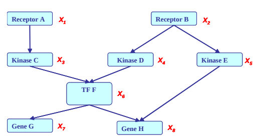
Machine Learning

10-701/15-781, Spring 2010

Bayesian Networks
Learning and Inference



Eric Xing

Lecture 14, March 15, 2010

Reading: Chap. 8, C.B book

Recap of BN Representation

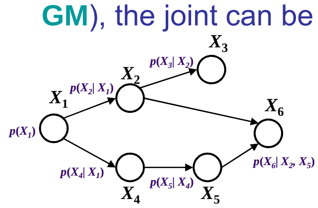
- Joint probability dist. on multiple variables:

$$
\begin{aligned}
&P(X_1, X_2, X_3, X_4, X_5, X_6) \\
&= P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_1, X_2)P(X_4 \mid X_1, X_2, X_3)P(X_5 \mid X_1, X_2, X_3, X_4)P(X_6 \mid X_1, X_2, X_3, X_4, X_5)
\end{aligned}
$$

- If X_i's are **independent**: ($P(X_i|\cdot)= P(X_i)$)

$$
\begin{aligned}
&P(X_1, X_2, X_3, X_4, X_5, X_6) \\
&= P(X_1)P(X_2)P(X_3)P(X_4)P(X_5)P(X_6) = \prod_i P(X_i)
\end{aligned}
$$

- If X_i's are **conditionally independent** (as described by a **GM**), the joint can be factored to simpler products, e.g.,



$$
\begin{aligned}
&P(X_1, X_2, X_3, X_4, X_5, X_6) \\
&= P(X_1)\, P(X_2| X_1)\, P(X_3| X_2)\, P(X_4| X_1)\, P(X_5| X_4)\, P(X_6| X_2, X_5)
\end{aligned}
$$

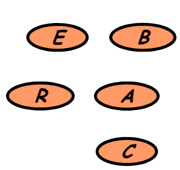
Inference and Learning

- We now have compact representations of probability distributions: **BN**
- A BN *M* describes a unique probability distribution *P*
- Typical tasks:
  - Task 1: How do we answer **queries** about *P*?
    - We use **inference** as a name for the process of computing answers to such queries
  - Task 2: How do we estimate a **plausible model** *M* from data *D*?
    1. We use **learning** as a name for the process of obtaining point estimate of *M*.
    2. But for *Bayesian*, they seek $p(M \mid D)$, which is actually an **inference** problem.
    3. When not all variables are observable, even computing point estimate of *M* need to do **inference** to impute the *missing data*.

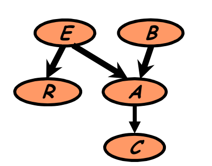
Learning BNs

The goal:

Given set of independent samples (*assignments* of random variables), find the *best* (the most likely?) Bayesian Network (both DAG and CPDs)



(B,E,A,C,R)=(T,F,F,T,F)
(B,E,A,C,R)=(T,F,T,T,F)
....
(B,E,A,C,R)=(F,T,T,T,F)

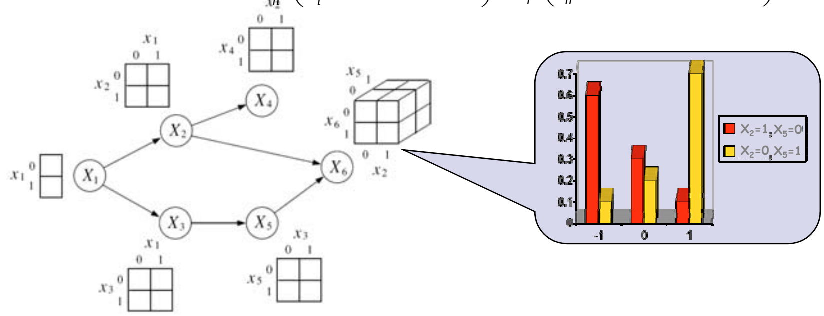


E	B	P(A | E,B)	
e	b	0.9	0.1
e	$\bar{b}$	0.2	0.8
$\bar{e}$	b	0.9	0.1
$\bar{e}$	$\bar{b}$	0.01	0.99

MLE for general BN parameters

- If we assume the parameters for each CPD are globally independent, and all nodes are **fully observed**, then the log-likelihood function decomposes into a sum of local terms, one per node:

$$\ell(\theta;D) = \log p(D \mid \theta) = \log \prod_{n} \left(\prod_{i} p(x_{n,i} \mid \mathbf{x}_{n,\pi_i}, \theta_i) \right) = \sum_{i} \left(\sum_{n} \log p(x_{n,i} \mid \mathbf{x}_{n,\pi_i}, \theta_i) \right)$$

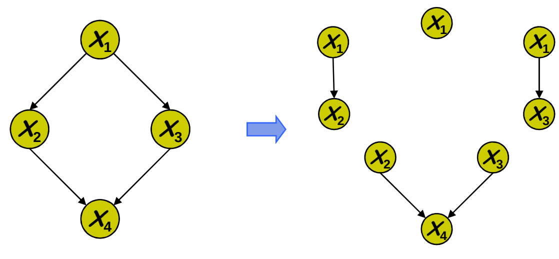


Example: decomposable likelihood of a directed model

- Consider the distribution defined by the directed acyclic GM:

$$p(x \mid \theta) = p(x_1 \mid \theta_1) p(x_2 \mid x_1, \theta_1) p(x_3 \mid x_1, \theta_3) p(x_4 \mid x_2, x_3, \theta_1)$$

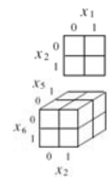
- This is exactly like learning four separate small BNs, each of which consists of a node and its parents.



E.g.: MLE for BNs with tabular CPDs

- Assume each CPD is represented as a table (multinomial) where

$$\theta_{ijk} \overset{\text{def}}{=} p(X_i = j \mid X_{\pi_i} = k)$$



  - Note that in case of multiple parents, $\mathbf{X}_{\pi_i}$ will have a composite state, and the CPD will be a high-dimensional table
  - The sufficient statistics are counts of family configurations

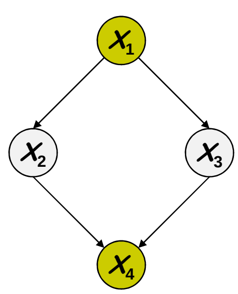
$$n_{ijk} \overset{\text{def}}{=} \sum_n x_{n,i}^j x_{n,\pi_i}^k$$

- The log-likelihood is $\ell(\theta; D) = \log \prod_{i,j,k} \theta_{ijk}^{n_{ijk}} = \sum_{i,j,k} n_{ijk} \log \theta_{ijk}$
- Using a Lagrange multiplier to enforce $\sum_j \theta_{ijk} = 1$, we get: $\theta_{ijk}^{ML} = \dfrac{n_{ijk}}{\sum_{i,j',k} n_{ij'k}}$

What if some nodes are not observed?

- Consider the distribution defined by the directed acyclic GM:

$$p(x \mid \theta) = p(x_1 \mid \theta_1) p(x_2 \mid x_1, \theta_1) p(x_3 \mid x_1, \theta_3) p(x_4 \mid x_2, x_3, \theta_1)$$



- Need to compute $p(x_H|x_V)$ → inference

Probabilistic Inference

- Computing statistical queries regarding the network, e.g.:
  - Is node X independent on node Y given nodes Z,W ?
  - What is the probability of X=true if (Y=false and Z=true)?
  - What is the joint distribution of (X,Y) if Z=false?
  - What is the likelihood of some full assignment?
  - What is the most likely assignment of values to all or a subset the nodes of the network?

- General purpose algorithms exist to fully automate such computation
  - Computational cost depends on the topology of the network
  - Exact inference:
    - The junction tree algorithm
  - Approximate inference;
    - Loopy belief propagation, variational inference, Monte Carlo sampling

Inferential Query 1: Likelihood

- Most of the queries one may ask involve **evidence**
  - Evidence $\mathbf{x}_v$ is an assignment of values to a set $\mathbf{X}_v$ of nodes in the GM over varialbe set $\mathbf{X}=\{X_1, X_2, \ldots, X_n\}$
  - Without loss of generality $\mathbf{X}_v=\{X_{k+1}, \ldots, X_n\}$,
  - Write $\mathbf{X}_H=\mathbf{X}\backslash\mathbf{X}_v$ as the set of hidden variables, $\mathbf{X}_H$ can be $\varnothing$ or $\mathbf{X}$

- Simplest query: compute probability of evidence

$$P(\mathbf{x}_v) = \sum_{\mathbf{x}_H} P(\mathbf{X}_H, , \mathbf{X}_v) = \sum_{x_1} \cdots \sum_{x_k} P(x_1, \ldots, x_k, \mathbf{x}_v)$$

  - this is often referred to as computing the **likelihood** of $\mathbf{x}_v$

Inferential Query 2: Conditional Probability

- Often we are interested in the **conditional probability distribution** of a variable given the evidence

$$P(\mathbf{X_H} \mid \mathbf{X_V} = \mathbf{x_V}) = \frac{P(\mathbf{X_H}, \mathbf{x_V})}{P(\mathbf{x_V})} = \frac{P(\mathbf{X_H}, \mathbf{x_V})}{\sum_{\mathbf{x_H}} P(\mathbf{X_H} = \mathbf{x_H}, \mathbf{x_V})}$$

  - this is the ***a posteriori* belief** in $\mathbf{X_H}$, given evidence $\mathbf{x_v}$

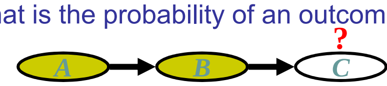
- We usually query a subset $\mathbf{Y}$ of all hidden variables $\mathbf{X_H}=\{\mathbf{Y},\mathbf{Z}\}$ and "don't care" about the remaining, $\mathbf{Z}$:

$$P(\mathbf{Y} \mid \mathbf{x_V}) = \sum_{\mathbf{z}} P(\mathbf{Y}, \mathbf{Z} = \mathbf{z} \mid \mathbf{x_V})$$

  - the process of summing out the "don't care" variables z is called **marginalization**, and the resulting $P(\mathbf{Y}|\mathbf{x_v})$ is called a **marginal** prob.

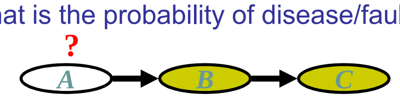
Applications of *a posteriori* Belief

- **Prediction**: what is the probability of an outcome given the starting condition



  - the query node is a descendent of the evidence

- **Diagnosis**: what is the probability of disease/fault given symptoms



  - the query node an ancestor of the evidence

- **Learning** under partial observation
  - fill in the unobserved values under an "EM" setting (more later)

- The directionality of information flow between variables is not restricted by the directionality of the edges in a GM
  - probabilistic inference can combine evidence form all parts of the network

Inferential Query 3: Most Probable Assignment

- In this query we want to find the **most probable joint assignment** (MPA) for ***some*** variables of interest
- Such reasoning is usually performed under some given evidence $\mathbf{x}_v$, and ignoring (the values of) other variables $\mathbf{Z}$:

$$\mathbf{Y}^* \mid \mathbf{x}_V = \arg\max_{\mathbf{y}} P(\mathbf{Y} \mid \mathbf{x}_V) = \arg\max_{\mathbf{y}} \sum_{\mathbf{z}} P(\mathbf{Y}, \mathbf{Z} = \mathbf{z} \mid \mathbf{x}_V)$$

  - this is the **maximum *a posteriori*** configuration of $\mathbf{Y}$.

Complexity of Inference

Thm:

Computing $P(X_H=x_H \mid x_v)$ in an arbitrary BN is NP-hard

- **Hardness does not mean we cannot solve inference**
  - **It implies that we cannot find a general procedure that works efficiently for arbitrary BNs**
  - **For particular families of BNs, we can have provably efficient procedures**

Approaches to inference

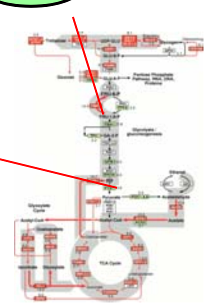
- Exact inference algorithms
  - The elimination algorithm √
  - The junction tree algorithms √ (but will not cover in detail here)
- Approximate inference techniques
  - Stochastic simulation / sampling methods √
  - Markov chain Monte Carlo methods √
  - Variational algorithms (will be covered in advanced ML courses)

Marginalization and Elimination

- A signal transduction pathway:

$$A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$$

What is the likelihood that protein E is active?



- Query: $P(e)$

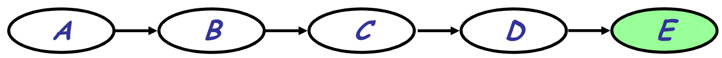
$$P(e) = \underbrace{\sum_d \sum_c \sum_b}\sum_a P(a,b,c,d,e)$$

a naïve summation needs to enumerate over an exponential number of terms

- By chain decomposition, we get

$$= \sum_d \sum_c \sum_b \sum_a P(a)P(b\,|\,a)P(c\,|\,b)P(d\,|\,c)P(e\,|\,d)$$

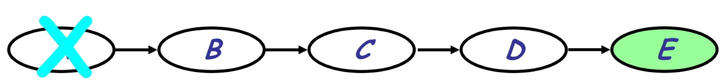
Elimination on Chains



- Rearranging terms ...

$$P(e) = \sum_d \sum_c \sum_b \sum_a P(a)P(b \mid a)P(c \mid b)P(d \mid c)P(e \mid d)$$
$$= \sum_d \sum_c \sum_b P(c \mid b)P(d \mid c)P(e \mid d) \sum_a P(a)P(b \mid a)$$

Elimination on Chains

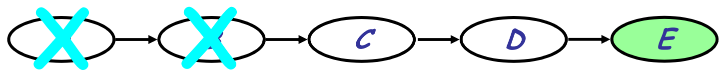


- Now we can perform innermost summation

$$P(e) = \sum_d \sum_c \sum_b P(c \mid b)P(d \mid c)P(e \mid d) \underbrace{\sum_a P(a)P(b \mid a)}$$
$$= \sum_d \sum_c \sum_b P(c \mid b)P(d \mid c)P(e \mid d)\, p(b)$$

- This summation "eliminates" one variable from our summation argument at a "local cost".

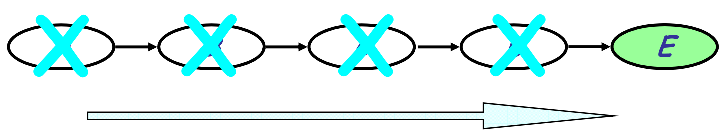
Elimination in Chains



- Rearranging and then summing again, we get

$$
\begin{aligned}
P(e) &= \sum_d \sum_c \sum_b P(c \mid b) P(d \mid c) P(e \mid d) p(b) \\
&= \sum_d \sum_c P(d \mid c) P(e \mid d) \sum_b P(c \mid b) p(b) \\
&= \sum_d \sum_c P(d \mid c) P(e \mid d) p(c)
\end{aligned}
$$

Elimination in Chains



- Eliminate nodes one by one all the way to the end, we get

$$P(e) = \sum_d P(e \mid d) p(d)$$

- Complexity:
  - Each step costs $O(|Val(X_i)|*|Val(X_{i+1})|)$ operations: $O(nk^2)$
  - Compare to naïve evaluation that sums over joint values of n-1 variables $O(k^n)$

Inference on General BN via Variable Elimination

General idea:

- Write query in the form

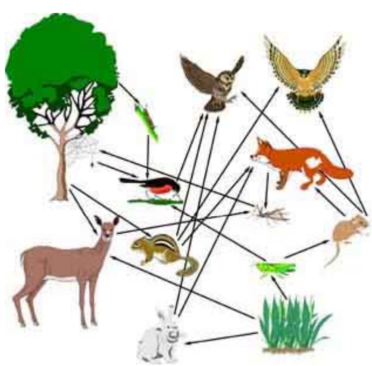
$$P(X_1, \boldsymbol{e}) = \sum_{x_n} \cdots \sum_{x_3} \sum_{x_2} \prod_i P(x_i \mid pa_i)$$

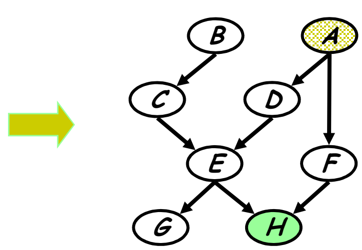
  - this suggests an "elimination order" of latent variables to be marginalized
- Iteratively
  - Move all irrelevant terms outside of innermost sum
  - Perform innermost sum, getting a new term
  - Insert the new term into the product
- wrap-up

$$P(X_1 \mid \boldsymbol{e}) = \frac{P(X_1, \boldsymbol{e})}{P(\boldsymbol{e})}$$

A more complex network

A food web

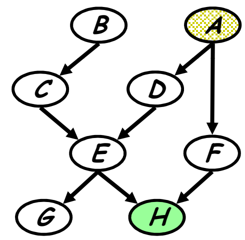




What is the probability that hawks are leaving given that the grass condition is poor?

Example: Variable Elimination

- Query: $P(A \mid h)$
  - Need to eliminate: B,C,D,E,F,G,H



- Initial factors:

$$P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)$$

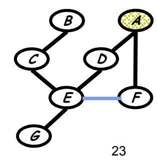
- Choose an elimination order: H,G,F,E,D,C,B

- Step 1:
  - **Conditioning** (fix the evidence node (i.e., h) on its observed value (i.e., $\tilde{h}$)):

$$m_h(e,f) = p(h = \tilde{h} \mid e, f)$$

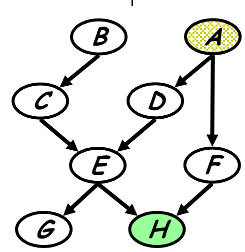
  - This step is isomorphic to a marginalization step:

$$m_h(e,f) = \sum_h p(h \mid e,f)\delta(h = \tilde{h})$$



Example: Variable Elimination

- Query: $P(B \mid h)$
  - Need to eliminate: B,C,D,E,F,G



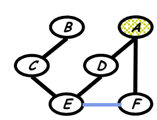
- Initial factors:

$$P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)m_h(e,f)$$

- Step 2: Eliminate G
  - compute

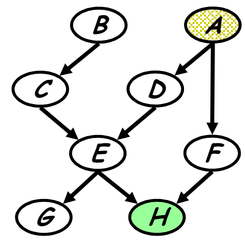
$$m_g(e) = \sum_g p(g \mid e) = 1$$

$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)m_g(e)m_h(e,f)$$
$$= P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)m_h(e,f)$$



Example: Variable Elimination

- Query: *P(B | h)*
  - Need to eliminate: *B,C,D,E,F*



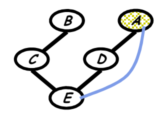
- Initial factors:

$$P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$$
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)\underline{P(f|a)m_h(e,f)}$$

- Step 3: Eliminate *F*
  - compute

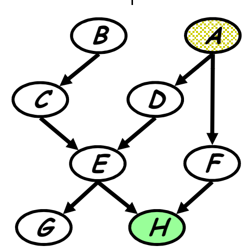
$$m_f(e,a) = \sum_f p(f|a)m_h(e,f)$$

$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)\underline{m_f(a,e)}$$



Example: Variable Elimination

- Query: *P(B | h)*
  - Need to eliminate: *B,C,D,E*



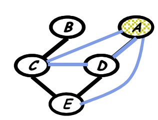
- Initial factors:

$$P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$$
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)\underline{P(e|c,d)m_f(a,e)}$$

- Step 4: Eliminate *E*
  - compute

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

$$\Rightarrow P(a)P(b)P(c|b)P(d|a)\underline{m_e(a,c,d)}$$

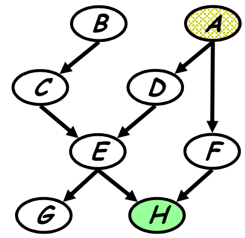


Example: Variable Elimination

- Query: $P(B \mid h)$
  - Need to eliminate: B, C, D

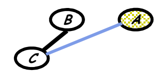
- Initial factors:

$$P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$$
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)m_f(a,e)$$
$$\Rightarrow P(a)P(b)P(c|b)\underline{P(d|a)m_e(a,c,d)}$$



- Step 5: Eliminate D
  - compute

$$m_d(a,c) = \sum_d p(d|a)m_e(a,c,d)$$



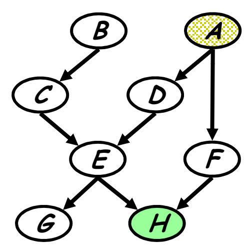
$$\Rightarrow P(a)P(b)P(c|d)\underline{m_d(a,c)}$$

Example: Variable Elimination

- Query: $P(B \mid h)$
  - Need to eliminate: B, C

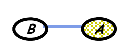
- Initial factors:

$$P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)m_f(a,e)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)m_e(a,c,d)$$
$$\Rightarrow P(a)P(b)\underline{P(c|d)m_d(a,c)}$$



- Step 6: Eliminate C
  - compute

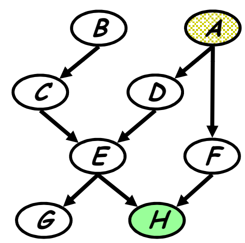
$$m_c(a,b) = \sum_c p(c|b)m_d(a,c)$$



$$\Rightarrow P(a)P(b)P(c|d)\underline{m_d(a,c)}$$

Example: Variable Elimination

- Query: *P(B | h)*
  - Need to eliminate: *B*
- Initial factors:



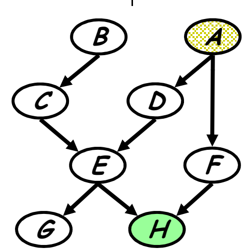
$$P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)m_f(a,e)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)m_e(a,c,d)$$
$$\Rightarrow P(a)P(b)P(c|d)m_d(a,c)$$
$$\Rightarrow P(a)\underline{P(b)m_c(a,b)}$$

- Step 7: Eliminate *B*
  - compute $$m_b(a) = \sum_b p(b)m_c(a,b)$$

$$\Rightarrow P(a)\underline{m_b(a)}$$

Example: Variable Elimination

- Query: *P(B | h)*
  - Need to eliminate: *B*
- Initial factors:



$$P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)m_f(a,e)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)m_e(a,c,d)$$
$$\Rightarrow P(a)P(b)P(c|d)m_d(a,c)$$
$$\Rightarrow P(a)P(b)m_c(a,b)$$
$$\Rightarrow P(a)m_b(a)$$

- Step 8: Wrap-up $$p(a,\tilde{h}) = p(a)m_b(a), \quad p(\tilde{h}) = \sum_a p(a)m_b(a)$$

$$\Rightarrow P(a|\tilde{h}) = \frac{p(a)m_b(a)}{\sum p(a)m_b(a)}$$

Complexity of variable elimination

- Suppose in one elimination step we compute

$$m_x(y_1, \dots, y_k) = \sum_x m'_x(x, y_1, \dots, y_k)$$

$$m'_x(x, y_1, \dots, y_k) = \prod_{i=1}^{k} m_i(x, \mathbf{y}_{c_i})$$

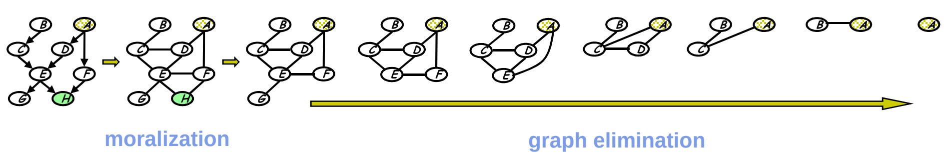
This requires

- $k \bullet |\mathrm{Val}(X)| \bullet \prod_i |\mathrm{Val}(\mathbf{Y}_{C_i})|$ multiplications
  - For each value of $x, y_1, \dots, y_k$, we do k multiplications
- $|\mathrm{Val}(X)| \bullet \prod_i |\mathrm{Val}(\mathbf{Y}_{C_i})|$ additions
  - For each value of $y_1, \dots, y_k$, we do $|Val(X)|$ additions

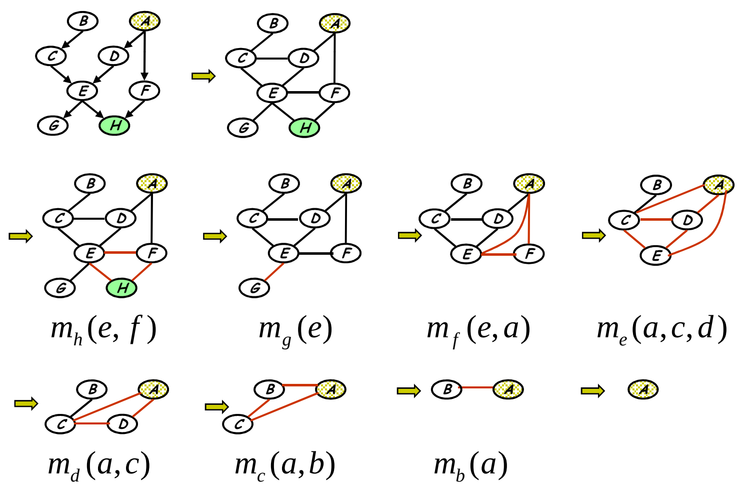
Complexity is **exponential** in number of variables in the intermediate factor

Understanding Variable Elimination

- A graph elimination algorithm

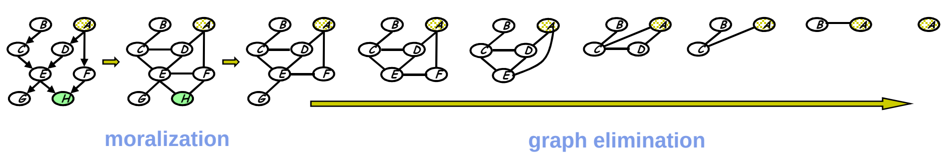


Elimination Cliques



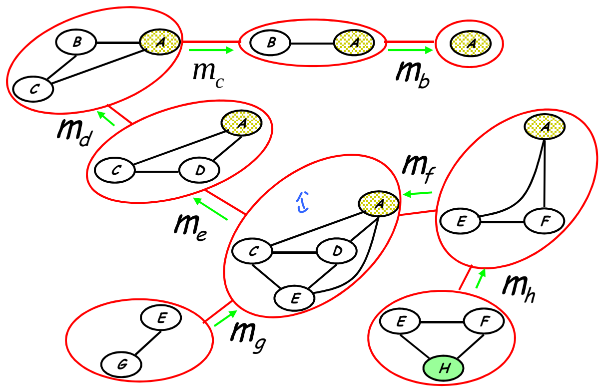
Understanding Variable Elimination

- A graph elimination algorithm



- Intermediate terms correspond to the cliques resulted from elimination
  - "good" elimination orderings lead to **small cliques** and hence reduce complexity
    (what will happen if we eliminate "e" first in the above graph?)
  - finding the optimum ordering is NP-hard, but for many graph optimum or near-optimum can often be heuristically found
- Applies to undirected GMs

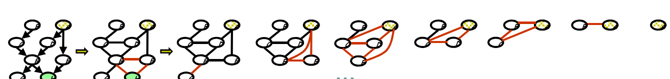
A clique tree



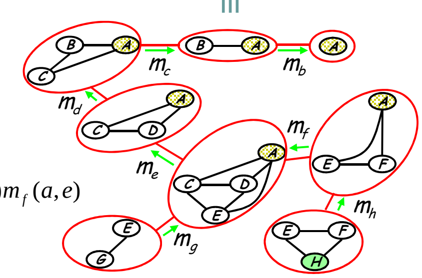
$$m_e(a,c,d)$$
$$= \sum_e p(e|c,d) m_g(e) m_f(a,e)$$

From Elimination to Message Passing

- Our algorithm so far answers only one query (e.g., on one node), do we need to do a complete elimination for every such query?
- Elimination ≡ message passing on a **clique tree**



≡

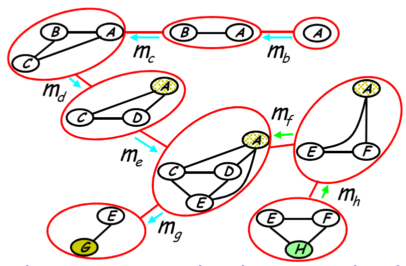


$$m_e(a,c,d)$$
$$= \sum_e p(e|c,d) m_g(e) m_f(a,e)$$

- Messages can be reused

From Elimination to Message Passing

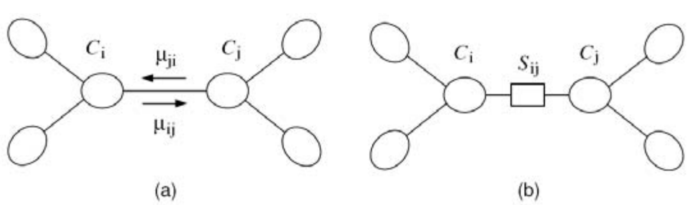
- Our algorithm so far answers only one query (e.g., on one node), do we need to do a complete elimination for every such query?
- Elimination ≡ message passing on a **clique tree**
  - **Another query ...**



- Messages m_f and m_h are reused, others need to be recomputed

The Junction Tree Algorithm

- Shafer-Shenoy algorithm



- Message from clique *i* to clique *j* :

$$\mu_{i \to j} = \sum_{C_i \setminus S_{ij}} \psi_{C_i} \prod_{k \neq j} \mu_{k \to i}(S_{ki})$$

- Clique marginal

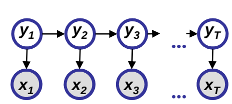
$$p(C_i) \propto \psi_{C_i} \prod_k \mu_{k \to i}(S_{ki})$$

A Sketch of the Junction Tree Algorithm

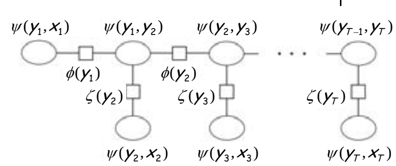
- **The algorithm**
  - Construction of junction trees --- a special **clique tree**
  - Propagation of probabilities --- a message-passing protocol
- Results in marginal probabilities of all cliques --- solves all queries in a single run
- A **generic** exact inference algorithm for any GM
- **Complexity**: exponential in the size of the maximal clique --- a good elimination order often leads to small maximal clique, and hence a good (i.e., thin) JT
- Many well-known algorithms are special cases of JT
  - Forward-backward, Kalman filter, Peeling, Sum-Product ...

The Junction tree algorithm for HMM

- A junction tree for the HMM

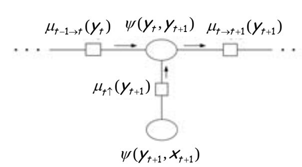


$\Rightarrow$



- Rightward pass

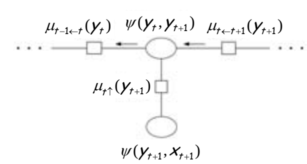
$$\mu_{t\to t+1}(y_{t+1}) = \sum_{y_t}\psi(y_t,y_{t+1})\mu_{t-1\to t}(y_t)\mu_{t\uparrow}(y_{t+1})$$
$$= \sum_{y_t} p(y_{t+1}\mid y_t)\mu_{t-1\to t}(y_t)p(x_{t+1}\mid y_{t+1})$$
$$= p(x_{t+1}\mid y_{t+1})\sum_{y_t} a_{y_t,y_{t+1}}\mu_{t-1\to t}(y_t)$$



  - This is exactly the forward algorithm!

- Leftward pass …

$$\mu_{t-1\leftarrow t}(y_t) = \sum_{y_{t+1}}\psi(y_t,y_{t+1})\mu_{t\leftarrow t+1}(y_{t+1})\mu_{t\uparrow}(y_{t+1})$$
$$= \sum_{y_{t+1}} p(y_{t+1}\mid y_t)\mu_{t\leftarrow t+1}(y_{t+1})p(x_{t+1}\mid y_{t+1})$$



  - This is exactly the backward algorithm!

Summary

- Represent dependency structure with a directed acyclic graph
  - Node <-> random variable
  - Edges encode dependencies
    - Absence of edge -> conditional independence
  - Plate representation
  - A BN is a database of prob. Independence statement on variables

- The factorization theorem of the joint probability
  - Local specification → globally consistent distribution
  - Local representation for exponentially complex state-space
- Support efficient inference and learning