An Algorithm for Finding Minimum d-Separating Sets in Belief Networks*

Silvia ACID, Luis M. de CAMPOS
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada
18071-Granada, Spain

Abstract

The criterion commonly used in directed acyclic graphs (dags) for testing graphical independence is the well-known d-separation criterion. It allows us to build graphical representations of dependency models (usually probabilistic dependency models) in the form of belief networks, which make possible an easy interpretation and management of independence relationships, without reference to numerical parameters (conditional probabilities). In this paper we study the following combinatorial problem: to find the minimum d-separating set for two nodes in a dag. This set would represent the minimum information necessary to prevent these two nodes to influence each other. The solution of this basic problem and of some of its extensions can be useful in several ways, as we will see later. Our solution is based on a two-steps process: first, we reduce the original problem to the simpler one of finding a minimum separating set in an undirected graph, and second, we develop an algorithm for solving it.

*This work has been supported by the DGICYT under Project PB92-0939

1 Introduction

Belief networks have become common knowledge representation tools capable of representing and handling independence relationships. The reasons for the success of belief networks are their ability to efficiently perform correct inferences by using local computations, and their reduced storage needs. Indeed, independence can modularize knowledge in such a way that we only need to consult the pieces of information relevant to the specific question in which we are interested, instead of having to explore a whole knowledge base; moreover, the storage requirements of, for example, a joint probability distribution are usually excessive, whereas the memory requirements of a suitable factorization of this distribution, taking into account the independence relationships, may be much smaller.

The capacity of belief networks for representing independence statements is based on the well-known graphical independence criterion called d-separation: if the dependency model satisfies certain properties then we can assert and use many (or all) of the independencies which are true in the model by checking for d-separation statements in the corresponding dag. However, d-separation is quite a subtle concept, more difficult to manage and interpret than its counterpart for undirected graph, i.e., the separation criterion. In this paper we propose and solve an optimization problem related to d-separation. The basic problem can be formulated as follows: *given a pair of nodes x and y in a dag G, find the set of nodes with minimum size that d-separates x and y*. An obvious extension of this basic problem is to find the minimum set d-separating two sets of nodes instead of two single nodes. A second extension of the basic problem is the following: given two sets of nodes X and Y, and given a third set of nodes Z, find the minimum set, say S, that together with Z, d-separates X and Y. It is worth noting that an algorithm for solving this last problem may also be used for testing d-separation: if the minimum set S is empty, then X and Y are d-separated by Z; otherwise, they are not d-separated.

The first question we have to answer is: apart from the possible theoretical interest that this problem may have, does it have some practical utility? In our opinion the answer is positive. In general, the solution of this problem represents the minimum information which is necessary to know, in order to prevent two sets of nodes to influence each other, either in absence of any other information (first extension), or in presence of some previous knowledge

(second extension of the basic problem). This can be useful in several ways:

Let us consider the following situation: we have a database containing instances of some variables in a given domain (or a joint probability distribution for these variables), and we also have a dag which is guessed to be an appropriate graphical representation for this domain. This means that the d-separation statements in the dag should correspond with true conditional independence statements in the domain. In order to test this assumption we could select, for example, pairs of non adjacent nodes in the dag, next finding d-separating sets for these pairs, and then testing the corresponding conditional independencies in the database (or in the joint distribution). However, the complexity of testing these conditional independence statements increases exponentially (and, in the case of using a database, the reliability of the result decreases) with the size of the d-separating sets. So, it may be quite interesting to select d-separating sets of size as small as possible, instead of using some more obvious sets (as, for example, the parent set of one of the two nodes in the pair). This idea has been recently used for the design of an algorithm for learning belief networks from databases [1].

Another possible application is the following: consider an already constructed belief network, and suppose that we are interested in obtaining information about a given variable (for example, a classification variable). Let us suppose also that we are trying to gather information relevant for this variable, and we have to decide what other variables we should know, in order to improve our information about the variable of interest. The problem arises when knowing the value of each variable has a different cost, and the differences may be significant (for example, some variables may represent more or less expensive medical tests). In these circumstances, if we can find a minimum (and inexpensive) set d-separating the variable of interest from other expensive variable (or set of variables), then we could avoid the observation of this variable, by replacing it by the d-separating set, without any decrease in the quality of the obtained information.

So, we think that an algorithm for solving our basic problem and its extensions may be an useful tool during the phases of construction and use of belief networks.

The rest of the paper is organized as follows: in Section 2, we briefly describe several concepts which are basic for subsequent development, such as the separation and d-separation criteria, as well as the relationship between them proposed by Lauritzen et al. [9]. Section 3 shows how to reduce our

original optimization problem about d-separation in dags to a simpler equivalent problem involving separation in undirected graphs. In Section 4 we propose an algorithm that, taking into account the previous results, finds a minimum d-separating set for any two non adjacent nodes in a dag. We also analize how to cope with the two proposed extensions of the basic problem. Finally, Section 5 contains the concluding remarks.

2 Preliminaries

In this Section, we are going to describe the notation and some basic concepts used throughout the paper.

A *Dependency Model* [10] is a pair $M = (\mathcal{U}, I)$, where $\mathcal{U}$ is a finite set of elements or variables, and $I(.,.|.)$ is a rule that assigns truth values to a three place predicate whose arguments are disjoint subsets of $\mathcal{U}$. Single elements of $\mathcal{U}$ will be denoted by standard or Greek lowercase letters, such as x, y, z, α, $\beta \ldots$, whereas subsets of $\mathcal{U}$ will be represented by capital letters, such as X, Y, $Z \ldots$ The intended interpretation of $I(X, Y|Z)$ (read X is independent of Y given Z) is that having observed Z, no additional information about X could be obtained by also observing Y. For example, in a probabilistic model [6, 9, 12], $I(X, Y|Z)$ holds if and only if

$$P(\mathtt{x}|\mathtt{z}, \mathtt{y}) = P(\mathtt{x}|\mathtt{z}) \text{ whenever } P(\mathtt{z}, \mathtt{y}) > 0,$$

for every instantiation $\mathtt{x}$, $\mathtt{y}$ and $\mathtt{z}$ of the sets of variables X, Y and Z. However, dependency models are applicable to many situations far beyond probabilistic models [2, 3, 4, 10, 11, 14]. The study of the concept of conditional independence in probability theory and that of embedded multivalued dependence in database theory has resulted in the identification of several properties that may be reasonable to demand of any relationship which attempts to capture the intuitive notion of independence. These properties are the well-known *graphoid axioms* [10].

On the other hand, a graphical representation of a dependency model $M = (\mathcal{U}, I)$ is a graph, $G = (\mathcal{U}, \mathcal{E})$, where $\mathcal{E}$ is the set of arcs or edges of G, such that the topology of G reflects some properties of I. The way we relate independence assertions in I with some topological property of a graph depends on the kind of graph we use; this property is *separation* for the case of undirected graphs [8, 10] and *d-separation* for directed acyclic ones [10, 13]:

- *Separation*: Given an undirected graph G, two subsets of nodes, X and Y, are said to be separated by the set of nodes Z, and this is denoted by $\langle X, Y|Z\rangle_G^s$, if Z intercepts all chains between the nodes in X and those in Y, or, in other words, if the removal of the set of nodes Z from the graph together with their associated edges, disconnects the nodes in X from those in Y.

- *d-separation*: Given a dag G, a chain C (a chain in a directed graph is a sequence of adjacent nodes, the direction of the arrows does not matter) from node α to node β is said to be blocked by the set of nodes Z, if there is a vertex $\gamma \in C$ such that, either

  - $\gamma \in Z$ and arrows of C do not meet head to head at γ, or
  - $\gamma \notin Z$, nor has γ any descendants in Z, and the arrows of C do meet head to head at γ.

  A chain that is not blocked by Z is said to be active. Two subsets of nodes, X and Y, are said to be d-separated by Z, and this is denoted by $\langle X, Y|Z\rangle_G^d$, if all chains between the nodes in X and the nodes in Y are blocked by Z.

In [9], an equivalent criterion to d-separation was proposed, which will be especially useful for our purposes. Several previous concepts are necessary to establish this equivalence: Let G be a dag; given a node α, the nodes β such that there is a path in G from α to β are the *descendants* of α, written $de(\alpha)$. Similarly, the nodes β such that there is a path in G from β to α are called the *ancestors* of α, denoted by $an(\alpha)$. A subset of nodes X is an *ancestral set* if it contains its own ancestors, i.e., if $\mathrm{an}(\alpha) \subseteq X\ \forall \alpha \in X$. We denote $\mathrm{An}(X)$ to the smallest ancestral set containing X, i.e., $\mathrm{An}(X) = X \cup (\cup_{\alpha \in X} \mathrm{an}(\alpha))$. The *moral graph* G^m of the dag G is the undirected graph with the same set of nodes but with α and β adjacent in G^m if and only if they are adjacent in G or if there is a node γ such that $\alpha \to \gamma$ and $\beta \to \gamma$ are arcs in G. In other words, the moral graph is obtained from the original dag by 'marrying' parents with a common child and then dropping directions on arrows.

The following important result was proven in [9]: Let X, Y and Z be disjoint subsets of nodes in a dag G. Then, Z d-separates X from Y if and only if Z separates X from Y in $(G_{\mathrm{An}(X \cup Y \cup Z)})^m$, where $(G_{\mathrm{An}(X \cup Y \cup Z)})^m$ is the

moral graph of the subgraph of G whose set of nodes is the smallest ancestral set containing $X \cup Y \cup Z$. In symbols:

$$\langle X, Y | Z \rangle_G^d \iff \langle X, Y | Z \rangle^s_{(G_{\mathrm{An}(X \cup Y \cup Z)})^m} \quad (1)$$

3 Reducing d-Separation to Separation

The problem we are trying to solve is the following: Find a d-separating set of minimum size for two sets of nodes X and Y in a given dag G.

This is a combinatorial optimization problem, and, in principle, it does not seem us easy to solve, mainly because the d-separation criterion is difficult to manage, and is rather subtle: in some cases, the instantiation of some nodes (non head to head nodes) blocks the chains, and in some other (head to head nodes) unblocks them. So, we would like to transform the problem into an equivalent one, but avoiding the use of the d-separation criterion, and replacing it by a more 'uniform' criterion. The separation criterion for undirected graphs represents a good option. Therefore, the methodology we are going to use to solve our problem will first transform it into an equivalent separation problem.

The result in eq.(1) seems quite relevant to our purposes, because it relates d-separation with separation. However, this result is not directly applicable, because the transformed (undirected) graph where we have to test for the separation of X from Y depends on X, Y and also on the d-separating set Z, the set we are looking for. Thus, we would have to test for the separation of X from Y given Z in all the undirected graphs that can be formed by varying Z, and then selecting that Z of minimum cardinality, a completely prohibitive brute force approach.

However, in this Section we prove that it is possible to transform our problem into a separation problem, where the undirected graph in which we have to look for the minimum set separating X from Y depends only on X and Y. After, in the next Section, we will apply this result to develop an efficient algorithm that solves our original problem.

The next proposition shows that if we want to test a d-separation relationship between two sets of nodes X and Y in a dag, where the d-separating set is included in the smallest ancestral set of $X \cup Y$, then we can test this relationship in a more reduced dag, whose set of nodes is formed only by the ancestors of X and Y.

Proposition 1 *Given a dag $G = (\mathcal{U}, \mathcal{E})$, $X, Y \subseteq \mathcal{U}$, and $Z \subseteq An(X \cup Y)$, let $H = G_{\mathrm{An}(X \cup Y)}$ be the subgraph of G whose set of nodes is $An(X \cup Y)$. Then*

$$\langle X, Y | Z \rangle_G^d \Longleftrightarrow \langle X, Y | Z \rangle_H^d$$

Proof: The necessary condition is obvious, because H is a subgraph of G. Let us prove the sufficient condition: suppose that $\langle X, Y | Z \rangle_H^d$ but $\neg \langle X, Y | Z \rangle_G^d$. Then, in G, there is at least one chain C linking one node x in X and one node y in Y, such that for all $\gamma \in C$, if γ is not a head to head node then $\gamma \notin Z$, and if γ is head to head, then either $\gamma \in Z$ or $\mathrm{de}(\gamma) \cap Z \neq \emptyset$.

If the chain C were only formed by nodes from $\mathrm{An}(X \cup Y)$, then C would be a chain in H not blocked by Z, hence $\neg \langle X, Y | Z \rangle_H^d$, in contradiction with the hypothesis. Therefore, there are nodes in C which do not belong to $\mathrm{An}(X \cup Y)$. Let γ_0 be one of these nodes, i.e., $\gamma_0 \in C$, $\gamma_0 \notin \mathrm{An}(X \cup Y)$. As γ_0 belongs to a chain linking x and y, and γ_0 is not an ancestor of x or y, then γ_0 has to be either a head to head node of C or an ancestor of a head to head node of C. As all the head to head nodes of C belong to Z or are ancestors of nodes that belong to Z, and since $Z \subseteq \mathrm{An}(X \cup Y)$, then in either case γ_0 also belongs to $\mathrm{An}(X \cup Y)$, which is again a contradiction. Therefore we have $\langle X, Y | Z \rangle_G^d$. □

The following proposition establishes the basic result necessary to solve our optimization problem: it says that any d-separating set for the sets of nodes X and Y, which does not contain 'superfluous' nodes, must be included in the smallest ancestral set $\mathrm{An}(X \cup Y)$. In other words, all the minimal d-separating sets are formed exclusively by nodes which are ancestors of either X or Y.

Proposition 2 *Given a dag $G = (\mathcal{U}, \mathcal{E})$, $X, Y \subseteq \mathcal{U}$, let $Z \subseteq \mathcal{U}$ be a set of nodes such that $\langle X, Y | Z \rangle_G^d$ and $\neg \langle X, Y | Z' \rangle_G^d$, $\forall Z' \subset Z$. Then $Z \subseteq An(X \cup Y)$.*

Proof: Let us suppose that $Z \not\subseteq \mathrm{An}(X \cup Y)$. Then there is at least one node $z_0 \in Z$ such that $z_0 \notin \mathrm{An}(X \cup Y)$. Let $T = Z \setminus \{z_0\}$. Then, by the hypothesis we have $\neg \langle X, Y | T \rangle_G^d$ but $\langle X, Y | Z \rangle_G^d$. Therefore, using the d-separation criterion, we have:

-For every chain C linking nodes in X and nodes in Y, $\exists \gamma_c \in C$ such that: if γ_c is not head to head, then $\gamma_c \in Z$, and if γ_c is head to head, then $\gamma_c \notin Z$ and $\mathrm{de}(\gamma_c) \cap Z = \emptyset$.

-There is at least one chain C_{z_0} linking one node x in X and one node y in Y, such that for all $\gamma \in C_{z_0}$, if γ is not head to head then $\gamma \notin T$, and if γ is head to head, then either $\gamma \in T$ or $\mathrm{de}(\gamma) \cap T \neq \emptyset$.

It is simple to check that the chain C_{z_0} must include the node z_0 (otherwise, the chain C_{z_0} would not be blocked by Z, and therefore $\neg\langle X, Y|Z\rangle_G^d$, against the hypothesis). If z_0 is a head to head node in C_{z_0}, then z_0 cannot be the node in Z that blocks the chain C_{z_0}, hence it must exist another node $\alpha \in C_{z_0}$ such that either $\alpha \in Z$ is not head to head, or α is head to head, $\alpha \notin Z$, $\mathrm{de}(\alpha) \cap Z = \emptyset$. But in the first case $\alpha \in Z \setminus \{z_0\} = T$, and in the second case $\alpha \notin Z \setminus \{z_0\} = T$ and $\mathrm{de}(\alpha) \cap (Z \setminus \{z_0\}) = \emptyset$. In either case we obtain a contradiction with the fact that T does not block the chain C_{z_0}.

So, z_0 must be a non head to head node in C_{z_0}. As $z_0 \notin \mathrm{An}(X \cup Y)$, then it has to exist at least one node β in the chain C_{z_0}, such that β is head to head and z_0 is an ancestor of β. But in this case we have $\beta \in T$, hence $\beta \in Z$. Moreover, β cannot belong to $\mathrm{An}(X \cup Y)$ (otherwise, z_0 would also belong to $\mathrm{An}(X \cup Y)$). So, β is a node in Z that is not necessary to block the chain C_{z_0}. Then we could eliminate it from Z and the set $Z \setminus \{\beta\}$ perhaps would continue being a d-separating set; if this were true, then we would have a contradiction with the fact that Z is a minimal d-separating set. The only way of avoiding the contradiction is to assume that β is necessary to block another chain, different from C_{z_0}. In this case, as $\beta \in Z$ and $\beta \notin \mathrm{An}(X \cup Y)$, then we can repeat for β the same reasoning used so far for node z_0. This reasoning will give rise to find another chain C_β, such that β is a non head to head node in this chain and is an ancestor of a head to head node δ in C_β. As we cannot iterate this reasoning indefinitely, because the number of nodes is finite, then necessarily in a finite number of steps we will find a node in Z which can be eliminated from Z without destroying the d-separation relationship, which contradicts the minimality of Z. Therefore, we have to conclude that $Z \subseteq \mathrm{An}(X \cup Y)$. □

Corollary 1 *Given a dag $G = (\mathcal{U}, \mathcal{E})$, $X, Y, Z \subseteq \mathcal{U}$, if $\langle X, Y|Z\rangle_G^d$ and $\neg\langle X, Y|Z'\rangle_G^d$ $\forall Z' \subset Z$, then $\langle X, Y|Z\rangle_H^d$, where $H = G_{\mathrm{An}(X \cup Y)}$.*

The proof follows inmediately from propositions 1 and 2. This result says

that by replacing the dag G by its subgraph $G_{\mathrm{An}(X\cup Y)}$, we do not lose any minimal d-separation relationship between X and Y, i.e., all the minimal d-separation relationships between X and Y in G are kept in $G_{\mathrm{An}(X\cup Y)}$.

Corollary 2 *Given a dag $G = (\mathcal{U}, \mathcal{E})$, $X, Y \subseteq \mathcal{U}$, $H = G_{\mathrm{An}(X\cup Y)}$, let us define the sets $\mathcal{S}_G = \{Z \subseteq \mathcal{U} \mid \langle X, Y|Z\rangle_G^d\}$ and $\mathcal{S}_H = \{Z \subseteq An(X \cup Y) \mid \langle X, Y|Z\rangle_H^d\}$. Then*

$$T \in \mathcal{S}_G \text{ and } |T| = \min_{Z\in\mathcal{S}_G} |Z| \Longleftrightarrow T \in \mathcal{S}_H \text{ and } |T| = \min_{Z\in\mathcal{S}_H} |Z|.$$

Proof: From proposition 1 we deduce $\mathcal{S}_H \subseteq \mathcal{S}_G$, and therefore $\min_{Z\in\mathcal{S}_H} |Z| \geq \min_{Z\in\mathcal{S}_G} |Z|$.

Necessary condition: If $|T| = \min_{Z\in\mathcal{S}_G} |Z|$, then $\forall T' \subset T$ we have $T' \notin \mathcal{S}_G$, and from corollary 1 we obtain $T \in \mathcal{S}_H$. So, we have $|T| \geq \min_{Z\in\mathcal{S}_H} |Z| \geq \min_{Z\in\mathcal{S}_G} |Z| = |T|$, hence $|T| = \min_{Z\in\mathcal{S}_H} |Z|$.

Sufficient condition: If $|T| = \min_{Z\in\mathcal{S}_H} |Z| > \min_{Z\in\mathcal{S}_G} |Z| = |Z_0|$, then we have that $\forall Z' \subset Z_0$, $Z' \notin \mathcal{S}_G$, and therefore, once again from corollary 1, we get $Z_0 \in \mathcal{S}_H$ and $|Z_0| \geq \min_{Z\in\mathcal{S}_H} |Z| = |T|$, which is a contradiction. Thus, $|T| = \min_{Z\in\mathcal{S}_G} |Z|$. □

The previous corollary establishes that, in order to solve the problem of finding a minimum d-separating set for X and Y in a dag G, we can solve the equivalent problem of finding a minimum d-separating set for X and Y in the subdag $G_{\mathrm{An}(X\cup Y)}$, which only has the smallest ancestral set of $X \cup Y$ as set of nodes. The only remaining task is to transform this last problem into an equivalent problem involving separation instead of d-separation:

Proposition 3 *Given a dag $G = (\mathcal{U}, \mathcal{E})$, $X, Y \subseteq \mathcal{U}$, the subdag $H = G_{\mathrm{An}(X\cup Y)}$, and its moral graph H^m, let us define the sets $\mathcal{S}_H = \{Z \subseteq An(X \cup Y) \mid \langle X, Y|Z\rangle_H^d\}$, and $\mathcal{S}_H^m = \{Z \subseteq An(X \cup Y) \mid \langle X, Y|Z\rangle_{H^m}^s\}$. Then $\mathcal{S}_H = \mathcal{S}_H^m$.*

Proof: Let Z be any subset of $\mathrm{An}(X \cup Y)$. Then taking into account the characteristics of ancestral sets, it is clear that $H_{\mathrm{An}(X\cup Y\cup Z)} = H$. Then, by applying eq.(1) to the graph H, we have

$$Z \in \mathcal{S}_H \Leftrightarrow \langle X, Y|Z\rangle_H^d \Leftrightarrow \langle X, Y|Z\rangle_{(H_{\mathrm{An}(X\cup Y\cup Z)})^m}^s \equiv \langle X, Y|Z\rangle_{H^m}^s \Leftrightarrow Z \in \mathcal{S}_H^m$$

Hence $\mathcal{S}_H = \mathcal{S}_H^m$. □

Theorem 1 *The problem of finding a minimum d-separating set for X and Y in a dag G is equivalent to the problem of finding a minimum separating set for X and Y in the undirected graph $(G_{\mathrm{An}(X \cup Y)})^m$.*

The proof follows inmediately from corollary 2 and proposition 3.

Before finishing this Section, let us see on an example the practical significance of the previous results. Let us consider the graph in Figure 1, where we have numbered the nodes in an ordering compatible with the graph structure (i.e., the parents of any node appear before their children in the ordering).

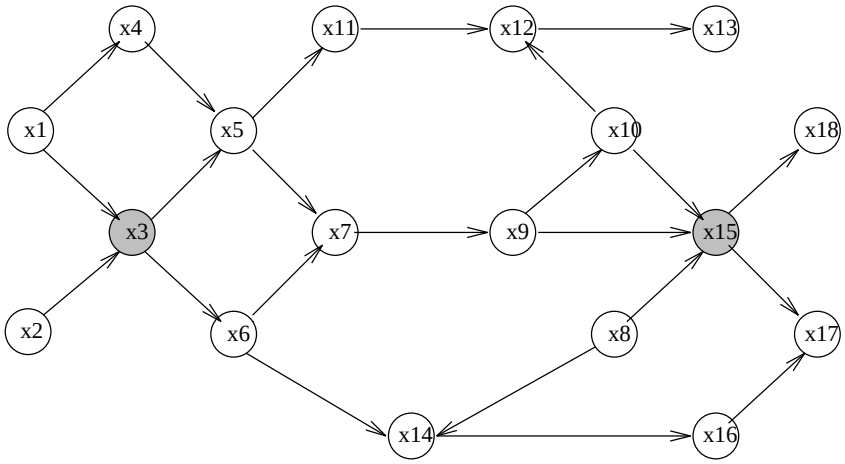

Figure 1: A dag G having 18 nodes

Let us suppose that we select the pair of nodes x_3 and x_{15}, and we want to find a set d-separating them with minimum size. Of course, we know that any node x is d-separated from all the other nodes by the set of the parents of x, the children of x and the parents of x's children. So, in our case, we know that the two sets $\{x_1, x_2, x_4, x_5, x_6\}$ and $\{x_8, x_9, x_{10}, x_{16}, x_{17}, x_{18}\}$ d-separate x_3 from x_{15}. We also know that any node x is d-separated from all its non-descendants by the set of x's parents. Therefore, as x_3 is not a descendant of x_{15}, we can be sure that x_3 and x_{15} are d-separated by the set $\{x_8, x_9, x_{10}\}$. However, can we find a smaller set that still d-separates these

two nodes? To answer this question we would have to examine every possible chain linking x_3 and x_{15}, to face the enormous amount of d-separating sets that we can obtain from the graph G, and next select the set of minimum size.

The result in corollary 2 allows us to reduce considerably the searching space where we have to look for the d-separating sets, by removing the nodes that do not belong to the set $\mathrm{An}(x_3, x_{15})$. The subgraph $G_{\mathrm{An}(x_3,x_{15})}$, which has 11 nodes, is shown in Figure 2. The complexity of the graph has decreased, thus reducing the number of chains we have to explore.

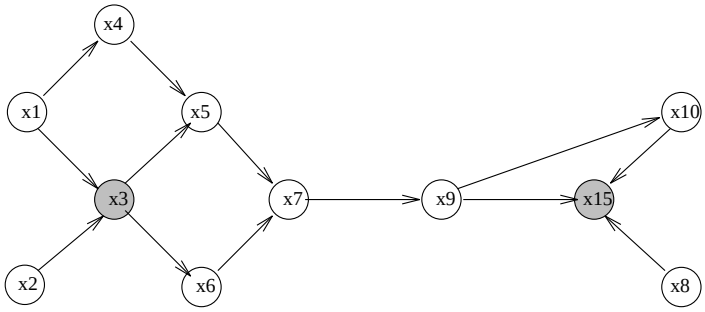

Figure 2: Dag $G_{\mathrm{An}(x_3,x_{15})}$

Finally, the corresponding moralized graph, $(G_{\mathrm{An}(x_3,x_{15})})^m$, where, by virtue of theorem 1, we have to search for the minimum separating set of x_3 and x_{15}, is shown in Figure 3

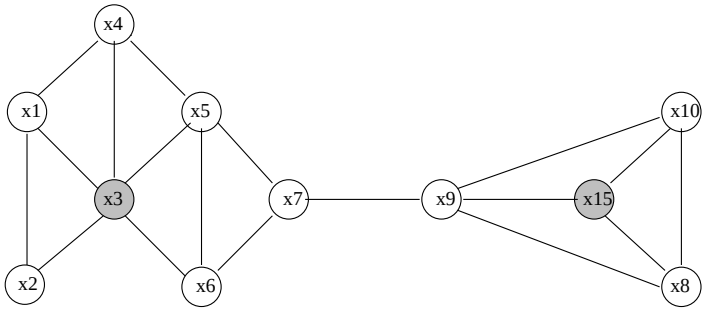

Figure 3: Moral graph $(G_{\mathrm{An}(x_3,x_{15})})^m$

4 The Algorithm

In this section we develop an algorithm to solve the basic problem stated in the introduction. After, we will see how to modify this algorithm to solve the two proposed extensions of the basic problem.

4.1 The algorithm for the basic problem

We want to develop an algorithm for finding a minimum d-separating set for two given nodes x and y in a dag G. However, it may happen that there is more than one d-separating set for x and y with minimum size. In this case we have to provide an additional criterion to select one of these sets. Our proposal is the following: as G is an acyclic graph, either x is not a descendant of y or y is not a descendant of x; suppose for example that x is not a descendant of y. Then a natural d-separating set for x and y in G would be the parent set of y, pa(y); however, we want to select a set of minimum size. If pa(y) had minimum size, then this set would be chosen; otherwise, we should replace some of (or all) the parents of y by other nodes, as long as this replacement diminishes the size of the d-separating set. But, in order to be coherent, we should use nodes as near as possible from y. For example, for the dag displayed in Figure 4, to d-separate x from y using a set of minimun size, we would select the set $\{t\}$ instead of the set $\{z\}$.

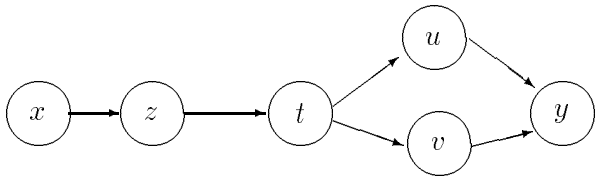

Figure 4: Dag where x and y can be d-separated by the sets $\{z\}$ and $\{t\}$

Therefore, our additional criterion to choose only one minimum d-separating set is the following: among all the minimum d-separating sets for x and y, and supposing that x is not a descendant of y, select the one which is nearer from y.

Starting from the results obtained in the previous Section, to solve the basic problem of finding a minimum d-separating set for a pair of nodes x and y in a dag G, it suffices to find a minimum separating set for x and y in the undirected graph $(G_{\mathrm{An}(X\cup Y)})^m$.

So, bearing in mind the discussion above, our specific objective in this subsection will be the following: given an undirected graph $H = (\mathcal{V}, \mathcal{E})$, and given two nodes $x, y \in \mathcal{V}$, find a set of minimum size that separates x from y; if there is more than one of these sets, select the one which is nearer from y (the proximity being measured in terms of the length of the chains linking y and the nodes in the separating set).

To design an algorithm for solving the problem above we will take advantage of the strong relationship that exists between problems of connectivity and flow problems in graphs.

In general, a flow problem arises when, given a directed graph, we want to determine the value of the maximum flow that can be transmitted from a specified source node s of the graph, to a specified sink node t. In this context, every arc of the graph has associated a number that represents the largest amount of flow that can be transmitted along the arc (the capacity of the arc). A method of solution of this *maximum flow* problem was developed by Ford and Fulkerson [7, 5]: their algorithm uses a labelling technique and a search tree to iteratively build up the flow in the network, and is based on an important result which establishes the relation between maximum flows and minimum cuts [7]: the value of the maximum flow from s to t in a graph is equal to the value of the minimum cut-set separating s from t.

Unfortunately, the term cut-set does not refer to a separating set containing nodes but to a separating set containing arcs: a cut-set separating s from t is a set of arcs such that all the paths in the graph going from s to t must pass along some arc in the cut-set; the value of a cut-set is the sum of the capacities of its arcs. Therefore, if the capacity of each arc were 1, then the Ford-Fulkerson algorithm would identify a cut-set of minimum size. So, although this result may be useful, it is not exactly that we need: we look for separating node-sets in undirected graphs, whereas the previous result refers to separating arc-sets (cut-sets) in directed graphs.

However, we can easily transform separating arc-sets for directed graphs into separating node-sets for undirected graphs: we can see any undirected graph $H = (\mathcal{V}, \mathcal{E})$ as a directed one, $\vec{H} = (\mathcal{V}, \vec{\mathcal{E}})$, by simply considering every

edge u–$v \in \mathcal{E}$ as the pair of arcs $u \rightarrow v$, $u \leftarrow v \in \vec{\mathcal{E}}$. Moreover, we can turn a problem of node connectivity in $\vec{H}$ into a problem of arc connectivity in an auxiliary graph $\vec{H}_{\text{aux}} = (\mathcal{V}', \vec{\mathcal{E}}_{\text{aux}})$, in the following way:

- Every node $u \in \mathcal{V}$ corresponds to two nodes $u^+, u^- \in \mathcal{V}'$.
- For every arc $u \rightarrow v \in \vec{\mathcal{E}}$ corresponds an arc $u^- \rightarrow v^+ \in \vec{\mathcal{E}}_{\text{aux}}$.
- We also introduce in $\vec{\mathcal{E}}_{\text{aux}}$ the arcs $u^+ \rightarrow u^-$.

The transformation from graph H into graph $\vec{H}_{\text{aux}}$ is illustrated in Figure 5.

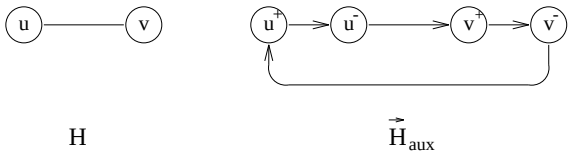

Figure 5: Transformation of H into $\vec{H}_{\text{aux}}$

Moreover we give all the arcs in $\vec{H}_{\text{aux}}$ capacity equal to 1. Then, to search for a minimum node-set separating s and t in H is equivalent to search for a minimum cut-set separating s^- from t^+ in $\vec{H}_{\text{aux}}$: since the total flow entering a node u^+ must, by necessity, travel along the arc $u^+ \rightarrow u^-$ whose capacity is 1, the maximum flow in the graph $\vec{H}_{\text{aux}}$ must correspond to a minimum cut-set containing only arcs of the form $u^+ \rightarrow u^-$; therefore, the nodes u in H that correspond to the arcs $u^+ \rightarrow u^-$ of the cut-set in $\vec{H}_{\text{aux}}$ comprise a separating node-set in H of size equal to the value of the cut-set.

So, to solve the problem of finding a minimum set separating two nodes x and y in an undirected graph H, we can apply the Ford-Fulkerson algorithm to find the maximum flow from y^- to x^+ in the auxiliary directed graph $\vec{H}_{\text{aux}}$. The reason for using the maximum flow from y^- to x^+ instead of the flow from x^- to y^+ is that the own dynamics of the Ford-Fulkerson algorithm will favor the presence of nodes close to the source y^- in the separating set, as required. Fortunately, it is not really necessary to transform explicitly the original graph H into the auxiliary graph $\vec{H}_{\text{aux}}$: we have developed an algorithm that works directly over H (in fact, our implementation of the algorithm uses the original dag G where we want to test for d-separation, and all the graph transformations, from G to $H = (G_{\text{An}(X \cup Y)})^m$ and to $\vec{H}_{\text{aux}}$ are implicit; however, for explanatory purposes, we will describe the algorithm

using the graph H).

Basically, the algorithm iteratively finds disjoint chains linking y and x (i.e., chains having only their endpoints in common). The maximum number of disjoint chains will be the minimum size of any separating set. The algorithm uses a search tree technique for finding disjoint chains. More precisely, we use a breadth first search to find the shortest disjoint chains.

The process of finding a disjoint chain can be seen as composed of a *forward* and a *backward* procedure. The forward procedure starts in the node y. From y, if the search tree reaches x, then a new chain has been found. During the search the algorithm uses a labelling procedure to trace the chain employed to go from y to x: while the candidate chains are being considered, every explored node is labelled with the name of the node from which it has been reached (using up to two labels), until no more nodes can be explored (the search tree is blocked) or the node x has been reached. The backward procedure is carried out only when the forward procedure has finished successfully in x. It tries to recover the new chain just found: every node in this chain is marked as belonging to some of the current chains found so far, and the involved edges are marked with the direction they have been used. During this process, some of the chains previously found may be modified (to accomplish the requirement of disjoint chains); next all the labels are removed. These two procedures are repeated until all search trees are blocked before reaching x. Then, starting from node y, and going through marked edges, the algorithm determines the minimum separating node-set.

After this informal explanation of the algorithm, let us describe it more precisely. The algorithm uses and updates the following information about nodes and edges:

A node can only be in one of three possible states:

- *labelled* and *scanned*, i.e., it has some label and all adjacent nodes have been 'processed'),
- *labelled* and *unscanned*, i.e., it has some label but not all its adjacent nodes have been processed),
- *unlabelled*, i.e., it has no label.

A node v may have simultaneously two labels, *label-p* and *label-n* (in order to implement the connectivity model shown in Figure 5: implicitly, v represents the two nodes v^+ and v^-. When node v has been reached forward from node u through the arc $u^- \rightarrow v^+$, the positive label of v, label-p(v), is set to u; when v is reached backward from u through the arc $v^- \leftarrow u^+$, the negative label, label-n(v), is set to u). If label-p(v) = u, this means that the edge u–v is part of a candidate chain linking x and y; if label-n(v) = u, then the edge u–v is part of an already existing chain linking x and y which may be modified by removing precisely the edge u–v. Each node v has also a boolean attribute, *in*: in(v) = true means that node v belongs to one of the current disjoint chains linking x and y.

On the other hand, each edge u–v has also two attributes, *marked* and *dir*: marked(u–v) = true means that this edge is part of some current chain linking x and y; in that case dir(u–v) = (u, v) indicates that this chain, if traversed in the direction from y to x, passes first through u and next through v. Finally, the algorithm also uses a queue Q to store the labelled and unscanned nodes.

The algorithm works as follows: Initially, all nodes are unlabelled and their *in*(.) attribute is false.

Step 1. Set label-p(y) = y. Enqueue y. y is now labelled an unscanned and all other nodes are unlabelled.

Step 2. Choose the first labelled and unscanned node u (and remove it from Q).

2.1 If in(u) = false then perform *f-search*(u).

2.2 If in(u) = true and label-n(u) is empty (i.e., u has only a positive label) then perform *b-search*(u).

2.3 If in(u) = true and label-n(u) is not empty (u has either only a negative label or both positive and negative labels) then perform both *f-search*(u) and *b-search*(u).

2.4 Put node u as labelled and scanned.

f-search(u) explores all the nodes v adjacent to u and gives a positive label (label-p(v) = u) to those which are unlabelled and verify marked(u–v) = false.

b-search(u) identifies the single node, t, adjacent to u, such that marked(u–t)=true and dir(u–t) = (t, u). If the negative label of t is empty, then label-n(y) is set to u. If t was already scanned, it is marked as unscanned (and therefore added to Q).

Step 3. Repeat Step 2 until either x is labelled in which case proceed to step 4 or x is unlabelled and no more labels can be placed (the queue Q is empty) in which case proceed to step 7.

The three steps above constitute that we have called the *forward procedure*.

Step 4. Set $u = x$ and $w = x$.

Step 5. Do the following:

5.1 If u has only positive label, z =label-p(u) then set marked(u–z)=true and dir(u–z) = (z, u). If $z \neq y$ then set in(z)=true.

5.2 If u has only negative label, z =label-n(u) then set marked(u–z)=false. If z has only negative label then set in(z)=false.

5.3 If u has both positive and negative labels and u = label-n(w) and z =label-p(u) then set marked(u–z)=true and dir(u–z) = (z, u). If $z \neq y$ then set in(z)=true.

5.4 If u has both positive and negative labels and u = label-p(w) and z =label-n(u) then set marked(u–z)=false.

Step 6. If $z \neq y$ then set $w = u$ and $u = z$ and return to step 5. Otherwise, erase all labels, empty the queue Q, and return to step 1.

The three steps above form that we have called the *backward procedure*. They construct the new disjoint chain linking x and y that was found at the end of step 3, and modify, if necessary, some of the previously found chains.

Step 7. Add to the separating set every node u adjacent to y such that marked(y–u)=true and the two labels of u are empty. For every node u adjacent to y verifying marked(y–u)=true such that some label of u is not empty do the following: starting from u continue through the chain containing only marked edges until finding the last node having a non empty label, and then add this node to the separating set.

Let us illustrate the use of this algorithm on the network in Figure 3: the first chain found linking x_{15} and x_3 is x_{15}–x_9–x_7–x_5–x_3, the in(.) values for the involved nodes are set to true, and the respective edges are marked. The forward procedure (using only *f-search*) finds this chain, that does not intersect with any previous chain, so the chain is also easily recovered by the backward procedure. Then no more search tree can reach x_3 from x_{15}, so that the forward-backward process stops. Finally, the only marked edge found from x_{15} is x_{15}–x_9, and x_9 is the last labelled node in the chain linking x_{15} and x_3 containing marked edges. So, the algorithm gives x_9 as the separating set.

To illustrate the behavior of the algorithm in a more complex situation, as the appeared when the new chain found at the end of step 3 intersects with a previous one, and therefore *b-search* has to manage negative labels, let us consider the graph G in Figure 6.

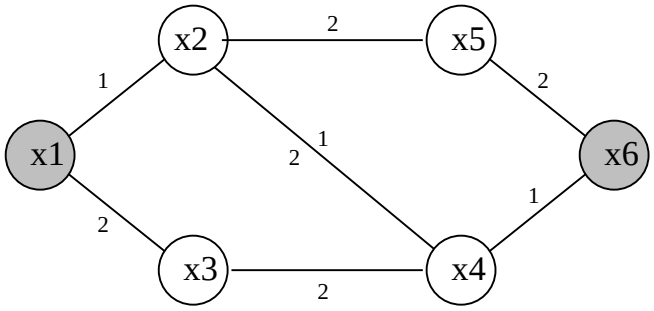

Figure 6: Graph G, with selected nodes x_1 and x_6

Let us suppose that the first chain linking $x1$ and $x6$ found by the algorithm is the one whose edges are numbered with 1 on the graph G. The edges x_1–x_2, x_2–x_4 and x_4–x_6 are marked and the in(.) values for the involved

nodes are set to true. Next, *f-search* is applied to go from $x6$ to x_2 (whose value in(x_2) is true). As the current chain cuts the previous one, then *b-search*(x_2) goes back to x_4 and the negative label of x_4 is set to x_2. There the chain may continue through the edges x_4–x_3 and x_3–x_1 by using the *f-search* process from x_4. This new chain is displayed in Figure 6 by means of edges numbered with 2. As the two chains are not disjoint, they share the edge x_2–x_4, the backward process that recovers the chain, after marking the edges x_1–x_3 and x_3–x_4, will notice it when it finds label-n(x_4) = x_2. It will remove the mark of the edge x_2–x_4 and will continue marking x_2–x_5 and x_5–$x6$, the result being the two disjoint chains x_6–x_5–x_2–x_1 and x_6–x_4–x_3–x_1. Then no more search tree can reach x_1 from x_6, and finally the algorithm gives the set $\{x_4, x_5\}$ as outcome.

4.2 Coping with the extensions

In this subsection we study how the algorithm previously developed can be modified to deal with the extensions of the basic problem.

For the first extension, i.e., to find the minimum set d-separating two subsets of nodes X and Y (instead of two single nodes) in a dag $G = (\mathcal{U}, \mathcal{E})$, the solution is very simple: first, we build the undirected graph $(G_{\mathrm{An}(X\cup Y)})^m = (\mathrm{An}(X \cup Y), \mathcal{E}^m_{\mathrm{An}(X\cup Y)})$, i.e., the moral graph of the subgraph of G whose set of nodes is the smallest ancestral set containing $X \cup Y$; next, starting from this graph we construct a new undirected graph $G^{XY} = (\mathcal{V}, \mathcal{F})$ as follows:

- $\mathcal{V} = \mathrm{An}(X \cup Y) \cup \{\alpha_X, \beta_Y\}$,
- $\mathcal{F} = \mathcal{E}^m_{\mathrm{An}(X\cup Y)} \cup \{\alpha_X\text{–}z \mid \exists x \in X \text{ s.t. } z\text{–}x \in \mathcal{E}^m_{\mathrm{An}(X\cup Y)})\} \cup \{\beta_Y\text{–}y \mid \exists y \in Y \text{ s.t. } z\text{–}y \in \mathcal{E}^m_{\mathrm{An}(X\cup Y)})\}$.

Put in words: we add two artificial nodes α_X and β_Y, and connect α_X and β_Y to those nodes that are adjacent to some node in X and Y, respectively. It can be easily proven that

- $\langle \alpha_X, \beta_Y | Z \rangle^s_{G^{XY}} \iff \langle X, Y | Z \rangle^s_{(G_{\mathrm{An}(X\cup Y)})^m}$, and
- if $\langle \alpha_X, \beta_Y | W \rangle^s_{G^{XY}}$ and $W \cap (X \cup Y) \neq \emptyset$, then $\langle \alpha_X, \beta_Y | W \setminus (X \cup Y) \rangle^s_{G^{XY}}$.

So, the separation of X and Y in $(G_{\mathrm{An}(X\cup Y)})^m$ is equivalent to the separation of α_X and β_Y in G^{XY}. Moreover, the minimum separating set for α_X and β_Y

in G^{XY} cannot contain nodes from $(X \cup Y)$. Therefore, in order to find the minimum d-separating set for X and Y in G, it suffices to find the minimum separating set for α_X and β_Y in the auxiliar graph G^{XY}. In this way, we have reduced the problem to one of separation for single nodes, which can be solved using the previous algorithm.

The second extension of the basic problem was the following: given two sets of nodes X and Y, and given a third set of nodes Z, find the minimum set, say S, such that $\langle X, Y | Z \cup S \rangle_G^d$. In this case we try to find the minimum d-separating set for X and Y but with the restriction that some nodes in the d-separating set are fixed. It can be proven that all the propositions stated in Section 3 can be extended to deal with this restriction, i.e., we can prove the following results (the proofs are almost identical to those in Section 3):

- If $S \subseteq \mathrm{An}(X \cup Y \cup Z)$, and $H = G_{\mathrm{An}(X \cup Y \cup Z)}$ then
$$\langle X, Y | Z \cup S \rangle_G^d \iff \langle X, Y | Z \cup S \rangle_H^d.$$
- If $\langle X, Y | Z \cup S \rangle_G^d$ and $\neg \langle X, Y | Z \cup S' \rangle_G^d$, $\forall S' \subset S$, then $S \subseteq \mathrm{An}(X \cup Y \cup Z)$.

So, by once again applying the result stated in [9], the problem of finding, in a dag G, a minimum d-separating set for X and Y which contains the set Z, is equivalent to the problem of finding a minimum separating set for X and Y, containing Z, in the undirected graph $(G_{\mathrm{An}(X \cup Y \cup Z)})^m$. Now, it suffices to eliminate from this last graph the set Z, i.e., to search for the minimum separating set for X and Y in the graph $((G_{\mathrm{An}(X \cup Y \cup Z)})^m)_{\mathrm{An}(X \cup Y \cup Z) \setminus Z}$.

5 Concluding Remarks

We have studied the problem of finding minimum d-separating sets for pairs of variables in belief networks, and developed an efficient algorithm for solving it. Our method is based on a theoretical study that allows us to transform the original problem into an equivalent problem of separation in undirected graphs. The proposed algorithm implicitly uses this equivalence, and is based on a suitable modification of a well-known algorithm from the Operations Research literature, the Ford-Fulkerson algorithm of maximum flow in networks with capacities. We have also studied some extensions of the basic problem: finding minimum d-separating sets for subsets of variables, and finding

minimum d-separating sets for variables or subsets of variables, with the restriction that some variables in the d-separating sets must be fixed. Our basic algorithm is also able to manage these extensions with minor modifications. Actual and potential applications of this research include learning belief networks from data [1] and problems related with the selection of the variables to be instantiated when using belief networks for inference tasks.

Another problem that is quite similar to that one considered here, which perhaps could be efficiently solved using similar methods, is the following: Given some variable of interest in a belief network, and given a set of instantiated variables, what is the minimum subset of these variables that we actually need to propagate, taking into account the independence relationships displayed by the network?

From a more theoretical point of view, another interesting problem is the following: our algorithm can be seen as a method to find, given a network, the conditional independence relationships involving the minimum number of variables. Can this process be reversed? In other words, given this set of conditional independence assertions, can all the other original independence relationships be recovered? This would lead to the definition of a new concept, similar to that of *causal input list* or *recursive basis* [13], but with a more 'local' character. We plan to study these topics in the future.

References

[1] S. Acid, L.M. de Campos, BENEDICT: An algorithm for learning probabilistic belief networks, To appear in the Proceedings of the IPMU-96 Conference.

[2] L.M. de Campos, Independence relationships in possibility theory and their application to learning belief networks, in: G. Della Riccia, R. Kruse, R. Viertl, eds., Mathematical and Statistical Methods in Artificial Intelligence, CISM Courses and Lectures 363, (Springer Verlag, Wien, 1995) 119–130.

[3] L.M. de Campos, J.F. Huete, Independence concepts in upper and lower probabilities, in: B. Bouchon-Meunier, L. Valverde, R.R. Yager, eds., Uncertainty in Intelligent Systems (North-Holland, Amsterdam, 1993) 49–59.

[4] L.M. de Campos, J.F. Huete, Learning non probabilistic belief networks, in: M. Clarke, R. Kruse, S. Moral, eds., Symbolic and Quantitative Approaches to Reasoning and Uncertainty, Lect. Notes Comput. Sc. 747 (Springer Verlag, Berlin, 1993) 57–64.

[5] N. Christofides, Graph Theory, an Algorithmic Approach (Academic Press, London, 1975).

[6] A.P. Dawid, Conditional independence in statistical theory, J.R. Statist. Soc. Ser. B 41 (1979) 1–31.

[7] L.R. Ford, D.R. Fulkerson, Flows in Networks (Princeton Univ. Press, Princeton, NJ, 1962).

[8] S.L. Lauritzen, Lectures on Contingency Tables, 2nd ed. (University of Aalborg Press, Aalborg, Denmark, 1982).

[9] S.L. Lauritzen, A.P. Dawid, B.N. Larsen, H.-G. Leimer, Independence properties of directed Markov fields, Networks 20 (1990) 491–505.

[10] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference (Morgan and Kaufmann, San Mateo, 1988).

[11] P.P. Shenoy, Conditional independence in uncertainty theories, in: D. Dubois, M.P. Wellman, B. D'Ambrosio, P. Smets, eds., Uncertainty in Artificial Intelligence, Proceedings of the Eighth Conference (Morgan and Kaufmann, San Mateo, 1992), 284–291.

[12] W. Spohn, Stochastic independence, causal independence and shieldability, Journal of Philosophical Logic 9 (1980) 73–99.

[13] T. Verma, J. Pearl, Causal networks: Semantics and expressiveness, in: R.D. Shachter, T.S. Levitt, L.N. Kanal, J.F. Lemmer, eds., Uncertainty in Artificial Intelligence 4 (North-Holland, Amsterdam, 1990), 69–76.

[14] N. Wilson, Generating graphoids from generalized conditional probability, in: D. Poole, R. López de Mántaras, eds., Uncertainty in Artificial Intelligence, Proceedings of the Tenth Conference (Morgan Kaufmann, San Francisco, 1994) 583–590.