Reasoning with Bayesian Networks

Rina Dechter
Donald Bren School of Computer Science
University of California, Irvine, USA

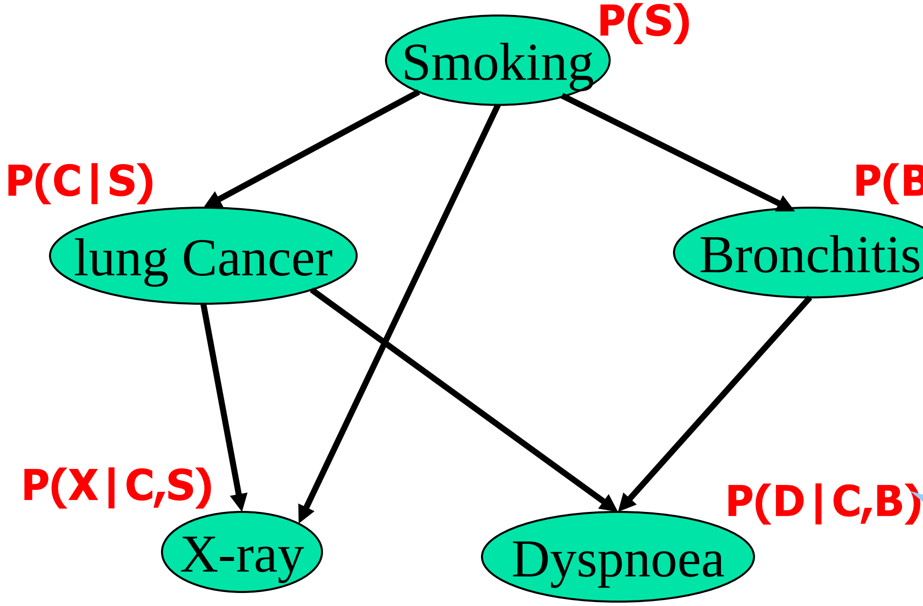
Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
- Search
- Sampling
- Hybrid of search and inference
- Modeling and learning
- Software

Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
- Search
- Sampling
- Hybrid of search and inference
- Modeling and learning
- Software

Bayesian Networks (Pearl, 1988)



BN = (G, Θ)

CPD:

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

P(S, C, B, X, D) = ***P(S) P(C|S) P(B|S) P(X|C,S) P(D|C,B)***

Belief Updating:

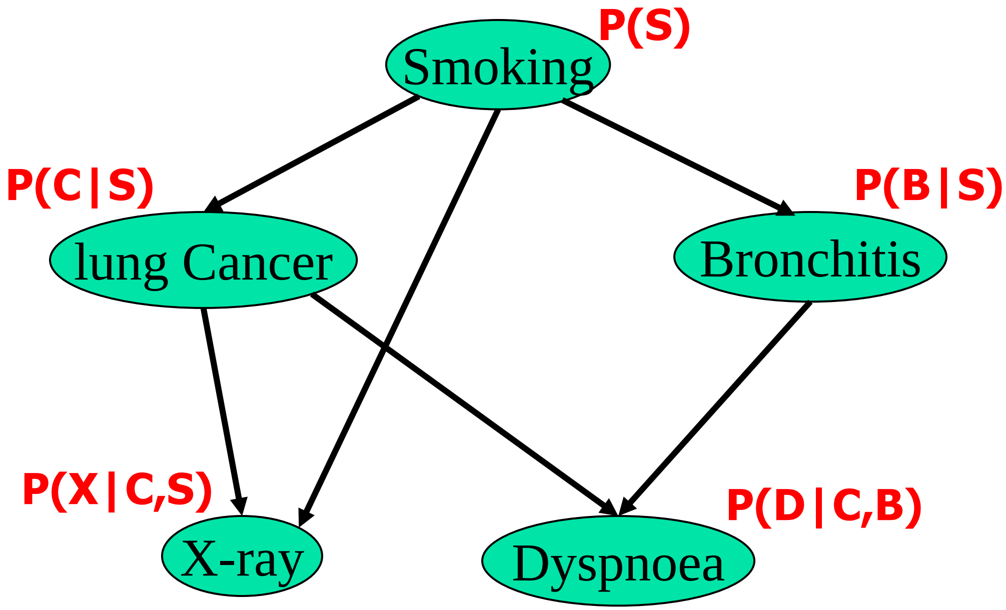
P (lung cancer=yes | smoking=no, dyspnoea=yes) = ?

Most likely explanation (MPE):

MPE = find argmax P(S)· P(C|S)· P(B|S)· P(X|C,S)· P(D|C,B)

Bayesian Networks encode independencies

Causal relationship

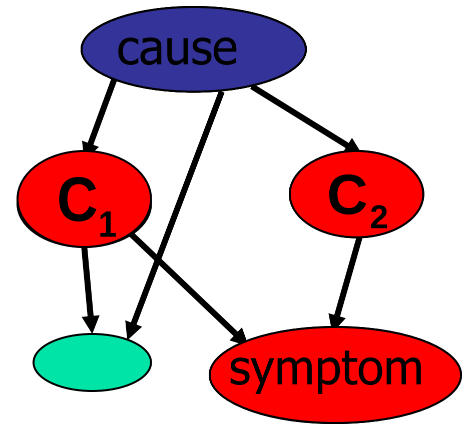


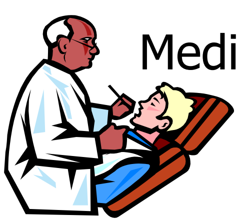
$$\mathbf{BN} = (\mathbf{G}, \mathbf{\Theta})$$

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

What are they good for?

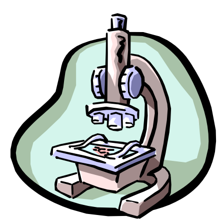
- Diagnosis: P(cause|symptom)=?
- Prediction: P(symptom|cause)=?
- Classification: $\max_{class} P(class|data)$
- Decision-making (given a cost function)





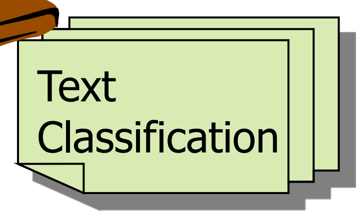
Medicine

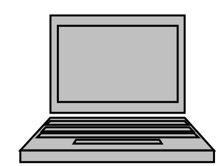
Speech recognition



Bio-informatics

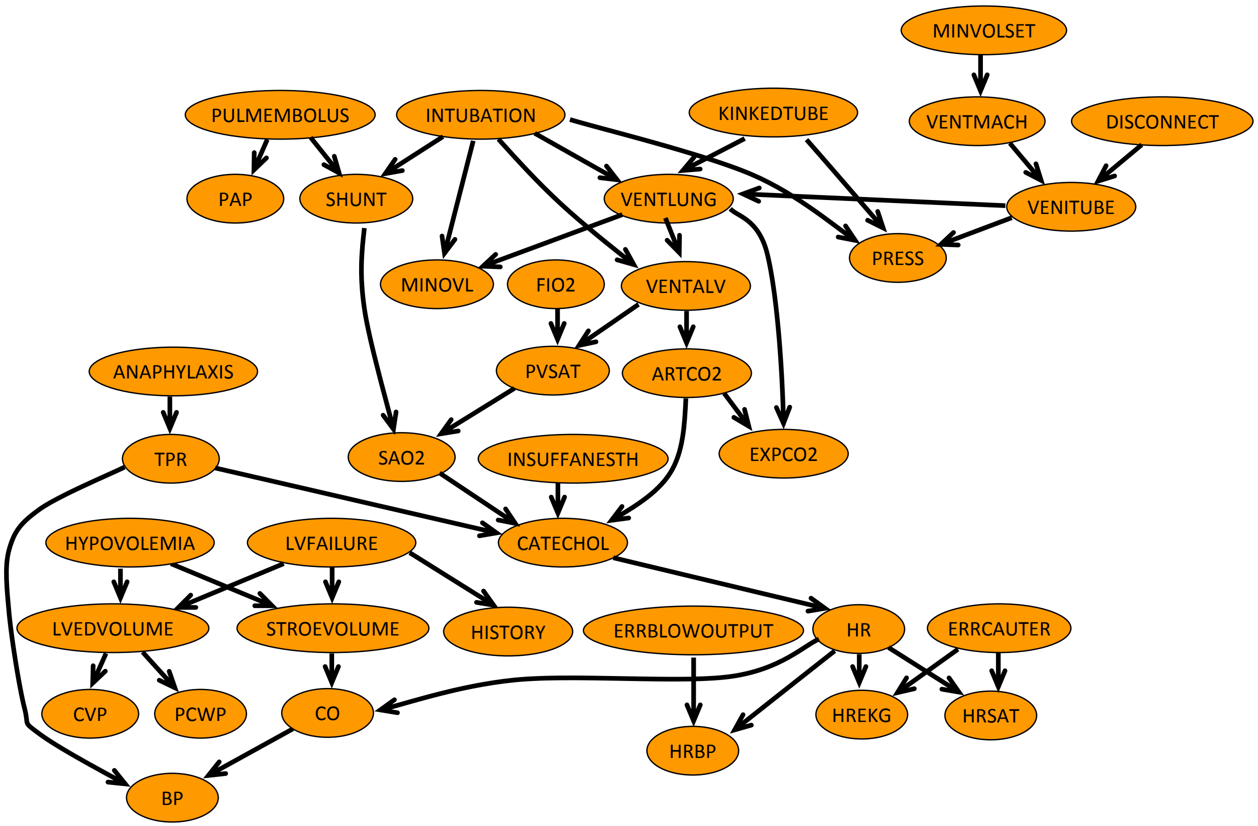
Stock market





Computer troubleshooting

Monitoring Intensive-Care Patients

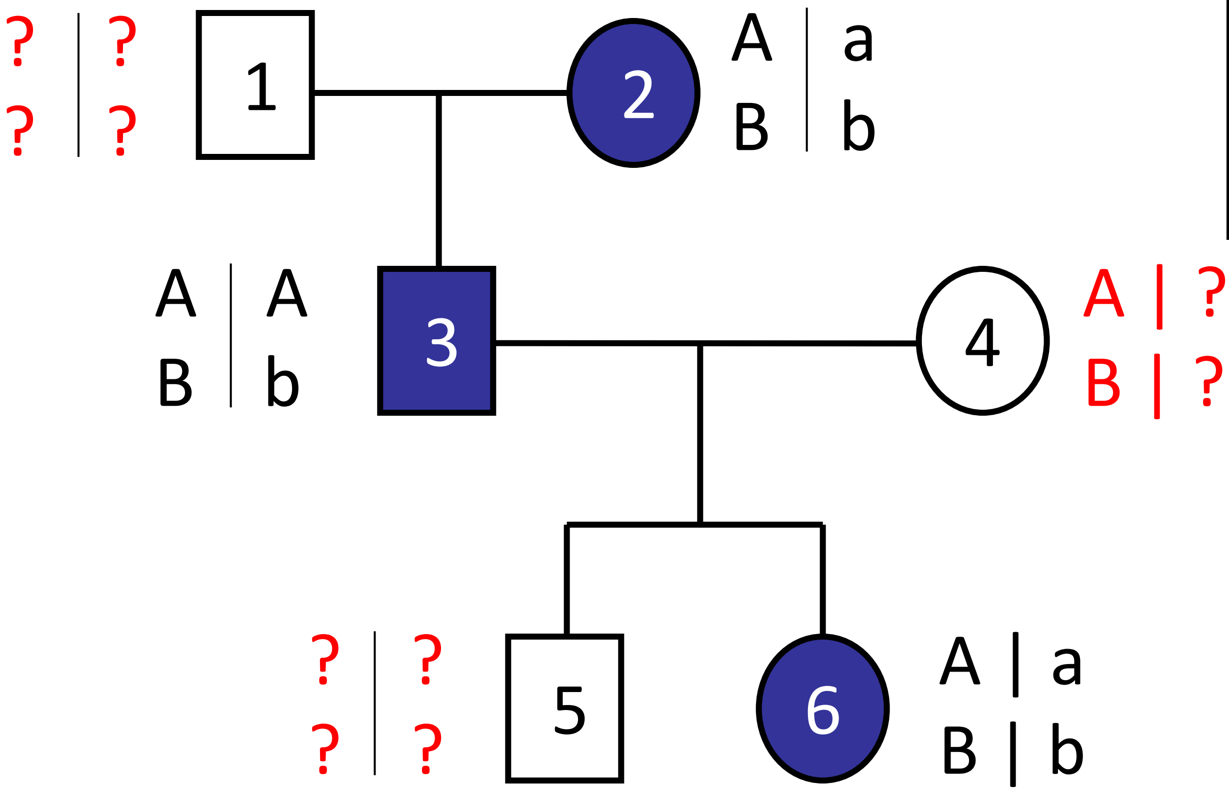


Alarm network

37 variables
509 parameters

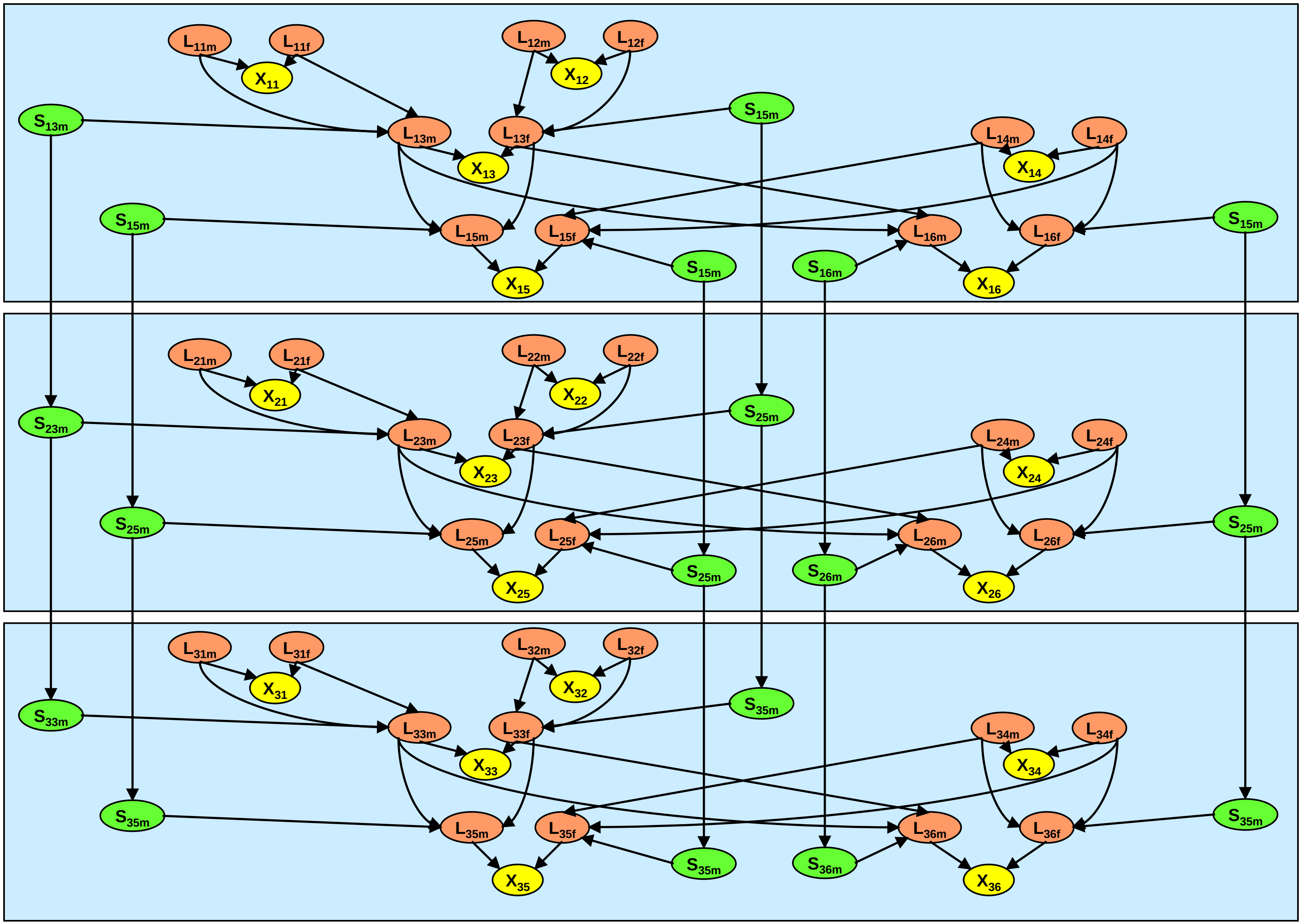
<< 2^{37}

Linkage Analysis



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

Pedigree: 6 people, 3 markers



Constraint Networks

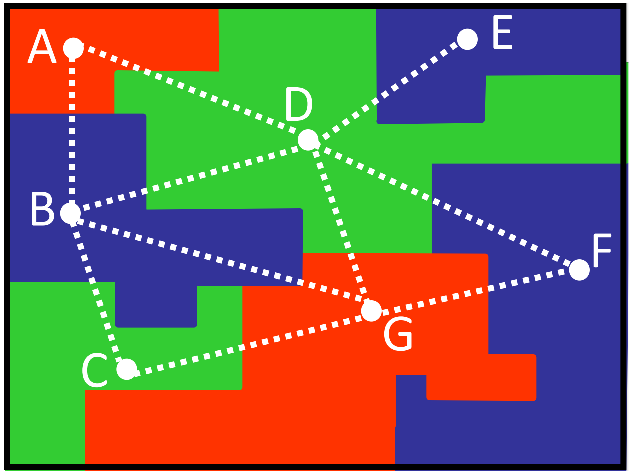
Map coloring

Variables: countries (A B C etc.)

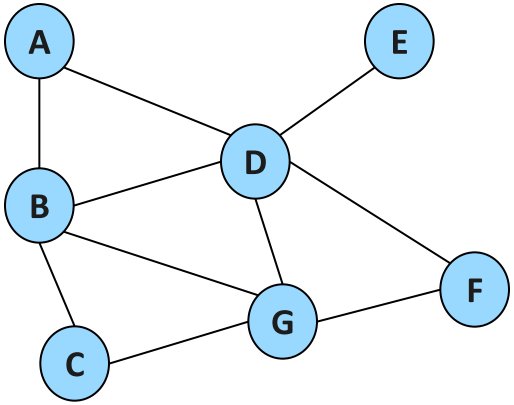
Values: colors (red green blue)

Constraints: $A \neq B$, $A \neq D$, $D \neq E$, …

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



Constraint graph



Graphical Models

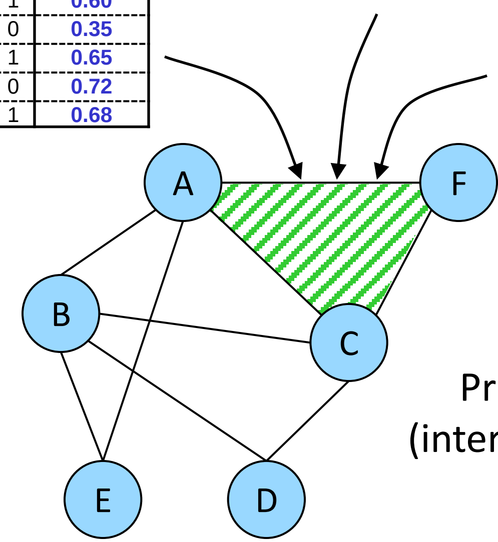
- A graphical model $(\mathbf{X},\mathbf{D},\mathbf{F})$:
  - $\mathbf{X} = \{X_1,\ldots X_n\}$ variables
  - $\mathbf{D} = \{D_1, \ldots D_n\}$ domains
  - $\mathbf{F} = \{f_1,\ldots,f_m\}$ functions
- Operators:
  - combination
  - elimination (projection)
- Tasks:
  - **Belief updating**: $\Sigma_{X-y}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

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

Relation

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

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



Primal graph (interaction graph)

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

Type of CPD

- Discrete variable
  - Tables
  - Noisy-or, noisy-and,
  - Decision trees
  - If/then rules
  - multinomial
- Continuous variables
  - Linear Gaussian

Example of Networks

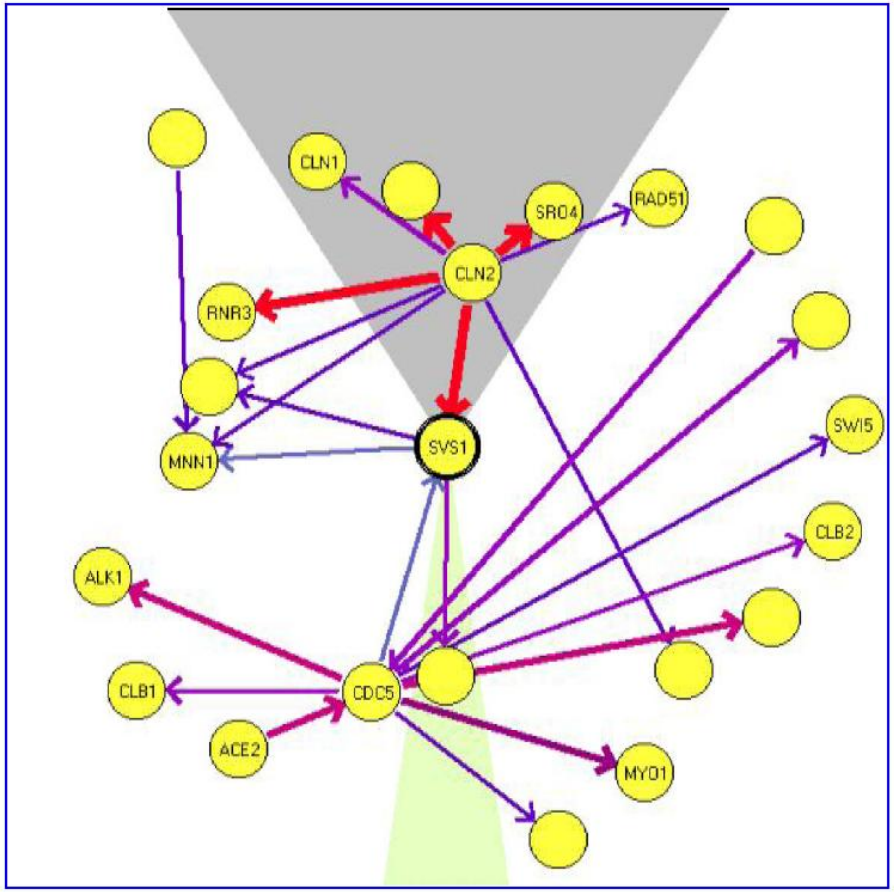
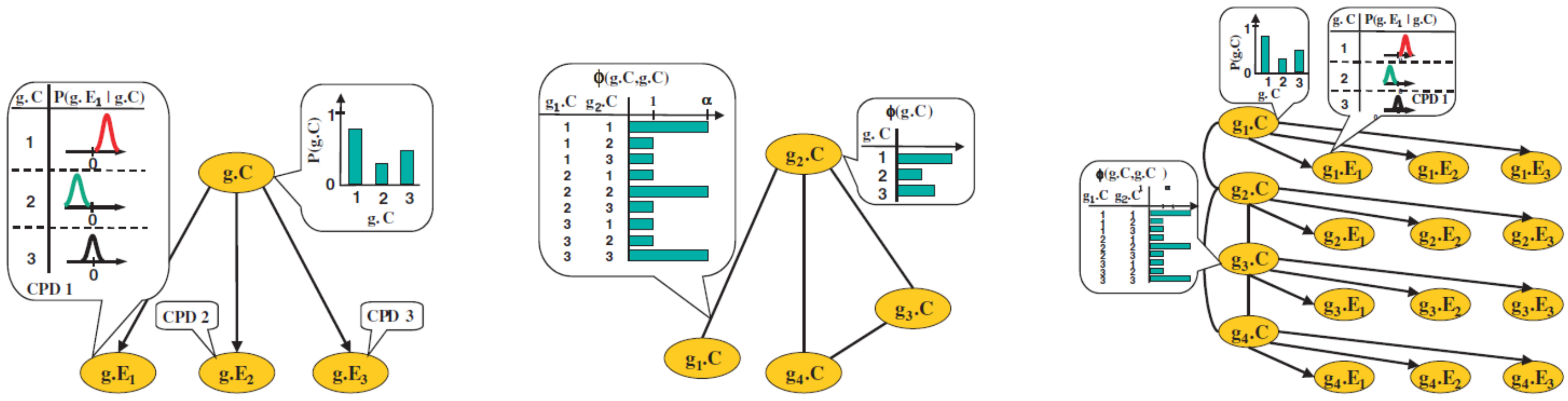


FIG. 2. An example of the graphical display of Markov features. This graph shows a "local map" for the gene SVS1. The width (and color) of edges corresponds to the computed confidence level. An edge is directed if there is a sufficiently high confidence in the order between the genes connected by the edge. This local map shows that CLN2 separates SVS1 from several other genes. Although there is a strong connection between CLN2 to all these genes, there are no other edges connecting them. This indicates that, with high confidence, these genes are conditionally independent given the expression level of CLN2.

Using Bayesian Networks to analyze expression data (Friedman et, al. 2001)

Example of networks



(a) (b) (c)

Fig. 2. (a) Naive Bayes model over 3 classes, for an expression data set with 3 expression measurements for each gene. A multinomial distribution is associated with $g.C$ (shown as a histogram). For each class $g.C$, each experiment is associated with a Gaussian CPD (shown in CPD 1). (b) Protein interaction model for a dataset with 4 genes in which the interactions are between: g_1 and g_2; g_2 and g_3; g_2 and g_4; and g_3 and g_4. Shown is the resulting Markov network, with its two types of potentials: $\phi_i(g_i.C)$ and $\phi_e(g_i.C, g_j.C)$. (c) Resulting unified partially-directed model.

Segal, Wang and Koller, 2003 “Discovering molecular pathways from Protein interaction and gene expression

Sample Domains for Graphical Models

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

…

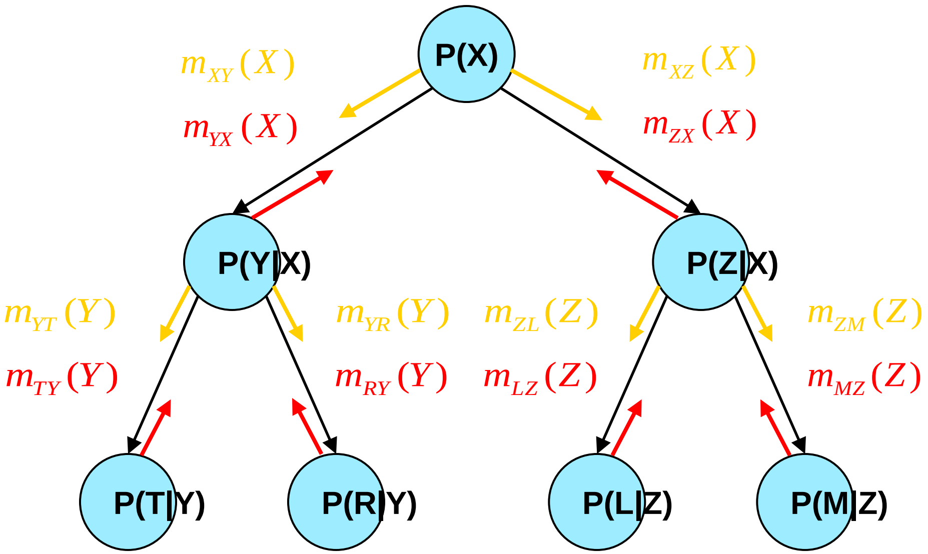
Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
- Search
- Sampling
- Hybrid of search and inference
- Modeling and learning
- Software

Tree-solving is easy

Belief updating (sum-prod)

CSP – consistency (projection-join)



MPE (max-prod)

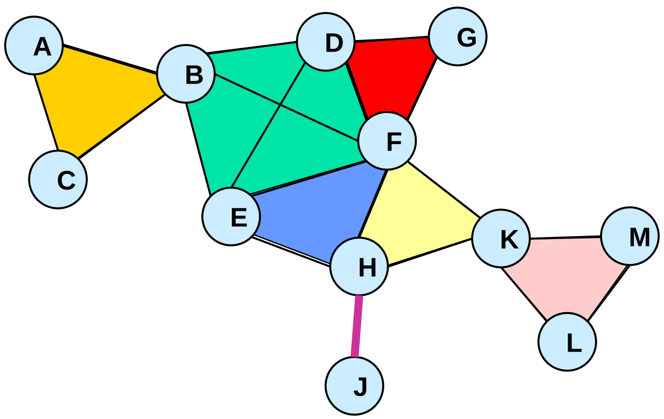
#CSP (sum-prod)

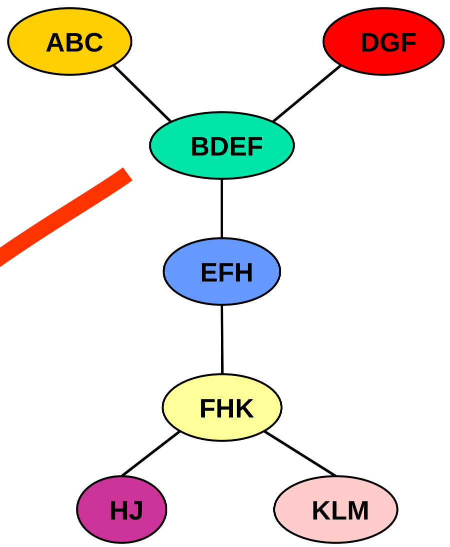
Trees are processed in linear time and memory

Transforming into a Tree

- **By Inference (thinking)**
  - Transform into a single, equivalent tree of sub-problems
- **By Conditioning (guessing)**
  - Transform into many tree-like sub-problems.

Inference and Treewidth





Inference algorithm:

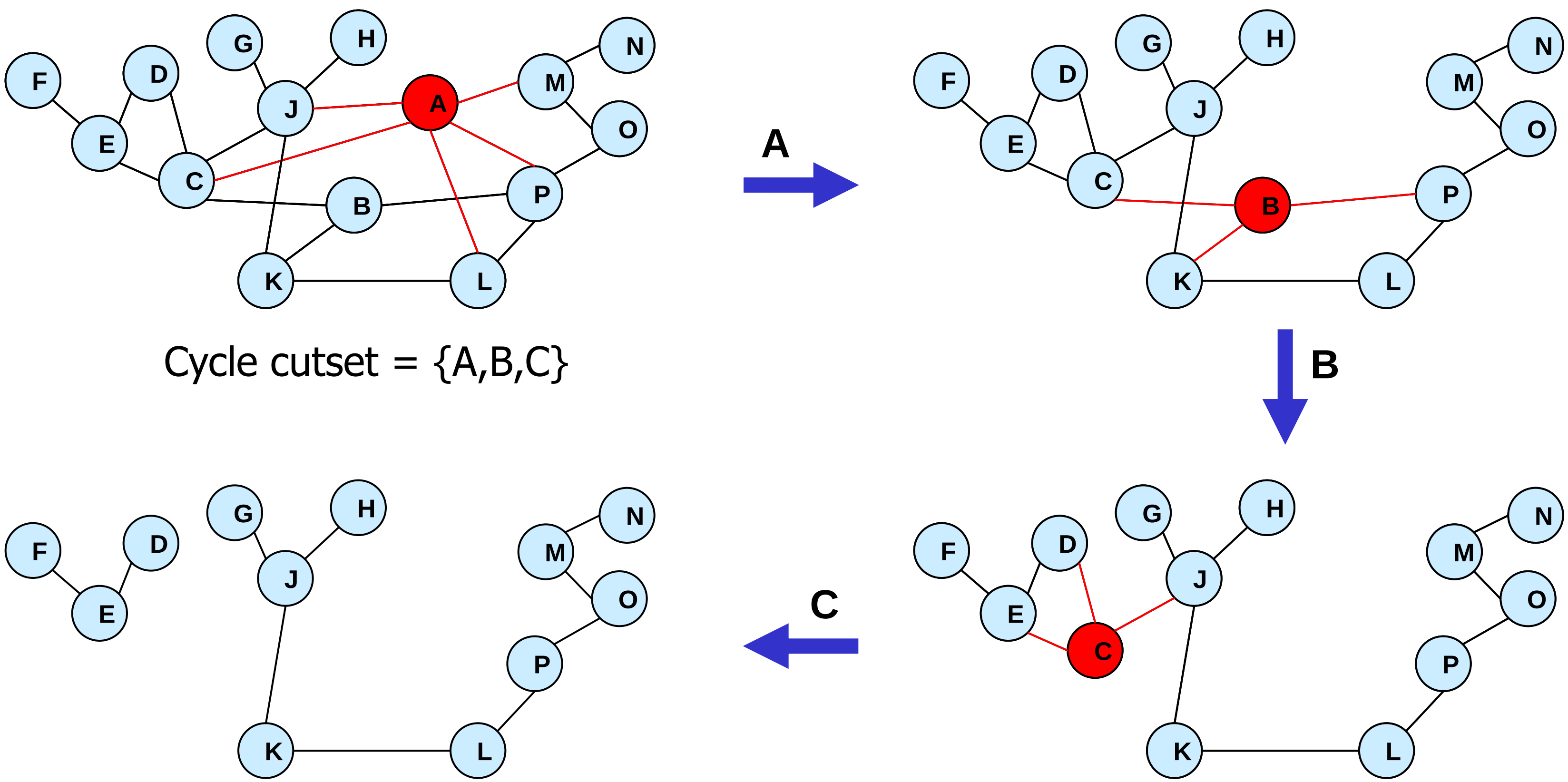
Time: exp(tree-width)

Space: exp(tree-width)

treewidth = 4 - 1 = 3

treewidth = (maximum cluster size) - 1

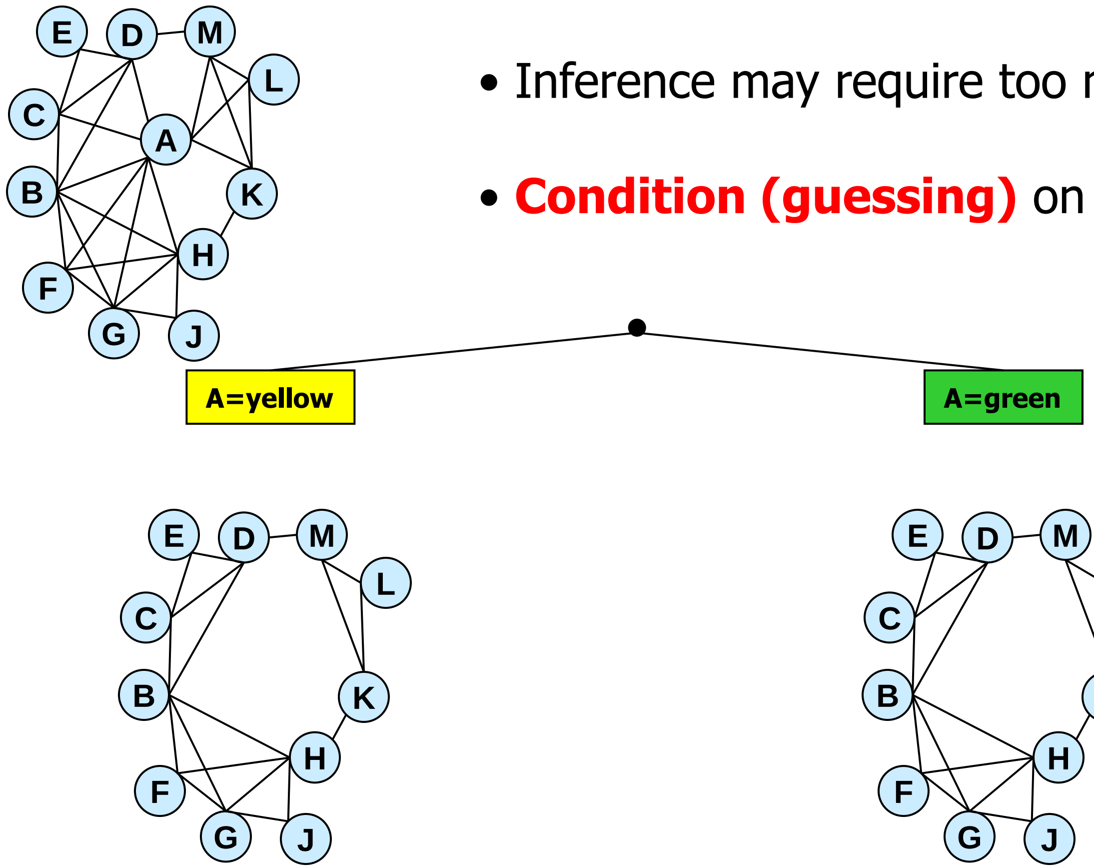
Conditioning and Cycle cutset



Cycle cutset = {A,B,C}

Search over the Cutset

Graph Coloring problem

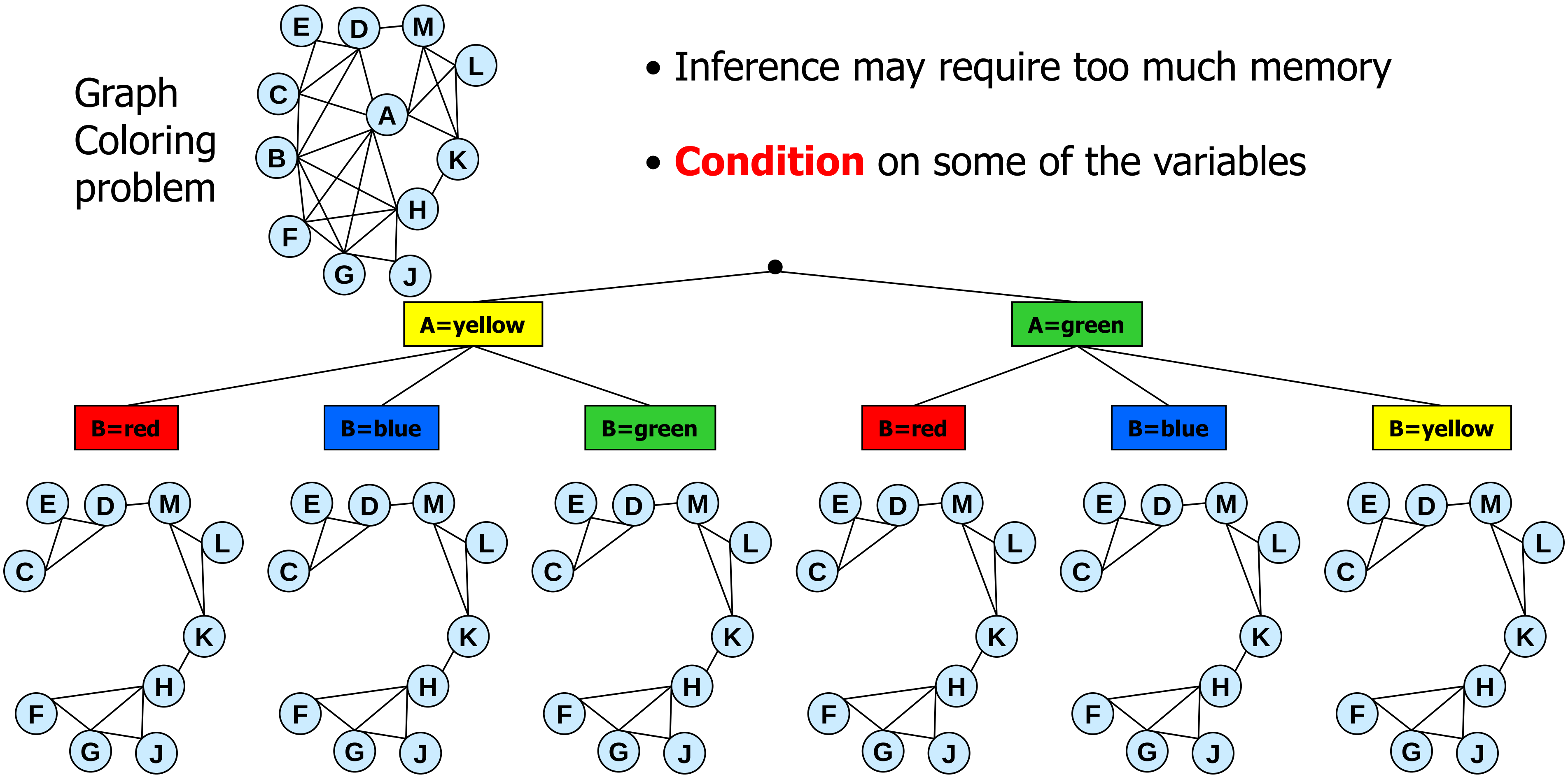


- Inference may require too much memory
- **Condition (guessing)** on some of the variables

Search over the Cutset (cont)

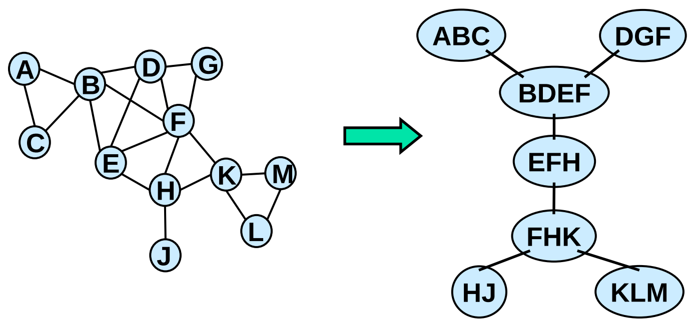
- Inference may require too much memory
- **Condition** on some of the variables

Graph Coloring problem



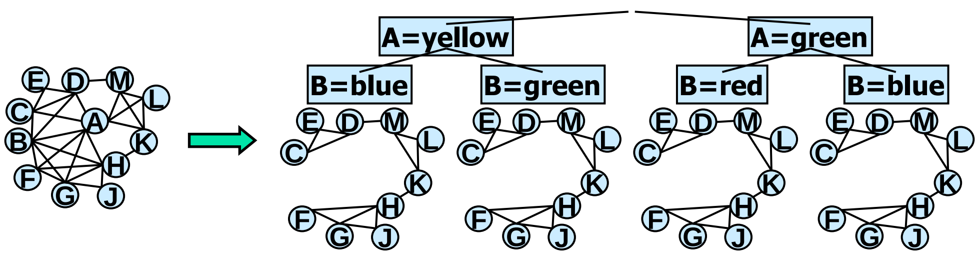
Inference vs. Conditioning

- By Inference (thinking)



Exponential in treewidth
Time and memory

- By Conditioning (guessing)



Exponential in cycle-cutset
Time-wise, linear memory

Solution Techniques, State of the art

AND/OR search

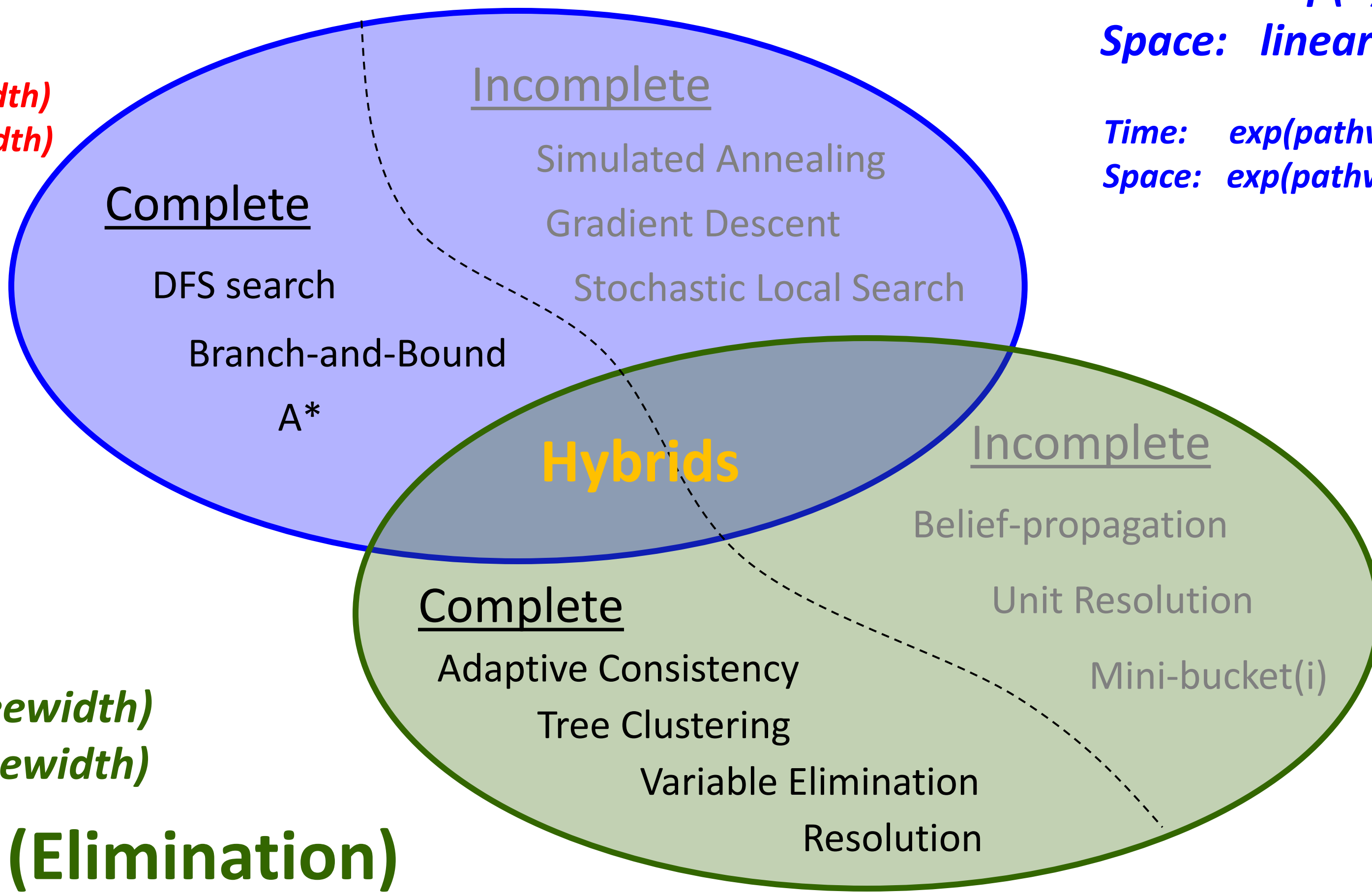
*Time: exp(treewidth * log n)*
Space: linear

Space: exp(treewidth)
Time: exp(treewidth)

Search (Conditioning)

Time: exp(n)
Space: linear

Time: exp(pathwidth)
Space: exp(pathwidth)



Time: exp(treewidth)
Space: exp(treewidth)

Inference (Elimination)

Solution techniques and queries

I will present algorithms that are uniformly applicable to both likelihood and optimizations, first.

As time permits, will focus on specific tasks:
Likelihood: belief, probability of evidence

optimization: mpe vs map

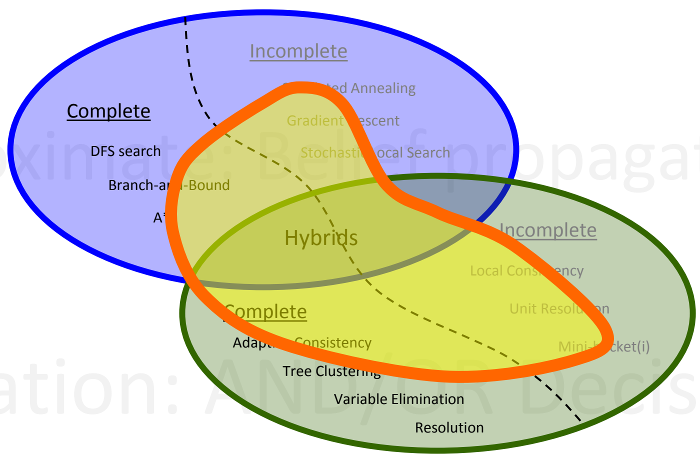
Will focus on **discrete** variables and assume table representation

Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
- Search
- Sampling
- Hybrid of search and inference
- Modeling and learning
- Software

Outline

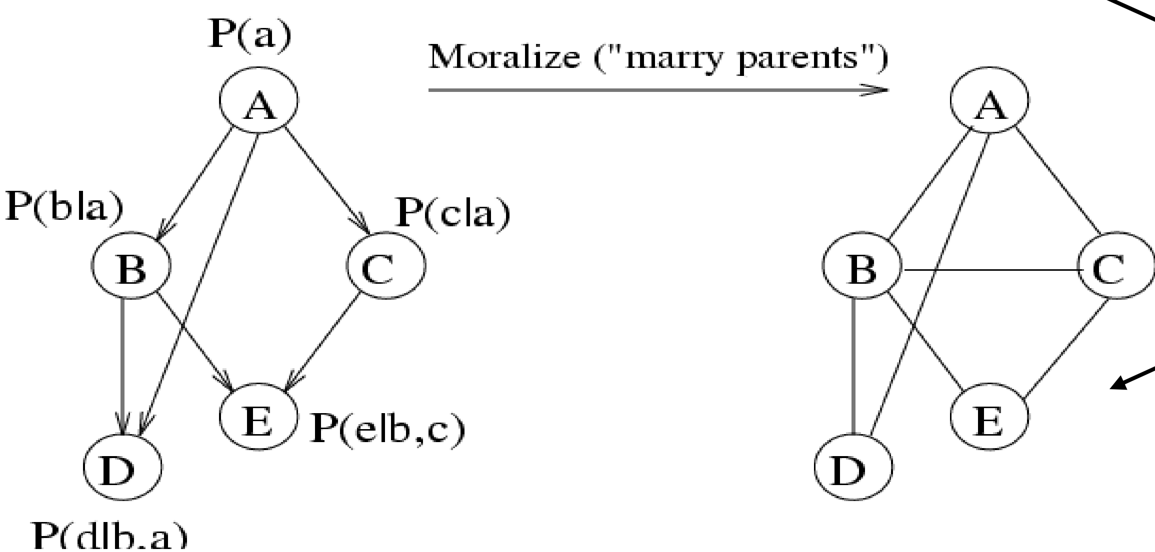
- Introduction
- Inference
  - Exact: Variable elimination, bucket elimination
  - Exact: cluster-tree propagation (join/junction-trees)
  - Approximate: Belief propagation



- Search
- Compilation: AND/OR Decision Diagrams
- Software

"Moral" Graph

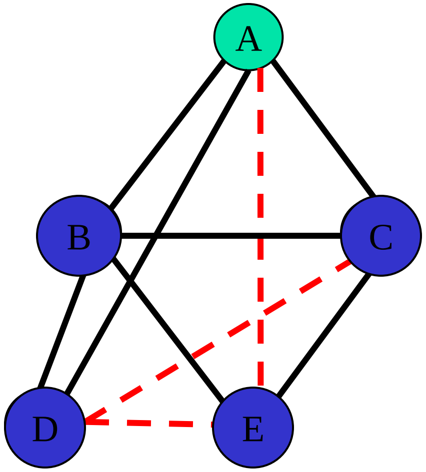
$$P(X_1,\ldots,X_n) = \prod_{i=1}^{n} \underbrace{P(X_i \mid parents(X_i))}$$



Conditional Probability Distribution (CPD)

Clique in moral graph ("family")

Belief updating: P(X|evidence)=?



"Moral" graph

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

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

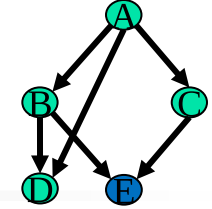
$$P(a)\sum_{e=0}\sum_{d}\sum_{c}P(c|a)\sum_{b}P(b|a)P(d|b,a)P(e|b,c)$$

Variable Elimination

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

Bucket elimination

Algorithm *BE-bel* (Dechter 1996)



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

$\sum_b \prod$ ← Elimination operator

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

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

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

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

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

P(e=0)

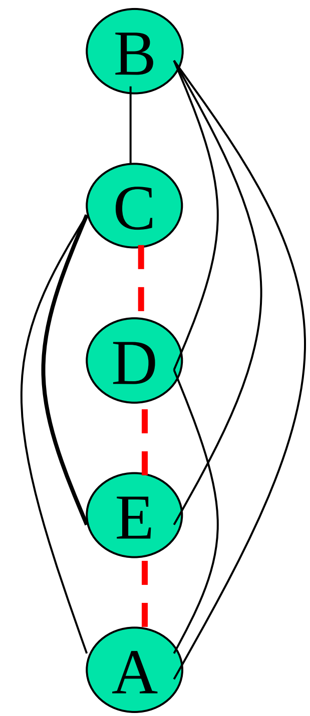
P(a|e=0)

W*=4

"induced width"

(max clique size)

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



The operation in a bucket

- Multiplying functions
- Marginalizing (summing-out) functions

Combination of Cost Functions

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

●

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

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

$= 0.1 \times 0.8$

Factors: Multiplication Operation

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

D	E	f_2
true	true	0.448
true	false	0.192
false	true	0.112
false	false	0.248

The result of multiplying the above factors:

B	C	D	E	$f_1(B, C, D)f_2(D, E)$
true	true	true	true	$0.4256 = (.95)(.448)$
true	true	true	false	$0.1824 = (.95)(.192)$
true	true	false	true	$0.0056 = (.05)(.112)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
false	false	false	false	$0.2480 = (1)(.248)$

Thanks to Darwiche

Factors: Sum-Out Operation

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

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

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

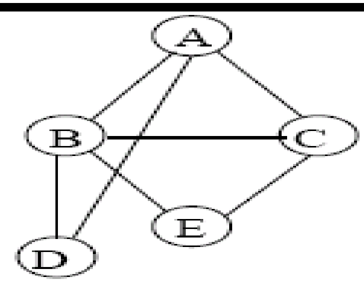
B	C	D	f_1
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

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

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

Thanks to Darwiche

Bucket Elimination and Induced Width



Ordering: a, e, d, c, b

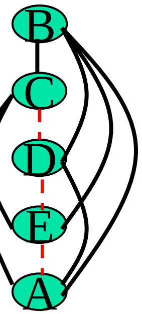
$bucket(B) = \quad P(e|b,c), P(d|a,b), P(b|a)$

$bucket(C) = \quad P(c|a) \quad || \quad \lambda_B(a,c,d,e)$

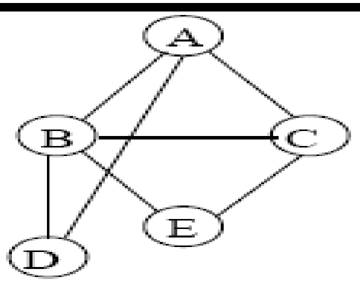
$bucket(D) = \quad || \ \lambda_C(a,d,e)$

$bucket(E) = \quad e = 0 \quad || \ \lambda_D(a,c)$

$bucket(A) = \quad P(a) \quad || \quad \lambda_E(a)$



Bucket Elimination and Induced Width



Ordering: a, b, c, d, e

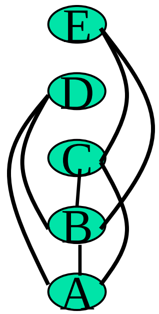
$bucket(E) = \quad P(e|b,c),\ e = 0$

$bucket(D) = \quad P(d|a,b)$

$bucket(C) = \quad P(c|a) \quad || \ P(e=0|b,c)$

$bucket(B) = \quad P(b|a) \quad || \quad \lambda_D(a,b), \lambda_C(b,c)$

$bucket(A) = \quad P(a) \quad || \quad \lambda_B(a)$



W*=2

Ordering: a, e, d, c, b

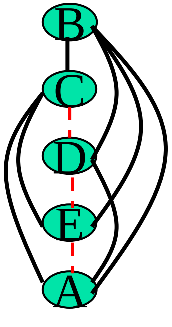
$bucket(B) = \quad P(e|b,c), P(d|a,b), P(b|a)$

$bucket(C) = \quad P(c|a) \quad || \quad \lambda_B(a,c,d,e)$

$bucket(D) = \quad || \ \lambda_C(a,d,e)$

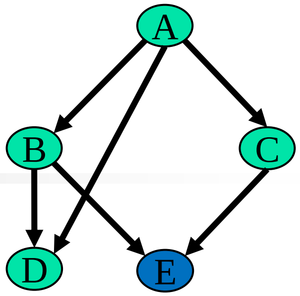
$bucket(E) = \quad e = 0 \quad || \ \lambda_D(a,c)$

$bucket(A) = \quad P(a) \quad || \quad \lambda_E(a)$

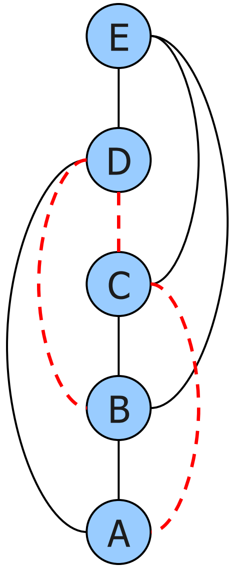


W*=4

Induced-width



- **Width** along ordering *d*, w(d):
  - max # of previous neighbors (parents)

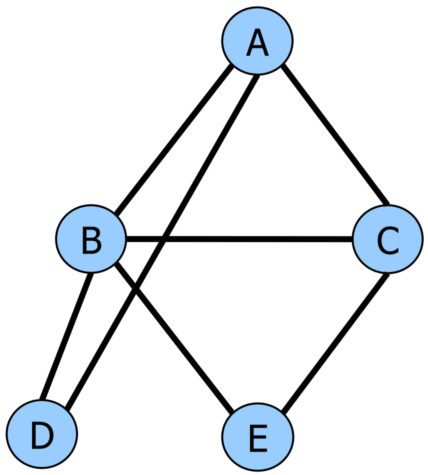


- **Induced width** along ordering d, w*(d):
  - The width in the ordered induced graph, obtained by connecting "parents" of each node X, recursively from top to bottom

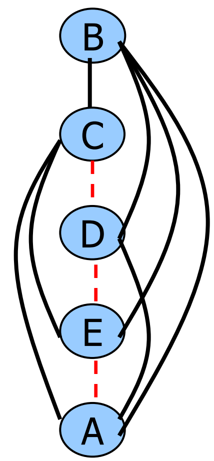
Induced width (continued)

$w^*(d)$ – the induced width of the primal graph along ordering d

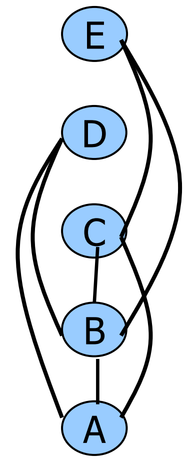
The effect of the ordering:



constraint graph



$w^*(d_1) = 4$



$w^*(d_2) = 2$

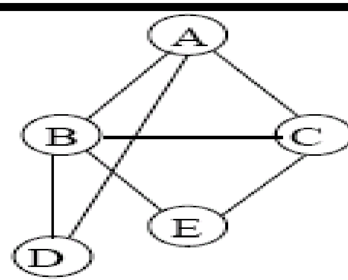
Finding smallest induced-width is hard!
Greedy algorithms (min-fill) works well.
Significant research area

BE-BEL

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

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

Handling Observations



Observing $b = 1$

Ordering: a, e, d, c, b

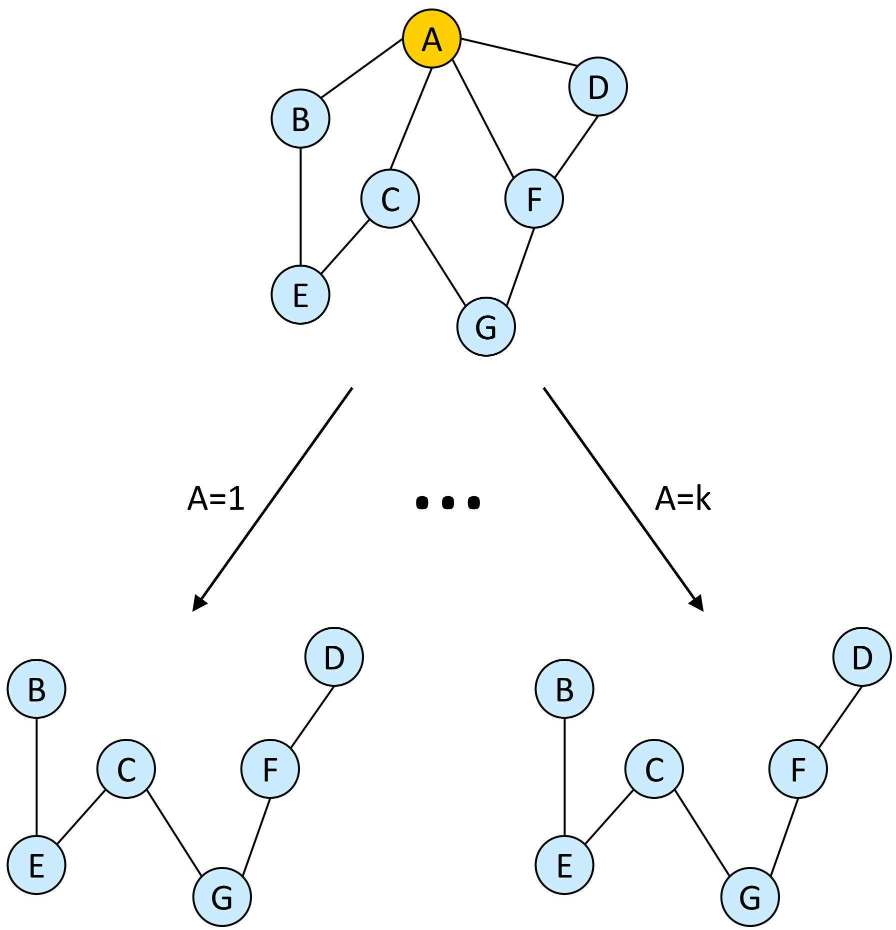
$bucket(B) = \quad P(e|b,c), P(d|a,b), P(b|a), b = 1$
$bucket(C) = \quad P(c|a), \quad || \quad P(e|b = 1, c)$
$bucket(D) = \quad || \quad P(d|a, b = 1)$
$bucket(E) = \quad e = 0 \quad || \quad \lambda_C(e, a)$
$bucket(A) = \quad P(a), \quad || \quad P(b = 1|a) \ \lambda_D(a), \lambda_E(e, a)$

Ordering: a, b, c, d, e

$bucket(E) = \quad P(e|b,c), \ e = 0$
$bucket(D) = \quad P(d|a,b)$
$bucket(C) = \quad P(c|a) \quad || \quad \lambda_E(b, c)$
$bucket(B) = \quad P(b|a), b = 1 \quad || \quad \lambda_D(a, b), \lambda_C(a, b)$
$bucket(A) = \quad P(a) \quad || \quad \lambda_B(a)$

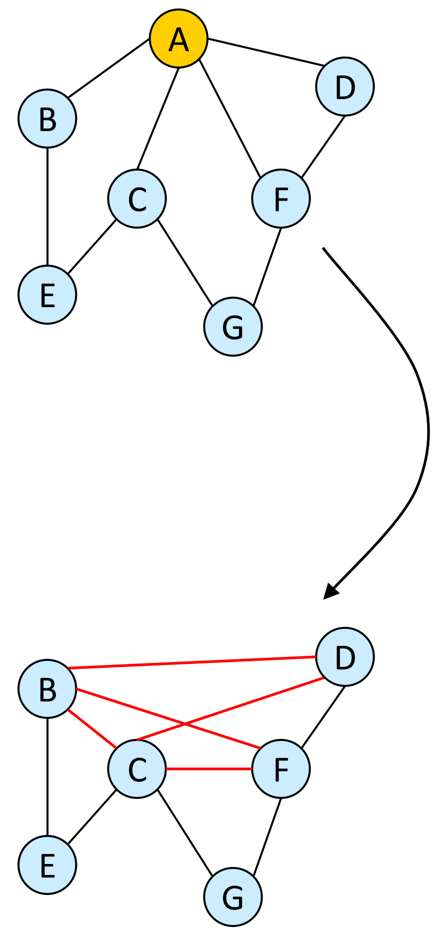
Search vs. Inference

Search (conditioning)



k “sparser” problems

Inference (elimination)



1 “denser” problem

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

Algorithm *BE-mpe* (Dechter 1996)

$$\sum \text{ is replaced by } \boldsymbol{max}:$$

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

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

Algorithm *BE-mpe* (Dechter 1996)

$$\sum \text{ is replaced by } \boldsymbol{max}:$$

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

$\max_b \prod$ ← Elimination operator

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

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

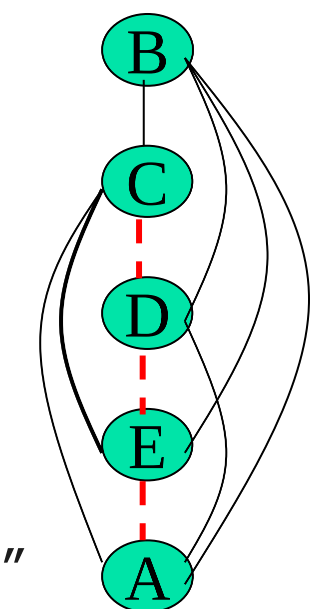
bucket D: $h^C(a,d,e)$

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

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

MPE

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



Generating the MPE-tuple

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

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

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

2. $e' = 0$

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

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

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

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

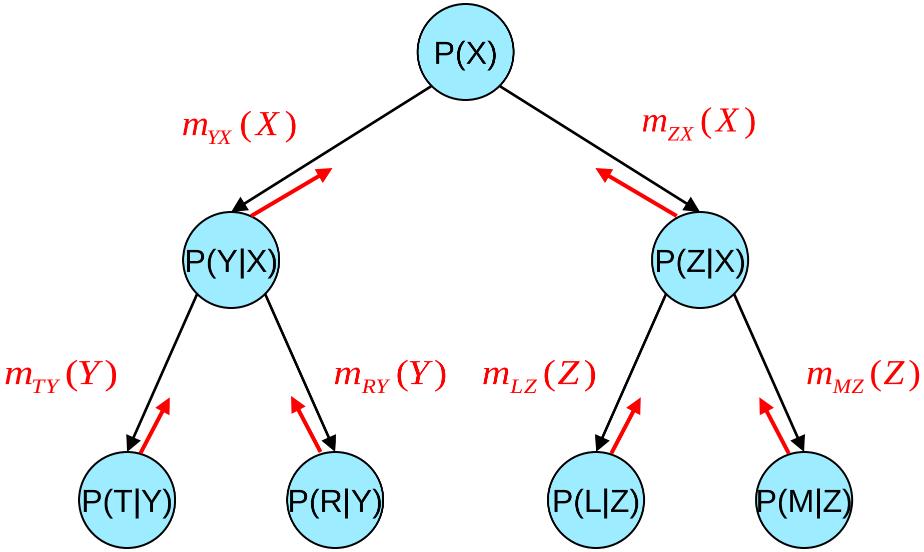
E: e=0 $h^D(a,e)$

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

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

Complexity of Bucket-elimination

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



bucket-elimination
Sends messages
From leaves to root

Belief Updating Example

SUM-PROD operators
POLY-TREE structure

H	P(H)
0	.9
1	.1

F	P(F)
0	.99
1	.01

*

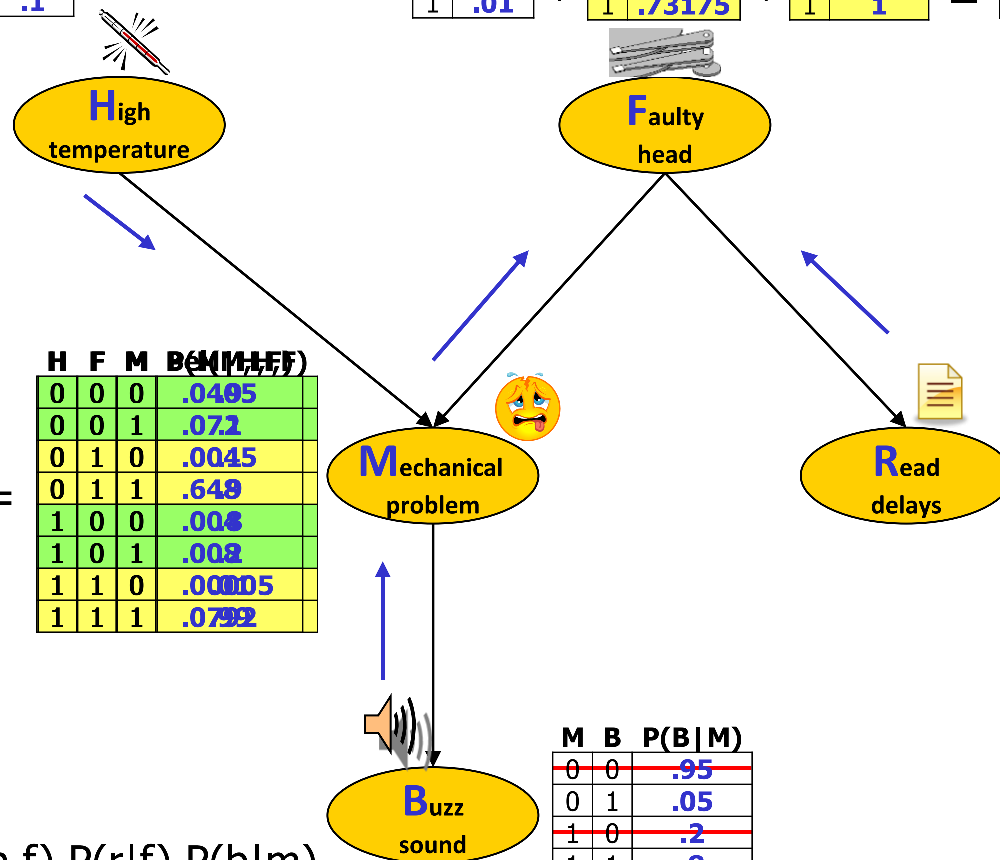
F	h_3(F)
0	.1245
1	.73175

*

F	h_4(F)
0	1
1	1

=

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



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

*

M	h_1(M)
0	.05
1	.8

*

H	h_2(H)
0	.9
1	.1

=

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

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

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

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

P(F | B=1) = ?

P(B=1) = .19643

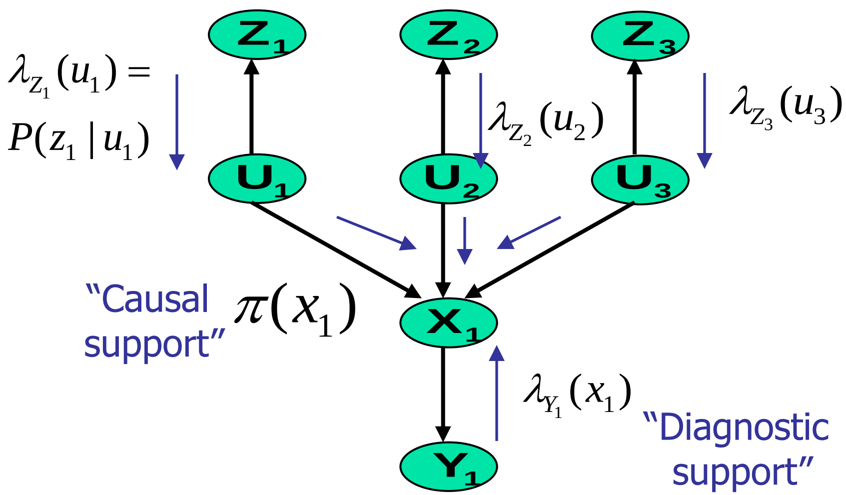
Probability of evidence

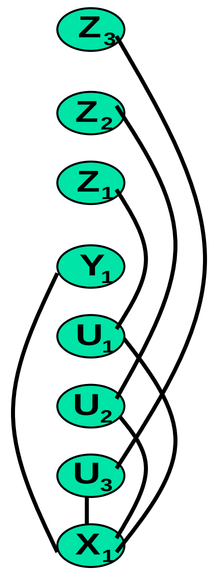
P(F=1|B=1) = .3725

Updated belief

Relationship with Pearl's belief propagation on poly-trees (Pearl 1988)

I



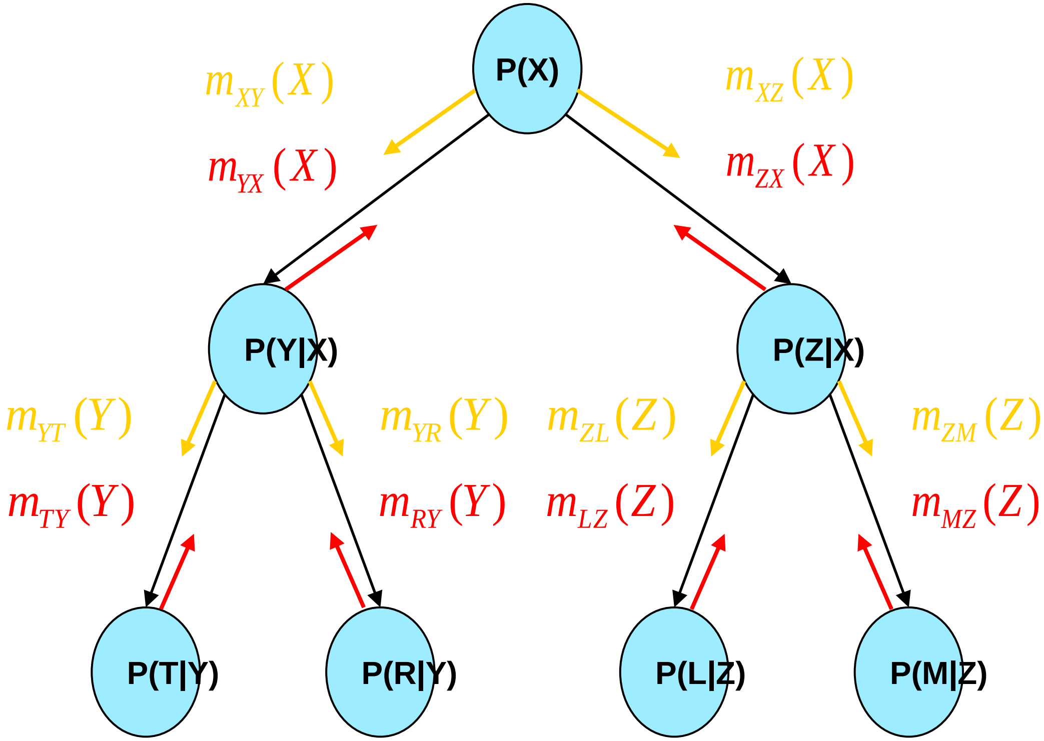


Pearl's belief propagation for single-root query ⟺ BE-bel using topological ordering

On a trees induced-width is 1: message-passing is linear.
On poly-tree **width = induced-width**, message-passing is linear.
But message propagation can go both ways

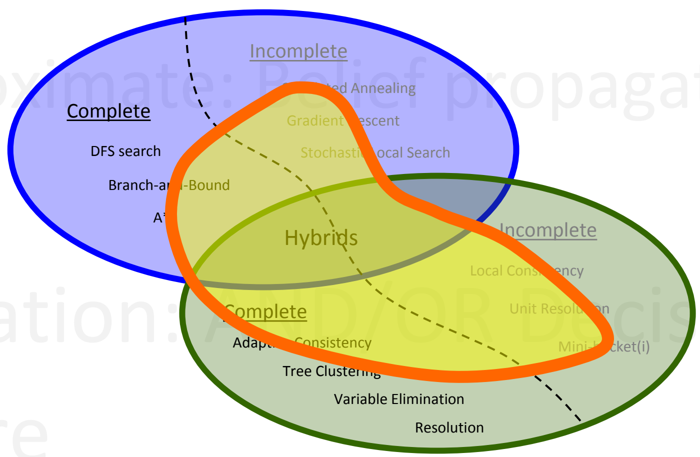
Propagation in both directions

- Messages can propagate both ways and we get beliefs for each variable



Outline

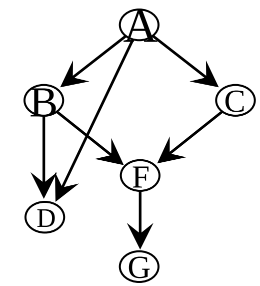
- Introduction
- Inference
  - Exact: Variable elimination, bucket elimination
  - cluster-tree propagation (join/junction-trees)



  - Approximate: Belief propagation
- Search
- Compilation: AND/OR Decision Diagrams
- Software

From Bucket elimination to bucket-tree elimination

If we want the marginal on D?



Bucket G: $P(G|F)$

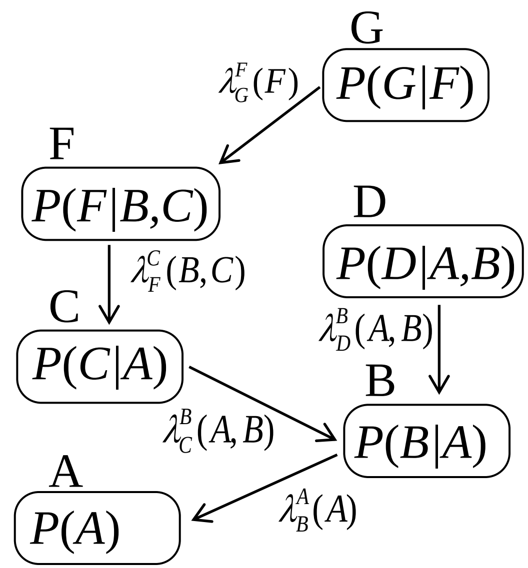
Bucket F: $P(F|B,C)$ $\lambda_G^F(F)$

Bucket D: $P(D|A,B)$

Bucket C: $P(C|A)$ $\lambda_F^C(B,C)$

Bucket B: $P(B|A)$ $\lambda_D^B(A,B)$ $\lambda_C^B(A,B)$

Bucket A: $P(A)$ $\lambda_B^A(A)$



BTE: allows messages both ways

Each bucket can
Compute its
marginal probability

Bucket G: $P(G/F)$ → $\pi_F^G(F)$

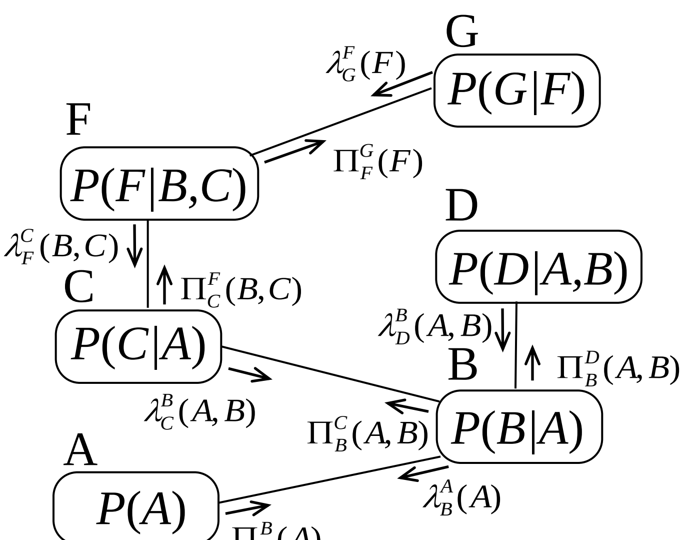
Bucket F: $P(F/B,C)$ $\lambda_G^F(F)$ → $\pi_C^F(B,C)$

Bucket D: $P(D/A,B)$ → $\pi_B^D(A,B)$

Bucket C: $P(C/A)$ $\lambda_F^C(B,C)$ → $\pi_B^C(A,B)$

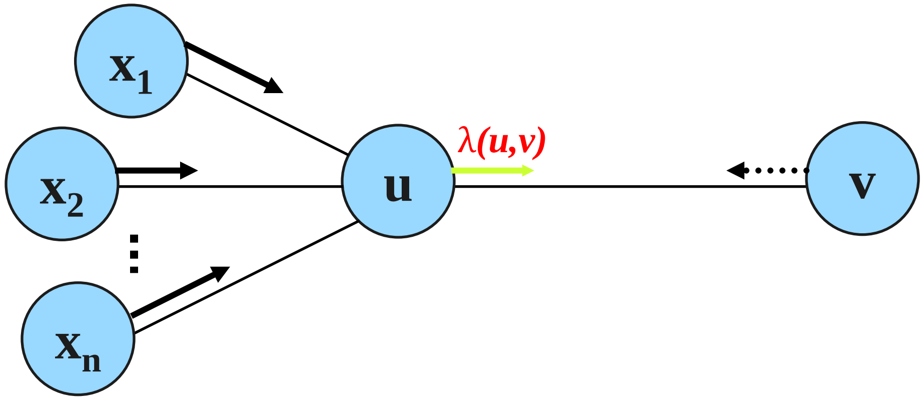
Bucket B: $P(B/A)$ $\lambda_D^B(A,B)$ $\lambda_C^B(A,B)$ → $\pi_A^B(A)$

Bucket A: $P(A)$ $\lambda_B^A(A)$



$$\pi_A^B(a) = P(a)$$
$$\pi_B^C(c,a) = P(b|a)\lambda_D^B(a,b)\pi_A^B(a)$$
$$\pi_B^D(a,b) = P(b|a)\lambda_C^B(a,b)\pi_A^B(a,b)$$
$$\pi_C^F(c,b) = \sum_a P(c|a)\pi_B^C(a,b)$$
$$\pi_F^G(f) = \sum_{b,c} P(f|b,c)\pi_C^F(c,b)$$

Same Message Passing rule up and down



$$bucket(u) = P(u) \cup \{\lambda(x_1,u), \lambda(x_2,u), \ldots, \lambda(x_n,u), \lambda(v,u)\}$$

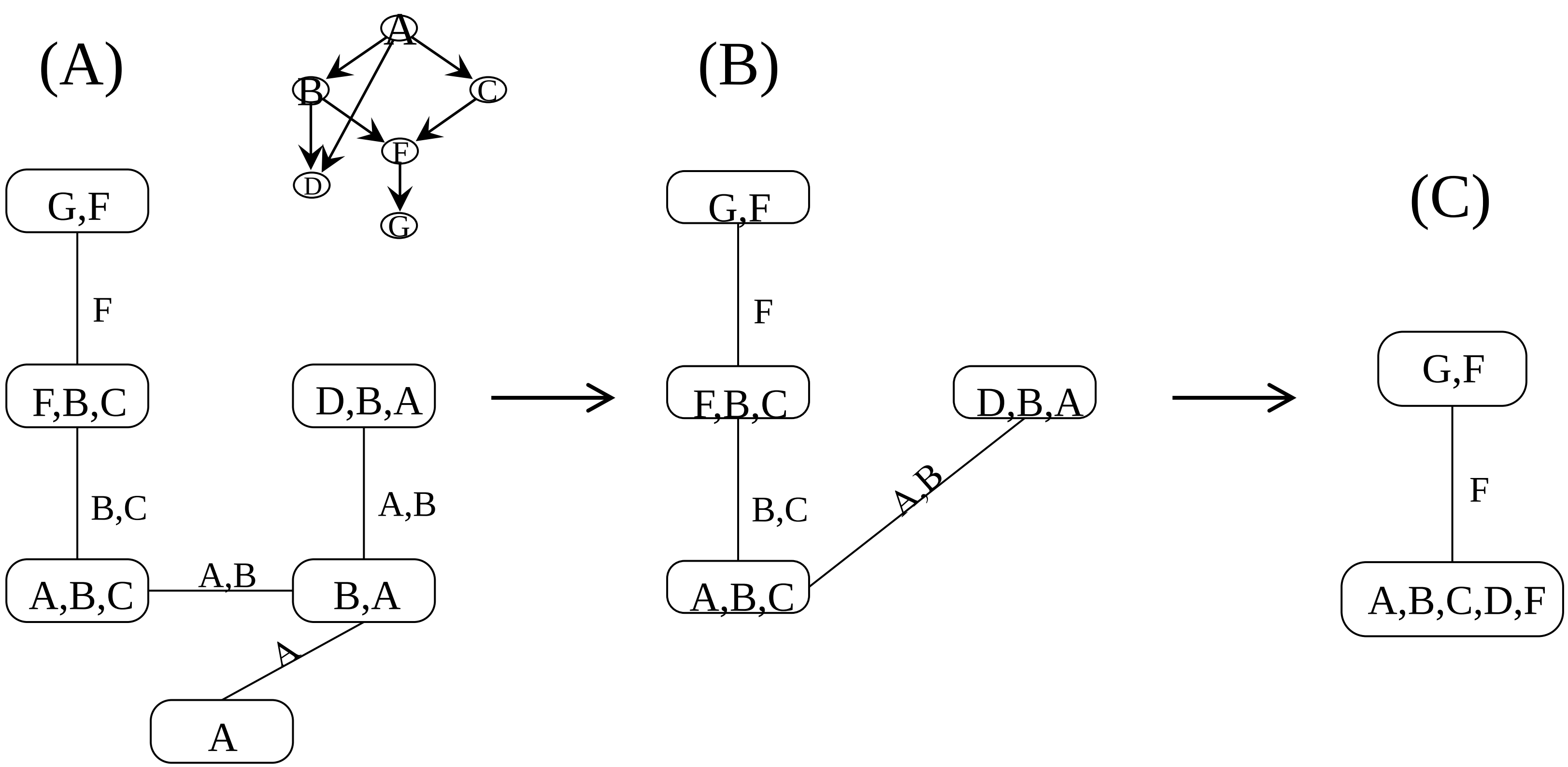
Compute the message:

$$\lambda(u,v) = \sum_{elim(u,v)} \prod_{f \in bucket(u) - \{\lambda(v,u)\}} f$$

Elim(u,v) = cluster(u)-sep(u,v)

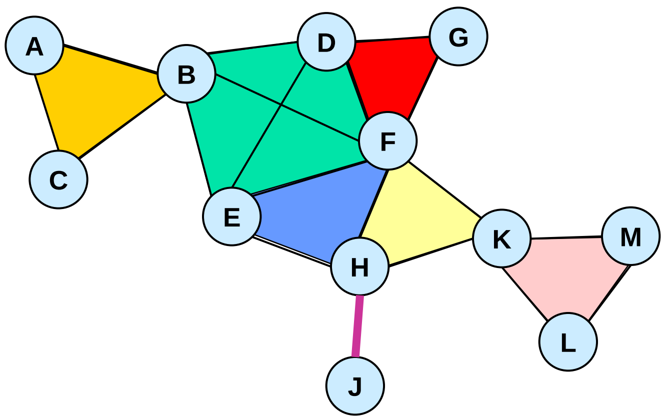
From a bucket-tree to a join-tree

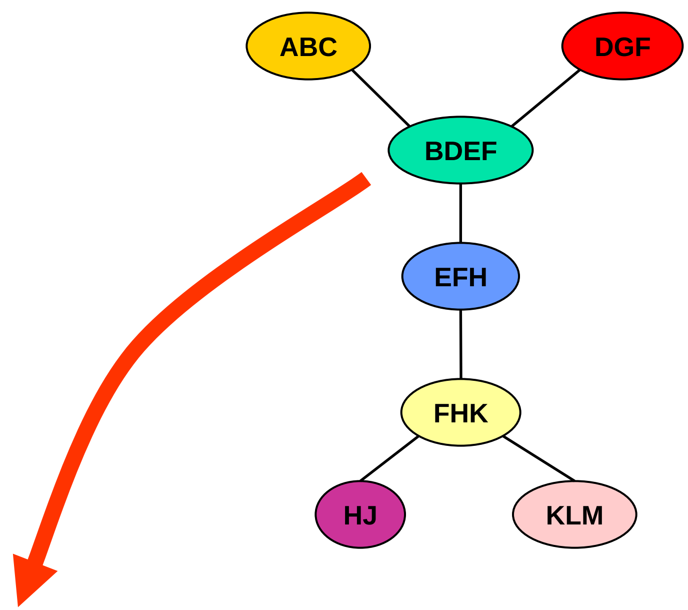
- Merge non-maximal buckets into maximal clusters.
- Connect clusters into a tree: each cluster to one with which it shares a largest subset of variables.
- Separators are variable- intersection on adjacent clusters.



A super-bucket-tree is an i-map of the Bayesian network

The general tree-decomposition





Inference algorithm:

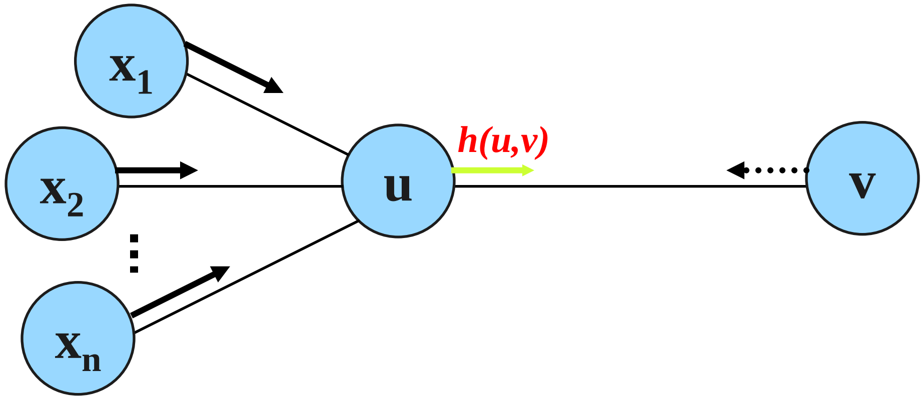
Time: exp(tree-width)

Space: exp(tree-width)

treewidth = 4 - 1 = 3

treewidth = (maximum cluster size) - 1

The general Message Passing on a general tree-decomposition



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

For max-product
Just replace Σ
With max.

Compute the message :

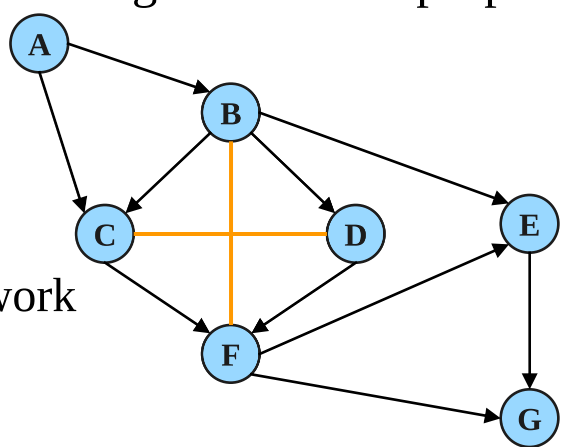
$$h(u,v) = \sum_{elim(u,v)} \prod_{f \in cluster(u) - \{h(v,u)\}} f$$

Elim(u,v) = cluster(u)-sep(u,v)

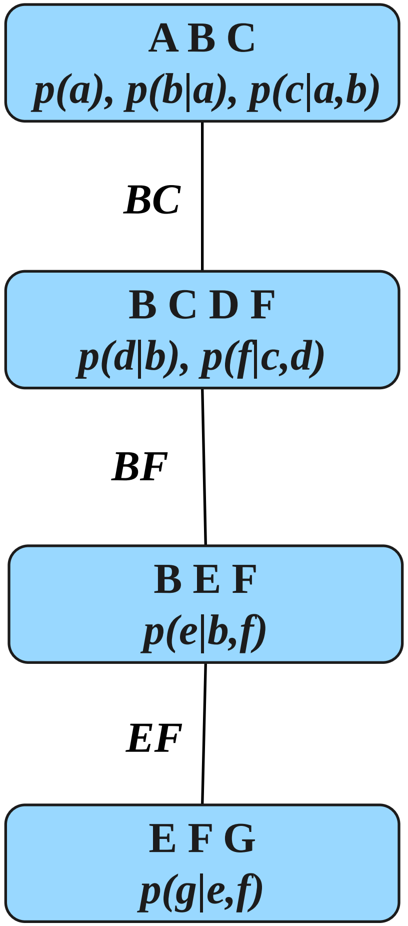
Tree decompositions (formal)

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

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

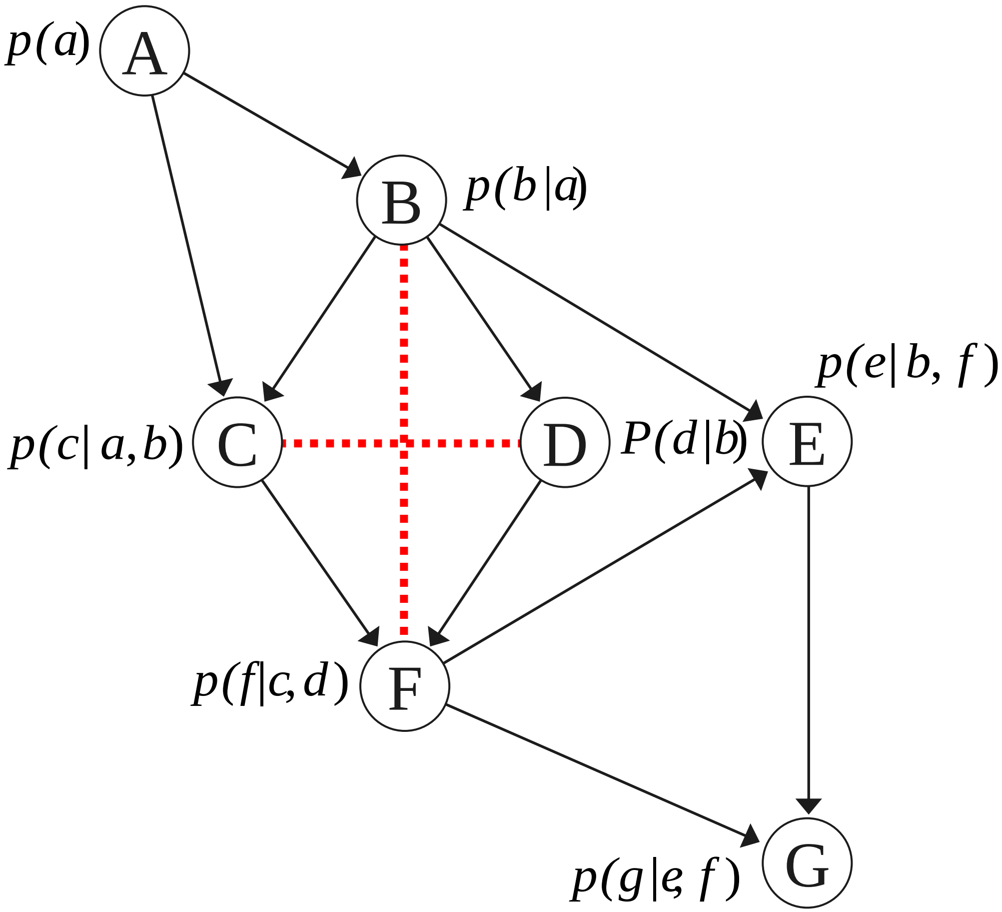


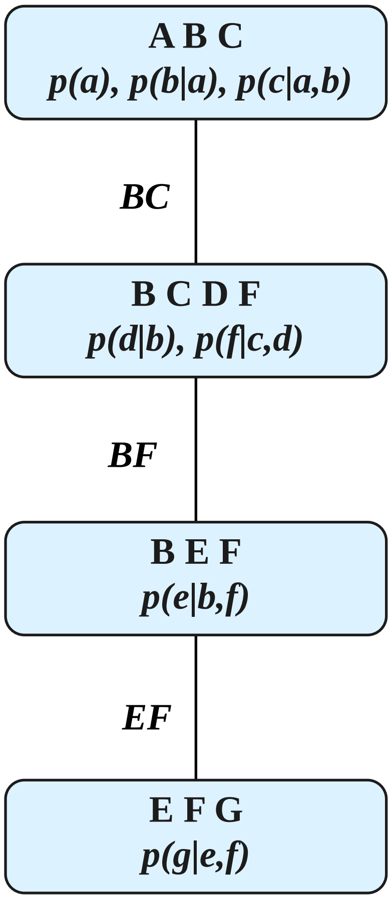
Belief network



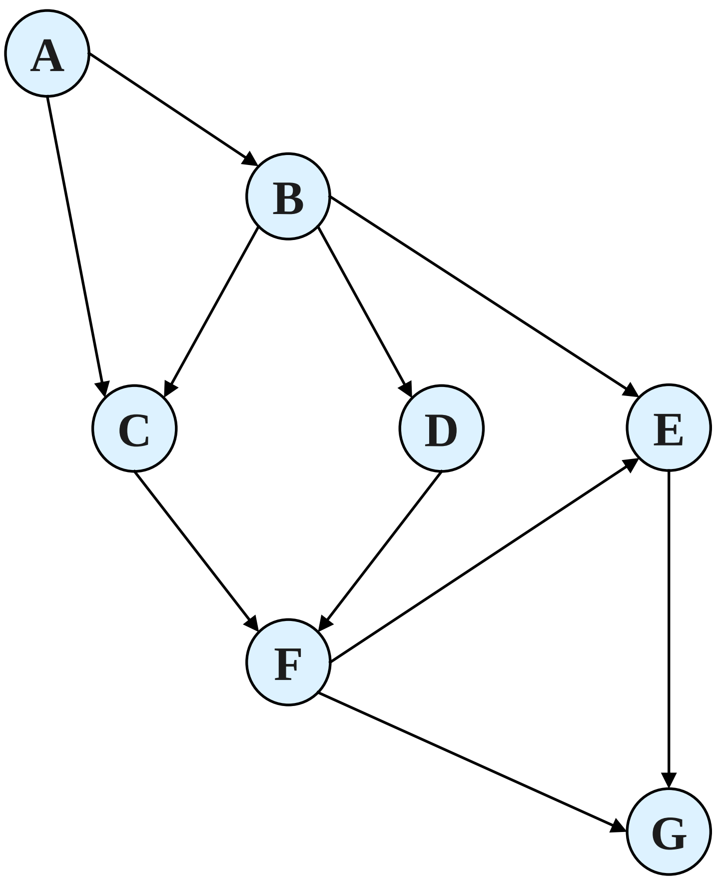
Tree decomposition

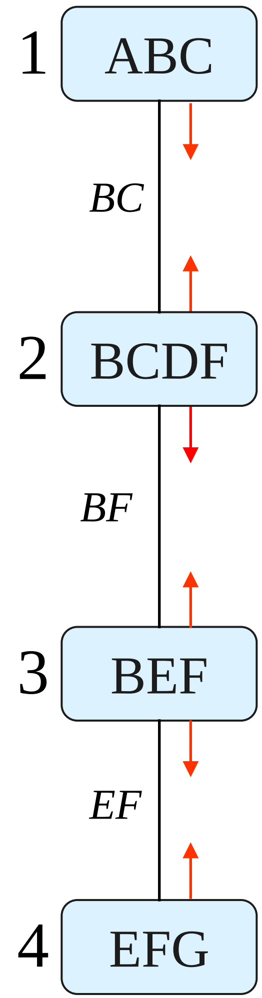
Tree Decomposition for belief updating





CTE: Cluster Tree Elimination





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

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

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

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

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

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

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

For each cluster P(X|e) is computed, also P(e)

Algorithm cluster-tree elimination (CTE)

Input: A tree decomposition $< T, \chi, \psi >$ for a problem $M =< X, D, F, \prod\} >$, $X = \{X_1, ..., X_n\}$, $F = \{f_1, ..., f_r\}$.

Output: An augmented tree whose vertices are clusters containing the original functions as well as messages received from neighbors. A solution computed from the augmented clusters.

Compute messages:

For every edge (u, v) in the tree, do

- Let $m_{(u,v)}$ denote the message sent by vertex u to vertex v.
- Let $cluster(u) = \psi(u) \cup \{m_{(i,u)} | (i, u) \in T\}$.
- If vertex u has received messages from all adjacent vertices other than v, then compute and send to v,

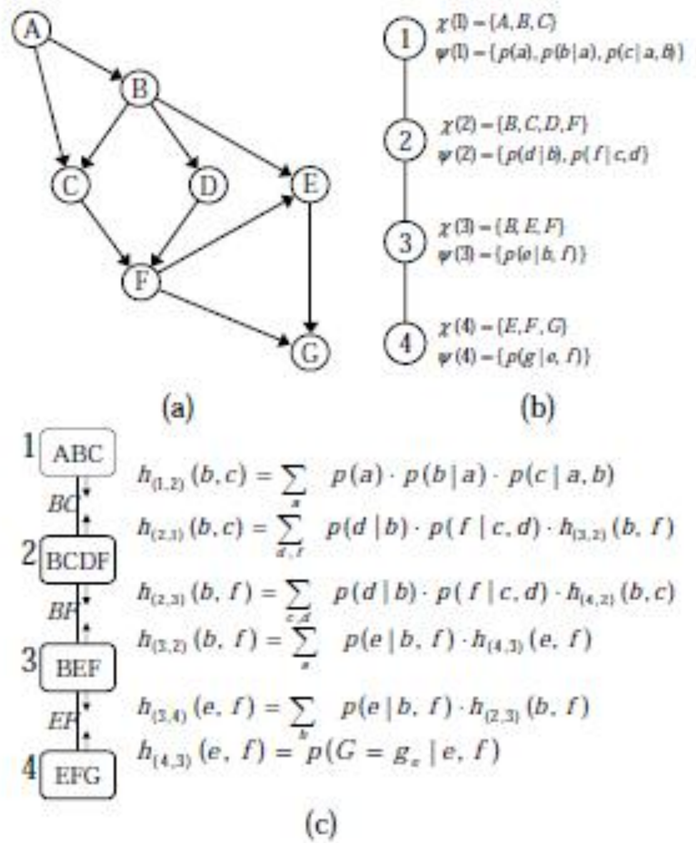
$$m_{(u,v)} = \sum_{sep(u,v)} \Big(\prod_{f \in cluster(u), f \neq m_{(v,u)}} f \Big)$$

Endfor

Note: functions whose scope does not contain elimination variables do not need to be processed, and can instead be directly passed on to the receiving vertex.

Return: A tree-decomposition augmented with messages, and for every $v \in T$

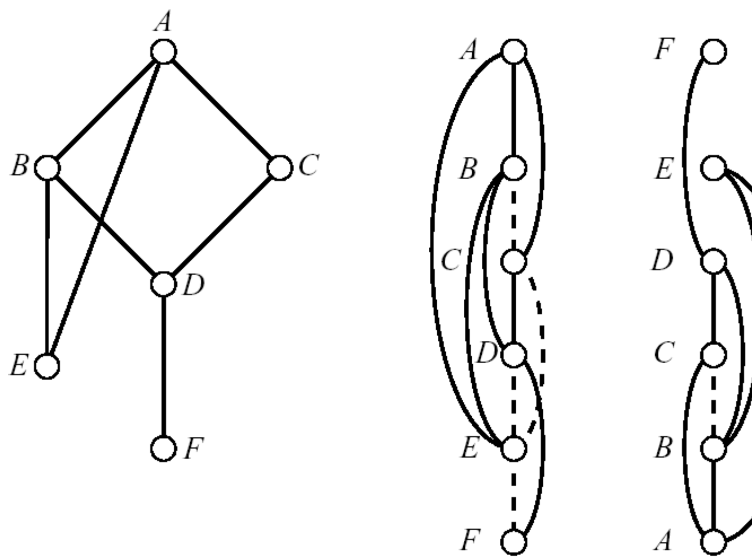
CTE (continued)



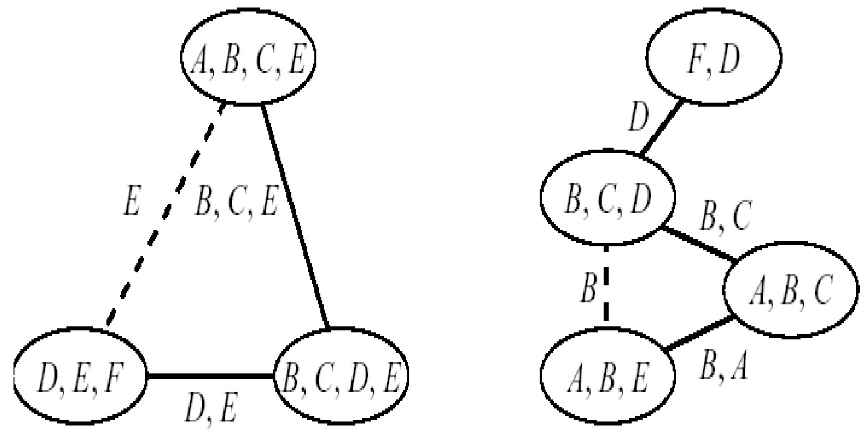
Let Ci and Cj two adjacent clusters and sep(i,j) be their separator

$$bel(sep) = \sum_{elim(i,j)} \prod_{f \in C_i} f = \sum_{elim(j,i)} \prod_{f \in C_j} f = h_{(i,j)} \bullet h_{(j,i)}$$

Examples of tree-clustering



(a) (b) (c)



(a) (b)

CTE - properties

- Correctness and completeness: Algorithm CTE is correct, i.e. it computes the exact joint probability in each cluster and therefore of every single variable and the evidence.

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

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

where

deg = the maximum degree of a node in the cluster-tree

n = number of variables (= number of CPTs)

N = number of nodes in the tree decomposition

k = the maximum domain size of a variable

*w** = the induced width

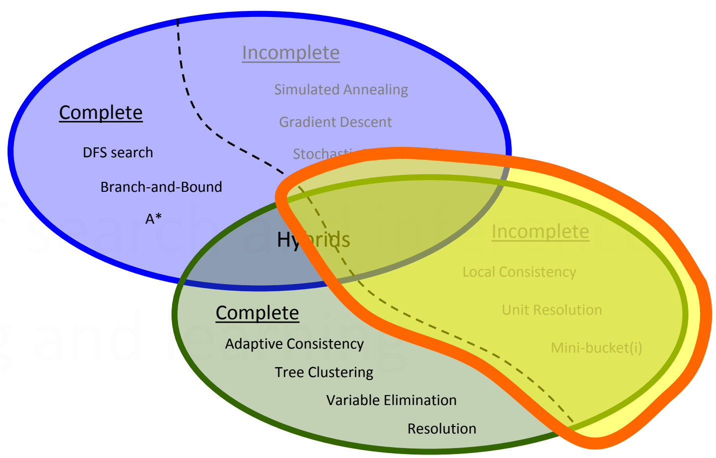
sep = the separator size

Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
- Search
- Sampling
- Hybrid of search and inference
- Modeling and learning
- Software

Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
  - Mini-buckets, mini-clusters
  - Belief propagation, Generalized belief propagation
- Search
- Sampling
- Hybrid of search
- Modeling and
- Software



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

Local computation: bound the size of recorded dependencies

Split a bucket into mini-buckets =>bound complexity

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

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

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

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

$$\mathbf{h^X \leq g^X}$$

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

Mini-Bucket Elimination

$\max_B \prod$ $\max_B \prod$

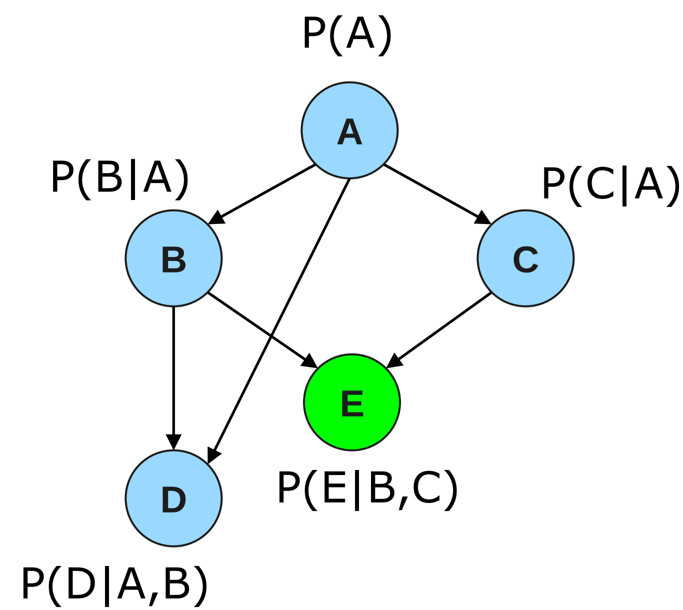
Bucket B   P(E|B,C)   P(B|A)   P(D|A,B)

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

Bucket D h^B (A,D)

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

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

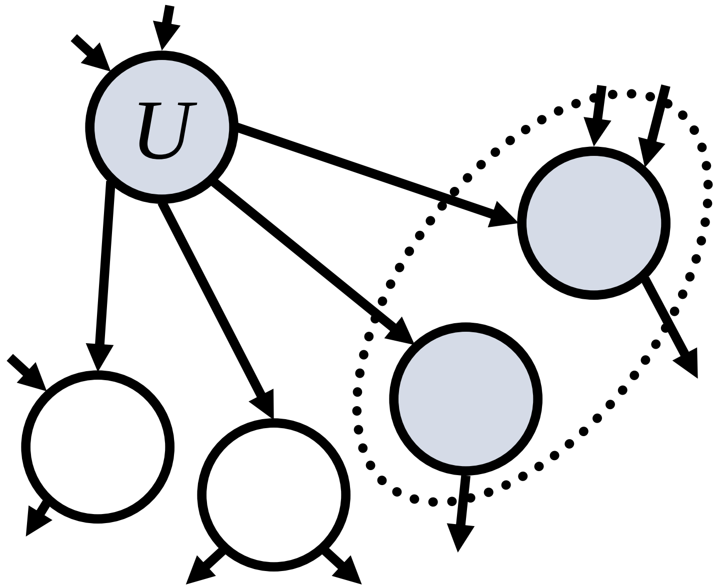


MPE* is an upper bound on MPE --U
Generating a solution yields a lower bound--L

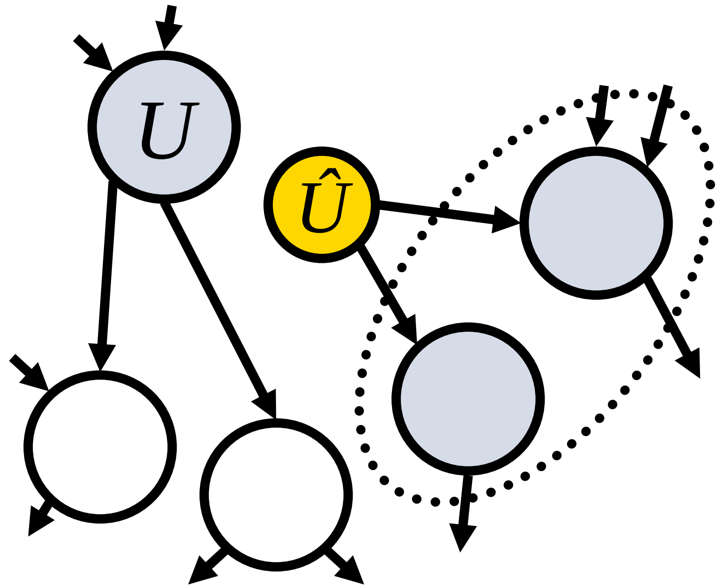
Semantics of Mini-Bucket: Splitting a Node

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

Before Splitting:
Network *N*



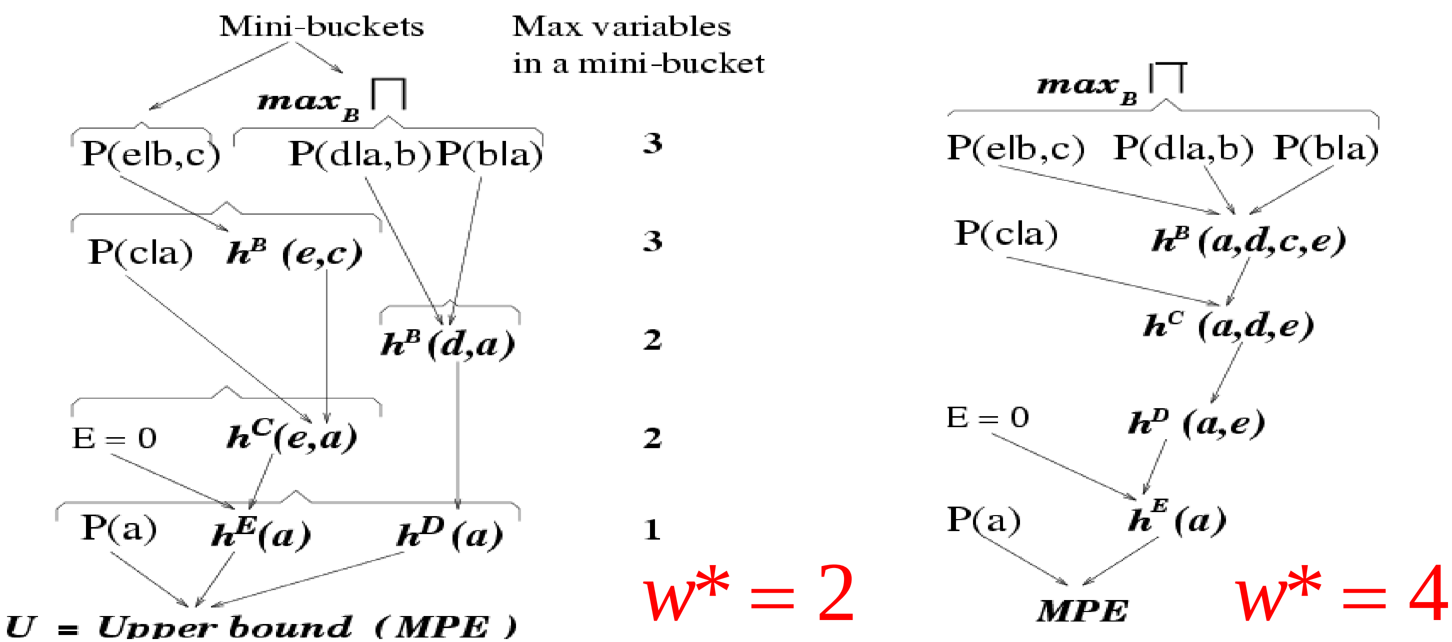
After Splitting:
Network *N'*



MBE(i) (Dechter and Rish 1997)

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

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



Properties of MBE(i)

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

Anytime Approximation

anytime-mpe(ε)

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

MBE for likelihood computation

- Idea mini-bucket is the same:

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

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

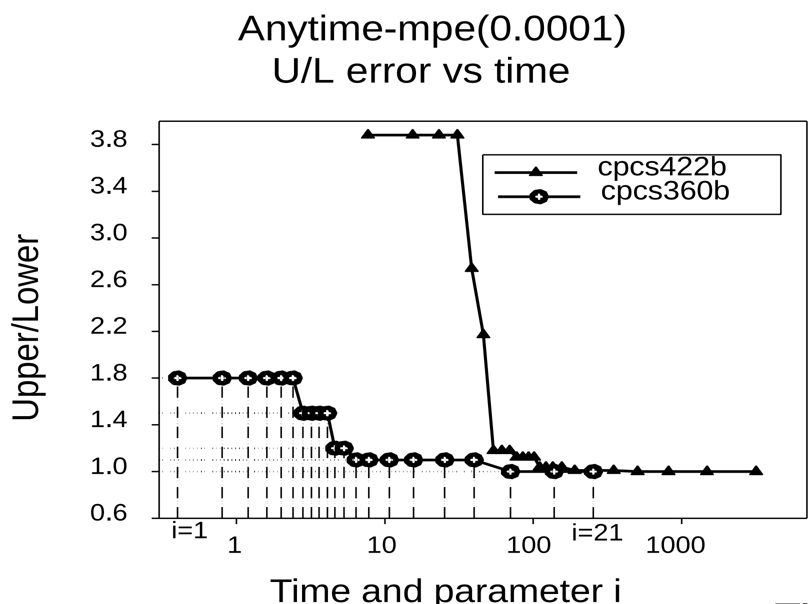
- So we can apply a sum in each mini-bucket, or better, one sum and the rest max, or min (for lower-bound)

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

- MBE-map(i,m): max buckets will be maximized, sum buckets will be sum-max. Approximates BE-map.

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

Test case: no evidence



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

Mini-Clustering (for sum-product)

Split a cluster into mini-clusters => bound complexity

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

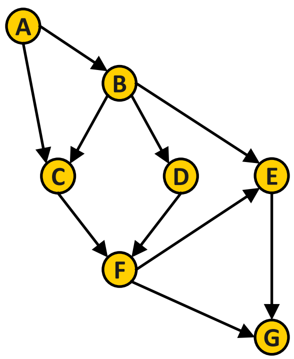
APPROXIMATE algorithm

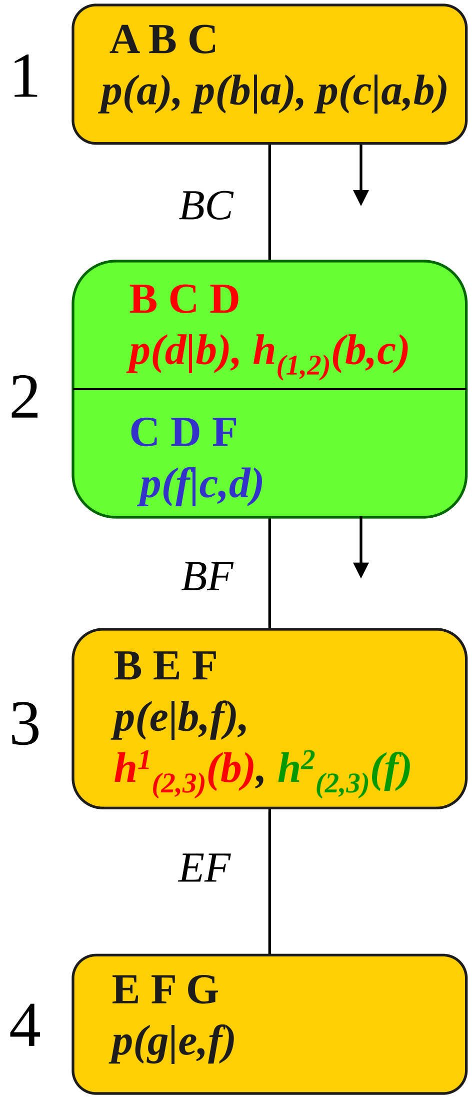
$$\underbrace{\{h_1,\dots,h_r\}} \qquad \underbrace{\{h_{r+1},\dots,h_n\}}$$

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

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

Mini-Clustering, i-bound=3





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

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

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

APPROXIMATE algorithm

Time and space:

exp(i-bound)

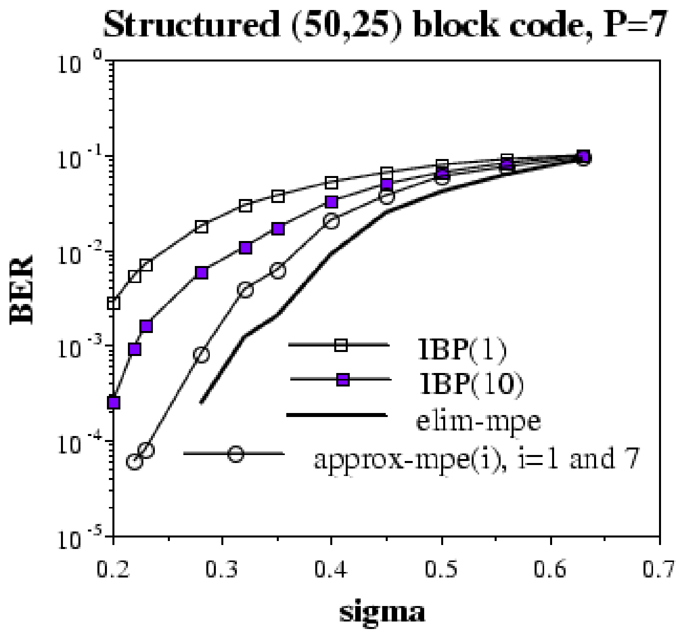
↓

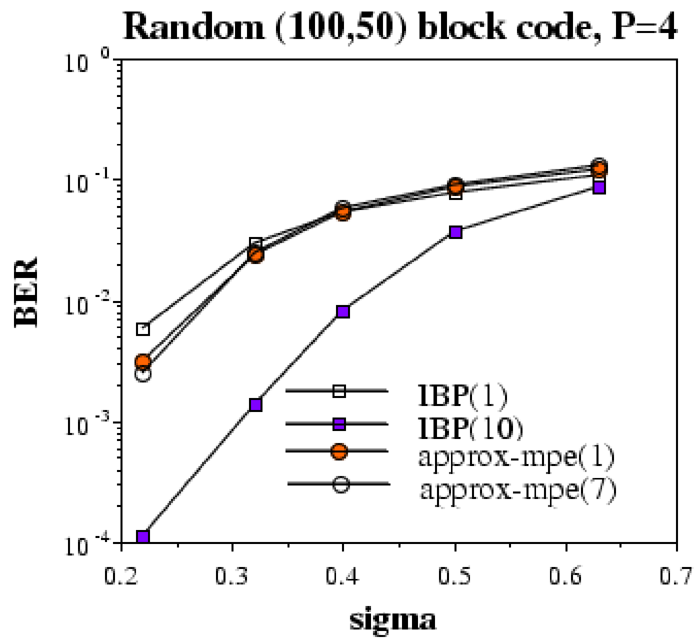
Number of variables in a mini-cluster

MBE-mpe vs. IBP

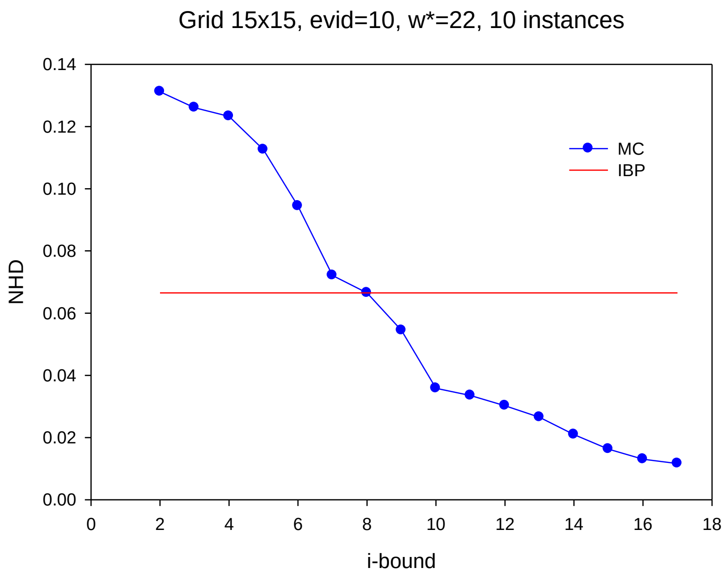
approx - mpe is better on low - w * codes
IBP is better on randomly generated (high - w *) codes

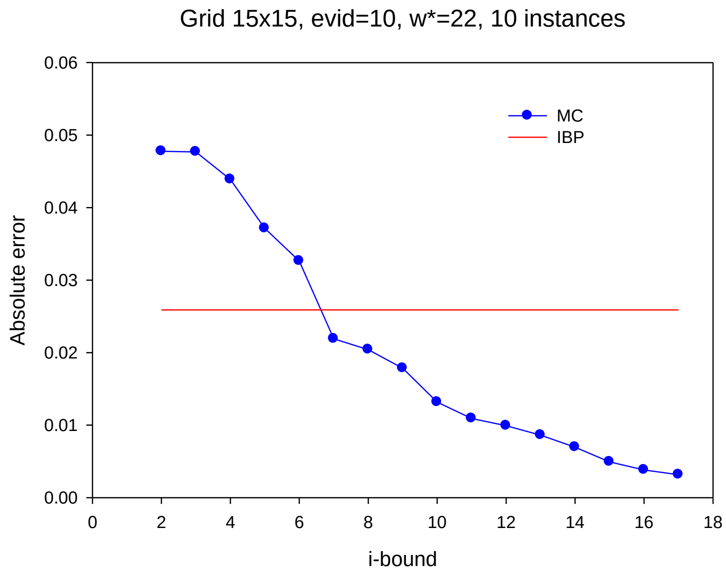
Bit error rate (BER) as a function of noise (sigma):

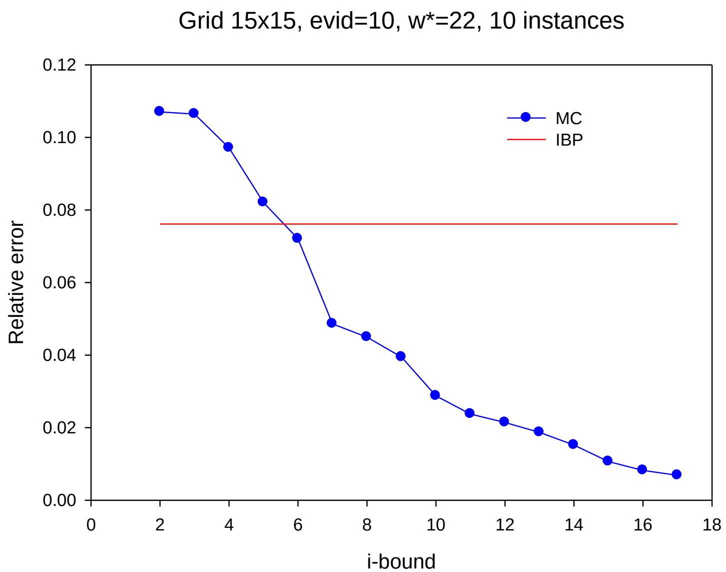


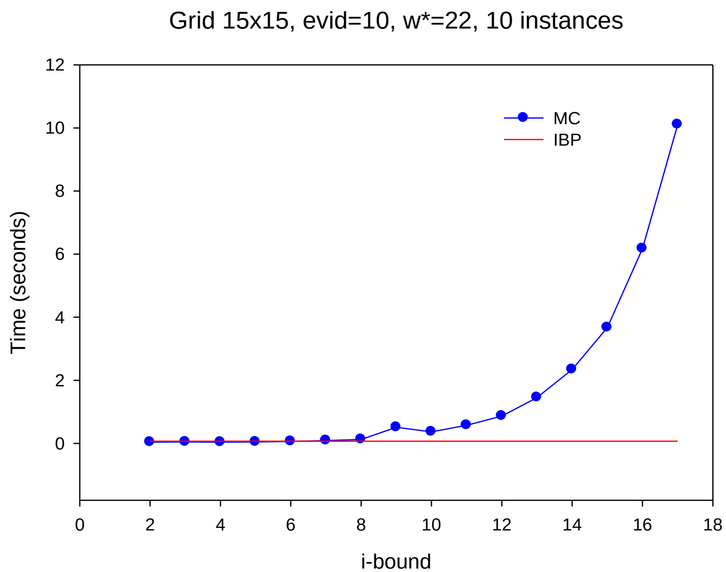


Grid 15x15 - 10 evidence





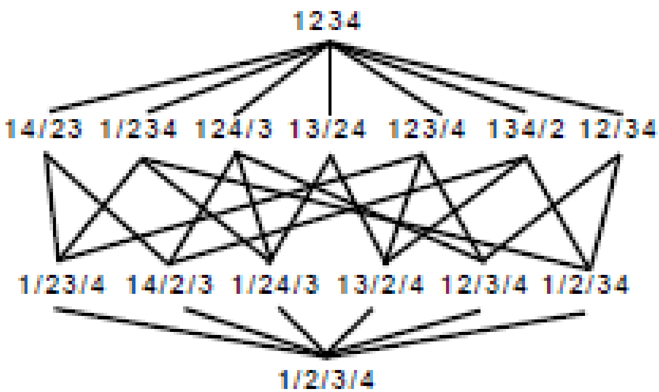




Heuristics for partitioning

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

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



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

- *Log relative error:*

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

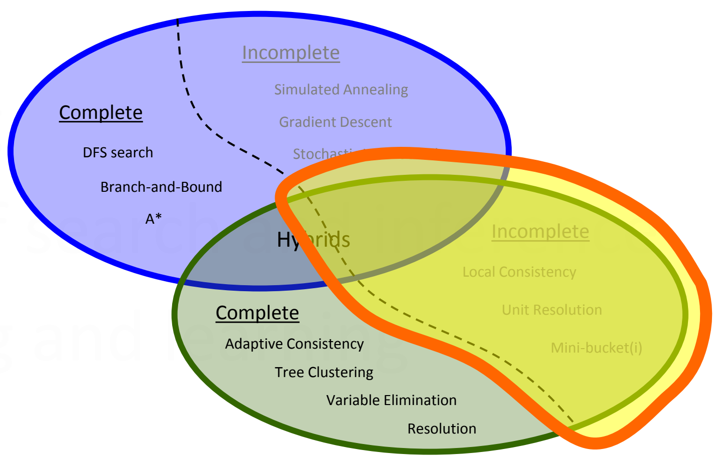
- *Max log relative error:*

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

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

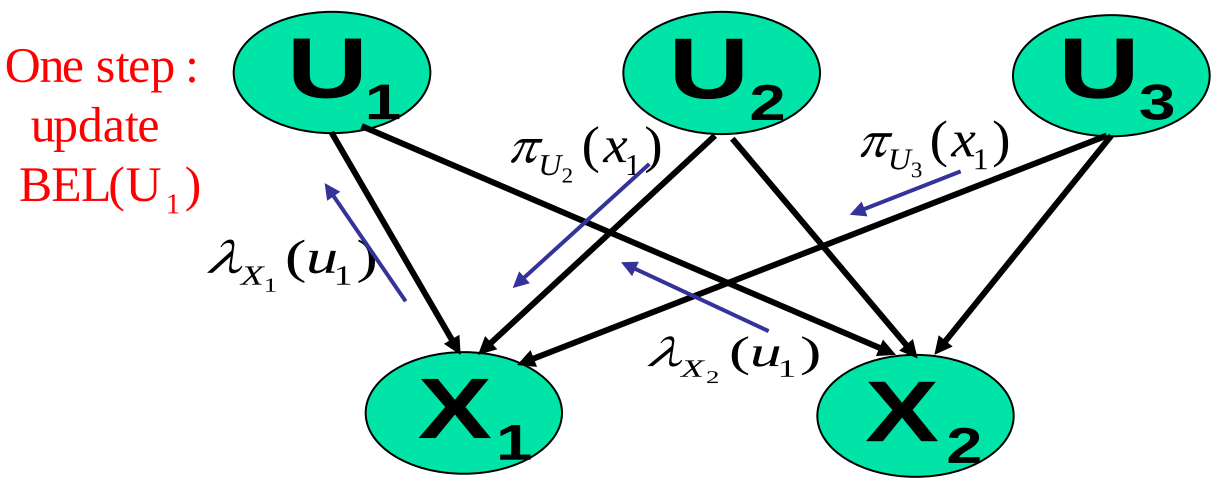
Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
  - Mini-buckets, mini-clusters
  - Belief propagation, Generalized belief propagation
- Search
- Sampling
- Software



Iterative Belief Proapagation

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

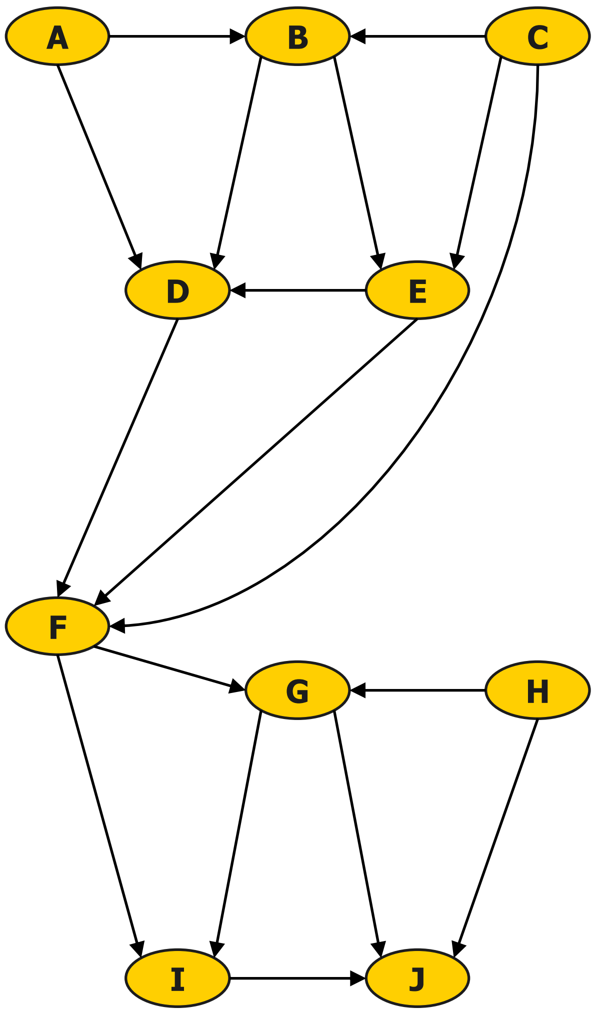


- No guarantees for convergence
- Works well for many coding networks
- Lets combine iterative-nature with anytime--IJGP

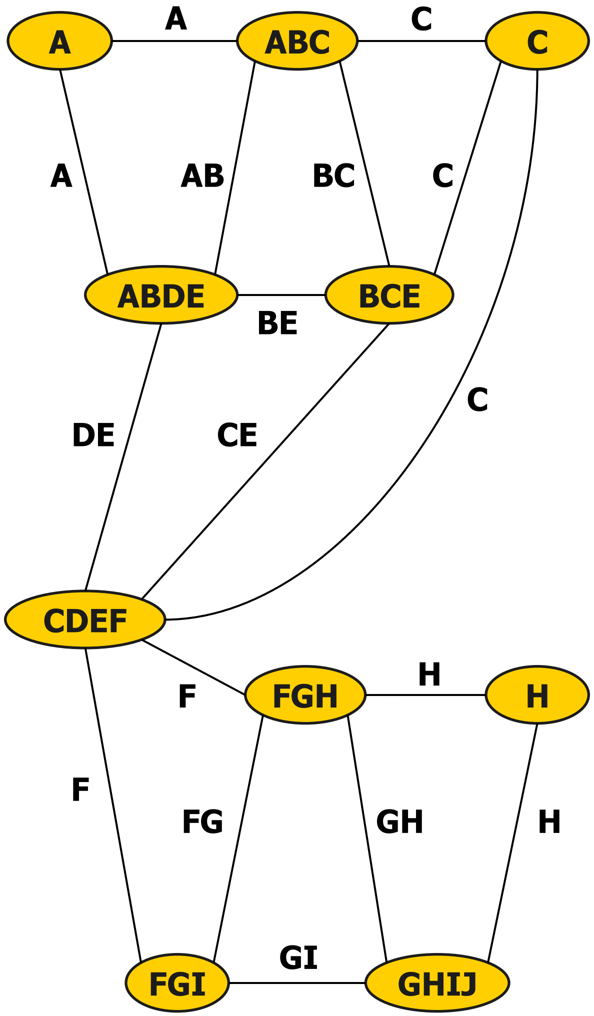
BP works on dual graph

- Need a slide saying the belief propagation operates on the dual graph

IJGP - Example

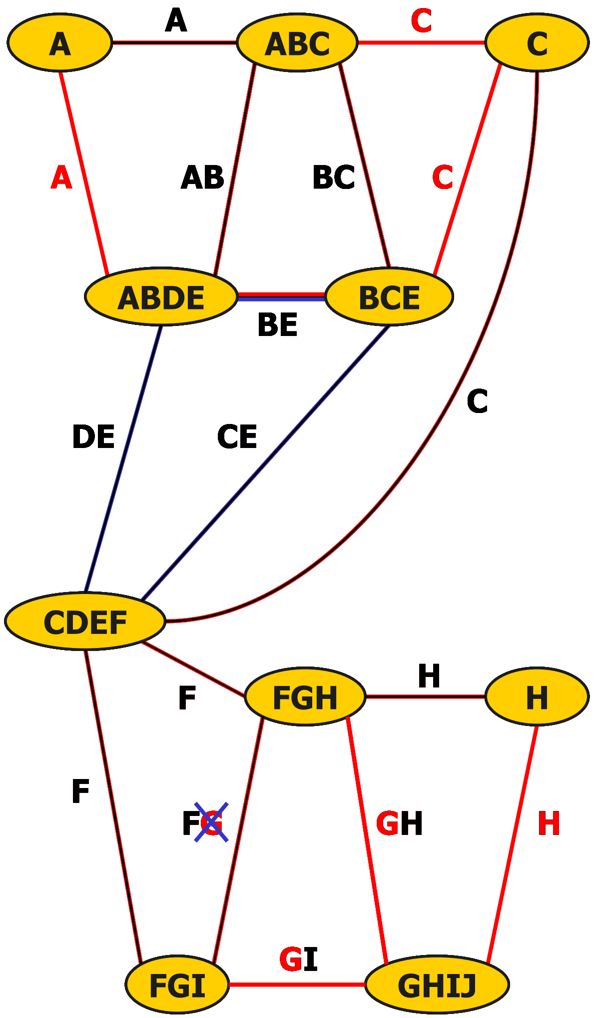


Belief network



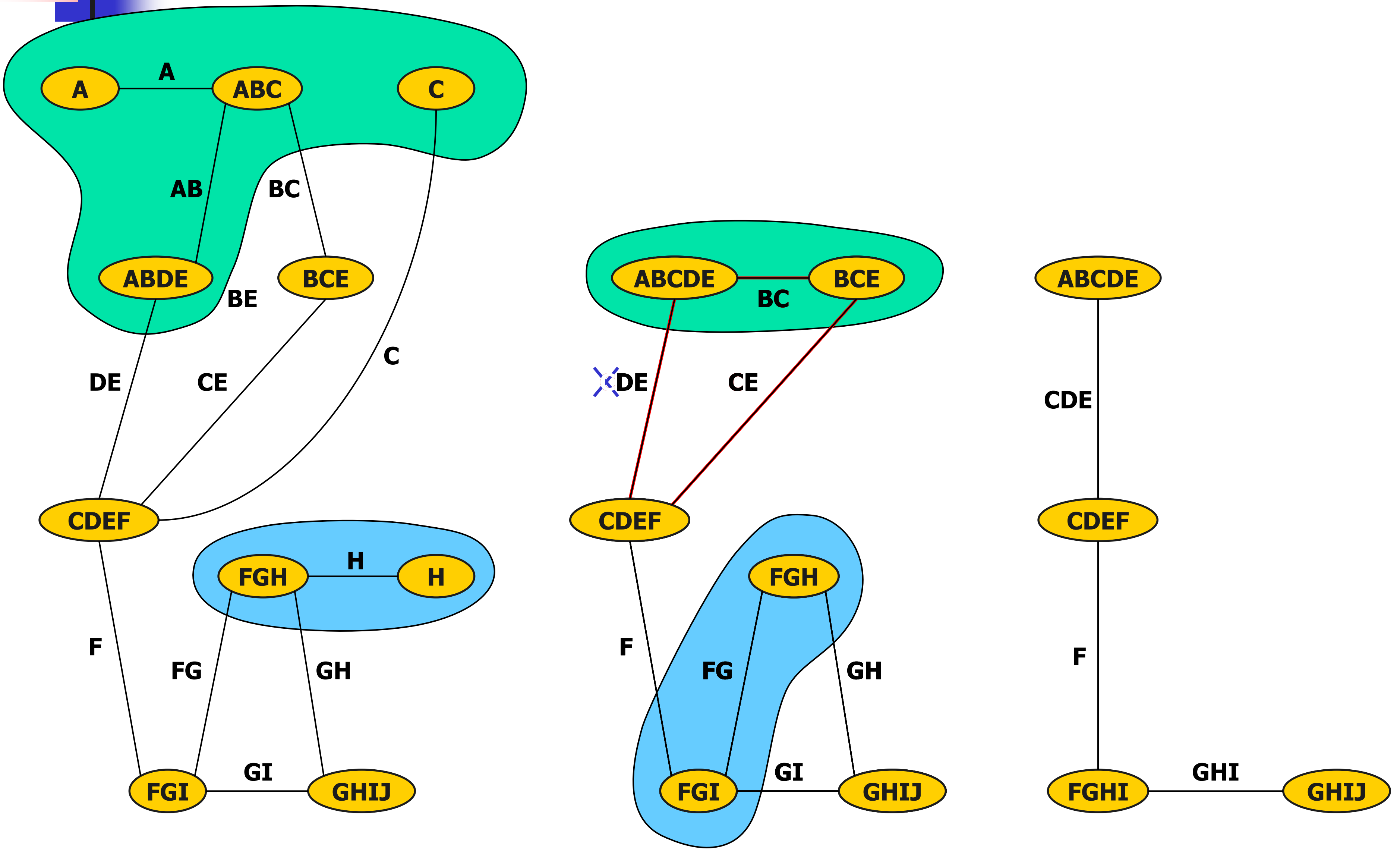
Loopy BP graph

Arc-Minimal Join-Graph

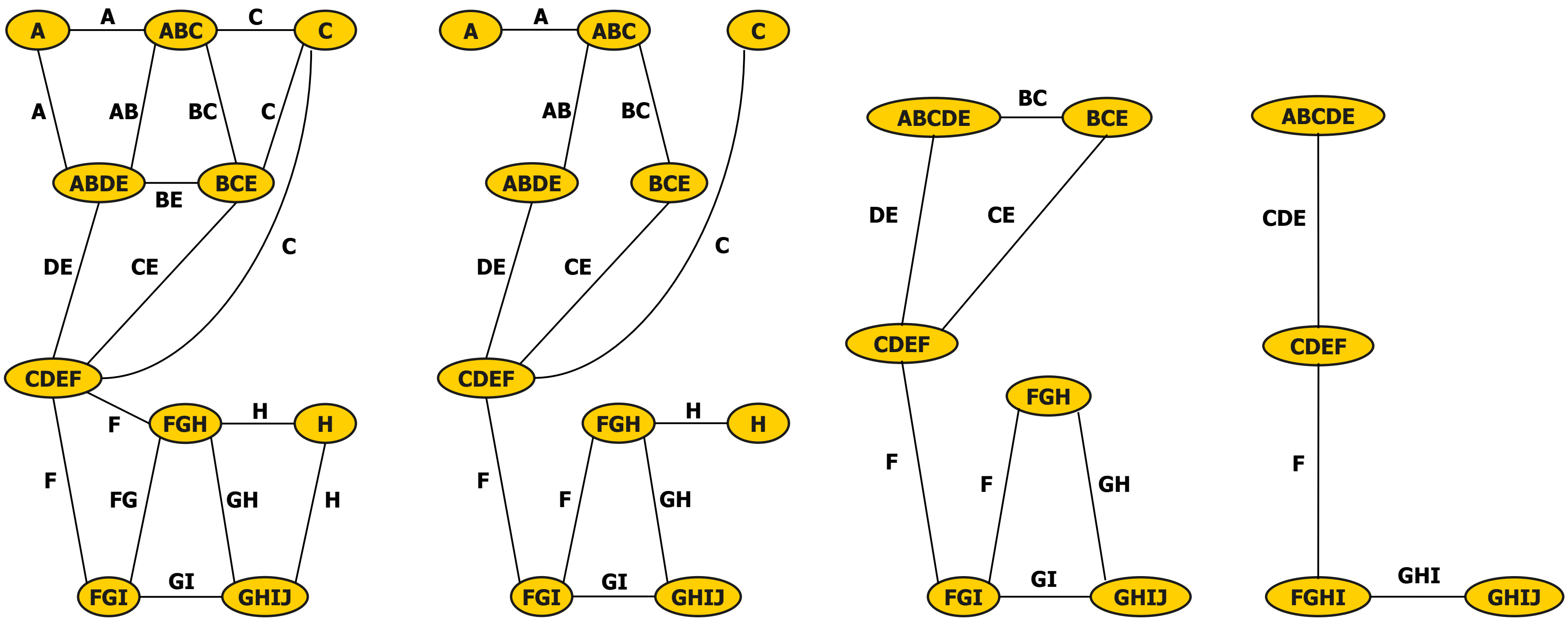


Arcs labeled with any single variable should form a TREE

Collapsing Clusters



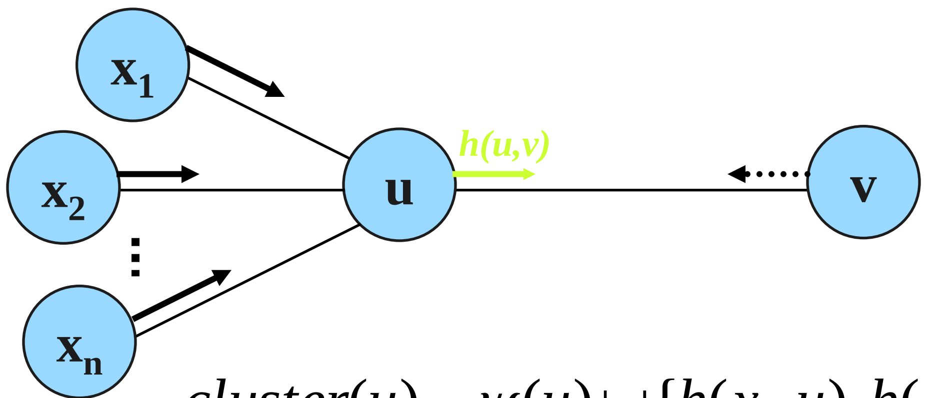
Join-Graphs



more accuracy

less complexity

Belief Propagation



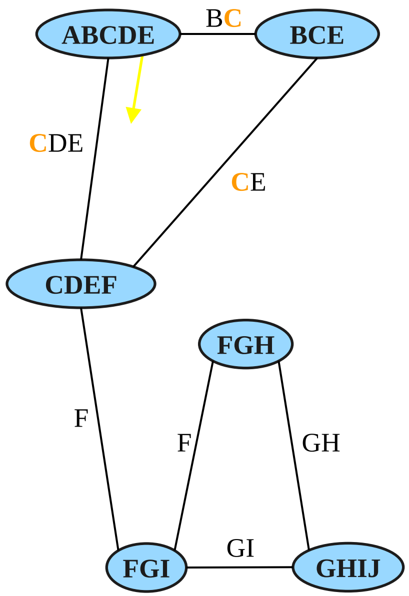
$$cluster(u) = \psi(u) \cup \{h(x_1,u), h(x_2,u), \ldots, h(x_n,u), h(v,u)\}$$

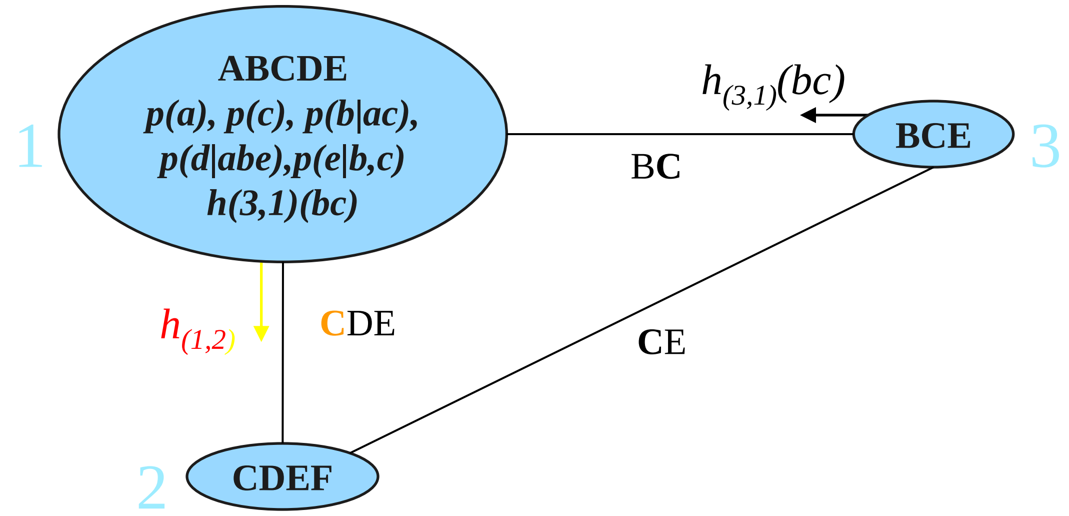
Compute the message :

$$h(u,v) = \sum_{elim(u,v)} \prod_{f \in cluster(u) - \{h(v,u)\}} f$$

For max-product: IJGP replaces summation with maximization

Message propagation





Minimal arc-labeled:

sep(1,2)={D,E}

elim(1,2)={A,B,C}

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

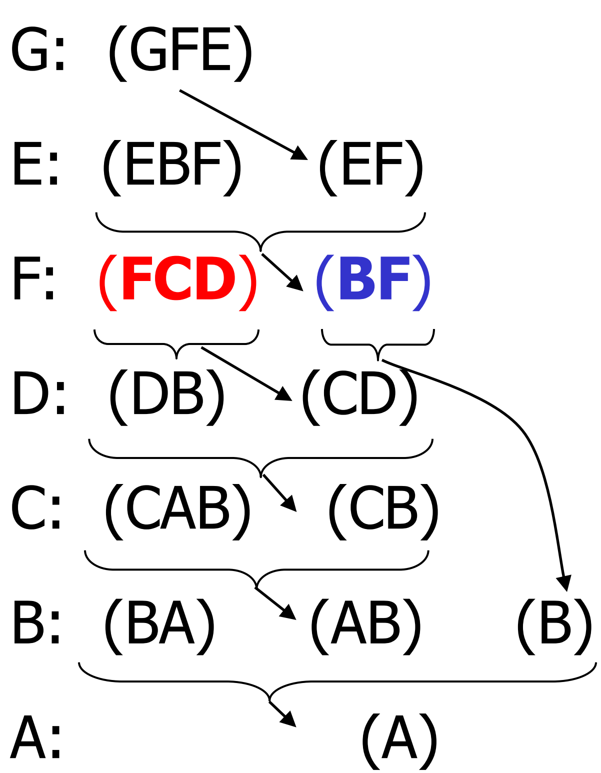
Non-minimal arc-labeled:

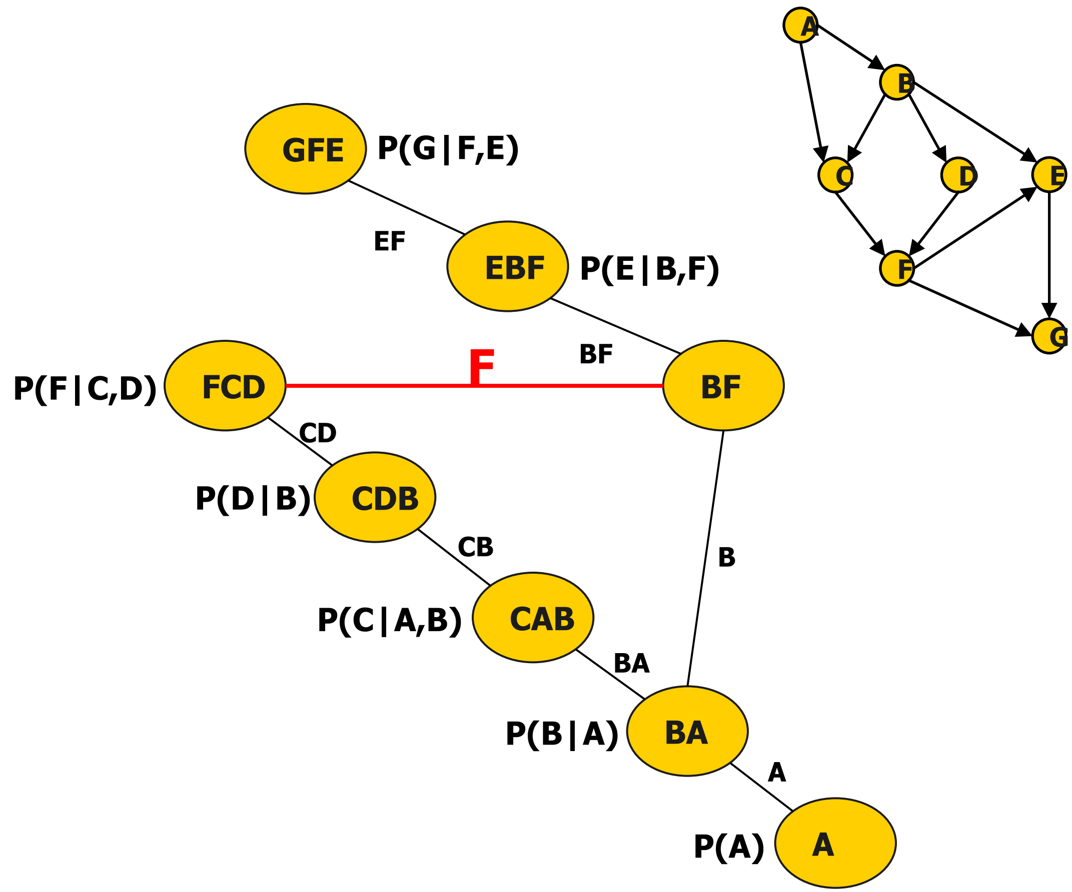
sep(1,2)={C,D,E}

elim(1,2)={A,B}

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

Constructing Join-Graphs

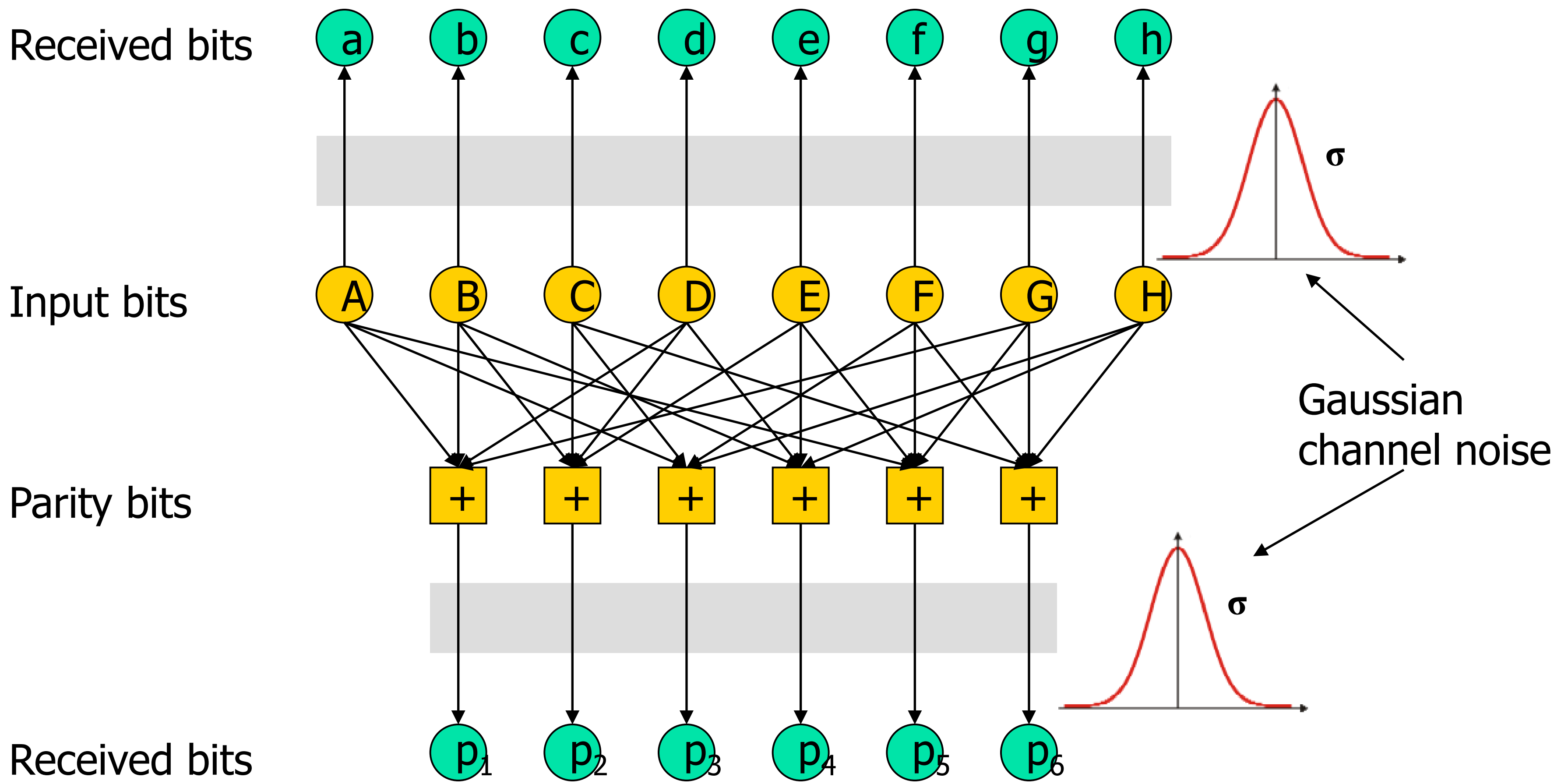




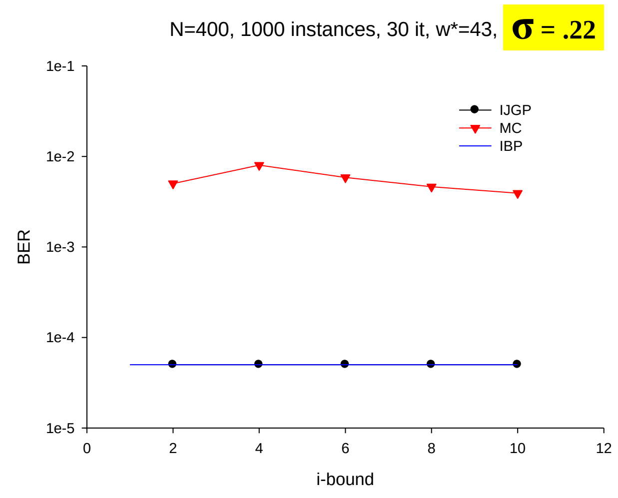
a) schematic mini-bucket(i), i=3

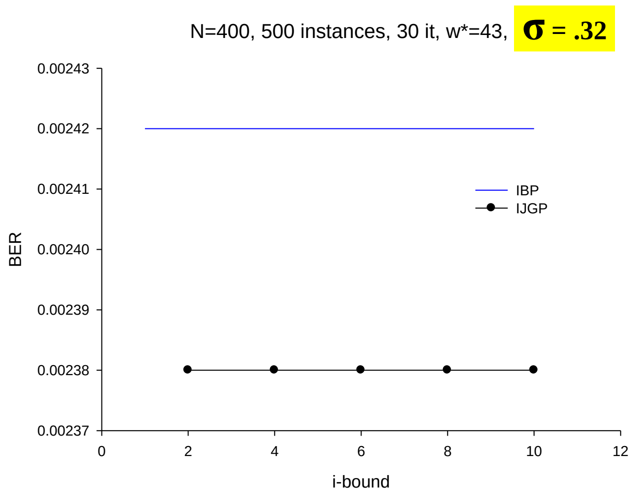
b) arc-labeled join-graph decomposition

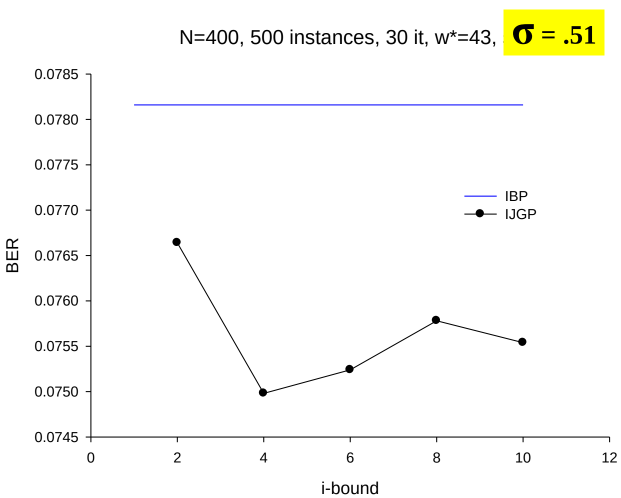
Linear Block Codes

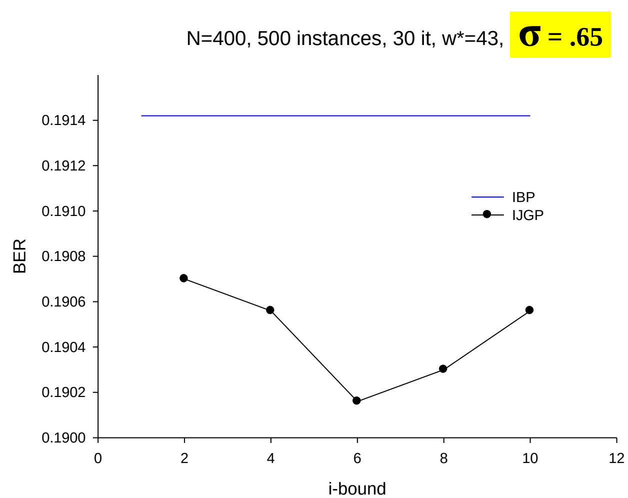


Coding Networks – Bit Error Rate

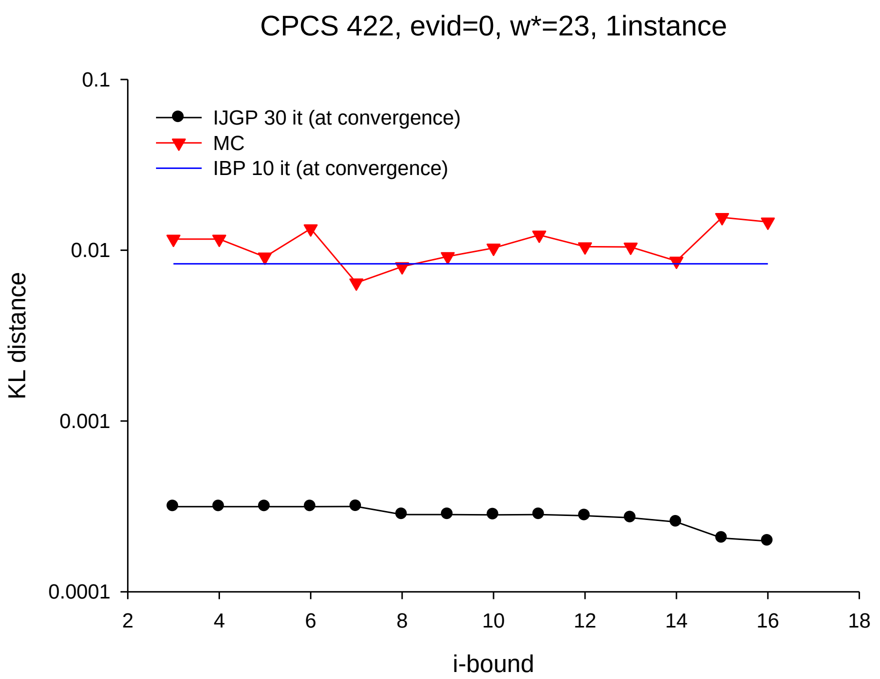




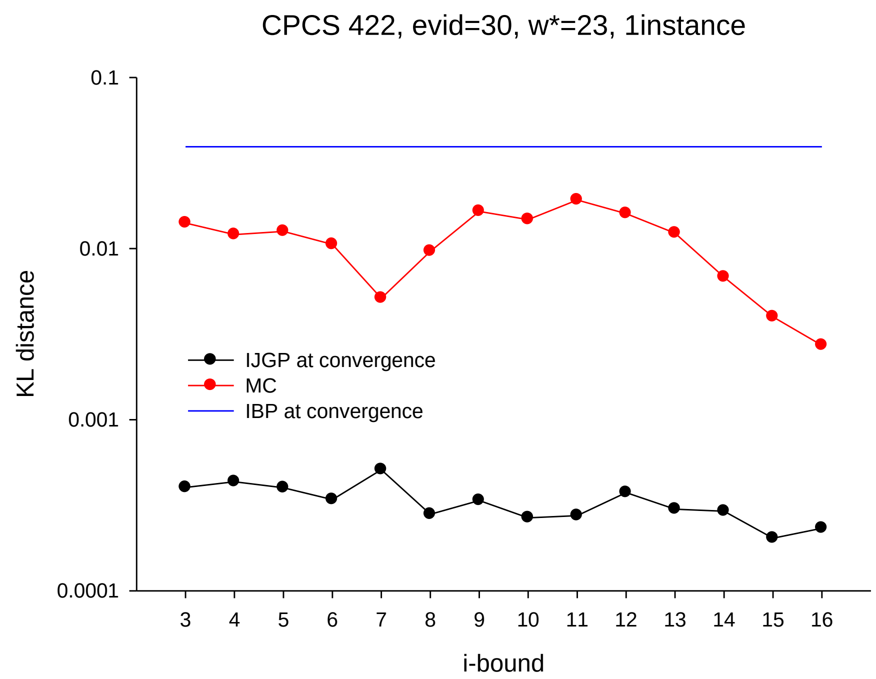




CPCS 422 – KL Distance

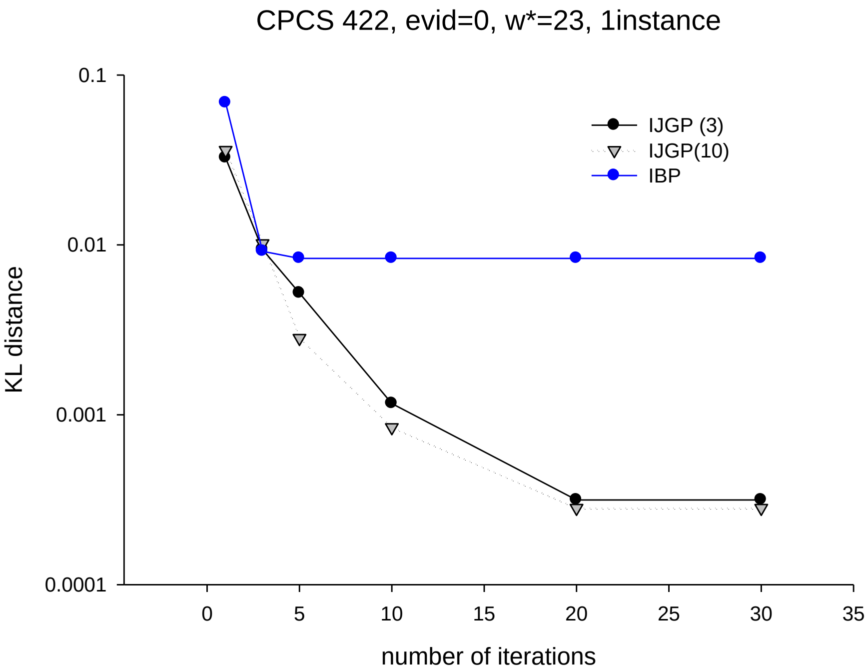


evidence=0

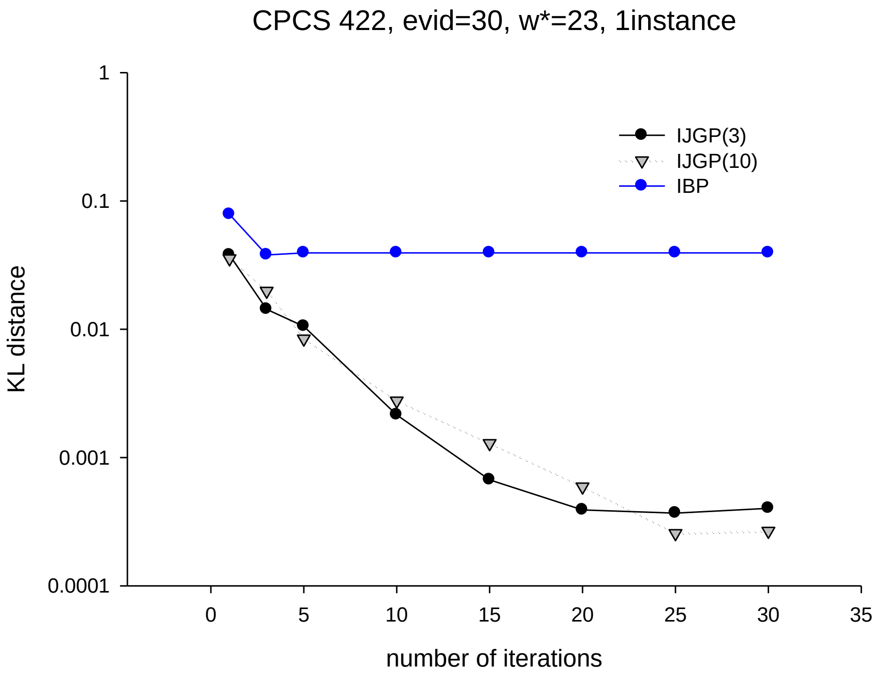


evidence=30

CPCS 422 – KL vs. Iterations



evidence=0



evidence=30

More On the Power of Belief Propagation

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

Optimizing the KL-Divergence

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

Therse results also extend to generalizzed BP

Summary of IJGP so far

A spectrum of approximations.

IBP: results from applying IJGP to the dual joingraph.

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

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

Constraint networks

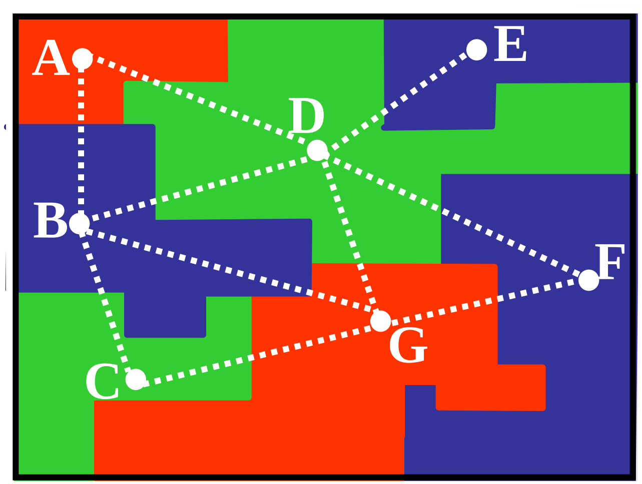
Map coloring

Variables: countries (A B C etc.)

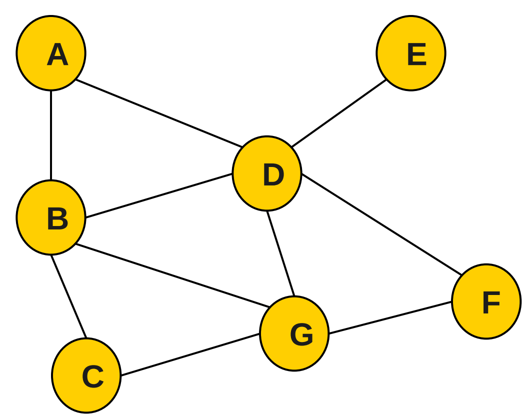
Values: colors (red green blue)

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

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

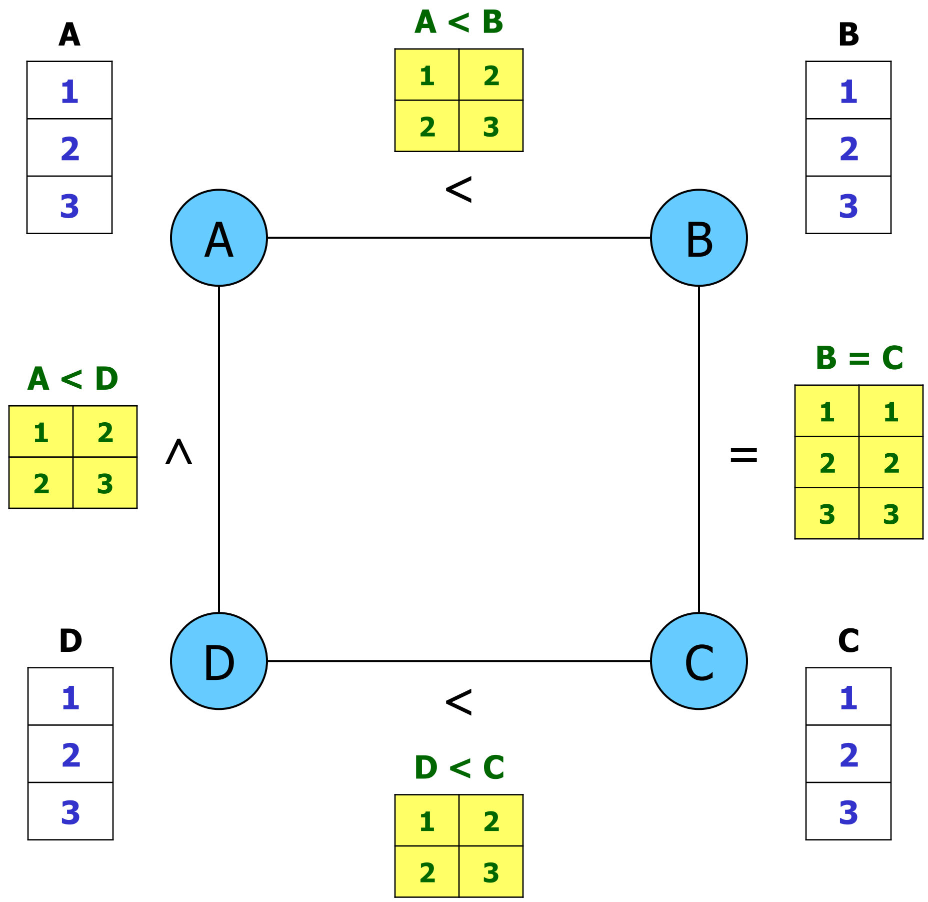


Constraint graph

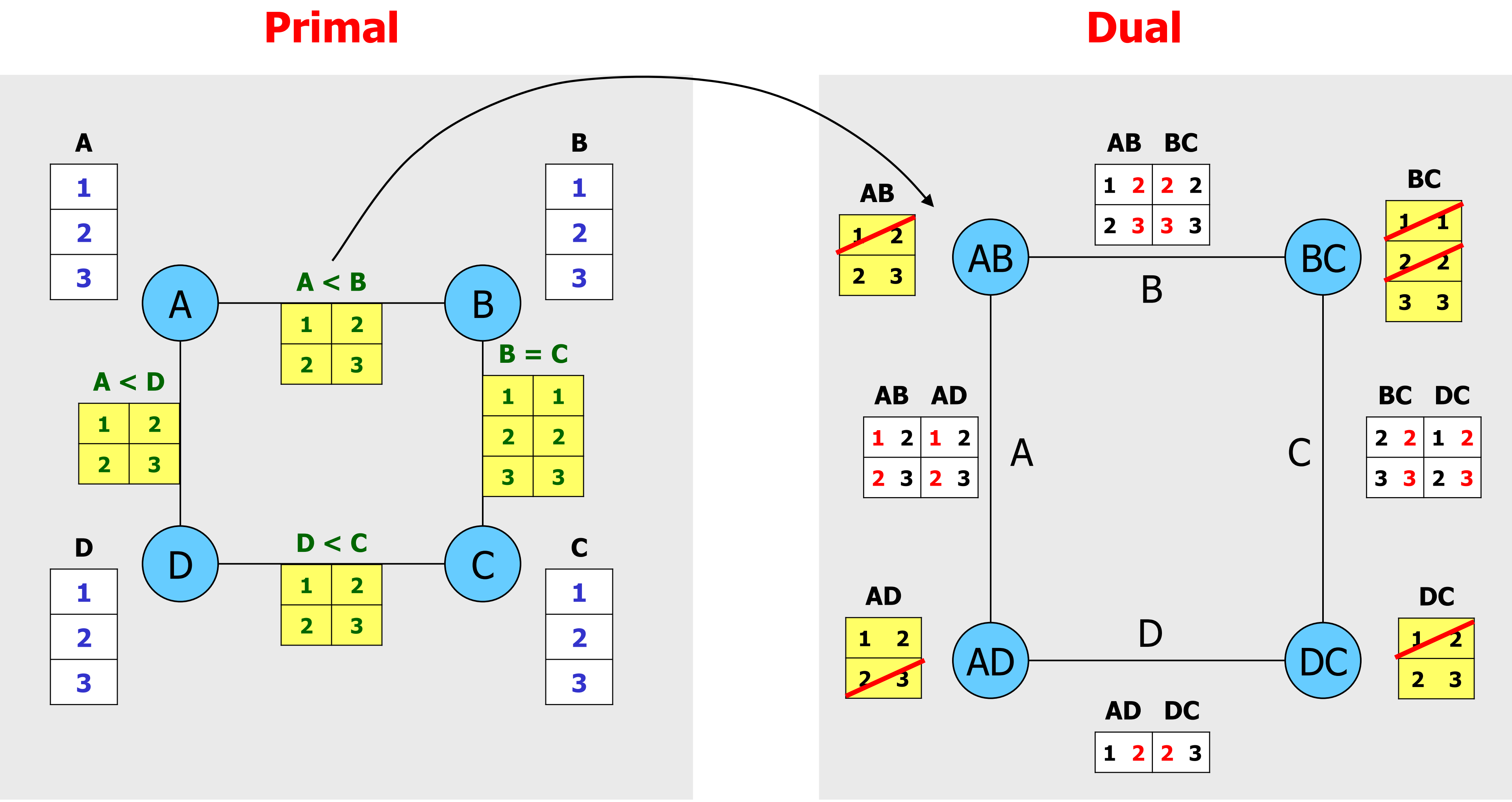


Arc-consistency

- Sound
- Incomplete
- Always converges (polynomial)



Relational Distributed Arc-Consistency

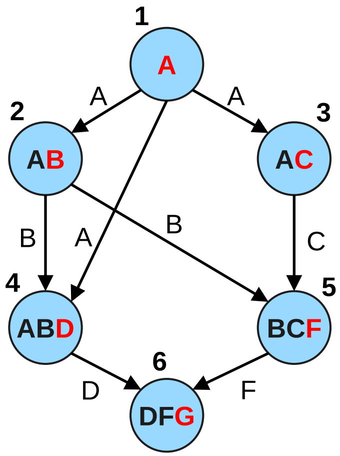


Flattening the Bayesian Network

A	P(A)
1	.2
2	.5
3	.3
…	0

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

A	C	P(C|A)
1	2	1
3	2	1
…	…	0



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

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

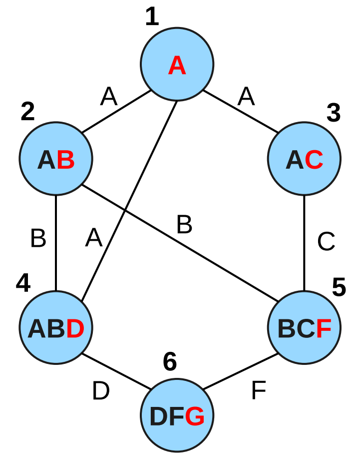
D	F	G	P(G|D,F)
1	2	3	1
2	1	3	1
…	…	…	0

Belief network

A
1
2
3

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

A	C
1	2
3	2



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

B	C	F
1	2	3
3	2	1

D	F	G
1	2	3
2	1	3

Flat constraint network

IBP – inference power for zero beliefs

- **Theorem:**

  Trace of zero beliefs of Iterative Belief Propagation =
  Trace of invalid tuples of arc-consistency on flat network

- **Soundness:**
  - The inference of zero beliefs by Loopy BP converges in a finite number of iterations
  - all the inferred zero beliefs are correct

- **Incompleteness:**
  - Loopy BP may not infer all the true zero beliefs

Properties of ijgp

- Properties of the sum-product algorithm
  - If/when the algorithm converges, the covergence is a stationary point of the KL distance to the posterior distribution
- Properties of the max-product algorithm
  - If the max-marginals agree…

IJGP summary

- IJGP borrows the iterative feature from IBP and the anytime virtues of bounded inference from MC
- Empirical evaluation showed the potential of IJGP, which improves with iteration and most of the time with i-bound, and scales up to large networks
- IJGP is almost always superior, often by a high margin, to IBP and MC
- Based on all our experiments, we think that IJGP provides a practical breakthrough to the task of belief updating

Exact Reasoning by Search

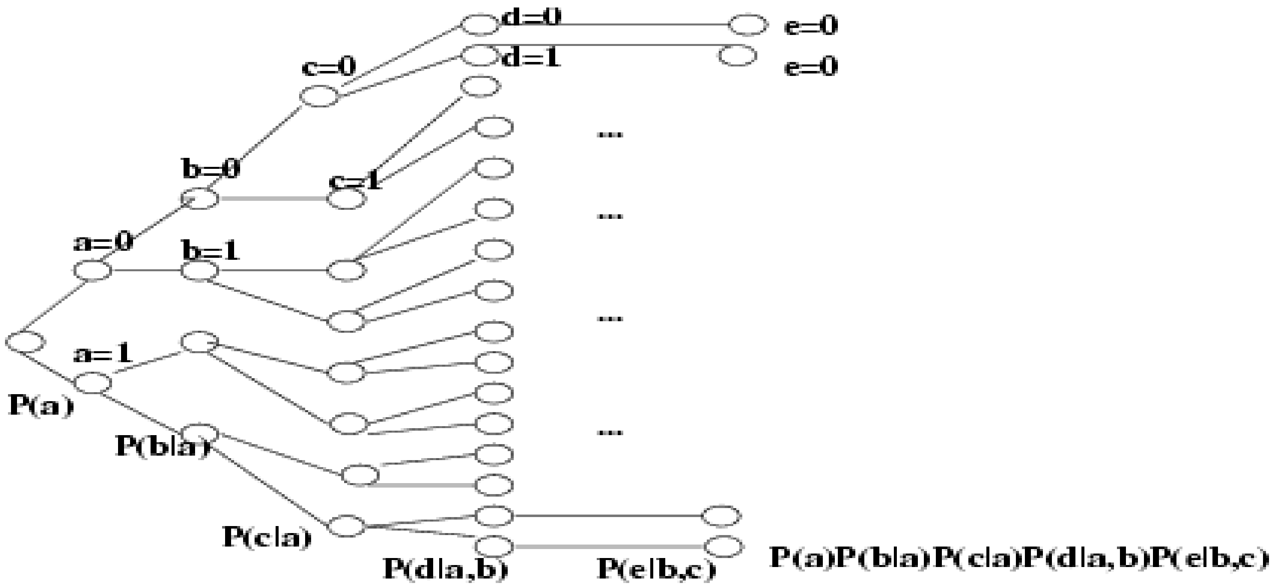
- Why consider search?
- Can we do any better in search?
- Can we combine search and inference?

Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
- Search
- Sampling
- Hybrid of search and inference
- Modeling and learning
- Software

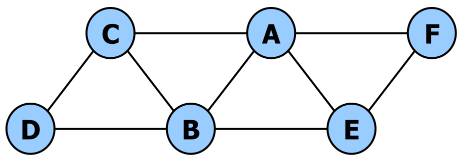
Conditioning generates the probability tree

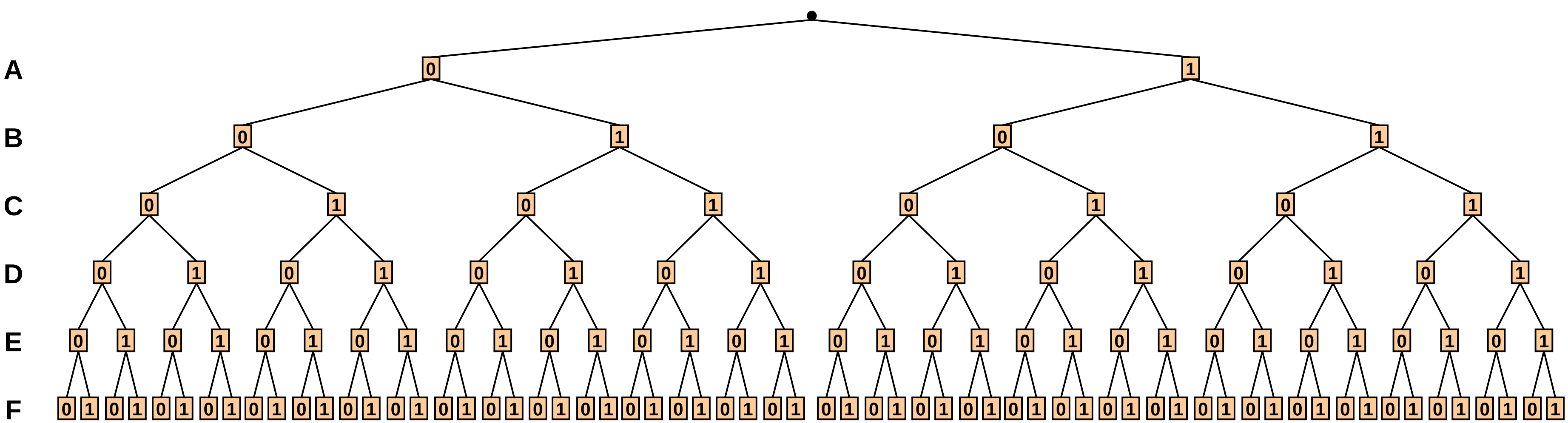
$$P(a, e=0) = P(a)\sum_{b} P(b \mid a)\sum_{c} P(c \mid a)\sum_{b} P(d \mid a, b)\sum_{e=0} P(e \mid b, c)$$



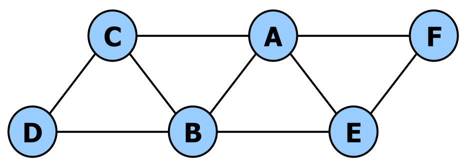
Complexity of conditioning: exponential time, linear space

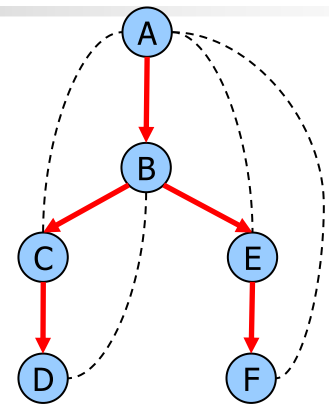
Classic OR Search Space



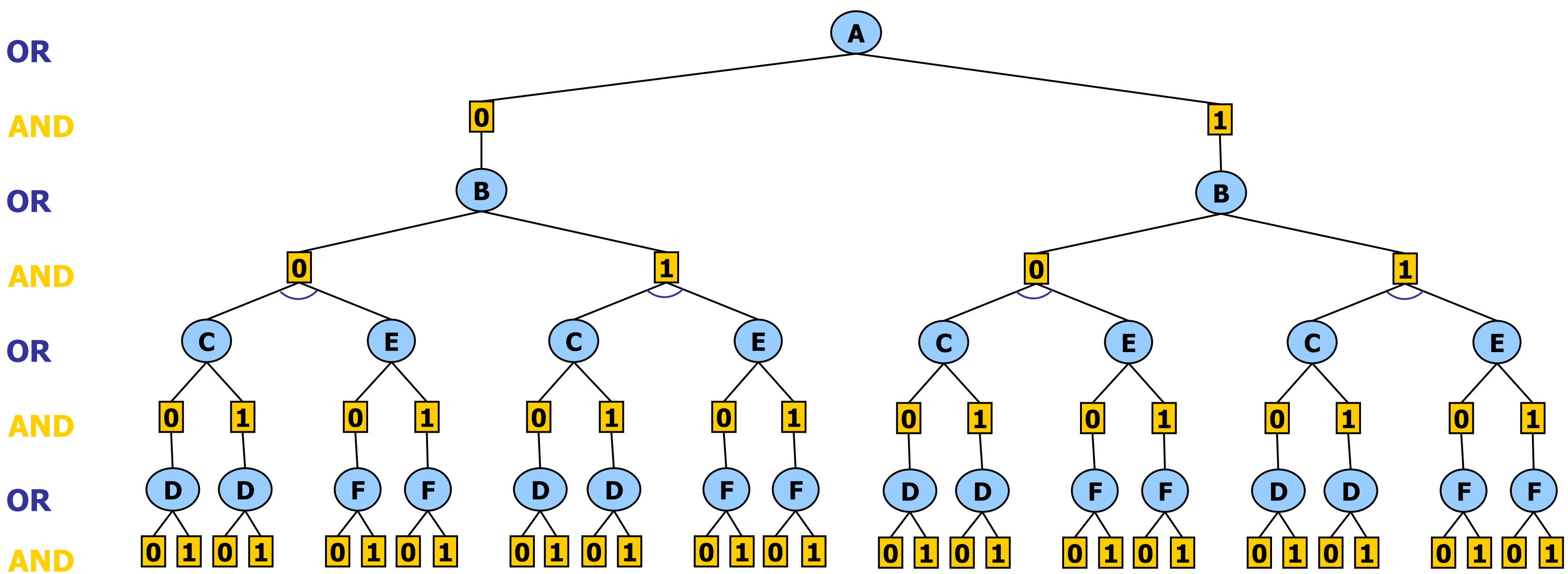


The AND/OR Search Tree

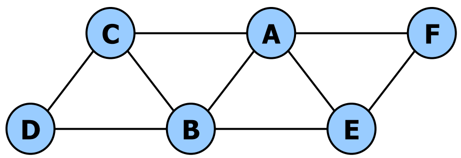


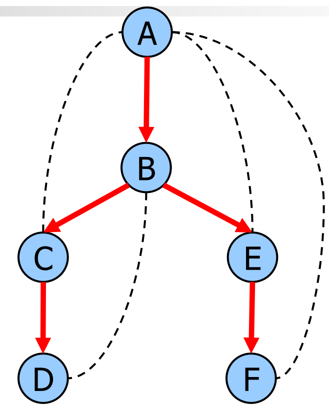


Pseudo tree (Freuder and Quinn,IJCAI85)

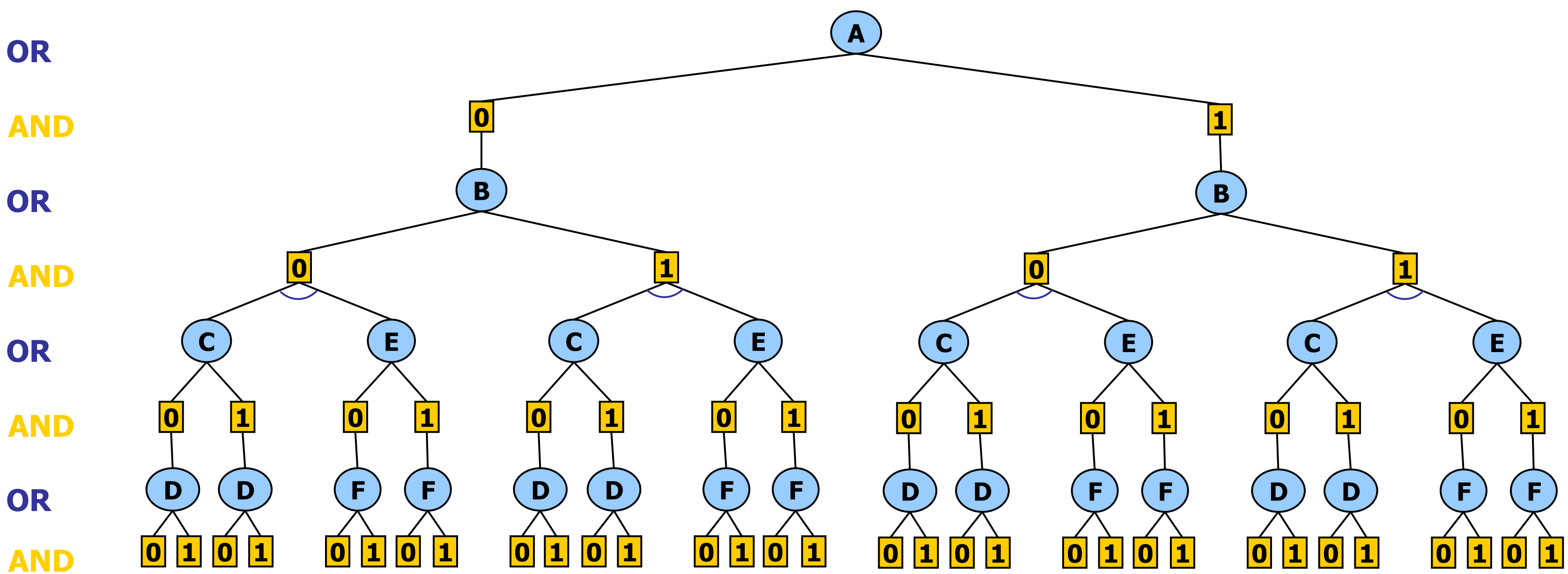


The AND/OR Search Tree





Pseudo tree



A solution subtree is (A=0, B=1, C=0, D=0, E=1, F=1)

Weighted AND/OR Tree

$P(E \mid A, B)$

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

Evidence: E=0

$P(B \mid A)$

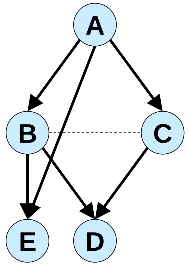
A	B=0	B=1
0	.4	.6
1	.1	.9

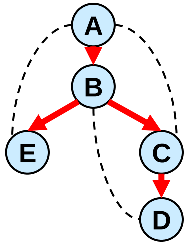
$P(C \mid A)$

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

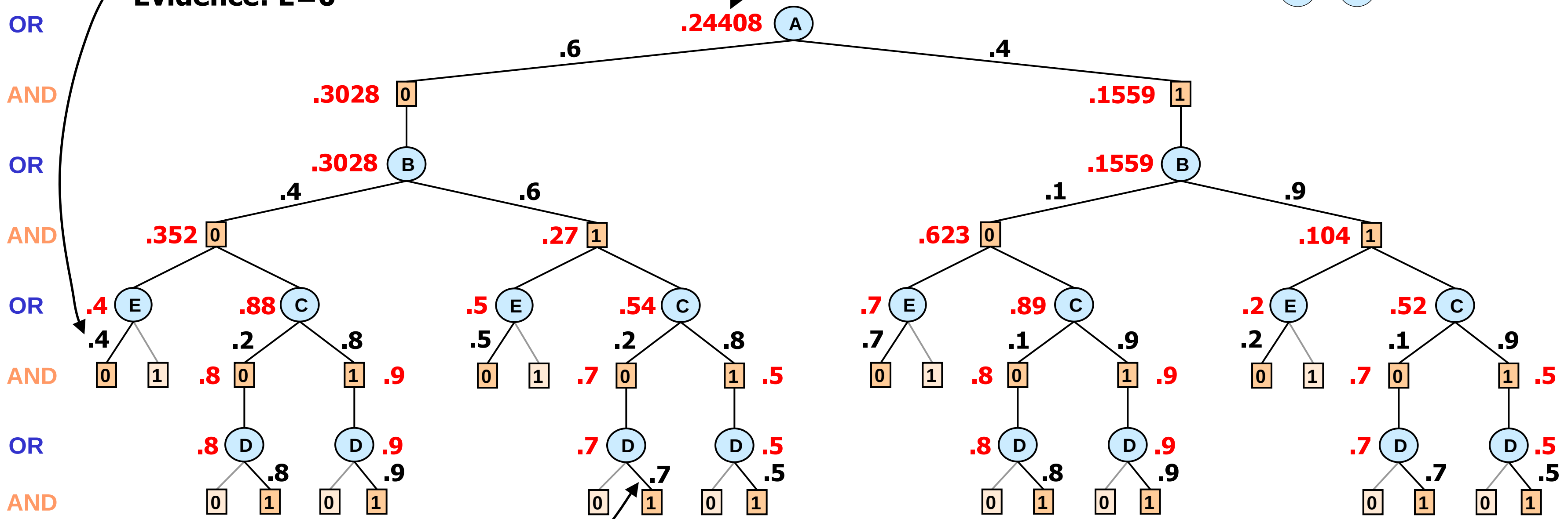
$P(A)$

A	P(A)
0	.6
1	.4





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



$P(D \mid B, C)$

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

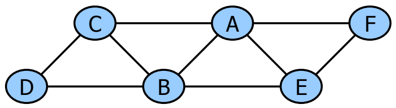
Evidence: D=1

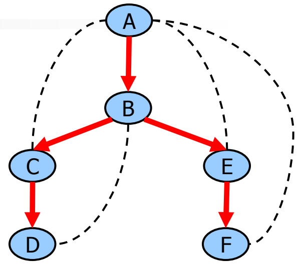
OR node: Marginalization operator (summation)

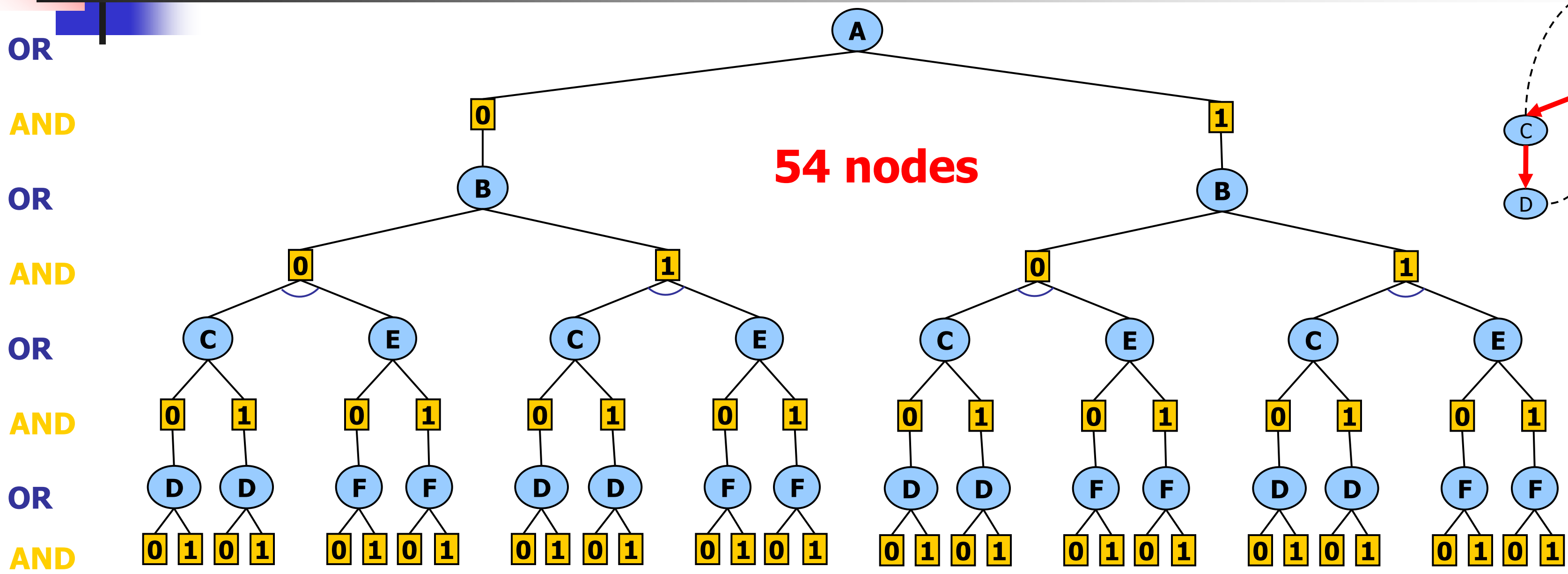
AND node: Combination operator (product)

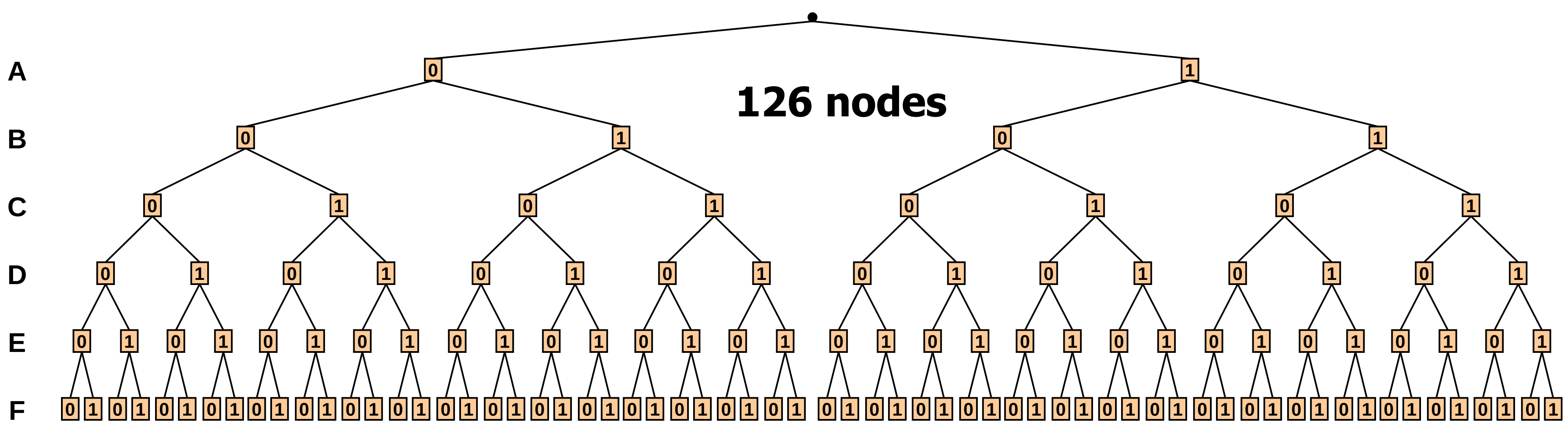
Value of node = updated belief for sub-problem below

AND/OR vs. OR Spaces









AND/OR vs. OR Spaces

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

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

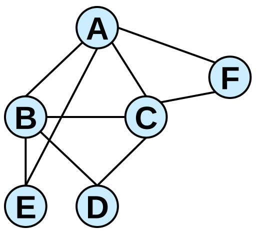
Complexity of AND/OR Tree Search

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

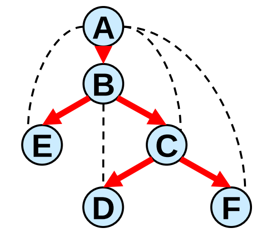
d = domain size
t = depth of pseudo-tree
n = number of variables
w* = treewidth

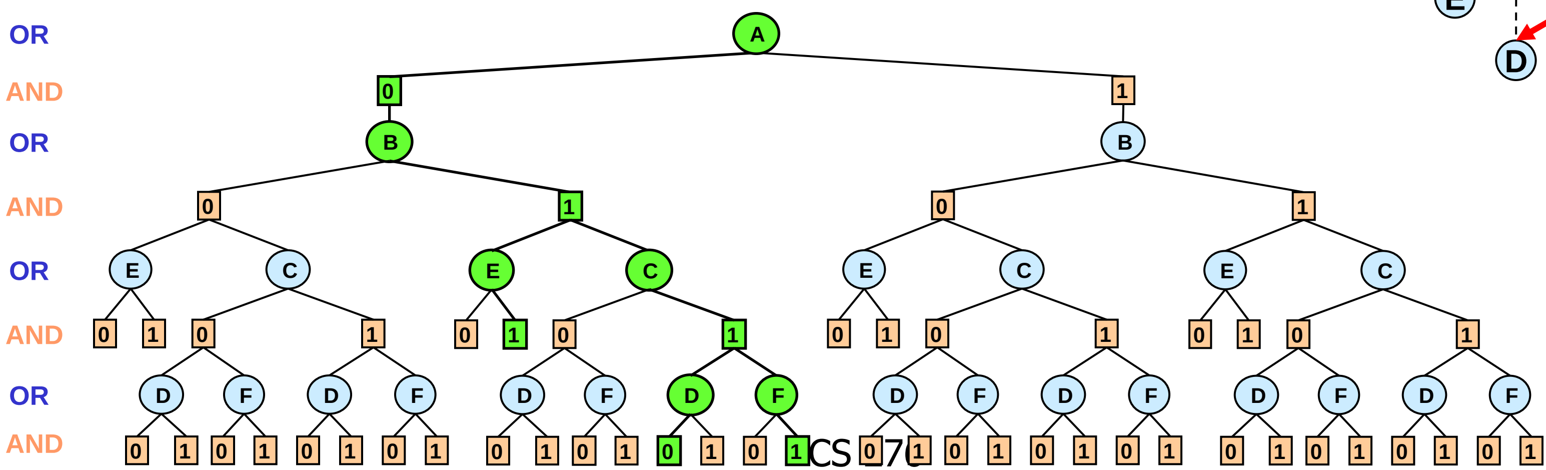
AND/OR search tree for graphical models

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



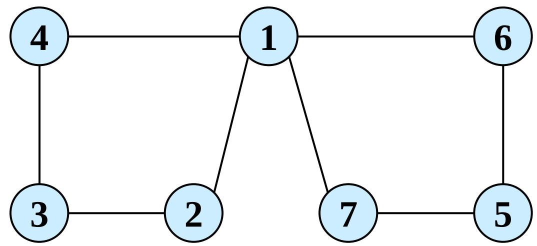
- **A solution is a consistent subtree**
- **Task: compute the value of the root node**





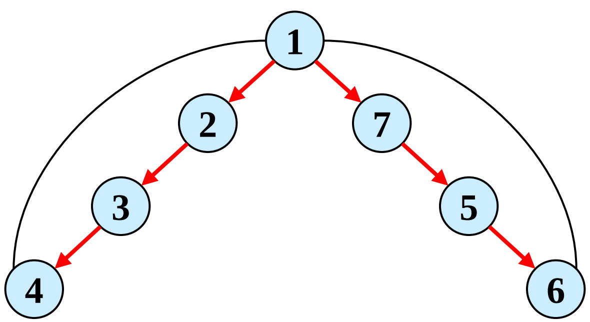
Pseudo-Trees

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

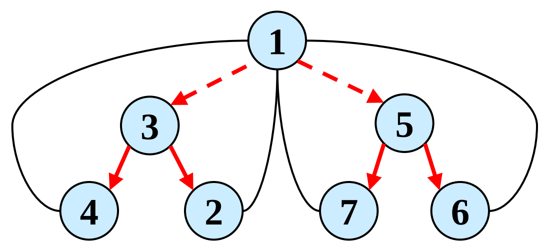


(a) Graph

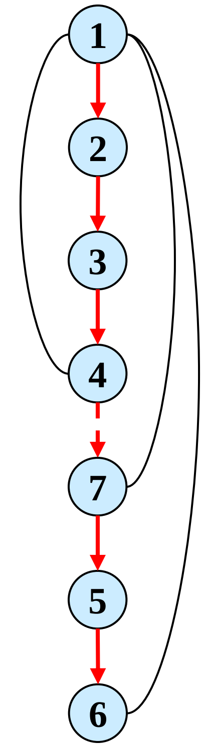
$t <= w^* \log n$



(b) DFS tree
depth=3



(c) pseudo- tree
depth=2



(d) Chain
depth=6

AND/OR tree search (P(evidence))

Weighted AND/OR
Has weights on arcs

$P(E \mid A,B)$

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

$P(B \mid A)$

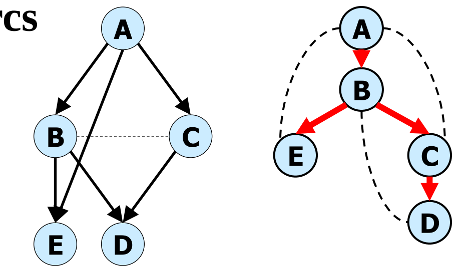
A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C \mid A)$

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

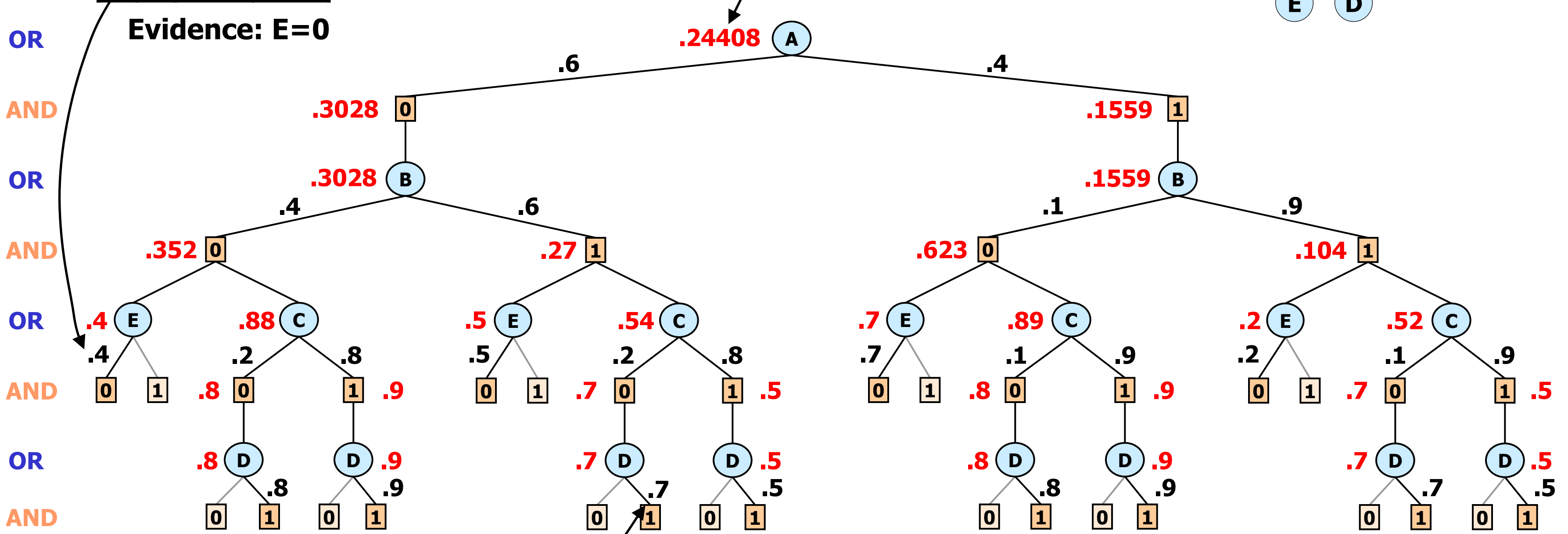
$P(A)$

A	P(A)
0	.6
1	.4



Evidence: E=0

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



$P(D \mid B,C)$

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

Evidence: D=1

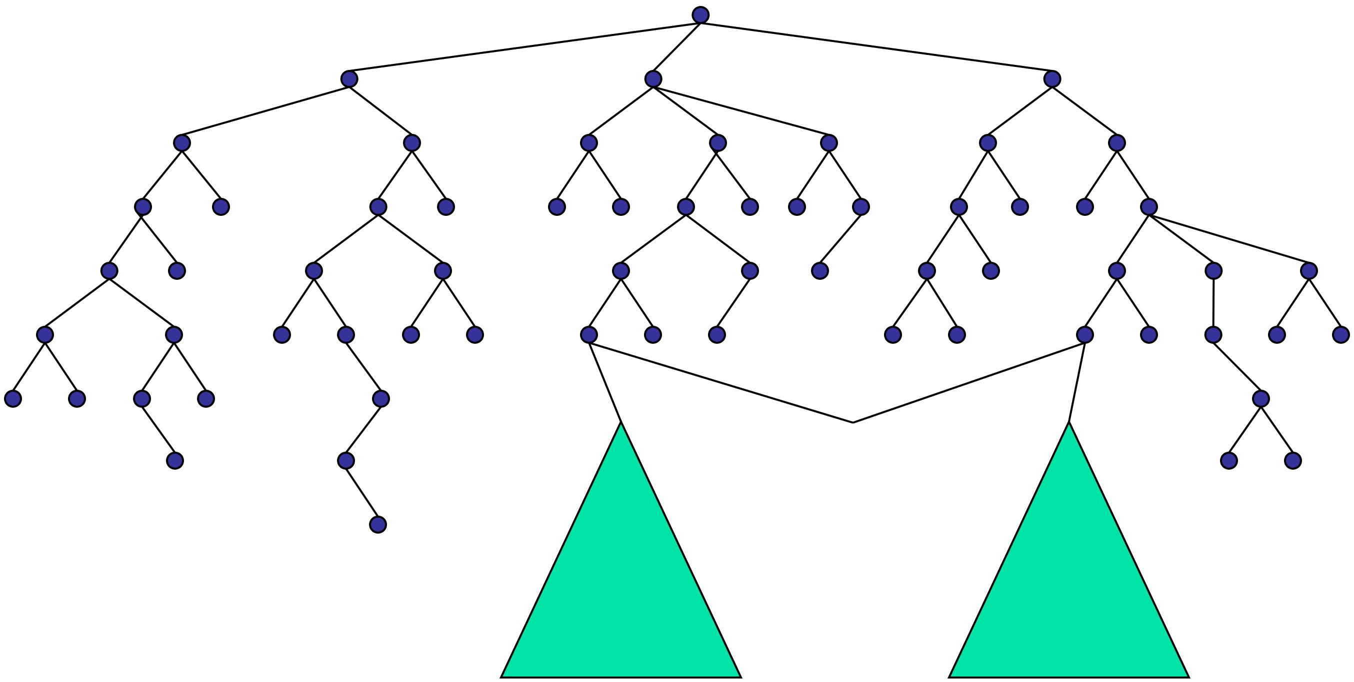
OR node: Marginalization operator (summation)

AND node: Combination operator (product)

Value of node = updated belief for subproblem below

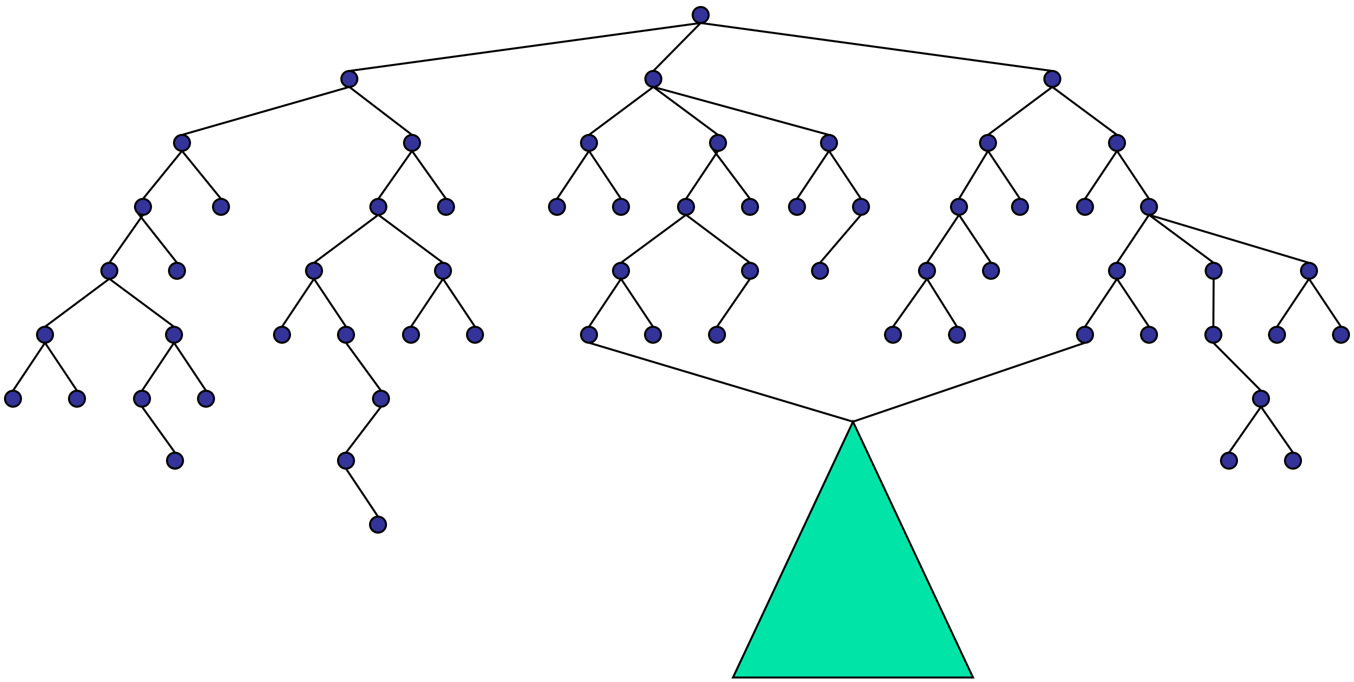
From search trees to search graphs

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

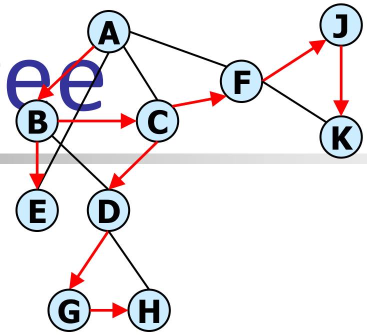


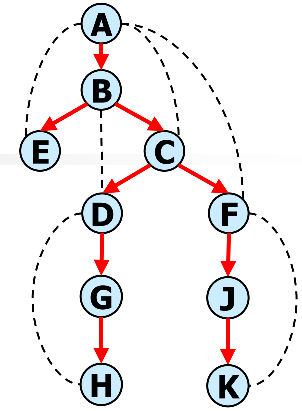
From search trees to search graphs

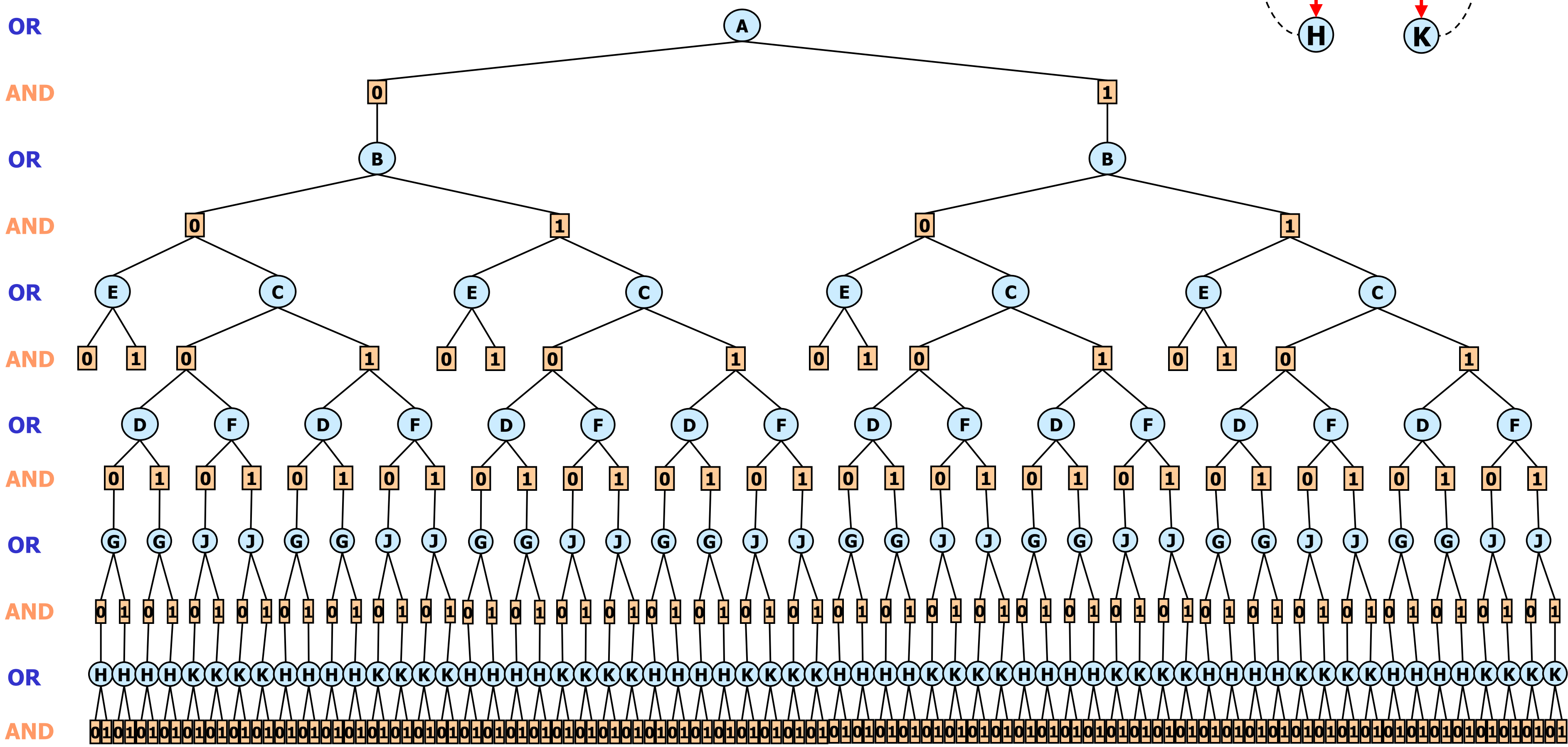
- Any two nodes that root identical subtrees (subgraphs) can be merged



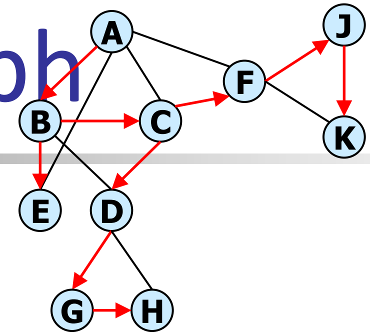
From AND/OR Tree

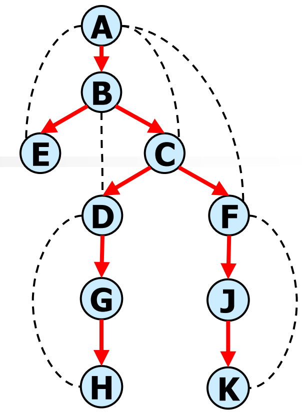


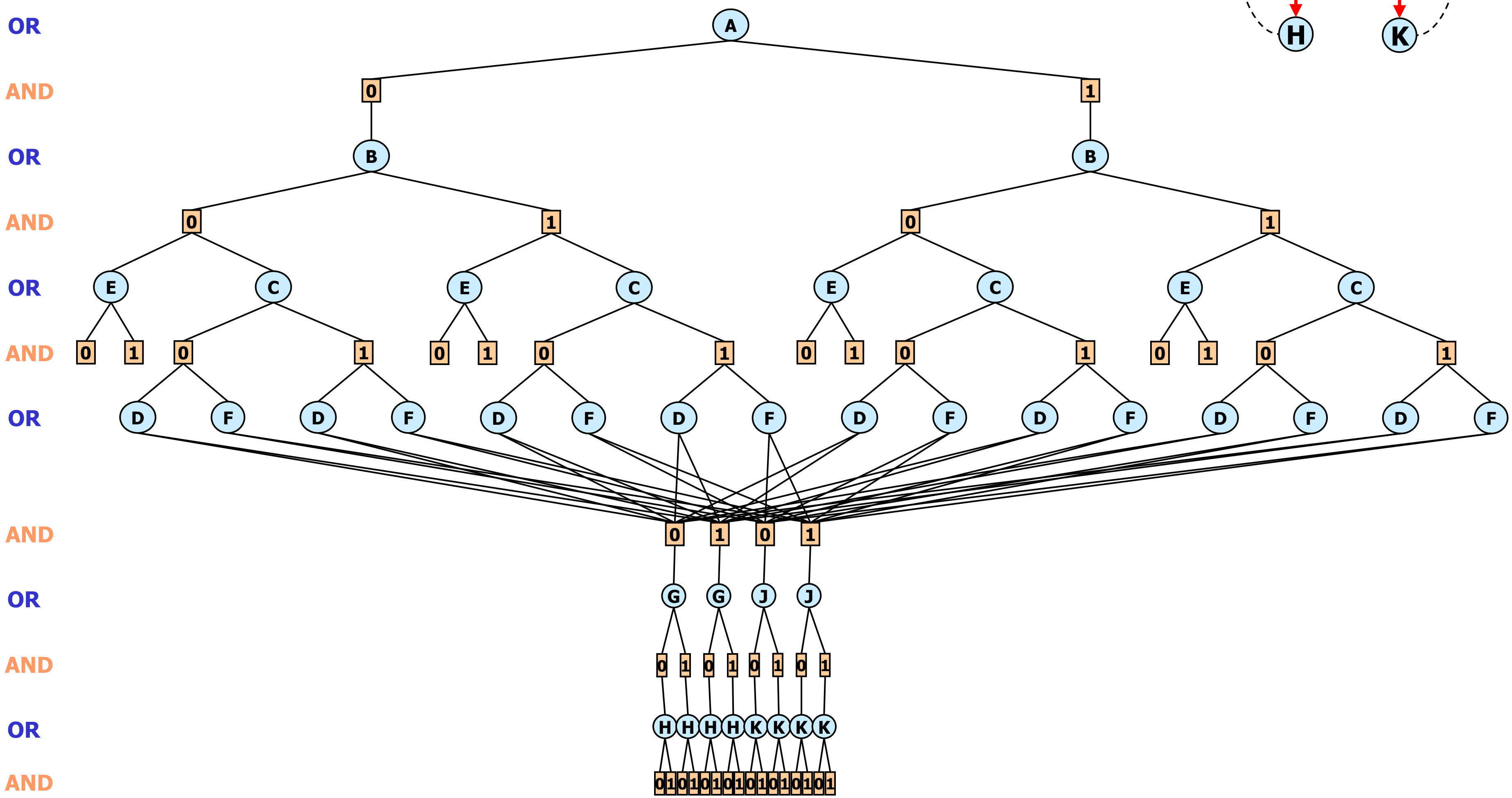




An AND/OR Graph

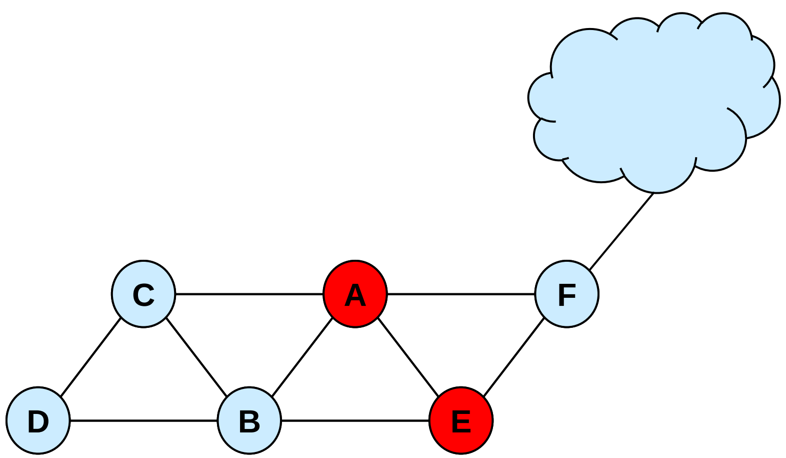


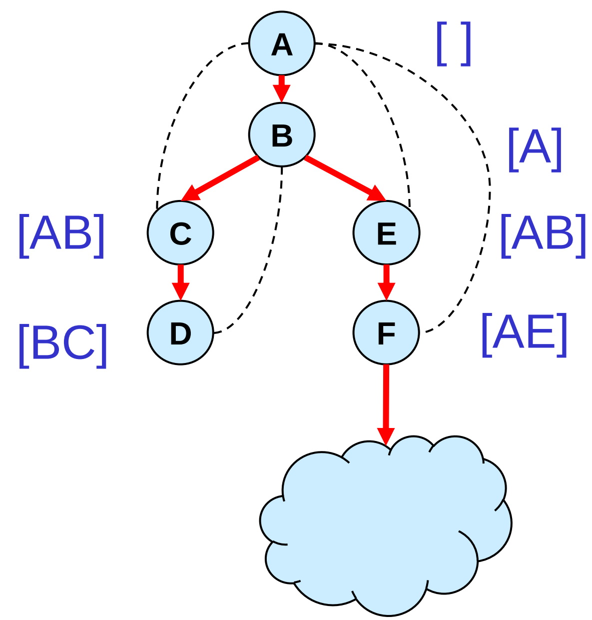




Merging based on context

context (X) = ancestors of X connected to
- X
- descendants of X



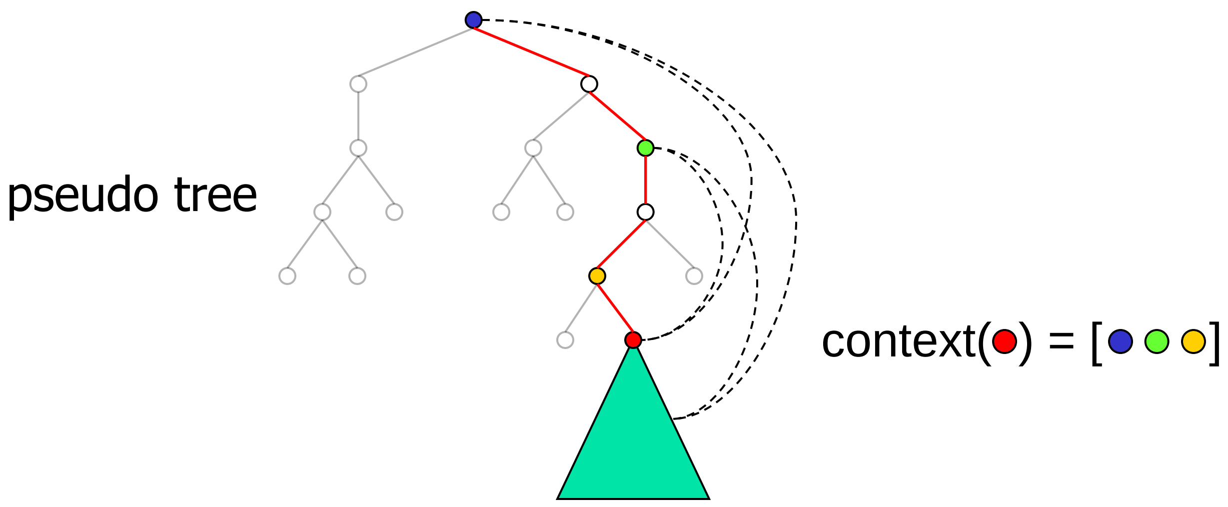


How big is the context?

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

context (X) = parents in the induced graph

max |context| = induced width = treewidth



AND/OR Tree DFS Algorithm

(Belief Updating)

$P(E \mid A,B)$

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

Evidence: E=0

$P(B \mid A)$

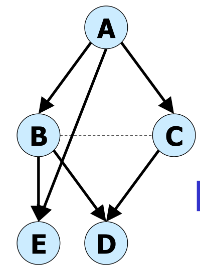
A	B=0	B=1
0	.4	.6
1	.1	.9

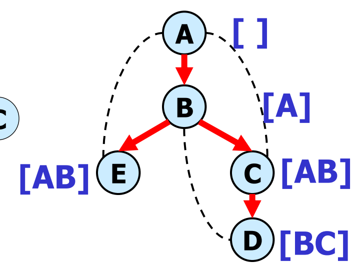
$P(C \mid A)$

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

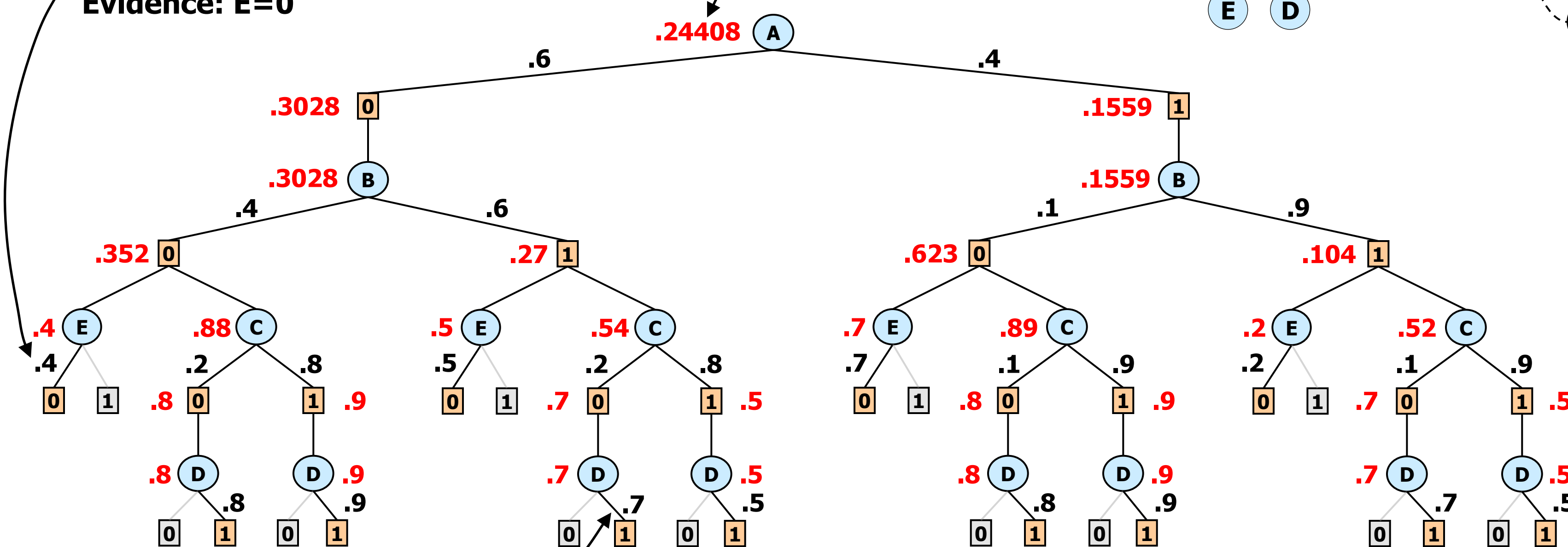
$P(A)$

A	P(A)
0	.6
1	.4





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



$P(D \mid B,C)$

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

Evidence: D=1

OR node: Marginalization operator (summation)

AND node: Combination operator (product)

Value of node = updated belief for sub-problem below

AND/OR Graph DFS Algorithm

(Belief Updating)

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

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

Evidence: E=0

$P(B\,|\,A)$

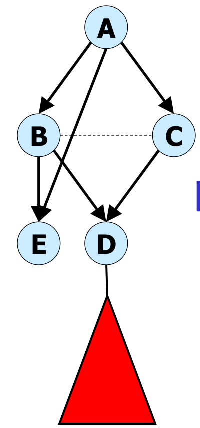
A	B=0	B=1
0	.4	.6
1	.1	.9

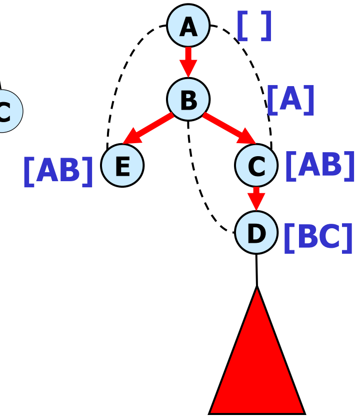
$P(C\,|\,A)$

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

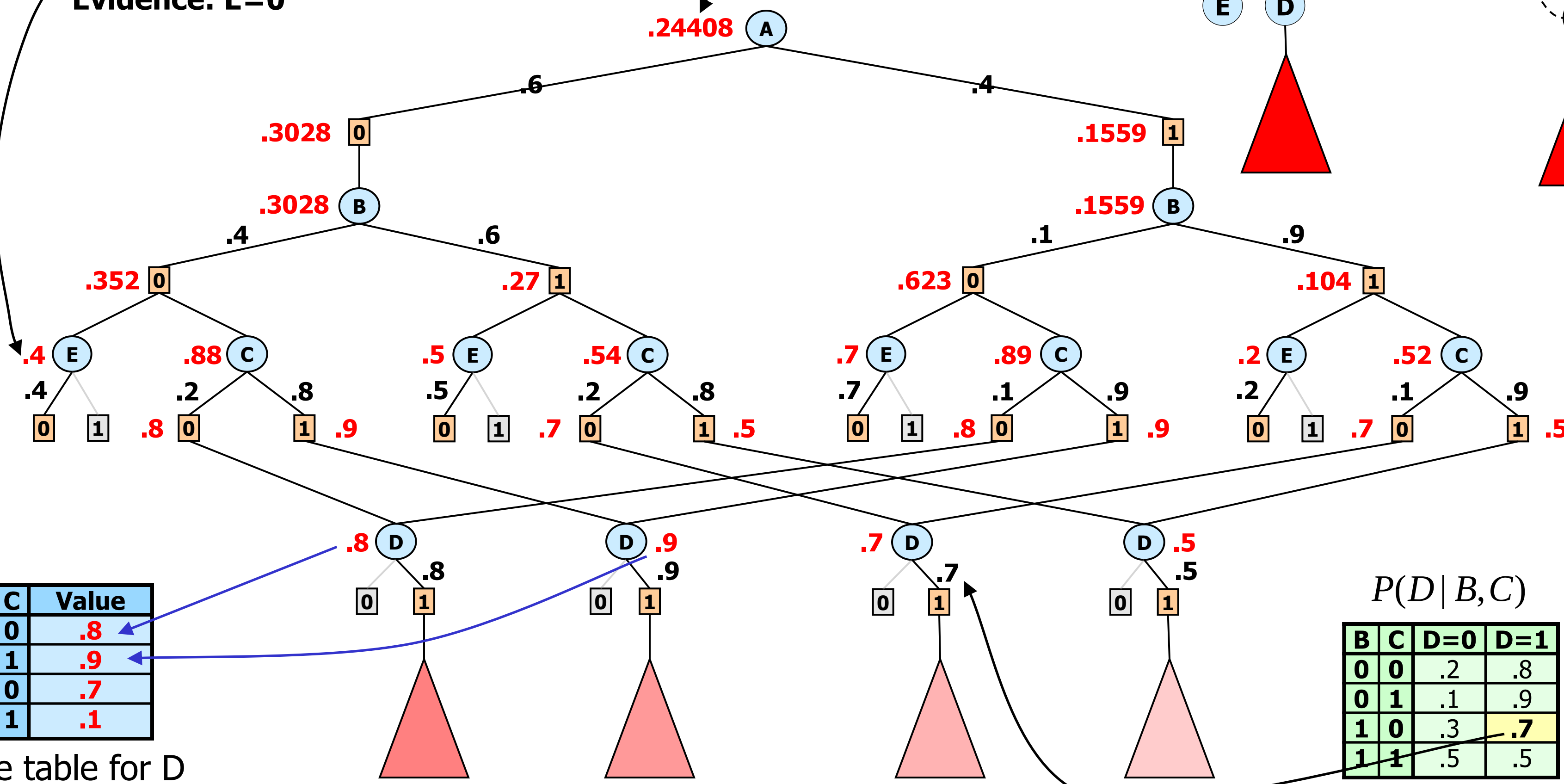
$P(A)$

A	P(A)
0	.6
1	.4





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



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

Cache table for D

$P(D\,|\,B,C)$

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

Evidence: D=1

Complexity of AND/OR Graph Search

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

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

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

Constructing Pseudo Trees

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

Quality of the Pseudo Trees

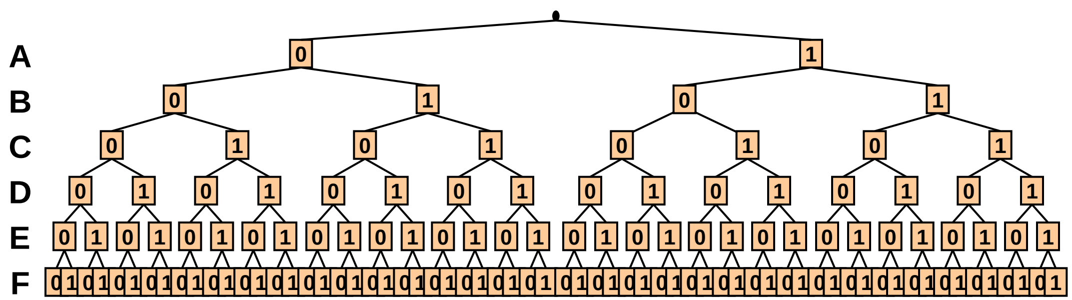
Network	hypergraph		min-fill	
	width	depth	width	depth
barley	7	**13**	7	23
diabetes	7	**16**	4	77
link	21	**40**	15	53
mildew	5	**9**	4	13
munin1	12	**17**	12	29
munin2	9	**16**	9	32
munin3	9	**15**	9	30
munin4	9	**18**	9	30
water	11	**16**	10	15
pigs	11	**20**	11	26

Bayesian Networks Repository

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

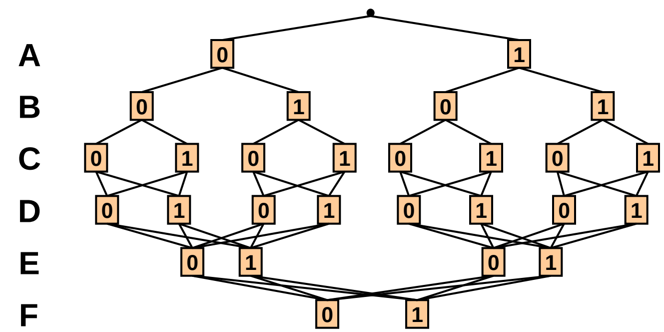
SPOT5 Benchmarks

All Four Search Spaces



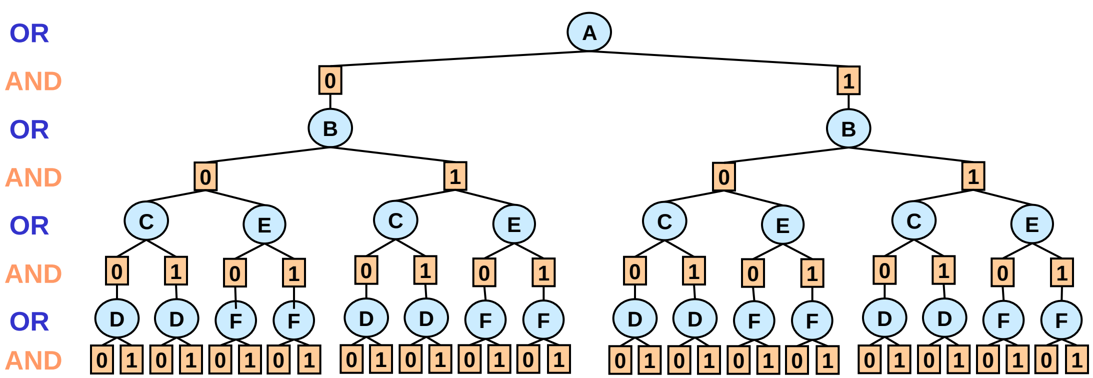
Full OR search tree

126 nodes



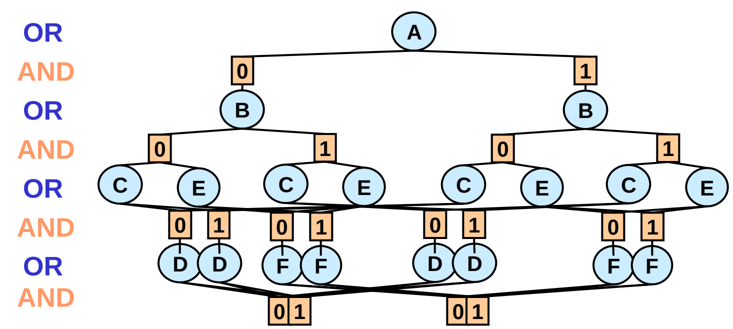
Context minimal OR search graph

28 nodes



Full AND/OR search tree

54 AND nodes

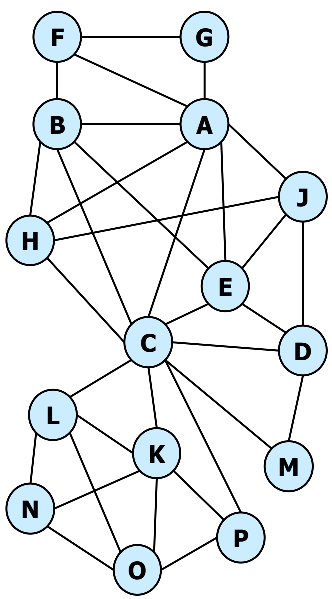


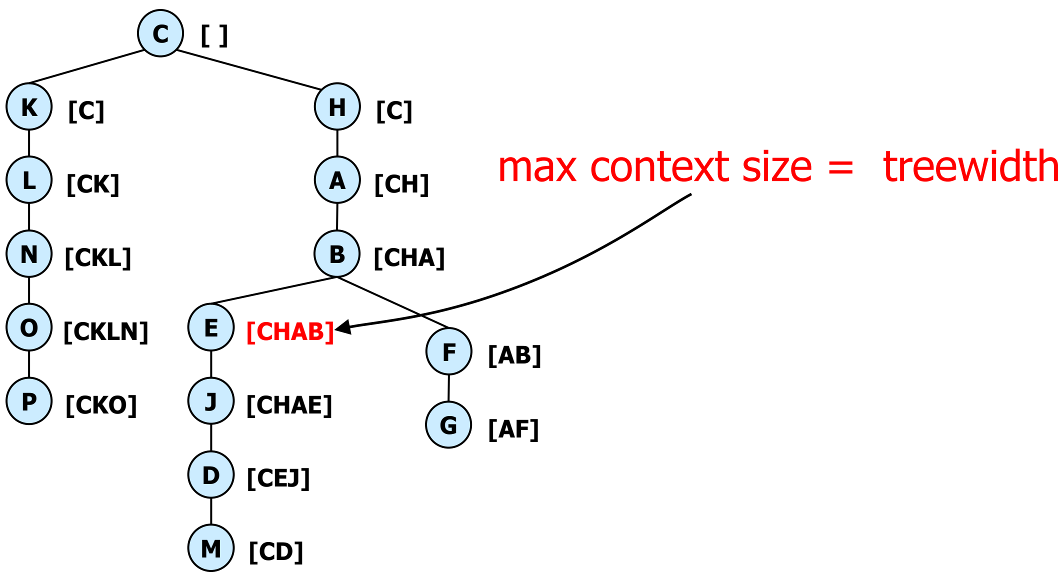
Context minimal AND/OR search graph

18 AND nodes

How Big Is The Context?

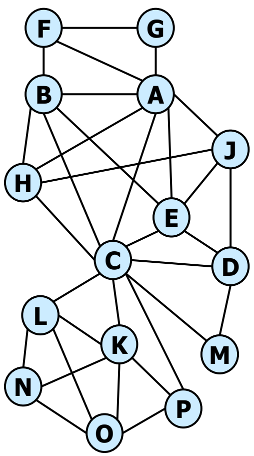
Theorem: The maximum context size for a pseudo tree is equal to the treewidth of the graph along the pseudo tree.

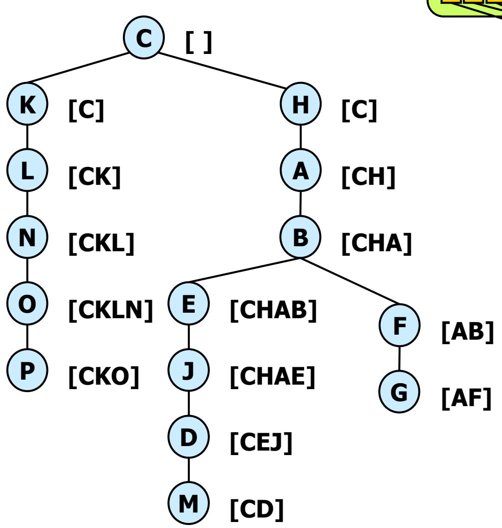


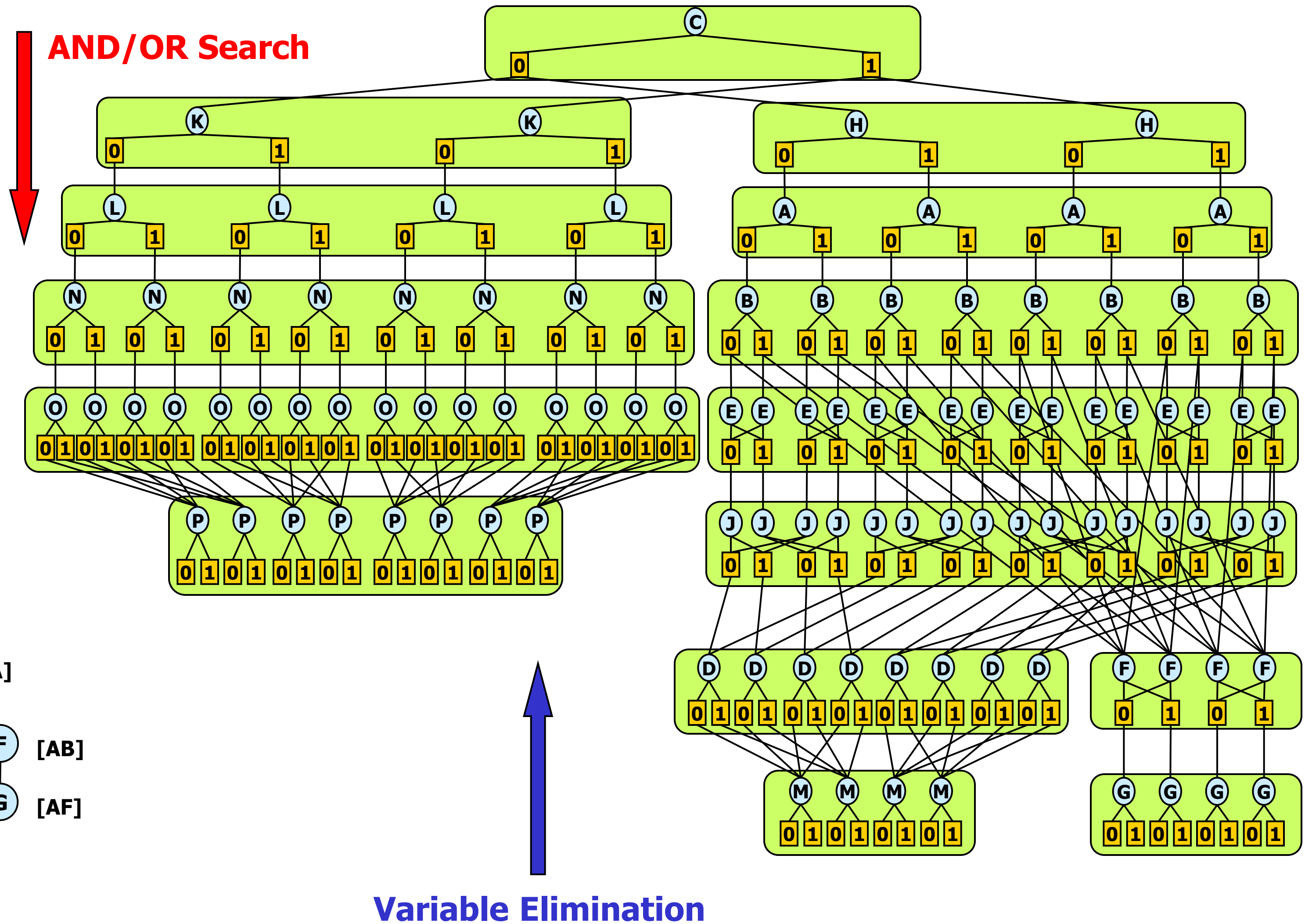


(C K H A B E J L N O D P M F G)

AND/OR Context Minimal Graph



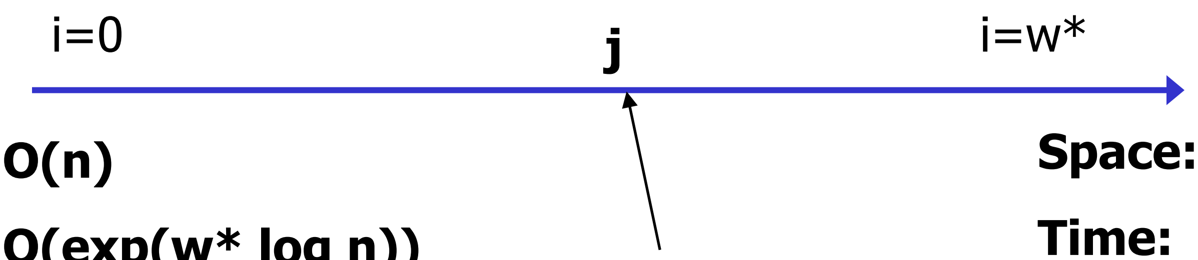




(C K H A B E J L N O D P M F G)

Searching AND/OR Graphs

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



Space: O(n)

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

Space: O(exp(j))

Time: O(exp(m_j+j)

Space: O(exp w*)

Time: O(exp w*)

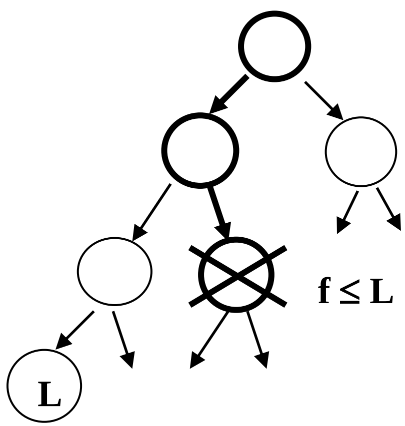
Search for MPE/MAP problem

- Searching the AND?OR space by
  - Branch and bound
  - Best-first

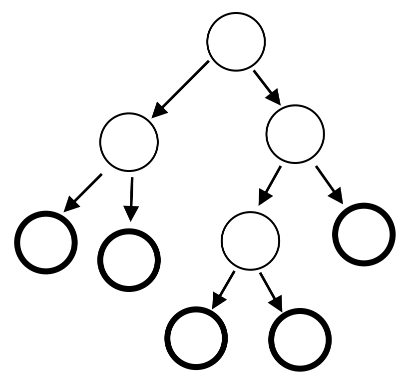
Searching the AND/OR space for MPE/MAP

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

1. DF Branch-and-Bound
Use heuristic function **$f(x^p)$** to prune the depth-first search tree
Linear space



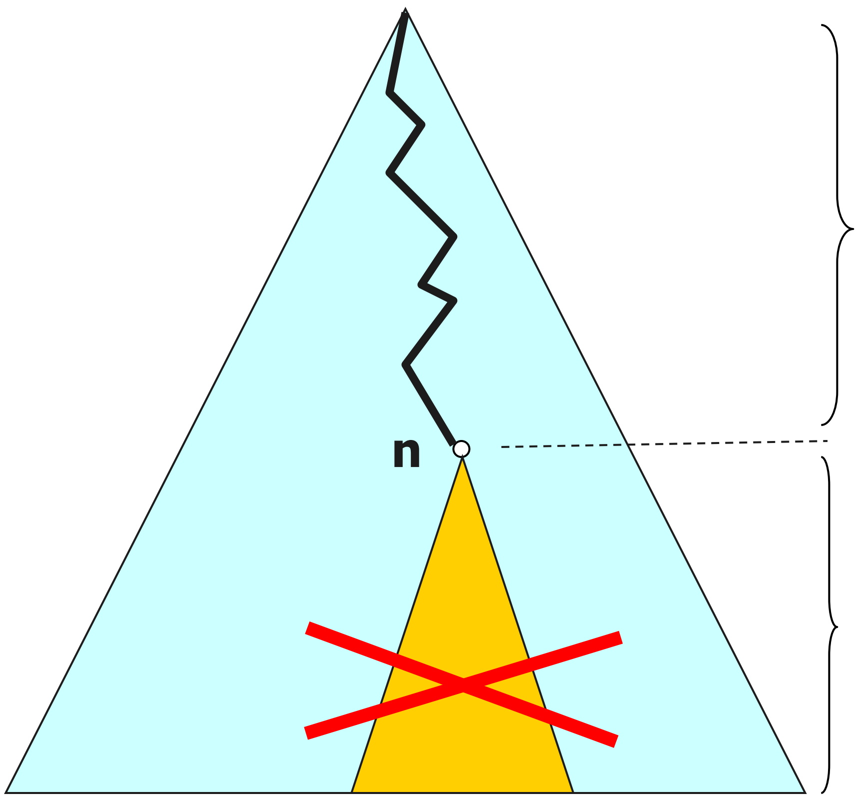
2. Best-First Search
Always expand the node with the highest heuristic value **$f(x^p)$**
Needs lots of memory



AND/OR Branch-and-Bound (AOBB)

(Marinescu & Dechter, IJCAI'05)

Maintain
ub = best solution found so far



g(n)

lb(n) = g(n) + h(n)

h(n)

estimates the optimal cost below n

OR Branch-and-Bound

Prune subtree below n if lb(n) ≥ ub

Mini-Bucket Approximation

(Dechter & Rish, 1997)

Split a bucket into mini-buckets => bound complexity

bucket (X) =

{ h_1, ..., h_r, h_{r+1}, ..., h_n }

$$h^X = \min_X \sum_{i=1}^{n} h_i$$

{ h_1, ..., h_r } **{ h_{r+1}, ..., h_n }**

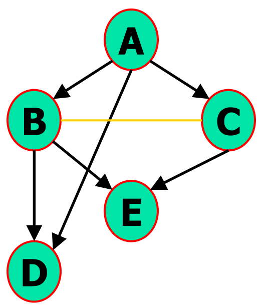
$$g^X = \left(\min_X \sum_{i=1}^{r} h_i\right) + \left(\min_X \sum_{i=r+1}^{n} h_i\right)$$

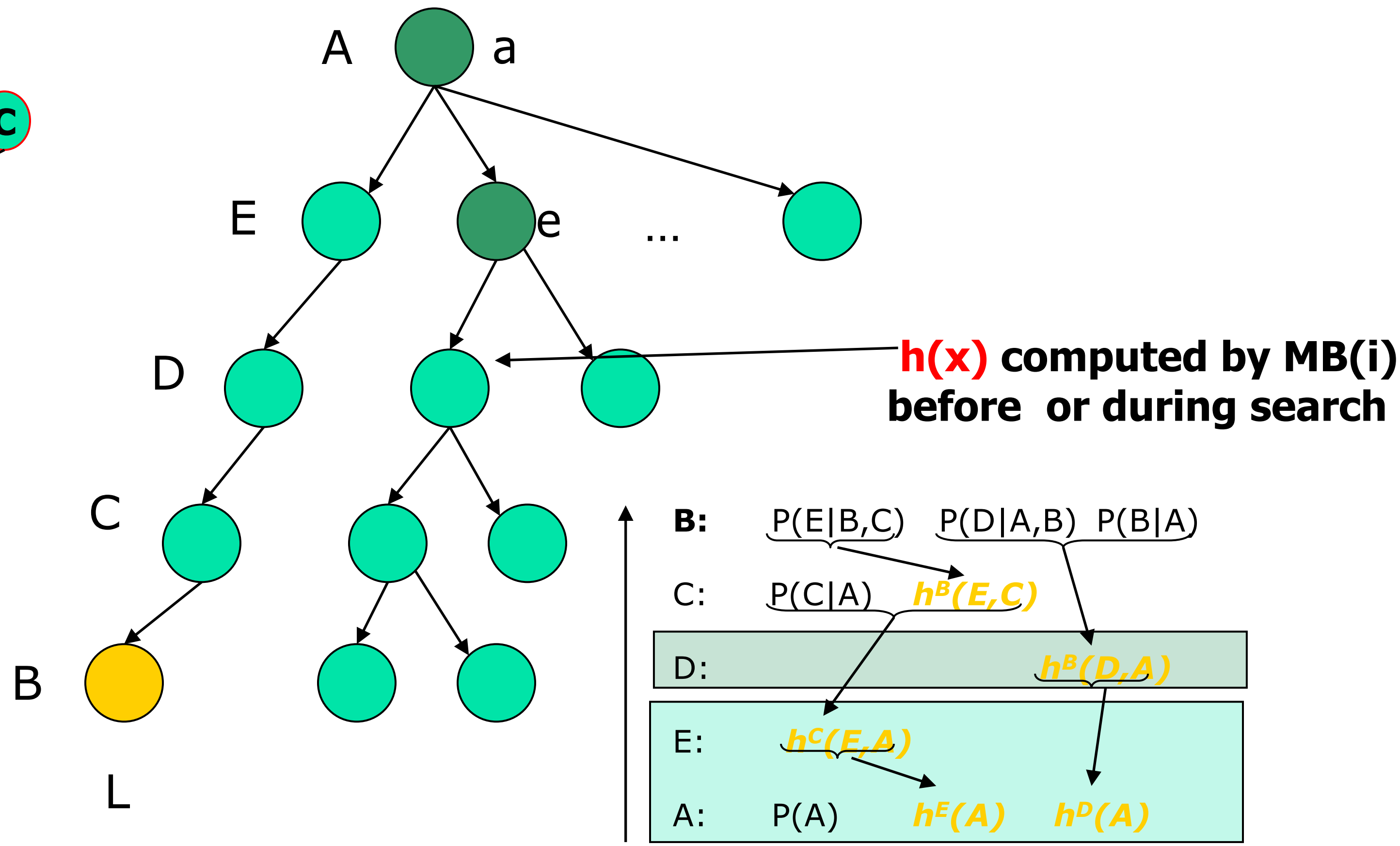
$$g^X \leq h^X$$

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

Mini-bucket Heuristics for BB search

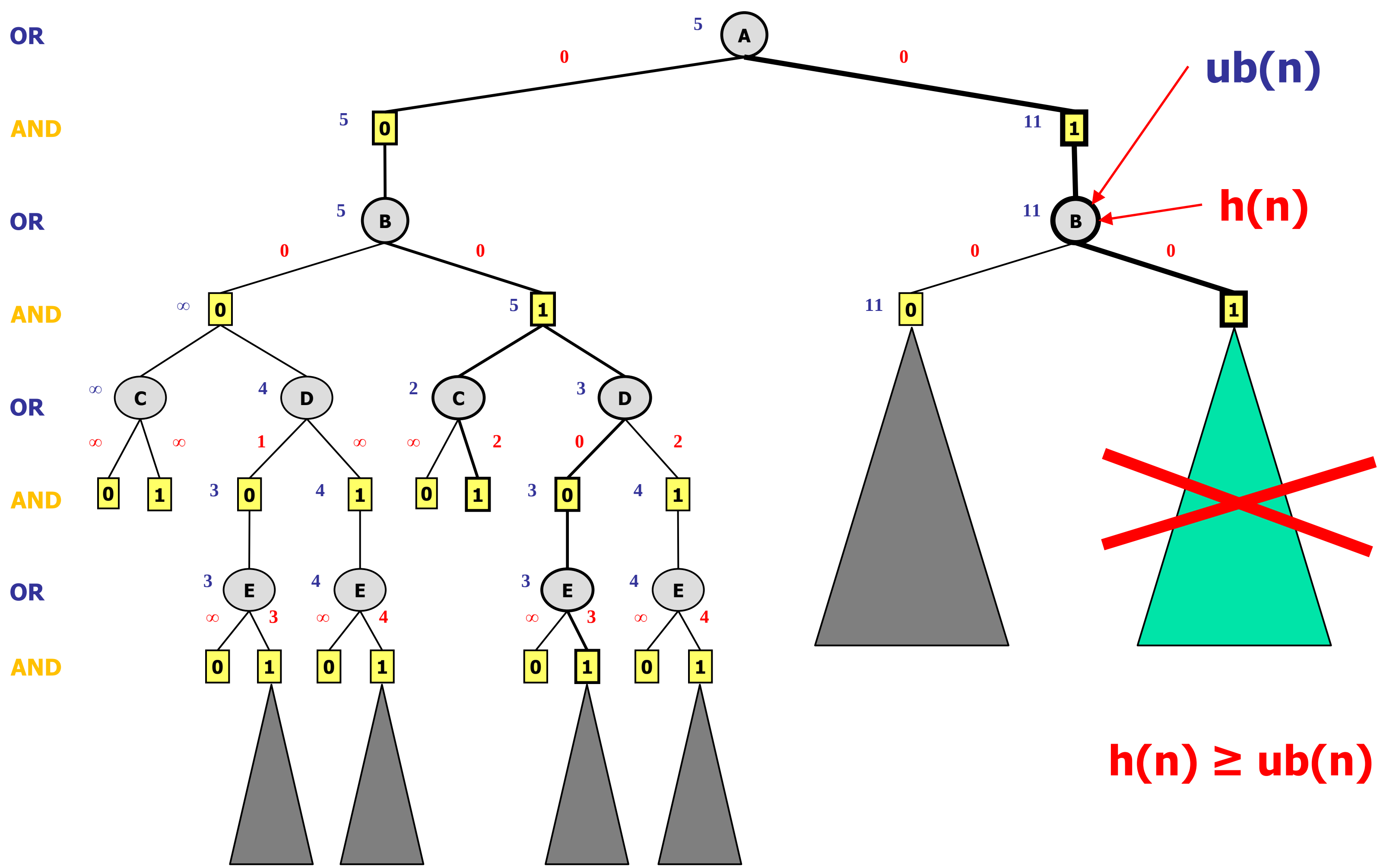
(Kask and dechterAIJ, 2001, Kask, Dechter and Marinescu UAI 2003)





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

AND/OR Branch-and-Bound (contd.)



AND/OR Branch and Bound for Constraint Optimization

(Marinescu and Dechter, IJCAI 2005, UAI 2005, AAAI 2006, ECAI 2006)

- Search AND/OR Context-minimal graph
  - exploit decomposition and equivalence
- Prune irrelevance via mini-bucket heuristics, and constraint propagation
- Depth-first (AOBB) and best-first (AOBF)
- Dynamic variable orderings
- Applied to MPE and weighted CSPs
- Applied to Integer Programming

Experiments

- **Benchmarks**
  - Belief Networks (BN)
  - Weighted CSPs (WCSP)
- **Algorithms**
  - AOBB-C – AND/OR Branch-and-Bound w/ caching
  - AOBF-C – Best-first AND/OR Search
  - Samlam
  - Superlink
  - Toolbar (DFBB+EDAC), Toolbar-BTD (BTD+EDAC)
- **Heuristics**
  - Mini-Bucket heuristics

Genetic Linkage Analysis

pedigree (w*, h) (n, d)	**SamIam** **Superlink**	**BB-C+SMB(i)** **AOBB+SMB(i)** **AOBB-C+SMB(i)** **AOBF-C+SMB(i)** i=12 time	nodes	**BB-C+SMB(i)** **AOBB+SMB(i)** **AOBB-C+SMB(i)** **AOBF-C+SMB(i)** i=14 time	nodes	**BB-C+SMB(i)** **AOBB+SMB(i)** **AOBB-C+SMB(i)** **AOBF-C+SMB(i)** i=16 time	nodes	**BB-C+SMB(i)** **AOBB+SMB(i)** **AOBB-C+SMB(i)** **AOBF-C+SMB(i)** i=18 time	nodes
ped30 (23, 118) (1016, 5)	out 13095.83	- - 10212.70 out	- - 93,233,570	- - 8858.22 out	- - 82,552,957	- - - out	- - -	- 214.10 34.19 **30.39**	- 1,379,131 193,436 72,798
ped33 (37, 165) (581, 5)	out -	- 2804.61 1426.99 out	- 34,229,495 11,349,475	- 737.96 307.39 140.61	- 9,114,411 2,504,020 407,387	- 3896.98 1823.43 out	- 50,072,988 14,925,943	- 159.50 86.17 **74.86**	- 1,647,488 453,987 134,068
ped42 (25, 76) (448, 5)	out 561.31	- - - out	- - -	- - - out	- - -	- - 2364.67 **133.19**	- - 22,595,247 93,831	out	

Min-fill pseudo tree. Time limit 3 hours.

Algorithms for AND/OR Space are currently superior

- **Back-jumping** for CSPs
  (Gaschnig 1977), (Dechter 1990), (Prosser, Bayardo and Mirankar, 1995)

- **Pseudo-search re-arrangement**, for any CSP task
  (Freuder and Quinn 1985)

- **Pseudo-tree search for soft constraints**
  (Larrosa, Meseguer and Sanchez, 2002)

- **Recursive Conditioning**
  (Darwiche, 2001), explores the AND/OR tree or graph for any query

- **BTD: Searching tree-decompositions** for optimization
  (Jeagou and Terrioux, 2004)

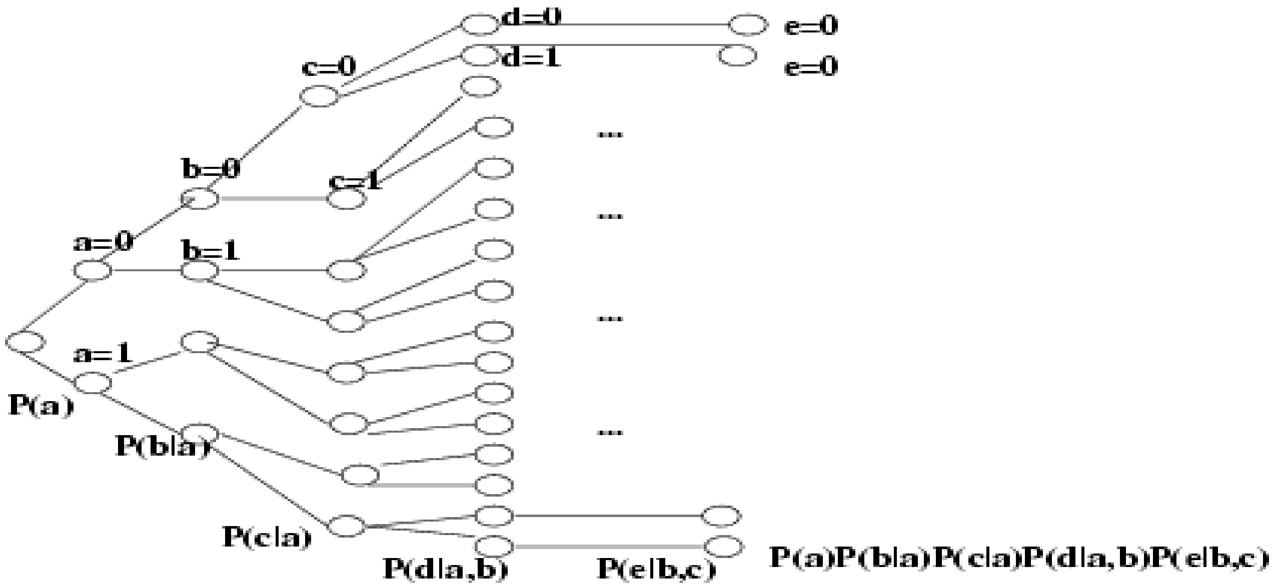
- **Value Elimination**
  (Bacchus, Dalmao and Pittasi, 2003)

Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
- Search
- Hybrid of search and inference
- Sampling
- Modeling and learning
- Software

Conditioning generates the probability tree

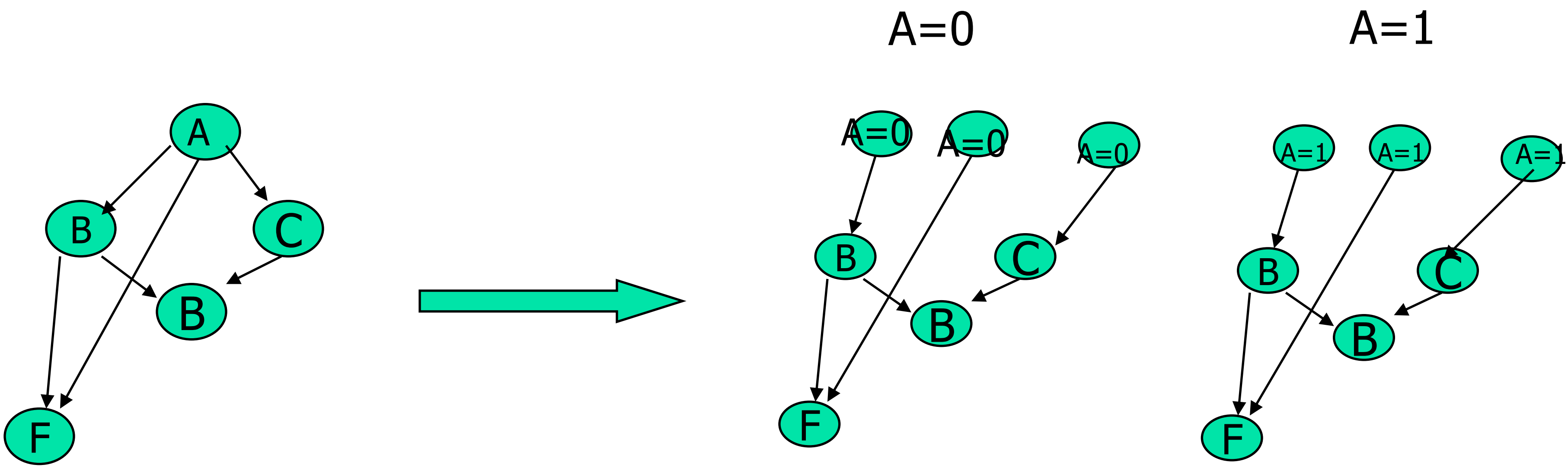
$$P(a, e=0) = P(a)\sum_b P(b \mid a)\sum_c P(c \mid a)\sum_b P(d \mid a,b)\sum_{e=0} P(e \mid b,c)$$



Complexity of conditioning: exponential time, linear space

Loop-cutset decomposition

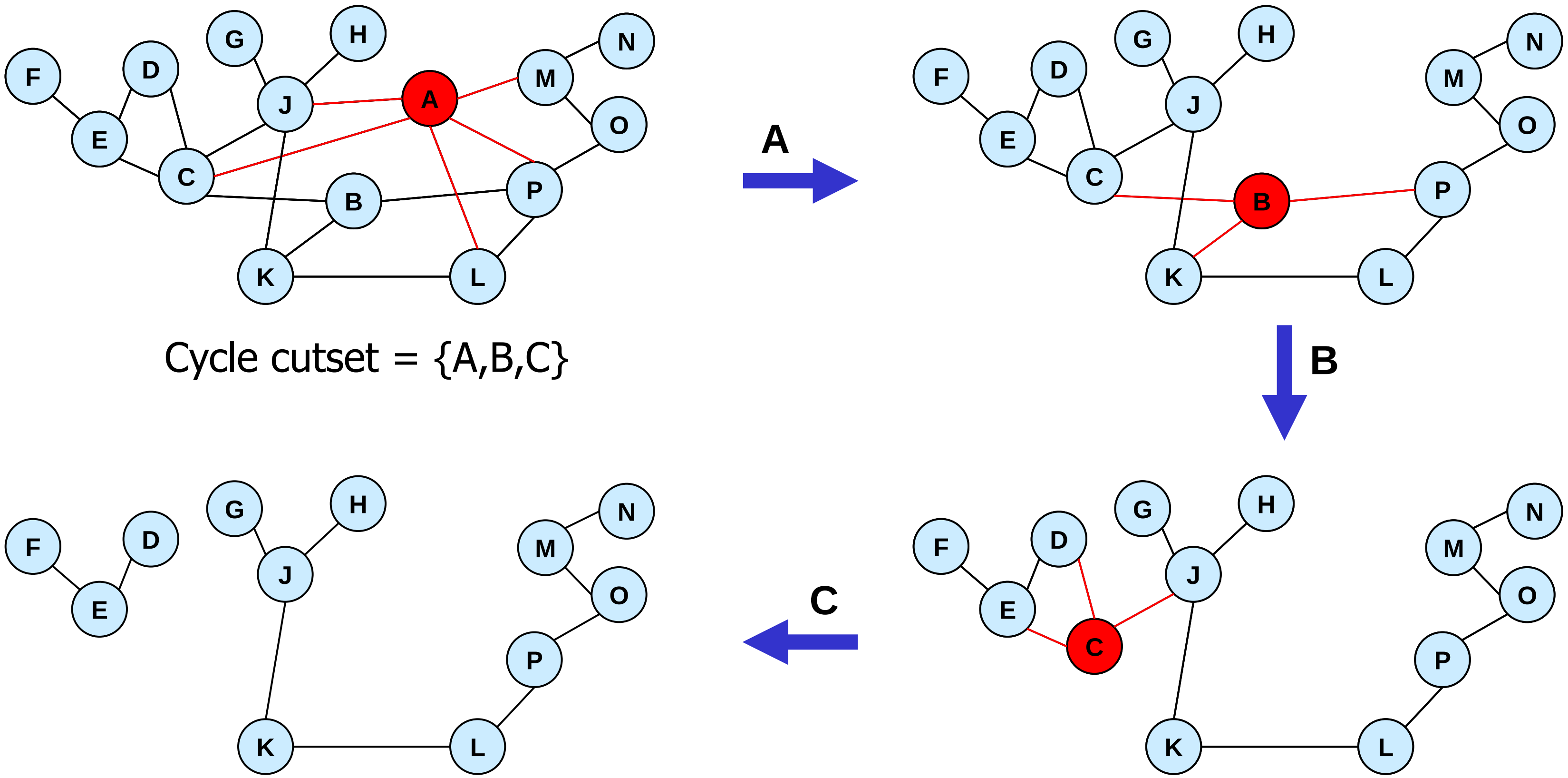
- You condition until you get a polytree



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

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

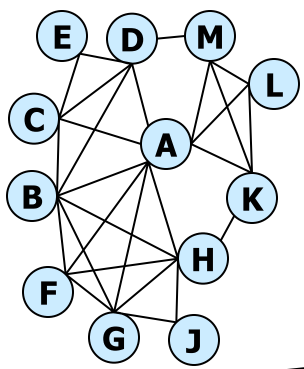
Conditioning and Cycle cutset



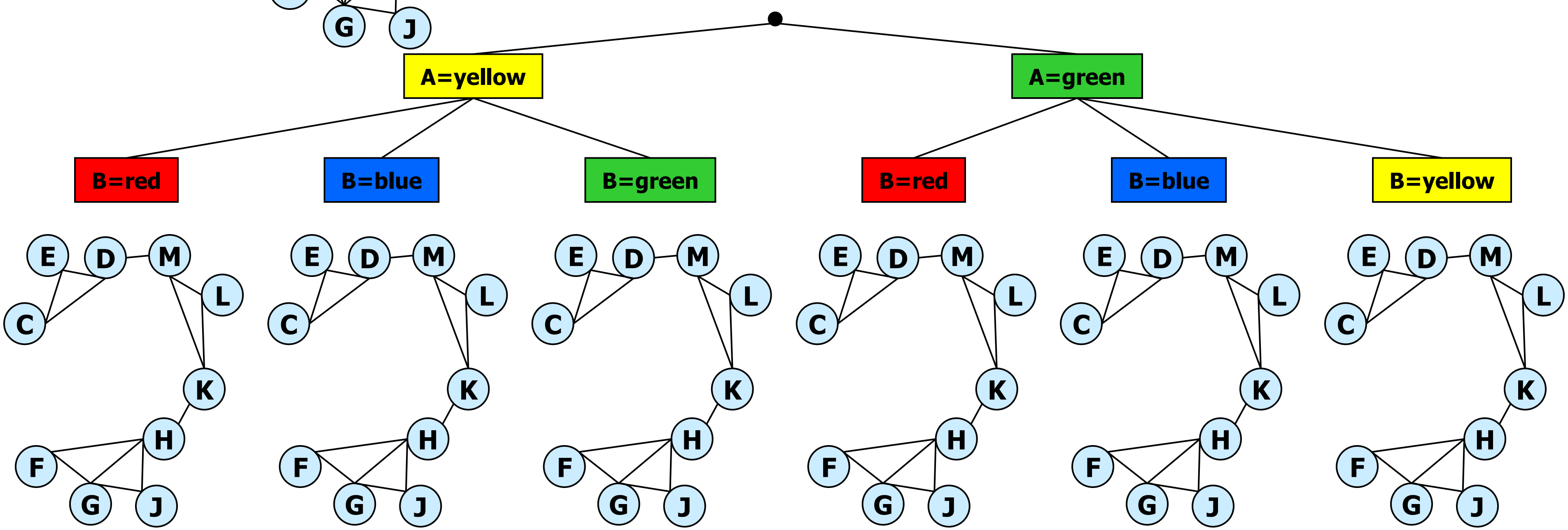
Cycle cutset = {A,B,C}

Search over the Cutset (cont)

Graph Coloring problem



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



Variable elimination with conditioning; w-cutset algorithms

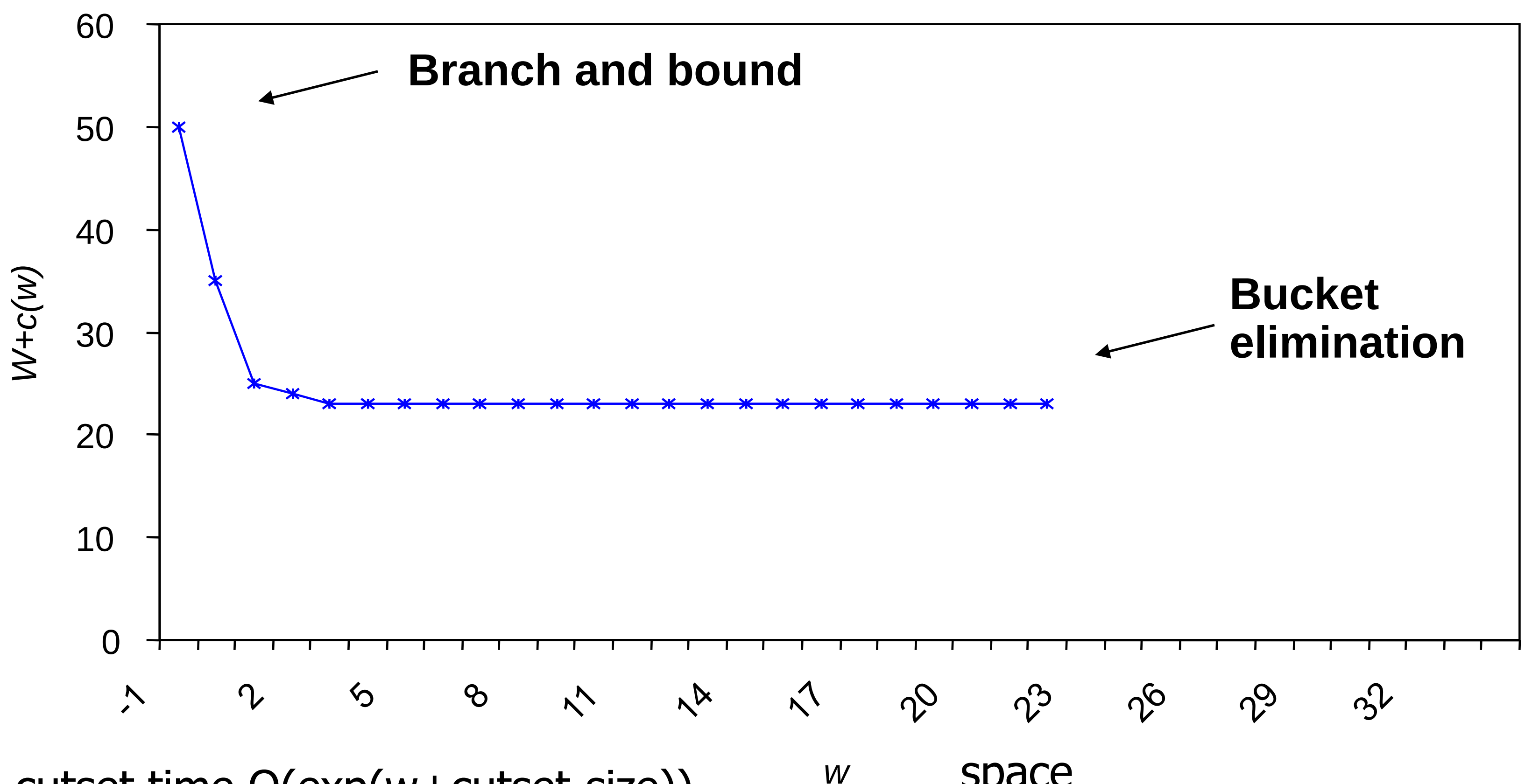
- Identify an w-cutset c_w of the network
- For each assignment to the cutset c_w solve the conditioned sub-problem by *CTE*
- Aggregate the solutions over all c_w assignments.
- Time complexity: $O(k^{c_w+w})$
- Space complexity: $O(k^{\omega})$
- What w should we use?
  - W=1? W=0? W=w*
  - Depends on the graph
  - Practice: use the largest w allowed by space
- Alternate conditioning and elimination?

Time vs Space for w-cutset

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

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

time



space

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

Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
- Search
- Sampling
- Hybrid of search and inference
- Modeling and learning
- Software

Sampling: Approximation of Search

1. Importance Sampling
2. Markov Chain Monte Carlo: Gibbs Sampling
3. Sampling in presence of Determinism
4. Rao-Blackwellisation
5. AND/OR importance sampling

See :Sampling Techniques for Probabilistic and Deterministic Graphical models PDF Tutorial, AAAI 2010, Atlanta, GA, July 12, 2010:

http://www.ics.uci.edu/~dechter/talks.html

Sampling for Probabity Inference

- **Logic Sampling**
- Importance Sampling
  - Likelihood Sampling
  - Choosing a Proposal Distribution
- Markov Chain Monte Carlo (MCMC)
  - Metropolis-Hastings
  - Gibbs sampling
- Variance Reduction

Logic Sampling: No Evidence (Henrion 1988)

Input: Bayesian network

$X = \{X_1,\ldots,X_N\}$, N- #nodes, T - # samples

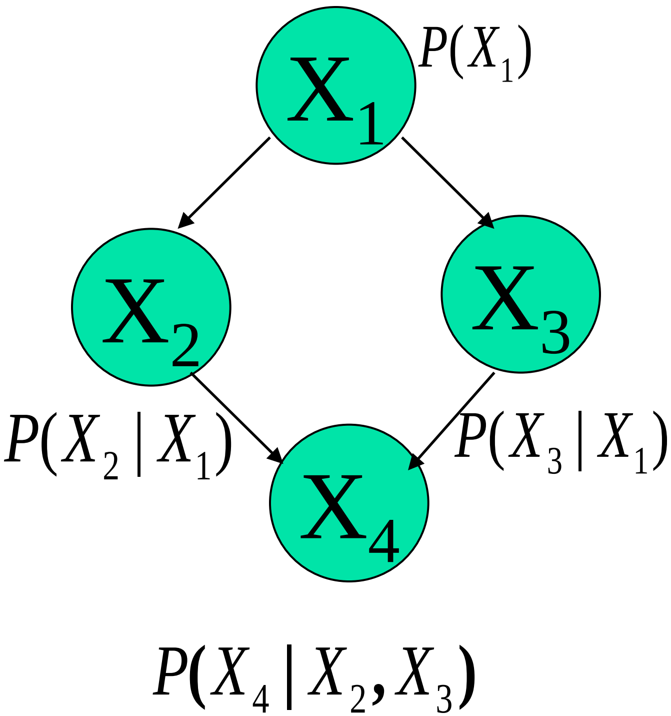
Output: T samples

Process nodes in topological order – first process the ancestors of a node, then the node itself:

1. For t = 0 to T
2. For i = 0 to N
3. $X_i \leftarrow$ sample x_i^t from $P(x_i \mid pa_i)$

Logic sampling (example)

$$P(X_1, X_2, X_3, X_4) = P(X_1) \times P(X_2 \mid X_1) \times P(X_3 \mid X_1) \times P(X_4 \mid X_2, X_3)$$



No Evidence

// generate sample k

1. Sample x_1 from $P(x_1)$
2. Sample x_2 from $P(x_2 \mid X_1 = x_1)$
3. Sample x_3 from $P(x_3 \mid X_1 = x_1)$
4. Sample x_4 from $P(x_4 \mid X_2 = x_2, X_3 = x_3)$

Logic Sampling w/ Evidence

Input: Bayesian network

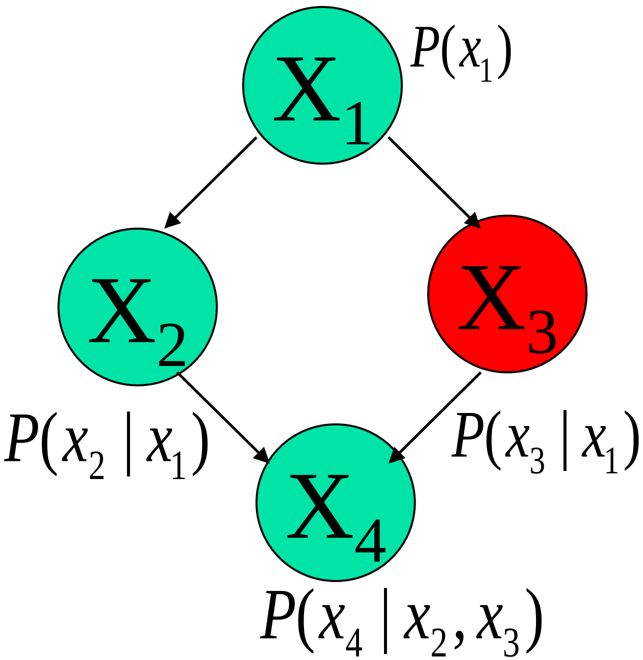
$X = \{X_1,\ldots,X_N\}$, N- #nodes

E – evidence, T - # samples

Output: T samples consistent with E

1. For t=1 to T
2. For i=1 to N
3. $X_i \leftarrow$ sample x_i^t from $P(x_i \mid pa_i)$
4. If X_i in E and $X_i \neq x_i$, reject sample:
5. Goto Step 1.

Logic Sampling (example)



Evidence: $X_3 = 0$

// generate sample k

1. Sample x_1 from $P(x_1)$
2. Sample x_2 from $P(x_2 | x_1)$
3. Sample x_3 from $P(x_3 | x_1)$
4. If $x_3 \neq 0$, reject sample and start from 1, otherwise
5. Sample x_4 from $P(x_4 | x_{2,} x_3)$

Monte Carlo Estimate

- **Estimator:**
  - An estimator is a function of the samples.
  - It produces an estimate of the unknown parameter of the sampling *distribution.*

Given i.i.d. samples $S^1, S^2, \ldots S^T$ drawn from P,

the Monte carlo estimate of $E_P[g(x)]$ is given by :

$$\hat{g} = \frac{1}{T} \sum_{t=1}^{T} g(S^t)$$

Example: Monte Carlo estimate

- Given:
  - A distribution P(X) = (0.3, 0.7).
  - g(X) = 40 if X equals 0

    = 50 if X equals 1.
- Estimate $E_P[g(x)]=(40x0.3+50x0.7)=47$.
- Generate k samples from P: 0,1,1,1,0,1,1,0,1,0

$$\hat{g} = \frac{40 \times \#samples(X=0) + 50 \times \#samples(X=1)}{\#samples}$$

$$= \frac{40 \times 4 + 50 \times 6}{10} = 46$$

Importance sampling: Main idea

- Express query as the expected value of a random variable w.r.t. to a distribution Q.
- Generate random samples from Q.
- Estimate the expected value from the generated samples using a monte carlo estimator (average).

Importance sampling for P(e)

$Let\ Z = X \setminus E,$

Let Q(Z) be a (proposal) distribution, satisfying

$P(z,e) > 0 \Rightarrow Q(z) > 0$

Then, we can rewrite P(e) as:

$$P(e) = \sum_z P(z,e) = \sum_z P(z,e)\frac{Q(z)}{Q(z)} = E_Q\left[\frac{P(z,e)}{Q(z)}\right] = E_Q[w(z)]$$

Monte Carlo estimate:

$$\hat{P}(e) = \frac{1}{T}\sum_{t=1}^{T} w(z^t), \text{ where } z^t \leftarrow Q(Z)$$

Likelihood Weighting

(Fung and Chang, 1990; Shachter and Peot, 1990)

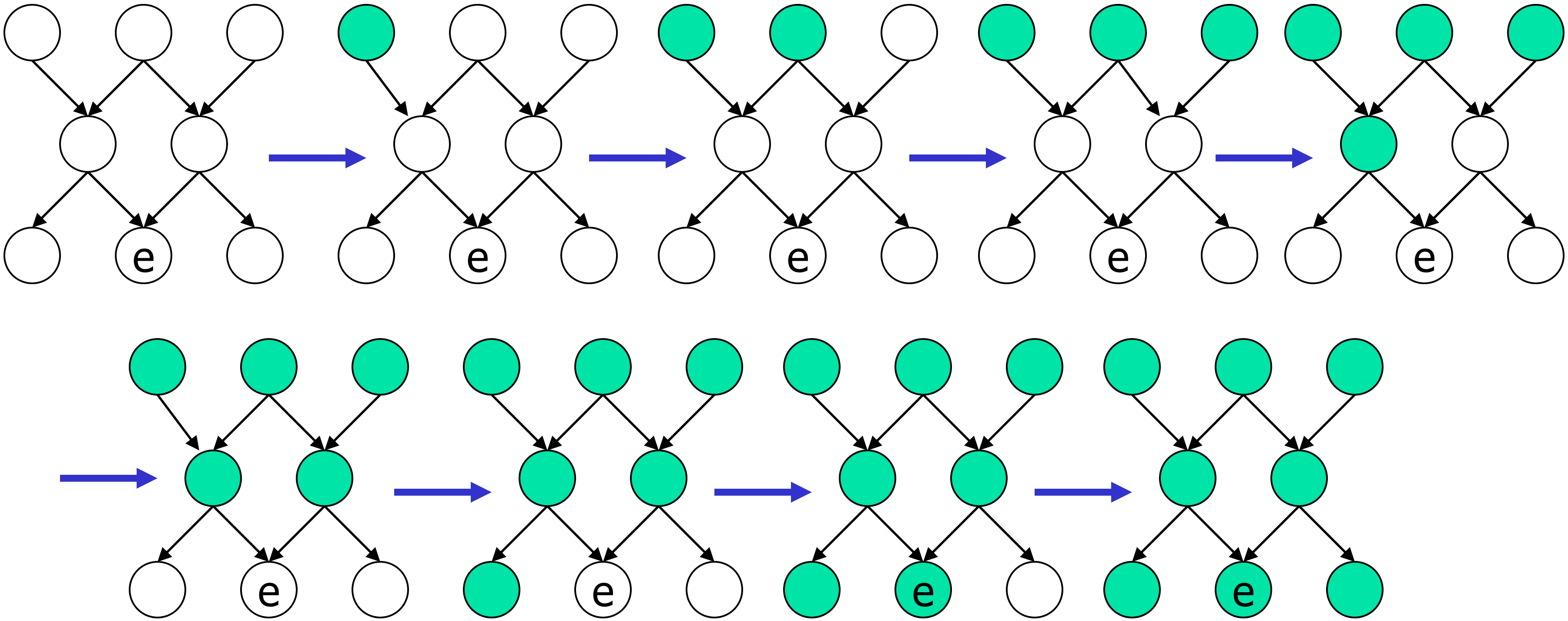
Is an instance of importance sampling!

"Clamping" evidence+
logic sampling+
weighing samples by evidence likelihood

Works well for *likely evidence!*

Likelihood Weighting: Sampling

Sample in topological order over **X** !



Clamp evidence, Sample $x_i \leftarrow P(X_i|pa_i)$,

$P(X_i|pa_i)$ is *a look-up* in CPT!

Likelihood Weighting: Proposal Distribution

$$Q(X \setminus E) = \prod_{X_i \in X \setminus E} P(X_i \mid pa_i, e)$$

Notice: Q is another Bayesian network

Example:

Given a Bayesian network: $P(X_1, X_2, X_3) = P(X_1) \times P(X_2 \mid X_1) \times P(X_3 \mid X_1, X_2)$ and

Evidence $X_2 = x_2$.

$Q(X_1, X_3) = P(X_1) \times P(X_3 \mid X_1, X_2 = x_2)$

Weights:

Given a sample: $x = (x_1, .., x_n)$

$$w = \frac{P(x, e)}{Q(x)} = \frac{\prod_{X_i \in X \setminus E} P(x_i \mid pa_i, e) \times \prod_{E_j \in E} P(e_j \mid pa_j)}{\prod_{X_i \in X \setminus E} P(x_i \mid pa_i, e)}$$

$$= \prod_{E_j \in E} P(e_j \mid pa_j)$$

Likelihood Weighting: Estimates

Estimate P(e): $\hat{P}(e) = \frac{1}{T}\sum_{t=1}^{T} w^{(t)}$

Estimate Posterior Marginals:

$$\hat{P}(x_i \mid e) = \frac{\hat{P}(x_i, e)}{\hat{P}(e)} = \frac{\sum_{t=1}^{T} w^{(t)} g_{x_i}(x^{(t)})}{\sum_{t=1}^{T} w^{(t)}}$$

$$g_{x_i}(x^{(t)}) = 1 \text{ if } \mathrm{x_i} = x_i^t \text{ and equals zero otherwise}$$

Likelihood Weighting

- Converges to exact posterior marginals
- Generates Samples Fast
- Sampling distribution is close to prior (especially if $E \subset$ Leaf Nodes)
- Increasing sampling variance

$\Rightarrow$Convergence may be slow

$\Rightarrow$Many samples with $P(x^{(t)})=0$ rejected

Avoid rejection

- Gibbs Sampling: An MCMC approach
- Likelihood weighting: An importance sampling approach
- Exploit structure
  - Cutset-sampling (likelihood and Gibbs)
  - SamplingSearch (avoid inconsistency)

Overview

1. Probabilistic Reasoning/Graphical models
2. Importance Sampling
3. Markov Chain Monte Carlo: Gibbs Sampling
4. Sampling in presence of Determinism
5. Rao-Blackwellisation
6. AND/OR importance sampling

Gibbs Sampling (Geman&Geman,1984)

- Gibbs sampler is an algorithm to generate a sequence of samples from the **joint probability distribution** of two or more random variables
- Sample new variable value one variable at a time from the variable's conditional distribution:

$$P(X_i) = P(X_i \mid x_1^t,..,x_{i-1}^t, x_{i+1}^t,...,x_n^t\} = P(X_i \mid x^t \setminus x_i)$$

- Samples from a Markov chain with stationary distribution *P(X|e)*

Ordered Gibbs Sampler

Generate sample x^{t+1} from x^t :

Process All Variables In Some Order

$$X_1 = x_1^{t+1} \leftarrow P(X_1 \mid x_2^t, x_3^t, \ldots, x_N^t, e)$$

$$X_2 = x_2^{t+1} \leftarrow P(X_2 \mid x_1^{t+1}, x_3^t, \ldots, x_N^t, e)$$

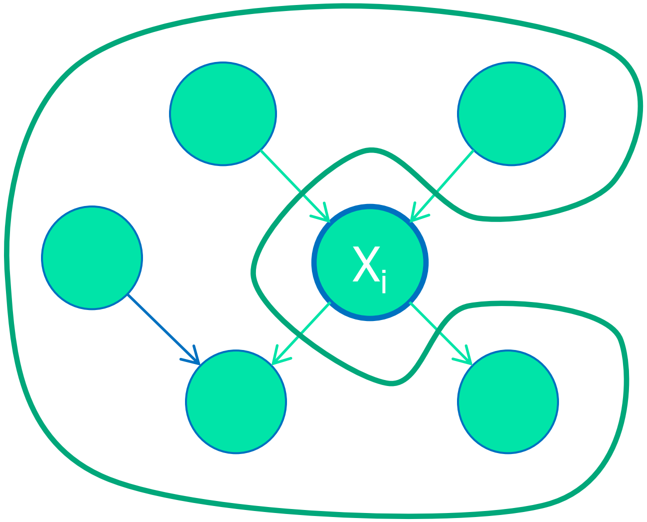
$$\ldots$$

$$X_N = x_N^{t+1} \leftarrow P(X_N \mid x_1^{t+1}, x_2^{t+1}, \ldots, x_{N-1}^{t+1}, e)$$

In short, for i=1 to N:

$$X_i = x_i^{t+1} \leftarrow \text{sampled from } P(X_i \mid x^t \setminus x_i, e)$$

Transition Probabilities in BN



Given *Markov blanket* (parents, children, and their parents), X_i is independent of all other nodes

Markov blanket:

$$markov(X_i) = pa_i \cup ch_i \cup \Big(\bigcup_{X_j \in ch_j} pa_j \Big)$$

$$P(X_i \mid x^t \setminus x_i) = P(X_i \mid markov_i^t):$$

$$P(x_i \mid x^t \setminus x_i) \propto P(x_i \mid pa_i) \prod_{X_j \in ch_i} P(x_j \mid pa_j)$$

Computation is linear in the size of Markov blanket!

Ordered Gibbs Sampling Algorithm (Pearl,1988)

Input: *X, E=e*

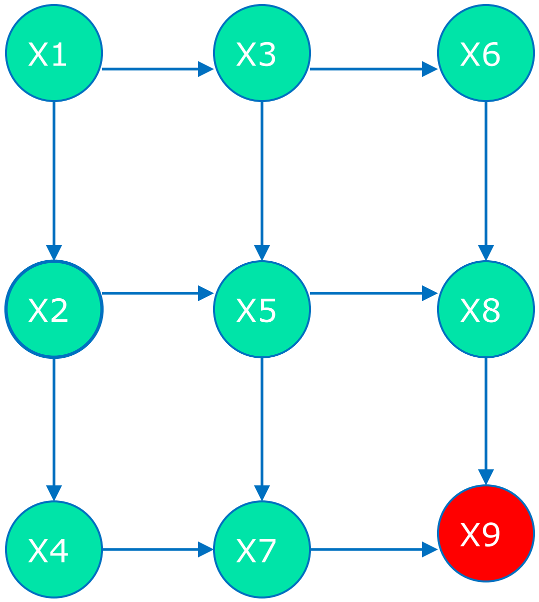
Output: *T* samples $\{x^t\}$

Fix evidence E=e, initialize x^0 at random

1. For t = 1 to T (compute samples)
2. For i = 1 to N (loop through variables)
3. $x_i^{t+1} \leftarrow P(X_i \mid markov_i^t)$
4. *End For*
5. *End For*

Gibbs Sampling Example - BN

$$X = \{X_1, X_2, \ldots, X_9\}, E = \{X_9\}$$



$$X_1 = x_1^0$$

$$X_6 = x_6^0$$

$$X_2 = x_2^0$$

$$X_7 = x_7^0$$

$$X_3 = x_3^0$$

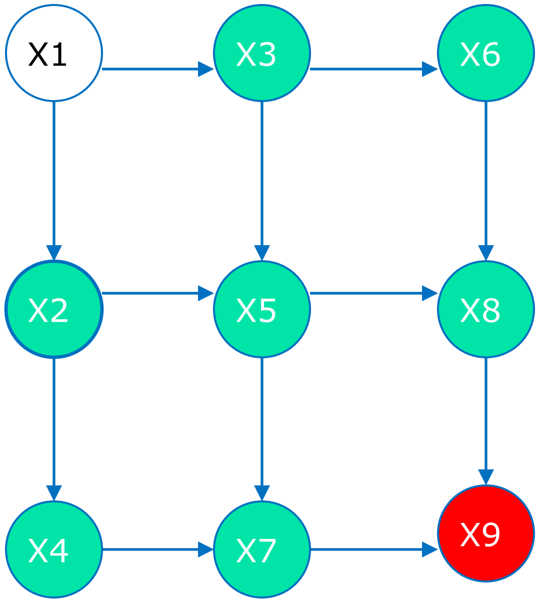
$$X_8 = x_8^0$$

$$X_4 = x_4^0$$

$$X_5 = x_5^0$$

Gibbs Sampling Example - BN

$$X = \{X_1, X_2, \ldots, X_9\}, E = \{X_9\}$$



$$x_1^1 \leftarrow P(X_1 \mid x_2^0, \ldots, x_8^0, x_9)$$

$$x_2^1 \leftarrow P(X_2 \mid x_1^1, \ldots, x_8^0, x_9)$$

$$\ldots$$

Answering Queries $P(x_i \,|\, e) = ?$

- **Method 1**: count # of samples where $X_i = x_i$ (*histogram estimator*):

Dirac delta f-n

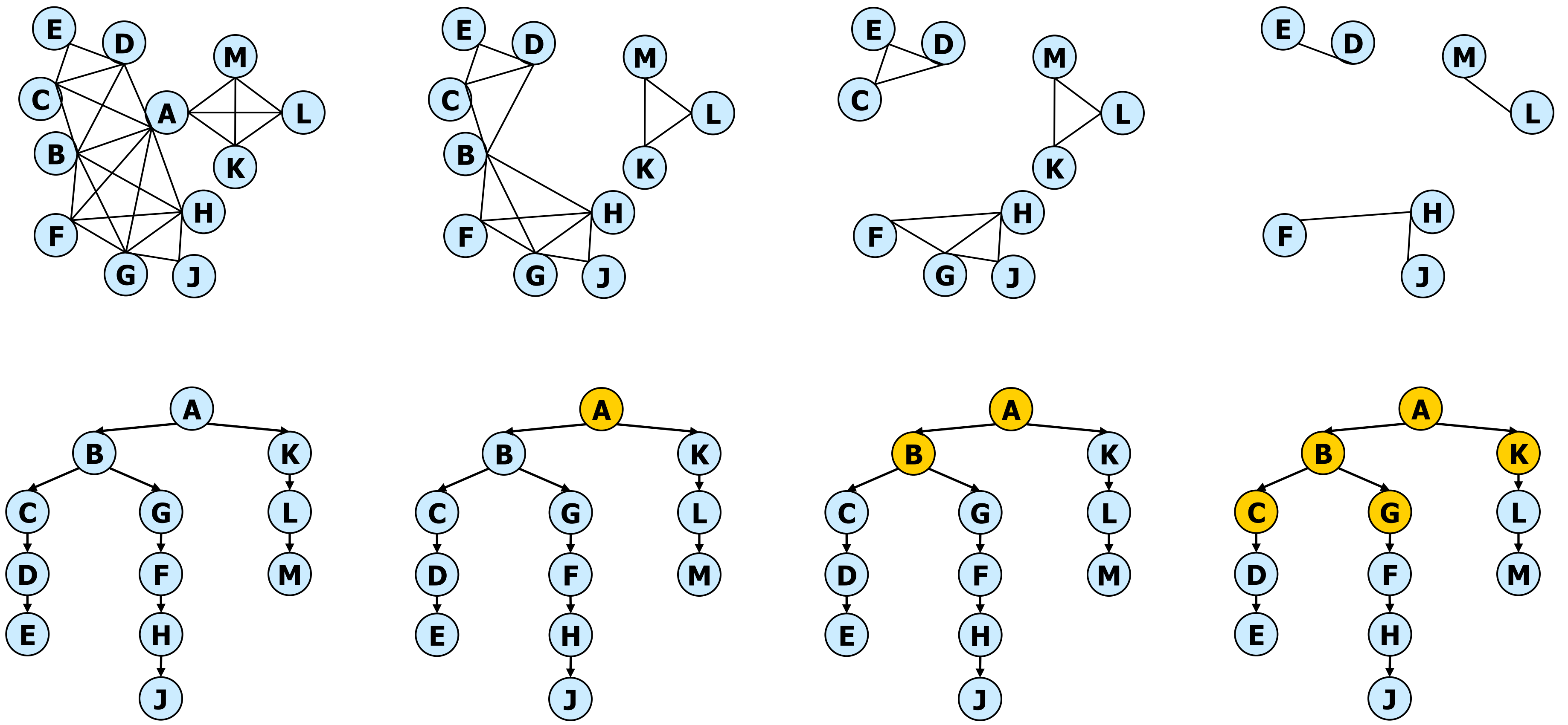
$$\overline{P}(X_i = x_i) = \frac{1}{T}\sum_{t=1}^{T}\delta(x_i, x^t)$$

- **Method 2**: average probability (*mixture estimator*):

$$\overline{P}(X_i = x_i) = \frac{1}{T}\sum_{t=1}^{T}P(X_i = x_i \,|\, markov_i^t)$$

- Mixture estimator converges faster (consider estimates for the unobserved values of X_i; prove via Rao-Blackwell theorem)

AND/OR w-cutset



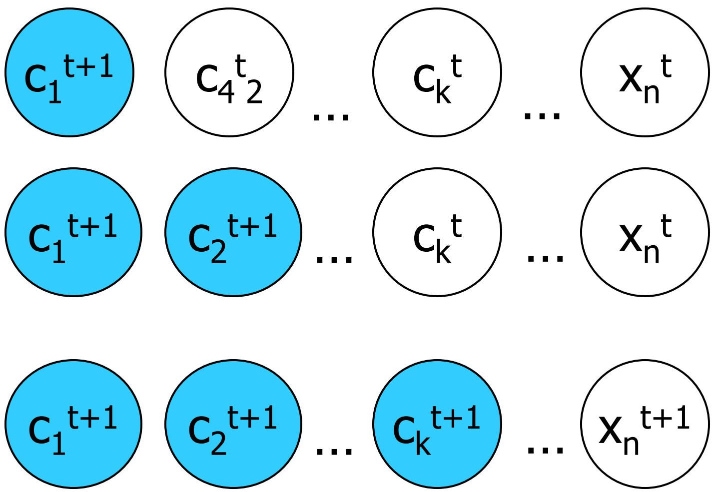
3-cutset

2-cutset

1-cutset

Cutset Sampling

Generate sample c^{t+1} from c^t , $C \subset X$:



$$C_1 = c_1^{t+1} \leftarrow P(c_1 \mid c_2^t, c_3^t, \ldots, c_K^t, e)$$

$$C_2 = c_2^{t+1} \leftarrow P(c_2 \mid c_1^{t+1}, c_3^t, \ldots, c_K^t, e)$$

...

$$C_K = c_K^{t+1} \leftarrow P(c_K \mid c_1^{t+1}, c_2^{t+1}, \ldots, c_{K-1}^{t+1}, e)$$

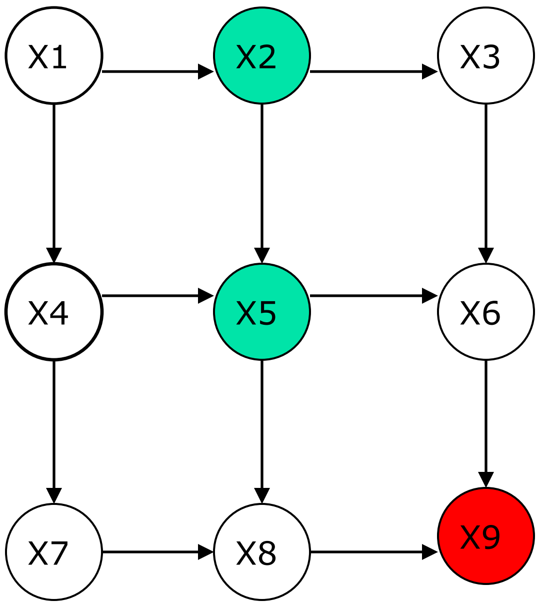
Queries:

$$\overline{P}(c_i|e) = \frac{1}{T}\sum_{t=1}^{T} P(c_i \mid c_{-i}^t, e)$$

$$\overline{P}(x_i|e) = \frac{1}{T}\sum_{t=1}^{T} P(x_i \mid c^t, e)$$

Cutset Sampling Example

Sample a new value for X_2 :



$$c^0 = \{x_2^0, x_5^0\}$$

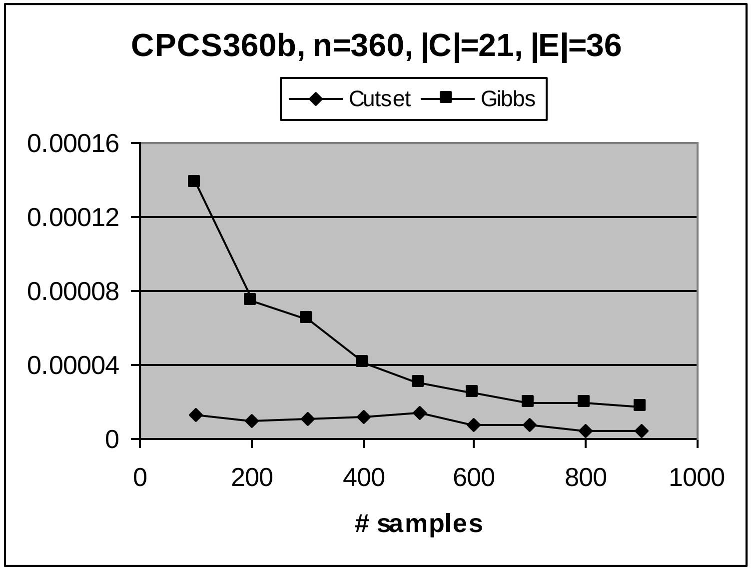
$$BTE(x_2^{'}, x_5^0, x_9)$$

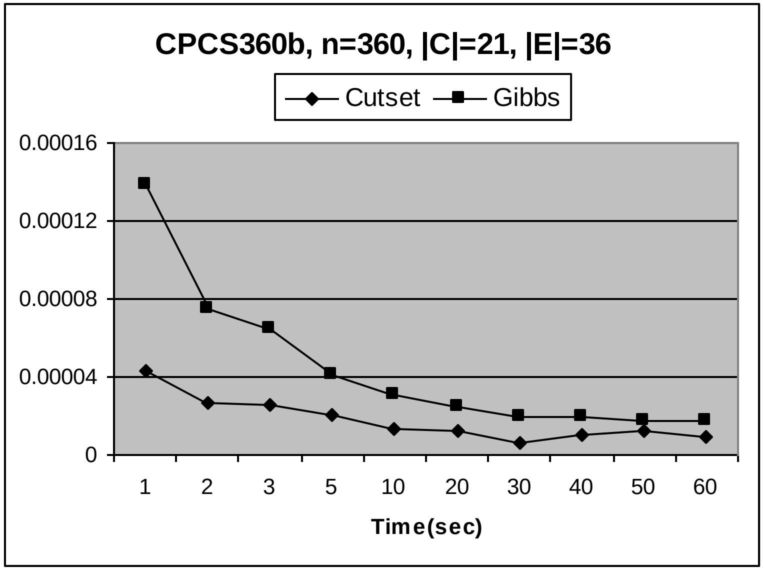
$$BTE(x_2^{''}, x_5^0, x_9)$$

$$x_2^1 \leftarrow P(x_2 | x_5^0, x_9) = \frac{1}{\alpha}\left[\begin{array}{l} BTE(x_2^{'}, x_5^0, x_9) \\ + BTE(x_2^{''}, x_5^0, x_9) \end{array}\right]$$

W-cutset sampling, Bidyuk and Dechter JAIR 2007

CPCS360b Test Results





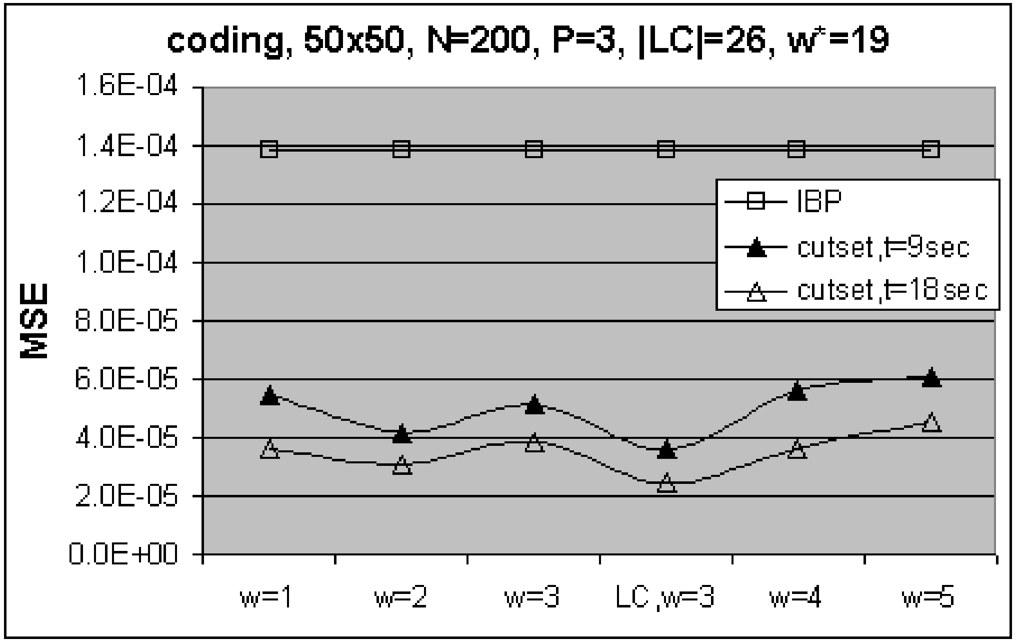
MSE vs. #samples (left) and time (right)

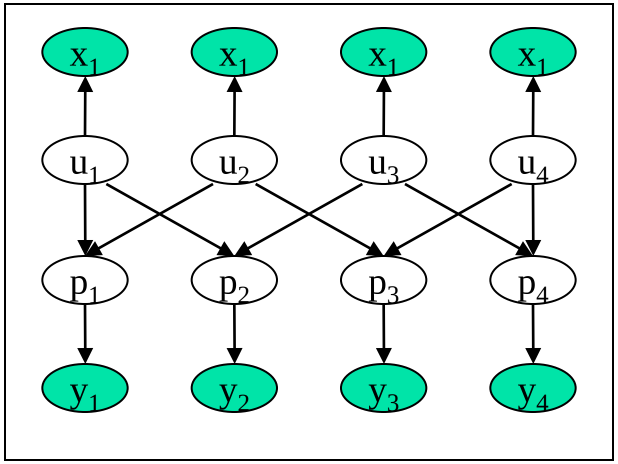
Ergodic, $|X| = 360$, $D(X_i)=2$, $|C| = 21$, $|E| = 36$

Exact Time > 60 min using Cutset Conditioning

Exact Values obtained via Bucket Elimination

Coding Networks, MSE vs. *w*





LCS,#samples=450
1-cutset,#samples=800
2-cutset,#samples=600
3-cutset,#samples=250
4-cutset,#samples=150
5-cutset,#samples=100

Note:
1-cutset=All Code Bits

SampleSearch

- Combining importance sampling with backtracking search.

Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
- Search
- Sampling
- Hybrid of search and inference
- Software
- Modeling and learning

Software & Competitions

- **How to use the software**
  - http://graphmod.ics.uci.edu/group/Software
  - http://mulcyber.toulouse.inra.fr/projects/toulbar2

- **Reports on competitions**
  - UAI-2006, 2008, 2010 Competitions
    - PE, MAR, MPE tasks
  - CP-2006 Competition
    - WCSP task

Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
- Search
- Sampling
- Hybrid of search and inference
- Modeling and learning
- Software

Modeling with Bayesian Networks

Bayesian networks will be constructed in three consecutive steps.

Step 1

Define the network variables and their values.

- A query variable is one which we need to ask questions about, such as compute its posterior marginal.
- An evidence variable is one which we may need to assert evidence about.
- An intermediary variable is neither query nor evidence and is meant to aid the modeling process by detailing the relationship between evidence and query variables.

The distinction between query, evidence and intermediary variables is not a property of the Bayesian network, but of the task at hand.

Modeling with Bayesian Networks

Bayesian networks will be constructed in three consecutive steps.

Step 2

Define the network structure (edges).

We will be guided by a causal interpretation of network structure.

The determination of network structure will be reduced to answering the following question about each network variable X: what set of variables we regard as the direct causes of X?

What about the boundary strata?

Modeling with Bayesian Networks

Step 3

Define the network CPTs.

- CPTs can sometimes be determined completely from the problem statement by objective considerations.
- CPTs can be a reflection of subjective beliefs.
- CPTs can be estimated from data.

Diagnosis I: Model from Expert

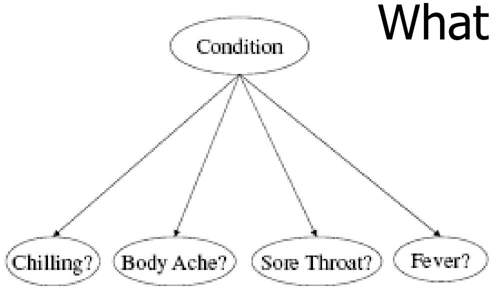
Example

The flu is an acute disease characterized by fever, body aches and pains, and can be associated with chilling and a sore throat. The cold is a bodily disorder popularly associated with chilling and can cause a sore throat. Tonsillitis is inflammation of the tonsils which leads to a sore throat and can be associated with fever.

Our goal here is to develop a Bayesian network to capture this knowledge and then use it to diagnose the condition of a patient suffering from some of the symptoms mentioned above.

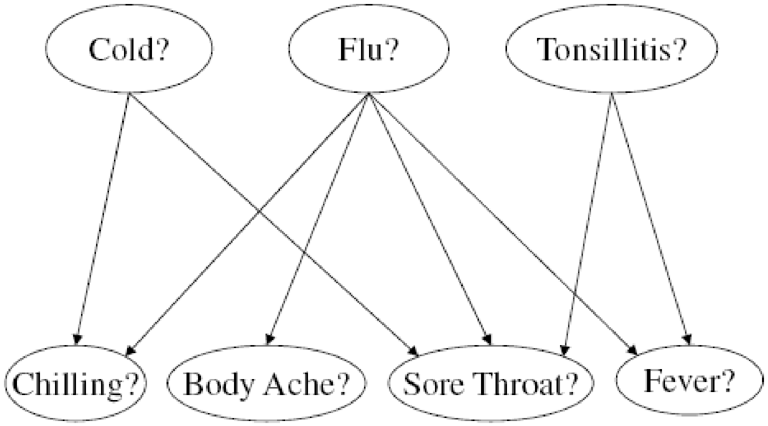
Variables? Arcs? Try it.

Diagnosis I: Model from Expert



What about?

A naive Bayes structure has the following edges C -> A1, . . . , C -> Am, where C is called the class variable and A1; : : : ;Am are called the attributes.



Variables are binary: values are either true or false. More refined information may suggest different degrees of body ache.

Diagnosis I: Learn the model from data

CPTs can be obtained from medical experts, who supply this information based on known medical statistics or subjective beliefs gained through practical experience.

CPTs can also be estimated from medical records of previous patients

Case	*Cold?*	*Flu?*	*Tonsillitis?*	*Chilling?*	*Bodyache?*	*Sorethroat?*	*Fever?*
1	true	false	?	true	false	false	false
2	false	true	false	true	true	false	true
3	?	?	true	false	?	true	false
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

? indicates the unavailability of corresponding data for that patient.

Diagnosis I: M

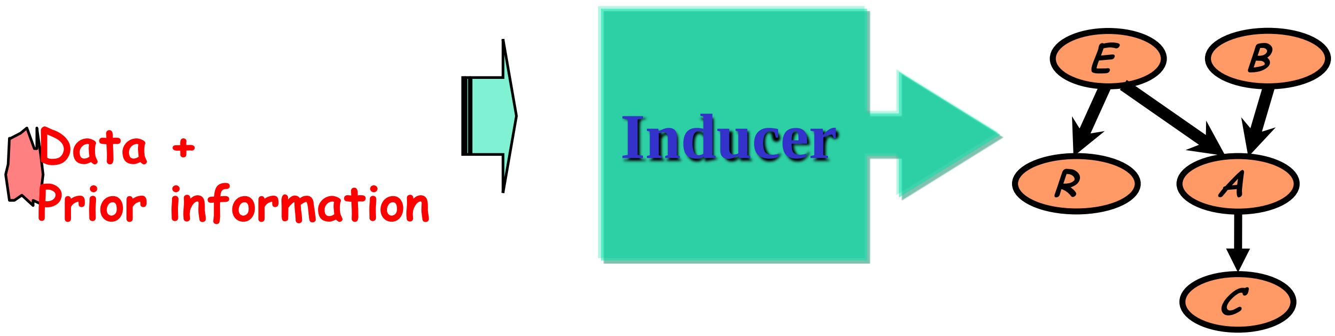
Learning the model

- Tools for Bayesian network inference can generate a network parameterization Θ, which tries to maximize the probability of seeing the given cases.
- If each case is represented by event $\mathbf{d}_i$, such tools will generate a parametrization Θ which leads to a probability distribution Pr that attempts to maximize:

$$\prod_{i=1}^{N} \mathrm{Pr}(\mathbf{d}_i).$$

- Term $\mathrm{Pr}(\mathbf{d}_i)$ represents the probability of seeing the case i.
- The product represents the probability of seeing all N cases (assuming the cases are independent).
-

Learning Bayesian networks



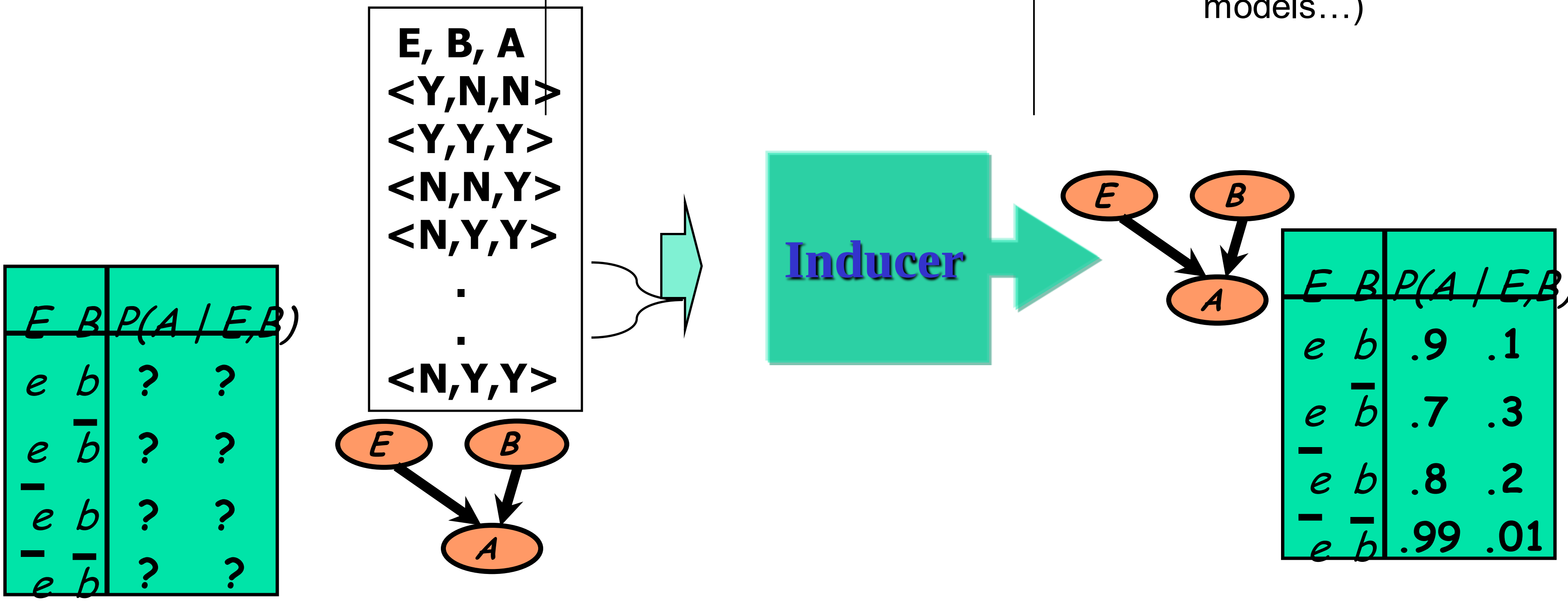
E	B	P(A | E,B)	
e	b	.9	.1
e	$\bar{b}$	.7	.3
$\bar{e}$	b	.8	.2
$\bar{e}$	$\bar{b}$	.99	.01

The Learning Problem

	Known Structure	**Unknown Structure**
Complete Data	Statistical parametric estimation (closed-form eq.)	Discrete optimization over structures (discrete search)
Incomplete Data	Parametric optimization (EM, gradient descent...)	Combined (Structural EM, mixture models…)

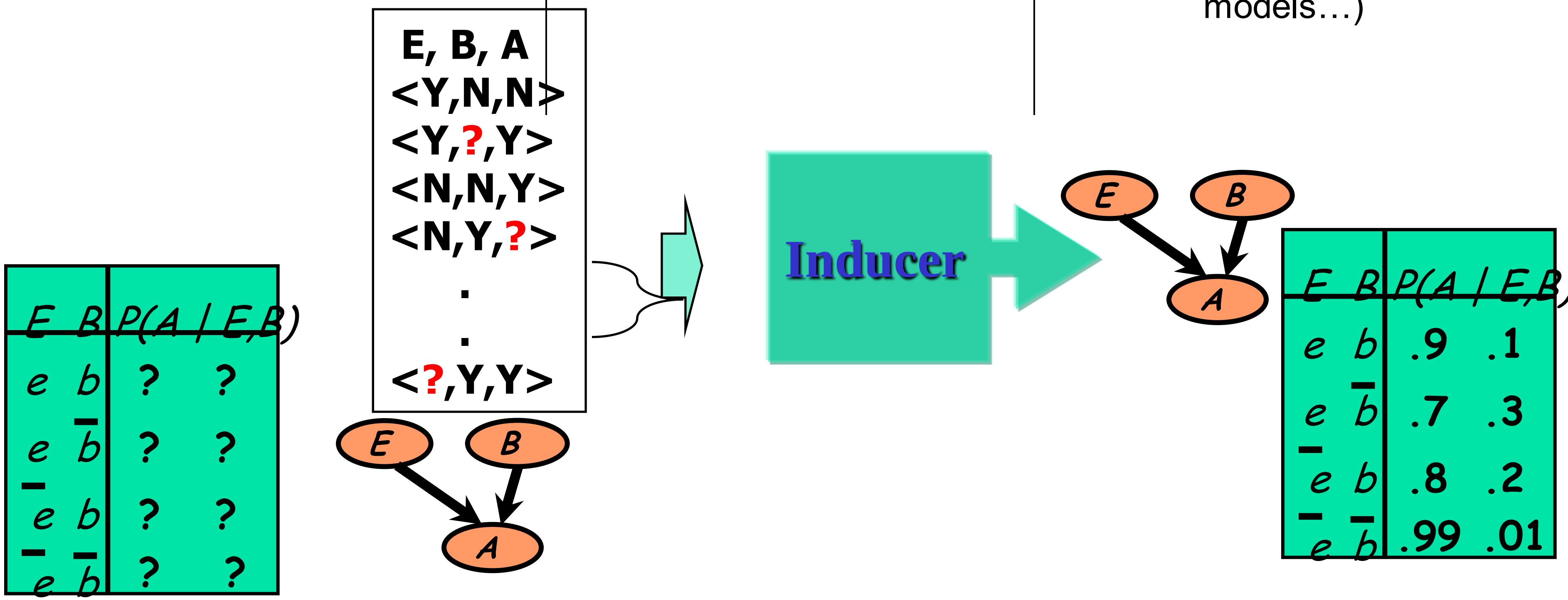
Learning Problem

	Known Structure	Unknown Structure
Complete	Statistical parametric estimation (closed-form eq.)	Discrete optimization over structures (discrete search)
Incomplete	Parametric optimization (EM, gradient descent...)	Combined (Structural EM, mixture models…)



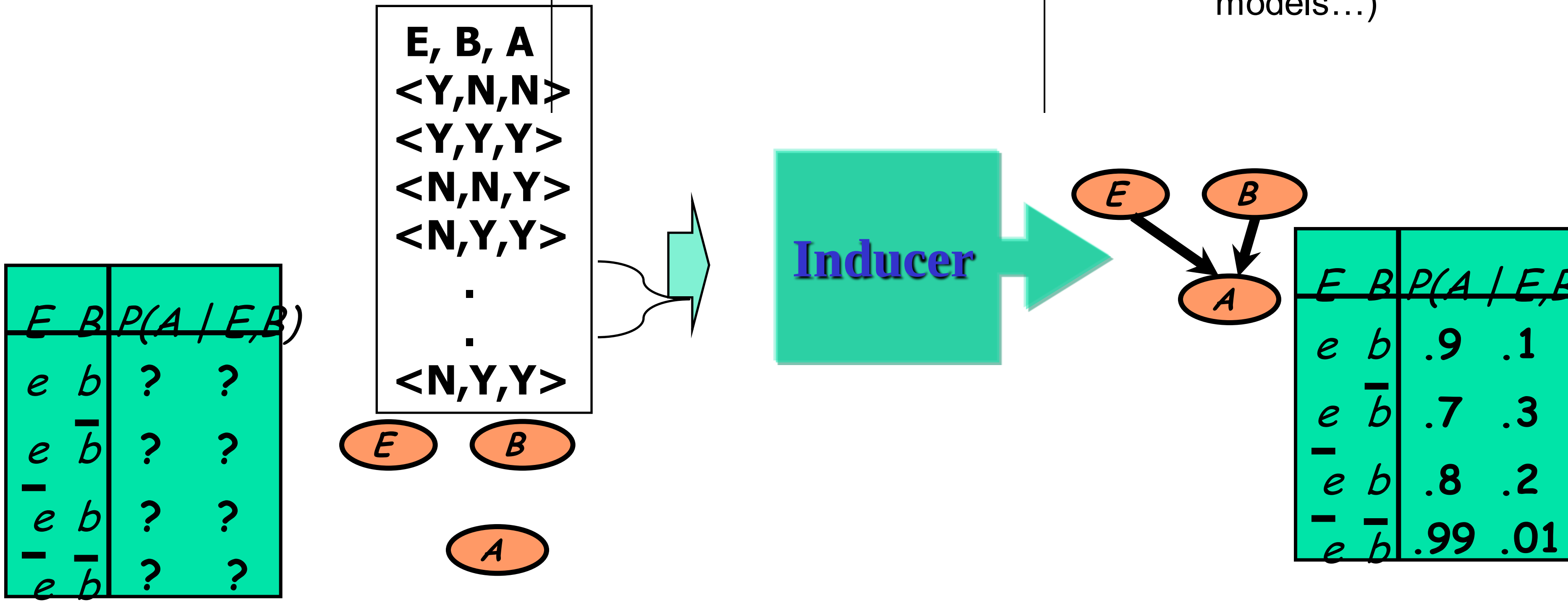
Learning Problem

	Known Structure	Unknown Structure
Complete	Statistical parametric estimation (closed-form eq.)	Discrete optimization over structures (discrete search)
Incomplete	Parametric optimization (EM, gradient descent...)	Combined (Structural EM, mixture models…)



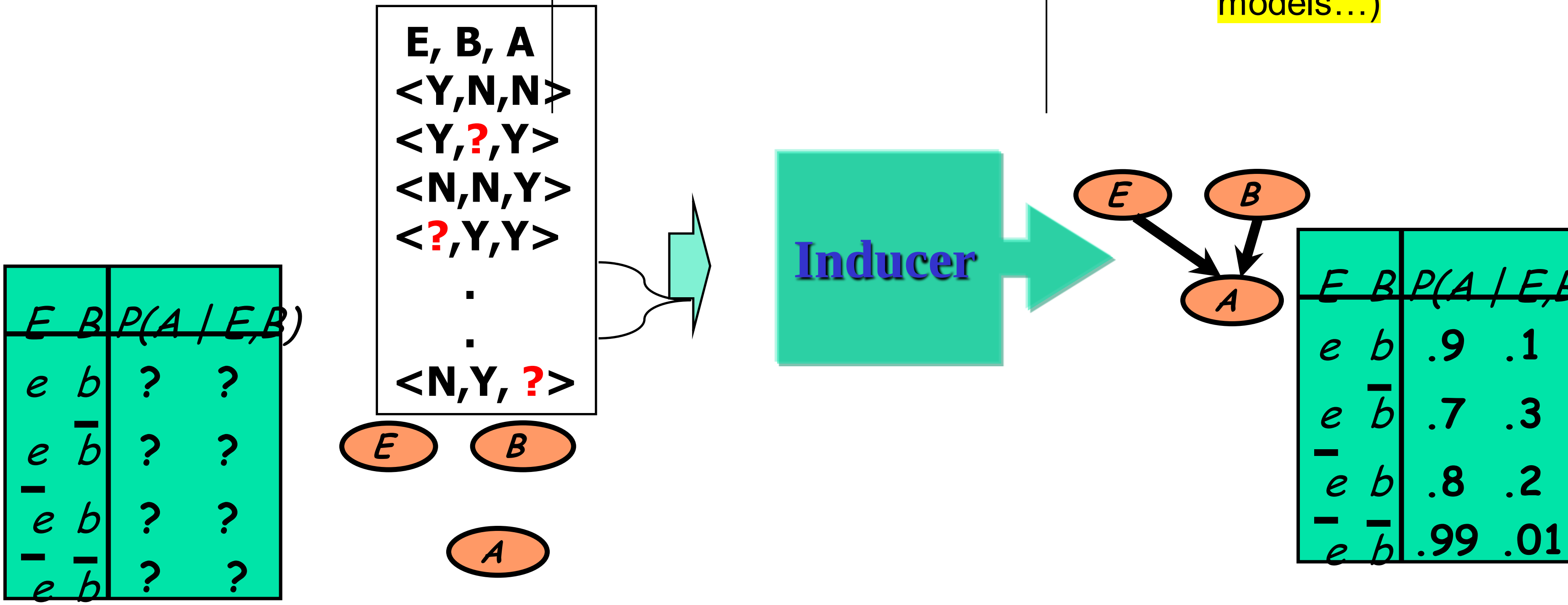
Learning Problem

	Known Structure	Unknown Structure
Complete	Statistical parametric estimation (closed-form eq.)	Discrete optimization over structures (discrete search)
Incomplete	Parametric optimization (EM, gradient descent...)	Combined (Structural EM, mixture models…)



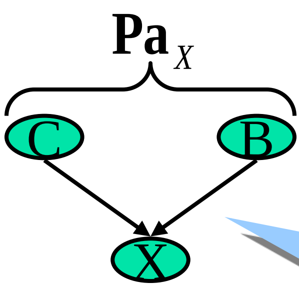
Learning Problem

	Known Structure	Unknown Structure
Complete	Statistical parametric estimation (closed-form eq.)	Discrete optimization over structures (discrete search)
Incomplete	Parametric optimization (EM, gradient descent...)	Combined (Structural EM, mixture models…)



Learning Parameters: complete data

- ML-estimate: $\max_{\Theta} \underbrace{\log P(D \mid \Theta)}$ - decomposable!



Multinomial

$$\theta_{x,\mathbf{pa}_X} = P(x \mid \mathbf{pa}_X)$$

counts

$$\mathrm{ML}(\theta_{x,\mathbf{pa}_X}) = \frac{N_{x,\mathbf{pa}_X}}{\sum_x N_{x,\mathbf{pa}_X}}$$

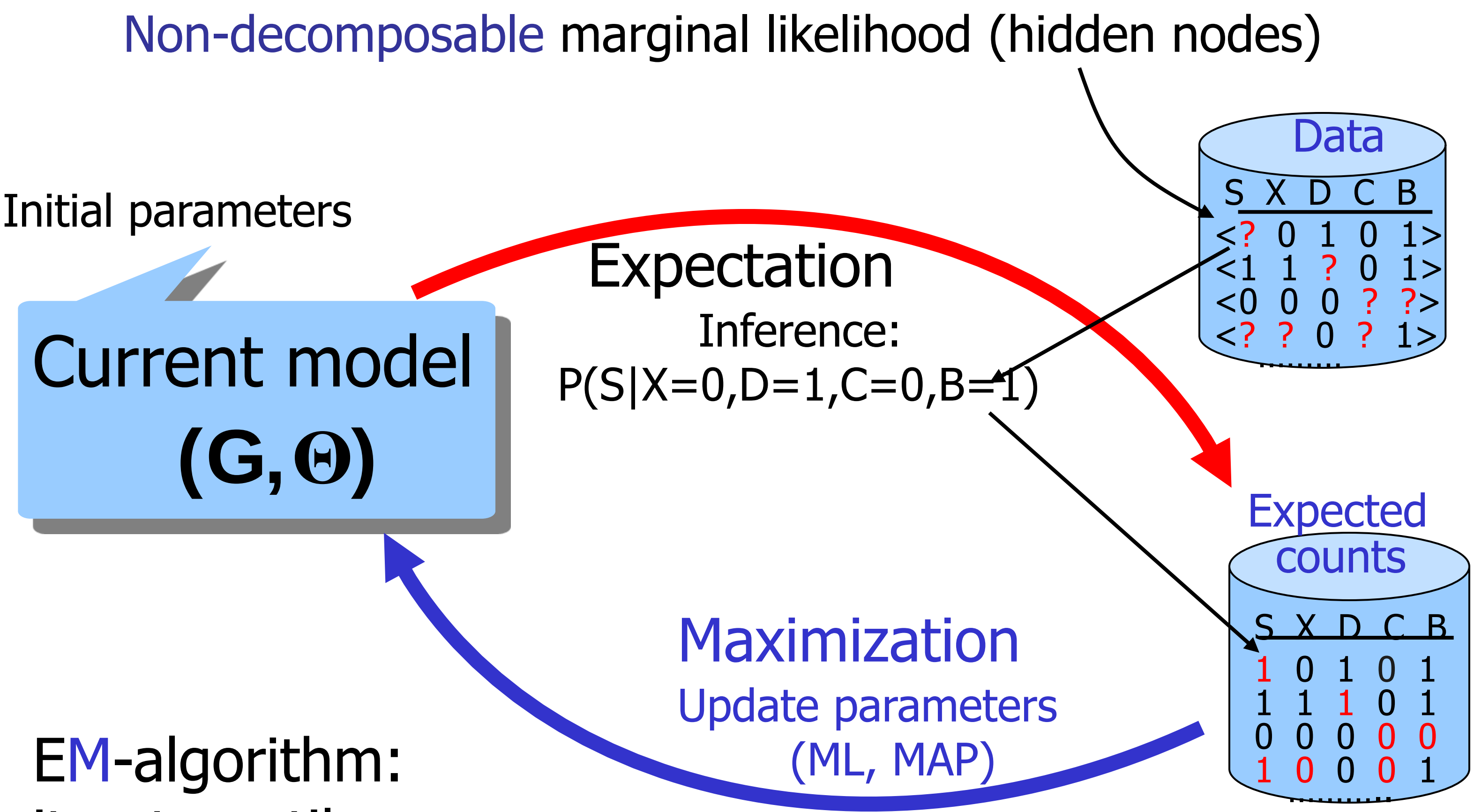
- MAP-estimate (Bayesian statistics) $\max_{\Theta} \underbrace{\log P(D \mid \Theta) P(\Theta)}$

Conjugate priors - **Dirichlet** $Dir(\theta_{\mathbf{pa}_X} \mid \alpha_{1,\mathbf{pa}_X}, \ldots, \alpha_{m,\mathbf{pa}_X})$

$$\mathrm{MAP}(\theta_{x,\mathbf{pa}_X}) = \frac{N_{x,\mathbf{pa}_X} + \alpha_{x,\mathbf{pa}_X}}{\sum_x N_{x,\mathbf{pa}_X} + \underbrace{\sum_x \alpha_{x,\mathbf{pa}_X}}}$$

Equivalent sample size (prior knowledge)

Learning Parameters: incomplete data



EM-algorithm:
iterate until convergence

Learning graph structure

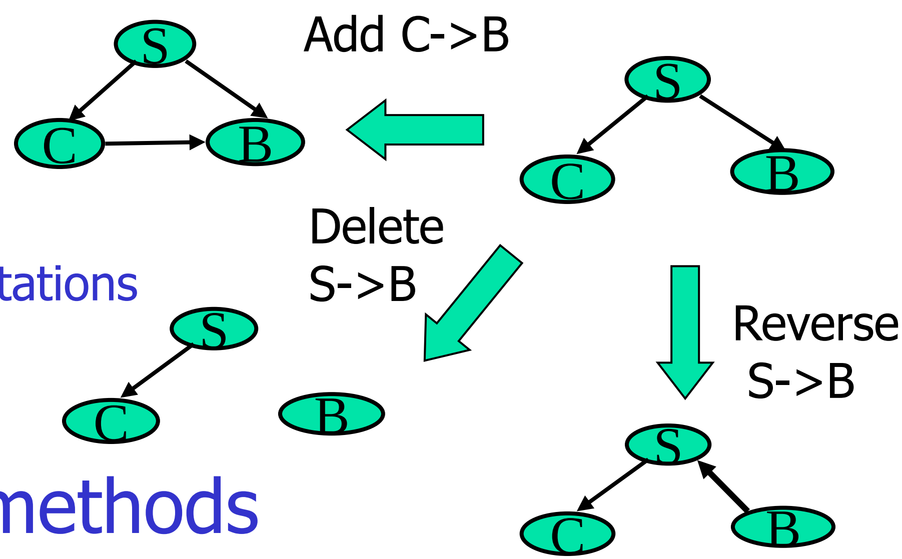
Find $\hat{G} = \arg\max_{G} \textbf{Score(G)}$

NP-hard optimization

- Heuristic search:
  - Greedy local search
  - Best-first search
  - Simulated annealing

  Complete data – local computations

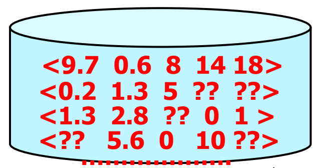
  Incomplete data (score non-decomposable): Structural EM

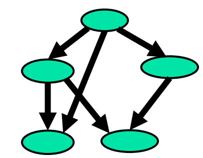


- Constrained-based methods
  - Data impose independence relations (constrains)

Scoring functions:
Minimum Description Length (MDL)

- Learning ⇔ data compression





$$MDL(BN \mid D) = \underbrace{-\log P(D \mid \Theta, G)}_{\text{DL(Data|model)}} + \underbrace{\frac{\log N}{2} \mid \Theta \mid}_{\text{DL(Model)}}$$

- Other: MDL = -BIC (Bayesian Information Criterion)
- Bayesian score (BDe) - asymptotically equivalent to MDL

Thank you