Introduction to
Artificial Intelligence

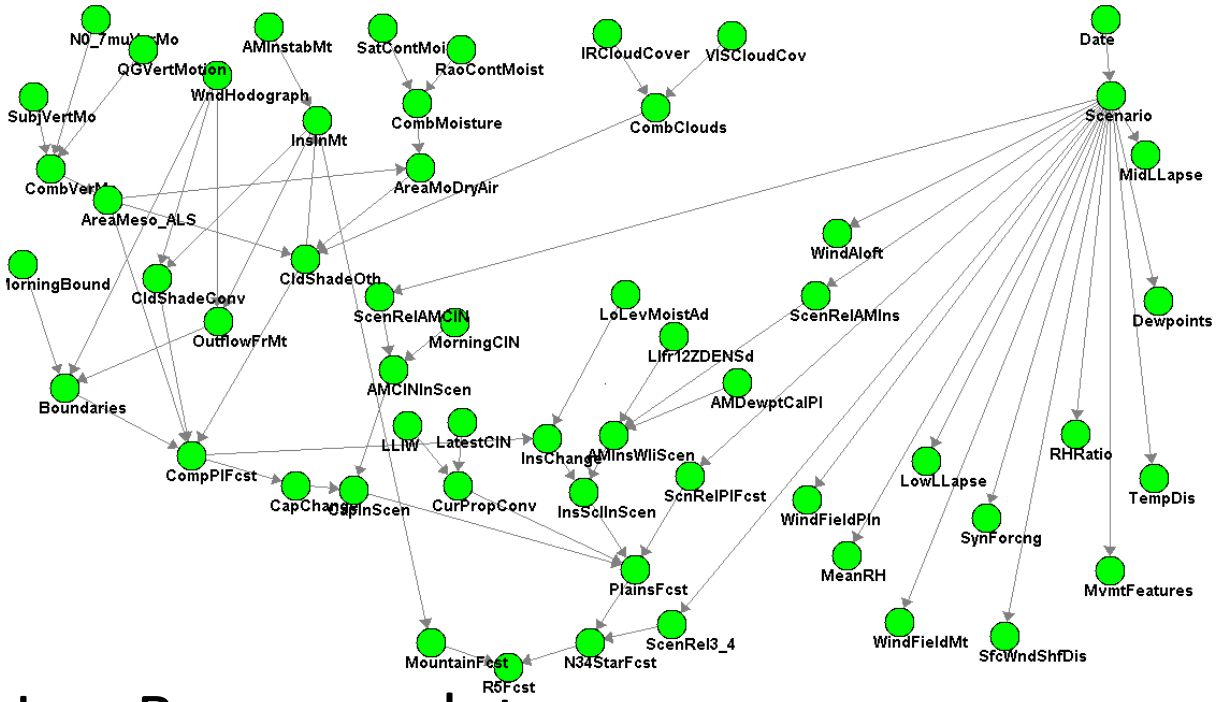
Lecture 13 – Approximate Inference

CS/CNS/EE 154

Andreas Krause

Bayesian networks

- Compact representation of distributions over large number of variables
- (Often) allows efficient exact inference (computing marginals, etc.)



JavaBayes applet

HailFinder

56 vars

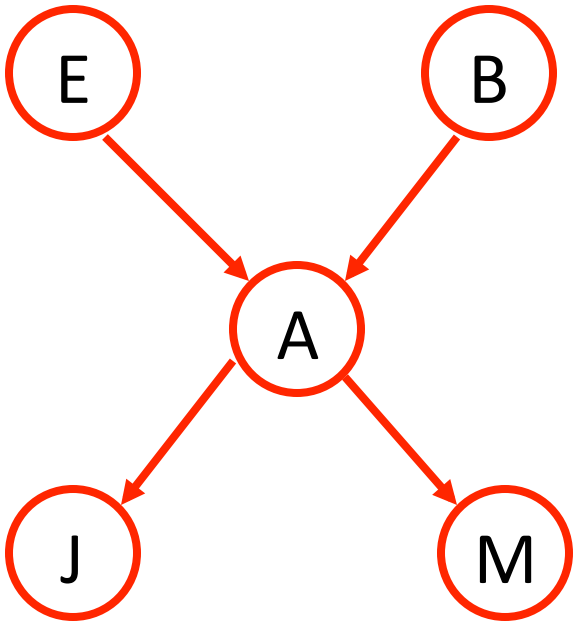
~ 3 states each

➔~10^{26} terms

> **10.000 years**

on Top supercomputers

Typical queries: Conditional distribution



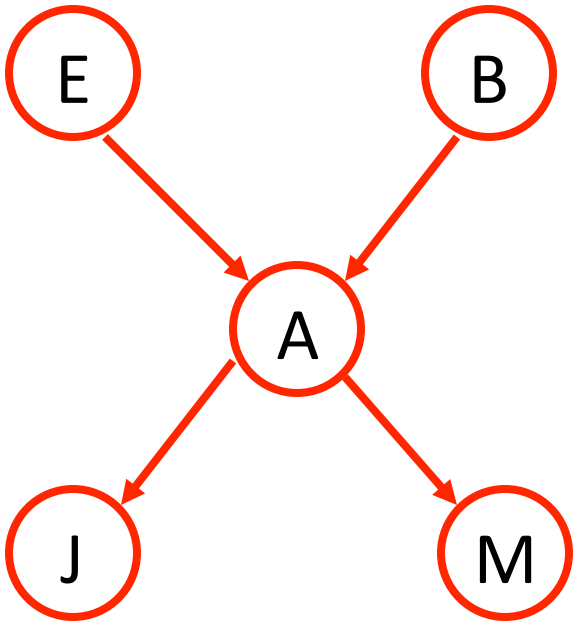
- Compute distribution of some variables given values for others

$P(E \mid J=T)$?

$P(E, B \mid J=T, M=F)$?

$= \frac{1}{z} P(E, B, J=T, M=F)$

Typical queries: Maximization



- MPE (Most probable explanation):

  Given values for some vars, compute most likely assignment to all remaining vars

$$(a^*, e^*, b^*) = \underset{e,b,a}{\operatorname{argmax}} \; P(E=e, B=b, A=a \mid J=T, M=F)$$

- MAP (Maximum a posteriori):

  Compute most likely assignment to some variables

More general than MPE

$$(e^*, b^*) = \underset{e,b}{\operatorname{argmax}} \; P(e, b \mid J=T, M=F)$$

Hardness of inference for general BNs

- Computing conditional distributions:
  - Exact solution: #P-complete
  - NP-hard to obtain any nontrivial approximation

    Eg. NP-hard to obtain $\hat{p}$ s.t. $|p - \hat{p}| < \frac{1}{2}$
- Maximization:
  - MPE: NP-complete
  - MAP: NP^{PP}-complete

- Inference in general BNs is really hard ☹

Inference

- Can exploit structure (conditional independence) to efficiently perform **exact inference** in many practical situations

- For BNs where exact inference is not possible, can use algorithms for **approximate inference** (later)

Variable elimination algorithm

- Given BN and Query P(X | **E**=**e**)
- Choose an ordering of $X_1,\ldots,X_n$
- Set up initial factors: $f_i = P(X_i \mid \mathbf{Pa_i})$
- For i =1:n, $X_i \notin \{X,\mathbf{E}\}$
  - Collect all factors f that include X_i
  - Generate new factor by marginalizing out X_i

$$g = \sum_{x_i} \prod_j f_j$$

  - Add g to set of factors
- Renormalize P(x,**e**) to get P(x | **e**)

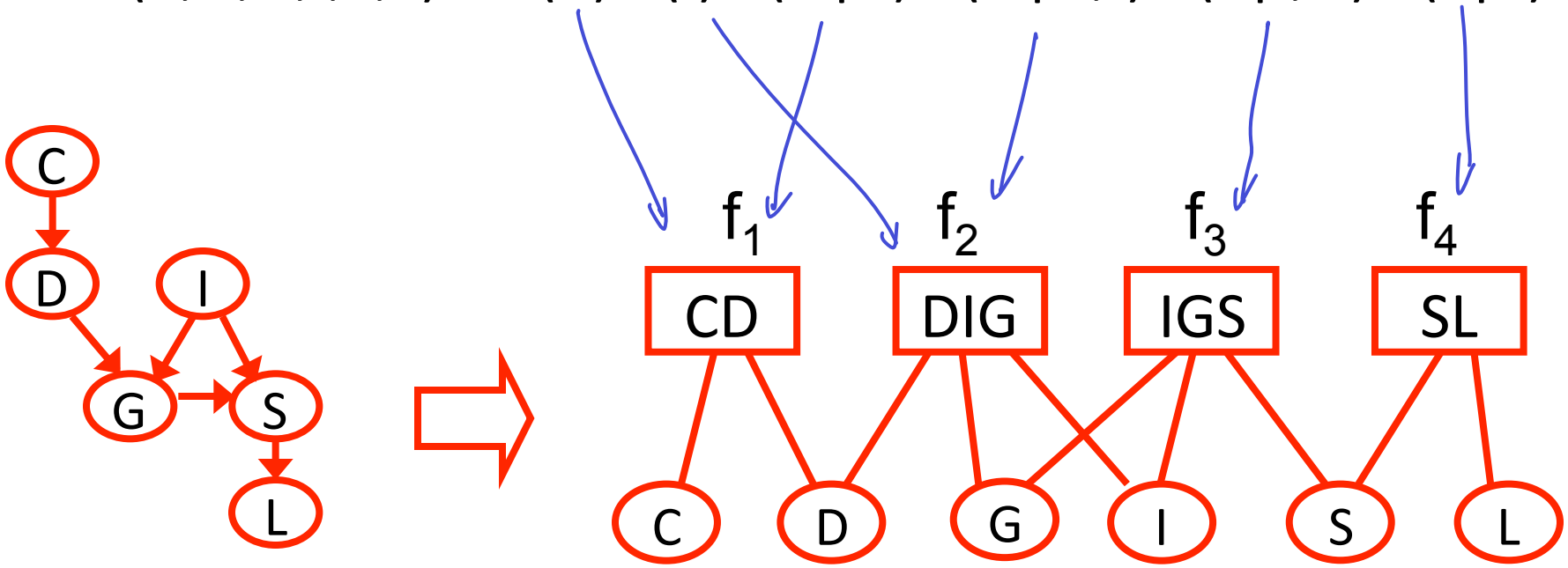
Reusing computation

- Often, want to compute conditional distributions of many variables, for fixed observations
- E.g., probability of *Pits* at different locations given observed *Breezes*
- Repeatedly performing variable elimination is wasteful (many factors are recomputed)
- Need right data-structure to avoid recomputation

➔ Message passing on factor graphs

Factor graphs

- P(C,D,G,I,S,L) = P(C) P(I) P(D|C) P(G|D,I) P(S|I,G) P(L|S)



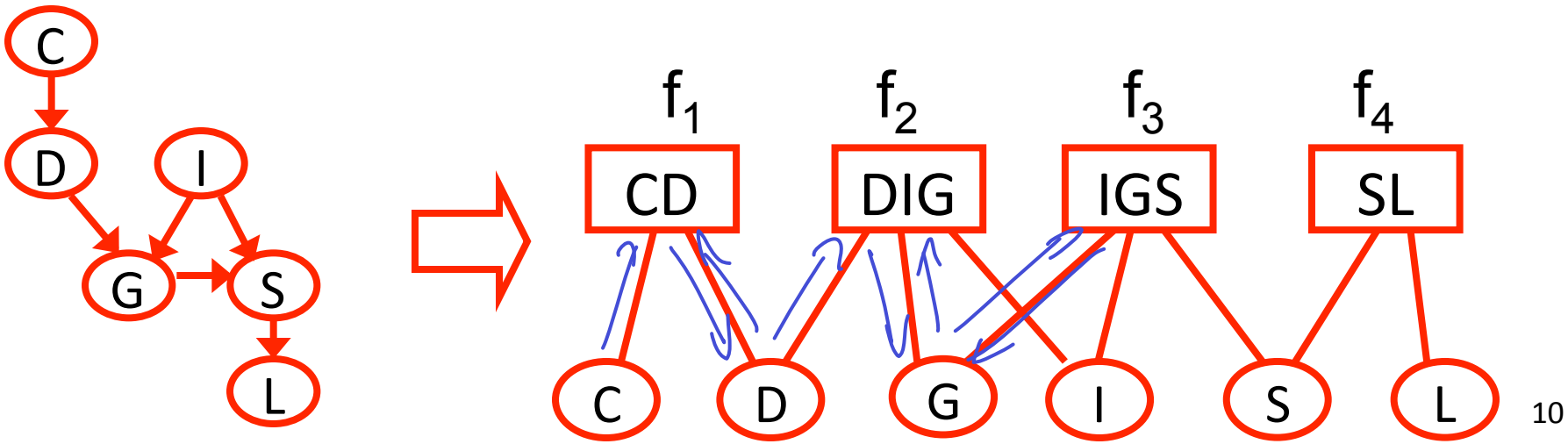
$f_1(C, D) = P(C)\,P(D|C)$

$f_2(D, I, G) = P(I)\,P(G|D, I)$

$f_3(I, G, S) = P(S|I, G)$

Factor graph

- A factor graph for a Bayesian network is a bipartite graph consisting of
  - Variables and
  - Factors
- Each factor is associated with a subset of variables, and all CPDs of the Bayesian network have to be assigned to one of the factor nodes



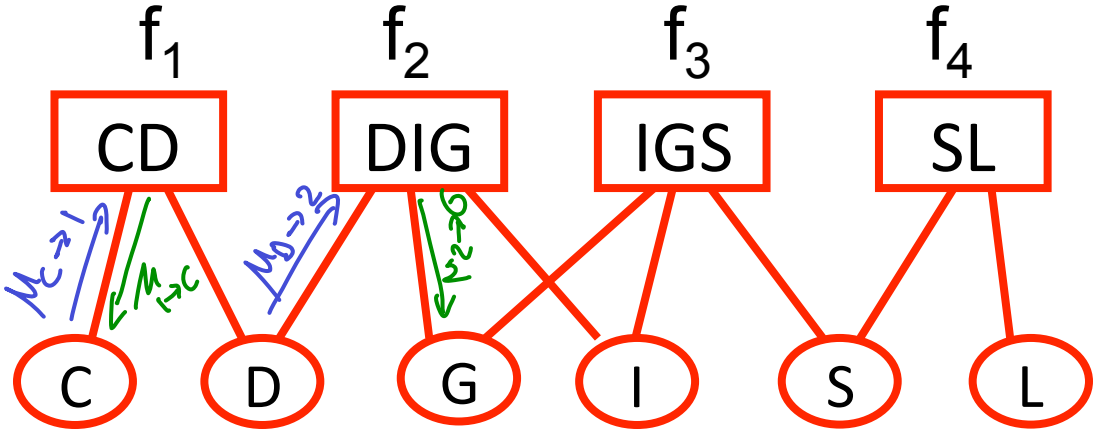
Sum-product message passing on factor graphs

- Messages from node *v* to factor *u*

$$\mu_{v \to u}(x_v) = \prod_{u' \in N(v) \setminus \{u\}} \mu_{u' \to v}(x_v)$$

- Messages from factor *u* to node *v*

$$\mu_{u \to v}(x_v) = \sum_{x_u \sim x_v} f_u(x_u) \prod_{v' \in N(u) \setminus \{v\}} \mu_{v' \to u}(x_{v'})$$

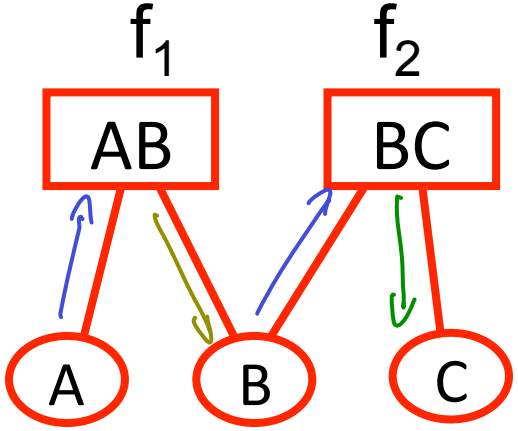


Example messages

$$\mu_{v\to u}(x_v) = \prod_{u'\in N(v)\setminus\{u\}} \mu_{u'\to v}(x_v)$$

$$\mu_{u\to v}(x_v) = \sum_{\mathbf{x}_u \sim x_v} f_u(\mathbf{x}_u) \prod_{v'\in N(u)\setminus\{v\}} \mu_{v'\to u}(x_{v'})$$

P(A)P(B|A) P(C|B)



$\mu_{A\to 1}(a) = 1$

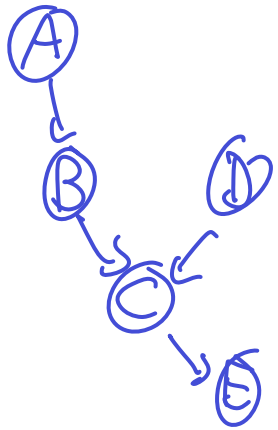
$\mu_{1\to B}(b) = \sum_a f_1(a,b)\cdot \mu_{A\to 1}(a) = \sum_a \underbrace{P(a)P(b|a)}_{P(a,b)} \cdot 1 = P(b)$

$\mu_{B\to 2}(b) = \mu_{1\to B}(b) = P(b)$

$\mu_{2\to C}(c) = \sum_b f_2(b,c)\cdot \mu_{B\to 2}(b) = \sum_b \underbrace{P(c|b)\cdot P(b)}_{P(b,c)} = P(c)$

Belief propagation on polytrees

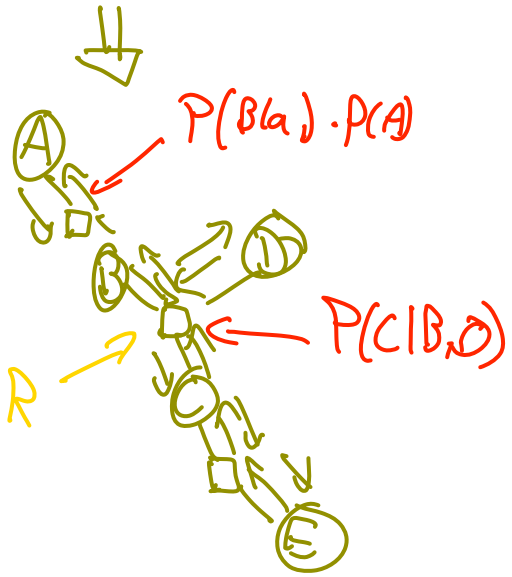
- Belief propagation (aka sum-product) is *exact* for polytree Bayesian networks
  - Factor graph of polytree is a tree
  - Choose one node as root
  - Send messages from leaves to root, and from root to leaves



- After convergence:

$$P(X_v = x_v) = \prod_{u \in N(v)} \mu_{u \to v}(x_v)$$

$$P(\mathbf{X}_u = \mathbf{x}_u) = f_u(\mathbf{x}_u) \prod_{v \in N(u)} \mu_{v \to u}(x_v)$$



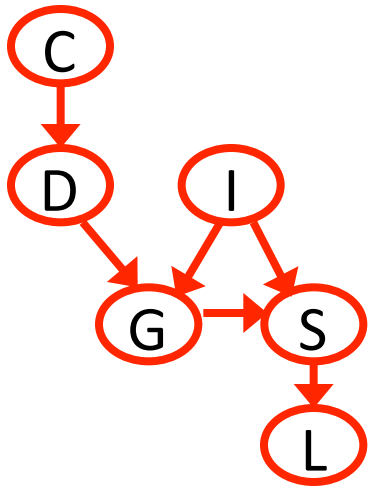
- Thus: immediately have correct values for **all** marginals!

What if we have loops?

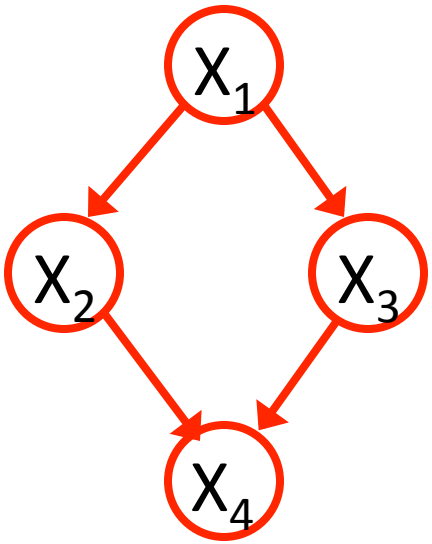
- Can still apply belief propagation even if we have loops
  - Just run it, close your eyes and hope for the best!
  - Use approximation:

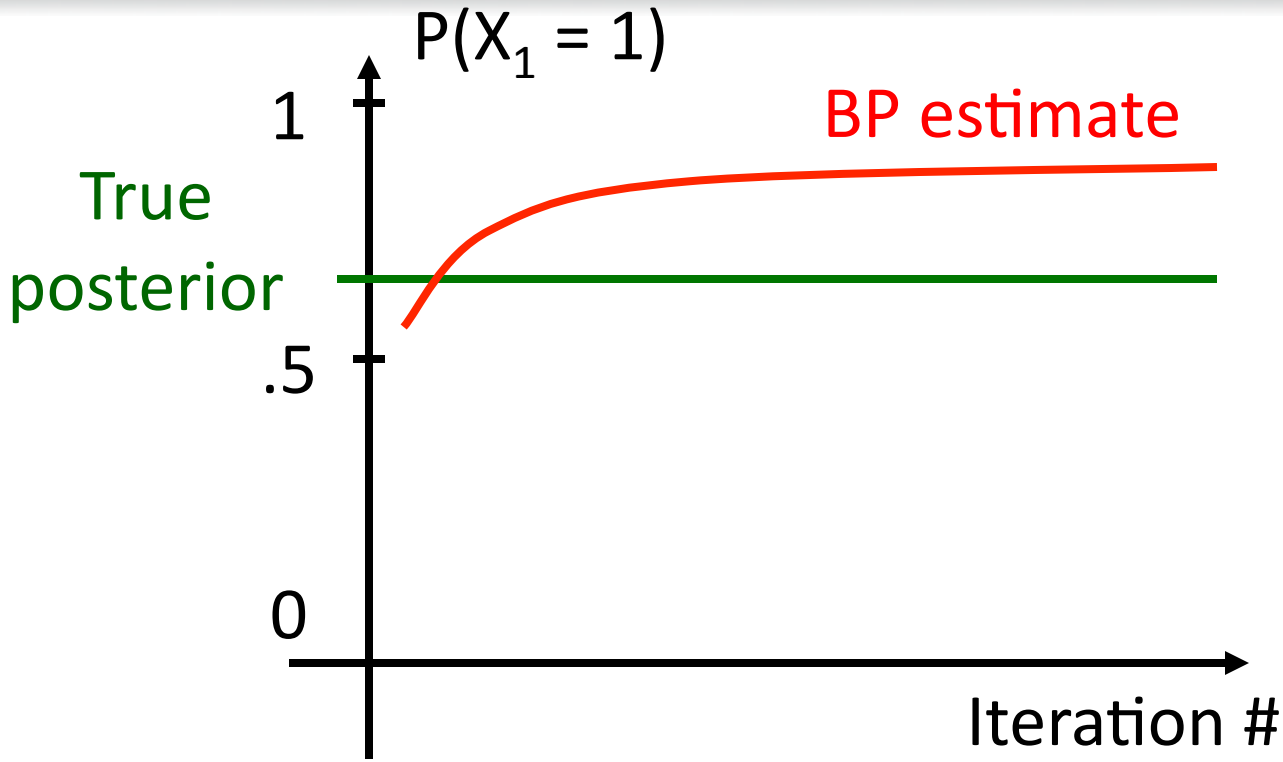
$$P(X_v = x_v) \approx \prod_{u \in N(v)} \mu_{u \to v}(x_v)$$

- In general, will not converge…
- Even if it converges, may converge to incorrect marginals…
- However, in practice often still useful!
  - E.g., turbo-codes, etc.
- "Loopy belief propagation"



Behavior of Loopy BP



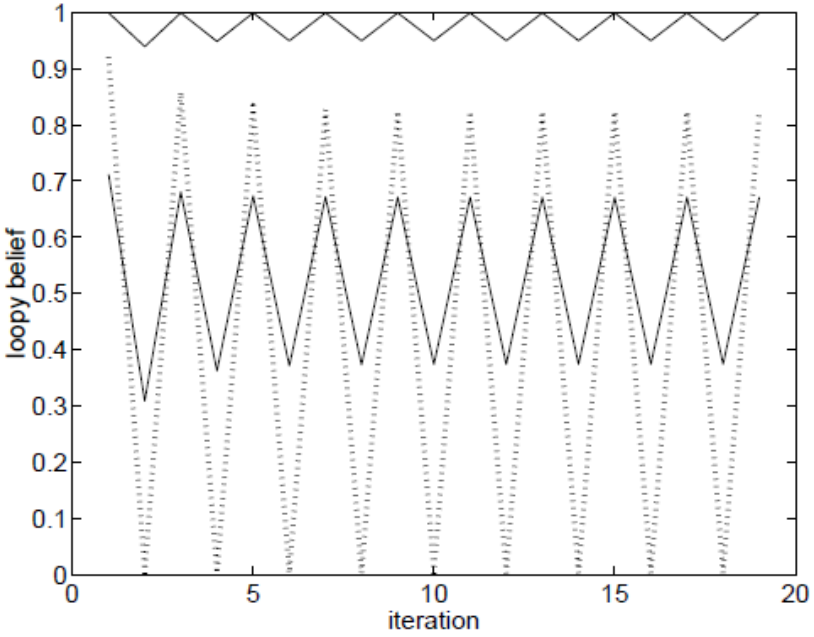


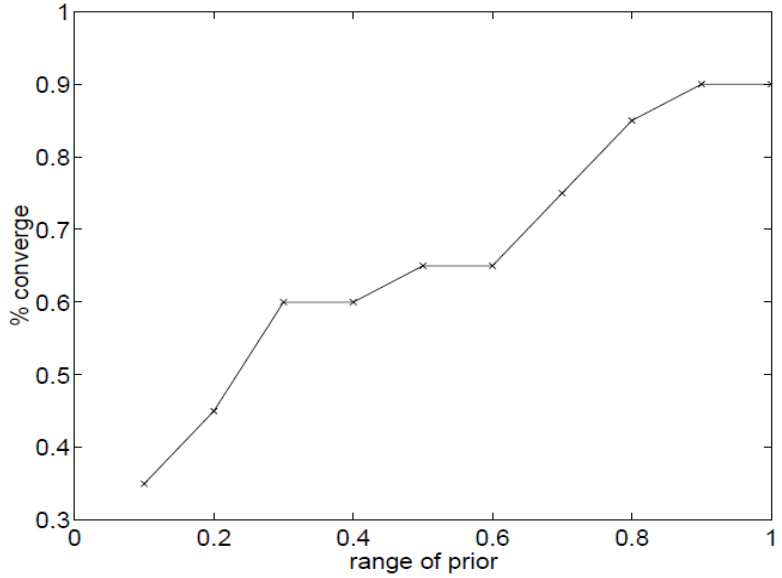
- Loopy BP multiplies same factors multiple times

  ➔ BP often overconfident

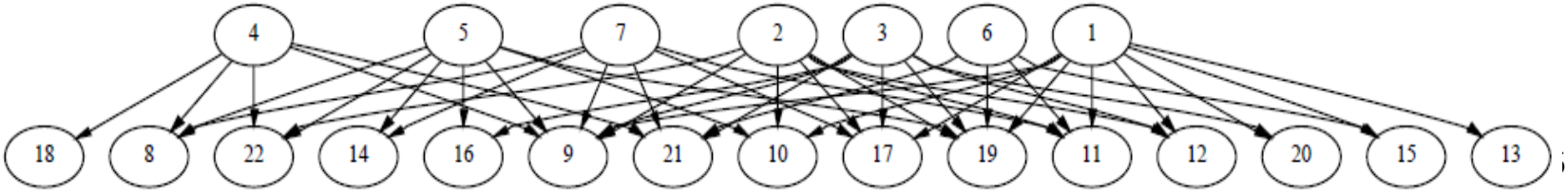
Does Loopy BP always converge?

- No! Can oscillate!
- Typically, oscillation the more severe the more "deterministic" the potentials



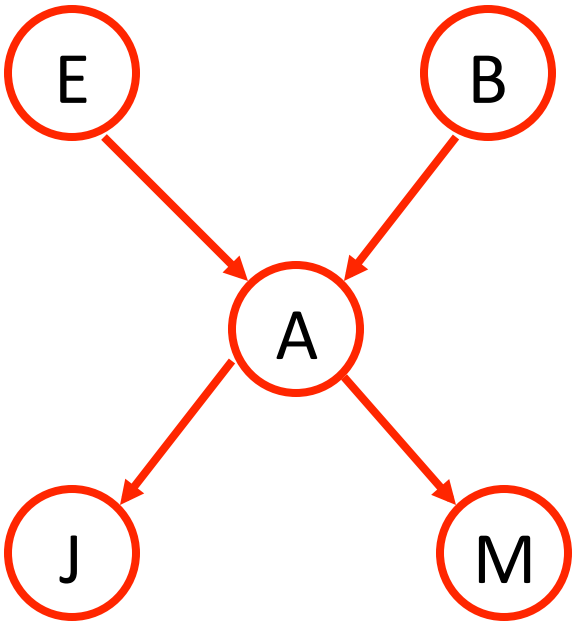


Graphs from K. Murphy UAI '99



What about MPE queries?

- E.g.,: What's the most likely assignment to the unobserved variables, given the observed ones?



$$(e^*, b^*, a^*) = \operatorname*{argmax}_{e,b,a} P(e, b, a \mid j, m)$$

$$= \operatorname*{argmax}_{e,b,a} P(e, b, a, j, m)$$

$$\max_{e,b,a} P(e, b, a, j, m)$$

$$= \max_{e,b,a} P(e)\,P(b)\,P(a \mid e, b)\,P(j \mid a)\,P(m \mid a)$$

$$= \max_{a} P(j \mid a)\,P(m \mid a) \max_{e} P(e) \underbrace{\max_{b} P(b)\,P(a \mid e, b)}_{g_b(e,a)}$$

- Use max-product
  (same as sum-product/BP, but with max instead of sums!)

Max-product message passing on factor graphs

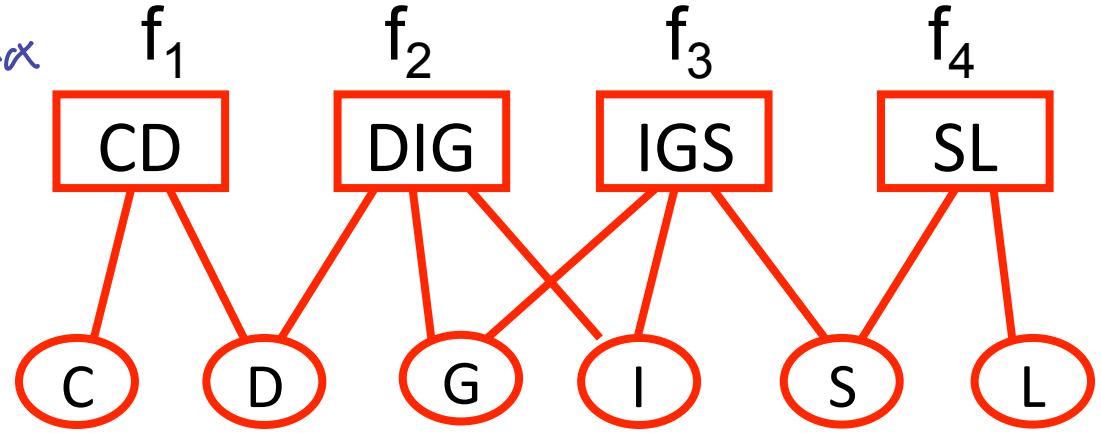
- Messages from nodes to factors

$$\mu_{v \to u}(x_v) = \prod_{u' \in N(v) \setminus \{u\}} \mu_{u' \to v}(x_v)$$

- Messages from factors to nodes

$$\mu_{u \to v}(x_v) = \max_{\mathbf{x}_u \sim x_v} f_u(\mathbf{x}_u) \prod_{v' \in N(u) \setminus \{v\}} \mu_{v' \to u}(x_{v'})$$

Replace $\sum$ by max



Sampling based inference

- So far: deterministic inference techniques
  - Variable elimination
  - (Loopy) belief propagation

- Will now introduce stochastic approximations
  - Algorithms that "randomize" to compute expectations
  - In contrast to the deterministic methods, guaranteed to converge to right answer (if wait looong enough..)
  - More exact, but slower than deterministic variants

Computing expectations

- Often, we're not necessarily interested in computing marginal distributions, but certain expectations:

- Moments (mean, variance, …)

$$\mathbb{E}_P[X^k] = \int x^k P(x) dx$$

- Event probabilities

$$P(X > c) = \mathbb{E}_P[I_{X>c}] = \int [x > c] P(x) dx$$

In general: $\mathbb{E}_P(f(X)) = \int P(x) f(x) dx$ for continuous

$= \sum_x P(x) f(x)$ for discrete

Sample approximations of expectations

- $x_1,\ldots,x_N$ samples from RV X
- Law of large numbers:

$$\mathbb{E}_P[f(X)] = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

- Hereby, the convergence is with probability 1 (almost sure convergence)
- Finite samples:

$$\mathbb{E}_P(f(X)) \simeq \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

How many samples do we need?

- Hoeffding inequality

  Suppose f is bounded in [0.C]. Then

$$P\left(\left|\mathbb{E}_P[f(X)] - \frac{1}{N}\sum_{i=1}^{N} f(x_i)\right| > \varepsilon\right) \leq 2\exp(-2N\varepsilon^2/C^2)$$

  Sample avg

- Thus, probability of error decreases exponentially in N!

  Suppose I want error at most ε with prob. at least $1-\delta$

$$\exp(-2N\varepsilon^2/C^2) \leq \delta/2$$
$$\Leftrightarrow \quad -2N\varepsilon^2/C^2 \leq \ln \delta/2$$
$$\Leftrightarrow \quad 2N\varepsilon^2/C^2 \geq \ln 2/\delta$$
$$\Leftrightarrow \quad N \geq \frac{1}{2}\frac{C^2}{\varepsilon^2}\ln\frac{2}{\delta}$$

- Need to be able to draw samples from P

Sampling from a Bernoulli distribution

- Most random number generators produce (approximately) uniformly distributed random numbers

- How can we draw samples from X ~ Bernoulli(p)?

$Y \sim U([0,1))$

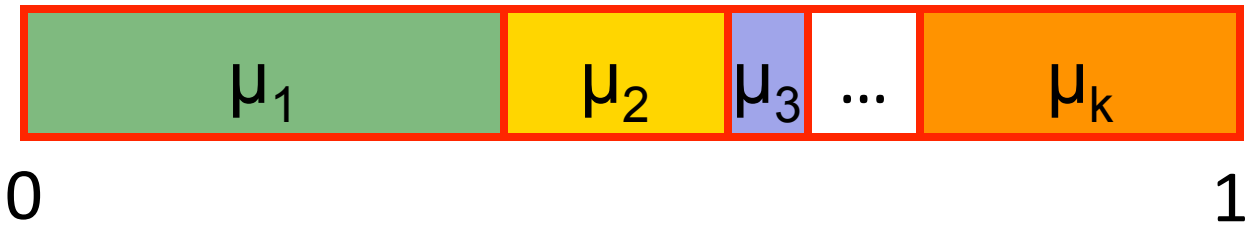
$X = 1$ if $Y < p$

$X = 0$ if $Y \geq p$

Sampling from a Multinomial

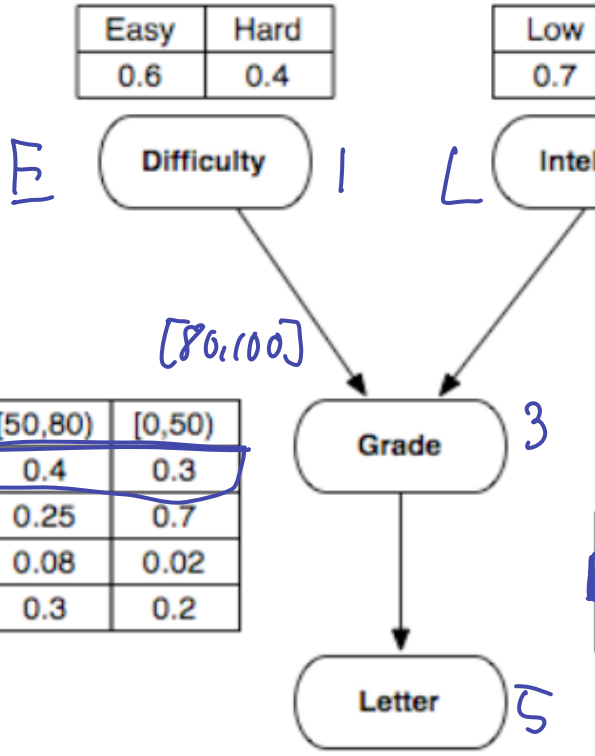
- $X \sim \text{Mult}([\mu_1,\ldots,\mu_k])$

  where $\mu_i = P(X=i)$; $\sum_i \mu_i = 1$



- Function g: [0,1]→{1,…,k} assigns state g(x) to each x
- Draw sample from uniform distribution on [0,1]
- Return $g^{-1}(x)$

Forward sampling from a BN



Easy	Hard
0.6	0.4

Low	High
0.7	0.3

Int	Diff	[80,100]	[50,80)	[0,50)
Low	Easy	0.3	0.4	0.3
Low	Hard	0.05	0.25	0.7
High	Easy	0.9	0.08	0.02
High	Hard	0.5	0.3	0.2

Int	Bad	Good
Low	0.95	0.05
High	0.2	0.8

Grade	Fail	Pass
[80,100]	0.1	0.9
[50,80)	0.4	0.6
[0,50)	0.99	0.01

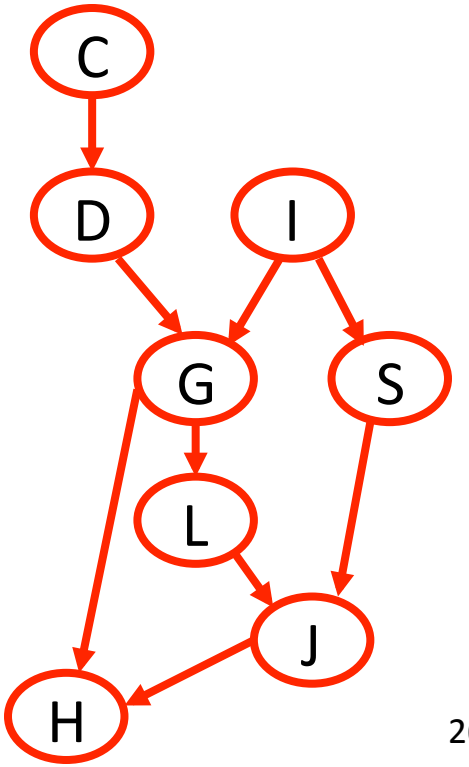
Sample

$\Rightarrow (D=E, I=L, G=[80,100], S=B, L=P)$

Monte Carlo sampling from a BN

- Sort variables in topological ordering $X_1,\ldots,X_n$
- For i = 1 to n do
  - Sample $x_i \sim P(X_i \mid X_1=x_1, \ldots, X_{i-1}=x_{i-1})$

- Works even with loopy models!



Computing probabilities through sampling

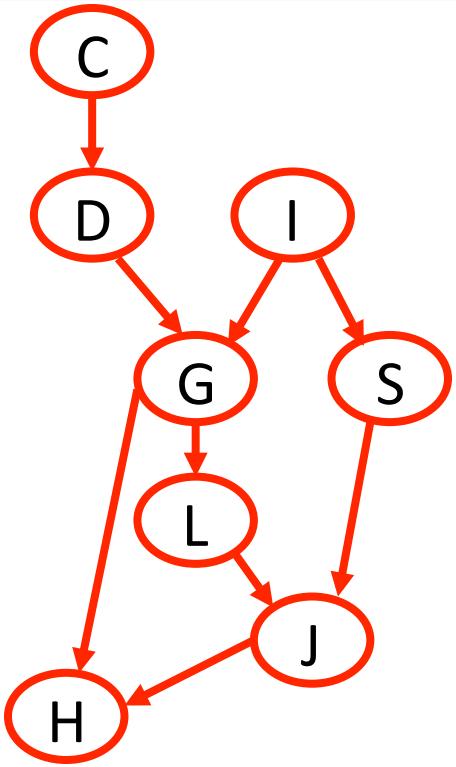
- Want to estimate probabilities
- Draw N samples from BN

$x_1 \ \ldots \ x_N$

- Marginals

$$P(L=T) \approx \frac{1}{N}\sum_{i=1}^{N} [L(x_i)=T] = \frac{Count(L=T)}{N}$$

$[L(x_i)=T]$ = 1 if $L(x_i)=T$, 0 otherwise



- Conditionals

$$P(L=T \mid H=F) = \frac{P(L=T, H=F)}{P(H=F)} \approx \frac{Count(L=T, H=F)}{Count(H=F)}$$

Rejection sampling

- Collect samples over all variables

$$\widehat{P}(\mathbf{X}_A = \mathbf{x}_A \mid \mathbf{X}_B = \mathbf{x}_B) \approx \frac{Count(\mathbf{x}_A, \mathbf{x}_B)}{Count(\mathbf{x}_B)}$$

- Throw away samples that disagree with $\mathbf{x_B}$
- Can be problematic if $P(\mathbf{X_B} = \mathbf{x_B})$ is rare event

Sample complexity for probability estimates

- Absolute error:

$$Prob\Big(\big|\widehat{P}(\mathbf{x}) - P(\mathbf{x})\big| > \varepsilon\Big) \leq$$

- Relative error:

$$Prob\Big(\widehat{P}(\mathbf{x}) < (1+\varepsilon)P(\mathbf{x})\Big) \leq 2\exp(-NP(\mathbf{x})\varepsilon^2/3)$$

if $P(x)$ exponentially small, need N exponentially large

Sampling from rare events

- Estimating conditional probabilities $P(X_A \mid \mathbf{X_B}=\mathbf{x_B})$ using rejection sampling is hard!
  - The more observations, the unlikelier $P(\mathbf{X_B} = \mathbf{x_B})$ becomes

- Want to directly sample from posterior distribution!

Gibbs sampling

- Start with initial assignment $\mathbf{x}^{(0)}$ to all variables
- For t = 1 to ∞ do
  - Set $\mathbf{x}^{(t)} = \mathbf{x}^{(t-1)}$
  - For each variable X_i
    - Set $\mathbf{v}_i$ = values of all $\mathbf{x}^{(t)}$ except x_i
    - Sample $x^{(t)}_i$ from $P(X_i \mid \mathbf{v}_i)$
- For large enough *t*, sampling distribution will be "close" to true posterior distribution!
- **Key challenge**: Computing conditional distributions $P(X_i \mid \mathbf{v}_i)$