Georgia Institute of Technology

CSIP Center for Signal & Image Processing

ECE/CS/ISYE 8803

Exact Inference in Graphical Models

Module 6: (Part A)
Message Passing Algorithm on General Graphs

Faramarz Fekri

Center for Signal and Information Processing

Overview

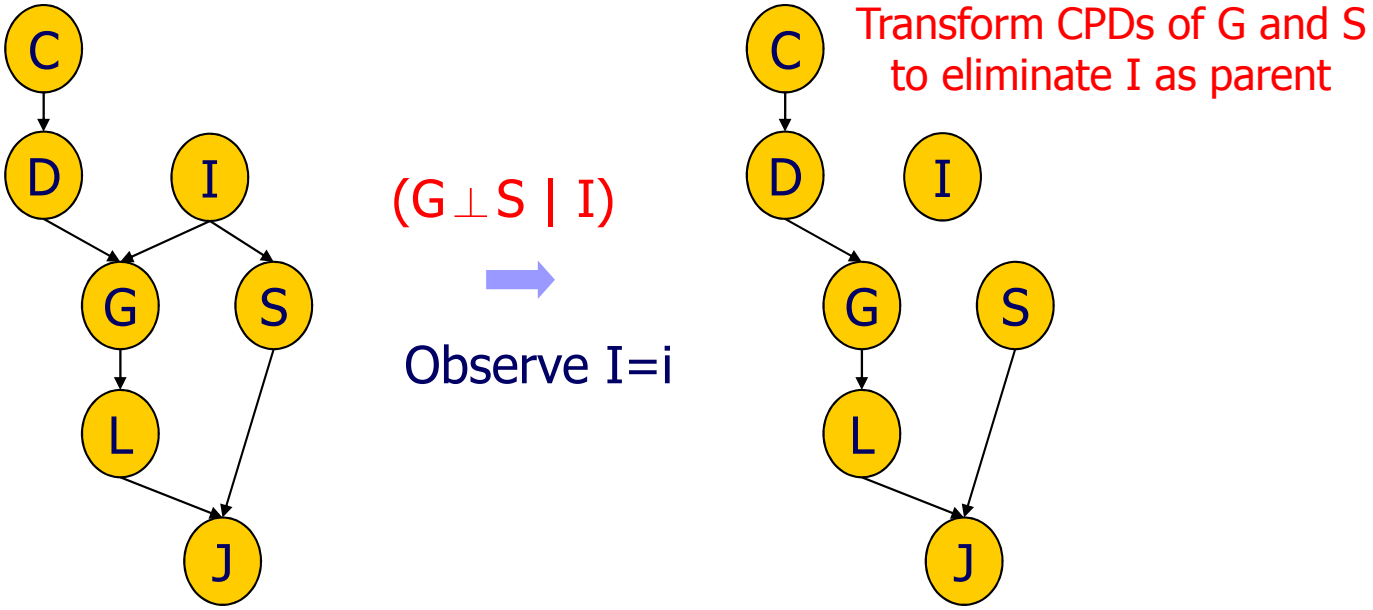
- Inference by Conditioning
- Exact Inference in general graphs
  - Cluster Graph and Clique Tree
  - Constructing clique/junction trees
    - Variable Elimination
    - Graph manipulation

Read Chapter 10 of K&F

Inference by Conditioning

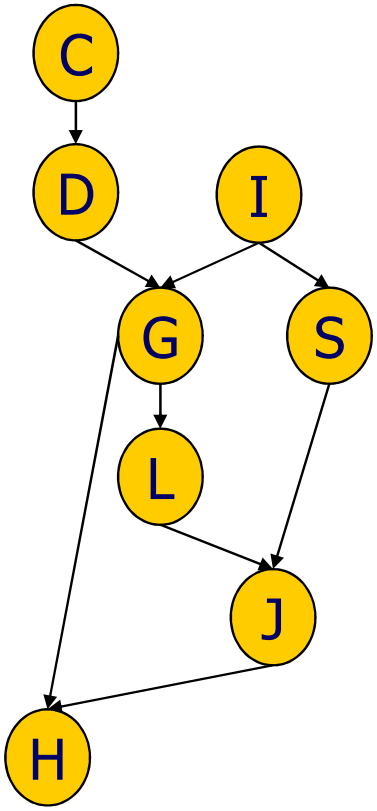
Observing the value of certain variables can simplify the variable elimination process:

- Goal: compute $P(J)$
- General idea
  - Enumerate the possible values i of a variable I
  - Apply Variable Elimination in a simplified network $P(J, I = i)$
  - Aggregate the results $P(J) = \sum_{i \in Val(I)} P(J, I = i)$

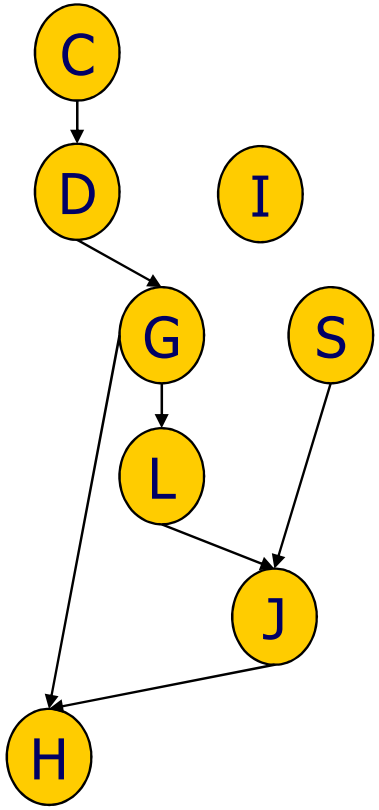


Cutset Conditioning (I)

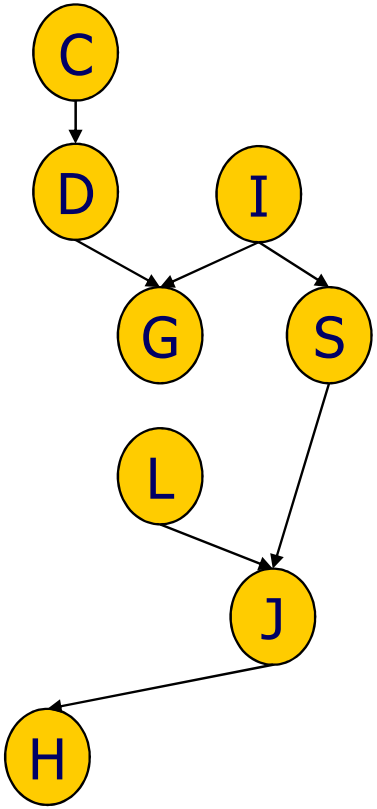
- Select a subset of nodes $\mathbf{X} \subset \mathbf{U}$
- $\mathbf{X}$ is a cutset in G if $G_{\mathbf{X}=\mathbf{x}}$ is a polytree (no loop)



I is not a cutset



G is a cutset



Cutset Conditioning (II)

- Select a subset of nodes $\mathbf{X} \subset \mathbf{U}$
- $\mathbf{X}$ is a cutset in G if $G_{\mathbf{X}=\mathbf{x}}$ is a polytree

- Define the conditional Bayesian network $G_{\mathbf{X}=\mathbf{x}}$
  - $G_{\mathbf{X}=\mathbf{x}}$ has the same variables as G
  - $G_{\mathbf{X}=\mathbf{x}}$ has the same structure as G except that all outgoing edges of nodes in $\mathbf{X}$ are deleted, and CPDs of nodes in which edges were deleted are updated to

$$P_{G_{X=x}}(Y \mid Pa(Y) - \boldsymbol{X}) = P_G(Y \mid Pa(Y), \boldsymbol{X} = \boldsymbol{x})$$

- Compute original $P(Y)$ query using:
  - Exponential in cutset

$$P_G(\boldsymbol{Y}) = \sum_{\boldsymbol{x} \in Val(\boldsymbol{X})} P_{G_{X=x}}(\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{Y})$$

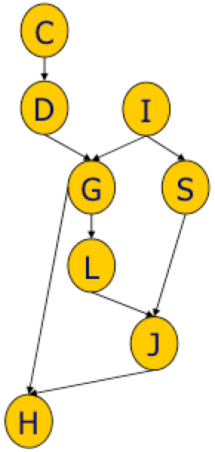
Cutset Conditioning vs Variable Elimination

- In general, both algorithms perform the same set of basic operations (sums and products).
- Cutset Conditioning advantages?
  - Memory saving (**in cases where the factors created by variable elimination are too big to fit in main memory**)
  - Forms the vehicle for a useful approximate inference algorithms (later)

Variable Elimination (another view)

- Variable elimination
  - Each step creates a factor π_i through multiplication
  - A variable is then eliminated in π_i to generate new factor τ_i
  - Process repeated until product contains only query variables

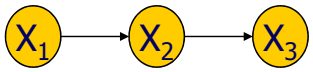
$$P(J) = \sum_L \sum_S P(J \mid L,S) \sum_G P(L \mid G) \sum_H P(H \mid G,J) \sum_I P(I) P(S \mid I) \sum_D P(G \mid D,I) \sum_C P(C) P(D \mid C)$$



- Clique tree inference
  - Another view of the above computation
  - General idea: π_j is a computational data structure which takes "messages" τ_i generated by other factors π_i and generates a message τ_j which is used by another factor π_k

Cluster Graph

- A cluster graph K for factors F is an undirected graph
  - Nodes are associated with a subset of variables $\mathbf{C_i} \subseteq \mathbf{U}$
  - The graph is family preserving: each factor $\phi \in F$ is associated with one node $\mathbf{C_i}$ such that Scope$[\phi] \subseteq \mathbf{C_i}$
  - Each edge $\mathbf{C_i}$–$\mathbf{C_j}$ is associated with a sepset $\mathbf{S_{i,j}} = \mathbf{C_i} \cap \mathbf{C_j}$



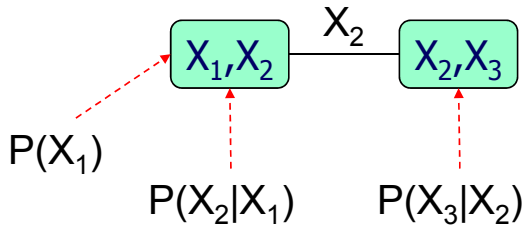
$$F=\{P(X_1),\ P(X_2 \mid X_1), P(X_3 \mid X_2)\}$$

Cluster graph

$C_1 = \{X_1, X_2\}$

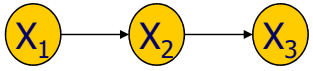
$C_2 = \{X_2, X_3\}$

$S_{1,2} = \{X_2\}$



Variable Elimination and Cluster Graph

- Key: variable elimination defines a cluster graph
  - Cluster $\mathbf{C_i}$ for each factor π_i used in the computation
  - Draw edge $\mathbf{C_i}$–$\mathbf{C_j}$ if the factor generated from π_i is used in the computation of π_j



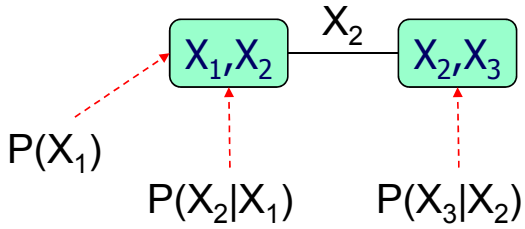
$F=\{P(X_1),\ P(X_2 \mid X_1), P(X_3 \mid X_2)\}$

Cluster graph

$C_1 = \{X_1, X_2\}$

$C_2 = \{X_2, X_3\}$

$S_{1,2} = \{X_2\}$



Cluster Trees

Definition: Running Intersection Property:

- *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 $\mathcal{E}_{\mathcal{T}}$ its edges. We say that $\mathcal{T}$ has the* running intersection property *if, whenever there is a variable X such that $X \in \boldsymbol{C}_i$ and $X \in \boldsymbol{C}_j$, then X is also in every cluster in the (unique) path in $\mathcal{T}$ between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$.*

Definition: Clique Tree:

- *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.

VE Algorithm and Clique Tree

Thm: The cluster graph induced by an execution of variable elimination Alg. is necessarily a clique tree:

Sketch of Proof

Cluster graphs are trees

- In VE, each intermediate factor π_i is used only once
- Hence, each cluster "passes" an edge (message τ_i) to exactly one other cluster

Cluster graphs obey the running intersection property

- If $X \in C_i$ and $X \in C_j$ then X is in each cluster in the (unique) path between C_i and C_j

Property of VE: Scope of the message in VE:

- *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 $C_i \cap C_j$.*

Summarizing VE Property

Remark: Running Intersection property in VE

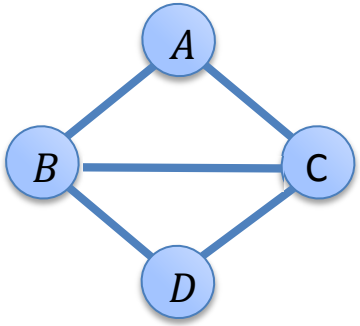
- *Let $\mathcal{T}$ be a cluster tree induced by a variable elimination algorithm over some set of factors Φ. Then $\mathcal{T}$ satisfies the running intersection property.*

Remark: The cluster graph induced by an execution of variable elimination Alg. is necessarily a clique tree.

A cluster in 'clique tree' is also called a clique (rather than a cluster)

Cluster Trees (I)

- Notation
  - Let G be a connected undirected graph. Let $D_1, \dots, D_k$ be the set of maximal cliques in G
  - Let T be a tree structured graph whose nodes are $D_1, \dots, D_k$
  - Let D_i and D_j be two cliques in the graph G connected by an edge, Let $S_{ij} = D_i \cap D_j$ be the separator set between D_i and D_j
  - Let $W_{ij} = D_i - S_{ij}$, the residue set



D_1={A,B,C}

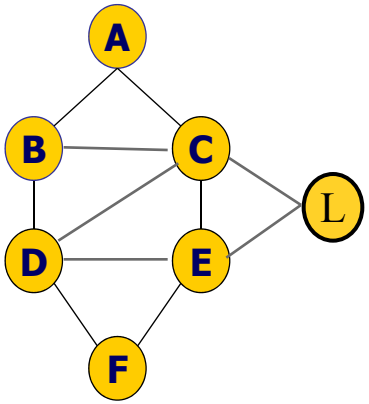
$D_2 = \{B, C, D\}$

S_{12}=$D_1 \cap D_2$={B,C}

Clique Trees (II)

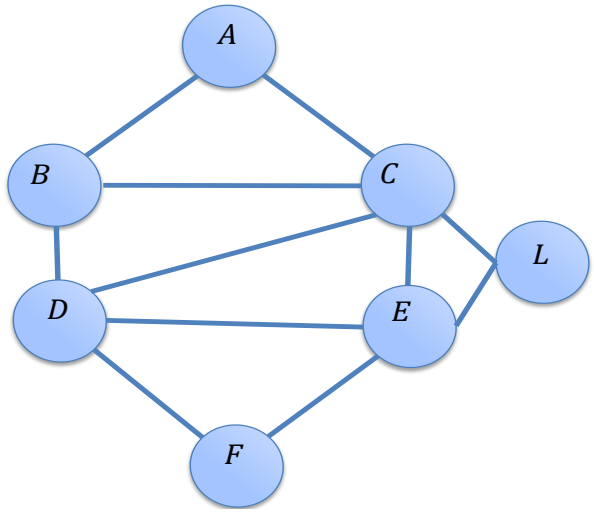
- A tree T is a clique tree for G if
  - Each node corresponds to a clique in G and each maximal clique in G is a node in T
  - Each separator set S_{ij} separates $W_{ij}$and W_{ji}
  - Tree has running intersection property.

- Every undirected chordal graph G has a clique tree T
  - Proof by induction (start from an triangle and add nodes)

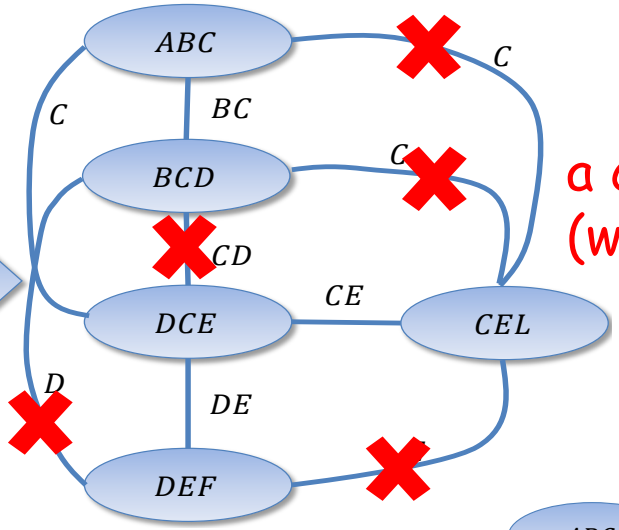


Example of Cluster Graph

- Each node in H corresponds to a clique in G and each maximal clique in G is a node in H
- Each edge is common set for two nodes D_i and D_j

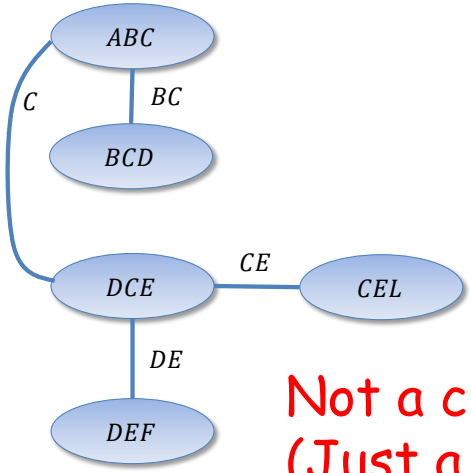


Graph G

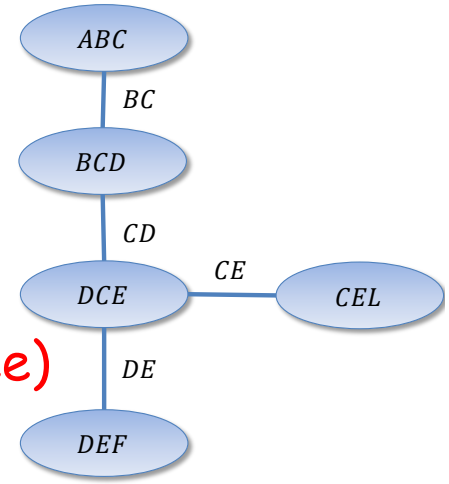


Graph H

a cluster graph (with all edges)



Not a clique tree
(Just a cluster tree)



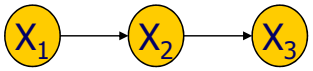
a clique tree
(i.e., junction tree)

Constructing Clique Trees

- Two basic approaches
  a. Using variable elimination (VE)
  b. Using direct graph manipulation

- Using VE
  - Create a cluster C_i for each factor used during a VE run
  - Create an edge between C_i and C_j when a factor generated by C_i is used directly by C_j (or vice versa)

Goal: $P(X_3)$



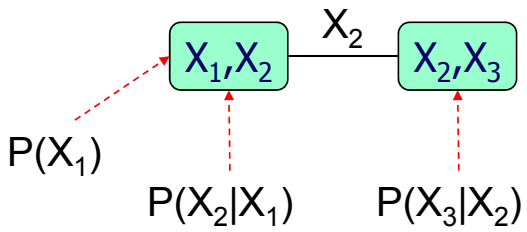
$F=\{P(X_1),\ P(X_2 \mid X_1), P(X_3 \mid X_2)\}$

Cluster graph

$C_1 = \{X_1, X_2\}$

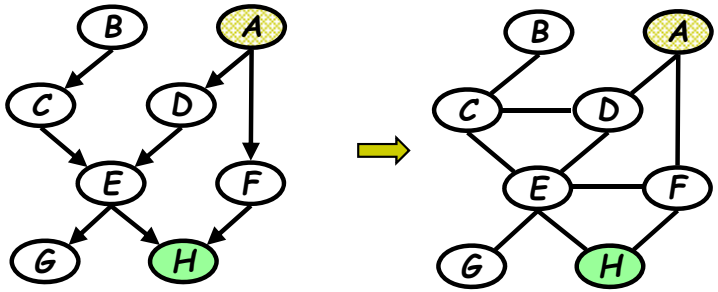
$C_2 = \{X_2, X_3\}$

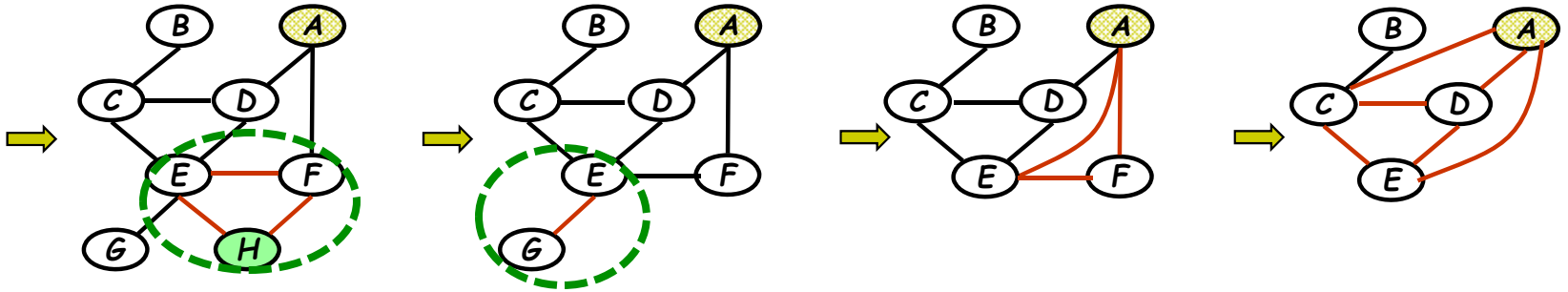
$S_{1,2} = \{X_2\}$



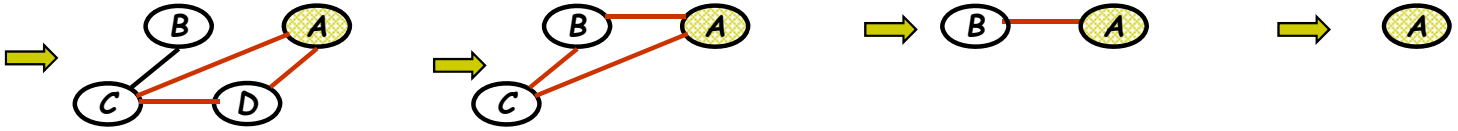
Another Example: VE creates Clique Tree (I)

moralization





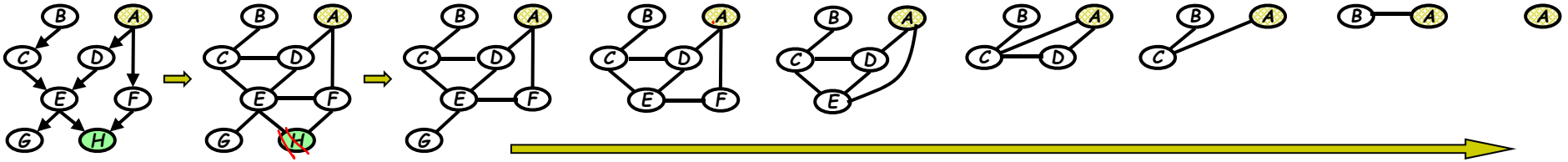
$m_h(e,f)$ $m_g(e)$ $m_f(e,a)$ $m_e(a,c,d)$



$m_d(a,c)$ $m_c(a,b)$ $m_b(a)$

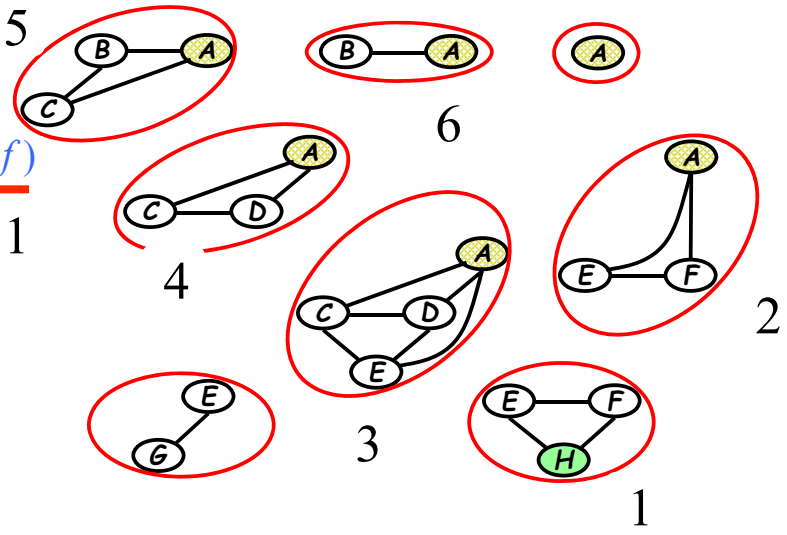
Another Example: VE creates Clique Tree (II)

- A graph elimination algorithm

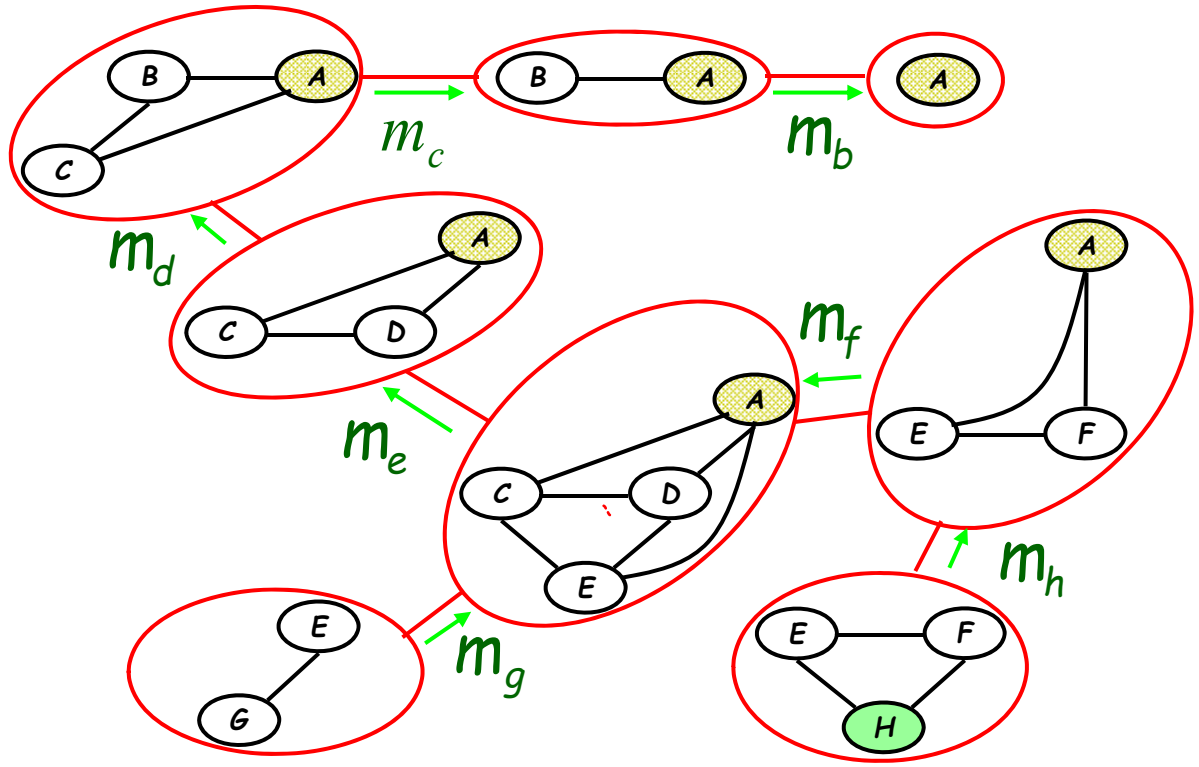


Intermediate terms correspond to the cliques resulted from elimination

$$P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f) \quad 1$$
$$P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)m_h(e,f) \quad 2$$
$$P(a)P(b)P(c|b)P(d|a)P(e|c,d)m_f(a,e) \quad 3$$
$$P(a)P(b)P(c|b)P(d|a)m_e(a,c,d) \quad 4$$
$$P(a)P(b)P(c|b)m_d(a,c) \quad 5$$
$$P(a)P(b)m_c(a,b) \quad 6$$
$$P(a)m_b(a)$$



Resulting Clique Tree, and Message Passing



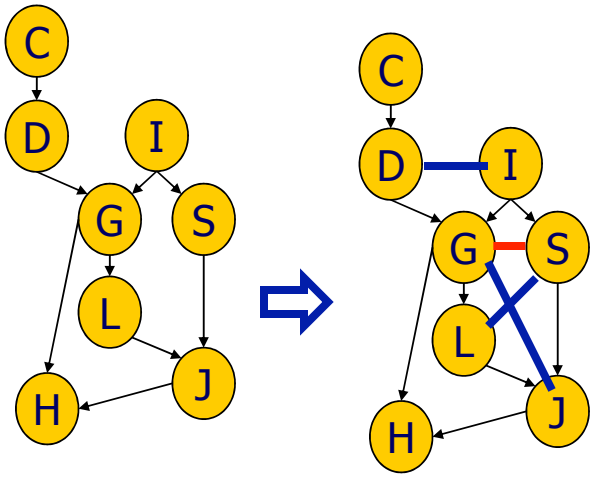
$$m_e(a,c,d) = \sum_e p(e \mid c,d) m_g(e) m_f(a,e)$$

- The overall complexity is determined by the number of the largest elimination clique
  - "good" elimination orderings lead to **small cliques** and hence reduce complexity

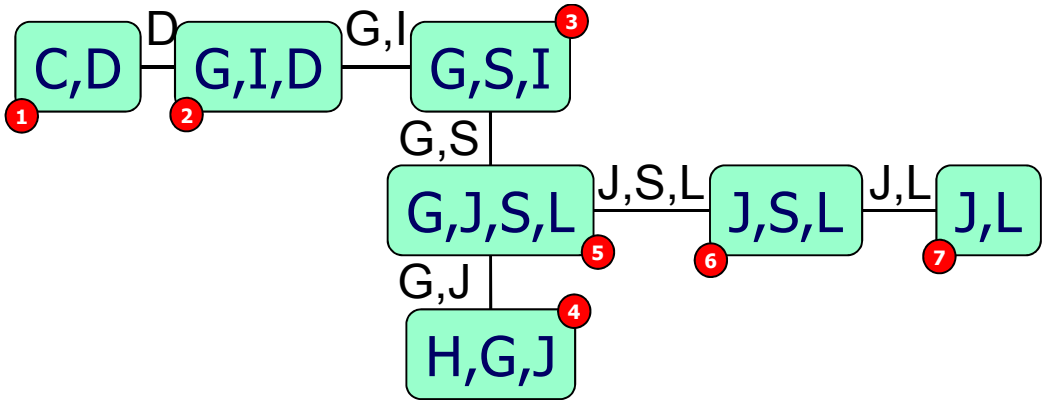
-Xing

Examples: Clique tree via VE

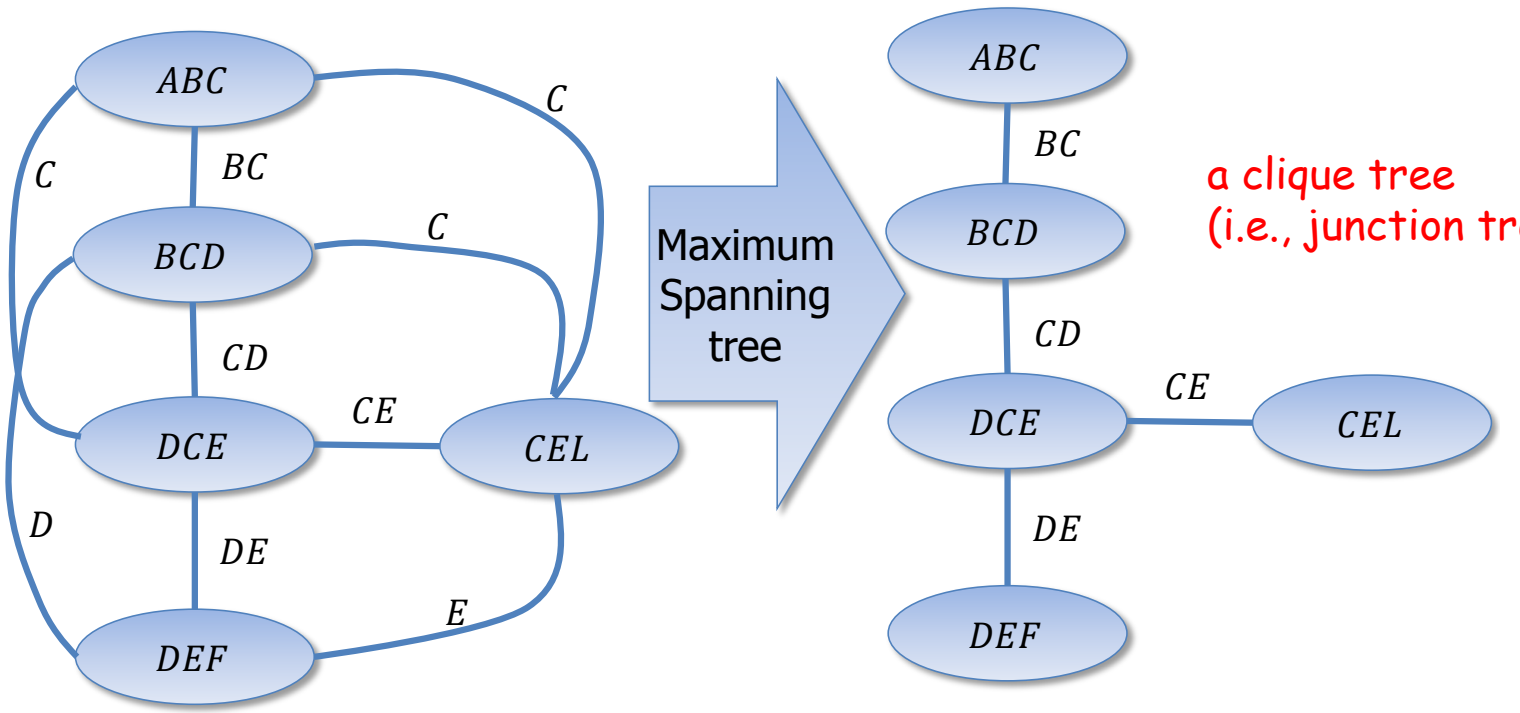
Goal: P(J), Eliminate: C,D,I,H,G,S,L



Moralized/undirected



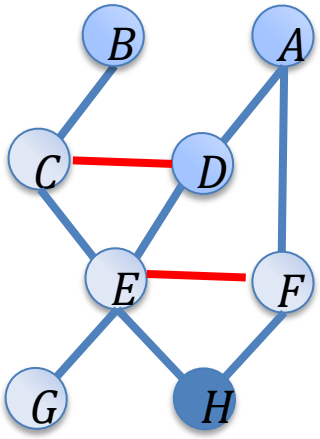
Obtaining Junction/Clique tree from Cluster Graph

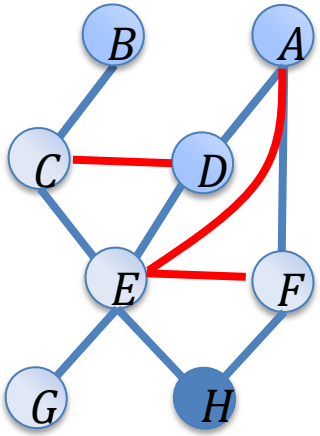


- Run maximum weight spanning tree algorithm on the cluster graph
- Edge weight is the size of the variable on the edge

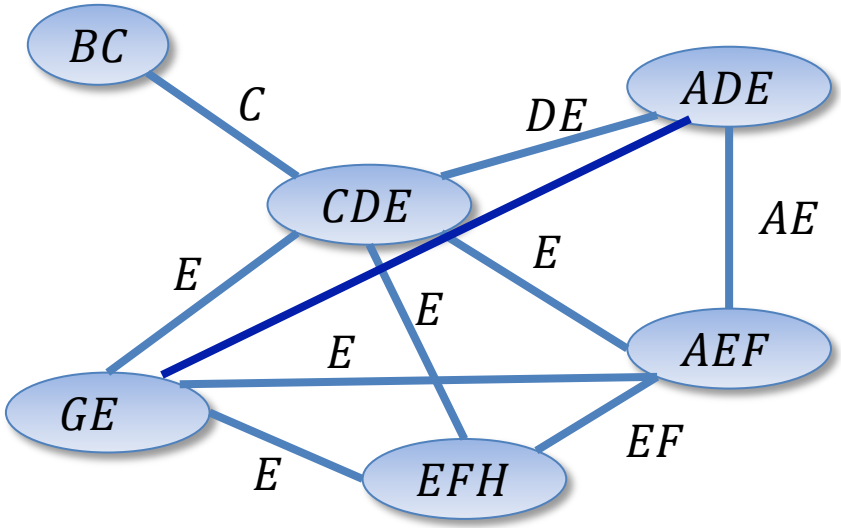
Constructing Clique Tree via Graph Manipulation

- Moralize the graph G





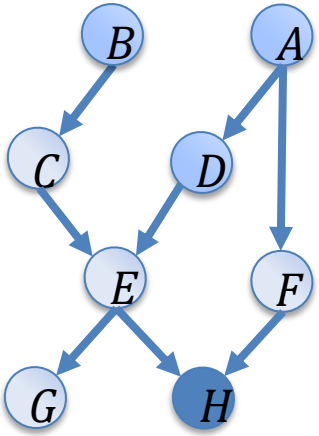
- Triangulate the graph
  - To construct a chordal graph H
- Obtain cluster graph
  - Finding maximal cliques in H



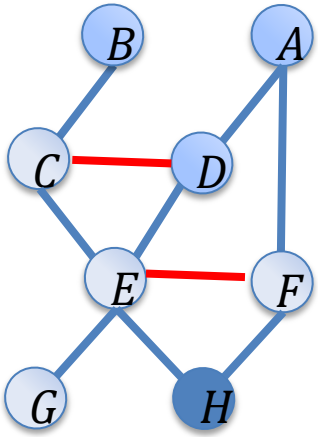
- Obtain junction tree
  - Use maximum weight spanning tree algorithm

Inference: Run local message passing on clique level instead

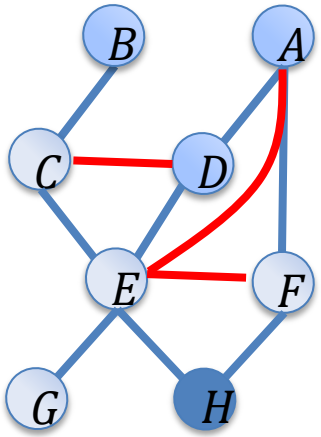
Example: Forming Clique Tree via Graph Manipulation

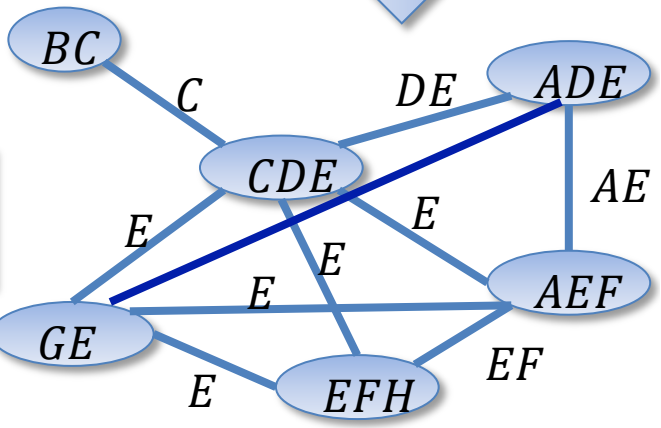


Moralize



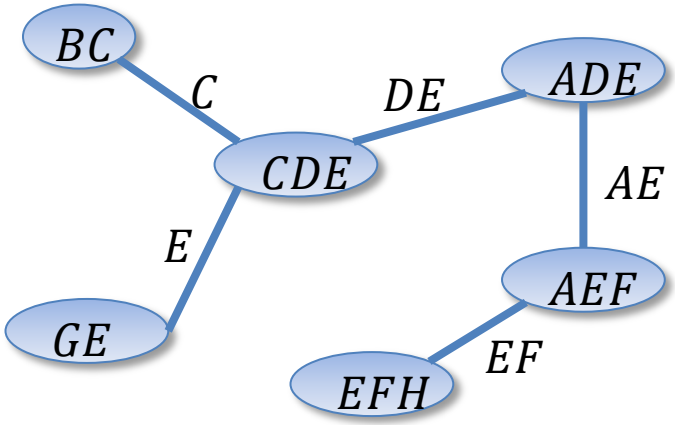
Triangulate





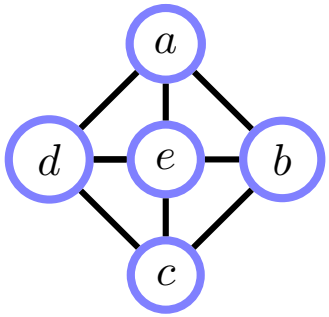
Clique Graph

Maximum spanning tree



Junction Tree

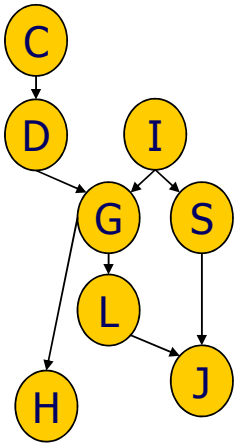
Trangularization



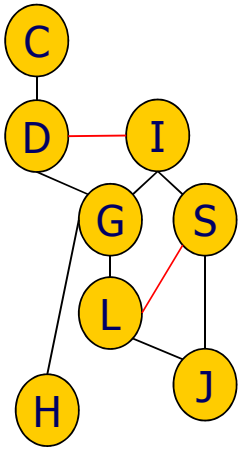
This graph is not triangulated, despite its 'triangular' appearance. The loop a–b–c–d–a does not have a chord.

- There is a Greedy variable elimination Alg. to generate a triangularized graph

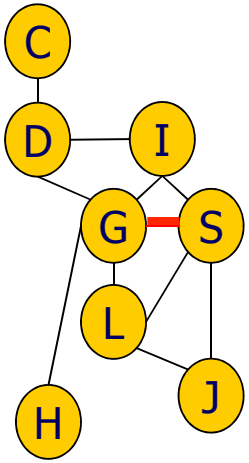
Another Example: Clique Tree via Graph Manipulation



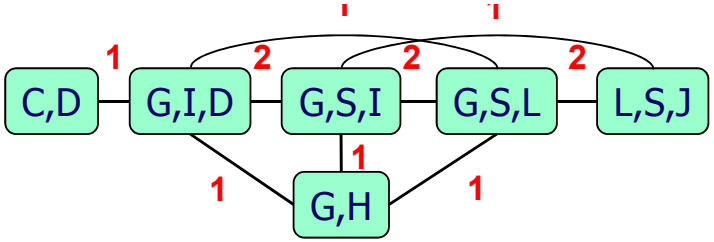
Moralize



Chordal/ trangularize



Cluster Graph



Clique tree