Exact Inference: Variable Elimination

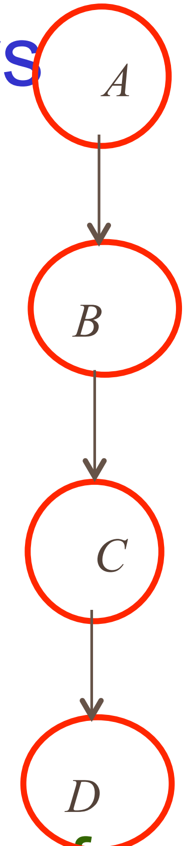
Sargur Srihari
srihari@cedar.buffalo.edu

Topics

- Exact Inference
- Variable Elimination (VE)
- Sum-Product Algorithm
- Variable Ordering for VE

Principles of Exact Inference

- We show that same BN structure that allows compaction of complex distributions also helps support inference
  - Consider BN: $A \rightarrow B \rightarrow C \rightarrow D$
    - E.g., sequence of words: CPDs are first order word probabilities



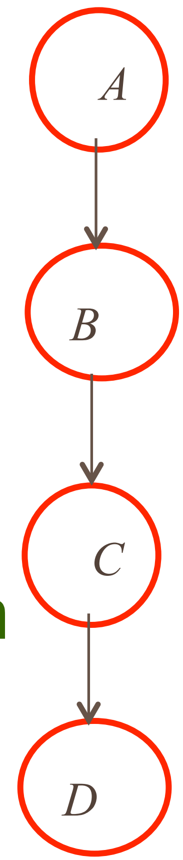
- We consider phased computation
    - Probabilities of four words: *The, quick, brown, fox*
  - Use results of a previous phase in computation of next phase
  - Then reformulate this process in terms of a global computation on the joint distribution

Exact Inference: Variable Elimination

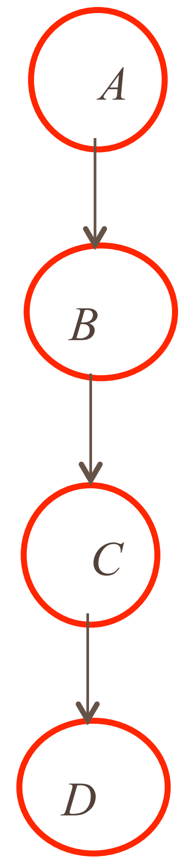
- To compute $P(B)$,
  - i.e., distribution of values b of B, we have

$$P(B) = \sum_a P(A,B) = \sum_a P(a)P(B \mid a)$$

  - required $P(a)$, $P(b|a)$ available in BN
- If A has k values and B has m values
  - For each b: k multiplications and k-1 addition
  - Since there are m values of B, process is repeated for each value of b:
    - this computation is $O(k \times m)$



Moving Down BN



- Assume we want to compute $P(C)$
- Using same analysis

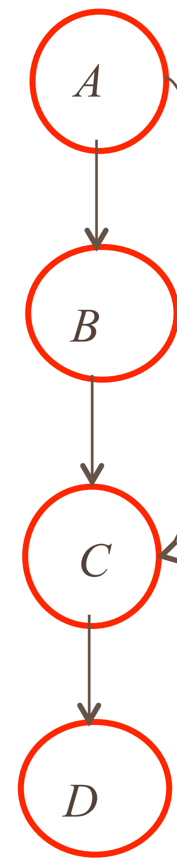
$$P(C) = \sum_b P(B,C) = \sum_b P(b)P(C \mid b)$$

  - $P(c|b)$ is given in CPD
  - But $P(B)$ is not given as network parameters
  - It can be computed using

$$P(B) = \sum_a P(A,B) = \sum_a P(a)P(B \mid a)$$

  - If B and C have k values each, complexity is $O(k^2)$

Computation depends on Structure

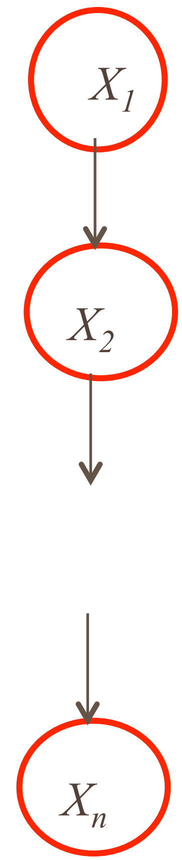


1. Structure of BN is critical for computation
   - If A had been a parent of C

$$P(C) = \sum_b P(b)P(C \mid b)$$

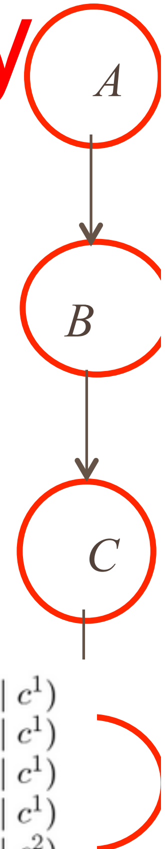
   - would not have sufficed
2. Algorithm does not compute single values but sets of values at a time
   - $P(B)$ over all possible values of B are used to compute $P(C)$

Complexity of General Chain



- In general, if we have $X_1 \rightarrow X_2 \rightarrow \ldots.. \rightarrow X_n$
- and there are k values of X_i, total cost is $O(nk^2)$
- Naïve evaluation
- Generate entire joint and summing it out
- Would generate k^n probabilities for the events $x_1, \ldots x_n$
- In this example, despite exponential size of joint distribution **we can do inference in linear time**

Insight that avoids exponentiality



- The joint probability decomposes as

$$P(A,B,C,D)=P(A)P(B|A)P(C|B)P(D|C)$$

  - To compute $P(D)$ we need to sum together all entries where $D=d^1$
    - And separately entries where $D=d^2$
  - Exact computation for $P(D)$ is
  - Examine summation
    - 3^{rd} & 4^{th} terms of first 2 terms:
    - $P(c^1|b^1)P(d^1|c^1)$
    - Modify to first compute
    - $P(a^1)P(b^1|a^1)+P(a^2)P(b^1|a^2)$
    - then multiply by common term

$$\begin{array}{lllll}
 & P(a^1) & P(b^1 \mid a^1) & P(c^1 \mid b^1) & P(d^1 \mid c^1) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^1 \mid b^1) & P(d^1 \mid c^1) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^1 \mid b^2) & P(d^1 \mid c^1) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^1 \mid b^2) & P(d^1 \mid c^1) \\
+ & P(a^1) & P(b^1 \mid a^1) & P(c^2 \mid b^1) & P(d^1 \mid c^2) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^2 \mid b^1) & P(d^1 \mid c^2) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^2 \mid b^2) & P(d^1 \mid c^2) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^2 \mid b^2) & P(d^1 \mid c^2)
\end{array}$$

$$\begin{array}{lllll}
 & P(a^1) & P(b^1 \mid a^1) & P(c^1 \mid b^1) & P(d^2 \mid c^1) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^1 \mid b^1) & P(d^2 \mid c^1) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^1 \mid b^2) & P(d^2 \mid c^1) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^1 \mid b^2) & P(d^2 \mid c^1) \\
+ & P(a^1) & P(b^1 \mid a^1) & P(c^2 \mid b^1) & P(d^2 \mid c^2) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^2 \mid b^1) & P(d^2 \mid c^2) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^2 \mid b^2) & P(d^2 \mid c^2) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^2 \mid b^2) & P(d^2 \mid c^2)
\end{array}$$

First Transformation of sum

- Same structure is repeated throughout table
- Performing the same transformation we get the summation for $P(D)$ as

$$\begin{array}{llll} & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) & P(c^1 \mid b^1) & P(d^1 \mid c^1) \\ + & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) & P(c^1 \mid b^2) & P(d^1 \mid c^1) \\ + & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) & P(c^2 \mid b^1) & P(d^1 \mid c^2) \\ + & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) & P(c^2 \mid b^2) & P(d^1 \mid c^2) \end{array}$$

$$\begin{array}{llll} & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) & P(c^1 \mid b^1) & P(d^2 \mid c^1) \\ + & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) & P(c^1 \mid b^2) & P(d^2 \mid c^1) \\ + & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) & P(c^2 \mid b^1) & P(d^2 \mid c^2) \\ + & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) & P(c^2 \mid b^2) & P(d^2 \mid c^2) \end{array}$$

  - Observe certain terms are repeated several times in this expression
  - $P(a^1)P(b^1|a^1)+P(a^2)P(b^1|a^2)$ and
  - $P(a^1)P(b^2|a^1)+P(a^2)P(b^2|a^2)$

    are repeated four times

2nd & 3rd transformation on the sum

- Defining τ_1: $Val(B) \rightarrow R$
  - where $\tau_1(b^1)$ and $\tau_1(b^2)$ are the two expressions, we get

$$\begin{array}{ll} & \tau_1(b^1) \quad P(c^1 \mid b^1) \quad P(d^1 \mid c^1) \\ + & \tau_1(b^2) \quad P(c^1 \mid b^2) \quad P(d^1 \mid c^1) \\ + & \tau_1(b^1) \quad P(c^2 \mid b^1) \quad P(d^1 \mid c^2) \\ + & \tau_1(b^2) \quad P(c^2 \mid b^2) \quad P(d^1 \mid c^2) \end{array}$$

$$\begin{array}{ll} & \tau_1(b^1) \quad P(c^1 \mid b^1) \quad P(d^2 \mid c^1) \\ + & \tau_1(b^2) \quad P(c^1 \mid b^2) \quad P(d^2 \mid c^1) \\ + & \tau_1(b^1) \quad P(c^2 \mid b^1) \quad P(d^2 \mid c^2) \\ + & \tau_1(b^2) \quad P(c^2 \mid b^2) \quad P(d^2 \mid c^2) \end{array}$$

– Can reverse the order of a sum and product

- sum first, product next

$$\begin{array}{ll} & (\tau_1(b^1)P(c^1 \mid b^1) + \tau_1(b^2)P(c^1 \mid b^2)) \quad P(d^1 \mid c^1) \\ + & (\tau_1(b^1)P(c^2 \mid b^1) + \tau_1(b^2)P(c^2 \mid b^2)) \quad P(d^1 \mid c^2) \end{array}$$

$$\begin{array}{ll} & (\tau_1(b^1)P(c^1 \mid b^1) + \tau_1(b^2)P(c^1 \mid b^2)) \quad P(d^2 \mid c^1) \\ + & (\tau_1(b^1)P(c^2 \mid b^1) + \tau_1(b^2)P(c^2 \mid b^2)) \quad P(d^2 \mid c^2) \end{array}$$

Fourth Transformation of sum

- Again notice shared expressions that are better computed once and used multiple times
  - We define τ_2: $Val(C) \rightarrow R$

$$\tau_2(c^1)=\tau_1(b^1)P(c^1|b^1)+\tau_1(b^2)P(c^1|b^2)$$

$$\tau_2(c^2)=\tau_1(b^1)P(c^2|b^1)+\tau_1(b^2)P(c^2|b^2)$$

$$\begin{array}{ll} & \tau_2(c^1) \quad P(d^1 \mid c^1) \\ + & \tau_2(c^2) \quad P(d^1 \mid c^2) \end{array}$$

$$\begin{array}{ll} & \tau_2(c^1) \quad P(d^2 \mid c^1) \\ + & \tau_2(c^2) \quad P(d^2 \mid c^2) \end{array}$$

Summary of computation

- We begin by computing $\tau_1(B)$
- Requires 4 multiplications and 2 additions
- Using it we can compute $\tau_2(C)$ which also requires 4 multis and 2 adds
- Finally we compute $P(D)$ at same cost
- Total no of ops is 18
- Joint distribution requires 16 x $3=48$ mps and 14 adds

Computation Summary

- Transformation we have performed has steps

$$P(D)=\sum_C\sum_B\sum_A P(A)P(B\mid A)P(C\mid B)P(D\mid C)$$

- We push the first summation resulting in

$$P(D)=\sum_C P(D\mid C)\sum_B P(C\mid B)\sum_A P(A)P(B\mid A)$$

- We compute the product $\psi_1(A,B)=P(A)P(B|A)$ and sum out A to obtain the function $\tau_1(B)=\sum_A \psi_1(A,B)$
  - For each value of b, we compute

$$\tau_1(b)=\sum_A \psi_1(A,b)=\sum_A P(A)P(b\mid A)$$

- We then continue

$$\psi_2(B,C)=\tau_1(B)P(C\mid B)$$

$$\tau_2(C)=\sum_B \psi_2(B,C)$$

  - Resulting $\tau_2(C)$ is used to compute $P(D)$

Computation is Dynamic Programming

- Naiive way for $P(D)=\sum_C \sum_B \sum_A P(A)P(B \mid A)P(C \mid B)P(D \mid C)$

  would have us compute every

  $$P(b)=\sum_A P(A)P(b \mid A)$$

  – many times, once for every value of C and D
- For a chain of length n this would be computed exponentially many times
- Dynamic Programming inverts order of computation– performing it inside out rather than outside in
  – First computing once for all values in $\tau_1(B)$, that allows us to compute $\tau_2(C)$ once for all, etc.

Ideas that prevented exponential blowup

- Because of structure of BN, some subexpressions depend only on a small no. of variables
- By computing and caching these results we can avoid generating them exponential no. of times

Variable Elimination: Use of Factors

- To formalize VE need concept of factors ϕ
- χ is a set of r.v.s, $\boldsymbol{X}$ is a subset $\boldsymbol{X} \subseteq \chi$
- We say $Scope[\phi]= \boldsymbol{X}$
- Factor associates a real value for each setting of it arguments ϕ*: $Val(\boldsymbol{X}) \rightarrow R$*
- Factor in BN is a product term
  - say $\phi(A,B,C)= P(A,B/C)$

Factors in BNs and MNs

- Useful in both BNs and MNs
- Factor in BN is a product term, say $\phi(A,B,C)=P(A,B/C)$
- Factor in MN comes from Gibbs distribution, say $\phi(A,B)$
  - Definition of Gibbs:

$$P_\Phi(X_1,..X_n)=\frac{1}{Z}\tilde{P}(X_1,..X_n)$$

where

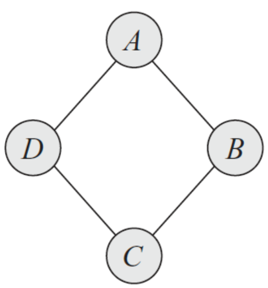
$$\tilde{P}(X_1,..X_n)=\prod_{i=1}^{m}\phi_i(D_i)$$

is an unnomalized measure and

$$Z=\sum_{X_1,..X_n}\tilde{P}(X_1,..X_n) \text{ is a normalizing constant}$$

called the partition function

  - Example:

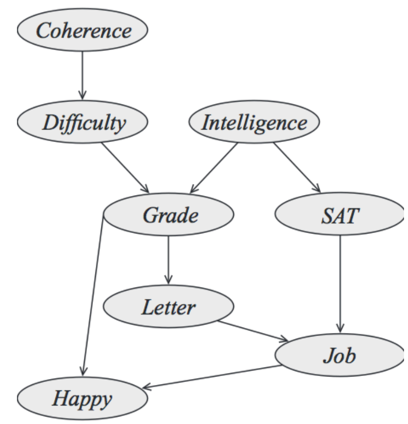


$\phi_1[A,B]$		
a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10

Role of Factor Operations

- The joint distribution is a product of factors

$$P(C,D,I,G,S,L,J,H) = P(C)P(D|C)P(I)P(G|I,D)P(S|I)P(L|G)P(J|L)P(H|G,J) =$$
$$\phi_C(C)\ \phi_D(D,C)\ \phi_I(I)\ \phi_G(G,I,D)\ \phi_S(S,I)\ \phi_L(L,G)\ \phi_J(J,L,S)\ \phi_H(H,G,J)$$



- Inference is a task of marginalization

$$\boxed{P(J) = \sum_L \sum_S \sum_G \sum_H \sum_I \sum_D \sum_C P(C,D,I,G,S,L,J,H)}$$

- We wish to systematically eliminate all variables other than J

About Factors

- Inference Algorithms manipulate factors
- Occur in both directed and undirected PGMs
- Need two operations:
  - Factor Product: $\Phi_1(X,Y)\ \Phi_2(Y,Z)$
  - Factor Marginalization: $\psi(\boldsymbol{X}) = \sum_{Y} \phi(\boldsymbol{X}, Y)$

Factor Product

- Let X, Y and Z be three disjoint sets of variables and let $\Phi_1(X,Y)$ and $\Phi_2(Y,Z)$ be two factors.
- The factor product is the mapping $Val(X,Y,Z) \rightarrow R$ as follows

$$\psi(X,Y,Z) = \Phi_1(X,Y)\ \Phi_2(Y,Z)$$

- An example:

Φ_1: $3 \times 2 = 6$ entries

Φ_2: $2 \times 2 = 4$ entries

yields

ψ: $3 \times 2 \times 2 = 12$ entries

a^1	b^1	0.5
a^1	b^2	0.8
a^2	b^1	0.1
a^2	b^2	0
a^3	b^1	0.3
a^3	b^2	0.9

b^1	c^1	0.5
b^1	c^2	0.7
b^2	c^1	0.1
b^2	c^2	0.2

a^1	b^1	c^1	$0.5 \cdot 0.5 = 0.25$
a^1	b^1	c^2	$0.5 \cdot 0.7 = 0.35$
a^1	b^2	c^1	$0.8 \cdot 0.1 = 0.08$
a^1	b^2	c^2	$0.8 \cdot 0.2 = 0.16$
a^2	b^1	c^1	$0.1 \cdot 0.5 = 0.05$
a^2	b^1	c^2	$0.1 \cdot 0.7 = 0.07$
a^2	b^2	c^1	$0 \cdot 0.1 = 0$
a^2	b^2	c^2	$0 \cdot 0.2 = 0$
a^3	b^1	c^1	$0.3 \cdot 0.5 = 0.15$
a^3	b^1	c^2	$0.3 \cdot 0.7 = 0.21$
a^3	b^2	c^1	$0.9 \cdot 0.1 = 0.09$
a^3	b^2	c^2	$0.9 \cdot 0.2 = 0.18$

Factor Marginalization

- $\boldsymbol{X}$ is a set of variables and $Y \notin \boldsymbol{X}$ is a variable
- $\phi(\boldsymbol{X},Y)$ is a factor
- We wish to eliminate Y
- Factor marginalization of Y is a factor ψ s.t.

$$\psi(\boldsymbol{X}) = \sum_{Y} \phi(\boldsymbol{X},Y)$$

a^1	b^1	c^1	0.25
a^1	b^1	c^2	0.35
a^1	b^2	c^1	0.08
a^1	b^2	c^2	0.16
a^2	b^1	c^1	0.05
a^2	b^1	c^2	0.07
a^2	b^2	c^1	0
a^2	b^2	c^2	0
a^3	b^1	c^1	0.15
a^3	b^1	c^2	0.21
a^3	b^2	c^1	0.09
a^3	b^2	c^2	0.18

$\Phi(A,B,C)$

a^1	c^1	0.33
a^1	c^2	0.51
a^2	c^1	0.05
a^2	c^2	0.07
a^3	c^1	0.24
a^3	c^2	0.39

$\psi(A,C)$

Example of Factor Marginalization: Summing-out $Y=B$ when $\boldsymbol{X}=\{A,C\}$

- Process is called summing out of Y in Φ
- *We sum up entities in the table only when the values of $\boldsymbol{X}$ match up*
- If we sum out all variables we get a factor which is a single value of 1
- If we sum out all of the variables in an unnormalized distribution $\tilde{P}_{\phi} = \prod_{i=1}^{N} \phi_i(D_i)$ we get the partition function

Distributivity of product over sum

Example with nos.

$a.b_1+a.b_2=a(b_1+b_2)$: product is distributive
$(a+b_1).(a+b_2)$. ne. $a+(b_1 b_2)$: sum is not
Product distributivity allows fewer operations

$$\psi(A,B) = \sum_{A=a_1}^{a_2} \sum_{B=b_1}^{b_2} A \cdot B = a_1 b_1 + a_1 b_2 + a_2 b_1 + a_2 b_2 \quad \text{requires 4 products, 3 sums}$$

Alternative formulation requires 2 sums, 2 products

$$\psi(A,B) = \sum_{A=a_1}^{a_2} A \cdot \tau(B)$$

$$where \quad \tau(B) = \sum_{B=b_1}^{b_2} B = b_1 + b_2$$

$$\psi(A,B) = a_1 \tau(B) + a_2 \tau(B)$$

Sum first
Product next
Saves ops over
Product first
Sum next

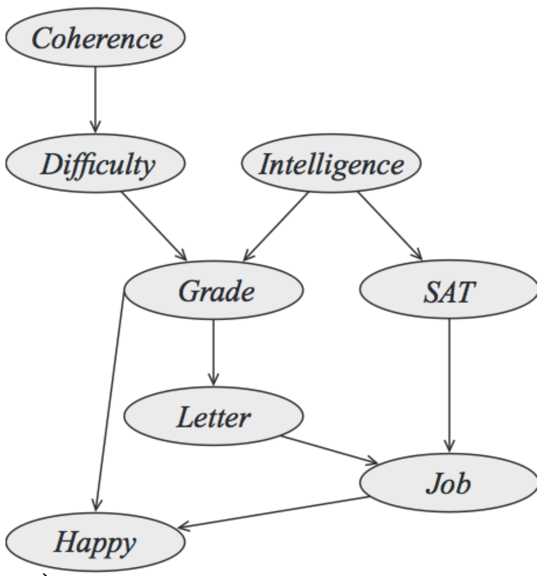
- Factor product and summation behave exactly like product and summation over nos.
- If $X \notin Scope(\phi_1)$ then $\sum_X (\phi_1 \cdot \phi_2) = \phi_1 \sum_X \phi_2$

Sum-Product Variable Elimination Algorithm

- Task of computing the value of an expression of the form $$\sum_{Z} \prod_{\phi \in \Phi} \phi$$
- Called sum-product inference task
  - Sum of Products
- Key insight is that scope of the factors is limited
  - Allowing us to push in some of the *summations*, performing them over the product of only some of the factors
  - We sum out variables one at a time

Inference using Variable Elimination

- Example: Extended Student BN



- We wish to infer $P(J)$

$$P(J) = \sum_H \sum_L \sum_S \sum_G \sum_I \sum_D \sum_C P(C,D,I,G,S,L,J,H)$$

- By chain rule:

$$P(C,D,I,G,S,L,J,H) =$$

$$P(C)P(D|C)P(I)P(G|I,D)P(S|I)P(L|G)P(J|L)P(H|G,J)$$

  – Which is a Sum of Product of factors

Sum-product VE

$$P(J)=\sum_L\sum_S\sum_G\sum_H\sum_I\sum_D\sum_C P(C,D,I,G,S,L,J,H)$$

$P(C,D,I,G,S,L,J,H) = P(C)P(D|C)P(I)P(G|I,D)P(S|I)P(L|G)P(J|L)P(H|G,J) =$

$\phi_C(C)\ \phi_D(D,C)\ \phi_I(I)\ \phi_G(G,I,D)\ \phi_S(S,I)\ \phi_L(L,G)\ \phi_J(J,L,S)\ \phi_H(H,G,J)$

Elimination ordering $C,D,I,H.G,S,L$

1. Eliminating C: $\quad \psi_1(C,D)=\phi_C(C)\phi_D(D,C) \qquad \tau_1(D)=\sum_C \psi_1(C,D)$

Each step involves factor product and factor marginalization

Compute the factors

2. Eliminating D: $\quad \psi_2(G,I,D)=\phi_G(G,I,D)\tau_1(D) \qquad \tau_2(G,I)=\sum_D \psi_2(G,I,D)$

Note we already eliminated one factor with D, but introduced τ_1 involving D

3. Eliminating I: $\quad \psi_3(G,I,S)=\phi_I(I)\phi_S(S,I)\tau_2(G,I) \qquad \tau_3(G,S)=\sum_I \psi_3(G,I,S)$
4. Eliminating H: $\quad \psi_4(G,J,H)=\phi_H(H,G,J) \qquad \tau_4(G,J)=\sum_H \psi_4(G,J,H)$

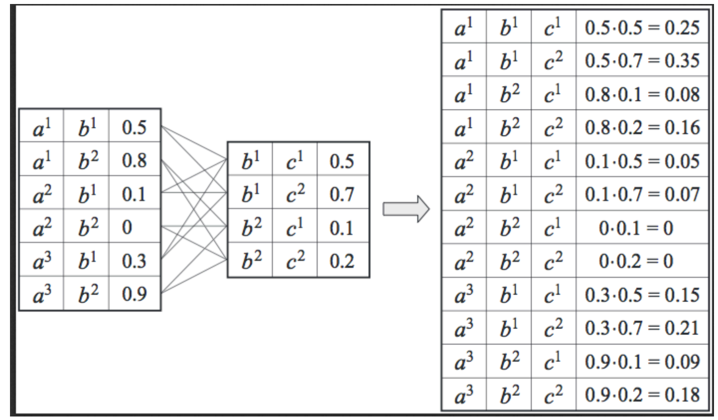
Note $\tau_4(G,J)=1$

5. Eliminating G: $\quad \psi_5(G,J,L,S)=\tau_4(G,J)\tau_3(G,S)\phi_L(L,G) \qquad \tau_5(J,L,S)=\sum_G \psi_5(G,J,L,S)$
6. Eliminating S: $\quad \psi_6(J,L,S)=\tau_5(J,L,S)\cdot\phi_J(J,L,S) \qquad \tau_6(J,L)=\sum_S \psi_6(J,L,S)$
7. Eliminating L: $\quad \psi_7(J,L)=\tau_6(J,L) \qquad \tau_7(J)=\sum_L \psi_7(J,L)$

Computing $\tau(A,C)$= $\Sigma_B \psi(A,B,C) = \Sigma_B \phi(A,B)\phi(B,C)$

1. Factor product

$$\psi(A,B,C)=\phi(A,B)\phi(B,C)$$



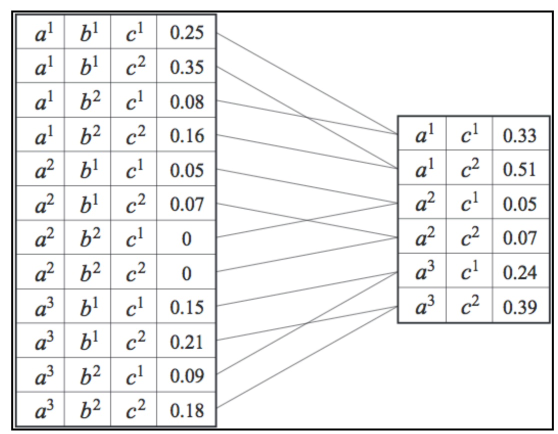
a^1	b^1	0.5
a^1	b^2	0.8
a^2	b^1	0.1
a^2	b^2	0
a^3	b^1	0.3
a^3	b^2	0.9

b^1	c^1	0.5
b^1	c^2	0.7
b^2	c^1	0.1
b^2	c^2	0.2

a^1	b^1	c^1	$0.5 \cdot 0.5 = 0.25$
a^1	b^1	c^2	$0.5 \cdot 0.7 = 0.35$
a^1	b^2	c^1	$0.8 \cdot 0.1 = 0.08$
a^1	b^2	c^2	$0.8 \cdot 0.2 = 0.16$
a^2	b^1	c^1	$0.1 \cdot 0.5 = 0.05$
a^2	b^1	c^2	$0.1 \cdot 0.7 = 0.07$
a^2	b^2	c^1	$0 \cdot 0.1 = 0$
a^2	b^2	c^2	$0 \cdot 0.2 = 0$
a^3	b^1	c^1	$0.3 \cdot 0.5 = 0.15$
a^3	b^1	c^2	$0.3 \cdot 0.7 = 0.21$
a^3	b^2	c^1	$0.9 \cdot 0.1 = 0.09$
a^3	b^2	c^2	$0.9 \cdot 0.2 = 0.18$

2. Factor marginalization

$$\tau(A,C) = \Sigma_B \psi(A,B,C)$$



a^1	b^1	c^1	0.25
a^1	b^1	c^2	0.35
a^1	b^2	c^1	0.08
a^1	b^2	c^2	0.16
a^2	b^1	c^1	0.05
a^2	b^1	c^2	0.07
a^2	b^2	c^1	0
a^2	b^2	c^2	0
a^3	b^1	c^1	0.15
a^3	b^1	c^2	0.21
a^3	b^2	c^1	0.09
a^3	b^2	c^2	0.18

a^1	c^1	0.33
a^1	c^2	0.51
a^2	c^1	0.05
a^2	c^2	0.07
a^3	c^1	0.24
a^3	c^2	0.39

Sum-Product VE Algorithm

- To compute

$$\sum_{Z} \prod_{\phi \in \Phi} \phi$$

- First procedure specifies ordering of k variables Z_i
- Second procedure eliminates a single variable Z (contained in factors Φ') and returns factor τ

Procedure Sum-Product-VE (
    Φ, // Set of factors
    $\boldsymbol{Z}$, // Set of variables to be eliminated
    $\prec$ // Ordering on $\boldsymbol{Z}$
)
    Let $Z_1, \ldots, Z_k$ be an ordering of $\boldsymbol{Z}$ such that
        $Z_i \prec Z_j$ if and only if $i < j$
    for $i = 1, \ldots, k$
        $\Phi \leftarrow$ Sum-Product-Eliminate-Var(Φ, Z_i)
    $\phi^* \leftarrow \prod_{\phi \in \Phi} \phi$
    return ϕ^*

Procedure Sum-Product-Eliminate-Var (
    Φ, // Set of factors
    Z // Variable to be eliminated
)
    $\Phi' \leftarrow \{\phi \in \Phi : Z \in Scope[\phi]\}$
    $\Phi'' \leftarrow \Phi - \Phi'$
    $\psi \leftarrow \prod_{\phi \in \Phi'} \phi$
    $\tau \leftarrow \sum_{Z} \psi$
    return $\Phi'' \cup \{\tau\}$

Two runs of Variable Elimination

- Elimination Ordering: *C,D,I,H,G,S,L*

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	$\tau_7(J)$

- Elimination Ordering: *G,I,S,L,H,C,D*

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

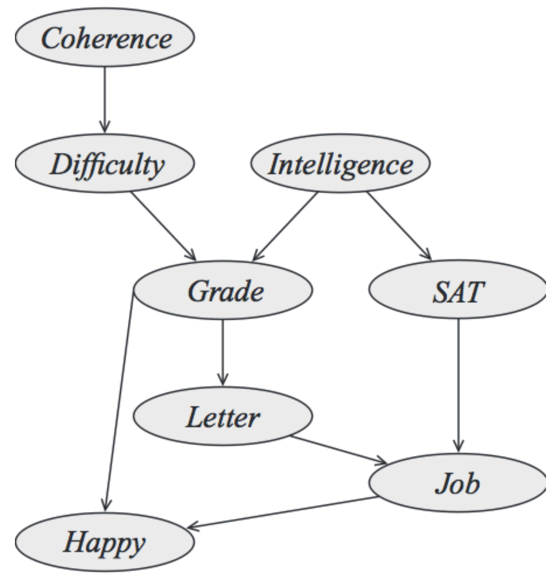
Factors with much larger scope

Dealing with Evidence

- We observe student is intelligent (i^1) and is unhappy (h^0)
- What is the probability that student has a job?

$$P(J \mid i^1, h^0) = \frac{P(J, i^1, h^0)}{P(i^1, h^0)}$$

  - For this we need unnormalized distribution $P(J,i^1,h^0)$. Then we compute conditional distribution by renormalizing by $P(e)=P(i^1,h^0)$



BN with evidence e is Gibbs with $Z=P(e)$

Defined by original factors reduced to context $E=e$

- B is a BN over χ and $E=e$ an observation. Let $W=\chi-E$.
  - Then $P_B(W|e)$ is a Gibbs distribution with factors

    $\Phi=\{\phi_{Xi}\}$ $X_i \varepsilon \chi$ where $\phi_{Xi}=P_B(X_i|Pa_{Xi})[E=e]$

    - Partition function for Gibbs distribution is $P(e)$. Proof follows:

$$P_B(\chi)=\prod_{i=1}^{N} P_B\left(X_i \mid Pa_{X_i}\right)$$

$$P_B(W \mid E=e)=\frac{P_B(W)\left[E=e\right]}{P_B(E=e)}=\frac{\prod_{i=1}^{N} P_B\left(X_i \mid Pa_{X_i}\right)\left[E=e\right]}{\sum_W P_B(\chi)\left[E=e\right]}=\frac{\prod_{i=1}^{N} P_B\left(X_i \mid Pa_{X_i}\right)\left[E=e\right]}{\sum_W \prod_{i=1}^{N} P_B\left(X_i \mid Pa_{X_i}\right)\left[E=e\right]}$$

- Thus any BN conditioned on evidence can be regarded as a Markov network
  - and use techniques developed for MN analysis

Sum-Product for Conditional Probabilities

- Apply Sum-product VE to χ-Y-E
- Returned factor ϕ^* is $P(Y,e)$
- Renormalize by $P(e)$, sum over entries in unormalized distribution

Procedure Cond-Prob-VE (
  $\mathcal{K}$, // A network over $\mathcal{X}$
  $\boldsymbol{Y}$, // Set of query variables
  $\boldsymbol{E} = \boldsymbol{e}$ // Evidence
)
  $\Phi \leftarrow$ Factors parameterizing $\mathcal{K}$
  Replace each $\phi \in \Phi$ by $\phi[\boldsymbol{E} = \boldsymbol{e}]$
  Select an elimination ordering $\prec$
  $\boldsymbol{Z} \leftarrow\ = \mathcal{X} - \boldsymbol{Y} - \boldsymbol{E}$
  $\phi^* \leftarrow$ Sum-Product-VE($\Phi, \prec, \boldsymbol{Z}$)
  $\alpha \leftarrow \sum_{\boldsymbol{y} \in Val(\boldsymbol{Y})} \phi^*(\boldsymbol{y})$
  return α, ϕ^*

Run of Sum-Product VE

- Computing

$P(J,i^1,h^0)$

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	$\tau'_7(J)$

Compare with previous elimination ordering:

- Steps *3,4* disappear
- Since I and H need not be eliminated
- By not eliminating I we avoid step that correlates G and I

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	$\tau_7(J)$

Complexity of VE: Simple Analysis

- If n random variables and m initial factors:
  - We have $m=n$ in a BN
  - In a MN we may have more factors than variables
- VE picks a variable X_i then multiplies all factors involving that variable
  - Result is a single factor ψ_i
- If N_i is no. of factors in ψ_i and $N_{max}=max\ N_i$
- Overall amount of work required is $O(mN_{max})$
- Inevitable exponential blowup is exponential size of factors ψ_i

Complexity: Graph-Theoretic Analysis

- VE can be viewed as operating on an undirected graph with factors Φ
- If P is distribution defined by multiplying factors in Φ
  - Defining $\boldsymbol{X} = Scope[\Phi]$

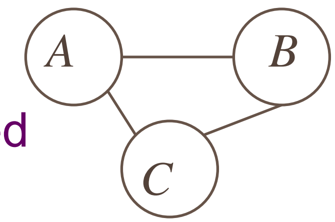
$$P(\boldsymbol{X}) = \frac{1}{Z}\prod_{\phi\in\Phi}\phi \quad \text{where } Z = \sum_{\boldsymbol{X}}\prod_{\phi\in\Phi}\phi$$

Then the directed graph defined by VE algorithm is precisely the Moralized BN

Factor Reduction: Reduced Gibbs

- Factor $\psi(A,B,C)$
- Context $C=c^1$

Moralized BN

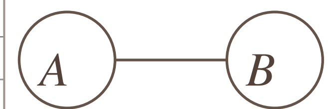


a^1	b^1	c^1	$0.5 \cdot 0.5 = 0.25$
a^1	b^1	c^2	$0.5 \cdot 0.7 = 0.35$
a^1	b^2	c^1	$0.8 \cdot 0.1 = 0.08$
a^1	b^2	c^2	$0.8 \cdot 0.2 = 0.16$
a^2	b^1	c^1	$0.1 \cdot 0.5 = 0.05$
a^2	b^1	c^2	$0.1 \cdot 0.7 = 0.07$
a^2	b^2	c^1	$0 \cdot 0.1 = 0$
a^2	b^2	c^2	$0 \cdot 0.2 = 0$
a^3	b^1	c^1	$0.3 \cdot 0.5 = 0.15$
a^3	b^1	c^2	$0.3 \cdot 0.7 = 0.21$
a^3	b^2	c^1	$0.9 \cdot 0.1 = 0.09$
a^3	b^2	c^2	$0.9 \cdot 0.2 = 0.18$

$C=c^1$

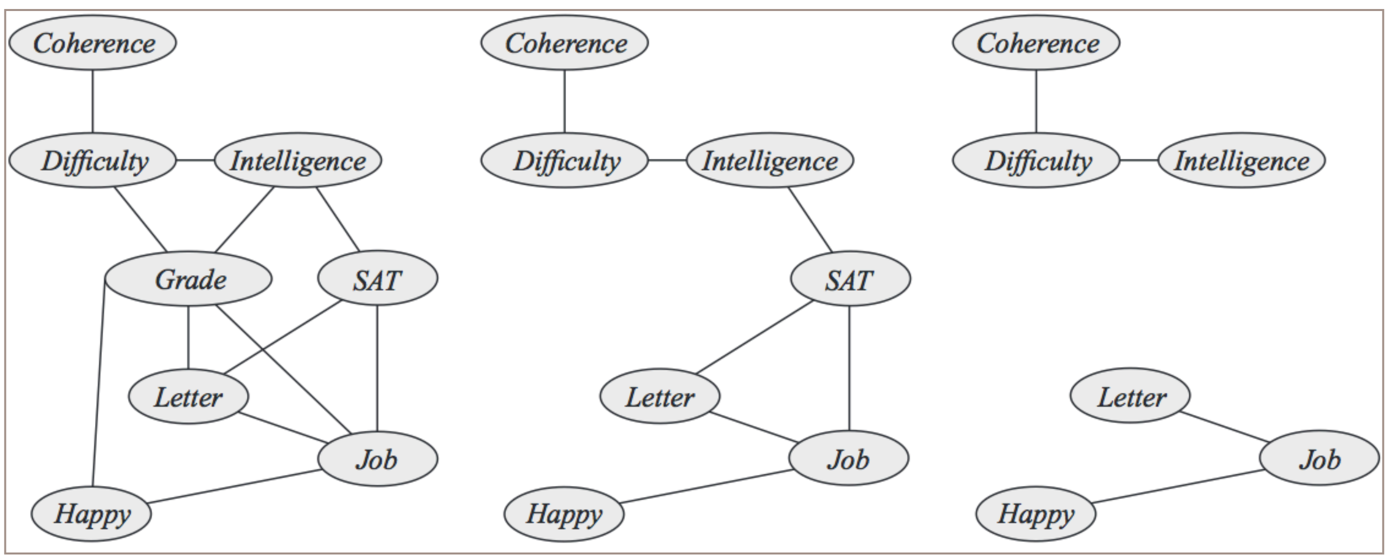
Value of C determines the factor $\tau(A,B)$

a^1	b^1	c^1	0.25
a^1	b^2	c^1	0.08
a^2	b^1	c^1	0.05
a^2	b^2	c^1	0
a^3	b^1	c^1	0.15
a^3	b^2	c^1	0.09



$$\tau(A,B) = \Sigma_{C=c^1} \psi(A,B,C)$$

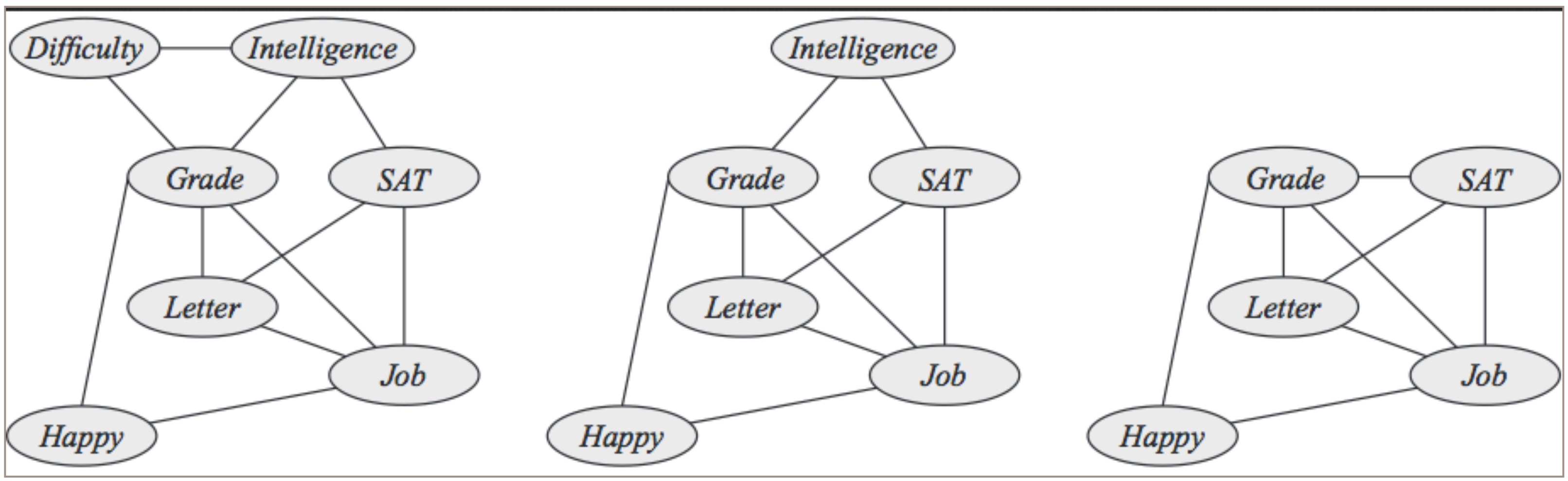
Initial Set of Factors | Context $G=g$ | Context $G=g$, $S=s$



VE as graph transformation

When a variable X is eliminated from Φ,

Fill edges are introduced in Φ_X



After eliminating C

After eliminating D
No fill edges

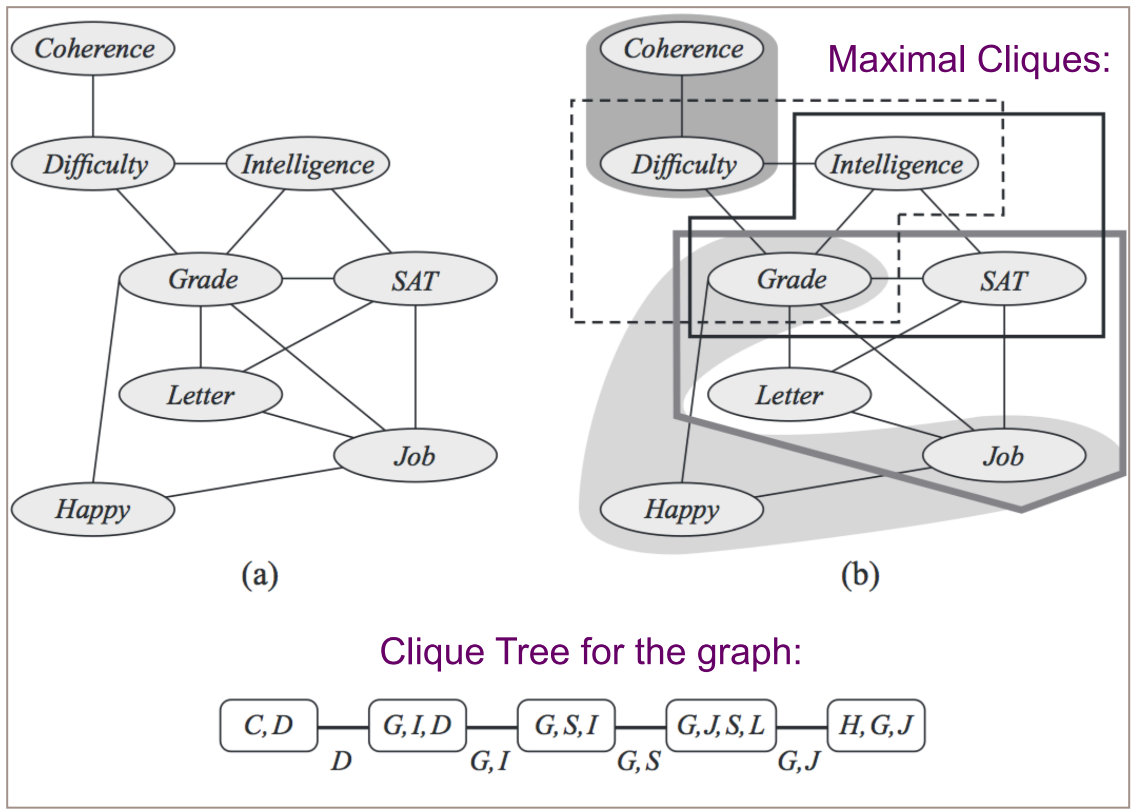
After eliminating I
Fill edge G-S

Induced Graph

- Union of all graphs generated by VE
- Every factor generated is a clique
- Every maximal clique is the scope of some intermediate factor

Induced Graph due to VE using elimination order:

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	$\tau_7(J)$

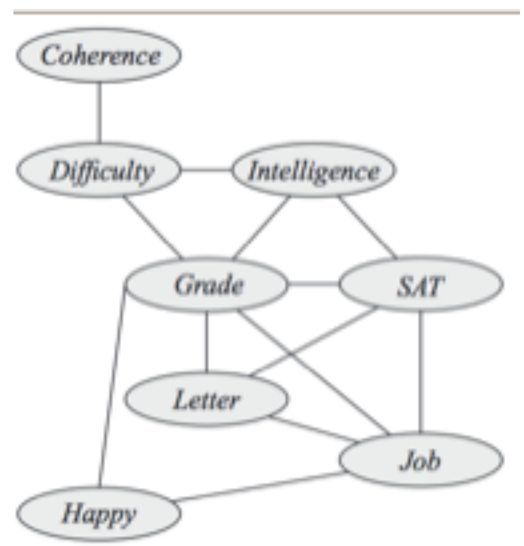


Width of induced graph=
no. of nodes in largest clique minus 1

Minimal induced width over all orderings is bound on VE performance

Finding Elimination Orderings

- Max-cardinality Search
  - Induced graphs are chordal
    - Every minimal loop is of length 3
      - $G \rightarrow L \rightarrow J \rightarrow H$ is cut by chord $G \rightarrow J$
- Greedy Search



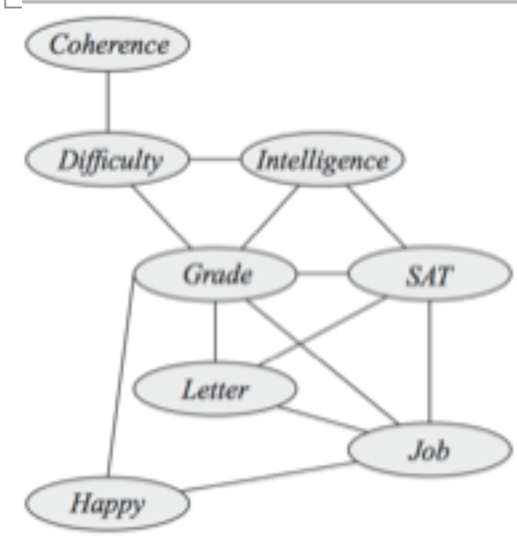
Max-Cardinality Search

- Procedure Max-Cardinality (

  H // An undirected graph over χ

)

```
Initialize all nodes in 𝒳 as unmarked
for k = |𝒳| ... 1
   X ← unmarked variable in 𝒳 with largest number of marked neighbors
   π(X) ← k
   Mark X
return π
```



Select S first
Next is a neighbor, say J
Largest no of marked neighbors are H and I

Greedy Search

- Procedure Greedy- Ordering(

  H // An undirected graph over χ

  s // An evaluation metric

)

Initialize all nodes in $\mathcal{X}$ as unmarked
for $k = 1 \ldots |\mathcal{X}|$
  Select an unmarked variable $X \in \mathcal{X}$ that minimizes $s(\mathcal{H}, X)$
  $\pi(X) \leftarrow k$
  Introduce edges in $\mathcal{H}$ between all neighbors of X
  Mark X
return π

Evaluation metric $s(H,X)$:

- Min-neighbors
- Min-weight
- Min-fill
- Weighted min-fill

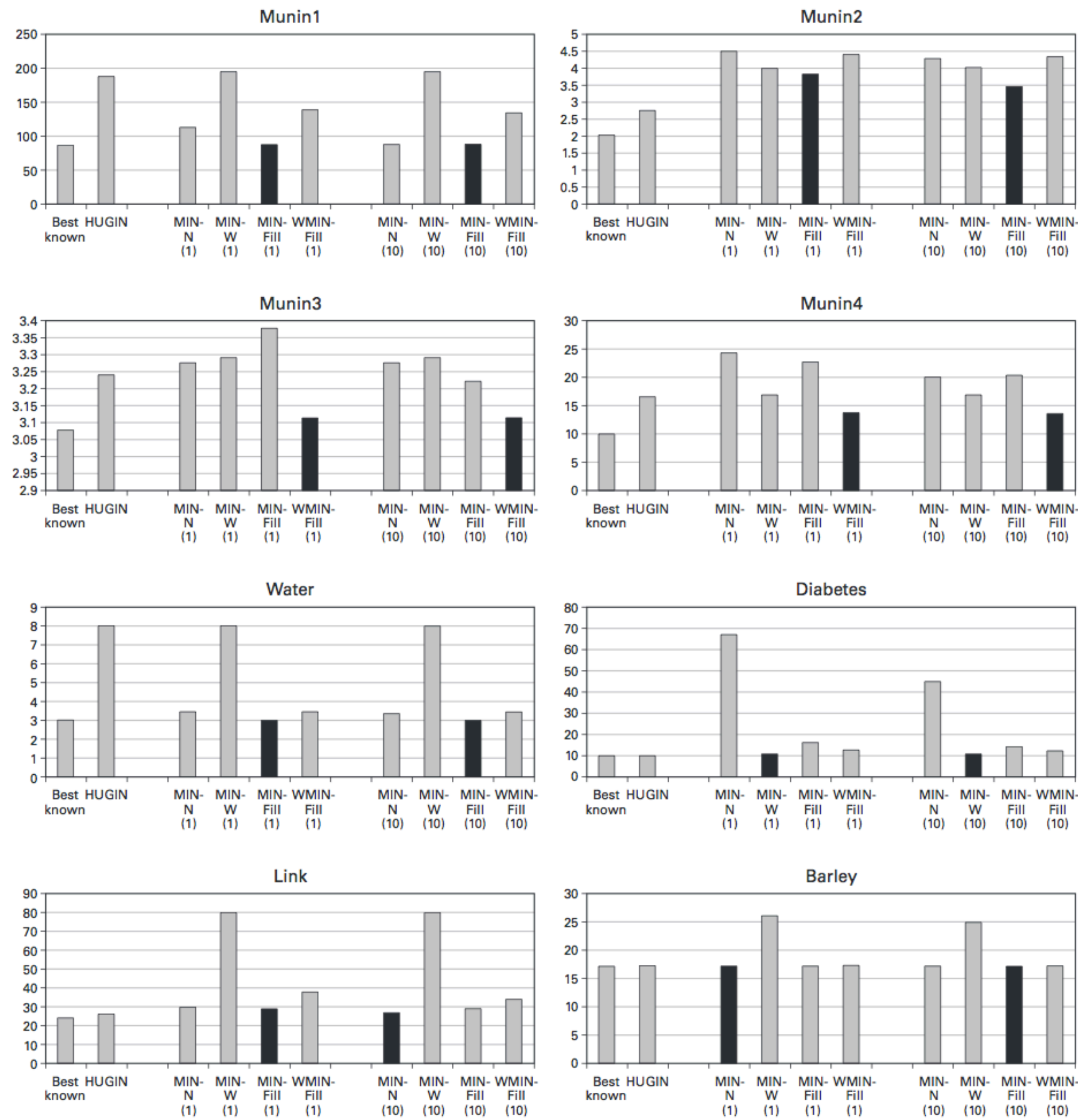
Comparison of VE Orderings

- Different heuristics for variable orderings
- Testing data:
  - 8 standard BNs ranging from 8 to 1,000 nodes
- Methods:
  - Simulated annealing, BN package
  - Four heuristics

Comparison of VE variable ordering algorithms

- Evaluation metric $s(H,X)$:
- Min-neighbors
- Min-weight
- Min-fill
- Weighted min-fill

- For large networks worthwhile to run several heuristic algorithms to find best ordering

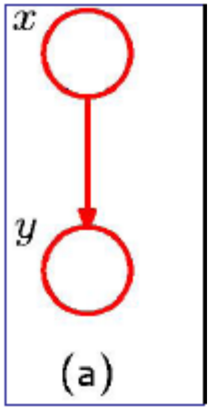


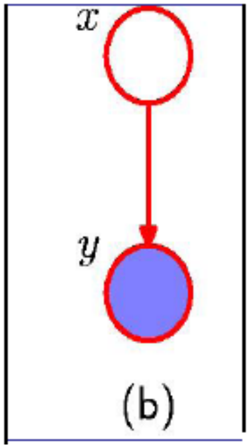
Two Simple Inference Cases

1. Bayes theorem as inference
2. Inference on a chain

1. Bayes Theorem as Inference

- Joint distribution $p(x,y)$ *over two variables* x *and* y
  - Factors $p(x,y)=p(x)p(y|x)$
    - represented as directed graph (a)
    - We are given CPDs $p(x)$ *and* $p(y|x)$
- If we observe value of y as in (b)
  - Can view marginal $p(x)$ *as prior*
  - Over latent variable x
- Analogy to 2-class classifier
  - Class $x \in \{0,1\}$ and feature y is continuous
  - Wish to infer *a posteriori distribution* $p(x|y)$





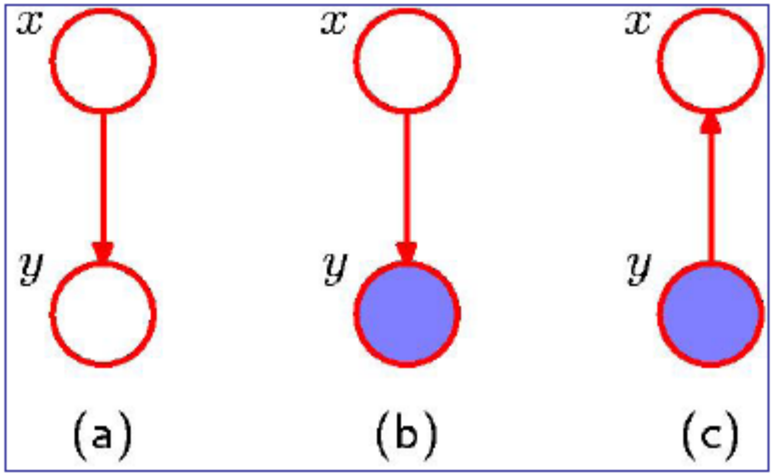
Inferring posterior using Bayes

- Using sum and product rules, we can evaluate marginal $p(y) = \sum_{x'} p(y \mid x') p(x')$
  - Need to evaluate a summation
- Which is then used in Bayes rule to calculate

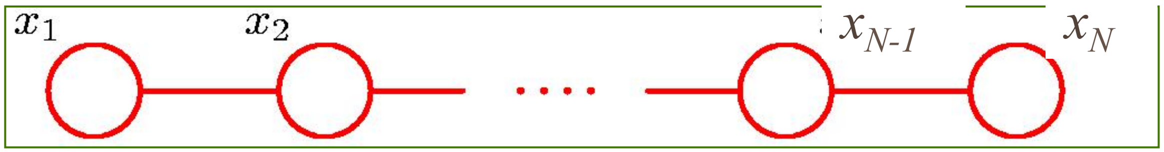
$$p(x \mid y) = \frac{p(y \mid x) p(x)}{p(y)}$$

- Observations
  - Joint is now expressed as

    $p(x,y)=p(y)p(x|y)$
    - Which is shown in (c)
  - Thus knowing value of y we know distribution of x



2. Inference on a Chain



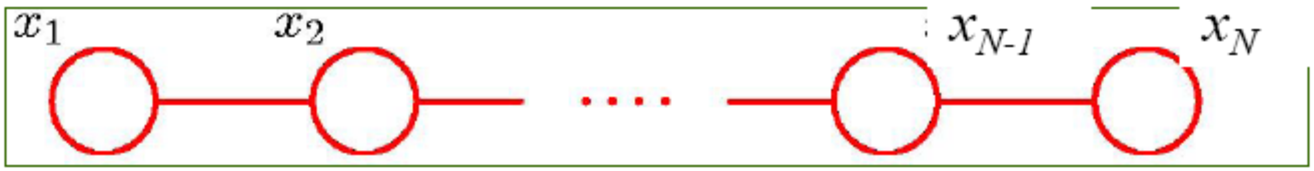
- Graphs of this form are known as Markov chains
  - Example: $N = 365$ days and x is weather (cloudy,rainy,snow..)
- Analysis more complex than previous case
- In this case directed and undirected are exactly same since there is only one parent per node (no additional links needed)
- Joint distribution has form

$$p(\mathrm{x}) = \frac{1}{Z}\psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)\ldots\psi_{N-1,N}(x_{N-1},x_N)$$

Product of potential functions over pairwise cliques

- Specific case of N discrete variables
  - Potential functions are K x K tables
  - Joint distribution has $(n\text{-}1)K^2$ parameters

Inferring marginal of a node

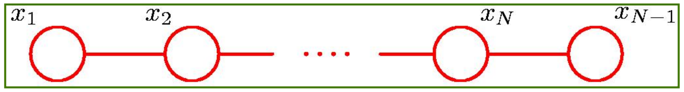


- Wish to evaluate marginal distribution $p(x_n)$
  - What is the weather on November 11?
- For specific node x_n part way along chain
- As yet there are no observed nodes
- Required marginal obtained summing joint distribution over all variables except x_n

$$p(x_n) = \sum_{x_1} .. \sum_{x_{n-1}} \sum_{x_{n+1}} .. \sum_{x_N} p(\mathrm{x})$$

By application of sum rule

Naïve Evaluation of marginal



$$p(x_n) = \sum_{x_1} .. \sum_{x_{n-1}} \sum_{x_{n+1}} .. \sum_{x_N} p(\mathrm{x})$$

$$= \sum_{x_1} .. \sum_{x_{n-1}} \sum_{x_{n+1}} .. \sum_{x_N} \underbrace{\frac{1}{Z} \psi_{1,2}(x_1, x_2) \psi_{2,3}(x_2, x_3) \ldots \psi_{N-1,N}(x_{N-1}, x_N)}_{\text{Joint}}$$

1. Evaluate joint distribution
2. Perform summations explicitly

- Joint can be expressed as set of numbers one for each value of x
- There are N variables with K states
  - K^N values for x
- Evaluation of both joint and marginal
  - Exponential with length N of chain
  - Impossible with $K=10$ and $N=365$

Efficient Evaluation

$$p(x_n) = \sum_{x_1} .. \sum_{x_{n-1}} \sum_{x_{n+1}} .. \sum_{x_N} \frac{1}{Z} \psi_{1,2}(x_1, x_2) \psi_{2,3}(x_2, x_3) ... \psi_{N-1,N}(x_{N-1}, x_N)$$

- We are adding a bunch of products
- But multiplication is distributive over addition

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

  - Perform summation first and then do product
  - LHS involves 3 arithmetic ops,
  - RHS involves 2
- Sum-of-products evaluated as sums first

Efficient evaluation: exploiting conditional independence properties

$$p(x_n)=\sum_{x_1}..\sum_{x_{n-1}}\sum_{x_{n+1}}..\sum_{x_N}\frac{1}{Z}\psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)...\psi_{N-1,N}(x_{N-1},x_N)$$

- Rearrange order of summations/multiplications
  - to allow marginal to be evaluated more efficiently
- Consider summation over x_N
  - Potential $\psi_{N-1,N}(x_{N-1},x_N)$ is only one that depends on x_N
  - So we can perform $\sum_{x_N}\psi_{N-1,N}(x_{N-1},x_N)$
  - To give a function of x_{N-1}
- Use this to perform summation over x_{N-1}
- Each summation removes a variable from distribution or removal of node from graph

Marginal Expression

- Group potentials and summations together to give marginal

$$p(x_n) = \frac{1}{Z}$$

$$\underbrace{\left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \ldots \left[\sum_{x_2} \psi_{2,3}(x_2, x_3) \left[\sum_{x_1} \psi_{1,2}(x_1, x_2)\right]\right]..\right]}_{\mu_\alpha(x_n)}$$

$$\underbrace{\left[\sum_{x_{n-1}} \psi_{n,n+1}(x_n, x_{n+1}) \ldots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)\right]..\right]}_{\mu_\beta(x_n)}$$

Key concept:
Multiplication is distributive over addition
ab+ac=a(b+c)
LHS involves 3 arithmetic ops, RHS involves 2

Computational cost

- Evaluation of marginal using reordered expression
- N-1 summations
  - Each with K states
  - Each a function of 2 variables
  - Summation over x_1 involves only $\psi_{1,2}(x_1,x_2)$
    - A table of K x K numbers
  - Sum table over x_1 for each x_2
  - $O(K^2)$ cost
- Total cost is $O(NK^2)$
- Linear in chain length *vs.* exponential cost of naïve approach
  - Able to exploit many conditional independence properties of simple graph

Interpretation as Message Passing

- Calculation viewed as message passing in graph
- Expression for marginal decomposes into

$$p(x_n) = \frac{1}{Z}\mu_\alpha(x_n)\mu_\beta(x_n)$$

- Interpretation
  - Message passed forwards along chain from node x_{n-1} to x_n is $\mu_\alpha(x_n)$
  - Message passed backwards from node x_{n+1} to x_n is $\mu_\beta(x_n)$
  - Each message comprises of K values one for each choice of x_n

Recursive evaluation of messages

- Message $\mu_\alpha(x_n)$ can be evaluated as

$$\mu_\alpha(x_\mathrm{n}) = \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \left[\sum_{x_{n-2}} \cdots \right]$$

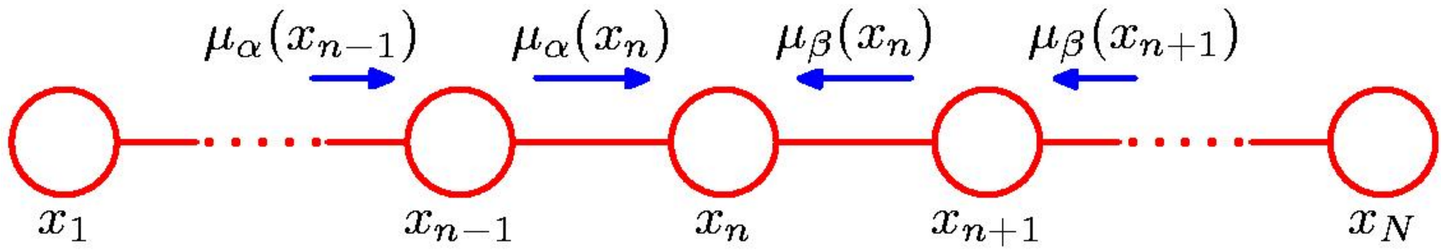
$$= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \mu_\alpha(x_{n-1}) \qquad (1)$$

- Therefore first evaluate

$$\mu_\alpha(x_2) = \sum_{x_1} \psi_{1,2}(x_1, x_2)$$

- Apply (1) repeatedly until we reach desired node
- Note that outgoing message $\mu_\alpha(x_n)$ in (1) is obtained by
  - multiplying incoming message $\mu_\alpha(x_{n-1})$ by the local potential involving the node variable and
  - the outgoing variable
  - and summing over node variable

Recursive message passing



- Similarly message $\mu_b(x_n)$ can be evaluated recursively starting with node x_n

$$\mu_\beta(x_n) = \sum_{x_{n+1}} \psi_{n+1,n}(x_{n+1}, x_n) \left[\sum_{x_{n+2}} \cdots \right]$$

$$= \sum_{x_{n+1}} \psi_{n+1,n}(x_{n+1}, x_n) \mu_\beta(x_{n+1})$$

Message passing equations known as *Chapman-Kolmogorov* equations for Markov processes

- Normalization constant Z is easily evaluated
  - By summing $\frac{1}{Z}\mu_\alpha(x_n)\mu_\beta(x_n)$ over all state of x_n
  - An $O(K)$ computation

Evaluating marginals for every node

- Evaluate $p(x_n)$ for every node $n = 1,..N$
- Simply applying above procedure is $O(N^2M^2)$
- Computationally wasteful with duplication
  - To find $p(x_1)$ we need to propagate message $m_b(.)$ from node x_N back to x_2
  - To evaluate $p(x_2)$ we need to propagate message $m_b(.)$ from node x_N back to x_3
- Instead
  - launch message $m_b(x_{N-1})$ starting from node x_N and propagate back to x_1
  - launch message $m_a(x_2)$ starting from node x_2 and propagate forward to x_N
  - Store all intermediate messages along the way
  - Then any node can evaluate its marginal by $p(x_n) = \frac{1}{Z}\mu_\alpha(x_n)\mu_\beta(x_n)$
  - Computational cost is only twice as finding marginal of single node instead of N times

Joint distribution of neighbors

- Wish to calculate joint distribution $p(x_{n-1},x_n)$ for neighboring nodes
- Similar to previous computation
- Required joint distribution can be written as

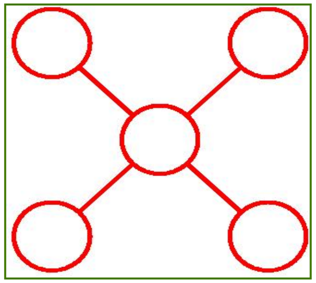
$$p(x_{n-1},x_n)=\frac{1}{Z}\mu_\alpha(x_{n-1})\psi_{n-1},n(x_{n-1},x_n)\mu_\beta(x_n)$$

- Obtained once message passing for marginals is completed
- Useful result if we wish to use parametric forms for conditional distributions

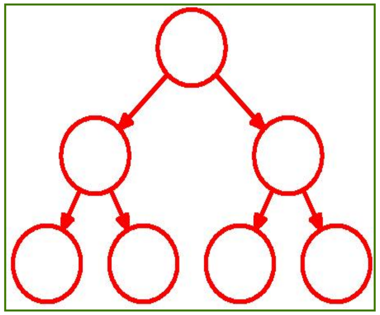
Tree structured graphs

- Local message passing can be performed efficiently on trees
- Message passing can be generalized to give *sum-product algorithm*
- Tree
  - a graph with only one path between any pair of nodes
  - Such graphs have no loops
  - In directed graphs a tree has a single node with no parents called a *root*
  - Directed to undirected will not add moralization links since every node has only one parent
- Polytree
  - A directed graph has nodes with more than one parent but there is only one path between nodes (ignoring arrow direction)
  - Moralization will add links

Undirected tree



Directed tree



Directed polytree