國立成功大學
National Cheng Kung University
1931

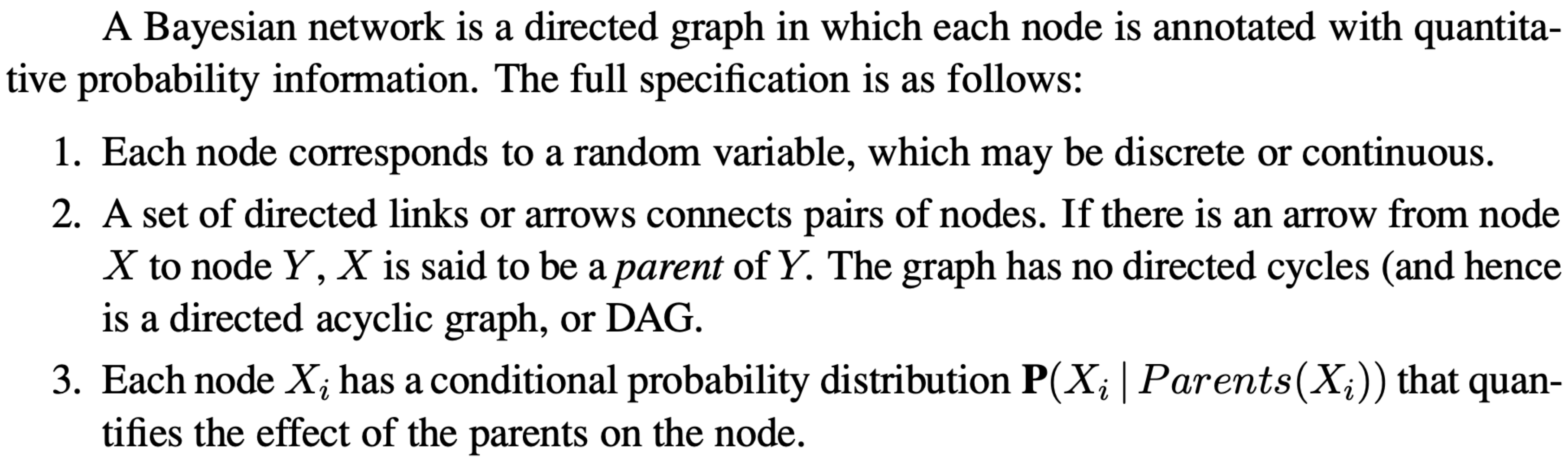
Introduction to
Artificial Intelligence

Chapter 13
Probabilistic Reasoning

Wei-Ta Chu (朱威達)

Representing Knowledge

- This chapter introduces **Bayesian network**.



A Bayesian network is a directed graph in which each node is annotated with quantitative probability information. The full specification is as follows:

1. Each node corresponds to a random variable, which may be discrete or continuous.
2. A set of directed links or arrows connects pairs of nodes. If there is an arrow from node X to node Y, X is said to be a *parent* of Y. The graph has no directed cycles (and hence is a directed acyclic graph, or DAG.
3. Each node X_i has a conditional probability distribution $\mathbf{P}(X_i \mid Parents(X_i))$ that quantifies the effect of the parents on the node.

Representing Knowledge

- Recall the variables *Toothache*, *Cavity*, *Catch*, and *Weather*. We argued that *Weather* is independent of the other variables; we argued that *Toothache* and *Catch* are conditionally independent, given *Cavity*.
- The conditional independence of *Toothache* and *Catch*, given *Cavity*, is indicated by the **absence** of a link between *Toothache* and *Catch*.

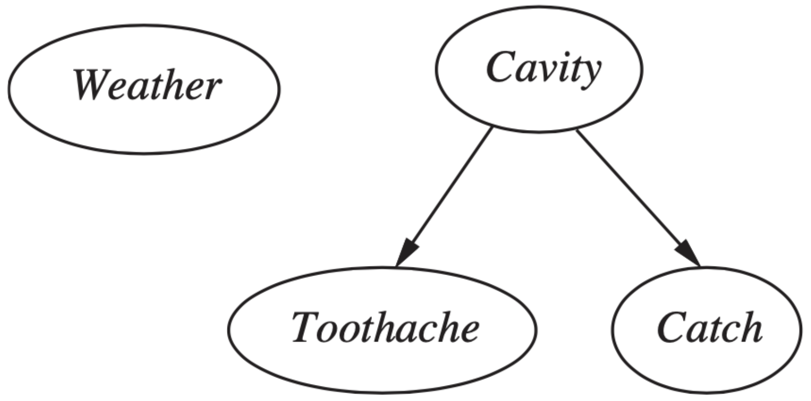


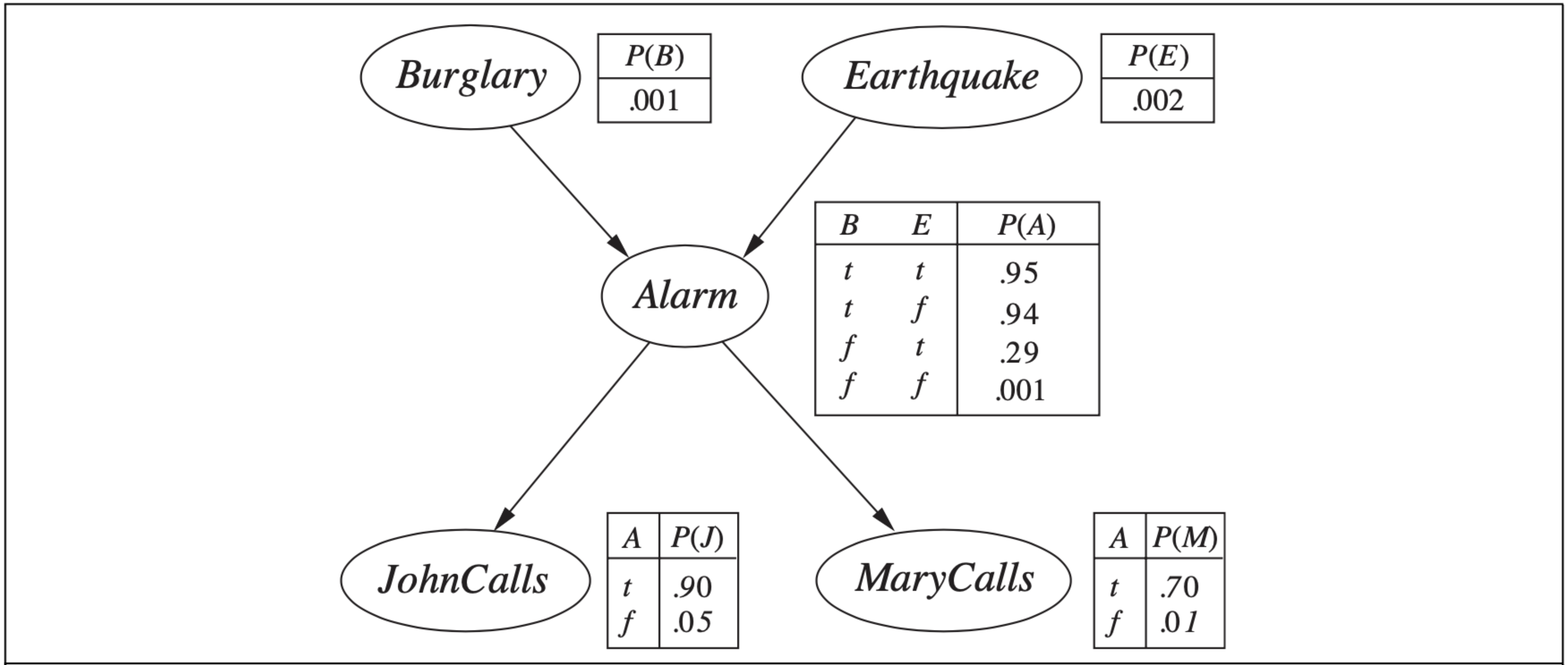
Figure 14.1 A simple Bayesian network in which *Weather* is independent of the other three variables and *Toothache* and *Catch* are conditionally independent, given *Cavity*.

Representing Knowledge

- You have a new burglar alarm (竊盜警報器) installed at home. It is fairly reliable at detecting a burglary, but also responds on occasion to minor earthquakes.

- You also have two neighbors, John and Mary, who have promised to call you at work when they hear the alarm. John nearly always calls when he hears the alarm, but sometimes confuses the telephone ringing with the alarm and calls then, too. Mary, on the other hand, likes rather loud music and often misses the alarm altogether.

Representing Knowledge



P(B)
.001

P(E)
.002

B	E	P(A)
t	t	.95
t	f	.94
f	t	.29
f	f	.001

A	P(J)
t	.90
f	.05

A	P(M)
t	.70
f	.01

Figure 14.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters *B*, *E*, *A*, *J*, and *M* stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.

國立成功大學 National Cheng Kung University

Representing Knowledge

- The network structure shows that burglary and earthquakes directly affect the probability of the alarm's going off, but whether John and Mary call depends only on the alarm.

- The network thus represents our assumptions that they do not perceive burglaries directly, they do not notice minor earthquakes, and they do not confer (協商) before calling.

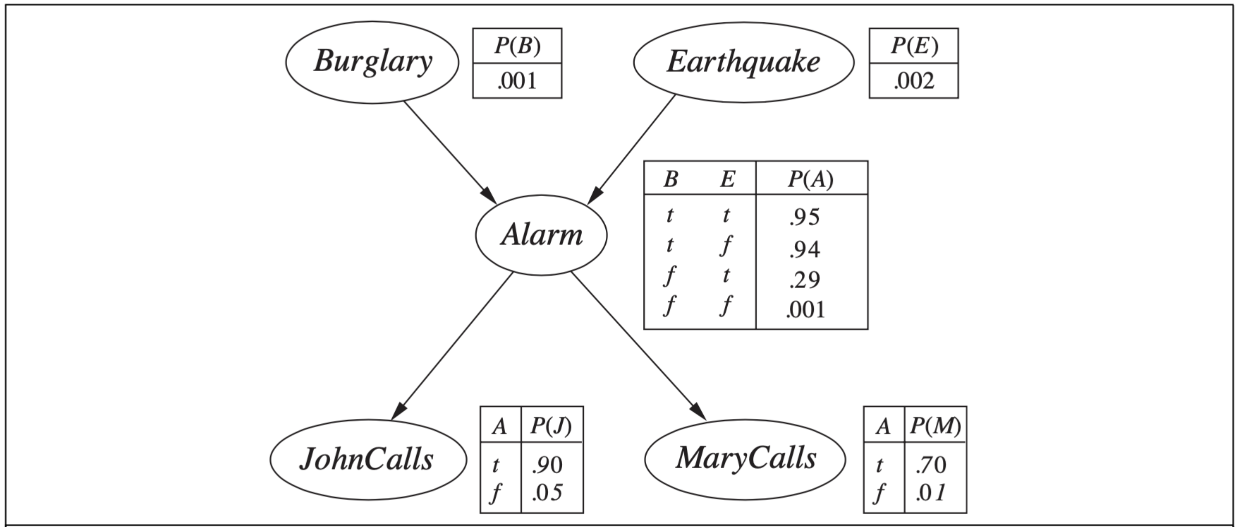
Representing Knowledge

- The conditional distributions are shown as a conditional probability table, or CPT.
- A node with no parents has only one row, representing the **prior probabilities** of each possible value of the variable.
- The probabilities actually summarize a potentially infinite set of circumstances in which the alarm might fail to go off (high humidity, power failure, dead battery, cut wires, a dead mouse stuck inside the bell, etc.) or John or Mary might fail to call and report it (out to lunch, on vacation, temporarily deaf, passing helicopter, etc.).
- In this way, a small agent can cope with a very large world, approximately.

Representing the full joint distribution

- We can calculate the probability that the alarm has sounded, but neither a burglary nor an earthquake has occurred, and both John and Mary call.

$$\begin{aligned} P(j, m, a, \neg b, \neg e) &= P(j \mid a)P(m \mid a)P(a \mid \neg b \wedge \neg e)P(\neg b)P(\neg e) \\ &= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998 = 0.000628 \end{aligned}$$



P(B)
.001

P(E)
.002

B	E	P(A)
t	t	.95
t	f	.94
f	t	.29
f	f	.001

A	P(J)
t	.90
f	.05

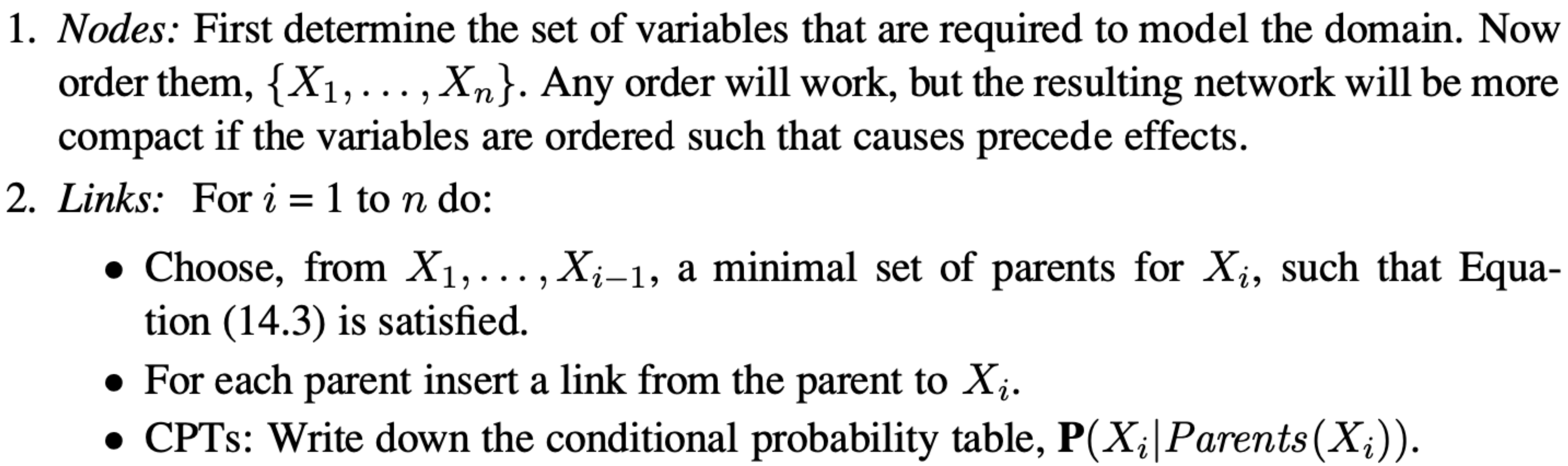
A	P(M)
t	.70
f	.01

Figure 14.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.

National Cheng Kung University

Representing the full joint distribution

- **A method for constructing Bayesian networks**



1. *Nodes:* First determine the set of variables that are required to model the domain. Now order them, $\{X_1, \ldots, X_n\}$. Any order will work, but the resulting network will be more compact if the variables are ordered such that causes precede effects.
2. *Links:* For $i = 1$ to n do:
   - Choose, from $X_1, \ldots, X_{i-1}$, a minimal set of parents for X_i, such that Equation (14.3) is satisfied.
   - For each parent insert a link from the parent to X_i.
   - CPTs: Write down the conditional probability table, $\mathbf{P}(X_i | Parents(X_i))$.

Representing the full joint distribution

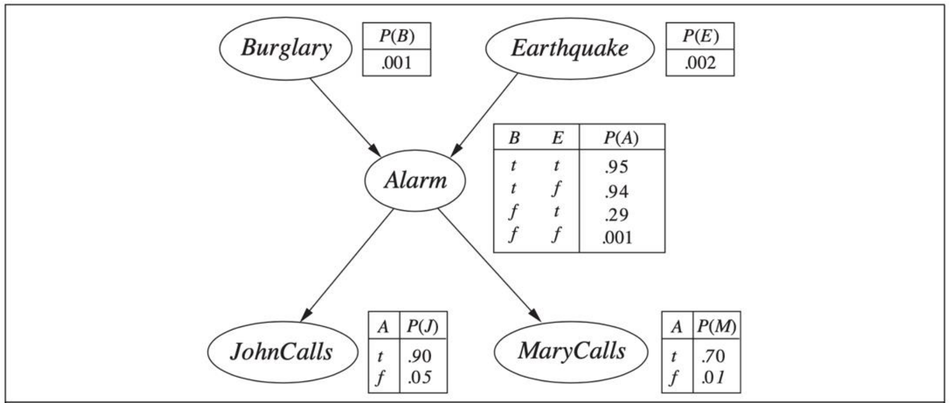
- Intuitively, the parents of node X_i should contain all those nodes in $X_1, \ldots,$ X_{i-1} that *directly influence* X_i.

$$\mathbf{P}(MaryCalls \mid JohnCalls, Alarm, Earthquake, Burglary) = \mathbf{P}(MaryCalls \mid Alarm)$$

- Because each node is connected only to earlier nodes, this construction method guarantees that the network is acyclic.

Representing the full joint distribution

- **Compactness and node ordering**
- One might object to our burglary network on the grounds that if there is an earthquake, then John and Mary would not call even if they heard the alarm, because they assume that the earthquake is the cause.
- Whether to add the link from *Earthquake* to *JohnCalls* and *MaryCalls* (and thus enlarge the tables) depends on comparing the importance of getting more accurate probabilities with the cost of specifying the extra information.



P(B)
.001

P(E)
.002

B	E	P(A)
t	t	.95
t	f	.94
f	t	.29
f	f	.001

A	P(J)
t	.90
f	.05

A	P(M)
t	.70
f	.01

Figure 14.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.

國立成功大學 National Cheng Kung University 1931

Representing the full joint distribution

- Even in a locally structured domain, we will get a compact Bayesian network only if we choose the node ordering well. What happens if we happen to choose the wrong order?
- Suppose we decide to add the nodes in the order *MaryCalls*, *JohnCalls*, *Alarm*, *Burglary*, *Earthquake*. We then get the somewhat more complicated network shown in Figure 14.3(a).

Representing the full joint distribution

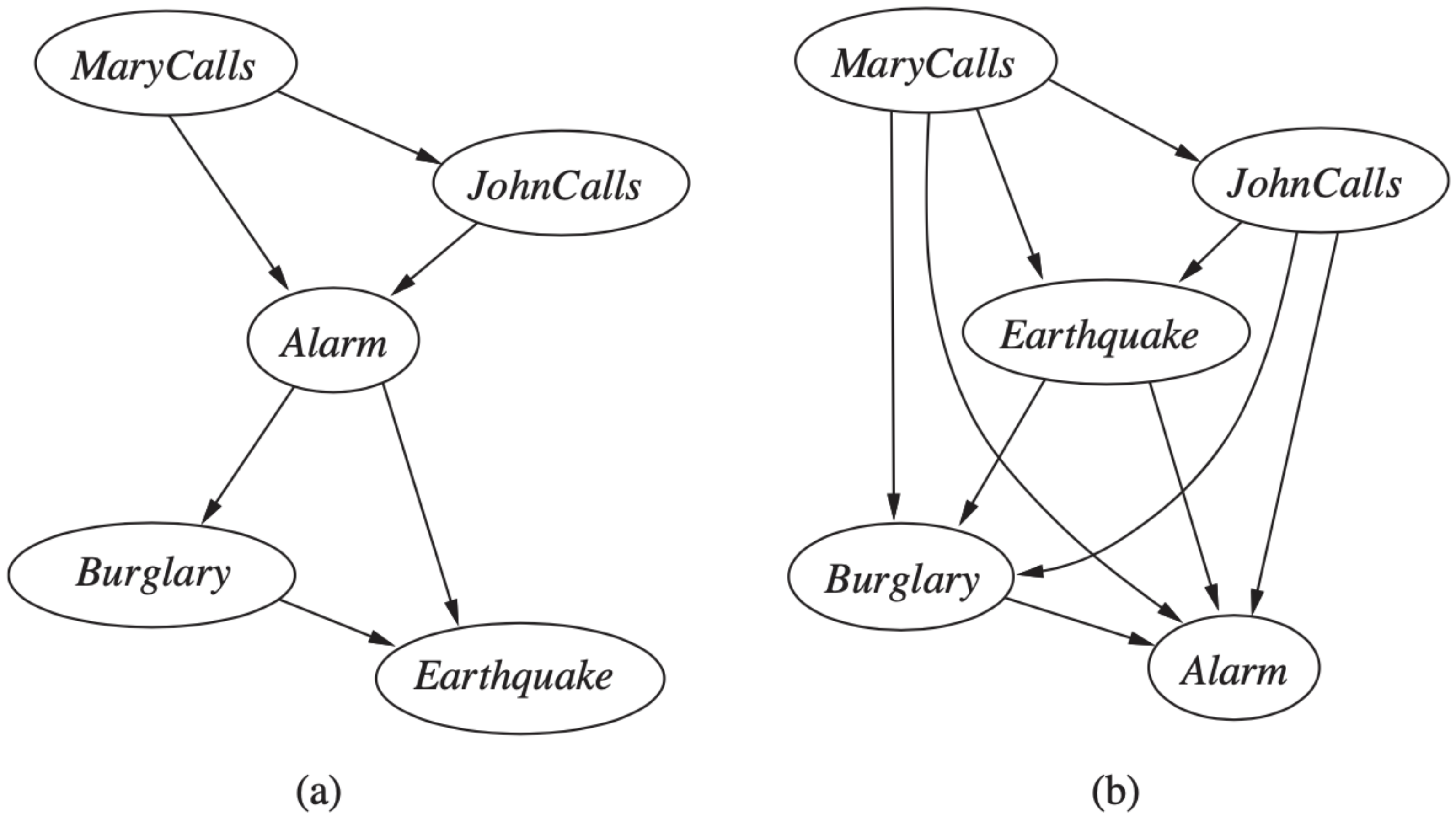


Figure 14.3 Network structure depends on order of introduction. In each network, we have introduced nodes in top-to-bottom order.

Representing the full joint distribution

- The process goes as follows:

- Adding *MaryCalls*: No parents.
- Adding *JohnCalls*: If Mary calls, that probably means the alarm has gone off, which of course would make it more likely that John calls. Therefore, *JohnCalls* needs *MaryCalls* as a parent.
- Adding *Alarm*: Clearly, if both call, it is more likely that the alarm has gone off than if just one or neither calls, so we need both *MaryCalls* and *JohnCalls* as parents.
- Adding *Burglary*: If we know the alarm state, then the call from John or Mary might give us information about our phone ringing or Mary's music, but not about burglary:

  $$\mathbf{P}(Burglary \mid Alarm, JohnCalls, MaryCalls) = \mathbf{P}(Burglary \mid Alarm) .$$

  Hence we need just *Alarm* as parent.
- Adding *Earthquake*: If the alarm is on, it is more likely that there has been an earthquake. (The alarm is an earthquake detector of sorts.) But if we know that there has been a burglary, then that explains the alarm, and the probability of an earthquake would be only slightly above normal. Hence, we need both *Alarm* and *Burglary* as parents.

Representing the full joint distribution

- The resulting network has two more links than the original network in Figure 14.2 and requires three more probabilities to be specified. What's worse, some of the links represent tenuous (稀薄的) relationships that require difficult and unnatural probability judgments, such as assessing the probability of *Earthquake*, given *Burglary* and *Alarm*.

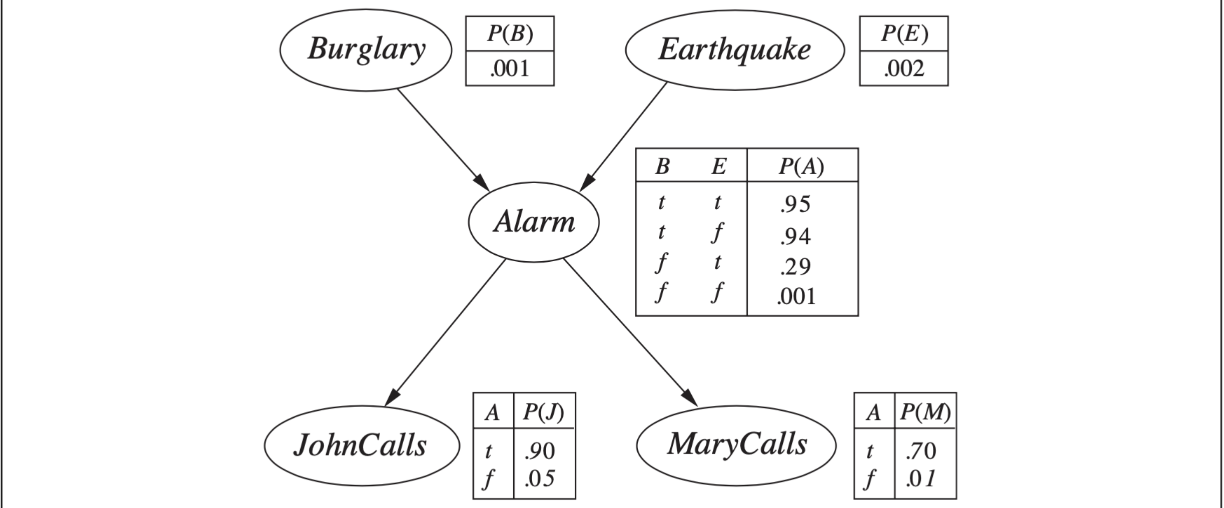


Figure 14.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.

Conditional independence relations in Bayesian networks

- The topological semantics specifies that each variable is conditionally independent of its non-**descendants**, given its parents (Figure 14.4(a)).
- A node is conditionally independent of all other nodes in the network, given its parents, children, and children's parents—that is, given its **Markov blanket**.

Conditional independence relations in Bayesian networks

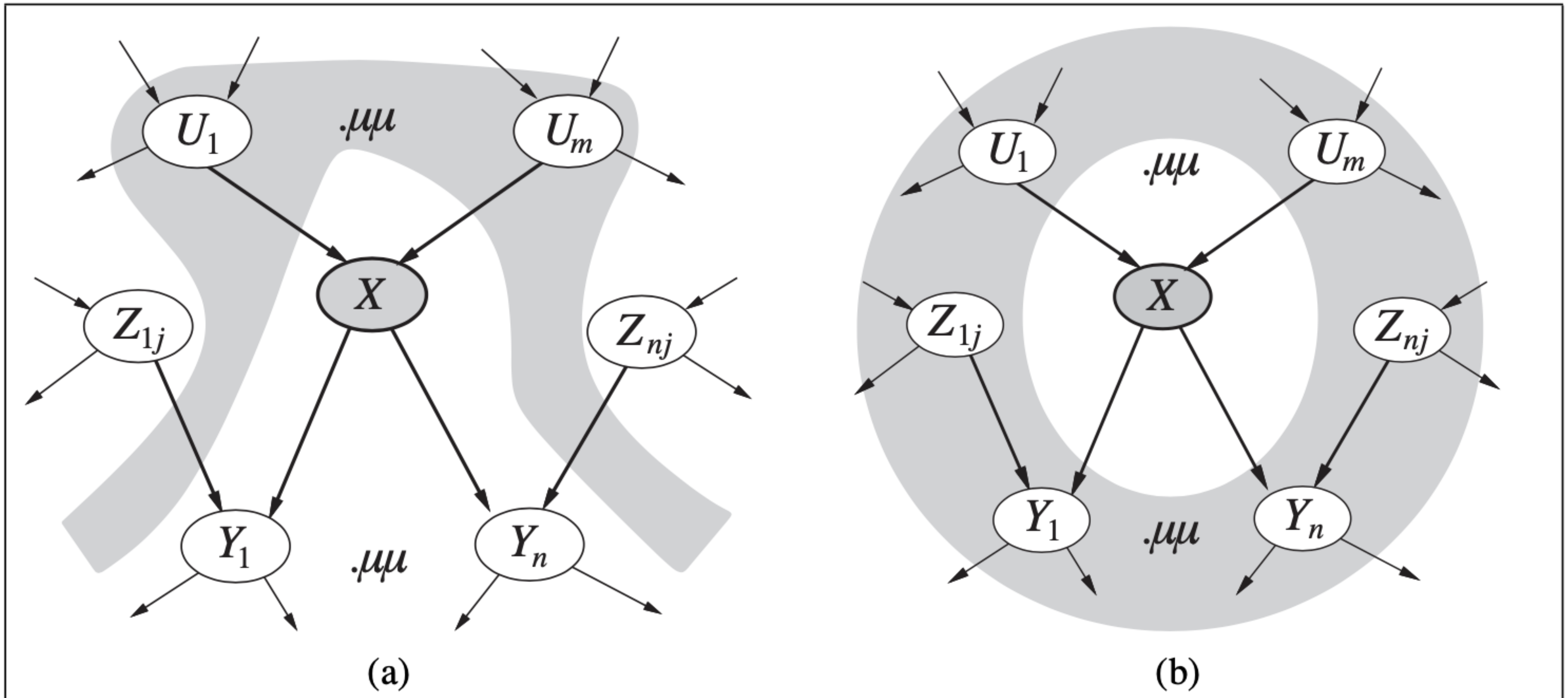


Figure 14.4 (a) A node X is conditionally independent of its non-descendants (e.g., the Z_{ij}s) given its parents (the U_is shown in the gray area). (b) A node X is conditionally independent of all other nodes in the network given its Markov blanket (the gray area).

Bayesian nets with continuous variables

- Many real-world problems involve continuous quantities.
- By definition, continuous variables have an infinite number of possible values, so it is impossible to specify conditional probabilities explicitly for each value.
- One possible way to handle continuous variables is using **discretization**—that is, dividing up the possible values into a fixed set of intervals.
- For example, temperatures could be divided into (<0C), (0C−100C), and (>100C). Discretization is sometimes an adequate solution, but often results in a considerable loss of accuracy and very large CPTs.

Bayesian nets with continuous variables

- The most common solution is to define standard families of probability density functions that are specified by a finite number of **parameters**. For example, a Gaussian (or normal) distribution $N(\mu,\sigma^2)(x)$ has the mean μ and the variance σ^2 as parameters.
- Yet another solution—sometimes called a **nonparametric** representation—is to define the conditional distribution implicitly with a collection of instances, each containing specific values of the parent and child variables.

Bayesian nets with continuous variables

- A network with both discrete and continuous variables is called a **hybrid Bayesian network**.

- Consider a customer buys some fruit depending on its cost, which depends in turn on the size of the harvest (收穫) and whether the government's subsidy (補貼) scheme is operating. The variable *Cost* is continuous and has continuous and discrete parents; the variable *Buys* is discrete and has a continuous parent.

Bayesian nets with continuous variables

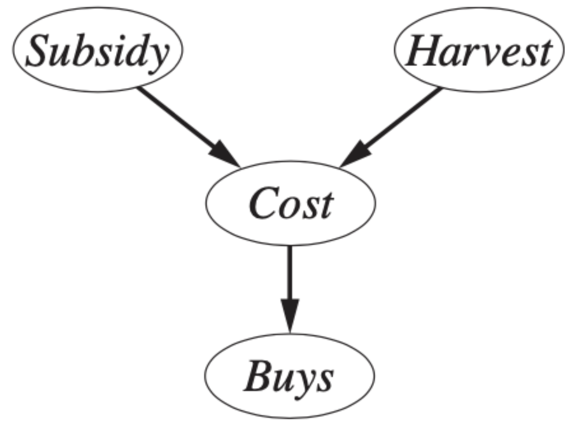


Figure 14.5 A simple network with discrete variables (*Subsidy* and *Buys*) and continuous variables (*Harvest* and *Cost*).

Bayesian nets with continuous variables

- For the *Cost* variable, we need to specify $\mathbf{P}(Cost \mid Harvest, Subsidy)$. The discrete parent is handled by enumeration—that is, by specifying both $P(Cost \mid Harvest, subsidy)$ and $P(Cost \mid Harvest, \neg subsidy)$.
- To handle *Harvest*, we specify how the distribution over the cost c depends on the continuous value h of *Harvest*. In other words, we specify the *parameters* of the cost distribution as a function of h. The most common choice is the **linear Gaussian** distribution, in which the child has a Gaussian distribution whose mean μ varies linearly with the value of the parent and whose standard deviation σ is fixed.

Bayesian nets with continuous variables

- We need two distributions, one for *subsidy* and one for ¬*subsidy*, with different parameters:

$$P(c \mid h, subsidy) = N(a_t h + b_t, \sigma_t^2)(c) = \frac{1}{\sigma_t \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{c-(a_t h + b_t)}{\sigma_t}\right)^2}$$

$$P(c \mid h, \neg subsidy) = N(a_f h + b_f, \sigma_f^2)(c) = \frac{1}{\sigma_f \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{c-(a_f h + b_f)}{\sigma_f}\right)^2}$$

- For this example, then, the conditional distribution for *Cost* is specified by naming the linear Gaussian distribution and providing the parameters a_t, b_t, σ_t, a_f, b_f, and σ_f. Figures 14.6(a) and (b) show these two relationships.

Bayesian nets with continuous variables

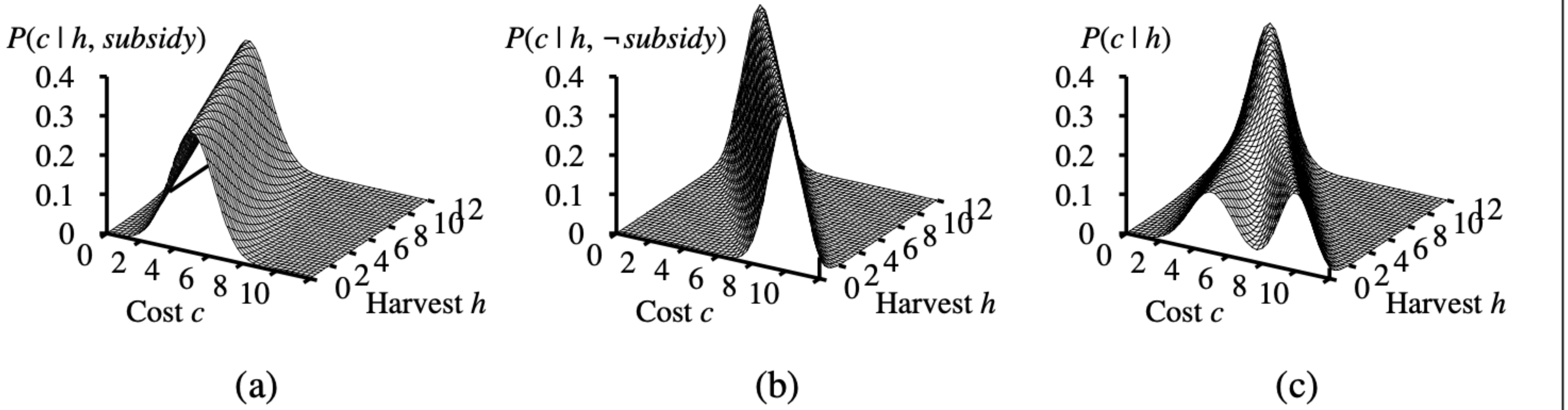


Figure 14.6 The graphs in (a) and (b) show the probability distribution over *Cost* as a function of *Harvest* size, with *Subsidy* true and false, respectively. Graph (c) shows the distribution $P(Cost \mid Harvest)$, obtained by summing over the two subsidy cases.

Bayesian nets with continuous variables

- Figure 14.6(c) shows the distribution $P(c \mid h)$, averaging over the two possible values of *Subsidy* and assuming that each has prior probability 0.5. This shows that even with very simple models, quite interesting distributions can be represented.

Bayesian nets with continuous variables

- It seems reasonable to assume that the customer will buy if the cost is low and will not buy if it is high and that the probability of buying varies smoothly in some intermediate region. The conditional distribution is like a "soft" threshold function. One way to make soft thresholds is to use the integral of the standard normal distribution:

$$\Phi(x) = \int_{-\infty}^{x} N(0,1)(x)dx$$

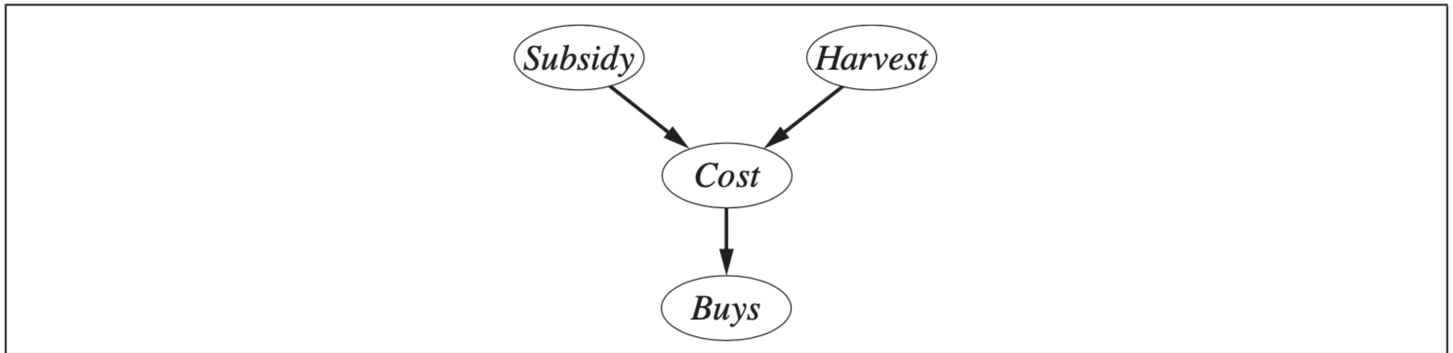


Figure 14.5 A simple network with discrete variables (*Subsidy* and *Buys*) and continuous variables (*Harvest* and *Cost*).

Bayesian nets with continuous variables

- Then the probability of *Buys* given *Cost* might be

$$P(buys \mid Cost = c) = \Phi((-c + \mu)/\sigma)$$

which means that the cost threshold occurs around μ, the width of the threshold region is proportional to σ, and the probability of buying decreases as cost increases.

- This **probit distribution** (pronounced "pro-bit" and short for "probability unit") is illustrated in Figure 14.7(b).

Bayesian nets with continuous variables

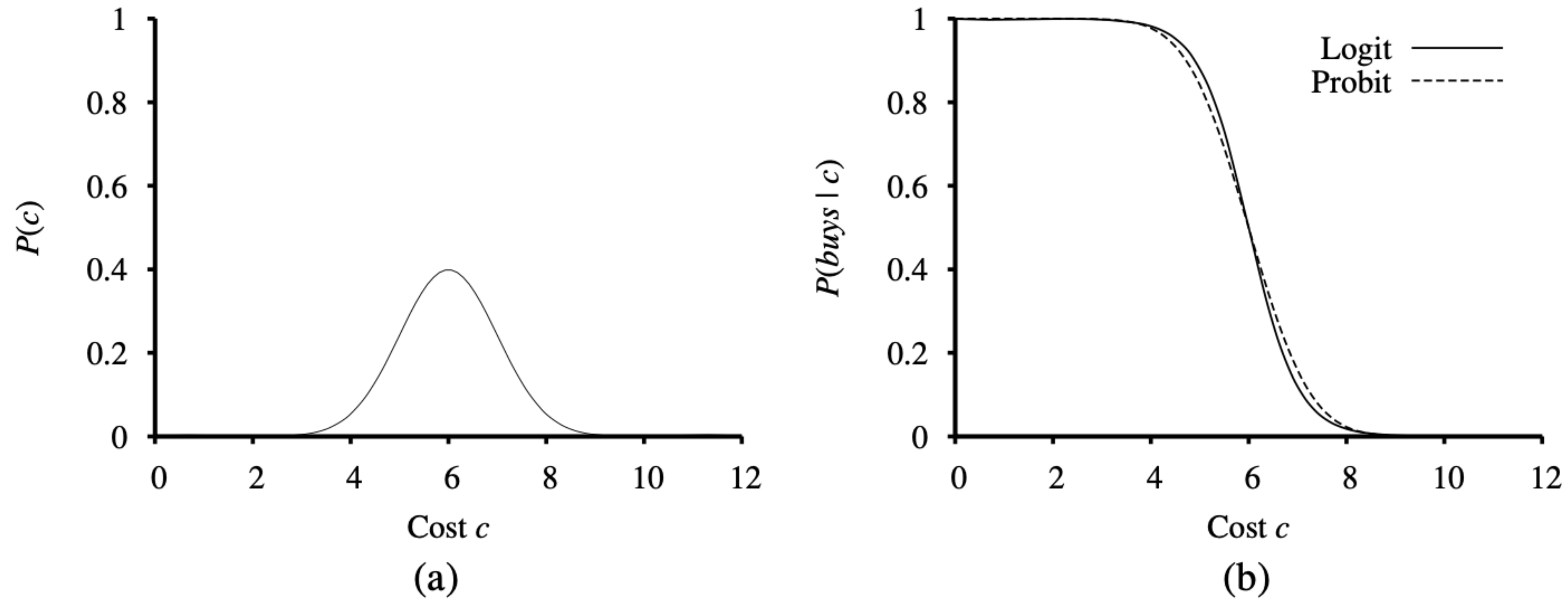


Figure 14.7 (a) A normal (Gaussian) distribution for the cost threshold, centered on $\mu = 6.0$ with standard deviation $\sigma = 1.0$. (b) Logit and probit distributions for the probability of *buys* given *cost*, for the parameters $\mu = 6.0$ and $\sigma = 1.0$.

Bayesian nets with continuous variables

- An alternative to the probit model is the **logit distribution** (pronounced “low-jit”). It uses the logistic function $1/(1 + e^{-x})$ to produce a soft threshold:

$$P(buys \mid Cost = c) = \frac{1}{1 + exp(-2\frac{-c+\mu}{\sigma})}$$

- This is illustrated in Figure 14.7(b). The two distributions look similar, but the logit actually has much longer “tails.” The probit is often a better fit to real situations, but the logit is sometimes easier to deal with mathematically. It is used widely in neural networks.

Exact Inference in Bayesian Networks

- The basic task for any probabilistic inference system is to compute the posterior probability distribution for a set of **query variables**, given some observed **event**.

- Let X denotes the query variable; **E** denotes the set of evidence variables $E_1, \ldots, E_m$, and **e** is a particular observed event; Y will denote the nonevidence, nonquery variables $Y_1, \ldots, Y_l$ (called the **hidden variables**). Thus, the complete set of variables is $\mathbf{X} = \{X\} \cup \mathbf{E} \cup \mathbf{Y}$. A typical query asks for the posterior probability distribution $\mathbf{P}(X \mid \mathbf{e})$.

Exact Inference in Bayesian Networks

- In the burglary network, we might observe the event in which *JohnCalls* = *true* and *MaryCalls* = *true*. We could then ask for, say, the probability that a burglary has occurred:

$$\mathbf{P}(Burglary \mid JohnCalls = true, MaryCalls = true) = \langle 0.284, 0.716 \rangle$$

Inference by enumeration

- Any conditional probability can be computed by summing terms from the full joint distribution.

$$\mathbf{P}(X \mid \mathbf{e}) = \alpha\,\mathbf{P}(X, \mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{e}, \mathbf{y})$$

- More specifically, Equation (14.2) shows that the terms $P(x, \mathbf{e}, \mathbf{y})$ in the joint distribution can be written as products of conditional probabilities from the network. Therefore, *a query can be answered using a Bayesian network by computing sums of products of conditional probabilities from the network.*

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid parents(X_i)) \,. \qquad (14.2)$$

Inference by enumeration

- Consider the query **P**(*Burglary* | *JohnCalls*=*true*, *MaryCalls*=*true*). The hidden variables for this query are *Earthquake* and *Alarm*. Using initial letters for the variables to shorten the expressions, we have

$$\mathbf{P}(B\,|\,j,m) = \alpha\,\mathbf{P}(B,j,m) = \alpha \sum_{e} \sum_{a} \mathbf{P}(B,j,m,e,a,)$$

- The semantics of Bayesian networks (Equation (14.2)) then gives us an expression in terms of CPT entries. For simplicity, we do this just for *Burglary* = *true*:

$$P(b\,|\,j,m) = \alpha \sum_{e} \sum_{a} P(b)P(e)P(a\,|\,b,e)P(j\,|\,a)P(m\,|\,a)$$

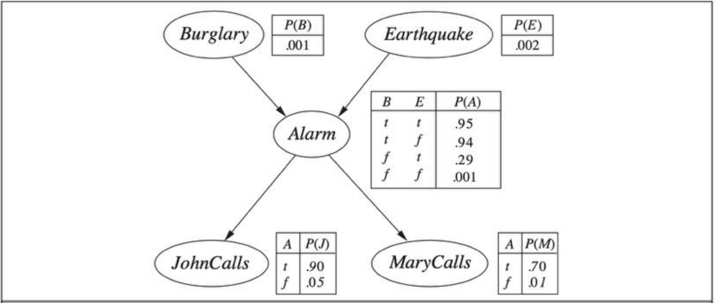


Figure 14.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.

$$\boxed{P(x_1,\ldots,x_n) = \prod_{i=1}^{n} P(x_i\,|\,parents(X_i))\,. \qquad (14.2)}$$

Inference by enumeration

- To compute this expression, we have to add four terms, each computed by multiplying five numbers. In the worst case, where we have to sum out almost all the variables, the complexity of the algorithm for a network with *n* Boolean variables is $O(n2^n)$.
- An improvement can be obtained from the following simple observations: the $P(b)$ term is a constant and can be moved outside the summations over *a* and *e*, and the $P(e)$ term can be moved outside the summation over *a*. Hence, we have

$$P(b \mid j, m) = \alpha P(b) \sum_e P(e) \sum_a P(a \mid b, e) P(j \mid a) P(m \mid a) \qquad (14.4)$$

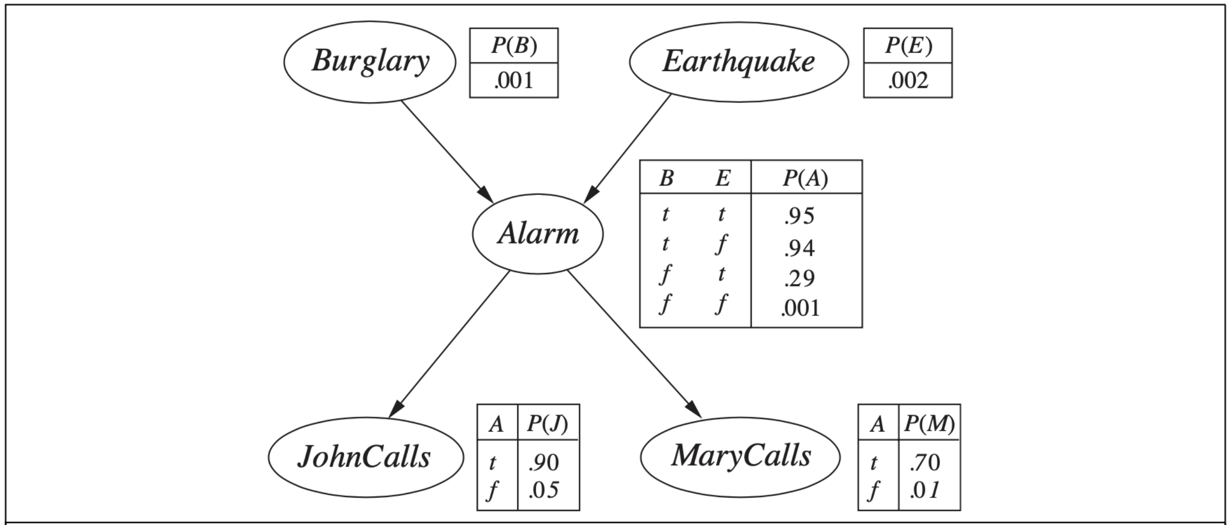
國立成功大學 National Cheng Kung University 1931

Inference by enumeration

- The structure of this computation is shown in Figure 14.8. Using the numbers from Figure 14.2, we obtain $P(b \mid j, m) = \alpha \times 0.00059224$. The corresponding computation for $\neg b$ yields $\alpha \times 0.0014919$; hence,

$$\mathbf{P}(B \mid j, m) = \alpha \langle 0.00059224, 0.0014919 \rangle \approx \langle 0.284, 0.716 \rangle$$

That is, the chance of a burglary, given calls from both neighbors, is about 28%.



P(B)
.001

P(E)
.002

B	E	P(A)
t	t	.95
t	f	.94
f	t	.29
f	f	.001

A	P(J)
t	.90
f	.05

A	P(M)
t	.70
f	.01

Figure 14.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.

Inference by enumeration

$$P(b \mid j, m) = \alpha P(b) \sum_e P(e) \sum_a P(a \mid b, e) P(j \mid a) P(m \mid a)$$

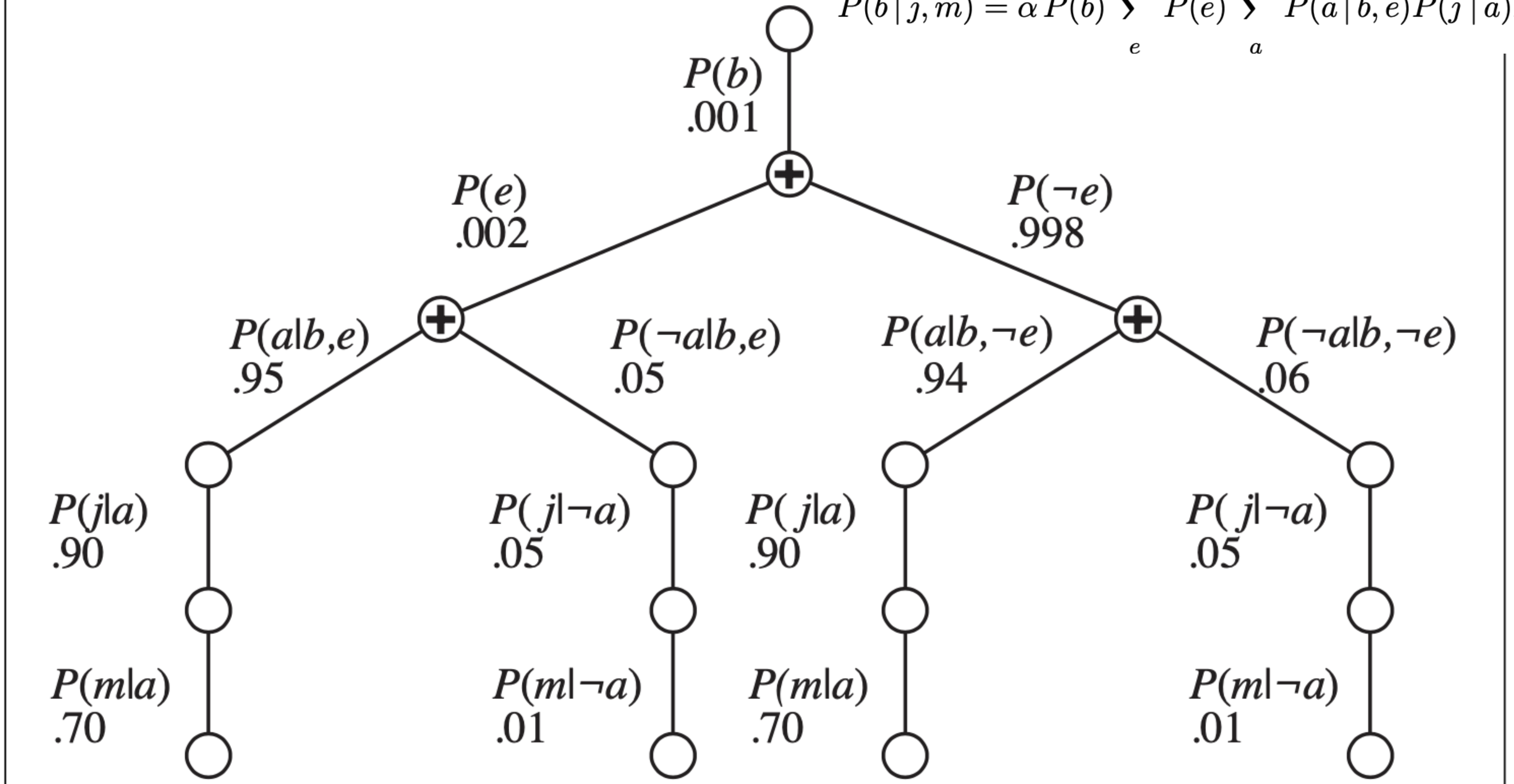


Figure 14.8 The structure of the expression shown in Equation (14.4). The evaluation proceeds top down, multiplying values along each path and summing at the "+" nodes. Notice the repetition of the paths for j and m.

The complexity of exact inference

- The complexity of exact inference in Bayesian networks depends strongly on the structure of the network.
- The burglary network has at most one path between any two nodes. These are called **singly connected** networks or **polytrees**. *The time and space complexity of exact inference in polytrees is linear in the size of the network*.

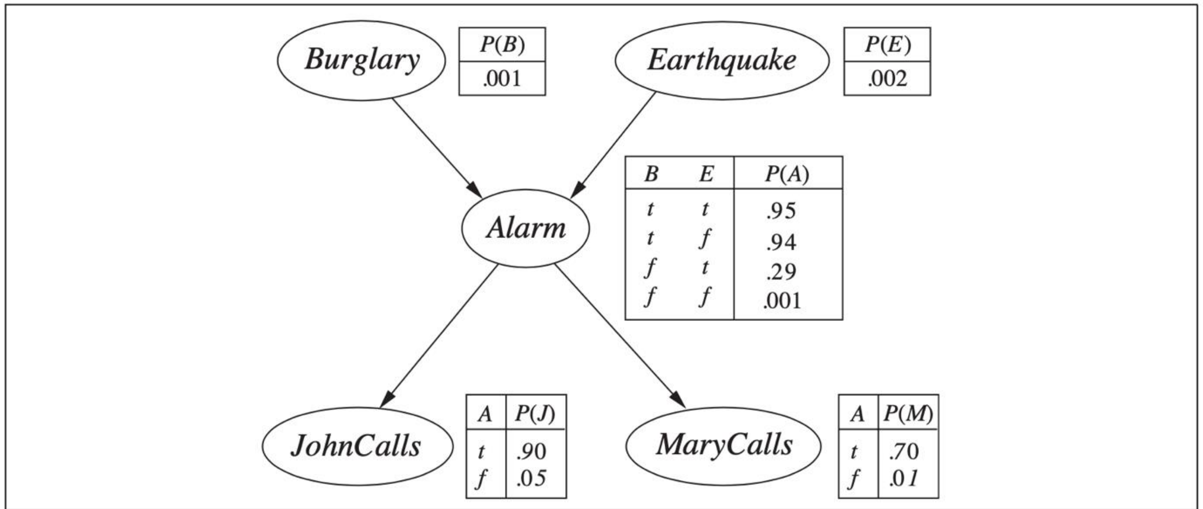
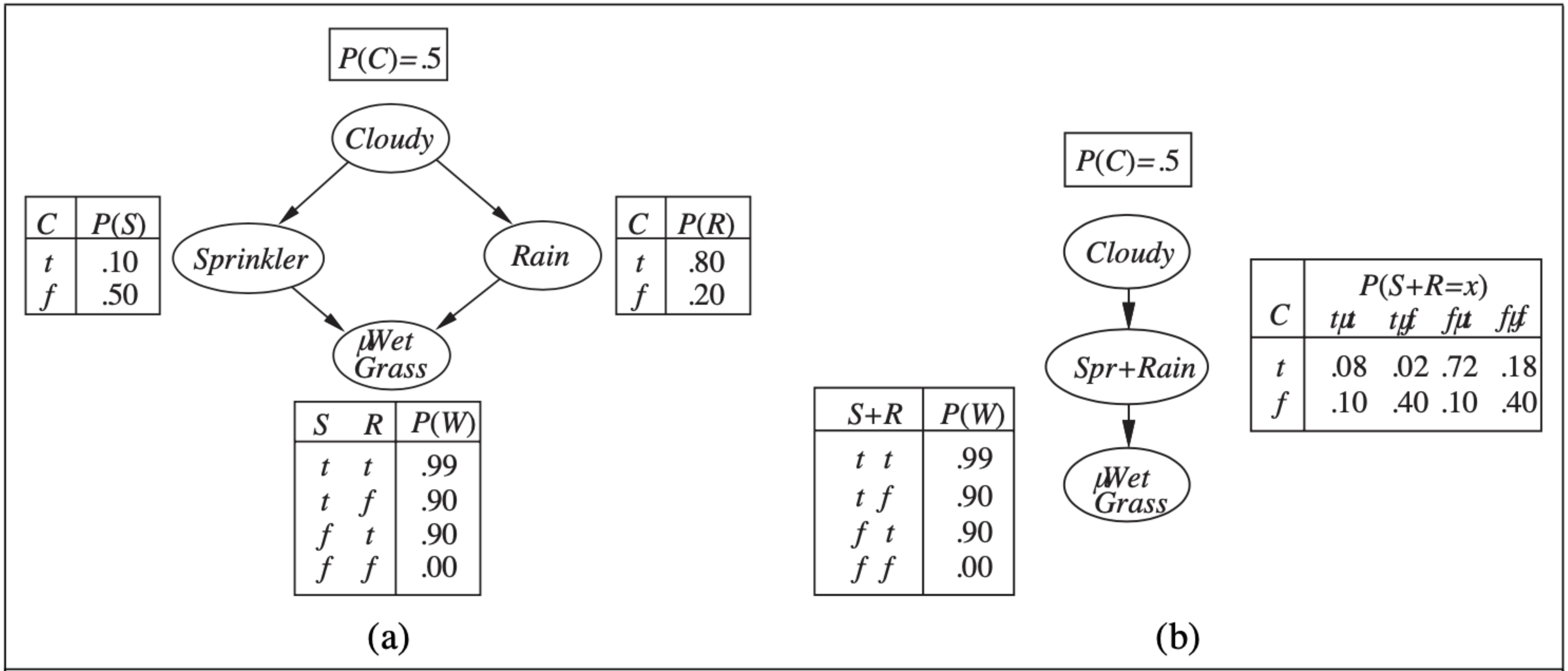


Figure 14.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.

The complexity of exact inference

- For **multiply connected** networks, such as that of Figure 14.12(a), variable elimination can have exponential time and space complexity in the worst case, even when the number of parents per node is bounded. In fact, it can be shown that the problem is an NP-hard problem.

The complexity of exact inference



C	P(S)
t	.10
f	.50

C	P(R)
t	.80
f	.20

S	R	P(W)
t	t	.99
t	f	.90
f	t	.90
f	f	.00

S+R	P(W)
t t	.99
t f	.90
f t	.90
f f	.00

C	P(S+R=x)			
	tt	tf	ft	ff
t	.08	.02	.72	.18
f	.10	.40	.10	.40

Figure 14.12 (a) A multiply connected network with conditional probability tables. (b) A clustered equivalent of the multiply connected network.

Approximate Inference in Bayesian Nets

- Given the intractability of exact inference in large, multiply connected networks, it is essential to consider approximate inference methods.
- **Monte Carlo** algorithms provide approximate answers whose accuracy depends on the number of samples generated.
- Monte Carlo algorithms are used in many branches of science to estimate quantities that are difficult to calculate exactly. In this section, we describe two families of algorithms: **direct sampling** and **Markov chain sampling**.

Direct sampling methods

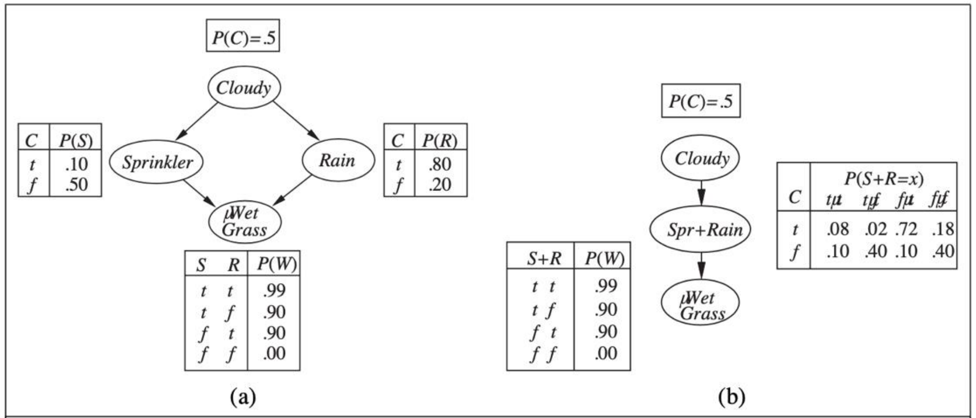
- The primitive element in any sampling algorithm is the generation of samples from a known probability distribution. For example, an unbiased coin can be thought of as a random variable *Coin* with values ⟨*heads*, *tails*⟩ and a prior distribution $\mathbf{P}(Coin) = \langle 0.5, 0.5 \rangle$.
- Sampling from this distribution is exactly like flipping the coin: with probability 0.5 it will return heads, and with probability 0.5 it will return tails.
- Sample each variable in turn, in topological order. The probability distribution from which the value is sampled is conditioned on the values already assigned to the variable's parents.

Direct sampling methods

- We can illustrate its operation on the network in Figure 14.12(a), assuming an ordering [*Cloudy*, *Sprinkler*, *Rain*, *WetGrass*]:

1. Sample from $\mathbf{P}(Cloudy) = \langle 0.5, 0.5 \rangle$, value is $true$.
2. Sample from $\mathbf{P}(Sprinkler \mid Cloudy = true) = \langle 0.1, 0.9 \rangle$, value is $false$.
3. Sample from $\mathbf{P}(Rain \mid Cloudy = true) = \langle 0.8, 0.2 \rangle$, value is $true$.
4. Sample from $\mathbf{P}(WetGrass \mid Sprinkler = false, Rain = true) = \langle 0.9, 0.1 \rangle$, value is $true$.

- In this case, PRIOR-SAMPLE returns the event [*true*, *false*, *true*, *true*].



P(C)=.5
Cloudy
Sprinkler
Rain
Wet Grass
C P(S): t .10, f .50
C P(R): t .80, f .20
S R P(W): t t .99, t f .90, f t .90, f f .00
(a)
P(C)=.5
Cloudy
Spr+Rain
Wet Grass
S+R P(W): t t .99, t f .90, f t .90, f f .00
C P(S+R=x) t,t t,f f,t f,f: t .08 .02 .72 .18; f .10 .40 .10 .40
(b)

Figure 14.12 (a) A multiply connected network with conditional probability tables. (b) A clustered equivalent of the multiply connected network.

Direct sampling methods

- It is easy to see that PRIOR-SAMPLE generates samples from the prior joint distribution specified by the network. First, let $S_{PS}(x_1, \ldots, x_n)$ be the probability that a specific event is generated by the PRIOR-SAMPLE algorithm. Just looking at the sampling process, we have

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

because each sampling step depends only on the parent values.

Direct sampling methods

- This expression should look familiar, because it is also the probability of the event according to the Bayesian net's representation of the joint distribution, as stated in Equation (14.2). That is, we have

$$S_{PS}(x_1 \ldots x_n) = P(x_1 \ldots x_n)$$

  This simple fact makes it easy to answer questions by using samples.

Direct sampling methods

- Suppose there are N total samples, and let $N_{PS}(x_1, \ldots, x_n)$ be the number of times the specific event $x_1, \ldots, x_n$ occurs in the set of samples. We expect this number, as a fraction of the total, to converge in the limit to its expected value according to the sampling probability:

$$\lim_{N\to\infty} \frac{N_{PS}(x_1, \ldots, x_n)}{N} = S_{PS}(x_1, \ldots, x_n) = P(x_1, \ldots, x_n)$$

For example, consider the event produced earlier: [*true*, *false*, *true*, *true*]. The sampling probability for this event is

$$S_{PS}(true, false, true, true) = 0.5 \times 0.9 \times 0.8 \times 0.9 = 0.324$$

in the limit of large N, we expect 32.4% of the samples to be of this event.

Direct sampling methods

- Whenever we use ("≈"), we mean that **the estimated probability becomes exact in the large-sample limit**. Such an estimate is called **consistent**.
- For example, one can produce a consistent estimate of the probability of any partially specified event $x_1, \ldots, x_m$, where $m \leq n$, as follows:

$$P(x_1, \ldots, x_m) \approx N_{PS}(x_1, \ldots, x_m)/N \tag{14.6}$$

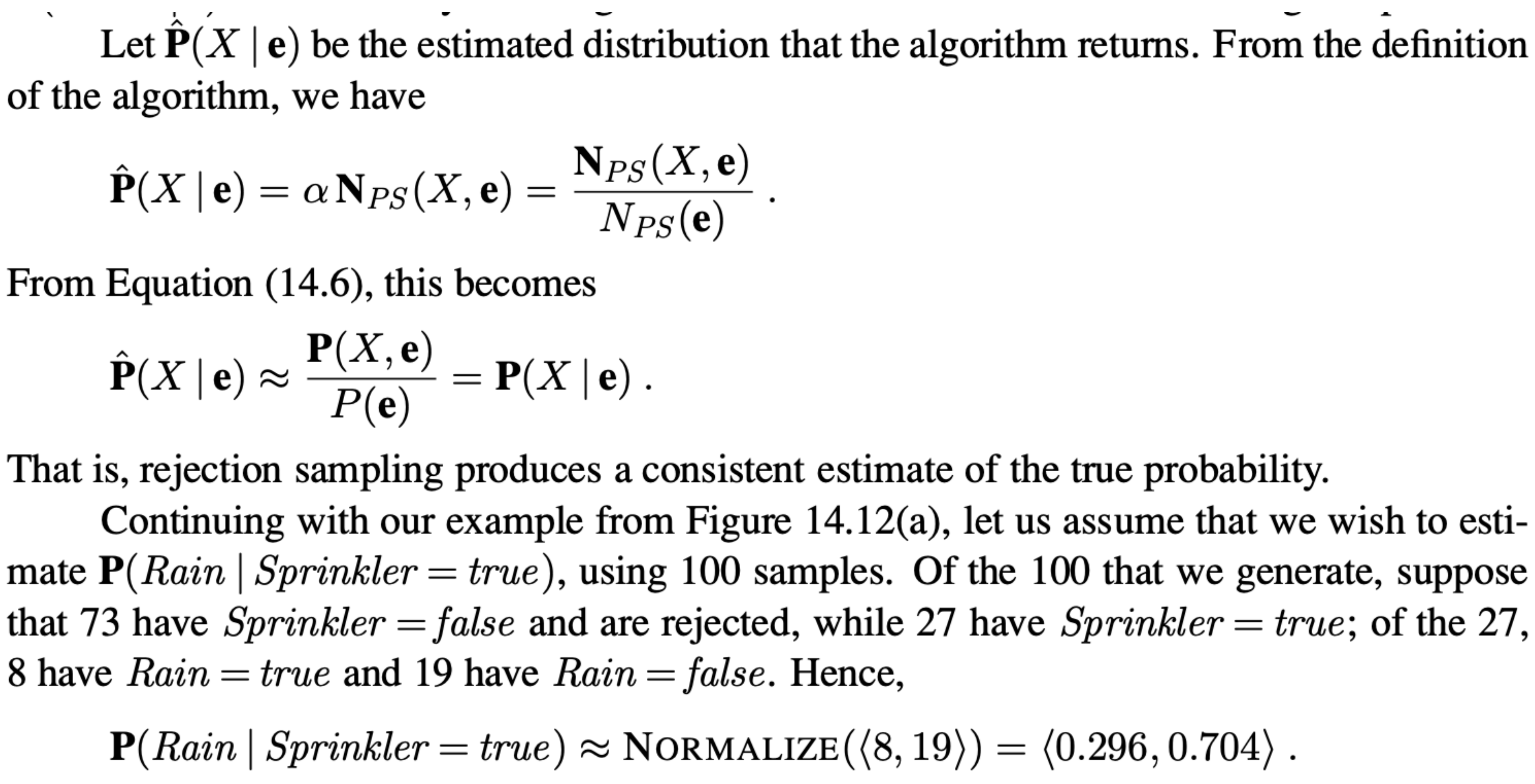
That is, the probability of the event can be estimated as the fraction of all complete events generated by the sampling process that match the partially specified event. For example, if we generate 1000 samples from the sprinkler network, and 511 of them have *Rain* = *true*, then the estimated probability of rain, written as $P(Rain = true)$, is 0.511.

Direct sampling methods

- **Rejection sampling in Bayesian networks**
- **Rejection sampling** is a general method for producing samples from a hard-to-sample distribution given an easy-to-sample distribution. In its simplest form, it can be used to compute conditional probabilities—that is, to determine $P(X \mid \mathbf{e})$.
- The REJECTION-SAMPLING algorithm first generates samples from the prior distribution specified by the network. Then, it rejects all those that do not match the evidence.

  Finally, the estimate $P(X = x \mid \mathbf{e})$ is obtained by counting how often $X = x$ occurs in the remaining samples.

Direct sampling methods



Let $\hat{\mathbf{P}}(X \mid \mathbf{e})$ be the estimated distribution that the algorithm returns. From the definition of the algorithm, we have

$$\hat{\mathbf{P}}(X \mid \mathbf{e}) = \alpha\, \mathbf{N}_{PS}(X, \mathbf{e}) = \frac{\mathbf{N}_{PS}(X, \mathbf{e})}{N_{PS}(\mathbf{e})} .$$

From Equation (14.6), this becomes

$$\hat{\mathbf{P}}(X \mid \mathbf{e}) \approx \frac{\mathbf{P}(X, \mathbf{e})}{P(\mathbf{e})} = \mathbf{P}(X \mid \mathbf{e}) .$$

That is, rejection sampling produces a consistent estimate of the true probability.

Continuing with our example from Figure 14.12(a), let us assume that we wish to estimate $\mathbf{P}(Rain \mid Sprinkler = true)$, using 100 samples. Of the 100 that we generate, suppose that 73 have $Sprinkler = false$ and are rejected, while 27 have $Sprinkler = true$; of the 27, 8 have $Rain = true$ and 19 have $Rain = false$. Hence,

$$\mathbf{P}(Rain \mid Sprinkler = true) \approx \textsc{Normalize}(\langle 8, 19 \rangle) = \langle 0.296, 0.704 \rangle .$$

Direct sampling methods

- The true answer is ⟨0.3, 0.7⟩. As more samples are collected, the estimate will converge to the true answer. The standard deviation of the error in each probability will be proportional to $1/\sqrt{n}$, where *n* is the number of samples used in the estimate.
- The biggest problem with rejection sampling is that it rejects so many samples! The fraction of samples consistent with the evidence **e** drops exponentially as the number of evidence variables grows, so the procedure is simply unusable for complex problems.

Direct sampling methods

- Notice that rejection sampling is very similar to the estimation of conditional probabilities directly from the real world. For example, to estimate **P**(*Rain* | *RedSkyAtNight* = true), one can simply count how often it rains after a red sky is observed the previous evening— ignoring those evenings when the sky is not red. (Here, the world itself plays the role of the sample-generation algorithm.) Obviously, this could take a long time if the sky is very seldom red, and that is the weakness of rejection sampling.

Inference by Markov chain simulation

- **Markov chain Monte Carlo** (MCMC) algorithms work quite differently from rejection sampling and likelihood weighting. Instead of generating each sample from scratch, MCMC algorithms generate each sample by making a random change to the preceding sample.
- Here we describe a particular form of MCMC called **Gibbs sampling**, which is especially well suited for Bayesian networks.

Inference by Markov chain simulation

- **Gibbs sampling in Bayesian networks**
- The Gibbs sampling algorithm for Bayesian networks starts with an arbitrary state (with the evidence variables fixed at their observed values) and generates a next state by randomly sampling a value for one of the nonevidence variables X_i. The sampling for X_i is done conditioned on the current values of the variables in the Markov blanket of X_i. (Recall that the Markov blanket of a variable consists of its parents, children, and children's parents.) The algorithm therefore flips one variable at a time, but keeping the evidence variables fixed.

Inference by Markov chain simulation

- Consider the query **P**(*Rain* | *Sprinkler* = *true*, *WetGrass* = *true*) applied to the network in Figure 14.12(a). The evidence variables *Sprinkler* and *WetGrass* are fixed to their observed values and the nonevidence variables *Cloudy* and *Rain* are initialized randomly— let us say to *true* and *false* respectively. Thus, the initial state is [*true*, *true*, *false*, *true*]. Now the nonevidence variables are [*Cloudy*, *Sprinkler*, *Rain*, *WetGrass*] sampled repeatedly in an arbitrary order.

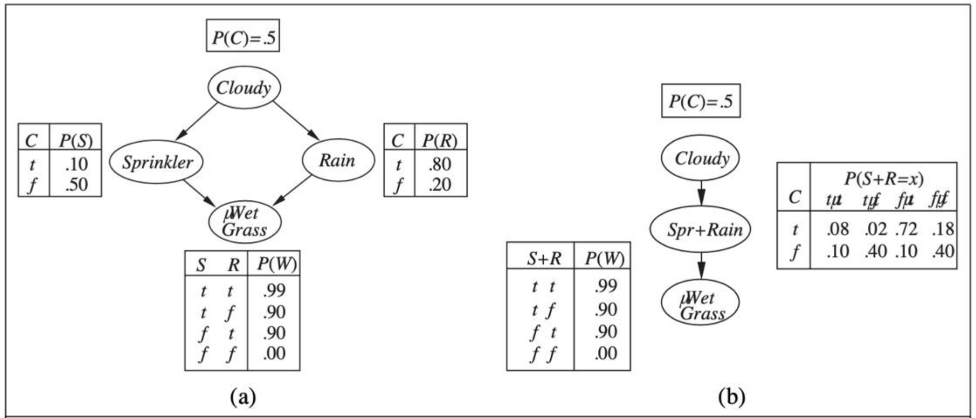
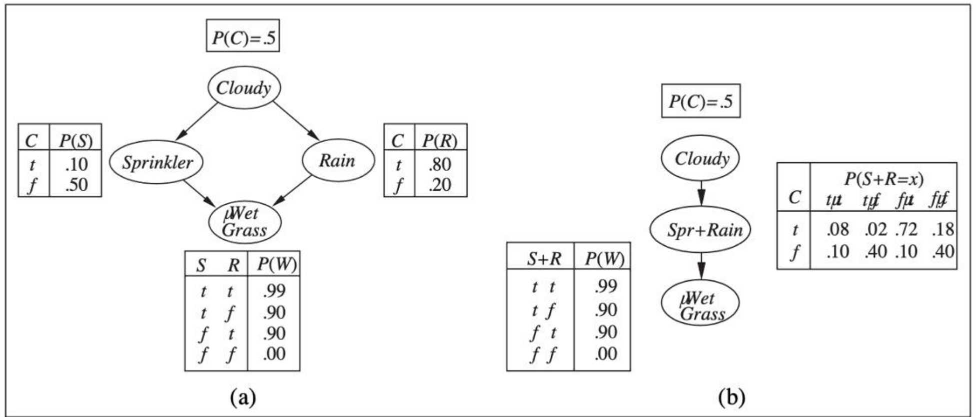


Figure 14.12 (a) A multiply connected network with conditional probability tables. (b) A clustered equivalent of the multiply connected network.

Inference by Markov chain simulation

1. *Cloudy* is sampled, given the current values of its Markov blanket variables: in this case, we sample from $\mathbf{P}(Cloudy \mid Sprinkler = true, Rain = false)$. (Shortly, we will show how to calculate this distribution.) Suppose the result is $Cloudy = false$. Then the new current state is $[false, true, false, true]$.
2. *Rain* is sampled, given the current values of its Markov blanket variables: in this case, we sample from $\mathbf{P}(Rain \mid Cloudy = false, Sprinkler = true, WetGrass = true)$. Suppose this yields $Rain = true$. The new current state is $[false, true, true, true]$.

Each state visited during this process is a sample that contributes to the estimate for the query variable *Rain*. If the process visits 20 states where *Rain* is true and 60 states where *Rain* is false, then the answer to the query is $\text{NORMALIZE}(\langle 20, 60\rangle) = \langle 0.25, 0.75\rangle$. The complete algorithm is shown in Figure 14.16.



C	P(S)
t	.10
f	.50

C	P(R)
t	.80
f	.20

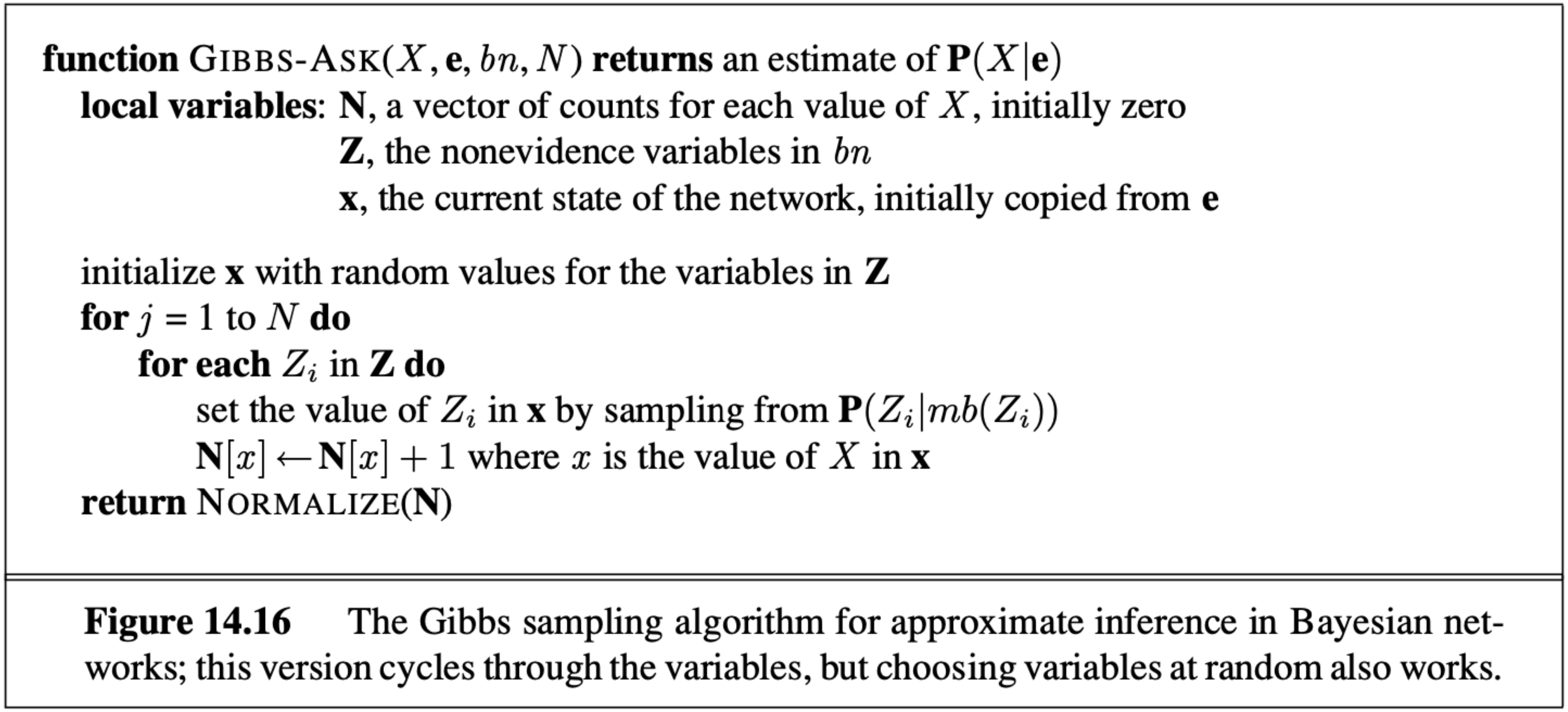
S	R	P(W)
t	t	.99
t	f	.90
f	t	.90
f	f	.00

S+R	P(W)
t t	.99
t f	.90
f t	.90
f f	.00

C	P(S+R=x)			
	tt	tf	ft	ff
t	.08	.02	.72	.18
f	.10	.40	.10	.40

Figure 14.12 (a) A multiply connected network with conditional probability tables. (b) A clustered equivalent of the multiply connected network.

Inference by Markov chain simulation



```
function GIBBS-ASK($X$, $\mathbf{e}$, $bn$, $N$) returns an estimate of $\mathbf{P}(X|\mathbf{e})$
  local variables: $\mathbf{N}$, a vector of counts for each value of $X$, initially zero
                   $\mathbf{Z}$, the nonevidence variables in $bn$
                   $\mathbf{x}$, the current state of the network, initially copied from $\mathbf{e}$

  initialize $\mathbf{x}$ with random values for the variables in $\mathbf{Z}$
  for $j$ = 1 to $N$ do
      for each $Z_i$ in $\mathbf{Z}$ do
          set the value of $Z_i$ in $\mathbf{x}$ by sampling from $\mathbf{P}(Z_i|mb(Z_i))$
          $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where $x$ is the value of $X$ in $\mathbf{x}$
  return NORMALIZE($\mathbf{N}$)
```

Figure 14.16 The Gibbs sampling algorithm for approximate inference in Bayesian networks; this version cycles through the variables, but choosing variables at random also works.