

Exact Inference: Clique Trees

Sargur Srihari
srihari@cedar.buffalo.edu

Topics

1. Overview
2. Variable Elimination and Clique Trees
3. Message Passing: Sum-Product
 - VE in a Clique Tree
 - Clique-Tree Calibration
4. Message Passing: Belief Update
5. Constructing a Clique Tree

Overview

- Two methods of inference using factors Φ over variables χ

1. Variable elimination (VE) algorithm

- uses factor representation and local operations instead of generating entire distribution (See next slide)

2. Clique Trees: alternative implementation of same insight

- Use a more global data structure for scheduling operations

Sum-product VE

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

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

Elimination ordering C, D, I, H, G, S, L

1. Eliminating C :

$$\psi_1(C, D) = \phi_C(C) \phi_D(D, C) \quad \tau_1(D) = \sum_C \psi_1(C, D)$$

Each step involves factor product and factor marginalization

Compute the factors

2. Eliminating D :

$$\psi_2(G, I, D) = \phi_G(G, I, D) \tau_1(D) \quad \tau_2(G, I) = \sum_D \psi_2(G, I, D)$$

Note we already eliminated one factor with D , but introduced τ_1 involving D

3. Eliminating I :

$$\psi_3(G, I, S) = \phi_I(I) \phi_S(S, I) \tau_2(G, I) \quad \tau_3(G, S) = \sum_I \psi_3(G, I, S)$$

4. Eliminating H :

$$\psi_4(G, J, H) = \phi_H(H, G, J) \tau_3(G, S) \quad \tau_4(G, J) = \sum_H \psi_4(G, J, H)$$

Note $\tau_4(G, J) = 1$

5. Eliminating G :

$$\psi_5(G, J, L, S) = \tau_4(G, J) \tau_3(G, S) \phi_L(L, G) \quad \tau_5(J, L, S) = \sum_G \psi_5(G, J, L, S)$$

6. Eliminating S :

$$\psi_6(J, L, S) = \tau_5(J, L, S) \cdot \phi_J(J, L, S) \quad \tau_6(J, L) = \sum_S \psi_6(J, L, S)$$

7. Eliminating L :

$$\psi_7(J, L) = \tau_6(J, L) \quad \tau_7(J) = \sum_L \psi_7(J, L)$$

Unnormalized Measure with Factors

1. We deal with unnormalized measure here

$$\tilde{P}_{\Phi}(\chi) = \prod_{\phi_i \in \Phi} \phi_i(\mathbf{X}_i)$$

2. For a BN

1. without evidence

- factors are CPDs and $\tilde{P}_{\Phi}(\chi)$ is a normalized distribution

2. with evidence $E=e$,

1. factors are CPDs restricted to e and $\tilde{P}_B(\chi) = P_B(\chi, e)$

3. For a Gibbs distribution,

1. factors are potentials

2. $\tilde{P}_{\Phi}(\chi)$ is the unnormalized Gibbs measure

Marginalize with Unnormalized

Unnormalized Conditional Measure
equivalent to Normalized Conditional
Probability

$$\begin{aligned}\tilde{P}_{\Phi}(X|Y) &= P_{\Phi}(X|Y) \quad \text{since} \\ \tilde{P}_{\Phi}(\mathbf{X}|\mathbf{Y}) &= \frac{\tilde{P}_{\Phi}(\mathbf{X}, \mathbf{Y})}{\tilde{P}_{\Phi}(\mathbf{Y})} = \frac{\prod_{\phi_i \in \Phi} \phi_i(D_i)}{\sum_X \prod_{\phi_i \in \Phi} \phi_i(D_i)} \\ P_{\phi}(\mathbf{X}|\mathbf{Y}) &= \frac{P_{\Phi}(X, Y)}{P_{\Phi}(Y)} = \frac{\frac{1}{Z} \prod_{\phi_i \in \Phi} \phi_i(D_i)}{\frac{1}{Z} \sum_X \prod_{\phi_i \in \Phi} \phi_i(D_i)}\end{aligned}$$

Factor Product

- Let X , Y and Z be three disjoint sets of variables and let $\Phi_1(X,Y)$ and $\Phi_2(Y,Z)$ be two factors.
- The factor product is the mapping $Val(X,Y,Z) \rightarrow R$ as follows

$$\psi(X,Y,Z) = \Phi_1(X,Y) \Phi_2(Y,Z)$$

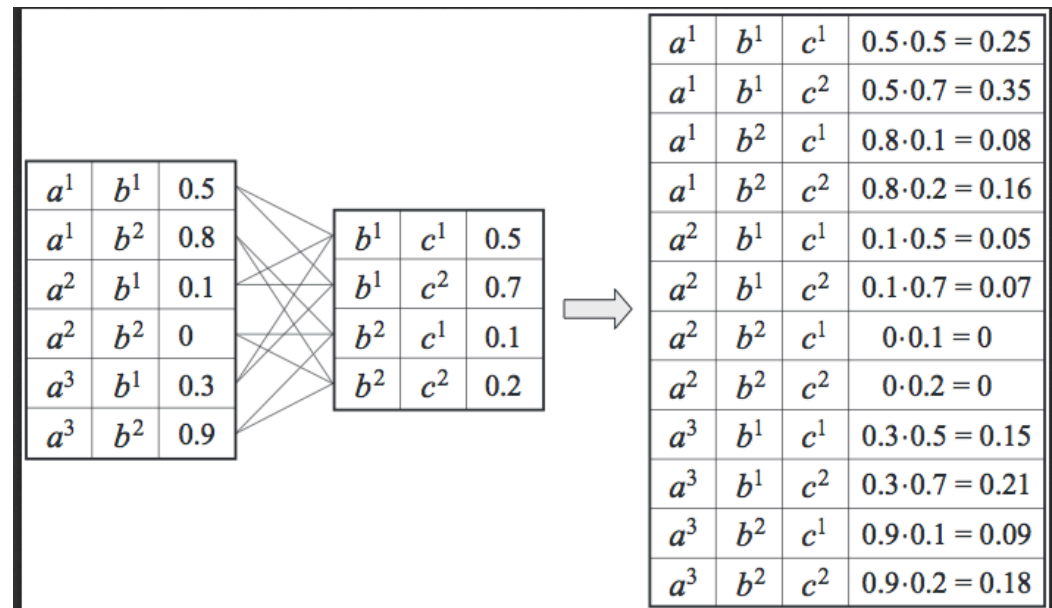
- An example:

$\Phi_1: 3 \times 2 = 6$ entries

$\Phi_2: 2 \times 2 = 4$ entries

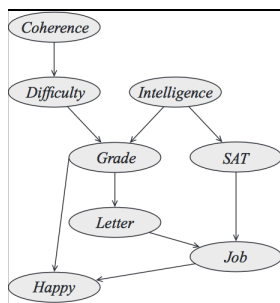
$\psi = \Phi_1 \times \Phi_2$ has

$3 \times 2 \times 2 = 12$ entries



VE and Factor Creation

- In variable elimination
 - each step creates a factor ψ_i by multiplying existing factors
 - A variable is then eliminated to create a factor τ_i which is then used to create another factor



$$P(C,D,I,G,S,L,J,H) = P(C)P(D|C)P(I)P(G|I,D)P(S|I)P(L|G)P(J|L)P(H|G,J) = \Phi_C(C) \Phi_D(D,C) \Phi_I(I) \Phi_G(G,I,D) \Phi_S(S,I) \Phi_L(L,G) \Phi_J(J,L,S) \Phi_H(H,G,J)$$

$$\psi_1(C,D) = \phi_C(C)\phi_D(D,C)$$

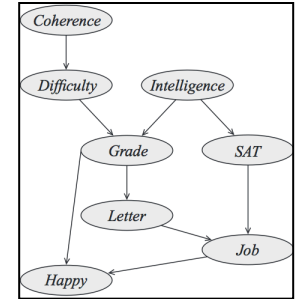
$$\tau_1(D) = \sum_C \psi_1(C,D)$$

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	$\tau_7(J)$

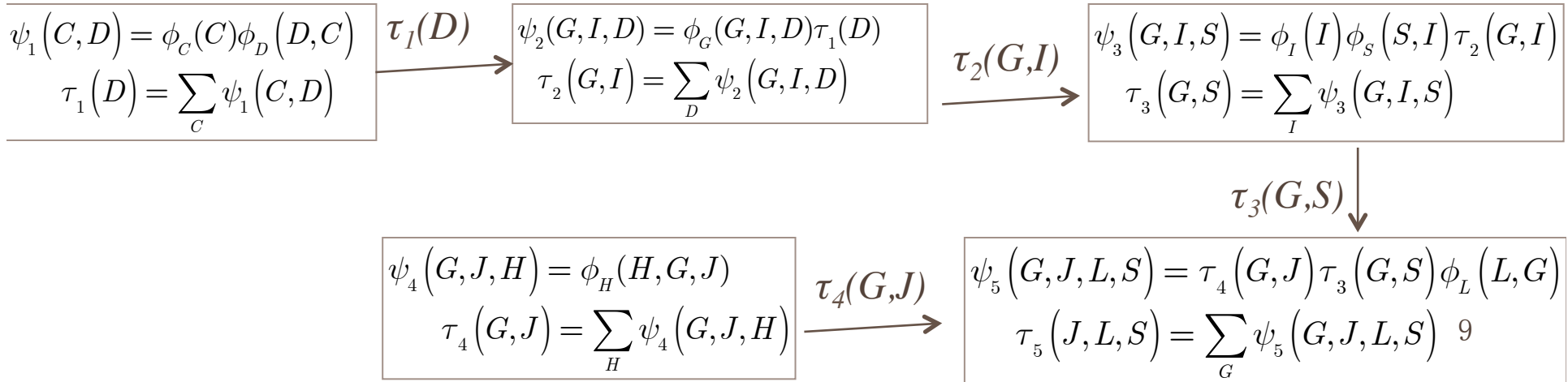
VE Alternative View

Alternative view

- We take ψ_i to be a data-structure
 - takes messages τ_l generated by other factors ψ_j
 - and generates message τ_i used by another factor ψ_l



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	$\tau_7(J)$



Example of Cluster Graph

- VE execution defines cluster graph (a flow-chart)

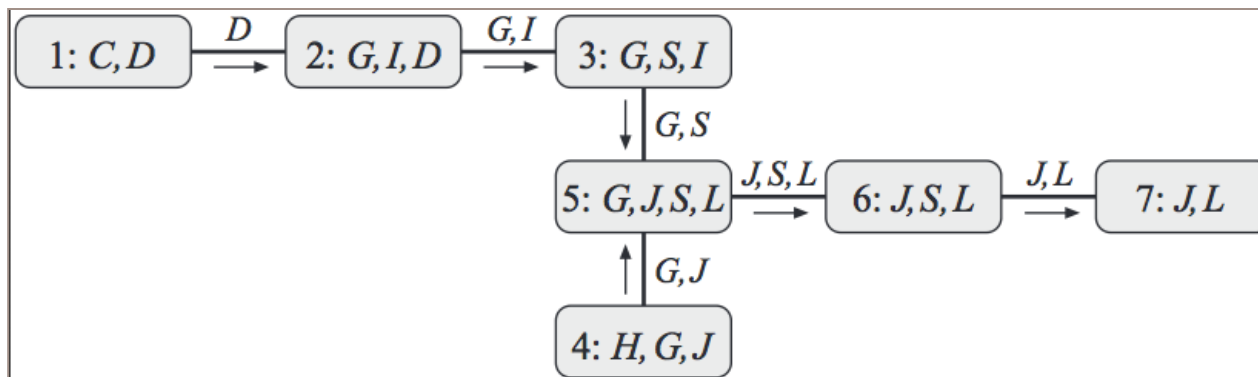
A cluster for each factor ψ_i

Draw edge between clusters C_i and C_j if message τ_i produced by eliminating a variable in ψ_i is used in the computation of τ_j

$$\psi_1(C, D) = \phi_C(C) \phi_D(D, C) \quad \tau_1(D) = \sum_C \psi_1(C, D)$$

$$\psi_2(G, I, D) = \phi_G(G, I, D) \tau_1(D) \quad \tau_2(G, I) = \sum_D \psi_2(G, I, D)$$

- Edge between C_1 and C_2 since message $\tau_1(D)$ produced by eliminating C is used for $\tau_2(G, I)$

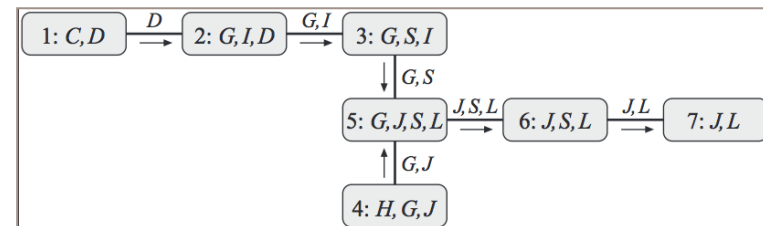


Arrows indicate flow of messages
 $\tau_1(D)$ generated from $\psi_1(C, D)$ participates in the computation of ψ_2

Cluster Graph Definition

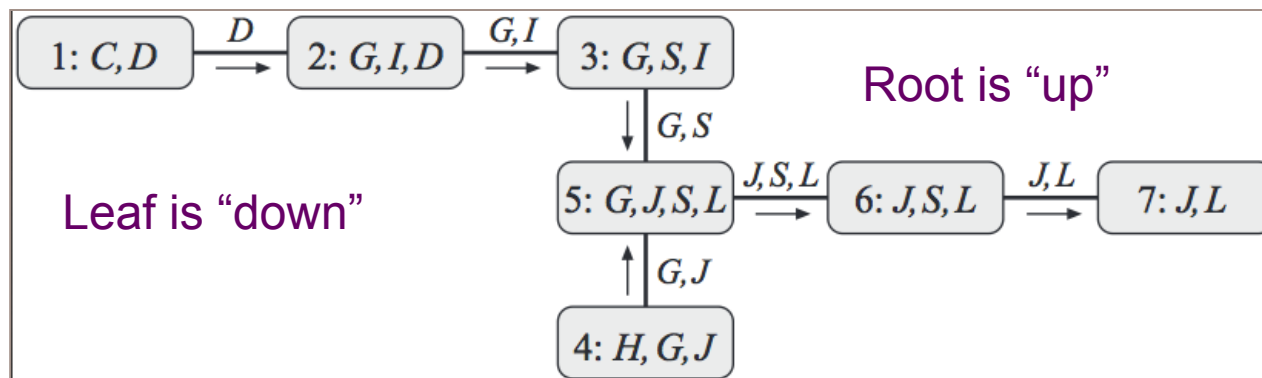
- Cluster graph U for factors Φ over χ is an undirected graph
 - Each of whose nodes i is associated with a subset $C_i \subseteq \chi$
 - Each edge between pair of clusters C_i and C_j is associated with a sepset $S_{i,j} \subseteq C_i \cap C_j$
- Cluster graph is family-preserving
 - Each factor ϕ must be associated with a cluster C_i , denoted $\alpha(\phi)$ such that $Scope(\phi) \subseteq C_i$

E.g., $D \subseteq \{C, D\} \cap \{D, I, G\}$



Cluster Graph is a Directed Tree

- In a tree there are no cycles
- Directions for this tree are specified by messages
 - Since intermediate factor τ_i is used only once
 - Otherwise there would be more than one link for a node
- Called Clique Tree (or *Junction Tree* or *Join Tree*)



Definition of Tree

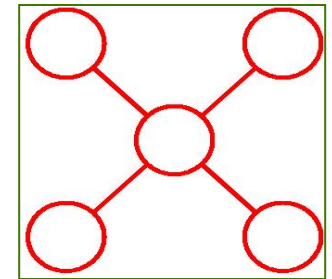
- Tree

- a graph with only one path between any pair of nodes
- Such graphs have no loops
- In directed graphs a tree has a single node with no parents called a *root*
- Directed to undirected will not add moralization links since every node has only one parent

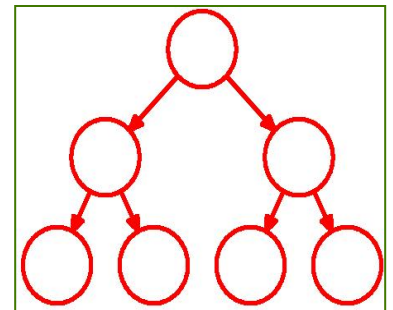
- Polytree

- A directed graph has nodes with more than one parent but there is only one path between nodes (ignoring arrow direction)
- Moralization will add links

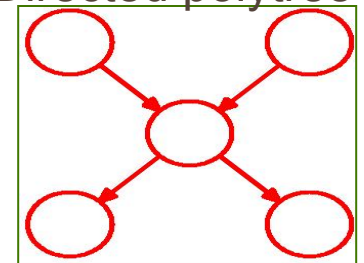
Undirected tree



Directed tree



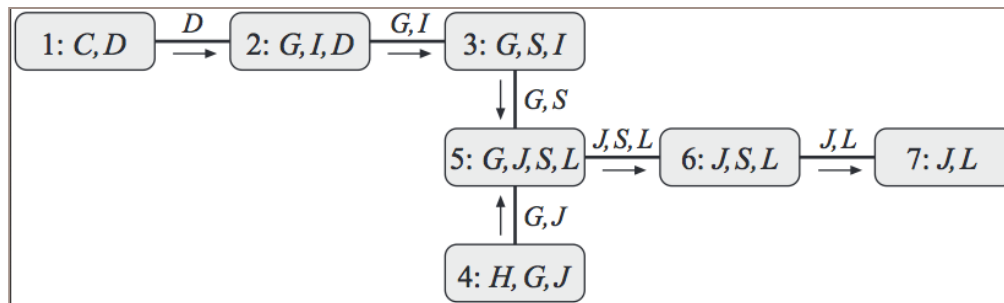
Directed polytree



Running Intersection Property

1. Definition

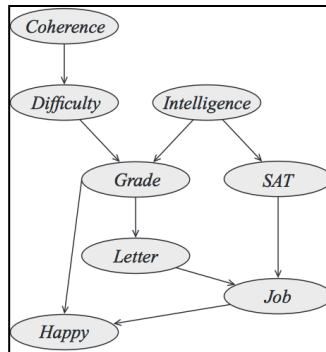
- If $X \in C_i$ & $X \in C_j$ then X is in every clique inbetween
 - In a clique, every pair of nodes is connected
 - In a maximal clique no more nodes can be added
- Ex: in cluster graph below, G is present in C_2 and C_4 and also present in clique inbetween: C_3 and C_4



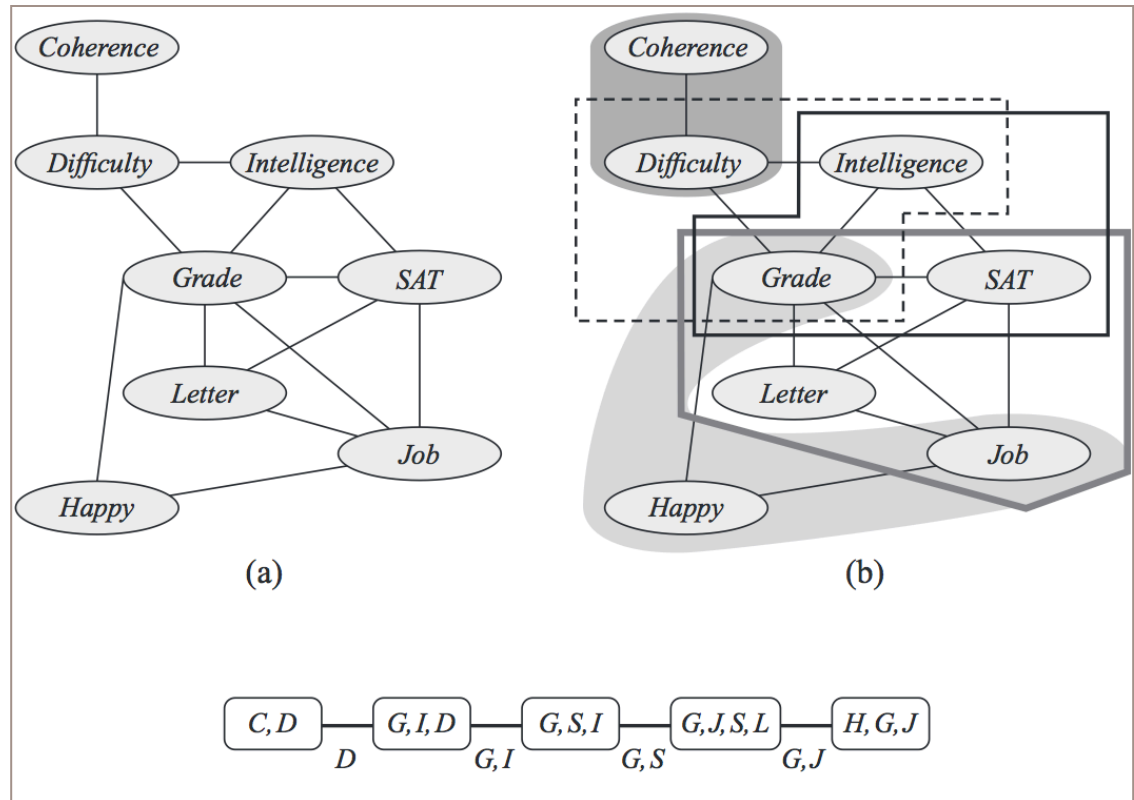
- ## 2. A VE generated cluster graph satisfies running intersection property

Clique Tree

1. BN



2. Induced Graph

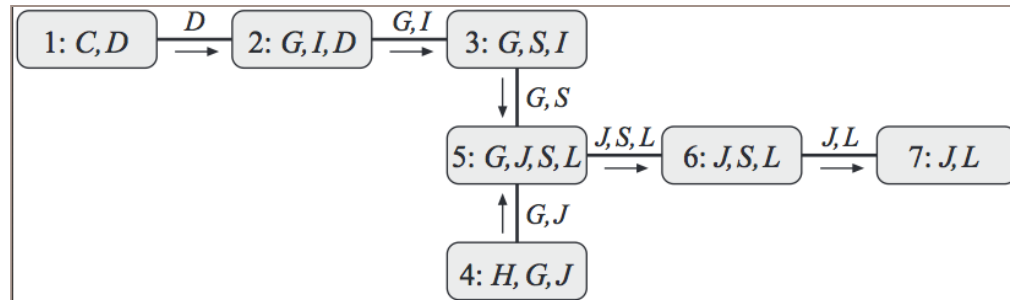


3. Some Cliques: $\{C, D\}, \{G, I, D\}, \{G, S, I\}, \{G, J, S, L\}, \{H, G, J\}$

4. A clique Tree that satisfies running intersection

Clique Tree Definition

- A tree T is a clique tree for graph H if
 - Each node in T corresponds to a clique in H and each maximal clique in H is a node in T



- Each sepset $S_{i,j}$ separates $W_{<I_j,j)}$ and $W_{<(j,i)}$ in H
 - Edge $S_{2,3}=\{G,I\}$ separates $W_{<(2,3)}=\{G,I,D\}$ and $W_{<(3,2)}=\{G,S,I\}$

Message Passing: Sum Product

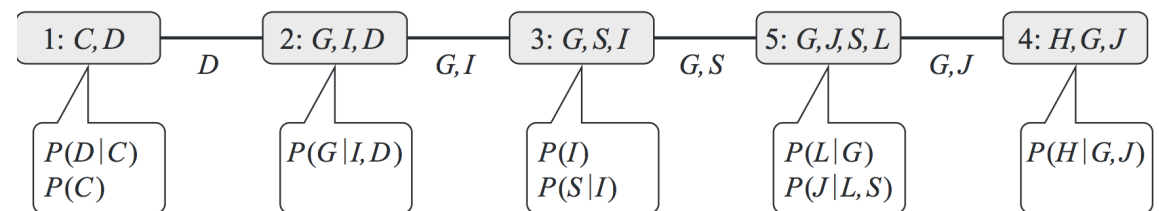
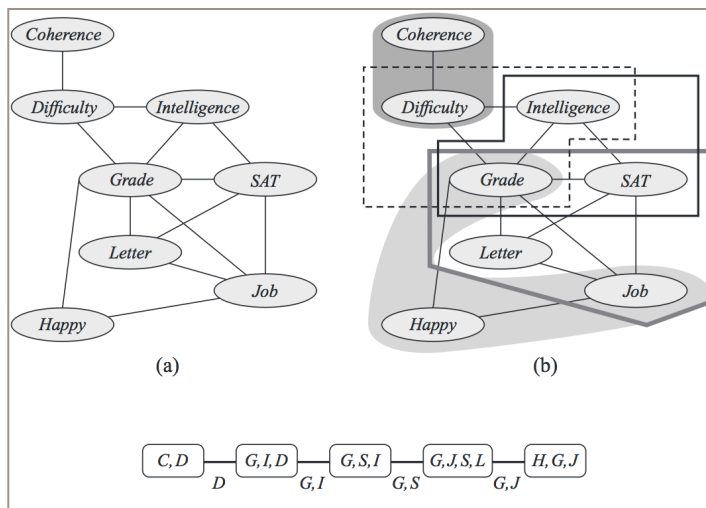
- Proceed in opposite direction of VE algorithm:
 - Starting from a clique tree, how to perform VE
- Clique Tree is a very versatile Data Structure

Variable Elimination in a Clique Tree

- Clique Tree can be used as guidance for VE
- Factors are computed in the cliques and messages are sent along edges

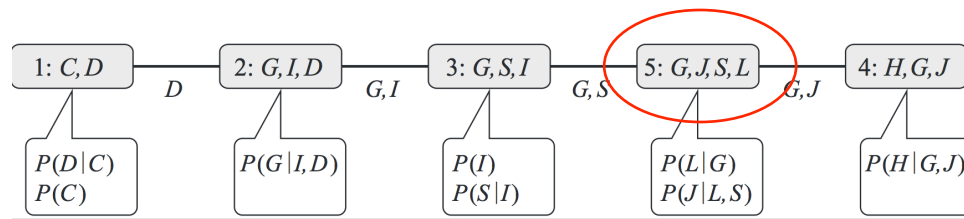
Variable Elimination in a Clique Tree

- A Clique Tree for Student Network
 - This tree satisfies Running Intersection Property
 - i.e., If $X \in C_i$ & $X \in C_j$ then X is in every clique in between
 - Family Preservation property
 - i.e., each factor is associated with a cluster



Example of VE in a Clique Tree

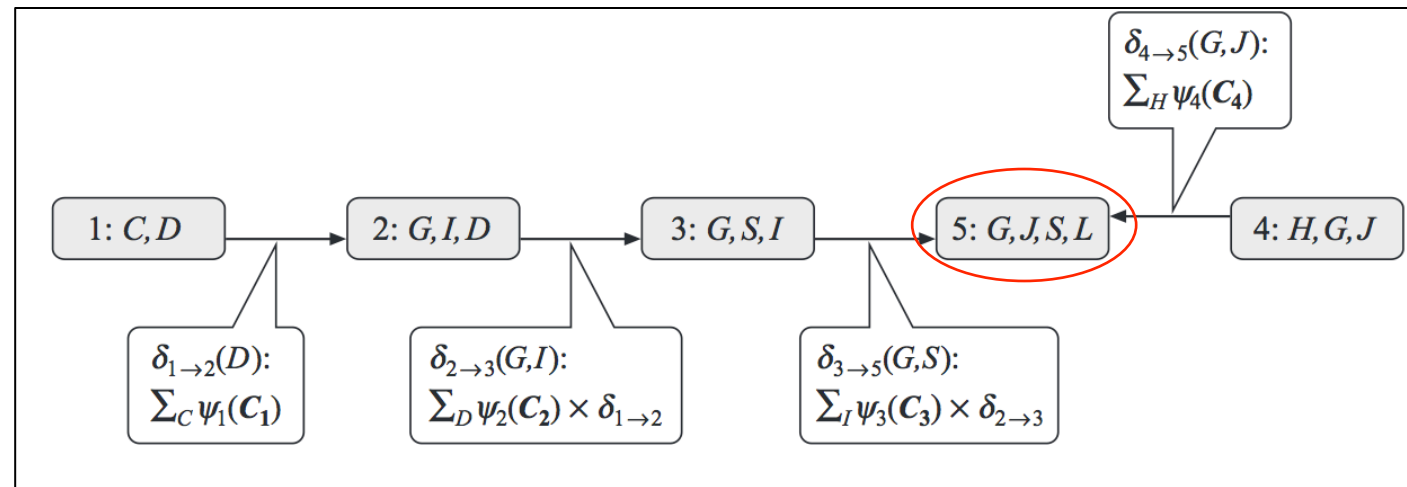
- A Clique Tree for Student Network
 - Non-maximal cliques C_6 and C_7 are absent



- Assign α : initial factors (CPDs) to cliques
- First step: Generate initial set of potentials by multiplying out the factors
 - E.g., $\psi_5(J, L, G, S) = \phi_L(L, G) * \phi_J(J, L, S)$
- Root is selected to have variable J , since we are interested in determining $P(J)$, e.g., C_5

Message Propagation in a Clique Tree

Root = C_5
 To compute $P(J)$

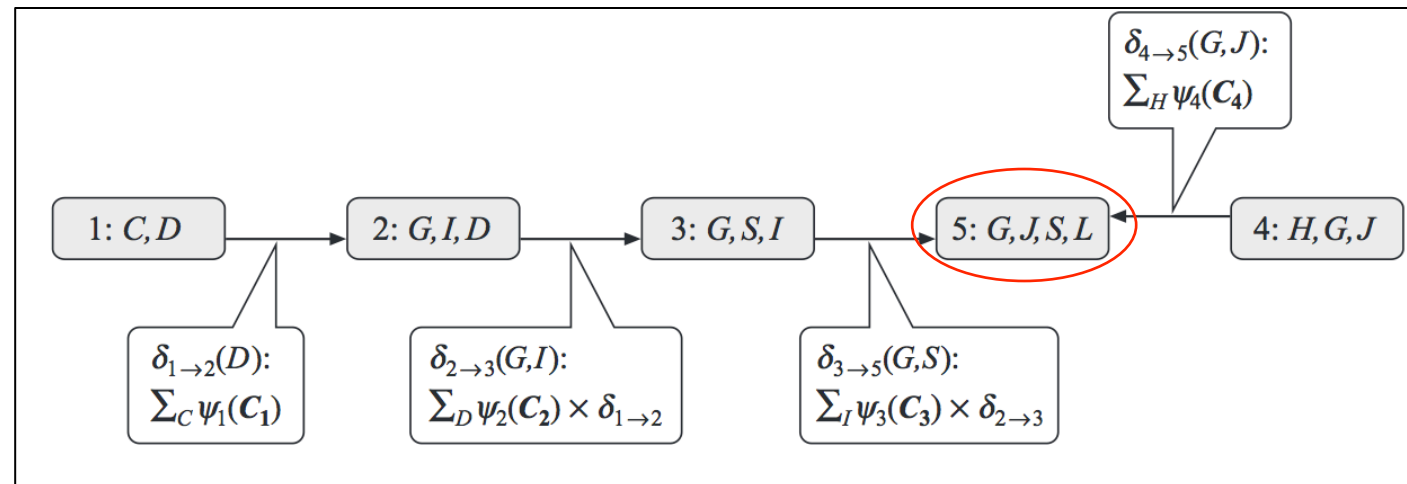


In C_1 : eliminate C by performing $\sum_C \psi_1(C, D)$
 The resulting factor has scope D . We send it as a message $\delta_{1 \rightarrow 2}(D)$ to C_2

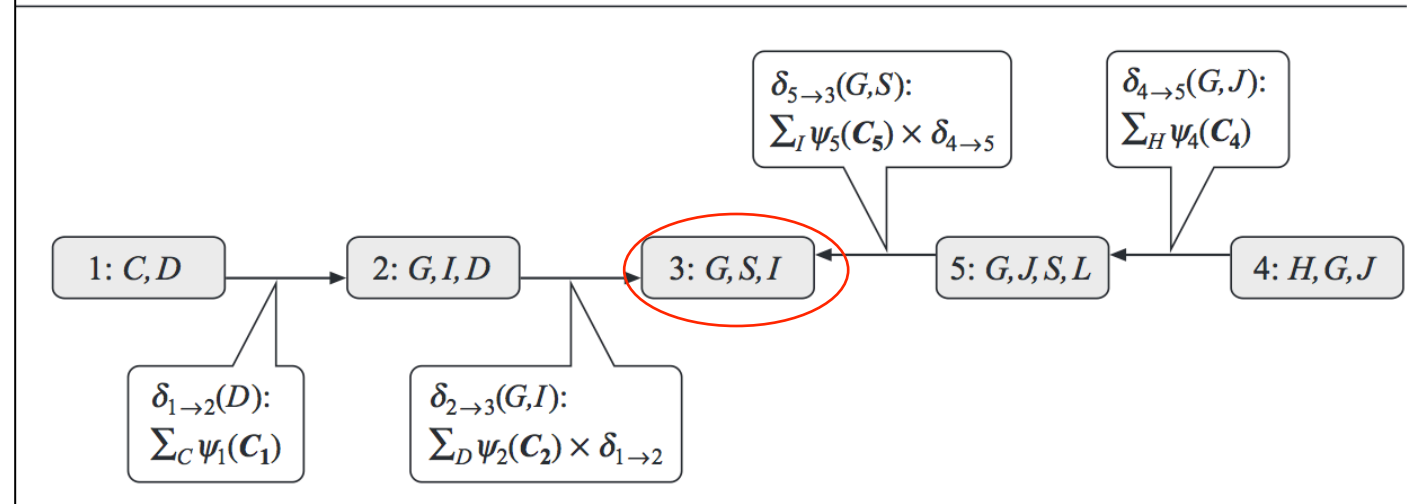
In C_2 : We define $\beta_2(G, I, D) = \delta_{1 \rightarrow 2}(D) \psi_2(G, I, D)$. We then eliminate D to get a factor over G, I . The resulting factor is $\delta_{2 \rightarrow 3}(G, I)$ which is sent to C_3 .

Message Propagation in a Clique Tree

Root= C_5
To compute $P(J)$



Root= C_3
To compute $P(G)$



VE as Clique Tree Message Passing

1. Let T be a clique tree with Cliques C_1, \dots, C_k
2. Begin by multiplying factors assigned to each clique, resulting in initial potentials $\psi_j(C_j) = \prod_{\phi: \alpha(\phi)=j} \phi$
3. Begin passing messages between neighbor cliques sending towards root node

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

4. Message passing culminates at root node
 - Result is a factor called *beliefs* denoted $\beta_r(C_r)$ which is equivalent to

$$\tilde{P}_\phi(C_r) = \sum_{\chi - C_r} \prod_\phi \phi$$

Algorithm: Upward Pass of VE in Clique Tree

Procedure *Ctree-SP-Upward* (

Φ , // Set of factors
 \mathcal{T} , // Clique tree over Φ
 α , // 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     $\delta_{i \rightarrow p_r(i)}(S_{i, p_r(i)}) \leftarrow \text{SP-Message}(i, p_r(i))$ 
5     $\beta_r \leftarrow \psi_r \cdot \prod_{k \in \text{Nb}_{C_r}} \delta_{k \rightarrow r}$ 
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   $\psi(C_i) \leftarrow \psi_i \cdot \prod_{k \in (\text{Nb}_i - \{j\})} \delta_{k \rightarrow i}$ 
2   $\tau(S_{i,j}) \leftarrow \sum_{C_i - S_{i,j}} \psi(C_i)$ 
3  return  $\tau(S_{i,j})$ 

```


Clique Tree Calibration

- We have seen how to use the same clique tree to compute probability of any variable
- We wish to compute the probability of a large number of variables
 - Consider task of computing the posterior distribution over every random variable in network
 - As with HMMs with several latent variables

Ready Clique

- C_i is *ready* to transmit to neighbor C_j
 - when C_i has messages from all of its neighbors except from C_j
- Sum-product belief propagation algorithm
 - Uses yet another layer of dynamic programming
 - Defined asynchronously

Sum-Product Belief Propagation

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

Procedure *CTree-SP-Calibrate* (

Φ , // Set of factors
 \mathcal{T} // Clique tree over Φ
)

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

Result at End of Algorithm

- Computes beliefs of all cliques by
 - Multiplying the initial potential with each of the incoming messages
- For each clique i , β_i is computed as

$$\beta_i(C_i) = \sum_{\chi - C_i} \tilde{P}_{\Phi}(\chi)$$

- Which is the unnormalized marginal distribution of variables in C_i

Calibration Definition

- If X appears in two cliques they must agree on its marginal
- Two adjacent cliques C_i and $C_k=j$ are said to be calibrated if

$$\sum_{C_i - S_{i,j}} \beta_i(C_i) = \sum_{C_j - S_{i,j}} \beta_j(C_j)$$
- Clique tree T is calibrated if all adjacent pairs of cliques are calibrated
- Terminology:
 - Clique Beliefs: $\beta_i(C_i)$
 - Sepset Beliefs: $\mu_{i,j}(S_{i,j}) = \sum_{C_i - S_{i,j}} \beta_i(C_i) = \sum_{C_j - S_{i,j}} \beta_j(C_j)$

Calibration Tree as a Distribution

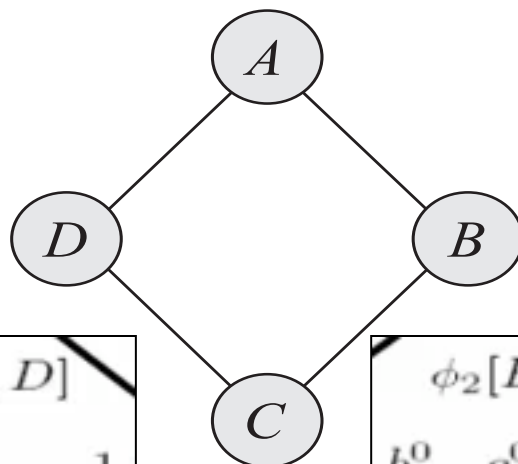
- A calibrated clique tree
 - Is more than a data structure that stores results of probabilistic inference
 - It can be viewed as an alternative representation of P_Φ
- At convergence of clique tree calibration algorithm

$$\tilde{P}_\Phi(\chi) = \frac{\prod_{i \in V_T} \beta_i(C_i)}{\prod_{(i,j) \in E_T} \mu_{i,j}(S_{i,j})}$$

Misconception Markov Network

$$\phi_4[D, A]$$

d^0	a^0	100
d^0	a^1	1
d^1	a^0	1
d^1	a^1	100



$$\phi_1[A, B]$$

a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10

Factors in terms of potentials

$$\phi_3[C, D]$$

c^0	d^0	1
c^0	d^1	100
c^1	d^0	100
c^1	d^1	1

$$\phi_2[B, C]$$

b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

Gibbs Distribution

$$P(a, b, c, d) = \frac{1}{Z} \phi_1(a, b) \cdot \phi_2(b, c) \cdot \phi_3(c, d) \cdot \phi_4(d, a)$$

where

$$Z = \sum_{a, b, c, d} \phi_1(a, b) \cdot \phi_2(b, c) \cdot \phi_3(c, d) \cdot \phi_4(d, a)$$

$$Z = 7,201,840$$

Assignment				Unnormalized	Normalized
a^0	b^0	c^0	d^0	300000	0.04
a^0	b^0	c^0	d^1	300000	0.04
a^0	b^0	c^1	d^0	300000	0.04
a^0	b^0	c^1	d^1	30	$4.1 \cdot 10^{-6}$
a^0	b^1	c^0	d^0	500	$6.9 \cdot 10^{-5}$
a^0	b^1	c^0	d^1	500	$6.9 \cdot 10^{-5}$
a^0	b^1	c^1	d^0	5000000	0.69
a^0	b^1	c^1	d^1	500	$6.9 \cdot 10^{-5}$
a^1	b^0	c^0	d^0	100	$1.4 \cdot 10^{-5}$
a^1	b^0	c^0	d^1	1000000	0.14
a^1	b^0	c^1	d^0	100	$1.4 \cdot 10^{-5}$
a^1	b^0	c^1	d^1	100	$1.4 \cdot 10^{-5}$
a^1	b^1	c^0	d^0	10	$1.4 \cdot 10^{-6}$
a^1	b^1	c^0	d^1	100000	0.014
a^1	b^1	c^1	d^0	100000	0.014
a^1	b^1	c^1	d^1	100000	0.014

Beliefs for Misconception example

- One clique tree consists cliques $\{A,B,D\}$ and $\{B,C,D\}$ with sepset $\{B,D\}$



- Tree obtained either from (i) VE or from (ii) triangulation (constructing a chordal graph)
- Final clique potentials and sepset

Assignment				max _C				Assignment				max _{A,C}			
a ⁰	b ⁰	d ⁰		600,000				b ⁰	c ⁰	d ⁰	300,000	b ⁰	c ⁰	d ⁰	300,000
a ⁰	b ⁰	d ¹		300,030				b ⁰	c ⁰	d ¹	1,300,130	b ⁰	c ⁰	d ¹	300,030
a ⁰	b ¹	d ⁰		5,000,500				b ⁰	c ¹	d ⁰		b ⁰	c ¹	d ⁰	
a ⁰	b ¹	d ¹		1,000				b ⁰	c ¹	d ¹		b ⁰	c ¹	d ¹	
a ¹	b ⁰	d ⁰		200				b ¹	c ⁰	d ⁰		b ¹	c ⁰	d ⁰	
a ¹	b ⁰	d ¹		1,000,100				b ¹	c ⁰	d ¹		b ¹	c ⁰	d ¹	
a ¹	b ¹	d ⁰		100,010				b ¹	c ¹	d ⁰		b ¹	c ¹	d ⁰	
a ¹	b ¹	d ¹		200,000				b ¹	c ¹	d ¹		b ¹	c ¹	d ¹	
$\beta_1(A,B,D)$				$\mu_{1,2}(B,D)$				$\beta_2(B,C,D)$							

- Potential from Gibbs and Clique Tree are same:

$$\frac{\tilde{P}_{\Phi}(a^1, b^0, c^1, d^0) = 100}{\mu_{1,2}(b^0, d^0)} = \frac{\beta_1(a^1, b^0, d^0) \beta_2(b^0, c^1, d^0)}{600 \cdot 200} = 100$$

Message Passing: Belief Update

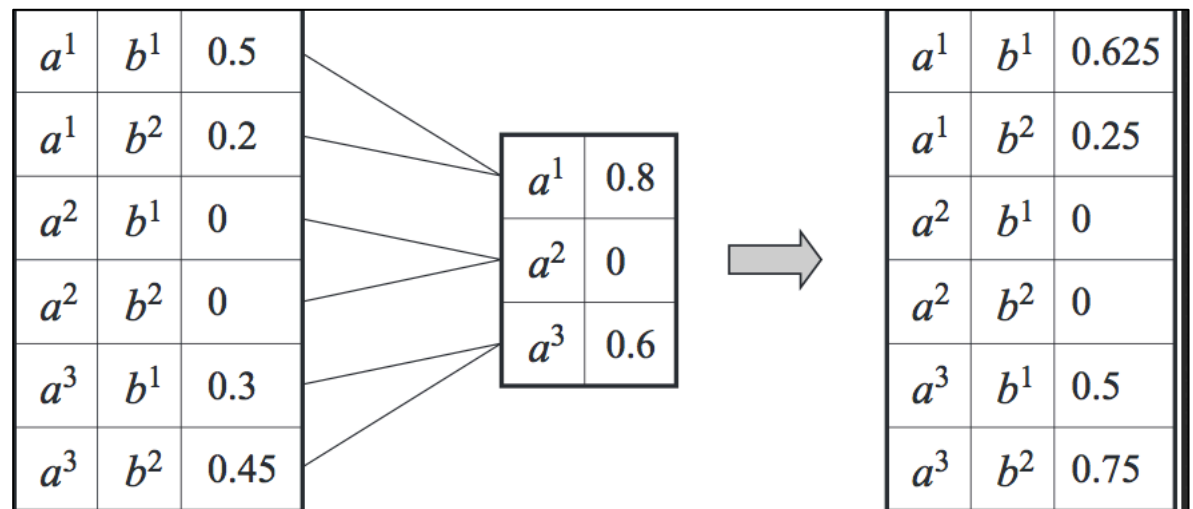
- Alternative Message Passing Scheme
- Involves operations on reparameterized distribution in terms of
 - cliques $\{\beta_i(C_i)\}$, $i \in V_T$ and
 - sepset beliefs $\{\mu_{i,j}(S_{i,j})\}$, $(i-j) \in V_T$

Message Passing with Division

- Multiply all the messages and then divide the resulting factor by $\delta_{j \rightarrow i}$

Factor Division

- Message Passing with Division
- An example of factor division



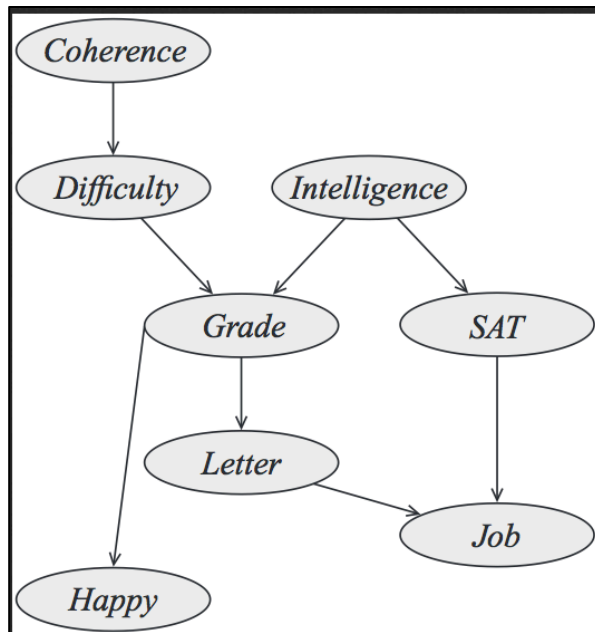
Constructing a Clique Tree

- Two approaches to construct a clique tree from a graph
 - From Variable Elimination
 - From Chordal Graphs

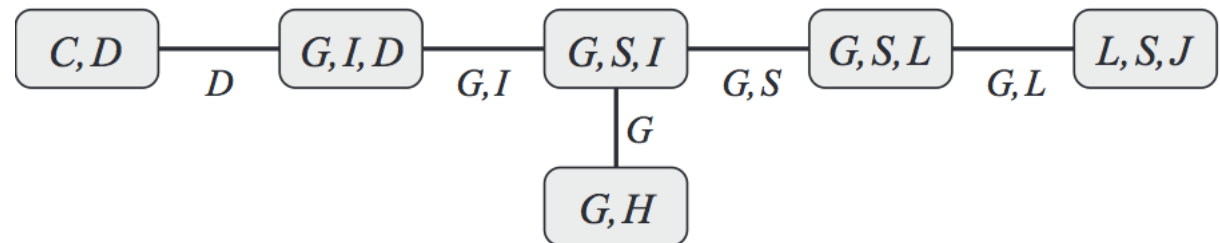
Clique Tree from VE

- Execution of Variable Elimination can be associated with a cluster graph
 - Satisfies running intersection property and is hence a clique tree

Unambitious Student

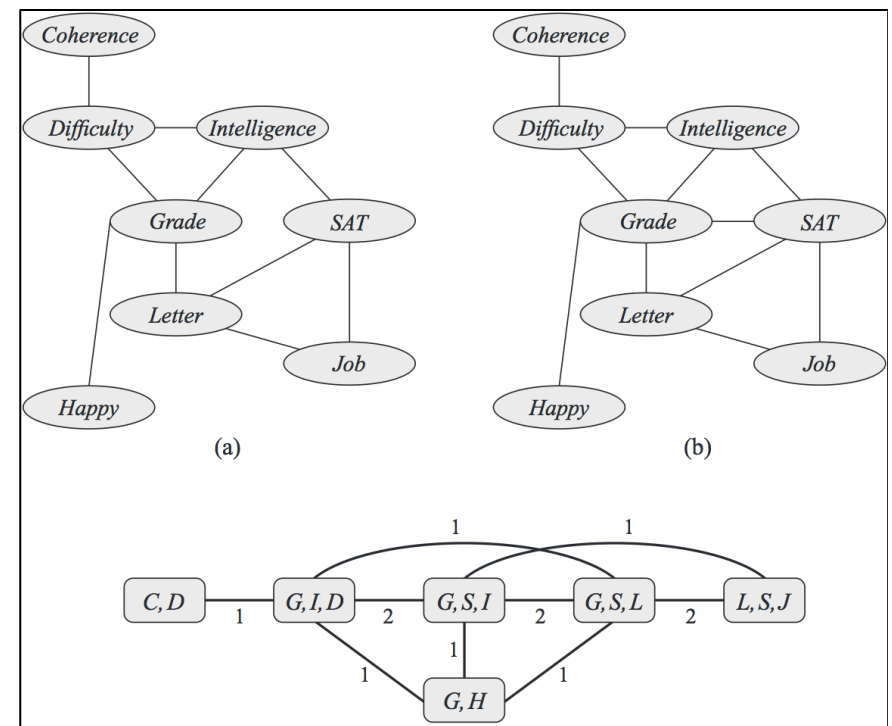


Variable Elimination with ordering J, L, S, H, C, D, I, G results in clique tree:



Clique Tree from Chordal Graphs

- There exists a clique tree for Φ whose cliques are precisely the maximal cliques in $I_{\Phi, <}$
 - Triangulation: construct chordal graph subsuming existing graph
 1. Undirected factor graph
 2. A triangulation
 3. Cluster graph
 - With edge weights



Algorithm: Clique Tree from Chordal Graph

- Given a set of factors, construct the undirected graph H_Φ
- Triangulate H_Φ to construct Chordal Graph H^*
- Find cliques in H^* , and make each one a node in a cluster graph
- Run the maximal spanning tree algorithm on the cluster graph to construct a tree