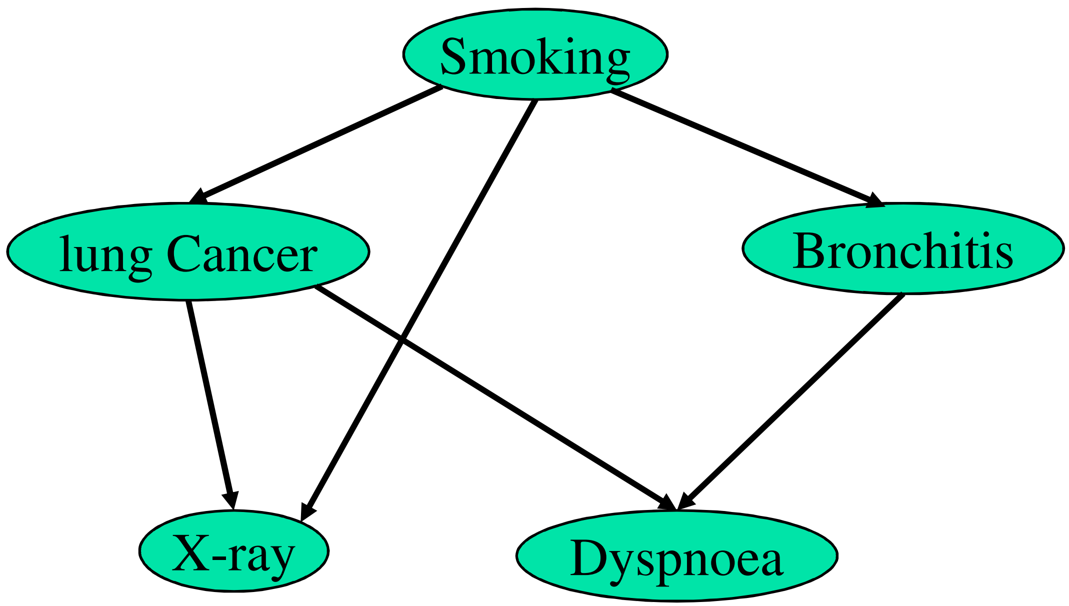
Exact Inference Algorithms for Probabilistic Reasoning;

COMPSCI 276

Fall 2007

Belief Updating



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

Probabilistic Inference Tasks

- Belief updating:

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

- Finding most probable explanation (MPE)

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

- Finding maximum a-posteriory hypothesis

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

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

- Finding maximum-expected-utility (MEU) decision

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

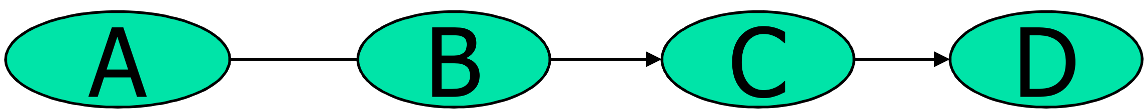
$D \subseteq X$: **decision variables**
$U(\bar{x})$: **utility function**

Belief updating is NP-hard

- Each sat formula can be mapped to a bayesian network query.
- Example:
- (u,~v,w) and (~u,~w,y) sat?

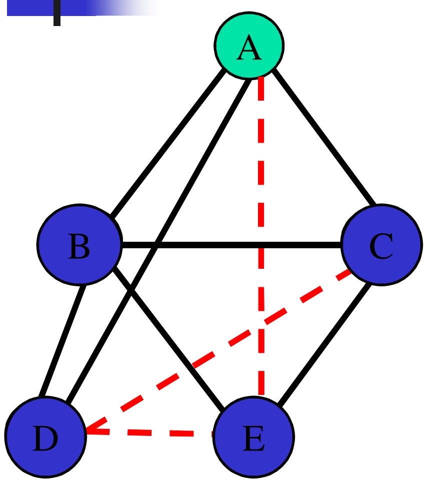
Motivation

- Given a chain show how it works
- How can we compute P(D)? P(D|A=0)?
- P(A|D=0)?



- Brute force O(k^4)

Belief updating: P(X|evidence)=?



"Moral" graph

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

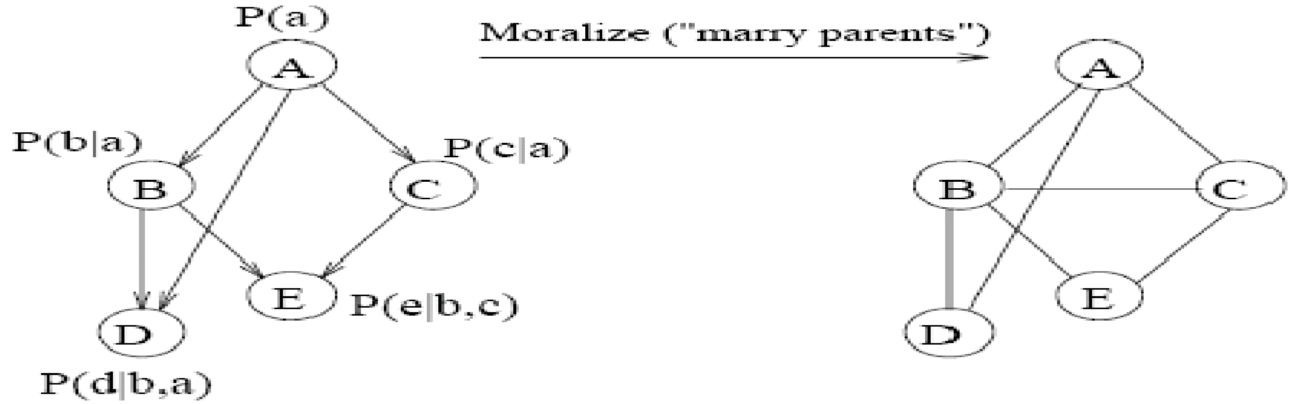
$$\sum_{e=0,d,c,b} P(a)\underbrace{P(b|a)}P(c|a)\underbrace{P(d|b,a)P(e|b,c)}=$$

$$P(a)\sum_{e=0}\sum_{d}\sum_{c}P(c|a)\sum_{b}\underbrace{P(b|a)P(d|b,a)P(e|b,c)}_{h^{B}(a,d,c,e)}$$

Variable Elimination

Belief Updating

$P(a|e=0) = \alpha P(a, e=0).$



Ordering: $a,\ b,\ c,\ d,\ e$

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

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

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

Ordering: $a,\ e,\ d,\ c,\ b$

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

$P(a, e=0) = P(a) \sum_e \sum_d \sum_c P(c|a) \sum_b P(b|a) P(d|a, b)$
$P(e|b, c)$

Backwards Computation = Elimination

Ordering: a, b, c, d, e

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

$= P(a)\sum_b P(b|a)\sum_c P(c|a)P(e=0|b,c)\sum_d P(d|b,a)$
$= P(a)\sum_b P(b|a)\lambda_D(a,b)\sum_c P(c|a)P(e=0|b,c)$
$= P(a)\sum_b P(b|a)\lambda_D(a,b)\lambda_C(a,b)$
$= P(a)\lambda_B(a)$

The Bucket elimination process:

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

Backwards Computation, Different Ordering

Ordering: a, e, d, c, b

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

$P(a) \sum_{e=0} \sum_d \sum_c P(c|a) \lambda_B(a, d, c, e)$
$P(a) \sum_{e=0} \sum_d \lambda_C(a, d, e)$
$P(a) \sum_{e=0} \lambda_D(a, e)$
$P(a) \lambda_D(a, e = 0)$

The bucket elimination Process:

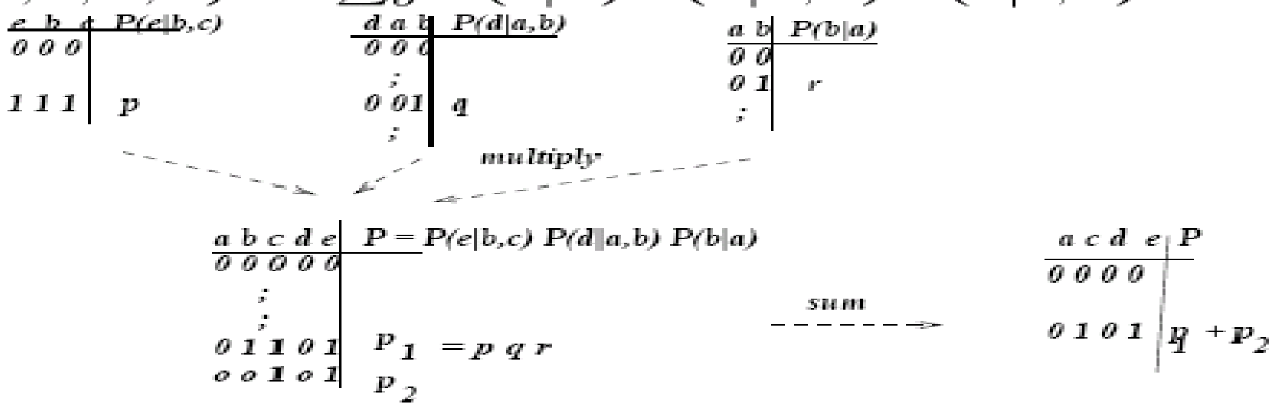
$bucket(B) = \quad P(e|b, c), P(d|a, b), P(b|a)$
$bucket(C) = \quad P(c|a) \quad || \quad \lambda_B(a, d, c, e)$
$bucket(D) = \quad || \quad \lambda_C(a, d, e)$
$bucket(E) = \quad e = 0 \quad || \quad \lambda_D(a, e)$
$bucket(A) = \quad P(a) \quad || \quad \lambda_D(a, e = 0)$

The Bucket Operation

Elimination: multiply and sum

$bucket(B) = \{P(e|b,c), P(d|a,b), P(b|a)\} \rightarrow$

$\lambda_B(a,c,d,e) = \sum_b P(b|a)P(d|a,b)P(e|b,c)$



Observed bucket:

$bucket(B) = \{P(e|b,c), P(d|a,b), P(b|a), b = 1\} \rightarrow$

$\lambda_B(a) = P(b = 1|a)$

$\lambda_B(a,d) = P(d|a, b = 1)$

$\lambda_B(e,c) = P(e|b = 1, c)$.

Bucket elimination

Algorithm *elim-bel* (Dechter 1996)

$\sum_b \prod$ ← Elimination operator

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

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

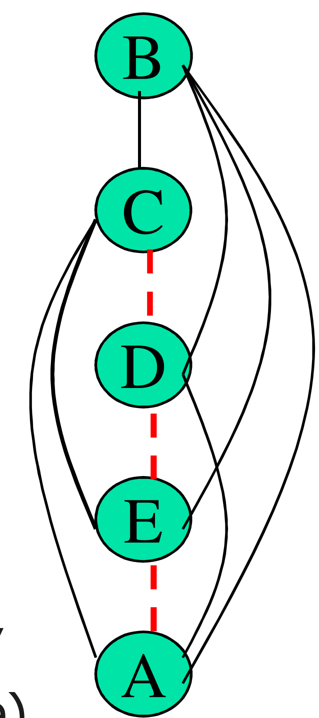
bucket D: $h^C(a, d, e)$

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

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

P(a|e=0)

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



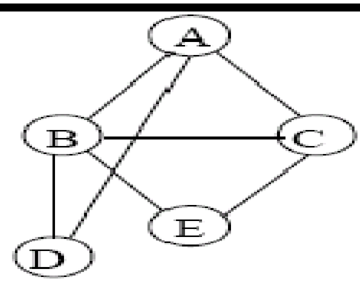
Elim-bel

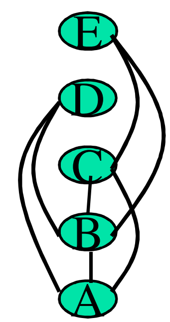
Input: A belief network $\{P_1, ..., P_n\}$, d,e.
Output: belief of X_1 given e.

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

Bucket Elimination and Induced Width





Ordering: a, b, c, d, e

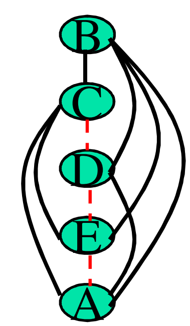
$bucket(E) = \quad P(e|b,c), \; e = 0$

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

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

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

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



Ordering: a, e, d, c, b

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

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

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

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

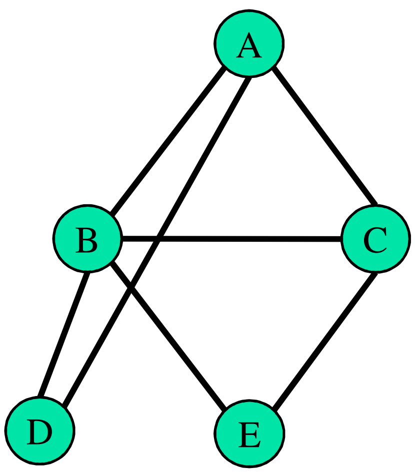
$bucket(A) = \quad P(a) \;\; || \;\; \lambda_E(a)$

Complexity of elimination

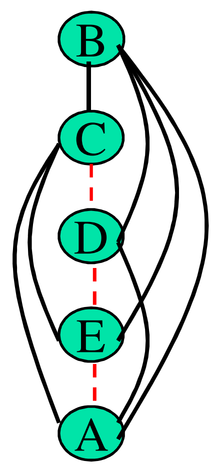
$$O(n\exp(w^*(d)))$$

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

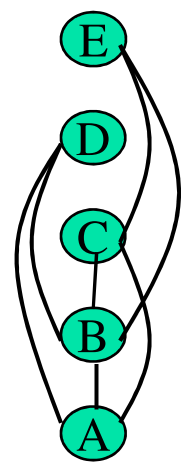
The effect of the ordering:



"Moral" graph



$w^*(d_1) = 4$

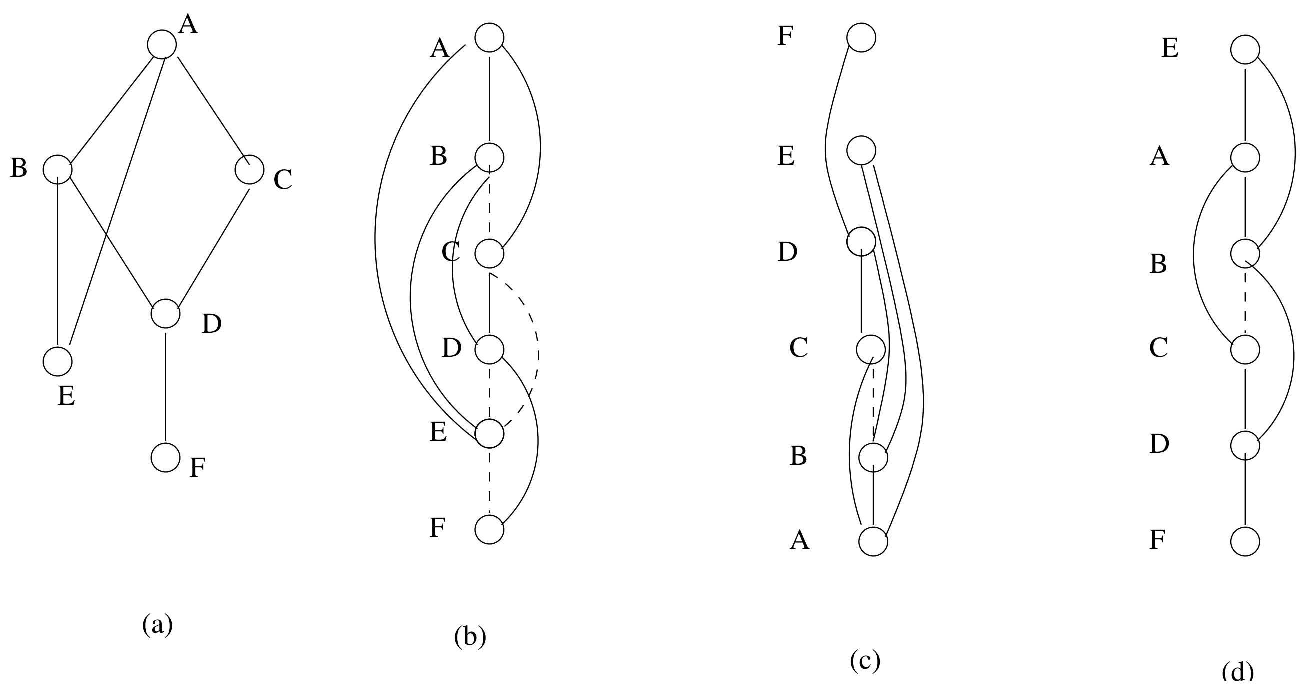


$w^*(d_2) = 2$

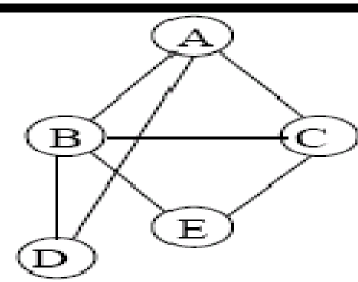
Finding small induced-width

- NP-complete
- A tree has induced-width of ?
- Greedy algorithms:
  - Min width
  - Min induced-width
  - Max-cardinality
  - Fill-in (thought as the best)
  - See anytime min-width (Gogate and Dechter)

Different Induced graphs



Handling Observations



Observing $b = 1$

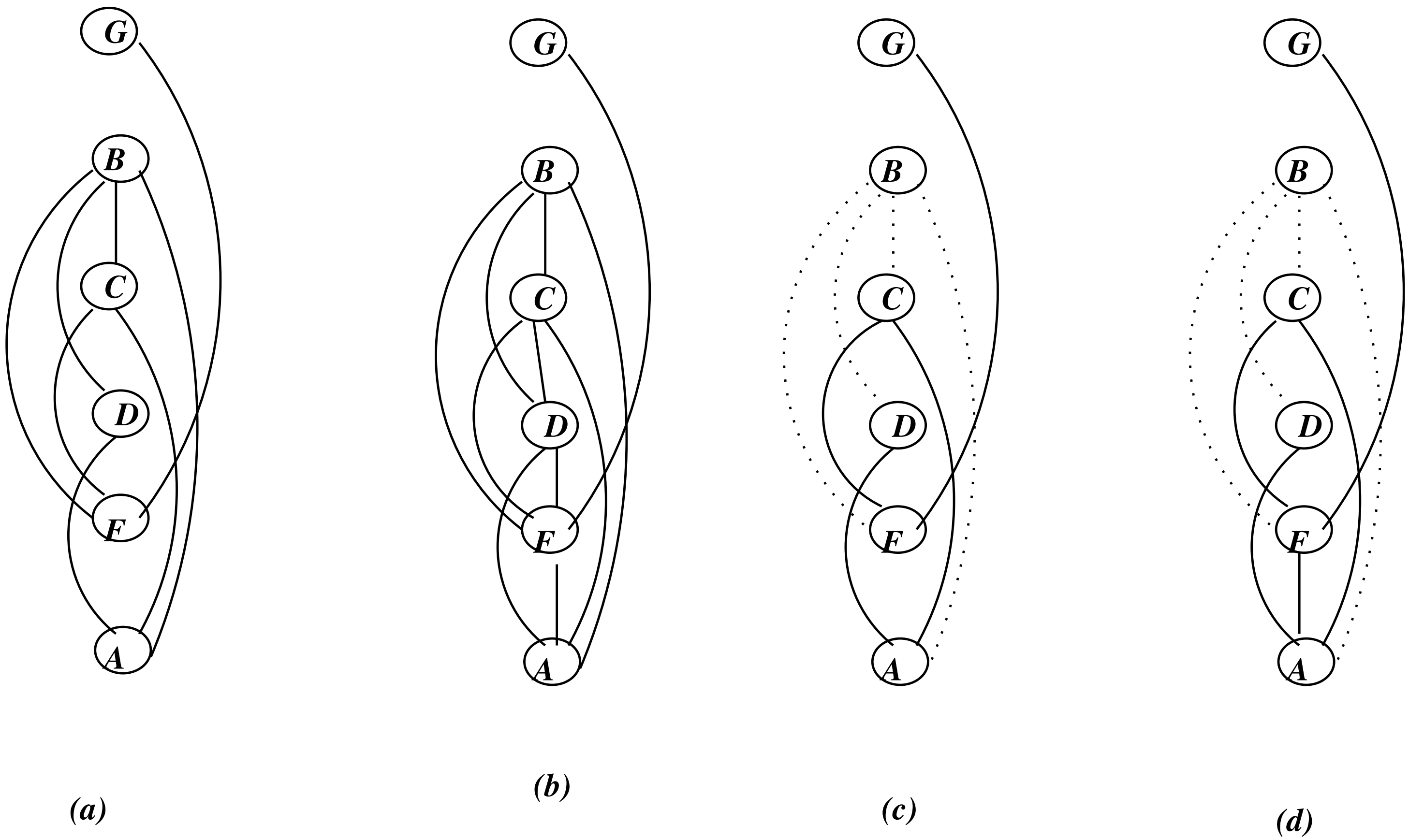
Ordering: a, e, d, c, b

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

Ordering: a, b, c, d, e

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

The impact of observations



Irrelevant buckets for elim-bel

Buckets that sum to 1 are **irrelevant**.
Identification: no evidence, no new functions.

Recursive recognition : ($bel(a|e)$)
$bucket(E) =$ $P(e|b,c)$, $e = 0$
$bucket(D) =$ $P(d|a,b)$,...skipable bucket
$bucket(C) =$ $P(c|a)$
$bucket(B) =$ $P(b|a)$
$bucket(A) =$ $P(a)$

Complexity: Use induced width in moral graph without irrelevant nodes, then update for evidence arcs.

Use the ancestral graph only

From Bucket elimination to bucket-tree elimination

Bucket G: $P(G|F)$

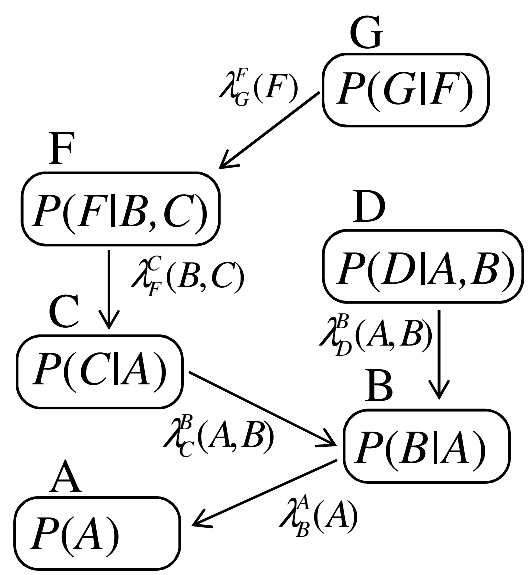
Bucket F: $P(F|B,C)$ $\lambda_G^F(F)$

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

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

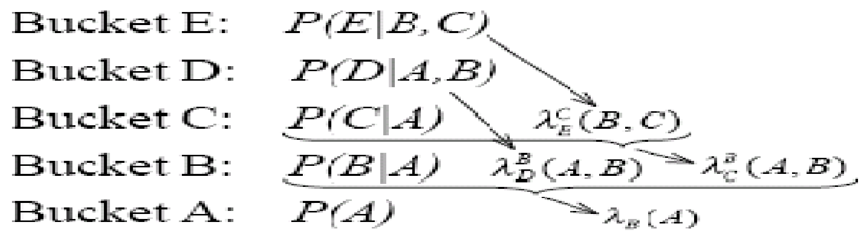
Bucket B: $P(B|A)$ $\lambda_D^B(A,B)$ $\lambda_C^B(A,B)$

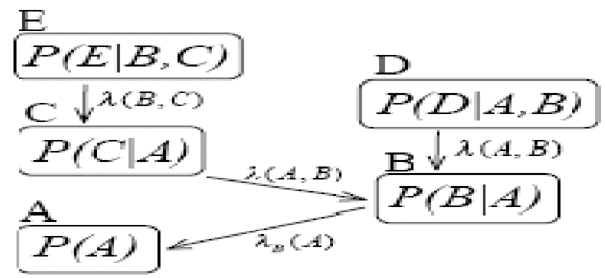
Bucket A: $P(A)$ $\lambda_B^A(A)$

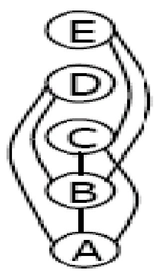


T

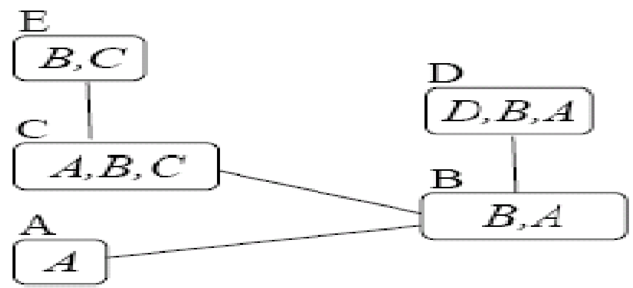
From Bucket-Elimination To Bucket Trees







$T=$



Definition: T is a bucket tree.

Theorem: T is an i-map of G.

- Variable-elimination can be viewed as message-passing (elimination) using a rooted bucket tree.
- Any variable (bucket) can be the root.

Propagation in a Bucket Tree

<u>Definitions:</u>

- Let G be a Bayesian network, d, an ordering and $B_1 \ldots B_n$ the final bucket created processing along $d = x_1 \ldots x_n$.
- Let B_i be the set of variables appearing in bucket i when it is processed.

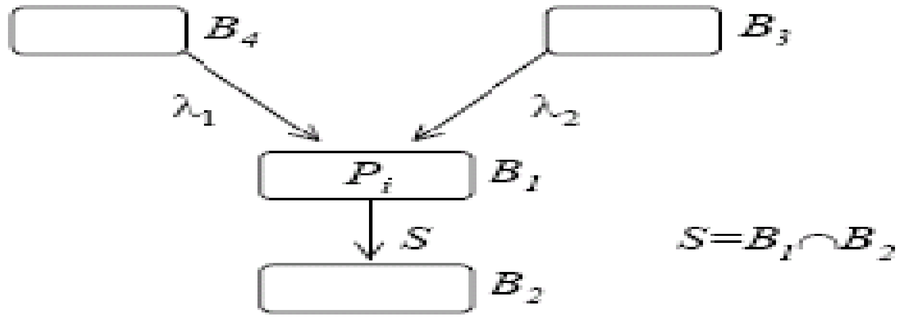
<u>Bucket Tree:</u>

- A bucket tree has each B_i cluster as a node and there is an arc from B_i to B_j if the function created at B_i was placed in B_j

<u>Graph-Based Definition:</u>

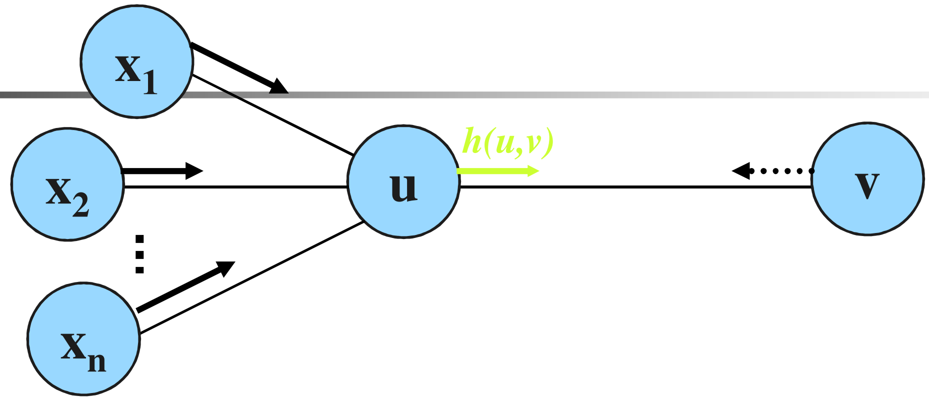
- Let G_d be the induced graph along d. Each variable x and it's earlier neighbors in a node, B_x. There is an arc from B_x to B_y if y is the closest parent of x.

Generalization:Eliminate (sum over) Variables Not in Separators



- Multiply all incoming messages, and P_i's in the bucket and sum over $B_1 \cap B_2$.
- $\lambda_{B_1}^{B_2}(s) = \sum_{B_1 - s} (\Pi \lambda_i) \bullet (\Pi P_i)$
- Given a rooted bucket tree, T, every node can be the "root" of the variables-elimination computation.
- If B_3 is the root, bucket B_2 and then Bucket B_1 should be processed; π-messages sent from B_2 to B_1 and from B_1 To B_3

Bucket-Tree propagation

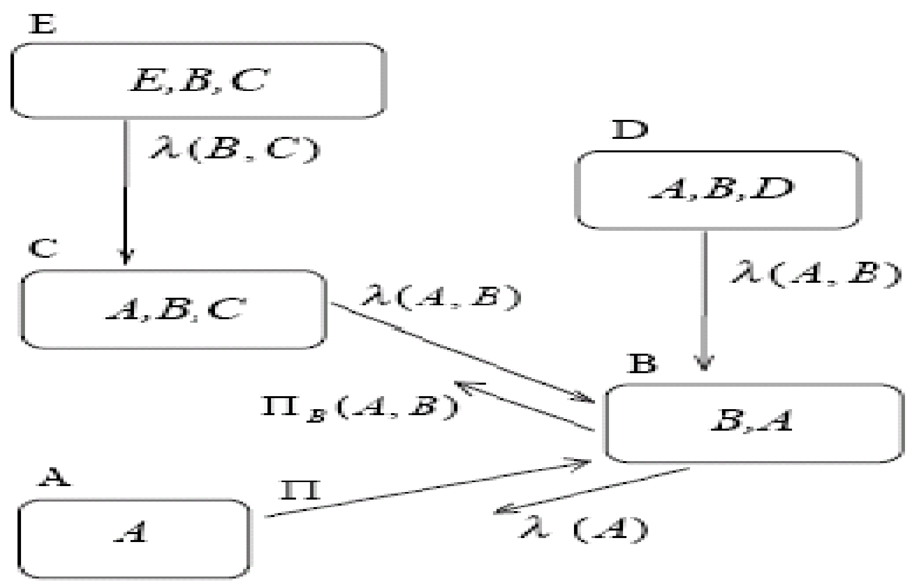


$$cluster(u) = \psi(u) \cup \{h(x_1,u), h(x_2,u), ..., h(x_n,u), h(v,u)\}$$

Compute the message :

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

Upwards Messages On The Bucket Tree



$$\Pi(A) = P(A)$$

$$\Pi_B^P(A,B) = P(B,A) \bullet \lambda_C^B(A,B)$$

$$\Pi_B^C(A,B) = P(B,A) \bullet \Pi(A) \bullet \lambda_D^B(A,B)$$

$$\Pi_C^E(B,C) = \sum_A P(C,A) \bullet \Pi_B^C(A,B)$$

BTE: full Execution

Bucket G: $P(G/F)$ $\quad\quad$ $\prod_F^G(F)$

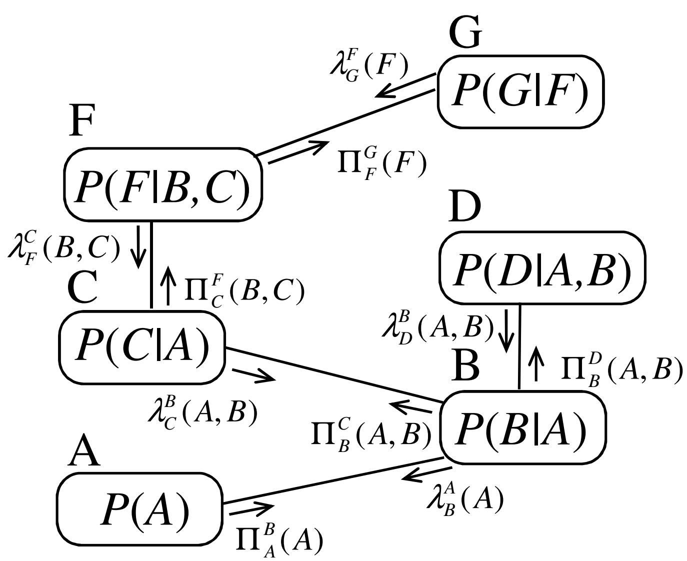
Bucket F: $P(F/B,C)$ $\lambda_G^F(F)$ $\quad\quad$ $\prod_C^F(B,C)$

Bucket D: $P(D/A,B)$ $\quad\quad$ $\prod_B^D(A,B)$

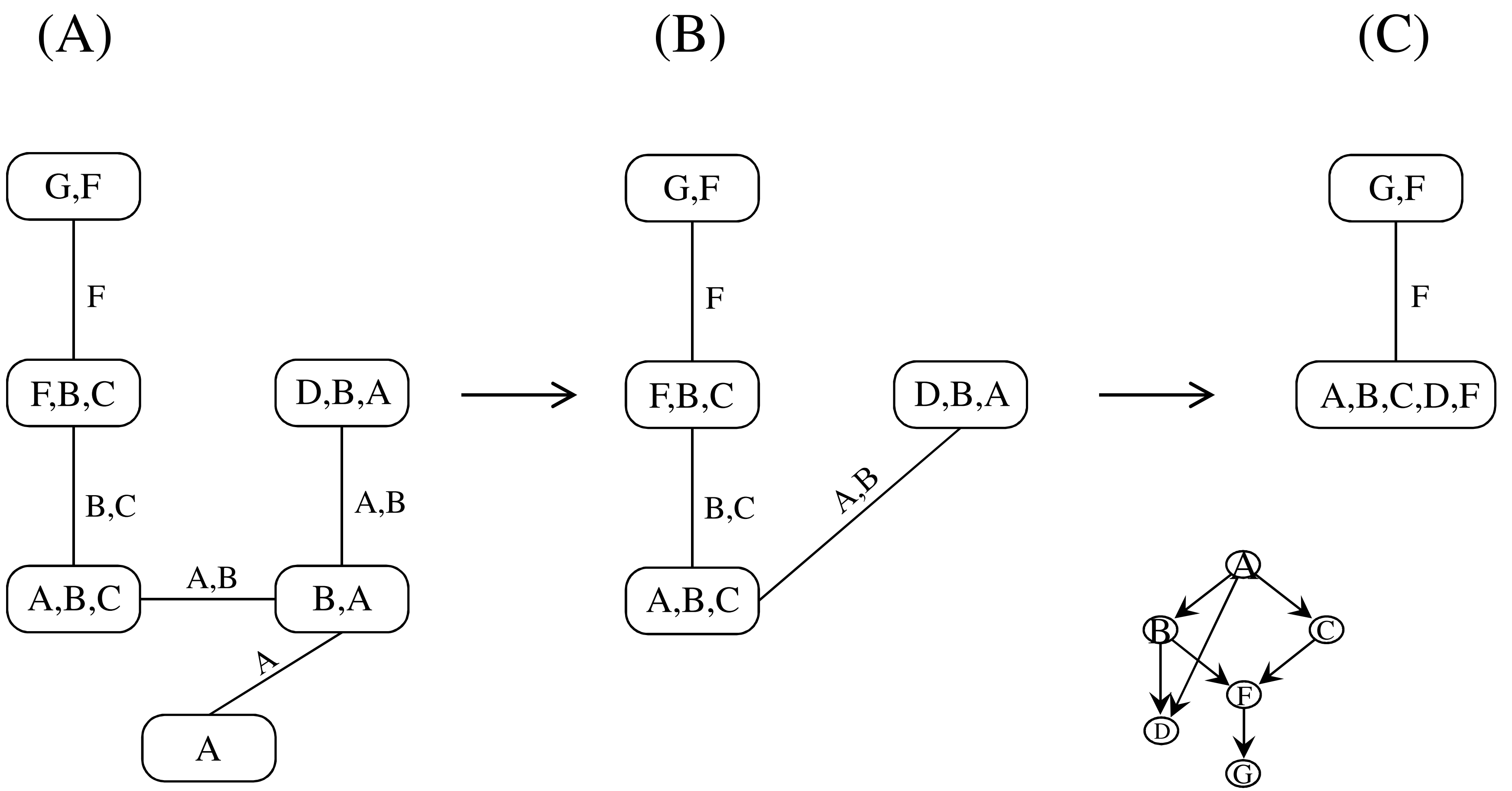
Bucket C: $P(C/A)$ $\lambda_F^C(B,C)$ $\quad\quad$ $\prod_B^C(A,B)$

Bucket B: $P(B/A)$ $\lambda_D^B(A,B)$ $\lambda_C^B(A,B)$ $\quad\quad$ $\prod_A^B(A)$

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



From buckets to superbucket to clusters



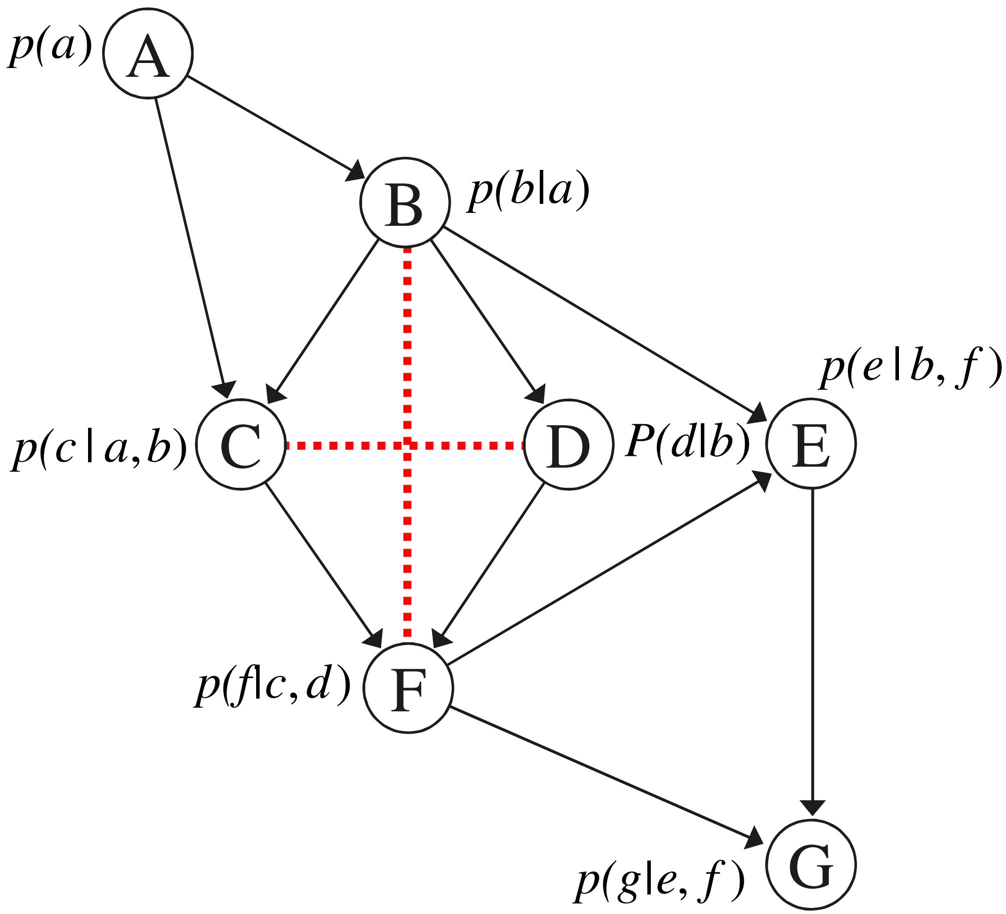
(Cluster) Tree Decomposition and elimination (CTE)

- A (cluster)-tree decomposition is a set of subsets of variables (clusters) connected by a tree structure:
  - 1. Every function (CPT) has at least one cluster that contains its scope. The function is assigned to one such cluster.
  - 2. The cluster-tree obeys the running intersection property.

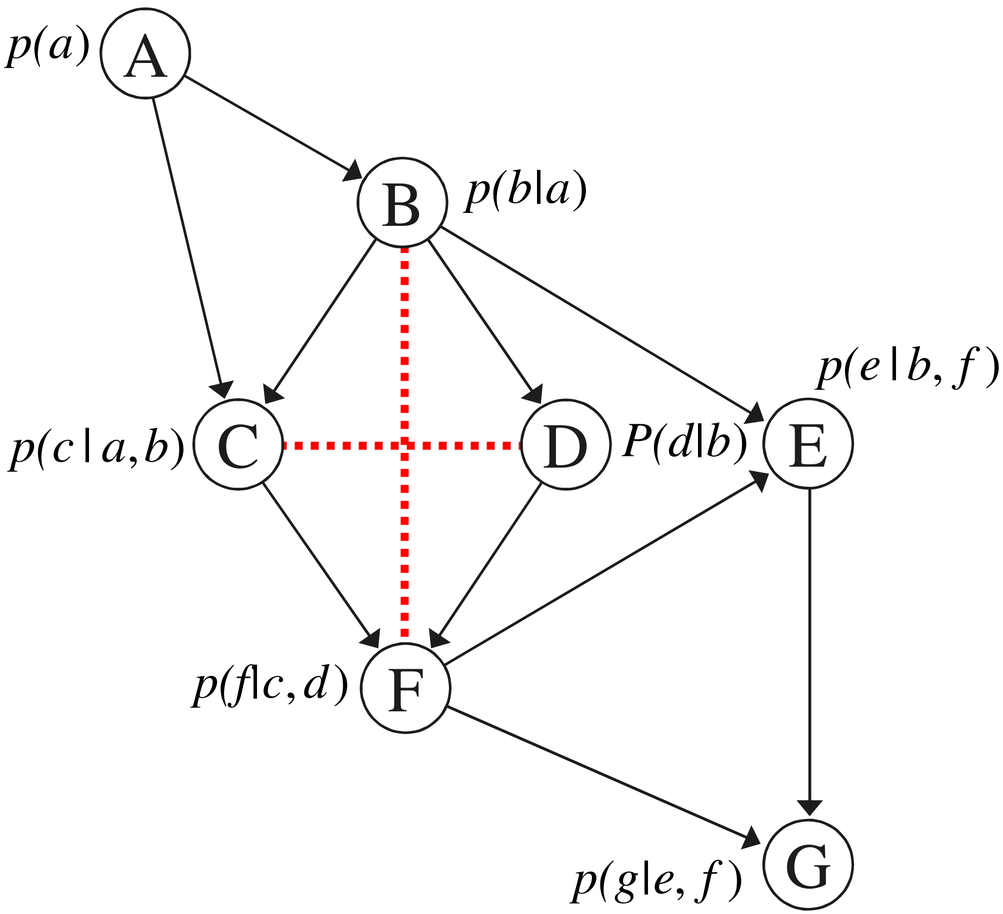
- Proposition: If T is a cluster-tree decomposition, then any tree obtained by merging adjacent clusters (the variable set and the functions) is also a tree-decomposition.

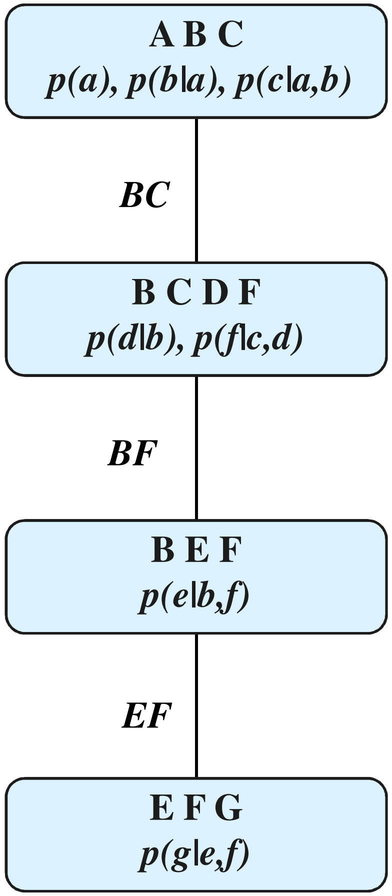
- Join-Tree: a tree-decomposition where all clusters are maximal.

Tree Decomposition for belief updating



Tree Decomposition

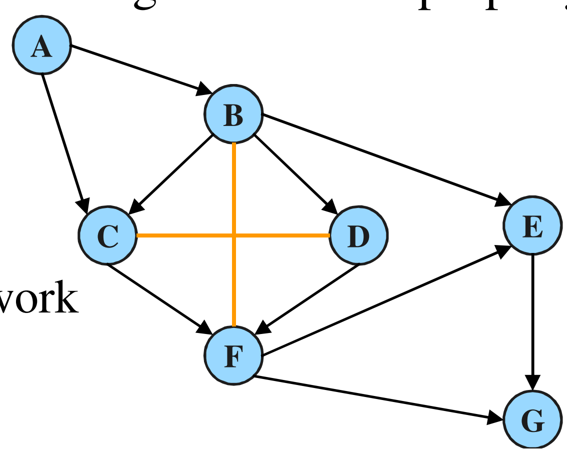




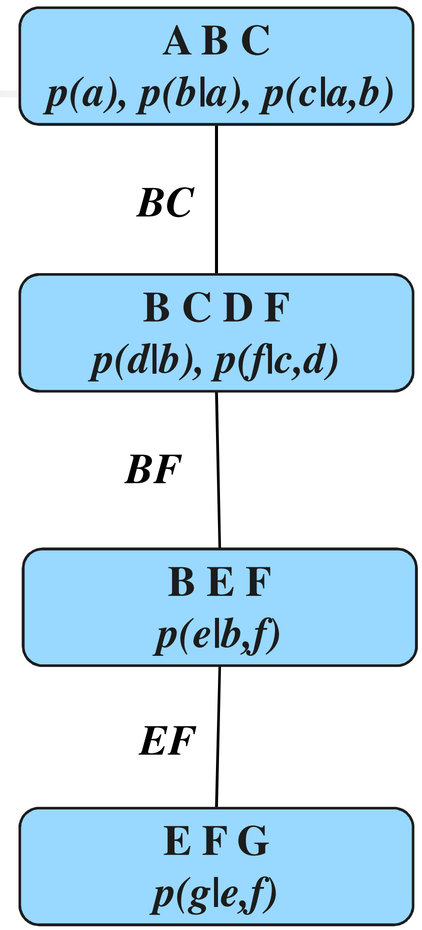
Tree decompositions (more formal)

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)

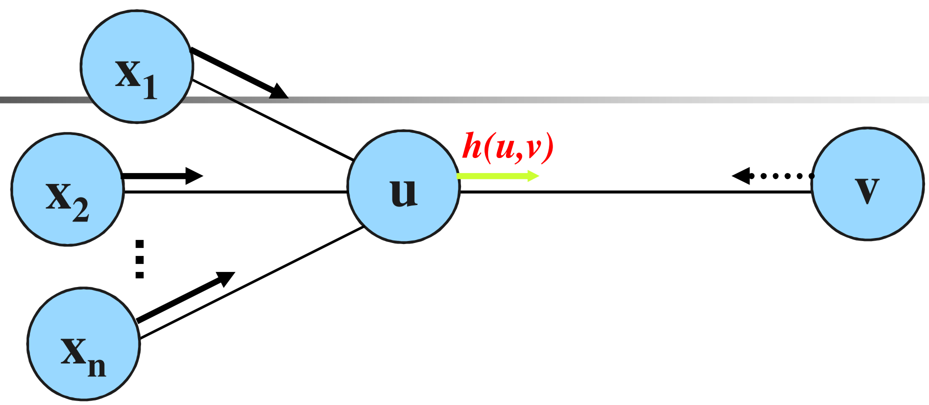


Belief network



Tree decomposition

Same Message Passing



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

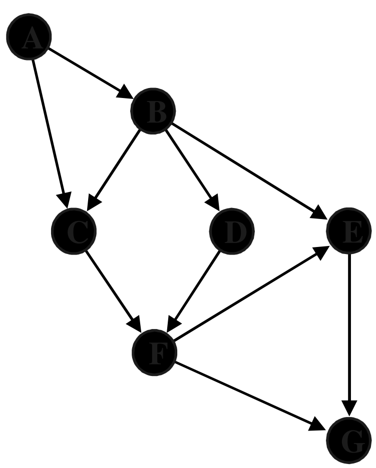
Compute the message :

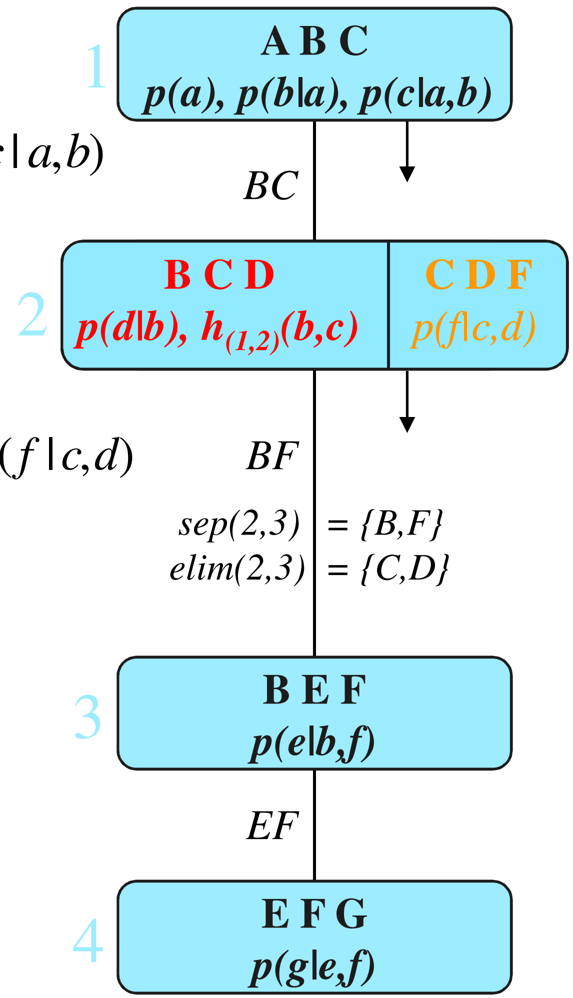
$$h(u,v) = \sum_{elim(u,v)} \prod_{f \in cluster(u) - \{h(v,u)\}} f$$

Cluster Tree Elimination

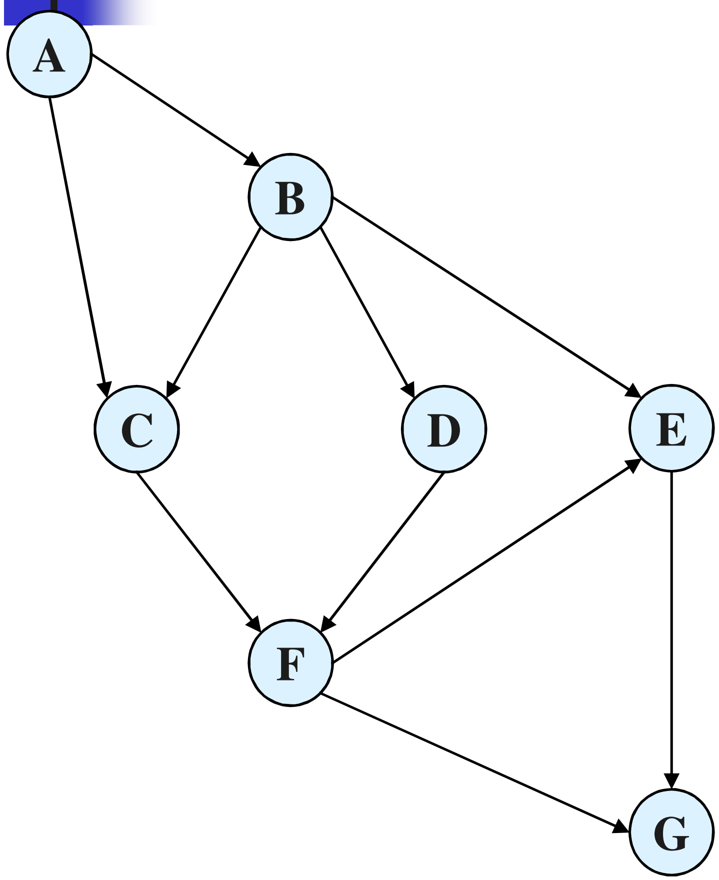
$$h_{(1,2)}(b,c) = \sum_{a} p(a) \cdot p(b \mid a) \cdot p(c \mid a,b)$$

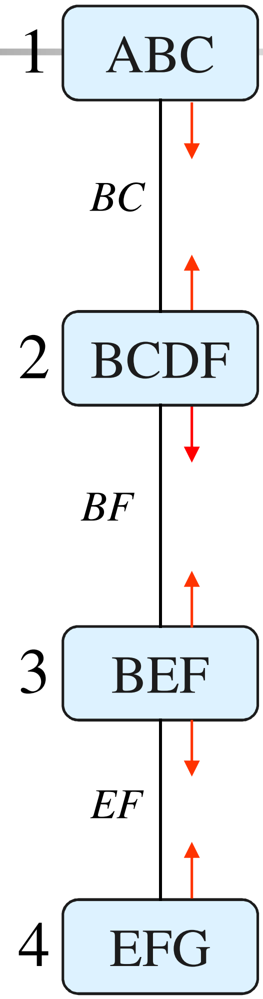
$$h_{(2,3)}(b,f) = \sum_{c,d} p(d \mid b) \cdot h^{1}_{(1,2)}(b,c) \cdot p(f \mid c,d)$$





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

Tree-Width & Separator

The *width* (also called tree-width) of a tree-decomposition $< T, \chi, \psi >$ is $\max_{v \in V} |\chi(v)|$, and its *hyper-width* is $\max_{v \in V} |\psi(v)|$. Given two adjacent vertices u and v of a tree-decomposition, a separator of u and v is defined as $sep(u, v) = \chi(u) \cap \chi(v)$.

Finding tree-decompositions

- Good Join-trees using triangulation
- Tree-width can be generated using induced-width ordering heuristics

CTE - properties

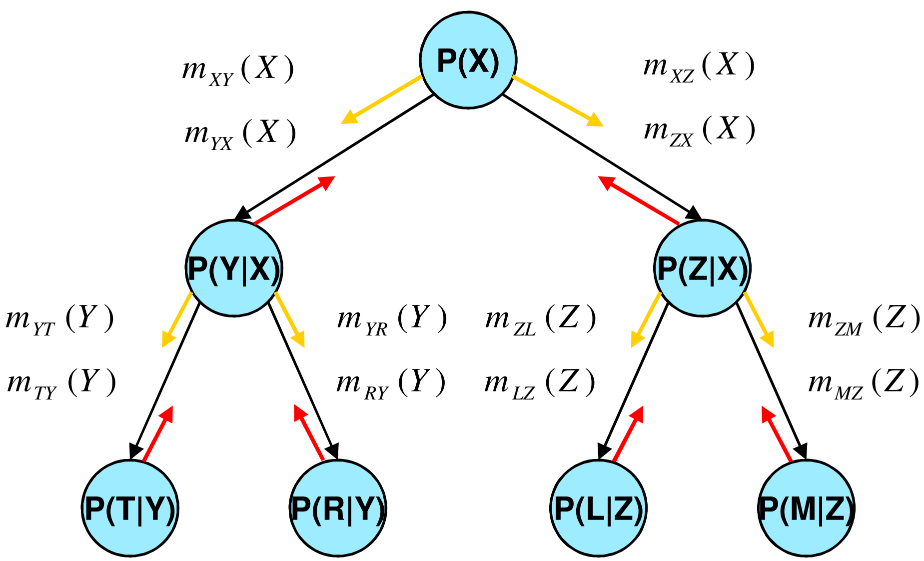
- Correctness and completeness: Algorithm CTE is correct, i.e. it computes the exact joint probability of a single variable and the evidence.

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

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

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

Inference on trees is easy and distributed



$$m_{MZ}(Z) = \sum_M P(M \mid Z)$$

$$m_{LZ}(Z) = \sum_L P(L \mid Z)$$

$$m_{ZX}(X) = \sum_Z P(Z \mid X) \cdot m_{MZ}(Z) \cdot m_{LZ}(Z)$$

$$m_{XZ}(X) = P(X) \cdot m_{YX}(X)$$

$$m_{ZL}(Z) = \sum_X P(Z \mid X) \cdot m_{XZ}(X) \cdot m_{MZ}(Z)$$

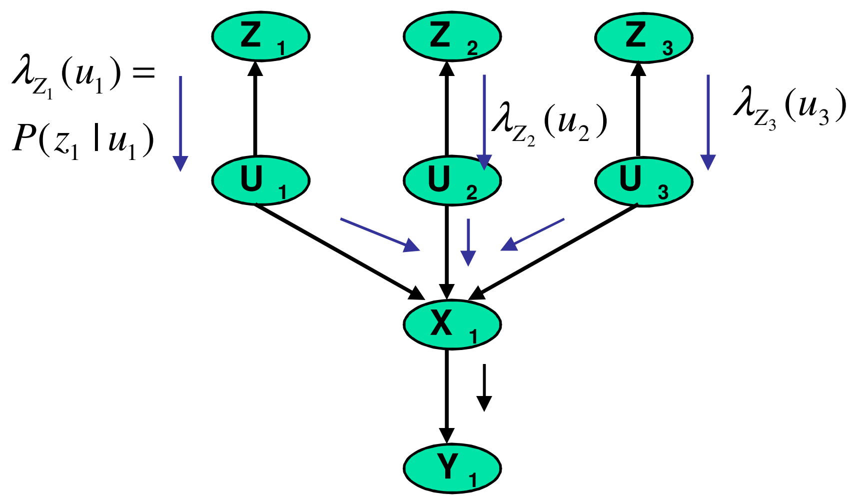
$$m_{ZM}(Z) = \sum_X P(Z \mid X) \cdot m_{XZ}(X) \cdot m_{LZ}(Z)$$

Belief updating = sum-prod

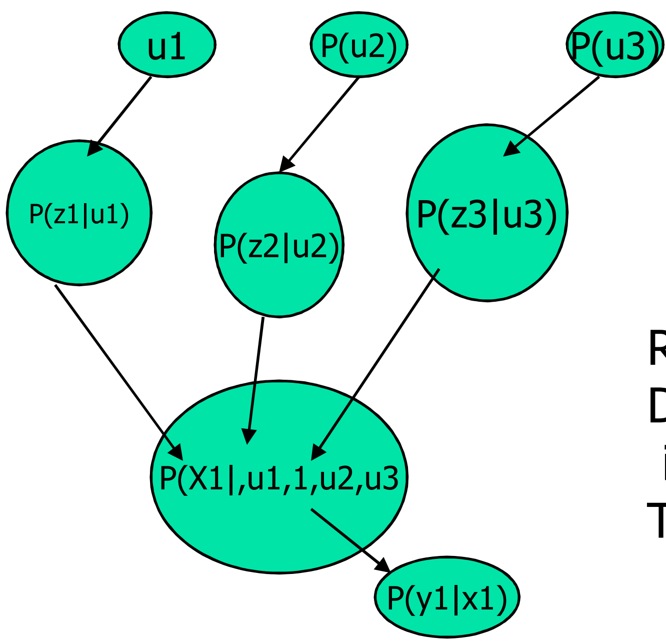
Inference is time and space linear on trees

Pearl's Belief Propagation

A polytree: a tree with
Larger families



A polytree decomposition

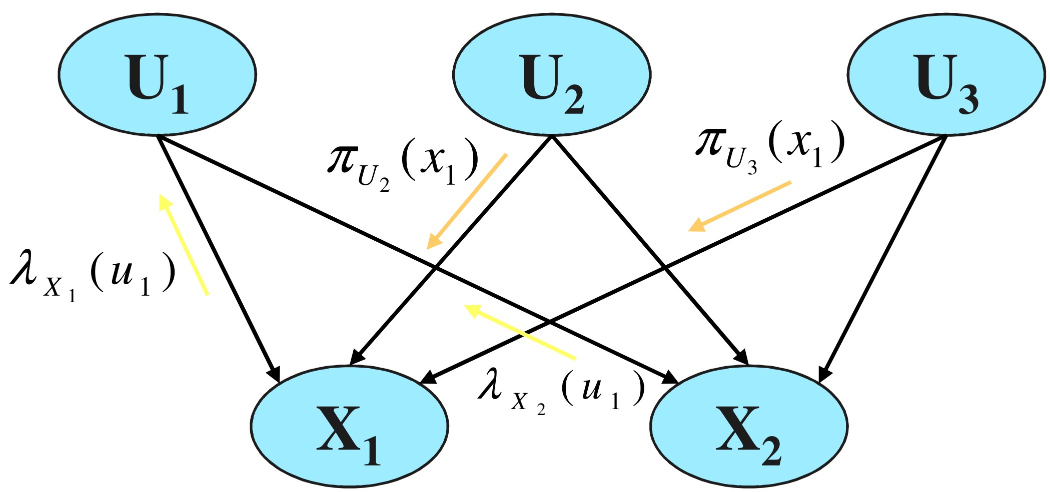


Running CTE = running Pearl's BP over the dual graph
Dual-graph: nodes are cpts, arcs connect non-empty
 intersections.
Time and space linear propagatin

Iterative Belief Propagation – IBP (Loopy belief propagation)

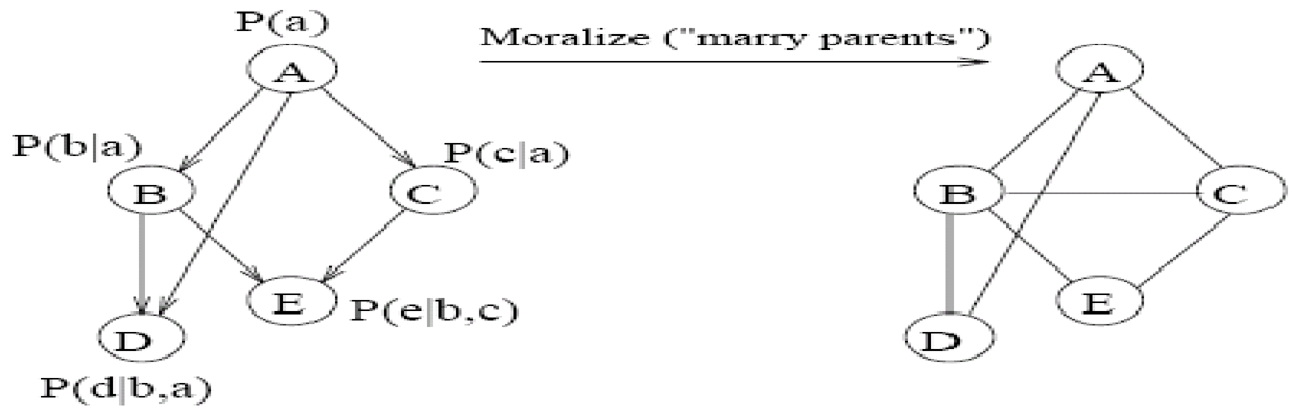
- Belief propagation is exact for poly-trees
- IBP - applying BP iteratively to cyclic networks

One step:
update BEL(U_1)



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

Finding the MPE
(An optimization task)



Ordering: $a, \; b, \; c, \; d, \; e$

$m = \max_{a,b,c,d,e=0} P(a,b,c,d,e) =$

$= \max_a P(a) \max_b P(b|a) \max_c P(c|a) \max_d P(d|b,a)$
$\max_{e=0} P(e|b,c)$

Ordering: $a, \; e, \; d, \; c, \; b$

$m = \max_{a,e=0,d,c,b} P(a,b,c,d,e)$

$m = \max_a P(a) \max_e \max_d \cdot$

$\max_c P(c|a) \max_b P(b|a) P(d|a,b) P(e|b,c)$

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

Algorithm *elim-mpe* (Dechter 1996)

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

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

$\max_b \prod$ ← Elimination operator

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

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

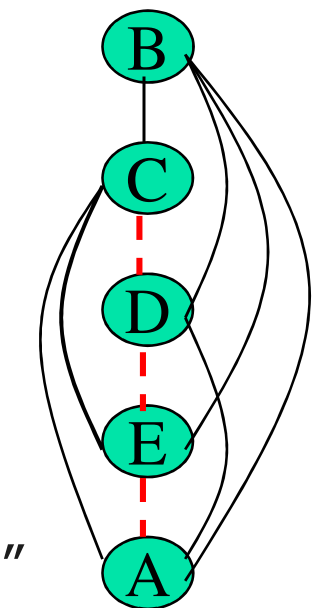
bucket D: $h^C(a, d, e)$

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')

Elim-mpe

Input: A belief network $\{P_1, ..., P_n\}$; d; e.
Output: mpe

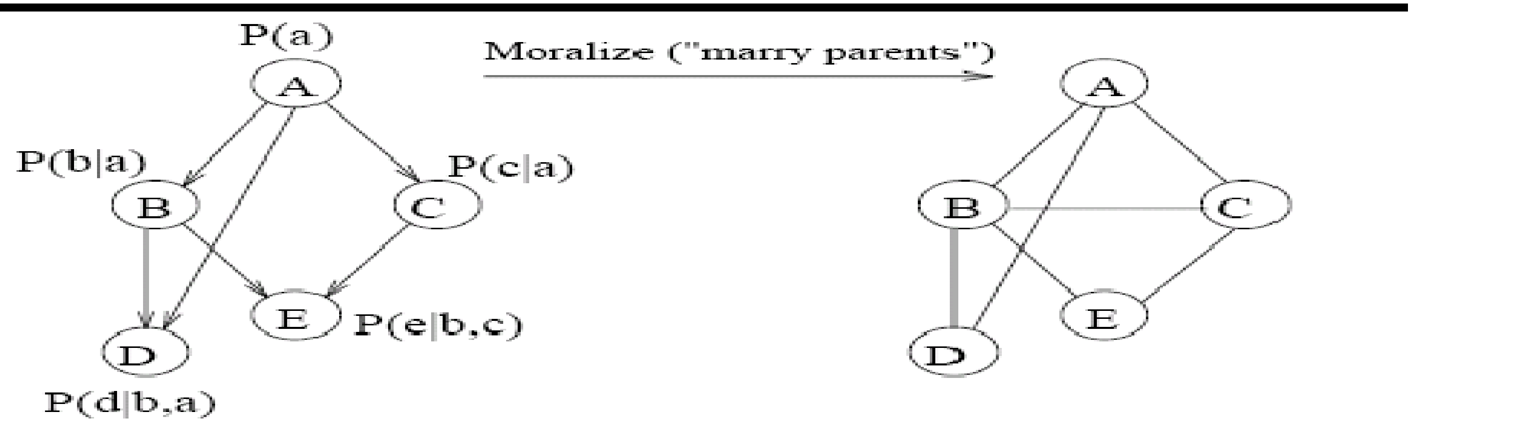
1. **Initialize:**

2. **Process buckets:** for $p = n$ to 1 do
   for matrices $h_1, h_2, ..., h_j$ in $bucket_p$ do
   - **If** (observed variable) assign $X_p = x_p$ to each h_i and put in buckets.
   - **Else,** (multiply and maximize)
     $h_p = \max_{X_p} \Pi_{i=1}^{j} h_i$.
     $x_p^{opt} = argmax_{X_p} h_p$.
     Add h_p to its bucket.
3. **Forward:** Assign values in ordering d

Theorem: Elim-mpe finds the value of the most probable tuple and a corresponding tuple.

Finding the MAP

(An optimization task)



Variables A and B are the hypothesis variables.

Ordering: $a,\ b,\ c,\ d,\ e$

$max_{a,b}P(a,b,e=0) = \max_{a,b}\sum_{c,d,e=0} P(a,b,c,d,e)$
$= \max_a P(a) \max_b P(b|a) \sum_c P(c|a) \sum_d P(d|b,a)$
$\sum_{e=0} P(e|b,c)$

Ordering: $a,\ e,\ d,\ c,\ b$ .... illegal ordering

$\max_{a,b} P(a,e,e=0) = \max_{a,b} \sum_P (a,b,c,d,e)$
$\max_{a,b} P(a,b,e=0) = \max_a P(a) \max_b P(b|a) \sum_d \cdot$
$\max_c P(c|a) P(d|a,b) P(e=0|b,c)$

Elim-map

Maximum aposteriori hypothesis (MAP):
Given $A = \{A_1, ...A_k\} \subseteq X$, find $a^o = ({a^o}_1, ...{a^o}_k)$ s.t. $p(a^o) = \max_{\bar{a}_k} \sum_{x_{X-A}} \Pi_{i=1}^{n} P(x_i|x_{pa_i}, e)$.

Input: A belief network and hypothesis $A = \{A_1, ..., A_k\}$, d, e.
Output: An map.

1. **Initialize:**
2. **Process buckets :** for $p = n$ **to** 1 **do**
   for matrices $\beta_1, \beta_2, ..., \beta_j$ in $bucket_p$ do
   - **If** observed variable, assign $X_p = x_p$.
   - **Else**, (multiply and sum or max)
     $\beta_p = \sum_{X_p} \Pi_{i=1}^{j} \beta_i$,
     $(X_p \in A)$ $\beta_p = \max_{X_p} \Pi_{i=1}^{j} \beta_i$
     $a^0 = argmax_{X_p} \beta_p$.
     Add β_p to its bucket.
3. **Forward:** Assign values to A.

Variable ordering is restricted: max-buckets should preceede (processed after) summation buckets.

Complexity of bucket elimination

Theorem

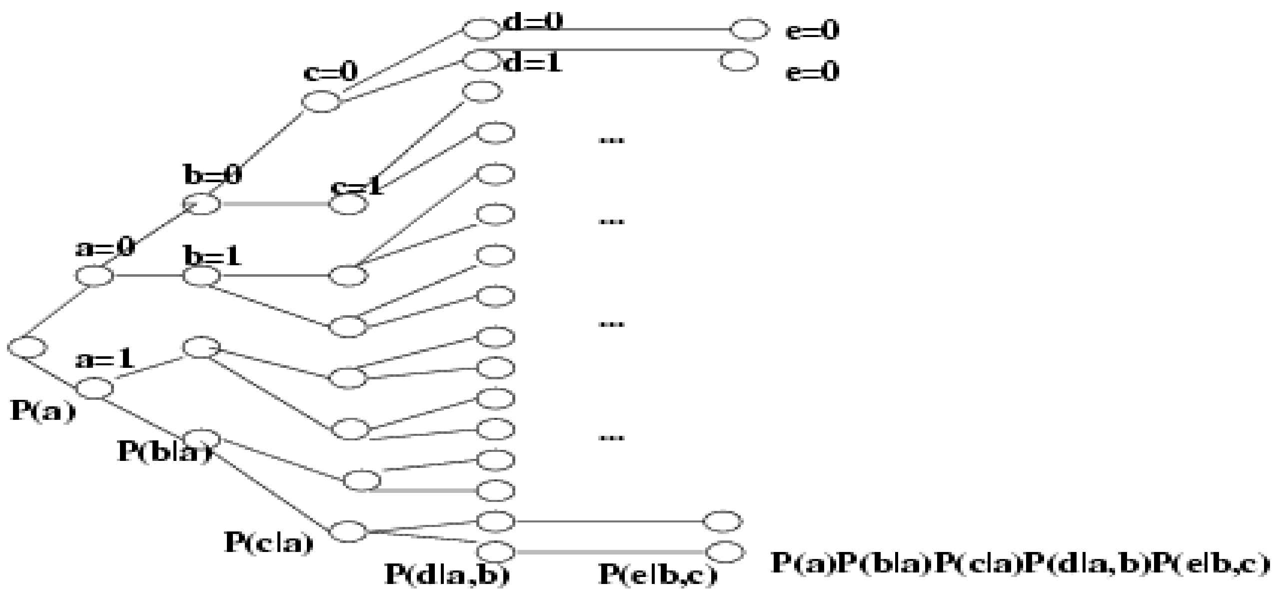
Given a belief network having n variables, observations e, the complexity of elim-mpe, elim-bel, elim-map along d, is time and space

$$O(n \cdot exp(w*(d))$$

where $w*(d)$ is the induced width of the moral graph whose edges connecting evidence to earlier nodes, were deleted.

Conditioning generates the probability tree

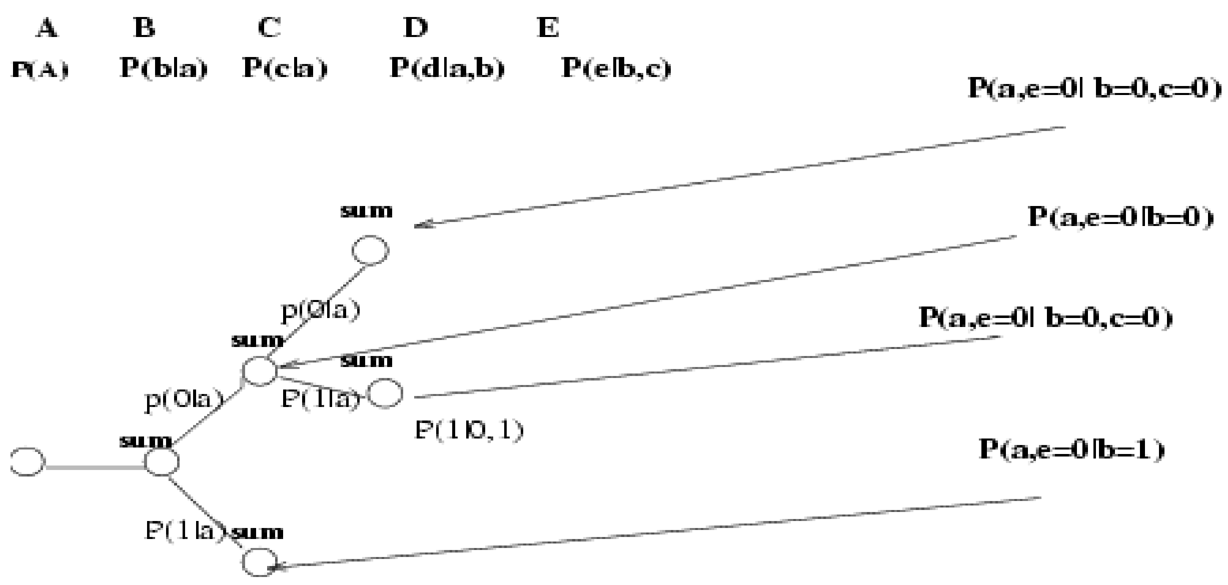
$$P(a,e=0) = P(a)\sum_{b} P(b \mid a)\sum_{c} P(c \mid a)\sum_{b} P(d \mid a,b)\sum_{e=0} P(e \mid b,c)$$



Complexity of conditioning: exponential time, linear space

Conditioning+Elimination

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

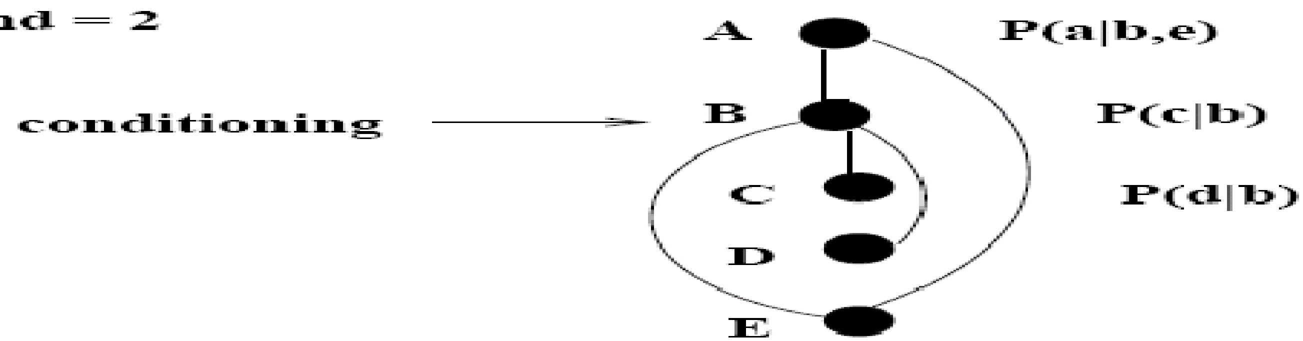


Idea: conditioning until w^* of a (sub)problem gets small

Conditioning + Elimination
Trading space for time

- Algorithm *elim-cond(b)*, b bounds width:
When $b > width$, apply conditioning.
- $b = 0$ is full conditioning,
- $b = w^*$ is pure bucket elimination
- $b = 1$ is the cycle-cutset method.
- Time $exp(b + |cond(b)|)$, space $exp(b)$

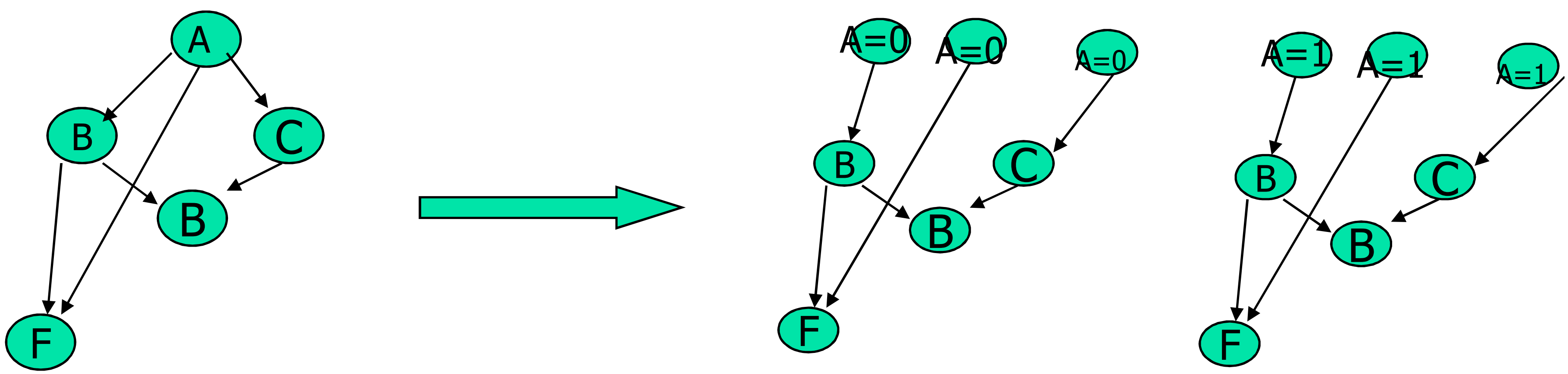
bound = 2



Loop-cutset decomposition

- You condition until you get a polytree

A=0 A=1



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

W-cutset algorithms

- Elim-cond-bel:
- Identify a w-cutset, C_w, of the network
- For each assignment to the cutset solve by CTE the conditioned subproblems
- Aggregate the solutions over all cutset assignments.
- Time complexity: exp(|C_w|+w)
- Space complexity: exp(w)

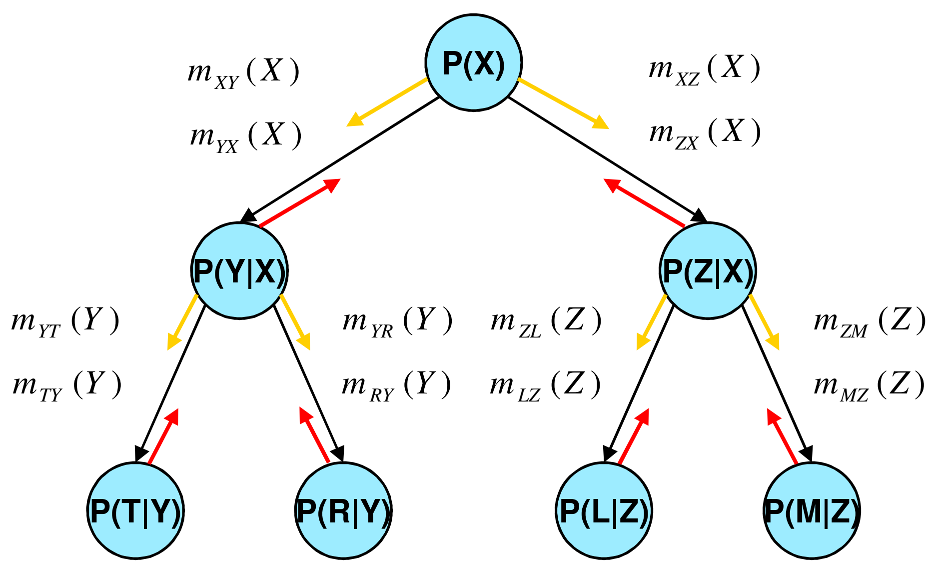
All algorithms genaralize to any graphical models

- Through general operations of combination and marginalization
- General BE, BTE, CTE, BP
- Applicable to Markov networks, to constraint optimization, to counting number of solutions in a SAT formula, etc.

Tree-solving

Belief updating (sum-prod)

CSP – consistency (projection-join)



MPE (max-prod)

#CSP (sum-prod)

Inference is time and space linear on trees

Graphical Models

- A graphical model **(X,D,F)**:
  - **X** = $\{X_1, \ldots X_n\}$ variables
  - **D** = $\{D_1, \ldots D_n\}$ domains
  - **F** = $\{f_1, \ldots, f_r\}$ functions

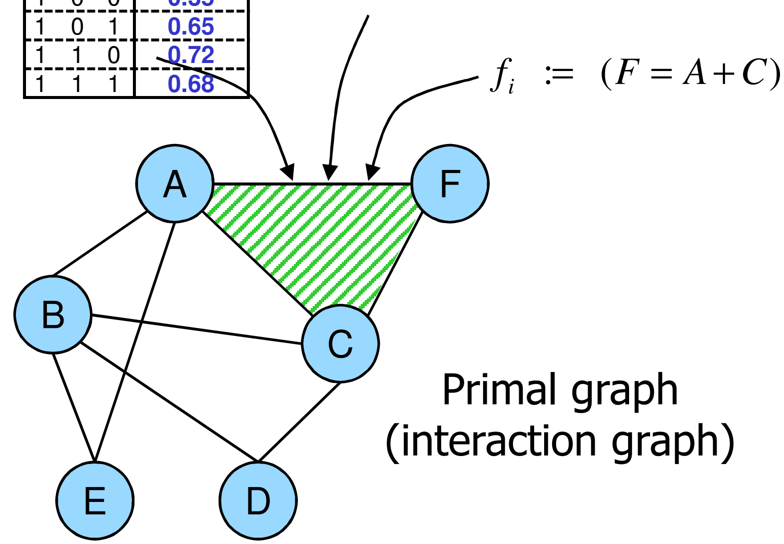
    (constraints, CPTS, CNFs …)

Conditional Probability Table (CPT)

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



Primal graph (interaction graph)

- Operators:
  - combination
  - elimination (projection)

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

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

Algorithm bucket-tree elimination (BTE)

Input: A problem $P =< X, D, F, \bigotimes, \Downarrow, \{x_1, ..., x_n\} >$, ordering d.

Output: Augmented buckets containing the original functions and all the π and λ functions received from neighbors in the bucket-tree. A solution to P computed from augmented buckets.

0. Pre-processing:
Place each function in the latest bucket, along d, that mentions a variable in its scope. Connect two buckets B_x and B_y if variable y is the lastest earlier neighbor of x in the induced graph G_d.

1. Bottom-up phase: λ messages (BE)
For $i = n$ to 1, process bucket B_{x_i}:
Let $\lambda_1, ...\lambda_j$ be all the functions in B_{x_i} at the time B_{x_i} is processed, including original functions of P. The message $\lambda_{x_i}^y$ sent from x_i to its parent y, is computed by

$$\lambda_{x_i}^y(sep(x_i, y)) = \Downarrow_{sep(x_i,y)} \bigotimes_{i=1}^{j} \lambda_i$$

where $sep(x_i, y)$ is the separator of x_i and y.

2. Top-down phase: π messages

For $i = 1$ to n, process bucket B_{x_i}:

Let $\lambda_1, ...\lambda_j$ be all the functions in B_{x_i} at the time B_{x_i} is processed, including the original functions of P. B_{x_i} takes the π message received from its parent y, $\pi_y^{x_i}$, and computes a message $\pi_{x_i}^{z_j}$ for each child bucket z_j by

$$\pi_{x_i}^{z_j}(sep(x_i, z_j)) = \Downarrow_{sep(x_i,z_j)} \pi_y^{x_i} \bigotimes (\bigotimes_i \lambda_i / \lambda_{z_j}^{x_i})$$

3. Compute solution: In each augmented bucket compute: $\Downarrow_{x_i} \bigotimes_{f \in bucket_i} f$,

Cluster-Tree Decomposition

Let $P=<X,D,F,\otimes,\Downarrow,\{Z_i\}>$ be an automated reasoning problem. A tree decomposition is $<T,\chi,\psi>$, such that

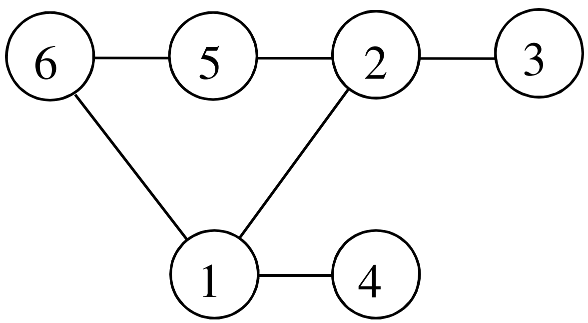
- $T=(V,E)$ is a tree
- χ associates a set of variables $\chi(v)\subseteq X$ with each node
- ψ associates a set of functions $\psi(v)\subseteq F$ with each node

such that

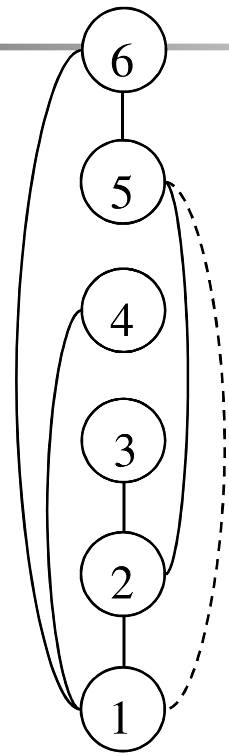
- $\forall f_i \in F$, there is exactly one v such that $f_i \in \psi(v)$ and scope$(f_i)\subseteq\chi(v)$.
- $\forall x\in X$, the set $\{v\in V|x\subseteq\chi(v)\}$ induces a connected subtree.
- $\forall i\ Z_i \subseteq\chi(v)$ for some $v\in V$.

Example: Cluster-Tree

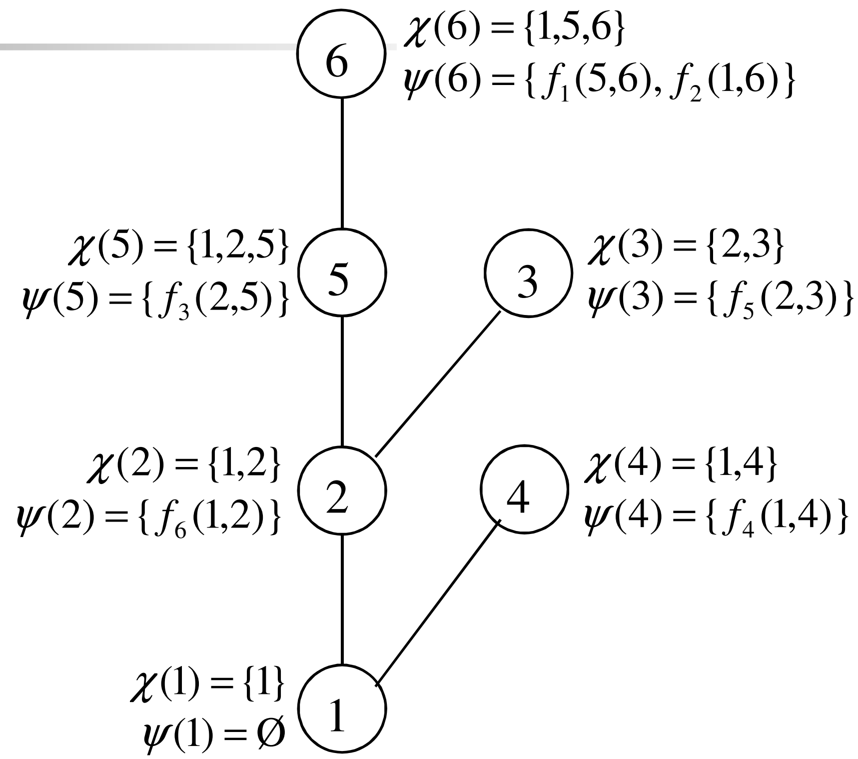
$C_{ij} = X_i \neq X_j$



(a)



(b)



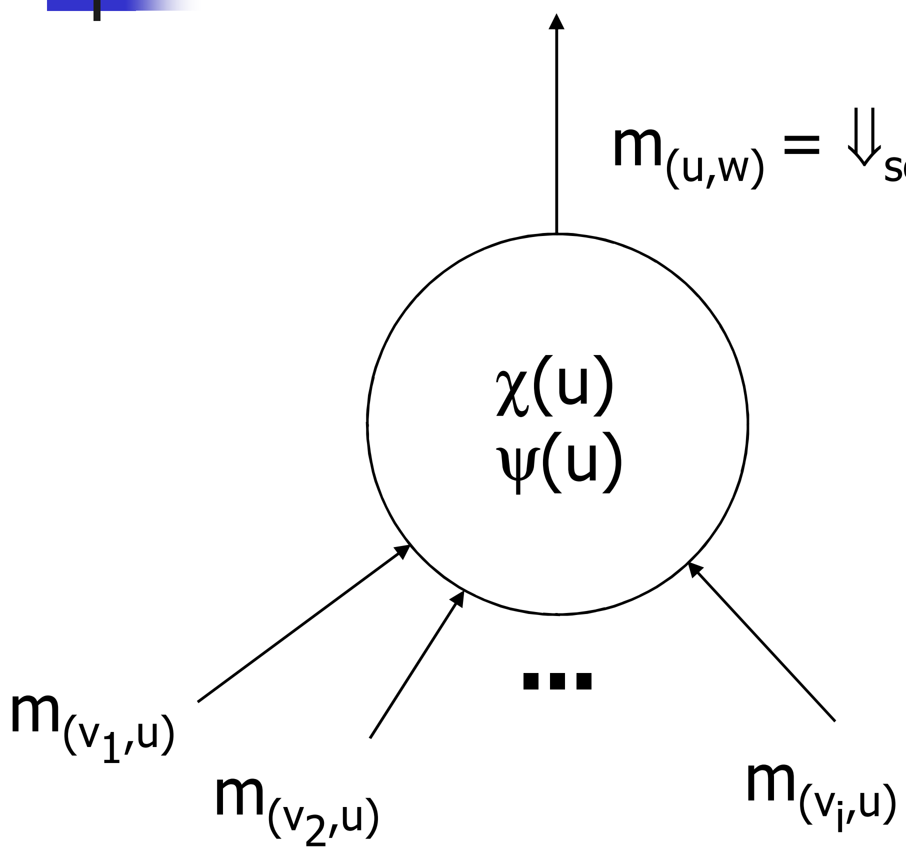
(c)

Tree-width = 3

sep(5,6) = {1, 5}

Cluster-Tree Elimination (CTE)

$$m_{(u,w)} = \Downarrow_{sep(u,w)} [\otimes_j \{m_{(v_j,u)}\} \otimes \psi(u)]$$



Example: Cluster-Tree Elimination