# Lecture 10: February 18, 2015
# cs 573: Probabilistic Reasoning
# Professor Nevatia
# Spring 2015

# Review

- Assignment #3 due Monday Feb 25
- Exam 1: Monday, March 2
  - Material covered: up to and including Feb 25
  - Closed book, closed notes
  - Will discuss more in Feb 23 class
- Last lecture:
  - Sum-Product Variable Elimination Algorithm
    - Applies to arbitrary directed and undirected graphs
  - Induced graph => Cluster graphs and clique trees
- Today's objective
  - Clique-tree construction
  - Inference on clique trees

# Inference using Clique Trees

- Variable elimination algorithm provides distribution of one variable at a time. We can re-run multiple times to get distribution of all variables but much of the computation would be repeated.
- Working with *clique trees* will allow us to get distribution of all variables efficiently
- Topics:
  - How to convert original Bayesian/Markov networks to clique trees?
  - Inference algorithms for clique trees
  - Generalization to cluster (clique) graphs

# Cluster Graph and Clique Tree

- *Cluster Graph*
  - Graph over a set of factors $\Phi$, $\boldsymbol{X}$ is set of nodes
  - A *node* in this graph is a "cluster" (subset of) variables, say $\mathbf{C}_i$
  - **Family preserving**: Each factor $\phi$ must be associated with some cluster, say $\mathbf{C}_i$ , called $\alpha(\phi)$. scope $[\phi] \subseteq \mathbf{C}_i$
  - Each *edge* between two nodes, say $\mathbf{C}_i$ and $\mathbf{C}_j$ , is associated with a set of nodes, called a *sepset*, $\mathbf{S}_{i,j}$ , $\mathbf{S}_{i,j} \subseteq \mathbf{C}_i \cap \mathbf{C}_j$ .
- *Clique Tree* (also called a *junction tree*): cluster graph that is a tree with **Running Intersection Property**
  - Let $T$ be a cluster tree over set of factors $\Phi$
  - Let $V_T$ be vertices of $T$ and let $\varepsilon_T$ be the edges
  - If X is in $\mathbf{C}_i$ and also in $\mathbf{C}_j$ then X is also in every cluster in the path between $\mathbf{C}_i$ and $\mathbf{C}_j$ .
  - Implies $\mathbf{S}_{i,j} = \mathbf{C}_i \cap \mathbf{C}_j$ .

# Constructing a Clique Tree

- How to convert a BN/MN into a clique tree?
- Basic technique is to convert graphs to chordal graphs that are I-maps of original graphs
- Two methods to do this:
  - Use the *induced g*raph generated by VE algorithm
  - If graph is not MN, convert to MN, triangulate to make a chordal graph and then infer a clique tree
- The tree is designed to maintain the original joint distribution
  - How to assign potentials to the nodes and factors in the generated tree?

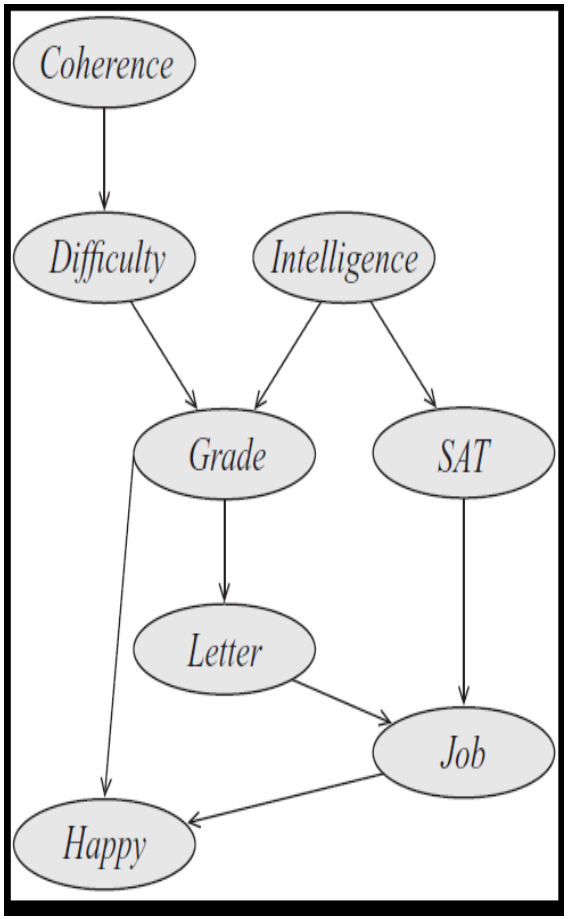
# Clique Tree from VE Induced Graph

- Induced graph:
  - Let $I_{\Phi,\prec}$ be the induced graph, $\prec$ is the elimination ordering
  - Two variables are connected in $I_{\Phi,\prec}$ if they both appear in a common factor.
- Each factor in $I_{\Phi,\prec}$, say $\psi_i$, corresponds to a cluster of variables, say $C_i$.
- Each such cluster becomes a node in the cluster graph
- Undirected edge between two clusters $C_i$ and $C_j$ if a message ($\tau_i$) is passed between them (directly) to construct $\psi_j$.
- Can be shown easily that the resulting graph is a tree
  - Each message ($\tau_i$) is used only once
- Can be shown that the resulting tree satisfies the running intersection property; hence it is a clique (junction) tree

# Theorems Associated with VE Clique Trees

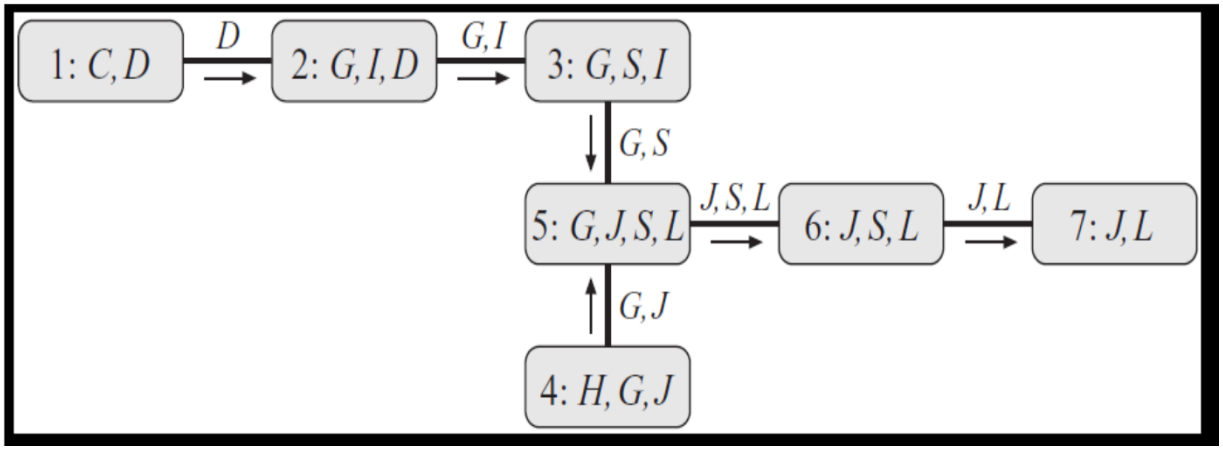
- Note: material distributed across chapters 4, 9, 10 (10.1, 10.4)
- Thm 9.6 shows that the induced graph generated by VE is necessarily chordal
- Thm 4.12 shows that a chordal graph can be always represented as a clique tree (and maintains the joint distribution)
- Thm 10.1: Tree generated by VE satisfies the running intersection property
- Thm 10.2: Tree satisfies the running intersection property *if and only if* $\mathbf{S}_{i,j}$ separates variables on "$\mathbf{C}_i$ side" of the tree from the "$\mathbf{C}_j$ side".
- Eliminate cliques that are a subset of another clique (i.e. maintain only the maximal cliques); running intersection property maintained.
- Examples: Figures 9.11, 10.1, 10.9

# Cluster Graph from VE Steps (Fig 10.1)

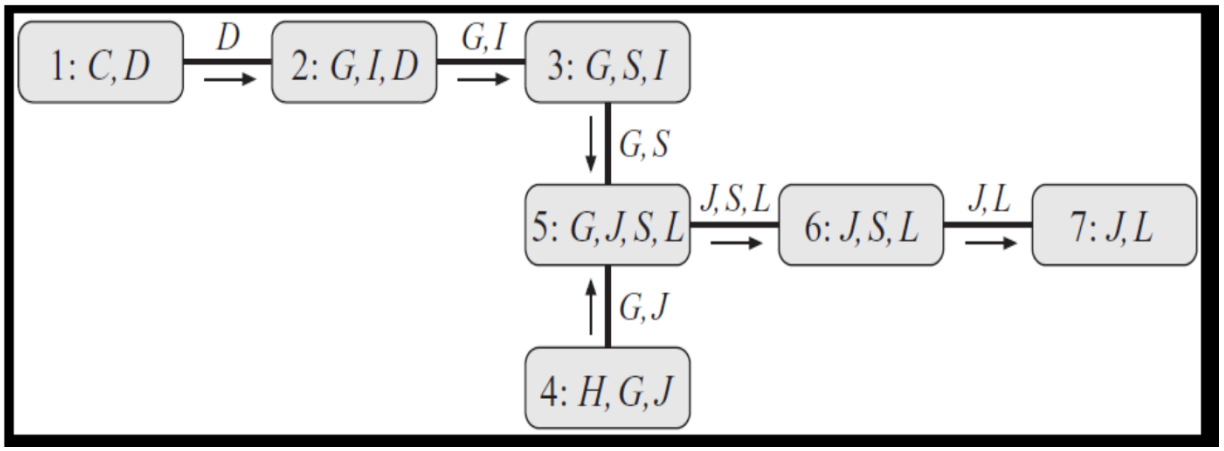



| Step | Variable eliminated | Factors used | Variables involved | New factor |
|---|---|---|---|---|
| 1 | $C$ | $\phi_C(C), \phi_D(D,C)$ | $C,D$ | $\tau_1(D)$ |
| 2 | $D$ | $\phi_G(G,I,D), \tau_1(D)$ | $G,I,D$ | $\tau_2(G,I)$ |
| 3 | $I$ | $\phi_I(I), \phi_S(S,I), \tau_2(G,I)$ | $G,S,I$ | $\tau_3(G,S)$ |
| 4 | $H$ | $\phi_H(H,G,J)$ | $H,G,J$ | $\tau_4(G,J)$ |
| 5 | $G$ | $\tau_4(G,J), \tau_3(G,S), \phi_L(L,G)$ | $G,J,L,S$ | $\tau_5(J,L,S)$ |
| 6 | $S$ | $\tau_5(J,L,S), \phi_J(J,L,S)$ | $J,L,S$ | $\tau_6(J,L)$ |
| 7 | $L$ | $\tau_6(J,L)$ | $J,L$ | $\tau_7(J)$ |

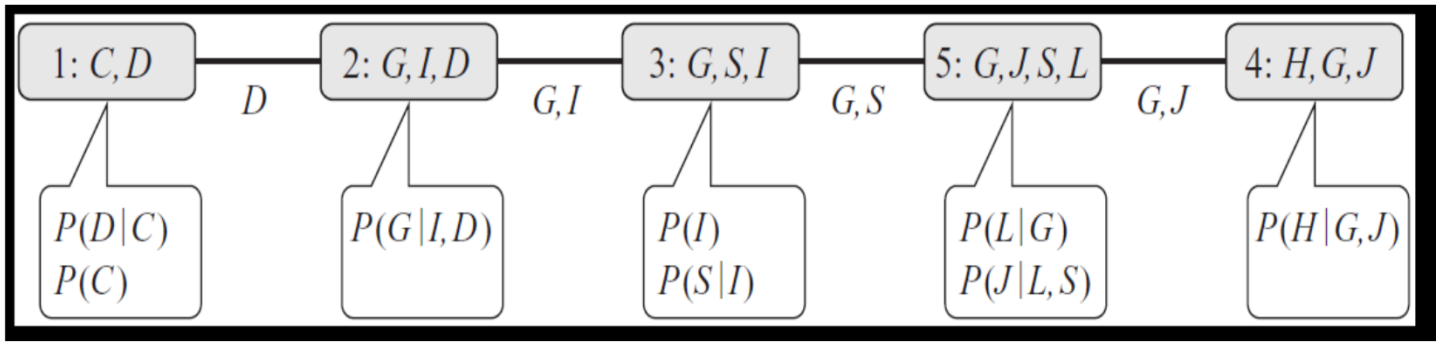
**Table 9.1 A run of variable elimination for the query $P(J)$**



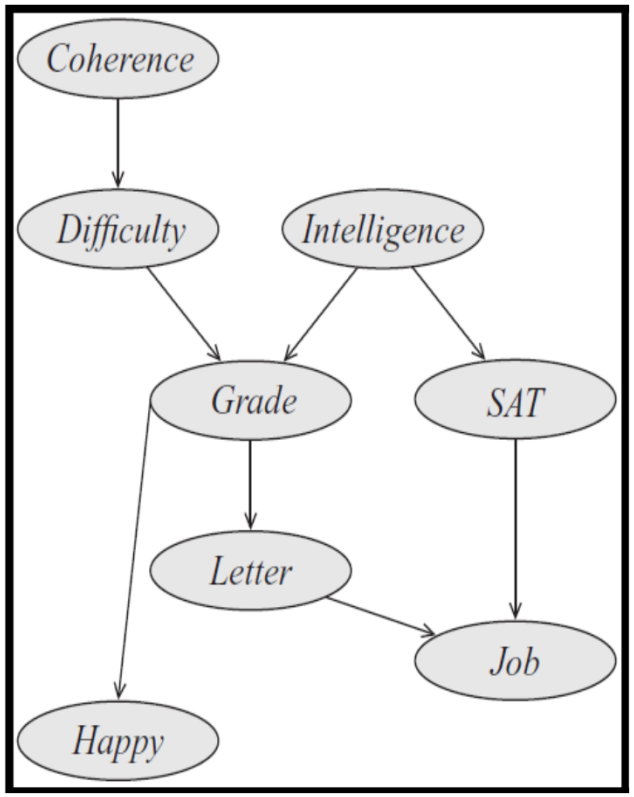
# Simplify the Cluster Tree



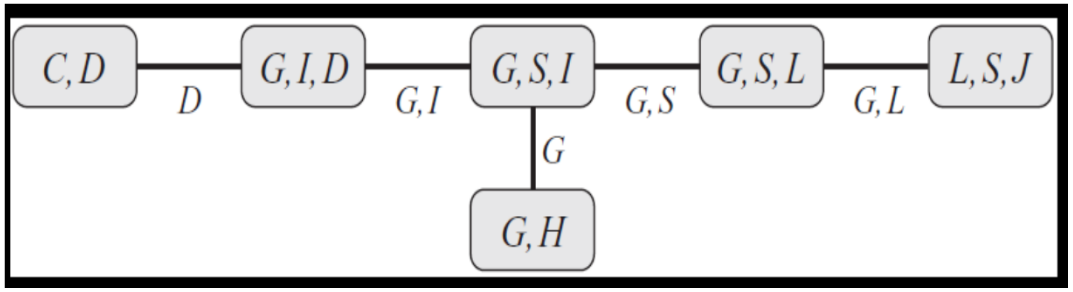

Eliminate cliques that are not maximal
(results in a chain for this example)



# Another Example (Different Graph): Fig 10.9




Steps of VE not provided; suggest students practice on their own




Not a chain this time

# Potentials in a Clique Tree

- What potentials to associate with each clique?
- Any factor that is a function of some or all variables in a clique node can be associated with this node
- Note: multiple choices for assignment of the factors to cliques may exist, *e.g.* P(I) could be in $2^{nd}$ or $3^{rd}$ cliques; either choice works (but one factor may be used in one clique only).
- Assign *initial potentials* to cliques by multiplying all the initial factors assigned to each clique.
- It should be clear that multiplication of all clique potentials yields the original distribution (same factor product)

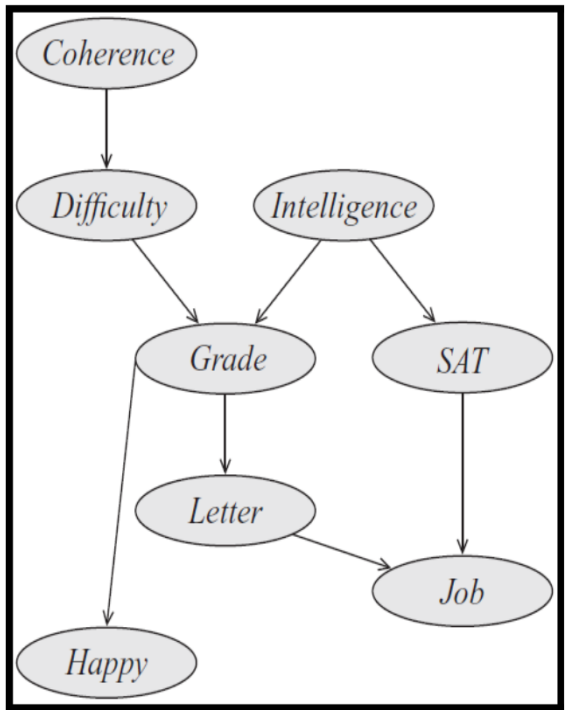
# BN/MN to Cliques

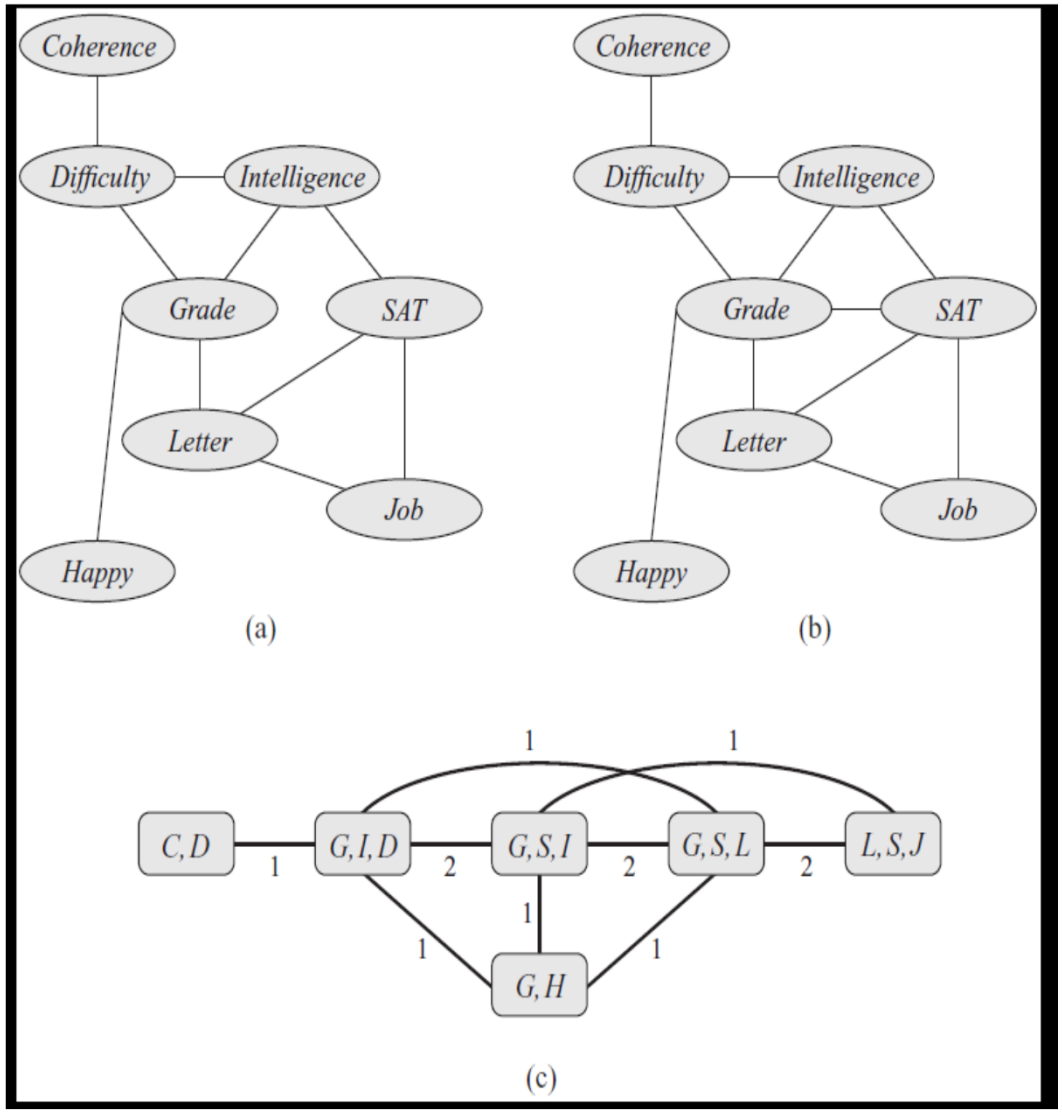
- If original graph was a directed graph, moralize it and convert to an undirected graph
- Triangulate the graph, *i.e.* make it chordal
  - Finding minimal triangulation (such that the size of the largest clique is the smallest possible) is NP-hard.
  - May use same heuristics as for VE ordering
- Find maximal cliques in the graph
  - NP-hard in general, but not for a triangulated graph
    - Maximum number of cliques is linear, not exponential in number of nodes
  - Start with any clique, add nodes until a max is achieved
  - Find node with maximum cardinality; collect corresponding cliques; repeat for remaining nodes

# Cliques to Clique Trees

- Make each maximal clique, create a node in a cluster graph
- Connect edges between nodes having common variables
  - May not result in a tree.
  - Assign weights to edges proportional to the number of shared variables
- Construct a maximal spanning tree (tree spans all nodes and sum of weights of its edges is maximal)
  - Yields the desired clique tree (with running intersect property)
    - Proof is not given in the book (Exercise 10.17).
  - Method used in HUGIN and various commercial packages

# Example


Coherence
Difficulty
Intelligence
Grade
SAT
Letter
Job
Happy


Coherence
Difficulty
Intelligence
Grade
SAT
Letter
Job
Happy
(a)
Coherence
Difficulty
Intelligence
Grade
SAT
Letter
Job
Happy
(b)
1
1
C,D
1
G,I,D
2
G,S,I
2
G,S,L
2
L,S,J
1
1
1
G,H
(c)

# Variable Elimination in a Clique Tree

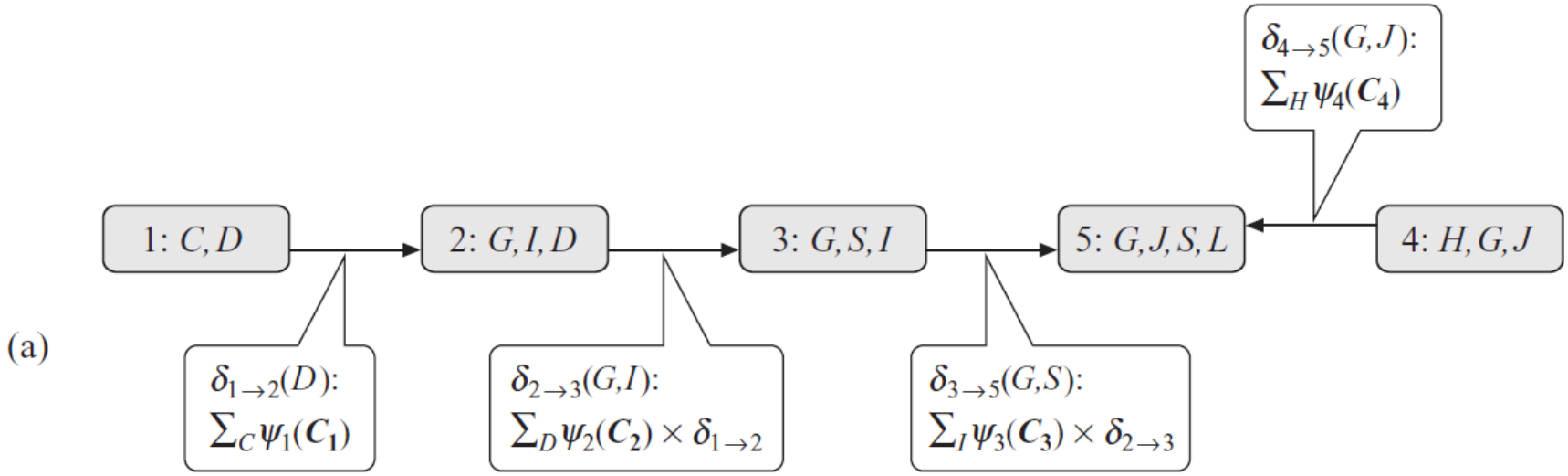
- Task: Compute P(J)
  - Choose a clique containing J as the root of the tree; all choices will give the same result.
  - Choose $C_5$ as an example

1. **In $C_1$:** We eliminate $C$ by performing $\sum_C \psi_1(C, D)$. The resulting factor has scope $D$. We send it as a message $\delta_{1\to 2}(D)$ to $C_2$.
2. **In $C_2$:** We define $\beta_2(G, I, D) = \delta_{1\to 2}(D) \cdot \psi_2(G, I, D)$. We then eliminate $D$ to get a factor over $G, I$. The resulting factor is $\delta_{2\to 3}(G, I)$, which is sent to $C_3$.
3. **In $C_3$:** We define $\beta_3(G, S, I) = \delta_{2\to 3}(G, I) \cdot \psi_3(G, S, I)$ and eliminate $I$ to get a factor over $G, S$, which is $\delta_{3\to 5}(G, S)$.
4. **In $C_4$:** We eliminate $H$ by performing $\sum_H \psi_4(H, G, J)$ and send out the resulting factor as $\delta_{4\to 5}(G, J)$ to $C_5$.
5. **In $C_5$:** We define $\beta_5(G, J, S, L) = \delta_{3\to 5}(G, S) \cdot \delta_{4\to 5}(G, J) \cdot \psi_5(G, J, S, L)$.

Sum out G,L,S to get P(J)

# Fig. 10.3 (a)

- $C_5$ selected as root; goal is to compute P(G)




(a)

# Different Root Clique

- Choose $C_4$ as root (figure (b) has $C_3$ as root)

1. **In $C_1$:** The computation and message are unchanged.
2. **In $C_2$:** The computation and message are unchanged.
3. **In $C_3$:** The computation and message are unchanged.
4. **In $C_5$:** We define $\beta_5(G, J, S, L) = \delta_{3\to 5}(G, S) \cdot \psi_5(G, J, S, L)$ and eliminate $S$ and $L$. We send out the resulting factor as $\delta_{5\to 4}(G, J)$ to $C_4$.
5. **In $C_4$:** We define $\beta_4(H, G, J) = \delta_{5\to 4}(G, S) \cdot \psi_4(H, G, J)$.

Sum out H,G to get P(J)

# Clique Tree Message Passing (Upward Pass)

- Let *T* be a clique tree, with cliques $\mathbf{C}_1$, $\mathbf{C}_2 \ldots \mathbf{C}_k$
- Set of factors $\Phi$ , each $\phi \in \Phi$ is assigned to some clique, say $\alpha(\phi)$
- Initial potential of $\mathbf{C}_j$ is given by $\psi_j(\mathbf{C}_j) = \Pi_{\alpha(\phi)=j} \phi$
- Note product of $\phi$ factors => $\psi$ factors
- Let $\mathbf{C}_r$ be the root of the clique tree
- $\mathrm{Nb}_i$ are *indices* of neighbors of $\mathbf{C}_i$
- $p_r(i)$ the upstream neighbor of $i$
- Each $\mathbf{C}_i$ the sends a message to $\mathbf{C}\, p_r(i)$
- Message from $\mathbf{C}_i$ to $\mathbf{C}_j$ is given by

$$\delta_{i \to j} = \sum_{C_i - S_{i,j}} \psi_i \cdot \prod_{k \in (\mathrm{Nb}_i - \{j\})} \delta_{k \to i}.$$

- Factor at root is denoted by $\beta_r(\mathbf{C}_r)$ then $\beta_r(C_r) = \sum_{\mathcal{X} - C_r} \tilde{P}_\Phi(\mathcal{X})$.

**Algorithm 10.1 Upward pass of variable elimination in clique tree**

**Procedure** CTree-SP-Upward (
  $\Phi$, // Set of factors
  $\mathcal{T}$, // Clique tree over $\Phi$
  $\alpha$, // Initial assignment of factors to cliques
  $C_r$ // Some selected root clique
)
1 Initialize-Cliques
2 **while** $C_r$ is not ready
3  Let $C_i$ be a ready clique
4  $\delta_{i \to p_r(i)}(S_{i,p_r(i)}) \leftarrow$ SP-Message$(i, p_r(i))$
5 $\beta_r \leftarrow \psi_r \cdot \prod_{k \in \text{Nb}_{C_r}} \delta_{k \to r}$
6 **return** $\beta_r$

**Procedure** Initialize-Cliques (

)
1 **for** each clique $C_i$
2  $\psi_i(C_i) \leftarrow \prod_{\phi_j \,:\, \alpha(\phi_j)=i} \phi_j$
3

**Procedure** SP-Message (
  i, // sending clique
  j // receiving clique
)
1 $\psi(C_i) \leftarrow \psi_i \cdot \prod_{k \in (\text{Nb}_i - \{j\})} \delta_{k \to i}$
2 $\tau(S_{i,j}) \leftarrow \sum_{C_i - S_{i,j}} \psi(C_i)$
3 **return** $\tau(S_{i,j})$

# Correctness

- Shown that the algorithm computes the desired expressions
- Proposition 10.2
  - Assume that X is eliminated when a message is sent from $\mathbf{C}_i$ to $\mathbf{C}_j$, then X does not appear anywhere in the tree on the $C_j$ side of (i-j)
  - Follows directly from the running intersection property
- Theorem 10.3:

  Notation: $\mathcal{F}_{\prec(i\rightarrow j)}$ Set of factors on the $\mathbf{C}_i$ side of the edge;

  $\mathcal{V}_{\prec(i_k\rightarrow i)}$ Set of variables on the $\mathbf{C}_i$ side of the edge but not in $S_{ij}$

*Let $\delta_{i\rightarrow j}$ be a message from $C_i$ to $C_j$. Then:*

$$\delta_{i\rightarrow j}(S_{i,j}) = \sum_{\mathcal{V}_{\prec(i\rightarrow j)}} \prod_{\phi\in\mathcal{F}_{\prec(i\rightarrow j)}} \phi.$$

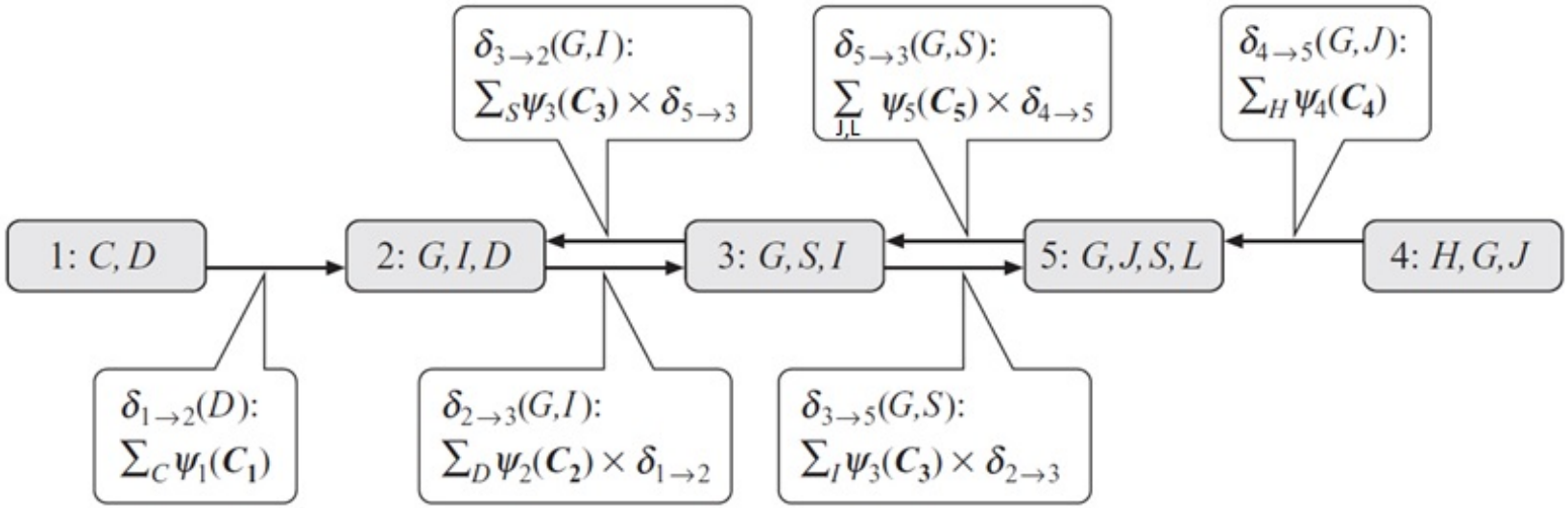
- Corollary 10.1 gives: $\beta_r(C_r) = \sum_{\mathcal{X}-C_r} \tilde{P}_\Phi(\mathcal{X})$.

# Clique Tree Calibration

- Compute probability of every non-evidence variable
- Run once for each clique, making it the root, cost is $K$ x $c$ (K is number of cliques, $c$ is cost of one upward pass)
- Example of Fig 10.2, consider roots to be $\mathbf{C}_5$ , $\mathbf{C}_4$ , and $\mathbf{C}_3$
  - Messages from $\mathbf{C}_1$ to $\mathbf{C}_2$ and $\mathbf{C}_2$ to $\mathbf{C}_3$ are same in all three cases but no message is sent from $\mathbf{C}_4$ to $\mathbf{C}_5$ when $\mathbf{C}_4$ is the root (message goes from $\mathbf{C}_5$ to $\mathbf{C}_4$ )
  - In general, there may be many common computations, we do not need to repeat them.
- In general, message sent from $\mathbf{C}_i$ to $\mathbf{C}_j$ does not depend on the root (but some messages may not be sent depending on the root).
  - Two messages associated with each edge so we only need to compute $2(c\text{-}1)$ messages at most, c is the number of cliques.
- Fig 10.5 shows some examples of both upward and downward pass messages

# Fig 10.5 (b)

Showing some steps of upward and downward messages



# Message Passing

- *Ready* Clique: clique is ready to transmit a message
- $\mathbf{C}_i$ is *ready* to transmit to neighbor $\mathbf{C}_j$ when $\mathbf{C}_i$ has messages from all its neighbors except $\mathbf{C}_j$ .
  - Can then compute $\delta_{i \to \mathbf{j}}(S_{i,j})$ by multiplying its *initial* potential with all incoming messages and eliminating the variables not in $S_{i,j}$ .
- Scheduling can be systematic: an upward and then a downward pass or be *asynchronous* with each clique sending a message whenever it is ready. Complexity is the same in either case.
- Compute $\beta_i$ by multiplying incoming messages with initial potential; would be same as computation in upward pass with root at $\mathbf{C}_i$ .
- Corollary 10.2: Applying algorithm 10.2 yields

$$\beta_i(C_i) = \sum_{\mathcal{X} - \boldsymbol{C}_i} \tilde{P}_\Phi(\mathcal{X}).$$

# Clique Tree Calibration Algorithm (10.2)

**Algorithm 10.2 Calibration using sum-product message passing in a clique tree**

**Procedure** CTree-SP-Calibrate (
  $\Phi$, // Set of factors
  $\mathcal{T}$ // Clique tree over $\Phi$
)

1 Initialize-Cliques
2 **while** exist $i, j$ such that $i$ is ready to transmit to $j$
3 $\delta_{i \to j}(\boldsymbol{S}_{i,j}) \leftarrow$ SP-Message$(i, j)$
4 **for** each clique $i$
5 $\beta_i \leftarrow \psi_i \cdot \prod_{k \in \text{Nb}_i} \delta_{k \to i}$
6 **return** $\{\beta_i\}$

# Calibration: Defnitions

- **Calibrated pair** *of* adjacent cliques

*Two adjacent cliques* $C_i$ *and* $C_j$ *are said to be* calibrated *if*

$$\sum_{C_i - S_{i,j}} \beta_i(C_i) = \sum_{C_j - S_{i,j}} \beta_j(C_j).$$

- **Tree is calibrated** if all adjacent pairs of cliques are calibrated.
  - $\beta_i$ ($\mathbf{C}_i$) is called the *clique belief*
- *Sepset belief* is given by:

$$\mu_{i,j}(S_{i,j}) = \sum_{C_i - S_{i,j}} \beta_i(C_i) = \sum_{C_j - S_{i,j}} \beta_j(C_j).$$

# Next Class

- Read sections 10.3, 11.3.1 to 11.3.5