Georgia Institute of Technology

CSIP
Center for Signal & Image Processing

ECE/ML/CS/ISYE 8803

Exact Inference in Graphical Models

Module 5:
Message Passing Algorithm

Faramarz Fekri

Center for Signal and Information Processing

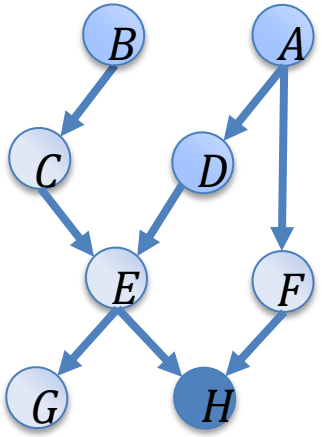
Overview

- Order of Variable Elimination and Complexity
- Graph Elimination
  - Message passing (Sum-Product, belief propagation)
    - Over chain
    - Over trees
  - Factor Graphs
    - Message Passing (Sum-Product) on Factor Trees
    - Max Product on Factor Trees

Read Chapter 9 of K&F----Chapters 3 and 4 of M. Jordan

Order of Variable Elimination (I)

- Elimination order H, G, F, E, B, C, D



- $P(A) =$
- $P(A) \sum_d P(d|A)(\sum_c(\sum_b P(b)P(c|b))(\sum_e P(e|c,d)(\sum_g P(g|e))(\sum_f P(f|A) \sum_h P(h|e,f))))$

$m_{BC}(c)$

$m_{GE}(e)$

$m_{H(EF)}(e,f)$

$m_{F(AE)}(A,e)$

$m_{E(ACD)}(A,c,d)$

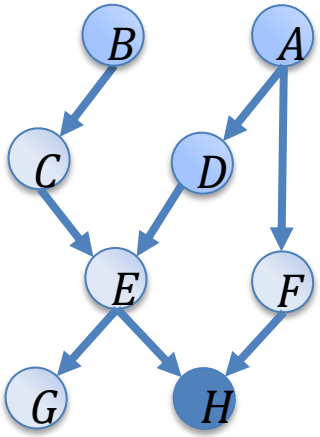
$m_{C(AD)}(A,d)$

$m_{DA}(A)$

4-way tables created!

Order of Variable Elimination (II)

- Elimination order G, H, F, B, C, D, E



- $P(A)$
- $= \sum_e (\sum_d P(d|A) \sum_c P(e|c,d) \left((\sum_b P(b)P(c|b))\right) \left((\sum_f P(f|A)(\sum_h P(h|e,f))\right) \sum_g P(g|e)\right))$

$m_{BC}(c)$

$m_{H(EF)}(e,f)\, m_{GE}(e)$

$m_{C(ED)}(e,d)$

$m_{F(AE)}(A,e)$

$m_{D(AE)}(A,e)$

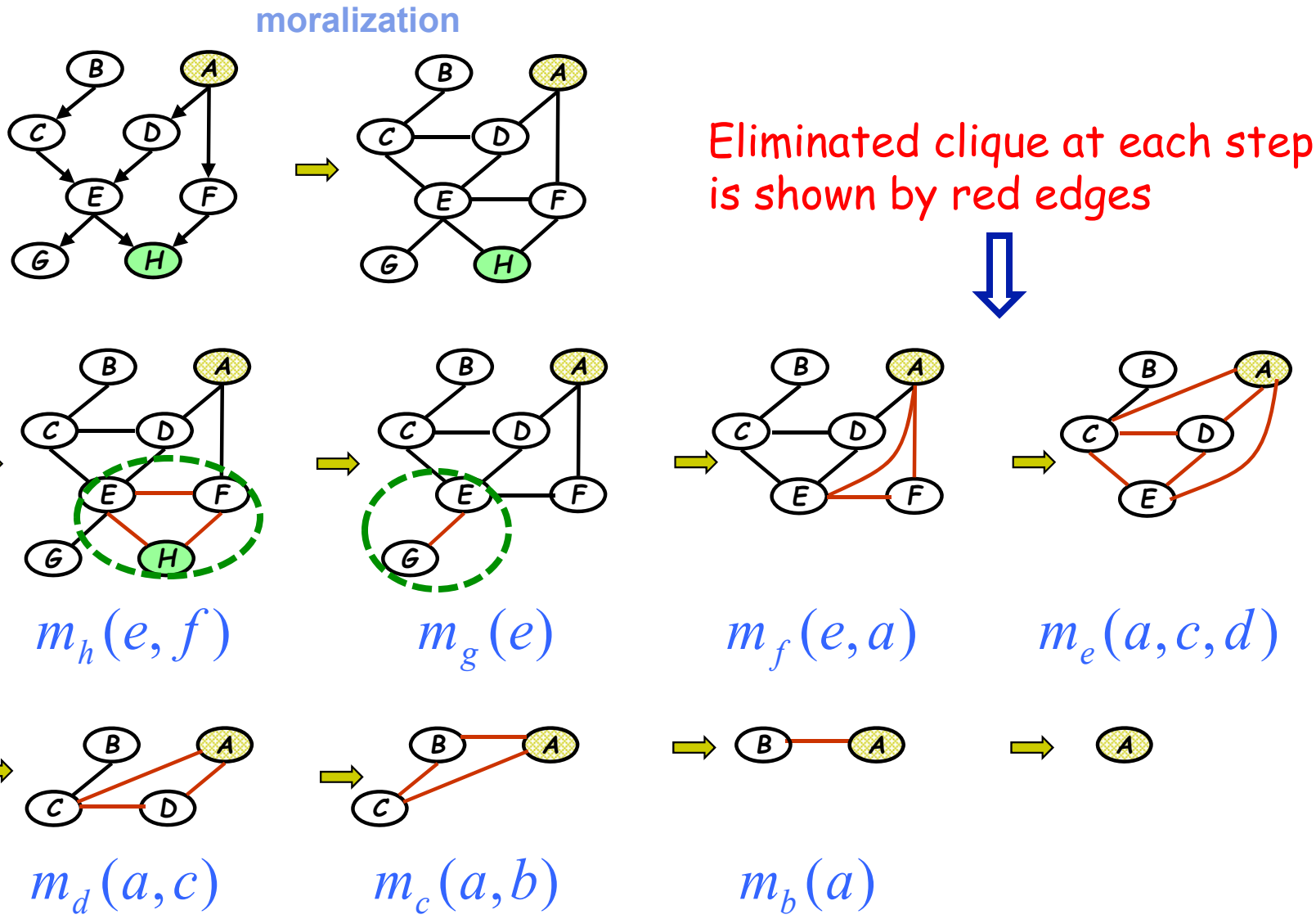
$m_{EA}(A)$

NO 4-way tables!

Graph Elimination Protocol

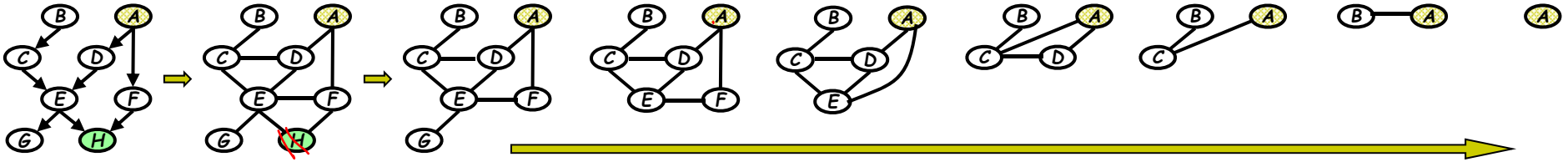
- Start from undirected GM or moralized BN
- Let $G(V, E)$ be Graph with elimination ordering set I
- Eliminate next node in ordering I
  - Removing the node from the graph
  - Connect remaining neighbors of the eliminated node
- The resulting (induced) graph $G'(V', E')$
  - Retain the edges that were created during the elimination procedure
  - Graph-theoretic property: the factors resulted during variable elimination are captured by recording the elimination clique

Example: Graph Elimination



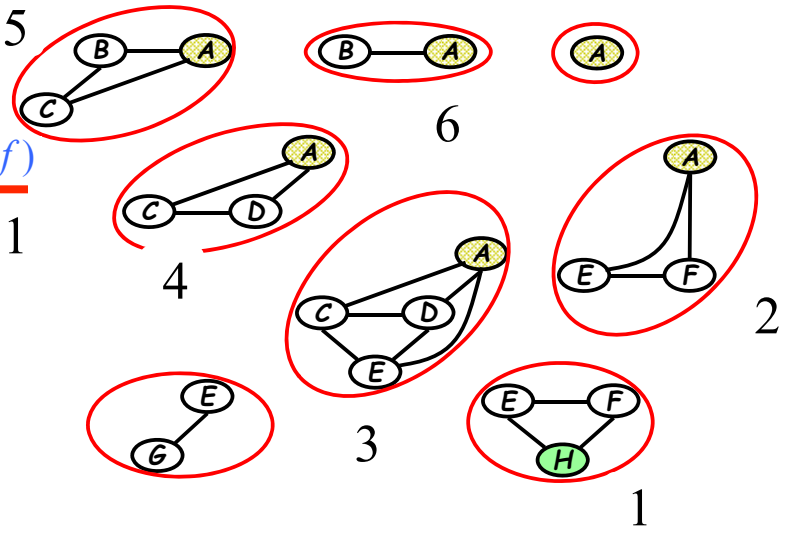
From Variable Elimination to Graph Elimination

- A graph elimination algorithm



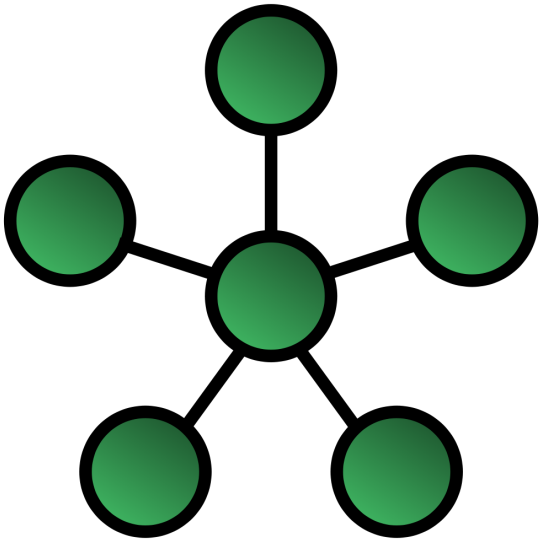
Intermediate terms correspond to the cliques resulted from elimination

$$P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f) \quad 1$$
$$P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)m_h(e,f) \quad 2$$
$$P(a)P(b)P(c|b)P(d|a)P(e|c,d)m_f(a,e) \quad 3$$
$$P(a)P(b)P(c|b)P(d|a)m_e(a,c,d) \quad 4$$
$$P(a)P(b)P(c|b)m_d(a,c) \quad 5$$
$$P(a)P(b)m_c(a,b) \quad 6$$
$$P(a)m_b(a)$$

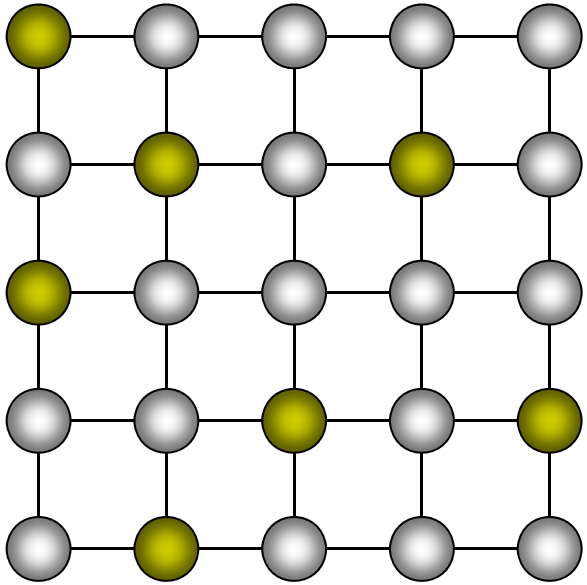


All Eliminated cliques at each step

Examples: What is the ordering?



Star Model



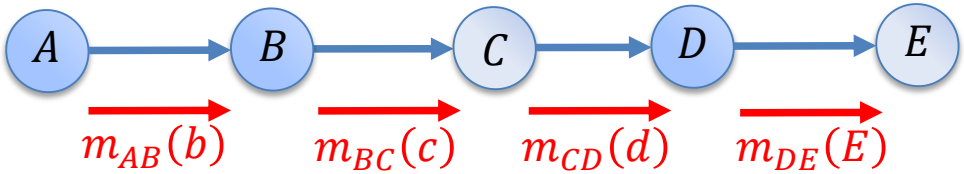
Ising Model

From Variable Elimination to Message Passing

- Recall Variable Elimination Algorithm
  - Choose an ordering in which the query node f is the final node
  - Eliminate node i by removing all potentials containing i, take sum/product over x_i
  - Place the resultant factor back

Query in a Chain (I)

- Query E



- Nice localization in computation
  - $P(E) = \sum_d \sum_c \sum_b \sum_a P(a)P(b|a)P(c|b)P(d|c)P(E|d)$
  - $P(E) = \sum_d P(E|d)(\sum_c P(d|c)\ (\sum_b P(c|b)(\sum_a P(b|a)P(a))))$

$m_{AB}(b)$

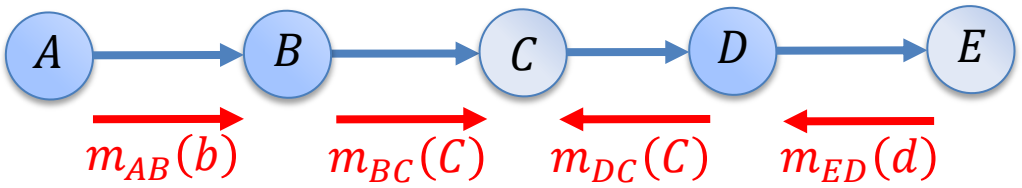
$m_{BC}(c)$

$m_{CD}(d)$

$P(E) = m_{DE}(E)$

Query in a Chain (II)

- Query C



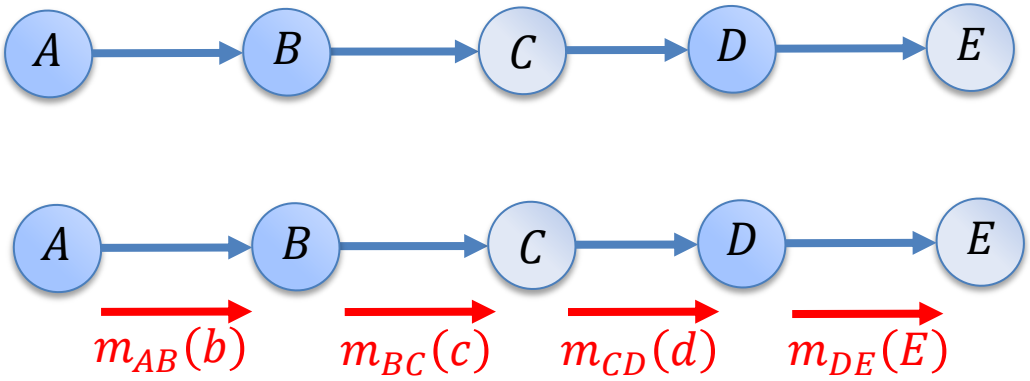
- Start elimination away from the query variable
  - $P(C) = \sum_d \sum_e \sum_b \sum_a P(a)P(b|a)P(c|b)P(d|c)P(e|d)$
  - $P(C) = (\sum_d P(d|C)(\sum_e P(e|d)))\ (\sum_b P(C|b)(\sum_a P(b|a)P(a)))$

$m_{DE}(d)$ $\quad$ $m_{AB}(b)$

$m_{DC}(C)$ $\quad$ $m_{BC}(C)$

$P(C) = m_{DC}(C)m_{BC}(C)$

Chain: Query Everybody (I)

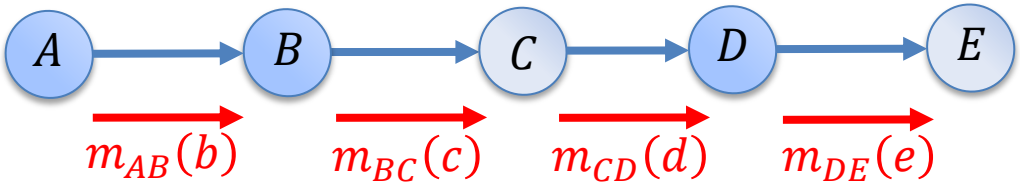


- Computational cost
  - Each message $O(K^2)$
  - Chain length is L
  - Cost for each query is about $O(LK^2)$
  - For L queries, cost is about $O(L^2K^2)$

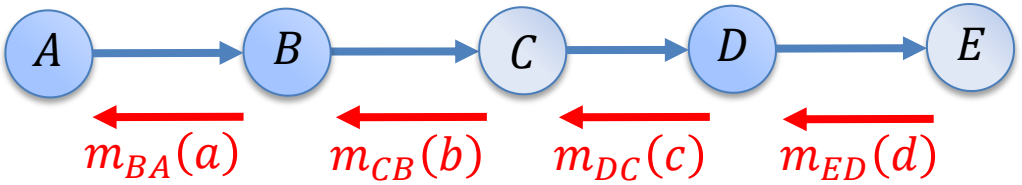
Chain: Query Everybody (II)

- Compute and cache the $2(L-1)$ unique messages

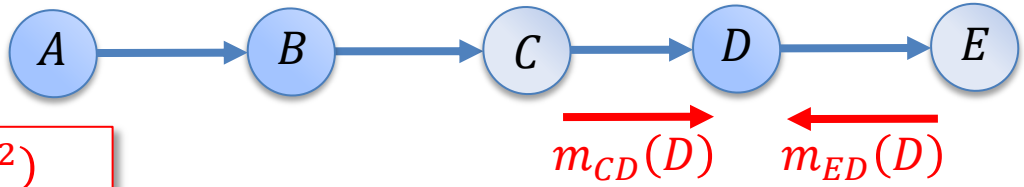
Forward pass:



Backward pass:



- In query time, just multiply together the messages from the neighbors
  - eg. $P(D) = m_{CD}(D)m_{ED}(D)$



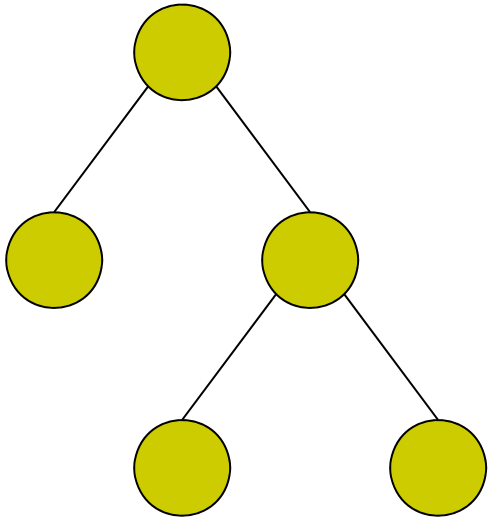
For all queries, $O(2LK^2)$

What do we know so far?

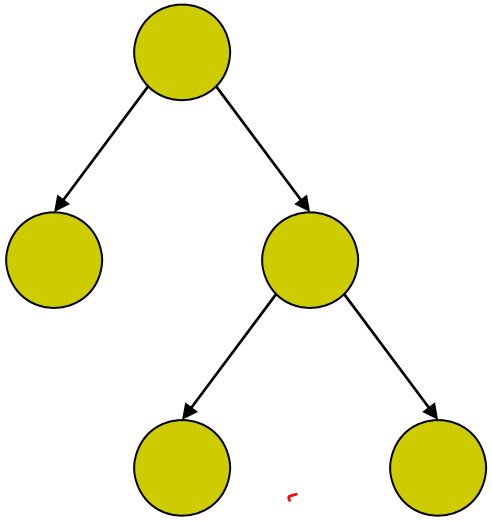
- Chain (directed/undirected)
  - forward-backward algorithm to find all queries.

- Can we generalize to other graphs? (trees, loopy graphs?)

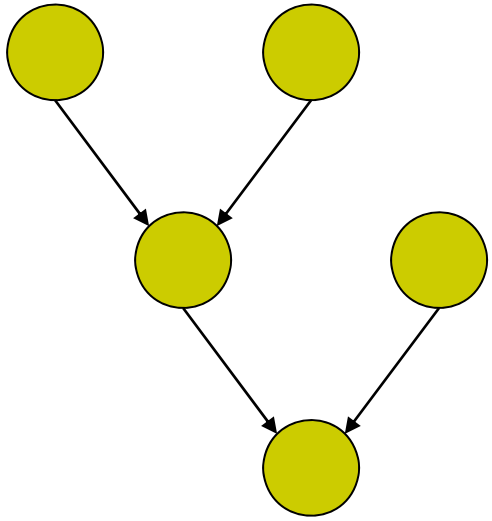
Tree Graphical Models



Undirected tree:
a unique path between any node pair



Directed tree:
all nodes (except root) have only one parent



Poly tree:
multiple parents
(later subject)

Equivalence of Directed and Undirected Trees

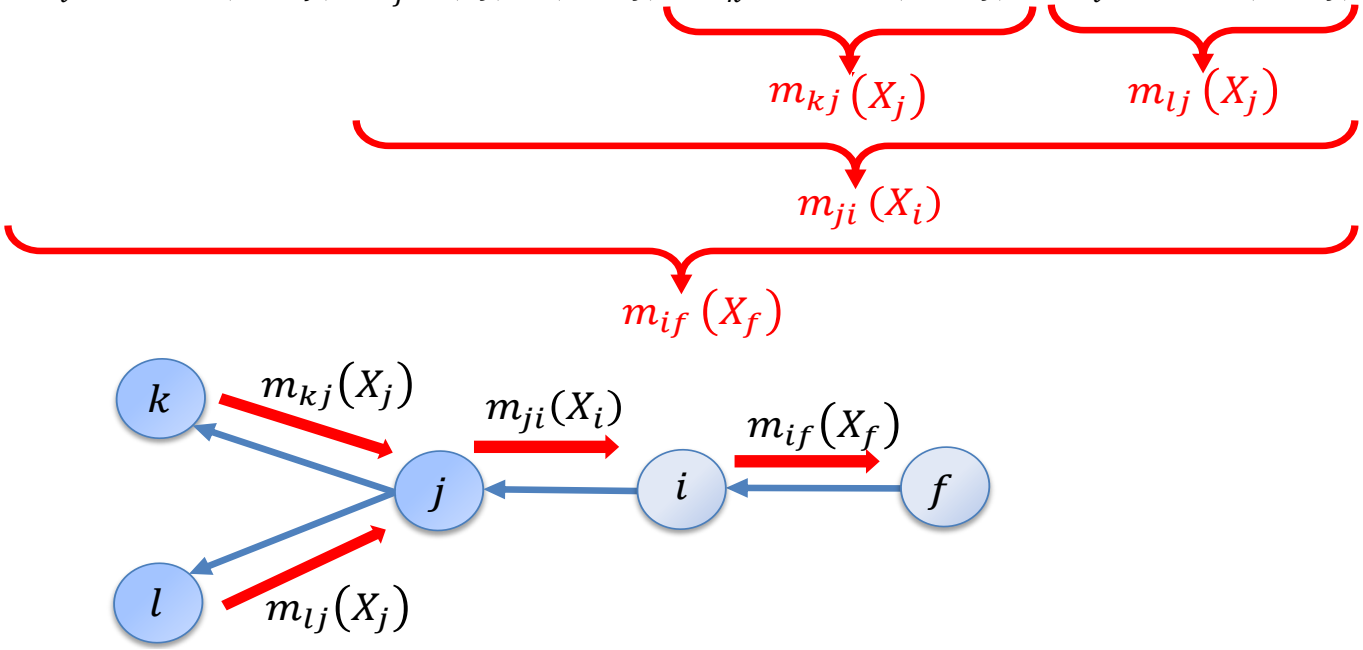
- Any undirected tree can be converted to a directed tree by choosing a root node and directing all edges away from it
- A directed tree and the corresponding undirected tree make the conditional independence assertions
- Parameterization are essentially the same
  - Undirected tree: $P(X) = \frac{1}{Z}\prod_{i\in \mathrm{V}} \Psi(X_i)\prod_{(i,j)\in \mathrm{E}} \Psi(X_i, X_j)$
  - Directed tree: $P(X) = P(X_r)\prod_{(i,j)\in E} P(X_j|X_i)$
  - Equivalence: $\Psi(X_i) = P(X_r), \Psi\left(X_i, X_j\right) = P\left(X_j\middle|X_i\right), Z = 1, \Psi(X_i) = 1$

Message Passing (Belief Propagation, Sum-Product)

- For a Tree graphical model:
  - Choose query node f as the root of the tree
  - View tree as a directed tree with edges pointing towards f
  - Elimination of each node can be considered as message-passing directly along tree branches, rather than on some transformed graphs
  - Use the tree itself as a data-structure to inference

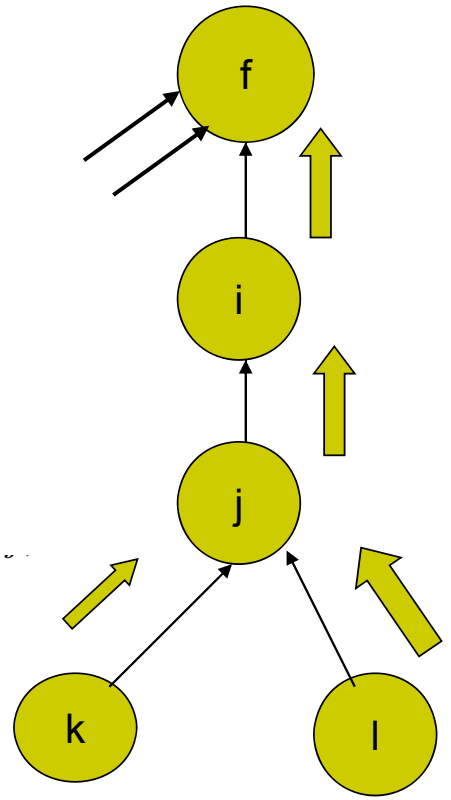
Message Passing (MP) on Trees

- Choose an ordering Z in which query node f is the final node (i.e., root)
- Place all potentials on an active list
  - Message passed along tree edges
  - $P(X_i, X_j, X_k, X_l, X_f) \propto$
    $\Psi(X_i)\,\Psi(X_j)\Psi(X_k)\Psi(X_l)\Psi(X_f)\Psi(X_i, X_j)\Psi(X_k, X_j)\Psi(X_l, X_j)\Psi(X_i, X_f)$
  - $P(f) =$
    $\Psi(X_f)\sum_{x_i}(\Psi(X_i)\Psi(X_i, X_f)\sum_{x_j}\Psi(X_j)\Psi(X_i, X_j)(\sum_{x_k}\Psi(X_k)\Psi(X_k, X_j))(\sum_{x_l}\Psi(X_l)\Psi(X_l, X_j)))$



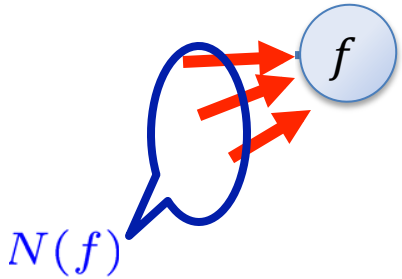
Message Formation on Trees

Let $m_{ji}(x_i)$ denote the factor resulting from eliminating variables from bellow up to i, which is a function of x_i (i.e., message sent from j to i):



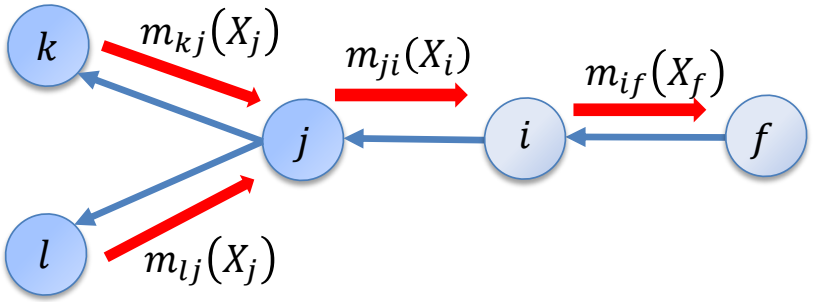
Root (query node)

Finally: $$p(x_f) \propto \psi(x_f) \prod_{e \in N(f)} m_{ef}(x_f)$$

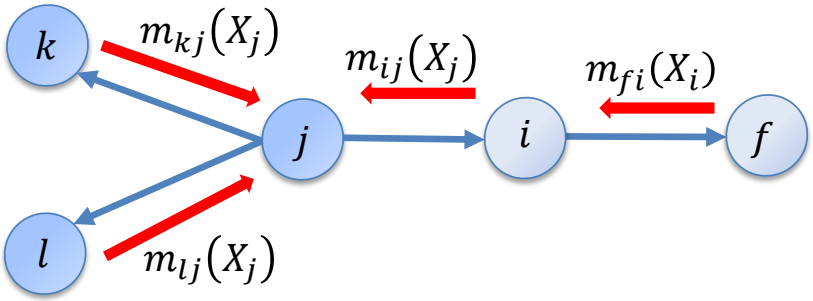


All Queries

Query f



Query j

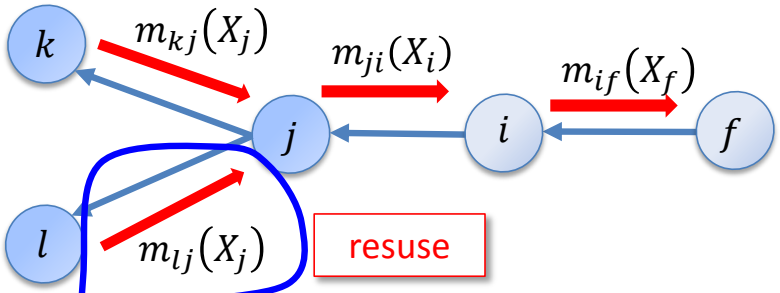


- Performing separate query for each:
  - Each message $O(K^2)$
  - Number of edges is L
  - Cost for each query is about $O(LK^2)$
  - For L queries, cost is about $O(L^2K^2)$

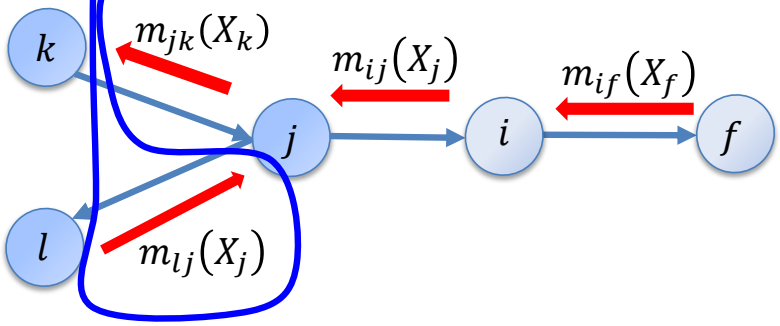
-Song

Forward-Backward Algorithm in Trees

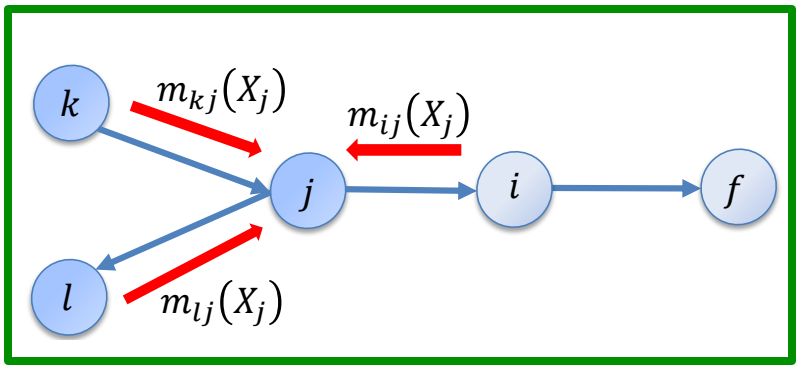
- Forward: pick one leave as root, compute all messages, cache



- Backward: pick another root, compute all messages, cache



Query j

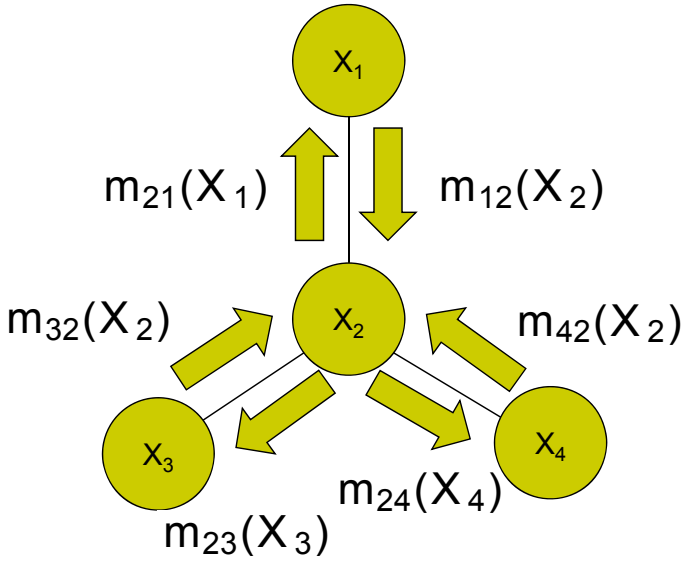


Complexity:

- Compute forward and backward messages for each edge, save them
- $2L$ unique messages
- Total Cost for all queries is about $O(2LK^2)$

Two-Pass MP on Trees

Essentially, a two-pass message passing algorithm over all edges of trees will provide all info we need to answer all queries for all node marginals.



- **Correctness of BP on tree:**

Theorem: The Message Passage (MP) guarantees obtaining all marginals in the tree correctly.

For non-trees: no proof of convergence, no guarantee of correctness!

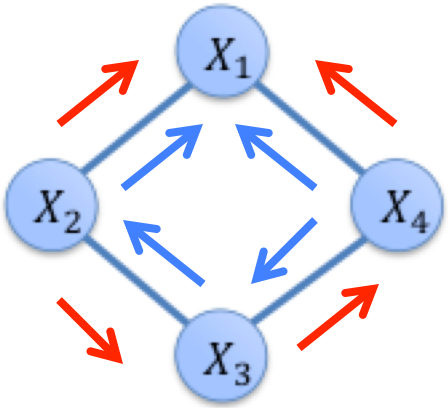
Message Passing Schedule

- Synchronous update:
  - X_j can send message to X_i when all incoming messages from $N(j)\backslash i$ arrive
  - Slow but provably correct for tree, may converge for loopy graphs as well.

- Asynchronous update:
  - X_j can send message to X_i when there is a change in any incoming messages from $N(j)\backslash i$
  - Fast, but not easy to prove convergence. However, empirically it often works

General Graphs?

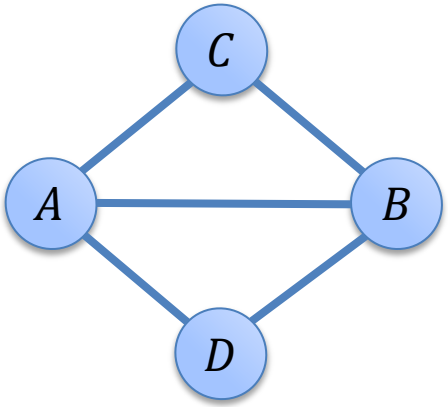
- Loopy graph is more complicated
  - Different elimination order results in different computational cost
  - Message passing on different paths results inconsistent inference outcomes



  - Can we somehow make loopy graph behave like trees? (Sometimes Factor Graphs will do the trick)

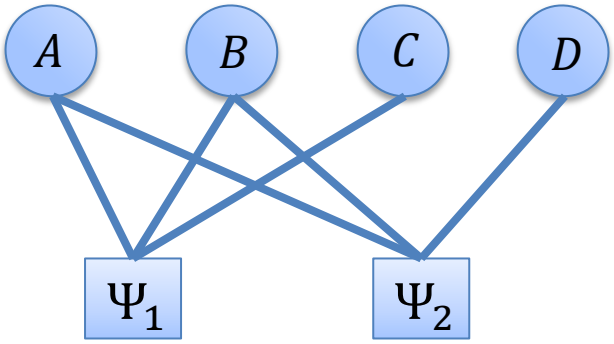
Example of FG for MRF

- Maximal clique specification
  - $P(A,B,C,D) = \frac{1}{Z}\,\Psi_1(A,B,C)\Psi_2(A,B,D)$

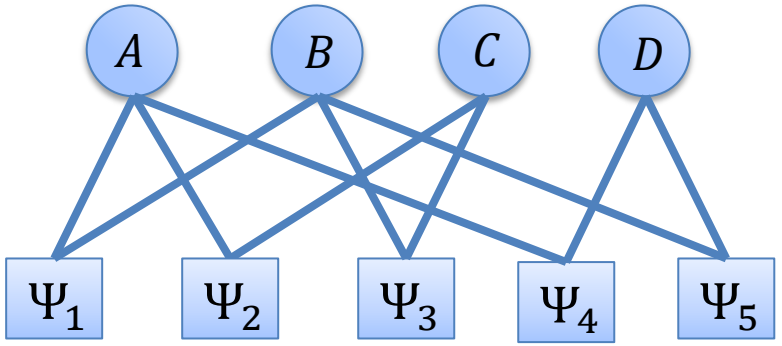


Markov network

- Pairwise Markov Networks
  - $P'(A,B,C,D) = \frac{1}{Z}\Psi(AC)\Psi(BC)\Psi(AB)\Psi(AD)\Psi(BD)$

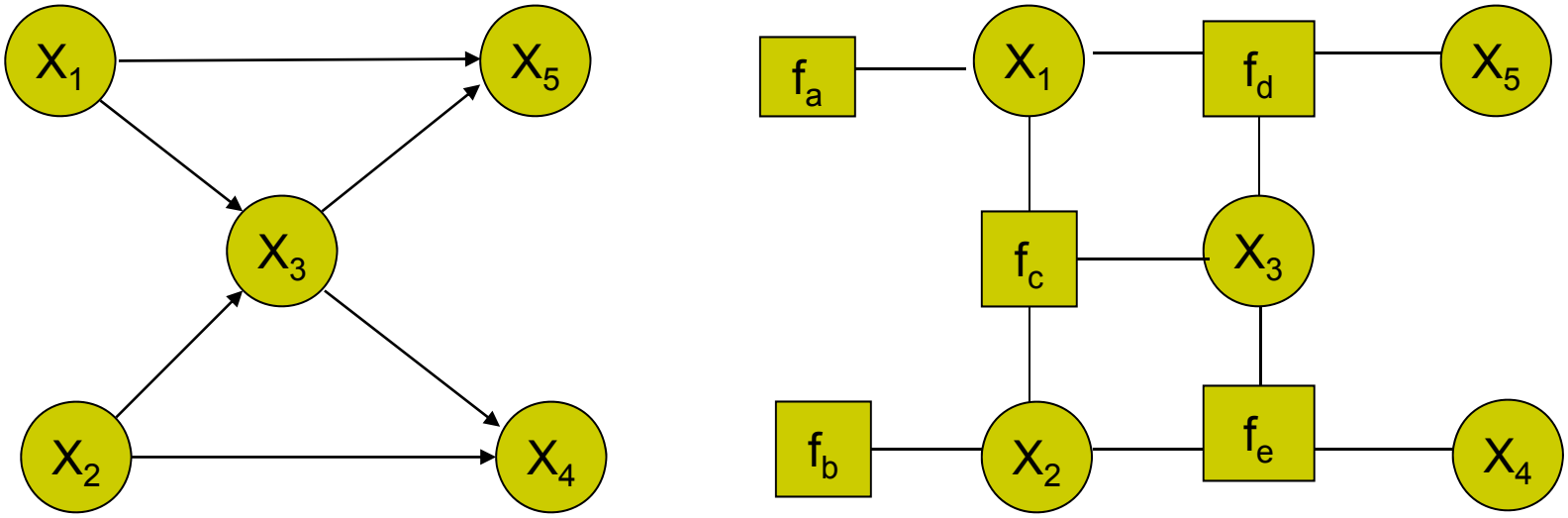


$$\frac{1}{Z}\,\Psi_1(A,B,C)\Psi_2(A,B,D)$$



$$\frac{1}{Z}\Psi_1(AB)\Psi_2(AC)\Psi_3(BC)\Psi_4(AD)\Psi_5(BD)$$

Example of FG for Bayesian Networks



$P(X_1)$ $P(X_2)$ $P(X_3|X_1,X_2)$ $P(X_5|X_1,X_3)$ $P(X_4|X_2,X_3)$

Factors in FG: $f_a(X_1)$ $f_b(X_2)$ $f_c(X_3,X_1,X_2)$ $f_d(X_5,X_1,X_3)$ $f_e(X_4,X_2,X_3)$

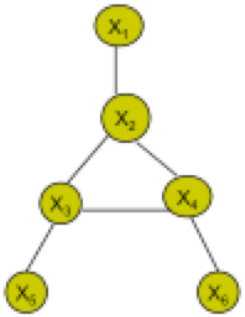
Factor Trees

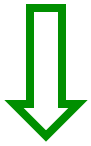
- Factor tree:
  - A Factor graph is a Factor Tree if the undirected graph obtained by ignoring the distinction between variable nodes and factor nodes is an undirected tree

Message Passing on Factor Graph (FG)

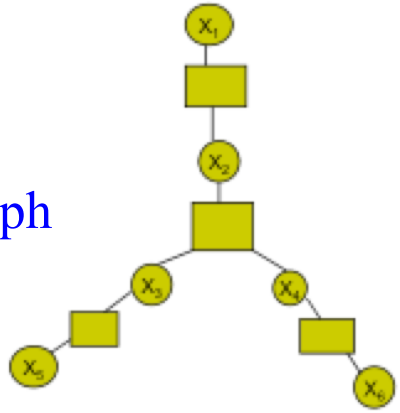
- Because factor graphs turn tree-like graphs to factor trees:
  - Any tree is a data-structure that guarantees correctness of message passing.

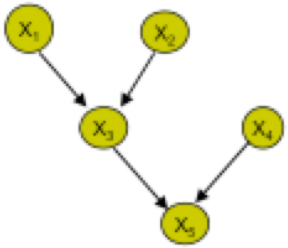
loopy graph





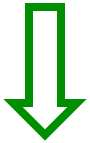
Factor graph



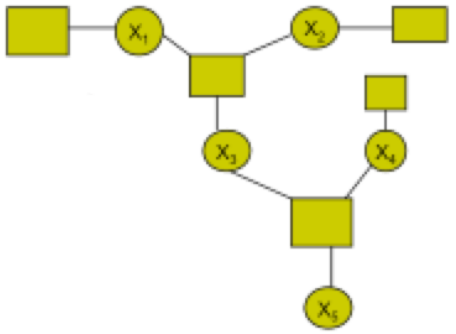


Poly tree

At most one path between any two variables



Factor graph

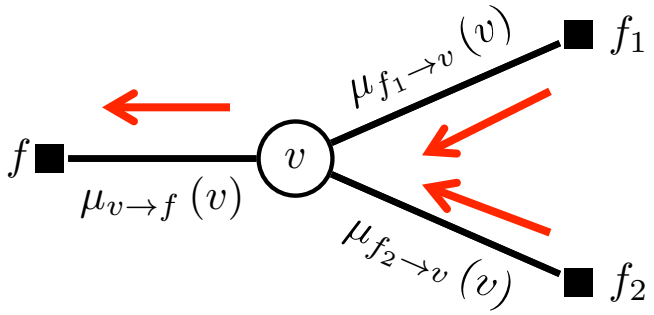


Message Passing on Factor Graph (FG)

- Two kinds of messages

1. **Variable to factor message**

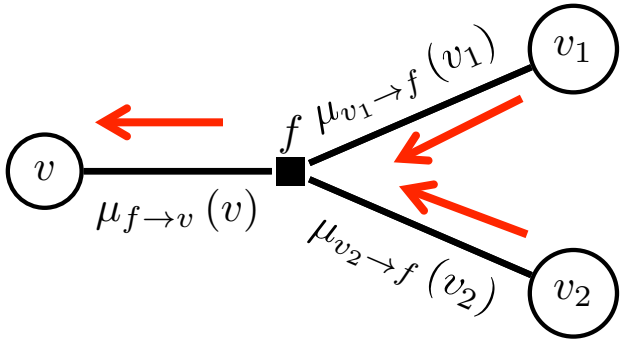
$$\mu_{v\to f}(v) = \prod_{f_i\sim v\setminus f} \mu_{f_i\to v}(v)$$



Messages from extremal variables are set to 1

2. **Factor to variable message**

$$\mu_{f\to v}(v) = \sum_{\{v_i\}} f(v,\{v_i\}) \prod_{v_i\sim f\setminus v} \mu_{v_i\to f}(v_i)$$



Messages from extremal factors are set to the factor

Marginal

$p(v) \propto \prod_{f_i\sim v} \mu_{f_i\to v}(v)$

Initialization of BP Algorithm over Factor Tree

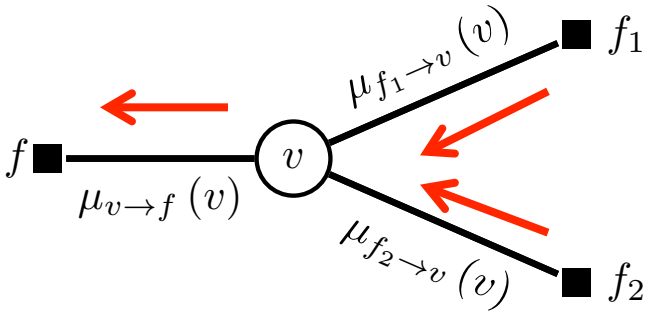
- Initialization at the leaf of the tree:
  - Set every variable node values at leaf of the tree to "1"
  - Set each factor node at the leaf of the tree to "$f(x)$". That is, if the factor node (with factor term "$f(x)$") send a message to the variable node x_i, then set the message as "$f(x_i)$",

Max Product on Factor Graph (FG): Computing MPE

- Two kinds of messages

1. Variable to factor message

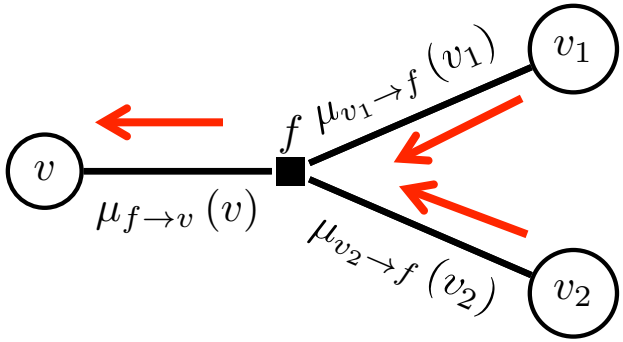
$$\mu_{v\to f}(v) = \prod_{f_i\sim v\setminus f} \mu_{f_i\to v}(v)$$



Messages from extremal variables are set to 1

2. Factor to variable message

$$\mu_{f\to v}(v) = \max_{\{v_i\}} f(v,\{v_i\}) \prod_{v_i\sim f\setminus v} \mu_{v_i\to f}(v_i)$$



Messages from extremal factors are set to the factor

MPE: $v^* = \operatorname{argmax}_v \prod_{f_i\sim v} \mu_{f_i\to v}(v)$

Max Product on Trees: Computing MPE

$$\arg\max_{\mathbf{x}} p(\mathbf{x}), \qquad p(\mathbf{x}) = \frac{1}{Z}\prod_{c\in C}\phi_c(\mathbf{x}_c)$$

(we assume any evidence has been subsumed into the potentials, as discussed in the last lecture)

- Since the normalization term is simply a constant, this is equivalent to

$$\arg\max_{\mathbf{x}} \prod_{c\in C}\phi_c(\mathbf{x}_c)$$

(called the *max-product* inference task)

- Furthermore, since log is monotonic, letting $\theta_c(\mathbf{x_c}) = \lg \phi_c(\mathbf{x_c})$, we have that this is equivalent to

$$\arg\max_{\mathbf{x}} \sum_{c\in C}\theta_c(\mathbf{x}_c)$$

(called *max-sum*)

Computing MPE on Trees

- Compare the sum-product problem with the max-product (equivalently, max-sum in log space):

$$\text{sum-product} \qquad \sum_{\mathbf{x}} \prod_{c \in C} \phi_c(\mathbf{x}_c)$$

$$\text{max-sum} \qquad \max_{\mathbf{x}} \sum_{c \in C} \theta_c(\mathbf{x}_c)$$

- Can exchange operators $(+, *)$ for $(\max, +)$
- We get "max-product variable elimination" and "max-product belief propagation"

Simple Example of MPE

Example

- Suppose we have a simple chain, $A - B - C - D$, and we want to find the MAP assignment,

$$\max_{a,b,c,d} \phi_{AB}(a,b)\phi_{BC}(b,c)\phi_{CD}(c,d)$$

- Just as we did before, we can push the maximizations inside to obtain:

$$\max_{a,b} \phi_{AB}(a,b) \max_{c} \phi_{BC}(b,c) \max_{d} \phi_{CD}(c,d)$$

  or, equivalently,

$$\max_{a,b} \theta_{AB}(a,b) + \max_{c} \theta_{BC}(b,c) + \max_{d} \theta_{CD}(c,d)$$

- To find the actual maximizing assignment, we do a traceback (or keep back pointers)

Max-product belief propagation (for tree-structured MRFs)

- Same as sum-product BP except that the messages are now:

$$m_{j\to i}(x_i) = \max_{x_j} \phi_j(x_j)\phi_{ij}(x_i, x_j) \prod_{k\in N(j)\setminus i} m_{k\to j}(x_j)$$

- After passing all messages, can compute single node *max-marginals*,

$$m_i(x_i) = \phi_i(x_i) \prod_{j\in N(i)} m_{j\to i}(x_i) \qquad \propto \qquad \max_{\mathbf{x}_{V\setminus i}} p(\mathbf{x}_{V\setminus i}, x_i)$$

- If the MAP assignment $\mathbf{x}^*$ is **unique**, can find it by locally decoding each of the single node max-marginals, i.e.

$$x_i^* = \arg\max_{x_i} m_i(x_i)$$

Exactly solving MPE, beyond trees

- a discrete optimization problem

$$\arg\max_{\mathbf{x}} \sum_{i \in V} \theta_i(x_i) + \sum_{ij \in E} \theta_{ij}(x_i, x_j)$$

- Very general discrete optimization problem – many hard combinatorial optimization problems can be written as this (e.g., 3-SAT)
- Studied in operations research communities, theoretical computer science, AI (constraint satisfaction, weighted SAT), etc.