Exact Inference: Variable Elimination and Clique Trees

Machine Learning: Data to Models

Prof. Suchi Saria

Readings:
See Murphy Chapter 20.3 for a summary of the Variable Elimination.
See K&F 10.2.2 for clique tree inference

See Koller and Friedman Chapter 10 for a more rigorous treatment with correctness proofs.

Outline

- Goal: Make inference efficient
  - Trick: Identify a new data structure where we can cache computations.
  - Intuitively, identify the operations that are taking place in variable elimination to motivate clique trees.

- What is a clique tree (special case of a cluster graph)
  - Family preservation property
  - Running intersection property

- Message passing on clique trees
  - Sum product or Belief Update [equivalent algorithms]

- Creating clique trees
  - Graph created by VE is a click tree.
    i.e. satisfies family preservation and running intersection prop.
  - Alternate (more general) procedure discussed in HW.

Student Example (BN)

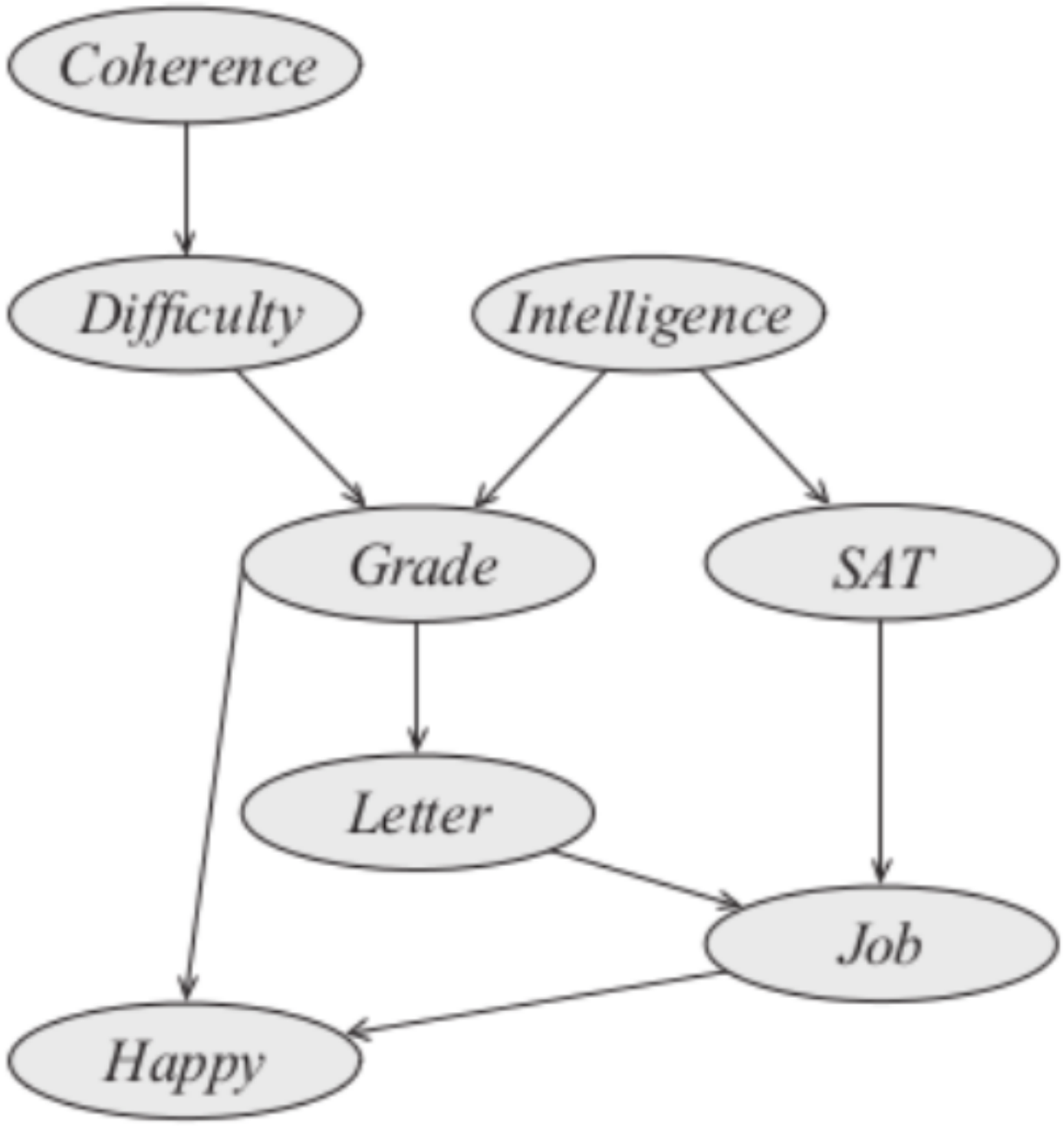


Figure : Student Example

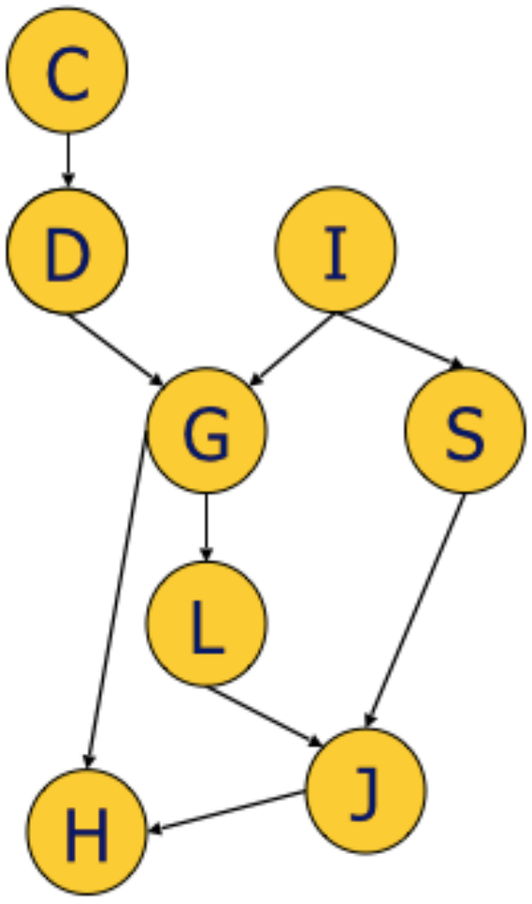
Query

Let's denote the nodes in the graph with their first letters. Now we have C, D, I, G, S, L, J, H.
What are the probability of $P(J)$ for different values of J?
The naive method will be compute the joint distribution of C, D, I, G, S, L, J, H and sum out J, which is computationally expensive.
We can use variable elimination based on the distribution rule in mathematics.

Variable Elimination

- Query variable can be anything.
- Different elimination orderings are equally valid.

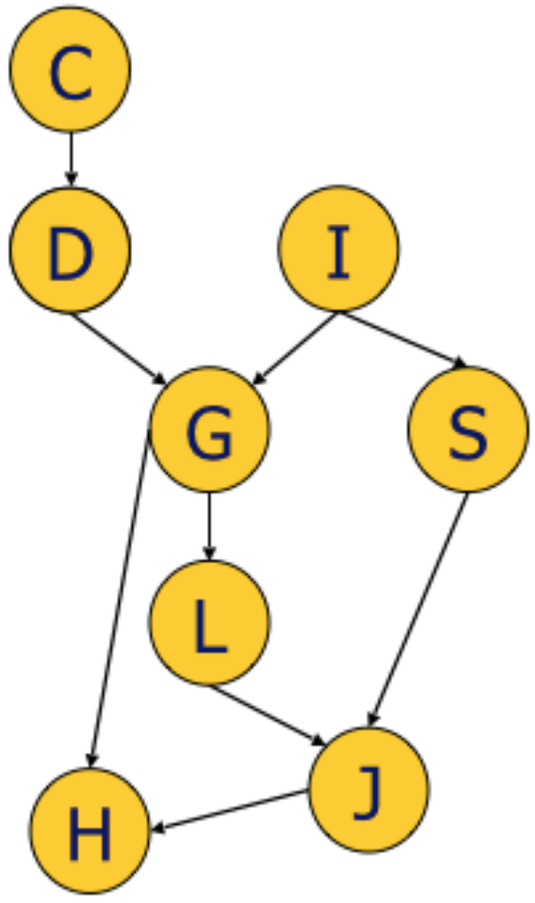
$$P(J) = \sum_{L,S,G,H,I,D,C} P(C, D, I, H, G, S, L, J)$$



Variable Elimination

- Query variable can be anything.
- Different elimination orderings are equally valid.

$$P(J) = \sum_{L,S,G,H,I,D,C} P(C, D, I, H, G, S, L, J)$$



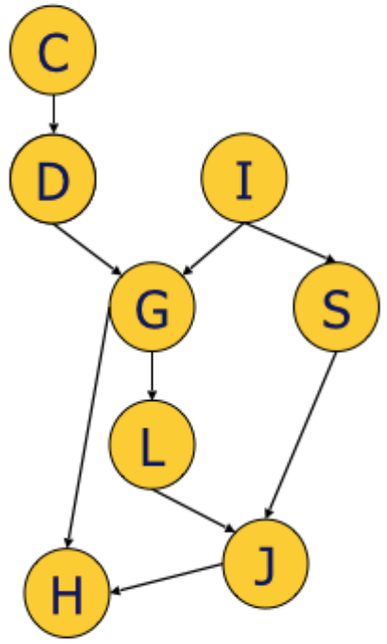
$$= \Sigma_L \Sigma_S \Sigma_G \Sigma_I \Sigma_D \Sigma_C P(J|L,S) P(H|G,J) P(L|G) P(I) P(S|I) P(G|D,I) P(D|C) P(C)$$

$$= \Sigma_L \Sigma_S P(J|L,S) \Sigma_G P(H|G,J) P(L|G) \Sigma_I P(I) P(S|I) \Sigma_D P(G|D,I) \Sigma_C P(D|C) P(C)$$

Variable Elimination

- Query variable can be anything.
- Different elimination orderings are equally valid.

$$P(J) = \sum_{L,S,G,H,I,D,C} P(C, D, I, H, G, S, L, J)$$



$$= \Sigma_L \Sigma_S \Sigma_G \Sigma_I \Sigma_D \Sigma_C P(J|L,S) P(H|G,J) P(L|G) P(I) P(S|I) P(G|D,I) P(D|C) P(C)$$

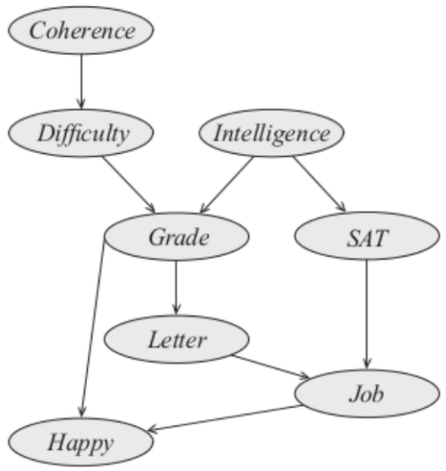
$$= \Sigma_L \Sigma_S P(J|L,S) \Sigma_G P(H|G,J) P(L|G) \Sigma_I P(I) P(S|I) \Sigma_D P(G|D,I) \Sigma_C P(D|C) P(C)$$

What are the operations here?

Variable Elimination

$$\Sigma_L \Sigma_S P(J|L,S) \Sigma_G P(H|G,J) P(L|G) \Sigma_I P(I) P(S|I) \Sigma_D P(G|D,I) \Sigma_C P(D|C) P(C)$$

$$\psi(J) = \sum_{L,S} \psi(J,L,S) \sum_{G} \psi(L,G) \sum_{H} \psi(H,G,J)$$
$$\times \sum_{I} \psi(S,I)\psi(I) \sum_{D} \psi(G,I,D)$$
$$\times \sum_{C} \psi(D,C)\psi(C)$$



Be careful of the notation, what we actually do is:

1. compute the summation from $\sum_C \psi(D,C)\psi(C)$ and this is called **factor marginalization**
2. compute $\sum_D \psi(G,I,D)(\sum_C \psi(D,C)\psi(C))$
3. do the product-sum layer by layer like peeling an onion and the reason is simply the distribution law

Factor Marginalization

Definition

Let $\mathbf{X}$ be a set of variables, and $Y \notin \mathbf{X}$ a variable. Let $\phi(\mathbf{X}, Y)$ be a factor. We define the factor marginalization of Y in ϕ, to be a factor ψ over $\mathbf{X}$ such that:

$$\psi(\mathbf{X}) = \sum_{Y} \phi(\mathbf{X}, Y)$$

This operation is called summing out of Y in ψ.

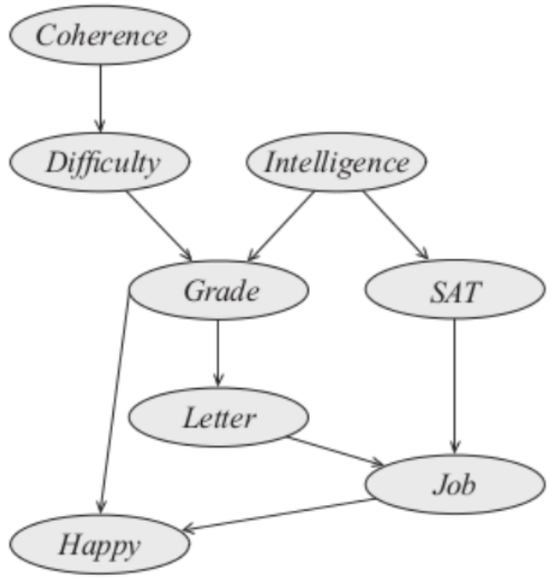
Commutative/Associative/Distribution Law

1. Commutative: $\phi_1 \cdot \phi_2 = \phi_2 \cdot \phi_1$
2. Associative: $(\phi_1 \cdot \phi_2) \cdot \phi_3 = \phi_1 \cdot (\phi_2 \cdot \phi_3)$
3. Distribution: $X \notin \mathit{Scope}[\phi_1]$, $\sum_X (\phi_1 \cdot \phi_2) = \phi_1 \cdot \sum_X \phi_2$

Variable Elimination

$$\Sigma_L \Sigma_S P(J|L,S) \Sigma_G P(L|G) \Sigma_H P(H|G,J) \Sigma_I P(I) P(S|I) \Sigma_D P(G|D,I) \Sigma_C P(D|C) P(C)$$

$$\psi(J) = \sum_{L,S} \psi(J,L,S) \sum_{G} \psi(L,G) \sum_{H} \psi(H,G,J)$$
$$\times \sum_{I} \psi(S,I)\psi(I) \sum_{D} \psi(G,I,D)$$
$$\times \sum_{C} \psi(D,C)\psi(C)$$



Be careful of the notation, what we actually do is:

1. compute the summation from $\sum_C \psi(D,C)\psi(C)$ and this is called **factor marginalization**
2. compute $\sum_D \psi(G,I,D)(\sum_C \psi(D,C)\psi(C))$
3. do the product-sum layer by layer like peeling an onion and the reason is simply the distribution law

Complexity

Suppose we have n variables $X_1, \ldots, X_n$, each variable can take on k values, we want to compute $P(X_1)$.
The naive method has complexity $O(k^n)$.
Variable elimination has complexity $O(\sum_{c \in C(X_1,\ldots,X_n)} k^{|c|})$ where C is the clique graph and $|c|$ is the number in each clique.
However, find an elimination ordering that minimize the total cost is NP-hard.

Variable Elimination and Clique Tree

$$P(J) = \sum_L \sum_S P(J \mid L,S) \sum_G P(L \mid G) \sum_H P(H \mid G,J) \sum_I P(I) P(S \mid I) \sum_D P(G \mid D,I) \sum_C P(C) P(D \mid C)$$

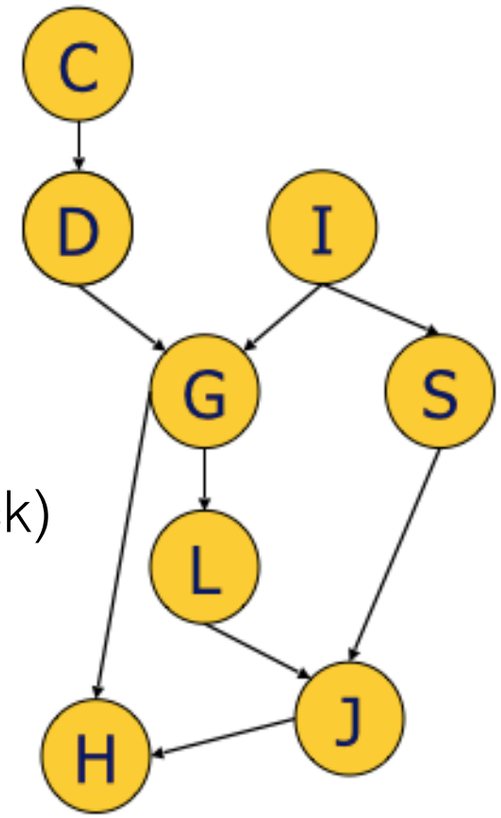
$$P(J) = \sum_{L,S} \phi_J(J,L,S) \sum_G \phi_L(L,G) \sum_H \phi_H(H,G,J) \sum_I \phi_I(I) \phi_S(S,I) \sum_D \phi_G(G,I,D) \sum_C \phi_D(C,D) \phi_C(C)$$

In variable elimination, each computation creates a factor ψ_i by multiplying existing factors. A variable is then eliminated in ψ_i to generate a new factor τ_i, which is then used to create another factor.

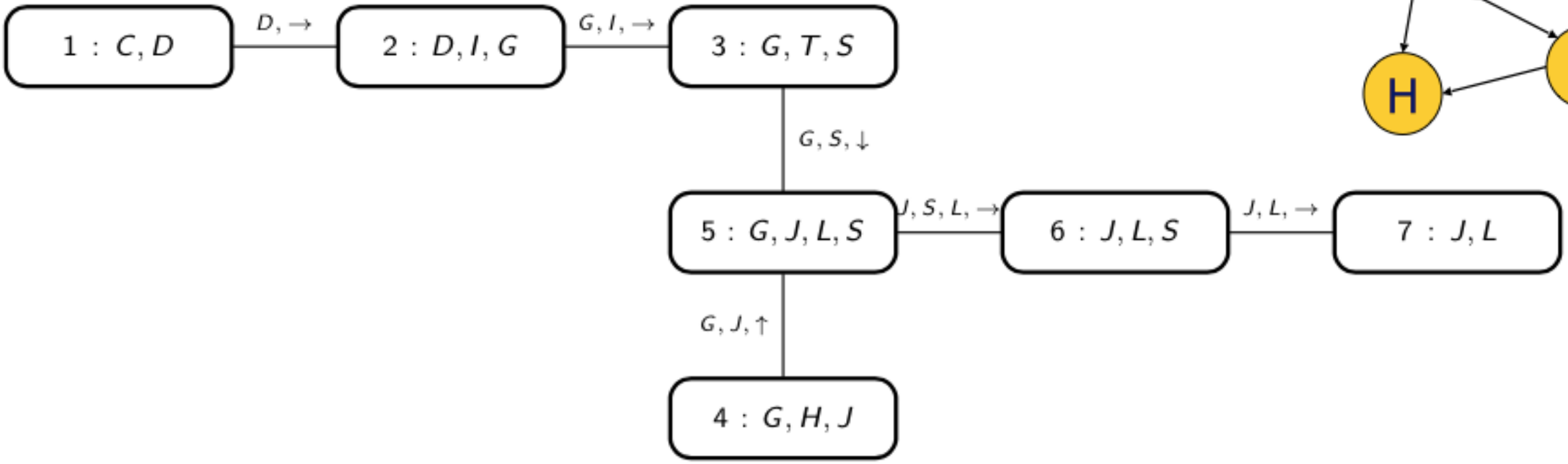
From the perspective of clique tree, we consider a factor ψ_i to be a computational data structure, which takes "message" τ_j generated by other factors ψ_j, and generated a message τ_i that is used by another factor ψ_l.

$$P(J) = \sum_{L,S} \sum_{G} P(J \mid L,S) \sum_{H} P(L \mid G) \sum_{I} P(H \mid G,J) \sum_{D} P(I) P(S \mid I) \sum_{C} P(G \mid D,I) P(C) P(D \mid C)$$

$$P(J) = \sum_{L,S} \phi_J(J,L,S) \sum_{G} \phi_L(L,G) \sum_{H} \phi_H(H,G,J) \sum_{I} \phi_I(I) \phi_S(S,I) \sum_{D} \phi_G(G,I,D) \sum_{C} \phi_D(C,D) \phi_C(C)$$



Typo: T should be I in cluster 3 of this figure (throughout this slide deck)



Example of a cluster graph generated from variable elimination in the extended student network.

Properties of Cluster Graphs for Variable Elimination

See Koller and Friedman chapter 10 for detailed explanations and proofs.

- The cluster graph induced by an execution of variable elimination is necessarily a tree.
- The cluster graph is defined to be undirected but variable elimination does define a direction for the edges, as induced by the flow of messages between the clusters.
- The directed graph induced by the messages is a directed tree, with all the messages flowing toward a single cluster where the final result is computed. This cluster is called the **root** of the directed tree.
- If C_i is on the path of C_j to the root, then C_i is the **upstream** from C_j and C_j is **downstream** from C_i.

Cluster Graphs

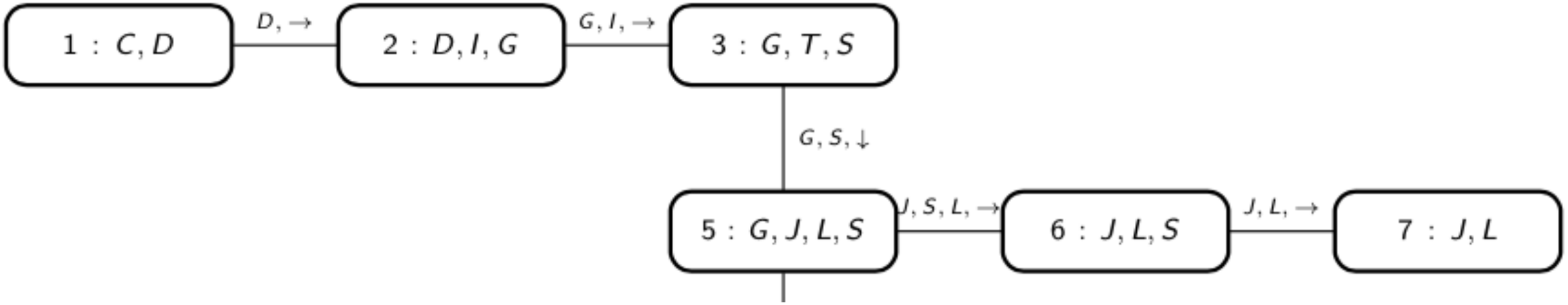
Definition

A **cluster graph** $\mathcal{U}$ for a set of factors Φ over $\mathcal{X}$ is an undirected graph, each of whose nodes i is associated with a subset $C_i \subseteq \mathcal{X}$. A cluster graph must be **family-preserving** — each factor $\phi \in \Phi$ must be associated with a cluster C_i, denoted $\alpha(\phi)$, such that $Scope[\phi] \subseteq C_i$. Each edge between a pair of clusters C_i and C_j is associated with a sepset (separation set) $S_{i,j} \subseteq C_i \cap C_j$.

Definition

Let $\mathcal{T}$ be a cluster tree over a set of factors Φ. We denote by $\mathcal{V}_{\mathcal{T}}$ the vertices of $\mathcal{T}$ and by $\varepsilon_{\mathcal{T}}$ its edges. We say that $\mathcal{T}$ has the **running intersection property** if, whenever there is a variable X such that $X \in C_i$ and $X \in C_j$, then X is also in every cluster in the (unique) path in $\mathcal{T}$ between C_i and C_j.

Note that running intersection property implies that $S_{i,j} = C_i \cap C_j$.



Definition

Let Φ be a set of factors over $\mathcal{X}$. A cluster tree over Φ that satisfies the running intersection property is called a **clique tree** (sometimes also called a junction tree or a join tree). In the case of a clique tree, the clusters are also called cliques.

Running Intersection Property

Theorem

Let $\mathcal{T}$ be a cluster tree induced by a variable elimination algorithm over some set of factors Φ. Then $\mathcal{T}$ satisfies the running intersection property.

See proof in Koller and Friedman pg 347 (Theorem 10.1)

Proposition

Let $\mathcal{T}$ be a cluster tree induced by a variable elimination algorithm over some set of factors Φ. Let C_i and C_j be two neighboring clusters, such that C_i passes the message τ_i to C_j. Then the scope of the message τ_i is precisely $C_i \cap C_j$.

Massage Passing

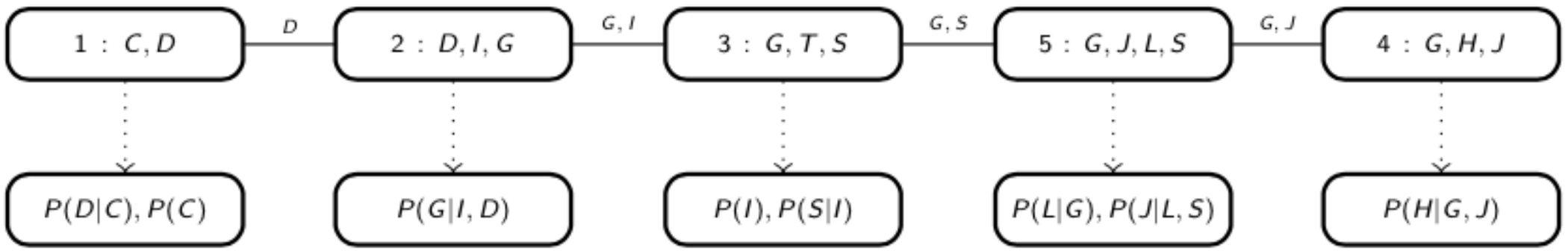


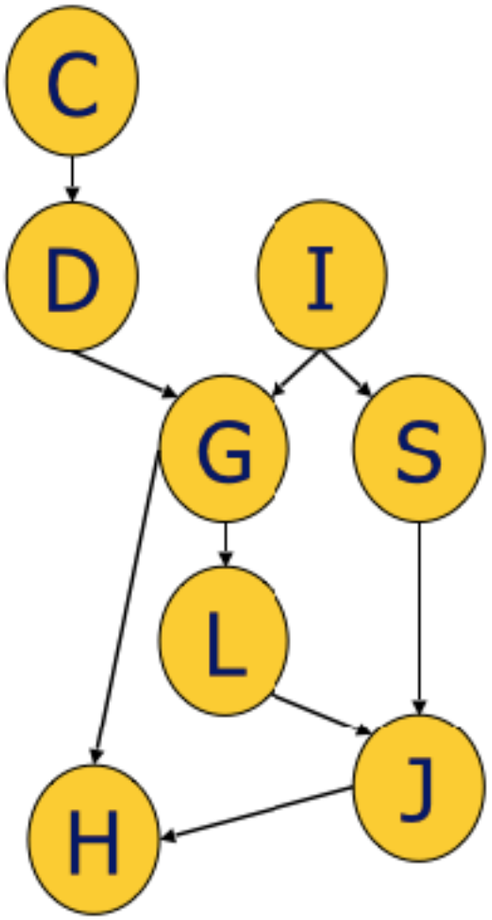
Figure : Simplified clique tree $\mathcal{T}$ for the extended student network

Another valid clique tree for the student example. In particular, the non-maximal cliques (C6 over {J,L,S} and C7 over {J,L}) are absent . See Koller and Friedman 10.2.1.1.

Clique Tree Inference

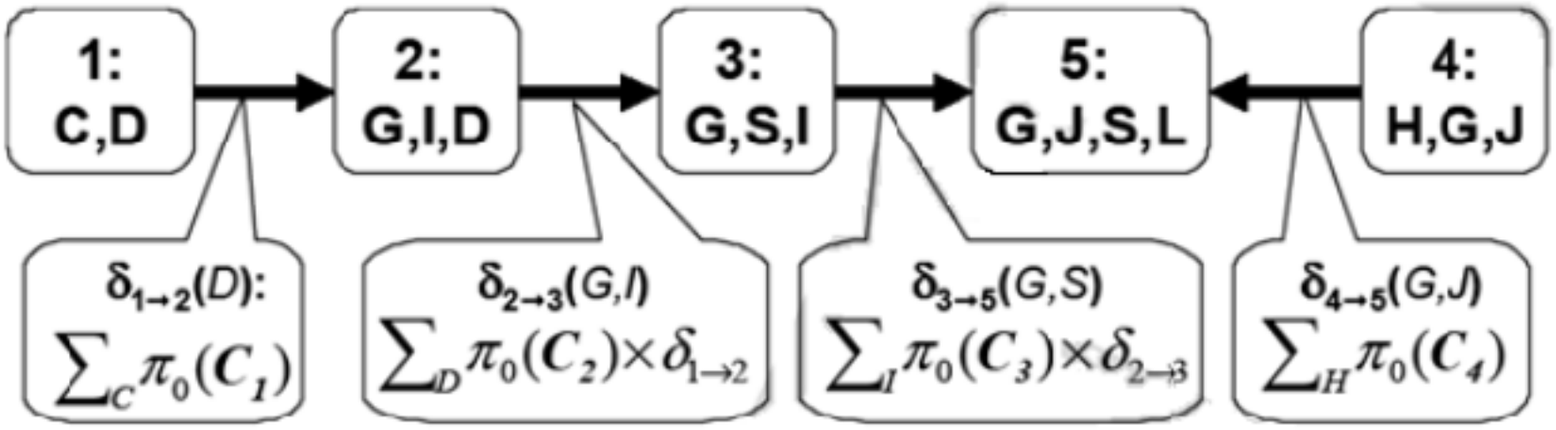
K&F 10.2.1.1

Task: Compute P(J)

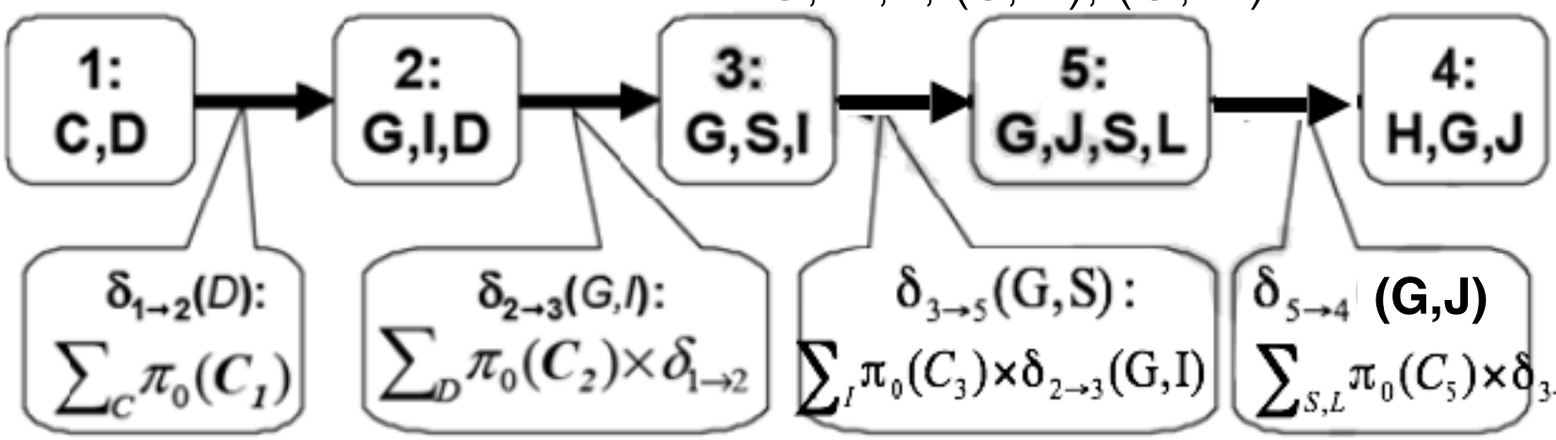


C5 as the root

C, D, I, H, (G, S, L)

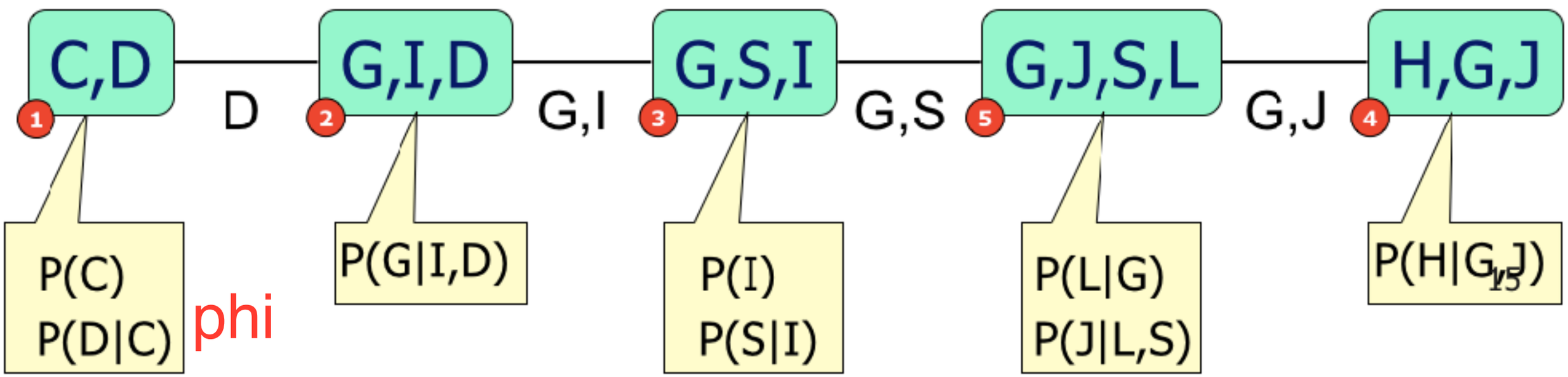


C, D, I, (S, L), (G, H)



psi

C4 as the root



phi

Clique-tree Message Passing

Let $\mathcal{T}$ be a clique tree which cliques $C_1, \cdots, C_k$.

We define the **initial clique potential** of C_j as:

$$\psi_j(C_j) = \prod_{\phi:\alpha(\phi)=j} \phi$$

As each factor is assigned to exactly one clique, we have:

$$\prod_{\phi} \phi = \prod_{j} \psi_j$$

Clique Tree Message Passing

Define C_r be the root. For each clique C_i, let NB_i be the set of indices of clique that are neighbors of C_i. Let $p_r(i)$ be the upstream neighbor, i.e. the next one in the path to the root.

For each clique C_i , the message is computed by multiplying incoming messages from its downstream neighbors with its initial clique potential, resulting in a factor which scope is the clique.

We sum out all the variables except those in the sepset between C_i and $C_{p_r(i)}$, and sends message to its upstream neighbor $C_{p_r(i)}$.

Mathematically, the message from C_i to C_j is computed as:

$$\delta_{i \to j} = \sum_{C_i - S_{i,j}} \psi_i \prod_{k \in (Nb_i - \{j\})} \delta_{k \to i}$$

Clique Tree Message Passing

This message passing process proceeds up the tree, culminating at the root clique. When the root clique has received all messages, it multiplies them with its own initial potential. The result is a factor called the beliefs, denoted $\beta_r(C_r)$:

$$\beta_r(C_r) = \tilde{P}_\Phi(C_r) = \sum_{\mathcal{X}-C_r} \prod_\phi \phi$$

This represents the unnormalized marginal probability on the clique C_r.

Summary thus far

- Goal: Make inference efficient
  - Trick: Identify a new data structure where we can cache computations.
  - Intuitively, identify the operations that are taking place in variable elimination to motivate clique trees.

- What is a clique tree (special case of a cluster graph)
  - Family preservation property
  - Running intersection property

- Message passing on clique trees
  - Sum product
  - Belief Update [equivalent algorithms] (K&F 10.3)

- General Procedure for creating clique trees (K&F 10.4)
  - You'll see an example in HW2

- We have not discussed correctness
  - Why does message passing on a clique tree yield the correct marginals?

- More than one query

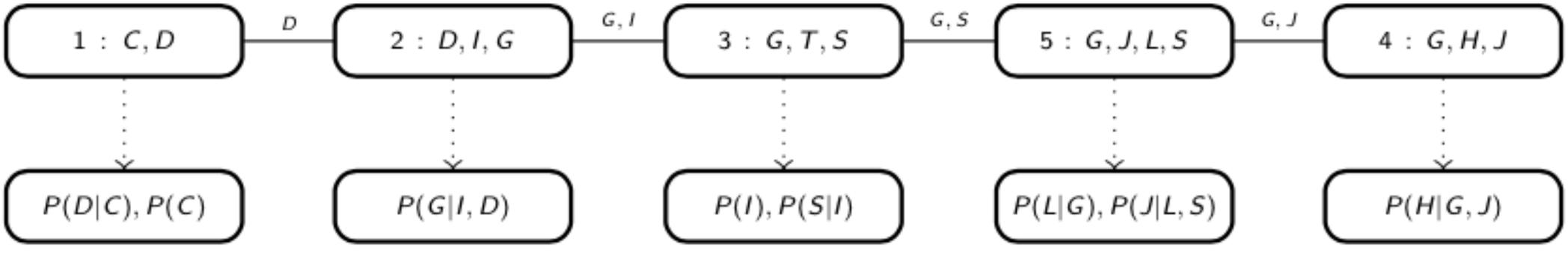
Correctness

let $(i - j)$ be some edge in the clique tree. We use $\mathcal{F}_{<(i \to j)}$ to denote the set of factors in the cliques on the C_i-side of the edge and $\mathcal{V}_{<(i \to j)}$ to denote the set of variables that appear on the C_i-side but are not in the sepset.

Using the simplified clique tree before,

$\mathcal{F}_{<(3 \to 5)} = \{P(C), P(D|C), P(G|I, D), P(I), P(S|I)\}$

$\mathcal{V}_{<(3 \to 5)} = \{C, D, I\}$.



Clique Tree and Separation

Running intersection property implies independence.

Let $\mathcal{T}$ be a cluster tree over Φ, and let $\mathcal{H}_\Phi$ be the undirected graph associated with this set of factors. Let $W_{<(i,j)}$ be the set of all variables in the scope of clusters in the C_i side of the tree, and $W_{<(j,i)}$ be the set of all variables in the scope of clusters in the C_j side of the tree.

Theorem

$\mathcal{T}$ satisfies the running intersection property if and only if, for every sepset $S_{i,j}$, we have that $W_{<(i,j)}$ and $W_{<(j,i)}$ are separated in $\mathcal{H}_\Phi$ given $S_{i,j}$.

Correctness

Proposition

Assume that X is eliminated when a message is sent from C_i to C_j. Then X does not appear anywhere in the tree on the C_j side of the edge $(i - j)$.

Theorem

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

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

$$P(Y) = \sum_{C_r - Y} \phi_r \prod_{k \in Nb_r} \delta_{k \to r}$$

Theorem

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

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

Proof: The proof proceeds by induction on the length of the path from the leaves. For the base case, the clique C_i is a leaf in the tree and the result follows from a simple elimination of the operation executed at the clique.

Now consider a clique C_i that is not a leaf, and consider the expression $\sum_{\mathcal{V}_{<(i\to j)}} \prod_{\phi\in\mathcal{F}_{<(i\to j)}} \phi$. Let $i_1, \cdots, i_m$ be the neighboring cliques of C_i other than C_j. It follows immediately from the proposition that $\mathcal{V}_{<(i\to j)}$ is a disjoint union of $\mathcal{V}_{<(i_k\to i)}$ and $\mathcal{F}_{<(i\to j)}$ is a disjoint union of $\mathcal{F}_{<(i_k\to i)}$ for $k = 1, \cdots, m$.

Proof of Correctness

$$\sum_{\mathcal{V}_{<(i\to j)}} \prod_{\phi\in\mathcal{F}_{<(i\to j)}} \phi$$

$$= \sum_{C_i - S_{i,j}} \sum_{\mathcal{V}_{<(i_1\to i)}} \cdots \sum_{\mathcal{V}_{<(i_m\to i)}} \left(\prod_{\phi\in\mathcal{F}_{<(i_1\to i)}} \phi\right) \cdots \left(\prod_{\phi\in\mathcal{F}_{<(i_m\to i)}} \phi\right) \left(\prod_{\phi\in\mathcal{F}_i} \phi\right)$$

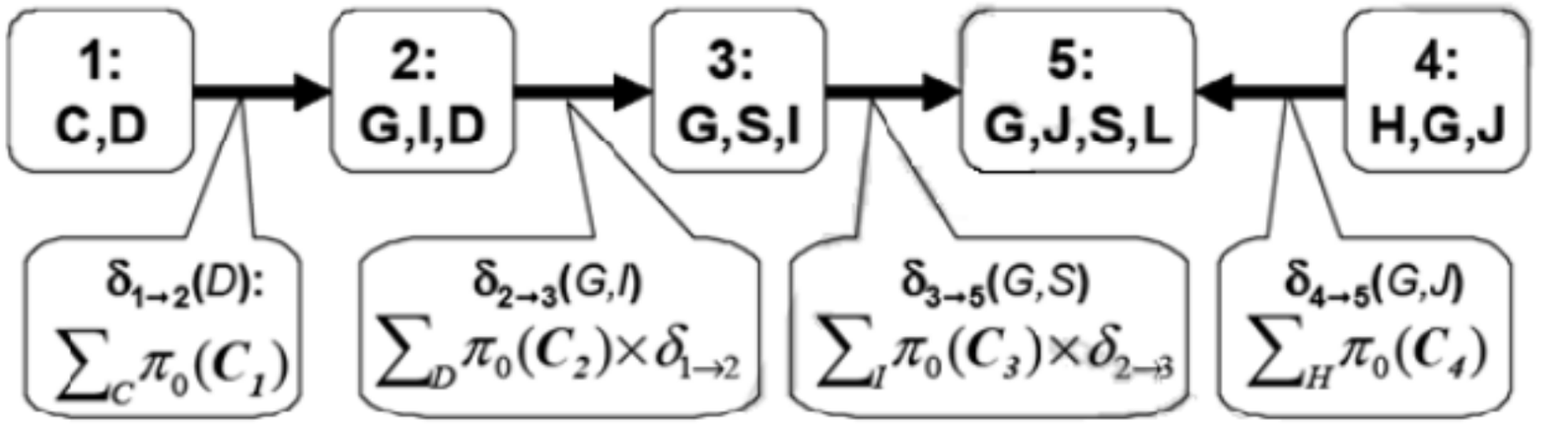
$$= \sum_{C_i - S_{i,j}} \left(\prod_{\phi\in\mathcal{F}_i} \phi\right) \sum_{\mathcal{V}_{<(i_1\to i)}} \left(\prod_{\phi\in\mathcal{F}_{<(i_1\to i)}} \phi\right) \cdots \sum_{\mathcal{V}_{<(i_m\to i)}} \left(\prod_{\phi\in\mathcal{F}_{<(i_m\to i)}} \phi\right)$$

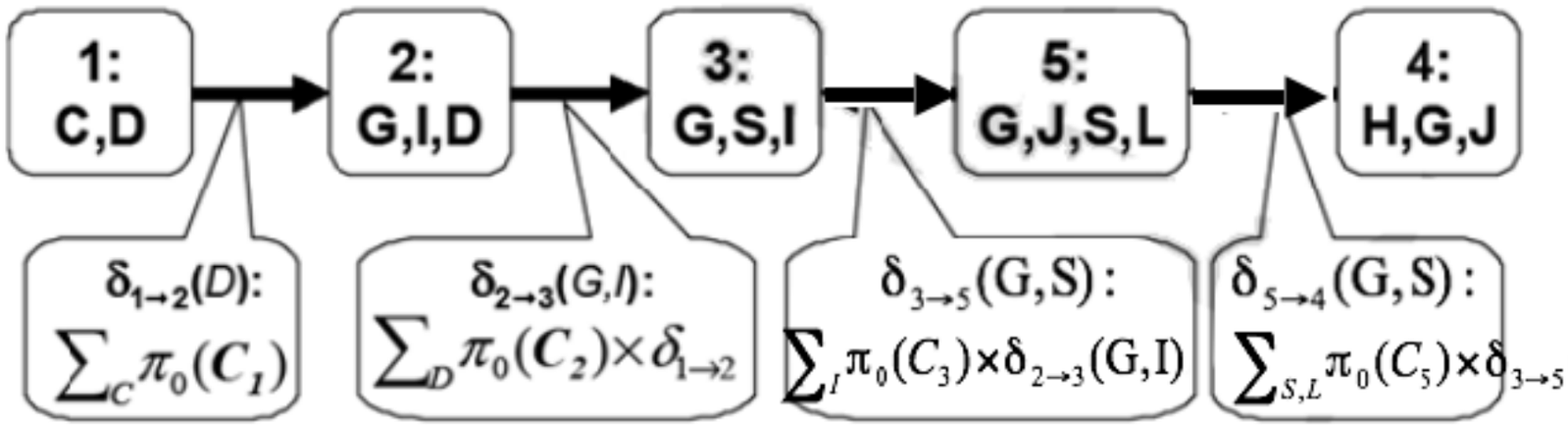
$$= \sum_{C_i - S_{i,j}} \psi_i(\delta_{i_1\to i} \ldots \delta_{i_m\to i})$$

This is the precisely the operation used to compute the message $\delta_{i\to j}$ and we will arrive at the beliefs which represent the unnormalized mariginal probability on the cliques.

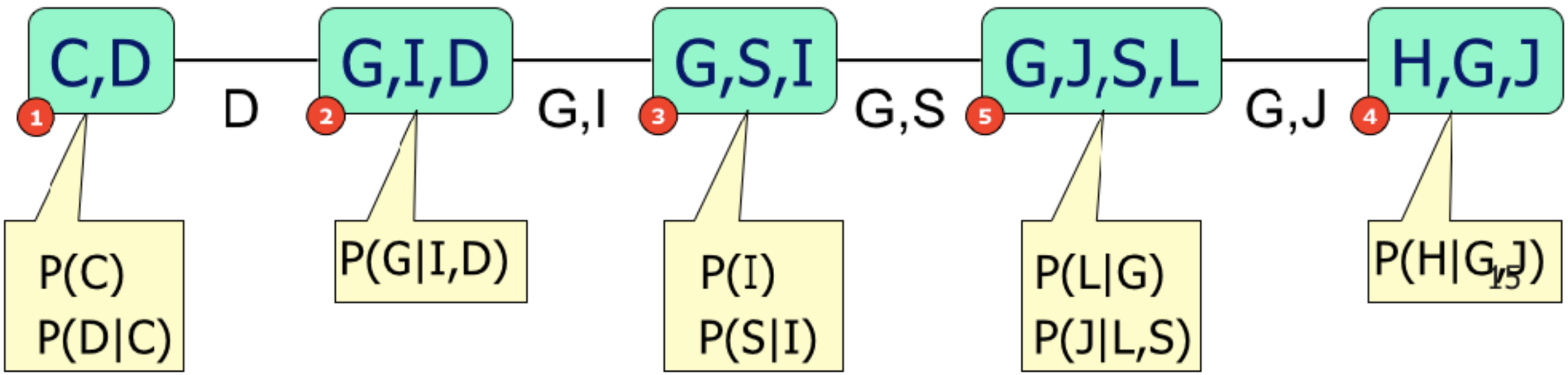
Compute query over another variable?

C5 as the root





C4 as the root



Clique Tree Calibration

Definition

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

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

A clique tree $\mathcal{T}$ is calibrated if all pairs of adjacent cliques are calibrated. For a calibrated clique tree, we use the term clique beliefs for $\beta_i(C_i)$ and sepset beliefs for

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

Clique Tree Calibration

The main advantage of the clique tree algorithm is that it computes the posterior probability of all variables in a graphical model using only twice the computation of the upward pass in the same tree.

If we want to compute the posterior distribution over every random variable in the network, the naive algorithm which does inference for each variable has cost nc where n is the number of variable and c is the cost for a single execution of clique tree inference. A better naive algorithm is that we run the algorithm once for each every clique, making it the root and it has cost Kc where K is the number of cliques.

If we compute the beliefs using calibration, the total cost will be only $2c$. The saving is considerable.

Note: Select a root note. Calibration occurs in one upstream and one downstream pass.

Answer Queries

1. Incremental Updates
2. Out-of-clique Queries

Belief Update

We now show that the entire message-parsing process can be executed in an equivalent way in term of the clique and sepset beliefs (β_i and $\mu_{i,j}$), without having to remember the initial potential ψ_i, or to compute explicitly the message $\delta_{i \to j}$.

Each clique C_i initializes β_i as ψ_i and then updates it by multiplying with message updates received from its neighbors. Each sepset $S_{i,j}$ maintains $\mu_{i,j}$ as the previous message passed along the edge $(i - j)$, regardless of the direction. The message is used to ensure that we do not double-count: whenever a new message is passed along the edge, it is divided by the old message, eliminating the previous message from the update to the clique.

Equivalence of Sum-Product and Belief Update Messages

Theorem

Consider a set of sum-product initial potentials $\{\psi_i : i \in \mathcal{V}_\mathcal{T}\}$ *and messages* $\{\delta_{i\to j}, \delta_{j\to i} : (i-j) \in \varepsilon_\mathcal{T}\}$, *and a set of belief-update beliefs* $\{\beta_i : i \in \mathcal{V}_\mathcal{T}\}$ *and messages* $\{\mu_{i,j} : (i-j) \in \varepsilon_\mathcal{T}\}$ *for which* $\beta_i = \psi_i \cdot \prod_{k\in Nb_i} \delta_{k\to i}$ *and* $\mu_{i,j}(S_{i,j}) = \delta_{j\to i}\delta_{i\to j}$ *hold. For any pair of neighboring cliques* C_i *and* C_j, *let* $\{\delta_{i\to j}, \delta_{j\to i} : (i-j) \in \varepsilon_\mathcal{T}\}$ *be the set of sum-product messages following an application of SP-Message(i, j), and* $\{\beta'_i : C_i \in \mathcal{T}\}$, $\{\mu'_{i,j} : (i-j) \in \varepsilon_\mathcal{T}\}$ *be the set of belief-update beliefs following an application of BU-Message(i, j). Then* $\beta_i = \psi_i \cdot \prod_{k\in Nb_i} \delta_{k\to i}$ *and* $\mu_{i,j}(S_{i,j}) = \delta_{j\to i}\delta_{i\to j}$ *also hold for the new beliefs* $\delta'_{i\to j}, \beta'_i, \mu'_{i,j}$.

Belief Update

At this point, clique C_i has all of the necessary information to compute its belief:

$$\beta_i = \psi_i \cdot \prod_{k \in Nb_i} \delta_{k \to i}$$

where $\delta_{i \to j} = \sum_{C_i - S_{i,j}} \psi_i \prod_{k \in (Nb_i - \{j\})} \delta_{k \to i}$.

We can also multiply all of the messages and then divide the resulting factor $\delta_{j \to i}$ by $\delta_{j \to i}$ to compute the messages:

$$\delta_{i \to j} = \frac{\sum_{C_i - S_{i,j}} \beta_i}{\delta_{j \to i}}$$

Construct a Clique Tree

1. Clique trees from variable elimination
2. Clique trees from chordal graphs
   - Given a set of factors, construct the undirected graph $\mathcal{H}_\phi$
   - Triangulate $\mathcal{H}_\phi$ to construct a chordal graph $\mathcal{H}^*$
   - Find cliques in $\mathcal{H}^*$, and make each one a node in a cluster graph
   - Run the maximum spanning tree algorithm on the cluster graph to construct a tree

A Calibrated Clique Tree as a Distribution

A calibrated clique tree is more than simply a data structure that stores the results of probabilistic inference for all of the cliques in the tree. It can be viewed as an alternative representation of the measure $\tilde{P}_\Phi$.

The calibrated clique tree can be represented as

$$Q = \{\beta_i : i \in \mathcal{V}_\mathcal{T}\} \cup \{\mu_{i,j} : (i - j) \in \varepsilon_\mathcal{T}\}$$

where $\mathcal{V}_\mathcal{T}$ is the set of vertexes in the clique tree and $\varepsilon_\mathcal{T}$ is the set of edges in clique tree.

A Calibrated Clique Tree as a Distribution

At calibration, we have:

$$\beta_i = \psi_i \cdot \prod_{k \in Nb_i} \delta_{k \to i}$$

We also have that:

$$\begin{aligned}
\mu_{i,j}(S_{i,j}) &= \sum_{C_i - S_{i,j}} \beta_i(C_i) \\
&= \sum_{C_i - S_{i,j}} \psi_i \prod_{k \in Nb_i} \delta_{k \to i} \\
&= \sum_{C_i - S_{i,j}} \psi_i \delta_{j \to i} \prod_{k \in (Nb_i - \{j\})} \delta_{k \to i} \\
&= \delta_{j \to i} \sum_{C_i - S_{i,j}} \psi_i \prod_{k \in (Nb_i - \{j\})} \delta_{k \to i} \\
&= \delta_{j \to i} \delta_{i \to j}
\end{aligned}$$

A Calibrated Clique Tree as a Distribution

At convergence of the clique tree calibration algorithm, we have that:

$$\tilde{P}_\phi(\mathcal{X}) = \frac{\prod_{i \in \mathcal{V}_\mathcal{T}} \beta_i(C_i)}{\prod_{(i-j) \in \varepsilon_\mathcal{T}} \mu_{i,j}(S_{i,j})}$$

We can define the measure induced by a calibrated clique tree $\mathcal{T}$ to be:

$$Q_\mathcal{T} = \frac{\prod_{i \in \mathcal{V}_\mathcal{T}} \beta_i(C_i)}{\prod_{(i-j) \in \varepsilon_\mathcal{T}} \mu_{i,j}(S_{i,j})}$$

where

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

We can view the clique tree as an alternative representation of the joint measure, one that directly reveals the clique marginals.