = {F}

Caching

context(D)={D}

context(F)={F}

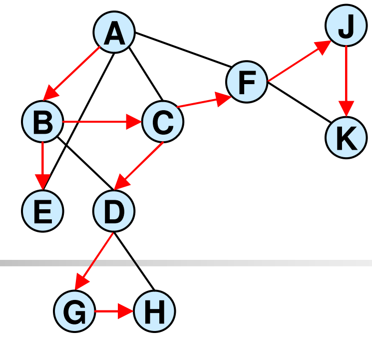


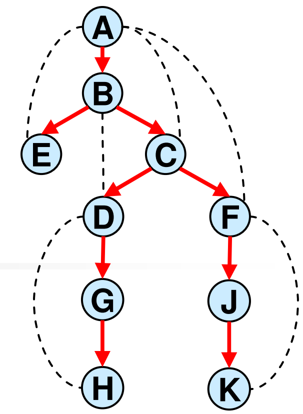


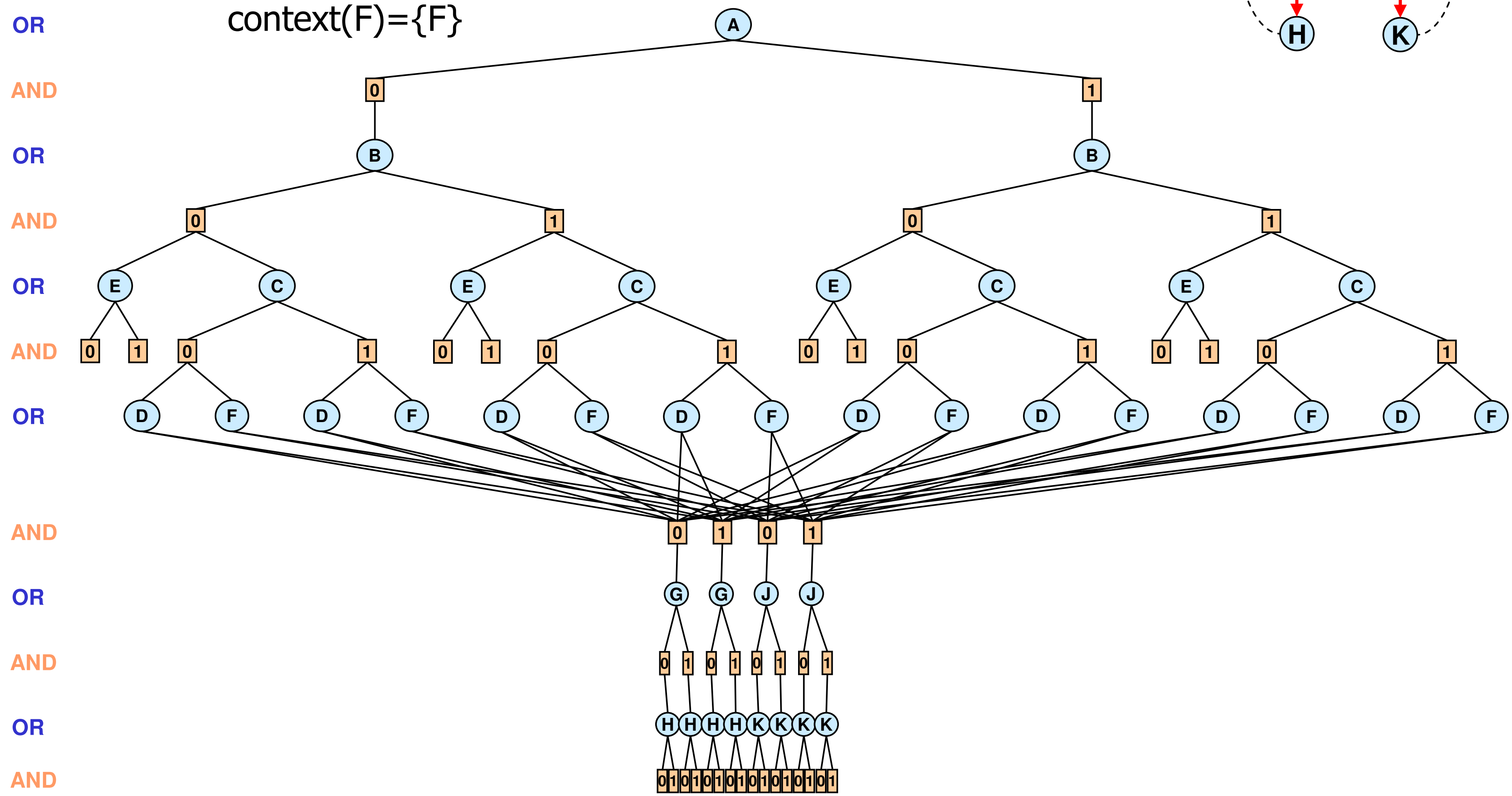
Caching

context(D)={D}

context(F)={F}







Size of minimal AND/OR search graphs

- Theorem:
  - Minimal AND/OR search graph is bounded exponentially by the tree-width w* of the graphical model
  - Minimal OR search graph is bounded exponentially by its path-width pw*

    ($w^* \leq pw^*$)

- Minimal OR – related to OBDDs
- Minimal AND/OR – related to tree-OBDDs

  (McMillan 94)

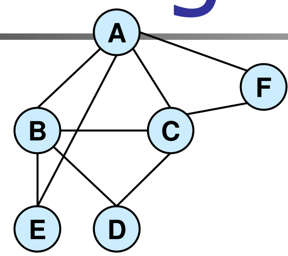
Outline

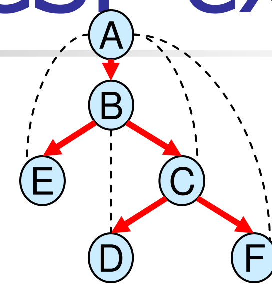
- AND/OR search spaces for graphical models:
  - AND/OR search trees
  - AND/OR search graphs (caching)
  - Algorithms
- Mixed networks
- Results for:
  - Counting solutions for CSP
  - Mixed networks

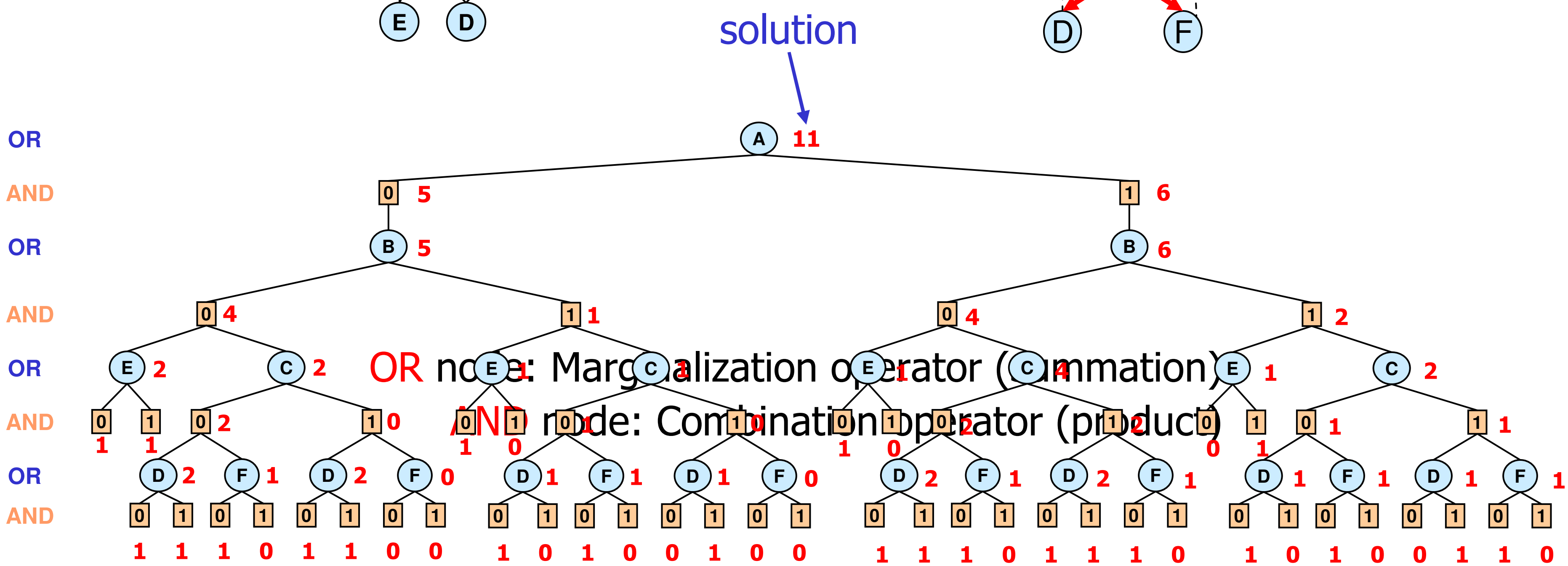
DFS algorithms for AND/OR spaces

- Solving a task involves:
  - *Combination* operator (e.g. multiplication, join) – AND
    Marginalization operator (summation, max, min, projection) – OR
- Each task defines a value function for nodes;
- Solution of task = value of root node
- Algorithm computes values of nodes in DFS manner

DFS algorithm (#CSP example)







OR node: Marginalization operator (summation)

AND node: Combination operator (product)

Value of node = number of solutions below it

AND/OR search algorithms

- AO(i)
  - i = the max size of a cache table (i.e. number of variables in a context)

i=0 → i=w*

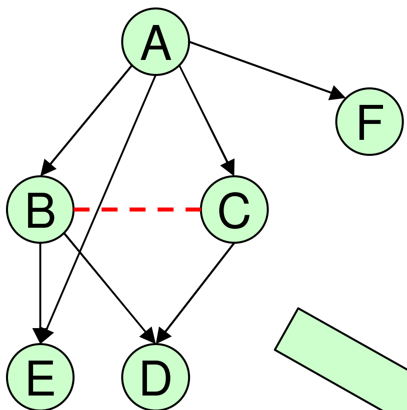
Space:	O(n)	Space:	O(exp w*)
Time:	O(exp(w* log n))	Time:	O(exp w*)

Outline

- AND/OR search spaces for graphical models:
  - AND/OR search trees
  - AND/OR search graphs (caching)
  - Algorithms
- Mixed networks
- Results for:
  - Counting solutions for CSP
  - Mixed networks

Mixed Networks

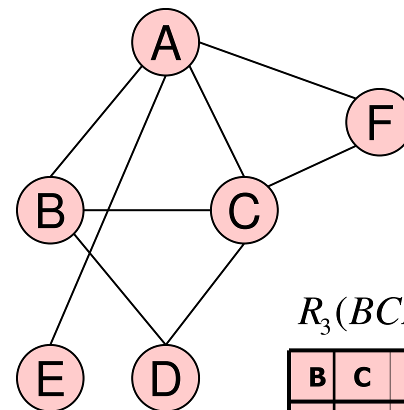
Belief Network



$P(D \mid B,C)$

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

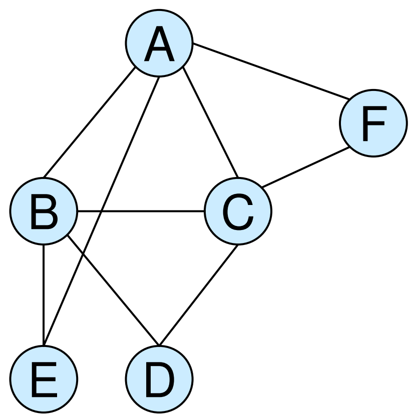
Constraint Network



$R_3(BCD)$

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

Moral mixed graph

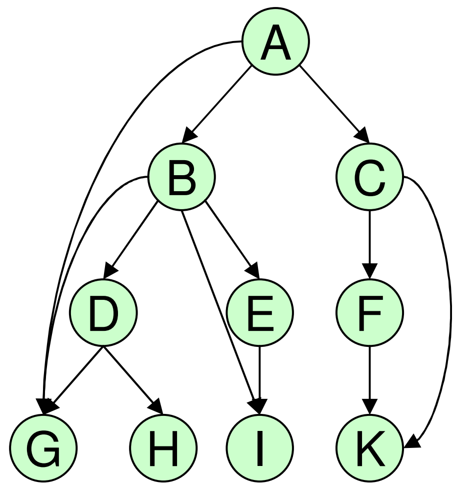


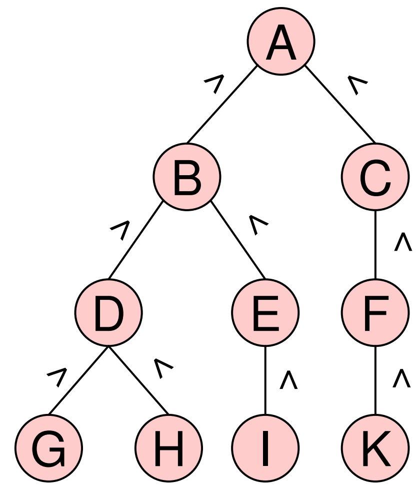
$$P_M(\bar{x}) = \begin{cases} P_B(\bar{x} \mid \bar{x} \in \rho) = \dfrac{P_B(\bar{x})}{P_B(\bar{x} \in \rho)}, \text{if } \bar{x} \in \rho \\ 0, \text{otherwise} \end{cases}$$

Benefits of mixed networks

- Constraint propagation techniques can be applied straightforwardly, maintaining their properties of convergence and fixed point
- The semantics is much clearer by separating probabilistic and deterministic information
- The algorithms can be made more efficient

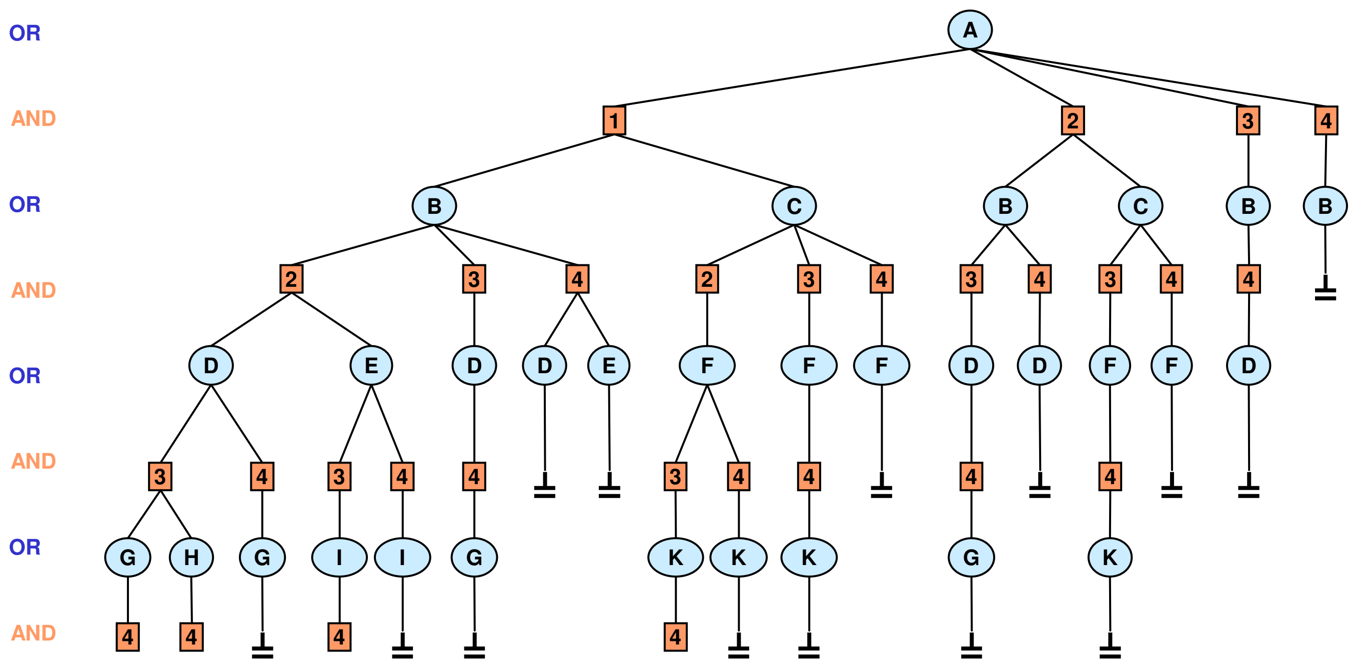
Using look-ahead – example (1)

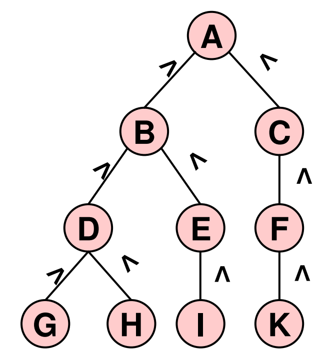




All domains are {1,2,3,4}

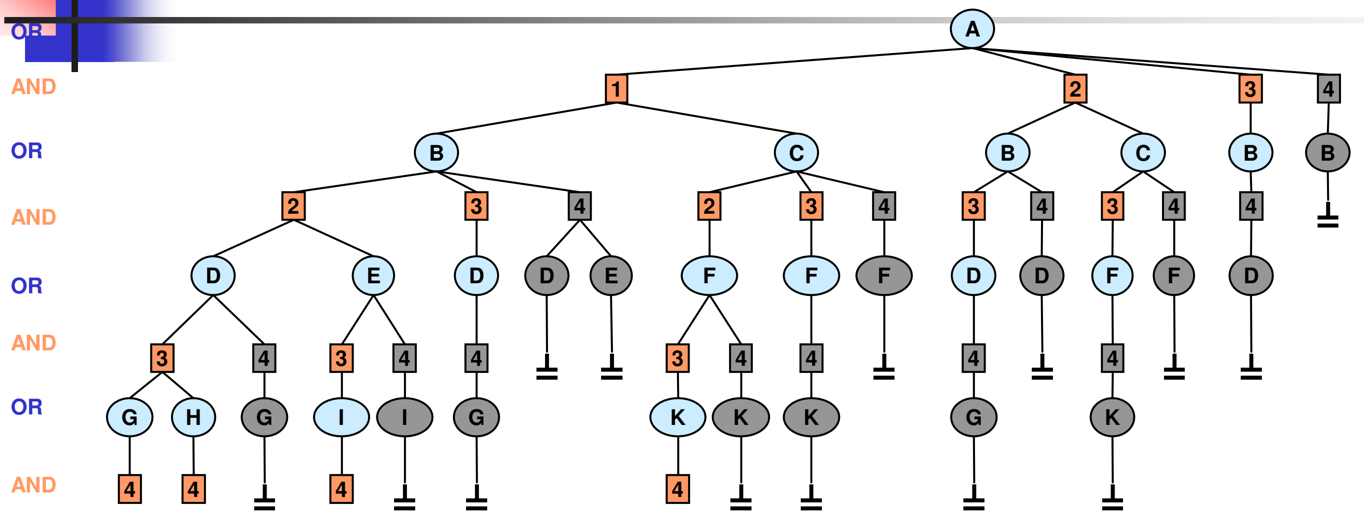
Using look-ahead – example (2) Constraint checking only



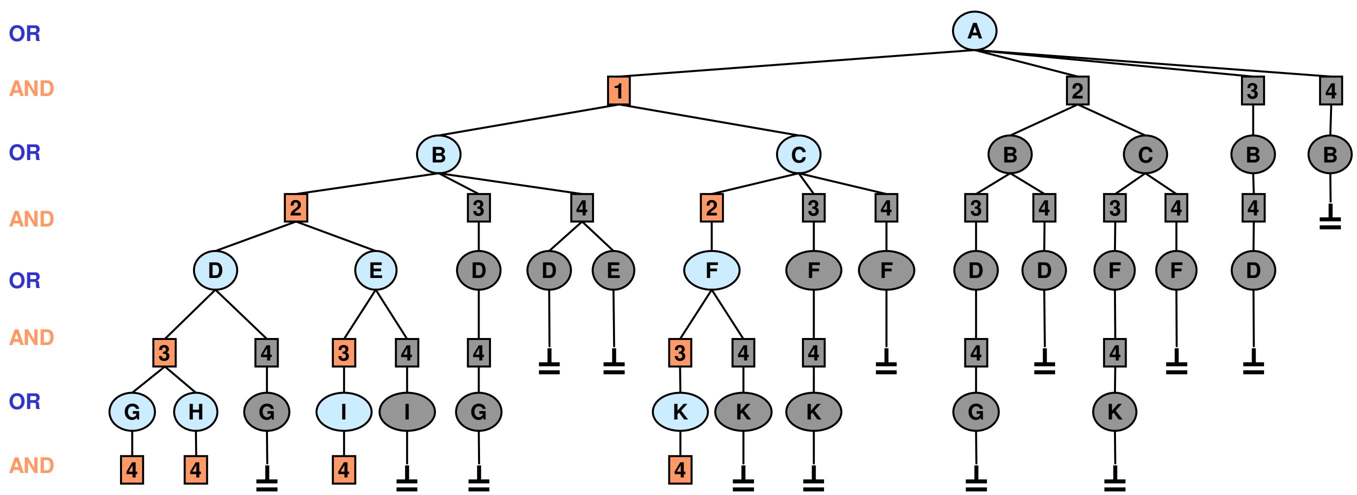


Domains are {1,2,3,4}

Using look-ahead – example (3)
Forward checking vs. Maintaining arc consistency



Forward Checking



Maintaining Arc Consistency

Outline

- AND/OR search spaces for graphical models:
  - AND/OR search trees
  - AND/OR search graphs (caching)
  - Algorithms
- Mixed networks
- Results for:
  - Counting solutions for CSP
  - Mixed networks

Experiments – counting CSP

- Constraint network - (N,K,C,S,t):
  - N = number of variables
  - K = number of values per variable
  - C = number of constraints
  - S = scope size of the constraints
  - t = tightness (no. of disallowed tuples per constraint)

- Measures:
  - Time
  - Number of nodes
  - Number of dead-ends

- Algorithms:
  - AO-FC   forward checking
  - AO-RFC   relational forward checking
  - OR-FC   forward checking
  - OR-RFC   relational forward checking

#CSP N40, K3, C50, P3, 20 inst, w*=13, depth=20
Time (seconds)

i	tightness	70%	60%	50\%	40%
	# solutions	0	0	46,582	147,898,575
i	BE	8.889	8.709	8.531	**8.637**
0	A/O FC	**0.110**	**0.454**	**3.129**	32.931
	OR FC	0.113	0.511	14.615	9737.823
3	A/O FC	0.111	**0.453**	**3.103**	31.277
	OR FC	0.112	0.509	14.474	9027.365
6	A/O FC	0.110	0.454	**3.006**	25.140
	OR FC	0.113	0.508	13.842	7293.472
9	A/O FC	0.114	0.453	**2.895**	21.558
	OR FC	0.111	0.509	12.336	5809.917
12	A/O FC	0.109	0.458	**2.703**	13.687
	OR FC	0.114	0.496	9.678	2598.778
13	A/O FC	0.111	0.457	**2.605**	**11.974**
	OR FC	0.123	0.494	8.703	1170.203

- AND/OR search is orders of magnitudes faster than OR search when problems are loose (consistent)
- AND/OR with full caching equivalent to BE: time and space O(exp w*)

#CSP N40, K2, 40, P3, 20 inst, w*=10, depth=17

tightness		60%	50%	40%	30%	20%	10%	0%
# solutions		0	0	13,533	2,414,724	190,430,000	21,549,650,000	1,099,511,627,776
Time (seconds)								
i=0	AO FC	0.005	0.011	0.065	0.289	1.931	7.979	30.094
i=3	AO FC	0.003	0.008	0.060	0.253	1.525	6.062	22.340
i=6	AO FC	0.003	0.009	0.052	0.182	0.883	2.873	8.847
i=9	AO FC	0.003	0.010	0.046	0.142	0.559	1.323	2.997
i=10	AO FC	**0.004**	**0.010**	**0.038**	**0.110**	**0.343**	**0.587**	**0.985**
i=10	OR FC	**0.004**	**0.012**	**0.671**	**24.912**	**1009.025**	**-**	**-**
Number of nodes								
i=0	AO FC	55	166	3,078	22,273	204,562	988,136	4,145,934
i=3	AO FC	55	155	1,503	8,747	57,778	236,466	870,866
i=6	AO FC	55	148	975	4,292	24,542	95,394	298,236
i=9	AO FC	55	142	922	3,364	15,235	38,088	90,101
i=10	AO FC	**55**	**135**	**746**	**2,365**	**8,646**	**15,050**	**25,717**
i=10	OR FC	**55**	**166**	**14,049**	**635,331**	**25,078,186**	**-**	**-**
Number of deadends								
i=0	AO FC	57	162	1,978	10,298	57,678	134,324	0
i=3	AO FC	57	159	1,662	8,569	45,336	92,263	0
i=6	AO FC	56	149	974	3,721	13,655	19,257	0
i=9	AO FC	55	139	641	1,850	4,517	5,372	0
i=10	AO FC	**55**	**125**	**533**	**1,312**	**2,313**	**1,887**	**0**
i=10	OR FC	**57**	**164**	**9,693**	**299,138**	**11,541,863**	**-**	**-**

#CSP N40, K2, 40, P3, 20 inst, w*=10, depth=17

tightness		60%	50%	40%	30%	20%	10%	0%
# solutions		0	0	13,533	2,414,724	190,430,000	21,549,650,000	1,099,511,627,776
Time (seconds)								
i=0	**AO FC**	0.005	0.011	0.065	0.289	1.931	**7.979**	**30.094**
i=3	**AO FC**	0.003	0.008	0.060	0.253	1.525	**6.062**	**22.340**
i=6	**AO FC**	0.003	0.009	0.052	0.182	0.883	**2.873**	**8.847**
i=9	**AO FC**	0.003	0.010	0.046	0.142	0.559	**1.323**	**2.997**
i=10	**AO FC**	0.004	0.010	0.038	0.110	0.343	**0.587**	**0.985**
i=10	**OR FC**	0.004	0.012	0.671	24.912	1009.025	-	-
Number of nodes								
i=0	**AO FC**	55	166	3,078	22,273	204,562	988,136	4,145,934
i=3	**AO FC**	55	155	1,503	8,747	57,778	236,466	870,866
i=6	**AO FC**	55	148	975	4,292	24,542	95,394	298,236
i=9	**AO FC**	55	142	922	3,364	15,235	38,088	90,101
i=10	**AO FC**	55	135	746	2,365	8,646	15,050	25,717
i=10	**OR FC**	55	166	14,049	635,331	25,078,186	-	-
Number of deadends								
i=0	**AO FC**	57	162	1,978	10,298	57,678	134,324	0
i=3	**AO FC**	57	159	1,662	8,569	45,336	92,263	0
i=6	**AO FC**	56	149	974	3,721	13,655	19,257	0
i=9	**AO FC**	55	139	641	1,850	4,517	5,372	0
i=10	**AO FC**	55	125	533	1,312	2,313	1,887	0
i=10	**OR FC**	57	164	9,693	299,138	11,541,863	-	-

Mixed networks results (1)

N=40, K=2, R=2, P=2, C=10, S=4, 20 instances, w*=12, depth=19											
tightness	i	Time			Nodes			Dead-ends			#sol
		AO-C	AO-FC	AO-RFC	AO-C	AO-FC	AO-RFC	AO-C	AO-FC	AO-RFC	
80%	0	0.671	0.056	0.022	153,073	4,388	1,066	95,197	3,299	962	1.6E+05
	6	0.479	0.055	0.022	75,397	3,213	936	57,306	3,168	940	
	12	0.103	0.044	**0.016**	16,579	2,273	683	2,638	1,537	398	
60%	0	2.877	0.791	1.094	774,697	167,921	158,007	239,991	40,069	36,119	7.7E+07
	6	1.409	0.445	0.544	183,286	35,325	31,607	107,362	27,575	24,153	
	12	0.189	**0.142**	0.149	27,848	9,148	7,357	3,343	3,997	3,048	
40%	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
	6	2.809	2.219	3.149	346,842	183,895	180,463	150,864	88,822	85,522	
	12	**0.255**	**0.331**	0.425	36,262	23,160	22,293	2,825	5,083	4,658	
20%	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
	6	5.305	6.286	9.061	626,405	519,258	518,029	127,683	98,100	96,872	
	12	**0.318**	0.543	0.714	44,340	39,550	39,524	1,431	2,647	2,659	
0%	0	23.595	27.129	41.744	7,450,537	7,450,537	7,450,537	0	0	0	1.1E+12
	6	8.325	11.528	16.636	956,965	956,965	956,965	0	0	0	
	12	**0.366**	0.681	0.884	50,616	50,616	50,616	0	0	0	

- Caching helps more on loose problems
- Constraint propagation helps more on tight problems

Mixed networks results(2)

N=100, K=2, R=10, P=2, C=30, S=3, 20 instances, w*=28, depth=38								
tightness	i	Time		Nodes		Dead-ends		#sol
		AO-FC	AO-RFC	AO-FC	AO-RFC	AO-FC	AO-RFC	
90%	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	
80%	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	
70%	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, depth=51								
tightness	i	Time		Nodes		Dead-ends		#sol
		AO-FC	AO-RFC	AO-FC	AO-RFC	AO-FC	AO-RFC	
90%	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	
80%	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	
70%	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	

- Large and tight problems are solvable by search, but they are infeasible for Junction Tree algorithms

Mixed networks results:
AND/OR Search vs. Bucket Elimination (Time)

N=70, K=2, R=5, P=2, C=30, S=3, 20 instances, w*=23, depth=31					
tightness	i	Time			#sol
		BE	AO-FC	AO-RFC	
60%	0	**26.400**	1.956	1.263	0
	10		1.872	**1.231**	
	20		1.878	1.291	
50%	0		30.652	35.570	1.64E+12
	10		18.583	18.872	
	20		12.444	**12.110**	
40%	0		396.754	511.434	7.02E+14
	10		167.852	182.451	
	20		**80.484**	83.601	

N=60, K=2, R=5, P=2, C=40, S=3, 20 instances, w*=23, depth=31					
tightness	i	Time			#sol
		BE	AO-FC	AO-RFC	
60%	0	**66.871**	0.655	0.603	0
	10		0.630	0.568	
	20		0.632	**0.566**	
50%	0		3.178	3.021	6.24E+04
	10		2.993	2.794	
	20		2.731	**2.578**	
40%	0		65.171	70.242	7.51E+08
	10		54.101	56.419	
	20		**39.606**	40.718	

- AND/OR search better than BE on tight problems
- AND/OR search with full caching similar to BE with constraint propagation

Conclusion

- AND/OR search spaces are a unifying framework applicable to a wide range of graphical models
- AND/OR search spaces exploit problem structure
- AND/OR time and space bounds are equal to state of the art algorithms
- Mixed networks combine belief and constraint networks borrowing specific strengths from both formalisms
- Empirical results
  - AND/OR search spaces are always more effective than traditional OR spaces
  - AND/OR allows a flexible tradeoff between space and time

Experiments

- Counting: CSPs
- CPE (cnf probability): Mixed networks
- **Optimization: Max-csps/mpe**

Optimization Tasks
(Mateescu and Dechter, 2004)

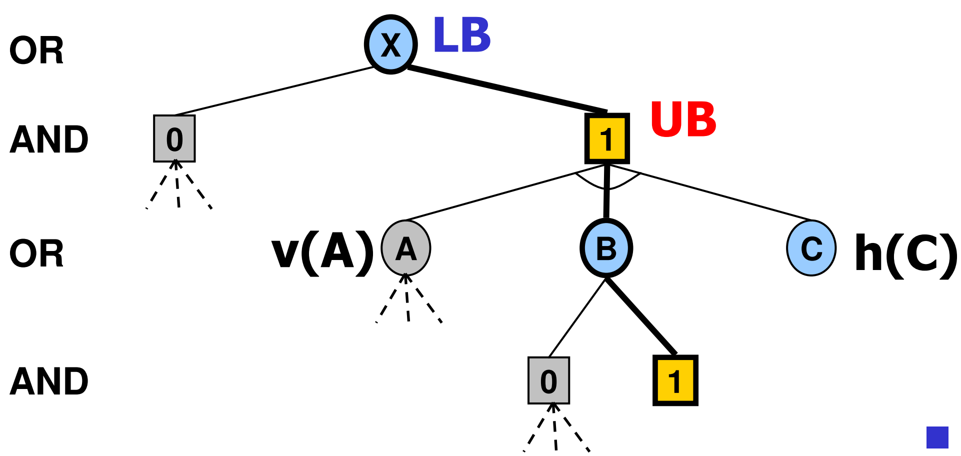
- MPE/MAP in Belief Networks
  - Find a maximum probability assignment to all/subset variables, given the evidence.
  - E.g. Medical diagnosis, speech recognition, coding

- Max-CSP in Constraint Networks
  - Find a complete assignment that violates the least number of constraints.
  - E.g. Over-constrained networks with no feasible solution

Classic BnB Search w/ MB(i) Heuristics

- Heuristic function **h(n)** computed using parameterized mini-bucket (MB(i)) approximation of variable elimination schemes, derived automatically from the problem's specification.
  - **Static BBMB(i)** – uses precompiled MB(i) functions
    - **[Kask and Dechter, 1999]**
  - **Dynamic BBMB(i)** – uses MB(i) at each node
    - **[Marinescu and Dechter, 2004]**
  - **Dynamic BBBT(i)** class – uses MBTE at each node
    - **[Kask, Dechter, Larrosa, 2001], [Kask, Marinescu, Dechter, 2003]**
- Competitive and in many cases superior to state of the art algorithms for solving Bayesian MPE/Max-CSP

AND/OR Branch and Bound Search



Partial solution subtree

- **OR** level
  - **lower bound on value v(n)**
  - Current best value backed up by the AND child nodes
- **AND** level
  - **upper bound on value v(n)**
  - Recursively computed using the current values/estimates of the OR child nodes

Prune if UB ≤ LB

AND/OR BnB w/ MB Heuristics

- **h(n)** is computed using MB(i) scheme:
  - Static AOMB class
    - Precompiled MB heuristics (similar to static-BBMB)
  - Dynamic AOMB class
    - Dynamically computes the MB heuristics (extends dynamic-BBMB)
    - The heuristics are computed relative to smaller independent sub-problems – this is more efficient!
- Any other heuristic function is suitable!

Bayesian MPE – Random Networks

Random Networks, 180 second time limit

s-AOMB d-AOMB s-BBMB d-BBMB BBBT i=2 % / time / nodes	s-AOMB d-AOMB s-BBMB d-BBMB BBBT i=4 % / time / nodes	s-AOMB d-AOMB s-BBMB d-BBMB BBBT i=6 % / time / nodes	s-AOMB d-AOMB s-BBMB d-BBMB BBBT i=8 % / time / nodes	s-AOMB d-AOMB s-BBMB d-BBMB BBBT i=10 % / time / nodes	s-AOMB d-AOMB s-BBMB d-BBMB BBBT i=12 % / time / nodes
N=100, K=2, P=2, C=90, w*=16.3, H=25.8					
58 / 121.6 / 4M	95 / 34.07 / 1.1M	100 / 8.659 / 269K	100 / 2.368 / 72.5K	100 / 0.776 / 26.5K	100 / 0.188 / 6.3K
100 / 9.583 / 29.3K	100 / 0.992 / 1.7K	100 / 0.520 / 731	100 / 0.365 / 381	100 / 0.382 / 225	100 / 0.531 / 181
1 / 179.9 / 9M	57 / 108.9 / 6.4M	96 / 24.19 / 1.6M	99 / 5.54 / 363K	100 / 2.606 / 179K	100 / 0.34 / 25K
70 / 85.04 / 229K	100 / 2.045 / 2.8K	100 / 0.616 / 698	100 / 0.467 / 387	100 / 0.465 / 229	100 / 0.653 / 191
41 / 136.2 / 29.4K	98 / 31.71 / 6.4K	100 / 6.354 / 1,058	100 / 1.848 / 245	100 / 1.474 / 135	100 / 1.623 / 112

- **AOMB** variants dominate **BBMB** variants for all reported i-bounds
  - Dynamic-AOMB dominates for **small** i-bounds
  - Static-AOMB dominates for **large** i-bounds

Bayesian MPE – Random Grids

s-AOMB d-AOMB s-BBMB d-BBMB **i=4** % / time / nodes	s-AOMB d-AOMB s-BBMB d-BBMB **i=6** % / time / nodes	s-AOMB d-AOMB s-BBMB d-BBMB **i=8** % / time / nodes	s-AOMB d-AOMB s-BBMB d-BBMB **i=10** % / time / nodes
Grid 9x9, K=3, w*=12, H=37, 10 observations			
100 / 139.4 / 6M **100 / 43.05 / 76K**	**100 / 1.23 / 61K** 100 / 1.93 / 546	**100 / 0.05 / 963** 100 / 0.83 / 151	**100 / 0.11 / 174** 100 / 0.79 / 91
20 / 261.5 / 17M 80 / 91.97 / 166K	95 / 27.95 / 1.8M 100 / 2.13 / 755	100 / 0.52 / 32K 100 / 0.84 / 158	**100 / 0.11 / 195** 100 / 0.84 / 92

Random Grid Networks, 300 second time limit

Bayesian MPE – Real World Networks

Network	w*	H	s-AOMB d-AOMB s-BBMB d-BBMB **i=2** time / nodes	s-AOMB d-AOMB s-BBMB d-BBMB **i=3** time / nodes	s-AOMB d-AOMB s-BBMB d-BBMB **i=4** time / nodes	s-AOMB d-AOMB s-BBMB d-BBMB **i=5** time / nodes
Barley (48,8,67)	7	17	- / 5.1M - / 281.6K - / 12.8M - / 2.1M	- / 10.7M **144.9 / 39.9K** - / 9M - / 829.5K	- / 11.4M **13.60 / 976** - / 6.8M 34.75 / 5.7K	266.1 / 5.4M **49.11 / 639** 541.9 / 7.3M - / 56.8K
Munin1 (189,5,21)	11	24	292.3 / 3.3M **78.31 / 223K** - / 2.9M - / 311K	39.27 / 480K **15.42 / 9.7K** - / 3.2M - / 327K	17.10 / 255K **12.49 / 2.3K** - / 4.3M - / 185K	**3.768 / 62.2K** 14.51 / 802 - / 3.7M 14.90 / 804
Munin3 (1044,5,21)	7	25	- / 2.9M - / 2.3M - / 371K - / 25.2K	- / 3.1M **91.5 / 62.6K** - / 405K - / 82.4K	5.844 / 53.8K **4.578 / 5.9K** - / 172K - / 38.3K	**0.64 / 6.8K** 3.515 / 3.8K - / 432K - / 23.7K

Bayesian Network Repository, 600 second time limit

Max-CSP – Random Networks

s-AOMB d-AOMB s-BBMB d-BBMB **i=2** % / time / nodes	s-AOMB d-AOMB s-BBMB d-BBMB **i=4** % / time / nodes	s-AOMB d-AOMB s-BBMB d-BBMB **i=6** % / time / nodes	s-AOMB d-AOMB s-BBMB d-BBMB **i=8** % / time / nodes	PFC-RDAC PFC-MRDAC PFC-MPRDAC
N=50, K=5, C=80, T=60%, w*=7.75, H=15.5				
70 / 88.65 / 1.2M **100 / 4.17 / 10.7K**	100 / 3.093 / 76K **100 / 0.791 / 250**	**100 / 0.131 / 2.4K** 100 / 0.838 / 80	**100 / 0.731 / 87** 100 / 1.717 / 53	100 / 3.142 / 227K **100 / 1.849 / 92K** 100 / 2.307 / 92K
0 / 180 / 6.5M 75 / 74.32 / 444K	75 / 68.19 / 2.8M 100 / 1.788 / 1.2K	100 / 2.743 / 146K 100 / 0.634 / 80	100 / 0.744 / 1.1K 100 / 1.673 / 50	

Random Networks, 180 second time limit

Overview/Conclusions

- Introduction and background for graphical models: inference and search
- Bounded inference: mini-bucket and mini-clustering, Generalized belief propagation
- Hybrid of inference and search**:** Heuristic generation and Brunch and Bound
- AND/OR search spaces for graphical models
- **Conclusions: Graphical models should always use AND/OR search with embedded inference**.