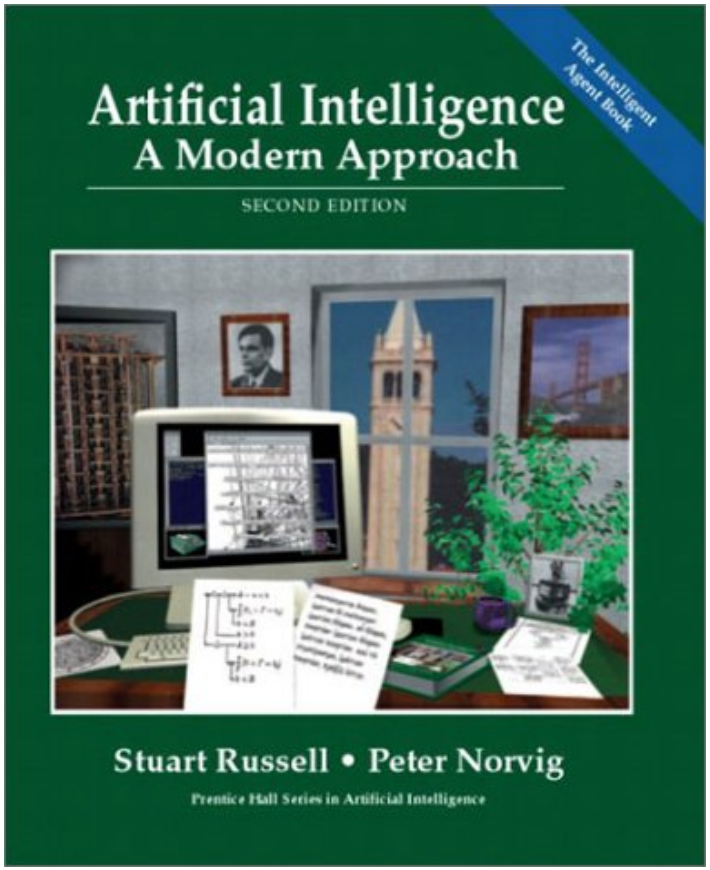
Web-Mining Agents
Data Mining

Prof. Dr. Ralf Möller
Dr. Özgür L. Özçep

Universität zu Lübeck
Institut für Informationssysteme

Tanya Braun (Lab)

Literature



- Chapter 14 (Section 1 and 2)

Syntax and Semantics of Bayesian Networks

Bayesian networks

- A simple, graphical notation for conditional independence assertions and hence for compact specification of full joint distributions

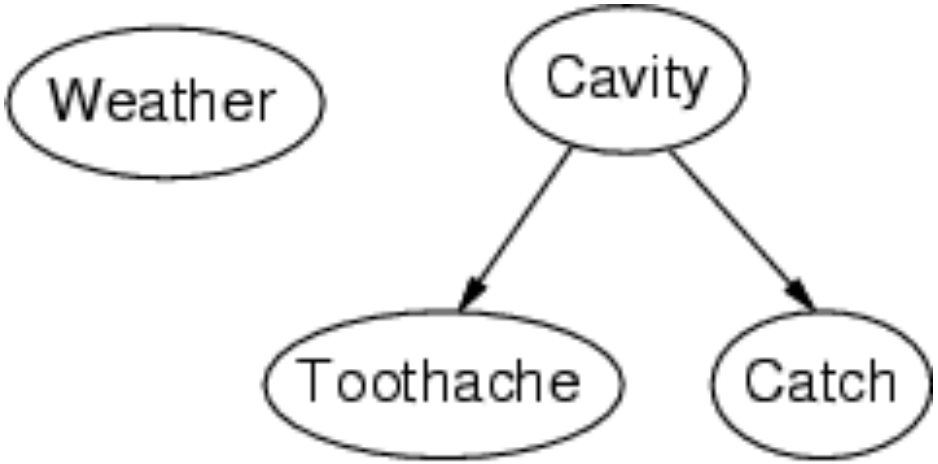
- Syntax:
  - a set of nodes, one per variable
  - a directed, acyclic graph (link ≈ "directly influences")
  - a conditional distribution for each node given its parents:

$$\mathbf{P}\,(X_i \mid \text{Parents}\,(X_i))$$

- In the simplest case, conditional distribution represented as a conditional probability table (CPT) giving the distribution over X_i for each combination of parent values

Example

- Topology of network encodes conditional independence assertions:



- Weather is independent of the other variables
- Toothache and Catch are conditionally independent given Cavity

Example

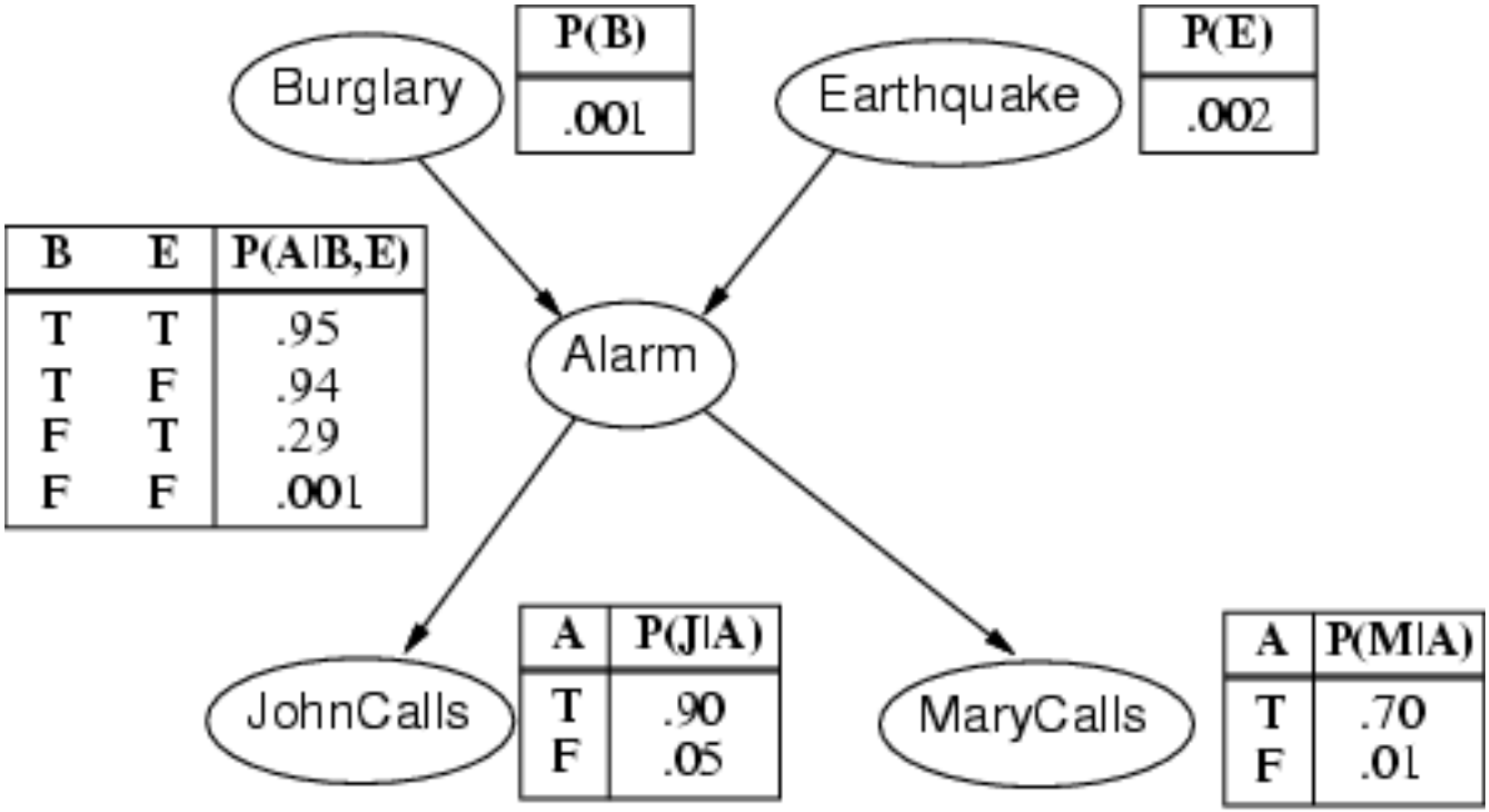
- I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?

- Variables: Burglary, Earthquake, Alarm, JohnCalls, MaryCalls

- In this case, the network topology reflects "causal" knowledge:
  - A burglar can set the alarm off
  - An earthquake can set the alarm off
  - The alarm can cause Mary to call
  - The alarm can cause John to call

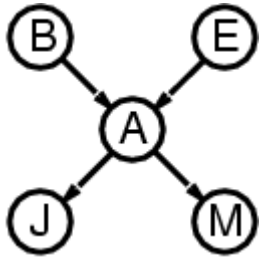
A formal in-depth treatment of causality will be given in lectures 5 to 9

Example contd.



Compactness

- A CPT for Boolean X_i with k Boolean parents has 2^k rows for the combinations of parent values

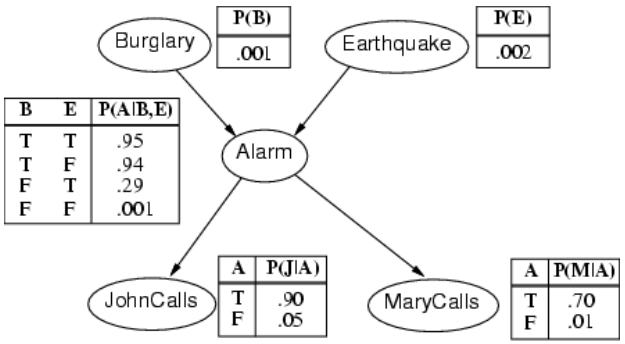


- Each row requires one number p for X_i = true (the number for X_i = false is just 1-p)
- If each variable has no more than k parents, the complete network requires $O(n \cdot 2^k)$ numbers
- i.e., grows linearly with n, vs. $O(2^n)$ for the full joint distribution
- For burglary net, 1 + 1 + 4 + 2 + 2 = 10 numbers (vs. 2^5-1 = 31)

Semantics

The full joint distribution of a BN is defined as the product of the local conditional distributions:

$$\mathbf{P}(X_1, \ldots, X_n) = \Pi_{i=1}^{n} \mathbf{P}(X_i \mid \text{Parents}(X_i))$$



e.g., $P(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$

$= P(j \mid a)\, P(m \mid a)\, P(a \mid \neg b, \neg e)\, P(\neg b)\, P(\neg e)$

$= 0.90 \times 0.7 \times 0.001 \times 0.999 \times 0.998$

≈ 0.00063

Constructing Bayesian networks

- 1. Choose an ordering of variables $X_1, \ldots, X_n$
- 2. For $i = 1$ to n
  - add X_i to the network
  - select parents from $X_1, \ldots, X_{i-1}$ such that

$$\mathbf{P}(X_i \mid \text{Parents}(X_i)) = \mathbf{P}(X_i \mid X_1, \ldots X_{i-1})$$

This choice of parents guarantees:

$$\mathbf{P}(X_1, \ldots, X_n) = \prod_{i=1}^{n} \mathbf{P}(X_i \mid X_1, \ldots, X_{i-1}) \quad \text{(chain rule)}$$

$$= \prod_{i=1}^{n} \mathbf{P}(X_i \mid \text{Parents}(X_i)) \quad \text{(by construction)}$$

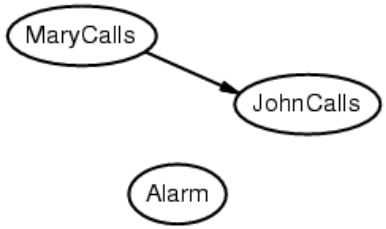
Example

- Suppose we choose the ordering M, J, A, B, E

$\boldsymbol{P}(J \mid M) = \boldsymbol{P}(J)?$

Example

- Suppose we choose the ordering M, J, A, B, E

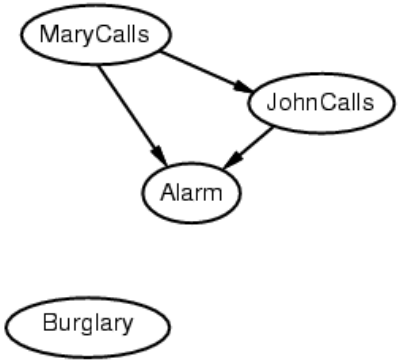


P(J | M) = **P**(J)? **No**

P(A | J, M) = **P**(A)? **P**(A | J, M) = **P**(A | J)?

Example

- Suppose we choose the ordering M, J, A, B, E



P(J | M) = **P**(J)? **No**

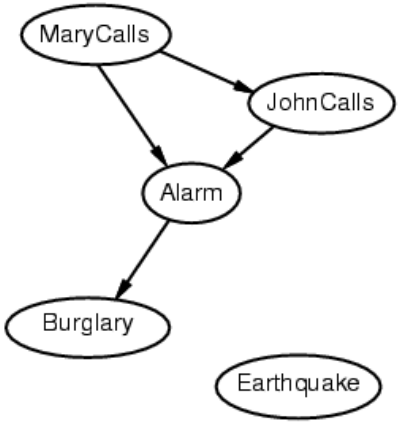
P(A | J, M) = **P**(A)? **P**(A | J, M) = **P**(A | J)? **No**

P(B | A, J, M) = **P**(B | A)?

P(B | A, J, M) = **P**(B)?

Example

- Suppose we choose the ordering M, J, A, B, E



P(J | M) = **P**(J)? **No**

P(A | J, M) = **P**(A | J)? **P**(A | J, M) = **P**(A)? **No**

P(B | A, J, M) = **P**(B | A)? **Yes**

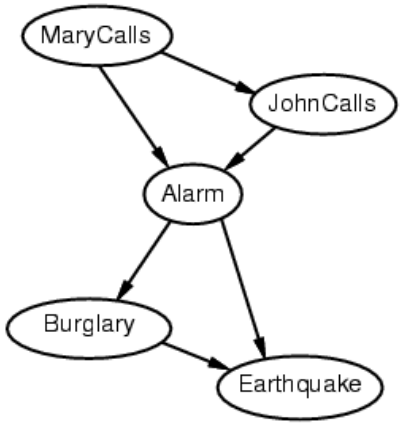
P(B | A, J, M) = **P**(B)? **No**

P(E | B, A ,J, M) = **P**(E | A)?

P(E | B, A, J, M) = **P**(E | A, B)?

Example

- Suppose we choose the ordering M, J, A, B, E



P(J | M) = **P**(J)? **No**

P(A | J, M) = **P**(A | J)? **P**(A | J, M) = **P**(A)? **No**

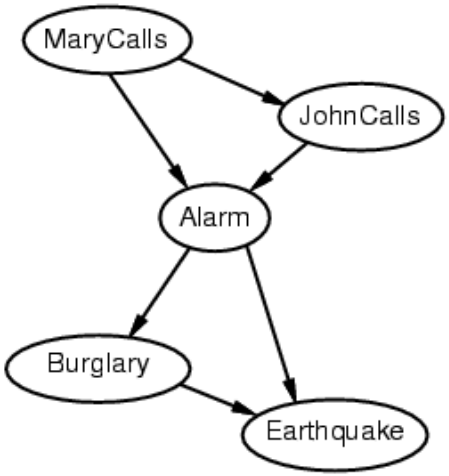
P(B | A, J, M) = **P**(B | A)? **Yes**

P(B | A, J, M) = **P**(B)? **No**

P(E | B, A ,J, M) = **P**(E | A)? **No**

P(E | B, A, J, M) = **P**(E | A, B)? **Yes**

Example contd.

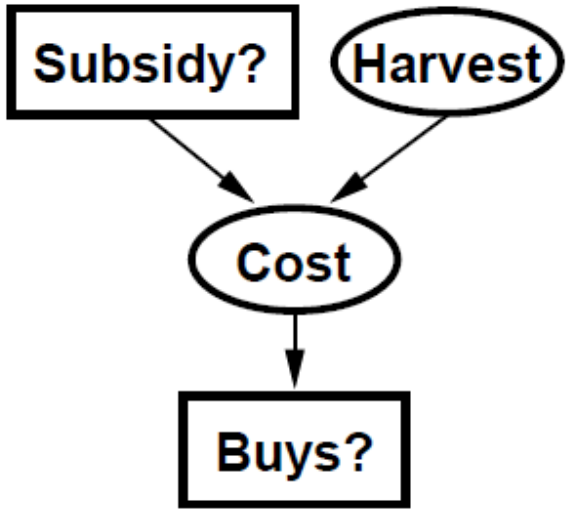


- Deciding conditional independence is hard in non-causal directions
- (Causal models and conditional independence seem hardwired for humans!)
- Network is less compact: 1 + 2 + 4 + 2 + 4 = 13 numbers needed instead of 10.

Hybrid (discrete+continuous) networks

BNs can also deal with continuous RVs or even hybrid ones

Discrete (*Subsidy?* and *Buys?*); continuous (*Harvest* and *Cost*)



Option 1: discretization—possibly large errors, large CPTs
Option 2: finitely parameterized canonical families

1) Continuous variable, discrete+continuous parents (e.g., *Cost*)
2) Discrete variable, continuous parents (e.g., *Buys?*)

Exact Inference on Bayesian Networks

Inference tasks

Simple queries: compute posterior marginal $\mathbf{P}(X_i|\mathbf{E}=\mathbf{e})$
e.g., $P(NoGas|Gauge=empty, Lights=on, Starts=false)$

Conjunctive queries: $\mathbf{P}(X_i, X_j|\mathbf{E}=\mathbf{e}) = \mathbf{P}(X_i|\mathbf{E}=\mathbf{e})\mathbf{P}(X_j|X_i, \mathbf{E}=\mathbf{e})$

Optimal decisions: decision networks include utility information;
probabilistic inference required for $P(outcome|action, evidence)$

Value of information: which evidence to seek next?

Sensitivity analysis: which probability values are most critical?

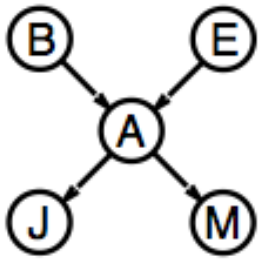
Explanation: why do I need a new starter motor?

Inference by enumeration

Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Simple query on the burglary network:

$$\begin{aligned}&\mathbf{P}(B|j,m)\\&= \mathbf{P}(B,j,m)/P(j,m)\\&= \alpha\mathbf{P}(B,j,m)\\&= \alpha \textstyle\sum_e \sum_a \mathbf{P}(B,e,a,j,m)\end{aligned}$$



Rewrite full joint entries using product of CPT entries:

$$\begin{aligned}&\mathbf{P}(B|j,m)\\&= \alpha \textstyle\sum_e \sum_a \mathbf{P}(B)P(e)\mathbf{P}(a|B,e)P(j|a)P(m|a)\\&= \alpha\mathbf{P}(B) \textstyle\sum_e P(e) \sum_a \mathbf{P}(a|B,e)P(j|a)P(m|a)\end{aligned}$$

Recursive depth-first enumeration: $O(n)$ space, $O(d^n)$ time

(Calculation over tree sum-product evaluation tree with path length n (=number of RVs) and degree d (=maximal number domain elements))

Enumeration algorithm

function ENUMERATION-ASK(X, $\mathbf{e}$, bn) **returns** a distribution over X
  inputs: X, the query variable
    $\mathbf{e}$, observed values for variables $\mathbf{E}$
    bn, a Bayesian network with variables $\{X\} \cup \mathbf{E} \cup \mathbf{Y}$

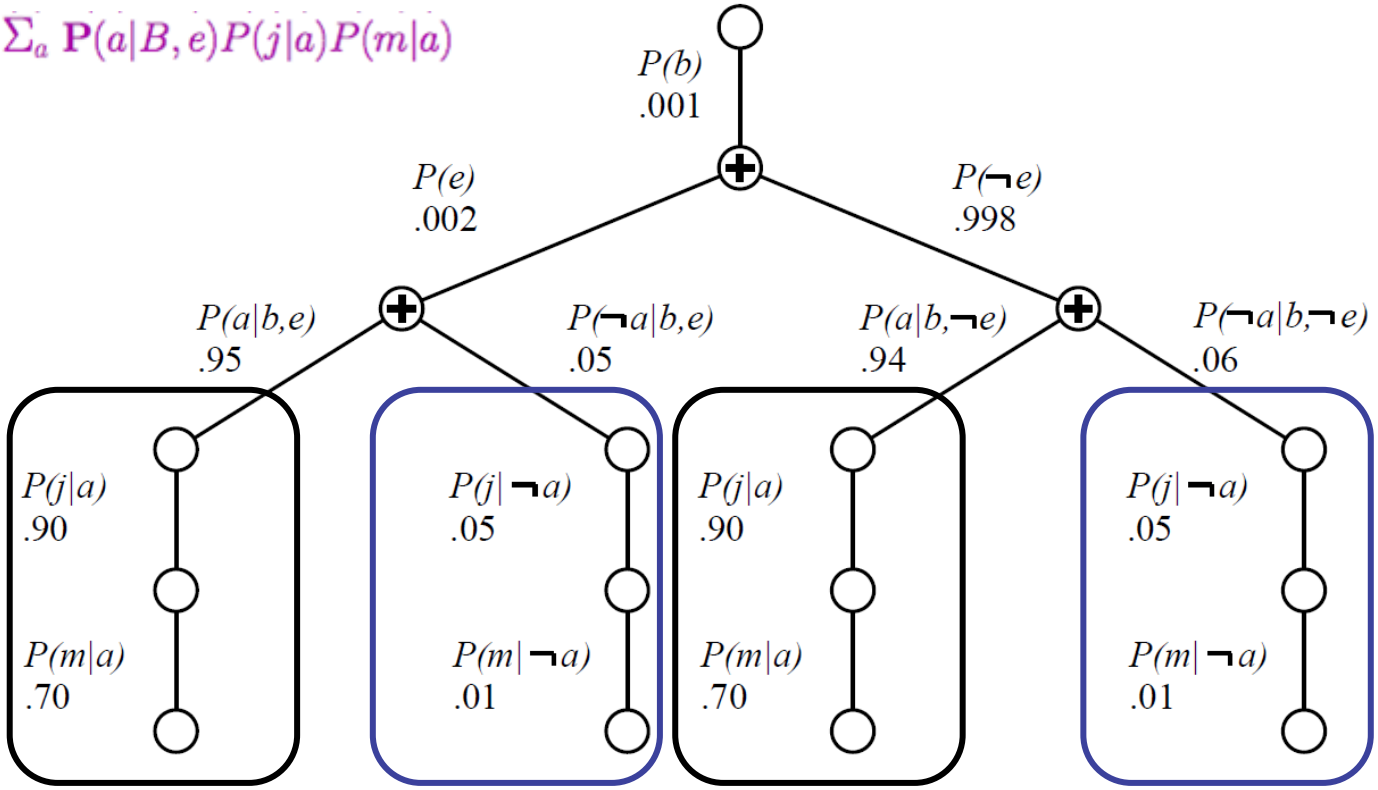
  $\mathbf{Q}(X) \leftarrow$ a distribution over X, initially empty
  for each value x_i of X **do**
    extend $\mathbf{e}$ with value x_i for X
    $\mathbf{Q}(x_i) \leftarrow$ ENUMERATE-ALL(VARS[bn], $\mathbf{e}$)
  return NORMALIZE($\mathbf{Q}(X)$)

---

function ENUMERATE-ALL($vars$, $\mathbf{e}$) **returns** a real number
  if EMPTY?($vars$) **then return** 1.0
  $Y \leftarrow$ FIRST($vars$)
  if Y has value y in $\mathbf{e}$
    then return $P(y \mid Pa(Y)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}$)
    else return $\Sigma_y \; P(y \mid Pa(Y)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}_y$)
      where $\mathbf{e}_y$ is $\mathbf{e}$ extended with $Y = y$

Evaluation Tree

$\mathbf{P}(B)\ \Sigma_e\ P(e)\ \Sigma_a\ \mathbf{P}(a|B,e)P(j|a)P(m|a)$



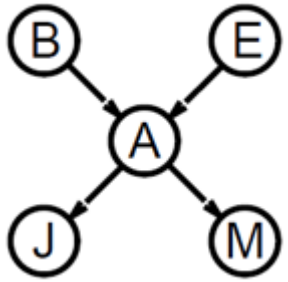
Enumeration is inefficient: repeated computation
e.g., computes $P(j|a)P(m|a)$ for each value of e

Can do better with Variable Elimination (VE)

Basic Objects of VE

$$\mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a|B,e) P(j|a) P(m|a)$$

- Track objects called factors (right-to-left, in tree: bottom up)
- Initial factors are local CPTs



$$\underbrace{P(B)}_{f_B(B)} \quad \underbrace{P(J|A)}_{f_J(A,J)} \quad \underbrace{P(A|B,E)}_{f_A(A,B,E)}$$

P(m|A)

$f_M(A)$

- During elimination create new factors
- Form of Dynamic Programming

Basic Operations: Pointwise Product

- Pointwise product of factors $\mathbf{f}_1$ and $\mathbf{f}_2$
  - for example: $\mathbf{f}_1(A,B) * \mathbf{f}_2(B,C) = \mathbf{f}(A,B,C)$
  - in general:
    $\mathbf{f}_1(X_1,\ldots,X_j,Y_1,\ldots,Y_k) * \mathbf{f}_2(Y_1,\ldots,Y_k,Z_1,\ldots,Z_l) = \mathbf{f}(X_1,\ldots,X_j,Y_1,\ldots,Y_k,Z_1,\ldots,Z_l)$
  - has 2^{j+k+l} entries (if all variables are binary)

Join by pointwise product

A	$f_{JM}(A)$
T	.9 * .7
F	.05 * .01

=

A	$f_J(A)$
T	.9
F	.05

A	$f_M(A)$
T	.7
F	.01

$$\mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a|B,e) P(j|a) P(m|a)$$

A	B	E	$f_{AJM}(A,B,E)$
T	T	T	.95 * .63
T	T	F	.94 * .63
T	F	T	.29 * .63
T	F	F	.001 * .63
F	T	T	.05 * .0005
F	T	F	.06 * .0005
F	F	T	.71 * .0005
F	F	F	.999 * .0005

=

A	B	E	$f_A \cdots(A,B,E)$
T	T	T	.95
T	T	F	.94
T	F	T	.29
T	F	F	.001
F	T	T	.05
F	T	F	.06
F	F	T	.71
F	F	F	.999

A	$f_{JM}(A)$
T	.63
F	.0005

Basic Operations: Summing out

- Summing out a variable from a product of factors
  - Move any constant factors outside the summation
  - Add up submatrices in pointwise product of remaining factors

$$\Sigma_x f_1 * \ldots * f_k = f_1 * \ldots * f_i * \Sigma_x f_{i+1} * \ldots * f_k$$
$$= f_1 * \ldots * f_i * f_X$$

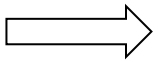
assuming $\mathbf{f}_1, \ldots, \mathbf{f}_i$ do not depend on X

Summing out

$$\mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a|B,e) P(j|a) P(m|a)$$

A	B	E	$f_{AJM}(A,B,E)$
T	T	T	.95 * .63
T	T	F	.94 * .63
T	F	T	.29 * .63
T	F	F	.001 * .63
F	T	T	.05 * .0005
F	T	F	.06 * .0005
F	F	T	.71 * .0005
F	F	F	.999 * .0005

Summing out a



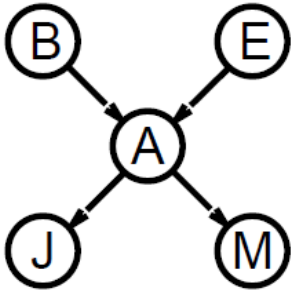
B	E	$f_{\bar{A}JM}(B,E)$
T	T	.95 * .63 + .05 * .0005 = .5985
T	F	.94 * .63 + .06 * .0005 = .5922
F	T	.29 * .63 + .71 * .0005 = .1830
F	F	.001 * .63 + .999 * .0005 = .001129

VE result for Burglary Example

$$
\begin{aligned}
&\mathbf{P}(B|j,m) \\
&\quad = \alpha \underbrace{\mathbf{P}(B)}_{B} \Sigma_e \underbrace{P(e)}_{E} \Sigma_a \underbrace{\mathbf{P}(a|B,e)}_{A} \underbrace{P(j|a)}_{J} \underbrace{P(m|a)}_{M} \\
&\quad = \alpha \mathbf{P}(B) \Sigma_e P(e) \Sigma_a \mathbf{P}(a|B,e) P(j|a) f_M(a) \\
&\quad = \alpha \mathbf{P}(B) \Sigma_e P(e) \Sigma_a \mathbf{P}(a|B,e) f_J(a) f_M(a) \\
&\quad = \alpha \mathbf{P}(B) \Sigma_e P(e) \Sigma_a f_A(a,b,e) f_J(a) f_M(a) \\
&\quad = \alpha \mathbf{P}(B) \Sigma_e P(e) f_{\bar{A}JM}(b,e) \text{ (sum out } A\text{)} \\
&\quad = \alpha \mathbf{P}(B) f_{\bar{E}\bar{A}JM}(b) \text{ (sum out } E\text{)} \\
&\quad = \alpha f_B(b) \times f_{\bar{E}\bar{A}JM}(b)
\end{aligned}
$$

Optimization by Finding Irrelevant Variables

Consider the query $P(JohnCalls|Burglary = true)$



$$P(J|b) = \alpha P(b) \sum_e P(e) \sum_a P(a|b,e) P(J|a) \sum_m P(m|a)$$

Sum over m is identically 1; M is **irrelevant** to the query

Thm 1: Y is irrelevant unless $Y \in Ancestors(\{X\} \cup \mathbf{E})$

Here, $X = JohnCalls$, $\mathbf{E} = \{Burglary\}$, and
$Ancestors(\{X\} \cup \mathbf{E}) =$ {Alarm, Earthquake, Burglary}
so $MaryCalls$ is irrelevant

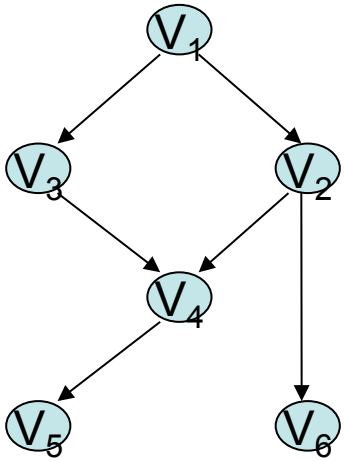
Irrelevant variables can be identified by a graph theoretical criterion on the associated moral graph by m-separation

What else can we do?

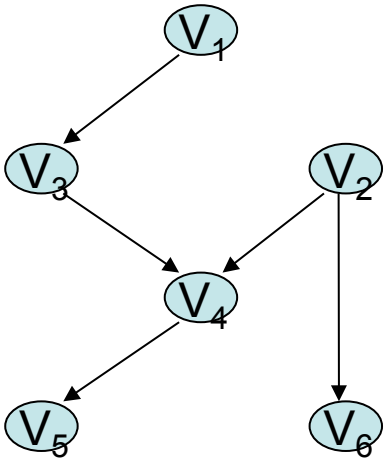
- General methodology: Consider special cases (special data structures)
- Here: Consider polytrees (singly connected networks)
- Following slides on Pearl‘s belief propagation algorithm based on slides by Tomas Singliar (adapted by Daniel Lowd)

Polytree Bayesian Networks

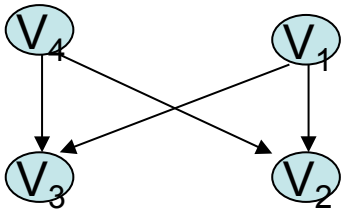
- Bayesian network graph is a *polytree (or singly connected network)* iff there is at most one path between any two nodes, V_i and V_k (iff the underlying non-directed graph is a tree)
- Exact BN inference is NP-hard but O(n) on polytree
- (Non-)Examples:



-



+



-

Our inference task

- We know the values of some evidence variables E:

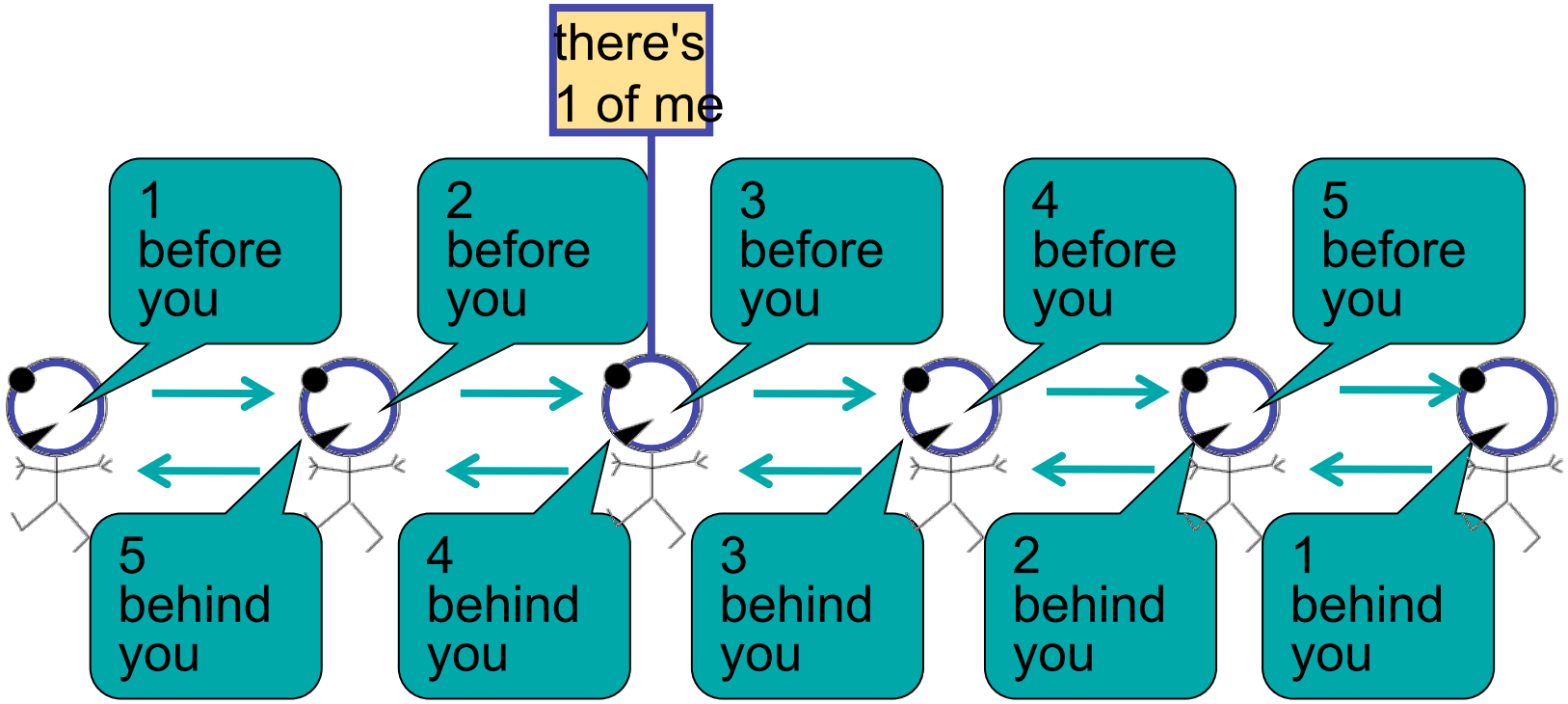
$$V_{e_1},\ldots,V_{e_{|E|}}$$

- We wish to compute the conditional probability $P(X_i \mid E)$ for all non-evidence variables X_i.

- IDEA:
  - Do variable elimination in parallel (locally on each node/ RV)
  - Send/receive required knowledge to/from other nodes

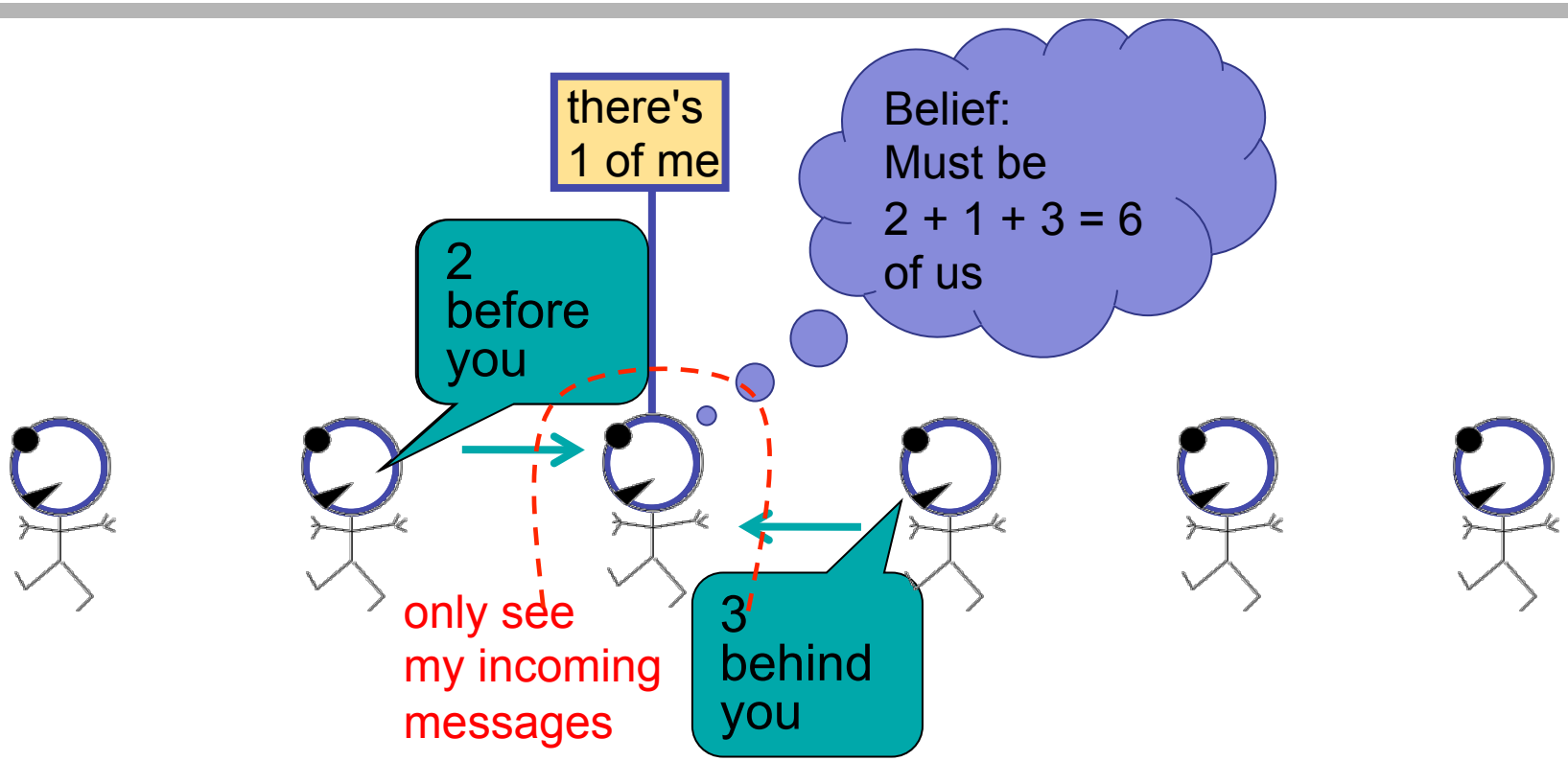
- Following soldier counting example on message passing adapted from slides of M. Gormley

Counting soldiers example (linear)

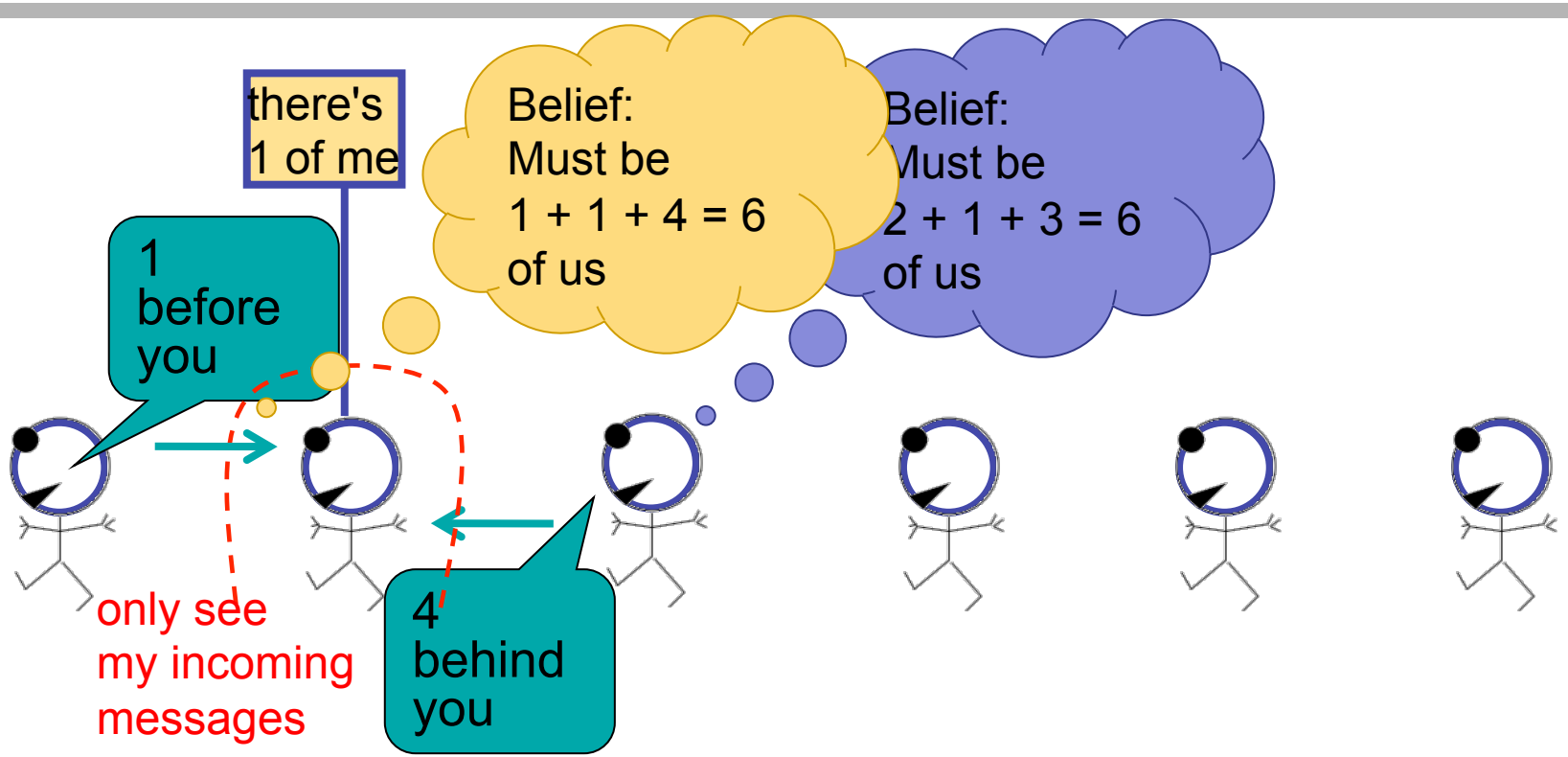


adapted from MacKay (2003): Information Theory, Inference and Learning Algorithms.

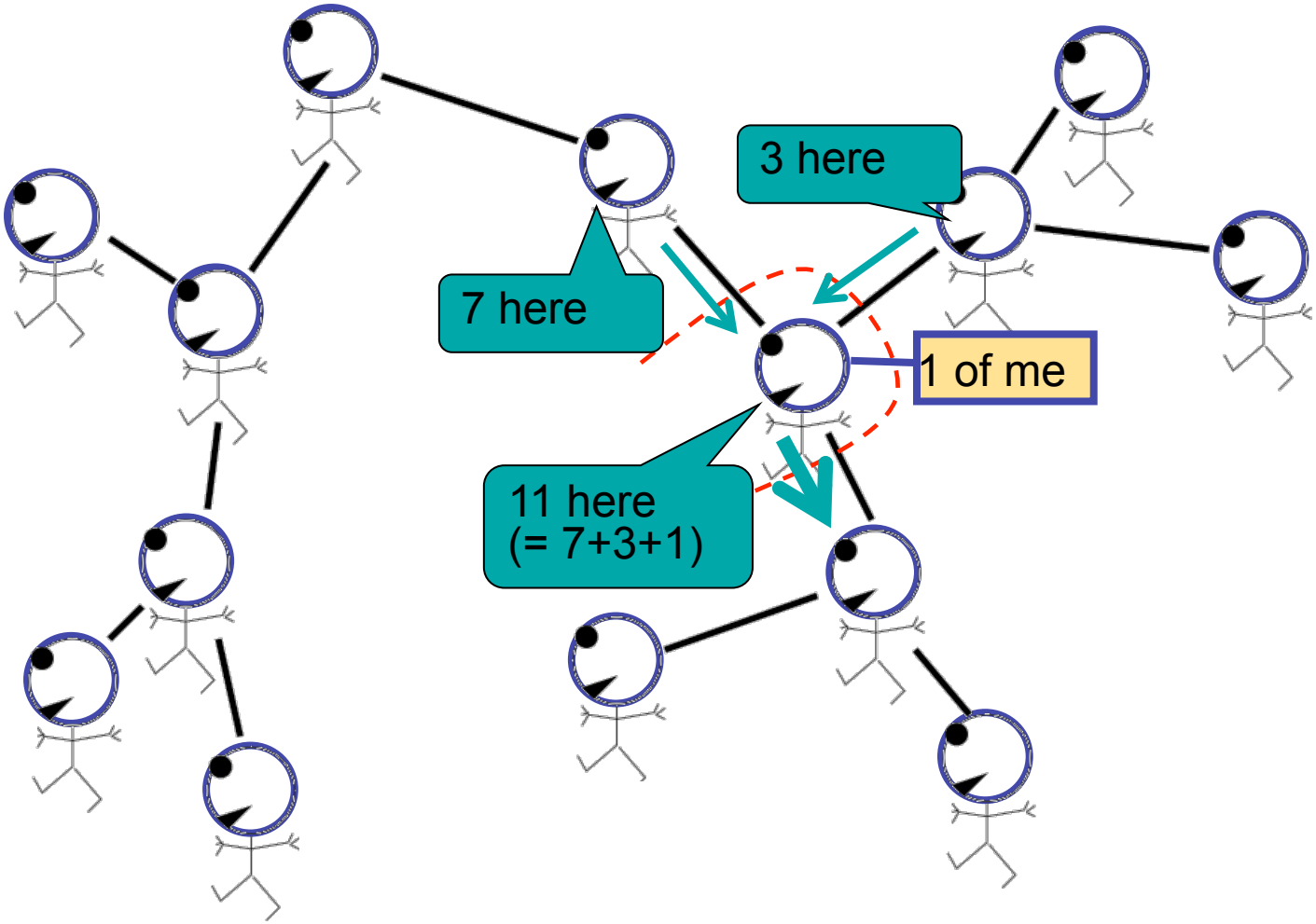
Counting soldiers example (beliefs)



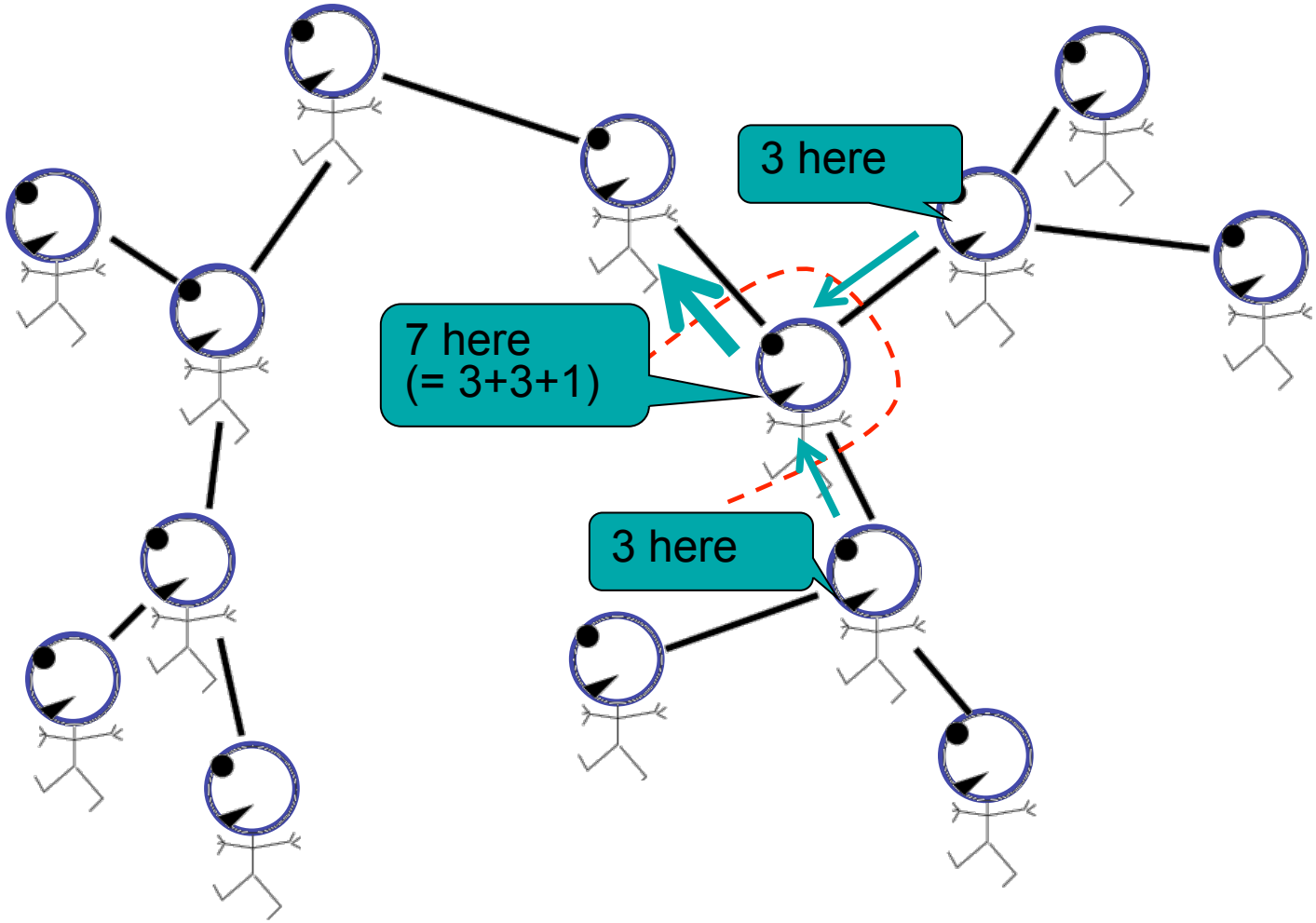
Counting soldiers example (locality)



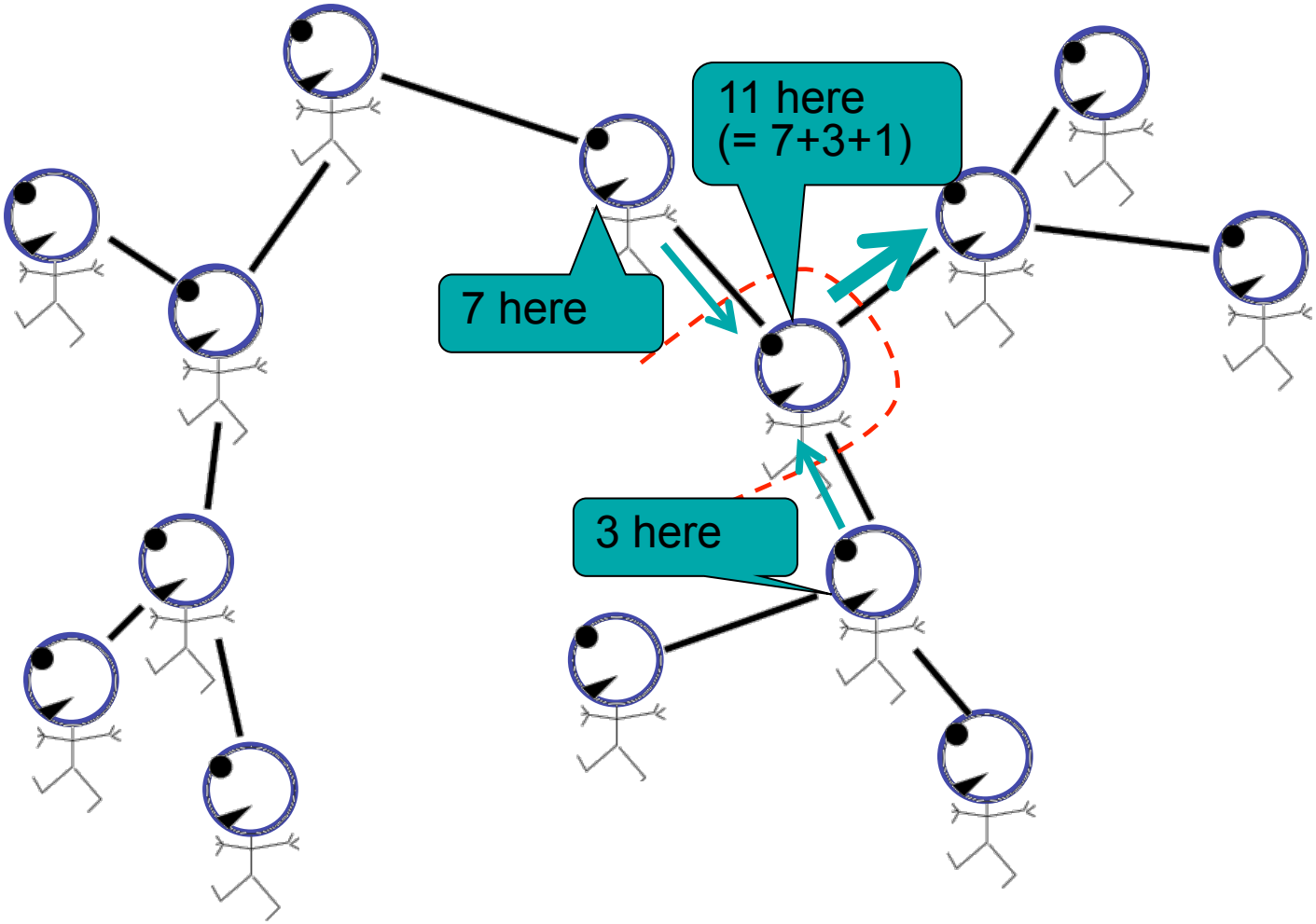
Counting soldiers example (in polytree)



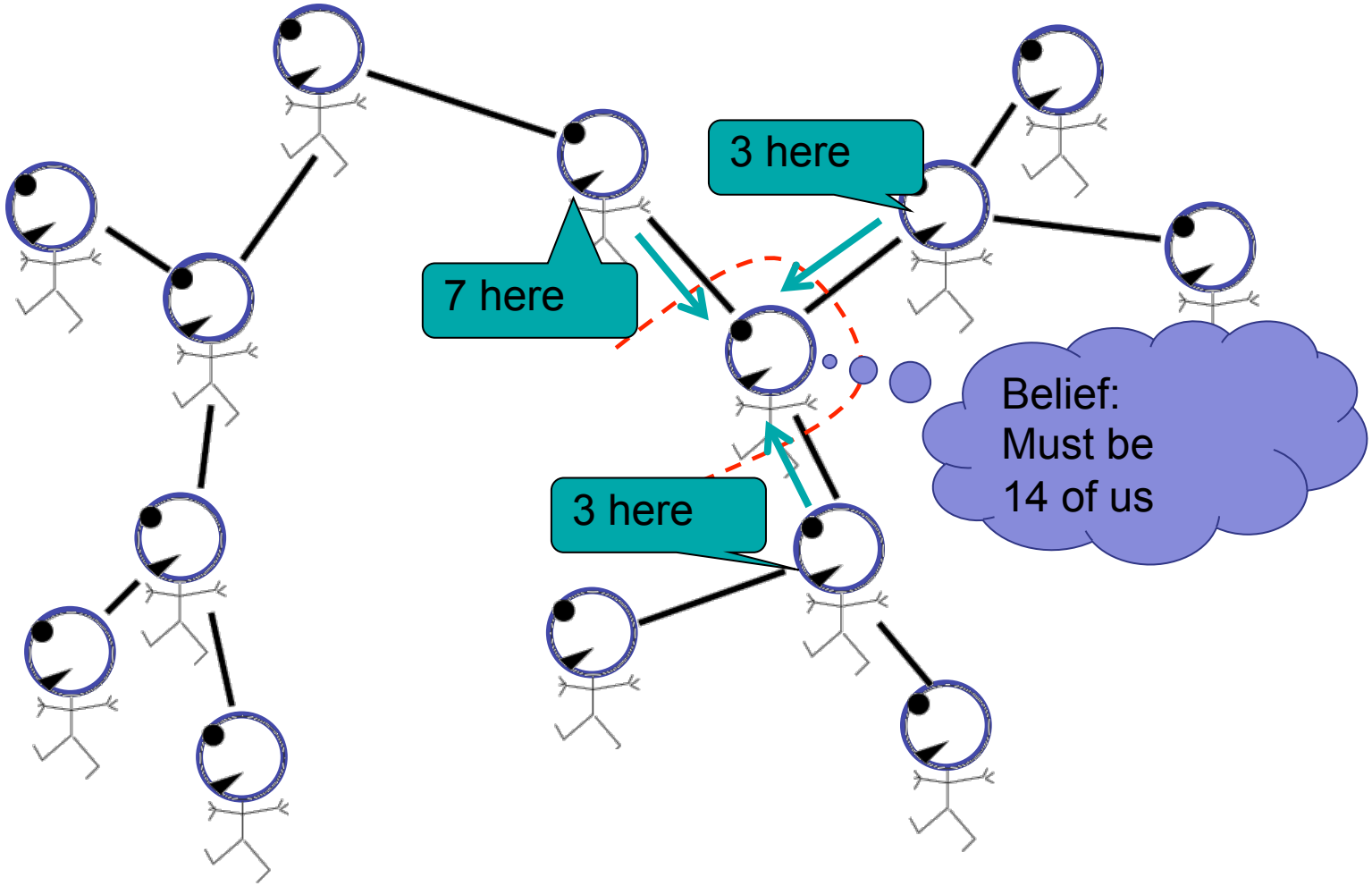
Counting soldiers example (in polytree)



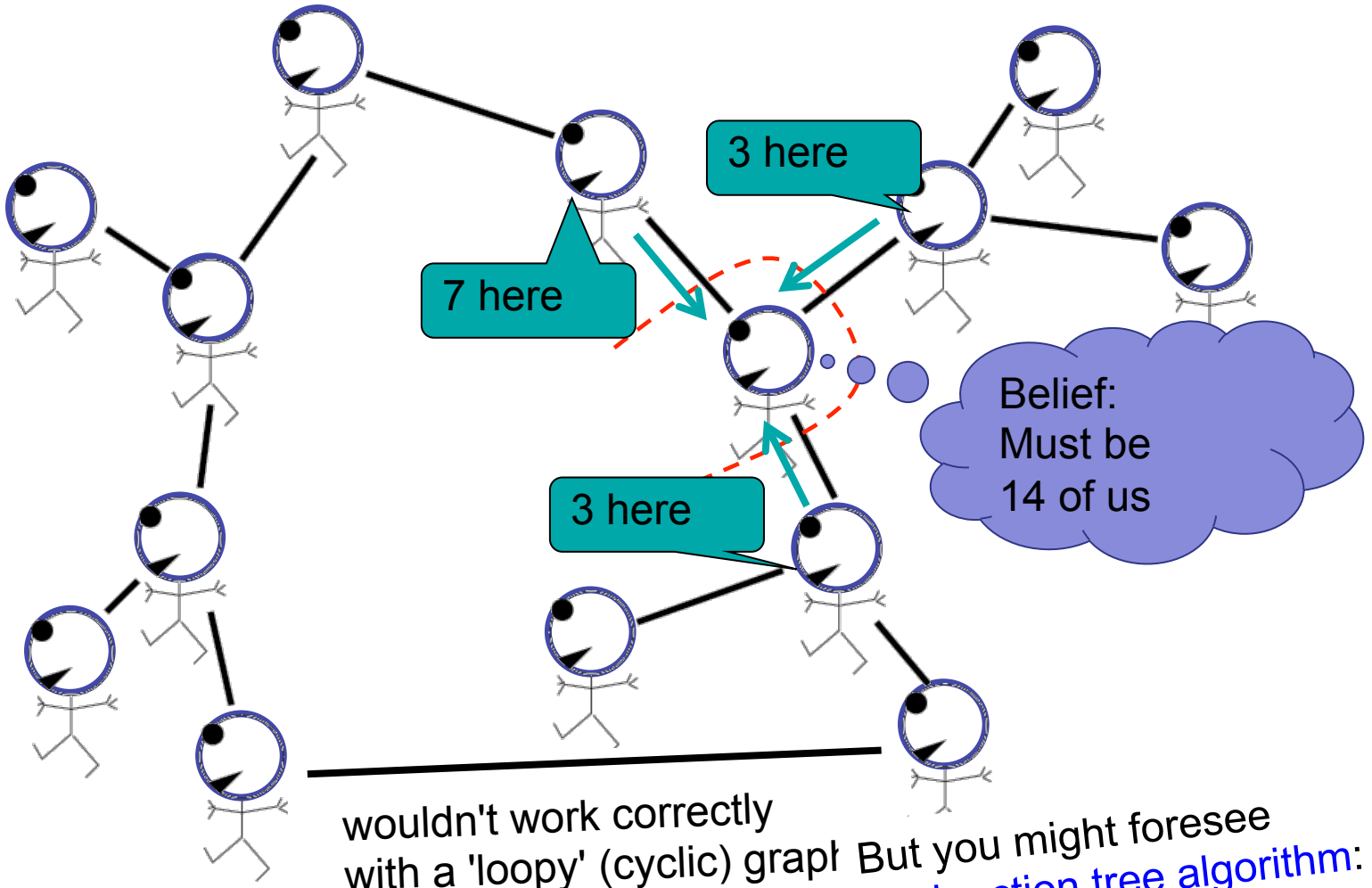
Counting soldiers example (in polytree)



Counting soldiers example (in polytree)



Counting soldiers example (non-circularity)



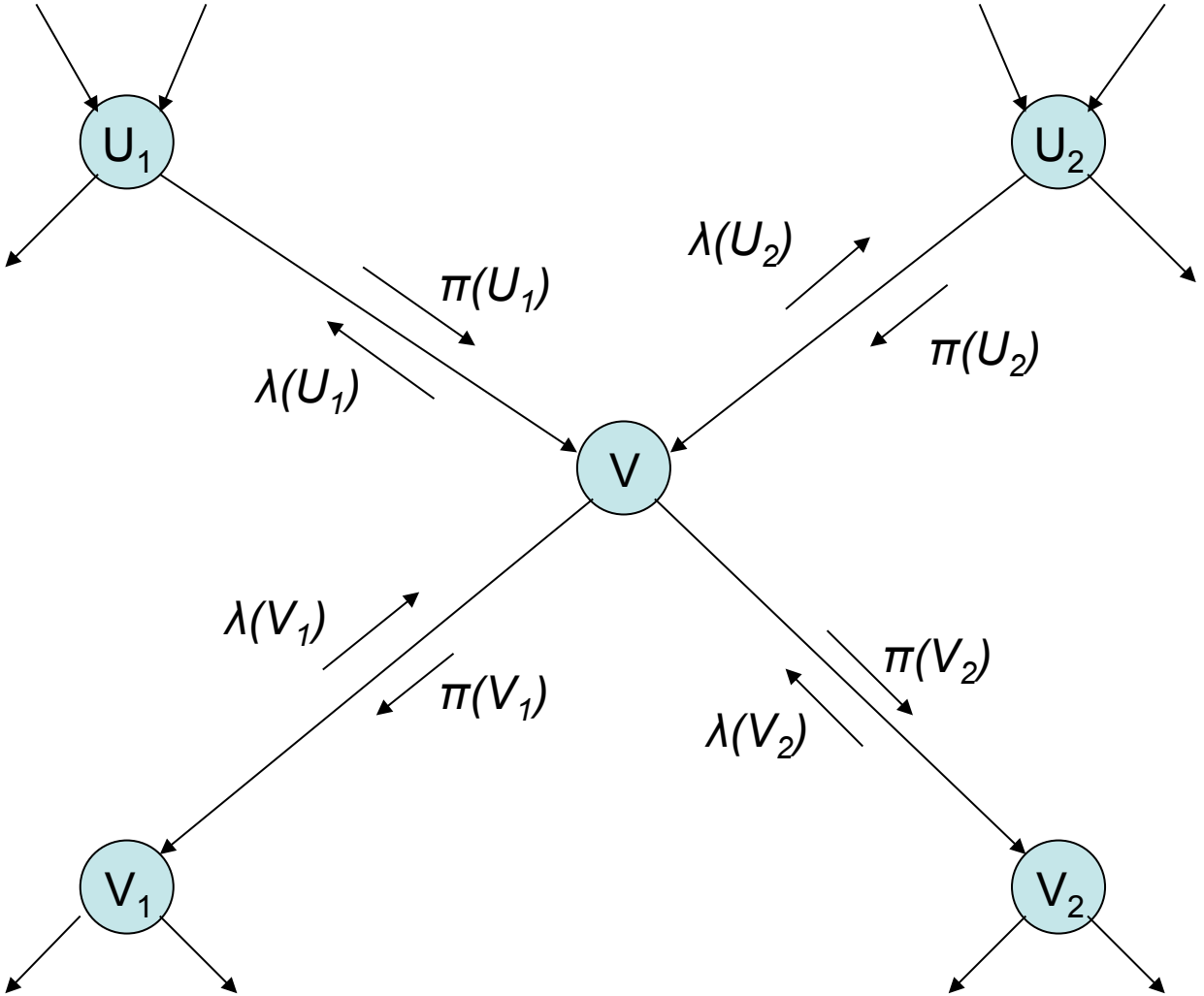
wouldn't work correctly
with a 'loopy' (cyclic) graph

But you might foresee
the junction tree algorithm:
One has to partition the graph
into a tree of clusters

Pearl‘s belief propagation algorithm

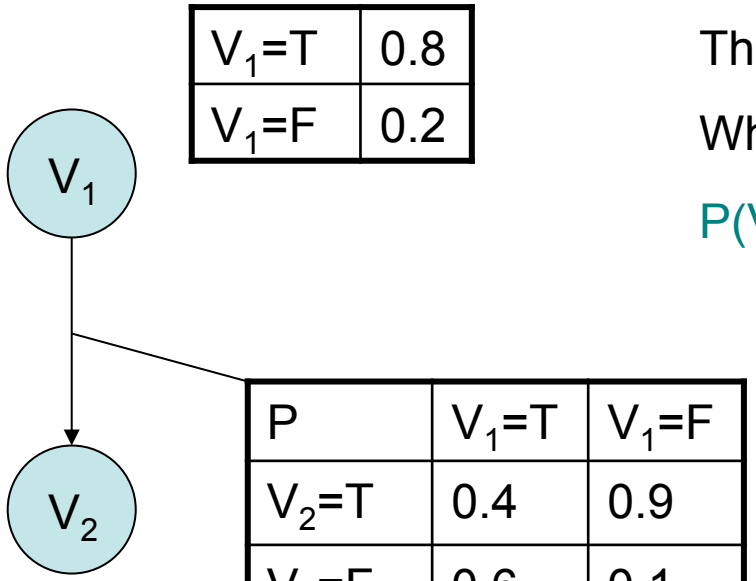
- Local computation for one node V desired
- Information flows through the links of G
  - flows as messages of two types – λ and π
- V splits network into two disjoint parts
  - Strong independence assumptions induced – crucial!
- Denote E_V^+ the part of evidence accessible through the parents of V (causal or predictive)
  - passed downward in π messages
- Analogously, let E_V^- be the diagnostic evidence
  - passed upwards in λ messages

Pearl's Belief Propagation



The π Messages

- What are the messages?
- For simplicity, let the nodes be binary



V_1=T	0.8
V_1=F	0.2

P	V_1=T	V_1=F
V_2=T	0.4	0.9
V_2=F	0.6	0.1

The message passes on information.

What information? Observe:

$P(V_2) = P(V_2|\ V_1=T)P(V_1=T)$

$+ P(V_2|\ V_1=F)P(V_1=F)$

The information needed is the CPT of $V_1 = \pi_V(V_1)$

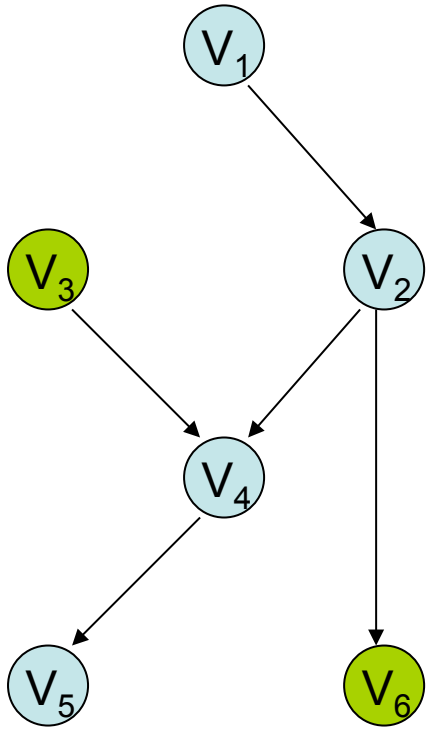
π Messages capture information passed from parent to child

The Evidence

- Evidence – values of observed nodes
  - $V_3 = T$, $V_6 = 3$
- Our belief in what the value of V_i ‘should’ be changes.
- This belief is propagated
- As if the CPTs became

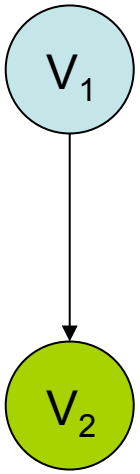
V_3=T	1.0
V_3=F	0.0

P	V_2=T	V_2=F
V_6=1	0.0	0.0
V_6=2	0.0	0.0
V_6=3	1.0	1.0



The Messages

- We know what the π messages are
- What about λ?



Assume E = { V_2 } and compute by Bayes rule:

$$P(V_1 \mid V_2) = \frac{P(V_1)P(V_2 \mid V_1)}{P(V_2)} = \alpha P(V_1)P(V_2 \mid V_1)$$

The information not available at V_1 is $P(V_2|V_1)$. To be passed upwards by a λ-message. Again, this is not in general exactly the CPT, but the belief based on evidence down the tree.

- The messages are $\pi(V)=P(V|E^+)$ and $\lambda(V)=P(E^-|V)$

Combination of evidence in node V

- Recall that $E_V = E_V^+ \cup E_V^-$ and let us compute

$$P(V \mid E) = P(V \mid E_V^+, E_V^-) = \alpha' P(E_V^+, E_V^- \mid V) P(V) =$$
$$\alpha' P(E_V^- \mid V) P(E_V^+ \mid V) P(V) = \alpha P(E_V^- \mid V) P(V \mid E_V^+) =$$
$$\alpha \lambda(V) \pi(V) = BEL(V)$$

- α is the normalization constant ($1/P(E^-|E^+)$)
- normalization is not necessary (can do it at the end)
- but may prevent numerical underflow problems

λ message combination

- Assume X received λ-messages from neighbors
- How to compute $\lambda(x) = P(e^-|x)$?
- Let $Y_1, \ldots, Y_c$ be the children of X
- $\lambda_{XY}(x)$ denotes the λ-message sent between X and Y

$$\lambda(x) = \prod_{j=1}^{c} \lambda_{Y_j X_i}(x)$$

Derivation clear: Assume two children Y, Z of X, then $P(e^-|x) = P(e^-_Y, e^-_Z|x) = P(e^-_Y|x) * P(e^-_Z|x)$ (siblings independent given parent)

π message combination

- Assume X received π -messages from neighbors
- How to compute $\pi(x) = P(x|e^+)$?
- Let $U_1, \ldots, U_p$ be the parents of X
- $\pi_{XY}(x)$ denotes the π-message sent between X and Y
- summation over the CPT

$$\pi(x) = \sum_{u_1,\ldots,u_p} P(x \mid u_1,\ldots,u_p) \prod_{j=1}^{p} \pi_{U_j X_i}(u_j)$$

Derivation: Condition on all U_i and note that each pair (parent U_i,evidence e^+_i) is independent of the other pairs (U_j,e^+_j)

Messages to pass

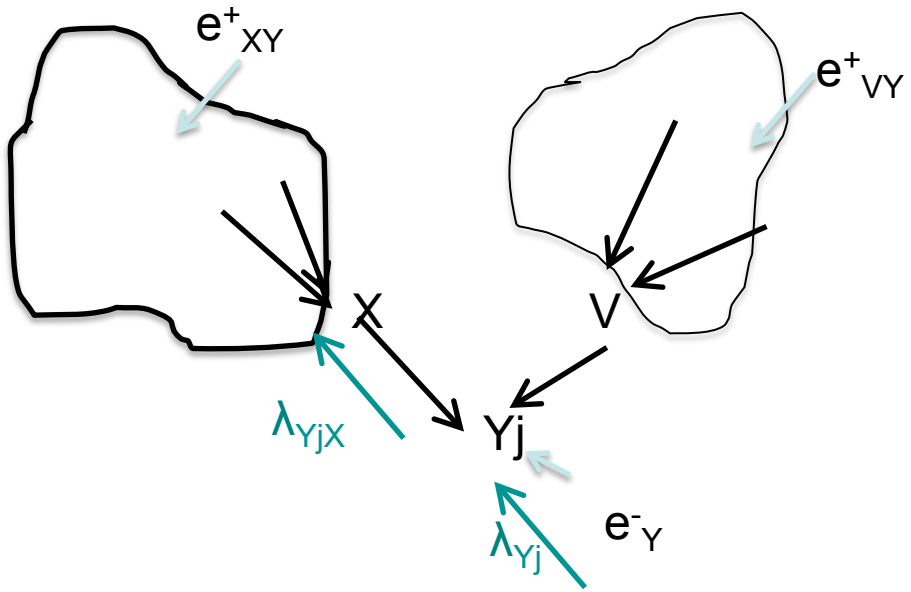
- We need to compute $\pi_{XY}(x)$

$$\pi_{XY_J}(x) = \alpha \pi_X(x) \prod_{k \neq j} \lambda_{Y_k X}(x)$$

Derivation

π_{XYJ}
$= P(x \mid e^{+}_{XYj})$
$= P(x \mid e - e^{-}_{XYj}) =$
= [Bel(x) when evidence e^{-}_{Xyj} is surpressed]
= Bel(x) setting $\lambda_{XYj}(x) = 1$
= formula above

Messages to pass



Derivation hint
- $\lambda_{YjX} = P(e^-_{XYj}|x)$
  $= P(e^+_{VYj}, e^-_{Yj}|X)$
- then condition on YJ and V,
- And use independences
  - Y_j seperates e^+_{VYj} from e^-_{Yj}
  - V seperates e^+_{VYj} from X

- Symbolically, group other (≠X) parents of Yj into single complex $V = V_1, \ldots, V_q$ and treat link X->Yj vs. V-> Yj

$$\lambda_{Y_jX}(x) = \sum_{y_j} \lambda_{Y_j}(y_j) \sum_{v_1,\ldots,v_q} p(y \mid v_1,\ldots,v_q) \prod_{k=1}^{q} \pi_{V_kY_j}(v_k)$$

Pearl's Belief Propagation Algorithm

- Initialization step
  - For all $V_i=e_i$ in E:
    - $\lambda(x_i) = 1$ whenever $x_i = e_i$; 0 otherwise
    - $\pi(x_i) = 1$ whenever $x_i = e_i$; 0 otherwise
  - For nodes without parents
    - $\pi(x_i) = p(x_i)$ - prior probabilities
  - For nodes without children
    - $\lambda(x_i) = 1$ uniformly (normalize at end)

Pearl's Belief Propagation Algorithm

- Iterate until no change occurs
  - (For each node X) if X has received all the π messages from its parents, calculate π(x)
  - (For each node X) if X has received all the λ messages from its children, calculate λ(x)
  - (For each node X) if π(x) has been calculated and X received all the λ-messages from all its children (except Y), calculate $\pi_{XY}(x)$ and send it to Y.
  - (For each node X) if λ(x) has been calculated and X received all the π-messages from all parents (except U), calculate $\lambda_{XU}(u)$ and send it to U.
- Compute BEL(X) = λ(x)π(x) and normalize

Complexity

- On a polytree, the BP algorithm converges in time proportional to diameter of network – at most linear
- Work done in a node is proportional to the size of CPT
- Hence BP is linear in number of network parameters
- For general BNs
  - Exact inference is NP-hard
  - Approximate inference is NP-hard

- Most BNs are not polytrees. What to do?
  - Clustering (e.g., junction tree) algorithm (somewhat later) to make graph tree like and then use BP
  - Approximate inference (next slides)

Approximate Inference over BNs

Approximation

- We are going to approximate answers to queries such as the posterior (PX|E).
- Here we assume an intuitive of approximation:
  - The answer may be erroneous up to some amount
- Formally treated in PAC theory (Probably Approximately Correct) by parameters (δ,ε)
  - Confidence (quantified by δ) in that found solution maximally deviates from true solution up to ε
- PAC discussed in one of the next lectures

Approximate Inference in Bayesian Networks

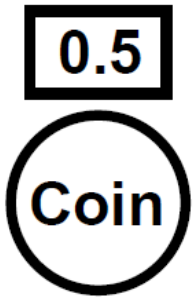
Monte Carlo algorithm

- Widely used to estimate quantities that are difficult to calculate exactly (Remember: for BNs NP-hardness)
- Randomized sampling algorithm
- Accuracy depends on the number of samples
- Two families
  - Direct sampling
  - Markov chain sampling

Inference by stochastic simulation

Basic idea:

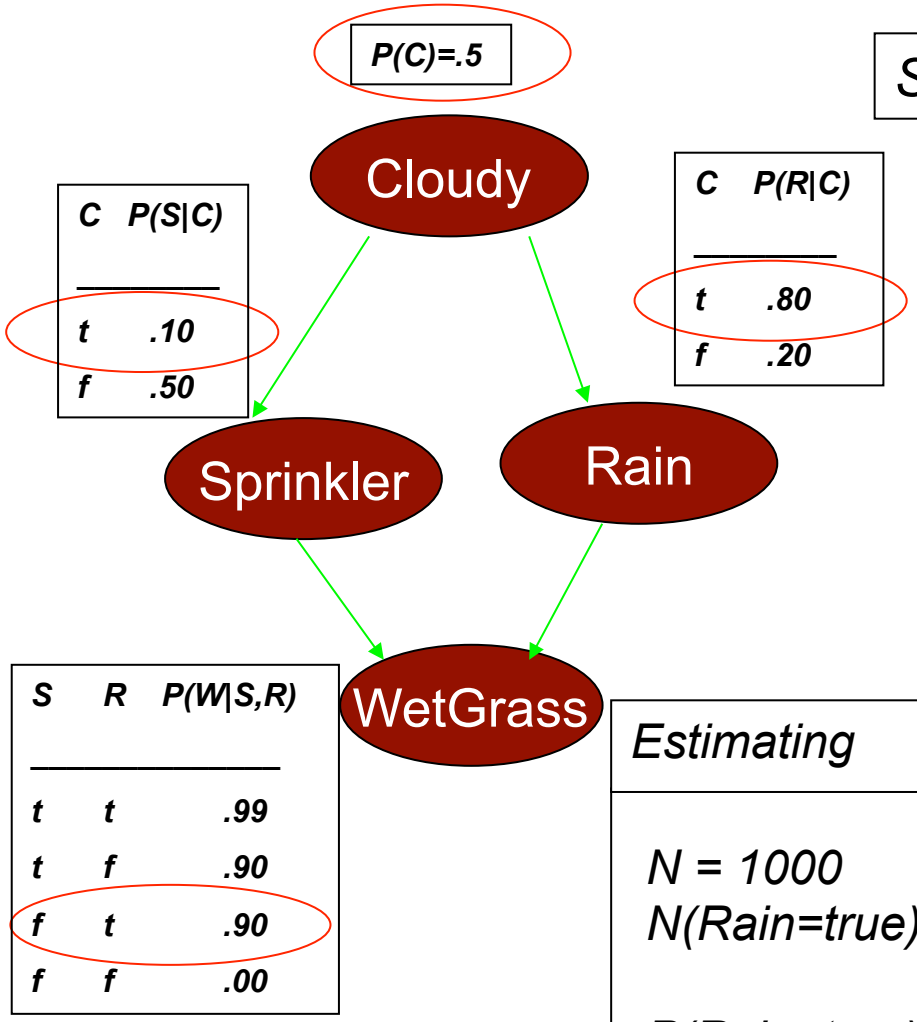
1) Draw N samples from a sampling distribution S
2) Compute an approximate posterior probability $\hat{P}$
3) Show this converges to the true probability P



Outline:

- Sampling from an empty network
- Rejection sampling: reject samples disagreeing with evidence
- Likelihood weighting: use evidence to weight samples
- Markov chain Monte Carlo (MCMC): sample from a stochastic process whose stationary distribution is the true posterior

Example in simple case



Sampling

[Cloudy, Sprinkler, Rain, WetGrass]

[true, , ,]

[true, false, ,]

[true, false, true,]

[true, false, true, true]

Estimating

N = 1000 *(# generated samples)*

N(Rain=true) = N([_ , _ , true, _]) = 511
(# of samples for this event)

P(Rain=true) = 0.511 *(approximated probability)*

Sampling from empty network

- Generating samples from a network that has no evidence associated with it (*empty* network)
- Basic idea
  - sample a value for each variable in topological order
  - using the specified conditional probabilities

function PRIOR-SAMPLE(bn) **returns** an event sampled from bn
  inputs: bn, a belief network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

  $\mathbf{x} \leftarrow$ an event with n elements
  for $i = 1$ **to** n **do**
    $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
      given the values of $Parents(X_i)$ in $\mathbf{x}$
  return $\mathbf{x}$

Properties

Probability that PRIORSAMPLE generates a particular event

$$S_{PS}(x_1 \ldots x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i)) = P(x_1 \ldots x_n)$$

i.e., the true prior probability

E.g., $S_{PS}(t, f, t, t) = 0.5 \times 0.9 \times 0.8 \times 0.9 = 0.324 = P(t, f, t, t)$

Let $N_{PS}(x_1 \ldots x_n)$ be the number of samples generated for event $x_1, \ldots, x_n$

Then we have

$$\begin{aligned} \lim_{N \to \infty} \hat{P}(x_1, \ldots, x_n) &= \lim_{N \to \infty} N_{PS}(x_1, \ldots, x_n)/N \\ &= S_{PS}(x_1, \ldots, x_n) \\ &= P(x_1 \ldots x_n) \end{aligned}$$

That is, estimates derived from PRIORSAMPLE are consistent

Shorthand: $\hat{P}(x_1, \ldots, x_n) \approx P(x_1 \ldots x_n)$

What if evidence is given?

- Sampling as defined above would generate cases that cannot be used

Rejection Sampling

- Used to compute conditional probabilities
- Procedure
  - Generating sample from prior distribution specified by the Bayesian Network
  - Rejecting all that do not match the evidence
  - Estimating probability

Rejection Sampling

$\hat{\mathbf{P}}(X|\mathbf{e})$ estimated from samples agreeing with $\mathbf{e}$

function REJECTION-SAMPLING($X, \mathbf{e}, bn, N$) **returns** an estimate of $P(X|\mathbf{e})$
  local variables: $\mathbf{N}$, a vector of counts over X, initially zero

  for $j = 1$ to N **do**
    $\mathbf{x} \leftarrow$ PRIOR-SAMPLE(bn)
    if $\mathbf{x}$ is consistent with $\mathbf{e}$ **then**
      $\mathbf{N}[x] \leftarrow \mathbf{N}[x]+1$ where x is the value of X in $\mathbf{x}$
  return NORMALIZE($\mathbf{N}[X]$)

Rejection Sampling Example

- Let us assume we want to estimate P(Rain|Sprinkler = true) with 100 samples
- 100 samples
  - 73 samples => Sprinkler = false
  - 27 samples => Sprinkler = true
    - 8 samples => Rain = true
    - 19 samples => Rain = false
- P(Rain|Sprinkler = true) = NORMALIZE((8,19)) = (0.296,0.704)
- Problem
  - It rejects too many samples

Analysis of rejection sampling

$$
\begin{aligned}
\hat{\mathbf{P}}(X|\mathbf{e}) &= \alpha \mathbf{N}_{PS}(X,\mathbf{e}) && \text{(algorithm defn.)} \\
&= \mathbf{N}_{PS}(X,\mathbf{e})/N_{PS}(\mathbf{e}) && \text{(normalized by } N_{PS}(\mathbf{e})\text{)} \\
&\approx \mathbf{P}(X,\mathbf{e})/P(\mathbf{e}) && \text{(property of PriorSample)} \\
&= \mathbf{P}(X|\mathbf{e}) && \text{(defn. of conditional probability)}
\end{aligned}
$$

Hence rejection sampling returns consistent posterior estimates

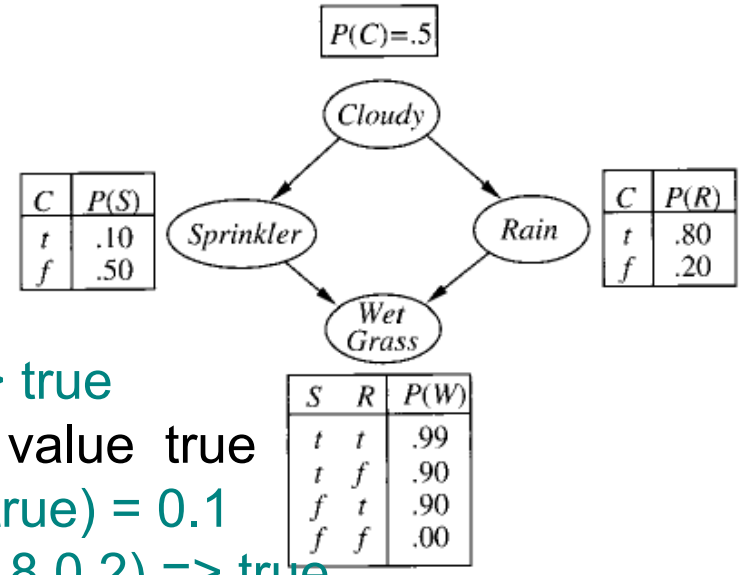
Problem: hopelessly expensive if $P(\mathbf{e})$ is small

$P(\mathbf{e})$ drops off exponentially with number of evidence variables!

Likelihood Weighting

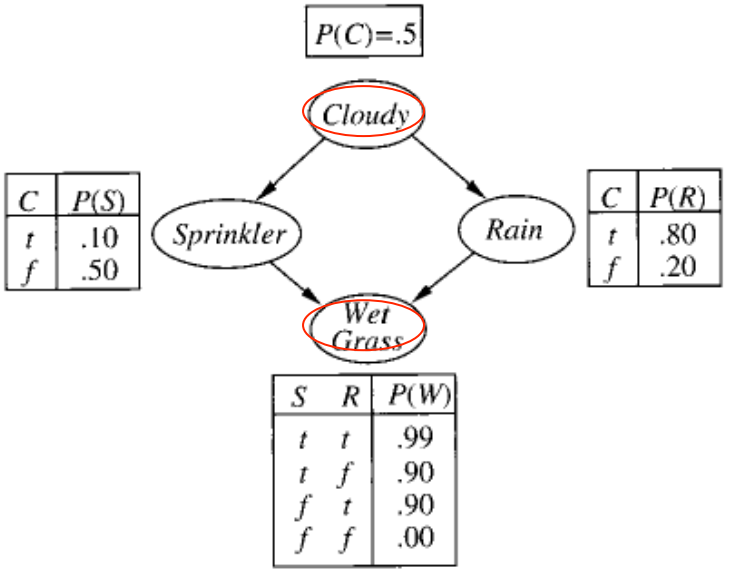
- Goal
  - Avoiding inefficiency of rejection sampling
- Idea
  - Generating only events consistent with evidence
  - Each event is weighted by likelihood that the event accords to the evidence

Likelyhood Weighting: Example



- **P**(Rain|Sprinkler=true, WetGrass = true)?
- Sampling
  - The weight is set to 1.0
  - Sample from **P**(Cloudy) = (0.5,0.5) => true
  - Sprinkler is an evidence variable with value true
    w ← w * P(Sprinkler=true | Cloudy = true) = 0.1
  - Sample from P(Rain|Cloudy=true)=(0.8,0.2) => true
  - WetGrass is an evidence variable with value true
    w ←w * P(WetGrass=true |Sprinkler=true, Rain = true) = 0.099
  - [true, true, true, true] with weight 0.099
- Estimating
  - Accumulating weights to either Rain=true or Rain=false
  - Normalize (= divide by sum of weights)

Likelyhood Weighting: Example



- **P**(Rain|Cloudy=true, WetGrass = true)?
- Sampling
  - Cloudy is an evidence
    w ← w * P(Cloudy = true) = 0.5
  - Sprinkler no evidence
    Sample from P(Sprinkler| Cloudy=true)=(0.1, 0.9) => false
  - Sample from P(Rain|Cloudy=true)=(0.8,0.2) => true
  - WetGrass is an evidence variable with value true
    w ←w * P(WetGrass=true |Sprinkler=false, Rain = true) = 0.45
  - [true, false, true, true] with weight 0.45

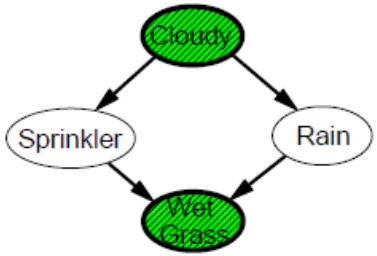
Likelihood analysis

Sampling probability for WeightedSample is

$$S_{WS}(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{l} P(z_i | parents(Z_i))$$

Note: pays attention to evidence in **ancestors** only

$\Rightarrow$ somewhere "in between" prior and posterior distribution



Weight for a given sample $\mathbf{z}, \mathbf{e}$ is

$$w(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{m} P(e_i | parents(E_i))$$

Weighted sampling probability is

$$\begin{aligned} & S_{WS}(\mathbf{z}, \mathbf{e}) w(\mathbf{z}, \mathbf{e}) \\ & = \prod_{i=1}^{l} P(z_i | parents(Z_i)) \prod_{i=1}^{m} P(e_i | parents(E_i)) \\ & = P(\mathbf{z}, \mathbf{e}) \text{ (by standard global semantics of network)} \end{aligned}$$

Hence likelihood weighting returns consistent estimates but performance still degrades with many evidence variables because a few samples have nearly all the total weight

Likelihood weighting

function LIKELIHOOD-WEIGHTING($X, \mathbf{e}, bn, N$) **returns** an estimate of $P(X|\mathbf{e})$

  local variables: $\mathbf{W}$, a vector of weighted counts over X, initially zero

  for $j = 1$ to N **do**

    $\mathbf{x}, w \leftarrow$ WEIGHTED-SAMPLE(bn)

    $\mathbf{W}[x] \leftarrow \mathbf{W}[x] + w$ where x is the value of X in $\mathbf{x}$

  return NORMALIZE($\mathbf{W}[X]$)

---

function WEIGHTED-SAMPLE($bn, \mathbf{e}$) **returns** an event and a weight

  $\mathbf{x} \leftarrow$ an event with n elements; $w \leftarrow 1$

  for $i = 1$ **to** n **do**

    if X_i has a value x_i in $\mathbf{e}$

      then $w \leftarrow w \times P(X_i = x_i \mid parents(X_i))$

      else $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$

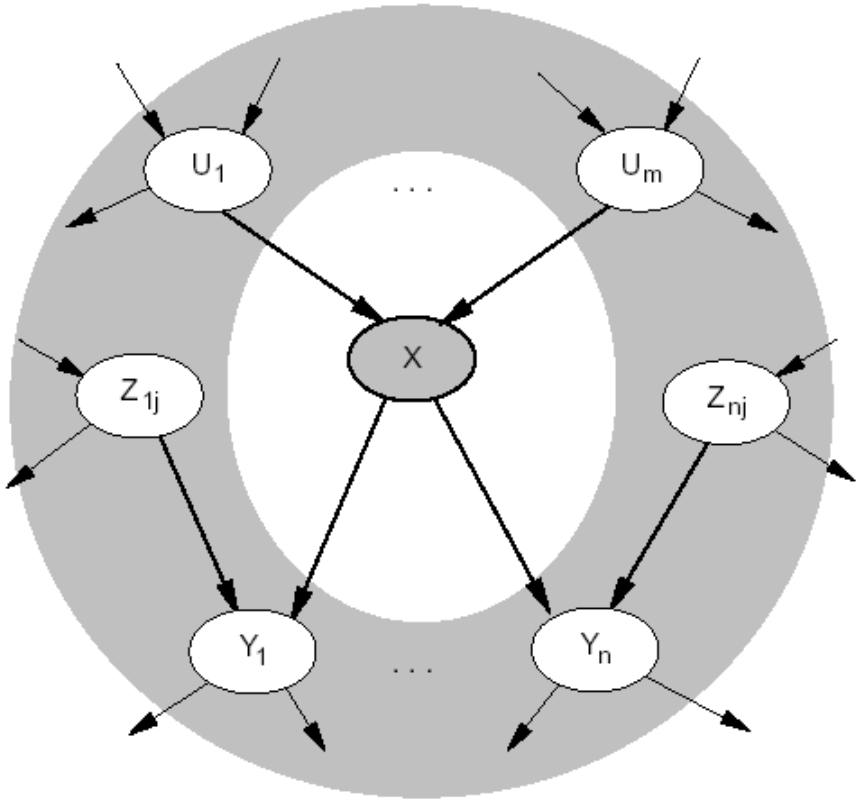
  return $\mathbf{x}$, w

Markov Chain Monte Carlo

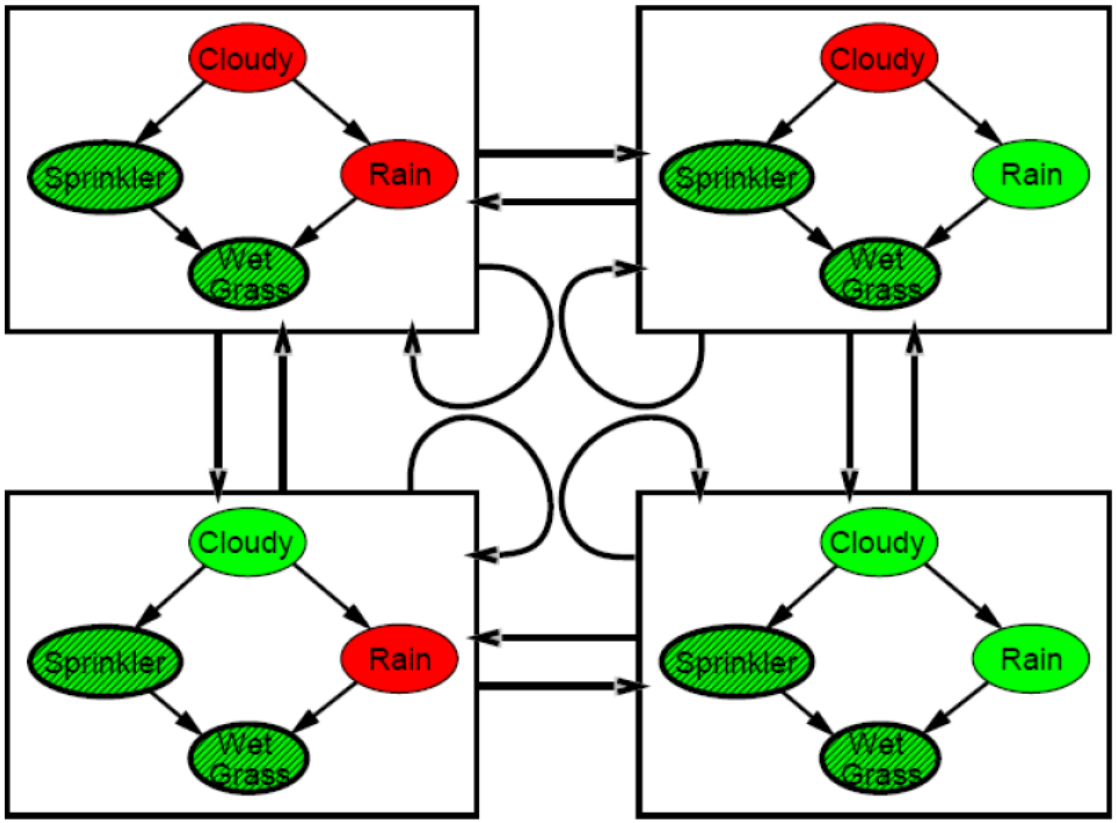
- Let's think of the network as being in a particular current state specifying a value for every variable
- MCMC generates each event by making a random change to the preceding event
- The next state is generated by randomly sampling a value for one of the non-evidence variables X_i, conditioned on the current values of the variables in the Markov blanket of X_i
- Likelihood Weighting only takes into account the evidences of the parents. (Problematic if evidence on leafs).

Markov Blanket

- Markov blanket: Parents + children + children's parents
- Node is conditionally independent of all other nodes in network, given its Markov Blanket



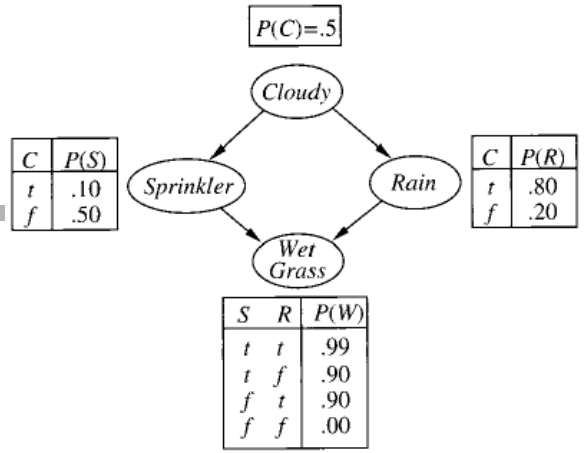
With $Sprinkler = true, WetGrass = true$, there are four states:



Wander about for a while, average what you see

Arrows describe transition probabilities; leads to a (the Markov) chain of states

Markov Chain Monte Carlo: Example



- P(Rain|Sprinkler = true, WetGrass = true) = ?
- Initial state is [true, true, false, true] (for [Cloudy,Sprinkler,Rain,WetGrass])

- The following steps are executed repeatedly:
  - Cloudy is sampled, given the current values of its Markov blanket variables
    So, we sample from P(Cloudy|Sprinkler= true, Rain=false)
    Suppose the result is Cloudy = false.
  - Now current state is [false, true, false, true] and counts are updated
  - Rain is sampled, given the current values of its Markov blanket variables
    So, we sample from P(Rain|Cloudy=false,Sprinkler=true, WetGrass=true)
    Suppose the result is Rain = true.
  - Then current state is [false, true, true, true]
- After all the iterations, let's say the process visited 20 states where Rain is true and 60 states where Rain is false then the answer of the query is NORMALIZE((20,60))=(0.25,0.75)

MCMC

"State" of network = current assignment to all variables.

Generate next state by sampling one variable given Markov blanket
Sample each variable in turn, keeping evidence fixed

function MCMC-ASK($X, \mathbf{e}, bn, N$) **returns** an estimate of $P(X|\mathbf{e})$
  local variables: $\mathbf{N}[X]$, a vector of counts over X, initially zero
    $\mathbf{Z}$, the nonevidence variables in bn
    $\mathbf{s}$, the current state of the network, initially copied from $\mathbf{e}$
  initialize $\mathbf{s}$ with random values for the variables in $\mathbf{Z}$
  for $j = 1$ to N **do**
    for each Z_i in $\mathbf{Z}$ **do**
      sample the value of Z_i in $\mathbf{s}$ from $\mathbf{P}(Z_i|mb(Z_i))$
        given the values of $MB(Z_i)$ in $\mathbf{s}$
      $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where x is the value of X in $\mathbf{s}$
  return NORMALIZE($\mathbf{N}[X]$)

Can also choose a variable to sample at random each time

Summary

- Bayesian networks provide a natural representation for (causally induced) conditional independence
- Topology + CPTs = compact representation of joint distribution
- Generally easy for domain experts to construct
- Exact inference by variable elimination
  - polytime on polytrees, NP-hard on general graphs
  - space can be exponential as well
- Exact Inference by belief propagation on polytress
- Approximate inference based on sampling and counting help to overcome complexity of exact inference