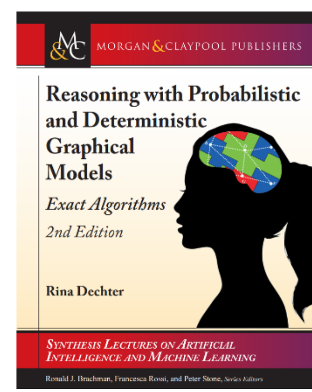
Causal and Probabilistic Reasoning

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)

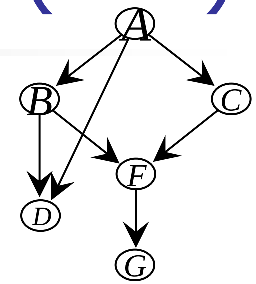
Outline

- 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
- 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?

Bucket G: $P(G|F)$

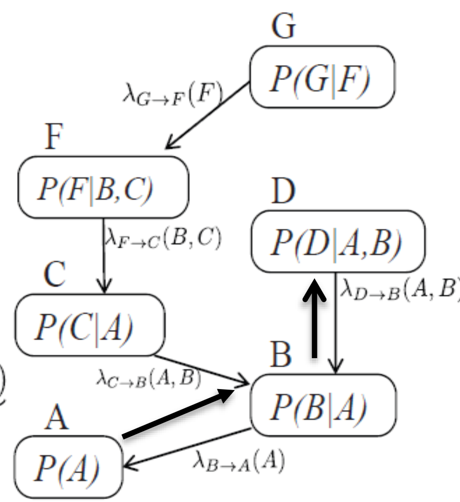
Bucket F: $P(F|B,C)$ $\lambda_{G\to F}(F)$

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

Bucket C: $P(C|A)$ $\lambda_{F\to C}(B,C)$

Bucket B: $P(B|A)$ $\lambda_{D\to B}(A,B)$ $\lambda_{C\to B}(A,B)$

Bucket A: $P(A)$ $\lambda_{B\to A}(A)$



P(D)?

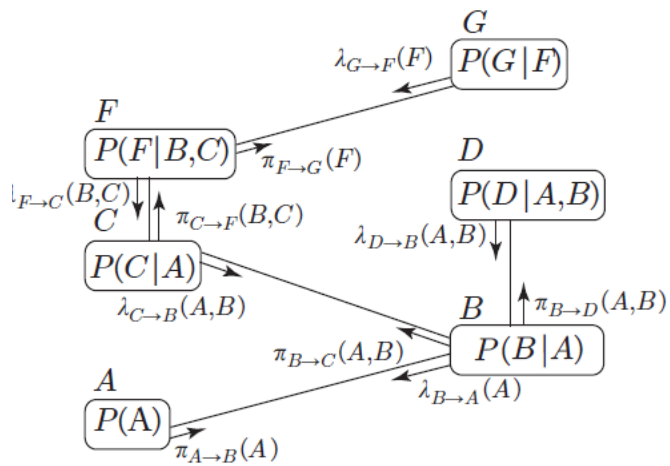
$$\pi_{A\to B}(a) = P(A),$$

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

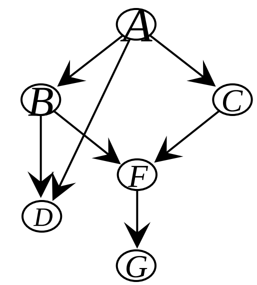
$$bel(d) = \alpha\sum_{a,b} P(d|a,b)\cdot\pi_{B\to D}(a,b).$$

BTE: Allows Messages Both Ways

Initial buckets
+ messages

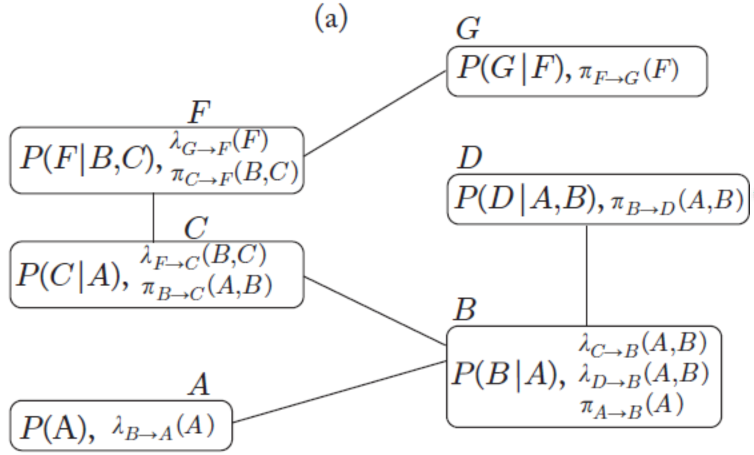


(a)



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)

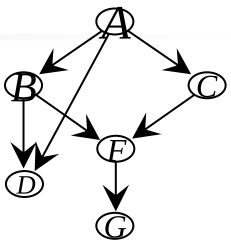
Algorithm bucket-tree elimination (BTE)

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)

All messages are called lambda

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
   $$\boxed{\lambda_{i \to j} \Leftarrow \sum_{elim(i,j)} \psi_i \cdot \left(\prod_{k \neq j} \lambda_{k \to i}\right)}$$
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

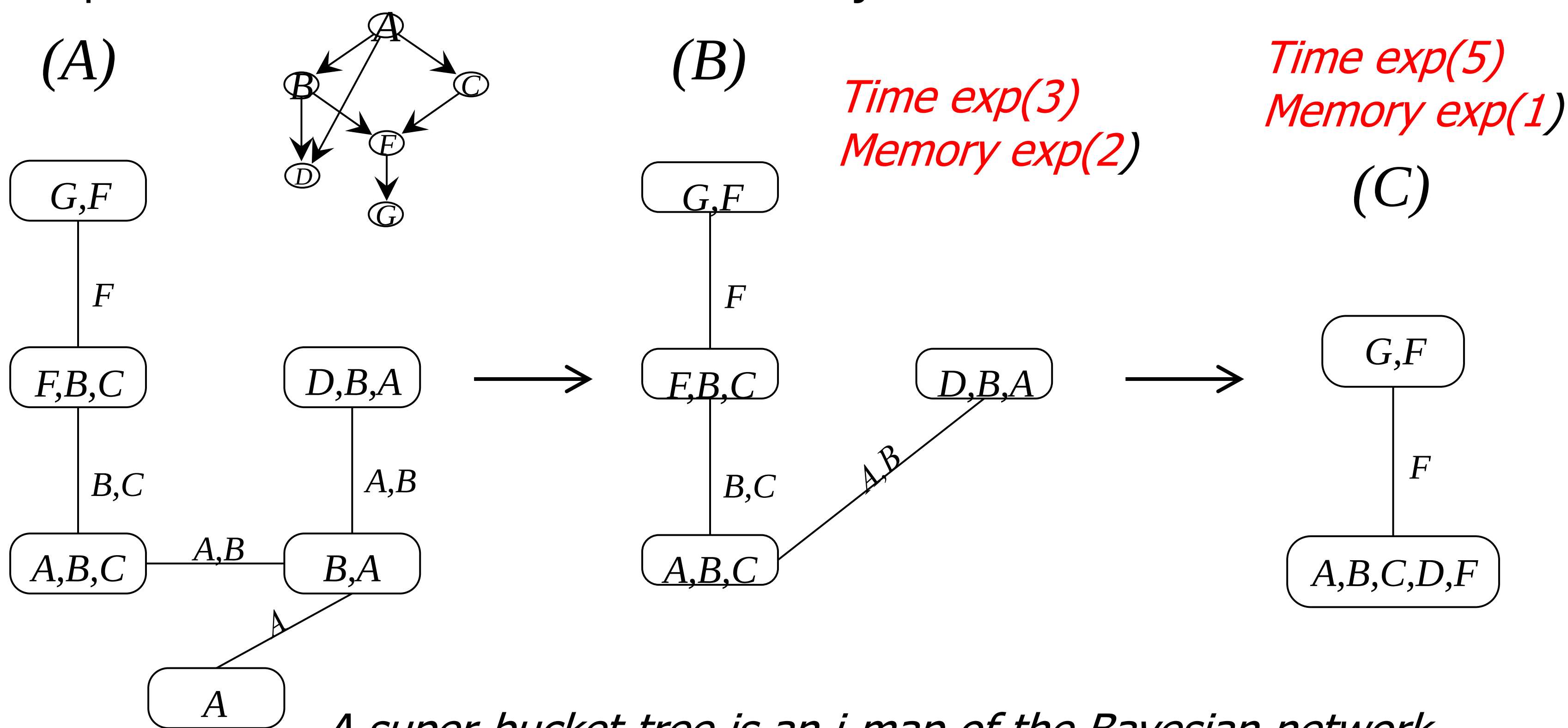
Theorem 5.6 Complexity of BTE. *Let $w^*(d)$ be the induced width of (G^*, d) where G is the primal graph of $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \prod, \sum \rangle$, r be the number of functions in $\mathbf{F}$ and k be the maximum domain size. The time complexity of BTE is $O(r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

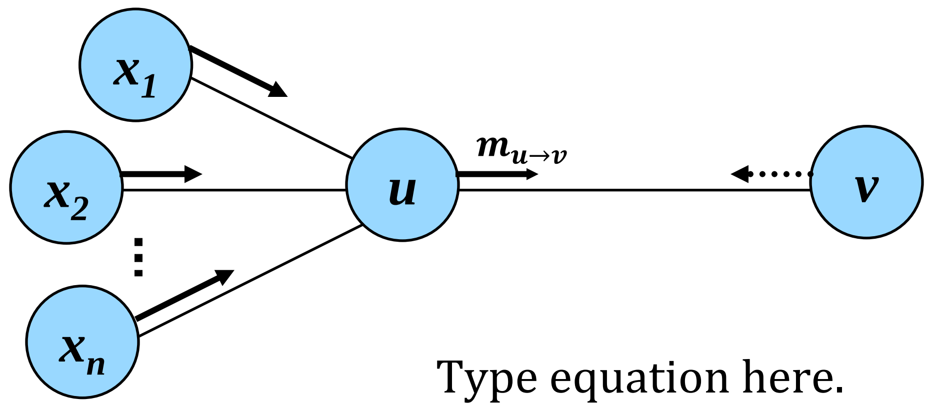
From Buckets to Tree-Clusters

- 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.

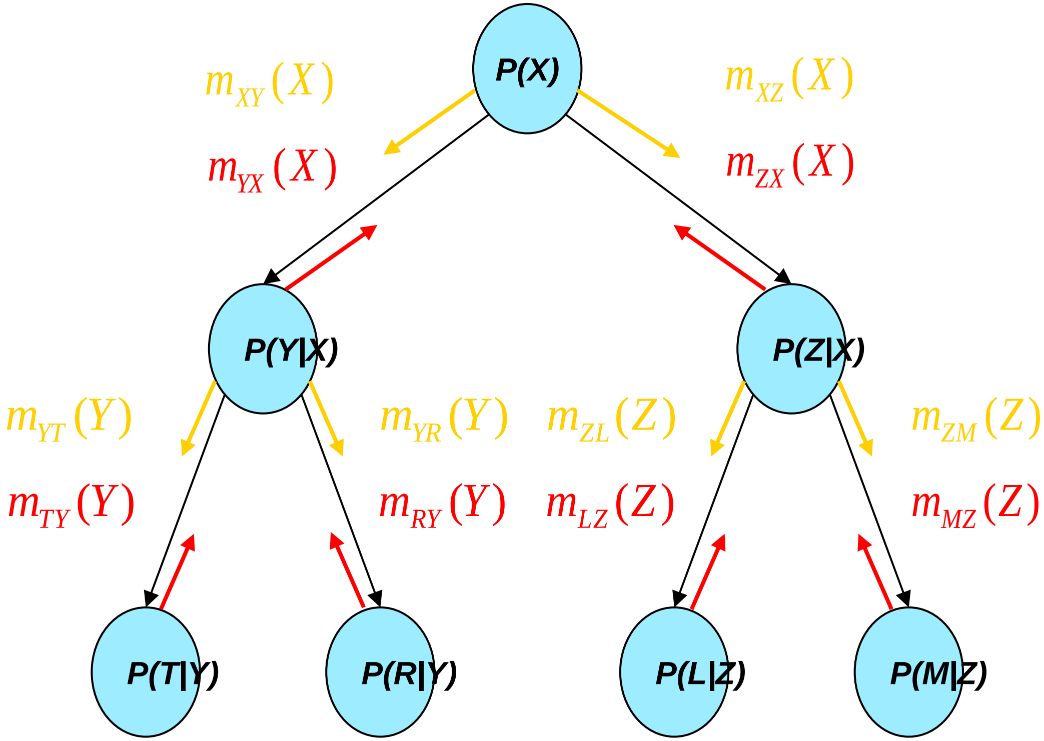
Cluster(u) = $\psi(u) \cup \{m_{X_{1\rightarrow u}}, m_{X_{1\rightarrow u}}, m_{X_{2\rightarrow u}}, \ldots m_{X_{n\rightarrow u}}\}$

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

$$\boldsymbol{m_{u\rightarrow v}} = \sum\nolimits_{elim(u,v)} \psi(u) \prod\nolimits_{r\, \in neighbor(u), r\neq v} \{\boldsymbol{m_{r\rightarrow u}}\}$$

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 (also called, junction-tree algorithm)
- Examples of CTE for Bayesian network
- Belief-propagation on acyclic probabilistic networks (poly-trees) and Loopy networks

Acyclic Graphical Models

- **Dual network**: Each scope of a CPT is a node and each arc is denoted by intersection.
- **Acylic network:** when the dual graph is a tree or has a join-tree
- Tree-clustering converts a network into an acyclic one.

From Acyclic Networks

Sometime the dual graph seems to not be a tree, but it is in fact, a tree. This is because some of its arcs are redundant and can be removed while not violating the original independency relationships that is captured by the graph. A [illegible]

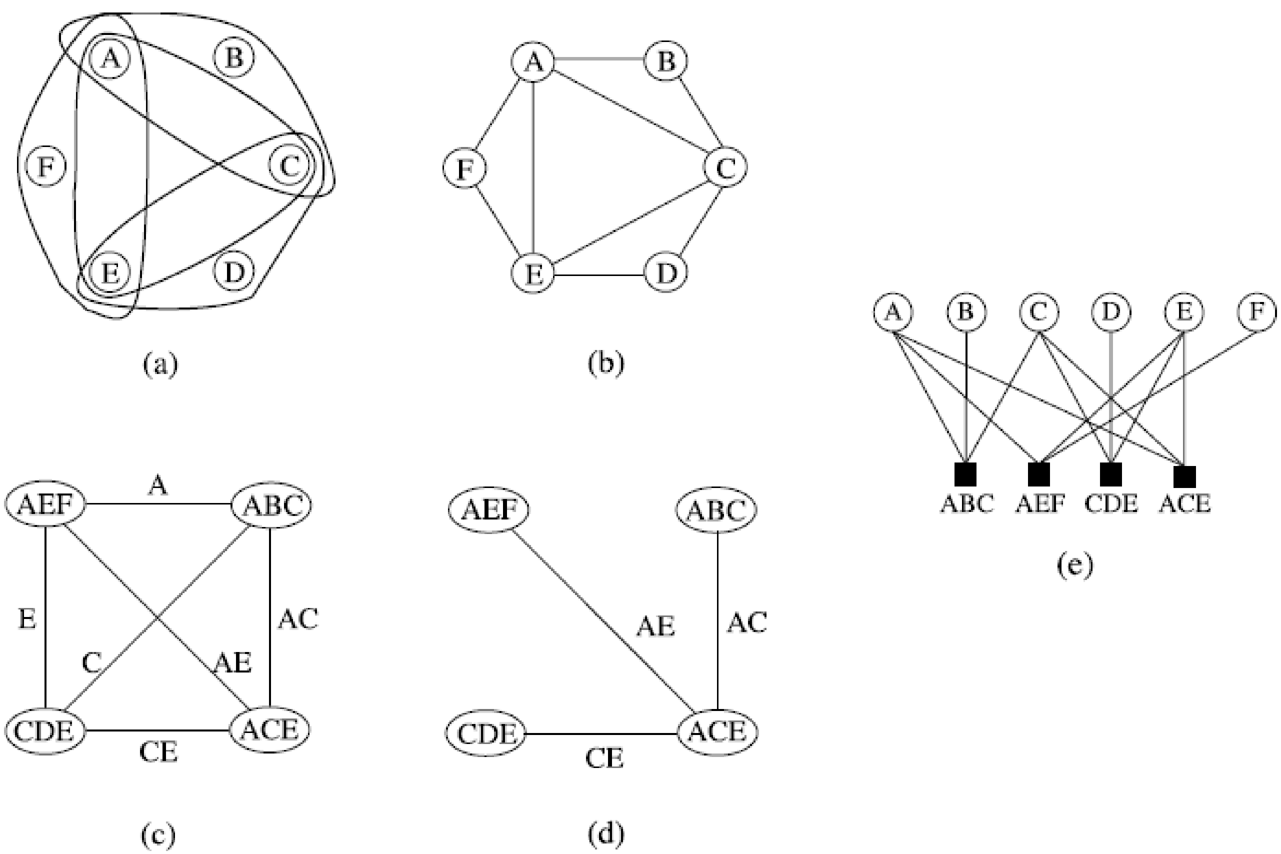


Figure 5.1: (a)Hyper, (b)Primal, (c)Dual and (d)Join-tree of a graphical model having scopes ABC, AEF, CDE and ACE. (e) the factor graph

Connectedness and Ascyclic Dual Graphs
(The Running Intersection Property)

Definition 5.11 Connectedness, join-trees. Given a dual graph of a graphical model $\mathcal{M}$, an arc subgraph of the dual graph satisfies the *connectedness* property iff for each two nodes that share a variable, there is at least one path of labeled arcs of the dual graph such that each contains the shared variables. An arc subgraph of the dual graph that satisfies the connectedness property is called a *join-graph* and if it is a tree, it is called a *join-tree*.

Definition: A graphical model whose dual graph has a join-tree is acyclic

Theorem: BTE is time and space linear on acyclic graphical models

Tree-decomposition: If we transform a general model into an acyclic one it can then be solved by a BTE/BTP scheme. Also known as tree-clustering

Tree Decompositions

A *tree decomposition* for a graphical model $< X,D,P >$ 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 :

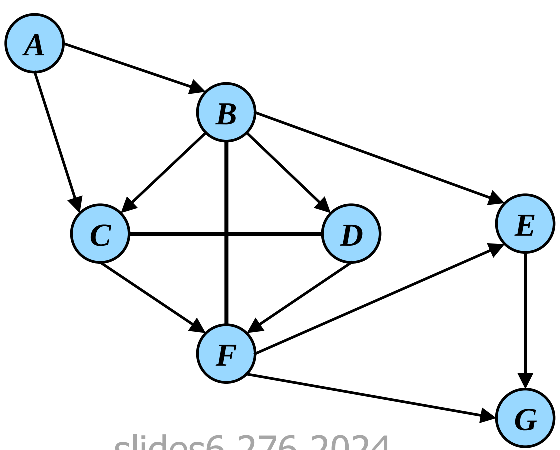
1. For each function $p_i \in P$ there is exactly one vertex such that $p_i \in \psi(v)$ and $scope(p_i) \subseteq \chi(v)$
2. For each variable $X_i \in X$ the set $\{v \in V | X_i \in \chi(v)\}$ forms a connected subtree (running intersection property)

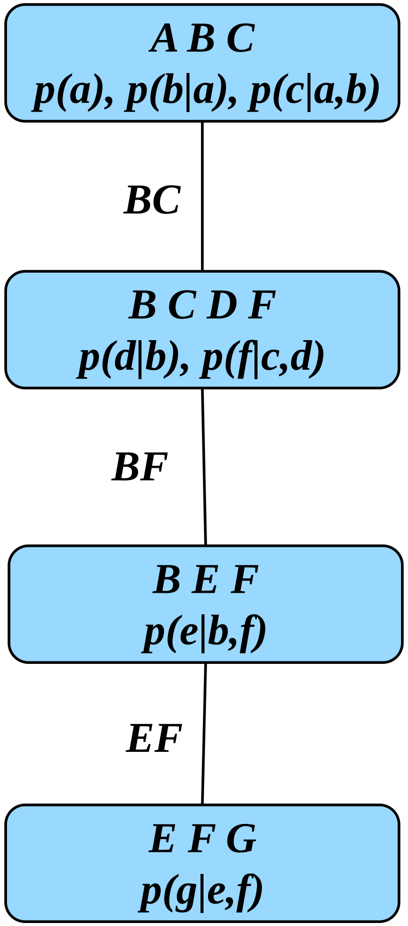
Tree Decompositions

A *tree decomposition* for a graphical model $< X,D,P >$ 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)

Connectedness, or
Running intersection property



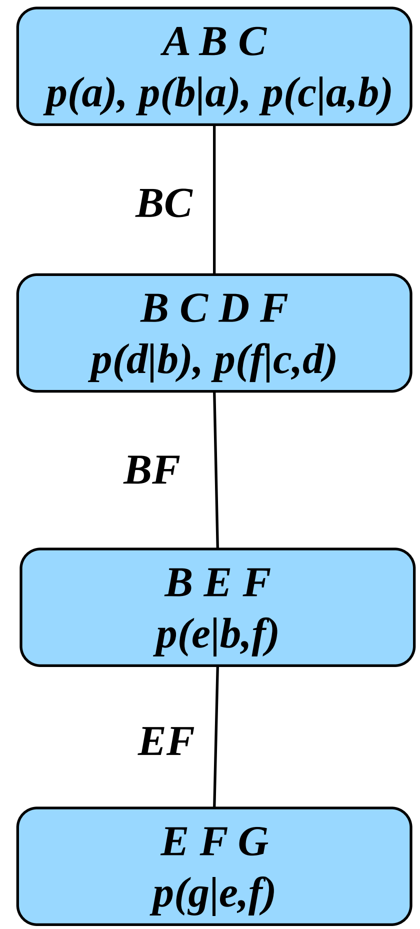


Tree decomposition

Tree Decompositions (TD)

A *tree decomposition* for a graphical model $< X,D,P >$ 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)



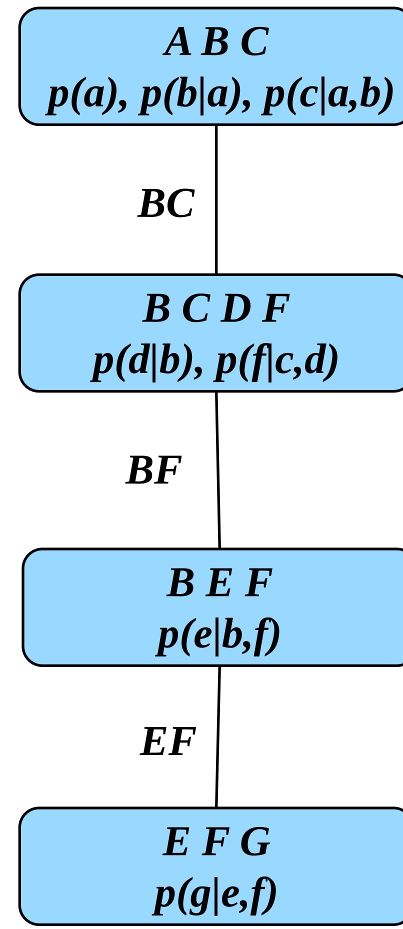
Tree decomposition

Treewidth: maximum number of variables in a node of TD – 1
Seperator-width: maximum intersection between adjacent nodes
Eliminator: elim(u,v) = χ(u) - χ(v)

H-Tree Decompositions (TD)

A *tree decomposition* for a graphical model $< X,D,P >$ 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)



Treewidth: maximum number of variables in a node of TD – 1
Seperator-width: maximum intersection between adjacent nodes
Eliminator: elim(u,v) = $\chi(u) - \chi(v)$
Hypertree-width = Max number off functions in a node

Hypertree-decomposition: 3. If every variable in a node is covered by a function scope.

Cluster-Tree Elimination

CLUSTER-TREE ELIMINATION (CTE)

Input: A tree decomposition $< T, \chi, \psi >$ for a problem $M =< X, D, F, \prod, \sum \} >$, $X = \{X_1, ..., X_n\}$, $F = \{f_1, ..., f_r\}$. Evidence $E = e$, $\psi_u = \prod_{f \in \psi(u)} f$

Output: An augmented tree decomposition whose clusters are all model explicit.
Namely, a decomposition $< T, \chi, \bar{\psi} >$ where $u \in T$, $\bar{\psi}(u)$ is model explicit relative to $\chi(u)$.

1. **Initialize.** (denote by $m_{u \to v}$ the message sent from vertex u to vertex v.)
2. **Compute messages:**

   For every node u in T, once u received messages from all neighbors but v,

   Process observed variables:

   For each node $u \in T$ assign relevant evidence to $\psi(u)$

   Compute the message:

   $$m_{u \to v} \leftarrow \sum_{\chi(u) - sep(u,v)} \psi_u \cdot \prod_{r \in neighbor(u), r \neq v} m_{r \to u}$$

   endfor

   Note: functions whose scopes do not contain any separator variable do not need to be combined and can be directly passed on to the receiving vertex.
3. **Return:** The explicit tree $< T, \chi, \bar{\psi} >$, where

   $\bar{\psi}(v) \Leftarrow \psi(v) \cup_{u \in neighbor(v)} \{m_{u \to v}\}$

   return the explicit function: for each v, $M_{\chi(v)} = \prod_{f \in \bar{\psi}(v)} f$

Properties of CTE

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

- 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
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, treewidth
sep = the separator size

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

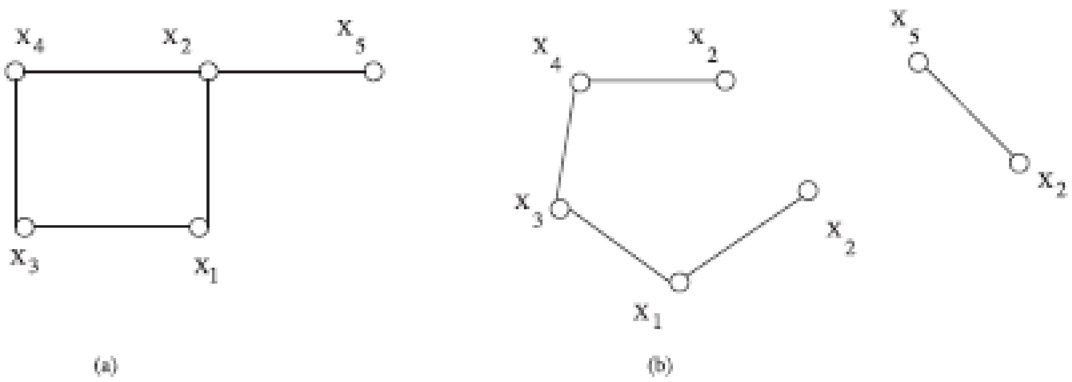
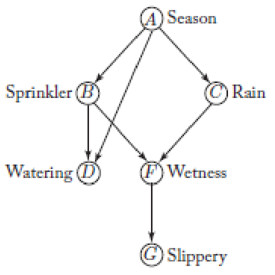


Figure 7.1: An instantiated variable cuts its own cycles.

Conditioning - the Probability Tree

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



(a) Directed Acyclic Graph

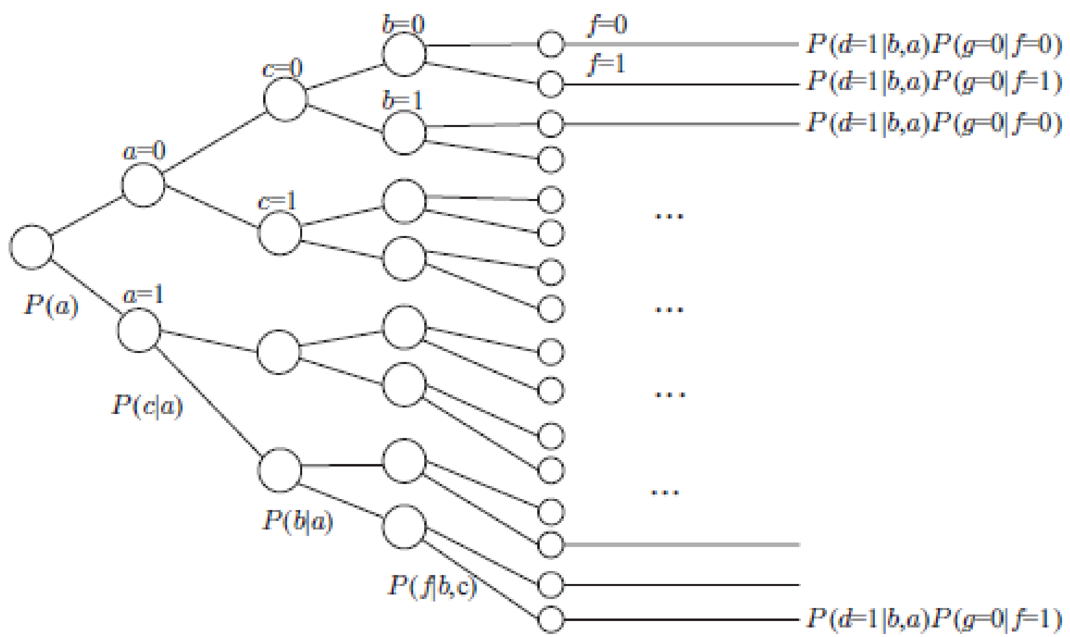
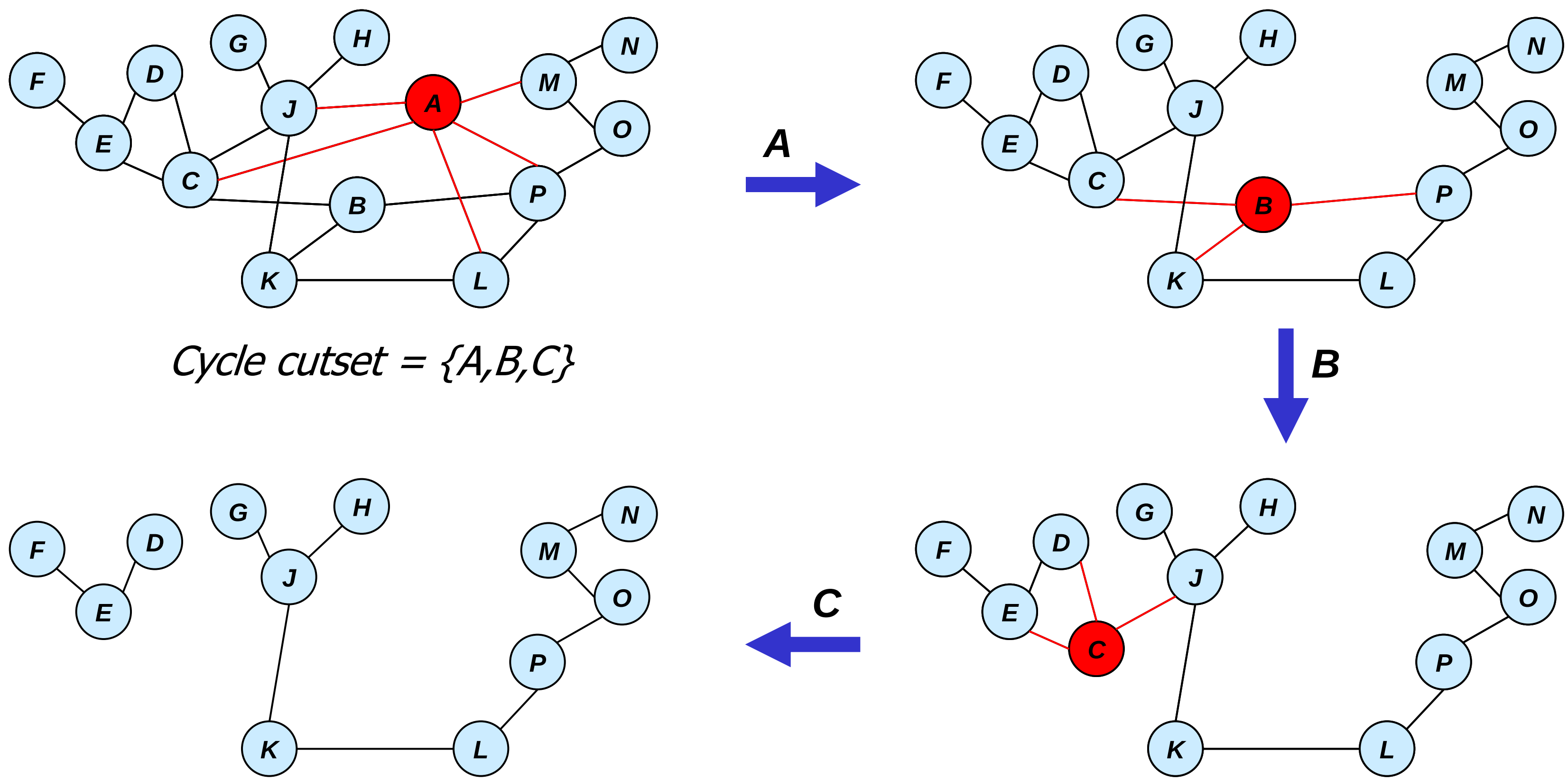


Figure 6.1: Probability tree for computing $P(d=1, g=0)$.

Complexity of conditioning: exponential time, linear space

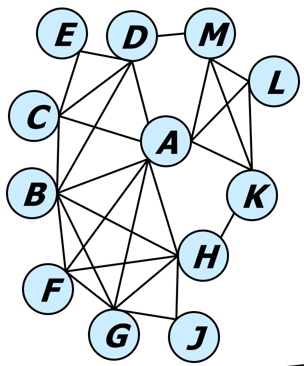
Cycle-Cutset Conditioning



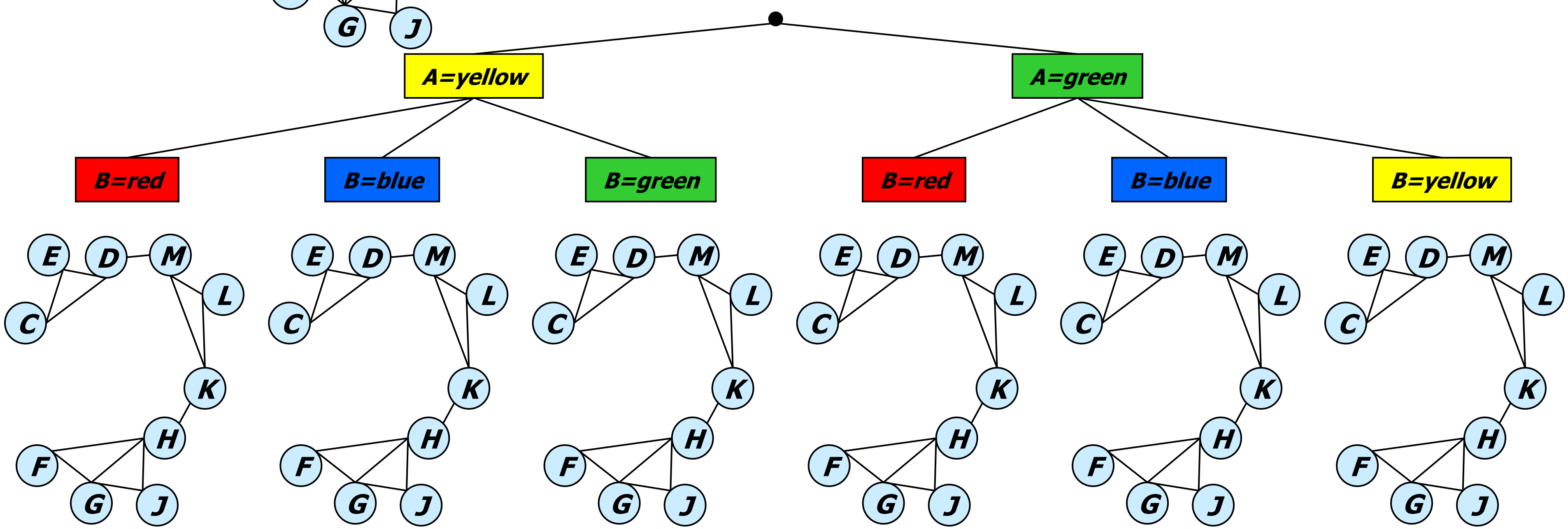
1-cutset = {A,B,C}, size 3

Search Over the Cutset (cont)

Graph Coloring problem



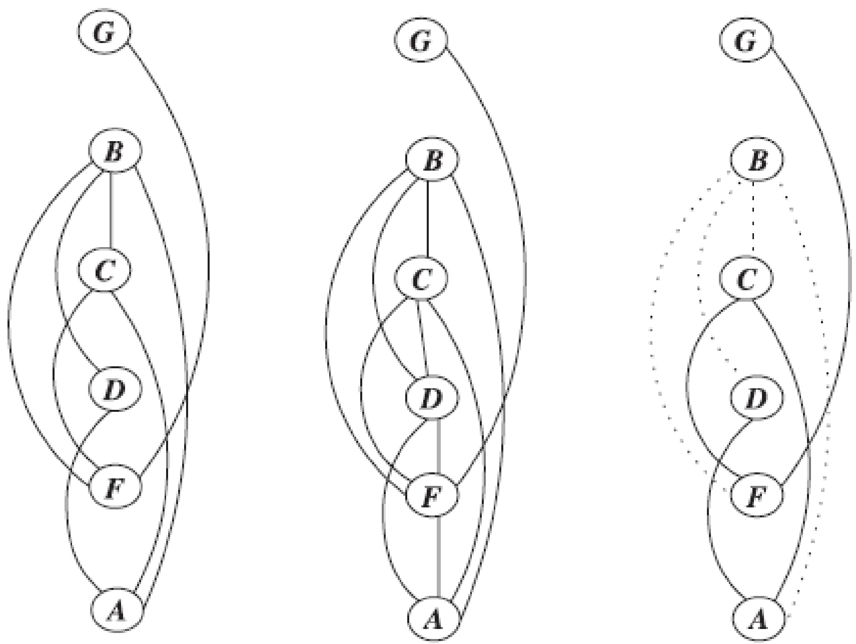
- *Inference may require too much memory*
- ***Condition*** *on some of the variables*



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

The Impact of Observations

(a) Directed acyclic graph (b) Moral graph



(a) (b) (c)

Figure 4.9: Adjusted induced graph relative to observing B.

Ordered graph *Induced graph* *Ordered conditioned graph*

The Idea of Cutset-Conditioning

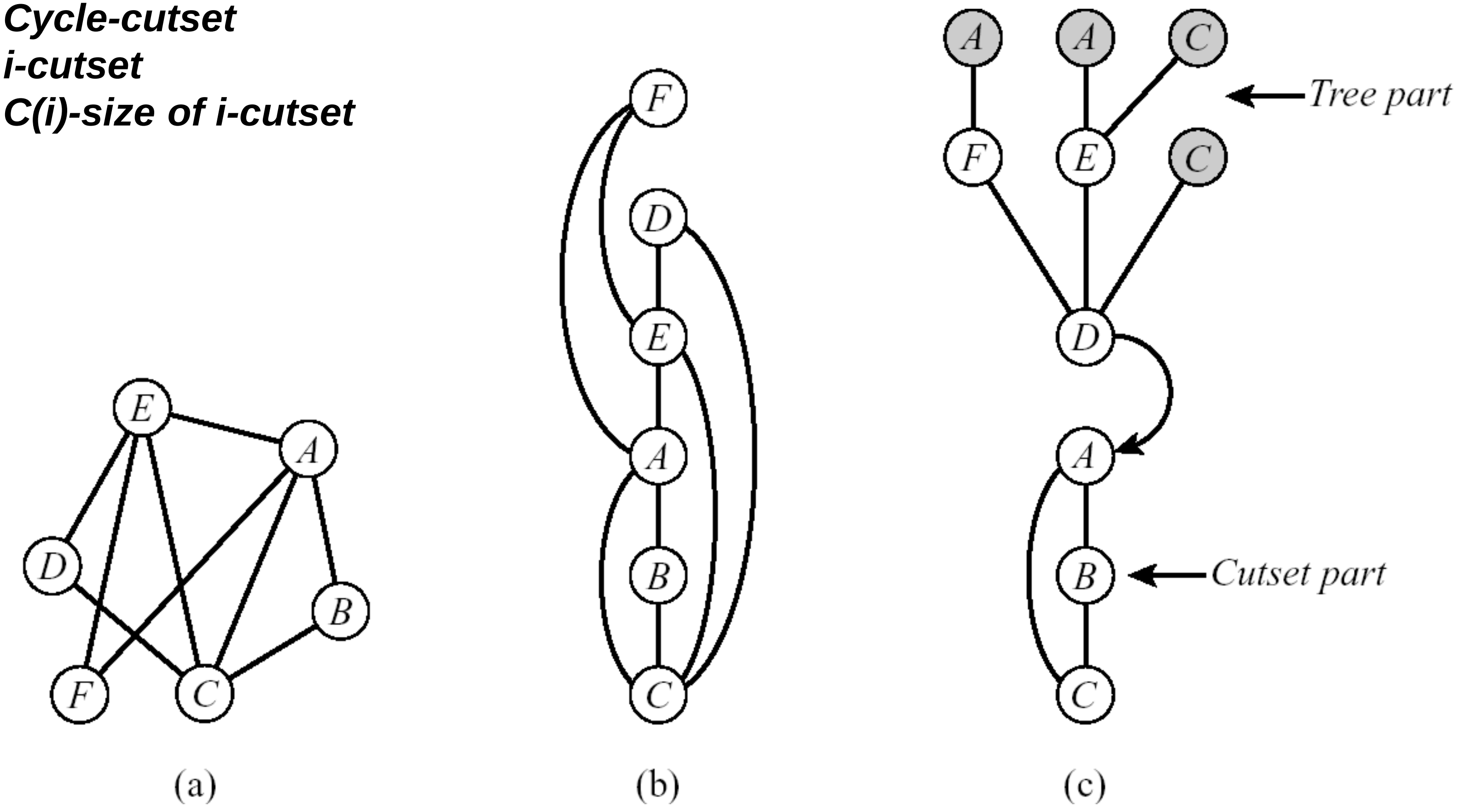
We observed that when variables are assigned, connectivity reduces.
The magnitude of saving is reflected through the "conditioned-induced graph"

- *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

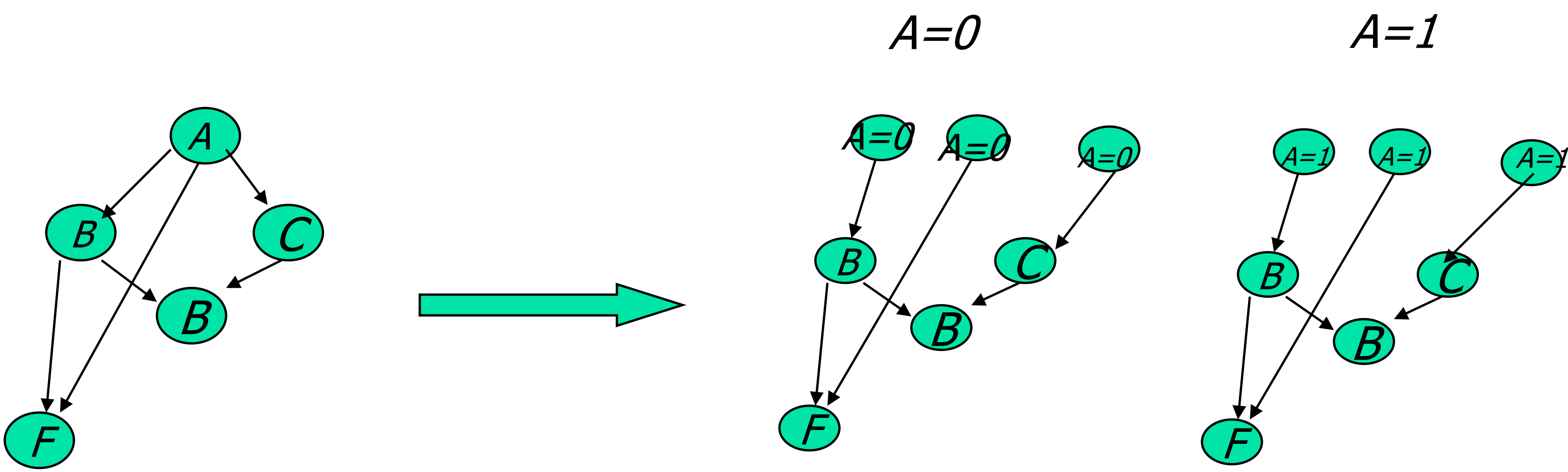
The Cycle-Cutset Scheme:
Condition Until Treeness

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



Loop-Cutset Conditioning

- You condition until you get a polytree



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

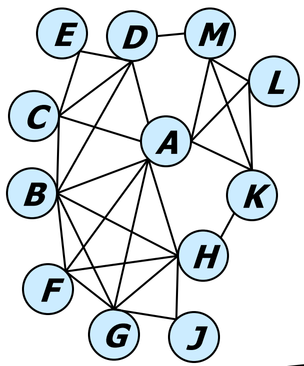
Loop-cutset method is time exponential in loop-cutset size
but linear space. For each cutset we can do BE (belief propagation.)

Loop-Cutset, q-Cutset, cycle-cutset

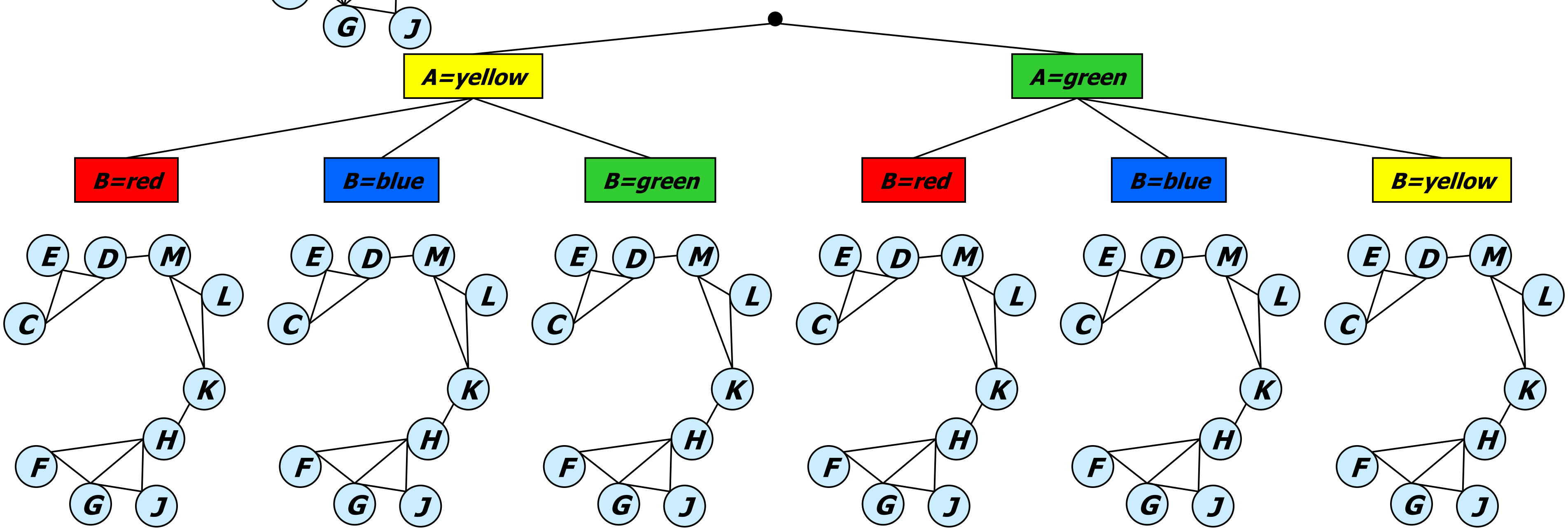
- A loop-cutset is a subset of nodes of a *directed* graph that when removed the remaining graph is a poly-tree
- A q-cutset is a subset of nodes of an *undirected* graph that when removed the remaining graph has an induced-width of q or less.
- A cycle-cutset is a q-cutset such that q=1.

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,*

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

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

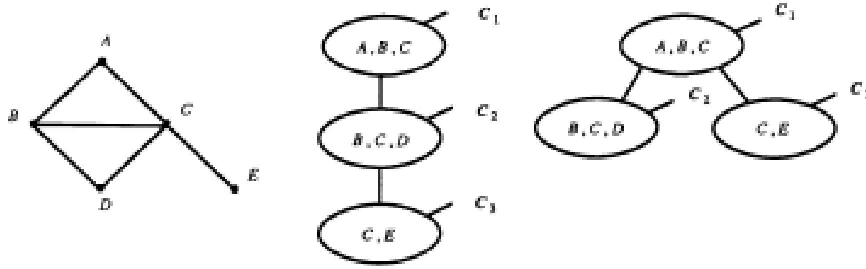
ASSEMBLING A JOIN TREE

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.

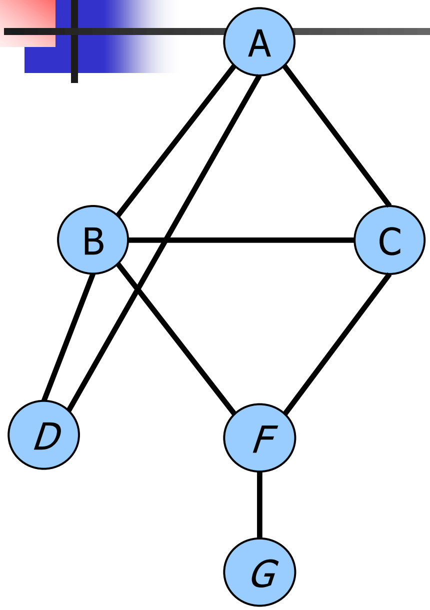


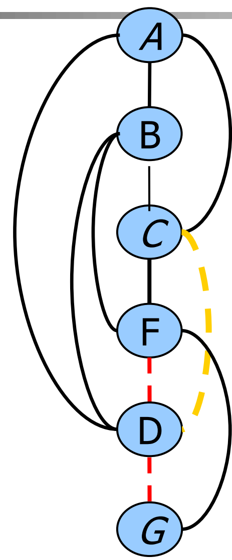
(a) (b) (c)

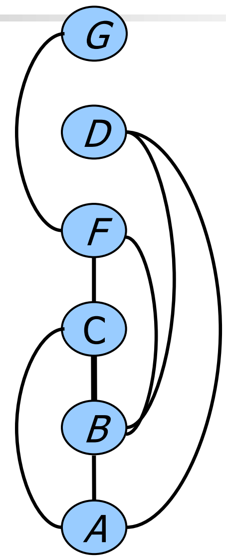
EXAMPLE: Consider the graph in Figure 3.9*a*. One maximum cardinality ordering is (A, B, C, D, E).

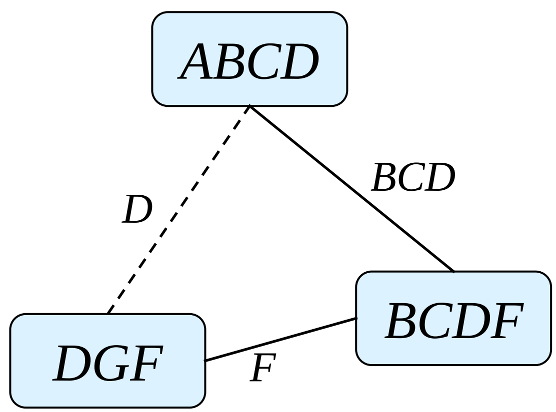
- Every vertex in this ordering has its preceding neighbors already connected, hence the graph is chordal and no edges need be added.
- The cliques are ranked C_1, C_2, and C_3 as shown in Figure 3.9*b*.
- $C_3 = \{C, E\}$ shares only vertex C with its predecessors C_2 and C_1, so either one can be chosen 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*.

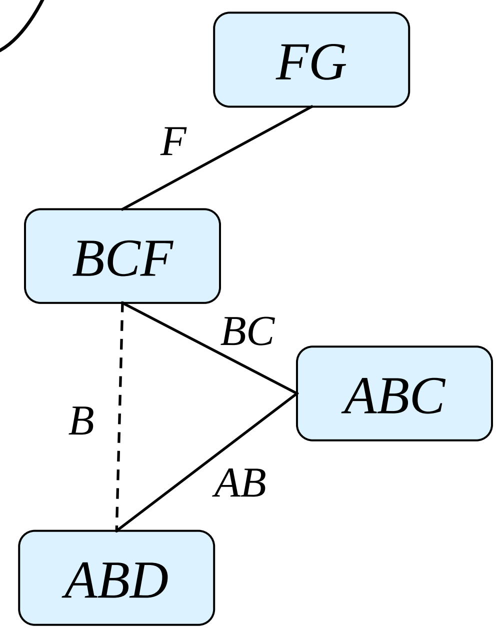
Examples of (Join)-Trees Construction











Hypertree width and EmpBN

Tree and Hypertree Decompositions

A ***tree decomposition*** *of a graphical model is a triple* $<T, \chi, \psi>$*, where* $T=<V,E>$ *is a tree and* χ *and* ψ *are labeling*
functions that associate with each vertex $v \in V$ *two sets,* $\chi(v) \subseteq X$ *and* $\psi(v) \subseteq F$*, that satisfy the following conditions:*

- *for each* $f_j \in F$*, there is at least one* $v \in V$ *such that* $f_j \in \psi(v)$*.*
- *if* $f_j \in \psi(v)$*, then* $scope(f_j) \subseteq \chi(v)$*.*
- *for each* $x_i \in X$*, the set* $\{v \in V \mid x_i \in \chi(v)\}$ *induces a connected subtree of T.*

The ***tree width*** *of T is* $\mathbf{w = max_{v \in V} |\chi(v)| - 1}$*. T is also a* ***hypertree decomposition*** *if it satisfies the following additional*
condition:

- *for each* $v \in V$*,* $\chi(v) \subseteq \bigcup_{f_j \in \psi(v)} scope(f_j)$*.*

In this case, the ***hypertree width*** *of T is* $\mathbf{hw = max_{v \in V} |\psi(v)|}$*.*

Finding tree and hypertree decompositions of minimal width is known to be NP-complete, therefore heuristic algorithms are employed in practice. Once a tree or hypertree decomposition is available, it can be processed by a suitable version of a message passing algorithm like ***Cluster-Tree Elimination (CTE)****.*

Empirical Factors Sparse representation

Table 80: 6 Variables with domain size 3

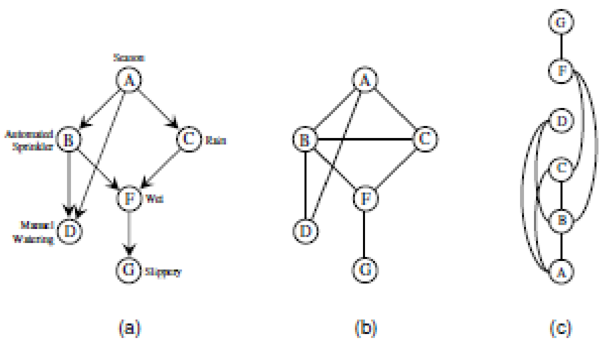
(a) Data Table

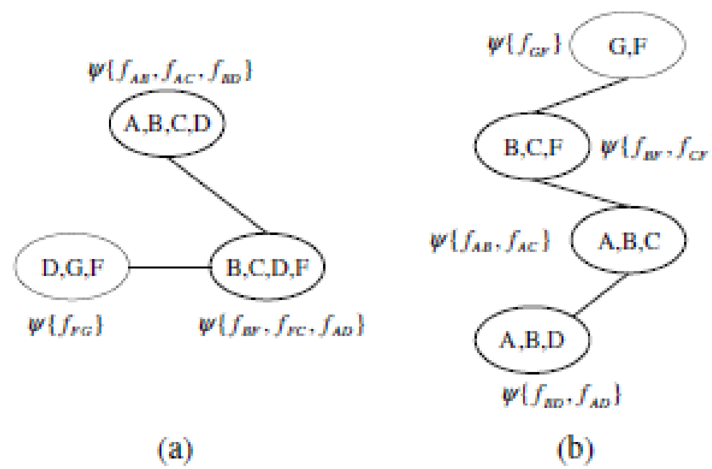
A	B	C	D	E	F
0	1	2	0	1	2
1	2	0	1	2	0
2	0	1	2	0	1
1	0	2	1	0	2
2	1	0	2	1	0
0	1	2	0	1	2
1	2	0	1	2	0
2	0	1	2	0	1
1	0	2	1	0	2
0	1	2	0	1	2
1	2	0	1	2	0
2	0	1	2	0	1
1	0	2	1	0	2
2	1	0	2	1	0
0	0	0	0	0	0
1	2	1	0	2	0
0	1	0	2	1	1
0	0	0	0	0	0
2	0	1	2	0	1
0	2	2	1	0	1

(b) Sparse Factor Table

A	B	C	D	E	F	Probability
0	1	2	0	1	2	0.20
1	2	0	1	2	0	0.20
2	0	1	2	0	1	0.20
1	0	2	1	0	2	0.15
2	1	0	2	1	0	0.10
0	0	0	0	0	0	0.10
1	2	1	0	2	0	0.05
0	1	0	2	1	1	0.05
0	2	2	1	0	1	0.05

Tree and Hypertree-Decompositions





W=3,
hw=3

W=2,
hw=2

Theorem: *A hypertree decomposition of a graphical model whose functions are sparse and bounded by t, can be processed in time and space exp(hw): O(m •deg • hw • log t •* t^{hw}*).*

Corollary: *Given graph G and data D, reasoning over empBN(G,D) is O(m • deg • hw • log t •* t^{hw} *).*

BN Hypergraphs and Hypertrees

BN hypergraphs. A BN can be associated with a dual graph or a hypergraph

$$P(V_7 \mid do(V_1)) = \sum_{V_2,V_3,V_4,V_5,V_6} P(V_6 \mid V_1,V_2,V_3,V_4,V_5) P(V_4 \mid V_1,V_2,V_3) P(V_2 \mid V_1)$$
$$\times \sum_{V_1'} P(V_7 \mid V_1',V_2,V_3,V_4,V_5,V_6) P(V_5 \mid V_1',V_2,V_3,V_4) P(V_3 \mid V_1',V_2) P(V_1')$$

.

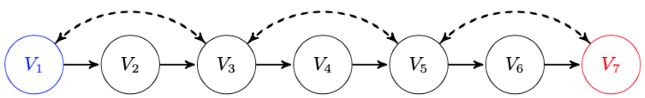
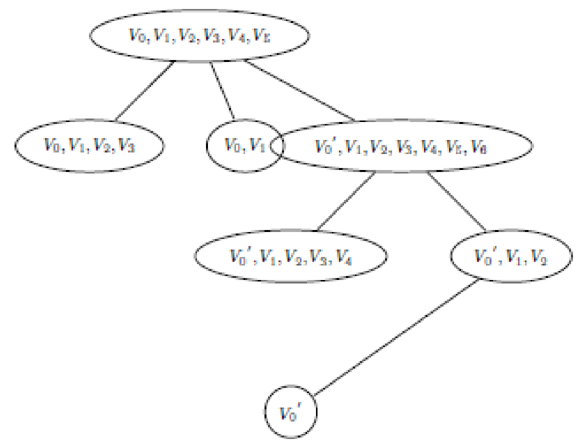


Figure 1: Chain Model with 7 observable variables and 3 latent variables



h$_{w=1,\ w=3}$ ***h***$_{w=1,\ w=5}$