1.2 Graphs and Probabilities

1.2.1 Graphical Notation and Terminology

A graph consists of a set V of *vertices* (or *nodes*) and a set E of *edges* (or *links*) that connect some pairs of vertices. The vertices in our graphs will correspond to variables (whence the common symbol V) and the edges will denote a certain relationship that holds in pairs of variables, the interpretation of which will vary with the application. Two variables connected by an edge are called *adjacent.*

Each edge in a graph can be either directed (marked by a single arrowhead on the edge), or undirected (unmarked links). In some applications we will also use "bidirected" edges to denote the existence of unobserved common causes (sometimes called *confounders*). These edges will be marked as dotted curved arcs with two arrowheads (see Figure 1.1(a)). If all edges are directed (see Figure 1.1(b)), we then have a *di-*

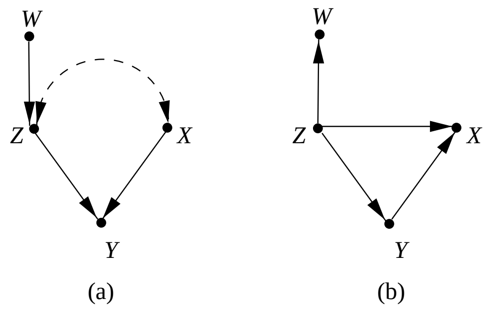

Figure 1.1: (a) A graph containing both directed and bidirected edges. (b) A directed acyclic graph (DAG) with the same skeleton as (a).

rected graph. If we strip away all arrowheads from the edges in a graph G, the resultant undirected graph is called the *skeleton* of G. A *path* in a graph is a sequence of edges (e.g., $((W,Z),(Z,Y),(Y,X),(X,Z))$ in Figure 1.1(a)) such that each edge starts with the vertex ending the preceding edge. In other words, a path is any unbroken, nonintersecting route traced out along the edges in a graph, which may go either along or against the arrows. If every edge in a path is an arrow that points from the first to the second vertex of the pair, we have a *directed path.* In Figure 1.1(a), for example, the path $((W,Z),(Z,Y))$ is directed but the paths $((W, Z), (Z, Y), (Y, X))$ and $((W, Z), (Z, X))$ are not. If there exists a path between two vertices in a graph then the two vertices are said to be *connected*; else they are *disconnected.*

Directed graphs may include directed cycles (e.g., $X \longrightarrow Y$, $Y \longrightarrow X$), representing mutual causation or feedback processes, but not self-loops (e.g., $X \longrightarrow X$). A graph (like the two in Figure 1.1) that contains no directed cycles is called *acyclic.* A graph that is both directed and acyclic (Figure 1.1(b)) is called a *directed acyclic graph* (DAG), and such graphs will occupy much of our discussion of causality. We make free use of the terminology of kinship (e.g., *parents, children, descendants, ancestors, spouses*) to denote various relationships in a graph. These kinship relations are defined along the full arrows in the graph, including arrows that form directed cycles but ignoring bidirected and undirected edges. In Figure 1.1(a), for example, Y has two parents (X and Z), three ancestors (X, Z, and W), and no children, while X has no parents (hence, no ancestors), one spouse (Z), and one child (Y). A $family$ in a graph is a set of nodes containing a node and all its parents. For example, $\{W\}, \{Z, W\}, \{X\}$, and $\{Y, Z, X\}$ are the families in the graph of Figure 1.1(a).

A node in a directed graph is called a *root* if it has no parents and a *sink* if it has no children. Every DAG has at least one root and at least one sink. A connected DAG in which every node has at most one parent is called a *tree*, and a tree in which every node has at most one child is called a *chain.* A graph in which every pair of nodes is connected by an edge is called *complete.* The graph in Figure 1.1(a), for instance, is connected but not complete, because the pairs (W, X) and (W, Y) are not adjacent.

1.2.2 Bayesian Networks

The role of graphs in probabilistic and statistical modeling is threefold:

1. to provide convenient means of expressing substantive assumptions;

2. to facilitate economical representation of joint probability functions; and

3. to facilitate efficient inferences from observations.

We will begin our discussion with item 2.

Consider the task of specifying an arbitrary joint distribution, $P(x_1, \ldots, x_n)$, for n dichotomous variables. To store $P(x_1, \ldots, x_n)$ explicitly would require a table with 2^n entries, an unthinkably large number by any standard. Substantial economy can be achieved when each variable depends on just a small subset of other variables. Such dependence information permits us to decompose large distribution functions into several small distributions—each involving a small subset of variables—and then to piece them together coherently to answer questions of global nature. Graphs play an essential role in such decomposition, for they provide a vivid representation of the sets of variables that are relevant to each other in any given state of knowledge.

Both directed and undirected graphs have been used by researchers to facilitate such decomposition. Undirected graphs, sometimes called *Markov networks* (Pearl 1988b), are used primarily to represent symmetrical spatial relationships (Isham 1981; Cox and Wermuth 1996; Lauritzen 1996). Directed graphs, especially DAGs, have been used to represent causal or temporal relationships (Lauritzen 1982; Wermuth and Lauritzen 1983; Kiiveri et al. 1984) and came to be known as *Bayesian networks*, a term coined in Pearl (1985) to emphasize three aspects: (1) the subjective nature of the input information; (2) the reliance on Bayes's conditioning as the basis for updating information; and (3) the distinction between causal and evidential modes of reasoning, a distinction that underscores Thomas Bayes's paper of 1763. Hybrid graphs (involving both directed and undirected edges) have also been proposed for statistical modeling (Wermuth and Lauritzen 1990), but in this book our main interest will focus on directed acyclic graphs, with occasional use of directed cyclic graphs to represent feedback cycles.

The basic decomposition scheme offered by directed acyclic graphs can be illustrated as follows. Suppose we have a distribution P defined on n discrete variables, which we may order arbitrarily as $X_1, X_2, \ldots, X_n$. The chain rule of probability calculus (equation (1.12)) always permits us to decompose P as a product of n conditional distributions:

$$P(x_1, \ldots, x_n) = \prod_j P(x_j | x_1, \ldots, x_{j-1}). \tag{1.30}$$

Now suppose that the conditional probability of some variable X_j is not sensitive to all the predecessors of X_j but only to a small subset of those predecessors. In other words, suppose that X_j is independent of all other predecessors, once we know the value of a select group of predecessors called PA_j. We can then write

$$P(x_j | x_1, \ldots, x_{j-1}) = P(x_j | pa_j) \tag{1.31}$$

in the product of (1.30), which will considerably simplify the input information required. Instead of specifying the probability of X_j conditional on all possible realizations of its predecessors $X_1, \ldots, X_{j-1}$, we need only concern ourselves with the possible realizations of the set PA_j. The set PA_j is called the *Markovian parents* of X_j, or *parents* for short. The reason for the name becomes clear when we build graphs around this concept.

Definition 1.2.1 (Markovian Parents)
Let $V = \{X_1, \ldots, X_n\}$ be an ordered set of variables, and let $P(v)$ be the joint probability distribution on these variables. A set of variables PA_j is said to be Markovian parents *of X_j if PA_j is a minimal set of predecessors of X_j that renders X_j independent of all its other predecessors. In other words, PA_j is any subset of $\{X_1, \ldots, X_{j-1}\}$ satisfying*

$$P(x_j | pa_j) = P(x_j | x_1, \ldots, x_{j-1}) \tag{1.32}$$

and such that no proper subset of PA_j satisfies (1.32).[5]

Definition 1.2.1 assigns to each variable X_j a select set PA_j of preceding variables that are sufficient for determining the probability of X_j; knowing the values of other preceding variables is redundant once we know the values pa_j of the parent set PA_j. This assignment can be represented in the form of a DAG in which variables are represented

[5]Lowercase symbols (e.g., x_j, pa_j) denote particular realizations of the corresponding variables (e.g., X_j, PA_j).

by nodes and arrows are drawn from each node of the parent set PA_j toward the child node X_j. Definition 1.2.1 also suggests a simple recursive method for constructing such a DAG: Starting with the pair (X_1, X_2), we draw an arrow from X_1 to X_2 if and only if the two variables are dependent. Continuing to X_3, we draw no arrow in case X_3 is independent of $\{X_1, X_2\}$; otherwise, we examine whether X_2 screens off X_3 from X_1 or X_1 screens off X_3 from X_2. In the first case, we draw an arrow from X_2 to X_3; in the second, we draw an arrow from X_1 to X_3. If no screening condition is found, we draw arrows to X_3 from both X_1 and X_2. In general: at the jth stage of the construction, we select any minimal set of X_j's predecessors that screens off X_j from its other predecessors (as in equation (1.32)), call this set PA_j, and draw an arrow from each member in PA_j to X_j. The result is a directed acyclic graph, called a Bayesian network, in which an arrow from X_i to X_j assigns X_i as a Markovian parent of X_j, consistent with Definition 1.2.1.

It can be shown (Pearl 1988b) that the set PA_j is unique whenever the distribution $P(v)$ is strictly positive (i.e., involving no logical or definitional constraints), so that every configuration v of variables, no matter how unlikely, has some finite probability of occurring. Under such conditions, the Bayesian network associated with $P(v)$ is unique, given the ordering of the variables.

Figure 1.2 illustrates a simple yet typical Bayesian network. It describes relationships among the season of the year (X_1), whether rain falls (X_2), whether the sprinkler is on (X_3), whether the pavement would get wet (X_4), and whether the pavement would be slippery (X_5). All variables in this figure are binary (taking a value of either true or false) except for the root variable X_1, which can take one of four values: spring, summer, fall, or winter. The network was constructed in accordance with Definition 1.2.1, using causal intuition as a guide. The absence of a direct link between X_1 and X_5, for example, captures our understanding that the influence of seasonal variations on the slipperiness of the pavement is mediated by other conditions (e.g., the wetness of the pavement). This intuition coincides with the independence condition of (1.32), since knowing X_4 renders X_5 independent of $\{X_1, X_2, X_3\}$.

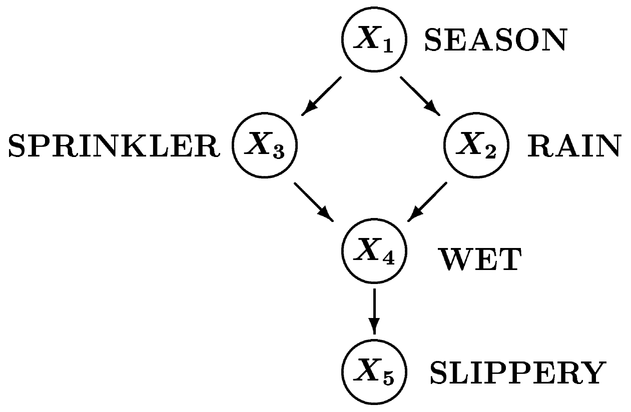

Figure 1.2: A Bayesian network representing dependencies among five variables.

The construction implied by Definition 1.2.1 defines a Bayesian network as a carrier of conditional independence relationships along the order of construction. Clearly, every distribution satisfying (1.32) must decompose (using the chain rule of (1.30)) into the product

$$P(x_1, ..., x_n) = \prod_i P(x_i \mid pa_i). \tag{1.33}$$

For example, the DAG in Figure 1.2 induces the decomposition

$$P(x_1, x_2, x_3, x_4, x_5) = P(x_1)P(x_2|x_1)P(x_3|x_1)P(x_4|x_2, x_3)P(x_5|x_4). \tag{1.34}$$

The product decomposition in (1.33) is no longer order-specific since, given P and G, we can test whether P decomposes into the product given by (1.33) without making any reference to variable ordering. We therefore conclude that a necessary condition for a DAG G to be a Bayesian network of probability distribution P is for P to admit the product decomposition dictated by G, as given in (1.33).

Definition 1.2.2 (Markov Compatibility)
If a probability function P admits the factorization of (1.33) *relative to DAG G, we say that G* represents *P, that G and P are* compatible*, or that P is* Markov relative *to G.*[6]

[6]The latter expression seems to gain strength in recent literature (e.g. Spirtes et al. 1993; Lauritzen 1996). Pearl (1988b, p. 116) used "*G is an I-map* of P."

Ascertaining compatibility between DAGs and probabilities is important in statistical modeling primarily because compatibility is a necessary and sufficient condition for a DAG G to *explain* a body of empirical data represented by P, that is, to describe a stochastic process capable of *generating* P (e.g. Pearl, 1988b, pp. 210–23). If the value of each variable X_i is chosen at random with some probability $P_i(x_i|pa_i)$, based solely on the values pa_i previously chosen for PA_i, then the overall distribution P of the generated instances $x_1, x_2, \ldots, x_n$ will be Markov relative to G. Conversely, if P is Markov relative to G then there exists a set of probabilities $P_i(x_i|pa_i)$ according to which we can choose the value of each variable X_i such that the distribution of the generated instances $x_1, x_2, \ldots, x_n$ will be equal to P. (In fact, the correct choice of $P_i(x_i|pa_i)$ would be simply $P(x_i|pa_i)$.)

A convenient way of characterizing the set of distributions compatible with a DAG G is to list the set of (conditional) independencies that each such distribution must satisfy. These independencies can be read off the DAG by using a graphical criterion called *d-separation* (Pearl 1988b; the d denotes *directional*), which will play a major role in many discussions in this book.

1.2.3 The *d*-Separation Criterion

Consider three disjoint sets of variables, X, Y, and Z, which are represented as nodes in a directed acyclic graph G. To test whether X is independent of Y given Z in any distribution compatible with G, we need to test whether the nodes corresponding to variables Z "block" all paths from nodes in X to nodes in Y. By *path* we mean a sequence of consecutive edges (of any directionality) in the graph, and blocking is to be interpreted as stopping the flow of information (or of dependency) between the variables that are connected by such paths, as defined next.

Definition 1.2.3 (*d*-Separation)
*A path p is said to be d-*separated *(or* blocked*) by a set of nodes Z if and only if*

1. *p contains a chain $i \rightarrow m \rightarrow j$ or a fork $i \leftarrow m \rightarrow j$ such that the middle node m is in Z, or*

2. *p contains an inverted fork (or* collider*) $i \longrightarrow m \longleftarrow j$ such that the middle node m is not in Z and such that no descendant of m is in Z.*

A set Z is said to d-separate X from Y if and only if Z blocks every path from a node in X to a node in Y.

The intuition behind d-separation is simple and can best be recognized if we attribute causal meaning to the arrows in the graph. In causal chains $i \longrightarrow m \longrightarrow j$ and causal forks $i \longleftarrow m \longrightarrow j$, the two extreme variables are marginally dependent but become independent of each other (i.e., blocked) once we condition on (i.e., know the value of) the middle variable. Figuratively, conditioning on m appears to "block" the flow of information along the path, since learning about i has no effect on the probability of j, given m. Inverted forks $i \longrightarrow m \longleftarrow j$, representing two causes having a common effect, act the opposite way; if the two extreme variables are (marginally) independent, they will become dependent (i.e., connected through unblocked path) once we condition on the middle variable (i.e., the common effect) or any of its descendants. This can be confirmed in the context of Figure 1.2. Once we know the season, X_3 and X_2 are independent (assuming that sprinklers are set in advance, according to the season); whereas finding that the pavement is wet or slippery renders X_2 and X_3 dependent, because refuting one of these explanations increases the probability of the other.

In Figure 1.2, $X = \{X_2\}$ and $Y = \{X_3\}$ are d-separated by $Z = \{X_1\}$, because both paths connecting X_2 and X_3 are blocked by Z. The path $X_2 \longleftarrow X_1 \longrightarrow X_3$ is blocked because it is a fork in which the middle node X_1 is in Z, while the path $X_2 \longrightarrow X_4 \longleftarrow X_3$ is blocked because it is an inverted fork in which the middle node X_4 and all its descendants are outside Z. However, X and Y are not d-separated by the set $Z' = \{X_1, X_5\}$: the path $X_2 \longrightarrow X_4 \longleftarrow X_3$ (an inverted fork) is not blocked by Z', since X_5, a descendant of the middle node X_4, is in Z'. Metaphorically, learning the value of the consequence X_5 renders its causes X_2 and X_3 dependent, as if a pathway were opened along the arrows converging at X_4.

At first glance, readers might find it a bit odd that conditioning on a node not lying on a blocked path may unblock the path. However, this corresponds to a general pattern of causal relationships: observations on a common consequence of two independent causes tend to render those causes dependent, because information about one of the causes tends to make the other more or less likely, given that the consequence has occurred. This pattern is known as *selection bias* or *Berkson's paradox* in the statistical literature (Berkson 1946) and as the *explaining away effect* in artificial intelligence (Kim and Pearl 1983). For example, if the admission criteria to a certain graduate school call for either high grades as an undergraduate or special musical talents, then these two attributes will be found to be correlated (negatively) in the student population of that school, even if these attributes are uncorrelated in the population at large. Indeed, students with low grades are likely to be exceptionally gifted in music, which explains their admission to graduate school.

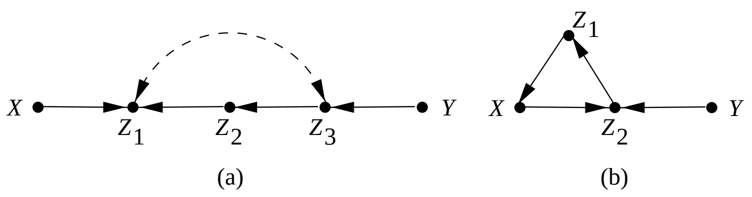

Figure 1.3: Graphs illustrating d-separation. In (a), X and Y are d-separated given Z_2 and d-connected given Z_1. In (b), X and Y cannot be d-separated by any set of nodes.

Figure 1.3 illustrates more elaborate examples of d-separation: example (a) contains a bidirected arc $Z_1 \dashleftarrow\dashrightarrow Z_3$ and (b) involves a directed cycle $X \longrightarrow Z_2 \longrightarrow Z_1 \longrightarrow X$. In Figure 1.3(a), the two paths between X and Y are blocked when none of $\{Z_1, Z_2, Z_3\}$ is measured. However, the path $X \longrightarrow Z_1 \dashleftarrow\dashrightarrow Z_3 \longleftarrow Y$ becomes unblocked when Z_1 is measured. This is so because Z_1 unblocks the "colliders" at both Z_1 and Z_3; the first because Z_1 is the collision node of the collider, the second because Z_1 is a descendant of the collision node Z_3 through the path $Z_1 \longleftarrow Z_2 \longleftarrow Z_3$. In Figure 1.3(b), X and Y cannot be d-separated by any set of nodes, including the empty set. If we condition on Z_2, we block the path $X \longleftarrow Z_1 \longleftarrow Z_2 \longleftarrow Y$ yet unblock the path $X \longrightarrow Z_2 \longleftarrow Y$. If we condition on Z_1, we again block the path $X \longleftarrow Z_1 \longleftarrow Z_2 \longleftarrow Y$ and unblock the path $X \longrightarrow Z_2 \longleftarrow Y$, because Z_1 is a descendant of the collision node Z_2.

The connection between d-separation and conditional independence is established through the following theorem due to Verma and Pearl (1988; see also Geiger et al. 1990).

Theorem 1.2.4 (Probabilistic Implications of *d*-Separation) *If sets X and Y are d-separated by Z in a DAG G, then X is independent of Y conditional on Z in every distribution compatible with G. Conversely, if X and Y are* not *d-separated by Z in a DAG G, then X and Y are dependent conditional on Z in at least one distribution compatible with G.*

The converse part of Theorem 1.2.4 is in fact much stronger—the absence of d-separation implies dependence in *almost all* distributions compatible with G. The reason is that a precise tuning of parameters is required to generate independency along an unblocked path in the diagram, and such tuning is unlikely to occur in practice (see Spirtes et al. 1993 and Sections 2.4 and 2.9.1).

In order to distinguish between the probabilistic notion of conditional independence $(X \perp\!\!\!\perp Y|Z)_P$ and the graphical notion of d-separation, for the latter we will use the notation $(X \perp\!\!\!\perp Y|Z)_G$. We can thereby express Theorem 1.2.4 more succinctly as follows.

Theorem 1.2.5 *For any three disjoint subsets of nodes (X, Y, Z) in a DAG G and for all probability functions P, we have:*

(i) $(X \perp\!\!\!\perp Y|Z)_G \Longrightarrow (X \perp\!\!\!\perp Y|Z)_P$ *whenever G and P are compatible, and*

(ii) *if $(X \perp\!\!\!\perp Y|Z)_P$ holds in all distributions compatible with G, it follows that $(X \perp\!\!\!\perp Y|Z)_G$.*

An alternative test for d-separation has been devised by Lauritzen et al. (1990), based on the notion of ancestral graphs. To test for $(X \perp\!\!\!\perp Y|Z)_G$, delete from G all nodes except those in $\{X, Y, Z\}$ and their ancestors, connect by an edge every pair of nodes that share a common child, and remove all arrows from the arcs. Then $(X \perp\!\!\!\perp Y|Z)_G$

holds if and only if Z intercepts all paths between X and Y in the resulting undirected graph.

Note that the ordering with which the graph was constructed does not enter into the d-separation criterion; it is only the topology of the resulting graph that determines the set of independencies that the probability P must satisfy. Indeed, the following theorem can be proven (Pearl 1988b, p. 120).

Theorem 1.2.6 (Ordered Markov Condition)
A necessary and sufficient condition for a probability distribution P to be Markov relative a DAG G is that, conditional on its parents in G, each variable be independent of all its predecessors in some ordering of the variables that agrees with the arrows of G.

A consequence of this theorem is an order-independent criterion for determining whether a given probability P is Markov relative to a given DAG G.

Theorem 1.2.7 (Parental Markov Condition)
A necessary and sufficient condition for a probability distribution P to be Markov relative a DAG G is that every variable be independent of all its nondescendants (in G), conditional on its parents.

This condition, which Kiiveri et al. (1984) and Lauritzen (1996) called the "local" Markov condition, is sometimes taken as the definition of Bayesian networks (Howard and Matheson 1981). In practice, however, the ordered Markov condition is easier to use.

Another important property that follows from d-separation is a criterion for determining whether two given DAGs are observationally equivalent—that is, whether every probability distribution that is compatible with one of the DAGs is also compatible with the other.

Theorem 1.2.8 (Observational Equivalence)
Two DAGs are observationally equivalent if and only if they have the same skeletons and the same sets of v-structures, that is, two converging arrows whose tails are not connected by an arrow (Verma and Pearl 1990).[7]

[7] An identical criterion was independently derived by Frydenberg (1990) in the context of chain graphs, where strict positivity is assumed.

Observational equivalence places a limit on our ability to infer directionality from probabilities alone. Two networks that are observationally equivalent cannot be distinguished without resorting to manipulative experimentation or temporal information. For example, reversing the direction of the arrow between X_1 and X_2 in Figure 1.2 would neither introduce nor destroy a v-structure. Therefore, this reversal yields an observationally equivalent network, and the directionality of the link $X_1 \longrightarrow X_2$ cannot be determined from probabilistic information. The arrows $X_2 \longrightarrow X_4$ and $X_4 \longrightarrow X_5$, however, are of different nature; there is no way of reversing their directionality without creating a new v-structure. Thus, we see that some probability functions P (such as the one responsible for the construction of the Bayesian network in Figure 1.2), when unaccompanied by temporal information, can constrain the directionality of some arrows in the graph. The precise meaning of such directionality constraints—and the possibility of using these constraints for inferring causal relationships from data—will be formalized in Chapter 2.

1.2.4 Inference with Bayesian Networks

Bayesian networks were developed in the early 1980s to facilitate the tasks of prediction and "abduction" in artificial intelligence (AI) systems. In these tasks, it is necessary to find a coherent interpretation of incoming observations that is consistent with both the observations and the prior information at hand. Mathematically, the task boils down to the computation of $P(y|x)$, where X is a set of observations and Y is a set of variables that are deemed important for prediction or diagnosis.

Given a joint distribution P, the computation of $P(y|x)$ is conceptually trivial and invokes straightforward application of Bayes's rule to yield

$$P(y|x) = \frac{\sum_{s} P(y, x, s)}{\sum_{y,s} P(y, x, s)}, \qquad (1.35)$$

where S stands for the set of all variables *excluding* X and Y. Because every Bayesian network defines a joint probability P (given by the product in (1.33)) it is clear that $P(y|x)$ can be computed from a DAG

G and the conditional probabilities $P(x_i|pa_i)$ defined on the families of G.

The challenge, however, lies in performing these computations efficiently and within the representation level provided by the network topology. The latter is important in systems that generate explanations for their reasoning processes. Although such inference techniques are not essential to our discussion of causality, we will nevertheless survey them briefly, for they demonstrate (i) the effectiveness of organizing probabilistic knowledge in the form of graphs and (ii) the feasibility of performing coherent probabilistic calculations (and approximations thereof) on such organization. Details can be found in the references cited.

The first algorithms proposed for probabilistic calculations in Bayesian networks used message-passing architecture and were limited to trees (Pearl 1982; Kim and Pearl 1983). With this technique, each variable is assigned a simple processor and permitted to pass messages asynchronously with its neighbors until equilibrium is achieved (in a finite number of steps). Methods have since been developed that extend this tree propagation (and some of its synchronous variants) to general networks. Among the most popular are Lauritzen and Spiegelhalter's (1988) method of join-tree propagation and the method of cut-set conditioning (Pearl 1988b, pp. 204–10; Jensen 1996). In the join-tree method, we decompose the network into clusters (e.g. cliques) that form tree structures and then treat the set variables in each cluster as a compound variable that is capable of passing messages to its neighbors (which are also compound variables). For example, the network of Figure 1.2 can be structured as a Markov-compatible chain of three clusters:

$$\{X_1, X_2, X_3\} \longrightarrow \{X_2, X_3, X_4\} \longrightarrow \{X_4, X_5\}.$$

In the cut-set conditioning method, a set of variables is instantiated (given specific values) such that the remaining network forms a tree. The propagation is then performed on that tree, and a new instantiation chosen, until all instantiations have been exhausted; the results are then averaged. In Figure 1.2, for example, if we instantiate X_1 as any specific value (say, $X_1 = \text{summer}$), then we break the pathway between X_2 and

X_3 and the remaining network becomes tree-structured. The main advantage of the cut-set conditioning method is that its storage-space requirement is minimal (linear in the size of the network), whereas that of the join-tree method might be exponential. Hybrid combinations of these two basic algorithms have also been proposed (Shachter et al. 1994; Dechter 1996) to allow flexible trade-off of storage versus time.

Whereas inference in general networks is "NP-hard" (Cooper 1990), the computational complexity for each of the methods cited here can be estimated prior to actual processing. When the estimates exceed reasonable bounds, an approximation method such as stochastic simulation (Pearl 1988b, pp. 210–23) can be used instead. This method exploits the topology of the network to perform Gibbs sampling on local subsets of variables, sequentially as well as concurrently.

Additional properties of DAGs and their applications to evidential reasoning in expert systems are discussed in Pearl (1988b), Lauritzen and Spiegelhalter (1988), Pearl (1993a), Spiegelhalter et al. (1993), Heckerman et al. (1995), and Shafer (1996b, 1997).