Readings:
K&F: 8.4, 12.1, 12.2, 9

Complexity of Var. Elim
MPE Inference
Junction Trees

Graphical Models – 10708

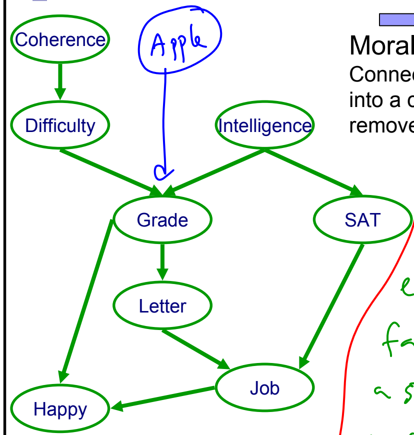
Carlos Guestrin
Carnegie Mellon University

October 20th, 2008

What's next

- Thus far: Variable elimination
  - (Often) Efficient algorithm for inference in graphical models
- Next: Understanding complexity of variable elimination
  - Will lead to cool junction tree algorithm later

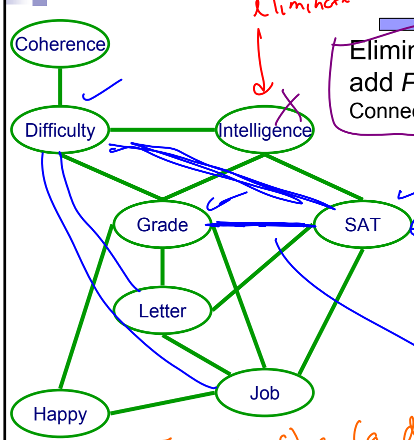
Complexity of variable elimination – Graphs with loops



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

Connect nodes that appear together in an initial factor

Eliminating a node – Fill edges

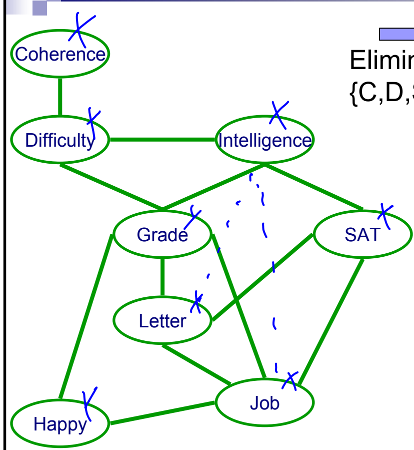


Eliminate variable
add *Fill Edges*:
Connect neighbors

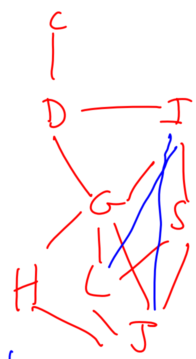
Induced graph

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

$\triangleleft \equiv \prec$



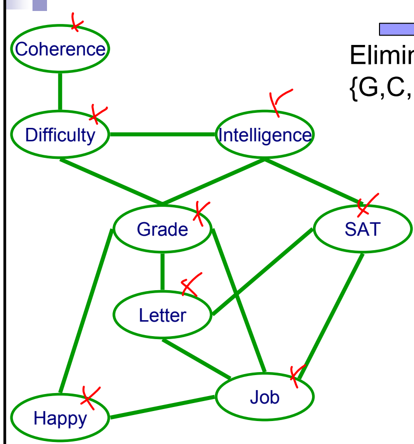
Elimination order:
{C,D,S,I,L,H,J,G}



each VE factor subset of maximal clique

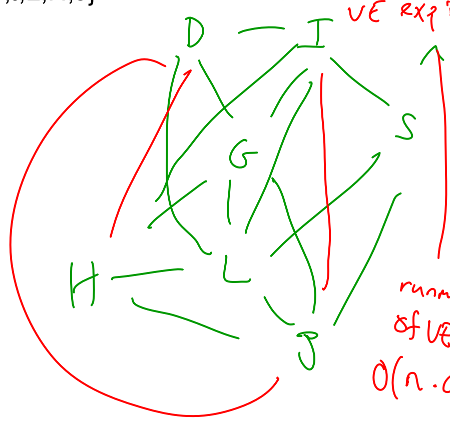
largest clique: {G,I,L,J} => VE exp. in 4

Different elimination order can lead to different induced graph



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

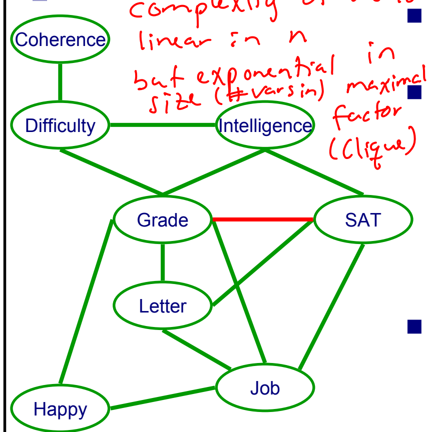
biggest clique: {D,I,G,L,S,H} VE exp in 6



running of VE $O(n \cdot d^6)$

Induced graph and complexity of VE

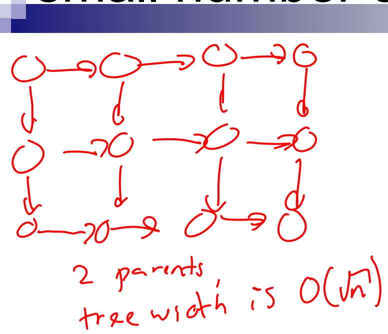
Read complexity from cliques in induced graph

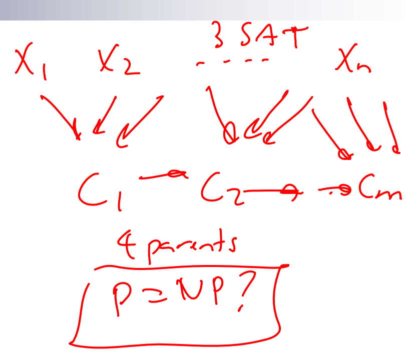


Elimination order:
{C,D,I,S,L,H,J,G}

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

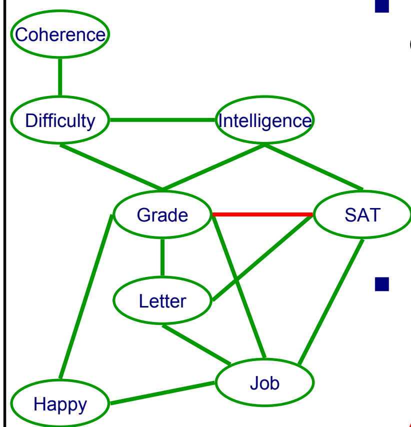
Example: Large induced-width with small number of parents





Compact representation ⇏ Easy inference ☹

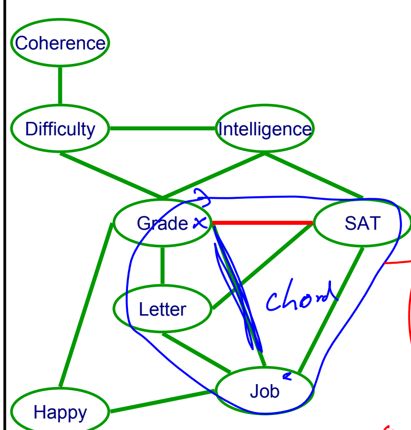
Finding optimal elimination order



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 ≤ K
- **Interpretation**:
  - Hardness of finding elimination order in addition to hardness of inference
  - Actually, can find elimination order in time exponential in size of largest clique – same complexity as inference

Induced graphs and chordal graphs

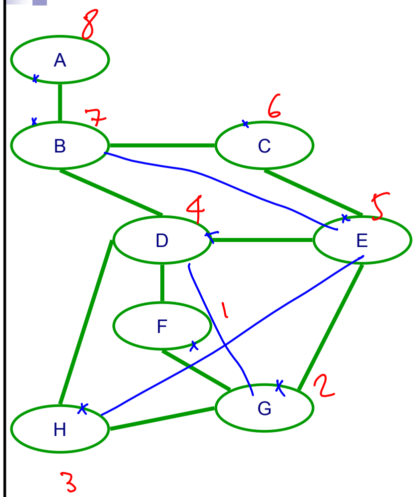


- **Chordal graph**:
  - Every cycle $X_1 - X_2 - \ldots - X_k - X_1$ with $k \geq 3$ has a chord
    - Edge $X_i - X_j$ for non-consecutive i & j
- **Theorem**:
  - Every induced graph is chordal
- "Optimal" elimination order easily obtained for chordal graph

add no new edges

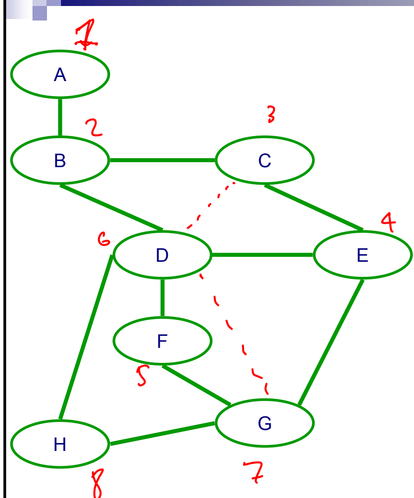
a tree is already chordal

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 ← unmarked var with most **marked** neighbors
  - ◁(X) ← k
  - Mark X
- **Theorem**: Obtains optimal order for chordal graphs
- Often, not so good in other graphs!

Minimum fill/size/weight heuristics

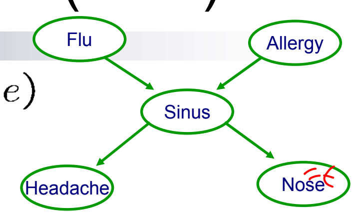


- Many more effective heuristics
  - see reading
- **Min (weighted) fill heuristic**
  - Often very effective
- Initialize unobserved nodes **X** as unmarked
- For k = 1 to |**X**|
  - X ← unmarked var whose elimination adds fewest edges
  - ◁(X) ← k
  - Mark X
  - Add fill edges introduced by eliminating 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: $\underset{x_1,\ldots,x_n}{\text{argmax}}\, P(x_1,\ldots,x_n \mid e)$

- Using defn of conditional probs:

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

- Normalization irrelevant:

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

Max-marginalization

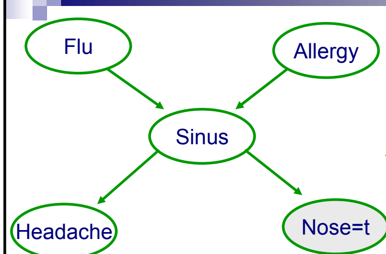
$a(b+c) = ab + ac$

$\max(ab; ac) = a \cdot \max(b; c)$

$$\max_a \max_s P(A=a)\, P(S=s \mid A=a)\, P(N=t \mid S=s)$$

$$= \max_{as} \begin{bmatrix} P(A=t)\, P(S=t \mid A=t)\, P(N=t \mid S=t), \\ P(A=f)\, P(S=t \mid A=f)\, P(N=t \mid S=t), \\ P(A=t)\, P(S=f \mid A=t)\, P(N=t \mid S=f), \\ P(A=f)\, P(S=f \mid A=f)\, P(N=t \mid S=f) \end{bmatrix}$$

Example of variable elimination for MLE – Forward pass



$$\max_{f,a,s,h} P(f,a,s,h,N=t) =$$

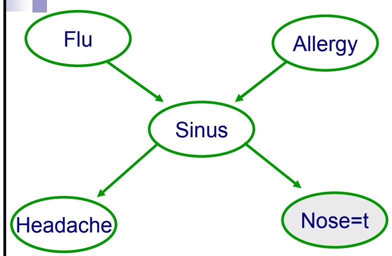
$$\max_{fas} P(f)\, P(a)\, P(s \mid fa)\, P(N=t \mid s) \underbrace{\max_h P(h \mid s)}_{g_1(s)}$$

$$= \max_{fa} P(f)\, P(a) \underbrace{\max_s P(s \mid f,a)\, P(N=t \mid s)\, g_1(s)}_{g_2(f,a)}$$

$$= \underbrace{\max_f P(f) \underbrace{\max_a P(a) \cdot g_2(f,a)}_{g_3(f)}}_{g_4} = g_4 = \max_{fash} P(f,a,s,h,N=t)$$

$\max_f P(f)\, g_3(f)$

Example of variable elimination for MLE – Backward pass



$f^* = \arg\max_f P(f)\, g_3(f)$

$a^* = \arg\max_a P(a)\, g_2(f^*, a)$

$s^* = \arg\max_s P(s \mid f^*, a^*) \cdot g_1(s)\, P(n{=}t \mid s)$

$h^* = \arg\max_h P(h \mid s^*)$

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

    only difference

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

  - 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:

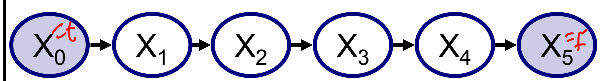
$$x_i^* \in \underset{x_i}{\text{argmax}} \prod_{j=1}^{k} f_j$$

can only depend on X_i or vars that were eliminated later

What you need to know about VE

- 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" (special recitation) context specific indep.
- 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?

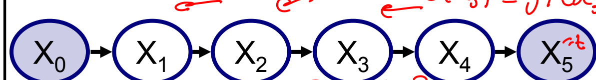
$O(n)$

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

$O(n^2)$

Reusing computation

get forwards-backwards: all $P(x_i|e)$ in $O(n)$



Compute:

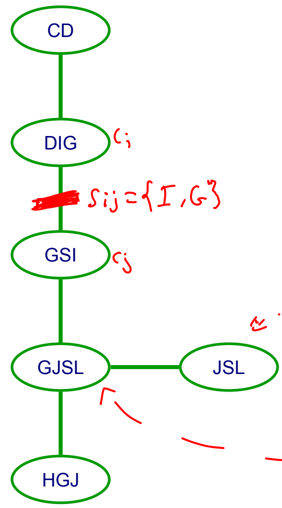
$$P(X_i \mid x_0, x_{n+1})$$

$$P(X_2|x_0,x_5) = \sum_{x_1,x_3,x_4} P(x_1|x_0)\,P(X_2|x_1)\,P(x_3|X_2)\,P(x_4|x_3)\,P(x_5=t|x_4)$$

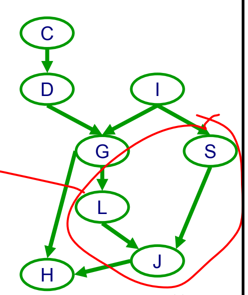
$$= \sum_{x_1,x_3} P(x_1|x_0)\,P(X_2|x_1)\,P(x_3|x_2) \underbrace{\sum_{x_4} P(x_4|x_3)\,P(x_5=t|x_4)}_{\beta(x_3),\ g_1(x_3)}$$

$$P(X_3|x_0,x_5) = \sum_{x_1,x_2} P(x_1|x_0)\,P(x_2|x_1)\,P(X_3|x_2) \underbrace{\sum_{x_4} P(x_4|X_3)\,P(x_5=t|x_4)}_{g_1(X_3)}$$

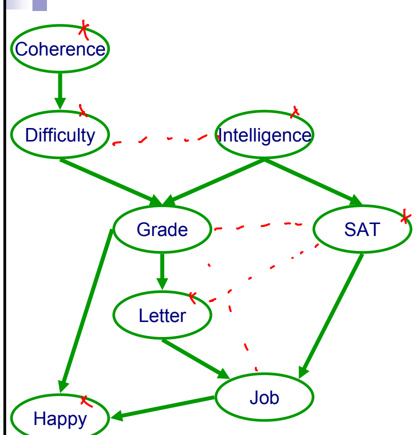
Cluster graph

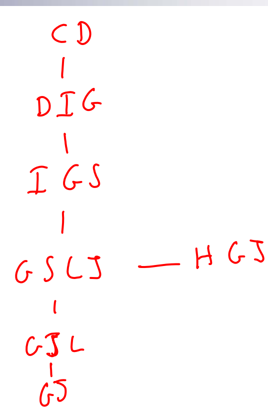


- **Cluster graph**: For set of factors *F*
  - Undirected graph
  - Each node i associated with a cluster $\mathbf{C}_i$
  - *Family preserving*: for each factor $f_j \in F$, $\exists$ node i such that scope[f_j] $\subseteq \mathbf{C}_i$
  - Each edge i – j is associated with a separator $\mathbf{S}_{ij}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



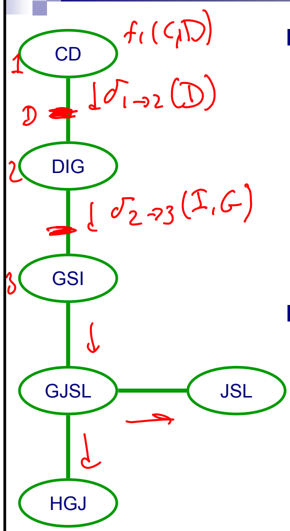
Factors generated by VE





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}] $\subseteq \mathbf{S}_{ij}$