CSC384 Summerr 2018

Week 9-10 - Reasoning Under Uncertainty - Part III

Ilir Dema

University of Toronto

July 23-30, 2018

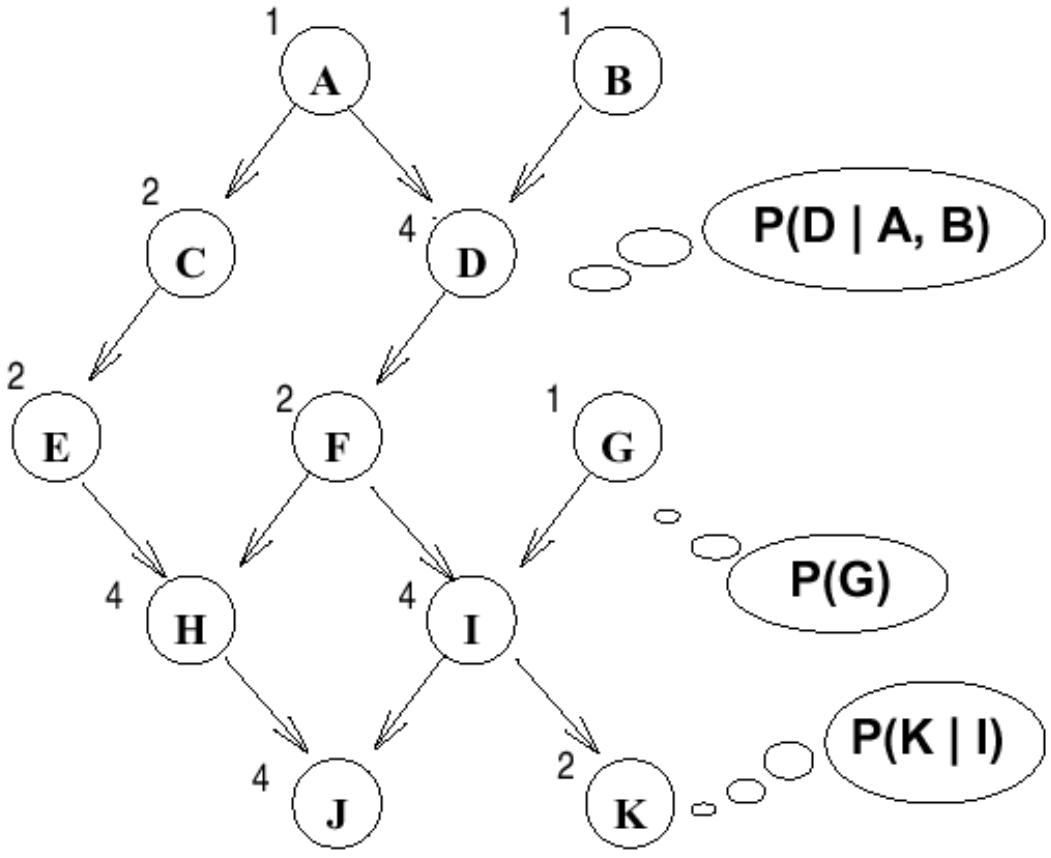
UNIVERSITY OF TORONTO MISSISSAUGA

Overview

Variable Elimination

- Variable elimination uses the product decomposition that defines the Bayes Net and the summing out rule to compute posterior probabilities from the information (CPTs) already in the network.

Example (Binary valued Variables)



$$P(A,B,C,D,E,F,G,H,I,J,K) =$$
$$P(A) \times P(B) \times P(C|A) \times P(D|A,B) \times P(E|C) \times P(F|D) \times P(G) \times$$
$$P(H|E,F) \times P(I|F,G) \times P(J|H,I) \times P(K|I)$$

Example (Binary valued Variables)

Say that $E = \{H = true, I = false\}$, and we want to know $P(D|h,i)$ (Notation: h means H is true, $-h$ means H is false)

1. Write as a sum for each value of D

$$\sum_{A,B,C,E,F,G,J,K} P(A,B,C,d,E,F,G,h,-i,J,K) = P(d,h,-i)$$

$$\sum_{A,B,C,E,F,G,J,K} P(A,B,C,-d,E,F,G,h,-i,J,K) = P(-d,h,-i)$$

2. $P(d,h,-i) + P(-d,h,-i) = P(h,-i)$
3. $P(d|h,-i) = P(d,h,-i)/P(h,-i)$
   $P(-d|h,-i) = P(-d,h,-i)/P(h,-i)$

Example (Binary valued Variables)

Say that $E = \{H = true, I = false\}$, and we want to know $P(D|h,i)$ (Notation: h means H is true, $-h$ means H is false)

1. Write as a sum for each value of D

$$\sum_{A,B,C,E,F,G,J,K} P(A,B,C,d,E,F,G,h,-i,J,K) = P(d,h,-i)$$

$$\sum_{A,B,C,E,F,G,J,K} P(A,B,C,-d,E,F,G,h,-i,J,K) = P(-d,h,-i)$$

2. $P(d,h,-i) + P(-d,h,-i) = P(h,-i)$
3. $P(d|h,-i) = P(d,h,-i)/P(h,-i)$
   $P(-d|h,-i) = P(-d,h,-i)/P(h,-i)$

Example (Binary valued Variables)

- So we are computing $P(d, h, -i)$ and $P(-d, h, -i)$ and then dividing by their sum:

  $[P(d, h, -i), P(-d, h, -i)]/(P(d, h, -i) + P(-d, h, -i))$
- This the same as normalizing the vector $[P(d, h, -i), P(-d, h, -i)]$.

Exercise

Compute $P(d, h, -1) = \sum_{A,B,C,E,F,G,J,K} P(A, B, C, d, E, F, G, h, -i, J, K)$

- Use Bayes net from slide 4
- Use properties of sums to rearrange summation
- Computer innermost sum $\sum_K P(K|-i) = P(k|-i) + P(-k|-i)$.
- Note this is a constant so can be factored.
- Conclude variable K has been eliminated.
- Similarly, eliminate J.

Exercise (continues)

- Now the sum must look like:

$$c_1 c_2 \sum_A P(A) \sum_B P(B) P(d|A,B) \sum_C P(C|A) \sum_E P(E|C)$$

$$\sum_F P(F|d) P(h|E,F) \sum_G P(G) P(-i|F,G)$$

- Next:

  $\sum_G P(G)P(-i|F,G) =$
  $P(g)P(-i|F,g) + P(-g)P(-i|F,-g) = f_1(F)$
  where f_1 is a function of variable F.
- Observe that after introducing the function $f_1(F)$ the summation on F yields a function of E which we denote f_2.
- That is variable F has been eliminated!

Exercise (conclusion)

- We can continue this way eliminating one variable after another. Until we sum out A to obtain a single number equal to $P(d, h, -i)$.
- A similar computation produces $P(-d, h, -i)$.
- Normalizing these two numbers gives us $P(d|h, i)$ and $P(-d|h, i)$.
- Or as we will see later, we can keep D as a variable, and compute a function of D, $f_k(D)$ whose two values $f_k(d)$ and $f_k(-d)$ are the values we want.

Variable Elimination (Dynamic Programming)

- This process is called variable elimination.
- By computing the intermediate functions $f_1(F)$, $f_2(E)$ etc. we are actually storing values that we can reuse many times during the computation.
- In this way variable elimination is a form of dynamic programming, where we save sub-computations to avoid re-computations.

Relevance

- Consider the term $\sum_K P(K|-i)$. The value of I has been fixed to $-i$ so at this point $K = k$ or $K = -k$ and there is no other possibility.
- Therefore $\sum_K P(K|-1) = 1$.
- Similarly, $\sum_K P(J|h,-i) = 1$.
- The we conclude variables K and J are not relevant to our query given the evidence h and $-i$.

Variable Elimination (VE)

- In general, at each stage VE will sum out the innermost variable, computing a new function over the variables in that sum.
- The function specifies one number for each different instantiation of its variables.
- We store these functions as a table with one entry per instantiation of the variables
- The size of these tables is exponential in the number of variables appearing in the sum, e.g.,

$$\sum_F P(F|D)P(h|E,F)f_1(F)$$

depends on the value of D and E, thus we will obtain $|Dom[D]| \times |Dom[E]|$ different numbers in the resulting table.

FACTORS

- We call these tables of values computed by VE factors.
- Note that the original probabilities that appear in the summation, e.g., $P(C|A)$, are also tables of values (one value for each instantiation of C and A).
- Each factor is a function of some variables, e.g., $P(C|A) = f(A, C)$: it maps each value of its arguments to a number.
  - A tabular representation is exponential in the number of variables in the factor.

Operations on Factors

- If we examine the inside-out summation process we see that various operations occur on factors.
- We can specify the algorithm for Variable Elimination by precisely specifying these operations.
- Notation: $f(\mathbf{X}, \mathbf{Y})$ denotes a factor over the variables $\mathbf{X} \cup \mathbf{Y}$ (where $\mathbf{X}$ and $\mathbf{Y}$ are sets of variables)

The Product of Two Factors

- Let $f(\mathbf{X},\mathbf{Y}), g(\mathbf{Y},\mathbf{Z})$ be two factors with variables $\mathbf{Y}$ in common
- The product of f and g, denoted $h = f * g$ (or sometimes just $h = fg$), is defined:

$$h(\mathbf{X},\mathbf{Y},\mathbf{Z}) = f(\mathbf{X},\mathbf{Y})g(\mathbf{Y},\mathbf{Z})$$

f(A,B)		g(B,C)		h(A,B,C)			
ab	0.9	bc	0.7	abc	0.63	ab~c	0.27
a~b	0.1	b~c	0.3	a~bc	0.08	a~b~c	0.02
~ab	0.4	~bc	0.8	~abc	0.28	~ab~c	0.12
~a~b	0.6	~b~c	0.2	~a~bc	0.48	~a~b~c	0.12

Summing a Variable Out of a Factor

- Let $f(X, \mathbf{Y})$ be a factor with variable X ($\mathbf{Y}$ is a set)
- We *sum out* variable X from f to produce a new factor $h = \sum_X f$, which is defined:

$$h(\mathbf{Y}) = \sum_{X \in Dom(X)} f(x, \mathbf{Y})$$

f(A,B)		h(B)	
ab	0.9	b	1.3
a~b	0.1	~b	0.7
~ab	0.4		
~a~b	0.6		

Restricting a Factor

- Let $f(X, \mathbf{Y})$ be a factor with variable X ($\mathbf{Y}$ is a set)
- We restrict factor f to $X = a$ by setting X to the value x and "deleting" incompatible elements of f's domain . Define $h = f_{X=a}$ as: $h(\mathbf{Y}) = f(a, \mathbf{Y})$.

f(A,B)		h(B) = $f_{A=a}$	
ab	0.9	b	0.9
a~b	0.1	~b	0.1
~ab	0.4		
~a~b	0.6		

Variable Elimination the Algorithm

Given query variable Q, evidence variables **E** (set of variables observed to have values **e**), remaining vars **Z**. Let F be original CPTs (conditional probability tables).

1. Replace each factor $f \in F$ that mentions a variable(s) in **E** with its restriction $f_{\mathbf{E}=\mathbf{e}}$ (this might yield a factor over no variables, a constant)
2. For each Z_j - in the order given - eliminate $Z_j \in \mathbf{Z}$ as follows:
   (A) Compute new factor $g_j = \sum_{Z_j} f_1 f_2 \dots f_k$, where the f_i are the factors in F that include Z_j.
   (B) Remove the factors f_i that mention Z_j from F and add new factor g_j to F.
3. The remaining factors refer only to the query variable Q. Take their product and normalize to produce $P(Q|E)$.

EXAMPLE

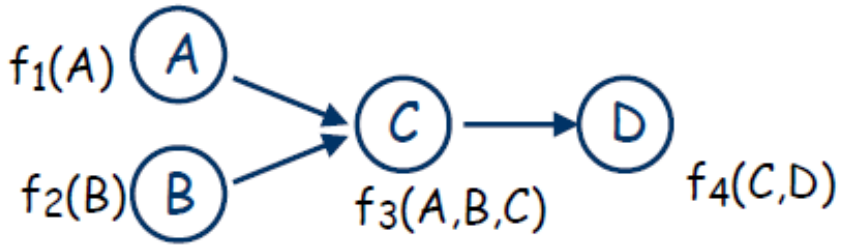
VE: Example

Factors: $f_1(A)$ $f_2(B)$ $f_3(A,B,C)$ $f_4(C,D)$

Query: P(A)?

Evidence: D = d

Elim. Order: C, B



Restriction: replace $f_4(C,D)$ with $f_5(C) = f_4(C,d)$

Step 1: Compute & Add $f_6(A,B) = \Sigma_C f_5(C) f_3(A,B,C)$

Remove: $f_3(A,B,C)$, $f_5(C)$

Step 2: Compute & Add $f_7(A) = \Sigma_B f_6(A,B) f_2(B)$

Remove: $f_6(A,B)$, $f_2(B)$

Last factors: $f_7(A)$, $f_1(A)$. The product $f_1(A) \times f_7(A)$ is (unnormalized) posterior. So...

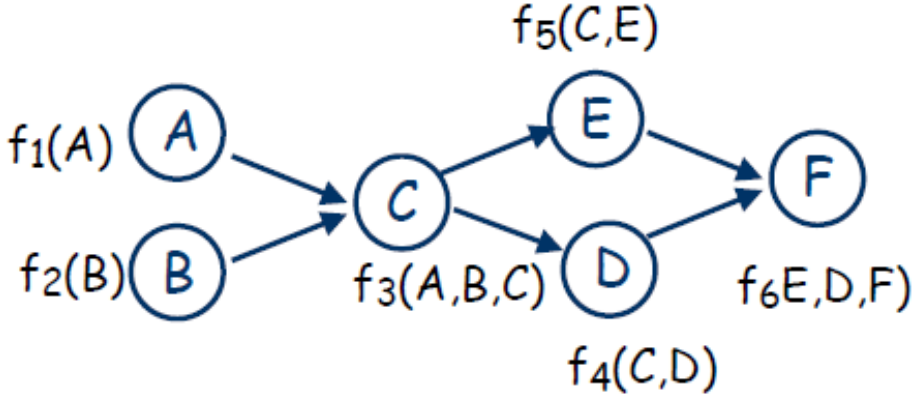
$P(A|d) = \alpha f_1(A) \times f_7(A)$

where $\alpha = 1/\Sigma_A f_1(A) f_7(A)$

TRACING VE

VE: Buckets as a Notational Device

Ordering:
C,F,A,B,E,D

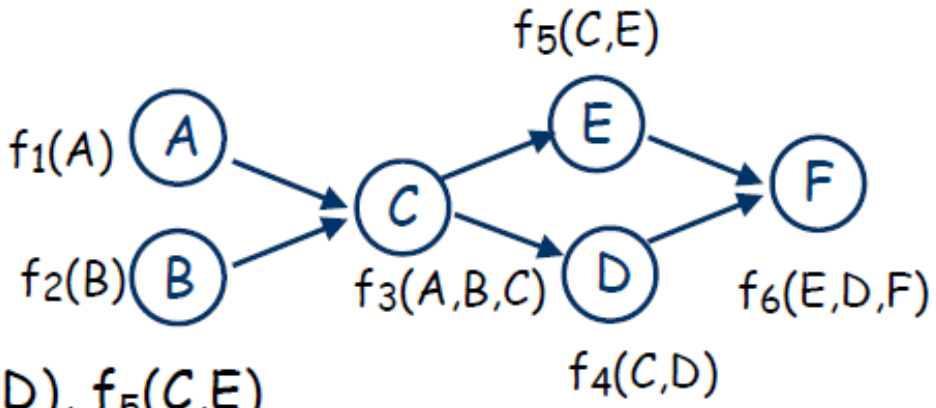


1. C:
2. F:
3. A:
4. B:
5. E:
6. D:

TRACING VE

VE: Buckets—Place Original Factors in first bucket that contains one of its variables

Ordering:
C,F,A,B,E,D

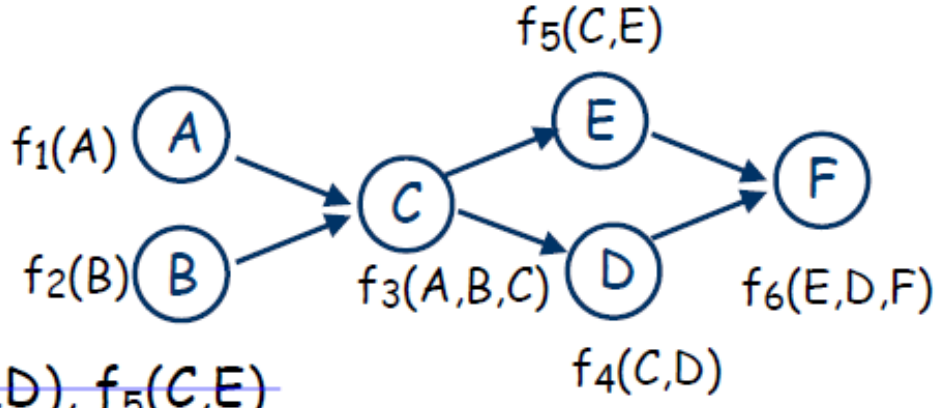


1. C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$
2. F: $f_6(E,D,F)$
3. A: $f_1(A)$
4. B: $f_2(B)$
5. E:
6. D:

TRACING VE

VE: Buckets—Place Original Factors in first bucket that contains one of its variables

Ordering:
C,F,A,B,E,D



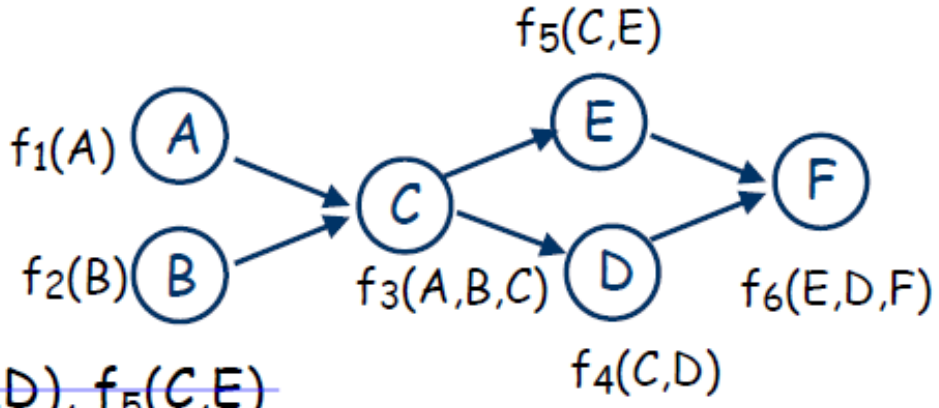
1. C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$
2. F: $f_6(E,D,F)$
3. A: $f_1(A)$, $f_7(A,B,D,E)$
4. B: $f_2(B)$
5. E:
6. D:

1. $\Sigma_C f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$
   $= f_7(A,B,D,E)$

TRACING VE

VE: Buckets—Place Original Factors in first bucket that contains one of its variables

Ordering:
C,F,A,B,E,D



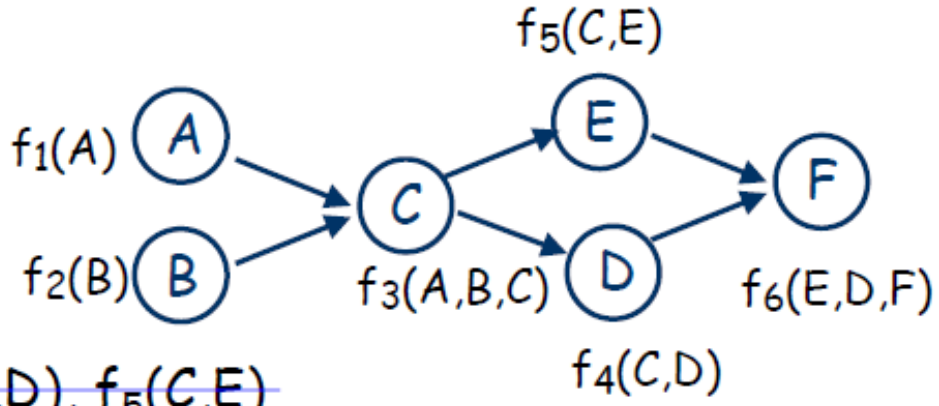
1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. A: $f_1(A)$, $f_7(A,B,D,E)$
4. B: $f_2(B)$
5. E: $f_8(E,D)$
6. D:

2. $\Sigma_F f_6(E,D,F) = f_8(E,D)$

TRACING VE

VE: Buckets—Place Original Factors in first bucket that contains one of its variables

Ordering:
C,F,A,B,E,D



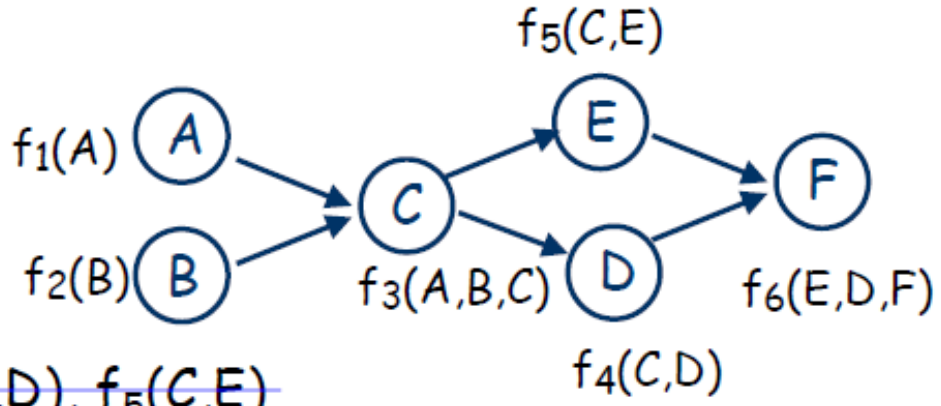
1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. ~~A: $f_1(A)$, $f_7(A,B,D,E)$~~
4. B: $f_2(B)$, $f_9(B,D,E)$
5. E: $f_8(E,D)$
6. D:

3. $\sum_A f_1(A), f_7(A,B,D,E) = f_9(B,D,E)$

TRACING VE

VE: Buckets—Place Original Factors in first bucket that contains one of its variables

Ordering:
C,F,A,B,E,D



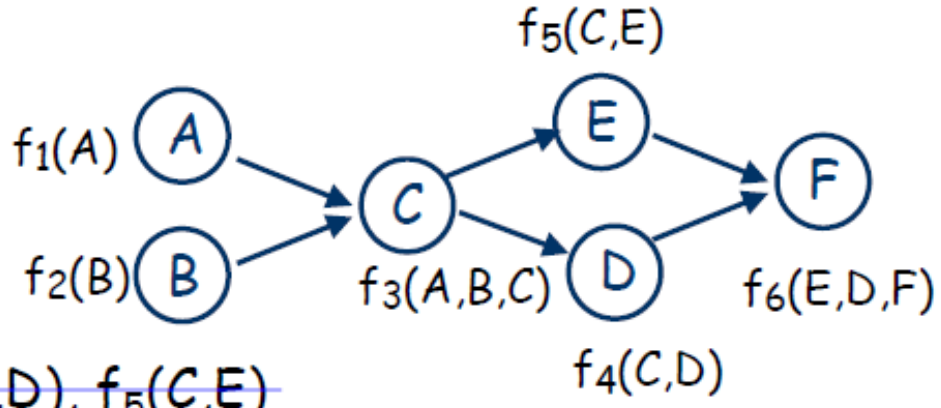
1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. ~~A: $f_1(A)$, $f_7(A,B,D,E)$~~
4. ~~B: $f_2(B)$, $f_9(B,D,E)$~~
5. E: $f_8(E,D)$, $f_{10}(D,E)$
6. D:

4. $\Sigma_B f_2(B), f_9(B,D,E) = f_{10}(D,E)$

TRACING VE

VE: Buckets—Place Original Factors in first bucket that contains one of its variables

Ordering:
C,F,A,B,E,D



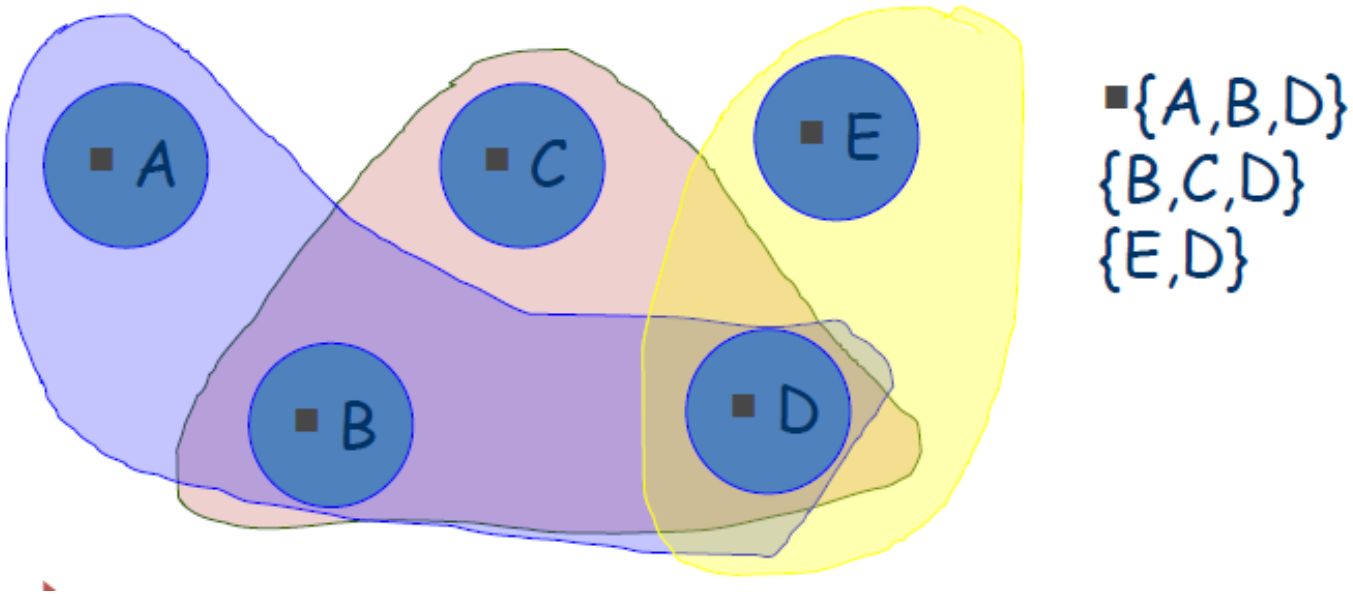
1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. ~~A: $f_1(A)$, $f_7(A,B,D,E)$~~
4. ~~B: $f_2(B)$, $f_9(B,D,E)$~~
5. ~~E: $f_8(E,D)$, $f_{10}(D,E)$~~
6. D: $f_{11}(D)$

5. Σ_E $f_8(E,D)$, $f_{10}(D,E)$
   = $f_{11}(D)$

f_{11} is he final answer, once we normalize it.

Complexity of variable elimination

- Lets define Hypergraph of Bayes Net.
- Hypergraph has vertices just like an ordinary graph, but instead of edges between two vertices it contains hyperedges.
- A hyperedge is a set of vertices (i.e., potentially more than one)

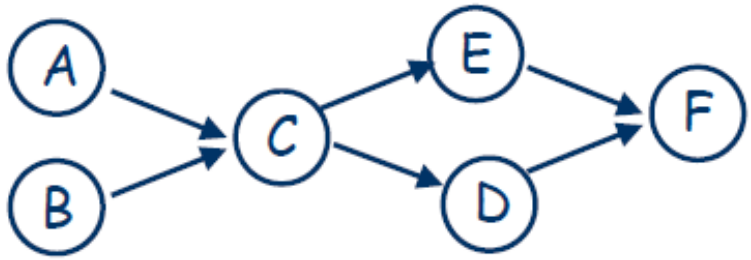


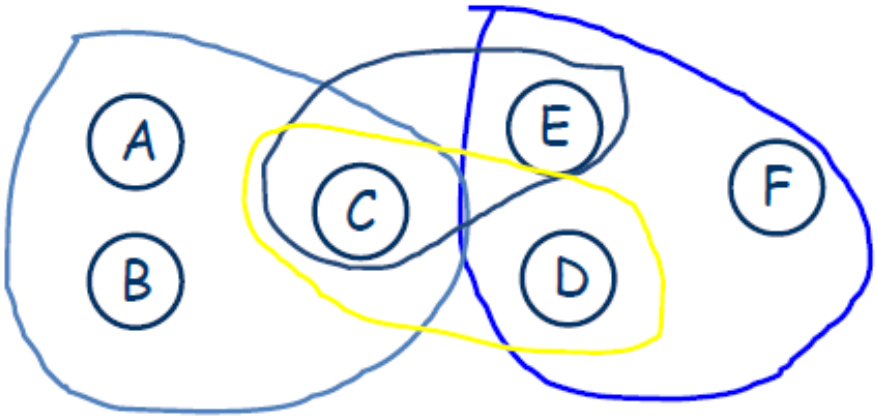
Hypergraph of Bayes Net.

- The set of vertices are precisely the nodes of the Bayes net.
- The hyperedges are the variables appearing in each CPT. $\{X_i\} \cup Par(X_i)$

Complexity of variable elimination

$$P(A,B,C,D,E,F) = P(A)Pr(B)P(C|A,B)P(E|C)P(D|C)P(F|E,D).$$





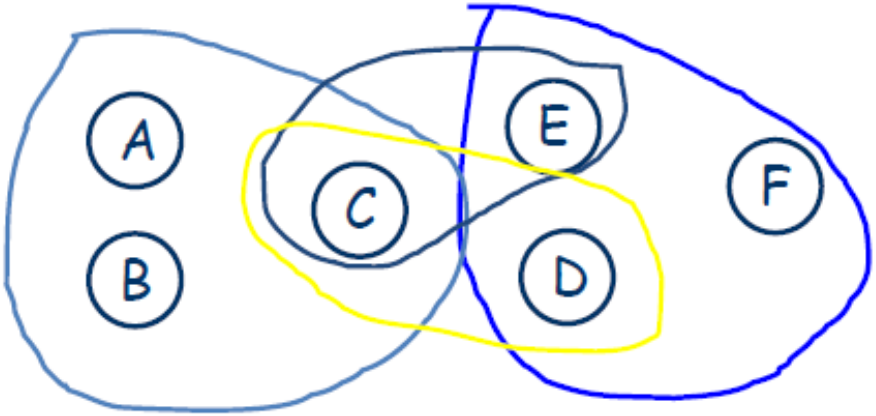
Variable Elimination in the HyperGraph

To eliminate variable X_i in the hypergraph we

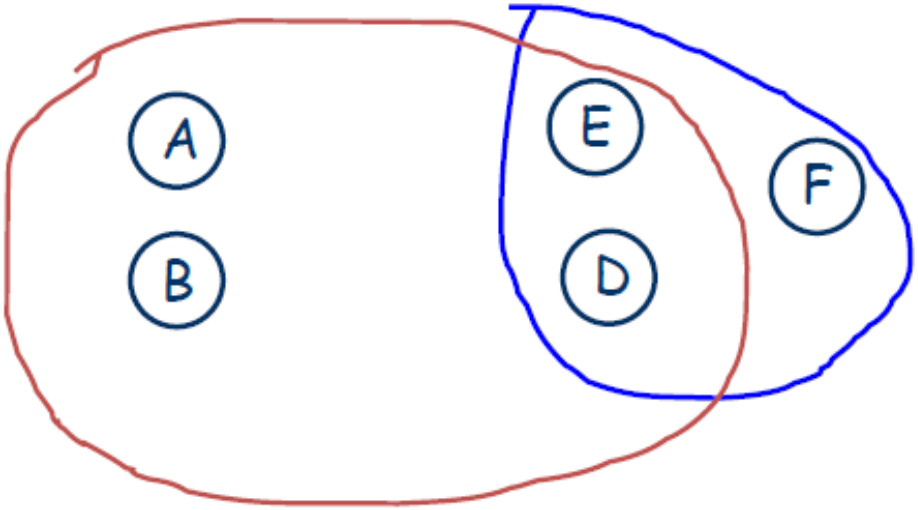
- remove the vertex Xi
- Create a new hyperedge H_i equal to the union of all of the hyperedges that contain X_i minus X_i
- Remove all of the hyperedges containing X_i from the hypergraph.
- Add the new hyperedge H_i to the hypergraph.

COMPLEXITY OF VARIABLE ELIMINATION

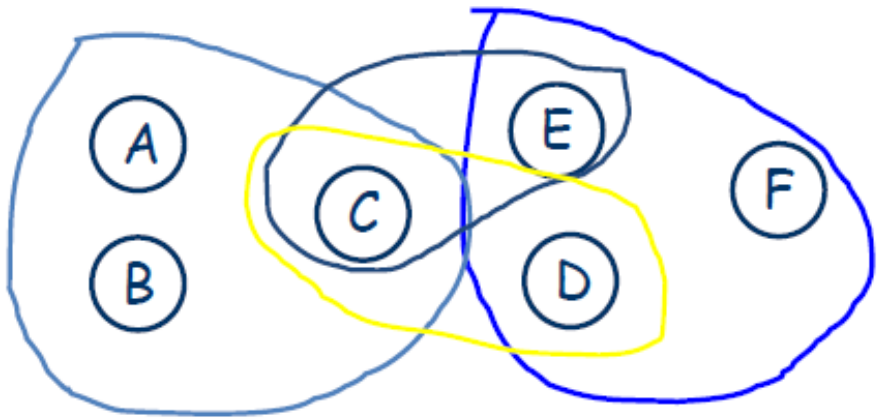
Complexity of Variable Elimination



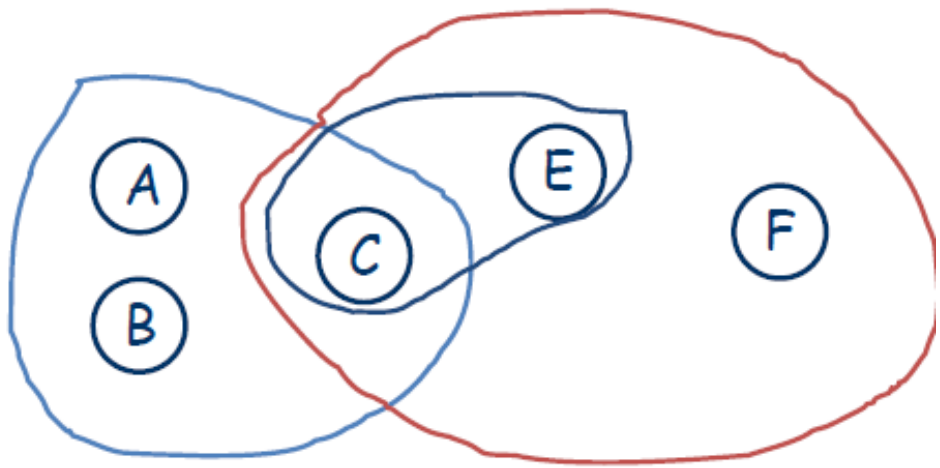
- Eliminate C



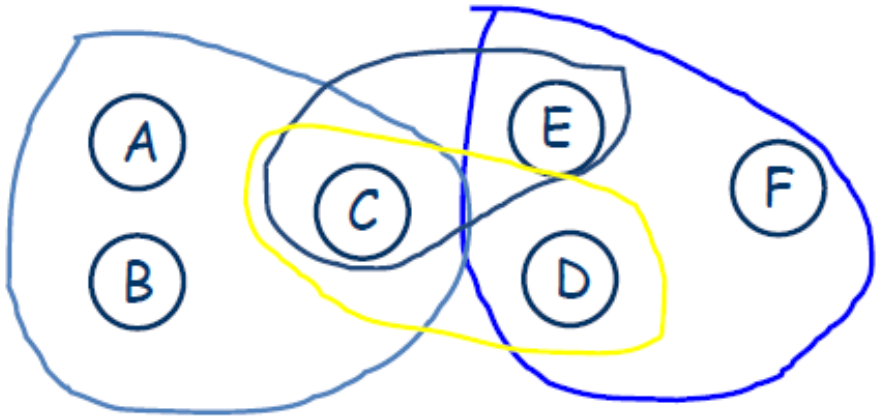
Complexity of variable elimination



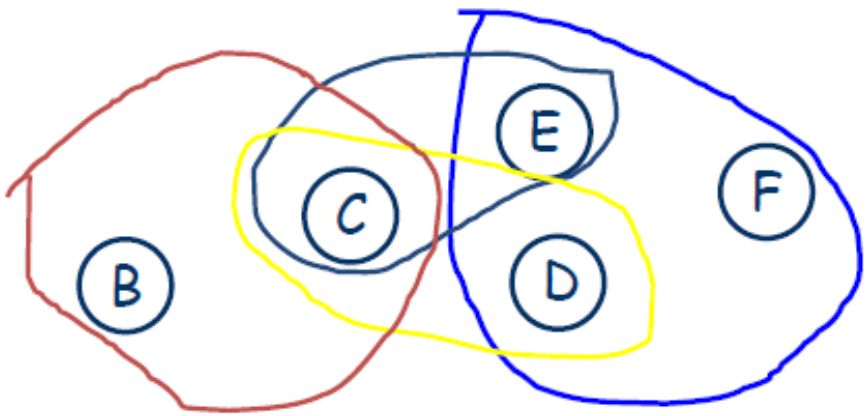
- Eliminate D



Complexity of variable elimination



- Eliminate A



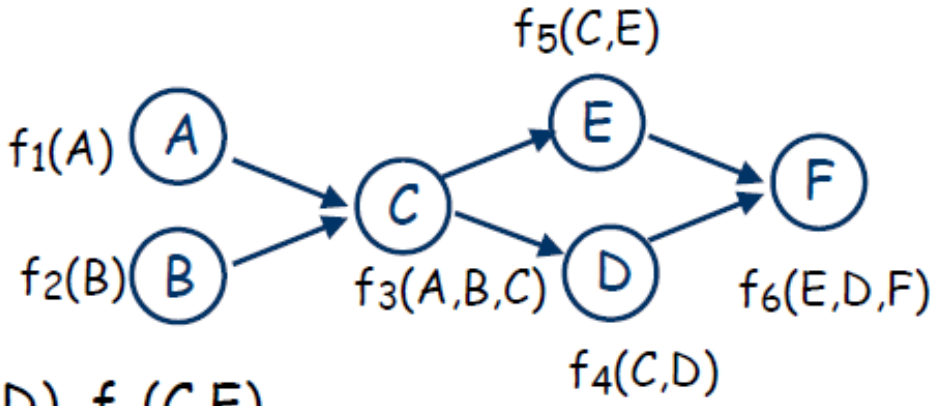
Variable Elimination

- Notice that when we start VE we have a set of factors consisting of the reduced CPTs. The unassigned variables for the vertices and the set of variables each factor depends on forms the hyperedges of a hypergraph H_1.
- If the first variable we eliminate is X, then we remove all factors containing X (all hyperedges) and add a new factor that has as variables the union of the variables in the factors containing X (we add a hyperdege that is the union of the removed hyperedges minus X).

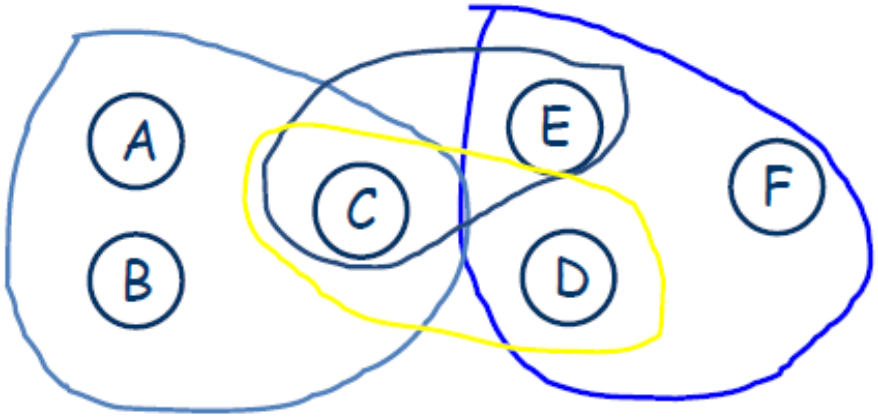
EXAMPLE

VE: Place Original Factors in first applicable bucket.

Ordering:
C,F,A,B,E,D



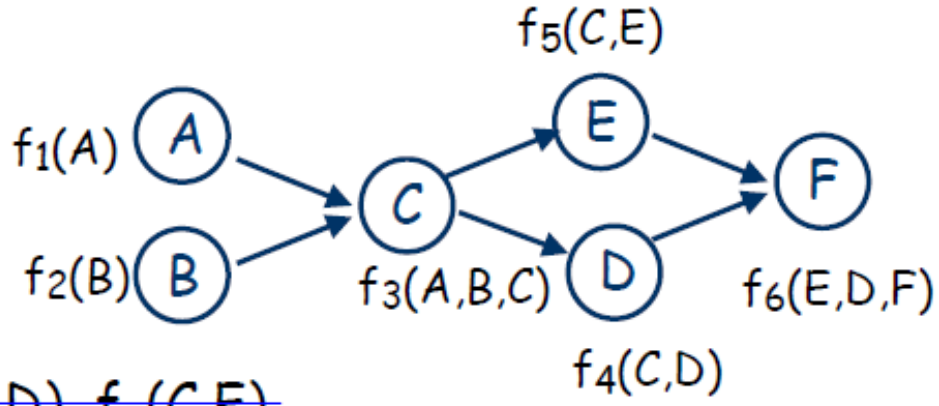
1. C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$
2. F: $f_6(E,D,F)$
3. A: $f_1(A)$
4. B: $f_2(B)$
5. E:
6. D:



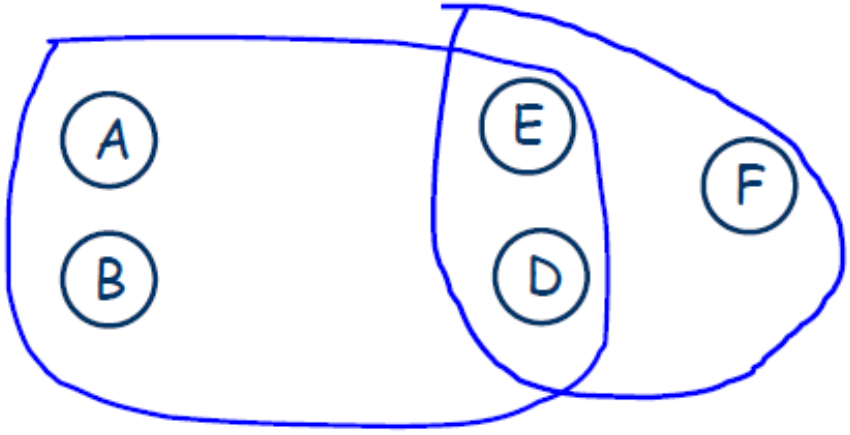
EXAMPLE

VE: Eliminate C, placing new factor f_7 in first applicable bucket.

Ordering:
C,F,A,B,E,D



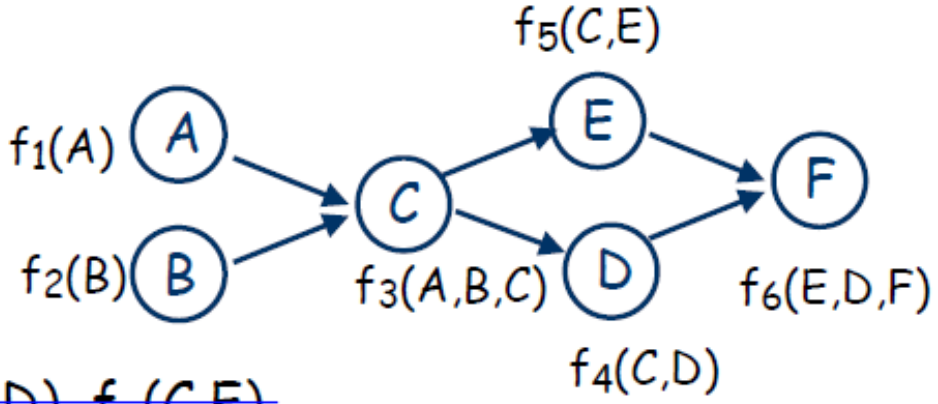
1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. F: $f_6(E,D,F)$
3. A: $f_1(A)$, $f_7(A,B,D,E)$
4. B: $f_2(B)$
5. E:
6. D:



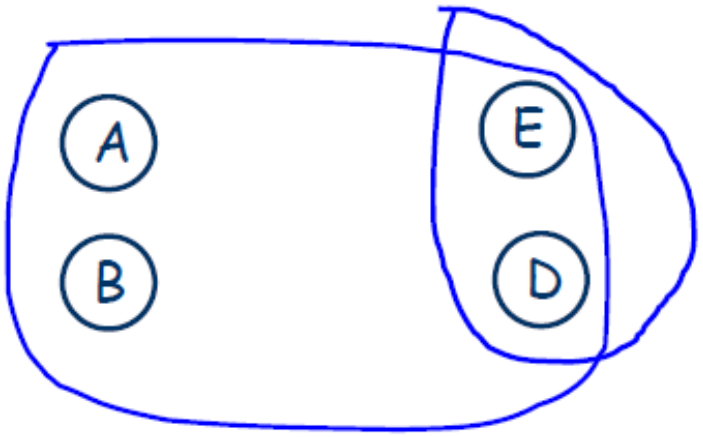
EXAMPLE

VE: Eliminate F, placing new factor f_8 in first applicable bucket.

Ordering:
C,F,A,B,E,D



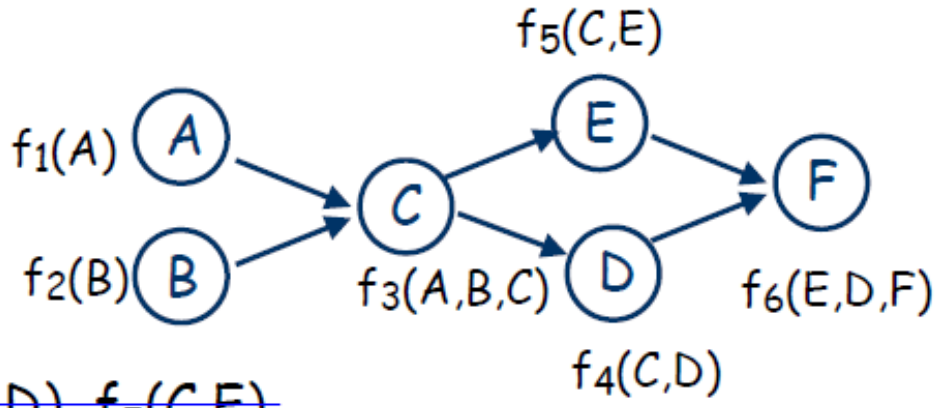
1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. A: $f_1(A)$, $f_7(A,B,D,E)$
4. B: $f_2(B)$
5. E: $f_8(E,D)$
6. D:



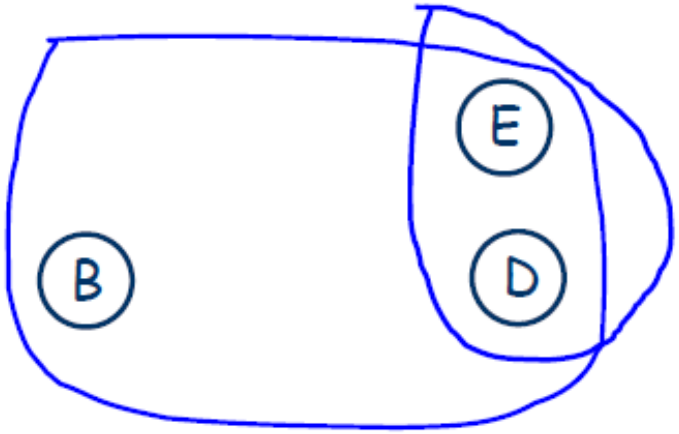
EXAMPLE

VE: Eliminate A, placing new factor f_9 in first applicable bucket.

Ordering:
C,F,A,B,E,D



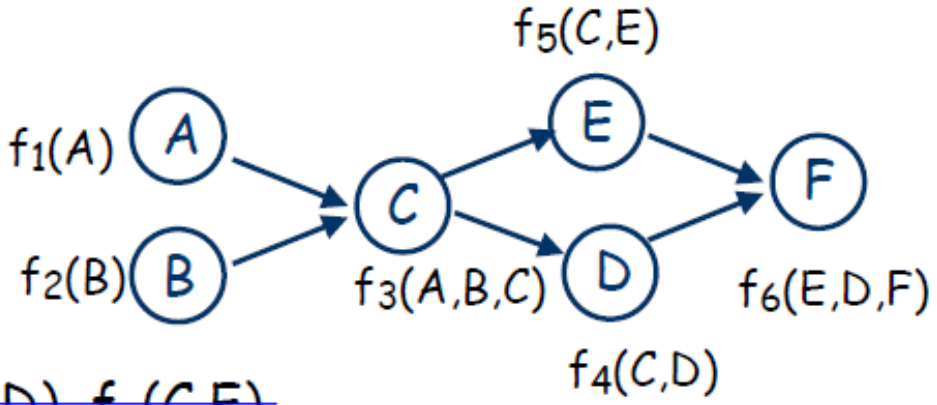
1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. ~~A: $f_1(A)$, $f_7(A,B,D,E)$~~
4. B: $f_2(B)$, $f_9(B,D,E)$
5. E: $f_8(E,D)$
6. D:



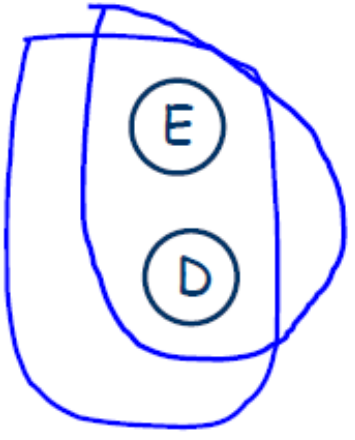
EXAMPLE

VE: Eliminate B, placing new factor f_{10} in first applicable bucket.

Ordering:
C,F,A,B,E,D



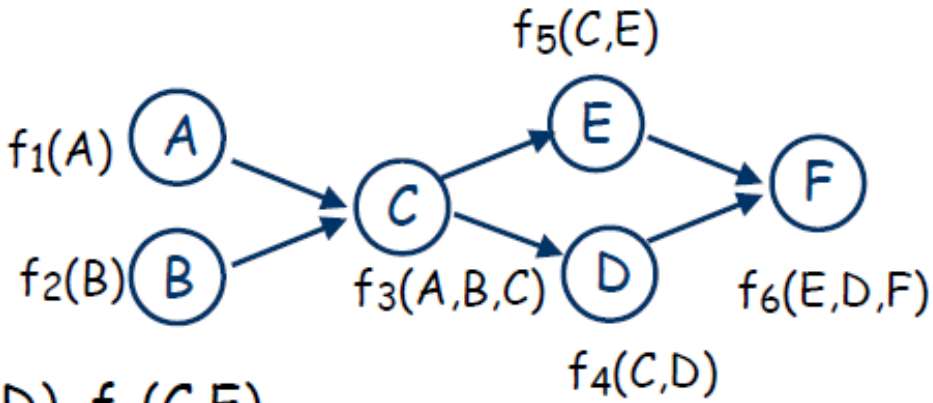
1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. ~~A: $f_1(A)$, $f_7(A,B,D,E)$~~
4. ~~B: $f_2(B)$, $f_9(B,D,E)$~~
5. E: $f_8(E,D)$, $f_{10}(D,E)$
6. D:



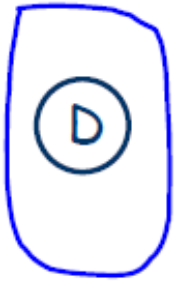
EXAMPLE

VE: Eliminate E, placing new factor f_{11} in first applicable bucket.

Ordering:
C,F,A,B,E,D



1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. ~~A: $f_1(A)$, $f_7(A,B,D,E)$~~
4. ~~B: $f_2(B)$, $f_9(B,D,E)$~~
5. ~~E: $f_8(E,D)$, $f_{10}(D,E)$~~
6. D: $f_{11}(D)$



Elimination Width

- Given an ordering π of the variables and an initial hypergraph $\mathcal{H}$ eliminating these variables yields a sequence of hypergraphs $\mathcal{H} = H_0, H_1, H_2, \ldots, H_n$.
- H_n contains only one vertex (the query variable).
- The elimination width of π is the maximum size (number of variables) of any hyperedge in any of the hypergraphs $H_0, H_1, \ldots, H_n$.
- The elimination width of the previous example was 4 ($\{A, B, E, D\}$ in H_1 and H_2).

Elimination Width

- If the elimination width of an ordering π is k, then the complexity of VE using that ordering is $2^{O(k)}$.
- Elimination width k means that at some stage in the elimination process a factor involving k variables was generated.
- That factor will require $2^{O(k)}$ space to store
- Then VE will require $2^{O(k)}$ space using this ordering
- And it will require $2^{O(k)}$ operations to process (either to compute in the first place, or when it is being processed to eliminate one of its variables).
- Then VE will require $2^{O(k)}$ time using this ordering.
- NOTE, that k is the elimination width of this particular ordering.

Elimination Width

- Given a hypergraph $\mathcal{H}$ with vertices $\{X_1, X_2, \ldots, X_n\}$ the elimination width of $\mathcal{H}$ is the MINIMUM elimination width of any of the $n!$ different orderings of the X_i minus 1.
- In the worst case the elimination width can be equal to the number of variables - exponential complexity.
- Under the best ordering VE will generate factors of size $2^{O(\omega)}$ where ω is the elimination width of the initial Bayes Net, and it will require this much space and time.

Complexity of Variable Elimination

- Note that VE input can already be larger than the number of variables.
- VE's inputs are the conditional probability tables $P(X|Par(X))$. If the largest CPT has k variables then VE's input will be $2^{O(k)}$.
- The table will have size equal to the product of the domain sizes of X and its parents.

Elimination width

- Exponential in the tree width is the best that VE can do.
- Finding an ordering that has minimum elimination width is NP-Hard.
- In practice there is no point in trying to speed up VE by finding the best possible elimination ordering.
- Heuristics are used to find orderings with good (low) elimination widths.
- In practice, this can be very successful. Elimination widths can often be relatively small, 8-10 even when the network has 1000s of variables.
- Thus VE can be much more efficient than simply summing the probability of all possible events (which is exponential in the number of variables).
- Sometimes, however, the elimination width is equal to the number of variables.

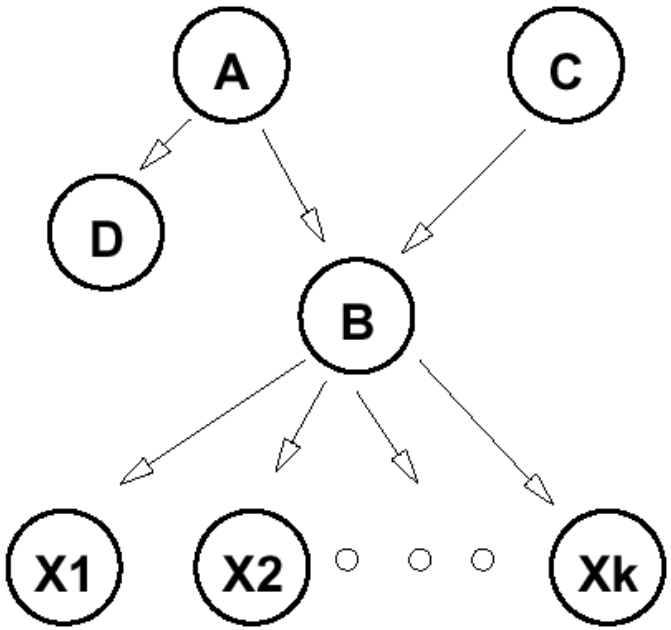
Finding Good Orderings

- A polytree is a singly connected Bayes Net: in particular there is only one path between any two nodes.
- A node can have multiple parents, but we have no cycles.
- Good orderings are easy to find for polytrees
  - At each stage eliminate a singly connected node.
  - Because we have a polytree we are assured that a singly connected node will exist at each elimination stage.
  - The size of the factors in the tree never increase!
  - Elimination width = size of largest input CPT

POLYTREES

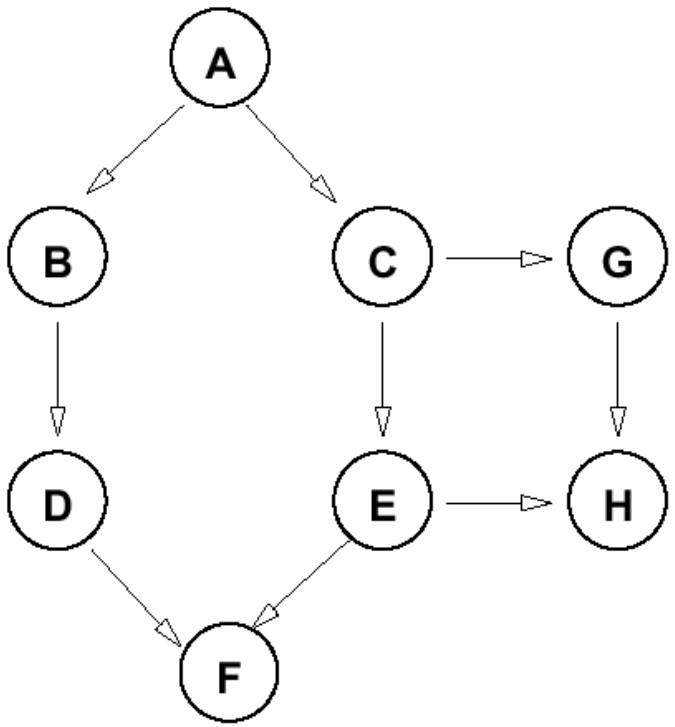
Eliminating singly connected nodes allows VE to run in time linear in size of network (not linear in the number of variables)

- e.g., in this network, eliminate $D, A, C, X_1, \ldots$; or eliminate $X_1, \ldots X_k, D, AC$; or mix up ...
- result: no factor ever larger than original CPTs
- eliminating B before these gives factors that include all of $A, C, X_1, \ldots X_k$!!!



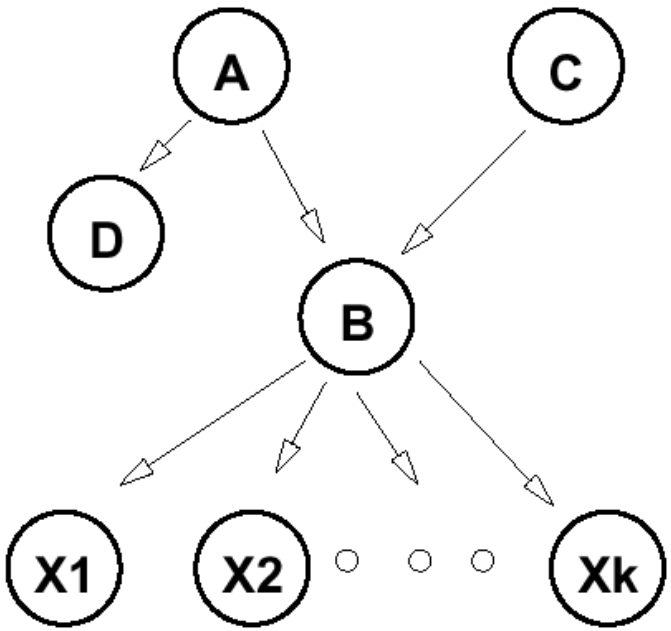
Effect of Different Orderings

- Suppose query variable is D. Consider different orderings for this network (not a polytree!)
- A,F,H,G,B,C,E: good
- E,C,A,B,G,H,F: bad



Min Fill Heuristic

- A fairly effective heuristic is always eliminate next the variable that creates the smallest size factor.
- This is called the min-fill heuristic.
- B creates a factor of size k+2
- A creates a factor of size 2
- D creates a factor of size 1
- The heuristic always solves polytrees in linear time.



Relevance

- Certain variables have no impact on the query. In network ABC, computing $P(A)$ with no evidence requires elimination of B and C.
- But when you sum out these vars, you compute a trivial factor (whose value are all ones); for example:
- eliminating C: $\sum_C P(C|B) = 1$ for any value of B (e.g., $P(c|b) + Pr(\neg c|b) = 1$)
- No need to think about B or C for this query

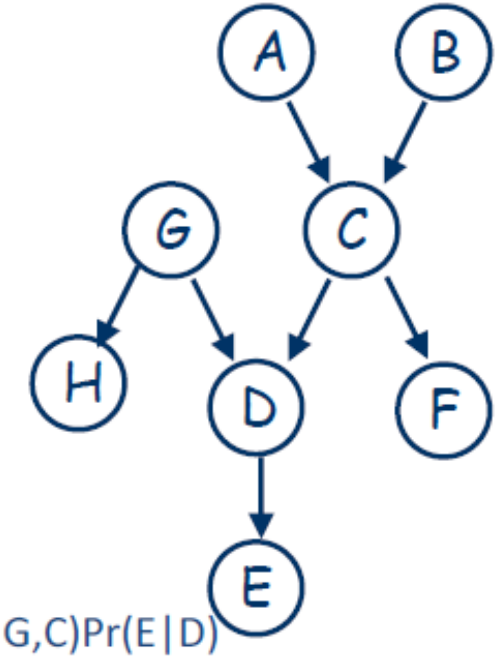
Relevance

- Can restrict attention to relevant variables. Given query Q, evidence **E**:
  - Q itself is relevant
  - if any node **Z** is relevant, its parents are relevant
  - if $e \in \mathbf{E}$ is a descendant of a relevant node, then **E** is relevant
- We can restrict our attention to the subnetwork comprising only relevant variables when evaluating a query Q

RELEVANCE: EXAMPLE

Relevance: Examples

- Query: P(F)
  - relevant: F, C, B, A
- Query: P(F|E)
  - relevant: F, C, B, A
  - **also: E, hence D, G**
  - intuitively, we need to compute P(C|E) to compute P(F|E)
- Query: P(F|H)
  - relevant F,C,A,B.



$Pr(A)Pr(B)Pr(C|A,B)Pr(F|C)\ Pr(G)Pr(h|G)Pr(D|G,C)Pr(E|D)$

$= \ldots Pr(G)Pr(h|G)Pr(D|G,C) \sum_E Pr(E|D)$ = a table of 1's

$= \ldots Pr(G)Pr(h|G) \sum_D Pr(D|G,C)$ = a table of 1's

$= [Pr(A)Pr(B)Pr(C|A,B)Pr(F|C)]\ [Pr(G)Pr(h|G)]$

$[Pr(G)Pr(h|G)] \neq 1$ but irrelevant

once we normalize, as it multiplies each value of F by the same number.