Approximate inference using loopy belief propagation

Yair Weiss
UC Berkeley
www.cs.berkeley.edu/ ~yweiss

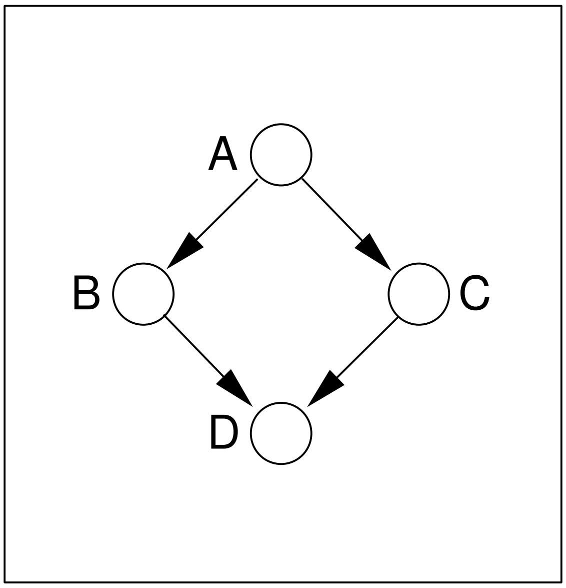
Collaborators: Bill Freeman (MERL), Jonathan Yedidia (MERL), Kevin Murphy (UC Berkeley), Mike Jordan (UC Berkeley)

Outline

- Exact inference using belief propagation.
- The need for approximation.
- Empirical success of loopy BP (and some failures).
- Theoretical results on loopy BP.
- Beyond ordinary BP.

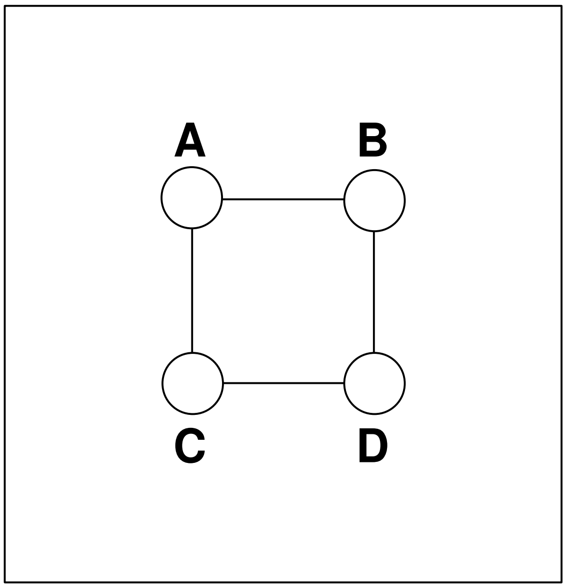
Graphical Models

Bayes nets



$\prod_x P(x|par(x))$

Markov Nets

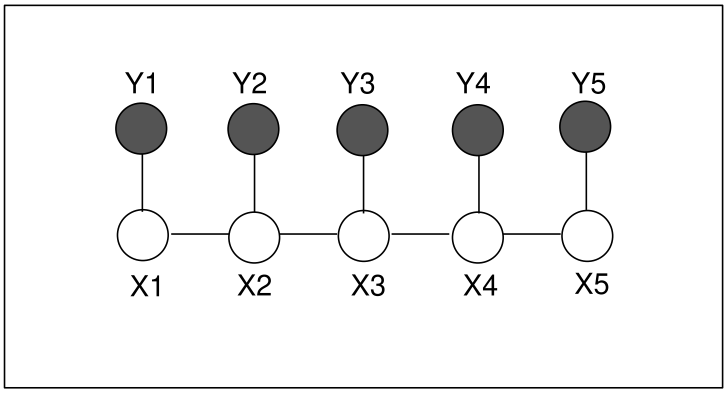


$\prod_c \Psi(x_c)$

Nodes: random variables.

Edges: probabilistic constraints between variables.

Inference in Graphical Models



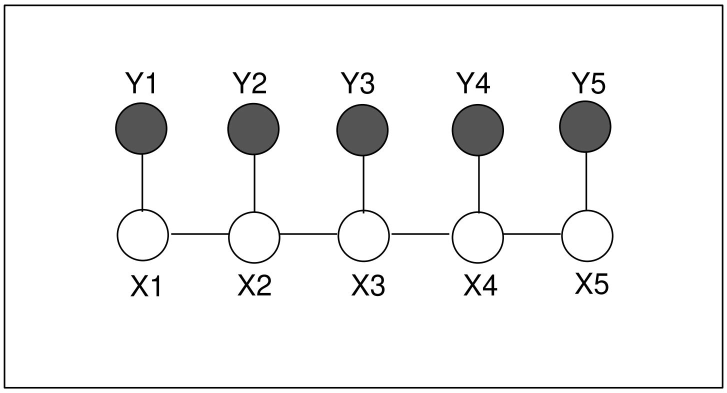
1. Marginals. Find $P(X_i = x_i|Y)$.

2. MAP assign. Find $\{x_i^*\}$ such that $P(\{X_i = x_i^*\}|Y)$ is max.

3. MM assign. Find $\{x_i^*\}$ such that $P(X_i = x_i^*|Y)$ is max for each i.

4. Likelihood. Calculate $P(Y)$.

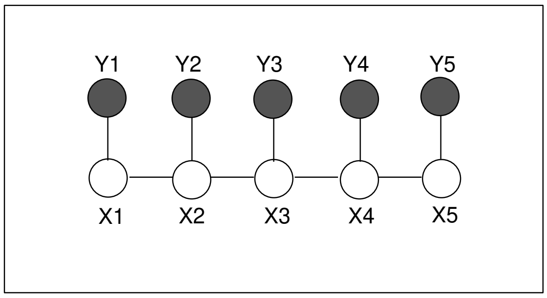
The forward-backward algorithm for HMMs



$$
\begin{aligned}
P(x_t|y) &\propto \alpha_t(x_t)\beta_t(x_t)P(y_t|x_t) \\
\alpha_t(x_t) &= P(x_t, y_{1,t-1}) \\
\beta_t(x_t) &= P(y_{t+1,T}|x_t)
\end{aligned}
$$

$$
\begin{aligned}
\beta_t(x_t) &= P(y_{t+1,T}|x_t) \\
&= \sum_{x_{t+1}} P(x_{t+1}, y_{t+1,T}|x_t) \\
&= \sum_{x_{t+1}} P(x_{t+1}|x_t)P(y_{t+1}|x_{t+1})\beta_{t+1}(x_t) \\
\alpha_t(x_t) &= \sum_{x_{t-1}} P(x_t|x_{t-1})P(y_{t-1}|x_{t-1})\alpha_{t-1}(x_t)
\end{aligned}
$$

Belief Propagation for undirected trees



Input: Graph, $\Psi_{ij}(x_i, x_j)$, $\Psi_{ii}(x_i, y_i)$.

Define:

$m_{ij}(x_j)$ message that X_i sends to X_j.

$b_i(x_i)$ "belief" at node X_i.

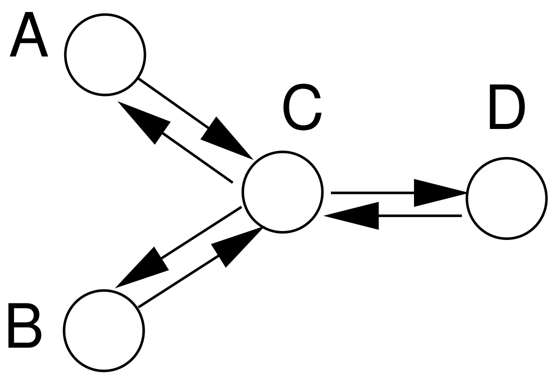
Iterate:

$$m_{ij}(x_j) \leftarrow \alpha \sum_{x_i} \Psi_{ij}(x_i, x_j) \Psi_i(x_i) \prod_{X_k \in N(X_i) \backslash X_j} m_{ki}(x_i)$$

finally:

$$b_i(x_i) \leftarrow \alpha \prod_{X_j \in N(X_i)} m_{ji}(x_i)$$

BP Dynamics:

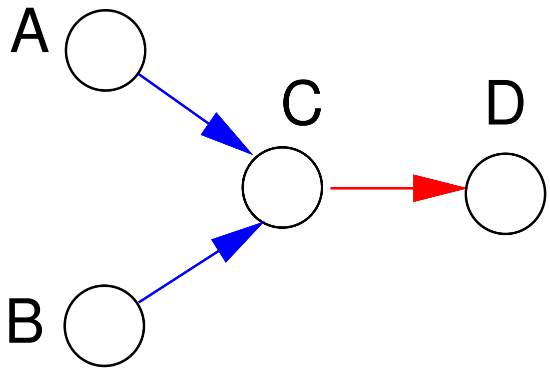


$$m_{ij}(x_j) \leftarrow \alpha \sum_{x_i} \Psi_{ij}(x_i, x_j) \Psi_i(x_i) \prod_{X_k \in N(X_i) \setminus X_j} m_{ki}(x_i)$$

A parallel message-passing algorithm.

- Every node sends a probability density to its neighbors.
- message to neighbor depends on messages received from all the *other neighbors*

BP Dynamics



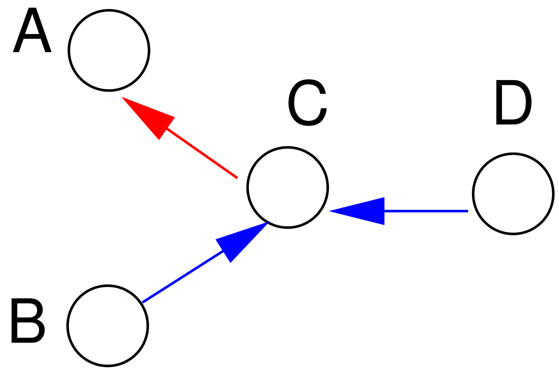
$$m_{ij}(x_j) \leftarrow \alpha \sum_{x_i} \Psi_{ij}(x_i, x_j) \Psi_i(x_i) \prod_{X_k \in N(X_i) \setminus X_j} m_{ki}(x_i)$$

A parallel message-passing algorithm.

- Every node sends a probability density to its neighbors.

- message to neighbor depends on messages received from all the *other neighbors*

BP Dynamics:



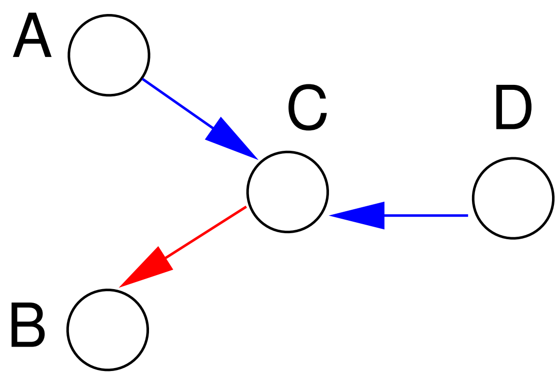
$$m_{ij}(x_j) \leftarrow \alpha \sum_{x_i} \Psi_{ij}(x_i, x_j) \Psi_i(x_i) \prod_{X_k \in N(X_i) \setminus X_j} m_{ki}(x_i)$$

A parallel message-passing algorithm.

- Every node sends a probability density to its neighbors.

- message to neighbor depends on messages received from all the *other neighbors*

BP Dynamics

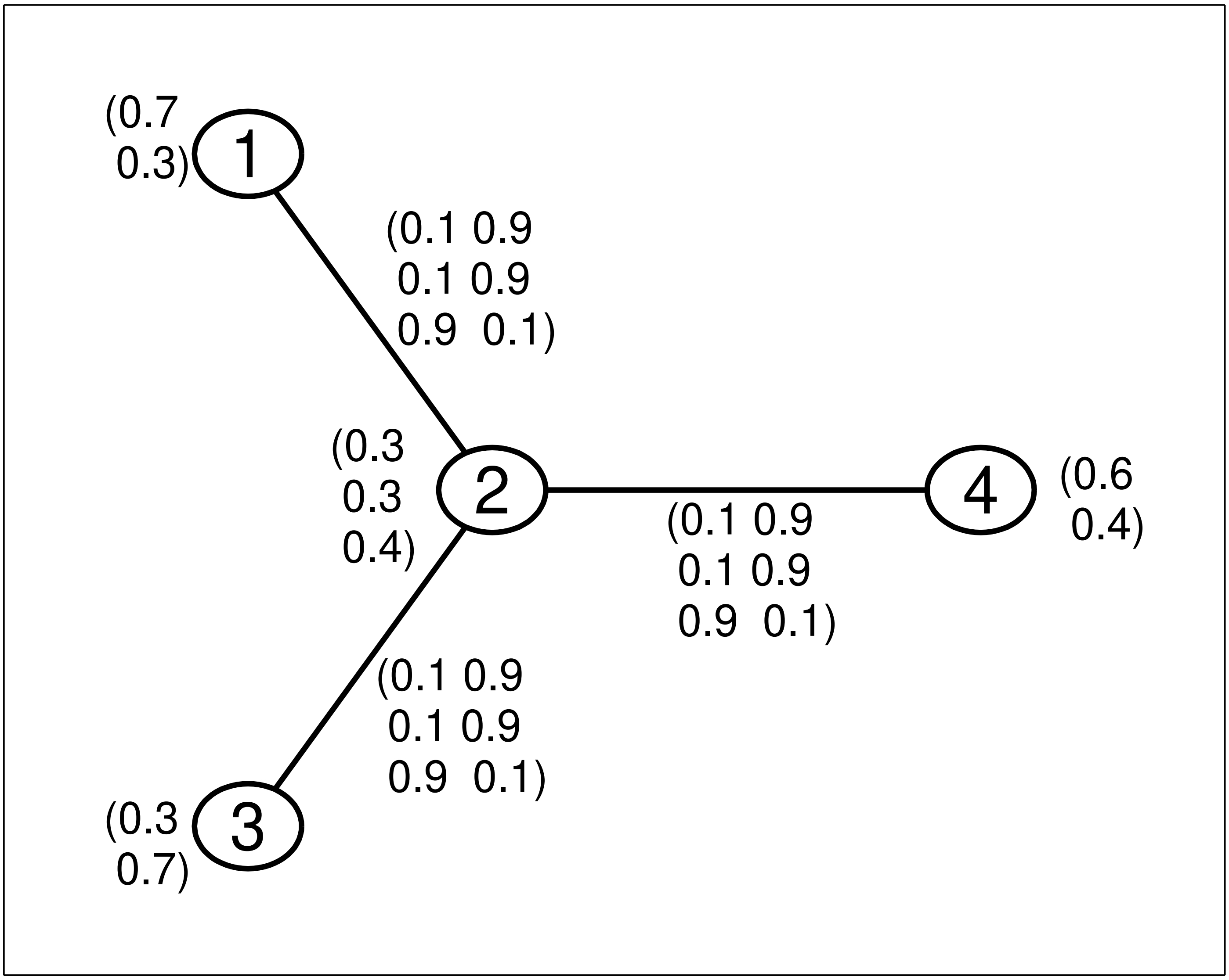


$$m_{ij}(x_j) \leftarrow \alpha \sum_{x_i} \Psi_{ij}(x_i, x_j) \Psi_i(x_i) \prod_{X_k \in N(X_i) \setminus X_j} m_{ki}(x_i)$$

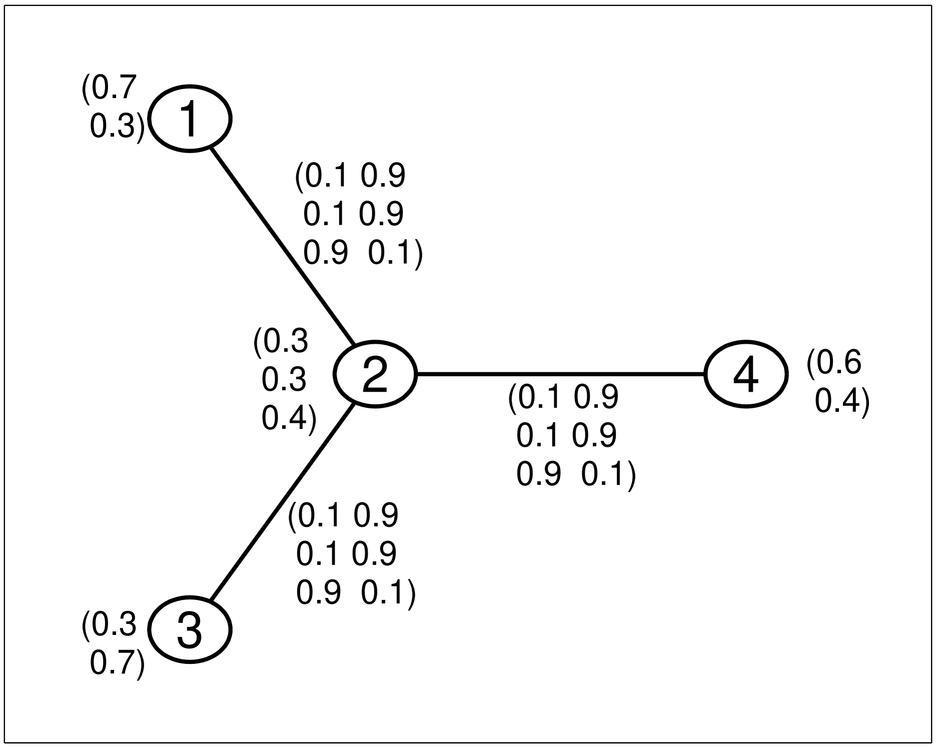
A parallel message-passing algorithm.

- Every node sends a probability density to its neighbors.
- message to neighbor depends on messages received from all the *other neighbors*

Numerical Example:



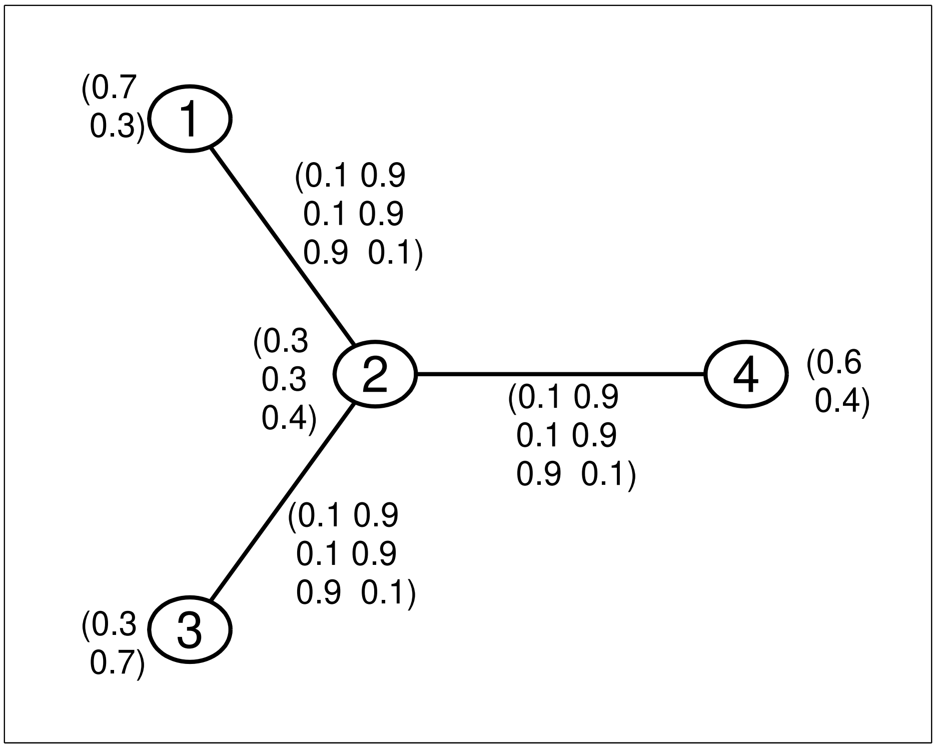
Numerical Example (cont):



$$m_{42} = \alpha \begin{pmatrix} 0.1 & 0.9 \\ 0.1 & 0.9 \\ 0.9 & 0.1 \end{pmatrix} \begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} = \begin{pmatrix} 0.2958 \\ 0.2958 \\ 0.4085 \end{pmatrix}$$

$$m_{32} = \alpha \begin{pmatrix} 0.1 & 0.9 \\ 0.1 & 0.9 \\ 0.9 & 0.1 \end{pmatrix} \begin{pmatrix} 0.3 \\ 0.7 \end{pmatrix} = \begin{pmatrix} 0.3976 \\ 0.3976 \\ 0.2048 \end{pmatrix}$$

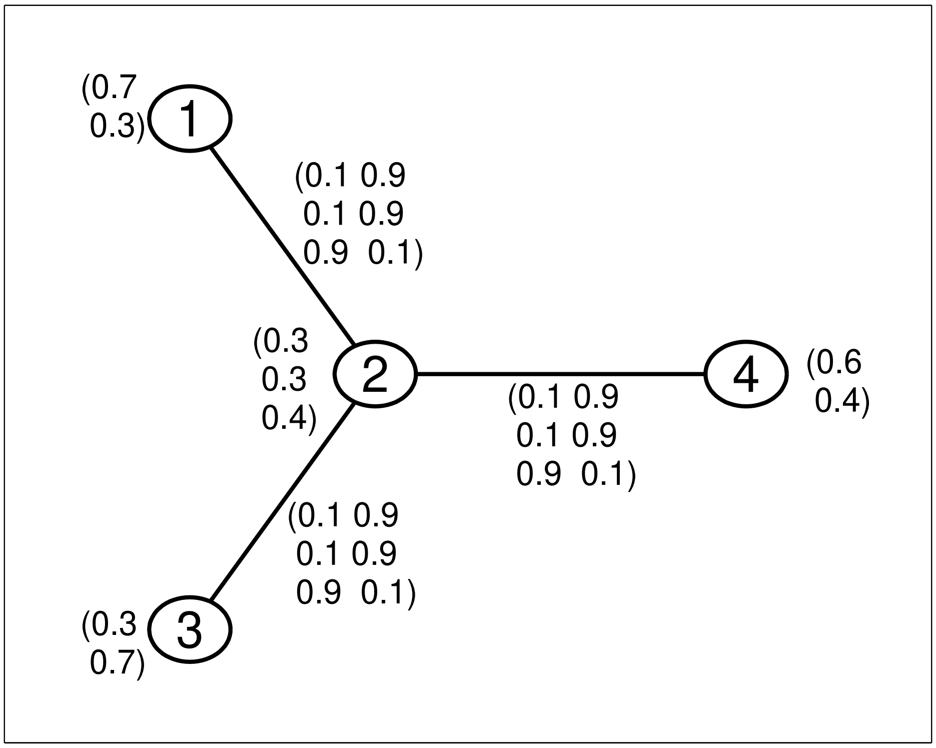
Numerical Example (cont):



$$m_{21} = \begin{pmatrix} 0.1 & 0.1 & 0.9 \\ 0.9 & 0.9 & 0.1 \end{pmatrix} \begin{pmatrix} 0.29 \\ 0.29 \\ 0.40 \end{pmatrix} * \begin{pmatrix} 0.39 \\ 0.39 \\ 0.20 \end{pmatrix} * \begin{pmatrix} 0.3 \\ 0.3 \\ 0.4 \end{pmatrix}$$

$$= \begin{pmatrix} 0.2869 \\ 0.7131 \end{pmatrix}$$

Numerical Example (cont):



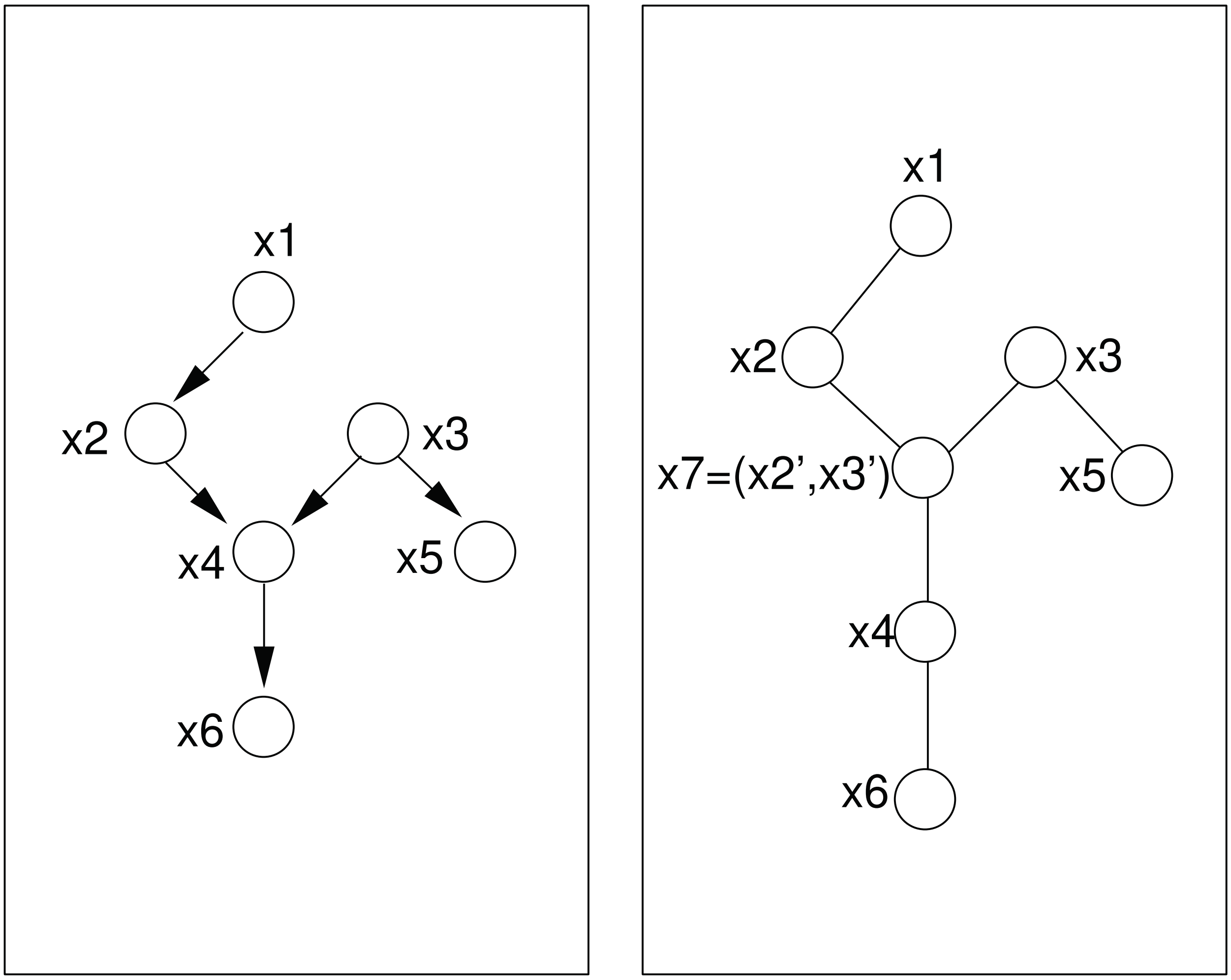
$$b_1 = \alpha \begin{pmatrix} 0.2869 \\ 0.7131 \end{pmatrix} * \begin{pmatrix} 0.7 \\ 0.3 \end{pmatrix} = \begin{pmatrix} 0.484 \\ 0.516 \end{pmatrix}$$

$$\begin{aligned} b_4 &= \alpha \begin{pmatrix} 0.29 \\ 0.29 \\ 0.40 \end{pmatrix} * \begin{pmatrix} 0.39 \\ 0.39 \\ 0.20 \end{pmatrix} * \begin{pmatrix} 0.253 \\ 0.253 \\ 0.492 \end{pmatrix} * \begin{pmatrix} 0.3 \\ 0.3 \\ 0.4 \end{pmatrix} \\ &= \begin{pmatrix} 0.253 \\ 0.253 \\ 0.492 \end{pmatrix} \end{aligned}$$

Belief propagation in directed polytrees

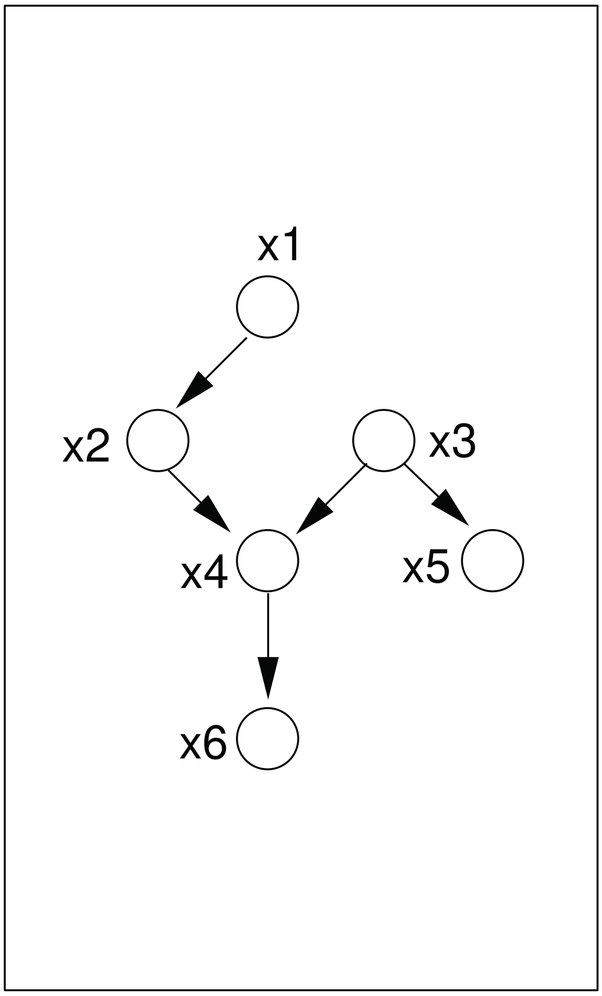
Many equivalent formulations (Including: Kim and Pearl 83, Jensen 94, Shafer and Shenoy 90, Aji and McEliece 97, Frey et al 97).

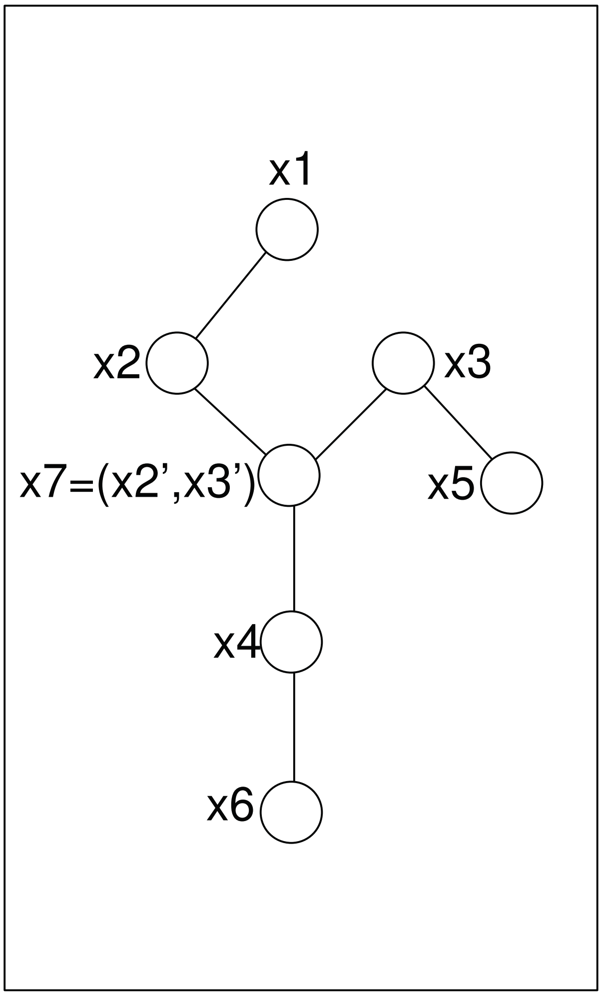
Equivalent to converting to undirected graph and running undirected BP.



Belief propagation in directed polytrees

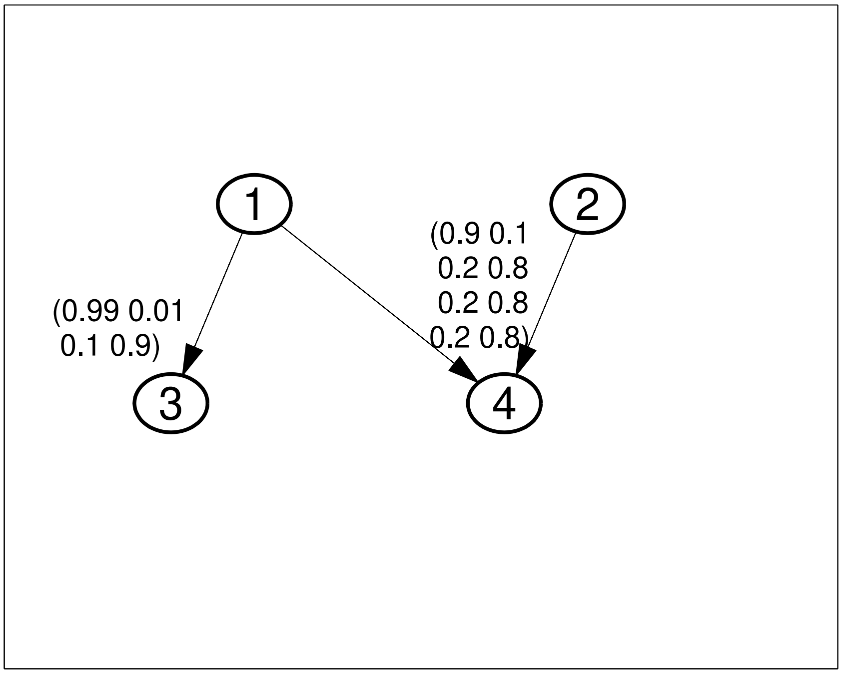
$$\begin{aligned}\Psi_{46}(x_4, x_6) &= P(x_6 | x_4) \\ \Psi_{74}(x_2', x_3', x_4) &= P(x_4 | x_2', x_3') \\ \Psi_{27}(x_2, x_2', x_3') &= \delta(x_2 - x_2')\end{aligned}$$

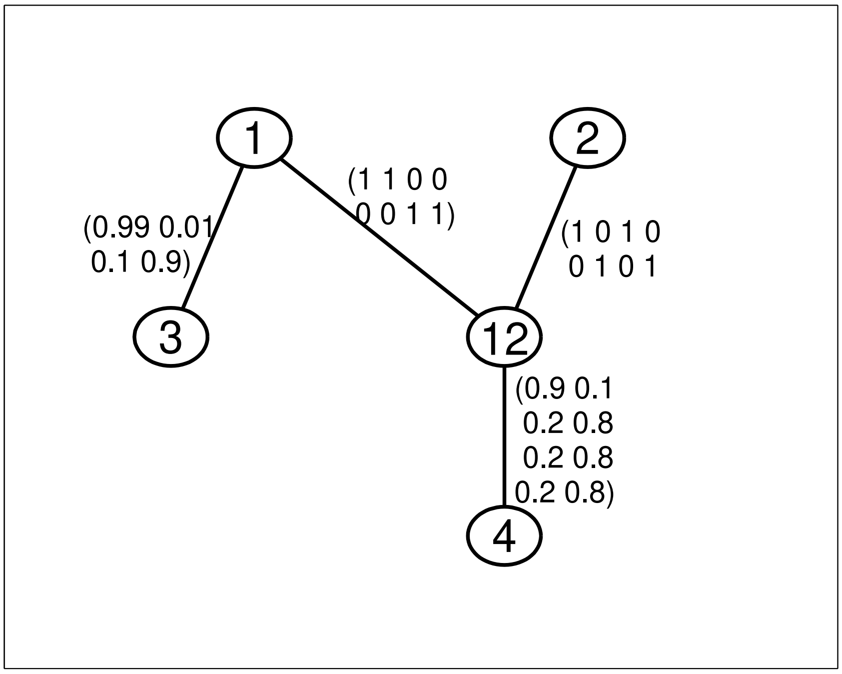




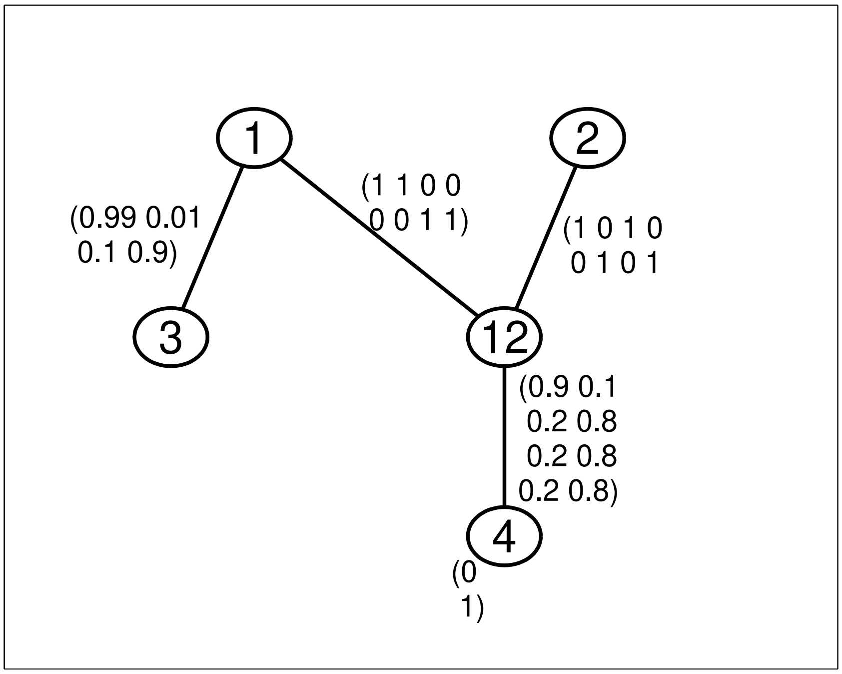
Example:

1:Earthquake,2:Burglary,3:Radio,4:Alarm.



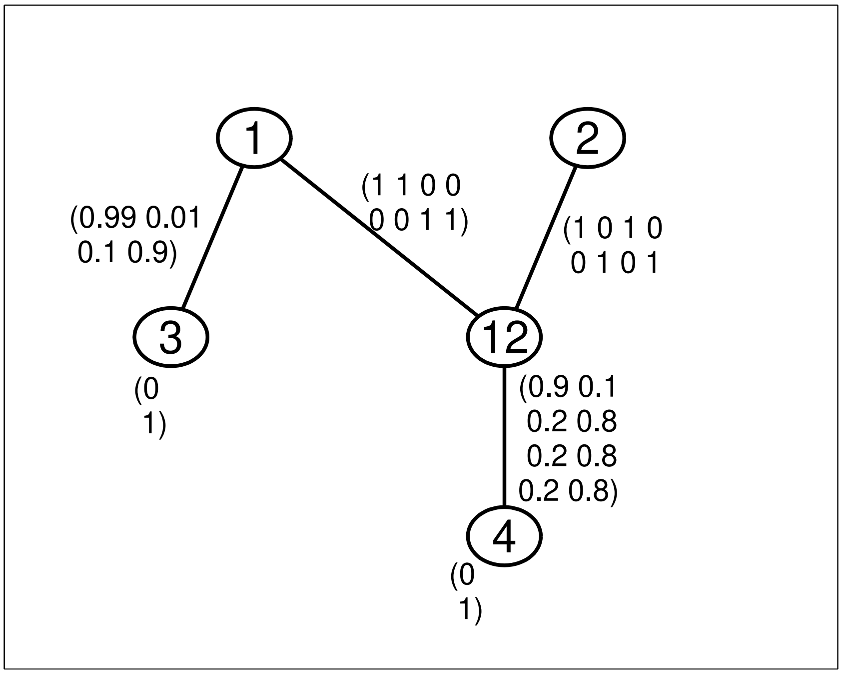


Numerical example (cont)



$$
\begin{aligned}
m_{12,2} &= \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \\ 0.2 & 0.8 \\ 0.2 & 0.8 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} 0.36 \\ 0.64 \end{pmatrix}
\end{aligned}
$$

Explaining away



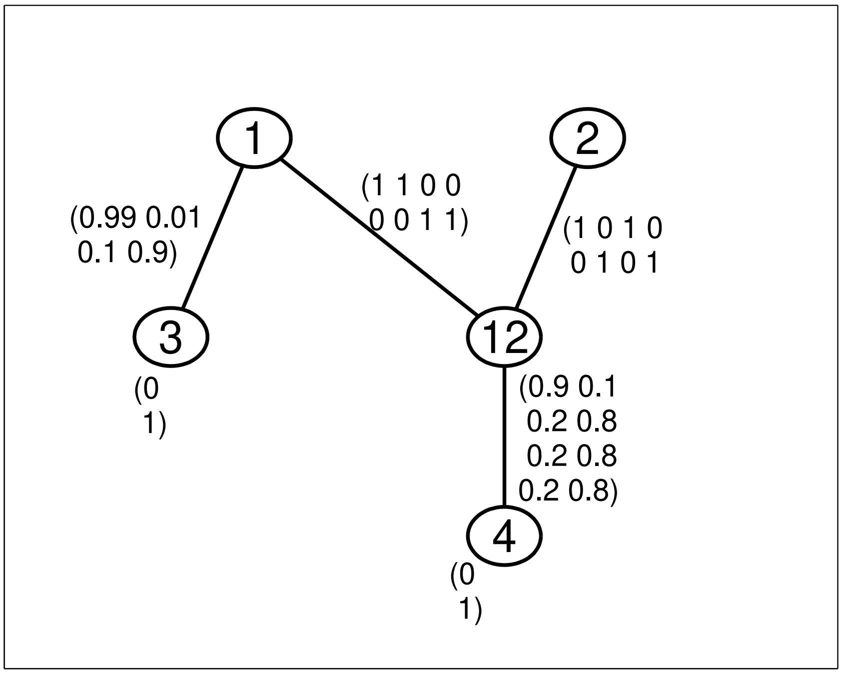
$$m_{1,12} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0.99 & 0.01 \\ 0.1 & 0.9 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0.01 \\ 0.01 \\ 0.49 \\ 0.49 \end{pmatrix}$$

$$\Rightarrow$$

$$m_{12,2} = \begin{pmatrix} 0.49 \\ 0.51 \end{pmatrix}$$

Q: How is this related to Pearl's algorithm?



A: it's exactly the same!

$$m_{12,2}(x_2) = \lambda(x_2).$$

$$m_{12,4}(x_4) = \pi(x_4)$$

Two variants of belief propagation

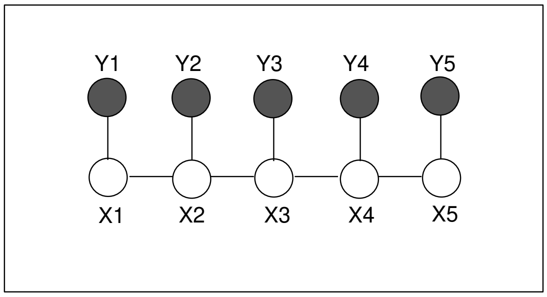
Sum-Product algorithm (a.k.a belief update)

$$m_{ij}(x_j) \leftarrow \alpha \sum_{x_i} \Psi_{ij}(x_i, x_j)\Psi_i(x_i) \prod_{X_k \in N(X_i) \setminus X_j} m_{ki}(x_i)$$

Max-Product algorithm (a.k.a belief revision, Viterbi algorithm, Dawid's algorithm)

$$m_{ij}(x_j) \leftarrow \alpha \max_{x_i} \Psi_{ij}(x_i, x_j)\Psi_i(x_i) \prod_{X_k \in N(X_i) \setminus X_j} m_{ki}(x_i)$$

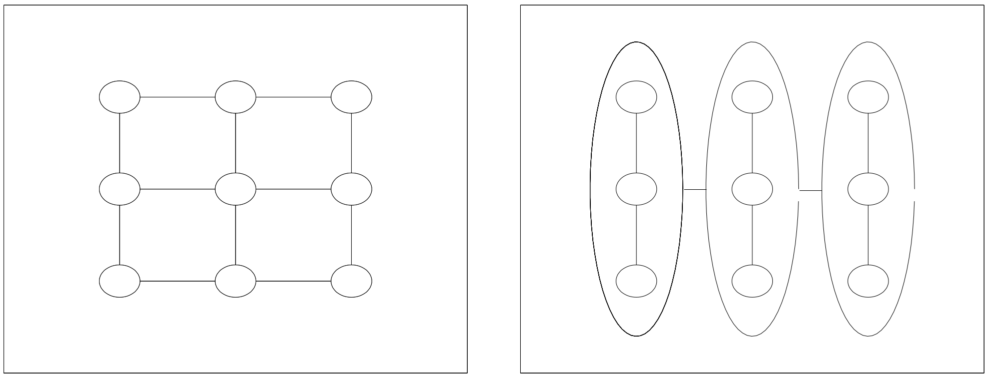
Belief Propagation in trees



When the graph is a tree (Pearl 86):

- both algorithms converge in finite time.
- at convergence: $b_i(x_i) = P(X_i = x_i|Y)$
- max-product assignment ($x_i^* = \max_{x_i} b_i(x_i)$ after convergence) gives MAP assignment.
- sum-product assignment ($x_i^* = \max_{x_i} b_i(x_i)$ after convergence) gives MM assignment.

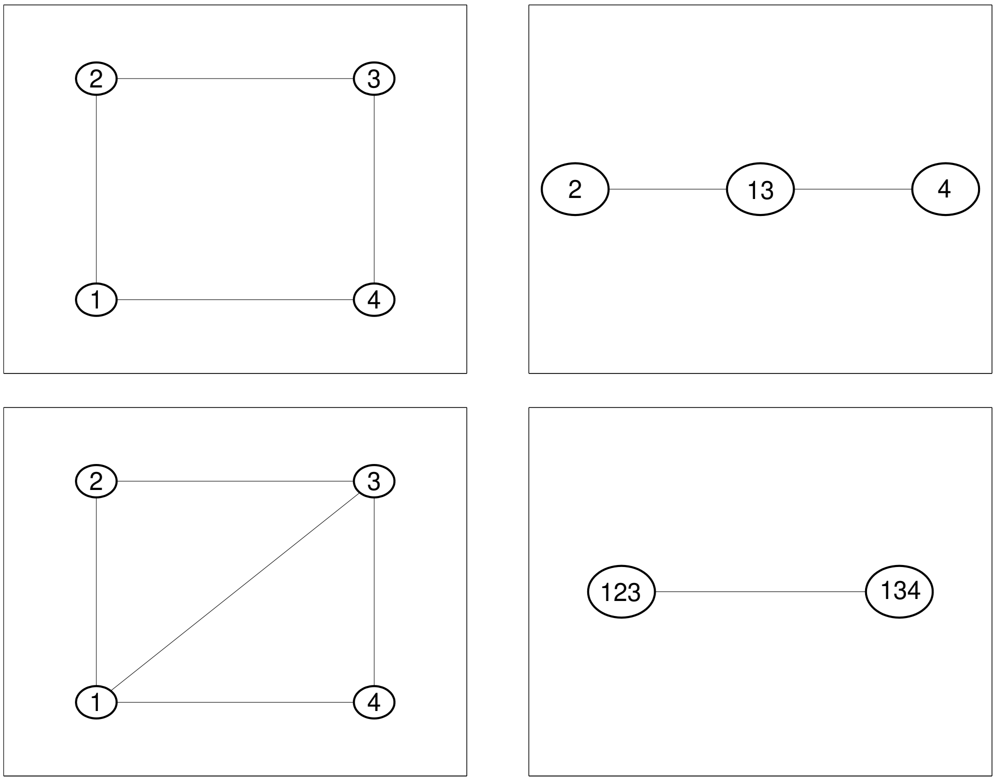
Exact inference in arbitrary graphical model



- Convert graph to an equivalent tree.

- run BP on the tree.

Converting arbitrary graph to a tree



- clustering (e.g. Pearl 86).
- Junction tree (e.g. Lauritzen 96)
- Bucket tree (e.g. Dechter 97)

Optimal conversion NP hard.

The need for approximate inference

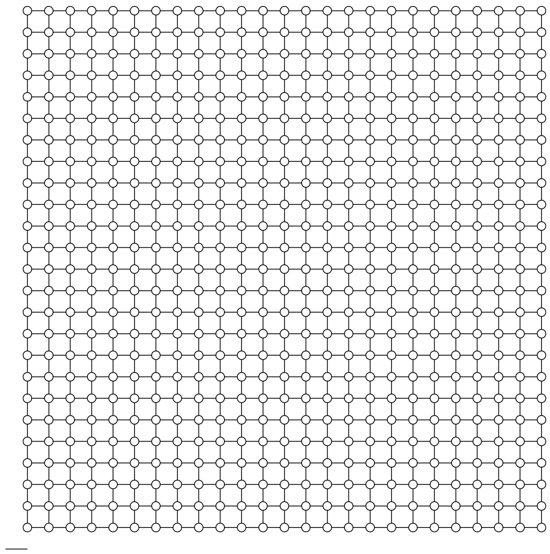
We can do exact inference in any graphical model by converting to a tree and running BP, but:

- Complexity of BP is exponential in the size of the nodes *.

- In most real world applications, node size in equivalent tree is huge.

$*$ some exceptions: noisy or CPTs, Gaussian CPTs.

Example: image analysis

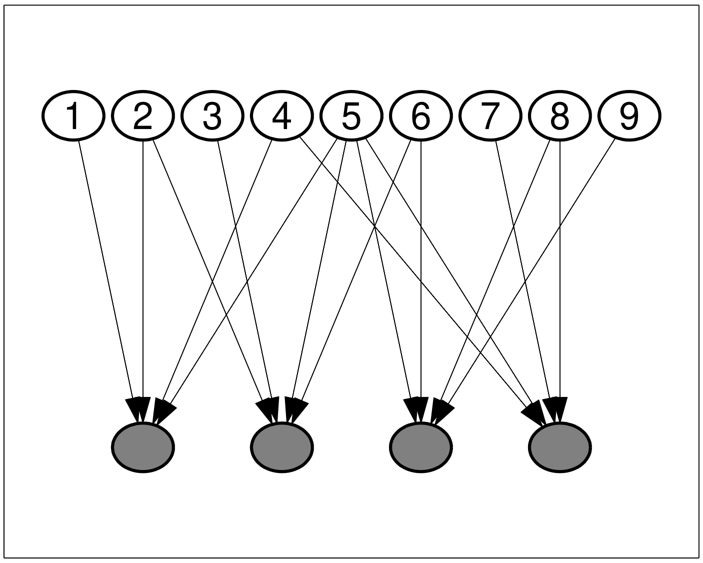


We want to classify pixels in an aerial photo as vegetation or not.

At each pixel we have color measurements and a prior that nearby pixels tend to have the same label.

Goal: calculate probability that a pixel is vegetation given measurements.

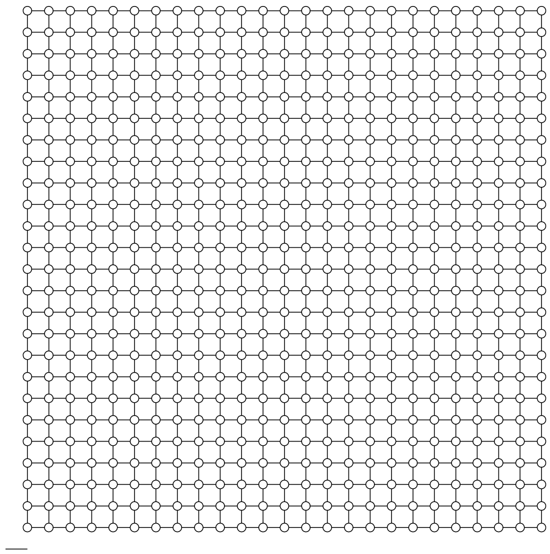
Example: LDPC Error correcting codes



Low power wireless communication $\Rightarrow$ noise. We use codewords in which subsets of the bits are constrained to have zero parity.

Goal: calculate probability that a bit is a one.

The computational bottleneck



Assume the image size is 256x256 then *exactly* calculating the probability that a pixel is vegetation takes order 2^{256} operations.

Similar numbers for exact inference in LDPCs.

Approximate inference in graphical models

- Monte-Carlo methods (e.g. Geman and Geman 84)

- Variational ("mean-field") approaches (e.g. Geiger and Girosi 91, Jordan et al. 98)

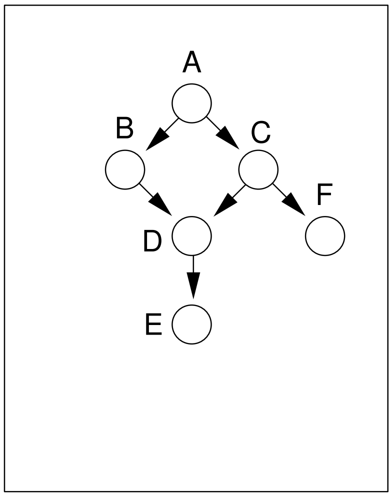
- "loopy" belief propagation.

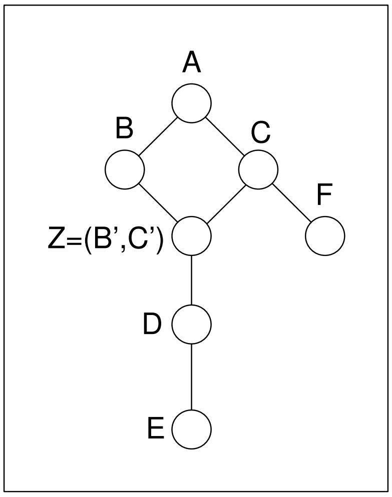
Loopy Belief propagation in directed graphs

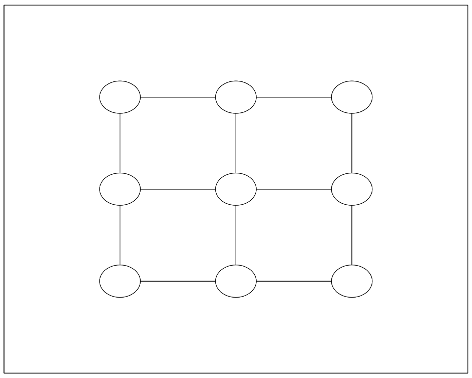
Convert graph to pairwise undirected graph.

Iterate, loopy BP equation:

$$m_{ij}(x_j) \leftarrow \alpha \sum_{x_i} \Psi_{ij}(x_i, x_j)\Psi_i(x_i) \prod_{X_k \in N(X_i) \backslash X_j} m_{ki}(x_i)$$







Loopy belief propagation

"When loops are present, the network is no longer singly connected and local propagation schemes will invariably run into trouble ... If we ignore the existence of loops and permit the nodes to continue communicating with each other as if the network were singly connected, messages may circulate indefinitely around the loops and the process may not converge to a stable equilibrium ... (even if it does) this asymptotic equilibrium is not coherent, in the sense that it does not represent the posterior probabilities of all nodes of the network" (Pearl 1988, p. 195)

Empirical successes of loopy propagation

- Gallager 63. Berrou at el. 93. Error correcting codes.
- Freeman and Pasztor 98, Saund 98. Image understanding.
- Shazeer et al. 99. Automatic crossword puzzle solution.
- Murphy et al. 99. Medical diagnosis.
- Frey 99. Factor analysis.
- Fridman and Mumford 99. Ising model.
- Frey 2001. Phase Unwrapping.

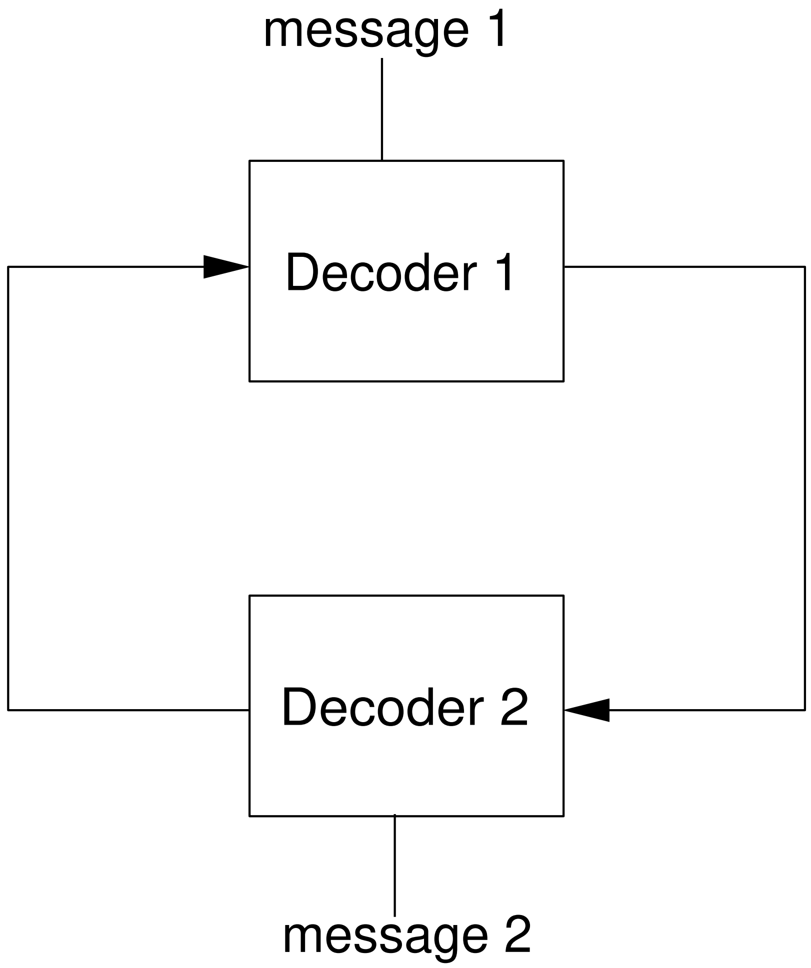
Error-correcting codes

X_i are bits. Original bits plus redundant "check" bits. Y_i are these bits sent over a noisy channel.

We want to design a code that:

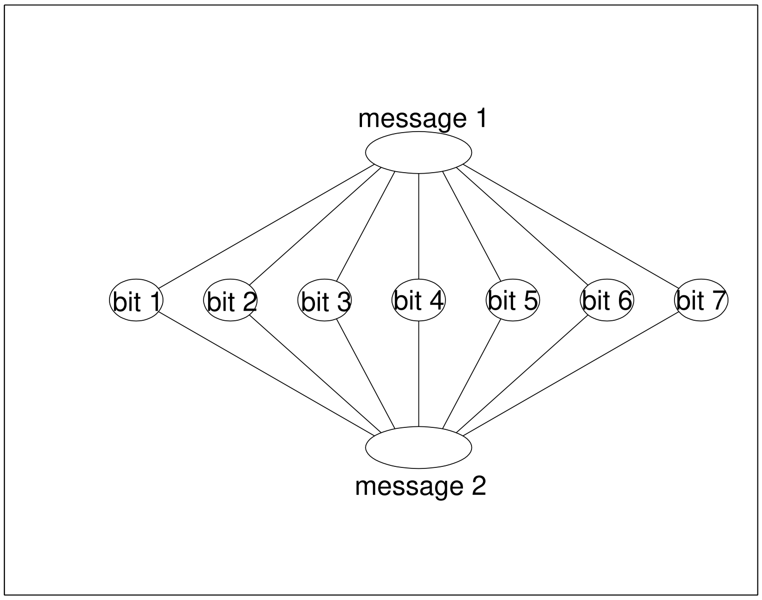
- can be decoded in reasonable time.
- the probability of error is as low as possible (Shannon's limit).

Turbo decoding algorithm



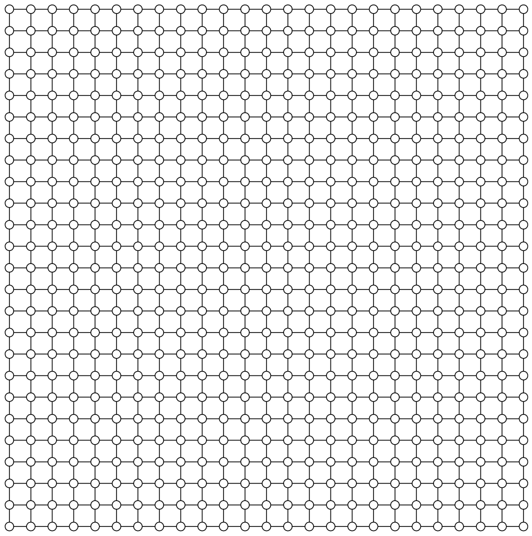
"A genuine and perhaps historic breakthrough" ... "the most exciting and potentially important development in coding theory in many years" (McEleice et al 95)

Turbo decoding is belief propagation



(McEliece et al 97, Wiberg 96). Belief propagation is not supposed to work in this case.

Example: image analysis

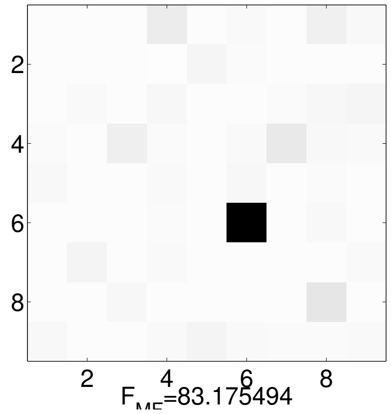


We want to classify pixels in an aerial photo as vegetation or not.

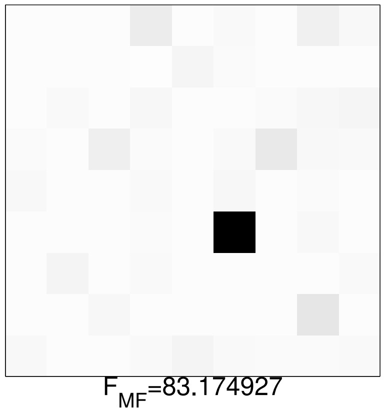
At each pixel we have color measurements and a prior that nearby pixels tend to have the same label.

Goal: calculate probability that a pixel is vegetation given measurements.

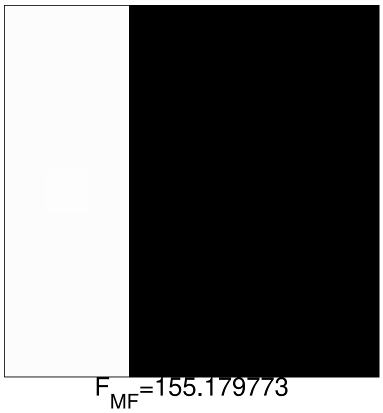
approximate inference results



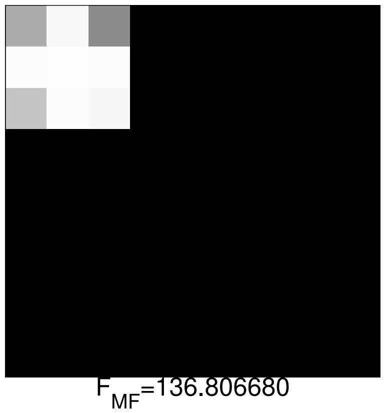
exact



loopy

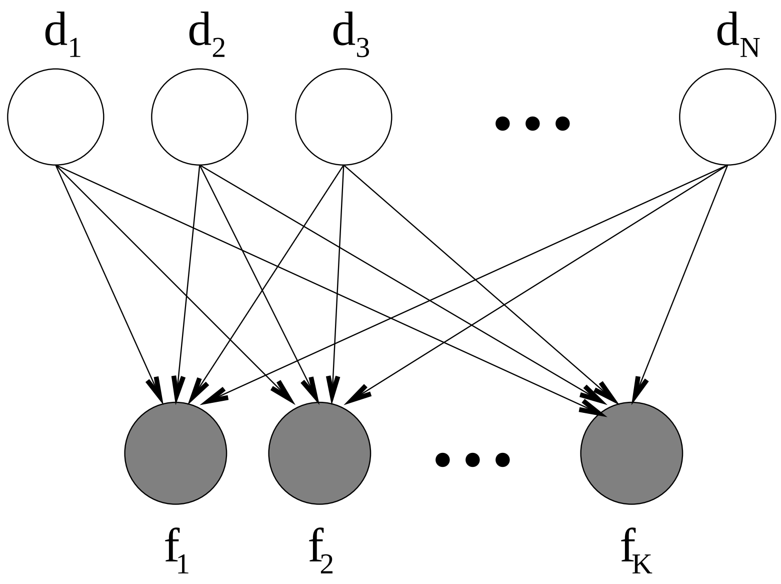


MF run 1



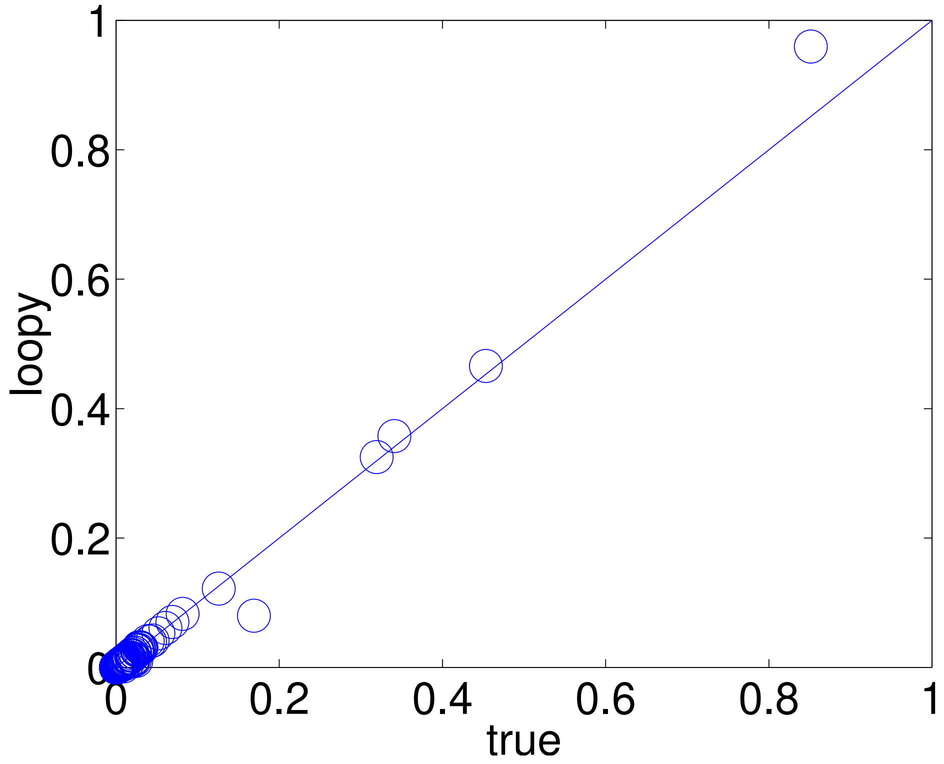
MF run 2

QMR network



approx. 600 diseases and 4000 symptoms.

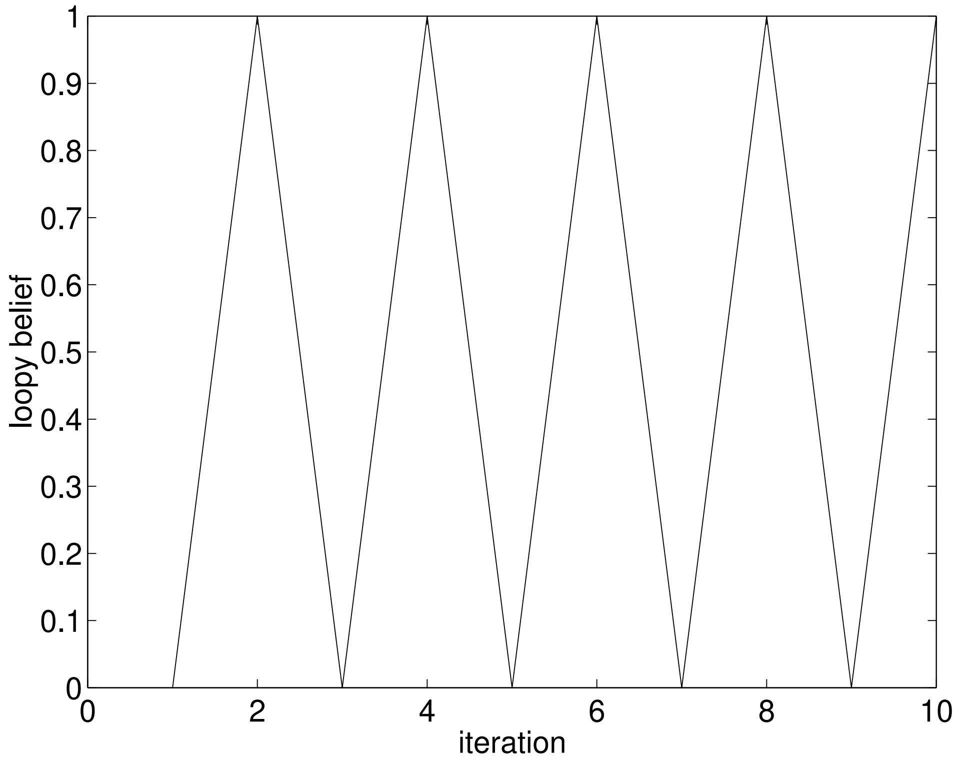
QMR results



case 32 works but case 46 does not converge.

(Murphy, Weiss and Jordan 99)

Failures of loopy BP



Two major failure modes:

- Beliefs oscillate wildly with iteration (more common).

- Beliefs converge to poor approximations.

Heuristics for avoiding oscillations

$$m_{ij}(x_j) \leftarrow \alpha \sum_{x_i} \Psi_{ij}(x_i, x_j) \Psi_i(x_i) \prod_{X_k \in N(X_i) \setminus X_j} m_{ki}(x_i)$$

- Asynchronous update.

- Damping.

$$m_{ij} = \alpha m_{ij}^{old} + (1 - \alpha) m_{ij}^{new}$$

Analysis of loopy BP

We now understand behavior of loopy BP in:

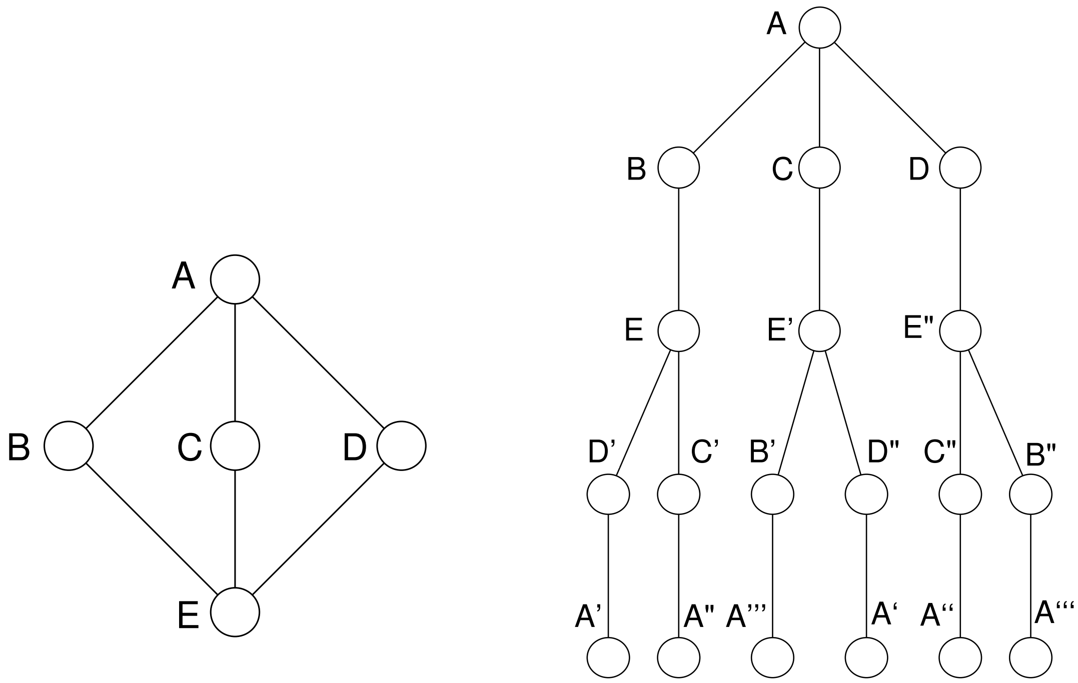
- Single-loop graphs. Convergence proof. Correctness of decision. (Weiss 97, Aji, Horn, and McElice 98, Forney et al 98).

- Gaussian graphs with arbitrary topology. Convergence conditions. Correctness of means. (Weiss and Freeman 99, Rusmevichientong and Van Roy 99).

- max-product in arbitrary graphs. semi-optimality of decision. (Weiss and Freeman 00)

- Average performance for LDPCs (Richardson and Urbanke 99)

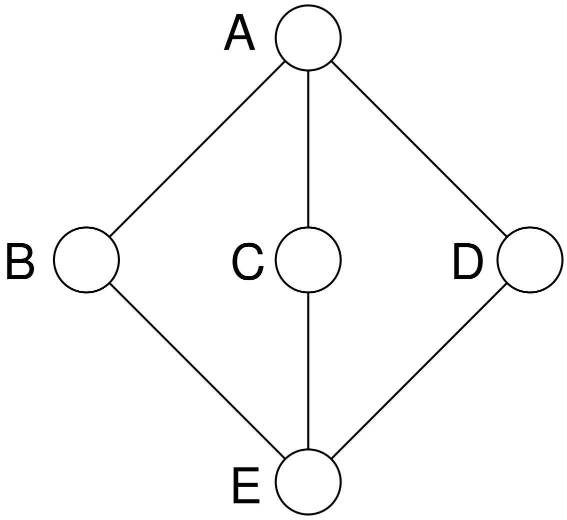
Main idea of proof:

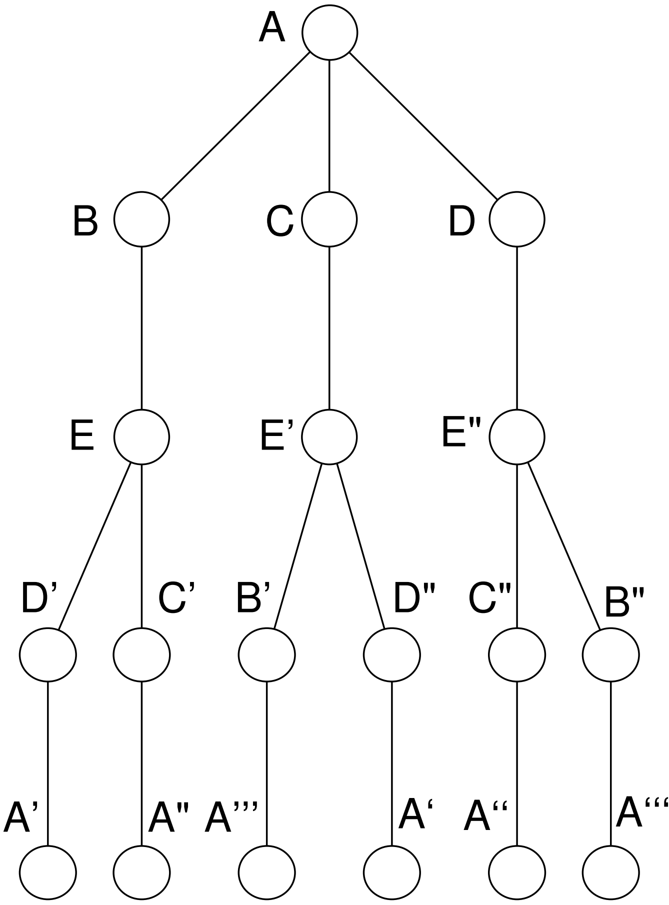


belief propagation performs exact inference in unwrapped graph. (cf. "computation tree" Gallager 63).

unwrapped graph has same local topology as loopy graph (universal covering).

After 4 iterations





Variational Interpertation:

For a graphical model of *arbitrary toplogy and potentials* the sum-product algorithm can only converge to local stationary points of the Bethe free energy:

$$
\begin{aligned}
F_\beta(\{b_{ij}, b_i\}) \;=\; & -\sum_{ij}\sum_{x_i,x_j} b_{ij}(x_i,x_j)\ln\psi_{ij}(x_i,x_j) \\
& -\sum_{i}\sum_{x_i} b_i(x_i)\ln\psi_i(x_i) \\
& +\sum_{ij}\sum_{x_i,x_j} b_{ij}(x_i,x_j)\ln b_{ij}(x_i,x_j) \\
& -\sum_{i}(q_i-1)\sum_{x_i} b_i(x_i)\ln b_i(x_i)
\end{aligned}
$$

where $b_{ij}(x_i, x_j)$ is joint belief of x_i, x_j,q_i is degree of node i.

Yedidia J.S., Freeman W.T. and Weiss Y. "Generalized Belief Propagation" NIPS 2000.

Bethe free energy in directed graphs

BP can be used to calculate $b(x_i, Par(x_i))$ by multiplying incoming messages and local evidence. Denote $f_i = (x_i, Par(x_i))$ " ith family".

Claim: $\{b_i(x_i)\}$,$\{b(f_i)\}$ fixed-points of BP if and only if constrained stationary points of:

$$
\begin{aligned}
F_\beta \;=\; & -\sum_i \sum_{f_i} b(f_i) \ln P(x_i|Par(x_i)) \\
& -\sum_i \sum_{x_i} b(x_i) \ln P(y_i|x_i) \\
& +\sum_i \sum_{f_i} b(f_i) \ln b(f_i) \\
& -\sum_i (q_i - 1) \sum_{x_i} b(x_i) \ln b(x_i)
\end{aligned}
$$

where q_i number of families that node x_i is in. Constraints: family beliefs marginalize down to singleton beliefs.

Sketch of proof

Introduce Lagrange multipliers that enforce marginalization constraints. Differentiate Lagrangian with respect to beliefs and Lagrange multipliers and set equal to zero. Resulting equations are equivalent to BP update rules with messages being logarithms of Lagrange multipliers (up to coordinate transformation). $\square$

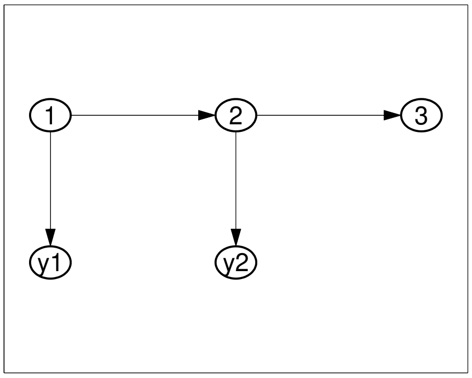
Free energies

$$
\begin{aligned}
F(\{b(x)\}) &= -\sum_x b(x) \ln P(x,y) + \sum_x b(x) \ln b(x) \\
&= -\sum_x b(x) \ln P(x|y) + \sum_x b(x) \ln b(x) \\
&\quad + \ln P(y)
\end{aligned}
$$

$$
\begin{aligned}
F_\beta &= -\sum_i \sum_{f_i} b(f_i) \ln P(x_i|Par(x_i)) \\
&\quad -\sum_i \sum_{x_i} b(x_i) \ln P(y_i|x_i) \\
&\quad +\sum_i \sum_{f_i} b(f_i) \ln b(f_i) \\
&\quad -\sum_i (q_i - 1) \sum_{x_i} b(x_i) \ln b(x_i)
\end{aligned}
$$

If graph is a tree, $F = F_\beta$

Example derivation



$$P(x,y) = P(x_2|x_1)P(x_3|x_2)P(y_1|x_1)P(y_2|x_2)$$

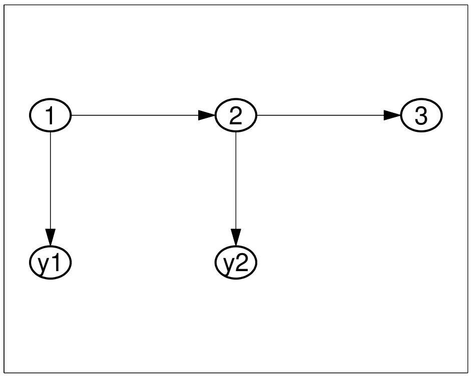
The average energy is $\sum_x b(x) \ln P(x,y) =$:

$$= \sum_x b(x) \left(\ln P(x_2|x_1) + \ln P(x_3|x_2) \cdots \right)$$

$$= \sum_{x_{12}} b(x_{12}) \ln P(x_2|x_1) + \sum_{x_{23}} b(x_{23}) \ln P(x_3|x_2)$$

$$+ \sum_{x_1} b(x_1) \ln P(y_1|x_1) + \sum_{x_2} b(x_2) \ln P(y_2|x_2)$$

Example derivation



$$P(x,y) = P(x_2|x_1)P(x_3|x_2)P(y_1|x_1)P(y_2|x_2)$$

we can restrict search to $b(x)$ that obey Markov properties:

$$\begin{aligned} b(x) &= b(x_1,x_2)b(x_3|x_2) \\ &= \frac{b(x_1,x_2)b(x_2,x_3)}{b(x_2)} \end{aligned}$$

The negative entropy is $\sum_x b(x) \ln b(x) =$:

$$\begin{aligned} &= \sum_x b(x) \ln \frac{b(x_1,x_2)b(x_2,x_3)}{b(x_2)} \\ &= \sum_{x_{12}} b(x_{12}) \ln b(x_{12}) + \sum_{x_{23}} b(x_{23}) \ln b(x_{23}) \\ &- \sum_{x_2} b(x_2) \ln b(x_2) \end{aligned}$$

Implications

- Existence of fixed-point for arbitrary graph.

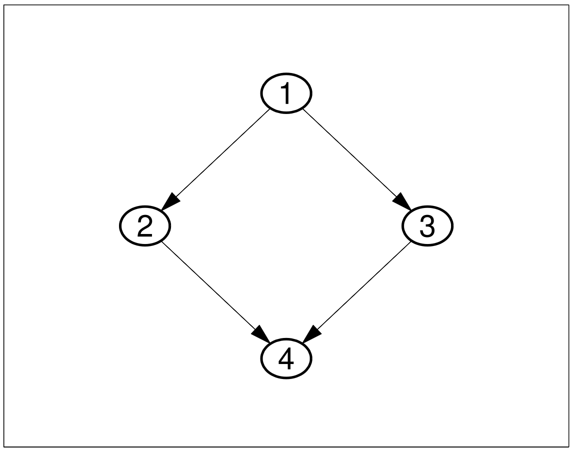
- Connection with variational approaches. MF free energy is Bethe free energy where family beliefs factorizes into product of singleton beliefs.

- Other ways of finding fixed-points are legitimate.

- Gives method for computing $P(y)$.

Beyond the Bethe approximation

In the Bethe approximation, the energy is exact and the entropy is approximated as a sum of simpler entropies minus "double-counted" single entropies.



$$\begin{aligned} H_\beta &= H(x_{12}) + H(x_{13}) + H(x_{234}) \\ &\quad - H(x_1) - H(x_2) - H(x_3) \end{aligned}$$

a better approximation is:

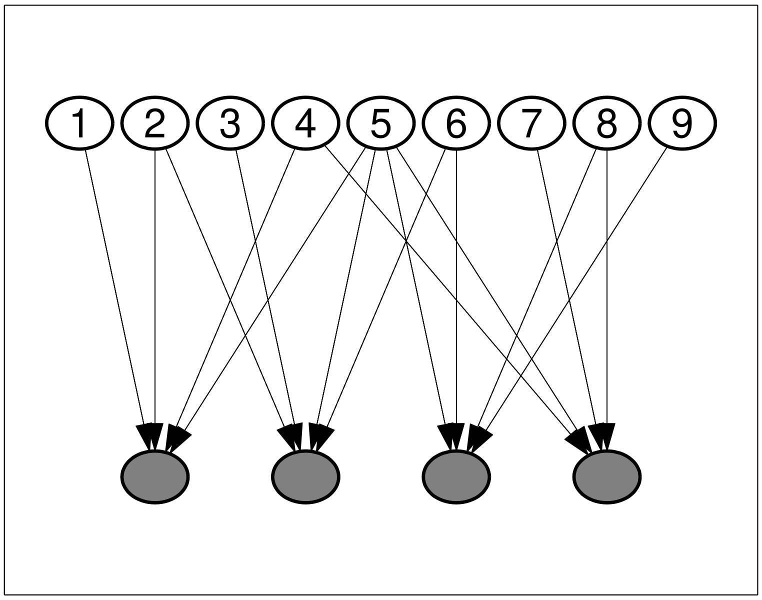
$$\begin{aligned} H_K &= H(x_{12}) + H(x_{13}) + H(x_{234}) \\ &\quad - H(1) - H(x_{23}) \end{aligned}$$

Kikuchi approximations to entropy

Approximate the entropy as sums of smaller region entropies weighted appropriately so that each node is counted exactly once.

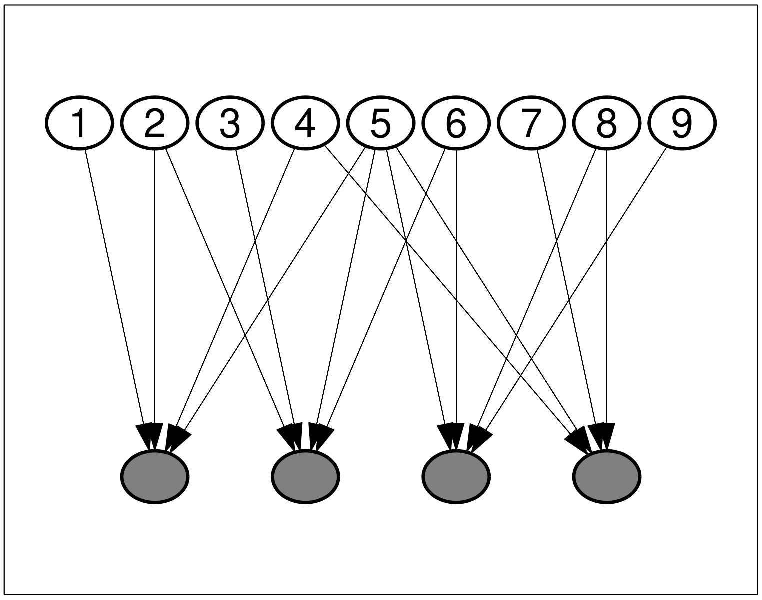
When regions are large enough, this becomes exact.

Examples of Kikuchi entropies



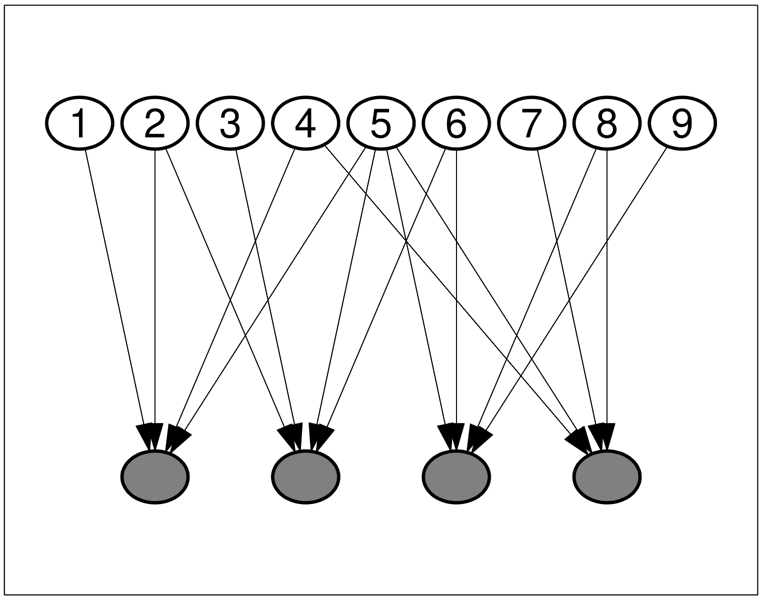
$$
\begin{aligned}
H_\beta &= \\
&= H(x_{1245}) + H(x_{2356}) + H(x_{4578}) + H(x_{5689}) \\
&\quad - H(x_2) - H(x_6) - H(x_8) - H(x_4) - 3H(x_5)
\end{aligned}
$$

Examples of Kikuchi entropies



$$
\begin{aligned}
H_K &= \\
&= H(x_{1245}) + H(x_{2356}) + H(x_{4578}) + H(x_{5689}) \\
&\quad - H(x_{25}) - H(x_{56}) - H(x_{58}) - H(x_4)
\end{aligned}
$$

Examples of Kikuchi entropies



$$
\begin{aligned}
H_K &= \\
&= H(x_{1245}) + H(x_{2356}) + H(x_{4578}) + H(x_{5689}) \\
&\quad -H(x_{25}) - H(x_{56}) - H(x_{58}) - H(x_{45}) \\
&\quad +H(x_5)
\end{aligned}
$$

Minimizing Kikuchi free energy

- The Kikuchi free energy is the Bethe energy minus the Kikuchi entropy.
- We have developed Generalized Belief Propagation algorithms that minimize Kikuchi free energies.
- Complexity is exponential in the size of regions.
- Sometimes, much better approximation with slight increase in complexity.

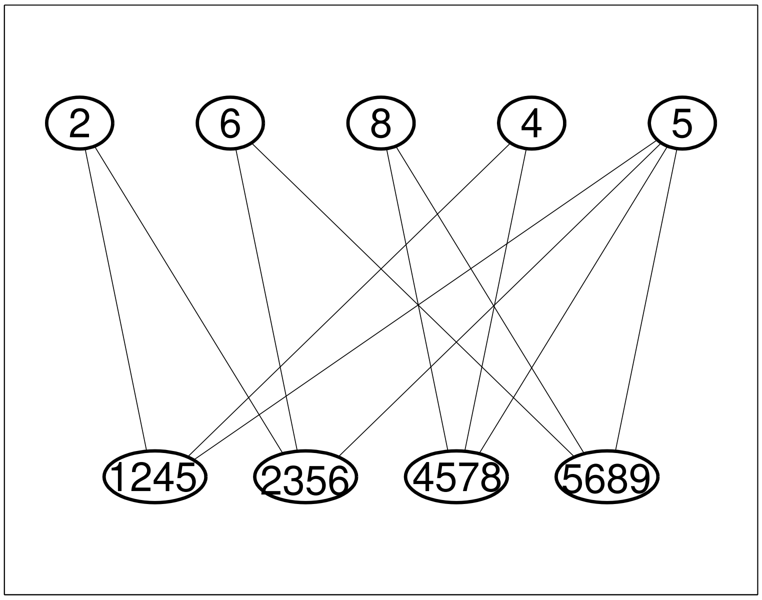
GBP Algorithms

- Many algorithms for a given free energy. Depends on how you write Lagrange multipliers.

- One algorithm: cluster nodes and run ordinary BP on clustered graph with additional update rule in certain graphs.

- We have only begun to explore the space of algorithms and energies.

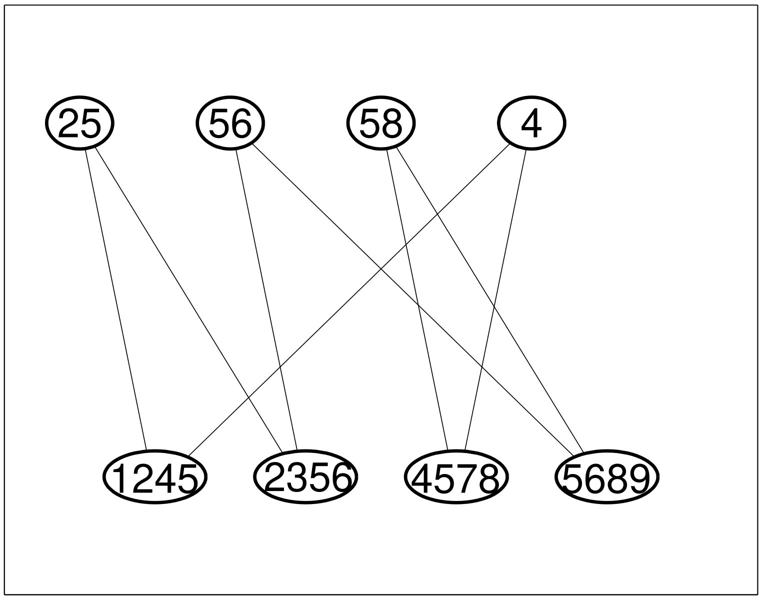
Bethe approximation



$$
\begin{aligned}
H_\beta = \\
& H(x_{1245}) + H(x_{2356}) + H(x_{4578}) + H(x_{5689}) \\
& -H(x_2) - H(x_6) - H(x_8) - H(x_4) - 3H(x_5)
\end{aligned}
$$

exact	0.1202	0.1814	0.1155	0.1778	0.8128
approx	0.1206	0.1409	0.1158	0.1391	0.9301

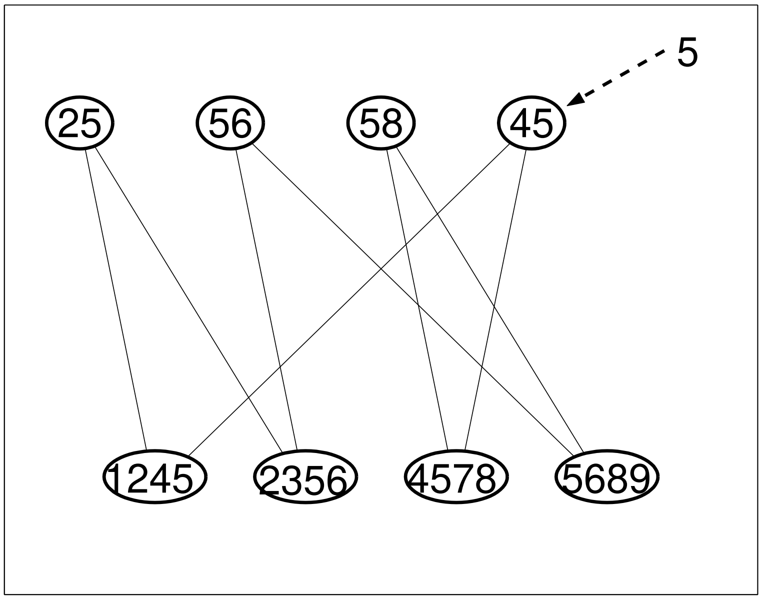
One Kikuchi approximation



$$H_K =$$
$$H(x_{1245}) + H(x_{2356}) + H(x_{4578}) + H(x_{5689})$$
$$-H(x_{25}) - H(x_{56}) - H(x_{58}) - H(x_4)$$

exact	0.1202	0.1814	0.1155	0.1778	0.8128
approx β	0.1206	0.1409	0.1158	0.1391	0.9301
approx $K1$	0.1189	0.1537	0.1135	0.1861	0.8419

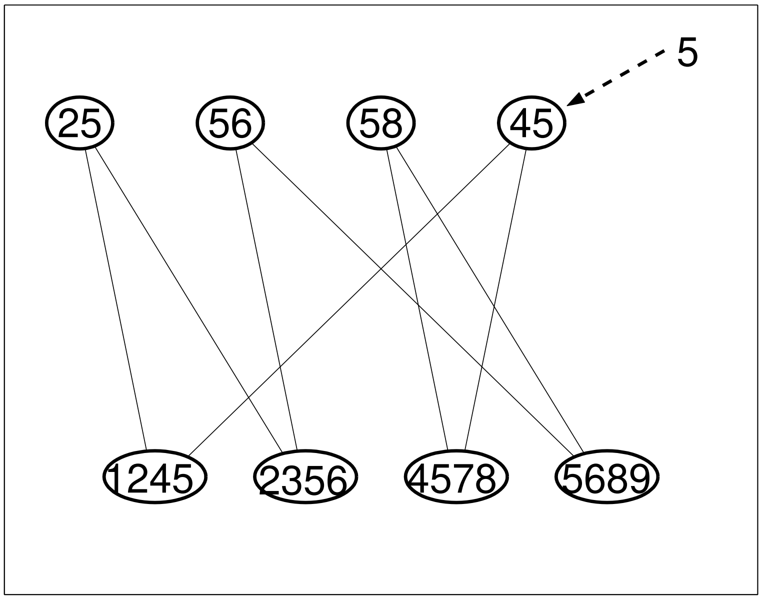
A better Kikuchi approximation



$$
\begin{aligned}
&H(x_{1245}) + H(x_{2356}) + H(x_{4578}) + H(x_{5689}) \\
&-H(x_{25}) - H(x_{56}) - H(x_{58}) - H(x_{45}) \\
&+H(x_5)
\end{aligned}
$$

exact	0.1202	0.1814	0.1155	0.1778	0.8128
approx β	0.1206	0.1409	0.1158	0.1391	0.9301
approx $K1$	0.1189	0.1537	0.1135	0.1861	0.8419
approx K_2	0.1227	0.1781	0.1177	0.1742	0.8216

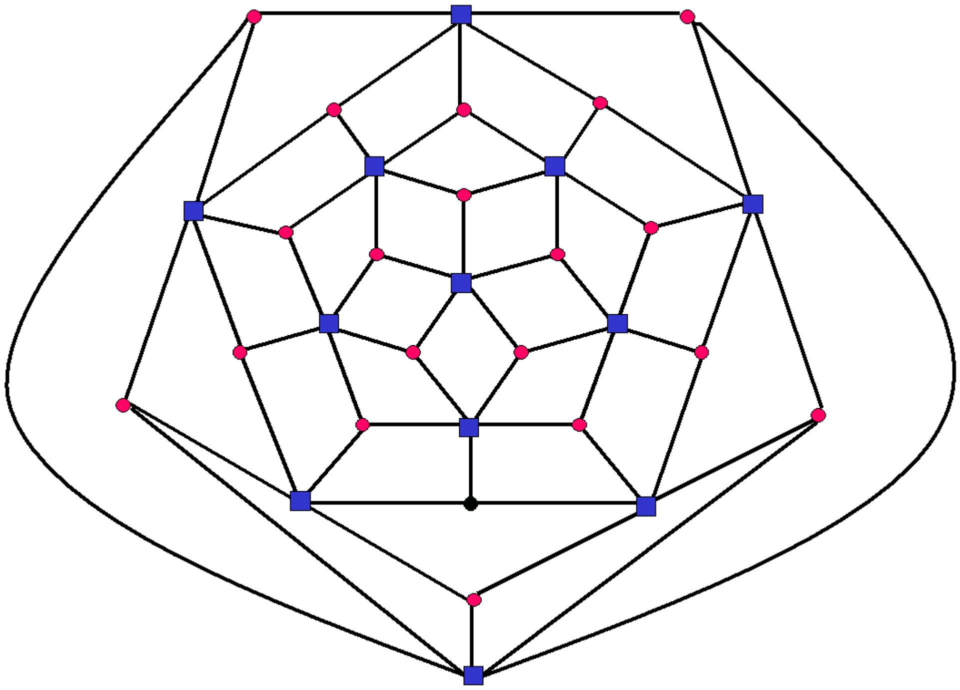
A better Kikuchi approximation

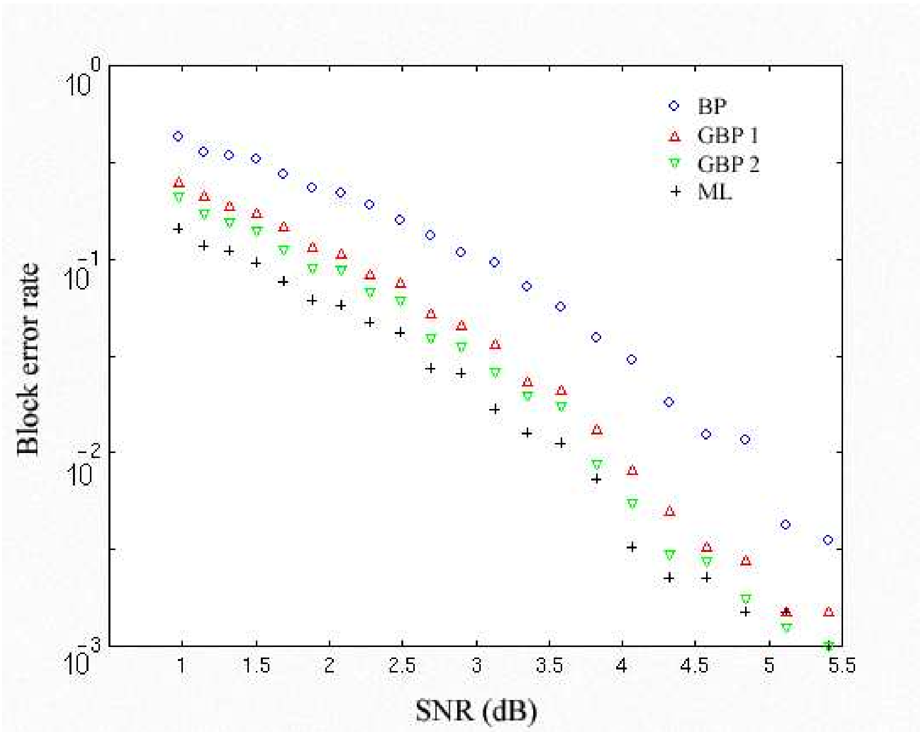


One new message:

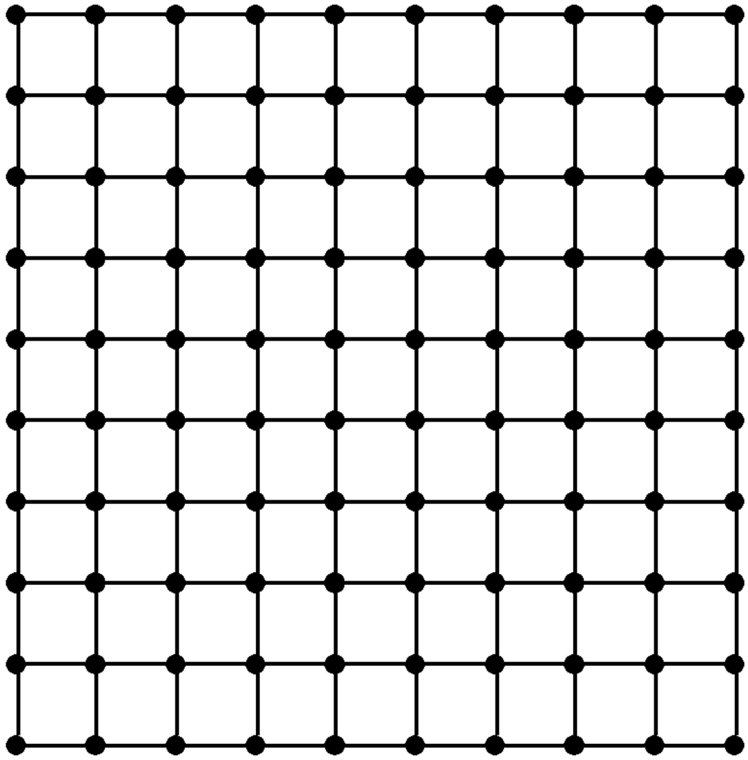
$$m_{5,45}(x_4, x_5) = m_{45,5}(x_5)^{-1/2}$$

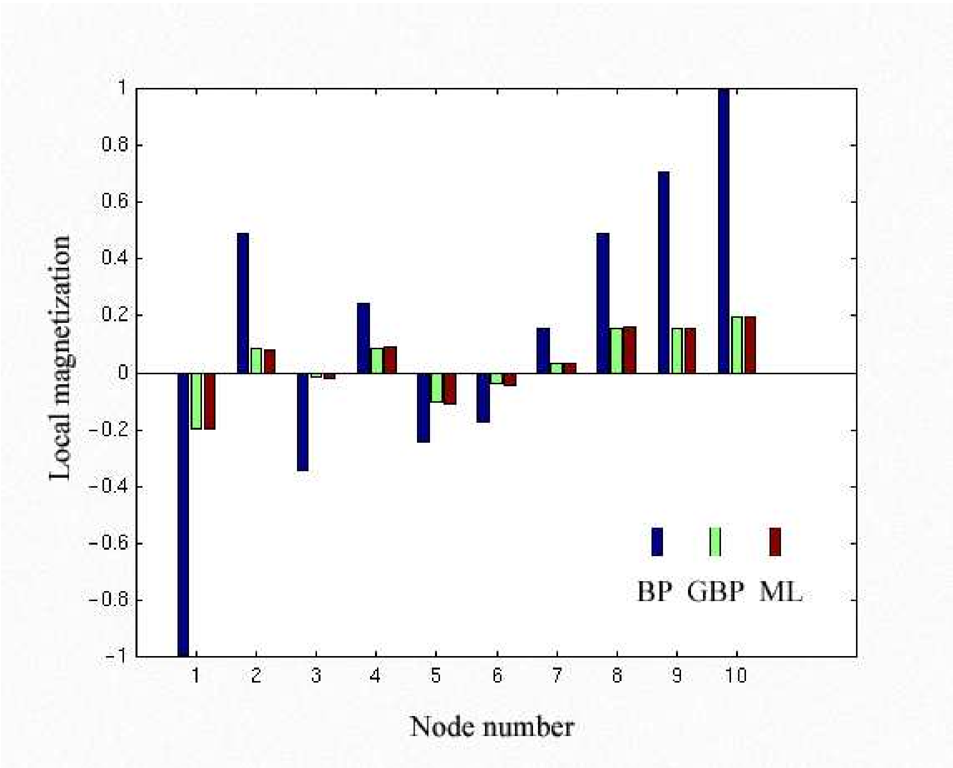
Comparison on error-correcting codes





Comparison on spin glass





Conclusions

- The need for approximate inference.

- loopy belief propagation. Was not supposed to work but dramatic empirical results — turbo codes.

- Theory of loopy BP: unwrapped tree and Bethe free energy.

- Kikuchi free energies extend Bethe. Can be minimized by generalized BP algorithms.

- Simple, universally applicable approximate inference algorithms.