New reading:
Chapter 7 of Koller&Friedman

Variable elimination 2
Clique trees

Graphical Models – 10708
Carlos Guestrin
Carnegie Mellon University

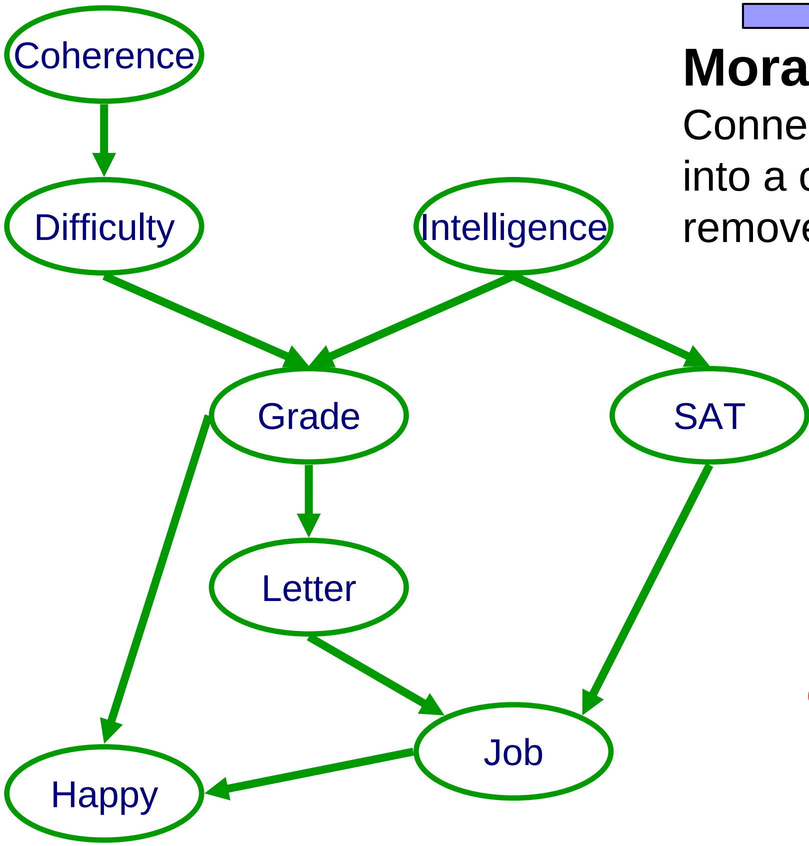
September 28th, 2005

Announcements

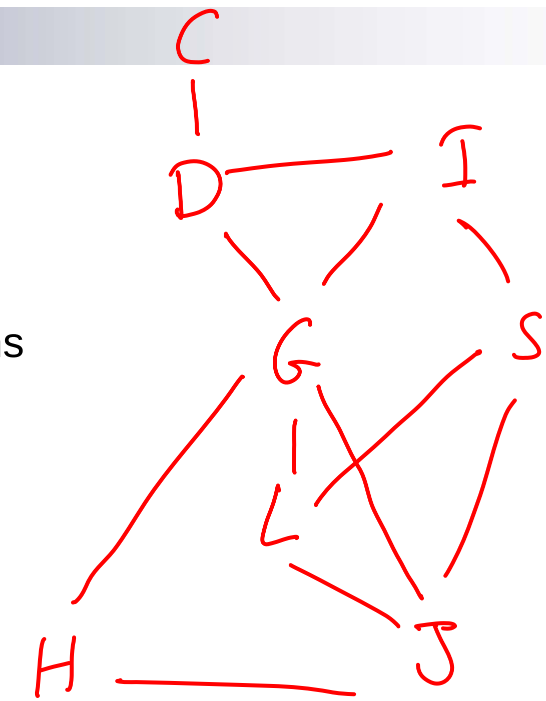
- **Recitation room change!!!**
  - **Wean Hall 4615A (Thursdays 5-6pm)**
- Waiting List
  - Anyone still wants to be registered?

Complexity of variable elimination – Graphs with loops

many trails!



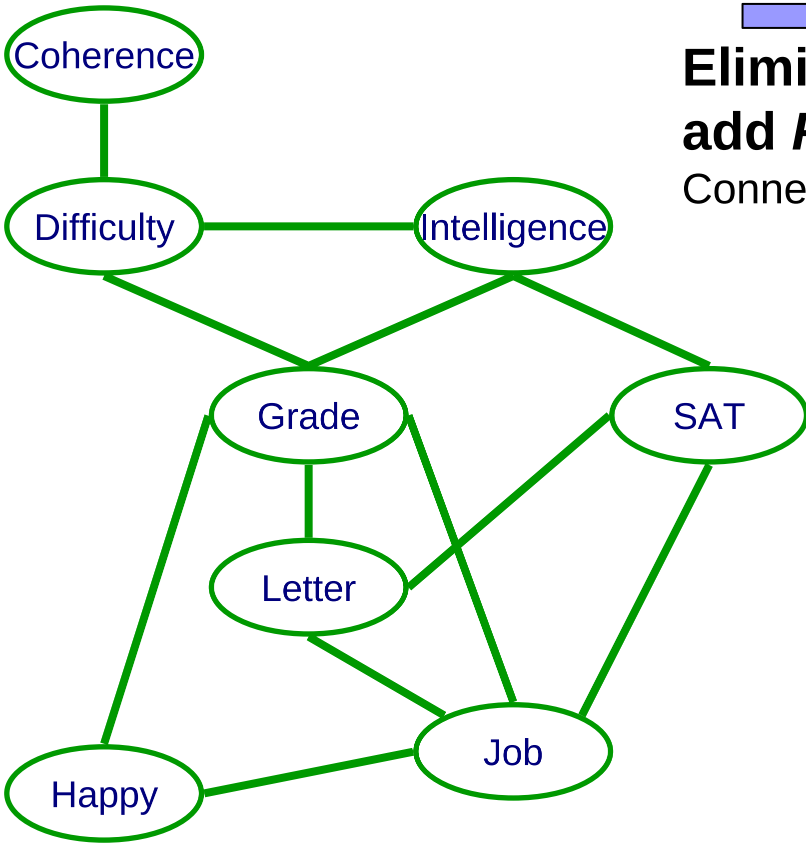
Moralize graph:
Connect parents
into a clique and
remove edge directions



any X_i, X_j that appear in same initial factor of VE connected in moral. graph

Connect nodes that appear together in an initial factor

Eliminating a node – Fill edges

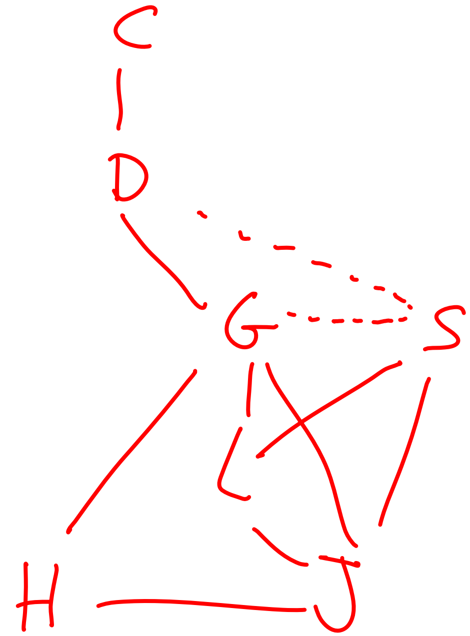


Eliminate variable add *Fill Edges*:
Connect neighbors

eliminate I first

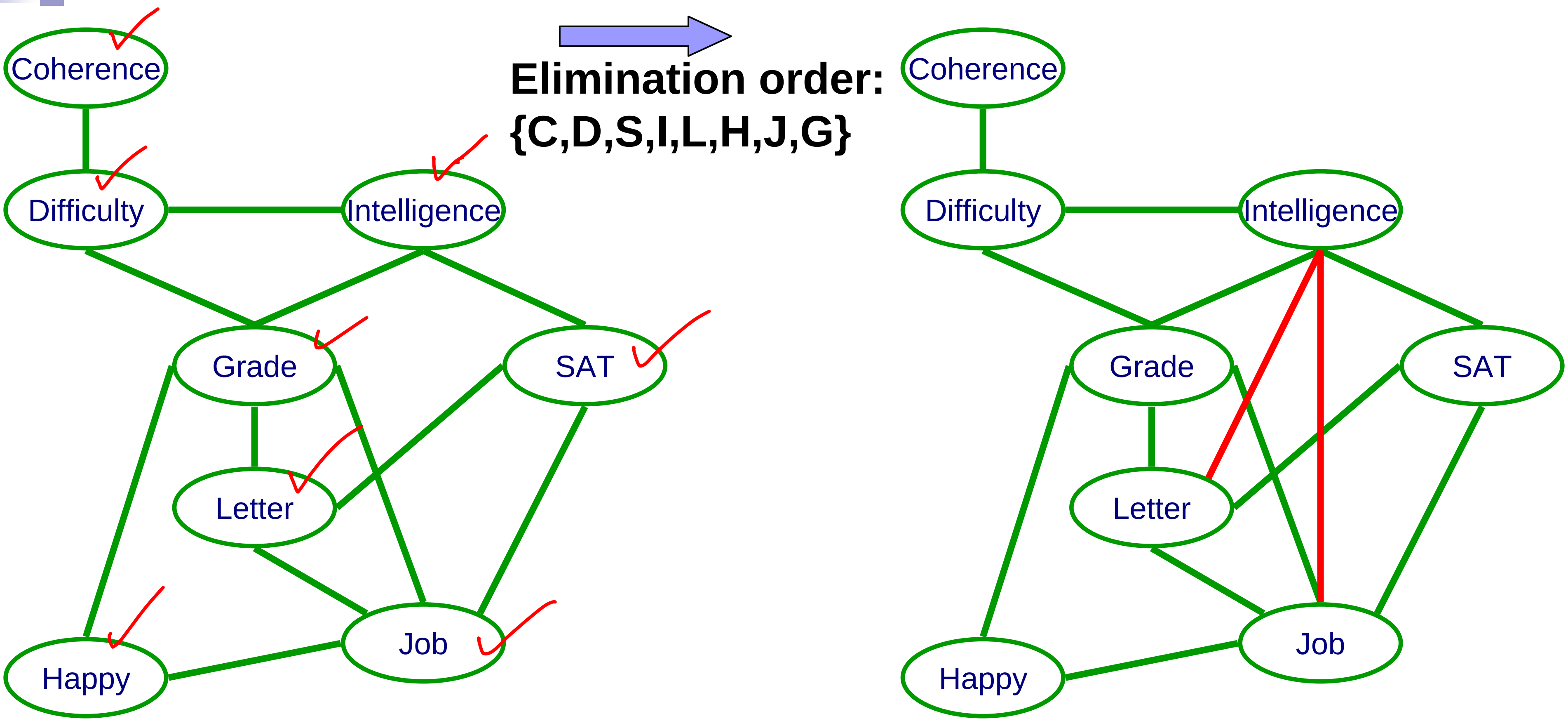
generate

$g_1(D,G,S)$



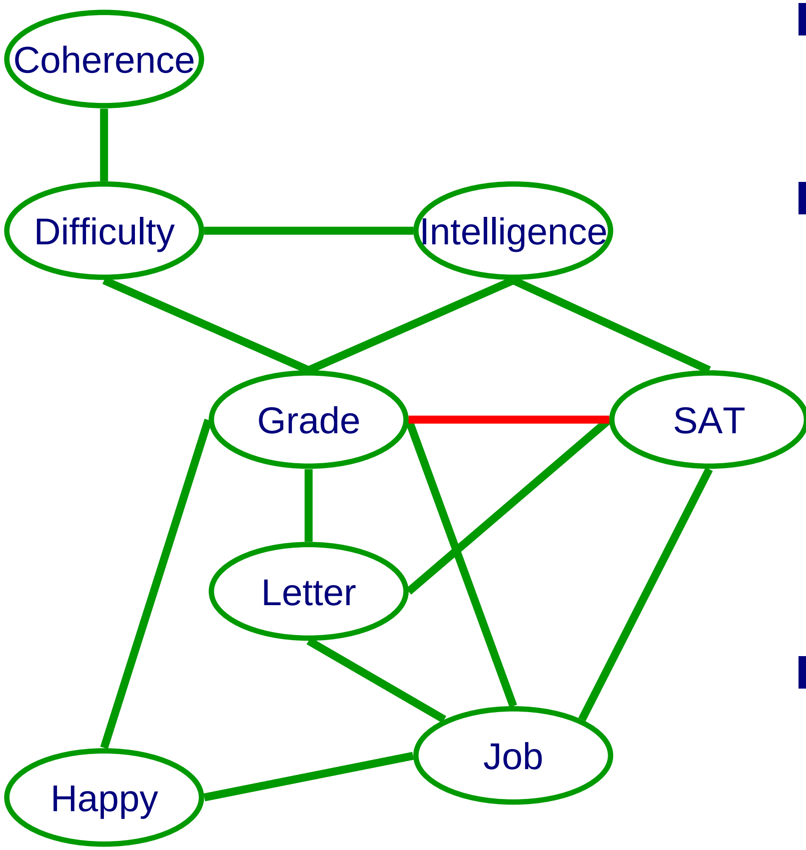
Induced graph

The **induced graph $\mathbf{I_{F\prec}}$** for elimination order $\prec$ has an edge $X_i – X_j$ if X_i and X_j appear together in a factor generated by VE for elimination order $\prec$ on factors F



Induced graph and complexity of VE

Read complexity from cliques in induced graph



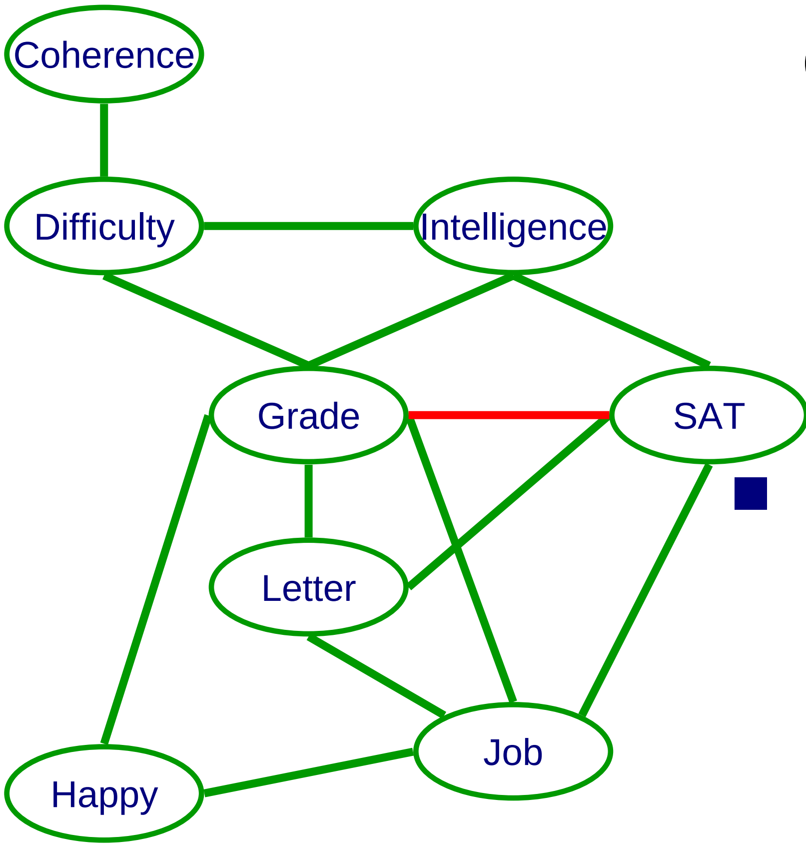
Elimination order:
{C,D,I,S,L,H,J,G}

- Structure of induced graph encodes complexity of VE!!!
- **Theorem**:
  - Every factor generated by VE subset of a maximal clique in $I_{F\prec}$
  - For every maximal clique in $I_{F\prec}$ corresponds to a factor generated by VE
- **Induced width** (or treewidth)
  - Size of largest clique in $I_{F\prec}$ minus 1
  - *Minimal induced width* – induced width of best order $\prec$

Example: Large induced-width with small number of parents

Compact representation $\not\Rightarrow$ Easy inference ☹

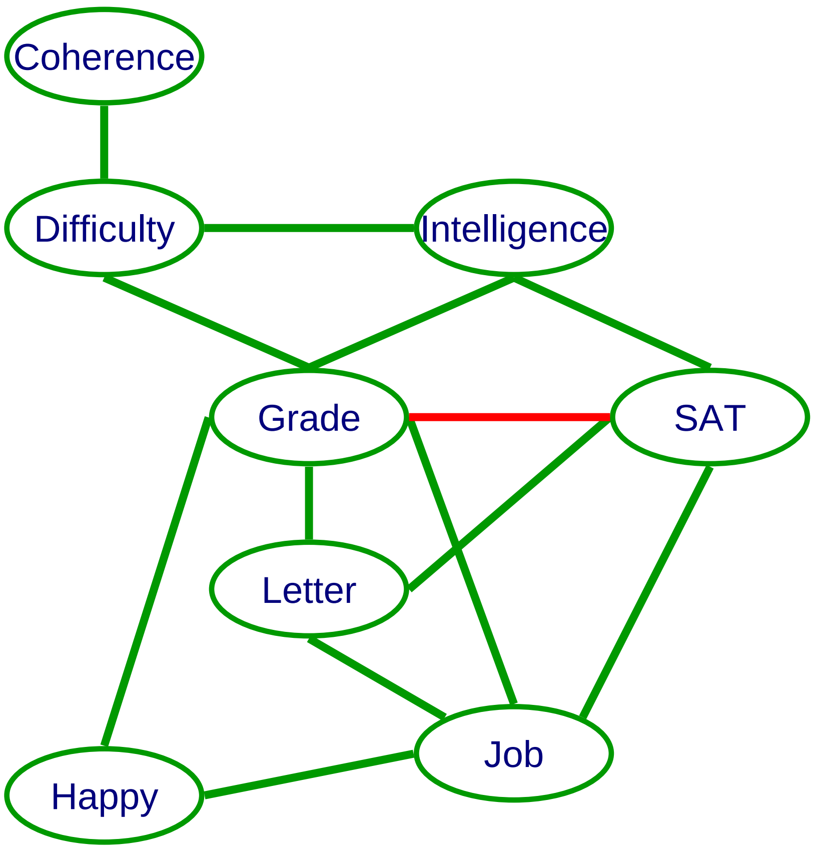
Finding optimal elimination order



Elimination order:
{C,D,I,S,L,H,J,G}

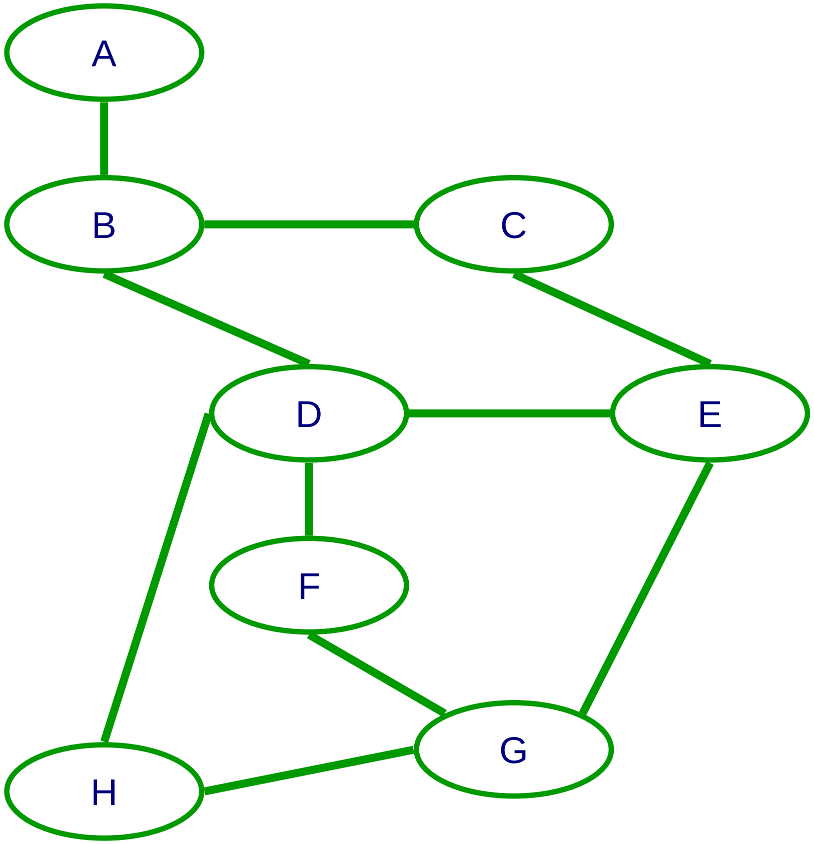
- **Theorem**: Finding best elimination order is NP-complete:
  - Decision problem: Given a graph, determine if there exists an elimination order that achieves induced width $\leq$ K
- **Interpretation**:
  - Hardness of elimination order "orthogonal" to hardness of inference
  - Actually, can find elimination order in time exponential in size of largest clique – same complexity as inference (next week)

Induced graphs and chordal graphs



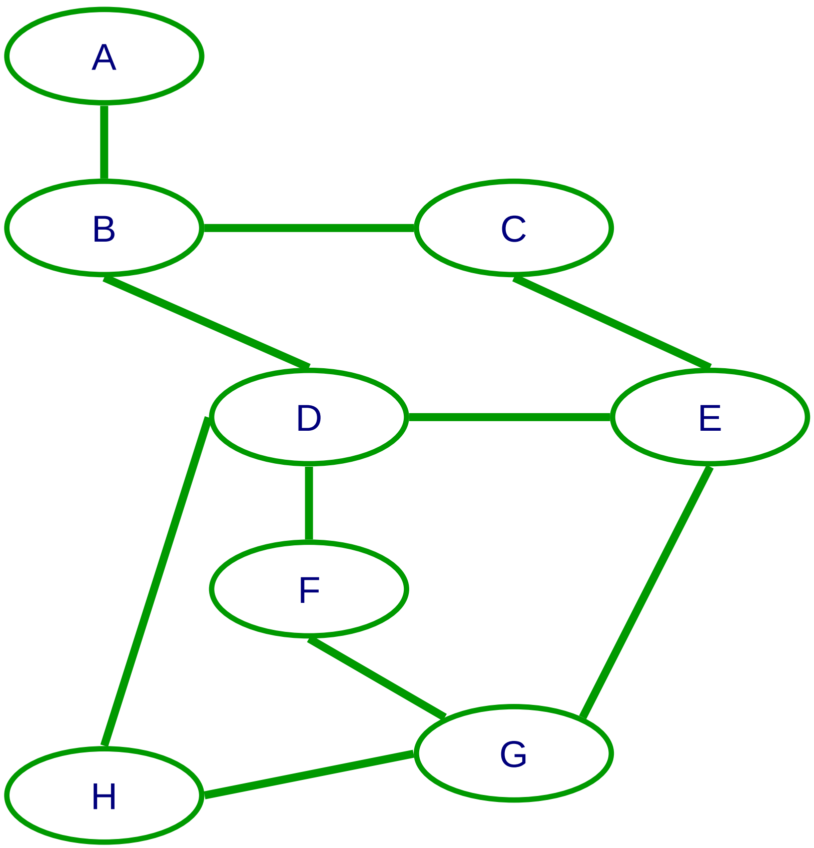
- **Chordal graph**:
  - Every cycle $X_1 - X_2 - \ldots - X_k - X_1$ with $k \geq 3$ has a chord
  - Edge $X_i - X_j$ for non-consecutive i & j
- **Theorem**:
  - Every induced graph is chordal
- "Optimal" elimination order easily obtained for chordal graph

Chordal graphs and triangulation



- **Triangulation**: turning graph into chordal graph
- **Max Cardinality Search**:
  - Simple heuristic
- Initialize unobserved nodes **X** as unmarked
- For k = |**X**| to 1
  - X $\leftarrow$ unmarked var with most marked neighbors
  - $\prec$(X) $\leftarrow$ k
  - Mark X
- **Theorem**: Obtains optimal order for chordal graphs
- Often, not so good in other graphs!

Minimum fill/size/weight heuristics

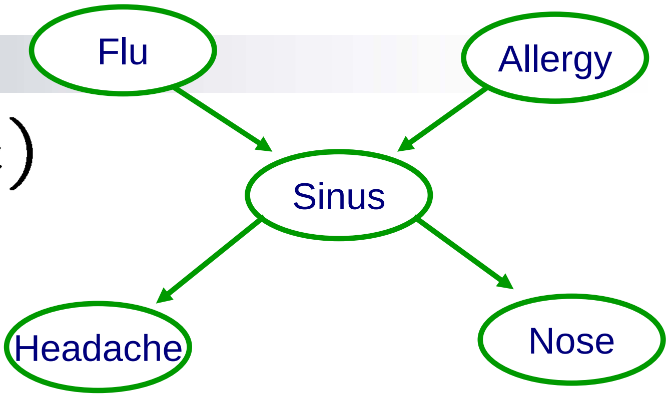


- Many more effective heuristics
  - page 262 of K&F
- **Min (weighted) fill heuristic**
  - Often very effective
- Initialize unobserved nodes **X** as unmarked
- For k = 1 to |**X**|
  - X $\leftarrow$ unmarked var whose elimination adds fewest edges
  - $\prec$(X) $\leftarrow$ k
  - Mark X
  - Add fill edges introduced by eliminating X
- Weighted version:
  - Consider size of factor rather than number of edges

Choosing an elimination order

- Choosing best order is NP-complete
  - Reduction from MAX-Clique
- Many good heuristics (some with guarantees)
- Ultimately, can't beat NP-hardness of inference
  - Even optimal order can lead to exponential variable elimination computation
- In practice
  - Variable elimination often very effective
  - Many (many many) approximate inference approaches available when variable elimination too expensive
  - Most approximate inference approaches build on ideas from variable elimination

Most likely explanation (MLE)



- Query: $\underset{x_1,\ldots,x_n}{\operatorname{argmax}} P(x_1,\ldots,x_n \mid e)$

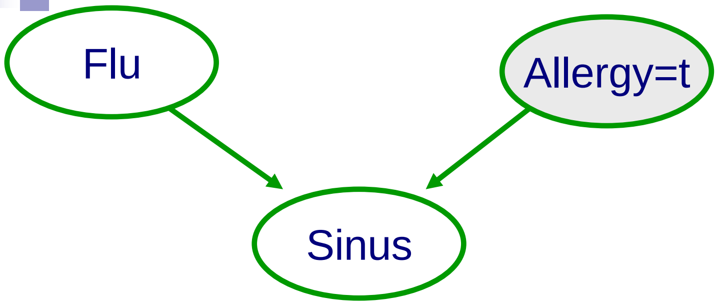
- Using Bayes rule:

$$\underset{x_1,\ldots,x_n}{\operatorname{argmax}} P(x_1,\ldots,x_n \mid e) = \underset{x_1,\ldots,x_n}{\operatorname{argmax}} \frac{P(x_1,\ldots,x_n,e)}{P(e)}$$

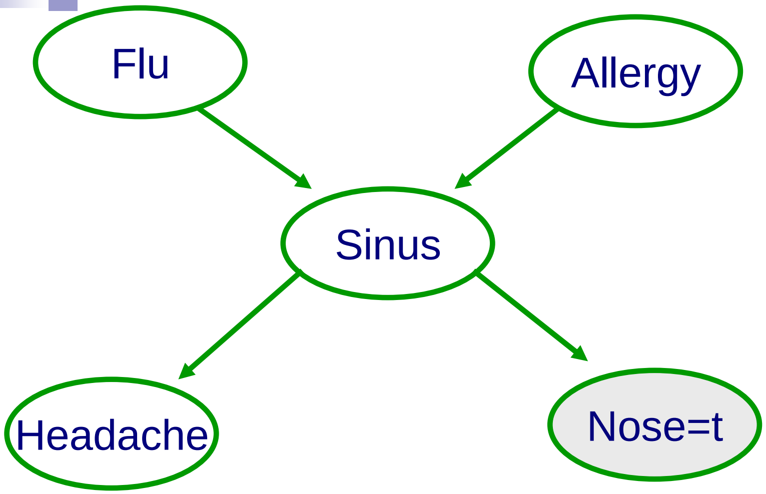
- Normalization irrelevant:

$$\underset{x_1,\ldots,x_n}{\operatorname{argmax}} P(x_1,\ldots,x_n \mid e) = \underset{x_1,\ldots,x_n}{\operatorname{argmax}} P(x_1,\ldots,x_n,e)$$

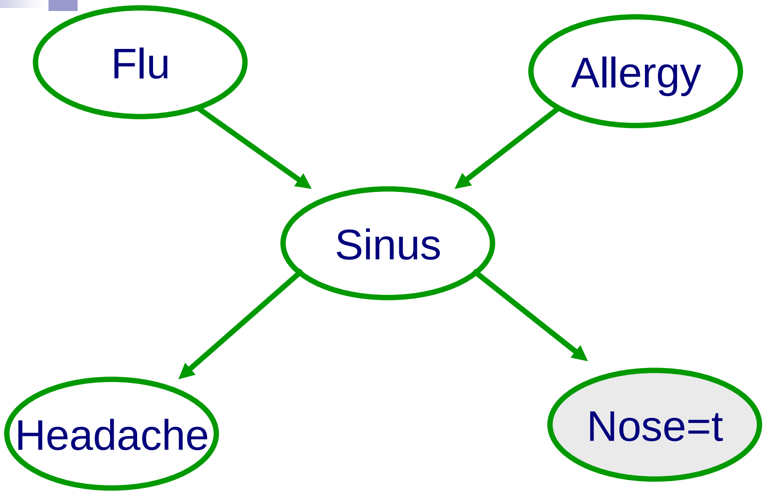
Max-marginalization



Example of variable elimination for MLE – Forward pass



Example of variable elimination for MLE – Backward pass



MLE Variable elimination algorithm – Forward pass

- Given a BN and a MLE query $\max_{x_1,\ldots,x_n} P(x_1,\ldots,x_n,\mathbf{e})$
- Instantiate evidence $\mathbf{E}=\mathbf{e}$
- Choose an ordering on variables, e.g., $X_1, \ldots, X_n$
- For i = 1 to n, If $X_i \notin \mathbf{E}$
  - Collect factors $f_1,\ldots,f_k$ that include X_i
  - Generate a new factor by eliminating X_i from these factors

$$g = \max_{x_i} \prod_{j=1}^{k} f_j$$

  - Variable X_i has been eliminated!

MLE Variable elimination algorithm – Backward pass

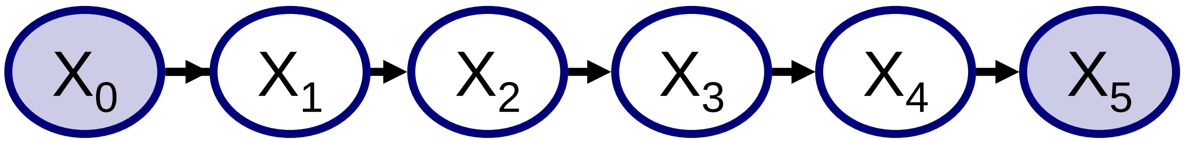
- $\{x_1^*, \ldots, x_n^*\}$ will store maximizing assignment
- For i = n to 1, If $X_i \notin \mathbf{E}$
  - Take factors $f_1, \ldots, f_k$ used when X_i was eliminated
  - Instantiate $f_1, \ldots, f_k$, with $\{x_{i+1}^*, \ldots, x_n^*\}$
    - Now each f_j depends only on X_i
  - Generate maximizing assignment for X_i:

$$x_i^* \in \underset{x_i}{\operatorname{argmax}} \prod_{j=1}^{k} f_j$$

What you need to know

- Variable elimination algorithm
  - Eliminate a variable:
    - Combine factors that include this var into single factor
    - Marginalize var from new factor
  - Cliques in induced graph correspond to factors generated by algorithm
  - Efficient algorithm ("only" exponential in induced-width, not number of variables)
    - If you hear: "Exact inference only efficient in tree graphical models"
    - You say: "No!!! Any graph with low induced width"
    - And then you say: "And even some with very large induced-width" (next week)
- Elimination order is important!
  - NP-complete problem
  - Many good heuristics
- Variable elimination for MLE
  - Only difference between probabilistic inference and MLE is "sum" versus "max"

What if I want to compute $P(X_i|x_0,x_{n+1})$ for each i?



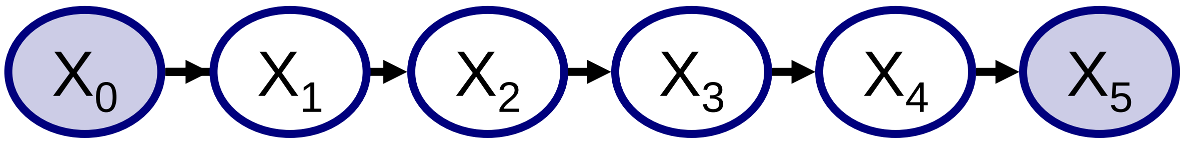
Compute:

$$P(X_i \mid x_0, x_{n+1})$$

Variable elimination for each i?

Variable elimination for each i, what's the complexity?

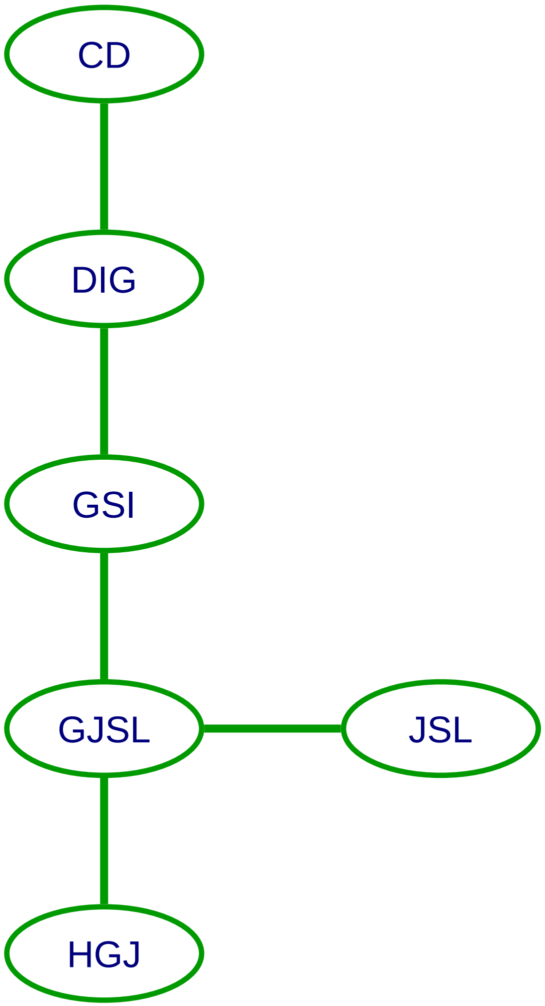
Reusing computation



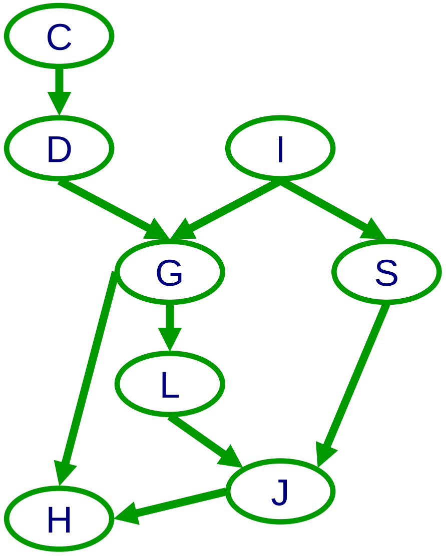
Compute:

$$P(X_i \mid x_0, x_{n+1})$$

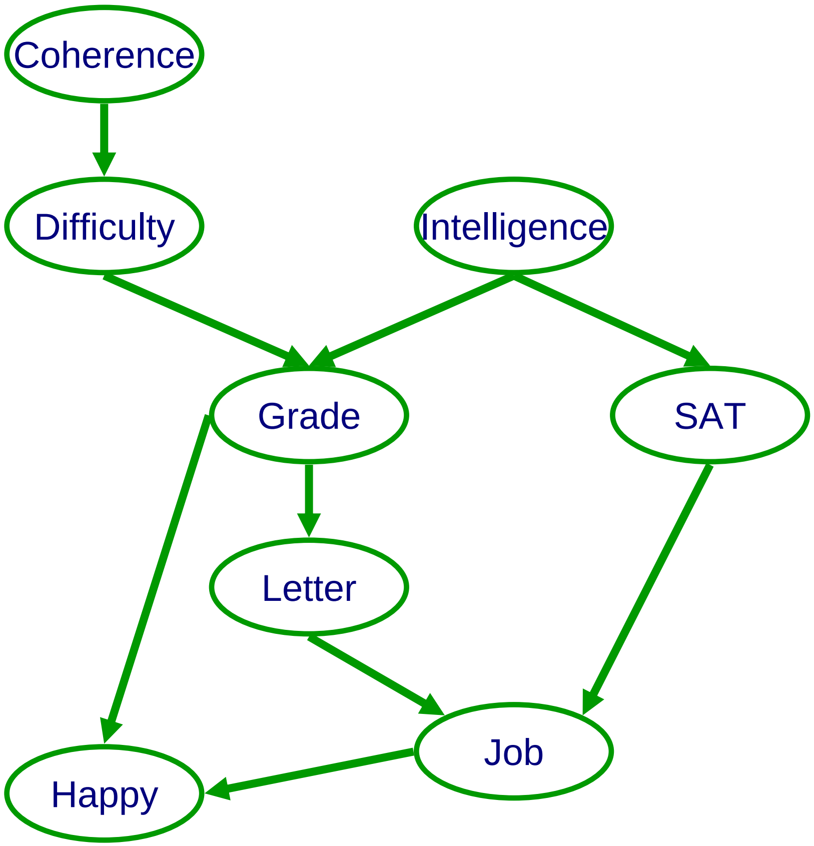
Cluster graph



- **Cluster graph**: For set of factors F
  - Undirected graph
  - Each node i associated with a cluster $\mathbf{C}_i$
  - *Family preserving*: for each factor $f_j \in F$, $\exists$ node i such that scope$[f_i] \subseteq \mathbf{C}_i$
  - Each edge i – j is associated with a separator $\mathbf{S}_{ij} = \mathbf{C}_i \cap \mathbf{C}_j$

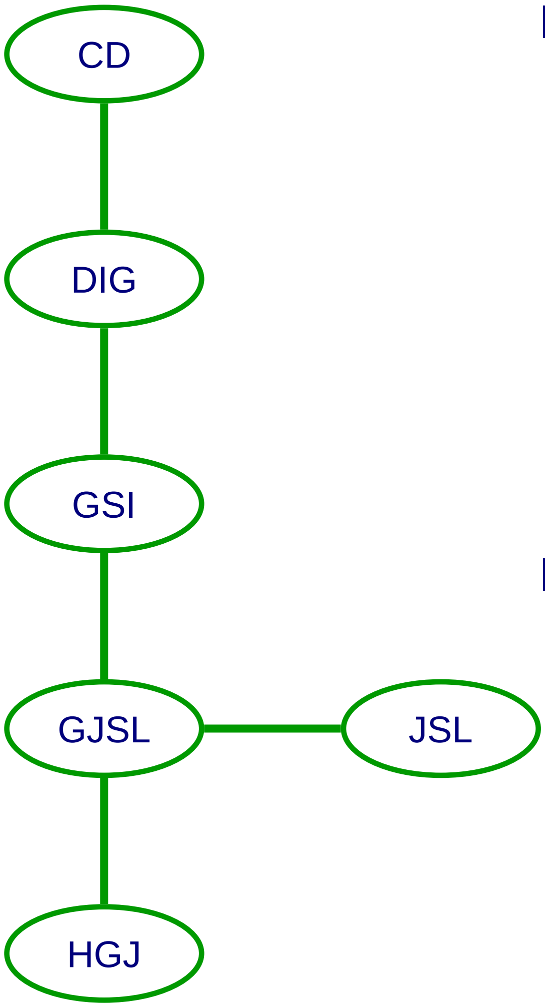


Factors generated by VE



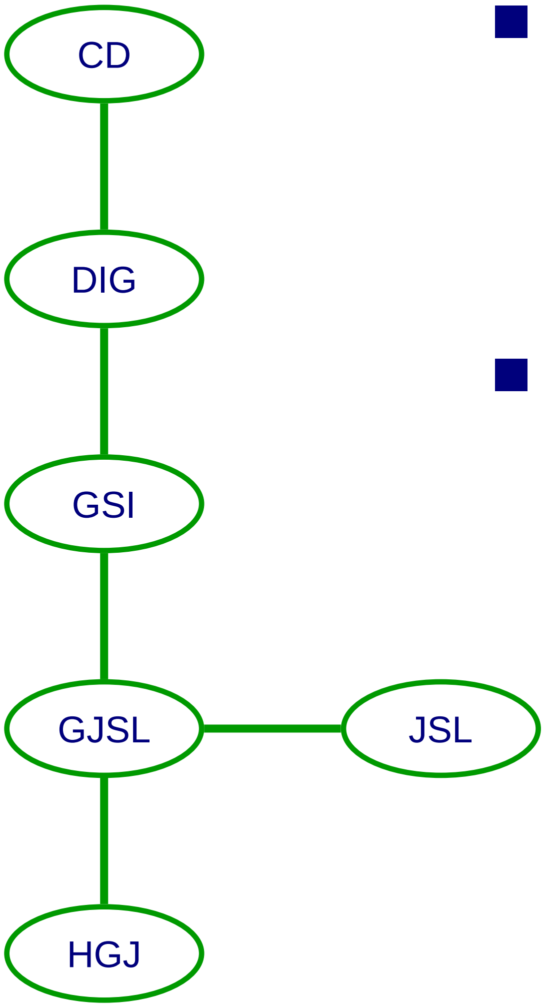
Elimination order:
{C,D,I,S,L,H,J,G}

Cluster graph for VE



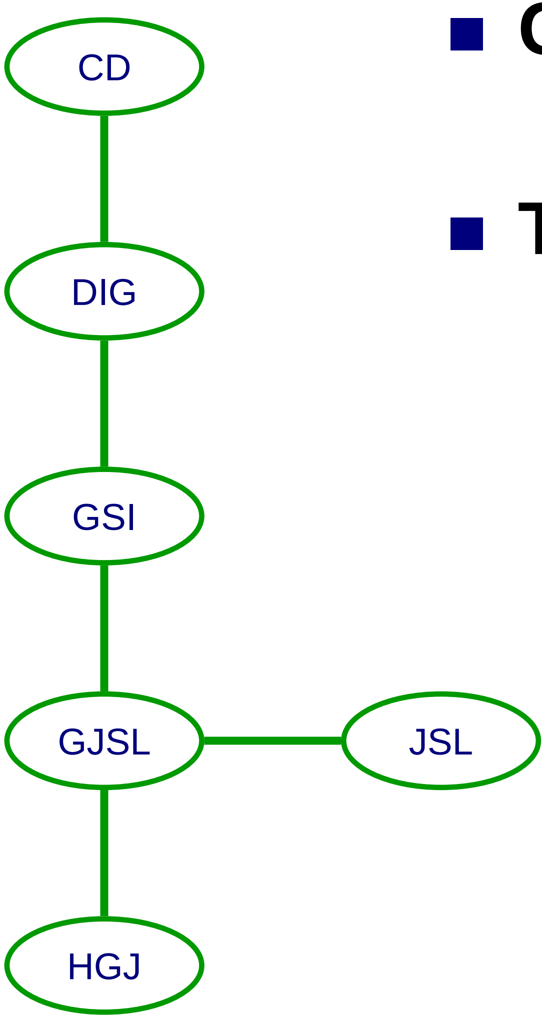
- **VE generates cluster tree!**
  - One clique for each factor used/generated
  - Edge i – j, if f_i used to generate f_j
  - "Message" from i to j generated when marginalizing a variable from f_i
  - Tree because factors only used once
- **Proposition**:
  - "Message" δ_{ij} from i to j
  - Scope[δ_{ij}] $\subseteq$ $\mathbf{S}_{ij}$

Running intersection property



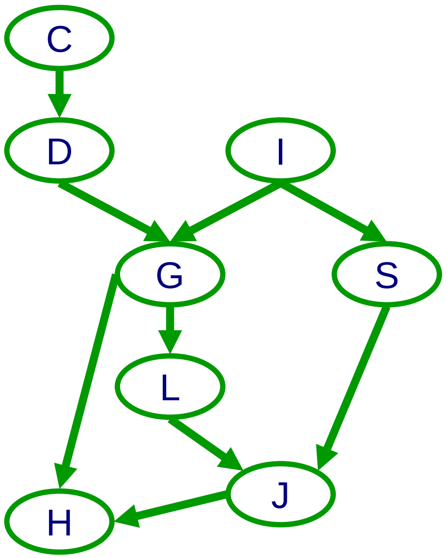
- **Running intersection property (RIP)**
  - Cluster tree satisfies RIP if whenever $X \in \mathbf{C}_i$ and $X \in \mathbf{C}_j$ then X is in every cluster in the (unique) path from $\mathbf{C}_i$ to $\mathbf{C}_j$
- **Theorem**:
  - Cluster tree generated by VE satisfies RIP

Clique tree & Independencies



- **Clique tree (or Junction tree)**
  - A cluster tree that satisfies the RIP
- **Theorem**:
  - Given some BN with structure *G* and factors *F*
  - For a clique tree *T* for *F* consider $\mathbf{C}_i - \mathbf{C}_j$ with separator $\mathbf{S}_{ij}$:
    - **X** – any set of vars in $\mathbf{C}_i$ side of the tree
    - **Y** – any set of vars in $\mathbf{C}_i$ side of the tree
  - Then, $(\mathbf{X} \perp \mathbf{Y} \mid \mathbf{S}_{ij})$ in BN
  - Furthermore, $I(T) \subseteq I(G)$

Variable elimination in a clique tree 1

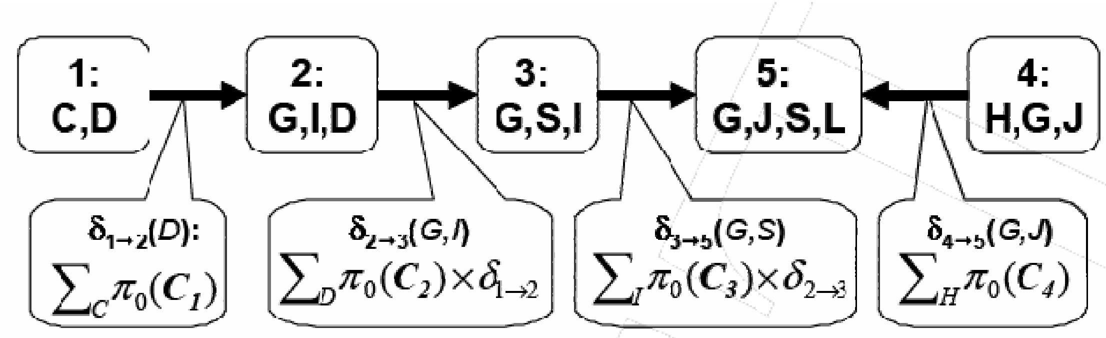


- **Clique tree for a BN**
  - Each CPT assigned to a clique
  - Initial potential $\pi_0(\mathbf{C}_i)$ is product of CPTs

Variable elimination in a clique tree 2

- **VE in clique tree to compute $P(X_i)$**
  - Pick a root (any node containing X_i)
  - Send messages recursively from leaves to root
    - Multiply incoming messages with initial potential
    - Marginalize vars that are not in separator
  - Clique *ready* if received messages from all neighbors

Belief from message

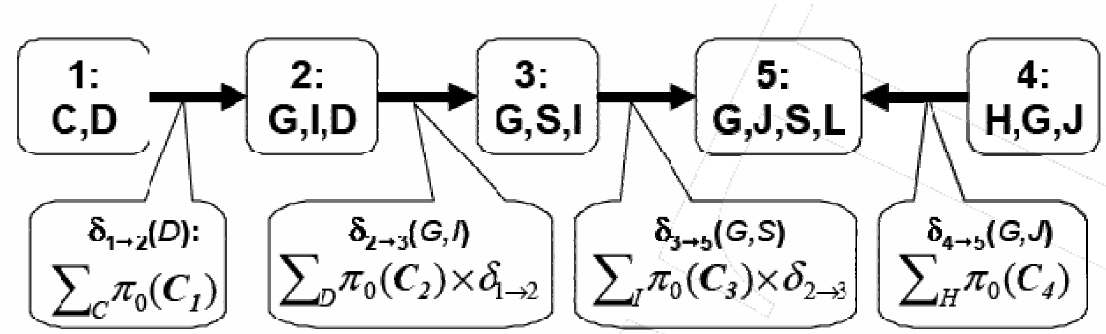


- **Theorem**: When **clique C_i is ready**
  - Receive messages from all neighbors
  - Belief $\pi_i(\mathbf{C}_i)$ is product of initial factor with messages:

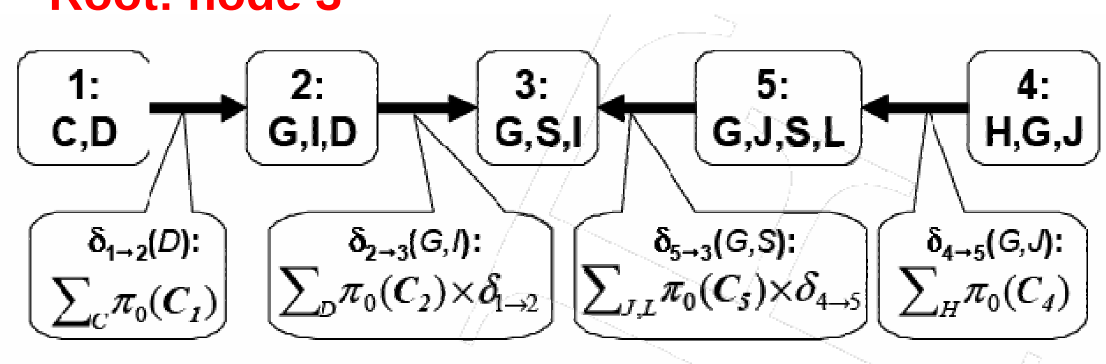
Choice of root

- **Message does not depend on root!!!**

Root: node 5

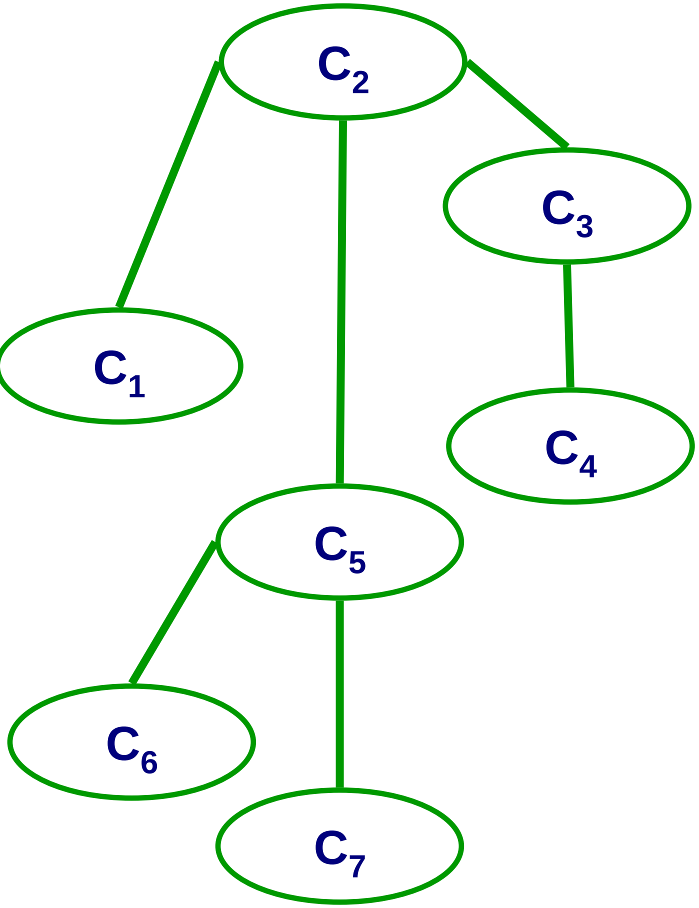


Root: node 3



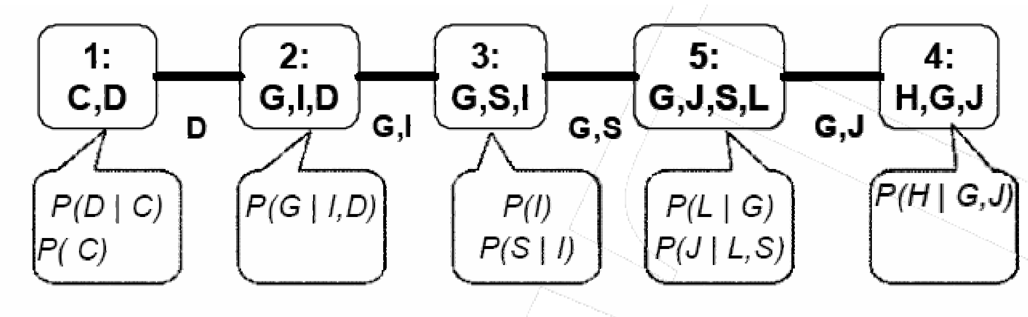
“Cache” computation: Obtain belief for all roots in linear time!!

Shafer-Shenoy Algorithm (a.k.a. VE in clique tree for all roots)



- Clique $\mathbf{C}_i$ *ready to transmit* to neighbor $\mathbf{C}_j$ if received messages from all neighbors but j
  - Leaves are always ready to transmit
- While $\exists$ $\mathbf{C}_i$ ready to transmit to $\mathbf{C}_j$
  - Send message $\delta_{i\to j}$
- Complexity: Linear in # cliques
  - One message sent each direction in each edge
- **Corollary**: At convergence
  - Every clique has correct belief

Calibrated Clique tree



- Initially, neighboring nodes don't agree on "distribution" over separators
- **Calibrated clique tree**:
  - At convergence, tree is *calibrated*
  - Neighboring nodes agree on distribution over separator