Probabilistic Graphical Models

Raquel Urtasun and Tamir Hazan

TTI Chicago

April 20, 2011

How to introduce evidence?

- We can apply variable elimination algorithm to the task of computing $P(\mathbf{Y}, \mathbf{e})$.
- Simply apply the algorithm to the set of factors in the network reduced to $\mathbf{E} = \mathbf{e}$, and eliminate the variables in $\mathcal{X} - \mathbf{Y} - \mathbf{e}$.
- The returned factor $\phi^*(\mathbf{Y})$ is $P(\mathbf{Y}, \mathbf{e})$.
- To obtain the conditional $P(\mathbf{Y}|\mathbf{e})$ we have to normalize the resulting product of factors.
- The normalization constant is $P(\mathbf{e})$.

Sum-product VE for conditional distributions

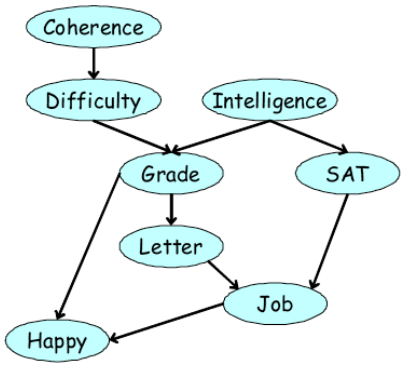
Algorithm 9.2 Using Sum-Product-Variable-Elimination **for computing conditional probabilities.**

Procedure Cond-Prob-VE (
  $\mathcal{K}$, // A network over $\mathcal{X}$
  $\boldsymbol{Y}$, // Set of query variables
  $\boldsymbol{E} = \boldsymbol{e}$ // Evidence
)

1 $\Phi \leftarrow$ Factors parameterizing $\mathcal{K}$
2 Replace each $\phi \in \Phi$ by $\phi[\boldsymbol{E} = \boldsymbol{e}]$
3 Select an elimination ordering $\prec$
4 $\boldsymbol{Z} \leftarrow = \mathcal{X} - \boldsymbol{Y} - \boldsymbol{E}$
5 $\phi^* \leftarrow$ Sum-Product-Variable-Elimination$(\Phi, \prec, \boldsymbol{Z})$
6 $\alpha \leftarrow \sum_{\boldsymbol{y} \in Val(\boldsymbol{Y})} \phi^*(\boldsymbol{y})$
7 **return** α, ϕ^*

Example

- We want to compute $P(J)$.



Step	Variable eliminated	Factors used	Variables involved	New factor
1	C	$\phi_C(C)$, $\phi_D(D,C)$	C,D	$\tau_1(D)$
2	D	$\phi_G(G,I,D)$, $\tau_1(D)$	G,I,D	$\tau_2(G,I)$
3	I	$\phi_I(I)$, $\phi_S(S,I)$, $\tau_2(G,I)$	G,S,I	$\tau_3(G,S)$
4	H	$\phi_H(H,G,J)$	H,G,J	$\tau_4(G,J)$
5	G	$\tau_4(G,J)$, $\tau_3(G,S)$, $\phi_L(L,G)$	G,J,L,S	$\tau_5(J,L,S)$
6	S	$\tau_5(J,L,S)$, $\phi_J(J,L,S)$	J,L,S	$\tau_6(J,L)$
7	L	$\tau_6(J,L)$	J,L	$\tau_7(J)$

- To compute $P(J, i^1, h^0)$

Step	Variable eliminated	Factors used	Variables involved	New factor
1'	C	$\phi_C(C)$, $\phi_D(D,C)$	C,D	$\tau'_1(D)$
2'	D	$\phi_G[I=i^1](G,D)$, $\phi_I[I=i^1]()$, $\tau'_1(D)$	G,D	$\tau'_2(G)$
5'	G	$\tau'_2(G)$, $\phi_L(L,G)$, $\phi_H[H=h^0](G,J)$	G,L,J	$\tau'_5(L,J)$
6'	S	$\phi_S[I=i^1](S)$, $\phi_J(J,L,S)$	J,L,S	$\tau'_6(J,L)$
7'	L	$\tau'_6(J,L)$, $\tau'_5(J,L)$	J,L	$\tau'_7(J)$

Complexity of Variable Elimination

- VE can be computationally much more efficient than a full enumeration.
- n random variables and m initial factors. In a BN $m = n$, in a MN we may have more factors than variables.
- Assume we run the algorithm until all variables are eliminated.
- At each step we pick a variable X_i and multiply all factors involving the variable, resulting in a single factor ψ_i. The variable gets sum out of ψ_i, resulting in a new factor τ_i with $scope(\tau) = scope(\phi) - X_i$.
- Let N_i be the number of entries in the factor ψ_i, and let $N_{max} = \max_i N_i$.
- The total number of multiplications is at most $(n + m)N_i \leq (n + m)N_{max}$, thus $\mathcal{O}(mN_{max})$.
- The total number of additions is nN_{max}.
- Therefore the total cost is $O(mN_{max})$.
- The exponential blowup is the potential exponential size of the factors ψ. if each variable has at most v values then $N_i \leq v^{k_i}$, for scope ψ_i which contains k_i variables.

Factors and complexity I

- The only aspect that affects complexity is the graph structure.
- Let's try to analyze the complexity in terms of the graph structure.
- The algorithm does not care if the graph is directed or undirected, only depends on the scope of the factors.
- Let's consider an undirected graph for simplicity.
- Let Φ be a set of factors, we define $Scope(\Phi) = \cup_{\phi \in \Phi} Scope(\phi)$.
- We define $\mathcal{H}_\Phi$ the undirected graph where we have an edge iff there exists a factor $\phi \in \Phi$ such that $X_i, X_j \in Scope(\phi)$.

Factors and complexity II

- Let P be a distribution defined as

$$P(\mathbf{X}) = \frac{1}{Z} \prod_{\phi \in \Phi} \phi$$

with $\mathbf{X} = Scope(\Phi)$ and Z the partition function. Then $\mathcal{H}_\Phi$ is the minimal Markov network I-map for P, and the factors Φ are a parameterization of this network that defines the distribution P.

- For a BN, the undirected graph $\mathcal{H}_\Phi$ is the **moralized graph**.
- In this case the distribution is normalized, i.e., $Z = 1$.
- The Markov network induced by the set of factors $\Phi[\mathbf{e}]$ defined by the reduction of the factors in a BN to some context $\mathbf{E} = \mathbf{e}$, is such that the variables $\mathbf{E}$ are removed from the factors, $Scope[\Phi_e] = \mathcal{X} - \mathbf{E}$.

Elimination as Graph Transformation

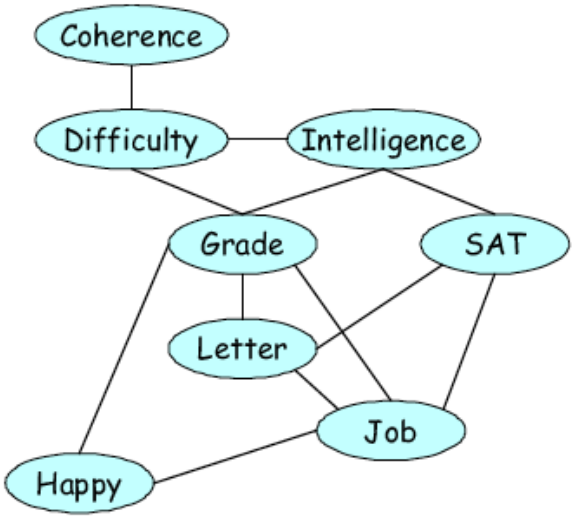
When a variable X is eliminated

- We create a single factor ψ that contains X and all of the variables $\mathbf{Y}$ with which it appears in factors.
- We eliminate X from ψ, replacing it with a new factor τ that contains all of the variables $\mathbf{Y}$, but not X. Let's call this Φ_X.

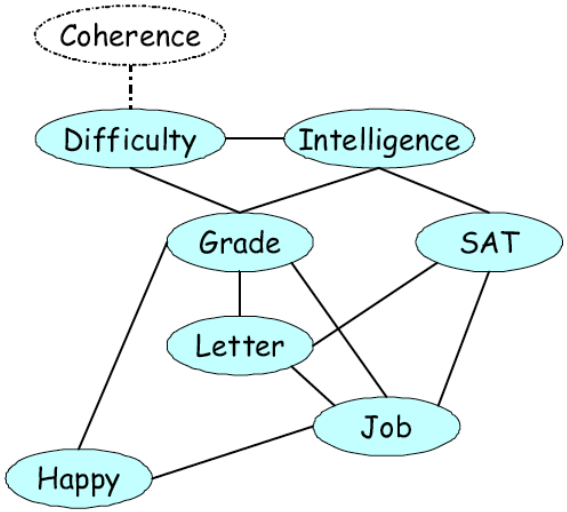
How does this modify the graph from $\mathcal{H}_\Phi$ to $\mathcal{H}_{\Phi_X}$?

- Constructing ψ generates edges between all of the variables $Y \in \mathbf{Y}$.
- Some of these edges were in $\mathcal{H}_\Phi$, some are new.
- The new edges are called **fill edges**.
- The step of removing X from ϕ to construct τ removes X and all it's incident edges from the graph.

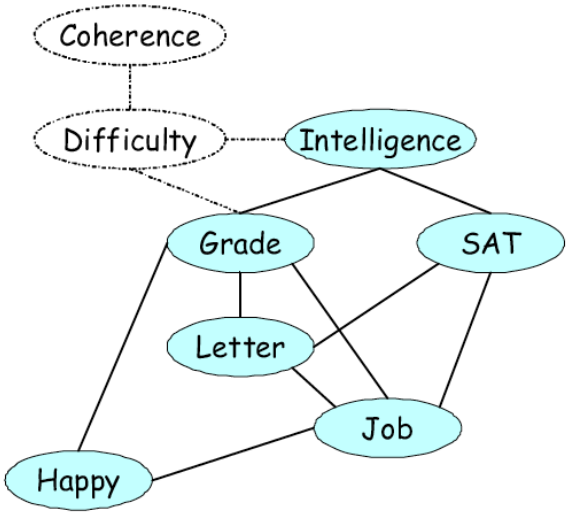
Example



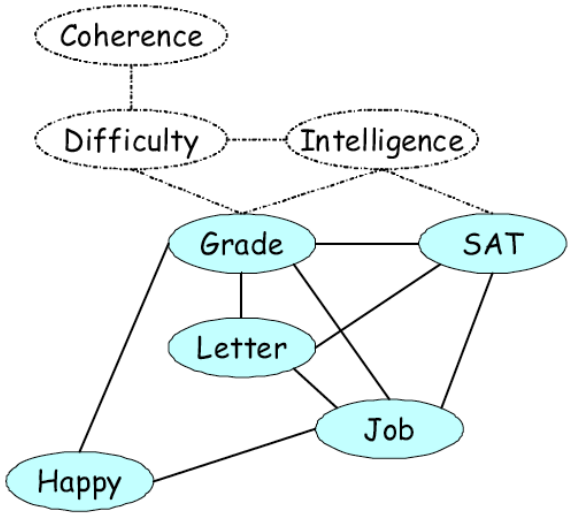
(Graph)



(Elim. C)



(Elim. D)

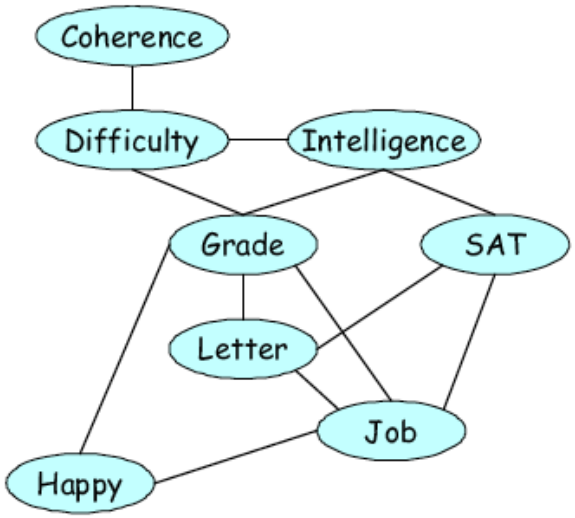


(Elim. I)

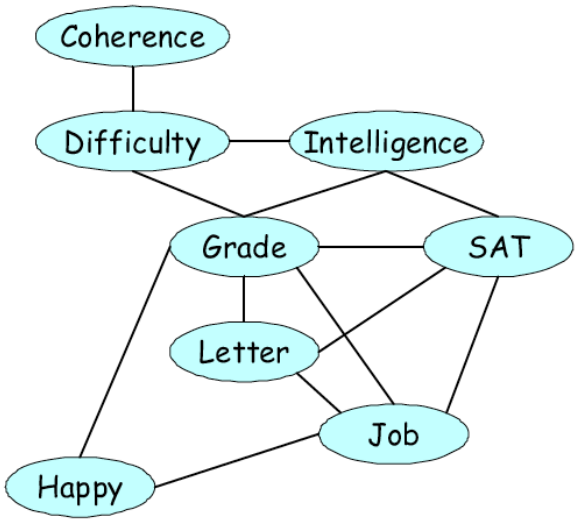
Induced Graph

- We can summarize the computation cost using a single graph that is the union of all the graphs resulting from each step of the elimination.
- **Def**: Let Φ be a set of factors over $\mathcal{X} = \{X_1, \cdots, X_n\}$ and $\prec$ be an elimination ordering for some subset $\mathbf{X} \subseteq \mathcal{X}$. The induced graph $\mathcal{I}_{\Phi,\prec}$ is an undirected graph over $\mathcal{X}$, where X_i and X_j are connected by and edge if both appear in some intermediate factor ψ generated by the VE algorithm using $\prec$ as an elimination ordering.
- In BN, the factors Φ correspond to the CPDs .
- In Markov networks they correspond to the clique potentials in $\mathcal{H}$.
- Each factor ψ corresponds to a complete subgraph in $\mathcal{I}_{\Phi,\prec}$, so it's a clique.

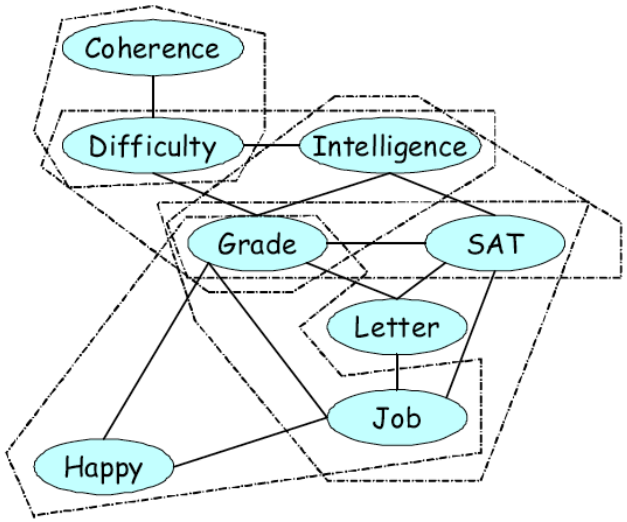
Example



Step	Variable eliminated	Factors used	Variables involved	New factor
1	C	$\phi_C(C)$, $\phi_D(D,C)$	C,D	$\tau_1(D)$
2	D	$\phi_G(G,I,D)$, $\tau_1(D)$	G,I,D	$\tau_2(G,I)$
3	I	$\phi_I(I)$, $\phi_S(S,I)$, $\tau_2(G,I)$	G,S,I	$\tau_3(G,S)$
4	H	$\phi_H(H,G,J)$	H,G,J	$\tau_4(G,J)$
5	G	$\tau_4(G,J)$, $\tau_3(G,S)$, $\phi_L(L,G)$	G,J,L,S	$\tau_5(J,L,S)$
6	S	$\tau_5(J,L,S)$, $\phi_J(J,L,S)$	J,L,S	$\tau_6(J,L)$
7	L	$\tau_6(J,L)$	J,L	$\tau_7(J)$



(Induced graph)



(Maximal Cliques)

More on Induced Graphs

Theorem: Let $\mathcal{I}_{\Phi,\prec}$ be the induced graph for a set of factors Φ and ordering $\prec$, then

1. Every factor generated during the VE has a scope that is a clique in $\mathcal{I}_{\Phi,\prec}$.
2. Every maximal clique in $\mathcal{I}_{\Phi,\prec}$ is the scope of some intermediate factor in the computation.

Proof of first statement:

- Consider a factor $\psi(Y_1, \cdots, Y_k)$ generated during VE.
- By definition of induced graph, there must be an edge between each Y_i and Y_j.
- Hence $Y_1, \cdots, Y_k$ is a clique.

More on Induced Graphs

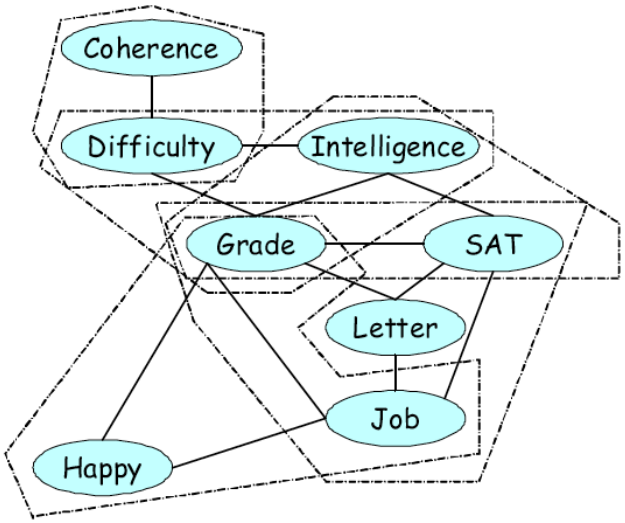
Th: Let $\mathcal{I}_{\Phi,\prec}$ be the induced graph for a set of factors Φ and ordering $\prec$, then

1. Every factor generated during the VE has a scope that is a clique in $\mathcal{I}_{\Phi,\prec}$.
2. Every maximal clique in $\mathcal{I}_{\Phi,\prec}$ is the scope of some intermediate factor in the computation.

Proof of second statement:

- Let $\mathbf{Y} = \{Y_1, \cdots, Y_k\}$ be a maximal clique, where Y_1 is the first of the variables to eliminate.
- As $\mathbf{Y}$ is a clique, there is an edge from Y_1 to all the other Y_i.
- Once Y_1 is eliminated it cannot appear in other factors, so no new edges can be added. Edges connecting Y_1 were added prior to this point in time.
- As there is an edge between Y_1 and Y_i, there is a factor containing both.
- When eliminating Y_1 all these factors are multiplied, therefore ψ contains $Y_1, \cdots, Y_k$.
- This factor cannot contain any other variables, otherwise there would also be an edge to all $Y_1, \cdots, Y_k$, and $Y_1, \cdots, Y_k$ would not be maximal connected subgraph.

Example



(Maximal Cliques)

Step	Variable eliminated	Factors used	Variables involved	New factor
1	C	$\phi_C(C)$, $\phi_D(D,C)$	C,D	$\tau_1(D)$
2	D	$\phi_G(G,I,D)$, $\tau_1(D)$	G,I,D	$\tau_2(G,I)$
3	I	$\phi_I(I)$, $\phi_S(S,I)$, $\tau_2(G,I)$	G,S,I	$\tau_3(G,S)$
4	H	$\phi_H(H,G,J)$	H,G,J	$\tau_4(G,J)$
5	G	$\tau_4(G,J)$, $\tau_3(G,S)$, $\phi_L(L,G)$	G,J,L,S	$\tau_5(J,L,S)$
6	S	$\tau_5(J,L,S)$, $\phi_J(J,L,S)$	J,L,S	$\tau_6(J,L)$
7	L	$\tau_6(J,L)$	J,L	$\tau_7(J)$

(VE)

- The maximal cliques in $\mathcal{I}_{\mathcal{G},\prec}$ are

$$\begin{aligned} \mathbf{C}_1 &= \{C,D\} \\ \mathbf{C}_2 &= \{D,I,G\} \\ \mathbf{C}_3 &= \{G,L,S,J\} \\ \mathbf{C}_4 &= \{G,J,H\} \end{aligned}$$

Concept of Width

- We define the **width** of an induced graph to be the number of nodes in the largest clique in the graph minus 1.
- We define the **induced width** $w_{\mathcal{K},\prec}$ of an ordering $\prec$ relative to a directed or undirected graph $\mathcal{K}$ to be the width of the graph $\mathcal{I}_{\mathcal{K},\prec}$ induced by applying VE to $\mathcal{K}$ using ordering $\prec$.
- We define the **minimal induced width** of a graph $\mathcal{K}$ to be

$$w^*_{\mathcal{K}} = \min_{\prec} w(\mathcal{I}_{\mathcal{K},\prec})$$

- The minimal induced width of the graph $\mathcal{K}$ provides a bound on the best performance we can hope by applying VE to a probability that factorizes over $\mathcal{K}$.

Finding elimination orderings

- How can we compute the minimal induced width of the graph?
- and the elimination ordering achieving that width?
- Given a graph and a bound K, determine whether there exist an elimination ordering achieving an induced width $\leq K$ is NP-complete.
- Finding the optimal elimination ordering is also NP-hard.
- Note that this is different from the NP-hardness of inference: even if we knew the best elimination ordering, the width might be too large and inference might be exponential.
- How to find good elimination orderings?

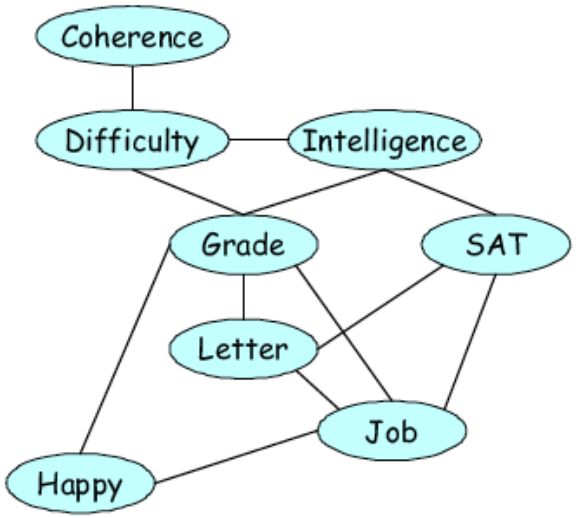
Chordal Graphs

An undirected graph is **chordal** if it contains no cycle of length greater than three that has no "shortcut", i.e., every minimal loop is length three.

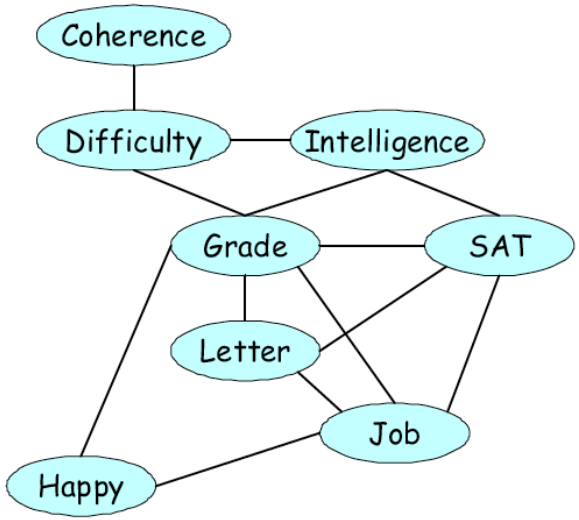
Theorem: Every induced graph is chordal

- Proof by contradiction: Assume we have such a cycle $X_1 - X_2 - \cdots - X_k - X_1$ for $k > 3$.
- Without lost of generality assume X_1 is the first variable to eliminate.
- Before we showed that no edge incident on X_1 is added after X_1 is eliminated, hence $X_1 - X_2$ and $X_1 - X_k$ must exist at this point.
- Therefore the edge $X_2 - X_k$ will be added at the same time, contradicting our assumption.

Example



(graph)



(induced graph)

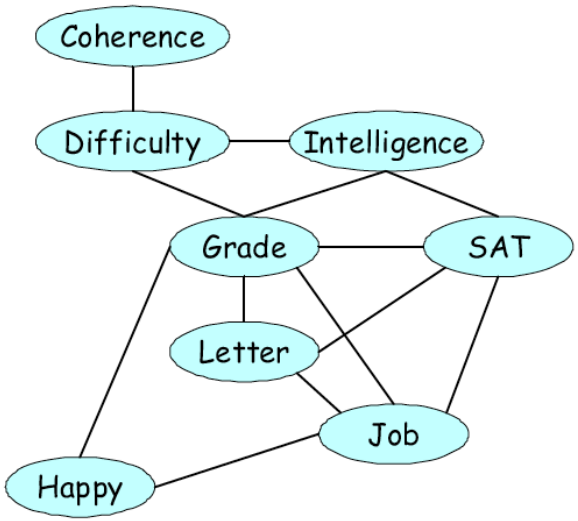
- The loop $H \rightarrow G \rightarrow L \rightarrow J \rightarrow H$ is cut by the chord $G \rightarrow J$

More on chordal graphs

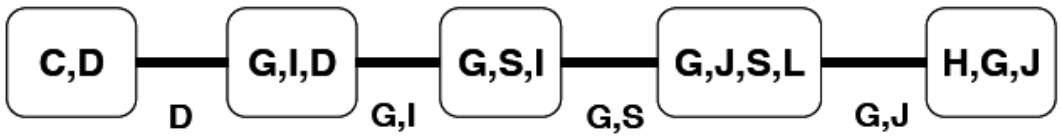
Theorem: Any chordal graph $\mathcal{H}$ admits an elimination ordering that does not introduce any fill edges into the graph.

- Proof by induction on the number of nodes in the tree.
- Let $\mathcal{H}$ a chordal graph with n nodes.
- There is a clique tree $\mathcal{T}$ for $\mathcal{H}$.
- Let $\mathbf{C}_k$ be a clique in the tree which is a leaf, i.e., it has only a single other clique as neighbor.
- Let X_i be a variable which is in $\mathbf{C}_k$ but not in its neighbor.
- Let $\mathcal{H}'$ be the graph by eliminating X_i.
- The neighbors of X_i are $\mathbf{C}_k - \{X_i\}$.
- As all the neighbors are in $\mathbf{C}_k$, they are connected to each other.
- Therefore eliminating X_i introduces no fill edges.
- As $\mathcal{H}'$ is also chordal, we can apply the inductive hypothesis.

Example



(graph)



(clique tree)

- Each sepset separates the two sides of the tree, e.g., $\{G, S\}$ separates $\{C, I, D\}$ and $\{L, J, H\}$.
- The elimination ordering C, D, I, H, G, S, L, J is an ordering that might arise from the construction of the previous theorem.
- The other ordering we saw in the previous day does not respect this construction and introduces edges

Maximum Cardinality I

- An alternative for constructing an ordering that introduces no edges in a chordal graph is the **Max-cardinality** algorithm.
- It does not use clique trees but operates in the graph.

Algorithm 9.3 Maximum Cardinality Algorithm for constructing an elimination ordering

```
    Procedure Max-Cardinality (
      H    // An undirected graph over X
    )
1     Initialize all nodes in X as unmarked
2     for k = |X| ... 1
3       X ← unmarked variable in X with largest number of marked neighbors
4       π(X) ← k
5       Mark X
6     return π
```

- The elimination ordering is REVERSE.
- Example on the board.

Maximum Cardinality II

- As max cardinality gives an ordering with is consistent with previous theorem, when applied to chordal graphs it introduces no fill edges.
- **Theorem**: Let $\mathcal{H}$ be a chordal graph. Let π be the ranking obtained by running Max-Cardinality on $\mathcal{H}$. Then Sum-product-VE with order π does not introduce any fill edges.
- Max-cardinality algorithm can also be used to construct elimination ordering for a non-chordal graph.
- However, in this case the algorithm is not as good as others that we will see later in the course.

Summary

- If $\mathcal{H}$ is a chordal graphs, then apply max-cardinality.
- If $\mathcal{H}$ is non-chordal then
  1. Triangulate
  2. Max-cardinality
- Finding the minimal triangulation is NP-Hard.
- There are other algorithms that we now going to discuss.

Minimum Fill/Size/Weight Search

- Goal: Find an ordering that induces a small graph.
- We cannot solve this exactly.
- Greedy algorithm eliminating algorithms one at a time.
- Question is which one at each step?

Algorithm 9.4 Greedy search for constructing an elimination ordering

Procedure Greedy-Ordering (
  $\mathcal{H}$ // An undirected graph over $\mathcal{X}$,
  s // An evaluation metric
)
1 Initialize all nodes in $\mathcal{X}$ as unmarked
2 **for** $k = 1 \ldots |\mathcal{X}|$
3 Select an unmarked variable $X \in \mathcal{X}$ that minimizes $s(\mathcal{H}, X)$
4 $\pi(X) \leftarrow k$
5 Introduce edges in $\mathcal{H}$ between all neighbors of X
6 Mark X **return** π

Evaluation metrics

Set of possible heuristics:

- **Min-neighbors**: The cost of a vertex is the number of neighbors it has in the current graph.
- **Min-weight**: the cost of a vertex is the product of weights (domain cardinality) of its neighbors.
- **Min-fill**: the cost of a vertex is the number of edges that need to be added to the graph due to its elimination.
- **Weighted-Min-Fill**: the cost of a vertex is the sum of weights of the edges that need to be added to the graph due to its elimination. Weight of an edge is the product of weights of its constituent vertices.

Which one better?

- None of these criteria is better than others.
- The search can be done deterministically or stochastic (at each step).
- If stochastic, do multiple trials with multiple criteria, as the cost of this is much less than the elimination itself.