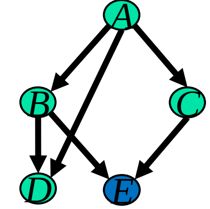
Inference for probabilistic networks (continued)

- Bucket elimination
  - Belief-updating, P(e), partition function
  - Marginals, probability of evidence
  - The impact of evidence
  - for MPE (→MAP)
  - for MAP (→ Marginal Map)
- Induced-Width (Dechter 3.4,3.5)

The Impact of Evidence?

Algorithm *BE-bel*



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

$\sum_b \prod$ ← *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: $\lambda^C(a,d,e)$

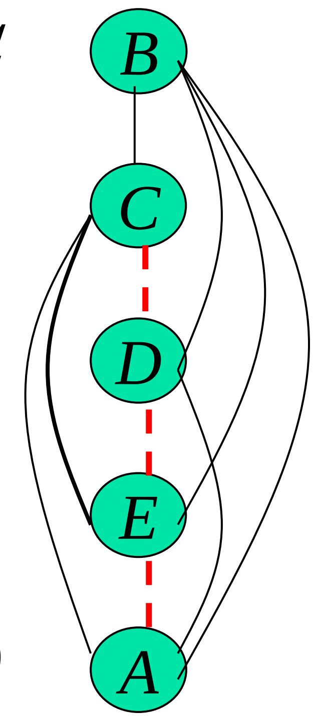
bucket E: $e=0$ $\lambda^D(a,e)$

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

$P(e=0)$

$P(a|e=0)$

B=1

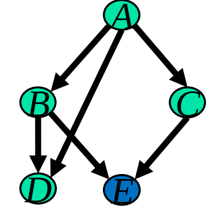


W=4*

"induced width"

(max clique size)

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



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

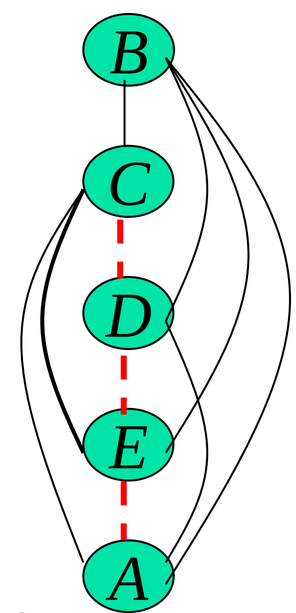
bucket D: $h^C(a,d,e)$

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')

Inference for probabilistic networks

- Bucket elimination
  - Belief-updating, P(e), partition function
  - Marginals, probability of evidence
  - The impact of evidence
  - for MPE (→MAP)
  - for MAP (→ Marginal Map)
- Induced-Width (Dechter 3.4,3.5)

Finding a Small Induced-Width

- NP-complete
- A tree has induced-width of ?
- Greedy algorithms:
  - Min width
  - Min induced-width
  - Max-cardinality and chordal graphs
  - Fill-in (thought as the best)
- Anytime algorithms
  - Search-based [Gogate & Dechter 2003]
  - Stochastic (CVO) [Kask, Gelfand & Dechter 2010]

Finding a Small Induced-Width

- NP-complete
- A tree has induced-width of ?
- Greedy algorithms:
  - Min width
  - Min induced-width
  - Max-cardinality and chordal graphs
  - Fill-in (thought as the best)
- Anytime algorithms
  - Search-based [Gogate & Dechter 2003]
  - Stochastic (CVO) [Kask, Gelfand & Dechter 2010]

Finding a Small Induced-Width

- NP-complete
- A tree has induced-width of ?
- Greedy algorithms:
  - Min width
  - Min induced-width
  - Max-cardinality and chordal graphs
  - Fill-in (thought as the best)
- Anytime algorithms
  - Search-based [Gogate & Dechter 2003]
  - Stochastic (CVO) [Kask, Gelfand & Dechter 2010]

Min-width Ordering

MIN-WIDTH (MW)

input: a graph $G = (V, E)$, $V = \{v_1, ..., v_n\}$
output: A min-width ordering of the nodes $d = (v_1, ..., v_n)$.
1. **for** $j = n$ to 1 by -1 do
2. $r \leftarrow$ a node in G with smallest degree.
3. put r in position j and $G \leftarrow G - r$.
   (Delete from V node r and from E all its adjacent edges)
4. **endfor**

Proposition: *algorithm min-width finds a min-width ordering of a graph*

What is the Complexity of MW?

O(e)

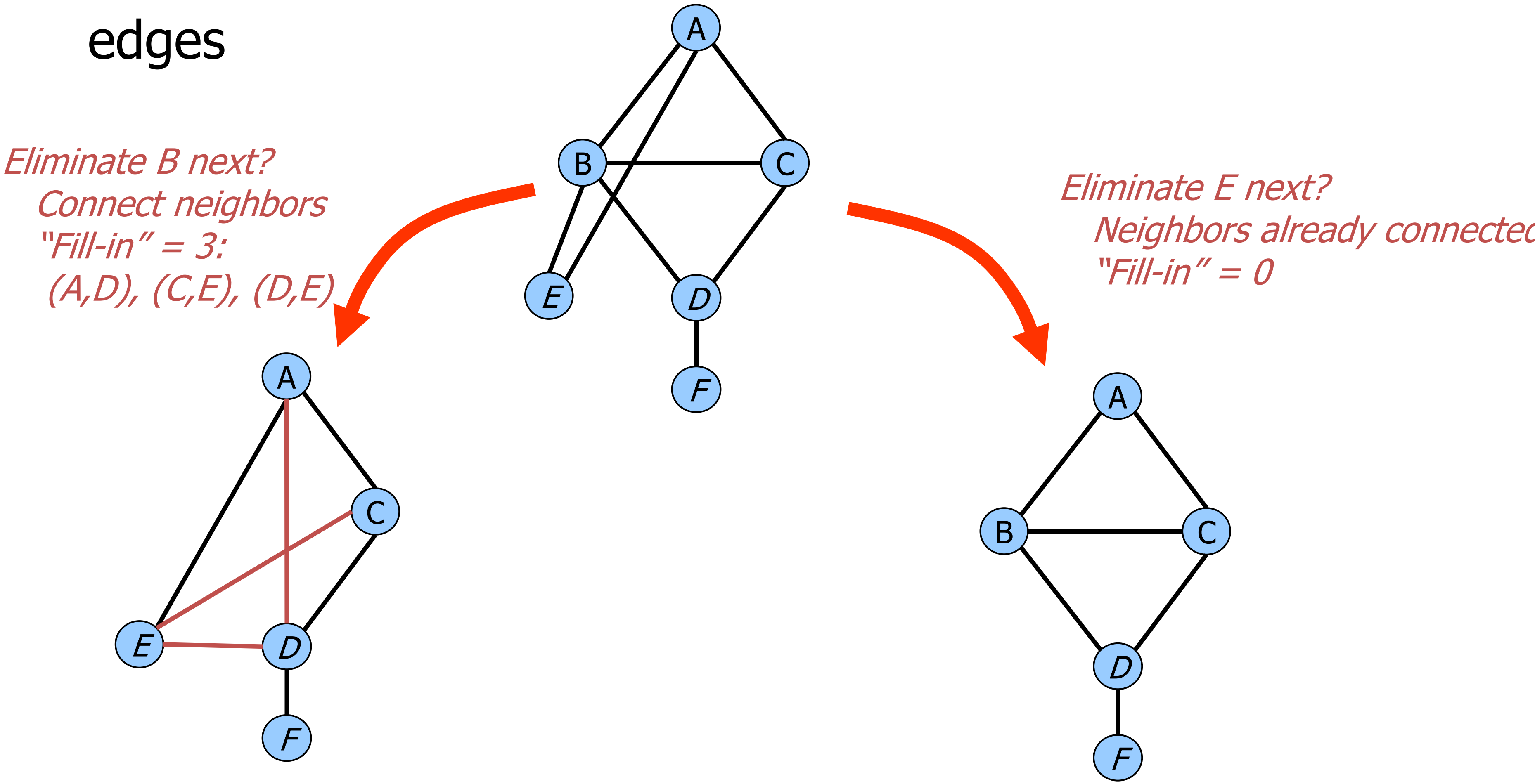
Greedy Orderings Heuristics

- Min-induced-width
  - From last to first, pick a node with smallest width, then connect parent and remove
- Min-Fill
  - From last to first, pick a node with smallest fill-edges

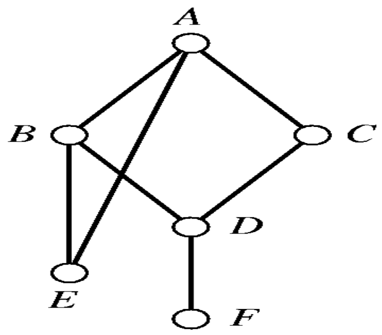
Complexity? $O(n^3)$

Min-Fill Heuristic

- Select the variable that creates the fewest "fill-in" edges

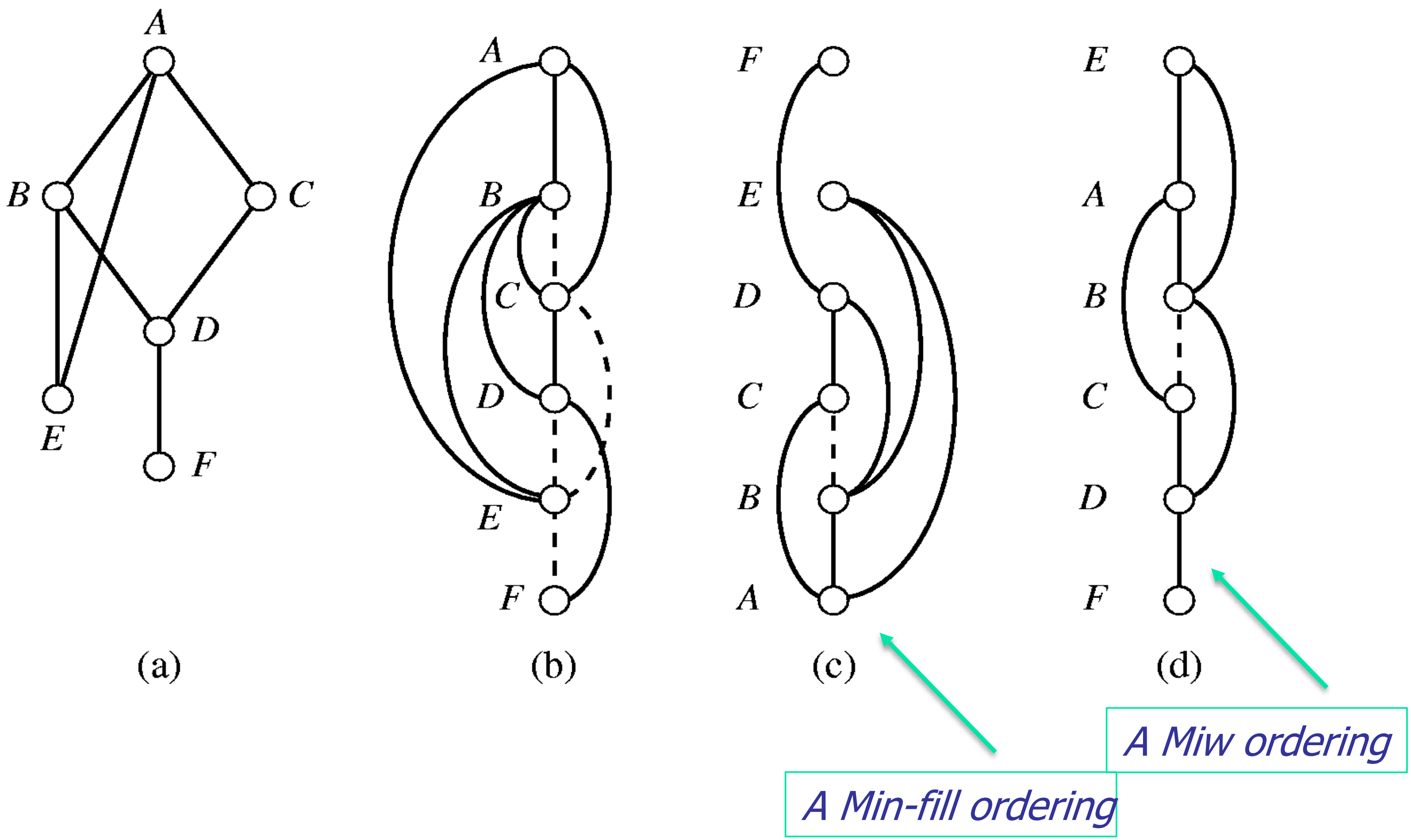


Example



(a)

Different Induced-Graphs



Which Greedy Algorithm is Best?

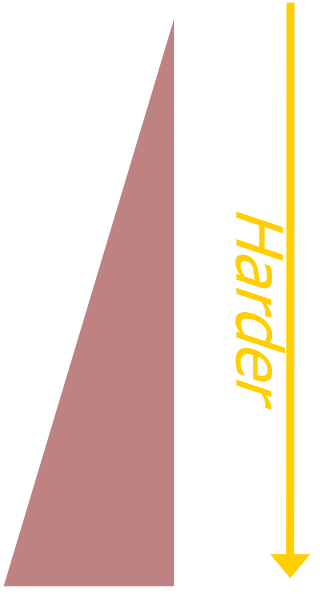
- Min-Fill, prefers a node who adds the least number of fill-in arcs.
- Empirically, fill-in is the best among the greedy algorithms (MW,MIW,MF,MC)
- Complexity of greedy orderings?
- MW is O(e), MIW: O(n^3), MF O(n^3), MC is O(e+n) (MC: read on your own)

Inference for probabilistic networks

- Bucket elimination (Dechter chapter 4)
  - Belief-updating, P(e), partition function
  - Marginals, probability of evidence
  - The impact of evidence
  - for MPE (→MAP)
  - for MAP (→ Marginal Map)
  - Influence diagrams ?
- Induced-Width (Dechter, Chapter 3.4)

Marginal Map

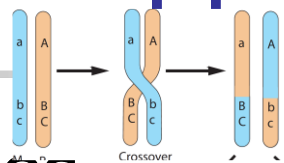
Max-Inference	$f(\mathbf{x}^*) = \max_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$
Sum-Inference	$Z = \sum_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$
Mixed-Inference	$f(\mathbf{x}_M^*) = \max_{\mathbf{x}_\mathrm{M}} \sum_{\mathbf{x}_S} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$



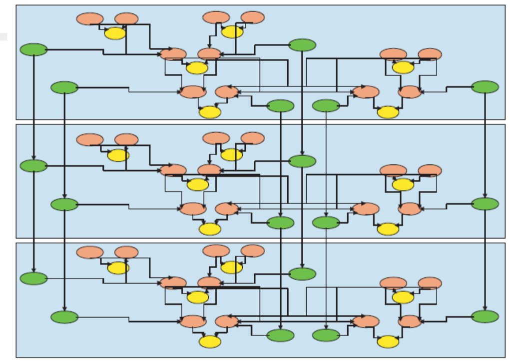
- **NP-hard**: exponentially many terms

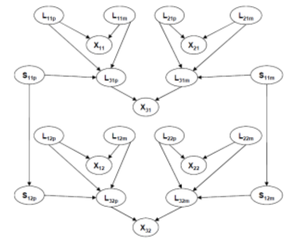
Example for MMAP Applications

- Haplotype in Family pedigrees



6 people, 3 markers





- Coding networks

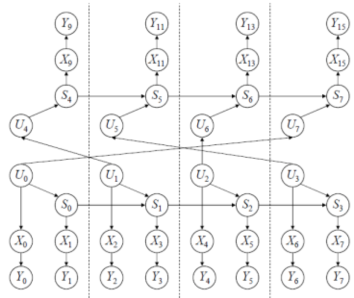
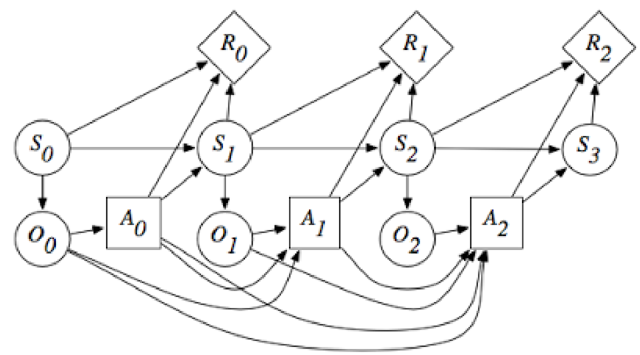
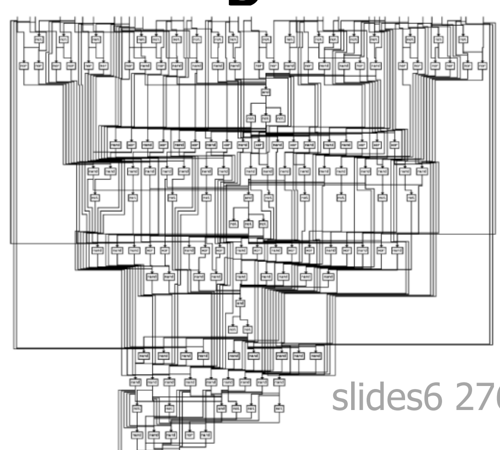


Figure 5.24: A Bayesian network for a turbo code.

- Probabilistic planning

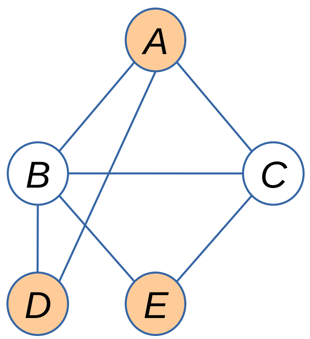


- Diagnosis



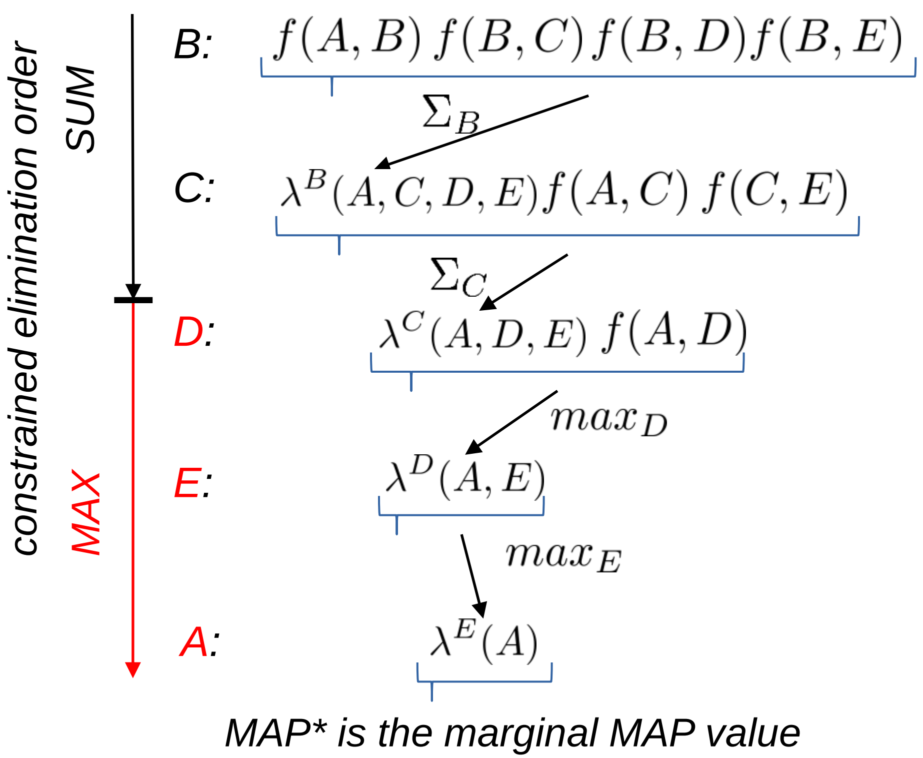
Bucket Elimination for MMAP

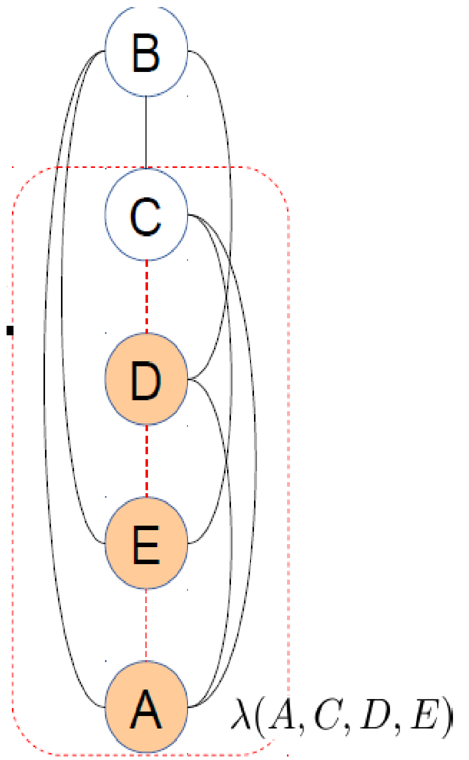
Bucket Elimination



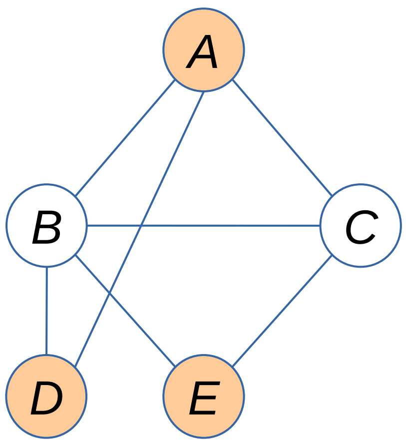
$$\mathbf{X}_M = \{A, D, E\}$$
$$\mathbf{X}_S = \{B, C\}$$

$$\max_{\mathbf{X}_M} \sum_{\mathbf{X}_S} P(\mathbf{X})$$

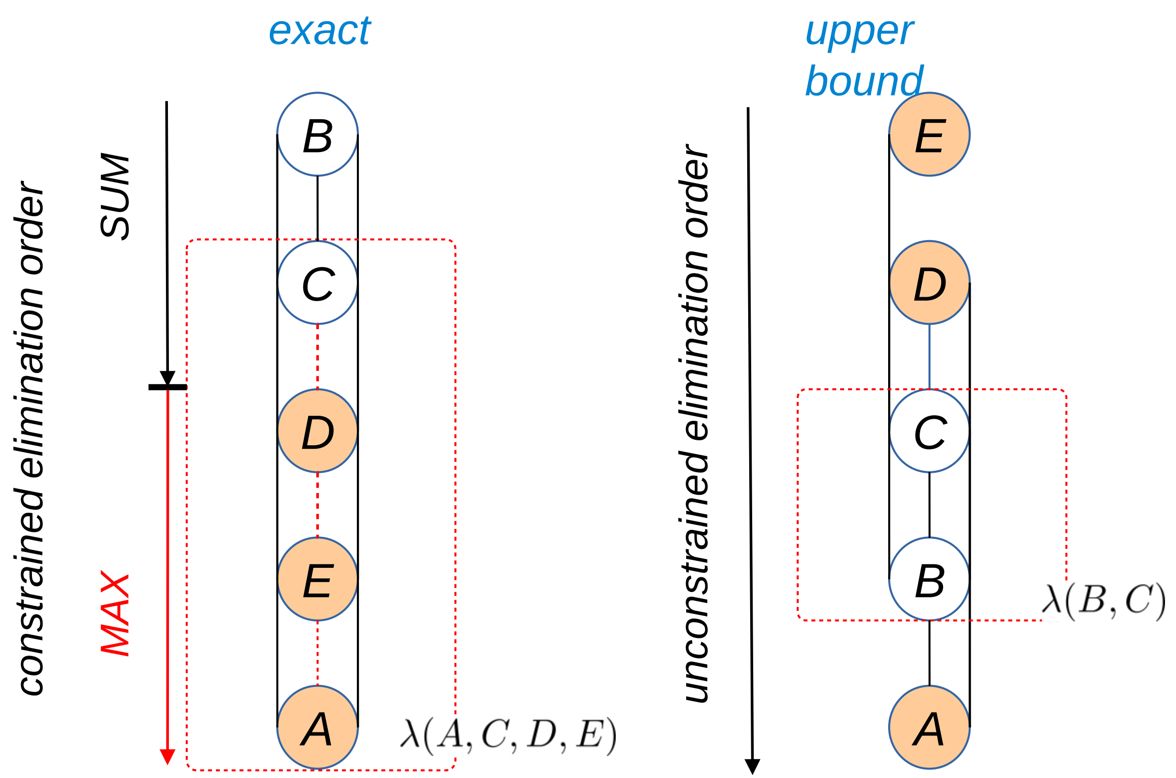




Why is MMAP harder?



$\mathbf{X}_M = \{A, D, E\}$
$\mathbf{X}_S = \{B, C\}$



$w^* = 4$ $w^* = 2$

In practice, constrained induced is much larger!

$$\max_X \sum_Y \phi \leq \sum_Y \max_X \phi$$

(Park & Darwiche, 2003)
(Yuan & Hansen, 2009)

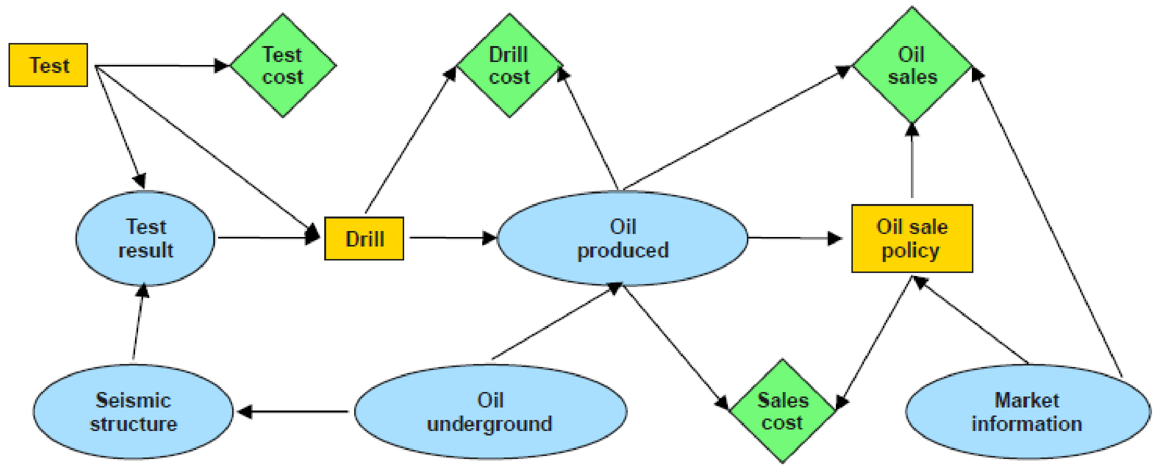
Inference for probabilistic networks

- Bucket elimination (Dechter chapter 4)
  - Belief-updating, P(e), partition function
  - Marginals, probability of evidence
  - The impact of evidence
  - for MPE (→MAP)
  - for MAP (→ Marginal Map)
- Induced-Width (Dechter, Chapter 3.4)
- Mixed networks
- Influence diagrams ?

Ex: “oil wildcatter”

e.g., [Raiffa 1968; Shachter 1986]

- Influence diagram:



- Three actions: test, drill, sales policy
- Chance variables:

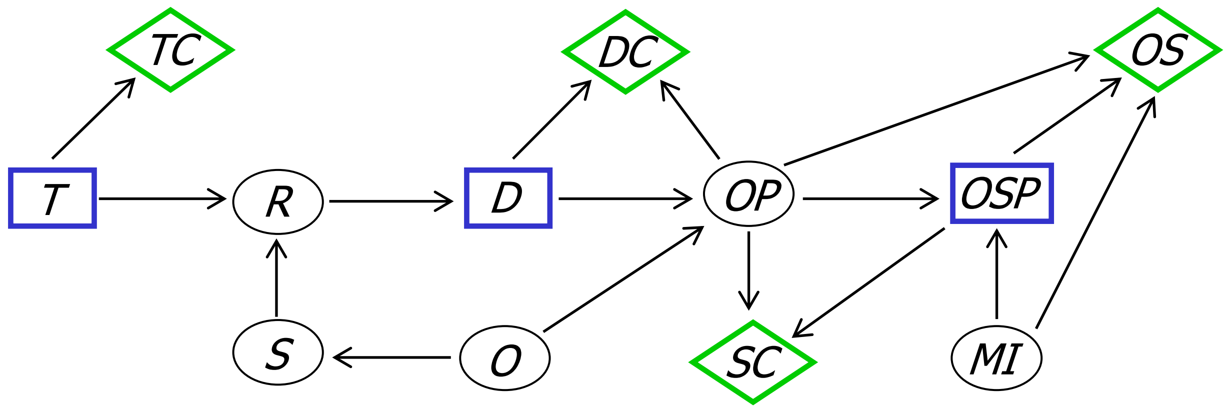
  P(oil) P(seismic|oil) P(result | seismic, test) P(produced | oil, drill) P(market)

- Utilities capture costs of actions, rewards of sale

  Oil sales - Test cost - Drill cost - Sales cost

Influence Diagrams

Influence diagram *ID = (X,D,P,R).*



Chance variables $X = X_1, \ldots, X_n$ ***over domains.***

Decision variables $D = D_1, \ldots, D_m$

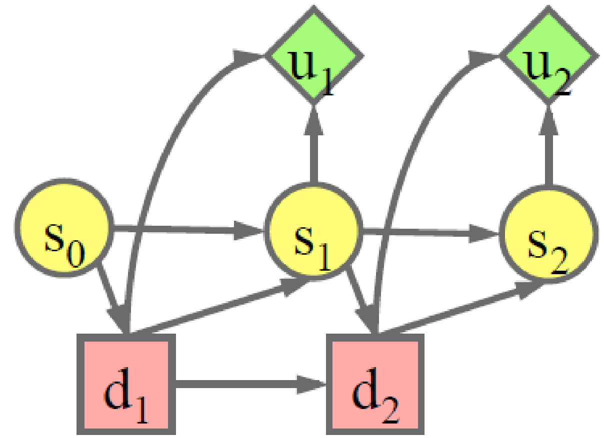
CPT's for chance variables $P_i = P(X_i \mid pa_i), i = 1..n$

Reward components $R = \{r_1, \ldots, r_j\}$

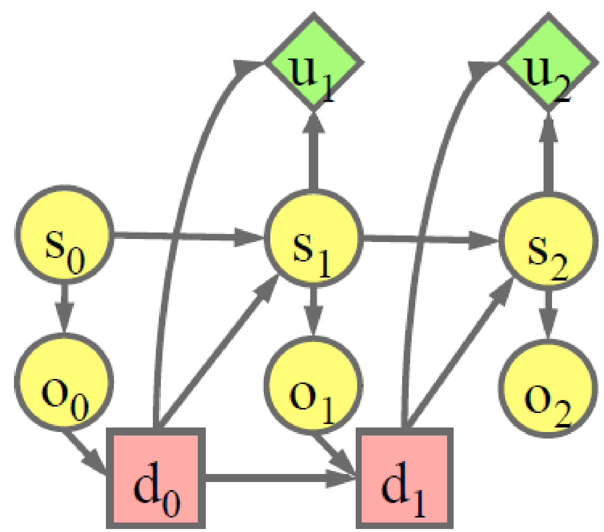
Utility function $u = \sum_i r_i$

Common examples

- Markov decision process
  - Markov chain state sequence
  - Actions “di” influence state transition
  - Rewards based on action, new state
  - Temporally homogeneous



- Partially observable MDP
  - Hidden Markov chain state sequence
  - Generate observations
  - Actions based on observations



Influence Diagrams (continue)

A decision rule *for* D_i *is a mapping:* $\delta_i : \Omega_{pa_{D_i}} \rightarrow \Omega_{D_i}$

where Ω_S *is the cross product of domains in* S.

A policy *is a list of decision rules* $\Delta = (\delta_1, \ldots., \delta_m)$

Task: *Find an optimal policy that maximizes the expected utility.*

$$E = \max_{\Delta=(\delta_1,\ldots,\delta_m)} \sum_{x=(x_1,\ldots,x_n)} \prod_i P_i(x) u(x)$$

General Graphical Models

Definition 2.2 Graphical model. A *graphical model* $\mathcal{M}$ is a 4-tuple, $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \bigotimes \rangle$, where:

1. $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a finite set of variables;
2. $\mathbf{D} = \{D_1, \ldots, D_n\}$ is the set of their respective finite domains of values;
3. $\mathbf{F} = \{f_1, \ldots, f_r\}$ is a set of positive real-valued discrete functions, defined over scopes of variables $\mathcal{S} = \{S_1, ..., S_r\}$, where $\mathbf{S}_i \subseteq \mathbf{X}$. They are called *local* functions.
4. $\bigotimes$ is a *combination* operator (e.g., $\bigotimes \in \{\prod, \sum, \bowtie\}$ (product, sum, join)). The combination operator can also be defined axiomatically as in [Shenoy, 1992], but for the sake of our discussion we can define it explicitly, by enumeration.

The graphical model represents a *global function* whose scope is $\mathbf{X}$ which is the combination of all its functions: $\bigotimes_{i=1}^{r} f_i$.

General Bucket Elimination

Algorithm General bucket elimination (GBE)

Input: $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \otimes \rangle$. $F = \{f_1, ..., f_n\}$ an ordering of the variables, $d = X_1, ..., X_n$; $\mathbf{Y} \subseteq \mathbf{X}$.

Output: A new compiled set of functions from which the query $\Downarrow_Y \otimes_{i=1}^n f_i$ can be derived in linear time.

1. **Initialize:** Generate an ordered partition of the functions into $bucket_1, ..., bucket_n$, where $bucket_i$ contains all the functions whose highest variable in their scope is X_i. An input function in each bucket ψ_i, $\psi_i = \otimes_{i=1}^n f_i$.
2. **Backward:** For $p \leftarrow n$ downto 1, do
for all the functions $\psi_p, \lambda_1, \lambda_2, ..., \lambda_j$ in $bucket_p$, do

- **If** (observed variable) $X_p = x_p$ appears in $bucket_p$, assign $X_p = x_p$ in ψ_p and to each λ_i and put each resulting function in appropriate bucket.
- **else,** (combine and marginalize)
$\lambda_p \leftarrow \Downarrow_{S_p} \psi_p \otimes (\otimes_{i=1}^j \lambda_i)$ and add λ_p to the largest-index variable in $scope(\lambda_p)$.

3. **Return:** all the functions in each bucket.

Theorem 4.23 Correctness and complexity. *Algorithm GBE is sound and complete for its task. Its time and space complexities is exponential in the* $w^*(d) + 1$ *and* $w^*(d)$*, respectively, along the order of processing* d.

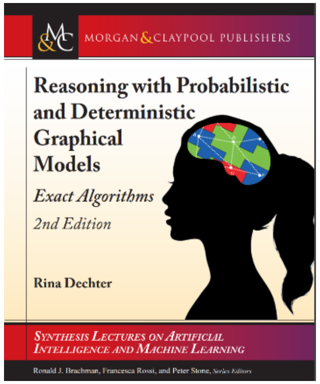
Causal and Probabilistic Reasoning

Slides Set 6:

Exact Inference Algorithms Tree-Decomposition Schemes

Rina Dechter

(Dechter chapter 5, Darwiche chapter 6-7)



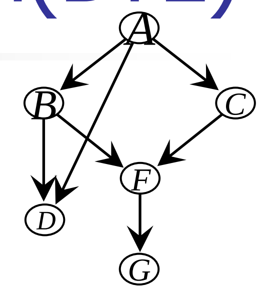
Outline

- From bucket-elimination (BE) to bucket-tree elimination (BTE)
- From BTE to CTE, Acyclic networks, the join-tree algorithm
- Generating join-trees, the treewidth
- Examples of CTE for Bayesian network
- Belief-propagation on acyclic probabilistic networks (poly-trees) and Loopy networks
- Conditioning with elimination (Dechter, 7.1, 7.2)

Outline

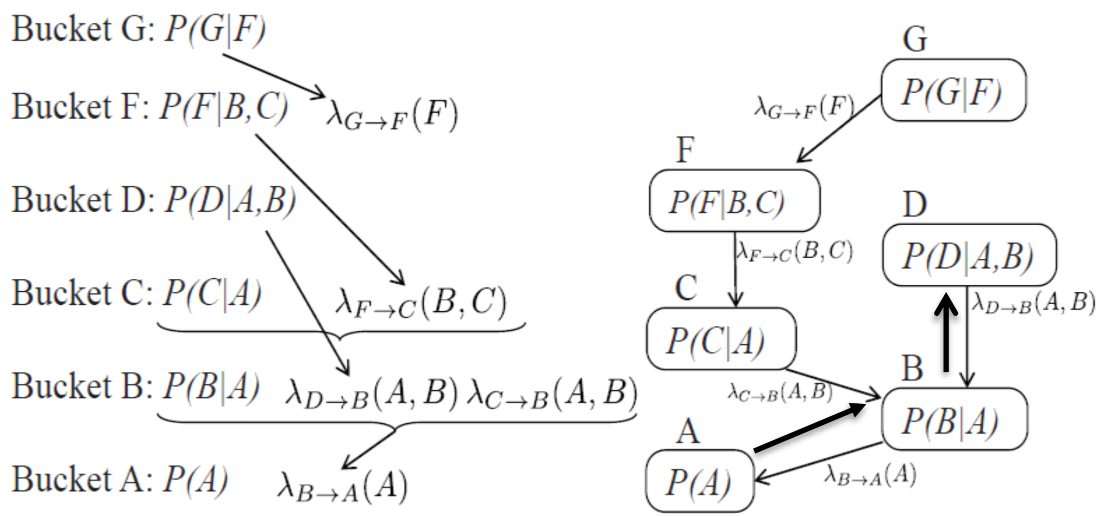
- From bucket-elimination (BE) to bucket-tree elimination (BTE)
- From BTE to CTE, Acyclic networks, the join-tree algorithm
- Generating join-trees, the treewidth
- Examples of CTE for Bayesian network
- Belief-propagation on acyclic probabilistic networks (poly-trees) and Loopy networks

From BE to Bucket-Tree Elimination(BTE)



First, observe the BE operates on a tree.

Second, What if we want the marginal on D?



P(D)?

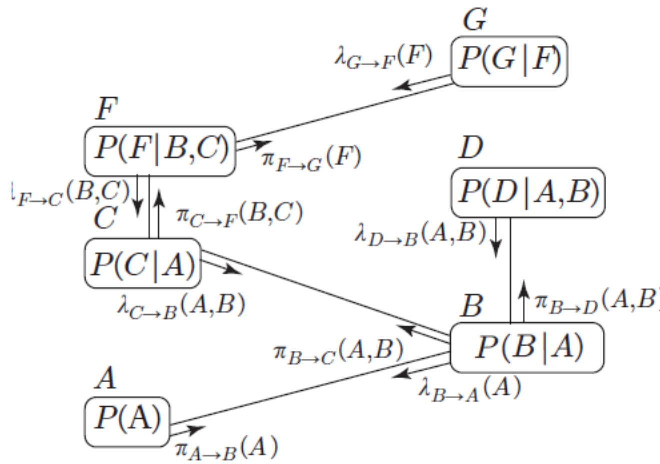
$$\pi_{A\to B}(a) = P(A),$$

$$\pi_{B\to D}(a,b) = p(b|a)\cdot \pi_{A\to B}(a)\cdot \lambda_{C\to B}(b)$$

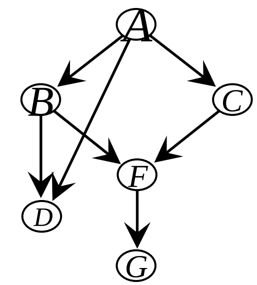
$$bel(d) = \alpha \sum_{a,b} P(d|a,b)\cdot \pi_{B\to D}(a,b).$$

BTE: Allows Messages Both Ways

Initial buckets
+ messages

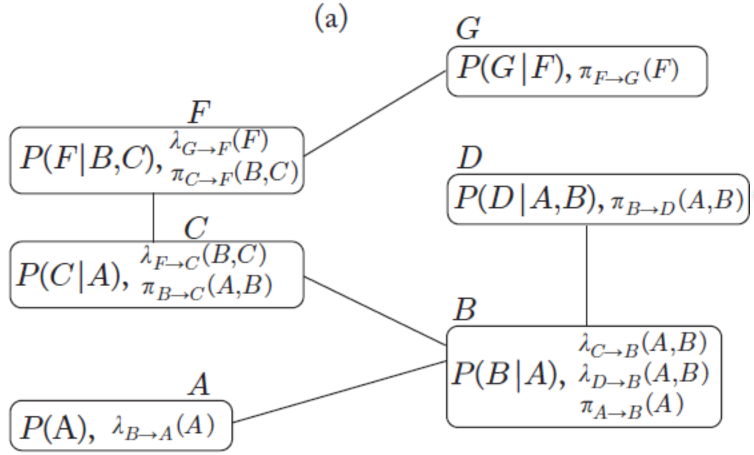


(a)



Output buckets

$$P(F) = \sum_{b,c} P(F|b,c)\pi_{C\to F}(b,c)\,\lambda_{G\to F}(F)$$



(b)

$$P(D) = \sum_{a,b} P(D|a,b)\,\pi_{B\to D}(a,b)$$

BTE

Theorem: *When BTE terminates The product of functions in each bucket is the beliefs of the variables joint with the evidence.*

elim(i,j) = scope(B_i)– scope(B_j)

ALGORITHM BUCKET-TREE ELIMINATION (BTE)

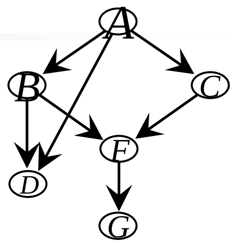
Input: A problem $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \prod, \sum \rangle$, ordering d.
$X = \{X_1, ..., X_n\}$ and $F = \{f_1, ..., f_r\}$
Evidence $E = e$.

Output: Augmented buckets $\{B'_i\}$, containing the original functions and all the π and λ functions received from neighbors in the bucket tree.

1. **Pre-processing:** Partition functions to the ordered buckets as usual and generate the bucket tree.
2. **Top-down phase:** λ messages (BE) **do**
   for $i = n$ to 1, in reverse order of d process bucket B_i:
   The message $\lambda_{i \to j}$ from B_i to its parent B_j, is:
   $\lambda_{i \to j} \Leftarrow \sum_{elim(i,j)} \psi_i \cdot \prod_{k \in child(i)} \lambda_{k \to i}$
   endfor
3. **bottom-up phase:** π **messages**
   for $j = 1$ to n, process bucket B_j **do**:
   B_j takes $\pi_{k \to j}$ received from its parent B_k, and computes a message $\pi_{j \to i}$ for each child bucket B_i by
   $\pi_{j \to i} \Leftarrow \sum_{elim(j,i)} \pi_{k \to j} \cdot \psi_j \cdot \prod_{r \neq i} \lambda_{r \to j}$
   endfor
4. **Output:** augmented buckets $B'_1, ..., B'_n$, where each B'_i contains the original bucket functions and the λ and π messages it received.

Figure 5.3: Algorithm bucket-tree elimination.

Bucket-Tree Construction From the Graph



1. Pick a (good) variable ordering, d.
2. Generate the induced ordered graph
3. From top to bottom, each bucket of X is mapped to pairs (variables, functions)
4. The variables are the clique of X, the functions are those placed in the bucket
5. Connect the bucket of X to earlier bucket of Y if Y is the closest node connected to X

Example: Create bucket tree for ordering A,B,C,D,F,G

Asynchronous BTE: Bucket-tree Propagation (BTP)

BUCKET-TREE PROPAGATION (BTP)

Input: A problem $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \prod, \sum \rangle$, ordering d. $X = \{X_1, ..., X_n\}$ and $F = \{f_1, ..., f_r\}$, $\mathbf{E} = \mathbf{e}$. An ordering d and a corresponding bucket-tree structure, in which for each node X_i, its bucket B_i and its neighboring buckets are well defined.

Output: Explicit buckets. Assume functions assigned with the evidence.

1. **for** bucket B_i **do:**
2. **for** each neighbor bucket B_j **do**,
   once all messages from all other neighbors were received, **do**
   compute and send to B_j the message

   $$\lambda_{i \to j} \Leftarrow \sum_{elim(i,j)} \psi_i \cdot (\prod_{k \neq j} \lambda_{k \to i})$$
3. **Output:** augmented buckets $B'_1, ..., B'_n$, where each B'_i contains the original bucket functions and the λ messages it received.

Query Answering

COMPUTING MARGINAL BELIEFS

Input: a bucket tree processed by BTE with augmented buckets: $B\prime_1, ..., B\prime_n$

output: beliefs of each variable, bucket, and probability of evidence.

$$bel(B_i) \Leftarrow \alpha \cdot \prod_{f \in B\prime_i} f$$

$$bel(X_i) \Leftarrow \alpha \cdot \sum_{B_i - \{X_i\}} \prod_{f \in B\prime_i} f$$

$$P(evidence) \Leftarrow \sum_{B_i} \prod_{f \in B\prime_i} f$$

Figure 5.4: Query answering.

Complexity of BTE/BTP on Trees

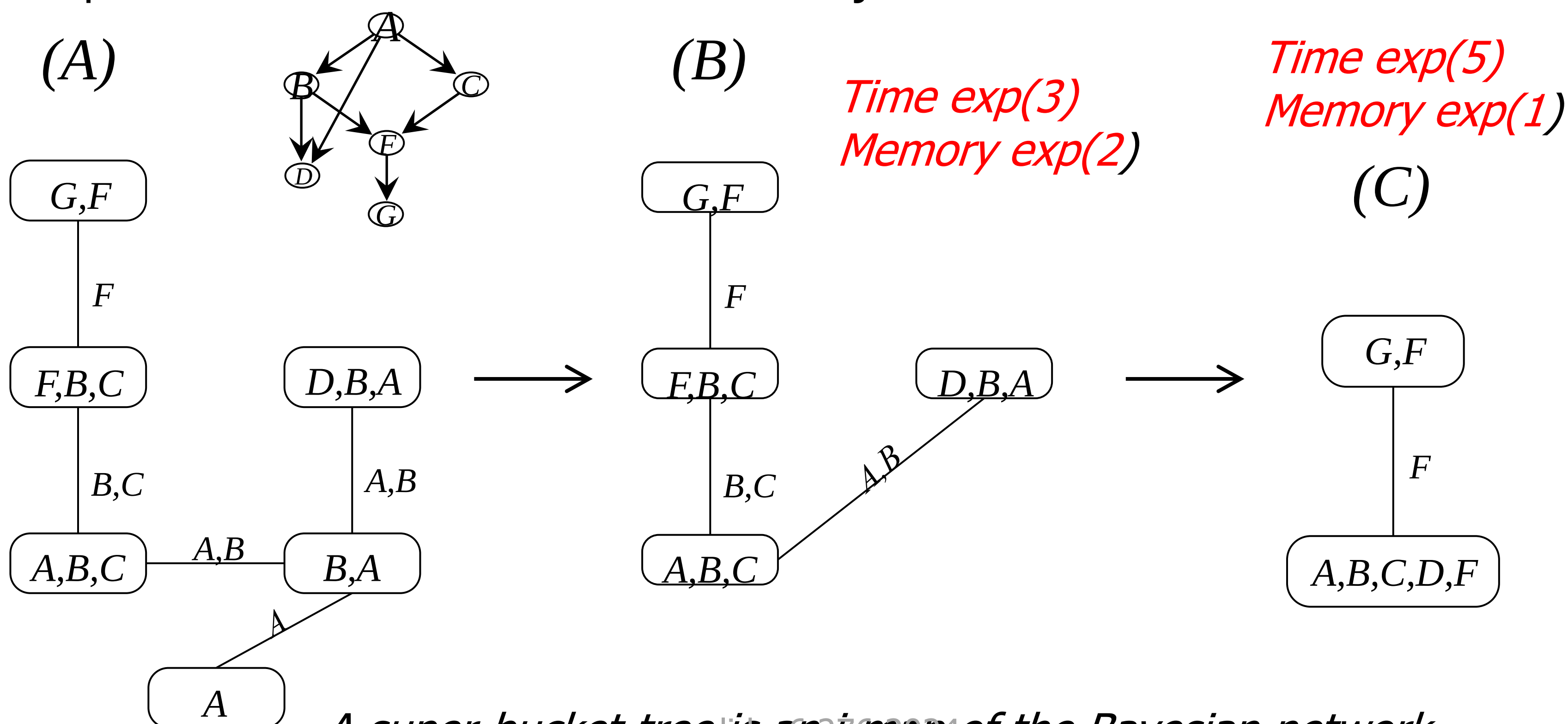
Theorem 5.6 Complexity of BTE. *Let $w^*(d)$ be the induced width of (G^*, d) where G is the primal graph of $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \prod, \sum \rangle$, r be the number of functions in $\mathbf{F}$ and k be the maximum domain size. The time complexity of BTE is $O(r \cdot deg \cdot k^{w^*(d)+1})$, where deg is the maximum degree of a node in the bucket tree. The space complexity of BTE is $O(n \cdot k^{w^*(d)})$.*

Proposition 5.8 BTE on trees *For tree graphical models, algorithms BTE and BTP are time and space $O(nk^2)$ and $O(nk)$, respectively, when k bound the domain size and n bounds the number of variables.*

This will be extended to acyclic graphical models shortly

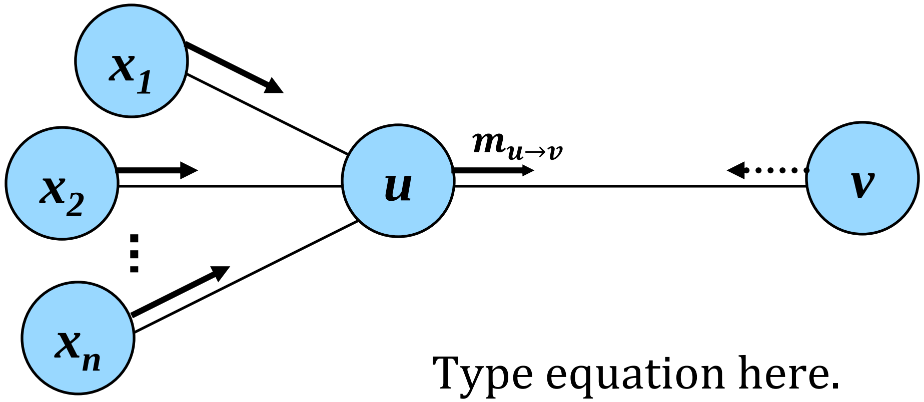
From Buckets to Tree-Clusters

- Merge non-maximal buckets into maximal clusters.
- Connect clusters into a tree: connect 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

Message Passing on a Tree Decomposition



Type equation here.

For max-product
Just replace $\sum$
With max.

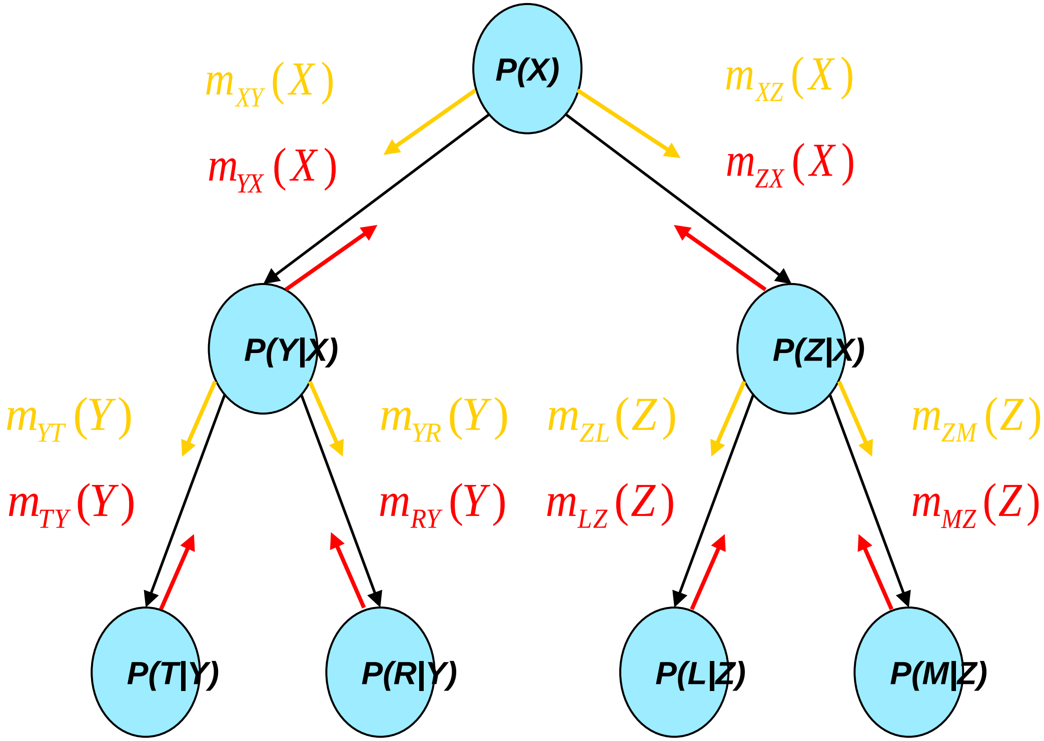
Cluster(u) = $\psi(u) \cup \{m_{X_{1 \to u}}, m_{X_{1 \to u}}, m_{X_{2 \to u}}, \ldots m_{X_{n \to u}}\}$

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

$$\boldsymbol{m_{u \to v}} = \sum_{elim(u,v)} \psi(u) \prod_{r \in neighbor(u), r \neq v} \{\boldsymbol{m_{r \to u}}\}$$

Propagation in Both Directions

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



Outline

- From bucket-elimination (BE) to bucket-tree elimination (BTE)
- From BTE to CTE, Acyclic networks, the join-tree algorithm
- Generating join-trees, the treewidth
- Examples of CTE for Bayesian network
- Belief-propagation on acyclic probabilistic networks (poly-trees) and Loopy networks
- Conditioning with elimination (Dechter, 7.1, 7.2)

The Idea of Cutset-Conditioning

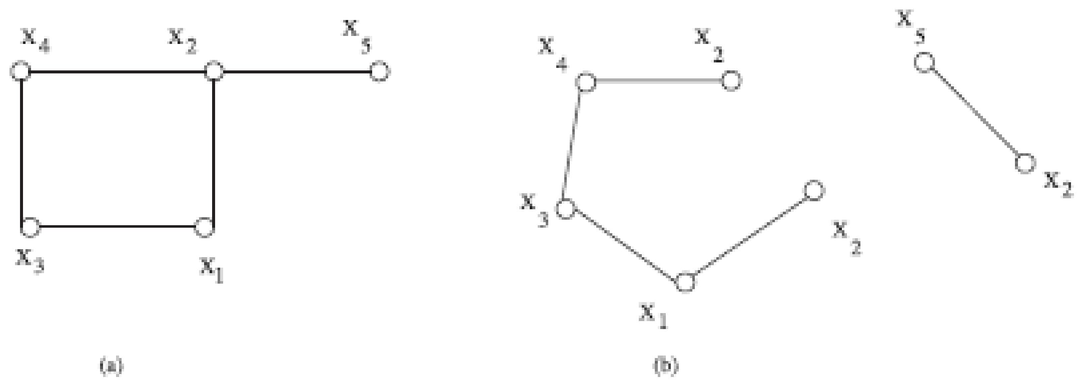
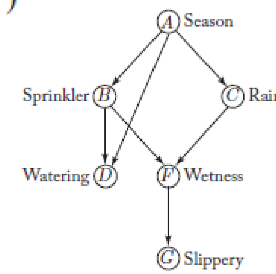


Figure 7.1: An instantiated variable cuts its own cycles.

Conditioning - the Probability Tree

$$P(D=1, G=0)) = \sum_a P(a) \sum_c P(c|a) \sum_b P(b|a) \sum_f P(f|b,c) P(d=1|b,a) P(g=0|f)$$



(a) Directed Acyclic Graph

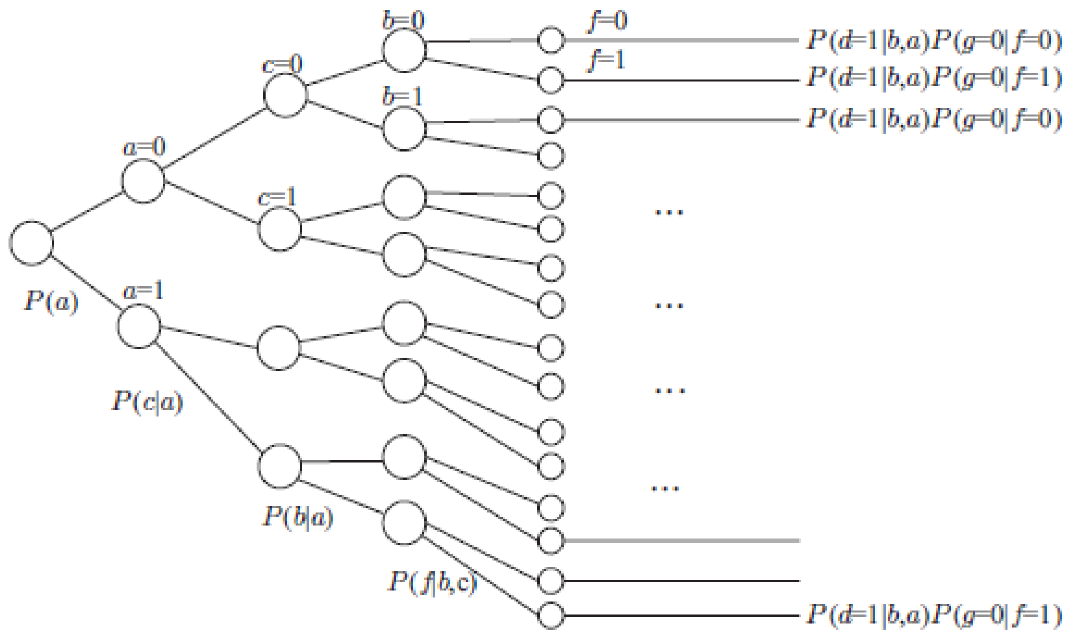
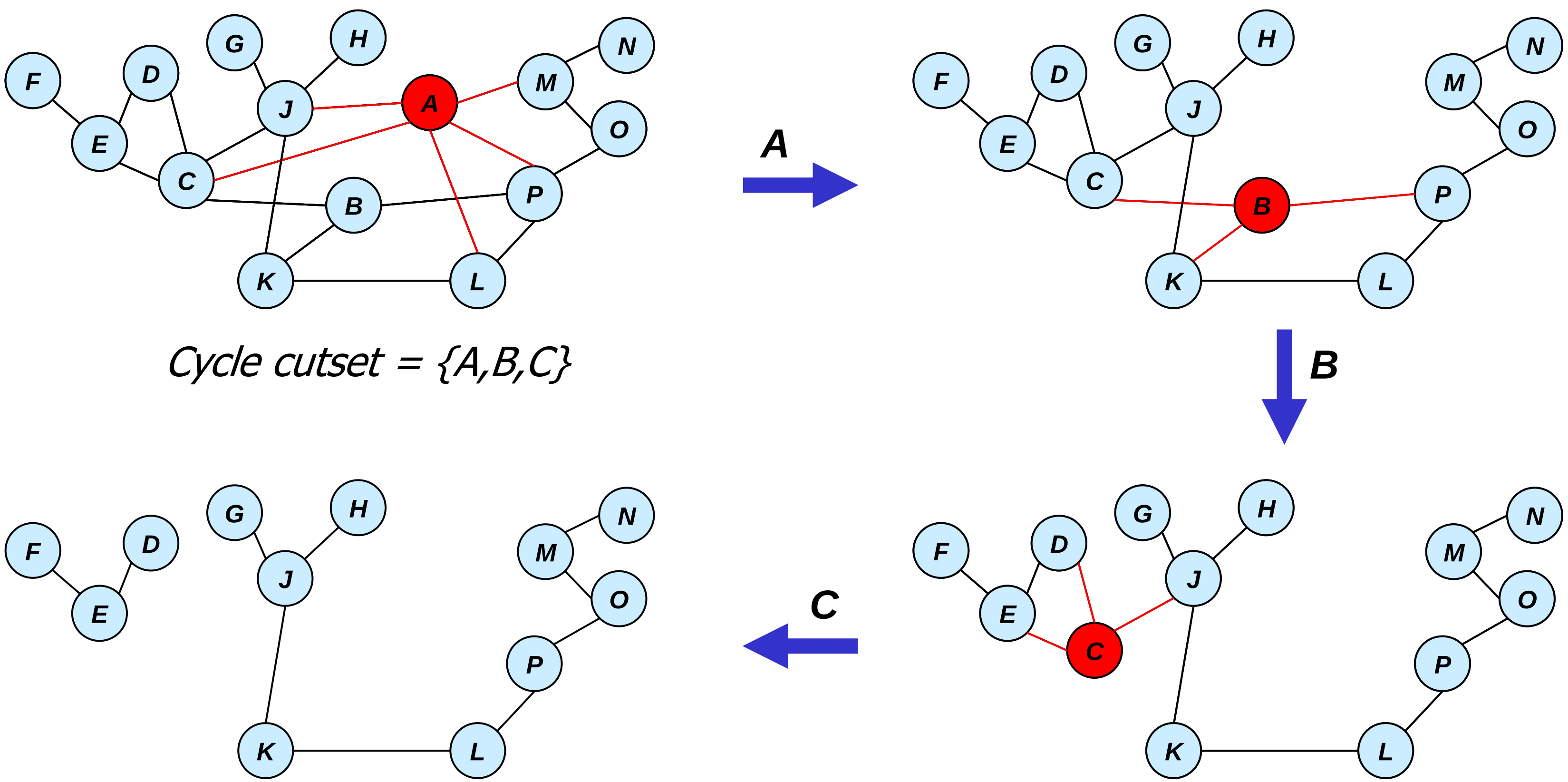


Figure 6.1: Probability tree for computing $P(d=1, g=0)$.

Complexity of conditioning: exponential time, linear space

Cycle-Cutset Conditioning

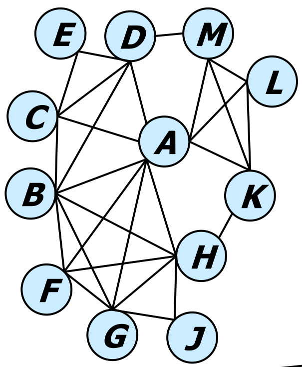


Cycle cutset = {A,B,C}

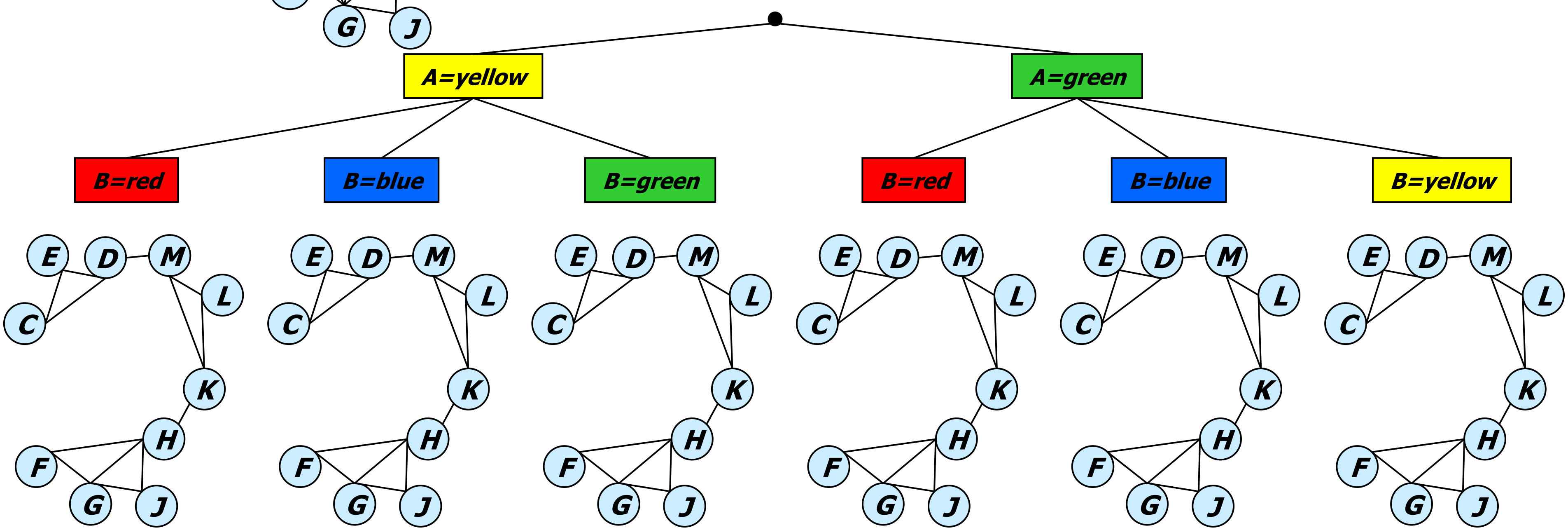
1-cutset = {A,B,C}, size 3

Search Over the Cutset (cont)

Graph Coloring problem



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



2-cutset = {A,B}, size =2

The Impact of Observations

(a) Directed acyclic graph (b) Moral graph

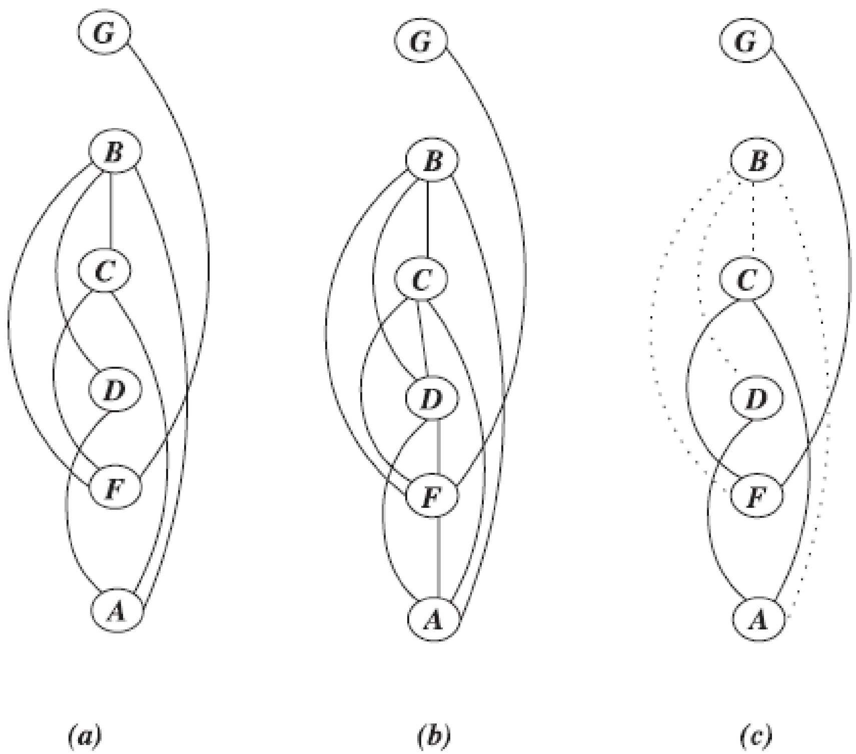


Figure 4.9: Adjusted induced graph relative to observing B.

Ordered graph *Induced graph* *Ordered conditioned graph*

The Idea of Cutset-Conditioning

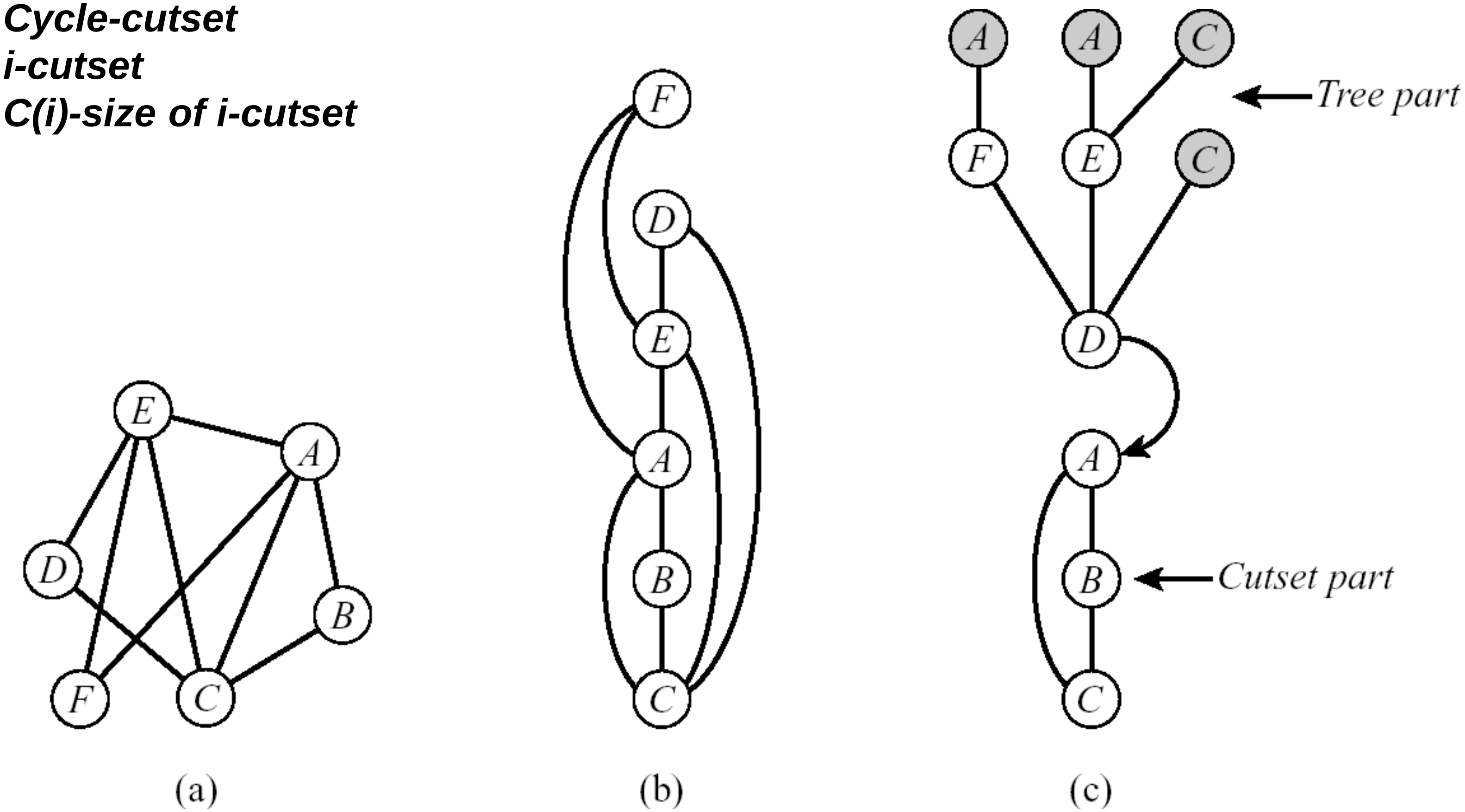
We observed that when variables are assigned, connectivity reduces.
The magnitude of saving is reflected through the "conditioned-induced graph"

- *Cutset-conditioning exploit this in a systematic way:*
- *Select a subset of variables, assign them values, and*
- *Solve the conditioned problem by bucket-elimination.*
- *Repeat for all assignments to the cutset.*

Algorithm VEC

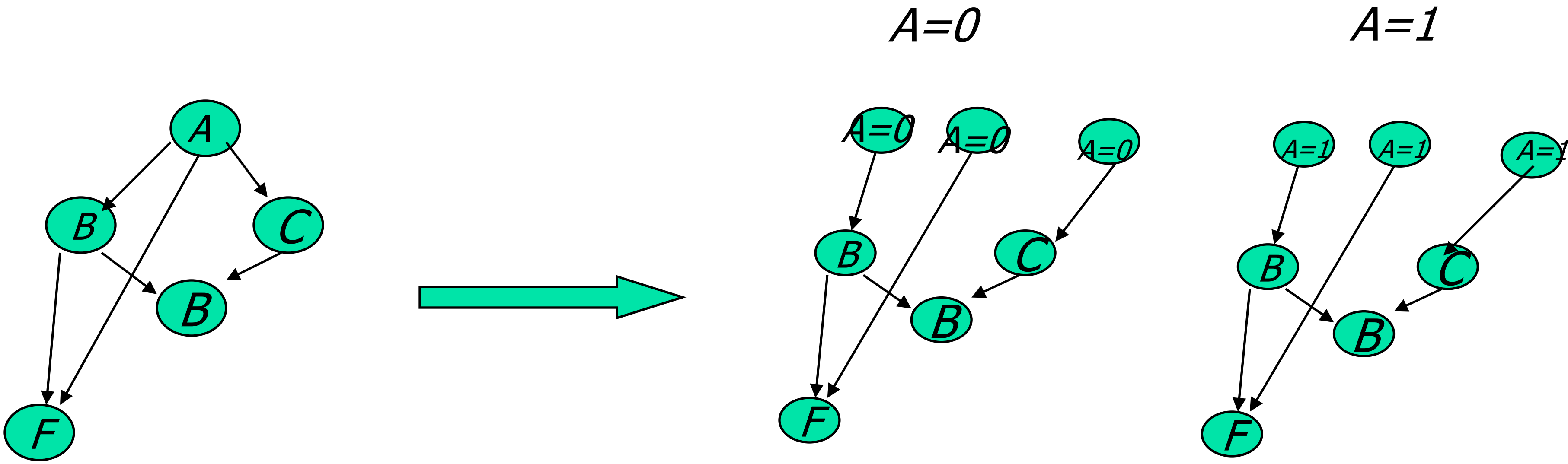
The Cycle-Cutset Scheme:
Condition Until Treeness

- ***Cycle-cutset***
- ***i-cutset***
- ***C(i)-size of i-cutset***



Loop-Cutset Conditioning

- You condition until you get a polytree



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

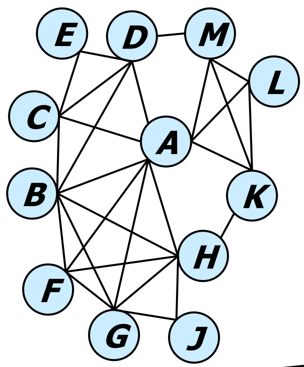
Loop-cutset method is time exponential in loop-cutset size
but linear space. For each cutset we can do BE (belief propagation.)

Loop-Cutset, q-Cutset, cycle-cutset

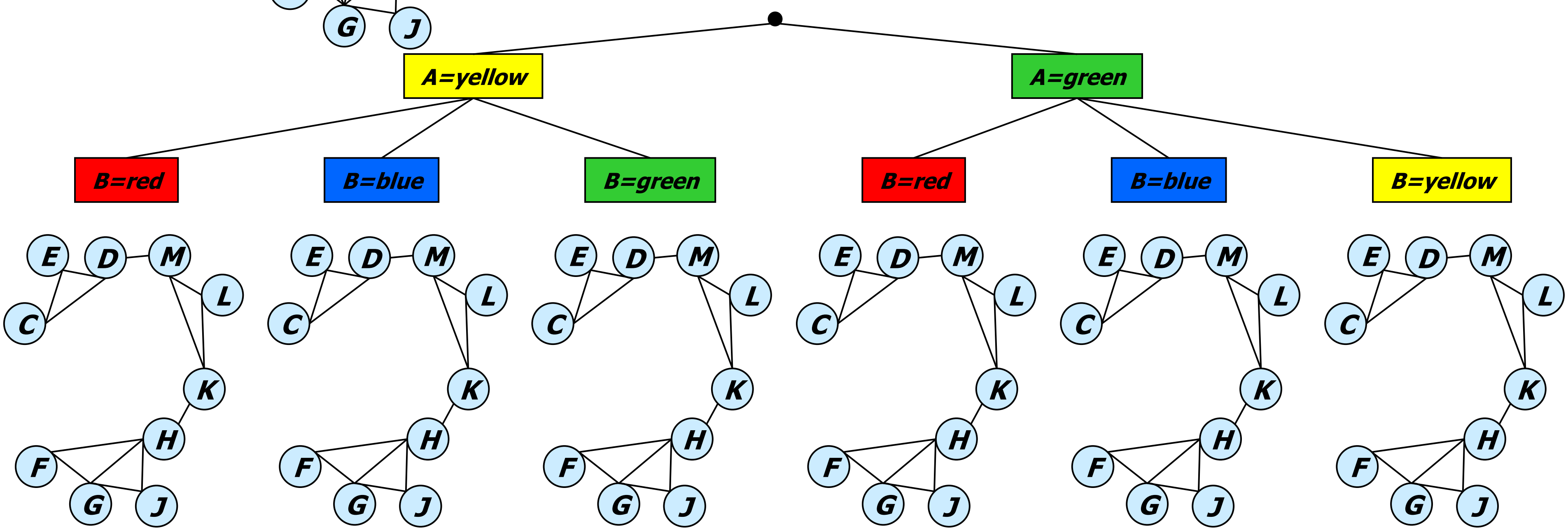
- A loop-cutset is a subset of nodes of a *directed* graph that when removed the remaining graph is a poly-tree
- A q-cutset is a subset of nodes of an *undirected* graph that when removed the remaining graph has an induced-width of q or less.
- A cycle-cutset is a q-cutset such that q=1.

Search Over the Cutset (cont)

Graph Coloring problem



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



2-cutset = {A,B}, size =2

VEC: Variable Elimination with Conditioning; or, q-cutset lgorithms

- VEC-bel:
  - Identify a q-cutset, C, of the network
  - For each assignment to C=c solve the conditioned sub-problem by CTE or BTE.
  - Accumulate probabilities.
  - Time complexity: nk^{c+q+1}
  - Space complexity: nk^q

Algorithm VEC (Variable-elimination with conditioning)

ALGORITHM VEC-EVIDENCE

Input: A belief network $\mathcal{B} = < \mathcal{X}, \mathcal{D}, \mathcal{G}, \mathcal{P} >$, an ordering $d = (x_1, \ldots, x_n)$; evidence e over E, a subset C of conditioned variables;

output: The probability of evidence $P(e)$

Initialize: $\lambda = 0$.

1. For every assignment $C = c$, do
   - $\lambda_1 \leftarrow$ The output of BE-bel with $c \cup e$ as observations.
   - $\lambda \leftarrow \lambda + \lambda_1$. (update the sum).

2. **Return** $P(e) = \alpha \cdot \lambda$ (α is a normalization constant.)

What Hybrid Should We Use?

- q=1? (loop-cutset?)
- q=0? (Full search?)
- q=w* (Full inference)?
- q in between?
- depends... on the graph
- What is relation between cycle-cutset and the induced-width?

Properties; Conditioning+Elimination

Definition 5.6.1 (cycle-cutset,w-cutset) *Given a graph G, a subset of nodes is called a w-cutset iff when removed from the graph the resulting graph has an induced-width less than or equal to w. A minimal w-cutset of a graph has a smallest size among all w-cutsets of the graph. A cycle-cutset is a 1-cutset of a graph.*

A cycle-cutset is known by the name a *feedback vertex set* and it is known that finding the minimal such set is NP-complete [41]. However, we can always settle for approximations, provided by greedy schemes. Cutset-decomposition schemes call for a new optimization task on graphs:

Definition 5.6.2 (finding a minimal w-cutset) *Given a graph $G = (V, E)$ and a constant w, find a smallest subset of nodes U, such that when removed, the resulting graph has induced-width less than or equal w.*

Tradeoff between w* and q-cutstes

Theorem 7.7 *Given graph G, and denoting by c_q^* its minimal q-cutset then,*

$$1 + c_1^* \geq 2 + c_2^* \geq ...q + c_q^*, ... \geq w^* + c_{w*}^* = w^*.$$

Proof. Let's assume that we have a q-cutset of size c_q. Then if we remove it from the graph the result is a graph having a tree decomposition whose treewidth is bounded by q. Let's T be this decomposition where each cluter has size $q + 1$ or less. If we now take the q-cutset variables and add them back to every cluster of T, we will get a tree decomposition of the whole graph (exercise: show that) whose treewidth is $c_q + q$. Therefore, we showed that for *every* c_q-size q-cutset, there is a tree decomposition whose treewidth is $c_q + q$. In particular, for an optimal q-cutset of size c_q^* we have that $w*$, the treewidth obeys, $w* \leq c_q^* + q$. This does not complete the proof because we only showed that for every q, $w* \leq c_q^* + q$. But, if we remove even a single node from a minimal q-cutset whose size is c_q^*, we get a $q + 1$ cutset by definition, whose size is $c_q^* - 1$. Therefore, $c_{q+1}^* \leq c_q^* - 1$. Adding q to both sides of the last inequality we get that for every $1 \leq q \leq w^*$, $q + c_q^* \geq q + 1 + c_{q+1}^*$, which completes the proof. □

Generating Join-trees (Junction-trees); a special type of tree-decompositions

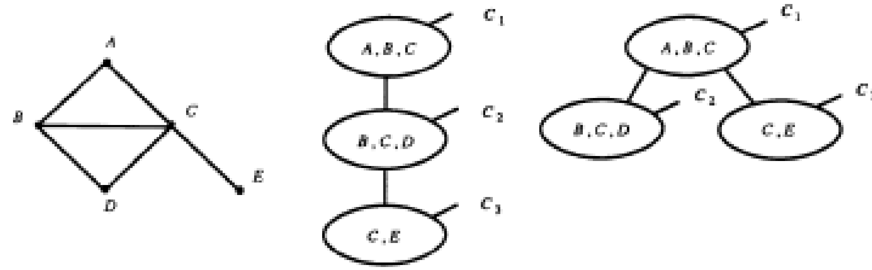
ASSEMBLING A JOIN TREE

1. Use the fill-in algorithm to generate a chordal graph G' (if G is chordal, $G = G'$).

2. Identify all cliques in G'. Since any vertex and its parent set (lower ranked nodes connected to it) form a clique in G', the maximum number of cliques is $|V|$.

3. Order the cliques $C_1, C_2, \ldots, C_t$ by rank of the highest vertex in each clique.

4. Form the join tree by connecting each C_i to a predecessor C_j ($j < i$) sharing the highest number of vertices with C_i.



(a) (b) (c)

EXAMPLE: Consider the graph in Figure 3.9*a*. One maximum cardinality ordering is (A, B, C, D, E).

- Every vertex in this ordering has its preceding neighbors already connected, hence the graph is chordal and no edges need be added.
- The cliques are ranked C_1, C_2, and C_3 as shown in Figure 3.9*b*.
- $C_3 = \{C, E\}$ shares only vertex C with its predecessors C_2 and C_1, so either one can be chosen as the parent of C_3.
- These two choices yield the join trees of Figures 3.9*b* and 3.9*c*.
- Now suppose we wish to assemble a join tree for the same graph with the edge (B, C) missing.
- The ordering (A, B, C, D, E) is still a maximum cardinality ordering, but now when we discover that the preceeding neighbors of node D (i.e., B and C) are nonadjacent, we should fill in edge (B, C).
- This renders the graph chordal, and the rest of the procedure yields the same join trees as in Figures 3.9*b* and 3.9*c*.

Examples of (Join)-Trees Construction