CS 3750 Machine Learning
Lecture 4

Markov Random Fields II

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Markov random fields

- **Probabilistic models with symmetric dependences.**
  - Typically models spatially varying quantities

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

$\phi_c(x_c)$ - A potential function (defined over factors)

- If $\phi_c(x_c)$ is strictly positive we can rewrite the definition in terms of a log-linear model :

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

- Gibbs (Boltzman) distribution

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

Graphical representation of MRFs

An undirected network (also called independence graph)

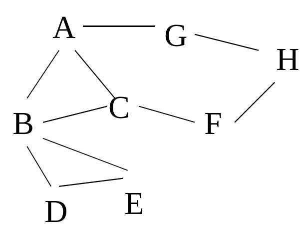
- G = (S, E)
  - S=1, 2, .. N correspond to random variables
  - $(i, j) \in E \Leftrightarrow \exists c : \{i, j\} \subset c$

    or x_i and x_j appear within the same factor c

Example:

- variables A,B ..H
- Assume the full joint of MRF

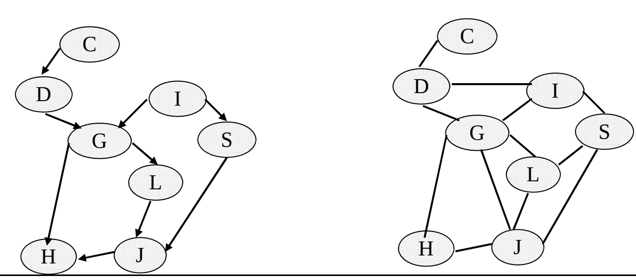
$$P(A, B, \ldots H) \sim$$
$$\phi_1(A, B, C)\phi_2(B, D, E)\phi_3(A, G)$$
$$\phi_4(C, F)\phi_5(G, H)\phi_6(F, H)$$



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:

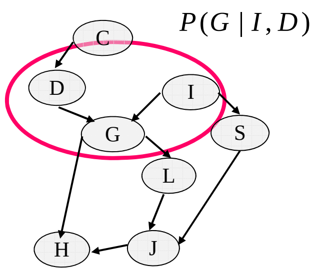
- There exists a directed edge between them in G.
- They are both parents of the same node in G.

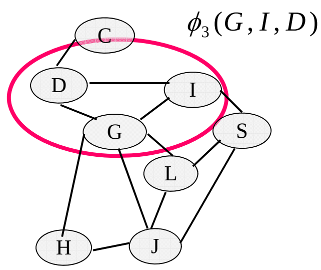


Moral Graphs

Why moralization?

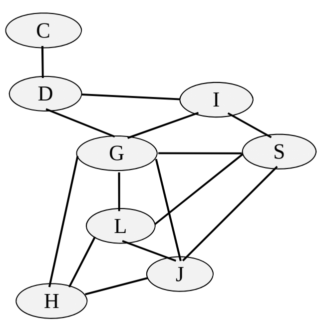
$$P(C,D,G,I,S,L,J,H) =$$
$$= P(C)P(D|C)P(G|I,D)P(S|I)P(L|G)P(J|L,S)P(H|G,J)$$
$$= \phi_1(C)\phi_2(D,C)\phi_3(G,I,D)\phi_4(S,I)\phi_5(L,G)\phi_6(J,L,S)\phi_7(H,G,J)$$



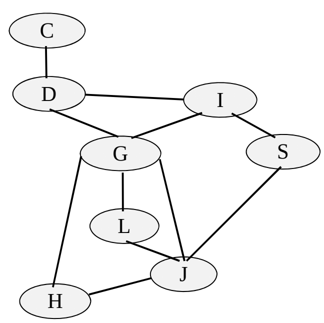


Chordal graphs

Chordal Graph: an undirected graph G whose minimum cycle contains 3 verticies.



Chordal.

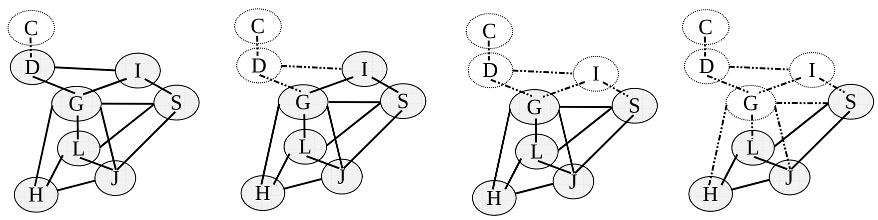


Not Chordal.

Chordal Graphs

Properties:

- There exists an elimination ordering that adds no edges.
- The minimal induced treewidth of the graph is equal to the size of the largest clique - 1.



Triangulation

The process of converting a graph G into a ***chordal graph*** is called Triangulation.

A new graph obtained via triangulation is:

1) Guaranteed to be chordal.
2) Not guaranteed to be (treewidth) optimal.

There exist exact algorithms for finding the ***minimal chordal graphs***, and heuristic methods with a guaranteed upper bound.

Chordal Graphs

- Given a minimum triangulation for a graph *G*, we can carry out the variable-elimination algorithm in the minimum possible time.
- **Complexity** of the optimal triangulation:
  - Finding the minimal triangulation is **NP-Hard**.
- **The inference limit:**
  - Inference time is exponential in terms of the largest clique (factor) in *G*.

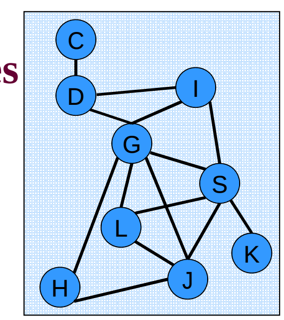
Inference: conclusions

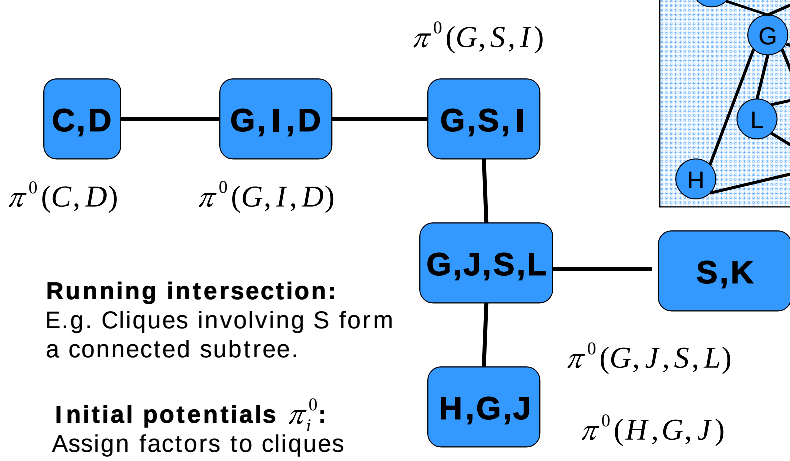
- We cannot escape **exponential costs in the treewidth.**
- But in many graphs the treewidth is much smaller than the total number of variables
- Still a problem: Finding the optimal decomposition is hard
  - But, paying the cost up front may be worth it.
  - Triangulate once, query many times.
  - Real cost savings if not a bounded one.

Clique tree properties

- **A clique tree:**
  - **a tree where nodes correspond to sets of variables**
  - **used for performing probabilistic inferences**
- **Sepset** $S_{ij} = C_i \cap C_j$
  - **separation** set: Variables **X** on one side of sepset are separated from the variables **Y** on the other side in the factor graph given variables in **S**
- **Running intersection property**
  - if C_i and C_j both contain X, then all cliques on the unique path between them also contain X

Inference in clique trees



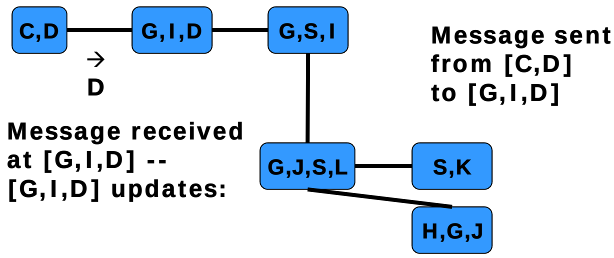


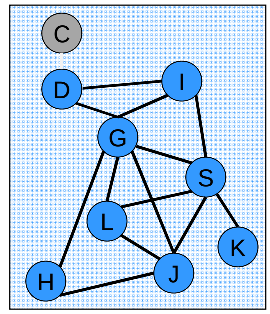
Running intersection:
E.g. Cliques involving S form a connected subtree.

Initial potentials π_i^0**:**
Assign factors to cliques and multiply them.

Message Passing VE

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

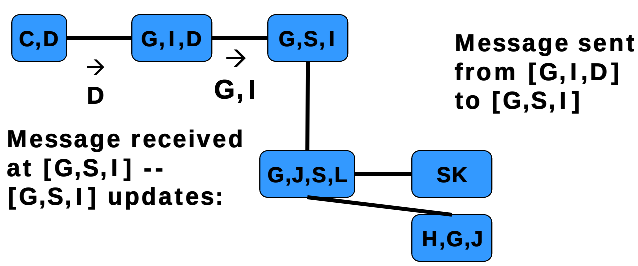


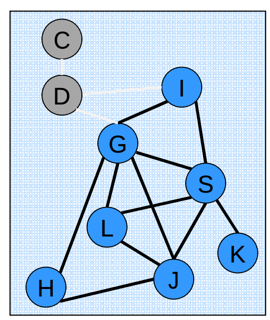


$$\pi_2[G,I,D] = \tau_1(D) \times \pi_2^0[G,I,D]$$

Message Passing VE

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

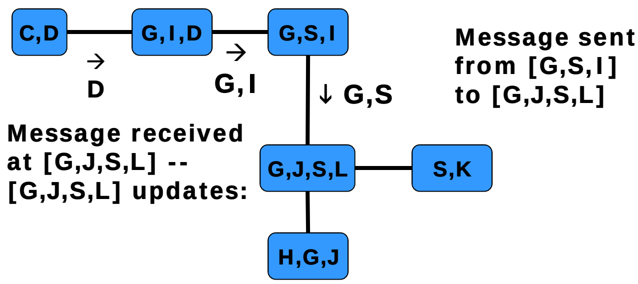


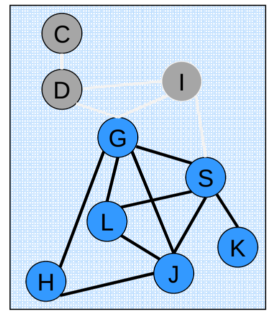


$$\pi_3[G,S,I] = \tau_2(G,I) \times \pi_3^0[G,S,I]$$

Message Passing VE

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



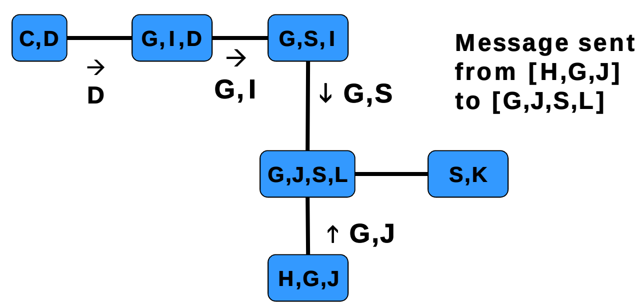


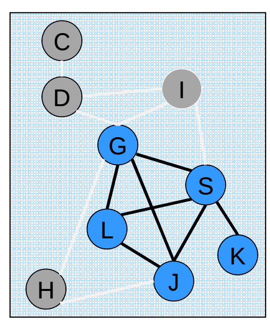
$$\pi_4[G,J,S,L]=\tau_3(G,S)\times\pi_4^0[G,J,S,L] \quad !$$

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

Message Passing VE

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





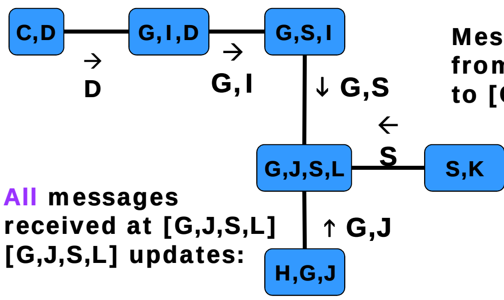
$$\pi_4[G,J,S,L]=\tau_3(G,S)\times\tau_4(G,J)\times\pi_4^0[G,J,S,L]$$

And …

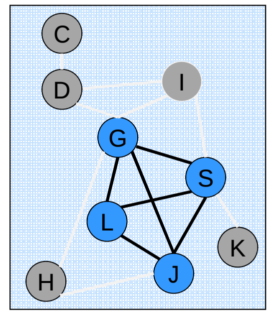
CS 3750 Advanced Machine Learning

Message Passing VE

- Query for P(J)
  - Eliminate K: $\tau_6(S) = \sum_K \pi^0[S,K]$



Message sent from [S,K] to [G,J,S,L]



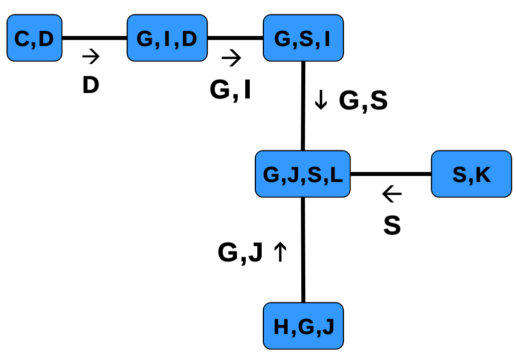
All messages received at [G,J,S,L] [G,J,S,L] updates:

$$\pi_4[G,J,S,L] = \tau_3(G,S) \times \tau_4(G,J) \times \tau_6(S) \times \pi_4^0[G,J,S,L]$$

And calculate P(J) from it by summing out G,S,L

Message Passing VE

- [G,J,S,L] clique potential
- … is used to finish the inference

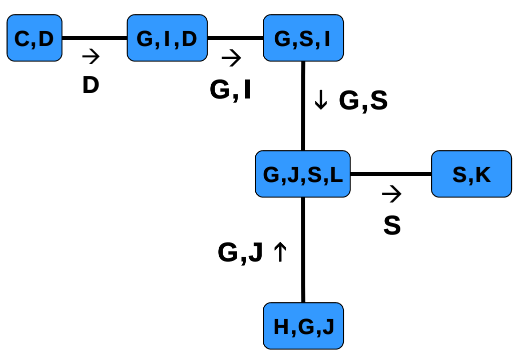


Message passing VE

- Often, **many marginals are desired**
  - Inefficient to re-run each inference from scratch
  - One distinct message per edge & direction
- **Methods :**
  - Compute (**unnormalized**) marginals for any vertex (clique) 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

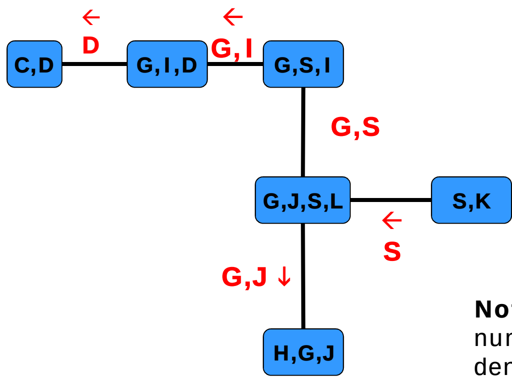
Two-pass message passing VE

- Chose the root clique, e.g. [S,K]
- Propagate messages to the root



Two-pass message passing VE

- Send messages back from the root



Notation:
number the cliques and
denote the messages

$\delta_{i \rightarrow j}$

Message Passing: BP

- Graphical model of a **distribution**
  - More edges = larger expressive power
  - Clique tree also a model of distribution
  - Message passing preserves the model but changes the parameterization
- Different but equivalent algorithm

Factor division

A=1	B=1	0.5
A=1	B=2	0.4
A=2	B=1	0.8
A=2	B=2	0.2
A=3	B=1	0.6
A=3	B=2	0.5

A=1	0.4
A=2	0.4
A=3	0.5

A=1	B=1	0.5/0.4=1.25
A=1	B=2	0.4/0.4=1.0
A=2	B=1	0.8/0.4=2.0
A=2	B=2	0.2/0.4=2.0
A=3	B=1	0.6/0.5=1.2
A=3	B=2	0.5/0.5=1.0

Inverse of factor product

Message Passing: BP

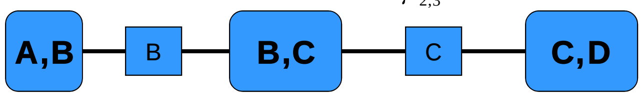
- Each node: multiply all the messages and divide by the one that is coming from node we are sending **the message to**
  - Clearly the same as VE

$$\delta_{i \to j} = \frac{\sum_{C_i - S_{ij}} \pi_i}{\delta_{j \to i}} = \frac{\sum_{C_i - S_{ij}} \prod_{k \in N(i)} \delta_{k \to i}}{\delta_{j \to i}} = \sum_{C_i - S_{ij}} \prod_{k \in N(i) \setminus j} \delta_{k \to i}$$

  - Initialize the messages on the edges to 1

Message Passing: BP

$\mu_{2,3} = 1$



$\pi_1^0(A,B)$ $\quad \pi_2^0(B,C)$ $\quad \longrightarrow \quad$ $\pi_3^0(C,D)$

$$\delta_{2\to3} = \left(\sum_B \pi_2^0(B,C)\right)$$

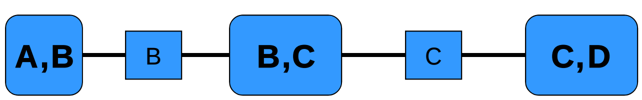
Store the last message on the edge and divide each passing message by the last stored.

$$\pi_3(C,D) = \pi_3^0(C,D)\frac{\delta_{2->3}}{\mu_{2,3}} = \pi_3^0(C,D)\sum_B \pi_2^0(B,C)$$

$$\mu_{2,3} = \delta_{2\to3} = \left(\sum_B \pi_2^0(B,C)\right)$$ New message

Message Passing: BP

$$\mu_{2,3} = \left(\sum_B \pi_2^0(B,C)\right)$$



$\pi_1^0(A,B)$ $\quad \pi_2^0(B,C)$ $\quad \longleftarrow \quad$ $\pi_3(C,D)$

Store the last message on the edge and divide each passing message by the last stored.

$$\pi_3(C,D) = \pi_3^0(C,D)\sum_B \pi_2^0(B,C) = \pi_3^0(C,D)\mu_{2,3}$$

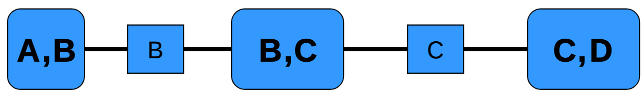
$$\delta_{3->2} = \left(\sum_D \pi_3(C,D)\right)$$

$$\pi_2(B,C) = \pi_2^0(B,C)\frac{\delta_{3->2}}{\mu_{2,3}(C)} = \frac{\pi_2^0(B,C)}{\mu_{2,3}(C)} \times \sum_D \pi_3^0(C,D) \times \mu_{2,3}(C) = \pi_2^0(B,C) \times \sum_D \pi_3^0(C,D)$$

$$\mu_{2,3} = \delta_{3->2} = \left(\sum_D \pi_3(C,D)\right) = \sum_D \pi_3^0(C,D)\sum_B \pi_2^0(B,C)$$ New message

Message Passing: BP

$$\mu_{2,3} = \sum_D \pi_3^0(C,D) \sum_B \pi_2^0(B,C)$$



$\pi_1^0(A,B)$ $\pi_2(B,C)$ $\leftarrow$ $\pi_3(C,D)$

Store the last message on the edge and divide each passing message by the last stored.

$$\pi_3(C,D) = \pi_3^0(C,D) \sum_B \pi_2^0(B,C)$$

$$\delta_{3->2} = \left(\sum_D \pi_3(C,D) \right)$$

$$\pi_2(B,C) = \pi_2^0(B,C) \times \sum_D \pi_3^0(C,D)$$

The same as before

$$\pi_2(B,C) = \pi_2(B,C) \frac{\delta_{3->2}}{\mu_{2,3}(C)} = \pi_2(B,C) \times \frac{\sum_D \pi_3^0(C,D) \times \sum_B \pi_2^0(B,C)}{\sum_D \pi_3^0(C,D) \times \sum_B \pi_2^0(B,C)} = \pi_2(B,C)$$

Message Propagation: BP

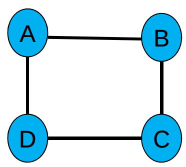
- **Lauritzen-Spiegelhalter algorithm**
- Two kinds of objects: clique and sepset potentials
  - Initial potentials not kept
- Improved "stability" of asynchronous algorithm (repeated messages cancel out)
- **New distribution representation**
  - clique tree potential

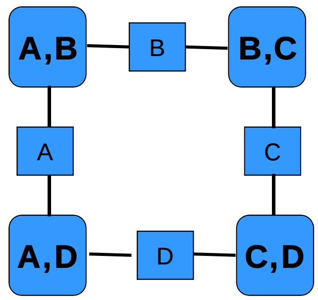
$$\pi_T = \frac{\prod_{C_i \in T} \pi_i(C_i)}{\prod_{(C_i \leftrightarrow C_j) \in T} \mu_{ij}(S_{ij})} = P_F(X)$$

  - Clique tree invariant = P_F

Loopy belief propagation

- The asynchronous BP algorithm works on clique trees
- What if we run the belief propagation algorithm on a non-tree structure?





- Sometimes converges
- If it converges it leads to an approximate solution
- **Advantage:** tractable for large graphs

Loopy belief propagation

- If the BP algorithm converges, it converges to an optimum of the Bethe free energy

See papers:

- Yedidia J.S., Freeman W.T. and Weiss Y. Generalized Belief Propagation, 2000
- Yedidia J.S., Freeman W.T. and Weiss Y. Understanding Belief Propagation and Its Generalizations, 2001