Readings:

K&F: 9.1, 9.2, 9.3, 9.4

K&F: 5.1, 5.2, 5.3, 5.4, 5.5

Clique Trees 2

Undirected Graphical Models

Here the couples get to swing!

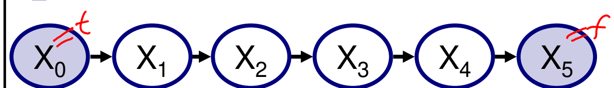
Graphical Models – 10708

Carlos Guestrin

Carnegie Mellon University

October 18th, 2006

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



Compute:

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

Variable elimination for each i? e.g., $X_1 \ldots X_{i-1}, X_{i+1} \ldots X_n$

eliminate X_1 $\quad g_1(X_2) = \sum_{x_1} P(x_0) \cdot P(x_1|x_0) \cdot P(X_2|x_1)$

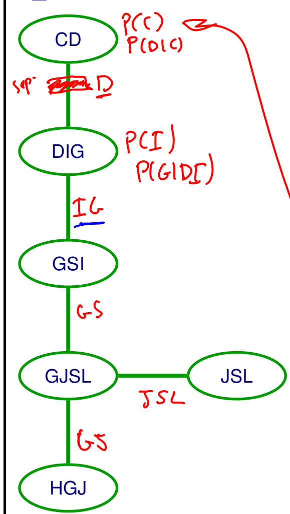
Complexity of $P(X_i \mid x_0, x_{n+1}) = O(n)$

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

run VE n times:

naive $\underline{O(n^2)}$

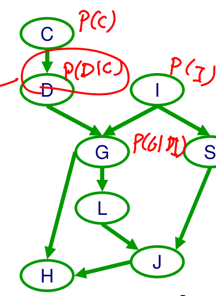
Cluster graph



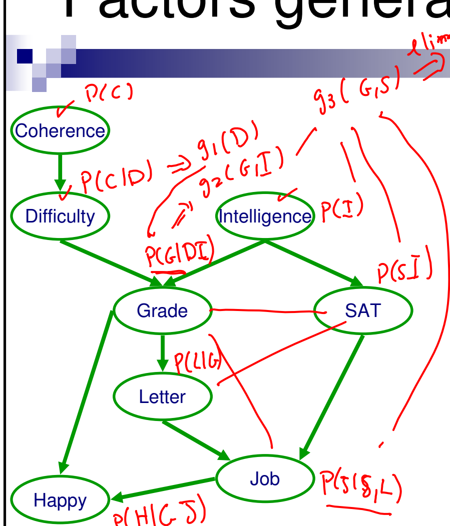
- **Cluster graph**: For set of factors *F*
  - Undirected graph
  - Each node i associated with a cluster $\mathbf{C}_i$
  - *Family preserving*: for each factor $f_j \in F$, $\exists$ node i such that $\text{scope}[f_j] \subseteq \mathbf{C}_i$
  - Each edge i – j is associated with a separator $\mathbf{S}_{ij}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

CPTs

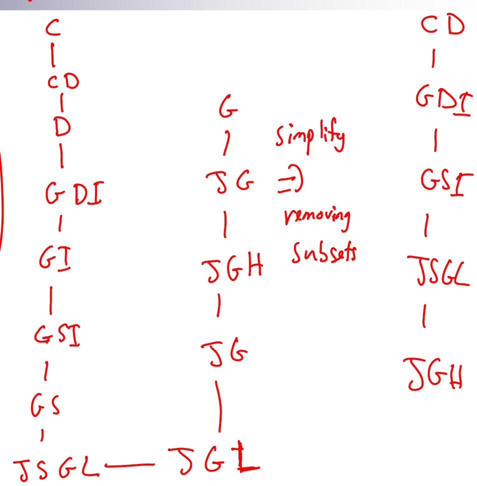
covers



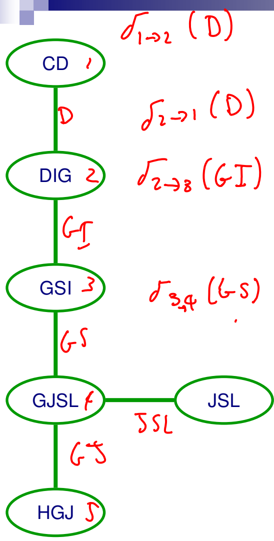
Factors generated by VE



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

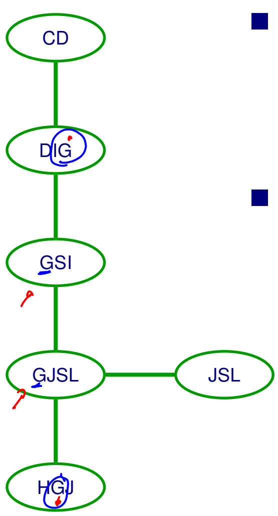


Cluster graph for VE



- **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$[\delta_{ij}] \subseteq \mathbf{S}_{ij}$

Running intersection property

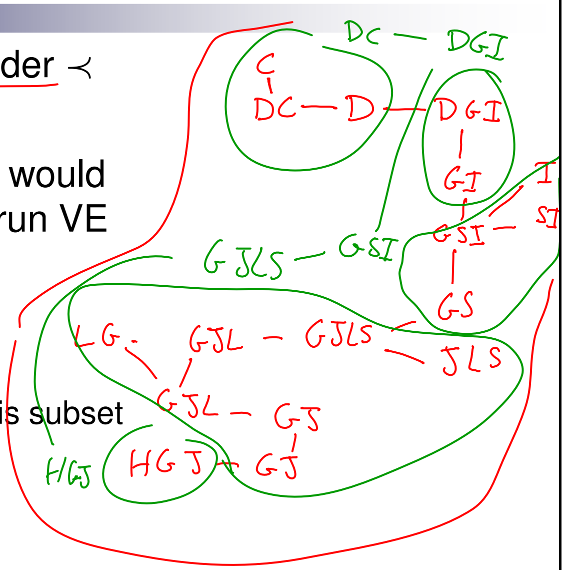


- **Running intersection property (RIP)**
  - Cluster tree satisfies RIP if whenever $X \in \mathbf{C}_i$ and $X \in \mathbf{C}_j$ then X is in every cluster in the (unique) path from $\mathbf{C}_i$ to $\mathbf{C}_j$
- **Theorem**:
  - Cluster tree generated by VE satisfies RIP

Cluster Tree that satisfies RIP is a Junction Tree

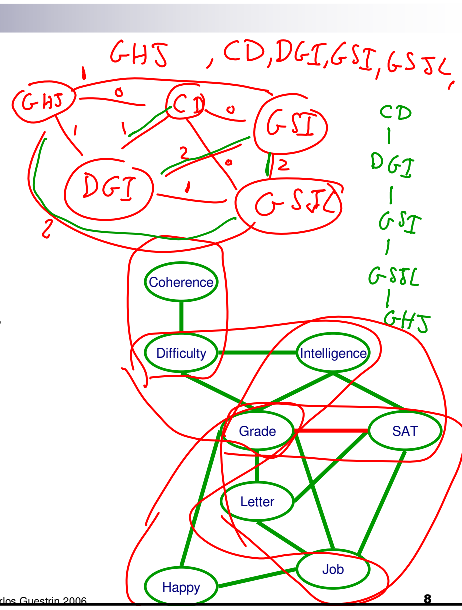
Constructing a clique tree from VE

- Select elimination order $\prec$
- Connect factors that would be generated if you run VE with order $\prec$
- Simplify!
  - Eliminate factor that is subset of neighbor

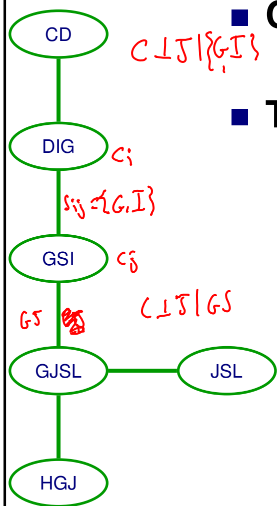


Find clique tree from chordal graph

- Triangulate moralized graph to obtain chordal graph
- Find all maximal cliques
  - NP-complete in general
  - Easy for chordal graphs
  - Max-cardinality search
- Maximum spanning tree finds clique tree satisfying RIP!!!
  - Generate weighted graph over cliques
  - Edge weights (i,j) is separator size – $|\mathbf{C}_i \cap \mathbf{C}_j|$



Clique tree & Independencies



$C \perp J \mid \{G, I\}$

C_i

$S_{ij} = \{G, I\}$

C_j

$C \perp J \mid GS$

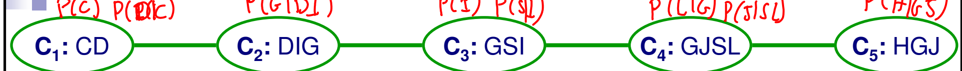
GS

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

if G is I-map for P

Then T still an I-map for P

Variable elimination in a clique tree 1



$P(C)\ P(D|C)$ | $P(G|DI)$ | $P(I)\ P(S|I)$ | $P(L|G)\ P(J|SL)$ | $P(H|GJ)$

$\pi_0(CD) = P(C) \cdot P(D|C)$

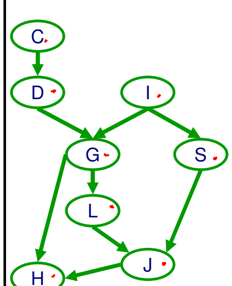
$\pi_0(DIG) = P(G|DI)$

$\pi_0(GSI) = P(I) \cdot P(S|I)$

$\pi_0(GJSL) = P(L|G) \cdot P(J|SL)$

$\pi_0(HGJ) = P(H|GJ)$

table $2 \times 2 \times 2 \times 2 = 16$



- **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

root

C_1: CD —D— C_2: DIG —GI— C_3: GSI —GS— C_4: GJSL —GJ— C_5: HGJ

$\Pi_0(CD)$ $\quad \delta_{1\to 2}(D)$ $\quad \Pi_0(DIG)$ $\quad \delta_{2\to 3}(GI)$ $\quad \Pi_0(GSI)$ $\quad \delta_{3\to 4}$ $\quad \Pi_0(GJSL)$ $\quad \delta_{5\to 4}$ $\quad \Pi_0(HGJ)$

$\delta_{1\to 2}(D) = \sum_c \Pi_0(c,D)$

$\delta_{2\to 3}(GI) = \sum_d \delta_{1\to 2}(d).\Pi_0(dIG)$

$\delta_{3\to 4}(GS) = \sum_i \delta_{2\to 3}(G,i).\Pi_0(G,S,i)$

$\delta_{5\to 4}(GJ) = \sum_h \Pi_0(h,G,J)$

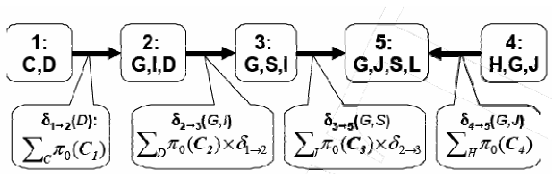
- **VE in clique tree to compute $P(X_i)$**
  - Pick a root (any node containing X_i)
  - Send messages recursively from leaves to root
    - Multiply incoming messages with initial potential
    - Marginalize vars that are not in separator
  - Clique *ready* if received messages from all neighbors

C_4 is ready:
$\Pi_0(GJSL).\delta_{3\to 4}(GS)$
$.\delta_{5\to 4}(GJ)$
$= \Pi(GJSL)$
$= P(GJSL)$

Belief from message

$S_{ij} \subseteq C_i$

$\Pi_i(C_i) = \Pi_0(C_i) \prod_{j \in Nb(i)} \delta_{i\to j}(S_{ij})$



- **Theorem**: When **clique C_i is ready**
  - Received messages from all neighbors
  - Belief $\pi_i(C_i)$ is product of initial factor with messages:

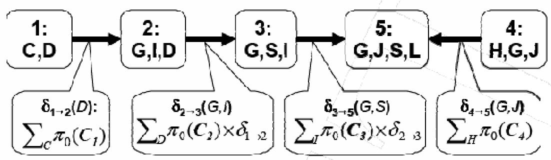
This process is equivalent to VE in the BN for some ordering

Choice of root

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

Root: node 5

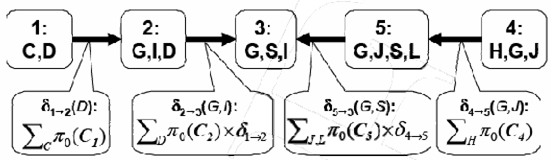
belief $\pi_5(G,J,S,L)$



only difference is computing $\delta_{3\to5}$ v. $\delta_{5\to3}$

Root: node 3

belief $\pi_3(G,S,I)$



compute $\delta_{1\to2}, \delta_{2\to3}, \delta_{35}, \delta_{5\to4}$

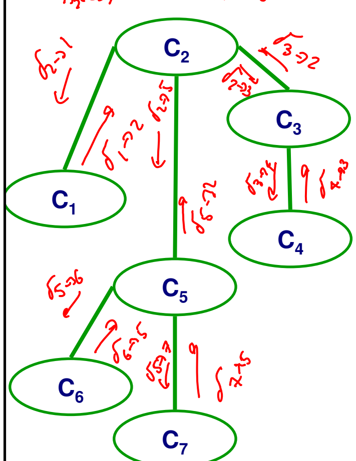
$\delta_{4\to5}, \delta_{5\to3}, \delta_{32}, \delta_{2\to1}$

can obtain all beliefs

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

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

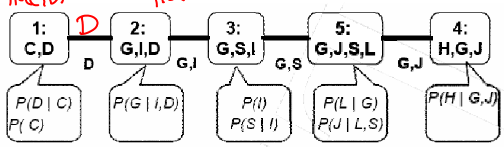
$\pi_2(C_2) = \pi_0(C_2) \cdot \delta_{1\to2}\, \delta_{5\to2} \cdot \delta_{3\to2} = P(C_i, e)$ ← if I had evidence e



- 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

$\pi_0(C,D) = P(C) \cdot P(D|C) = P(C,D)$ $\pi_0(G,I,D) = P(G|I,D)$



initially

$\sum_C \pi_0(C,D) = P(D)$

$\sum_{GI} \pi_0(G,I,D) = \sum_i \sum_g P(g|i,D) = \sum_i 1 = |Val[I]| = 2$

- Initially, neighboring nodes don't agree on "distribution" over separators
- **Calibrated clique tree**: $\sum_C \pi_1(C,D) = \sum_{GI} \pi_2(G,I,D)$
  - At convergence, tree is *calibrated*
  - Neighboring nodes agree on distribution over separator

Answering queries with clique trees

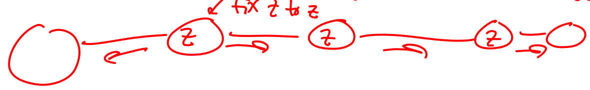
calibrated w.r.t. e

- Query within clique $P(X_i|e)$ look for any clique C_j, $X_i \in C_j$

  if $X_i \subseteq X$ and $X_i \subseteq C_j$ $\Rightarrow$ same thing $P(X_i|e) \propto \sum_{C_j - X_i} \pi_j(C_j)$

- Incremental updates – Observing evidence Z=z
  - Multiply some clique by indicator **1**(Z=z)

    then send messages ↙fix Z to z send outward message

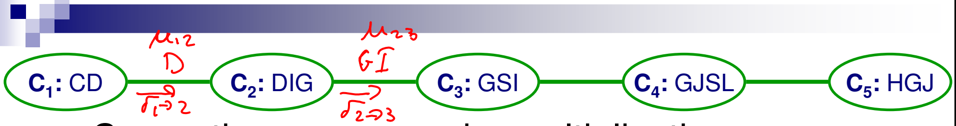


- Query outside clique
  - Use variable elimination!

    It's in your HW!!

$P(A,H|e)$?

Message passing with division



- Computing messages by multiplication:

$$\delta_{2\to3}(GI) = \sum_d \pi_0(d,G,I) \cdot \delta_{1\to2}(d)$$

$$\pi_2(DIG) = \pi_0(DGI) \cdot \delta_{1\to2}(D) \cdot \delta_{3\to2}(GI)$$

- Computing messages by division:

$\delta^{(0)}_{1\to2} = 1$

$\pi^{(0)} = \pi_0$

initialize $\pi_0(D,G,I)$, $\pi^{(t)}(DIG)$

new message $\delta^{(t)}_{1\to2}(D)$

$$\pi^{(t)}(DIG) = \pi^{(t-1)}(DIG) \cdot \frac{\delta^{(t)}_{1\to2}(D)}{\delta^{(t-1)}_{1\to2}(D)}$$

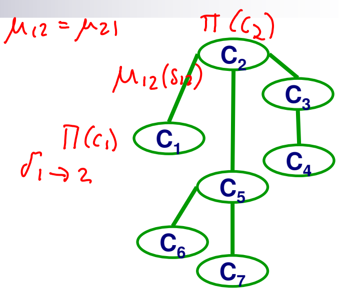
when $\delta^{(t+1)}_{3\to2}(GI)$

$$\pi^{(t+1)}(DIG) = \pi^{(t)}(DIG) \cdot \frac{\delta^{(t+1)}_{3\to2}(GI)}{\delta^{(t)}_{3\to2}(GI)}$$

Lauritzen-Spiegelhalter Algorithm (a.k.a. belief propagation)

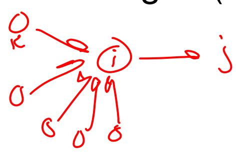
Simplified description
see reading for details

- Initialize all separator potentials to 1 $\mu_{12} = \mu_{21}$
  - $\mu_{ij} \leftarrow 1$
- All messages ready to transmit
- While ∃ $\delta_{i\to j}$ ready to transmit
  - $\mu_{ij}' \leftarrow \sum_{C_i - S_{ij}} \pi_i(C_i)$
  - If $\mu_{ij}' \neq \mu_{ij}$
    - $\delta_{i\to j} \leftarrow \mu_{ij}' / \mu_{ij}$
    - $\pi_j \leftarrow \pi_j \times \delta_{i\to j}$
    - $\mu_{ij} \leftarrow \mu_{ij}'$
    - ∀ neighbors k of j, k≠ i, $\delta_{j\to k}$ ready to transmit
- Complexity: Linear in # cliques
  - for the "right" schedule over edges (leaves to root, then root to leaves)
- **Corollary**: At convergence, every clique has correct belief



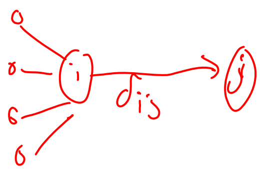
VE versus BP in clique trees

- VE messages (the one that multiplies)



$$\delta_{ij} = \sum_{C_i \setminus S_{ij}} \pi_0(C_i) \prod_{k \in \text{neighbours of } i \text{ except } j} \delta_{k \to i}$$

- BP messages (the one that divides)



$$\mu'_{ij} = \sum_{C_i / S_{ij}} \pi_i(C_i)$$

$$\delta_{i \to j} = \frac{\mu'_{ij}}{\mu_{ij}}$$

Clique tree invariant

- **Clique tree potential**:
  - Product of clique potentials divided by separators potentials
- Clique tree **invariant**:
  - P(**X**) = π_T(**X**)

Belief propagation and clique tree invariant

- **Theorem**: Invariant is maintained by BP algorithm!

- BP reparameterizes clique potentials and separator potentials
  - At convergence, potentials and messages are marginal distributions

Subtree correctness

- **Informed message** from i to j, if all messages into i (other than from j) are informed
  - Recursive definition (leaves always send informed messages)
- **Informed subtree**:
  - All incoming messages informed
- **Theorem**:
  - Potential of connected informed subtree *T'* is marginal over scope[*T'*]
- **Corollary**:
  - At convergence, clique tree is *calibrated*
    - π_i = P(scope[π_i])
    - μ_{ij} = P(scope[μ_{ij}])

Clique trees versus VE

- Clique tree advantages
  - Multi-query settings
  - Incremental updates
  - Pre-computation makes complexity explicit

- Clique tree disadvantages
  - Space requirements – no factors are "deleted"
  - Slower for single query
  - Local structure in factors may be lost when they are multiplied together into initial clique potential

Clique tree summary

- Solve marginal queries for all variables in only twice the cost of query for one variable
- Cliques correspond to maximal cliques in induced graph
- Two message passing approaches
  - VE (the one that multiplies messages)
  - BP (the one that divides by old message)
- Clique tree invariant
  - Clique tree potential is always the same
  - We are only reparameterizing clique potentials
- Constructing clique tree for a BN
  - from elimination order
  - from triangulated (chordal) graph
- Running time (only) exponential in size of largest clique
  - Solve **exactly** problems with thousands (or millions, or more) of variables, and cliques with tens of nodes (or less)