CMPUT463/563
Probabilistic Graphical Models

Exact Inference: Message Passing

Lili Mou

Dept. Computing Science, University of Alberta

lmou@ualberta.ca

Outline

- Variable elimination (last lecture)
  - Dynamic programming: pulling sum/max inside
  - Sum-product, max-product identical, due to semirings
  - To eliminate a node X: consider all potentials involving $X \Rightarrow$ marginalize/max X out $\Rightarrow$ Put resulting potential back, with scope $N(X)$
- Message passing
  - VE two passes => answer all queries of a variable
  - Belief propagation: Sum-product message passing
- Message passing on factor trees
- Message passing on junction trees

Variable Elimination

Input: Variables $\boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{X} = \boldsymbol{x}$, set of factors Φ

Elimination order (assuming $Z_1, Z_2, \cdots, Z_k$ wlog)

1. $\Phi \leftarrow \Phi$ with evidence potentials

2. For $i = 1, \cdots, k$

2.1 $\Phi' \leftarrow \{\phi \in \Phi' : Z_i \in \text{scope}(\phi)\}$

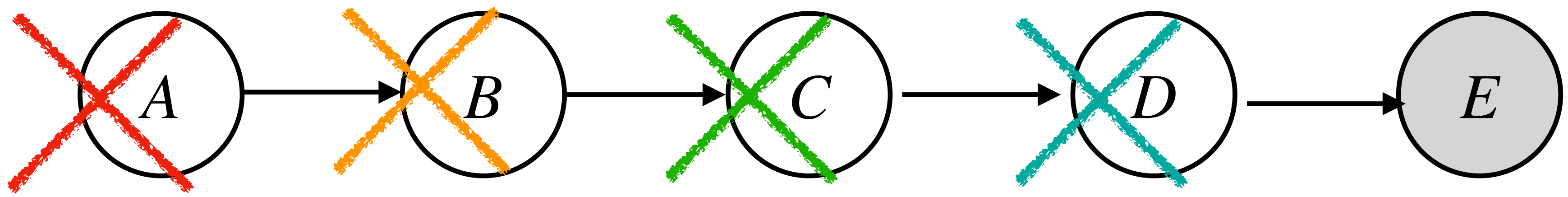
2.2. $\varphi \leftarrow \sum_{z_i} \prod_{\phi \in \Phi'} \phi$

2.3 $\Phi = \Phi \setminus \Phi' \cup \{\varphi\}$

Return $\prod_{\phi \in \Phi} \phi$ as a factor on $\boldsymbol{Y}$ given $\boldsymbol{X}$

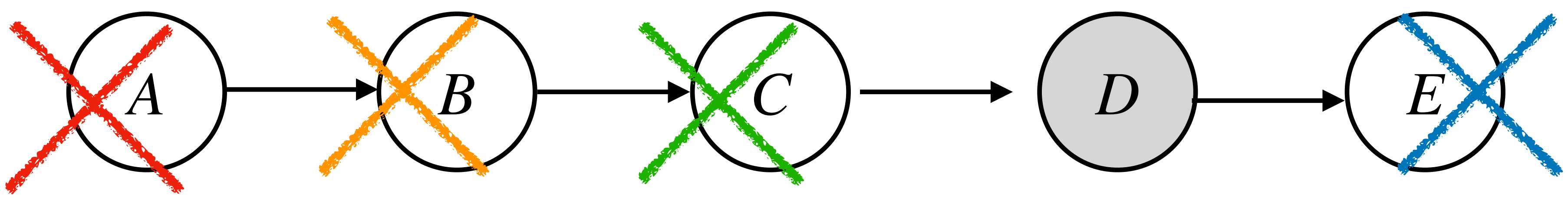
Examples

$P(e)$



$$P(e) = \sum_{d} \sum_{c} \sum_{b} \sum_{a} P(a)P(b \mid a)P(c \mid b)P(d \mid c)P(e \mid d)$$

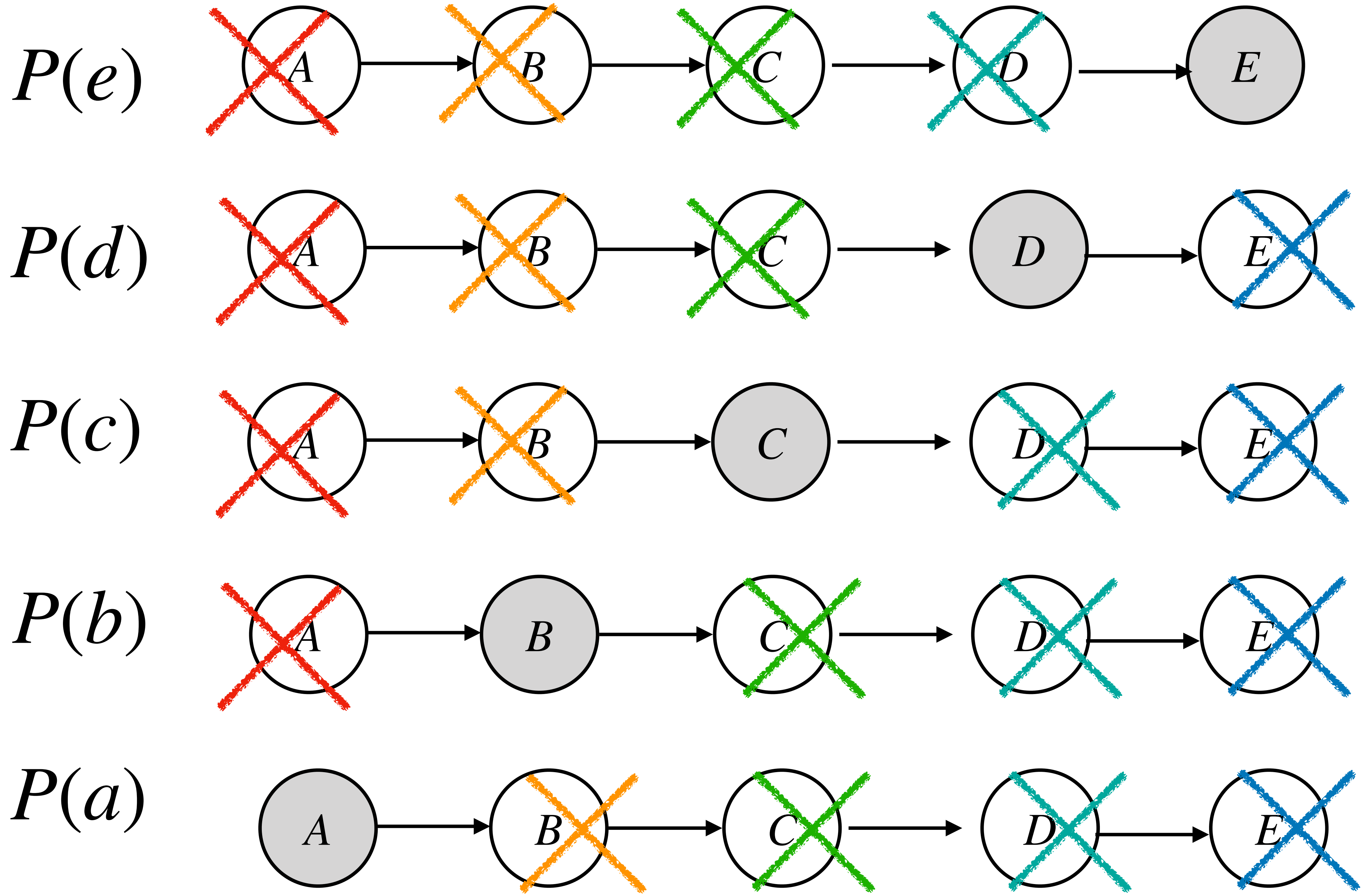
$P(d)$



$$P(d) = \sum_{e} \sum_{c} \sum_{b} \sum_{a} P(a)P(b \mid a)P(c \mid b)P(d \mid c)P(e \mid d)$$

Observation: If we have multiple queries for a graph, some DP results are reusable

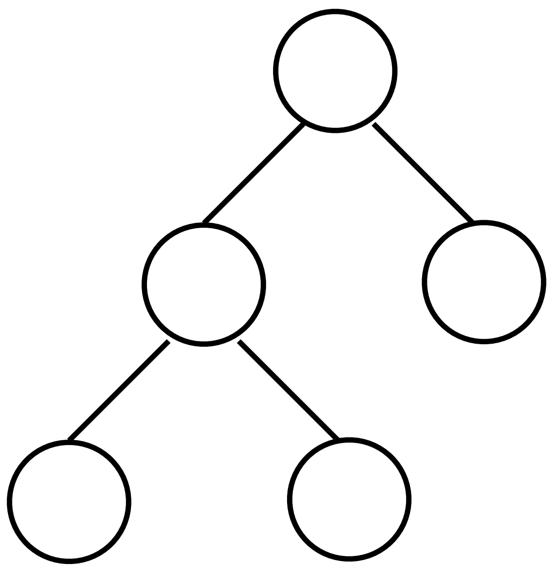
Even More

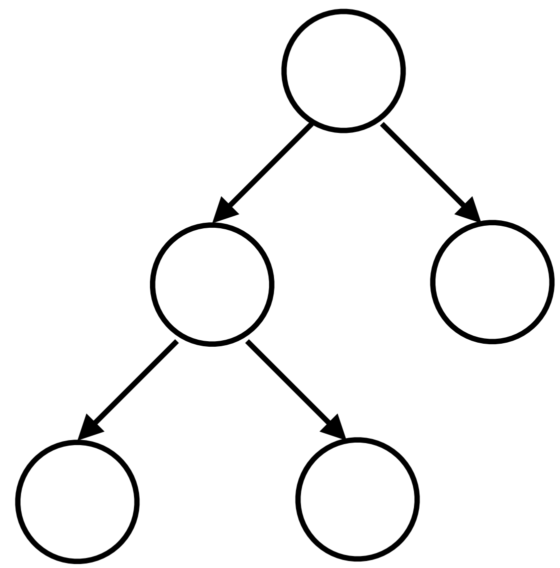


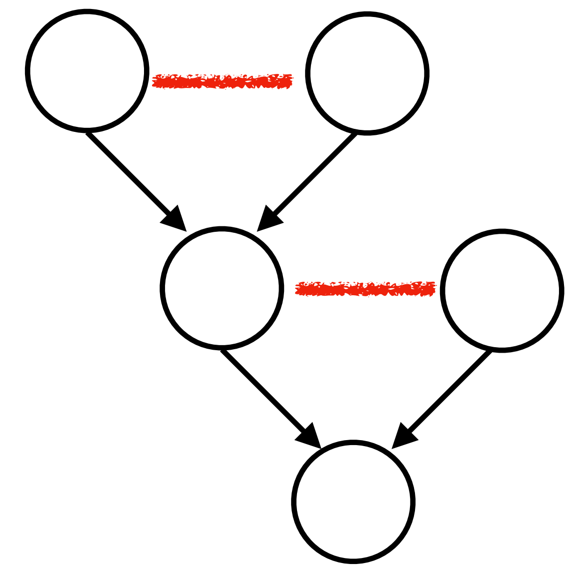
Observation: DP twice (from left and from right), all marginals are obtained. Congratulations! You are inventing the belief propagation algorithm

Trees

For certain graphs (namely, trees) DP results are reusable for different queries. A chain is a special case of trees.







An obvious order for variable elimination: Always eliminate a leaf. Will not introduce more edges during VE.

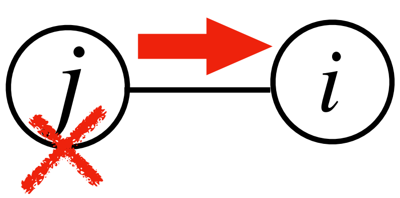
Any node separates a tree into two parts. VE needed for both sides, but is reusable

Work with factor graphs. Later

Message Passing

Without loss of generality, we assume having **one singleton** potential for each node and **one pairwise** potential for two connected nodes.

Initialization



Consider a leaf node j. If j does not have a neighbor, it's done. Otherwise, j can only have one neighbor, which we call i.

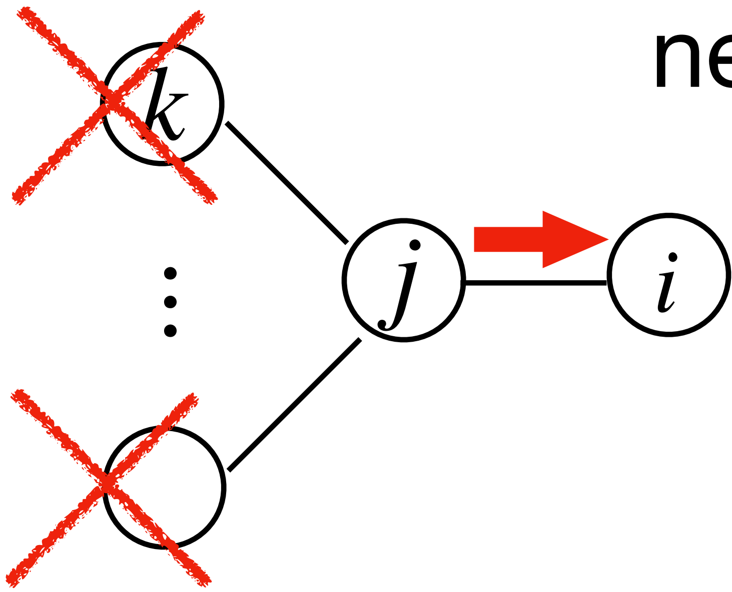
$$m_{ji}(x_i) = \sum_{x_j} \phi(x_j)\phi(x_i, x_j)$$

m_{ji} is called the *message* from j to i, a potential when j is eliminated.

Sum-product message passing is also called belief propagation; such a message is also called a belief.

Message Passing

Recursion



Suppose node j is now a leaf and can have at most one neighbor, which we call i. Other neighbors of j are already eliminated.

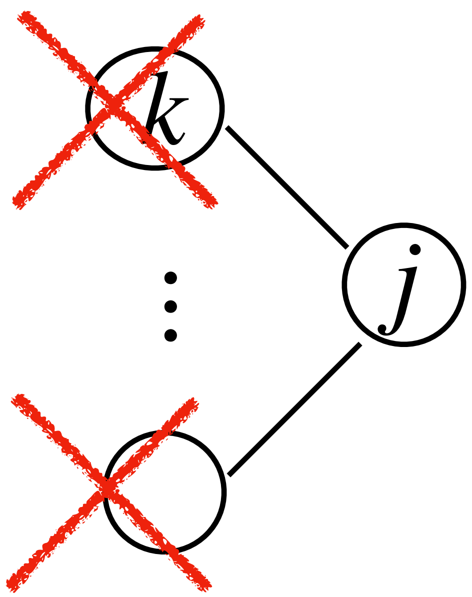
$$m_{ji}(x_i) = \sum_{x_j} \left(\phi(x_j)\phi(x_i, x_j) \prod_{k \in N(j) \setminus \{i\}} m_{kj}(x_j) \right)$$

m_{ji} is called the *message* from j to i, the potential after eliminating variables up to j before i, with a scope of i only.

Who can pass message? j can pass message to i if j has received messages from all its neighbors except i.

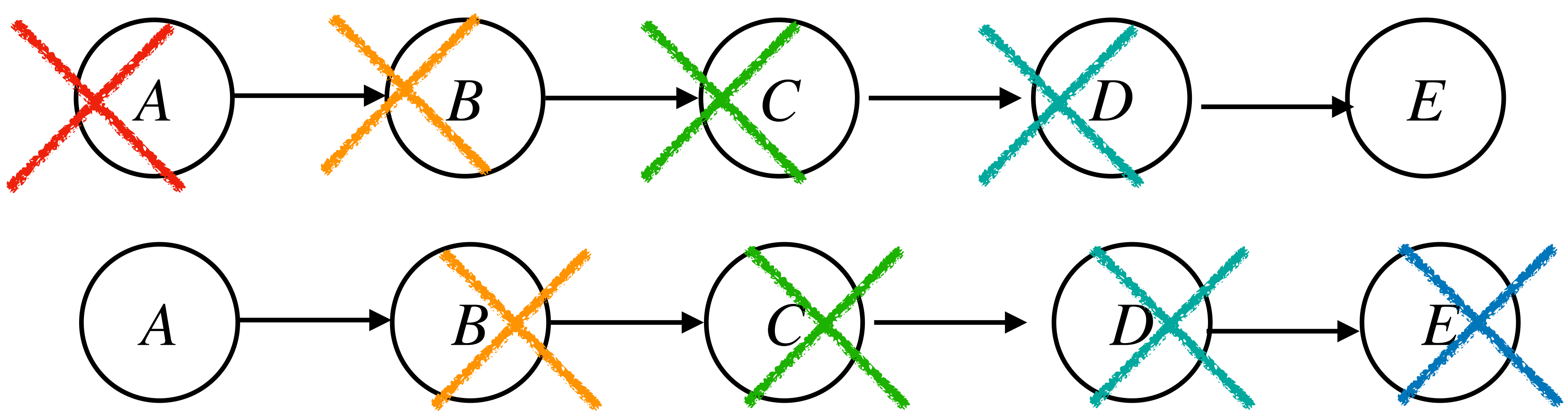
Message Passing

Termination



Suppose node j is now a leaf and it does not have other neighbors. Done!

But note: this is half of the story

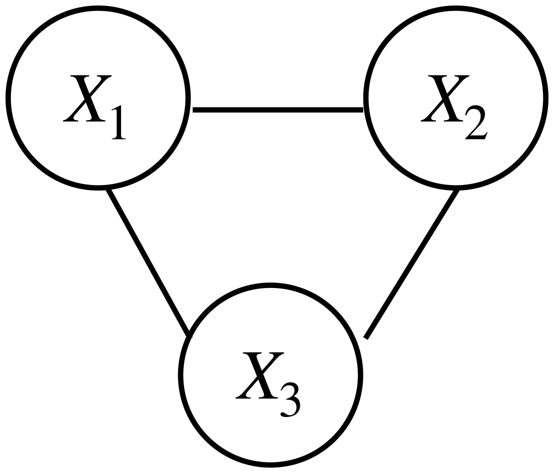


We need message passing the other way around.

It's also true for trees.

Non-Tree Structures

Assuming singleton and pairwise potentials



Suppose the query is $P(X_3)$. If we start sending messages from a node, say X_1, we have

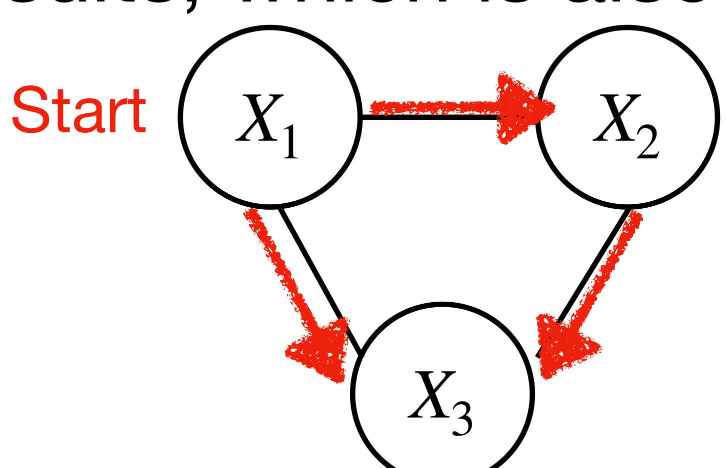
$$m_{12} = \sum_{X_1} \phi(X_1)\phi(X_1, X_2) \qquad m_{13} = \sum_{X_1} \phi(X_1)\phi(X_1, X_3)$$

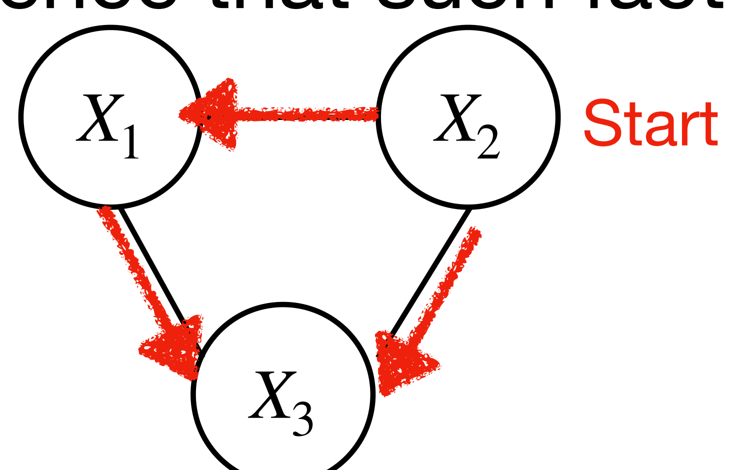
$$P(X_3) = \sum_{X_2} \sum_{X_1} \phi(X_1)\phi(X_2)\phi(X_3)\phi(X_1, X_2)\phi(X_2, X_3)\phi(X_1, X_3)$$

Factorization in such a style is fundamentally wrong

$$\sum_x xyz = \left(\sum_x xz\right)\left(\sum_x xy\right)$$

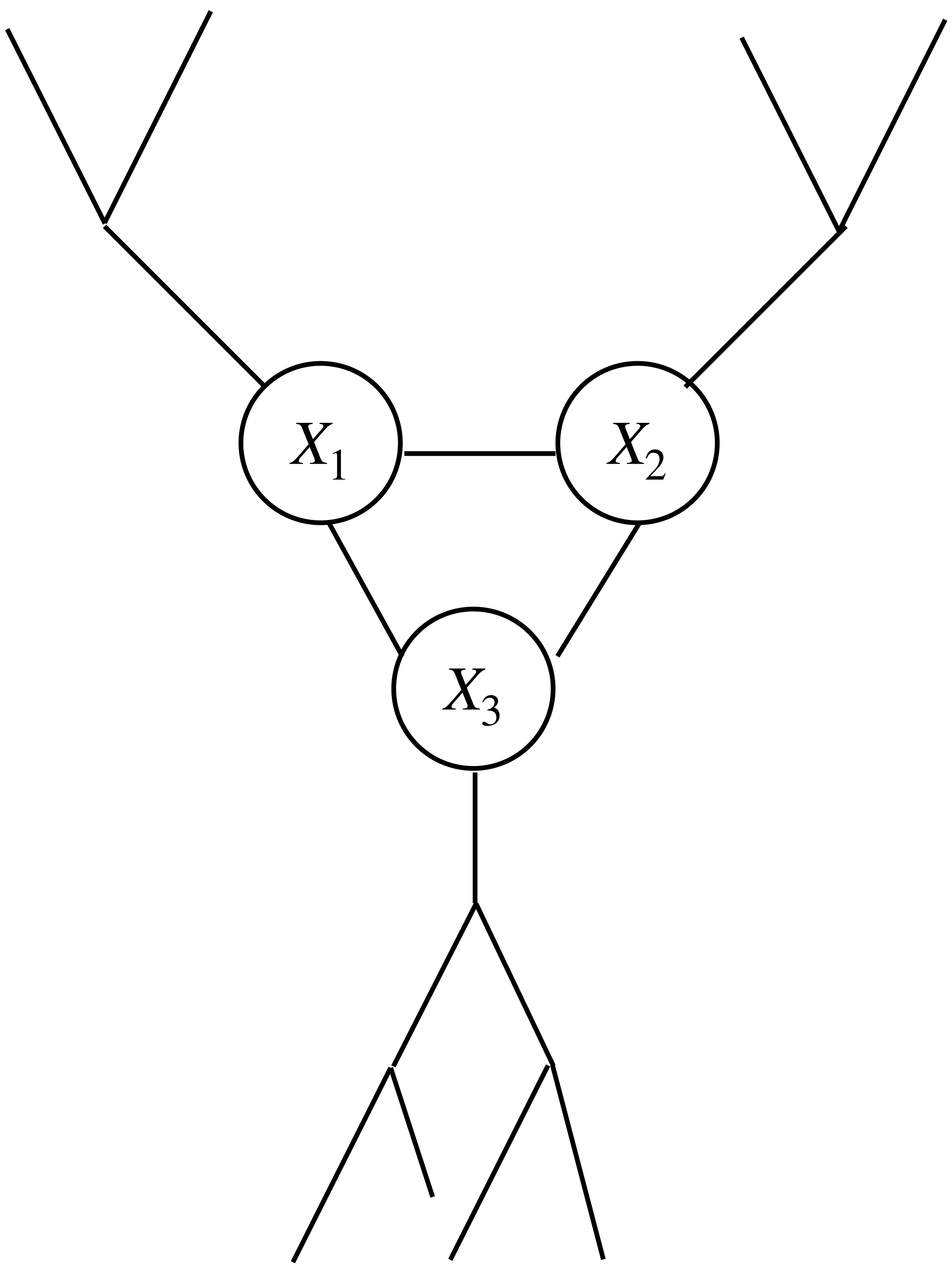
In fact, you can show that two message passings yield different results, which is also the evidence that such factorization is wrong

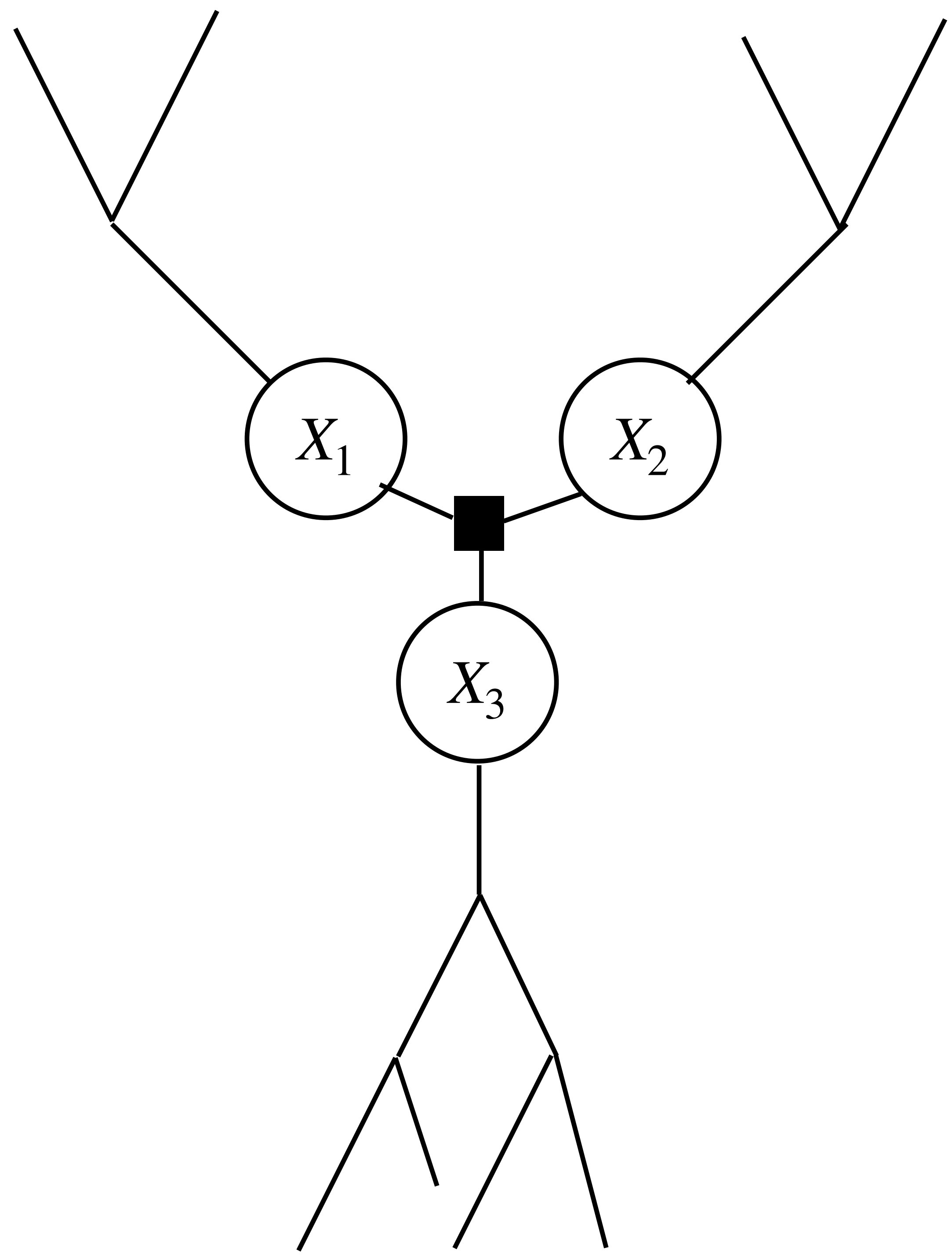




Non-Tree Structures

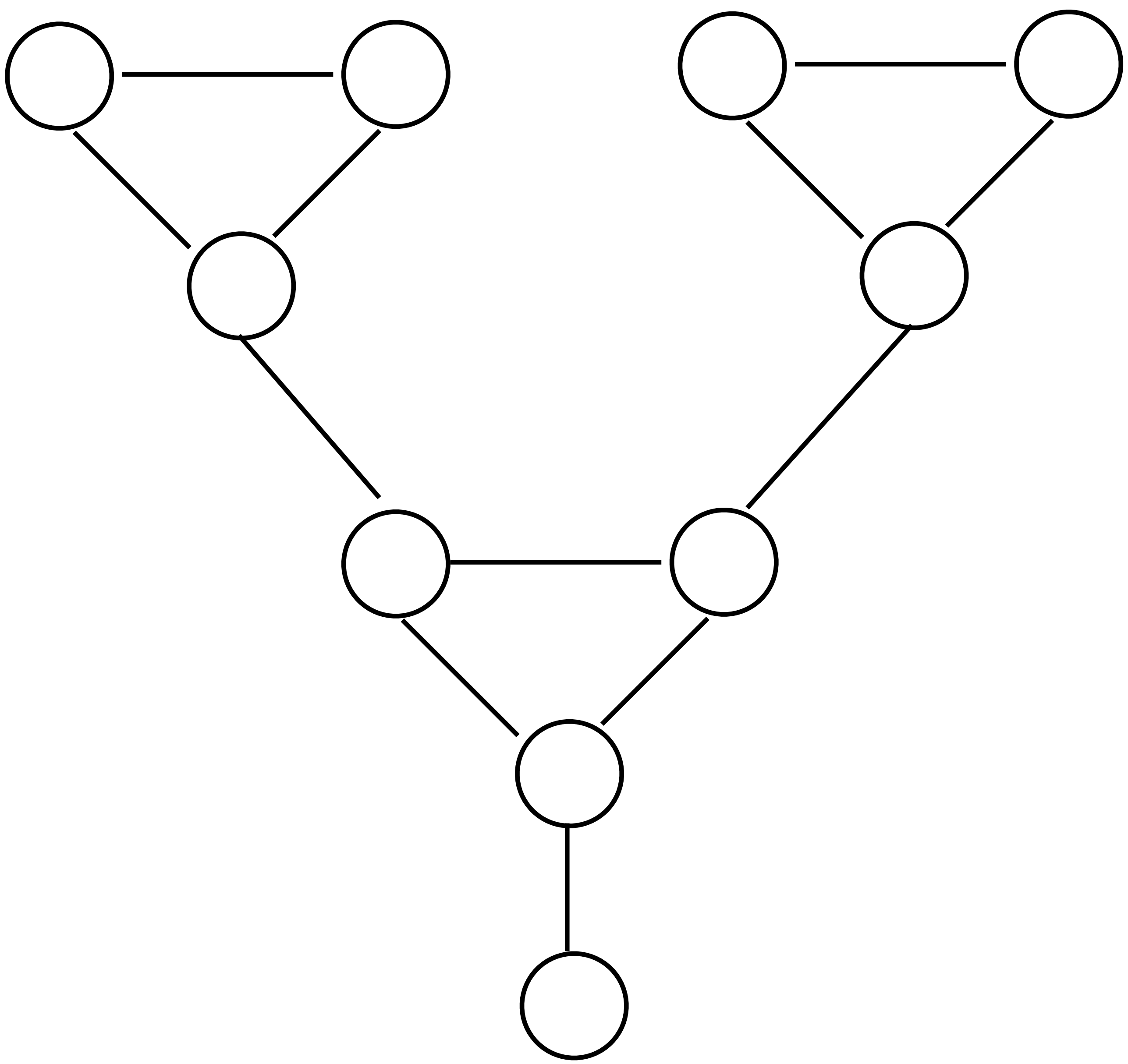
But suppose the remaining structures are tree-like, message passing can treat X_1, X_2, X_3 as a whole, as a factor

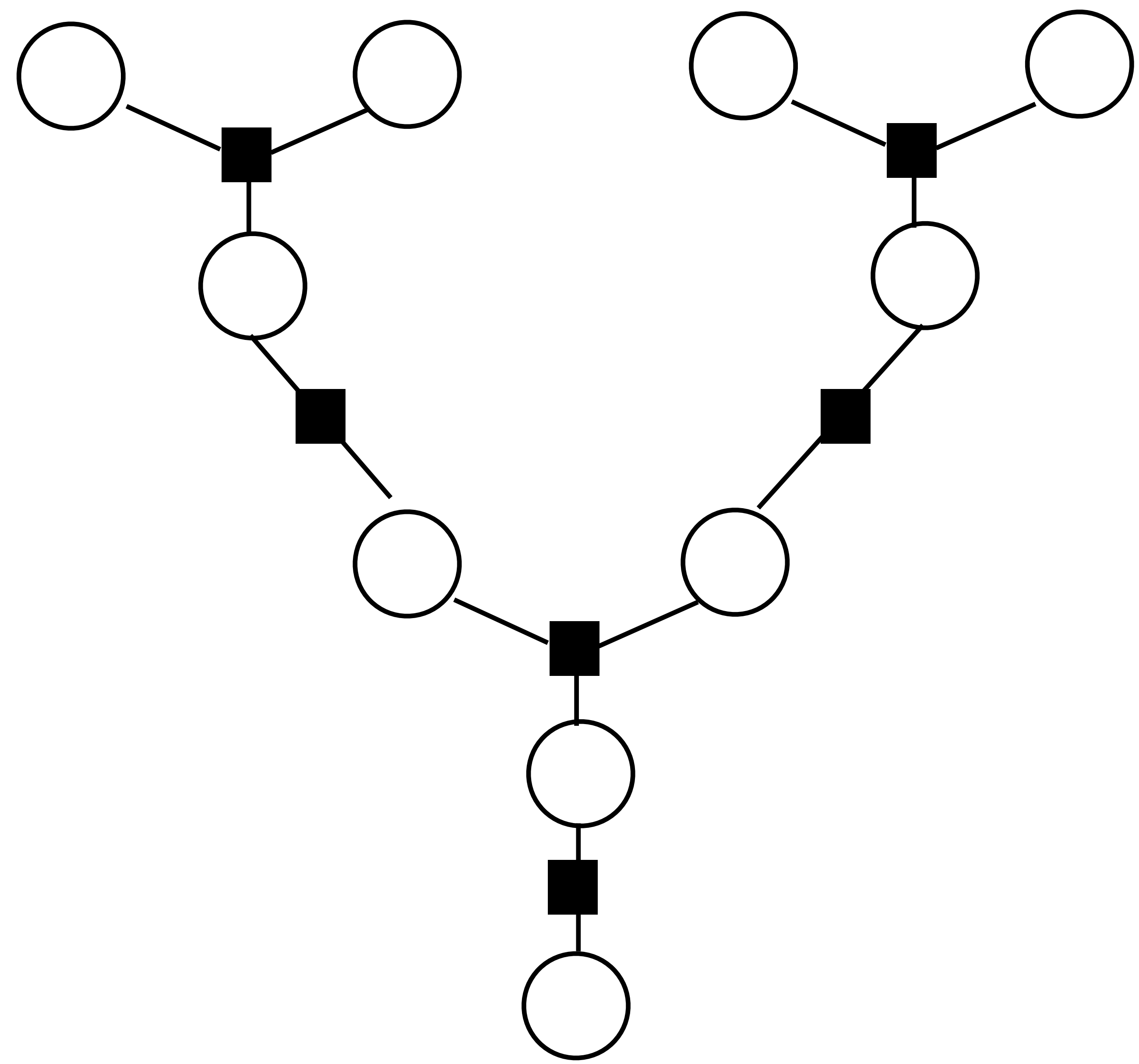




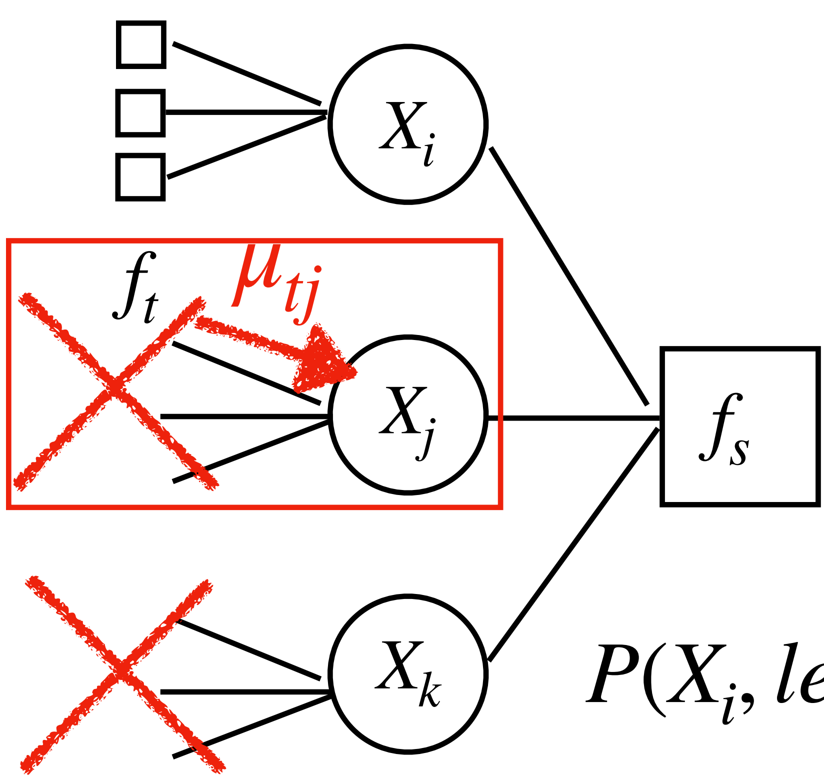
Message Passing on Factor Trees

Factor graph is a bipartite graph between variables and factors. We consider such graphs that are trees.





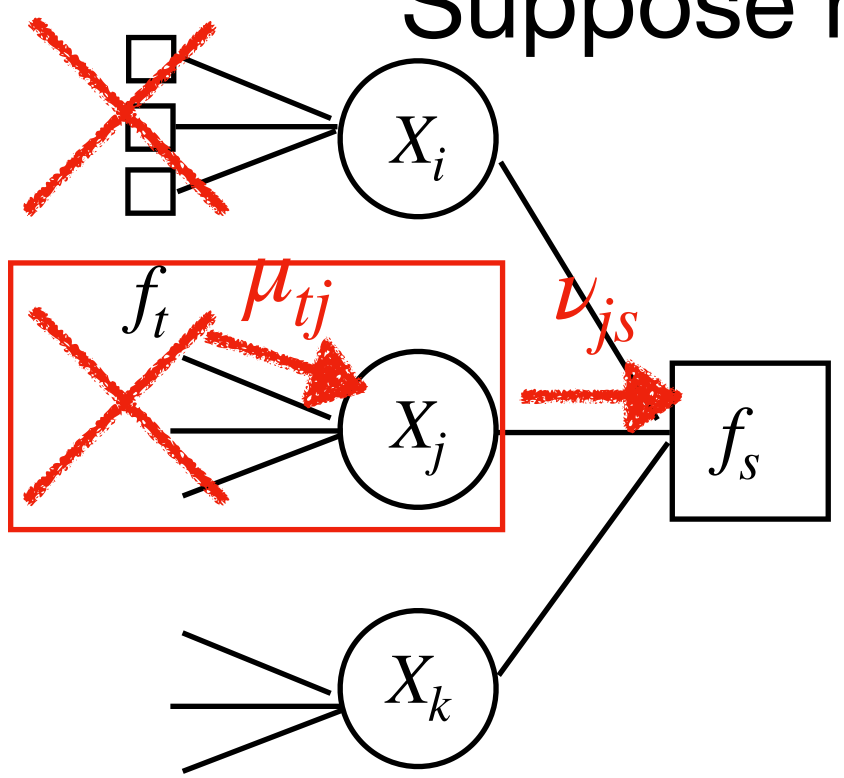
Message Passing on Factor Trees



Suppose we would like to eliminate X_j and X_k, whose left nodes are already eliminated with potentials μ_{tj} from factor t to node j

$$P(X_i, left(X_i)) \propto \phi(x_i, left(X_i)) \sum_{X_{N(s)\setminus i}} f_s(X_{N(s)}) \prod_{j\in N(s)\setminus i} \boxed{\prod_{t\in N(j)\setminus s} \mu_{tj}(x_j)}$$

Suppose now we would like to eliminate X_i and X_k

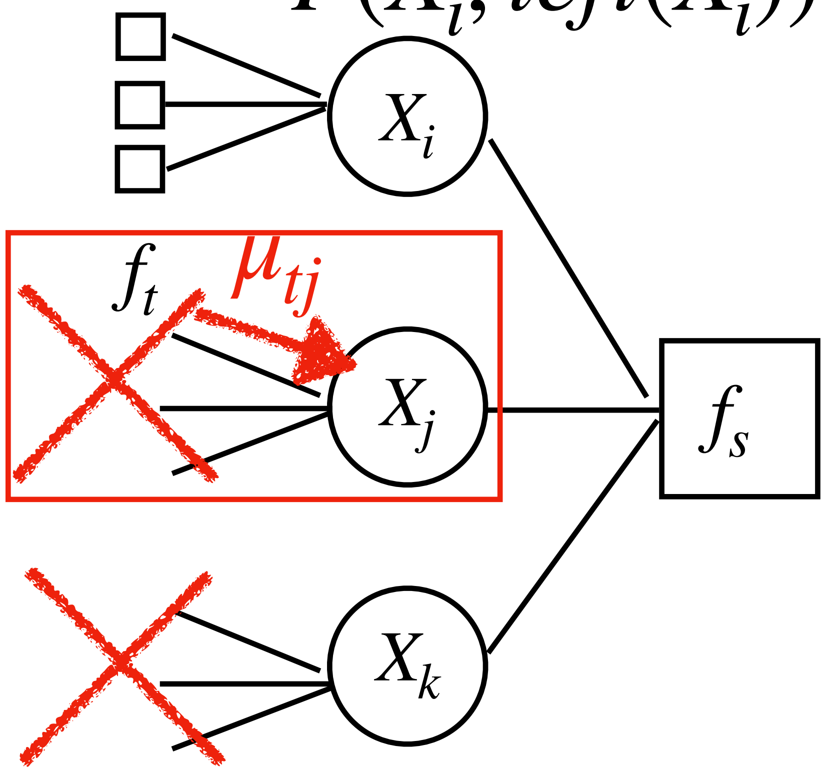


Reusable! DP is possible

$$\nu_{js} = \prod_{t\in N(j)\setminus s} \mu_{tj}(x_j)$$

Message Passing on Factor Trees

$$P(X_i, left(X_i)) \propto \phi(x_i, left(X_i)) \sum_{X_{N(s)\backslash i}} f_s(X_{N(s)}) \prod_{j \in N(s)\backslash i} \boxed{\prod_{t \in N(j)\backslash s} \mu_{tj}(x_j)}$$



Two types of messages

ν_{js}: variables j to factor s

$$\nu_{js}(x_i) = \prod_{t \in N(j)\backslash s} \mu_{tj}(x_j)$$

(marginalization is deferred with the factor f_s)

μ_{si}: factor s to variable i

$$\mu_{si}(x_i) = \sum_{X_{N(s)\backslash i}} f_s(X_{N(s)}) \prod_{j \in N(s)\backslash i} \nu_{js}(x_j)$$

(potential on X_i by eliminating other variables in the factor)

Message Passing on Factor Trees

$$P(X_i, left(X_i)) \propto \phi(x_i, left(X_i)) \underline{\sum_{X_{N(s)\setminus i}} f_s(X_{N(s)}) \prod_{j \in N(s)\setminus i} \underline{\prod_{t \in N(j)\setminus s} \mu_{tj}(x_j)}}$$

$$\underline{\nu_{js}(x_i) = \prod_{t \in N(j)\setminus s} \mu_{tj}(x_j)}$$

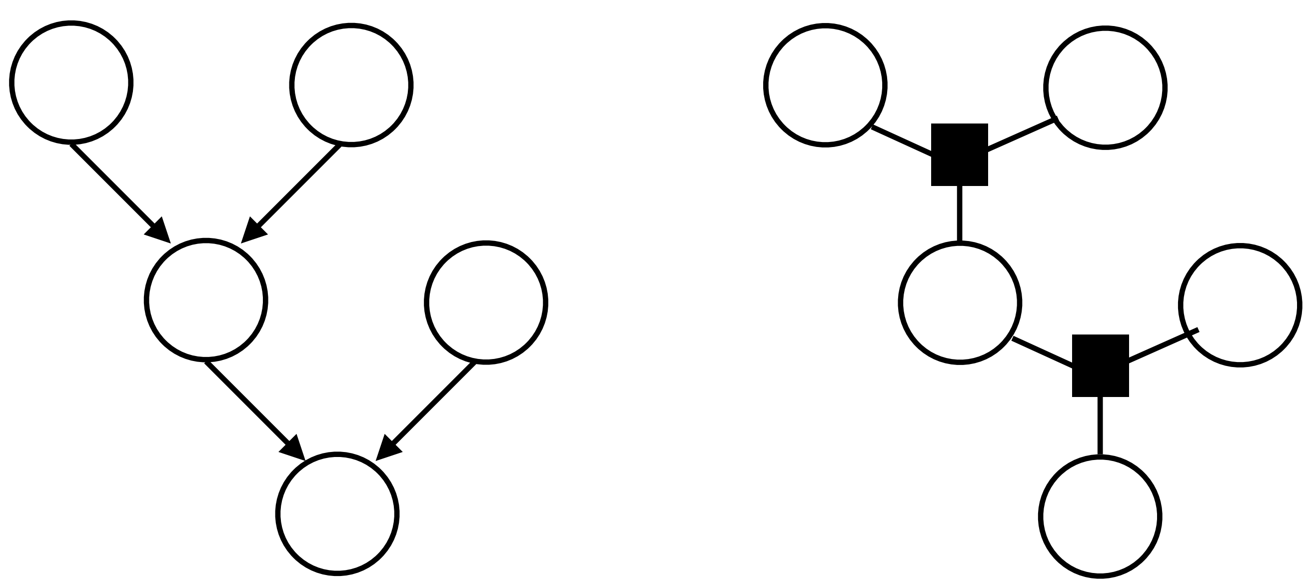
$$\underline{\mu_{si}(x_i)} = \sum_{X_{N(s)\setminus i}} f_s(X_{N(s)}) \underline{\prod_{j \in N(s)\setminus i} \nu_{js}(x_j)}$$

Message passing on factor trees is indeed an exact inference algorithm, as it’s reusing VE intermediate results

Complexity

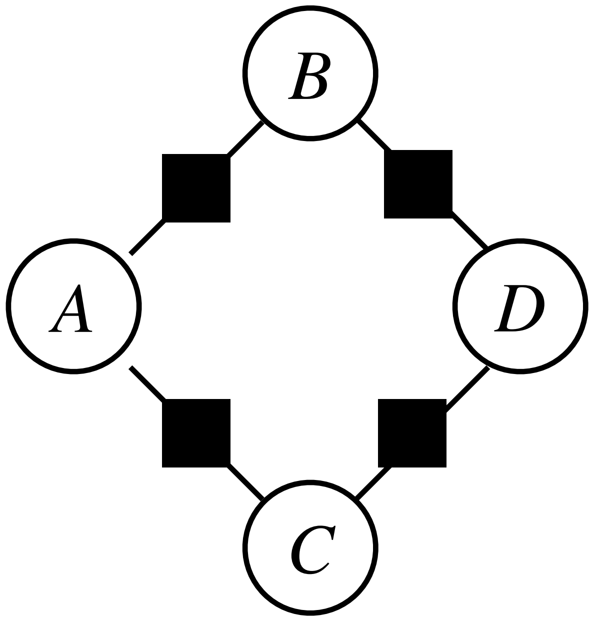
Complexity: exponential wrt largest factor

For ancestor trees, as efficient as representation the conditional probability

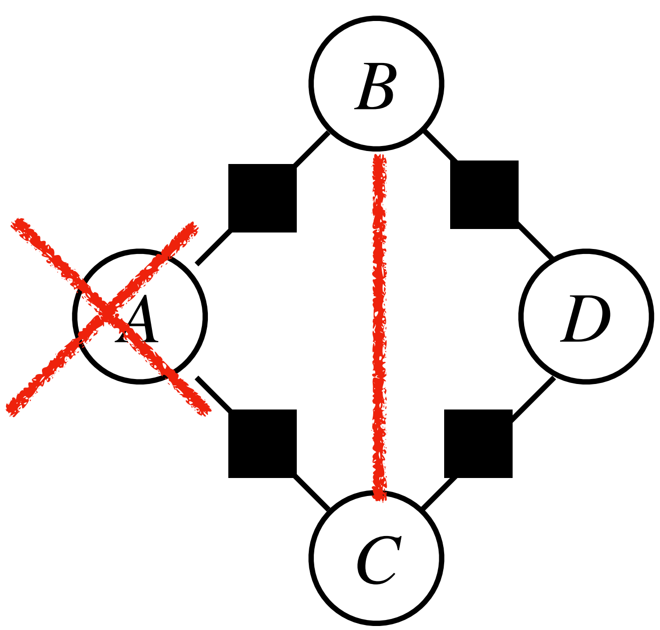


If the factor graph is cyclic, then we need to consider more variables at a time

What if factor graph is cyclic?



- The factor graph is cyclic. We cannot start message passing on the factor graph.
- However, we don't have to consider all variables either

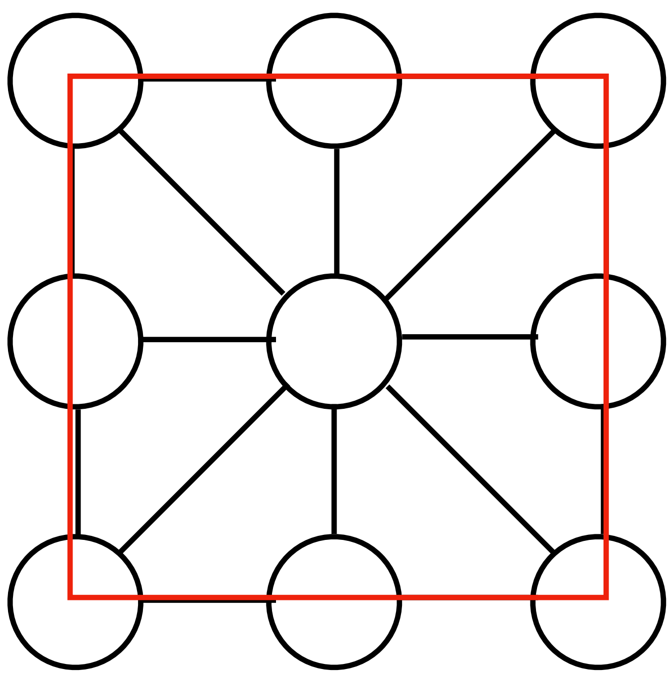


For example, when we eliminate A, we connect B, C without involving D

A special type of graphs: chordal graphs

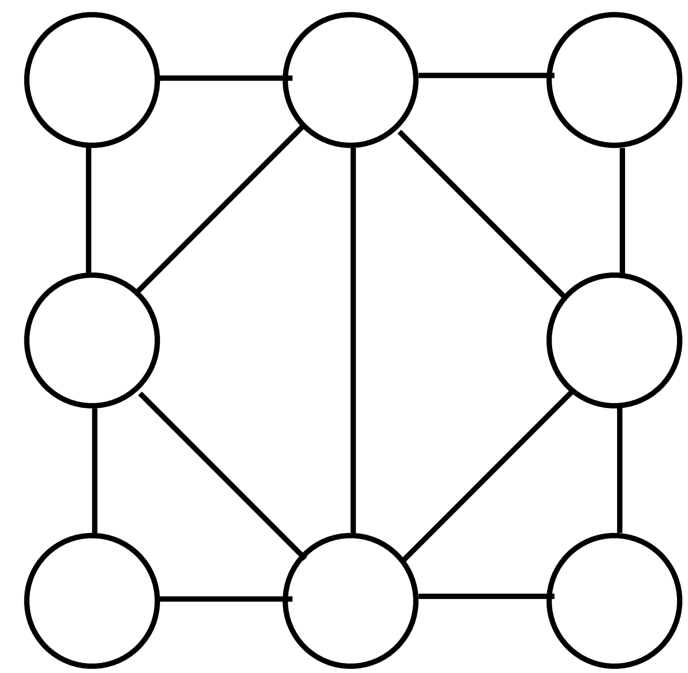
Chordal Graphs

Def: A graph is *chordal* (aka *triangulated*) if any loop of length >=4 has a chord (a straight line going across the loop)



←Not chordal

Chordal →



Many graph-theoretical results:

- For a chordal graph, there exists a VE order that does not introduce new edges
- The graph induced by VE is chordal

Junction Tree

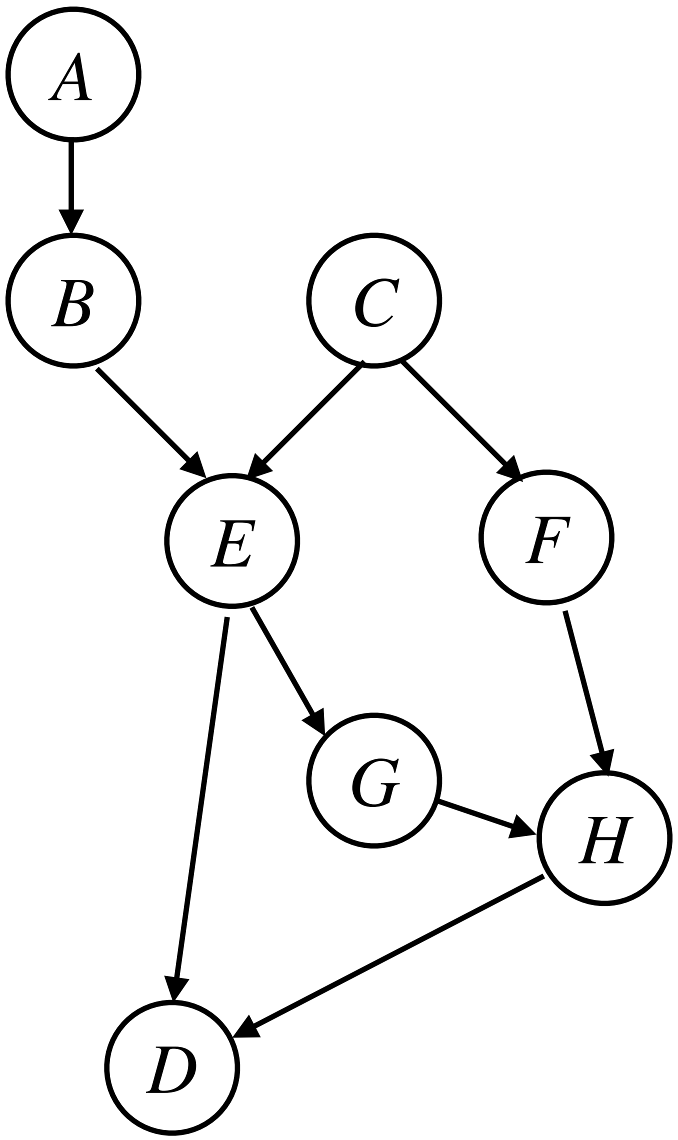
For a chordal graph, define a vertex C_i in the junction tree as a max clique. Connect the nodes by spanning tree algorithms (e.g., max spanning tree with intersecting variables)

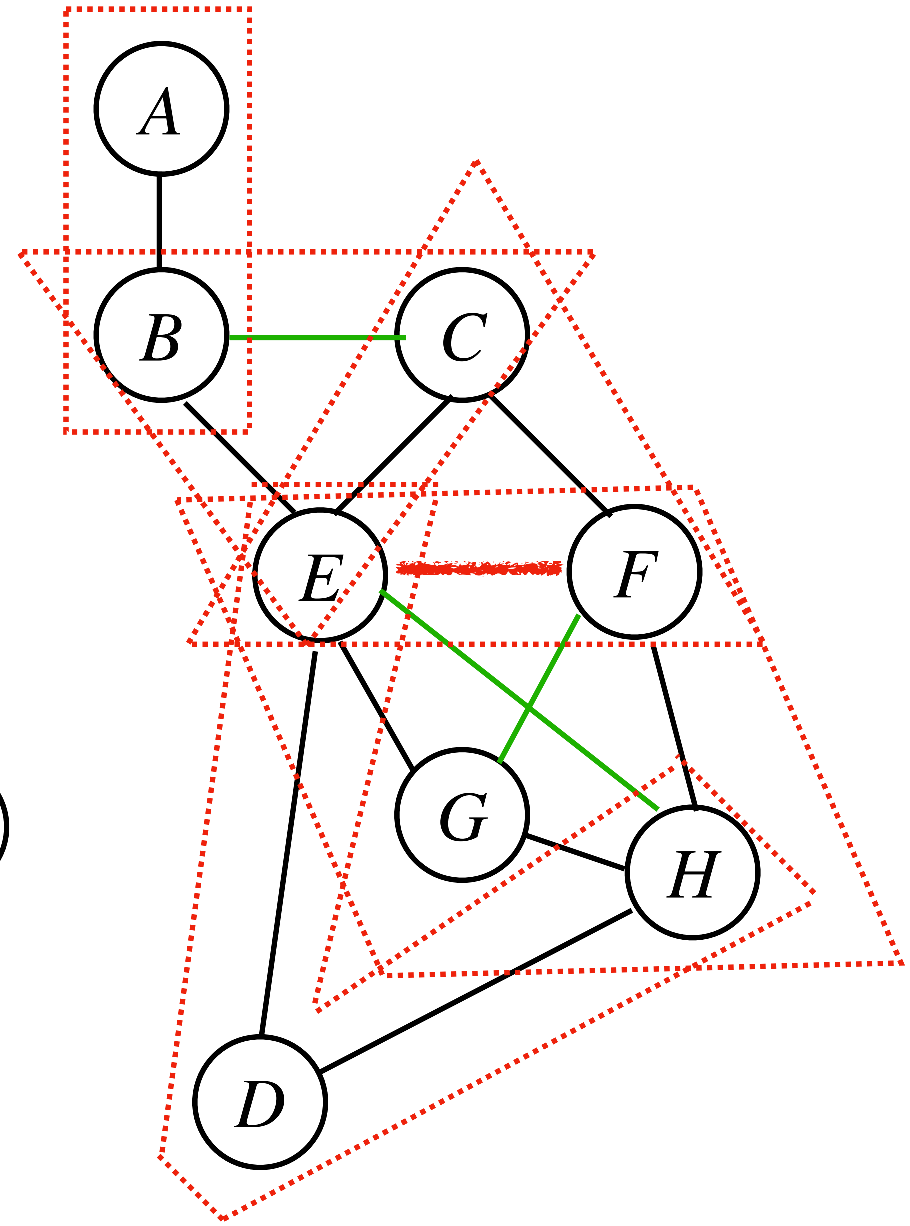
This will correspond to VE with some order such that $C_i - C_j$ if the message of i is used in computing the message of j.

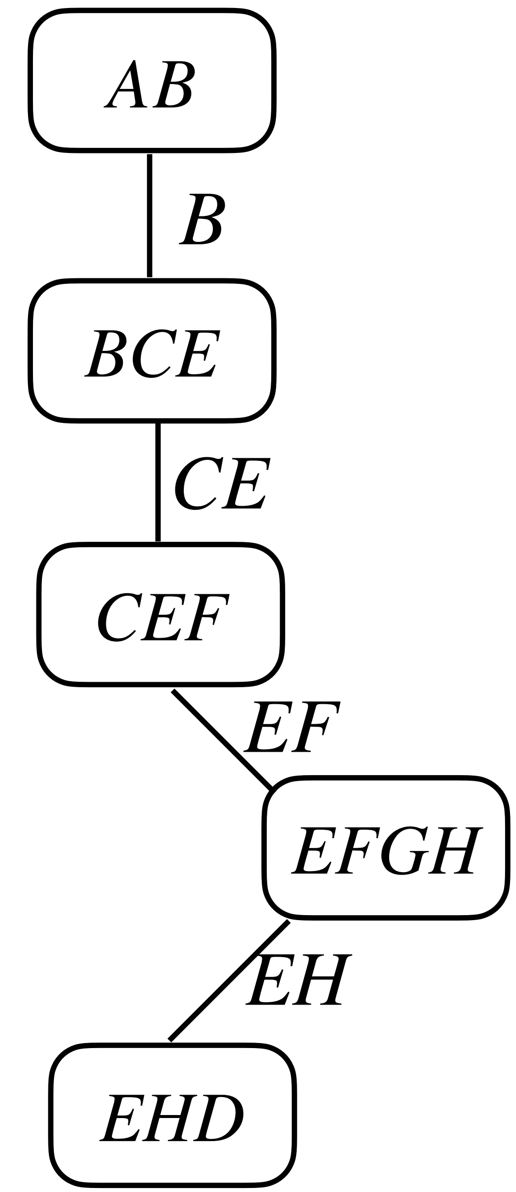
Running intersection property: If a variable occurs in C_i and C_j, then it must occur on every nodes in the trail $C_i - \cdots - C_j$. VE satisfies running intersection property because a variable eliminated will not recur.

Junction Tree Algorithm

Elimination order: $ABCDEFGH$







Moralization

Chordal

Absorb factors into a clique

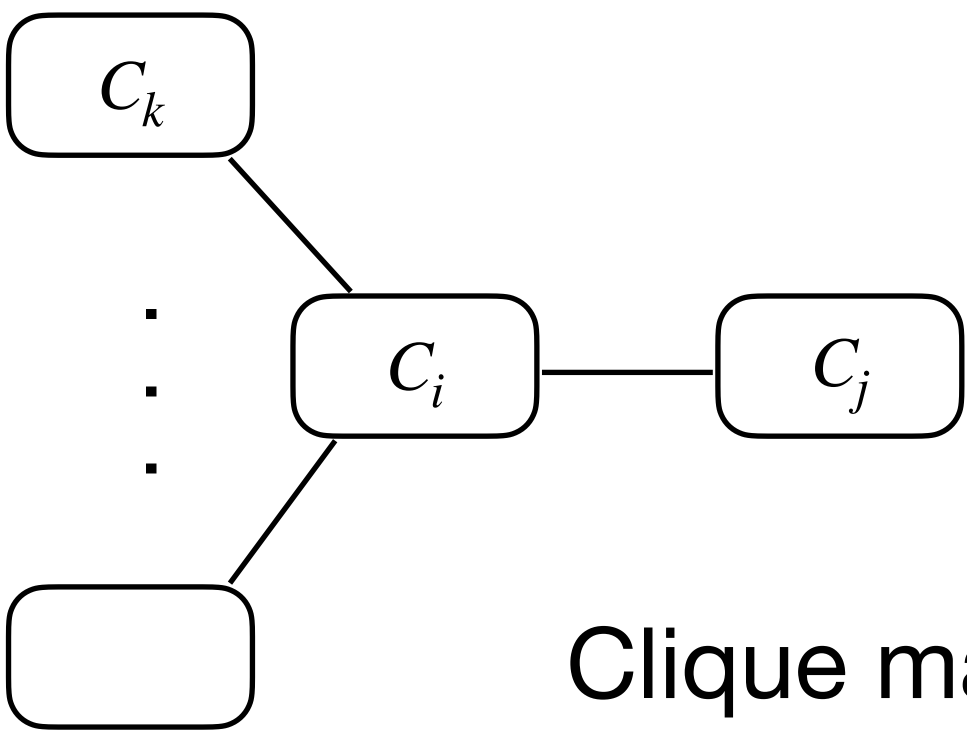
Now we can design a MP-like algorithm to eliminate variables
Results will be clique marginals

Junction Tree Algorithm

Let the variables of ith clique be C_i. Let $S_{ij} = C_i \cap C_j$

Sending message from C_i to C_j is to eliminate variables $C_i \backslash S_{ij}$

Shafer-Shenoy update:



$$\mu_{i \to j}(S_{ij}) = \sum_{C_i \backslash S_{ij}} \psi(C_i) \prod_{k \in N(i) \backslash \{j\}} \mu_{k \to i}(S_{ki})$$

Clique marginal

$$P(C_i) = \psi(C_i) \prod_{k \in N(i)} \mu_{k \to i}(S_{ki})$$

Junction Tree Algorithm

Shafer-Shenoy update: $\mu_{i\to j}(S_{ij}) = \sum_{C_i \backslash S_{ij}} \psi(C_i) \prod_{k\in N(i)\backslash\{j\}} \mu_{k\to i}(S_{ki})$

Clique marginal $$P(C_i) = \psi(C_i) \prod_{k\in N(i)} \mu_{k\to i}(S_{ki})$$

Hugin update

- Store the product in the potential. Store previous message on the edge and discount it during MP back.
- More time efficient

Initialization

$$\phi_i(C_i) = \psi(C_i)$$

$$\phi_{ij}(S_{ij}) = 1$$

Update from C_i to C_j over S_{ij}

$$\phi_{ij}(S_{ij})^{(\mathrm{new})} = \sum_{C_i \backslash S_{ij}} \phi_i(C_i)$$

$$\phi_j(C_j)^{(\mathrm{new})} = \phi_j(C_j)\frac{\phi_{ij}(C_{ij})^{(\mathrm{new})}}{\phi_{ij}(C_{ij})}$$

Denominator = 1 if first time passing an edge from i to j;

or = message from i to j, if second time passing from j to i

Junction Tree Algorithm

Summary

1. Moralize (easy)
2. Triangulate the graph (NP-hard)
3. Build the junction tree
4. Shafer-Shenoy update or Hugin update

$\left.\begin{array}{l}\text{2.}\\\text{3.}\end{array}\right\}$ Can be given by VE

Two-pass message passing => marginals of all potentials