

## CS 3750 Machine Learning Lecture 7

### Graphical models Exact inference

Milos Hauskrecht  
[milos@cs.pitt.edu](mailto:milos@cs.pitt.edu)  
5329 Sennott Square

The clique tree slides in this lecture  
were written by **Tomas Singliar**

---

CS 3750 Advanced Machine Learning

### Markov random fields

- **Probabilistic models with symmetric dependences.**

- Typically models spatially varying quantities

$$P(x) \propto \prod_{c \in cl(x)} f_c(x_c)$$

$f_c(x_c)$  - A potential function (defined over factors)

$$P(x) = \frac{1}{Z} \exp\left(- \sum_{c \in cl(x)} \phi_c(x_c)\right)$$

- Gibbs (Boltzman) distribution

$$Z = \sum_{x \in \{x\}} \exp\left(- \sum_{c \in cl(x)} \phi_c(x_c)\right) \quad \text{- A partition function}$$

---

CS 3750 Advanced Machine Learning

## Graphical representation of MRFs

### An undirected network (also called independence graph)

- $G = (S, E)$ 
  - $S = 1, 2, \dots, N$  correspond to random variables
  - $(i, j) \in E \Leftrightarrow \exists c : \{i, j\} \subset c$   
or  $x_i$  and  $x_j$  appear within the same factor  $c$

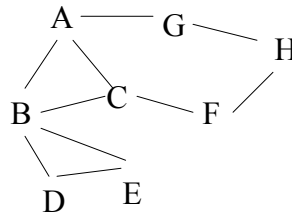
#### Example:

- variables  $A, B, \dots, H$
- Assume the full joint of MRF

$$P(A, B, \dots, H) =$$

$$\phi_1(A, B, C)\phi_2(B, D, E)\phi_3(A, G)$$

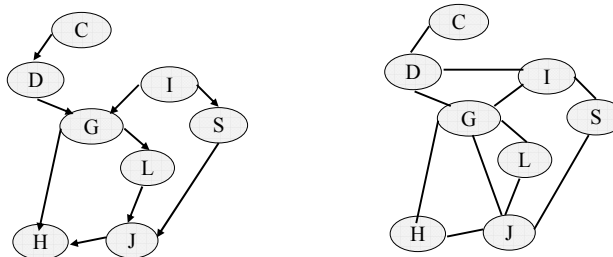
$$\phi_4(C, F)\phi_5(G, H)\phi_6(F, H)$$



## Converting BBNs to MRFs

**Moral-graph  $H[G]$ :** of a bayesian network over  $X$  is an undirected graph over  $X$  that contains an edge between  $x$  and  $y$  if:

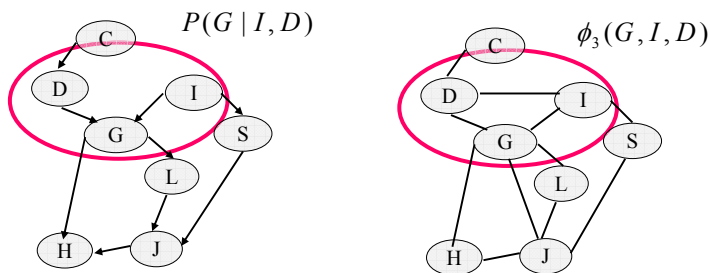
- There exists a directed edge between them in  $G$ .
- They are both parents of the same node in  $G$ .



## Moral Graphs

Why moralization?

$$\begin{aligned}
 P(C, D, G, I, S, L, J, H) &= \\
 &= P(C)P(D|C)P(G|I, D)P(S|I)P(L|G)P(J|L, S)P(H|G, J) \\
 &= \phi_1(C)\phi_2(D, C)\phi_3(G, I, D)\phi_4(S, I)\phi_5(L, G)\phi_6(J, L, S)\phi_7(H, G, J)
 \end{aligned}$$

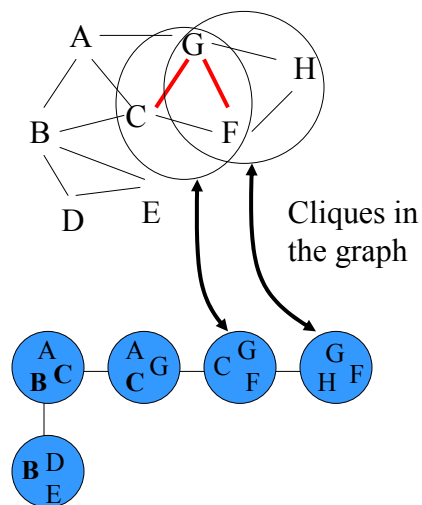


CS 3750 Advanced Machine Learning

## Tree decomposition of the graph

- **A tree decomposition of a graph  $G$ :**

- A tree  $T$  with a vertex set associated to every node.
- For all edges  $\{v, w\} \in G$ : there is a set containing both  $v$  and  $w$  in  $T$ .
- For every  $v \in G$ : the nodes in  $T$  that contain  $v$  form a connected subtree.

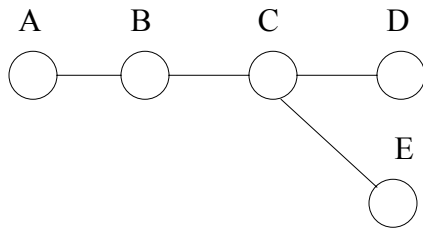


CS 3750 Advanced Machine Learning

## Trees

Why do we like trees?

- Inference in trees structures can be done in time **linear in the number of nodes**



## Clique tree

- Clique tree = a tree decomposition of the graph
- Can be constructed:
  - from the induced graph  
Built by running the variable elimination procedure
  - from the chordal graph  
Built by running the triangulation algorithm
- We have precompiled the clique tree.
- So how to take advantage of the clique tree to perform inferences?

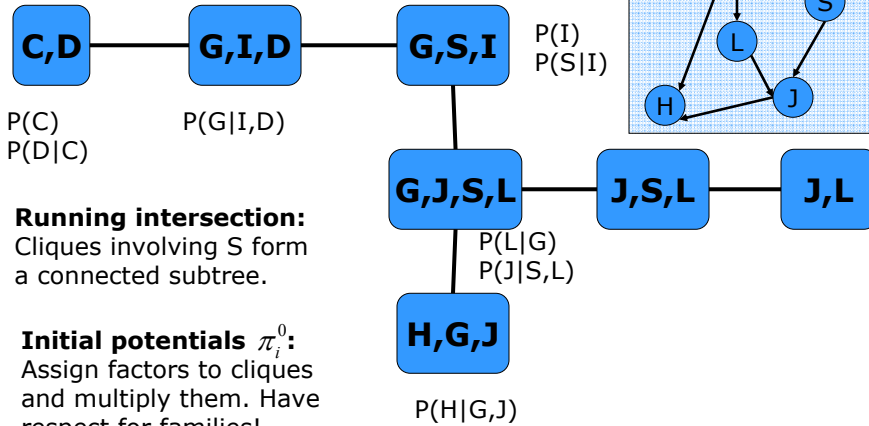
## Clique tree

- VE on the clique tree
  - works on *factors*
- Make factor a data structure
  - Sends and receives messages
- Cluster graph for set of factors  $\mathcal{F}$ , each node  $i$  is associated with a subset (**cluster**)  $C_i$  of  $\mathcal{X}$ .
  - Family-preserving: each factor's variables are completely embedded in a cluster

## Clique tree properties

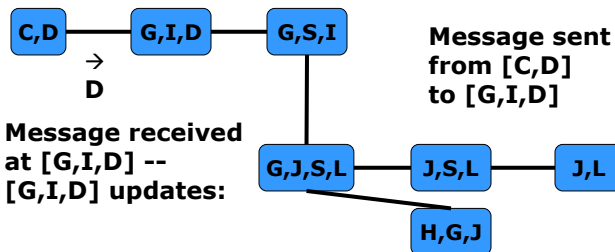
- Sepset  $S_{ij} = C_i \cap C_j$ 
  - **separation** set: Variables  $\mathbf{X}$  on one side of sepset are separated from the variables  $\mathbf{Y}$  on the other side in the factor graph given variables in  $\mathbf{S}$
- **Running intersection property**
  - if  $C_i$  and  $C_j$  both contain  $X$ , then all cliques on the unique path between them do

## Clique trees

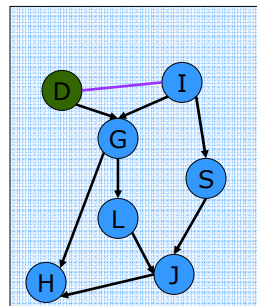
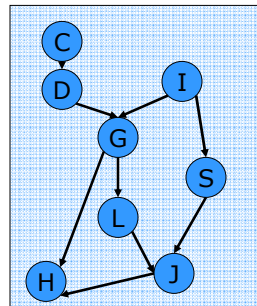


## Message Passing VE

- Query for  $P(J)$ 
  - Eliminate C:  $\tau_1(D) = \sum_C \pi_1^0[C,D]$

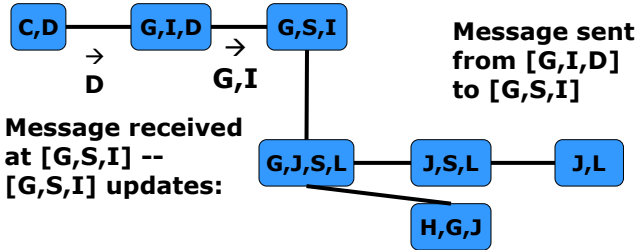


$$\pi_2[G,I,D] = \tau_1(D) \times \pi_2^0[G,I,D]$$

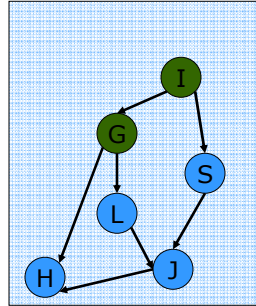
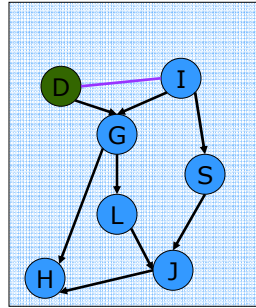


## Message Passing VE

- Query for  $P(J)$ 
  - Eliminate  $D$ :  $\tau_2(G,I) = \sum_D \pi_2[G,I,D]$

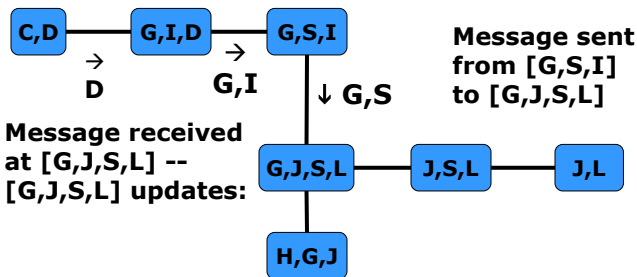


$$\pi_3[G,S,I] = \tau_2(G,I) \times \pi_3^0[G,S,I]$$



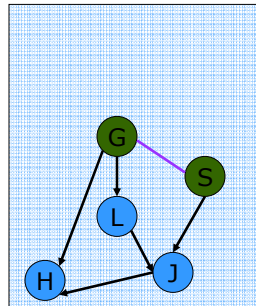
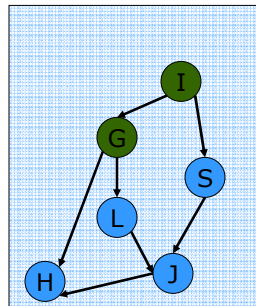
## Message Passing VE

- Query for  $P(J)$ 
  - Eliminate  $I$ :  $\tau_3(G,S) = \sum_I \pi_3[G,S,I]$



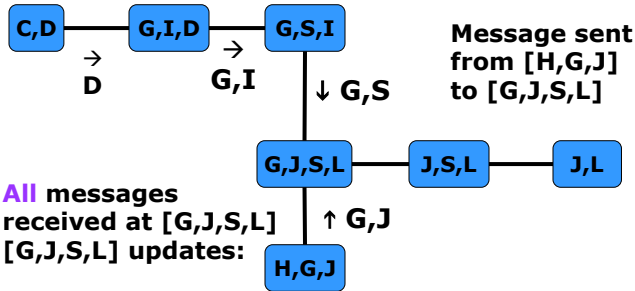
$$\pi_4[G,J,S,L] = \tau_3(G,S) \times \pi_4^0[G,J,S,L]$$

$[G,J,S,L]$  is not **ready!**



## Message Passing VE

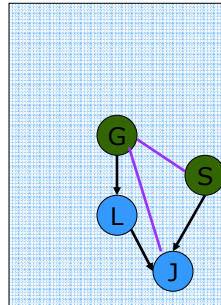
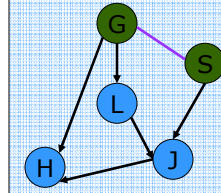
- Query for  $P(J)$ 
  - Eliminate H:  $\tau_4(G,J) = \sum_H \pi_5[H,G,J]$



All messages received at  $[G,J,S,L]$   
 $[G,J,S,L]$  updates:

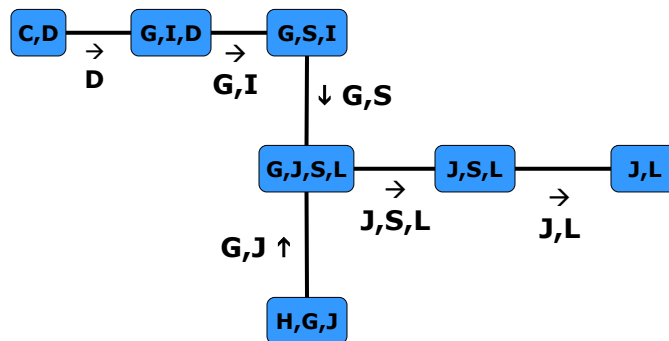
$$\pi_4[G,J,S,L] = \tau_3(G,S) \times \tau_4(G,J) \times \pi_4^0[G,J,S,L]$$

And so on...



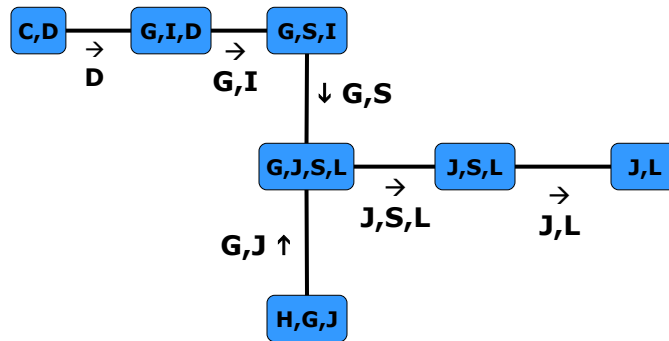
## Message Passing VE

- Chose  $[J,L]$  as the root clique
- ... and finish the inference



## Message Passing VE

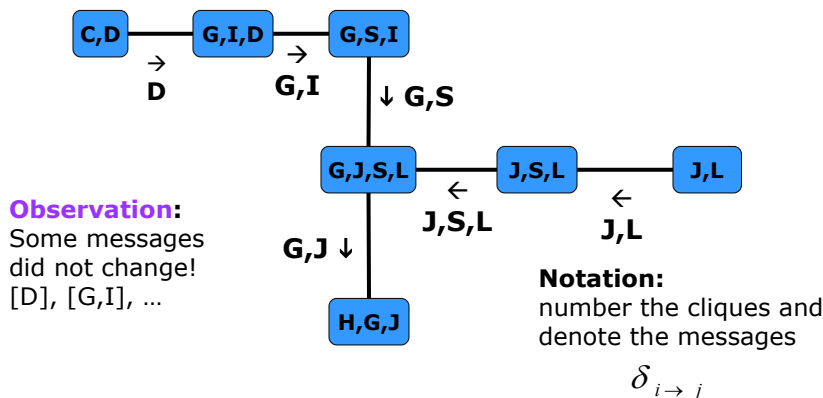
- Chose [J,L] as the root clique
- But could we have chosen otherwise?



CS 3750 Advanced Machine Learning

## Message Passing VE

- Choose [H,G,J] as the root clique



CS 3750 Advanced Machine Learning

## Message passing VE

- Often, many marginals desired
  - Inefficient to re-run inference from scratch
  - One distinct message per edge & direction
- **Methods :**
  - Compute (**unnormalized**) marginals for any node  $Y$  of the tree
  - Results in a *calibrated clique tree* 
$$\sum_{C_i - S_{ij}} \pi_i = \sum_{C_j - S_{ij}} \pi_j$$
- Recap: three kinds of factor objects
  - Initial potentials, final potentials and messages

## Message Passing VE

- **Shafer-Shenoy algorithm**
    - Define a root
    - asynchronous implementation of two passes: upward (send towards the root) and downward (send from the root)
- Asynchronously do:
- node  $i$  ready to send  $m$  to node  $j$  when it has received a message from all other nodes
  - Send message 
$$\delta_{i \rightarrow j} = \sum_{C_i - S_{ij}} \pi_i \times \prod_{k \in N(i) - j} \delta_{k \rightarrow i}$$
- Marginalize root clique's ancillary vars