2

Introducing Bayesian Networks

2.1 Introduction

Having presented both theoretical and practical reasons for artificial intelligence to use probabilistic reasoning, we now introduce the key computer technology for dealing with probabilities in AI, namely **Bayesian networks.** Bayesian networks (BNs) are graphical models for reasoning under uncertainty, where the nodes represent variables (discrete or continuous) and arcs represent direct connections between them. These direct connections are often causal connections. In addition, BNs model the quantitative strength of the connections between variables, allowing probabilistic beliefs about them to be updated automatically as new information becomes available.

In this chapter we will describe how Bayesian networks are put together (the **syntax**) and how to interpret the information encoded in a network (the **semantics**). We will look at how to model a problem with a Bayesian network and the types of reasoning that can be performed.

2.2 Bayesian network basics

A **Bayesian network** is a graphical structure that allows us to represent and reason about an uncertain domain. The nodes in a Bayesian network represent a set of random variables from the domain. A set of directed **arcs** (or links) connects pairs of nodes, representing the direct dependencies between variables. Assuming discrete variables, the strength of the relationship between variables is quantified by conditional probability distributions associated with each node. The only constraint on the arcs allowed in a BN is that there must not be any directed cycles: you cannot return to a node simply by following directed arcs. Such networks are called directed acyclic graphs, or simply **dags**.

There are a number of steps that a **knowledge engineer*** must undertake when building a Bayesian network. At this stage we will present these steps as a sequence;

* Knowledge engineer in the jargon of AI means a practitioner applying AI technology.

however it is important to note that in the real-world the process is not so simple. In Chapter 9 we provide a fuller description of BN knowledge engineering.

Throughout the remainder of this section we will use the following simple medical diagnosis problem.

Example problem: Lung cancer. *A patient has been suffering from shortness of breath (called* dyspnoea*) and visits the doctor, worried that he has lung cancer. The doctor knows that other diseases, such as tuberculosis and bronchitis, are possible causes, as well as lung cancer. She also knows that other relevant information includes whether or not the patient is a smoker (increasing the chances of cancer and bronchitis) and what sort of air pollution he has been exposed to. A positive X-ray would indicate either TB or lung cancer*[†].

2.2.1 Nodes and values

First, the knowledge engineer must identify the variables of interest. This involves answering the question: what are the nodes to represent and what values can they take? For now we will consider only nodes that take discrete values. The values should be both **mutually exclusive** and **exhaustive**, which means that the variable must take on exactly one of these values at a time. Common types of discrete nodes include:

- Boolean nodes, which represent propositions, taking the binary values true (T) and false (F). In a medical diagnosis domain, the node *Cancer* would represent the proposition that a patient has cancer.
- Ordered values. For example, a node *Pollution* might represent a patient's pollution exposure and take the values $\{$*low, medium, high*$\}$.
- Integral values. For example, a node called *Age* might represent a patient's age and have possible values from 1 to 120.

Even at this early stage, modeling choices are being made. For example, an alternative to representing a patient's exact age might be to clump patients into different age groups, such as $\{$*baby, child, adolescent, young, middleaged, old*$\}$. The trick is to choose values that represent the domain efficiently, but with enough detail to perform the reasoning required. More on this later!

For our example, we will begin with the restricted set of nodes and values shown in Table 2.1. These choices already limit what can be represented in the network. For instance, there is no representation of other diseases, such as TB or bronchitis, so the system will not be able to provide the probability of the patient having them. Another limitation is a lack of differentiation, for example between a heavy or a light smoker, and again the model assumes at least some exposure to pollution. Note that all these nodes have only two values, which keeps the model simple, but in general there is no limit to the number of discrete values.

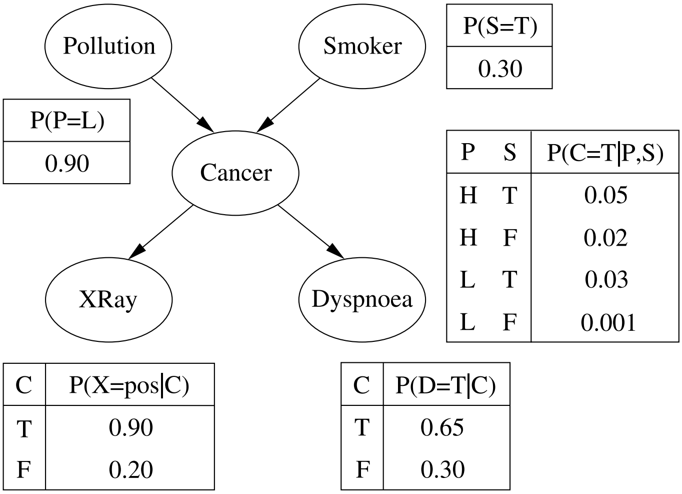
[†] This is a modified version of the so-called "Asia" problem [169], given in §2.5.3.

TABLE 2.1
Preliminary choices of nodes and values for the lung cancer example

Node name	Type	Values
Pollution	Binary	{*low, high*}
Smoker	Boolean	{*T, F*}
Cancer	Boolean	{*T, F*}
Dyspnoea	Boolean	{*T, F*}
X-ray	Binary	{*pos, neg*}

2.2.2 Structure

The structure, or topology, of the network should capture qualitative relationships between variables. In particular, two nodes should be connected directly if one affects or causes the other, with the arc indicating the direction of the effect. So, in our medical diagnosis example, we might ask what factors affect a patient's chance of having cancer? If the answer is "Pollution and smoking," then we should add arcs from *Pollution* and *Smoker* to *Cancer*. Similarly, having cancer will affect the patient's breathing and the chances of having a positive X-ray result. So we add arcs from *Cancer* to *Dyspnoea*. The resultant structure is shown in Figure 2.1. It is important to note that this is just one possible structure for the problem; we look at alternative network structures in §2.4.3.



P(S=T)
0.30

P(P=L)
0.90

P	S	P(C=T|P,S)
H	T	0.05
H	F	0.02
L	T	0.03
L	F	0.001

C	P(X=pos|C)
T	0.90
F	0.20

C	P(D=T|C)
T	0.65
F	0.30

FIGURE 2.1
A BN for the lung cancer problem.

Structure terminology and layout

In talking about network structure it is useful to employ a family metaphor: a node is a **parent** of a **child**, if there is an arc from the former to the latter. Extending the metaphor, if there is a directed chain of nodes, one node is an **ancestor** of another if it appears earlier in the chain, whereas a node is a **descendant** of another node if it comes later in the chain. In our example, the *Cancer* node has two parents, *Pollution* and *Smoker*, while *Smoker* is an ancestor of both *X-ray* and *Dyspnoea*. Similarly, *X-ray* is a child of *Cancer* and descendant of *Smoker* and *Pollution*. The set of parent nodes of a node X is given by *Parents(X)*.

Another useful concept is that of the **Markov blanket** of a node, which consists of the node's parents, its children, and its children's parents. Other terminology commonly used comes from the "tree" analogy (even though Bayesian networks in general are graphs rather than trees): any node without parents is called a **root** node, while any node without children is called a **leaf** node. Any other node (non-leaf and non-root) is called an **intermediate node**. Given a causal understanding of the BN structure, this means that root nodes represent original causes, while leaf nodes represent final effects. In our cancer example, the causes *Pollution* and *Smoker* are root nodes, while the effects *X-ray* and *Dyspnoea* are leaf nodes.

By convention, for easier visual examination of BN structure, networks are usually laid out so that the arcs generally point from top to bottom. This means that the BN "tree" is usually depicted upside down, with roots at the top and leaves at the bottom[‡]!

2.2.3 Conditional probabilities

Once the topology of the BN is specified, the next step is to quantify the relationships between connected nodes – this is done by specifying a conditional probability distribution for each node. As we are only considering discrete variables at this stage, this takes the form of a conditional probability *table* (CPT).

First, for each node we need to look at all the possible combinations of values of those parent nodes. Each such combination is called an **instantiation** of the parent set. For each distinct instantiation of parent node values, we need to specify the probability that the child will take each of its values.

For example, consider the *Cancer* node of Figure 2.1. Its parents are *Pollution* and *Smoking* and take the possible joint values $\{< H,T >, < H,F >, < L,T >, < L,F >\}$. The conditional probability table specifies in order the probability of cancer for each of these cases to be: $< 0.05, 0.02, 0.03, 0.001 >$. Since these *are* probabilities, and must sum to one over all possible states of the *Cancer* variable, the probability of no cancer is already implicitly given as one minus the above probabilities in each case; i.e., the probability of no cancer in the four possible parent instantiations is $< 0.95, 0.98, 0.97, 0.999 >$.

[‡] Oddly, this is the antipodean standard in computer science; we'll let you decide what that may mean about computer scientists!

Root nodes also have an associated CPT, although it is degenerate, containing only one row representing its prior probabilities. In our example, the prior for a patient being a smoker is given as 0.3, indicating that 30% of the population that the doctor sees are smokers, while 90% of the population are exposed to only low levels of pollution.

Clearly, if a node has many parents or if the parents can take a large number of values, the CPT can get very large! The size of the CPT is, in fact, exponential in the number of parents. Thus, for Boolean networks a variable with n parents requires a CPT with 2^{n+1} probabilities.

2.2.4 The Markov property

In general, modeling with Bayesian networks requires the assumption of the **Markov property**: there are no direct dependencies in the system being modeled which are not already explicitly shown via arcs. In our *Cancer* case, for example, there is no way for smoking to influence dyspnoea except by way of causing cancer (or not) — there is no hidden "backdoor" from smoking to dyspnoea. Bayesian networks which have the Markov property are also called **Independence-maps** (or, **I-maps** for short), since every independence suggested by the lack of an arc is real in the system.

Whereas the independencies suggested by a lack of arcs are generally required to exist in the system being modeled, it is not generally required that the arcs in a BN correspond to real dependencies in the system. The CPTs may be parameterized in such a way as to nullify any dependence. Thus, for example, every fully-connected Bayesian network can represent, perhaps in a wasteful fashion, any joint probability distribution over the variables being modeled. Of course, we shall prefer **minimal models** and, in particular, **minimal I-maps**, which are I-maps such that the deletion of any arc violates I-mapness by implying a non-existent independence in the system.

If, in fact, every arc in a BN happens to correspond to a direct dependence in the system, then the BN is said to be a **Dependence-map** (or, **D-map** for short). A BN which is both an I-map and a D-map is said to be a **perfect map**.

2.3 Reasoning with Bayesian networks

Now that we know how a domain and its uncertainty may be represented in a Bayesian network, we will look at how to use the Bayesian network to reason about the domain. In particular, when we observe the value of some variable, we would like to **condition** upon the new information. The process of conditioning (also called **probability propagation** or **inference** or **belief updating**) is performed via a "flow of information" through the network. Note that this information flow is *not* limited to the directions of the arcs. In our probabilistic system, this becomes the task of computing the posterior probability distribution for a set of **query** nodes, given values for some **evidence** (or **observation**) nodes.

2.3.1 Types of reasoning

Bayesian networks provide full representations of probability distributions over their variables. That implies that they can be conditioned upon any subset of their variables, supporting any direction of reasoning.

For example, one can perform **diagnostic reasoning**, i.e., reasoning from symptoms to cause, such as when a doctor observes *Dyspnoea* and then updates his belief about *Cancer* and whether the patient is a *Smoker*. Note that this reasoning occurs in the *opposite* direction to the network arcs.

Or again, one can perform **predictive reasoning**, reasoning from new information about causes to new beliefs about effects, following the directions of the network arcs. For example, the patient may tell his physician that he is a smoker; even before any symptoms have been assessed, the physician knows this will increase the chances of the patient having cancer. It will also change the physician's expectations that the patient will exhibit other symptoms, such as shortness of breath or having a positive X-ray result.

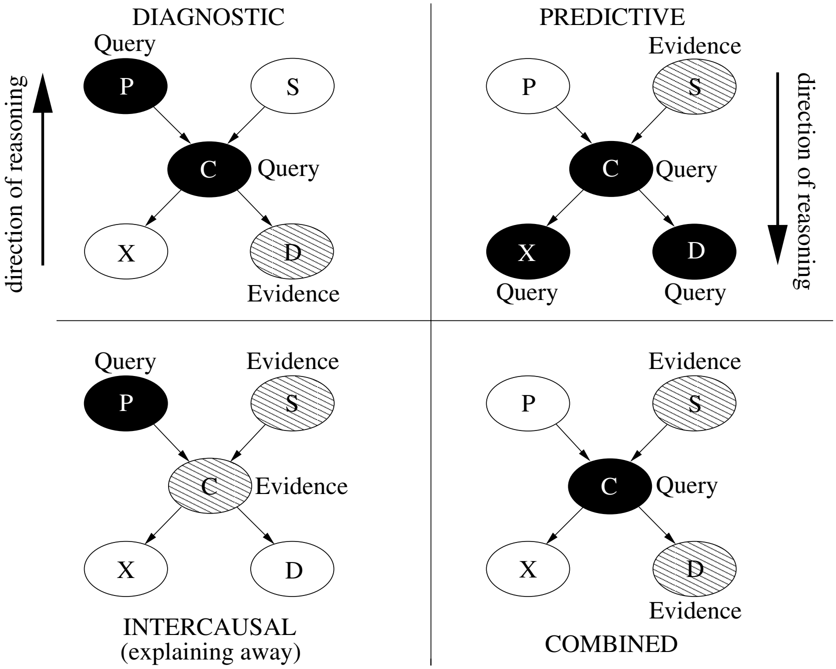


FIGURE 2.2
Types of reasoning.

A further form of reasoning involves reasoning about the mutual causes of a common effect; this has been called **intercausal reasoning**. A particular type called **explaining away** is of some interest. Suppose that there are exactly two possible causes of a particular effect, represented by a **v-structure** in the BN. This situation occurs in our model of Figure 2.1 with the causes *Smoker* and *Pollution* which have a common effect, *Cancer* (of course, reality is more complex than our example!). Initially, according to the model, these two causes are independent of each other; that is, a patient smoking (or not) does not change the probability of the patient being subject to pollution. Suppose, however, that we learn that Mr. Smith has cancer. This will raise our probability for both possible causes of cancer, increasing the chances both that he is a smoker and that he has been exposed to pollution. Suppose then that we discover that he is a smoker. This new information explains the observed cancer, which in turn *lowers* the probability that he has been exposed to high levels of pollution. So, even though the two causes are initially independent, with knowledge of the effect the presence of one explanatory cause renders an alternative cause less likely. In other words, the alternative cause has been *explained away*.

Since any nodes may be query nodes and any may be evidence nodes, sometimes the reasoning does not fit neatly into one of the types described above. Indeed, we can combine the above types of reasoning in any way. Figure 2.2 shows the different varieties of reasoning using the Cancer BN. Note that the last combination shows the simultaneous use of diagnostic and predictive reasoning.

2.3.2 Types of evidence

So Bayesian networks can be used for calculating new beliefs when new information – which we have been calling **evidence** – is available. In our examples to date, we have considered evidence as a definite finding that a node *X* has a particular value, *x*, which we write as $X = x$. This is sometimes referred to as **specific evidence**. For example, suppose we discover the patient is a smoker, then *Smoker=T*, which is specific evidence.

However, sometimes evidence is available that is not so definite. The evidence might be that a node Y has the value y_1 *or* y_2 (implying that all other values are impossible). Or the evidence might be that Y is *not* in state y_1 (but may take any of its other values); this is sometimes called a **negative evidence**.

In fact, the new information might simply be any new probability distribution over Y. Suppose, for example, that the radiologist who has taken and analyzed the X-ray in our cancer example is uncertain. He thinks that the X-ray looks positive, but is only 80% sure. Such information can be incorporated equivalently to Jeffrey conditionalization of §1.5.1, in which case it would correspond to adopting a new posterior distribution for the node in question. In Bayesian networks this is also known as **virtual evidence**. Since it is handled via likelihood information, it is also known as **likelihood evidence**. We defer further discussion of virtual evidence until Chapter 3, where we can explain it through the effect on belief updating.

2.3.3 Reasoning with numbers

Now that we have described qualitatively the types of reasoning that are possible using BNs, and types of evidence, let's look at the actual numbers. Even before we obtain any evidence, we can compute a prior belief for the value of each node; this is the node's prior probability distribution. We will use the notation Bel(X) for the posterior probability distribution over a variable X, to distinguish it from the prior and conditional probability distributions (i.e., $P(X)$, $P(X|Y)$).

The exact numbers for the updated beliefs for each of the reasoning cases described above are given in Table 2.2. The first set are for the priors and conditional probabilities originally specified in Figure 2.1. The second set of numbers shows what happens if the smoking rate in the population increases from 30% to 50%, as represented by a change in the prior for the *Smoker* node. Note that, since the two cases differ only in the prior probability of smoking ($P(S = T) = 0.3$ versus $P(S = T) = 0.5$), when the evidence itself is about the patient being a smoker, then the prior becomes irrelevant and both networks give the same numbers.

TABLE 2.2
Updated beliefs given new information with smoking rate 0.3 (top set) and 0.5 (bottom set)

Node	No	Reasoning Case				
P(S)=0.3	Evidence	Diagnostic	Predictive	Intercausal		Combined
		D=T	S=T	C=T	C=T	D=T
					S=T	S=T
Bel(P=high)	0.100	0.102	0.100	0.249	0.156	0.102
Bel(S=T)	0.300	0.307	1	0.825	1	1
Bel(C=T)	0.011	0.025	0.032	1	1	0.067
Bel(X=pos)	0.208	0.217	0.222	0.900	0.900	0.247
Bel(D=T)	0.304	1	0.311	0.650	0.650	1
P(S)=0.5						
Bel(P=high)	0.100	0.102	0.100	0.201	0.156	0.102
Bel(S=T)	0.500	0.508	1	0.917	1	1
Bel(C=T)	0.174	0.037	0.032	1	1	0.067
Bel(X=pos)	0.212	0.226	0.311	0.900	0.900	0.247
Bel(D=T)	0.306	1	0.222	0.650	0.650	1

Belief updating can be done using a number of exact and approximate inference algorithms. We give details of these algorithms in Chapter 3, with particular emphasis on how choosing different algorithms can affect the efficiency of both the knowledge engineering process and the automated reasoning in the deployed system. However, most existing BN software packages use essentially the same algorithm and it is quite possible to build and use BNs without knowing the details of the belief updating algorithms.

2.4 Understanding Bayesian networks

We now consider how to interpret the information encoded in a BN — the probabilistic **semantics** of Bayesian networks.

2.4.1 Representing the joint probability distribution

Most commonly, BNs are considered to be representations of joint probability distributions. There is a fundamental assumption that there is a useful underlying structure to the problem being modeled that can be captured with a BN, i.e., that not every node is connected to every other node. If such domain structure exists, a BN gives a more compact representation than simply describing the probability of every joint instantiation of all variables. **Sparse** Bayesian networks (those with relatively few arcs, which means few parents for each node) represent probability distributions in a computationally tractable way.

Consider a BN containing the n nodes, X_1 to X_n, taken in that order. A particular value in the joint distribution is represented by $P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$, or more compactly, $P(x_1, x_2, \ldots, x_n)$. The **chain rule** of probability theory allows us to factorize joint probabilities so:

$$\begin{aligned} P(x_1, x_2, \ldots, x_n) &= P(x_1) \times P(x_2|x_1) \ldots, \times P(x_n|x_1, \ldots, x_{n-1}) \\ &= \prod_i P(x_i|x_1, \ldots, x_{i-1}) \end{aligned} \tag{2.1}$$

Recalling from §2.2.4 that the structure of a BN implies that the value of a particular node is conditional *only* on the values of its parent nodes, this reduces to

$$P(x_1, x_2, \ldots, x_n) = \prod_i P(x_i|Parents(X_i))$$

provided $Parents(X_i) \subseteq \{x_1, \ldots, x_{i-1}\}$. For example, by examining Figure 2.1, we can simplify its joint probability expressions. E.g.,

$$\begin{aligned} &P(X = pos \wedge D = T \wedge C = T \wedge P = low \wedge S = F) \\ &\quad = P(X = pos|D = T, C = T, P = low, S = F) \\ &\quad\quad \times P(D = T|C = T, P = low, S = F) \\ &\quad\quad \times P(C = T|P = low, S = F)P(P = low|S = F)P(S = F) \\ &\quad = P(X = pos|C = T)P(D = T|C = T)P(C = T|P = low, S = F) \\ &\quad\quad \times P(P = low)P(S = F) \end{aligned}$$

2.4.2 Pearl's network construction algorithm

The condition that $Parents(X_i) \subseteq \{x_1, \ldots, x_{i-1}\}$ allows us to construct a network from a given ordering of nodes using Pearl's network construction algorithm [217, section 3.3]. Furthermore, the resultant network will be a unique minimal I-map, assuming the probability distribution is positive. The construction algorithm (Algorithm 2.1) simply processes each node in order, adding it to the existing network and adding arcs from a minimal set of parents such that the parent set renders the current node conditionally independent of every other node preceding it.

ALGORITHM 2.1
Pearl's Network Construction Algorithm

```
1. Choose the set of relevant variables $\{X_i\}$ that describe the domain.
2. Choose an ordering for the variables, $< X_1, \ldots, X_n >$.
3. While there are variables left:
    (a) Add the next variable $X_i$ to the network.
    (b) Add arcs to the $X_i$ node from some minimal set of nodes already in the net,
        $Parents(X_i)$, such that the following conditional independence property
        is satisfied:
            $$P(X_i|X'_1, \ldots, X'_m) = P(X_i|Parents(X_i))$$
        where $X'_1, \ldots, X'_m$ are all the variables preceding $X_i$ that are not in
        $Parents(X_i)$.
    (c) Define the CPT for $X_i$.
```

2.4.3 Compactness and node ordering

Using this construction algorithm, it is clear that a different node order may result in a different network structure, with both nevertheless representing the same joint probability distribution.

In our example, several different orderings will give the original network structure: *Pollution* and *Smoker* must be added first, but in either order, then *Cancer*, and then *Dyspnoea* and *X-ray*, again in either order.

On the other hand, if we add the symptoms first, we will get a markedly different network. Consider the order $< D, X, C, P, S >$. D is now the new root node. When adding *X*, we must consider "Is *X-ray* independent of *Dyspnoea*?" Since they have a common cause in *Cancer*, they will be dependent: learning the presence of one symptom, for example, raises the probability of the other being present. Hence, we have to add an arc from D to X. When adding *Cancer*, we note that *Cancer* is directly dependent upon both *Dyspnoea* and *X-ray*, so we must add arcs from both. For *Pollution*, an arc is required from C to P to carry the direct dependency. When the final node, *Smoker*, is added, not only is an arc required from C to S, but another from P to S. In our story S and P are independent, but in the new network, without this final arc, P and S are made dependent by having a common cause, so that effect must be counterbalanced by an additional arc. The result is two additional arcs and three new probability values associated with them, as shown in Figure 2.3(a). Given the order $< D, X, P, S, C >$, we get Figure 2.3(b), which is fully connected and requires as many CPT entries as a brute force specification of the full joint distribution! In such cases, the use of Bayesian networks offers no representational, or computational, advantage.

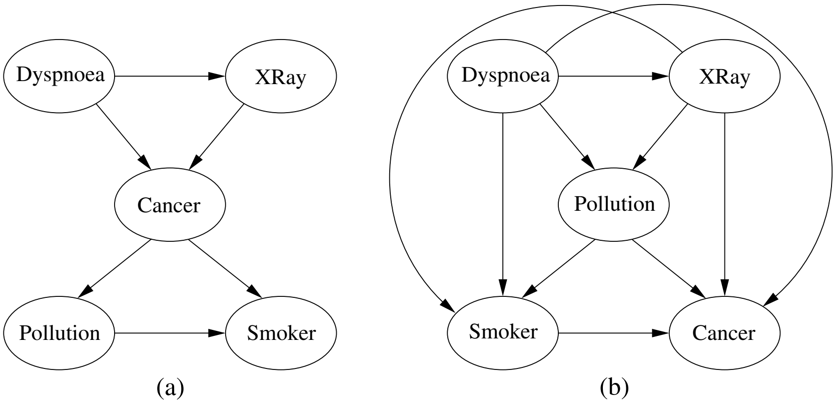


FIGURE 2.3

Alternative structures obtained using Pearl's network construction algorithm with orderings: (a) $< D, X, C, P, S >$; (b) $< D, X, P, S, C >$.

It is desirable to build the most compact BN possible, for three reasons. First, the more compact the model, the more tractable it is. It will have fewer probability values requiring specification; it will occupy less computer memory; probability updates will be more computationally efficient. Second, overly dense networks fail to represent independencies explicitly. And third, overly dense networks fail to represent the *causal* dependencies in the domain. We will discuss these last two points just below.

We can see from the examples that the compactness of the BN depends on getting the node ordering "right." The optimal order is to add the root causes first, then the variable(s) they influence directly, and continue until leaves are reached. To understand *why*, we need to consider the relation between probabilistic and causal dependence.

2.4.4 Conditional independence

Bayesian networks which satisfy the Markov property (and so are I-maps) explicitly express conditional independencies in probability distributions. The relation between conditional independence and Bayesian network structure is important for understanding how BNs work.

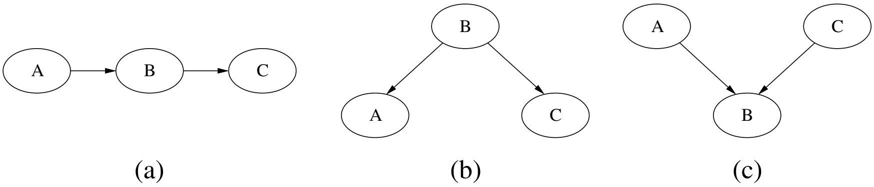
2.4.4.1 Causal chains

Consider a causal chain of three nodes, where A causes B which in turn causes C, as shown in Figure 2.4(a). In our medical diagnosis example, one such causal chain is "smoking causes cancer which causes dyspnoea." Causal chains give rise to conditional independence, such as for Figure 2.4(a):

$$P(C|A \wedge B) = P(C|B)$$

This means that the probability of C, given B, is exactly the same as the probability of C, given both B *and* A. Knowing that A has occurred doesn't make any difference to our beliefs about C *if we already know that B has occurred.* We also write this conditional independence as: $A \perp\!\!\!\perp C|B$.

In Figure 2.1(a), the probability that someone has dyspnoea depends directly only on whether they have cancer. If we don't know whether some woman has cancer, but we do find out she is a smoker, that would increase our belief both that she has cancer and that she suffers from shortness of breath. However, if we already *knew* she had cancer, then her smoking wouldn't make any difference to the probability of dyspnoea. That is, dyspnoea is conditionally independent of being a smoker *given* the patient has cancer.



(a) (b) (c)

FIGURE 2.4
(a) Causal chain; (b) common cause; (c) common effect.

2.4.4.2 Common causes

Two variables A and C having a common cause B is represented in Figure 2.4(b). In our example, cancer is a common cause of the two symptoms, a positive X-ray result and dyspnoea. Common causes (or common ancestors) give rise to the same conditional independence structure as chains:

$$P(C|A \wedge B) = P(C|B) \equiv A \perp\!\!\!\perp C|B$$

If there is no evidence or information about cancer, then learning that one symptom is present will increase the chances of cancer which in turn will increase the probability of the other symptom. However, if we already know about cancer, then an additional positive X-ray won't tell us anything new about the chances of dyspnoea.

2.4.4.3 Common effects

A common effect is represented by a network v-structure, as in Figure 2.4(c). This represents the situation where a node (the effect) has two causes. Common effects (or their descendants) produce the exact opposite conditional independence structure to that of chains and common causes. That is, the parents are marginally independent ($A \perp\!\!\!\perp C$), but become dependent given information about the common effect (i.e., they are **conditionally dependent**):

$$P(A|C \wedge B) \neq P(A|C) \equiv \neg(A \perp\!\!\!\perp C|B)$$

Thus, if we observe the effect (e.g., cancer), and then, say, we find out that one of the causes is absent (e.g., the patient does not smoke), this *raises* the probability of the other cause (e.g., that he lives in a polluted area) — which is just the inverse of explaining away.

Compactness again

So we can now see *why* building networks with an order violating causal order can, and generally will, lead to additional complexity in the form of extra arcs. Consider just the subnetwork $\{$ *Pollution, Smoker, Cancer* $\}$ of Figure 2.1. If we build the subnetwork in that order we get the simple v-structure *Pollution* $\rightarrow$ *Smoker* $\leftarrow$ *Cancer*. However, if we build it in the order $<$ *Cancer, Pollution, Smoker* $>$, we will first get *Cancer* $\rightarrow$ *Pollution*, because they are dependent. When we add *Smoker*, it will be dependent upon *Cancer*, because in reality there is a direct dependency there. But we shall also have to add a spurious arc to *Pollution*, because otherwise *Cancer* will act as a common cause, inducing a spurious dependency between *Smoker* and *Pollution*; the extra arc is necessary to reestablish marginal independence between the two.

2.4.5 d-separation

We have seen how Bayesian networks represent conditional independencies and how these independencies affect belief change during updating. The conditional independence in $A \perp\!\!\!\perp C|B$ means that knowing the value of B **blocks** information about C being relevant to A, and vice versa. Or, in the case of Figure 2.4(c), *lack* of information about B blocks the relevance of C to A, whereas learning about B **activates** the relation between C and A.

These concepts apply not only between pairs of nodes, but also between sets of nodes. More generally, given the Markov property, it is possible to determine whether a set of nodes $\mathbf{X}$ is independent of another set $\mathbf{Y}$, given a set of evidence nodes $\mathbf{E}$. To do this, we introduce the notion of **d-separation** (from **direction-dependent separation**).

Definition 2.1 Path (Undirected Path) *A* **path** *between two sets of nodes* **X** *and* **Y** *is any sequence of nodes between a member of* **X** *and a member of* **Y** *such that every adjacent pair of nodes is connected by an arc (regardless of direction) and no node appears in the sequence twice.*

Definition 2.2 Blocked path *A path is* **blocked**, *given a set of nodes* **E**, *if there is a node Z on the path for which at least one of three conditions holds:*

1. *Z is in* **E** *and Z has one arc on the path leading in and one arc out (chain).*
2. *Z is in* **E** *and Z has both path arcs leading out (common cause).*
3. *Neither Z nor any descendant of Z is in* **E**, *and both path arcs lead in to Z (common effect).*

Definition 2.3 d-separation *A set of nodes* **E d-separates** *two other sets of nodes* **X** *and* **Y** *if every path from a node in* **X** *to a node in* **Y** *is* **blocked** *given* **E**.

If **X** and **Y** are **d-separated** by **E**, then **X** and **Y** are **conditionally independent** given **E** (given the Markov property). Examples of these three blocking situations are shown in Figure 2.5. Note that we have simplified by using single nodes rather than sets of nodes; also note that the evidence nodes **E** are shaded.

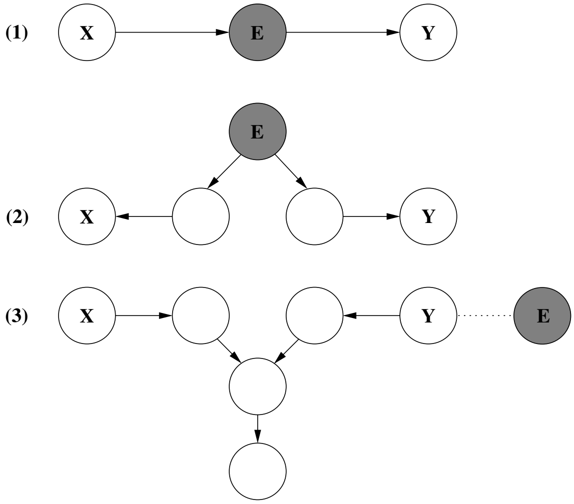


FIGURE 2.5

Examples of the three types of situations in which the path from X to Y can be blocked, given evidence E. In each case, X and Y are **d-separated** by E.

Consider d-separation in our cancer diagnosis example of Figure 2.1. Suppose an observation of the Cancer node is our evidence. Then:

1. P is d-separated from X and D. Likewise, S is d-separated from X and D (blocking condition 1).

2. While X is d-separated from D (condition 2).
3. However, if C had not been observed (and also not X or D), then S would have been d-separated from P (condition 3).

2.5 More examples

In this section we present further simple examples of BN modeling from the literature. We encourage the reader to work through these examples using BN software (see §B.4).

2.5.1 Earthquake

Example statement: *You have a new burglar alarm installed. It reliably detects burglary, but also responds to minor earthquakes. Two neighbors, John and Mary, promise to call the police when they hear the alarm. John always calls when he hears the alarm, but sometimes confuses the alarm with the phone ringing and calls then also. On the other hand, Mary likes loud music and sometimes doesn't hear the alarm. Given evidence about who has and hasn't called, you'd like to estimate the probability of a burglary (from [217]).*

A BN representation of this example is shown in Figure 2.6. All the nodes in this BN are Boolean, representing the true/false alternatives for the corresponding propositions. This BN models the assumptions that John and Mary do not perceive a burglary directly and they do not feel minor earthquakes. There is no explicit representation of loud music preventing Mary from hearing the alarm, nor of John's confusion of alarms and telephones; this information is summarized in the probabilities in the arcs from *Alarm* to *JohnCalls* and *MaryCalls*.

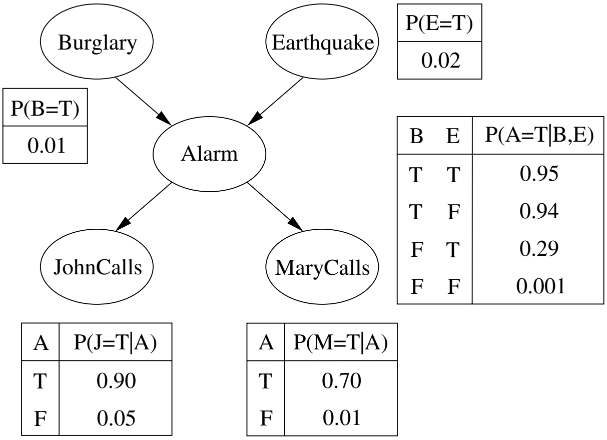
2.5.2 Metastatic cancer

Example statement: *Metastatic cancer is a possible cause of brain tumors and is also an explanation for increased total serum calcium. In turn, either of these could explain a patient falling into a coma. Severe headache is also associated with brain tumors. (This example has a long history in the literature [51, 217, 262].)*

A BN representation of this metastatic cancer example is shown in Figure 2.7. All the nodes are Booleans. Note that this is a *graph*, not a tree, in that there is more than one path between the two nodes M and C (via S and B).

2.5.3 Asia

Example Statement: *Suppose that we wanted to expand our original medical diagnosis example to represent explicitly some other possible causes of shortness of*



P(E=T)
0.02

P(B=T)
0.01

B	E	P(A=T|B,E)
T	T	0.95
T	F	0.94
F	T	0.29
F	F	0.001

A	P(J=T|A)
T	0.90
F	0.05

A	P(M=T|A)
T	0.70
F	0.01

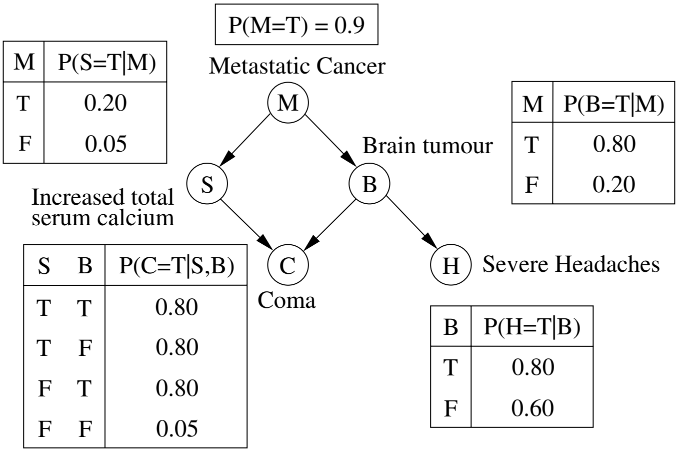
FIGURE 2.6
Pearl's Earthquake BN.

breath, namely tuberculosis and bronchitis. Suppose also that whether the patient has recently visited Asia is also relevant, since TB is more prevalent there.

Two alternative BN structures for the so-called Asia example are shown in Figure 2.8. In both networks all the nodes are Boolean. The left-hand network is based on the Asia network of [169]. Note the slightly odd intermediate node *TBorC*, indicating that the patient has either tuberculosis or bronchitis. This node is not strictly necessary; however it reduces the number of arcs elsewhere, by summarizing the similarities between TB and lung cancer in terms of their relationship to positive X-ray results and dyspnoea. Without this node, as can be seen on the right, there are two parents for *X-ray* and three for *Dyspnoea*, with the same probabilities repeated in different parts of the CPT. The use of such an intermediate node is an example of "divorcing," a model structuring method described in §9.3.2.

2.6 Summary

Bayes' theorem allows us to update the probabilities of variables whose state has not been observed given some set of new observations. Bayesian networks automate this process, allowing reasoning to proceed in any direction across the network of variables. They do this by combining qualitative information about direct dependencies (perhaps causal relations) in arcs and quantitative information about the strengths of those dependencies in conditional probability distributions. Computational speed gains in updating accrue when the network is sparse, allowing d-separation to take advantage of conditional independencies in the domain (so long as the Markov prop-



M	P(S=T|M)
T	0.20
F	0.05

M	P(B=T|M)
T	0.80
F	0.20

S	B	P(C=T|S,B)
T	T	0.80
T	F	0.80
F	T	0.80
F	F	0.05

B	P(H=T|B)
T	0.80
F	0.60

FIGURE 2.7
Metastatic cancer BN.

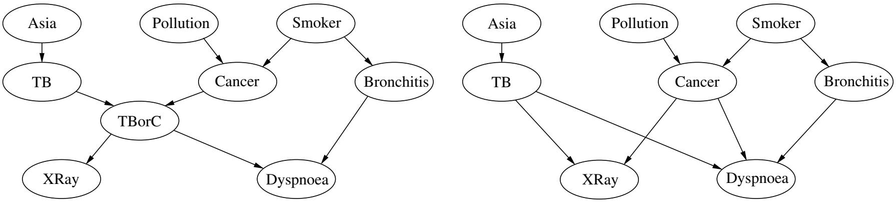


FIGURE 2.8
Alternative BNs for the "Asia" example.

erty holds). Given a known set of conditional independencies, Pearl's network construction algorithm guarantees the development of a minimal network, without redundant arcs. In the next chapter, we turn to specifics about the algorithms used to update Bayesian networks.

2.7 Bibliographic notes

The text that marked the new era of Bayesian methods in artificial intelligence is Judea Pearl's *Probabilistic Reasoning in Intelligent Systems* [217]. This text played no small part in attracting the authors to the field, amongst many others. Richard Neapolitan's *Probabilistic Reasoning in Expert Systems* [199] complements Pearl's book nicely, and it lays out the algorithms underlying the technology particularly well. A more current introduction is Finn Jensen's *Bayesian Networks and Decision*

A Quick Guide to Using Netica

Installation: Web Site `www.norsys.com`. Download Netica, which is available for MS Windows (95 / 98 / NT4 / 2000 / XP), and PowerPC MacIntosh. This gives you `Netica_Win.exe`, a self-extracting zip archive. Double-clicking will start the extraction process.

Network Files: BNs are stored in `.dne` files, with icon ▬. Netica comes with a folder of example networks, plus a folder of tutorial examples. To open an existing network:

- Select 📂; or
- Select `File`→`Open` menu option; or
- Double-click on the BN `.dne` file.

Compilation: Once a Netica BN has been opened, before you can see the initial beliefs or add evidence, you must first compile it:

- Click on ⚡; or
- Select `Network`→`Compile` menu option.

Once the network is compiled, numbers and bars will appear for each node state. Note that Netica beliefs are given out of 100, not as direct probabilities (i.e., not numbers between 0 and 1).

Evidence: To add evidence:

- Left-click on the node state name; or
- Right-click on node and select particular state name.

To remove evidence:

- Right-click on node and select `unknown`; or
- Select ☒ ; or
- Select `Network`→`Remove findings` menu option.

There is an option (`Network`→`Automatic Update`) to automatically re-compile and update beliefs when new evidence is set.

Editing/Creating a BN: Double-clicking on a node will bring up a window showing node features.

- Add a node by selecting either ⬭, or `Modify`→`Add nature node`, then "drag-and-drop" with the mouse.
- Add an arc by selecting either ↘, or `Modify`→`Add link`, then left-click first on the parent node, then the child node.
- Double-click on node, then click on the `Table` button to bring up the CPT. Entries can be added or changed by clicking on the particular cells.

Saving a BN: Select 💾 or the `File`→`Save` menu option. Note that the Netica Demonstration version only allows you to save networks with up to 15 nodes. For larger networks, you need to buy a license.

FIGURE 2.9
A quick guide to using Netica.

Graphs [128]. Its level and treatment is similar to ours; however, it does not go far with the machine learning and knowledge engineering issues we treat later. Two more technical discussions can be found in Cowell et al.'s *Probabilistic Networks and Expert Systems* [61] and Richard Neapolitan's *Learning Bayesian Networks* [200].

2.8 Problems

Modeling

These modeling exercises should be done using a BN software package (see **A Quick Guide to Using Netica** in Figure 2.9, or **A Quick Guide to Using Hugin** in Figure 3.14, and also Appendix B).

Also note that various information, including Bayesian network examples in Netica's `.dne` format, can be found at the book Web site:

`http://www.csse.monash.edu.au/bai`

Problem 1

Construct a network in which explaining away operates, for example, incorporating multiple diseases sharing a symptom. Operate and demonstrate the effect of explaining away. *Must* one cause explain away the other? Or, can the network be parameterized so that this doesn't happen?

Problem 2

"Fred's LISP dilemma." *Fred is debugging a LISP program. He just typed an expression to the LISP interpreter and now it will not respond to any further typing. He can't see the visual prompt that usually indicates the interpreter is waiting for further input. As far as Fred knows, there are* only *two situations that could cause the LISP interpreter to stop running: (1) there are problems with the computer hardware; (2) there is a bug in Fred's code. Fred is also running an editor in which he is writing and editing his LISP code; if the hardware is functioning properly, then the text editor should still be running. And if the editor is running, the editor's cursor should be flashing. Additional information is that the hardware is pretty reliable, and is OK about 99% of the time, whereas Fred's LISP code is often buggy, say 40% of the time*[§].

1. Construct a Belief Network to represent and draw inferences about Fred's dilemma.

[§] Based on an example used in Dean, T., Allen, J. and Aloimonos, Y. *Artificial Intelligence Theory and Practice* (Chapter 8), Benjamin/Cumming Publishers, Redwood City, CA. 1995. With permission.

First decide what your domain variables are; these will be your network nodes. Hint: 5 or 6 Boolean variables should be sufficient. Then decide what the causal relationships are between the domain variables and add directed arcs in the network from cause to effect. Finanly, you have to add the conditional probabilities for nodes that have parents, and the prior probabilities for nodes without parents. Use the information about the hardware reliability and how often Fred's code is buggy. Other probabilities haven't been given to you explicitly; choose values that seem reasonable and explain why in your documentation.

2. Show the belief of each variable before adding any evidence, i.e., about the LISP visual prompt not being displayed.
3. Add the evidence about the LISP visual prompt not being displayed. After doing belief updating on the network, what is Fred's belief that he has a bug in his code?
4. Suppose that Fred checks the screen and the editor's cursor is still flashing. What effect does this have on his belief that the LISP interpreter is misbehaving because of a bug in his code? Explain the change in terms of diagnostic and predictive reasoning.

Problem 3

"A Lecturer's Life." *Dr. Ann Nicholson spends 60% of her work time in her office. The rest of her work time is spent elsewhere. When Ann is in her office, half the time her light is off (when she is trying to hide from students and get research done). When she is not in her office, she leaves her light on only 5% of the time. 80% of the time she is in her office, Ann is logged onto the computer. Because she sometimes logs onto the computer from home, 10% of the time she is not in her office, she is still logged onto the computer.*

1. Construct a Bayesian network to represent the "Lecturer's Life" scenario just described.
2. Suppose a student checks Dr. Nicholson's login status and sees that she is logged on. What effect does this have on the student's belief that Dr. Nicholson's light is on?

Problem 4

"Jason the Juggler." *Jason, the robot juggler, drops balls quite often when its battery is low. In previous trials, it has been determined that when its battery is low it will drop the ball 9 times out of 10. On the other hand when its battery is not low, the chance that it drops a ball is much lower, about 1 in 100. The battery was recharged recently, so there is only a 5% chance that the battery is low. Another robot, Olga the observer, reports on whether or not Jason has dropped the ball. Unfortunately Olga's vision system is somewhat unreliable. Based on information from Olga, the* *task is to represent and draw inferences about whether the battery is low depending on how well Jason is juggling*¶.

1. Construct a Bayesian network to represent the problem.
2. Which probability tables show where the information on how Jason's success is related to the battery level, and Olga's observational accuracy, are encoded in the network?
3. Suppose that Olga reports that Jason has dropped the ball. What effect does this have on your belief that the battery is low? What type of reasoning is being done?

Problem 5

Come up with your own problem involving reasoning with evidence and uncertainty. Write down a text description of the problem, then model it using a Bayesian network. Make the problem sufficiently complex that your network has at least 8 nodes and is multiply-connected (i.e., not a tree or a polytree).

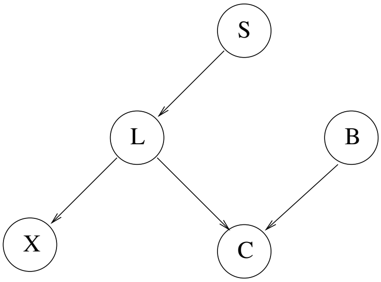
1. Show the beliefs for each node in the network before any evidence is added.
2. Which nodes are d-separated with no evidence added?
3. Which nodes in your network would be considered evidence (or observation) nodes? Which might be considered the query nodes? (Obviously this depends on the domain and how you might use the network.)
4. Show how the beliefs change in a form of diagnostic reasoning when evidence about at least one of the domain variables is added. Which nodes are d-separated with this evidence added?
5. Show how the beliefs change in a form of predictive reasoning when evidence about at least one of the domain variables is added. Which nodes are d-separated with this evidence added?
6. Show how the beliefs change through "explaining away" when particular combinations of evidence are added.
7. Show how the beliefs change when you change the priors for a root node (rather than adding evidence).

Conditional Independence

Problem 6

Consider the following Bayesian network for another version of the medical diagnosis example, where *B=Bronchitis*, *S=Smoker*, *C=Cough*, *X=Positive X-ray* and *L=Lung cancer* and all nodes are Booleans.

¶Variation of Exercise 19.6 in Nilsson, N.J. *Artificial Intelligence: A New Synthesis*, Copyright (1998). With permission from Elsevier.



List the pairs of nodes that are conditionally independent in the following situations:

1. There is no evidence for any of the nodes.
2. The cancer node is set to true (and there is no other evidence).
3. The smoker node is set to true (and there is no other evidence).
4. The cough node is set to true (and there is no other evidence).

Variable Ordering

Problem 7

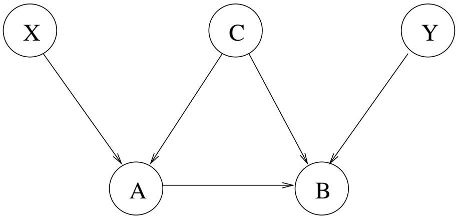
Consider the Bayesian network given for the previous problem.

1. What variable ordering(s) could have been used to produce the above network using the network construction algorithm (Algorithm 2.1)?
2. Given different variable orderings, what network structure would result from this algorithm? Use only pen and paper for now! Compare the number of parameters required by the CPTs for each network.

d-separation

Problem 8

Consider the following graph.



1. Find all the sets of nodes that d-separate X and Y (not including either X or Y in such sets).
2. Try to come up with a real-world scenario that might be modeled with such a network structure.

Problem 9

Design an internal representation for a Bayesian network structure; that is, a representation for the nodes and arcs of a Bayesian network (but not necessarily the parameters — prior probabilities and conditional probability tables). Implement a function which generates such a data structure from the Bayesian network described by a Netica `dne` input file. Use this function in the subsequent problems. (Sample `dne` files are available from the book Web site.)

Problem 10

Implement the network construction algorithm (Algorithm 2.1). Your program should take as input an ordered list of variables and prompt for additional input from the keyboard about the conditional independence of variables as required. It should generate a Bayesian network in the internal representation designed above. It should also print the network in some human-readable form.

Problem 11

Given as input the internal Bayesian network structure N (in the representation you have designed above), write a function which returns all undirected paths (Definition 2.1) between two sets **X** and **Y** of nodes in N.

Test your algorithm on various networks, including at least

- The d-separation network example from Problem 8, `dsepEg.dne`
- `Cancer_Neapolitan.dne`
- `ALARM.dne`

Summarize the results of these experiments.

Problem 12

Given the internal Bayesian network structure N, implement a **d-separation oracle** which, for any three sets of nodes input to it, **X**, **Y**, and **Z**, returns:

- **true** if $X \perp\!\!\!\perp Y|\mathbf{Z}$ (i.e., **Z** d-separates **X** and **Y** in N);
- **false** if $\neg(X \perp\!\!\!\perp Y|\mathbf{Z})$ (i.e., **Z** does not d-separate **X** and **Y** in N);
- some diagnostic (a value other than **true** or **false**) if an error in N is encountered.

Run your algorithm on a set of test networks, including at least the three network specified for Problem 11. Summarize the results of these experiments.

Problem 13

Modify your network construction algorithm from Problem 9 above to use the d-separation oracle from the last problem, instead of input from the user. Your new algorithm should produce exactly the same network as that used by the oracle whenever the variable ordering provided it is compatible with the oracle's network. Experiment with different variable orderings. Is it possible to generate a network which is simpler than the oracle's network?