Notes for Section 5

Today

- More motivation for graphical models
- A review of belief propagation
- Special-case: forward-backward algorithm
- From variable elimination to junction tree (mainly just intuition)

More belief propagation (sum-product)

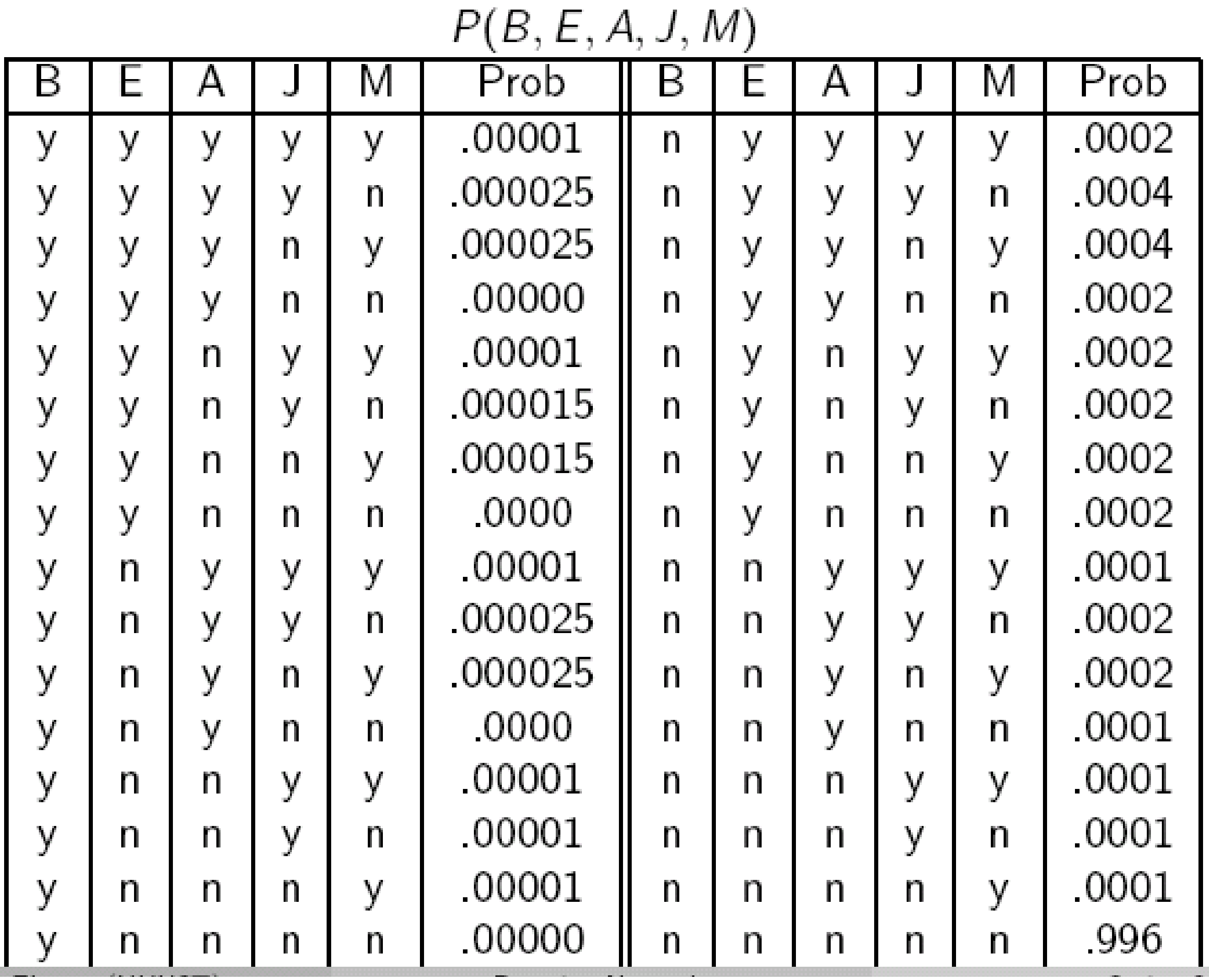
Consider an arbitrary probability distribution over N variables each of domain size D.

You can think of this as a table where each column corresponds to a variable, and for each row we specify a probability.

How many rows are there?

Suppose we condition on a variable x=v. What does this mean? It means picking out all rows with x=v and then renormalizing. Renormalizing is just dividing by the marginal probability of x=v.

Suppose we want to find the marginal probability of some variable x. What does this mean? For each value of x, it means finding all the rows with that value and adding their probabilities together. How many rows do we need to add?



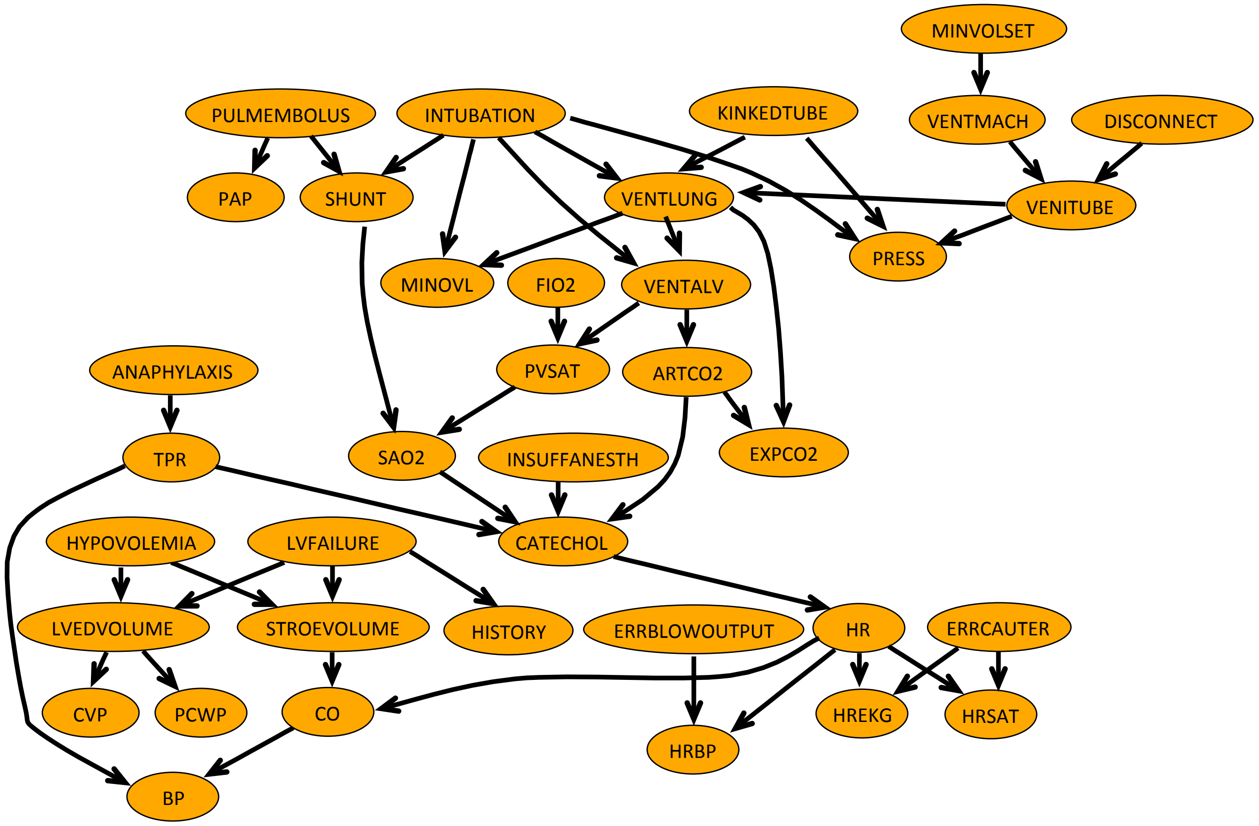
$P(B, E, A, J, M)$

B	E	A	J	M	Prob	B	E	A	J	M	Prob
y	y	y	y	y	.00001	n	y	y	y	y	.0002
y	y	y	y	n	.000025	n	y	y	y	n	.0004
y	y	y	n	y	.000025	n	y	y	n	y	.0004
y	y	y	n	n	.00000	n	y	y	n	n	.0002
y	y	n	y	y	.00001	n	y	n	y	y	.0002
y	y	n	y	n	.000015	n	y	n	y	n	.0002
y	y	n	n	y	.000015	n	y	n	n	y	.0002
y	y	n	n	n	.0000	n	y	n	n	n	.0002
y	n	y	y	y	.00001	n	n	y	y	y	.0001
y	n	y	y	n	.000025	n	n	y	y	n	.0002
y	n	y	n	y	.000025	n	n	y	n	y	.0002
y	n	y	n	n	.0000	n	n	y	n	n	.0001
y	n	n	y	y	.00001	n	n	n	y	y	.0001
y	n	n	y	n	.00001	n	n	n	y	n	.0001
y	n	n	n	y	.00001	n	n	n	n	y	.0001
y	n	n	n	n	.00000	n	n	n	n	n	.996

(slides: Rina Dechter)

- Naively, then, an arbitrary probability distribution requires O(N^D) operations.
- This might not seem so bad: but real world problems are large.

Monitoring Intensive-Care Patients

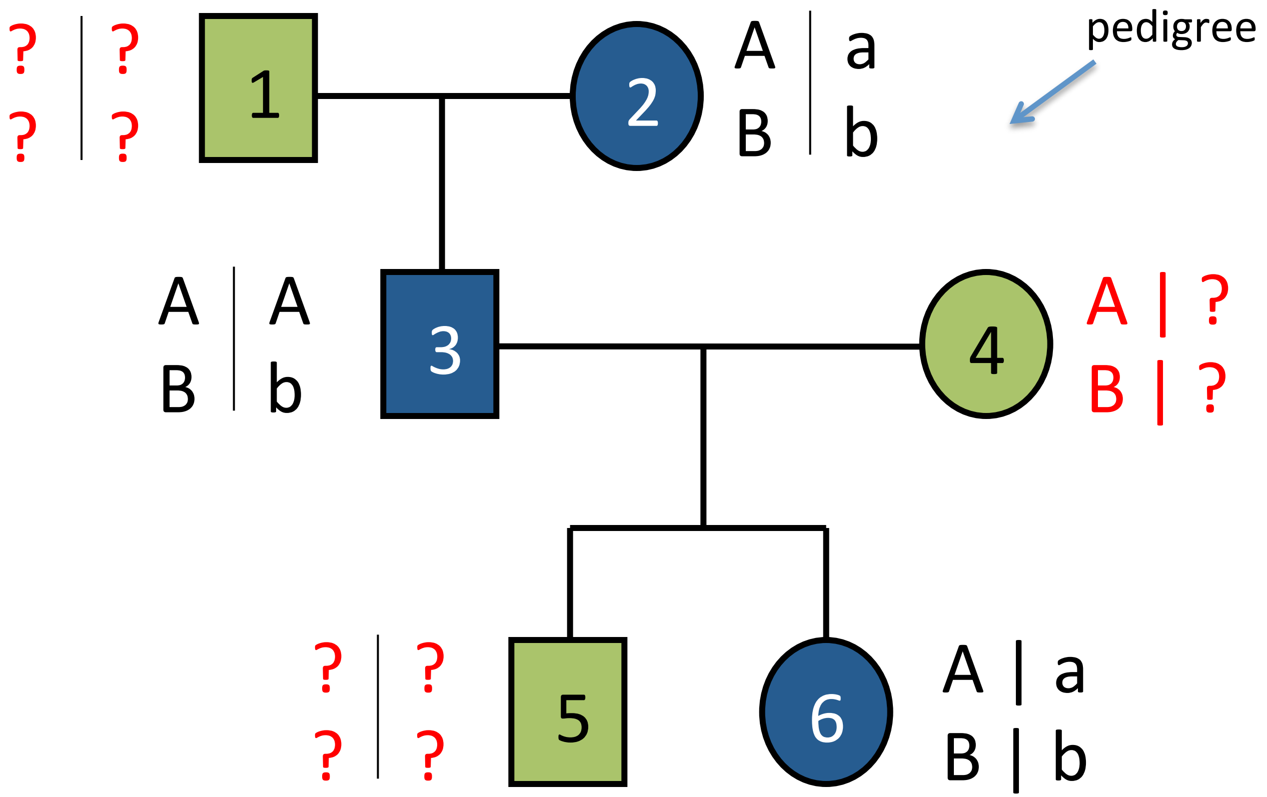


37 variables
509 parameters

<< 2^{37}

(slides: Rina Dechter)

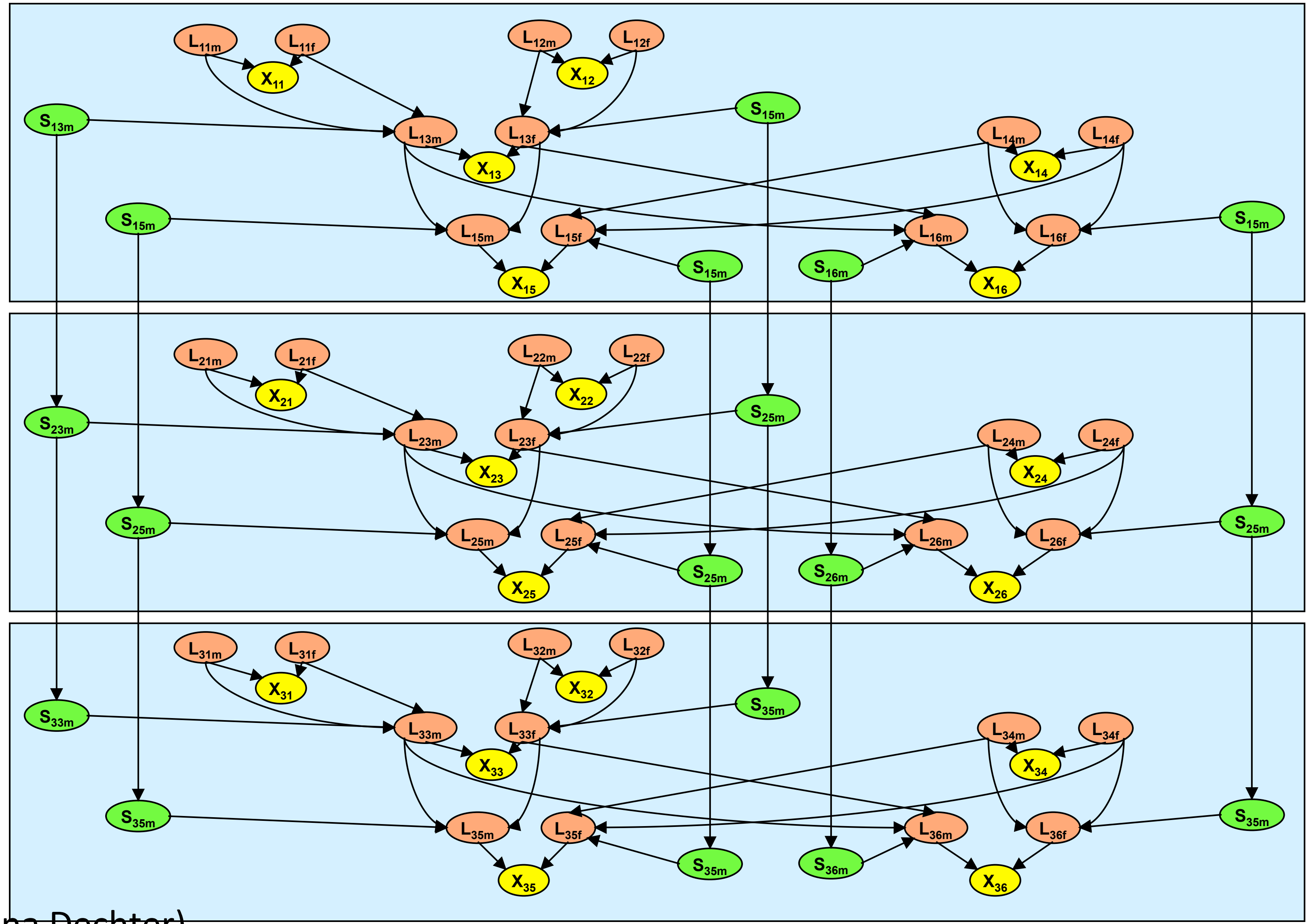
Another example: linkage analysis



- 6 individuals
- Haplotype: {2, 3}
- Genotype: {6}
- Unknown

(slides: Rina Dechter)

Pedigree: 6 people, 3 markers



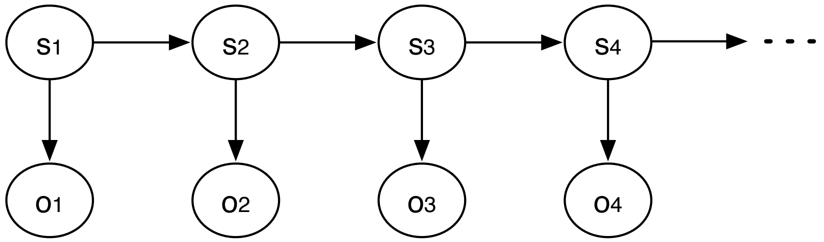
(slides: Rina Dechter)

But we can do better with message passing on trees.

- Using BP: choose root, sum over leaf variables and send message to parents, take the product, repeat, until you reach the root.
- Every summation is takes D^2 operations. At each parent, we need to multiply the messages from the children. There are at most N children, so at most N multiplications and one for each value in the parent variable.
- And if we go leaves to root and back this happens twice at each node, so O(N*D^2).

Forward backward algorithm

Decomposing the probability of an observation sequence



$$P(o_1, \ldots o_T) = \sum_{s_1,\ldots s_T} P(o_1, \ldots o_T, s_1, \ldots s_T)$$

$$= \sum_{s_1,\ldots s_T} P(s_1) \left(\prod_{t=2}^{T} P(s_t|s_{t-1}) \right) \left(\prod_{t=1}^{T} P(o_t|s_t) \right) \text{ (using the model)}$$

$$= \sum_{s_T} P(o_T|s_T) \sum_{s_1,\ldots s_{T-1}} P(s_T|s_{T-1}) P(s_1) \left(\prod_{t=2}^{T-1} P(s_t|s_{t-1}) \right) \left(\prod_{t=1}^{T-1} P(o_t|s_t) \right)$$

http://www.cs.mcgill.ca/~dprecup/courses/ML/Lectures/ml-lecture20.pdf

The forward algorithm

- Given an HMM model and an observation sequence $o_1, \ldots o_T$, define:

$$\alpha_t(s) = P(o_1, \ldots o_t, S_t = s)$$

- We can put these variables together in a vector α_t of size $\mathcal{S}$.
- In particular,

$$\alpha_1(s) = P(o_1, S_1 = s) = P(o_1|S_1 = s)P(S_1 = s) = q_{so_1}b_0(s)$$

- For $t = 2, \ldots T, \alpha_t(s) = p_{so_t} \sum_{s'} p_{s's}\alpha_{t-1}(s')$
- The solution is then

$$P(o_1, \ldots o_T) = \sum_s \alpha_T(s)$$

http://www.cs.mcgill.ca/~dprecup/courses/ML/Lectures/ml-lecture20.pdf

Computing state probabilities in general

- If we know the model parameters and an observation sequence, how do we compute $P(S_t = s|o_1, o_2, \ldots o_T)$?

$$\begin{aligned} P(S_t = s|o_1, \ldots o_T) &= \frac{P(o_1, \ldots o_T, S_t = s)}{P(o_1, \ldots o_T)} \\ &= \frac{P(o_{t+1}, \ldots o_T|o_1, \ldots o_t, S_t = s)P(o_1, \ldots o_t, S_t = s)}{P(o_1, \ldots o_T)} \\ &= \frac{P(o_{t+1}, \ldots o_T|S_t = s)P(o_1, \ldots o_t, S_t = s)}{P(o_1, \ldots o_T)} \end{aligned}$$

- The denominator is a normalization constant and second factor in the numerator can be computed using the forward algorithm (it is $\alpha_t(s)$)
- We now compute the first factor

http://www.cs.mcgill.ca/~dprecup/courses/ML/Lectures/ml-lecture20.pdf

The forward-backward algorithm

- Given the observation sequence, $o_1, \dots o_T$ we can compute the probability of any state at any time as follows:
  1. Compute all the $\alpha_t(s)$, using the forward algorithm
  2. Compute all the $\beta_t(s)$, using the backward algorithm
  3. For any $s \in S$ and $t \in \{1, \dots T\}$:

$$P(S_t = s | o_1, \dots o_T) = \frac{P(o_1, \dots o_t, S_t = s) P(o_{t+1}, \dots o_T | S_t = s)}{P(o_1, \dots o_T)} = \frac{\alpha_t(s)\beta_t(s)}{\sum_{s'} \alpha_T(s')}$$

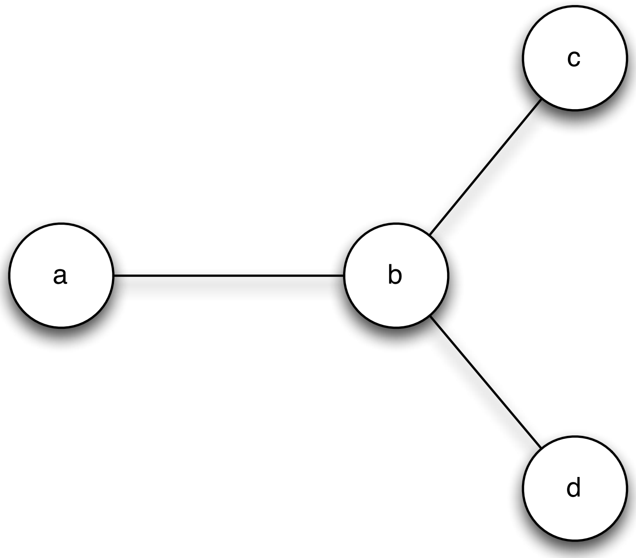
- The complexity of the algorithm is $O(|\mathcal{S}|T)$.

http://www.cs.mcgill.ca/~dprecup/courses/ML/Lectures/ml-lecture20.pdf

From VE to JT

- In most cases we don’t have trees
- BP doesn’t work because of “feedback”
- So we use variable elimination which is exp(treewidth). Tree width is roughly the size of the largest factor we generate during elimination.

The Junction Tree Algorithm

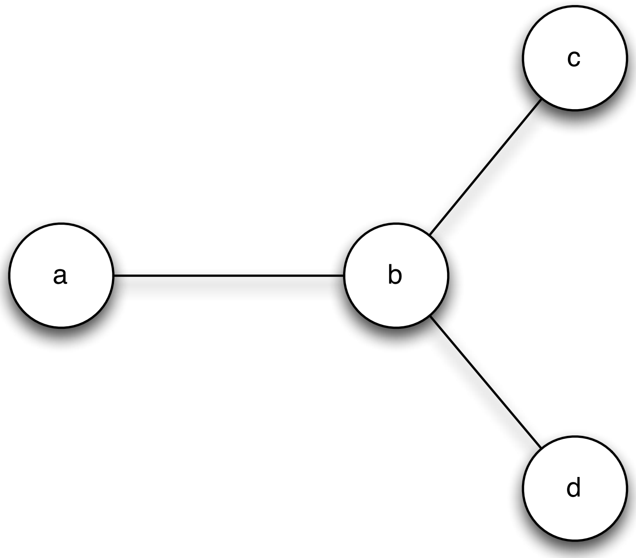


First, let's think about belief propagation in a slightly different way:

Every node in a undirected graphical model that is a tree is a SEPARATOR for the graph. If you remove it, it separates the graph into two unconnected components.

So we can do all the marginalizations in each component separately, and then send the resulting potentials over the separator to the separator node.

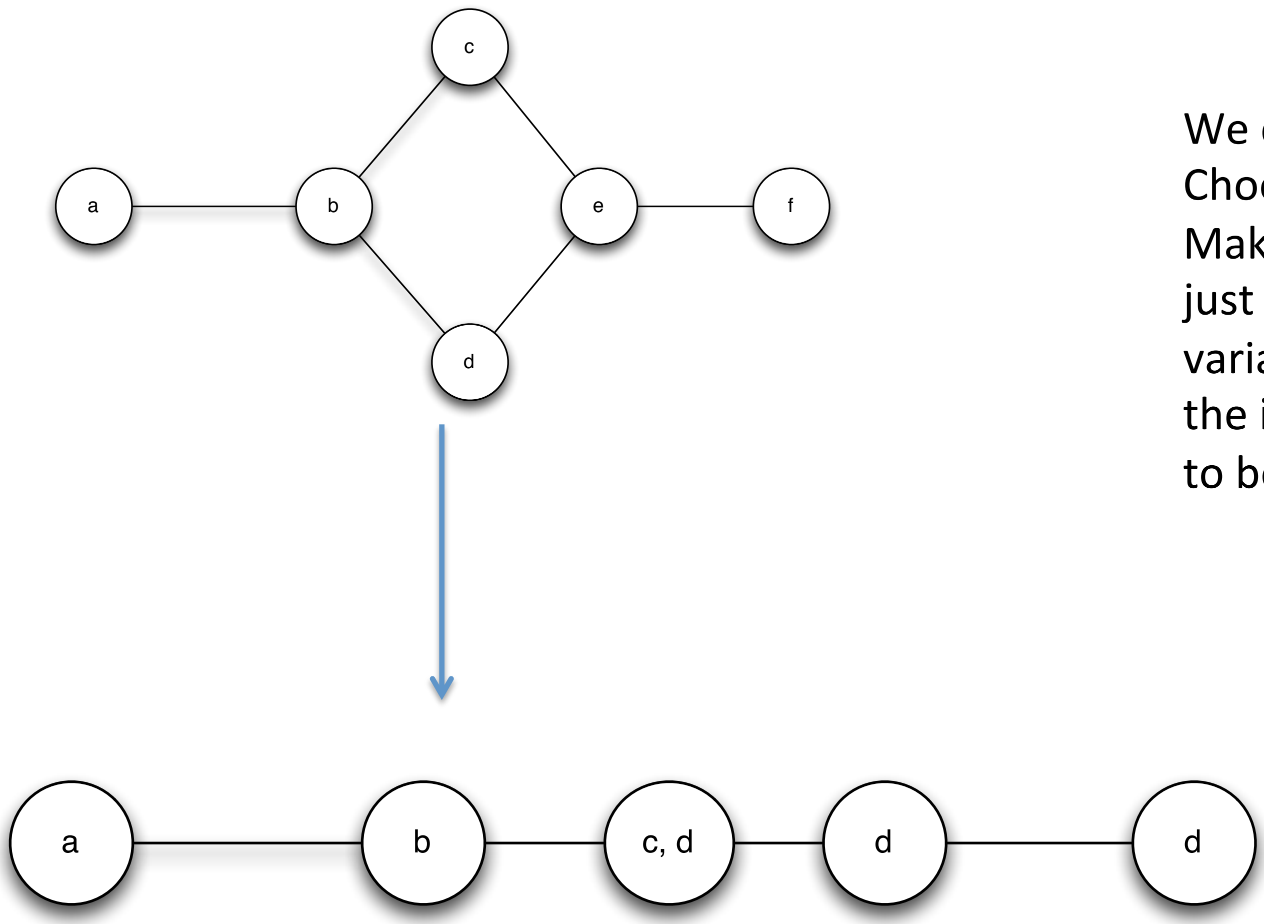
The Junction Tree Algorithm



This is convenient, because that means that in a tree we can break up the message calculations into nested sub-problems and we end up calculating the marginals by message passing.

But what do we do if we have cycles in our graph? …

The Junction Tree Algorithm

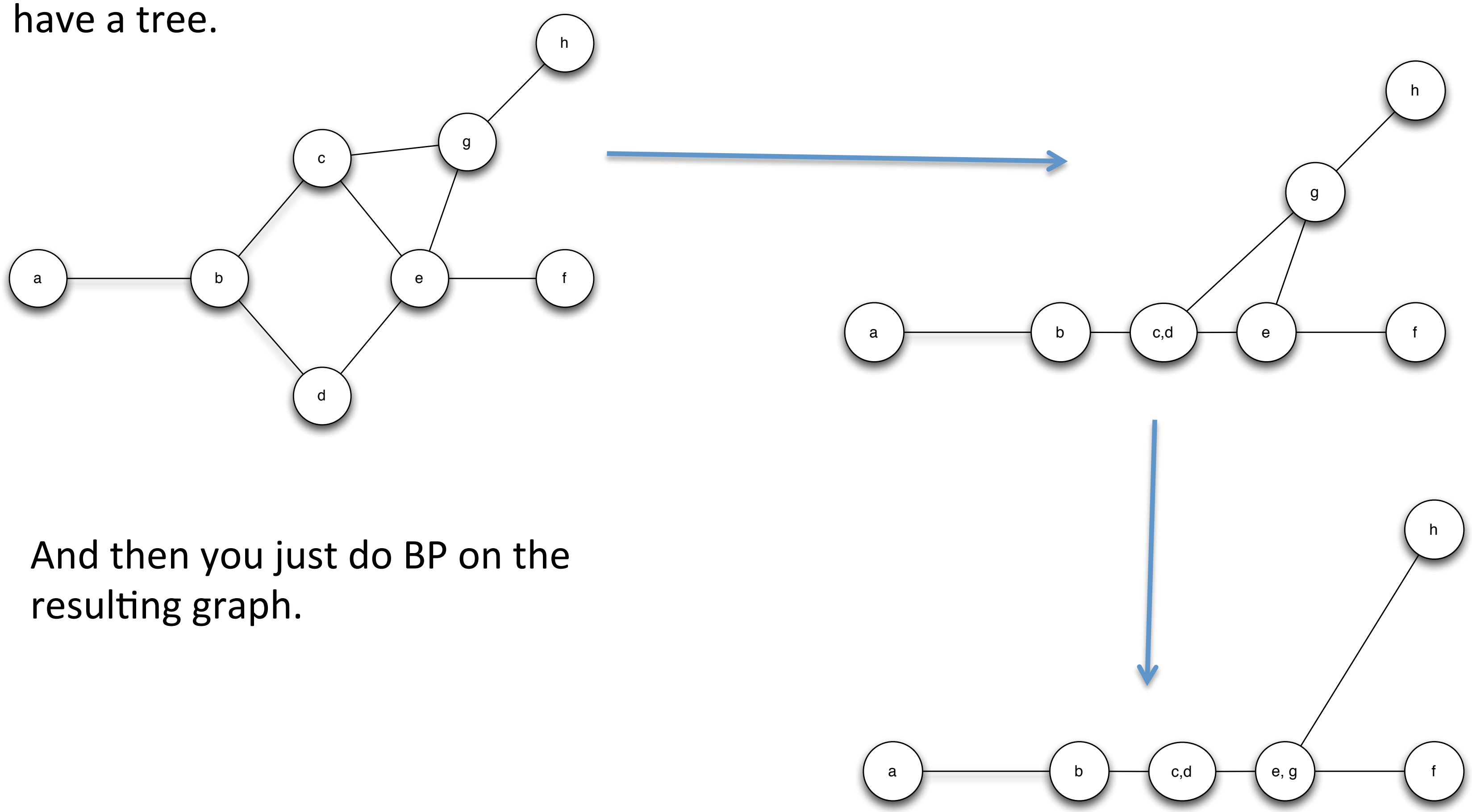


We can do the same thing!
Choose a separator set of variables.
Make that into a super-node which just the cartesian product of those variables and connect it to whatever the individual nodes were connected to before.

If you think about it, this has to represent the same distribution. It's just that if we want the marginal over c, first we get the marginal over (c,d) and then marginalize over d.

The Junction Tree Algorithm

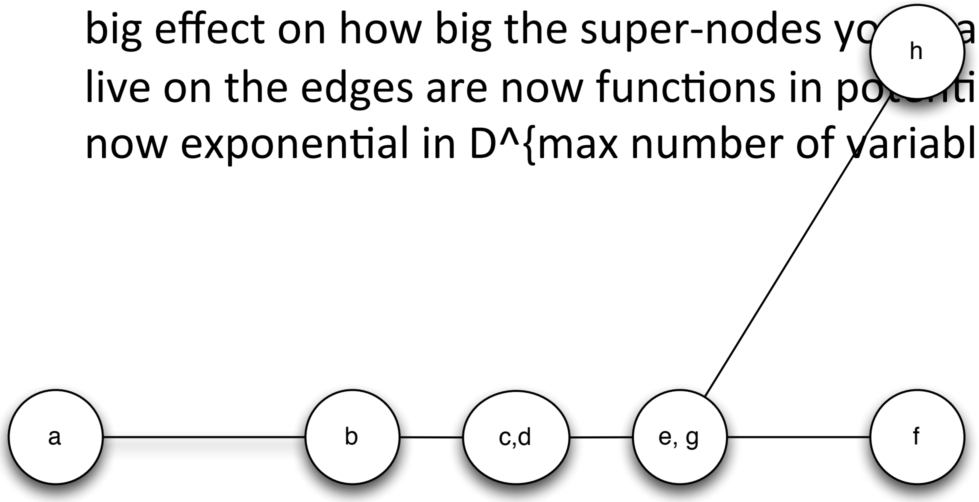
If you have more than one cycle, you just keep gathering together separators until you have a tree.

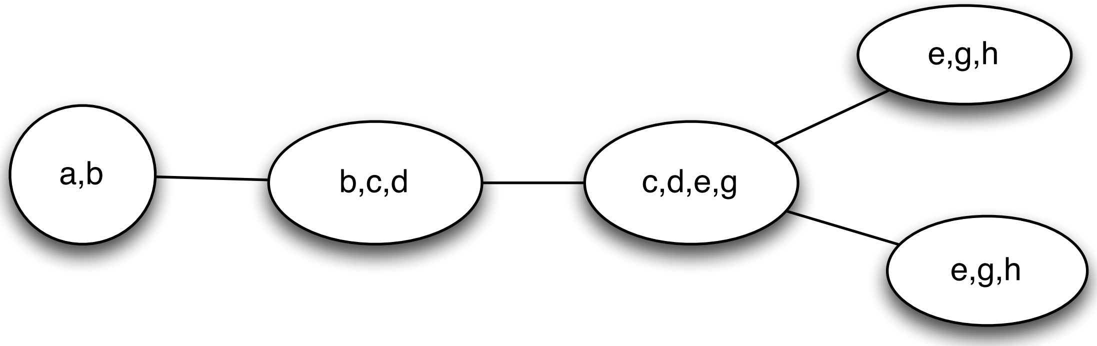


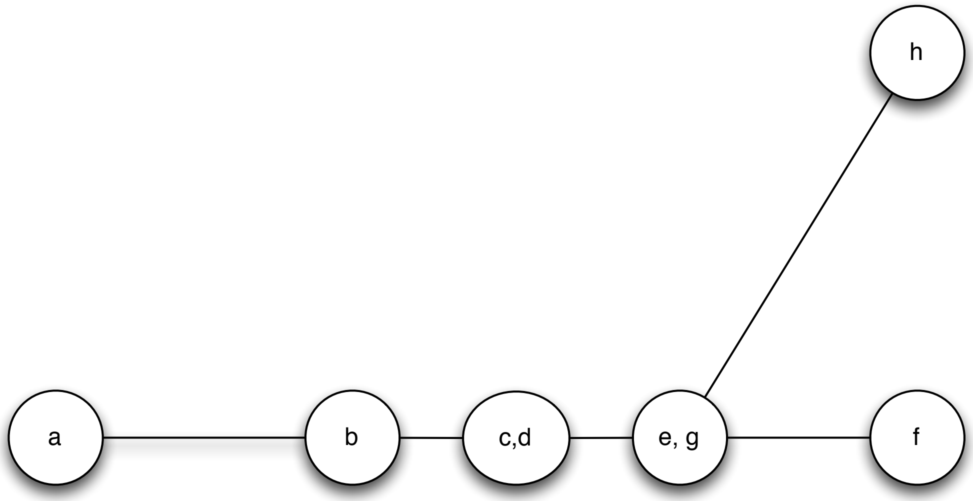
And then you just do BP on the resulting graph.

The Junction Tree Algorithm

Of course, there are often many separators and which separators you choose can have a big effect on how big the super-nodes you make are. And remember the functions that live on the edges are now functions in potentially many variables. So the running time is now exponential in D^{max number of variables connected by an edge}.

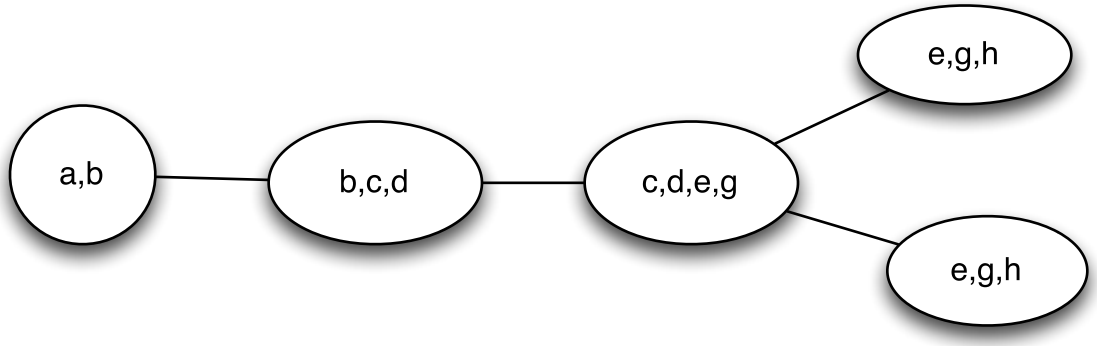






Just to make things confusing people in the field often talk about the dual graph which puts all the variables on an edge in a node and puts the nodes on the edges. Don't worry about it.

dual-graph (a.k.a junction tree)


e,g,h
a,b
b,c,d
c,d,e,g
e,g,h