Bayesian Belief Networks: Odds and Ends

Linda C. van der Gaag
Utrecht University, Department of Computer Science
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands

Abstract

In artificial intelligence research, the belief network framework for automated reasoning with uncertainty is rapidly gaining in popularity. The framework provides a powerful formalism for representing a joint probability distribution on a set of statistical variables. In addition, it offers algorithms for efficient probabilistic inference. At present, more and more knowledge-based systems employing the framework are being developed for various domains of application ranging from probabilistic information retrieval to medical diagnosis. This paper provides a tutorial introduction to the belief network framework and highlights some issues of ongoing research in applying the framework for real-life problem solving.

1 Introduction

Over the past few decades interest in the results of artificial intelligence research has been growing to an increasing extent. Especially the area of *knowledge-based systems* has attracted much attention. The phrase knowledge-based system, or *expert system*, is generally employed to denote computer systems in which some symbolic representation of human knowledge is incorporated and applied [Jackson, 1990, Lucas & van der Gaag, 1991]. Knowledge-based systems are typically designed to deal with real-life problems that require considerable human knowledge and expertise for their solution; examples range from medical diagnosis and technical trouble shooting to financial advice and product design. It is their ability to capture and reason with detailed human knowledge that allows knowledge-based systems to arrive at a performance comparable to that of human experts. Nowadays, knowledge-based systems have found their way from academic laboratories to the industrial world and are being integrated into conventional software environments.

As more and more knowledge-based systems are being developed for a large variety of problems, it becomes apparent that the knowledge required to solve these problems often is not precisely defined but instead is of an imprecise nature. In fact, many real-life problem domains are fraught with uncertainty. Human experts in these domains typically are able to form judgements and take decisions based on uncertain, incomplete, and sometimes even contradictory information. To be of practical use, a knowledge-based system has to deal with such information at least equally well. The development of formalisms for representing uncertainty and of algorithms for manipulating uncertain information by now has grown into a major research topic in artificial intelligence called *reasoning with uncertainty* or *plausible reasoning* [Shafer & Pearl, 1990].

Halfway through the 1980s, research on reasoning with uncertainty in knowledge-based systems resulted in the introduction of the *framework of Bayesian belief networks* [Pearl, 1988].

The framework has its origin in probability theory and is characterised by a powerful formalism for representing domain knowledge and the uncertainties that go with it — more in specific, the formalism provides for a concise representation of a joint probability distribution on a set of statistical variables. Associated with this formalism are algorithms for efficiently computing probabilities of interest and for processing evidence; these algorithms constitute the basic building blocks for reasoning with knowledge represented in the formalism. Since its introduction, the belief network framework has rapidly gained in popularity and by now is beginning to illustrate its worth in complex domains: practical applications are being developed for example for medical diagnosis and prognosis [Andreassen *et al.*, 1987, Heckerman *et al.*, 1992], for probabilistic information retrieval [Bruza & van der Gaag, 1994], and in computer vision [Jensen *et al.*, 1990b].

This paper gives a tutorial introduction to the belief network framework and highlights some issues of ongoing research in applying the framework for real-life problem solving. In Section 2 we briefly sketch the historical background of applying probability theory in knowledge-based systems. The Sections 3 and 4 introduce the framework of Bayesian belief networks: Section 3 details the belief network formalism and Section 4 outlines its associated algorithms. In Section 5 we briefly address building belief networks for real-life problem domains. Problem solving with belief networks is the topic of Section 6. The paper is rounded off with some conclusive discussion in Section 7.

2 Historical Background

As probability theory is a mathematically well-founded theory about uncertainty having a long and outstanding tradition of research and experience, it is not surprising that this theory takes a prominent place in research on reasoning with uncertainty in knowledge-based systems. Unfortunately, applying probability theory in a knowledge-based context is not as easy as it may seem at first sight. Straightforward application of the basic concepts of probability theory leads to insuperable problems of computational complexity: an explicit representation of a joint probability distribution requires exponential space (exponential in the number of variables discerned), and even if the distribution could be represented more economically, computing probabilities of interest by the basic rules of marginalisation and conditioning would still have an exponential time complexity. The historical background to the belief network framework shows various attempts to settle these problems. In this section, we sketch this historical background. We would like to note that our intention is not be complete, but merely to give an impression of the problems encountered by researchers pioneering in automated probabilistic inference; for a more elaborate overview, we refer the reader to [Horvitz *et al.*, 1988].

In our historical sketch, we focus on the task of (medical) diagnosis. For a given problem domain, we discern a set of possible hypotheses $H = \{h_1, \ldots, h_n\}$, $n \geq 1$, and a set of pieces of evidence $E = \{e_1, \ldots, e_m\}$, $m \geq 1$, that may be observed in relation with these hypotheses. A diagnostic problem in this domain is a set of pieces of evidence that is actually observed and needs to be explained in terms of the hypotheses. For ease of exposition, we assume that each of the hypotheses is either *true* or *false*; equally, we assume that each of the pieces of evidence is either *true* or *false*. A diagnosis for a problem $e \subseteq E$ under consideration now is a set of hypotheses $h \subseteq H$ that best explains e.

As early as in the 1960s several research efforts on automated reasoning with uncertainty were undertaken for the task of diagnosis [Warner *et al.*, 1961, Gorry & Barnett, 1968, de Dombal *et al.*, 1972]. The systems constructed in this period were based to a large extent on application of Bayes' Theorem; in the sequel, we will refer to the approach taken in these early systems as the *naive Bayesian approach.* The basic idea of computing a diagnosis for a set of actually observed pieces of evidence $e \subseteq E$ was to compute for all sets of hypotheses $h \subseteq H$ the conditional probability $\Pr(h \mid e)$ from the distribution Pr on the domain at hand, and then select a set $h' \subseteq H$ with highest probability. Since for real-life applications the conditional probabilities $\Pr(e \mid h)$ often were easier to come by than the conditional probabilities $\Pr(h \mid e)$, generally Bayes' Theorem was used for computing the required probabilities:

$$\Pr(h \mid e) = \frac{\Pr(e \mid h) \cdot \Pr(h)}{\Pr(e)}$$

It will be evident that this approach was computationally expensive: because a diagnosis could be composed of several different hypotheses, the number of probabilities to be computed equaled $2^n - 1$. To overcome this problem, a simplifying assumption was made: it was assumed that all hypotheses were mutually exclusive and collectively exhaustive. With this assumption only the n singleton hypothesis sets $\{h_i\}$ had to be considered as possible diagnoses. So, only the probabilities $\Pr(h_i \mid e)$ (writing h_i instead of $\{h_i\}$) for all $h_i \in H$ had to be computed. To this end, once more Bayes' Theorem was used:

$$\Pr(h_i \mid e) = \frac{\Pr(e \mid h_i) \cdot \Pr(h_i)}{\sum_{k=1}^{n} \Pr(e \mid h_k) \cdot \Pr(h_k)}$$

For automated application of Bayes' Theorem in this form, several probabilities were required from the joint probability distribution Pr on the domain at hand. In fact, conditional probabilities $\Pr(e \mid h_k)$, $k = 1, \ldots, n$, for *every* combination of pieces of evidence $e \subseteq E$ had to be available. Apart from the fact that it was hardly likely that these probabilities would be readily available in a real-life problem domain, this meant storing exponentially many probabilities. Hence, a second simplifying assumption was made: it was assumed that the pieces of evidence were conditionally independent given any of the hypotheses discerned. The two simplifying assumptions taken together allowed for computing the probabilities $\Pr(h_i \mid e)$ for all $h_i \in H$ given observed evidence $e = \{e_{j_1}, \ldots, e_{j_p}\}$, $1 \leq p \leq m$, from

$$\Pr(h_i \mid e_{j_1} \wedge \cdots \wedge e_{j_p}) = \frac{\Pr(e_{j_1} \mid h_i) \cdots \Pr(e_{j_p} \mid h_i) \cdot \Pr(h_i)}{\sum_{k=1}^{n} \Pr(e_{j_1} \mid h_k) \cdots \Pr(e_{j_p} \mid h_k) \cdot \Pr(h_k)}$$

It will be evident that for any problem now only $n - 1$ probabilities had to be computed, and that for this purpose only $m \cdot n$ conditional probabilities and $n - 1$ prior ones had to be stored.

The systems for automated reasoning with uncertainty constructed in the 1960s were rather small-scaled: they were devised for clear-cut problem domains with only a small number of hypotheses and restricted evidence. For these small systems, all probabilities necessary for applying Bayes' Theorem could be acquired from statistical analysis of empirical data. Despite the underlying (over-)simplifying assumptions, these systems performed considerably well [de Dombal *et al.*, 1974]. Nevertheless, interest in this naive Bayesian approach to reasoning with uncertainty faded in the late 1960s and early 1970s. One of the reasons for this decline in interest is that the approach was feasible only for highly restricted problem domains. For larger or more complex domains, the above-mentioned simplifying assumptions often were seriously violated, causing degeneration of system behaviour. In addition, for larger domains the approach inevitably became demanding, either computationally or from an assessment point of view.

At this stage, the first diagnostic knowledge-based systems began to emerge from artificial intelligence research. These systems mostly used *production rules* for representing human (experiential) knowledge in a modular form closely resembling logical implications — production rules are expressions of the form **if** ⟨*condition*⟩ **then** ⟨*conclusion*⟩. These so-called *rule-based* expert systems exhibited 'intelligent' reasoning behaviour by employing a heuristic reasoning algorithm that used the production rules for selective gathering of evidence and for pruning the search space of possible diagnoses. It is this pruning behaviour that rendered the rule-based expert systems capable of dealing with larger and complexer problem domains than the early naive-Bayesian systems were. The best-known rule-based expert system developed in the 1970s is the MYCIN system for assisting physicians in the diagnosis and treatment of bacterial infections [Buchanan & Shortliffe, 1984].

In the context of rule-based expert systems, the naive Bayesian approach to reasoning with uncertainty was no longer feasible due to the large number of probabilities to be computed: since during problem solving the search space of possible diagnoses was pruned by heuristic as well as probabilistic criteria, it became necessary to compute probabilities for all intermediate results derived by the production rules in addition to the probabilities of the separate hypotheses. To allow for efficient computation of all these probabilities, a set of computation rules was designed. These computation rules provided for computing the probability of an (intermediate) result from probabilities associated with the production rules that were used in its derivation; to this end, each production rule was assigned the conditional probability of its conclusion given its condition. Unfortunately, these computation rules did not always accord with the axioms of probability theory and could not even be considered rules for approximating probabilities. In the sequel, we will use the phrase *quasi-probabilistic* to refer to this approach. The most well-known illustration of the quasi-probabilistic approach is the *certainty factor model*, designed originally for dealing with uncertainty in the MYCIN system [Shortliffe & Buchanan, 1984]. The certainty factor model enjoys widespread use in rule-based expert systems built after MYCIN, even though it is widely known that the model is mathematically flawed. The relative success of the model can however be accounted for by its satisfactory behaviour in most applications and by its conceptual and computational simplicity [van der Gaag, 1994].

Although the quasi-probabilistic approach to reasoning with uncertainty on the one hand met with considerable success in the artificial intelligence community, it was criticised severely on the other hand because of its ad hoc character. The incorrectness of the approach from a mathematical point of view even led to a world-wide debate concerning the appropriateness of probability theory for handling uncertainty in a knowledge-based context. Here, we will not enter into this debate; for a wide range of diverging opinions, the reader is referred to [Cheeseman, 1988] with its ensuing discussions.

Although the above-mentioned debate had not yet subdued, in the mid-1980s the *belief network framework* was introduced as a novel approach to applying probability theory for reasoning with uncertainty in knowledge-based systems [Pearl, 1988]. When compared to the naive Bayesian approach on the one hand and the quasi-probabilistic approach on the other hand, the *belief network approach* offers advantages over both. In contrast with the quasi-probabilistic approach, the belief network approach has a firm mathematical foundation in probability theory. Contrasting the naive Bayesian approach, the belief network approach circumvents the need for simplifying assumptions by capturing and reasoning about actual independences among variables.

3 The Belief Network Formalism

The belief network framework offers a powerful and intuitively appealing formalism for representing a joint probability distribution for use in a knowledge-based system. In addition, the framework provides algorithms for reasoning with knowledge represented in this formalism. In this section, we focus on the belief network formalism; discussion of the associated algorithms is deferred to Section 4.

The *belief network formalism* provides for a concise representation of a joint probability distribution on a set of statistical variables; such a representation is called a *Bayesian belief network*, or *belief network* for short. Concision of representation is arrived at by explicit separation of information about the independences holding in the distribution at hand and the numerical quantities involved. To this end, a belief network comprises two parts: a *qualitative* part and a *quantitative* part.

The qualitative part of a belief network is a graphical representation of the independences holding among the variables in the distribution that is being represented. This part takes the form of an acyclic directed graph. In this digraph, each vertex represents a statistical variable that can take one of a finite set of values. The set of arcs of the digraph models the independences among these variables. Informally speaking, we take an arc $V_i \rightarrow V_j$ in the digraph to represent a direct influential or causal relationship between the linked variables V_i and V_j; the direction of the arc $V_i \rightarrow V_j$ designates V_j as the effect or consequence of the cause V_i. Absence of an arc between two vertices means that the corresponding variables do not influence each other directly, and hence, are (conditionally) independent.

Associated with the qualitative part of a belief network is a set of functions representing numerical quantities from the distribution at hand. With each vertex of the digraph is associated a *probability assessment function* which basically is a set of (conditional) probabilities describing the influence of the values of the vertex' predecessors on the probabilities of the values of this vertex itself. These probability assessment functions together constitute the quantitative part of the belief network. In the sequel, we will show that the probability assessment functions of a belief network provide all information necessary for uniquely defining a joint probability distribution that respects the independences portrayed by the qualitative part of the network.

Example 3.1 Consider the belief network shown in Figure 1 representing some fictitious medical 'knowledge' concerning the diagnosis of acute cardiac disorders. The information represented in the network pertains to patients presenting at a first aid clinic. We would like to stress that the belief network serves illustrative purposes only and should not be taken too seriously. The belief network comprises four binary variables: the variable S represents the *smoking history* of a patient, the variable M represents the presence or absence of a *myocardial infarction* (more commonly known as a heart attack), the variable P represents whether or not a patient is suffering from *pain on the chest*, and the variable F represents whether or not a patient complains of *tingling fingers*. In the digraph, the smoking history of a patient is modelled as having a direct influence on the presence or absence of a myocardial infarction in this patient. On having a myocardial infarction, a patient is likely to complain of pain on the chest: this is expressed through the high probability of chest pain given a myocardial infarction

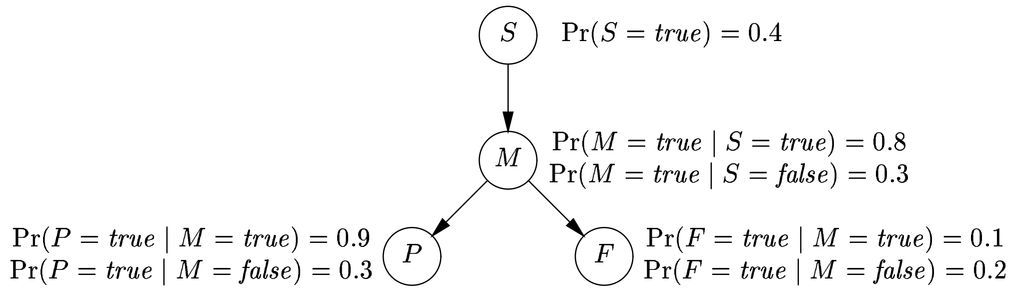
S
$\Pr(S = true) = 0.4$
M
$\Pr(M = true \mid S = true) = 0.8$
$\Pr(M = true \mid S = false) = 0.3$
P
$\Pr(P = true \mid M = true) = 0.9$
$\Pr(P = true \mid M = false) = 0.3$
F
$\Pr(F = true \mid M = true) = 0.1$
$\Pr(F = true \mid M = false) = 0.2$

Figure 1: A Fictitious Belief Network.

compared to the low probability of chest pain in the absence of a myocardial infarction. On the other hand, given a myocardial infarction a patient is not likely to have tingling fingers, which is expressed through the low probability $\Pr(F = true \mid M = true) = 0.1$; the presence of tingling fingers in fact suggests that the patient is suffering from another disorder such as hyperventilation, not modelled in our (incomplete) network. □

Before defining the concept of a belief network more formally, we provide some additional terminology and introduce our notational convention. In the sequel, the discussion will be restricted to binary variables taking one of the truth values *true* and *false*; the generalisation to variables with more than two discrete values, however, is straightforward. For abbreviation, we will use v_i to denote the proposition that the variable V_i takes the value *true*; $V_i = false$ will be denoted as $\neg v_i$. For a given set of variables V, the conjunction $C_V = \bigwedge_{V_i \in V} V_i$ of all variables from V is called the *configuration template* of V; a conjunction c_V of value assignments to the variables from V is called a *configuration* of V. Note that from the configuration template C_V of V, any configuration of V can be obtained by filling in appropriate values for all variables. In the sequel, we will use $\{C_V\}$ to denote the set of all configurations of V. To avoid abundance of braces, we will write C_{V_i} and c_{V_i} instead of $C_{\{V_i\}}$ and $c_{\{V_i\}}$, respectively, for singleton sets $\{V_i\}$. The independence relation of a joint probability distribution Pr will be denoted as $I_{\Pr}$. An *independence statement* $I_{\Pr}(X, Y, Z)$ signifies that in the distribution Pr the sets of variables X and Z are *conditionally independent* given the set of variables Y, that is, $\Pr(C_X \mid C_Y \wedge C_Z) = \Pr(C_X \mid C_Y)$; note that an independence statement applies to *all* configurations of the sets of variables involved.

Definition 3.2 *A* belief network *is a tuple* $B = (G, \Gamma)$ *where*

- $G = (V(G), A(G))$ *is an acyclic digraph with vertices* $V(G) = \{V_1, \ldots, V_n\}$, $n \geq 1$, *and arcs* $A(G)$*;*
- $\Gamma = \{\gamma_{V_i} \mid V_i \in V(G)\}$ *is a set of real-valued non-negative functions*

  $$\gamma_{V_i} \colon \{C_{V_i}\} \times \{C_{\rho_G(V_i)}\} \to [0, 1]$$

  called (conditional) probability assessment functions*, such that for each configuration* $c_{\rho_G(V_i)}$ *of the set* $\rho_G(V_i)$ *of (immediate) predecessors of vertex* V_i *in* G*, we have that* $\gamma_{V_i}(\neg v_i \mid c_{\rho_G(V_i)}) = 1 - \gamma_{V_i}(v_i \mid c_{\rho_G(V_i)})$, $i = 1, \ldots, n$.

Note that in the previous definition V_i is viewed as a vertex from the digraph and as a statistical variable, alternatively.

In order to link the qualitative and quantitative parts of a belief network, a probabilistic meaning is assigned to the topology of a digraph [Pearl, 1988].

Definition 3.3 *Let $G = (V(G), A(G))$ be an acyclic digraph and let s be a chain in G. We say that s is* blocked *by a set of vertices $W \subseteq V(G)$ if s contains three consecutive vertices X_1, X_2, X_3, for which one of the following conditions holds:*

1. *arcs $X_1 \leftarrow X_2$ and $X_2 \rightarrow X_3$ are on the chain s, and $X_2 \in W$;*
2. *arcs $X_1 \rightarrow X_2$ and $X_2 \rightarrow X_3$ are on the chain s, and $X_2 \in W$;*
3. *arcs $X_1 \rightarrow X_2$ and $X_2 \leftarrow X_3$ are on the chain s, and $\sigma^*(X_2) \cap W = \varnothing$, where $\sigma^*(X_2)$ denotes the set of vertices composed of X_2 and all its descendants.*

In defining the concept of a blocked chain, we have distinguished three conditions. Figure 2 serves as a reference for these conditions; in the two chains representing the conditions 1 and 2, vertex X_2 is drawn with shading to indicate that it is comprised in the blocking set W for the chain at hand.

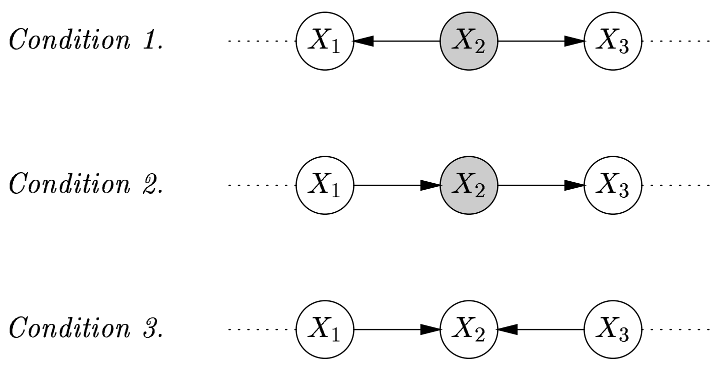

Figure 2: Chain Blocking.

The concept of blocking is defined to apply to single chains. Building on this concept we now define the *d-separation criterion* to apply to sets of chains.

Definition 3.4 *Let $G = (V(G), A(G))$ be an acyclic digraph and let $X, Y, Z \subseteq V(G)$ be mutually disjoint sets of vertices of G. The set Y is said to* d-separate *the sets X and Z in G, denoted as $\langle X | Y | Z \rangle_G^d$, if for each vertex $V_i \in X$ and each vertex $V_j \in Z$ every chain from V_i to V_j in G is blocked by Y.*

The following definition now relates the d-separation criterion to the concept of independence.

Definition 3.5 *Let $G = (V(G), A(G))$ be an acyclic digraph. Let* Pr *be a joint probability distribution on $V(G)$ and let I_{Pr} be its independence relation. Then, G is called an* I-map *for* Pr *if for all mutually disjoint sets of variables $X, Y, Z \subseteq V(G)$ we have: if $\langle X | Y | Z \rangle_G^d$ then $I_{\mathrm{Pr}}(X, Y, Z)$.*

Note that if a digraph is known to be an I-map for some joint probability distribution, the d-separation criterion provides for reading from the digraph independence statements holding in this distribution; this property allows for reasoning about independences without having to resort to probabilistic computations.

Example 3.6 Consider once more the belief network shown in Figure 1. From the digraph of the network we for example read that a patient's having pain on the chest is dependent upon this patient's smoking history. The smoking history, however, becomes irrelevant with respect to having chest pain once it has been established that the patient is suffering from a myocardial infarction. □

The following proposition now states that the probability assessment functions of a belief network provide all information necessary for uniquely defining a joint probability distribution on the variables discerned that respects the independence relation portrayed by the qualitative part of the network; henceforth, we will call this distribution the joint probability distribution *defined* by the network.

Proposition 3.7 *Let $B = (G, \Gamma)$ be a belief network. Then,*

$$\Pr(C_{V(G)}) = \prod_{V_i \in V(G)} \gamma_{V_i}(V_i \mid C_{\rho_G(V_i)})$$

defines a joint probability distribution $\Pr$ *on* $V(G)$ *such that* G *is an I-map for* $\Pr$.

Proof (Sketch). A digraph G without any directed cycles allows at least one total ordering of its vertices such that any successor to a vertex in the digraph follows it in the ordering. Any such ordering of the vertices can be taken as an ordering of the corresponding variables. Now consider applying the basic *chain rule* from probability theory to $\Pr(C_{V(G)})$ such that every variable from $V(G)$ is conditioned only on the variables preceding it in an ordering having the property mentioned above. The independences portrayed by G can then be exploited. By taking $\Pr(V_i \mid C_{\rho_G(V_i)}) = \gamma_{V_i}(V_i \mid C_{\rho_G(V_i)})$ for each V_i and its set of immediate predecessors $\rho_G(V_i)$, the property stated in the proposition follows immediately. □

Since the digraph of a belief network and its associated probability assessment functions with each other define a joint probability distribution on the variables discerned, any (prior or posterior) probability of interest can be computed from these functions. We will return to this observation in the following section.

Example 3.8 Consider once more the belief network shown in Figure 1. From Proposition 3.7, we have for the probability $\Pr(S = \mathit{true} \wedge M = \mathit{true} \wedge P = \mathit{false} \wedge F = \mathit{false})$ that

$$\begin{aligned}
&\Pr(S = \mathit{true} \wedge M = \mathit{true} \wedge P = \mathit{false} \wedge F = \mathit{false}) = \\
&\quad = \Pr(S = \mathit{true}) \cdot \Pr(M = \mathit{true} \mid S = \mathit{true}) \cdot \Pr(P = \mathit{false} \mid M = \mathit{true}) \cdot \\
&\qquad \cdot \Pr(F = \mathit{false} \mid M = \mathit{true}) = \\
&\quad = 0.4 \cdot 0.8 \cdot 0.1 \cdot 0.9 = 0.03
\end{aligned}$$

Note that for uniquely describing the joint probability distribution $\Pr$ on the four variables S, M, P, and F only seven probabilities are required in addition to the digraph; in contrast, a straightforward representation of the distribution would require fifteen probabilities. □

To conclude, we would like to note that the belief network formalism can be extended to the formalism of *influence diagrams* to allow for the representation of decision problems. A decision problem involves probabilistic information as well as information about viable decisions and preferences. While the former information can be expressed in the belief network formalism, the latter cannot: for representing this information, the belief network formalism is enhanced. Just like a belief network, an influence diagram comprises an acyclic directed graph. The influence diagram, however, models various different types of vertex and arc, bearing different meanings. Various algorithms have been designed for evaluating influence diagrams. These algorithms provide for computing the best decision(s) given the uncertainties and preferences involved [Howard & Matheson, 1981, Shachter, 1986]. In this paper, we focus on *probabilistic* reasoning only and therefore will not discuss the influence diagram formalism and its associated algorithms any further.

4 Probabilistic Inference

In the previous section, we have introduced the belief network formalism for representing a joint probability distribution on a set of statistical variables. Once a belief network is constructed, it can be used for *probabilistic inference*, that is, for making probabilistic statements concerning the represented variables. Since a belief network uniquely defines a joint probability distribution, any probability of interest can be computed from the network by explicitly generating the full distribution by means of Proposition 3.7, and then using the basic rules of marginalisation and conditioning. Such a straightforward approach, however, is computationally infeasible. More efficient algorithms for probabilistic inference have been designed that exploit the independences that are represented by the qualitative part of a belief network. The most well-known of these are the algorithms by J. Pearl [Pearl, 1988] and by S.L. Lauritzen and D.J. Spiegelhalter [Lauritzen & Spiegelhalter, 1988]. In the sequel, we will discuss Pearl's algorithm in some detail and briefly address other algorithms for efficient probabilistic inference with a belief network.

4.1 Pearl's Algorithm

The basic idea of Pearl's algorithm for probabilistic inference is best explained from an object-centered point of view. The digraph of a belief network is taken as a *computational architecture*: the vertices of the digraph are *autonomous objects*, and its arcs are *bi-directional communication channels*. Each vertex has a local processor that is capable of performing simple probabilistic computations and a local memory in which its associated probability assessment function is stored. Through the communication channels the vertices send each other *parameters* providing information about the joint probability distribution that is defined by the network and about the evidence obtained so far. Each vertex is able to compute the probabilities of its values from its own probability assessment function and the information it receives from its neighbours.

Initially, the belief network is in an *equilibrium* state: recomputation of the various parameters does not result in a change in any of them. Now, when a piece of evidence is entered for some vertex, this equilibrium is perturbed: the parameters this vertex sends to its neighbours are updated to reflect the entered evidence. After receiving updated parameters, these neighbours in turn compute new parameters to send to their neighbours, and so on: the impact of the evidence spreads through the network by *message-passing* between neighbouring vertices. In this process of parameter updating, each vertex is visited only once. The belief network will therefore reach a new equilibrium after a finite number of steps.

In this section, we detail the computations involved in Pearl's algorithm. A reader who is less interested in mathematical details may skip the remainder of this section and continue reading at Section 4.2.

4.1.1 Directed trees

In discussing Pearl's algorithm, we first focus on probabilistic inference with a belief network where the qualitative part is a *directed tree*, that is, in the network's digraph a vertex may have several successors but at most one predecessor. In Section 4.1.2 we address probabilistic inference with belief networks comprising a digraph of more general topology.

Before detailing the computations involved in Pearl's algorithm, we introduce some more terminology and notational convention that we will use in the sequel. Let G be the digraph of some belief network and let $V \subseteq V(G)$ be a subset of its set of vertices. A vertex $V_i \in V$ for which either $V_i = \mathit{true}$ or $V_i = \mathit{false}$ has been observed with certainty is called *instantiated*; if no evidence has been obtained as yet for a vertex, it is called *uninstantiated*. Now, let $X \subseteq V$ be the set of instantiated vertices from V. The configuration c_X of X is called a *partial configuration* of V and will be denoted as $\tilde{c}_V$. Note that the notation $\tilde{c}_V$ allows for referring to the subset of instantiated vertices from V without having to specify this subset explicitly.

At any time during probabilistic inference with a belief network, the probabilities of the values of a vertex are dependent upon all evidence entered so far.

Lemma 4.1 *Let $B = (G, \Gamma)$ be a belief network where $G = (V(G), A(G))$ is a directed tree and let* Pr *be the joint probability distribution defined by B. Let $V_i \in V(G)$ be a vertex in G and let $V_i^- = \sigma^*(V_i)$ and $V_i^+ = V(G) \setminus V_i^-$. Then,*

$$\Pr(V_i \,|\, \tilde{c}_{V(G)}) = \alpha_i \cdot \Pr(\tilde{c}_{V_i^-} \mid V_i) \cdot \Pr(V_i \mid \tilde{c}_{V_i^+})$$

where $\tilde{c}_{V(G)} = \tilde{c}_{V_i^-} \wedge \tilde{c}_{V_i^+}$ and α_i is a normalisation constant.

Proof. For the probabilities $\Pr(V_i \mid \tilde{c}_{V(G)})$ of the values of vertex V_i, we have

$$\Pr(V_i \mid \tilde{c}_{V(G)}) = \frac{\Pr(\tilde{c}_{V_i^-} \wedge \tilde{c}_{V_i^+} \mid V_i) \cdot \Pr(V_i)}{\Pr(\tilde{c}_{V_i^-} \wedge \tilde{c}_{V_i^+})}$$

by Bayes' Theorem. Now consider Figure 3 showing a fragment of the directed tree G. We observe that $\langle X | \{V_i\} | Y \rangle_G^d$ for all subsets $X \subseteq V_i^- \setminus \{V_i\}$ and $Y \subseteq V_i^+$. Since G is an I-map for the joint probability distribution Pr, we conclude that $I_{\Pr}(X, \{V_i\}, Y)$ for all $X \subseteq V_i^- \setminus \{V_i\}$, $Y \subseteq V_i^+$. Exploiting this observation, we find

$$\Pr(V_i \mid \tilde{c}_{V(G)}) = \frac{\Pr(\tilde{c}_{V_i^-} \mid V_i) \cdot \Pr(V_i \mid \tilde{c}_{V_i^+})}{\Pr(\tilde{c}_{V_i^-} \mid \tilde{c}_{V_i^+})}$$

Note that the factor $\frac{1}{\Pr(\tilde{c}_{V_i^-} | \tilde{c}_{V_i^+})}$ depends on the vertex V_i but not on its values. It therefore is a constant with respect to V_i. In the sequel, this constant will be denoted as α_i; the constant α_i is generally referred to as a *normalisation constant* because it can be computed

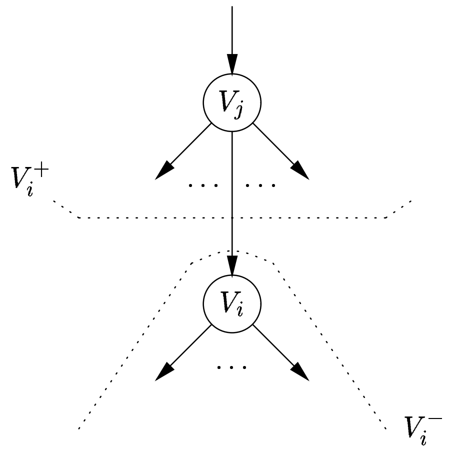

Figure 3: A Fragment of the Directed Tree G.

from $\Pr(v_i \mid \tilde{c}_{V(G)}) + \Pr(\neg v_i \mid \tilde{c}_{V(G)}) = 1$. The property stated in the lemma now follows by substitution. □

The previous lemma shows that the probabilities of the values of a vertex can be expressed in terms of two factors describing the influence of evidence entered for its descendants and for all other vertices, separately. The following definition introduces some new terminology for these separate factors.

Definition 4.2 *Let $B = (G, \Gamma)$ be a belief network as before and let* $\Pr$ *be the joint probability distribution defined by B. Let $V_i \in V(G)$ be a vertex in G and let V_i^- and V_i^+ be as in Lemma 4.1. The* compound causal parameter *π_{V_i} for V_i is the function $\pi_{V_i}\colon \{C_{V_i}\} \rightarrow [0,1]$ defined by*

$$\pi_{V_i}(V_i) = \Pr(V_i \mid \tilde{c}_{V_i^+})$$

The compound diagnostic parameter *λ_{V_i} for V_i is the function $\lambda_{V_i}\colon \{C_{V_i}\} \rightarrow [0,1]$ defined by*

$$\lambda_{V_i}(V_i) = \Pr(\tilde{c}_{V_i^-} \mid V_i)$$

We take a closer look at the previous definition for *uninstantiated* vertices having either no incoming or no outgoing arcs. A directed tree has one vertex W without any incoming arcs; this vertex is the *root* of the tree. We observe that for W the set W^+ is empty. So, $\tilde{c}_{W^+} = \mathit{true}$. The compound causal parameter π_W for W therefore equals $\pi_W(W) = \Pr(W)$. The directed tree may further comprise several vertices having no outgoing arcs; these vertices are the *leaves* of the tree. For a leaf V_i, we observe that the set V_i^- consists of V_i only. From V_i being uninstantiated, we then have that $\tilde{c}_{V_i^-} = \mathit{true}$. The compound diagnostic parameter λ_{V_i} for V_i therefore equals $\lambda_{V_i}(V_i) = 1$. To conclude, we consider *instantiated* vertices. For a vertex V_j for which the evidence $V_j = \mathit{true}$ has been entered, we find $\pi_{V_j}(V_j) = \Pr(V_j \mid \tilde{c}_{V_j^+})$, and $\lambda_{V_j}(v_j) = \Pr(\tilde{c}_{V_j^-} \mid v_j)$ and $\lambda_{V_j}(\neg v_j) = 0$; an analogous observation holds for the case where the evidence $V_j = \mathit{false}$ has been entered.

Using the definition of the compound causal and diagnostic parameters for a vertex, Lemma 4.1 can now be reformulated as

$$\Pr(V_i \mid \tilde{c}_{V(G)}) = \alpha_i \cdot \pi_{V_i}(V_i) \cdot \lambda_{V_i}(V_i)$$

In this form, the lemma is known as the *data fusion lemma* [Pearl, 1988]. Note that the data fusion lemma implies that the compound and diagnostic parameters for a vertex provide it with enough information for computing the probabilities of its values, that is, no further knowledge of the joint probability distribution is needed.

The compound diagnostic parameter for a vertex specifies probabilistic information from *all* its descendants combined; an analogous observation applies to the compound causal parameter for the vertex. To be able to exploit the digraph of a belief network as a computational architecture as outlined before, the compound parameters for a vertex have to be computed from separate parameters originating from its various neighbours. The following definition introduces such separate parameters; the Lemmas 4.4 and 4.5 will show the computation of the compound parameters from these separate ones.

Definition 4.3 *Let $B = (G, \Gamma)$ be a belief network as before and let Pr be the joint probability distribution defined by B. For each vertex $V \in V(G)$, let V^- and V^+ be as in Lemma 4.1. Now, let V_i be a vertex in G with a successor V_k. The* causal parameter $\pi_{V_k}^{V_i}$ from V_i to V_k *is the function $\pi_{V_k}^{V_i} : \{C_{V_i}\} \rightarrow [0,1]$ defined by*

$$\pi_{V_k}^{V_i}(V_i) = \Pr(V_i \mid \tilde{c}_{V_k^+})$$

Now, let V_i be a vertex in G having the predecessor V_j. The diagnostic parameter $\lambda_{V_i}^{V_j}$ from V_i to V_j *is the function $\lambda_{V_i}^{V_j} : \{C_{V_j}\} \rightarrow [0,1]$ defined by*

$$\lambda_{V_i}^{V_j}(V_j) = \Pr(\tilde{c}_{V_i^-} \mid V_j)$$

Note that for the root of the directed tree no diagnostic parameter has been defined and that for the leaves of the tree no causal parameters have been defined. In addition, note that for a variable V_i for which the evidence $V_i = true$ is observed, we find $\pi_{V_k}^{V_i}(v_i) = 1$ and $\pi_{V_k}^{V_i}(\neg v_i) = 0$; an analogous observation holds for the case where $V_i = false$ is observed. The separate causal and diagnostic parameters defined above are the messages the vertices send each other through the communication channels of the computational architecture and, hence, may be looked upon as associated with the arcs of the directed tree of the network.

The following lemma now shows that a vertex can compute its compound causal parameter from the causal parameter it receives from its predecessor and its own probability assessment function.

Lemma 4.4 *Let $B = (G, \Gamma)$ be a belief network as before. Let $V_i \in V(G)$ be a vertex in G with the predecessor V_j. Let π_{V_i} be the compound causal parameter for V_i and let $\pi_{V_i}^{V_j}$ be the causal parameter from V_j to V_i. Then,*

$$\pi_{V_i}(V_i) = \sum_{c_{V_j}} \gamma_{V_i}(V_i \mid c_{V_j}) \cdot \pi_{V_i}^{V_j}(c_{V_j})$$

Proof. Let Pr be the joint probability distribution defined by the belief network B. For vertex V_i, let V_i^+ be as before. Then, by definition we have

$$\pi_{V_i}(V_i) = \Pr(V_i \mid \tilde{c}_{V_i^+})$$

By conditioning on the values of V_i's predecessor V_j, we find

$$\pi_{V_i}(V_i) = \Pr(V_i \mid v_j \wedge \tilde{c}_{V_i^+}) \cdot \Pr(v_j \mid \tilde{c}_{V_i^+}) + \Pr(V_i \mid \neg v_j \wedge \tilde{c}_{V_i^+}) \cdot \Pr(\neg v_j \mid \tilde{c}_{V_i^+})$$

Now consider once more Figure 3 showing a fragment of the directed tree G. We observe that $\langle \{V_i\} | \{V_j\} | X \rangle_G^d$ for all subsets $X \subseteq V_i^+ \setminus \{V_j\}$. Since G is an I-map for Pr, we have that $I_{\Pr}(\{V_i\}, \{V_j\}, X)$ for all $X \subseteq V_i^+ \setminus \{V_j\}$. Exploiting this observation, we find

$$\pi_{V_i}(V_i) = \Pr(V_i \mid v_j) \cdot \Pr(v_j \mid \tilde{c}_{V_i^+}) + \Pr(V_i \mid \neg v_j) \cdot \Pr(\neg v_j \mid \tilde{c}_{V_i^+})$$

The probabilities $\Pr(V_i \mid V_j)$ have been specified as function values $\gamma_{V_i}(V_i \mid V_j)$ of the probability assessment function γ_{V_i}, and therefore are directly available to vertex V_i. In addition, V_i receives the probabilities $\Pr(V_j \mid \tilde{c}_{V_i^+})$ from its predecessor V_j as function values $\pi_{V_i}^{V_j}(V_j)$ of the causal parameter $\pi_{V_i}^{V_j}$. The property stated in the lemma now follows by substitution. □

A vertex can further compute its compound diagnostic parameter from the separate diagnostic parameters it receives from its successors. This property is stated more formally in the following lemma.

Lemma 4.5 *Let $B = (G, \Gamma)$ be a belief network as before. Let $V_i \in V(G)$ be an uninstantiated vertex in G with $\sigma(V_i) = \{V_{i_1}, \ldots, V_{i_m}\}$, $m \geq 1$. Furthermore, let λ_{V_i} be the compound diagnostic parameter for V_i and for each $V_{i_j} \in \sigma(V_i)$, let $\lambda_{V_{i_j}}^{V_i}$ be the diagnostic parameter from V_{i_j} to V_i. Then,*

$$\lambda_{V_i}(V_i) = \prod_{j=1,\ldots,m} \lambda_{V_{i_j}}^{V_i}(V_i)$$

Proof. Let Pr be the joint probability distribution defined by B. For each vertex $V \in V(G)$, let V^- be as before. Since V_i is an uninstantiated vertex, we have that $\tilde{c}_{V_i^-} = \tilde{c}_{V_{i_1}^-} \wedge \cdots \wedge \tilde{c}_{V_{i_m}^-}$; so,

$$\lambda_{V_i}(V_i) = \Pr(\tilde{c}_{V_{i_1}^-} \wedge \cdots \wedge \tilde{c}_{V_{i_m}^-} \mid V_i)$$

Now consider Figure 4 showing a fragment of the directed tree G. We observe that $\langle X | \{V_i\} | Y \rangle_G^d$

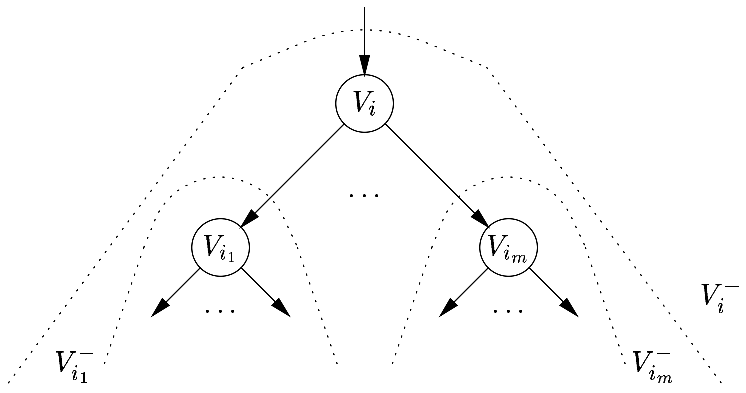

Figure 4: Exploiting d-Separation for Computing $\lambda_{V_i}(V_i)$.

for all subsets $X \subseteq V_{i_j}^-$ and $Y \subseteq \bigcup_{k=1,\ldots,m,k\neq j} V_{i_k}^-$, $j = 1,\ldots,m$. Since G is an I-map for Pr, we have that $I_{\Pr}(X, \{V_i\}, Y)$ for $X \subseteq V_{i_j}^-$, $Y \subseteq \bigcup_{k=1,\ldots,m,k\neq j} V_{i_k}^-$, $j = 1,\ldots,m$. It follows that

$$\lambda_{V_i}(V_i) = \Pr(\tilde{c}_{V_{i_1}^-} \mid V_i) \cdot \ldots \cdot \Pr(\tilde{c}_{V_{i_m}^-} \mid V_i)$$

The probabilities $\Pr(\tilde{c}_{V_{i_j}^-} \mid V_i)$ are sent to vertex V_i by its successor V_{i_j} as function values $\lambda_{V_{i_j}}^{V_i}(V_i)$ of the diagnostic parameter $\lambda_{V_{i_j}}^{V_i}$, $j = 1,\ldots,m$. The property stated in the lemma now follows by substitution. □

Note that the previous lemma applies to *uninstantiated* vertices only. However, the property mentioned in the lemma can be taken to hold for an instantiated vertex V_i as well, if entering evidence for V_i into the network is modelled by adding a 'dummy' successor D to V_i that sends an appropriate diagnostic parameter to V_i. For the evidence $V_i = \mathit{true}$, this 'dummy' successor sends the diagnostic parameter $\lambda_D^{V_i}$ with $\lambda_D^{V_i}(v_i) = 1$ and $\lambda_D^{V_i}(\neg v_i) = 0$; an analogous observation holds for the evidence $V_i = \mathit{false}$.

So far, we have shown that a vertex can compute the probabilities of its values from its own probability assessment function and the causal and diagnostic parameters it receives from its neighbours. Now observe that the vertex in turn has to compute parameters to send to its neighbours. The following lemma shows that a vertex can compute the diagnostic parameter to send to its predecessor from its own assessment function and the diagnostic parameters it receives from its successors. In other words, for this purpose it combines its own information about the joint probability distribution with the information it receives concerning the evidence obtained so far for its descendants. We state this lemma without proof.

Lemma 4.6 *Let $B = (G, \Gamma)$ be a belief network as before. Let $V_i \in V(G)$ be a vertex in G with the predecessor V_j. Let λ_{V_i} be the compound diagnostic parameter for V_i and let $\lambda_{V_i}^{V_j}$ be the diagnostic parameter from V_i to V_j. Then,*

$$\lambda_{V_i}^{V_j}(V_j) = \sum_{c_{V_i}} \lambda_{V_i}(c_{V_i}) \cdot \gamma_{V_i}(c_{V_i} \mid V_j)$$

Similarly, a vertex can compute the causal parameter to send to a successor from its compound causal parameter and the diagnostic parameters it receives from its other successors. The following lemma states this property more formally.

Lemma 4.7 *Let $B = (G, \Gamma)$ be a belief network as before. Let $V_i \in V(G)$ be an uninstantiated vertex in G with $\sigma(V_i) = \{V_{i_1}, \ldots, V_{i_m}\}$, $m \geq 1$. Furthermore, let π_{V_i} be the compound causal parameter for V_i. For each $V_{i_j} \in \sigma(V_i)$, let $\pi_{V_{i_j}}^{V_i}$ be the causal parameter from V_i to V_{i_j} and let $\lambda_{V_{i_j}}^{V_i}$ be the diagnostic parameter from V_{i_j} to V_i. Then,*

$$\pi_{V_{i_j}}^{V_i}(V_i) = \alpha_i \cdot \pi_{V_i}(V_i) \cdot \prod_{k=1,\ldots,m,k\neq j} \lambda_{V_{i_k}}^{V_i}(V_i)$$

where α_i is a normalisation constant.

The previous lemma applies to uninstantiated vertices only; the lemma, however, can be taken to hold for instantiated vertices as described before.

The data fusion lemma and the four computation rules provided by the Lemmas 4.4, 4.5, 4.6, and 4.7 with each other constitute Pearl's algorithm for probabilistic inference with a belief network comprising a directed tree for its qualitative part. Note that Pearl's algorithm provides for computing probabilities as well as for processing evidence. The computation rules for the separate causal and diagnostic parameters enable a vertex to pass on the impact of a piece of evidence correctly, and allow for the evidence to spread throughout the network. Close examination of the computation rules from the Lemmas 4.6 and 4.7 further reveals that a neighbour that sends an updated parameter will not receive a new parameter originating from the same evidence. A causal parameter or a diagnostic parameter *to* a vertex is not affected by the diagnostic parameter or the causal parameter, respectively, *from* that vertex; in addition, the topological property that in a directed tree there is at most one chain between any two vertices prohibits the process of parameter updating to reach this vertex along another chain. These properties with each other guarantee that feedback and circular reasoning are prevented and that evidence is propagated throughout the network in a single pass.

Example 4.8 Consider once more the belief network shown in Figure 1 representing fictitious medical 'knowledge' concerning the diagnosis of acute cardiac disorders. Suppose that we are interested in the prior probabilities of the presence and absence, respectively, of a myocardial infarction in a patient. These probabilities are computed by vertex M by application of the data fusion lemma:

$$\Pr(m) = \alpha_M \cdot \pi_M(m) \cdot \lambda_M(m)$$

$$\Pr(\neg m) = \alpha_M \cdot \pi_M(\neg m) \cdot \lambda_M(\neg m)$$

Vertex M computes the compound causal parameter π_M from its own probability assessment function γ_M and the causal parameter π_M^S it receives from its predecessor S by applying the computation rule from Lemma 4.4. From vertex S, it receives the information

$$\pi_M^S(s) = \pi_S(s) = 0.4$$

$$\pi_M^S(\neg s) = \pi_S(\neg s) = 0.6$$

So,

$$\pi_M(m) = \gamma_M(m \mid s) \cdot \pi_M^S(s) + \gamma_M(m \mid \neg s) \cdot \pi_M^S(\neg s) = 0.5$$

$$\pi_M(\neg m) = \gamma_M(\neg m \mid s) \cdot \pi_M^S(s) + \gamma_M(\neg m \mid \neg s) \cdot \pi_M^S(\neg s) = 0.5$$

Vertex M further computes the compound diagnostic parameter λ_M from the diagnostic parameters λ_P^M and λ_F^M it receives from its successors P and F, respectively, by applying the computation rule from Lemma 4.5. From P and F, it receives the information

$$\lambda_P^M(m) = 1$$

$$\lambda_P^M(\neg m) = 1$$

and

$$\lambda_F^M(m) = 1$$

$$\lambda_F^M(\neg m) = 1$$

respectively. So,

$$\lambda_M(m) = \lambda_P^M(m) \cdot \lambda_F^M(m) = 1$$

$$\lambda_M(\neg m) = \lambda_P^M(\neg m) \cdot \lambda_F^M(\neg m) = 1$$

Substitution of the compound parameters into the data fusion lemma and elimination of the normalisation constant α_M yields

$$\Pr(m) = 0.5$$

$$\Pr(\neg m) = 0.5$$

If the prior probability of the presence of a myocardial infarction seems to be rather high, then recall that the information from the network is conditional on a patient's presenting to a first aid clinic. For each vertex, the prior probabilities of its values are summarised in Figure 5.

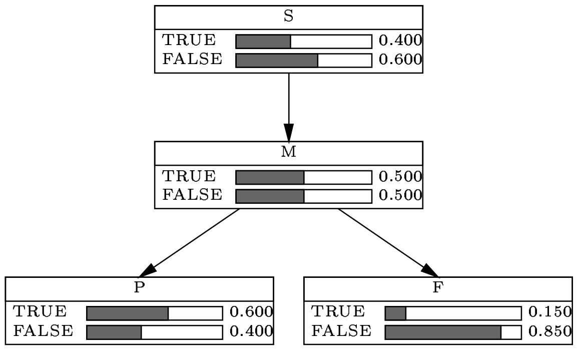

Figure 5: The Prior Probabilities.

Now suppose that a patient under consideration complains of pain on the chest, that is, the evidence $P = true$ is obtained. After entering this evidence, the posterior probabilities of the presence and absence of a myocardial infarction in this patient can be computed from the network. Vertex M once more applies the data fusion lemma:

$$\Pr(m \mid p) = \alpha_M \cdot \pi_M(m) \cdot \lambda_M(m)$$

$$\Pr(\neg m \mid p) = \alpha_M \cdot \pi_M(\neg m) \cdot \lambda_M(\neg m)$$

Since no evidence has been entered as yet for vertex S, this vertex sends the same causal parameter π_M^S to vertex M as before. The compound causal parameter π_M for vertex M therefore remains unaltered:

$$\pi_M(m) = 0.5$$

$$\pi_M(\neg m) = 0.5$$

From its successor F, vertex M also receives the same information as before. From vertex P, however, it receives an updated diagnostic parameter reflecting the entered evidence:

$$\lambda_P^M(m) = \gamma_P(p \mid m) = 0.9$$

$$\lambda_P^M(\neg m) = \gamma_P(p \mid \neg m) = 0.3$$

From the parameters λ_F^M and λ_P^M, vertex M computes its compound diagnostic parameter λ_M to be

$$\lambda_M(m) = \lambda_P^M(m) \cdot \lambda_F^M(m) = 0.9$$

$$\lambda_M(\neg m) = \lambda_P^M(\neg m) \cdot \lambda_F^M(\neg m) = 0.3$$

Substitution of the compound parameters into the data fusion lemma and elimination of the normalisation constant α_M yields

$$\Pr(m) = 0.75$$

$$\Pr(\neg m) = 0.25$$

Note that as a result of the evidence obtained, the likelihood of this patient having a myocardial infarction has increased considerably. For each vertex, the posterior probabilities of its values are summarised in Figure 6. □

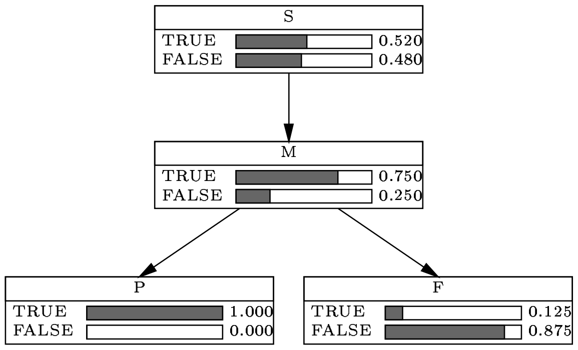

Figure 6: The Posterior Probabilities Given $P = true$.

4.1.2 General digraphs

So far, we have only considered probabilistic inference with a belief network comprising a directed tree. We will now sketch how Pearl's algorithm is extended to apply to belief networks comprising a digraph of more general topology.

We first consider extending Pearl's algorithm to apply to *singly connected digraphs*, that is, to digraphs where the underlying (undirected) graph is acyclic. The proofs of the lemmas presented in the previous section show that Pearl's computation rules for directed trees derive from exploiting independences. These independences are read from the qualitative part of a belief network by local inspection of a vertex' incoming and outgoing arcs only. The computation rules therefore make use explicitly of the property that there is at most one chain between any two vertices. Since singly connected digraphs share this property with directed trees, Pearl's algorithm is easily extended to apply to singly connected digraphs. In fact, only the computation rules for the compound causal parameter and the diagnostic parameter are adapted to allow for a vertex having multiple predecessors; all other computation rules remain unaltered. For further details, the reader is referred to [Pearl, 1988].

Pearl's algorithm, however, is not as easily extended to apply to acyclic *multiply connected digraphs* in general. Straightforward application of Pearl's algorithm to an acyclic digraph comprising one or more loops invariably leads to insuperable problems [Pearl, 1988, Suermondt & Cooper, 1990]. Vertices may indefinitely send updated messages to their neighbours, causing the network never to reach a new equilibrium. Yet, even if the network *does* reach a new equilibrium, it is not guaranteed to correctly reflect the updated joint probability distribution. Both problems originate from the same source: since there now is more than one chain between two vertices, local inspection of a vertex' incoming and outgoing arcs no longer serves for reading independences from the digraph of the network at hand.

Pearl has proposed several methods for probabilistic inference with a belief network comprising a multiply connected digraph, all of which retain to some extent the property of locality of computation [Pearl, 1988]. Of these, the method of *loop cutset conditioning* may be looked upon as a supplement to the basic algorithm. The idea underlying this method is that of *reasoning by assumption.* For a multiply connected digraph, vertices are selected that, upon instantiation, with each other effectively 'cut' or block all loops and cause the digraph to behave as if it were singly connected; the selected vertices are said to constitute a *loop cutset* of the digraph. Each configuration of the loop cutset now is looked upon as a (compound) assumption on which reasoning is performed. For each vertex, the probabilities of its values are computed by conditioning successively on all possible configurations of the loop cutset and subsequently weighting the results obtained. The probabilities of a vertex' values given some configuration of the loop cutset can be computed by Pearl's algorithm for singly connected digraphs. The interested reader is referred to [Pearl, 1988, Suermondt & Cooper, 1990] for further details on the method of loop cutset conditioning.

4.2 Other Algorithms

In the previous section, we have discussed Pearl's algorithm for probabilistic inference with a belief network. Pearl's algorithm, however, is not the only algorithm designed for this purpose: several different algorithms have been proposed in the course of the last decade.

In general, an algorithm for probabilistic inference with a belief network is built from two basic procedures: a procedure for (efficiently) computing probabilities of interest from a belief network, and a procedure for processing evidence, that is, for entering evidence into the network and subsequently (efficiently) computing the revised joint probability distribution given the evidence. Note that in Pearl's algorithm these basic procedures have been combined into a set of computation rules where they cannot easily be distinguished. In most other algorithms, however, these basic procedures are more readily discernable. The algorithms proposed further have two important properties in common: the qualitative part of a belief network is exploited more or less directly as a computational architecture and, after a piece of evidence has been processed in the network, again a belief network results. Note that the latter property allows for recursive processing of evidence.

Although all algorithms proposed for probabilistic inference build on the same notion of a belief network, they differ considerably with respect to their underlying concepts. To support this observation, we briefly review another, elegant algorithm for probabilistic inference with a belief network, designed by S.L. Lauritzen and D.J. Spiegelhalter [Lauritzen & Spiegelhalter, 1988]. Lauritzen and Spiegelhalter have observed that after a piece of evidence has become available, updating the joint probability distribution generally entails going against the 'direction' of the initially assessed conditional probabilities. From this observation they have concluded that the *directed* qualitative part of a belief network is not suitable as an architecture for probabilistic inference directly. Their algorithm therefore departs from an *undirected* representation of a joint probability distribution. Prior to probabilistic inference, the original belief network is transformed into a so-called *decomposable* belief network. A decomposable belief network again consists of a qualitative part and a quantitative part. The qualitative part is a decomposable, or *chordal*, graph; the quantitative part is a set of marginal distributions on the vertex sets of the cliques of this graph. The computational architecture for the algorithm derives from the qualitative part of this decomposable belief network and is termed a *junction tree* [Jensen *at al.*, 1990a]. In the junction-tree architecture, the cliques of the decomposable graph are the autonomous objects and the clique intersections give rise to the communication channels. From a decomposable belief network, a probability of interest is computed by further marginalisation of an appropriate marginal distribution. Processing evidence is performed per clique and by message-passing between neighbouring cliques in the junction tree.

The various algorithms proposed for probabilistic inference with a belief network also differ with respect to their computational complexity. We note that probabilistic inference with belief networks without any topological restrictions is known to be NP-hard [Cooper, 1990]. All algorithms proposed therefore have an exponential worst-case computational complexity. However, the algorithms can be shown to behave polynomially under certain restrictions. These restrictions concern the topology of a belief network's digraph and differ among the various algorithms. For example, digraphs for which Pearl's algorithm behaves polynomially may show exponential behaviour of Lauritzen and Spiegelhalter's algorithm, and vice versa [Suermondt & Cooper, 1991]. In general, however, the sparser the digraph of a belief network, the better most algorithms perform. Experience with constructing belief networks for real-life applications so far has indicated that a belief network's digraph in fact tends to be rather sparse.

5 Building a Bayesian Belief Network

The belief network framework generally is used for applying probability theory for reasoning with uncertainty in knowledge-based systems. For employing the framework for a real-life domain of application, relevant knowledge of the domain at hand is represented in the belief network formalism; the basic algorithms of the framework are taken as building blocks for shaping the system's intelligent problem-solving behaviour. In this section we address the task of building a Bayesian belief network for a domain of application. Before doing so, we would like to emphasise that as the belief network framework has not been around for a long time, methodologies for building belief networks do not yet abound.

Building a Bayesian belief network for a domain of application involves various tasks. The first of these is to identify the *variables* that are of importance in the domain at hand, along with their possible values. Identifying the important domain variables is typically performed with the help of one or more domain experts. This task is not specific for building a belief network, but instead is quite common in engineering knowledge-based systems. A knowledge engineer can therefore make use of the various elicitation techniques designed for engineering knowledge-based systems in general. We will not further elaborate on this task here and confine ourselves to emphasising that it needs to be performed with care since the domain variables that are modeled in the belief network demarcate the scope of the resulting system.

Once the important domain variables have been identified, each of them needs to be expressed as a *statistical variable* to allow for inclusion in the belief network in the making. A statistical variable is characterised by its values being *mutually exclusive* and *collectively exhaustive*; furthermore, to allow for inclusion in a belief network, a statistical variable has to take its value from a *finite* set of discrete values. Only if the set of values of a domain variable exhibits these properties, can the variable be included in the network as it is. A single-valued domain variable taking its value from an *infinite* set of values, for example, cannot be expressed directly as a statistical variable for inclusion in a belief network. The variable's set of values has to be discretised, that is, split up into a finite number of mutually exclusive subsets of values which subsequently are taken as the variable's new values. Also, a domain variable taking *multiple* values from a finite set of values cannot be expressed directly as a statistical variable, since its values are not mutually exclusive. The variable's values then have to be redefined to render them mutually exclusive and collectively exhaustive; alternatively, the domain variable can be decomposed into several variables which subsequently are modeled separately in the belief network in the making.

Once the domain variables of importance have been identified and expressed as statistical variables, the construction of the qualitative part of the belief network in the making can commence. Formally, for constructing the qualitative part of the network, the independence relation of the joint probability distribution on the variables discerned has to be identified and represented in an acyclic digraph. In general practice, however, the digraph is constructed directly without explicitly identifying all relevant independences.

For most domains of application, the qualitative part of a belief network has to be *hand-crafted* with the help of one or more domain experts. For eliciting the topology of the digraph of the network, often the concept of *causality* is used as a heuristic guiding rule during the interview with a domain expert; typical questions asked are "What could cause this effect ?", "What manifestations could this cause have ?" [Henrion, 1989]. The thus elicited causal relations among the variables discerned are easily expressed in graphical terms by taking the direction of causality for directing the arcs between related variables; this graphical representation can then be taken as a basis for feedback to the domain expert for further refinement. Building on the concept of causality has the advantage that domain experts are allowed to express their knowledge in either the *causal* or *diagnostic* direction. Since they are allowed to express their knowledge in a form they feel comfortable with, the experts' statements tend to be quite robust. Yet, not every influential relationship among variables can be interpreted as causal. If a non-causal influential relationship comes to the fore during an interview, a more elaborate analysis of the independences involved is required before it can be expressed in graphical terms. Also, causality is not a well-understood concept and therefore may leave room for multiple interpretations. We would like to note, however, that the task of eliciting relationships among variables from domain experts is not reserved for building Bayesian belief networks: the elicitation task has parallels with engineering knowledge-based systems in general for which several methodologies have been developed.

For some domains of application, the construction of the qualitative part of the belief network in the making can be performed automatically by exploiting carefully collected data. In various domains, data has been collected and maintained over numerous years of everyday problem solving. Such a data collection generally contains highly valuable information about the relationships among the variables discerned, albeit implicitly. The basic idea of an algorithm for *learning* the qualitative part of a belief network from data is to distill this information from the data collection and exploit it for constructing a digraph. Modern algorithms for learning the qualitative part of a belief network from data typically generate various different acyclic digraphs and compare these as to their ability to describe or explain the data at hand [Cooper & Herskovits, 1992, Lam & Bacchus, 1994]. For comparing digraphs, a learning algorithm makes use of a *quality measure*. A quality measure is a function that assigns to a digraph a numerical value expressing how well this digraph fits the data. The basic purpose of the algorithm now is to select from among all acyclic digraphs a digraph with *highest quality*. As it is not feasible from a computational point of view to generate all possible digraphs and compute their qualities, a learning algorithm incorporates a *search heuristic* for searching the set of all acyclic digraphs implicitly for digraphs that are *likely* to have a high quality; only for these digraphs is the quality given the data actually computed.

After the qualitative part of the belief network in the making has been constructed, is its quantitative part specified. Specifying the quantitative part amounts to defining the probability assessment functions for the variables modeled in the network. The task of assessing all required probabilities tends to be by far the hardest task in belief network building. In most domains, at least some information is readily available for this task, be it from literature or from domain experts. Although literature on the domain of application often provides abundant probabilistic information, it unfortunately is very seldom directly amenable to encoding in the belief network in the making: the information typically is not complete, it concerns variables that are not causally related, and so on. An additional, commonly found problem that prohibits direct use of probabilistic information from literature is that the characteristics of the population from which the information is derived, is not properly described or deviates seriously from the characteristics of the population for which the belief network is being developed [Korver & Lucas, 1993]. Literature, however, is not the only source of probabilistic information. In principle, probability assessments may also be obtained from data or from (other) models of domain knowledge. Unfortunately, experience learns that even if comprehensive data collections and models of domain knowledge are available, they very seldom contribute to the quantification task [Jensen, 1995, Korver & Lucas, 1993]. As a consequence, a large number of probabilities will have to be assessed by domain experts. The field of decision analysis offers various methods for elicitation of judgemental probabilities from experts [von Winterfeldt & Edwards, 1986]. These methods are designed to avert to at least some extent problems of bias and poor calibration typically found in human probability assessment. Straightforward application of these methods to belief network quantification, however, often is hampered by the large size and complex dependence structure of the belief network at hand. Decision-analytic methods therefore will have to be supplemented with methods tailored to belief network quantification; such methods are just beginning to arise [Druzdzel & van der Gaag, 1995]. Elicitation of probabilities from domain experts will nevertheless remain a very hard and time-consuming task.

In the foregoing, we have addressed the various tasks involved in building a Bayesian belief network separately. In practice, however, building a belief network is a cyclic process that iterates over the various tasks until a network results that is deemed satisfactory for the domain of application at hand. To conclude, we would like to stress that building a belief network requires careful tradeoff between the desire for a large and rich model to obtain accurate results on the one hand, and the cost of construction and maintenance and the complexity of probabilistic inference on the other hand.

6 Problem Solving with Belief Networks

Since its introduction, the belief network framework has been applied in knowledge-based systems for a wide range of complex real-life problems, most notably in the medical domain [Andreassen *et al.*, 1987, Heckerman *et al.*, 1992, Andreassen *et al.*, 1991, Bellazzi *et al.*, 1991, Shwe *et al.*, 1991]. As experience with applying the framework is building, it becomes apparent that, although the framework offers many advantages over earlier approaches to automated reasoning with uncertainty, it lacks with regard to *intelligent control over reasoning.* The provision of control over reasoning is generally considered one of the main contributions of artificial intelligence research to automated reasoning: knowledge-based systems thank their success to a large extent to their ability to apply specialised knowledge for pruning search spaces and for selectively gathering evidence. Since control over reasoning is a prerequisite for arriving at problem-solving behaviour that is satisfactory both from a computational and a user's point of view, it is not surprising that providing for control is an important issue in present-day belief-network research. In this section, we briefly address this issue.

6.1 A Problem-Solving Architecture

The main purpose of exerting control over reasoning is to shape efficient and intelligent problem-solving behaviour. Exerting control involves monitoring and reflecting upon the reasoning process as it develops and taking decisions as to how it should proceed. To this end, strategic knowledge about the domain at hand is employed. As this knowledge may be non-probabilistic in nature, control over belief-network reasoning generally cannot be implemented in the framework in itself. We therefore propose embedding the belief network framework in a general *problem-solving architecture* [van der Gaag & Wessels, 1993].

Our belief-network problem-solving architecture is composed of two *layers.* The first layer offers the belief network formalism and a variety of associated algorithms. The algorithms in this layer are characterised by their operating on a belief network directly: various algorithms for probabilistic inference are comprised in the layer as are algorithms for example for reading independences from the qualitative part of a network. In the sequel, we will call this layer the *probabilistic layer* of the architecture. The second layer of our problem-solving architecture is designed to provide for control over reasoning — it is termed the *control layer.* This layer offers a variety of methods for control for different types of problem solving and provides formalisms for representing the additional knowledge used by these methods. The layer for example offers methods for selectively gathering evidence for diagnostic applications as well as methods for intelligently pruning and focusing belief-network inference. These control methods are the basic building blocks for shaping complex, domain-dependent problem-solving behaviour. The two layers of our architecture are strictly separated and communicate in a highly restricted fashion. The control layer queries the probabilistic layer for information about the represented joint probability distribution and the evidence entered so far, and, based upon this information, takes strategic decisions as to how to proceed. The probabilistic layer computes and returns the information it is asked for by the control layer.

Our problem-solving architecture explicitly separates probabilistic reasoning from control over reasoning. Several advantages arise from such an explicit separation. A belief network can be developed and refined, without being hampered by any algorithmic issues. Moreover, the representation of the joint probability distribution on the domain at hand is not obscured by non-probabilistic knowledge. In addition, a belief network can be re-used in different contexts for different purposes; a similar observation holds for the methods of control comprised in the control layer of the architecture. We would like to note that our architecture also provides for modeling decision problems. Knowledge about viable decisions and the preferences over their consequences involved in a decision problem are represented in the control layer, along with methods for solving the problem; the probabilistic layer comprises a belief network that is used as a background knowledge base for providing the probabilities required. As such our problem-solving architecture is more flexible than the influence diagram framework. To conclude, we would like to mention that the idea of a meta-level problem-solving architecture is not a new one — in fact, the idea pervades many areas of artificial intelligence research.

6.2 Controlling Diagnostic Reasoning

As an example method for control over reasoning offered by the control layer of our problem-solving architecture, we will discuss a simple method for selective gathering of evidence for diagnostic problem solving with a belief network.

In diagnostic problem solving, the objective is to identify a most likely explanation for a problem under consideration — this explanation then is the *diagnosis* of the problem. Establishing a diagnosis is achieved by gathering information about the manifestations of the problem at hand by applying *tests* to the problem. In most domains, it is not necessary to collect evidence on all possible manifestations before an accurate diagnosis is reached: information from only a few tests generally suffices. Moreover, it often is not desirable to apply all tests available as testing may be costly or damaging. In diagnostic problem solving, therefore, tests are not applied as a matter of course but instead are selected carefully. *Selective evidence gathering*, or *test planning*, now amounts to selecting the most useful tests to apply to a problem under consideration.

In essence, selective evidence gathering is concerned with three tasks. The first of these is to select the test that is expected to yield the most useful information in the context of the evidence that is already available. When a test has been selected, the user is requested to apply the test and to enter the evidence yielded. The second task of selective evidence gathering is to process this evidence. The third task is to decide whether enough evidence has been obtained as yet to confirm a diagnosis to sufficient extent. If still further information is required, the three tasks are executed recursively. We now take a closer look at these tasks in view of diagnostic problem solving with a belief network. While the belief network framework offers algorithms for computing probabilities and for processing evidence, thus providing for the second task of selective evidence gathering, it does not provide for valuing and selecting tests nor for deciding when to stop gathering information: these tasks involve knowledge that cannot be expressed in the belief network formalism and require computations beyond probabilistic inference. These tasks therefore are provided for by the control layer of our problem-solving architecture.

In diagnostic problem solving, the variables discerned in the domain at hand play different roles; for example, some variables represent test outcomes, others represent unobservable, intermediate process states. For distinguishing between these different roles, we discern the following types of vertex in the digraph of a belief network [Henrion, 1989]: a *hypothesis vertex* represents one or more (mutually exclusive) hypotheses or disorders; an *evidence vertex* represents a variable whose value can be obtained by testing; all other vertices are *intermediate vertices*. In the sequel, we take the set of vertices of the digraph G of a belief network to be partitioned into three mutually disjoint sets of vertices: $H(G) = \{H_1, \ldots, H_n\}$, $n \geq 1$, is the set of hypothesis vertices; the set of evidence vertices is denoted as $E(G) = \{E_1, \ldots, E_m\}$, $m \geq 1$; $I(G)$ is the set of intermediate vertices. The roles of the various vertices are modelled in the control layer of our problem-solving architecture and are not known to the probabilistic layer. In addition to knowledge concerning the roles of the vertices discerned, the control layer also specifies the additional knowledge required for assessing for each test the (expected) usefulness of information yielded by testing, and the extra knowledge involved in deciding when to stop gathering information.

For selective evidence gathering in diagnostic problem solving with a belief network, generally two simplifying assumptions are made. First, a *myopic* approach to evidence gathering is taken, that is, evidence vertices to acquire information on are selected one by one. It is conceivable that in practical applications a non-myopic approach in which vertices are selected groupwise outperforms any method based on a myopic approach. Naively adopting a non-myopic approach, however, poses unsurmountable problems concerning computational complexity. The second simplifying assumption generally made is that the belief network at hand comprises *one* hypothesis vertex only, that is, it is assumed that all hypotheses discerned in the domain are mutually exclusive. Note that this assumption prohibits reasoning about multiple interacting disorders. Relaxing this assumption and straightforwardly applying selective evidence gathering in view of a set of hypothesis vertices also causes serious computational problems, since then all possible combinations of values for all hypothesis vertices have to be considered. We will here equally take up the two assumptions mentioned above. We would like to note, however, that recent research results indicate that the simplifying assumptions may be eased to some extent [Heckerman *et al.*, 1993, van der Gaag & Wessels, 1994].

Selective evidence gathering for diagnostic problem solving with a belief network may now be envisioned as outlined below in pseudo-code. The evidence-gathering procedure takes the digraph G of a belief network and the set E of all yet uninstantiated evidence vertices for its input and yields a diagnosis D for its output.

```
procedure evidence-gathering($G$,$E$,$D$)
  enough := $false$;
  while $E \neq \varnothing$ and not enough do
      dependent-vertices($H$,$E$,$E'$);
      if $E' \neq \varnothing$ then
          select-vertex($E'$,$E_j$);
          enter-evidence($E_j$);
          $E := E \setminus \{E_j\}$;
          enough := verify-enough()
      else enough := $true$
  od;
  diagnosis(D)
end
```

In principle, for selecting from a belief network an appropriate vertex to acquire information on, each yet uninstantiated evidence vertex has to be examined as to the (expected) usefulness of information yielded. To this end, several probabilities are computed from the belief network. These probabilities may reveal that for some of the uninstantiated evidence vertices, entering information has no influence whatsoever on the probabilities of the values of the hypothesis vertex and therefore is utterly useless in view of establishing a diagnosis. This property holds for all vertices that are independent of the hypothesis vertex given the evidence obtained so far. Now recall that the belief network formalism allows for identifying independences from the digraph of a network without having to resort to probabilistic computations. In the main evidence-gathering procedure, this property is exploited to save on the number of probabilities that has to be computed from the network. The dependent-vertices procedure is called upon to determine from a set E of uninstantiated evidence vertices the subset E' of those vertices that are not d-separated from the hypothesis vertex H given the evidence entered so far. For selecting an appropriate evidence vertex to acquire information on now only the vertices comprised in this set E' are examined. Here, we will not further elaborate on the dependent-vertices procedure; for the computational issues involved, the reader is referred to [Geiger *et al.*, 1990]. We would like to note that although the dependent-vertices procedure is called from the control layer of our problem-solving architecture, it itself is comprised in the probabilistic layer.

Once the set E' of relevant uninstantiated evidence vertices has been determined, the select-vertex procedure selects from this set the evidence vertex to best acquire information on. For discriminating between the various evidence vertices, this procedure employs a *utility function.* Such a function assigns to each value of every evidence vertex a numerical quantity expressing the desirability, or *utility*, of obtaining this value. The utility functions in use for selective evidence gathering differ considerably in the way they value information [Ben-Bassat, 1978, Glasziou & Hilden, 1989]. A utility function may be based on probabilistic information only and not involve any other information about the domain at hand. Yet, it may also incorporate non-probabilistic issues such as the cost of obtaining. For selecting an appropriate evidence vertex, the select-vertex procedure may employ any such utility function. As an example, we consider here a very simple utility function tailored to binary variables [van der Gaag & Wessels, 1993]: the *linear-value utility function* u is defined by

$$u(E_i) = |\Pr(h \mid c) - \Pr(h \mid c \wedge E_i)|$$

for each evidence vertex $E_i \in E'$, where H is the hypothesis vertex and c denotes the conjunction of all evidence obtained so far. Note that for an uninstantiated evidence vertex E_i, the difference between $\Pr(h \mid c)$ and $\Pr(h \mid c \wedge e_i)$ indicates the confidence gained in the hypothesis h if the evidence $E_i = true$ is observed; an analogous observation holds for the evidence $E_i = false$. The usefulness of acquiring information on an evidence vertex, however, does not depend on a single value as it is uncertain which test result will be yielded for the problem at hand. For examining an evidence vertex, therefore, the utilities of its separate values are weighted with the probabilities that these values will be found. The result models the *expected utility* of acquiring information on the vertex. When employing the linear-value utility function u, the expected utility $\hat{u}$ for an evidence vertex $E_i \in E'$ is computed from

$$\hat{u}(E_i) = \sum_{c_{E_i}} \Pr(c_{E_i} \mid c) \cdot u(c_{E_i})$$

To select the evidence vertex to best acquire information on, the select-vertex procedure computes the expected utilities for all relevant uninstantiated evidence vertices and then selects the vertex with highest expected utility.

Example 6.1 Consider once more the belief network shown in Figure 1 representing some knowledge concerning the diagnosis of acute cardiac disorders. Suppose that the vertices from the network are assigned the following roles: vertex M is the hypothesis vertex, and vertices S, P, and F are evidence vertices — there are no intermediate vertices in the network. Initially, no evidence is available. The select-vertex procedure selects the evidence vertex to best acquire information on from among the vertices S, P, and F. To this end, it computes their respective expected utilities as outline before. For example, for vertex P the utilities

$$u(p) = |\Pr(m) - \Pr(m \mid p)| = 0.25$$

$$u(\neg p) = |\Pr(m) - \Pr(m \mid \neg p)| = 0.375$$

are computed, and from these the expected utility

$$\hat{u}(P) = \Pr(p) \cdot u(p) + \Pr(\neg p) \cdot u(\neg p) = 0.3$$

From the three evidence vertices, vertex P appears to have the highest expected utility. The select-vertex procedure therefore selects this vertex to acquire information on. □

Once an appropriate evidence vertex has been selected, the enter-evidence procedure prompts the user for a value for this vertex. The value obtained from the user is entered and subsequently processed in the belief network at hand by the basic algorithms for probabilistic inference offered by the probabilistic layer of the problem-solving architecture. Note that the while-loop of the evidence-gathering procedure yields a sequence of prompts to the user concerning various evidence vertices.

The last task of selective evidence gathering is to investigate whether enough evidence has been collected to justify a decision to stop further gathering of information. For this task, a *stopping criterion* is employed. Such a stopping criterion may be based on several different principles. The principle of *sufficiency of confirmation* is to stop evidence gathering as soon as a diagnosis has been confirmed to sufficient extent by the available information: the probability of a (tentative) diagnosis is compared with a pre-set threshold value, and if this probability has surpassed the threshold value and is expected not to drop considerably, evidence gathering is stopped. Note that such a stopping criterion is based on probabilistic information only. Another principle a stopping criterion may be based upon is the principle of *sufficiency of information*. This principle is to stop evidence gathering if the expected utilities of all remaining evidence vertices have dropped below a pre-set threshold value; pursuing evidence gathering then is expected not to further contribute to establishing a diagnosis. A stopping criterion based on this principle may involve both probabilistic and non-probabilistic information from the domain at hand. In the evidence-gathering procedure the two principles are combined. The principle of sufficiency of information is seen in the condition of the main while-loop of the procedure: evidence gathering is stopped if all remaining uninstantiated evidence vertices are independent of the hypothesis vertex given the evidence obtained so far. The verify-enough function completes the stopping criterion by implementing a test on sufficiency of confirmation. We will not further elaborate here on the verify-enough function.

7 Conclusions

Probabilistic reasoning with the belief network framework is an exciting research area. First and foremost, the belief network framework may be looked upon as a mathematically sound computational framework for probabilistic inference. From this point of view, we have addressed the algorithms offered by the framework. We have argued that although these algorithms have an exponential worst-case time complexity, they tend to behave polynomially for most real-life belief networks. However, as applications of the framework grow larger, the belief networks involved increase in size accordingly. Networks comprising hundreds or even thousands of vertices are no exception. For belief networks of this size, the basic algorithms for probabilistic inference inevitably slow down problem solving despite their polynomial behaviour. Modern belief network research therefore aims at developing more efficient algorithms. Efficiency is sought after in many different ways: existing algorithms are further optimised, new algorithms are designed for example based on simulation techniques, yet other optimisations derive from knowledge-based pruning and focusing.

The belief network framework may also be looked upon as a framework for building knowledge-based systems. In fact, experience with developing applications of the framework is progressing rapidly. From this point of view, we have addressed the issue of building a belief network for a domain of application. In many respects, building a belief network resembles engineering a knowledge-based system more in general. Available general knowledge-engineering methodologies, however, will have to be supplemented with methodologies tailored to belief-network building. Such methodologies are now emerging. As experience with applying the belief network framework is building, the need for methods for knowledge-based control over reasoning is beginning to manifest itself. We have briefly addressed this issue and outlined shaping diagnostic problem solving with a belief network. Research on the provision of control over belief-network reasoning has scarcely begun and different control methods as demanded by various types of problem solving have yet to be designed.

Acknowledgement

A major part of this paper has been presented at the Sofsem'94 Winterschool. We further acknowledge that Section 5.2 is based to a large extent on joint work with Maria Wessels. To conclude, we would like to thank Peter Lucas for his valuable comments on an earlier draft of this paper.

References

[Andreassen *et al.*, 1987] S. Andreassen, M. Woldbye, B. Falck, and S.K. Andersen. MUNIN - A causal probabilistic network for interpretation of electromyographic findings. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, 1987, pp. 366 – 372.

[Andreassen *et al.*, 1991] S. Andreassen, R. Hovorka, J. Benn, K.G. Olesen, and E.R. Carson. A model-based approach to insulin adjustment. In: M. Stefanelli, A. Hasman, M. Fieschi, and J. Talmon. *Proceedings of the Third Conference on Artificial Intelligence in Medicine.* Lecture Notes in Medical Informatics 44, Springer Verlag, Berlin, 1991, pp. 239 – 248.

[Bellazzi *et al.*, 1991] R. Bellazi, C. Berzuini, S. Quaglini, D.J. Spiegelhalter, and M. Leaning. Cytotoxic chemotherapy monitoring using stochastic simulation on graphical models. In: M. Stefanelli, A. Hasman, M. Fieschi, and J. Talmon. *Proceedings of the Third*

Conference on Artificial Intelligence in Medicine. Lecture Notes in Medical Informatics 44, Springer Verlag, Berlin, 1991, pp. 227 – 238.

[Ben-Bassat, 1978] M. Ben-Bassat. Myopic policies in sequential classification. *IEEE Transactions on Computers*, vol. C-27, 1978, pp. 170 – 174.

[Bruza & van der Gaag, 1994] P.D. Bruza and L.C. van der Gaag. Index expression belief networks for information disclosure. *International Journal of Expert Systems: Research and Applications*, vol. 7, 1994, pp. 107 – 138.

[Buchanan & Shortliffe, 1984] B.G. Buchanan and E.H. Shortliffe. *Rule-based Expert Systems. The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, Reading, Massachusetts, 1984.

[Cheeseman, 1988] P. Cheeseman. An inquiry into computer understanding. *Computational Intelligence*, vol. 4, 1988, pp. 58 – 66.

[Cooper, 1990] G.F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, vol. 42, 1990, pp. 393 – 405.

[Cooper & Herskovits, 1992] G.F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, vol. 9, 1992, pp. 309 – 347.

[de Dombal *et al.*, 1972] F.T. de Dombal, D.J. Leaper, J.R. Staniland, A.P. McCann, and J.C. Horrocks. Computer-aided diagnosis of acute abdominal pain. *British Medical Journal*, vol. 2, 1972, pp. 9 – 13.

[de Dombal *et al.*, 1974] F.T. de Dombal, D.J. Leaper, J.C. Horrocks, J.R. Staniland, and A.P. McCann. Human and computer-aided diagnosis of abdominal pain: further report with emphasis on the performance of clinicians. *British Medical Journal*, vol. 4, 1974, pp. 376 – 380.

[Druzdzel & van der Gaag, 1995] M.J. Druzdzel, L.C. van der Gaag. Elicitation of probabilities for belief networks: combining qualitative and quantitative information. *Eleventh Conference on Uncertainty in Artificial Intelligence*, 1995, pp. 141 – 148.

[Geiger *et al.*, 1990] D.E. Geiger, T. Verma, and J. Pearl. *d*-separation: from theorems to algorithms. In: M. Henrion, R.D. Shachter, L.N. Kanal, and J.F. Lemmer. *Uncertainty in Artificial Intelligence 5*, Elsevier Science Publishers, Amsterdam, 1990, pp. 139 – 148.

[Glasziou & Hilden, 1989] P. Glasziou and J. Hilden. Test selection measures. *Medical Decision Making*, vol. 9, 1989, pp. 133 – 141.

[Gorry & Barnett, 1968] G.A. Gorry and G.O. Barnett. Experience with a model of sequential diagnosis. *Computers and Biomedical Research*, vol. 1, 1968, pp. 490 – 507.

[Heckerman *et al.*, 1992] D.E. Heckerman, E.J. Horvitz, and B.N. Nathwani. Toward normative expert systems. Part 1: The Pathfinder project. *Methods of Information in Medicine*, vol. 31, 1992, pp. 90 – 105.

[Heckerman *et al.*, 1993] D.E. Heckerman, E.J. Horvitz, and B. Middleton. An approximate nonmyopic computation for value of information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, 1993, pp. 292 – 298.

[Henrion, 1989] M. Henrion. Some practical issues in constructing belief networks. In: L.N. Kanal, T.S. Levitt, and J.F. Lemmer. *Uncertainty in Artificial Intelligence 3*, North-Holland Publishers, Amsterdam, 1989, pp. 161 – 173.

[Horvitz *et al.*, 1988] E.J. Horvitz, J.S. Breese, and M. Henrion. Decision theory in expert systems and artificial intelligence. *International Journal of Approximate Reasoning*, vol. 2, 1988, pp. 247 – 302.

[Howard & Matheson, 1981] R.A. Howard and J.E. Matheson. Influence diagrams. In: R.A. Howard and J.E. Matheson. *Readings on the Principles and Applications of Decision Analysis II*, 1981, pp. 721 – 762.

[Jackson, 1990] P. Jackson. *Introduction to Expert Systems.* Addison-Wesley, Wokingham, 1990.

[Jensen *at al.*, 1990a] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, vol. 4, 1990, pp. 269 – 282.

[Jensen *et al.*, 1990b] F.V. Jensen, J. Nielsen, and H.I. Christensen. *Use of Causal Probabilistic Networks as High Level Models in Computer Vision.* Technical Report R-90-39, University of Aalborg, Denmark, 1990.

[Jensen, 1995] A.L. Jensen. Quantification experience of a DSS for mildew management in winter wheat. In: M.J. Druzdzel, L.C. van der Gaag, M. Henrion, and F.V. Jensen. *Working Notes of the Workshop on Building Probabilistic Networks: Where Do the Numbers Come From ?*, 1995, pp. 23 –31.

[Korver & Lucas, 1993] M. Korver, P.J.F. Lucas. Converting a rule-based expert system into a belief network. *Medical Informatics*, vol. 18, 1993, pp. 219 – 241.

[Lam & Bacchus, 1994] W. Lam and F. Bacchus. Learning Bayesian belief networks, an approach based on the MDL principle. *Computational Intelligence*, vol. 10, 1994, pp. 269 – 293.

[Lauritzen & Spiegelhalter, 1988] S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, vol. 50, 1988, pp. 157 – 224.

[Lucas & van der Gaag, 1991] P.J.F. Lucas and L.C. van der Gaag. *Principles of Expert Systems.* Addison-Wesley, Wokingham, 1991.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems. Networks of Plausible Inference.* Morgan Kaufmann, Palo Alto, 1988.

[Shachter, 1986] R.D. Shachter. Evaluating influence diagrams. *Operations Research*, vol. 34, 1986, pp. 871 – 882.

[Shafer & Pearl, 1990] G. Shafer and J. Pearl. *Readings in Uncertain Reasoning.* Morgan Kaufmann, Palo Alto, 1990.

[Shortliffe & Buchanan, 1984] E.H. Shortliffe and B.G. Buchanan. A model of inexact reasoning in medicine. In: B.G. Buchanan and E.H. Shortliffe. *Rule-based Expert Systems. The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, Reading, Massachusetts, 1984, pp. 233 – 262.

[Shwe *et al.*, 1991] M.A. Shwe, B. Middleton, D.E. Heckerman, M. Henrion, E.J. Horvitz, H.P. Lehmann, and G.F. Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base I: the probabilistic model and inference algorithms. *Methods of Information in Medicine*, vol. 30, 1991, pp. 241 – 255.

[Suermondt & Cooper, 1990] H.J. Suermondt and G.F. Cooper. Probabilistic inference in multiply connected belief networks using loop cutsets. *International Journal of Approximate Reasoning*, vol. 4, 1990, pp. 283 – 306.

[Suermondt & Cooper, 1991] H.J. Suermondt and G.F. Cooper. A combination of exact algorithms for inference on Bayesian belief networks. *International Journal of Approximate Reasoning*, vol. 5, 1991, pp. 521 – 542.

[van der Gaag, 1994] L.C. van der Gaag. A pragmatic view of the certainty factor model. *The International Journal of Expert Systems: Research and Applications*, vol. 7, 1994, pp. 289 – 300.

[van der Gaag & Wessels, 1993] L.C. van der Gaag and M.L. Wessels. Selective evidence gathering for diagnostic belief networks. *AISB Quarterly*, no. 86, 1993, pp. 23 – 34.

[van der Gaag & Wessels, 1994] L.C. van der Gaag and M.L. Wessels. Multiple-disorder diagnosis with belief networks. *Proceedings of the Fifth International Workshop on Principles of Diagnosis — DX'94*, 1994, pp. 343 – 351.

[Warner *et al.*, 1961] H.R. Warner, A.F. Toronto, L.G. Veasy, and R. Stephenson. A mathematical approach to medical diagnosis: application to congenital heart disease. *Journal of the American Medical Association*, vol. 177, 1961, pp. 177 – 183.

[von Winterfeldt & Edwards, 1986] D. von Winterfeldt and W. Edwards. *Decision Analysis and Behavioral Research.* Cambridge University Press, New York, 1986.