J. Expt. Theor. Artif. Intell. 10(1998)511–549

Independency relationships and learning algorithms for singly connected networks

LUIS M. DE CAMPOS

Departamento de Ciencias de la Computación e I.A., Universidad de Granada, 18071-Granada, Spain
email: lci@decsai.ugr.es

Abstract. Graphical structures such as Bayesian networks or Markov networks are very useful tools for representing irrelevance or independency relationships, and they may be used to efficiently perform reasoning tasks. Singly connected networks are important specific cases where there is no more than one undirected path connecting each pair of variables. The aim of this paper is to investigate the kind of properties that a dependency model must verify in order to be equivalent to a singly connected graph structure, as a way of driving automated discovery and construction of singly connected networks in data. The main results are the characterizations of those dependency models which are isomorphic to singly connected graphs (either via the d-separation criterion for directed acyclic graphs or via the separation criterion for undirected graphs), as well as the development of efficient algorithms for learning singly connected graph representations of dependency models.

Keywords: graphical models, independency relationships, learning algorithms, singly connected networks.

Received 14 February 1997; revision received 11 June 1997; accepted 12 June 1997

1. Introduction

Graphical models have become common knowledge representation tools capable of efficiently representing and handling independency relationships as well as uncertainty in our knowledge. They comprise a qualitative and a quantitative component. The qualitative component is a graph displaying dependency/independency relationships: the absence of some links means the existence of certain conditional independency relationships between variables, and the presence of links may represent the existence of direct dependency relationships (if a causal interpretation is given, then the (directed) links signify the existence of direct causal influences between the linked variables). This is important because an appropriate use of independency or irrelevance relationships is crucial for the management of information, since independency 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. The quantitative component is a collection of numerical parameters, usually conditional probabilities, which give idea of the strength of the dependencies and measure our uncertainty. Therefore,

graphical models provide an intuitive graphical visualization of the available knowledge, including the interactions among the different knowledge components and the various sources of uncertainty. On the other hand, graphical models also encode probabilistic information in an economical way: the storage requirements of a joint probability distribution are usually excessive, whereas the memory requirements of a suitable factorization of this distribution, taking into account the independency relationships displayed by the graph, may be much smaller.

However, when the graphs representing the independency statements corresponding to a given domain of knowledge are very dense or contain a great number of variables, the processes needed to estimate them from empirical data (learning) (Chickering *et al.* 1994, Cooper and Herskovits 1992, Lam and Bacchus 1994, Pearl and Verma 1991, Spirtes 1991, 1993) and to use them for inference tasks (propagation) (Lauritzen and Spiegelhalter 1988, Pearl 1986, 1988, Shachter 1988) may still be time-consuming. Some simplified models, such as singly connected networks (SCNs) may alleviate these problems at the expense of losing some representation capabilities: these are graphs where no more than one (undirected) path connects every two nodes or variables. Using SCNs, we gain in efficiency and simplicity in the procedures for learning the networks as well as for propagating information through them (Pearl 1988). Therefore, SCNs and similar structures have received considerable attention within the artificial intelligence and statistical communities, from different points of view: learning and causality (Geiger *et al.* 1990, Huete and de Campos 1993, Rebane and Pearl 1989), classification (Friedman and Golszmidt 1996, Geiger 1992), propagation (Cano *et al.* 1993, Ng and Levitt 1994), data compression (Chow and Liu 1968), approximate models (Acid *et al.* 1991, Acid and de Campos 1995, Sarkar 1993). The price we have to pay for using SCNs is a less expressive power, because the kind of independency relationships that may be represented is more restricted for SCNs than for general multiply connected networks (MCNs).

So, the study of the kind of independency relationships which are associated to SCNs is interesting, not only from a purely theoretical point of view but also for practical reasons: if SCNs, such as forests, trees or polytrees, are to be used as approximations of more complex models, it is necessary to know the assumptions about independency that SCNs require. Moreover, some propagation methods (Lauritzen and Spiegelhalter 1988) are based on a clustering of variables that transforms the graph into a tree of cliques. Another method (Becker and Geiger 1994, Pearl 1986) is based on the ability to change the connectivity of the network and turns it into a singly connected one by instantiating a selected subset of variables. In those cases, our study could be applied to these SCNs. Furthermore, a theoretical study may create desiderata that could drive the automated construction of singly connected networks from data.

This paper's aim is twofold: first, to study the class of dependency models which are isomorphic to SCNs either via the d-separation criterion (Verma and Pearl 1990) for directed acyclic graphs (polytrees) or via the separation criterion for undirected graphs (forests and trees). This study should reveal some basic properties that could guide us in the design of algorithms for learning SCNs. So, the second objective is to develop efficient, exact and approximate algorithms for learning singly connected networks from data, by testing independency relationships between variables.

The paper is organized as follows: in Section 2, we briefly describe several concepts which are essential for subsequent development. Sections 3 and 4 are devoted to the undirected case: in Section 3 we prove characterizations of dependency models

isomorph to forests and trees; Section 4 uses the previous results to develop new algorithms for building undirected SCNs; comparisons with other algorithms and experimental results are also provided. The directed case is considered in Sections 5 and 6, where a study analogous to the previous one is carried out: first, in Section 5, we show a characterization of dependency models isomorph to polytrees; next, in Section 6, we develop efficient, exact and approximate algorithms for learning directed SCN representations of dependency model. Section 7 contains the concluding remarks and some proposals for future work. Finally, the Appendix contains the proofs of several technical lemmas which are necessary to establish the main results of the paper.

2. Preliminaries

In this section, we are going to describe the notation and some basic concepts used throughout the paper, although we shall omit the description of basic terminology for graphs.

A *Dependency Model* (Pearl 1988) is a pair $M = (U, I)$, where 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 U. Single elements of U will be denoted by standard or Greek lowercase letters, such as $q, s, t, \alpha, \beta \ldots$, whereas subsets of 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 (Dawid 1979, Geiger *et al.* 1991, Lauritzen *et al.* 1990, Spohn 1980, Studený 1989, 1990), $I(X, Y|Z)$ holds if and only if

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

for every instantiation $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ of the sets of variables X, Y and Z. However, dependency models are applicable to many situations far beyond probabilistic models (de Campos 1995, de Campos and Huete 1993a, b, Hunter 1991, Pearl 1988, 1989, Shenoy 1992, Smith 1989, Studený 1993, Verma and Pearl 1990b, Wilson 1994). In any case, the study of the concept of conditional independency in probability theory and that of embedded multivalued dependency in database theory (Fagin 1977) 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 independency. These properties are the following:

(A1) Symmetry:

$$(I(X, Y|Z) \Rightarrow I(Y, X|Z)) \forall X, Y, Z \subseteq U.$$

(A2) Decomposition:

$$(I(X, Y \cup W|Z) \Rightarrow I(X, Y|Z)) \forall X, Y, W, Z \subseteq U.$$

(A3) Weak union:

$$(I(X, Y \cup W|Z) \Rightarrow I(X, W|Z \cup Y)) \forall X, Y, W, Z \subseteq U.$$

(A4) Contraction:

$$(I(X, Y|Z) \text{ and } I(X, W|Z \cup Y) \Rightarrow I(X, Y \cup W|Z)) \forall X, Y, W, Z \subseteq U.$$

(A5) Intersection:

$$(I(X, Y|Z \cup W) \text{ and } I(X, W|Z \cup Y) \Rightarrow I(X, Y \cup W|Z)) \forall X, Y, W, Z \subseteq U.$$

The intuitive interpretation of these axioms is as follows: symmetry asserts that in any state of knowledge Z, if Y tells us nothing new about X, then X tells us nothing new about Y. Decomposition establishes that if two combined pieces of information Y and W are considered irrelevant to X, then each separate piece of information is also irrelevant. Weak union asserts that learning the irrelevant information Y cannot help the irrelevant information W to become relevant to X. Contraction states that if two pieces of information, X and W, are irrelevant to each other after knowing irrelevant information Y, then they were irrelevant before knowing Y too. Together, weak union and contraction mean that irrelevant information should not modify the nature of being relevant or irrelevant of other propositions in the system. Finally, intersection asserts that if two combined items of information, Y and W, are relevant to X, then at least one of them is also relevant to X, when the other is added to our previous state of knowledge, Z. Dependency models are called *semi-graphoids* if they verify the axioms A1–A4, and *graphoids* if they satisfy the axioms A1–A5 (Pearl 1988).

A graphical representation of a dependency model M is a direct correspondence between the elements in M and the set of nodes or vertices in a given graph, G, such that the topology of G reflects some properties of M. For simplicity in the notation, a node in the graph G will be referred to by the element in M associated with it. The way we relate independency assertions in M 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 (Lauritzen 1982, Pearl 1988) and *d-separation* for directed acyclic ones (dags) (Lauritzen and Spiegelhalter 1988, Pearl 1988, Shachter, 1988, Verma and Pearl 1990b):

- *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 $\langle X, Y \mid Z \rangle_G$, 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 $\langle X, Y \mid Z \rangle_G$, if all chains between the nodes in X and the nodes in Y are blocked by Z (separation and d-separation are denoted in the same way, because the context will avoid confusing them). In (Lauritzen *et al.* 1990), a criterion equivalent to d-separation was proposed (in terms of the separation of X from Y by Z in the moral graph of the smallest ancestral set containing $X \cup Y \cup Z$).

Given a dependency model, M, we say that an undirected graph (a dag, respectively), G, is an *Independency map* or *I-map* (Pearl 1988) if every separation (d-separation respectively) in G implies an independency in M:

$$\langle X, Y \mid Z \rangle_G \Rightarrow I(X, Y \mid Z).$$

On the other hand, an undirected graph (a dag, respectively), G, is called a *Dependency map* or *D-map* (Pearl 1988) if every independency relation in the model implies a separation (d-separation respectively) in the graph:

$$I(X, Y \mid Z) \Rightarrow \langle X, Y \mid Z \rangle_G.$$

A graph, G, is a *Perfect map* of M (Pearl 1988) if it is both an I-map and a D-map. M is said to be *graph-isomorphic* if a graph exists which is a perfect map of M.

3. Undirected graphs: independency relationships in forests and trees

In this section, we are going to show two axiomatic characterizations of dependency models isomorphic to forests and trees. These results will be applied in the next section to develop algorithms for learning the exact graphical representation of the previous models, as well as algorithms for learning forest or tree approximations of more general dependency models.

Let us consider the following axioms:

(F1) Symmetry:

$$(I(X, Y \mid Z) \Rightarrow I(Y, X \mid Z)) \forall X, Y, Z \subseteq U.$$

(F2) Decomposition:

$$(I(X, Y \cup W \mid Z) \Rightarrow I(X, Y \mid Z)) \forall X, Y, W, Z \subseteq U.$$

(F3) Strong union:

$$(I(X, Y \mid Z) \Rightarrow I(X, Y \mid Z \cup W)) \forall X, Y, W, Z \subseteq U.$$

(F4) Intersection:

$$(I(X, Y \mid Z \cup W) \text{ and } I(X, W \mid Z \cup Y) \Rightarrow I(X, Y \cup W \mid Z)) \forall X, Y, W, Z \subseteq U.$$

(F5) Transitivity:

$$(I(X, Y \mid Z) \Rightarrow I(X, \gamma \mid Z) \text{ or } I(\gamma, Y \mid Z) \forall \gamma \in U \setminus (X \cup Y \cup Z)) \forall X, Y, Z \subseteq U.$$

(F6) Atriangularity:

$$(\neg I(\alpha, \gamma \mid Z) \forall Z \subseteq U \setminus \{\alpha, \gamma\} \text{ and}$$

$$\neg I(\gamma, \beta \mid Z) \forall Z \subseteq U \setminus \{\gamma, \beta\} \Rightarrow I(\alpha, \beta \mid \gamma)) \forall \alpha, \beta, \gamma \in U.$$

Axioms F1–F5 are well known, and they have been proposed as the basic properties governing the separation relationships in undirected graphs. Axioms F1, F2 and F4 have already been commented upon. F3 is stronger than weak union, stating that the separating set Z can be unconditionally increased by additional variables without destroying the independence. In graph terms, if Z is a set separating X from Y, then when removing additional nodes W from the graph, X and Y are still separated. Contraction can easily be deduced from F3 and F4. So, the kind of dependency models that we consider here are graphoids. F5 establishes that if X is connected to some node γ and γ is connected to Y, then X must also be connected to Y. Axiom F6 intends to restrict the kind of graphs that we consider, by establishing that the graph cannot contain any triangle (a complete subgraph of three nodes), because the central node, γ, separates α from β.

In order to build a graph that represents the dependency model M, the idea is to introduce an edge into the graph for each pair of variables which are not independent in M, given any subset of U. Using strong union, it is immediately shown that

$$\neg I(\alpha, \beta \mid Z) \forall Z \subseteq U \backslash \{\alpha, \beta\} \Leftrightarrow \neg I(\alpha, \beta \mid U \backslash \{\alpha, \beta\}). \quad (1)$$

Therefore, in order to assert that two variables are not independent given any subset of U it is suffice to show that they are not independent given all the other variables. So, taking into account (1), the graph associated with a dependency model M is defined as follows:

Definition 1. *Given a dependency model $M = (U, I)$ verifying the axioms F1–F6, the (undirected) graph associated with M is $G_M = (U, E_M)$, where the set of edges E_M is*

$$E_M = \{(\alpha, \beta) \mid \alpha, \beta \in U, \neg I(\alpha, \beta \mid U \backslash \{\alpha, \beta\})\}. \quad (2)$$

Note that, using (1), axiom F6 can be reformulated in the following way:

(F6) Atriangularity:

$$(\neg I(\alpha, \gamma \mid U \backslash \{\alpha, \gamma\}) \text{ and } \neg I(\gamma, \beta \mid U \backslash \{\beta, \gamma\}) \Rightarrow I(\alpha, \beta \mid \gamma)) \forall \alpha, \beta, \gamma \in U.$$

It has been shown (Pearl and Paz 1985) that any dependency model verifying F1–F5 is isomorphic to its associated graph (using the usual separation criterion for undirected graphs). For example, the dependency model M, defined on $U = \{\alpha, \beta, \gamma, \delta\}$, where $I = \{I(\alpha, \beta \mid \gamma \cup \delta), I(\gamma, \delta \mid \alpha \cup \beta)\}$ verifies F1–F5; however, it does not satisfy F6. So, M is isomorphic to its associated graph G_M, which is depicted in figure 1, but G_M is not a forest. We are going to demonstrate that by adding the axiom F6, the associated graph necessarily becomes a forest. The next proposition proves a basic result in this direction.

Proposition 1. *If a dependency model M satisfies the axioms F1–F6, then its associated graph G_M is a forest, i.e. G_M has no cycle.*

Proof. First, let us prove that if $t_1 t_2 \cdots t_n$ is a chain in G_M, i.e. if $(t_i, t_{i+1}) \in E_M, \forall i = 1, \ldots, n-1$, then $I(t_1, t_n \mid t_2)$. We will use induction.

For $n = 3$, as $(t_1, t_2), (t_2, t_3) \in E_M$ then $\neg I(t_1, t_2 \mid U \backslash \{t_1, t_2\})$ and $\neg I(t_2, t_3 \mid U \backslash \{t_2, t_3\})$, and using atriangularity we obtain $I(t_1, t_3 \mid t_2)$. Suppose that the result is true for $n-1$, i.e. $I(t_1, t_{n-1} \mid t_2)$; then, by applying transitivity to this statement, we obtain $I(t_1, t_n \mid t_2)$ or $I(t_{n-1}, t_n \mid t_2)$. But $(t_{n-1}, t_n) \in E_M$ and we can deduce $\neg I(t_{n-1}, t_n \mid t_2)$ from (1); so, we have $I(t_1, t_n \mid t_2)$.

Now, let us suppose that $t_1 t_2 \cdots t_n t_1$ is a cycle in G_M. Then $t_1 t_2 \cdots t_n$ is a chain in G_M,

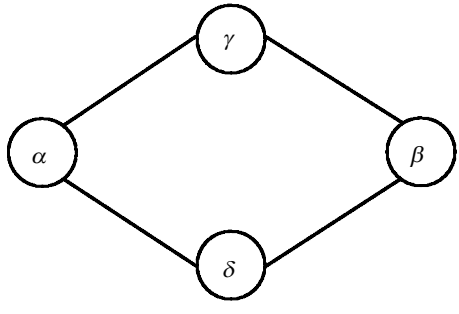

Figure 1. Graph G_M associated with the dependency model M.

and from the previous result we have $I(t_1, t_n \mid t_2)$. But (t_1, t_n) is an edge of G_M and therefore we also have $\neg I(t_1, t_n \mid U \backslash \{t_1, t_n\})$. As from (1) both results are contradictory, the conclusion is that G_M cannot have any cycle. □

Now, using the result in (Pearl and Paz 1985), we can easily prove the following characterization of forest-isomorphic dependency models.

Theorem 1. *A dependency model M is forest-isomorphic if, and only if, it verifies the axioms F*1–*F*6.

Proof. The necessary condition follows immediately from the fact that axioms F1–F5 are true for the separation relation in undirected graphs, and axiom F6 is also obviously true for the separation relation in forests.

The sufficient condition follows from proposition 1 and the result in (Pearl and Paz 1985), which establishes that a dependency model verifying the axioms F1–F5 is isomorphic to its associated undirected graph. □

The next result shows which is the additional axiom necessary to ensure that the graph (forest) G_M associated to M is connected, that is to say, to force G_M to be a tree.

Theorem 2. *A dependency model M is tree-isomorphic if, and only if, it verifies F*1–*F*6 *and the following additional axiom F*7:

(*F*7) *Connection*: $\neg I(\alpha, \beta \mid)\ \forall \alpha, \beta \in U$.

Proof. We already know that the axioms F1–F6 characterize forest-isomorphic models. So, to characterize tree-isomorphic models, we only have to find an axiom which guarantees that the forest G_M is connected: G_M is connected if and only if there is a chain linking every two nodes, or, in other words, if no pair of nodes are separated by the empty set: $\neg \langle \alpha, \beta \mid \rangle, \forall \alpha, \beta \in U$. But from theorem 1, this is equivalent to F7. □

To finish off this section, let us see why the characterization theorems of forest-isomorphic and tree-isomorphic dependency models cannot be refined.

Theorem 3. *The set of axioms F*1–*F*6 (*respectively F*1–*F*6 *and F*7) *constitutes a minimal set of axioms that characterize dependency models which are forest-isomorphic* (*respectively tree-isomorphic*).

Proof. We shall only prove the result for forest-isomorphic models; the proof for tree-isomorphic models is similar. According to theorem 1, in order to prove the result it is suffice to find dependency models which verify all the axioms except one.

(1) $M = (U, I)$, where $U = \{\alpha, \beta\}$, and $I = \{(\alpha, \beta \mid)\}$ verifies F2–F6, but it does not verify symmetry ($I(\alpha, \beta \mid)$ but $\neg I(\beta, \alpha \mid)$).

(2) $M = (U, I)$, where $U = \{\alpha, \beta, \gamma\}$, and $I = \{(\alpha, \beta \mid \gamma), (\alpha, \beta \cup \gamma \mid)$, symmetrical images$\}$ satisfies F1 and F3–F6, but it does not satisfy decomposition ($I(\alpha, \beta \cup \gamma \mid)$ but $\neg I(\alpha, \beta \mid)$ and $\neg I(\alpha, \gamma \mid)$).

(3) $M = (U, I)$, where $U = \{\alpha, \beta, \gamma\}$, and $I = \{(\alpha, \beta \mid)$, $(\alpha, \gamma \mid)$, $(\alpha, \gamma \mid \beta)$, symmetrical images$\}$ verifies F1, F2 and F4–F6, but it does not verify strong union ($I(\alpha, \beta \mid)$ but $\neg I(\alpha, \beta \mid \gamma)$).

(4) $M = (U, I)$, where $U = \{\alpha, \beta, \gamma, \delta\}$, and $I = \{(\alpha, \beta \mid \gamma)$, $(\alpha, \delta \mid \gamma)$, $(\alpha, \beta \mid \gamma \cup \delta)$, $(\alpha, \gamma \mid \beta \cup \delta)$, $(\alpha, \delta \mid \beta \cup \gamma)$, $(\beta, \gamma \mid \alpha \cup \delta)$, $(\beta, \delta \mid \alpha \cup \gamma)$, $(\gamma, \delta \mid \alpha \cup \beta)$, symmetrical images$\}$ satisfies F1–F3, F5 and F6, but it does not satisfy intersection (because we have $I(\beta, \gamma \mid \alpha \cup \delta)$ and $I(\beta, \delta \mid \alpha \cup \gamma)$ but $\neg I(\beta, \gamma \cup \delta \mid \alpha)$).

(5) $M = (U, I)$, where $U = \{\alpha, \beta, \gamma\}$, and $I = \{(\alpha, \beta|\ ),\ (\alpha, \beta|\gamma)$, symmetrical images$\}$ satisfies F1–F4 and F6, but it does not satisfy transitivity ($I(\alpha, \beta|\ )$ but $\neg\, I(\alpha, \gamma|\ )$ and $\neg\, I(\gamma, \beta|\ )$).

(6) $M = (U, I)$, where $U = \{\alpha, \beta, \gamma\}$, and $I =$ verifies F1–F5, but it does not verify atriangularity ($\neg\, I(\alpha, \gamma|\beta)$ and $\neg\, I(\gamma, \beta|\alpha)$ but $\neg\, I(\alpha, \beta|\gamma)$). □

4. Undirected graphs: learning algorithms

In this section, we apply the results from the previous section to obtain efficient algorithms for learning undirected SCNs. If the underlying dependency model is isomorphic to an SCN, then the algorithms will recover the corresponding graph. Otherwise, if the dependency model is not isomorphic to an SCN but it is isomorphic to an MCN, the algorithms may either output a 'fail' signal, meaning that the model is not isomorphic to an SCN, or produce a graph that is a (non-minimal) I-map of the model. In addition, the algorithms can be modified to always build an SCN which is intended as an approximation of the given dependency model.

4.1. *Algorithms for learning forests or trees*

As we have shown in Section 3, a dependency model is forest-isomorphic if, and only if, it verifies the axioms F1–F6. Observe that intersection (F4) and strong union (F3) imply the converse of decomposition (called composition). So, it is clear that any dependency model satisfying F1–F6 is completely defined by the set of independency type statements like $I(\alpha, \beta|Z)$, because $I(X, Y|Z) \Leftrightarrow I(\alpha, \beta|Z), \forall \alpha \in X, \forall \beta \in Y$. Furthermore, we are going to prove that only the set of statements $I(\alpha, \beta|\gamma)$ are necessary and sufficient to characterize this kind of dependency models.

Proposition 2. *Let M be a dependency model verifying the axioms F1–F6. Then, for all $\alpha, \beta \in U, Z \subseteq U \backslash \{\alpha, \beta\}, Z \neq$.*

$$I(\alpha, \beta|Z) \Leftrightarrow \exists \gamma \in Z \text{ such that } I(\alpha, \beta|\gamma). \tag{3}$$

Proof. The sufficient condition follows immediately from strong union. Let us prove the necessary condition: from Theorem 1 we know that independency statements in the model are equivalent to separation statements in a forest. As in a forest there is at most one chain linking every two nodes α and β, if α and β are separated by Z, then Z must contain at least one node γ in this chain, and this single node still separates α and β; if there is no chain linking α and β, then any single node in Z separates α and β. □

The previous proposition, together with the composition property, allows us to say that the dependency models considered are completely defined by independency statements with the form $I(\alpha, \beta|\gamma)$ (except in the trivial case where the model contains only two variables, i.e. $|U| = 2$). This fact is important for designing efficient algorithms for learning undirected SCNs, because from Proposition 2 we can easily deduce that, for the undirected graph G_M associated to M, and for every pair $\alpha, \beta \in U$,

$$\text{the edge } \alpha\text{–}\beta \text{ is in } G_M \Leftrightarrow \neg\, I(\alpha, \beta|\gamma)\ \forall \gamma \in U \backslash \{\alpha, \beta\}. \tag{4}$$

Equation (4) provides the basis for designing a simple and efficient algorithm to find the graph (forest or tree) associated with a dependency model which verifies F1–F6. It simply tests, for every pair of variables, the conditional independency of these two

variables given any other third variable; as soon as the algorithm finds a true conditional independency relationship, it removes the corresponding edge from the graph. This Exact Tree (ET) algorithm is depicted in figure 2.

It is clear that the complexity of the ET algorithm is $O(n^3)$ by the number of independence verifications, where n is the number of variables, $n = |U|$. Because of the equivalence $\neg I(\alpha, \beta | U \backslash \{\alpha, \beta\}) \Leftrightarrow \neg I(\alpha, \beta | \gamma) \forall \gamma \in U \backslash \{\alpha, \beta\}$, we could replace the set of tests $I(\alpha, \beta | \gamma)$ by the single test $I(\alpha, \beta | U \backslash \{\alpha, \beta\})$, and we would obtain an $O(n^2)$ algorithm. If we do so, we obtain Pearl and Paz's algorithm (Pearl 1988, Pearl and Paz 1985). But the latter type of independency test is much more difficult to apply if we do not know in advance the truth values of the independency statements, but we are going to estimate them from a data set: for computing the truth value of the statement $I(\alpha, \beta | U \backslash \{\alpha, \beta\})$, which involves all the variables in U, we need an exponential number of calculations, even in the case of binary variables. So, Pearl and Paz's algorithm quickly becomes unsuitable for constructing SCNs from data sets, even for a moderate number of variables. On the other hand, the truth value of the statement $I(\alpha, \beta | \gamma)$ can be computed in a time proportional to the size, m, of the data set, so that the complete process can be done in $O(n^3 m)$. An additional advantage of the ET algorithm is not related to efficiency, but to reliability: the truth value of $I(\alpha, \beta | \gamma)$ can be computed much more reliably than that of $I(\alpha, \beta | U \backslash \{\alpha, \beta\})$; this fact allows us to use smaller data sets as the inputs for the learning algorithms.

If the dependency model is not isomorphic to an SCN, but it is still isomorphic to an MCN, then the output of the previous algorithm is an MCN, which is a (non-minimal) I-map of the model. The next proposition proves this assertion:

Proposition 3. *Let M be a dependency model verifying the axioms F1–F5. Then the graph G, obtained by applying the ET algorithm, is an I-map of M. Moreover, if M does not satisfy F6, then G contains at least one triangle.*

Proof. The ET algorithm is exactly Pearl and Paz's (PP) algorithm except the condition of testing when to remove an edge, i.e. ET may remove less edges than PP. So, the ET algorithm outputs a graph that is a super graph of what Pearl and Paz output and so it must be an I-map because their output is an I-map.

Suppose now that M does not verify F6. This means that we can find nodes $\alpha, \beta, \gamma \in U$ such that $\neg I(\alpha, \gamma | U \backslash \{\alpha, \gamma\})$, $\neg I(\gamma, \beta | U \backslash \{\beta, \gamma\})$ and $\neg I(\alpha, \beta | \gamma)$. Using F3, from the

Exact Tree Algorithm

1. Start with a complete undirected graph G
2. **for** every pair of nodes $\alpha, \beta \in U$ **do**
   2.1. *removed* $\leftarrow$**false**
   2.2. $S \leftarrow U \setminus \{\alpha, \beta\}$
   2.3. **while** $S \neq \emptyset$ **and** *removed* =**false do**
      2.3.1. Select a node $\gamma \in S$
      2.3.2. $S \leftarrow S \setminus \{\gamma\}$
      2.3.3. **if** $I(\alpha, \beta | \gamma)$ =**true**
         then
         2.3.3.1. Remove the edge α—β from G
         2.3.3.2. *removed* $\leftarrow$**true**
3. **return** G

Figure 2. Algorithm for learning trees or forests.

first two statements we can deduce $\neg I(\alpha, \gamma \mid \delta)\ \forall \delta \in U \setminus \{\alpha, \gamma\}$ and $\neg I(\gamma, \beta \mid \delta)\ \forall \delta \in U \setminus \{\gamma, \beta\}$. Therefore G contains the edges α–γ and γ–β. If $I(\alpha, \beta \mid \delta_0)$ for some δ_0, then by using F5, we obtain $I(\alpha, \gamma \mid \delta_0)$ or $I(\gamma, \beta \mid \delta_0)$, which is a contradiction with the previous statements. So, we have $\neg I(\alpha, \beta \mid \delta)\ \forall \delta \in U \setminus \{\alpha, \beta\}$, and this means that G also contains the edge α–β; hence G contains a triangle. □

For example, a graph-isomorphic but not tree-isomorphic dependency model, together with the I-map constructed by the ET algorithm are depicted in figure 3.

In view of the result in Proposition 3, the ET algorithm may be easily modified to output either a singly connected network isomorph to the model, if such a one exists, or acknowledgment that no such a network exists (under the assumption that M is graph-isomorphic): it is only necessary to replace step 3 in the algorithm by a check for triangular structures, returning graph G when atriangularity holds or if not, a 'fail' sign appears. As this check can be made in polynomial time (cubic, in the worst case), the complexity of the modified algorithm is still $O(n^3)$. So, if we are interested in knowing whether a dependency model (or a data set) can be exactly represented by means of a tree, this modified algorithm offers an efficient way for finding it out.

Another interesting task consists in constructing an SCN which is an approximation of the dependency model being considered. In practice, an algorithm for doing so may be more useful than the previous algorithms, because we shall seldom find real problems that exactly fit a tree structure; however, it may be quite interesting to find a reasonable tree approximation, that is easy to build and use. To carry out this task, we need to replace the idea of an independency relation being true or false by a graded dependency relation which measures the degree of dependency between two variables: we can use any dependency function $Dep(X, Y \mid Z)$, whose value is zero if $I(X, Y \mid Z)$ and such that the more dependent X and Y are, given Z, the greater $Dep(X, Y \mid Z)$ is. For example, in a probabilistic dependency model we could use the well-known Kullback–Leibler cross entropy measure (Kullback and Leibler 1951), or any other of the *Dep* functions used in Acid *et al.* 1991 (such as the L_1 or L_2 norms). The basic idea is to preserve those edges representing the strongest dependency relations, but with the restriction that the resultant structure must be singly connected (this idea was also considered by Chow and Liu 1968). For each pair of nodes $\alpha, \beta \in U$, we can calculate several degrees of conditional dependency, one for each other node in the model, $Dep(\alpha, \beta \mid \gamma), \forall \gamma \in U \setminus \{\alpha, \beta\}$. In order to have a single measure of dependency, we could aggregate them in some way; any triangular norm $\otimes$ (Schweizer and Sklar 1983), such as, for example, the minimum or the product, could be an appropriate conjunctive operator. So, we define the global degree of dependency $Dep_t(\alpha, \beta)$ by means of

$$Dep_t(\alpha, \beta) = \bigotimes_{\gamma \in U \setminus \{\alpha, \beta\}} Dep(\alpha, \beta \mid \gamma) \tag{5}$$

To preserve the strongest dependencies compatible with a singly connected structure, we can use any Maximum Weight Spanning Tree (MWST) algorithm (Aho *et al.* 1987). Taking into account these ideas, the Tree Approximation (TA) algorithm is

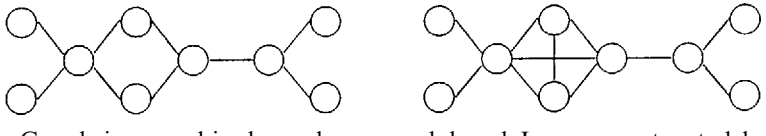

Figure 3. Graph-isomorphic dependency model and I-map constructed by the ET algorithm.

depicted in figure 4. In this algorithm $\otimes$ is a triangular norm and ε is a small non-negative real number (that may be zero) representing a threshold used to detect independency. The specific MWST algorithm used in Kruskal's (Christofides 1975).

Since the MWST algorithm (steps 3 and 4 of the algorithm in figure 4) takes, at most $O(n^2 \log(n))$, the complexity of the TA algorithm is $O(n^3)$ (or $O(n^3 m)$, if we include the cost of calculating the values of the dependency functions $Dep(\alpha, \beta | \gamma)$ from a data set of size m). The next proposition proves that this algorithm also finds the SCN isomorphic to a dependency model verifying F1–F6. Its advantage with respect to the ET algorithm is that the algorithm in figure 4 is also able to find a tree or forest approximation of any dependency model.

Proposition 4. *If M is a dependency model verifying the axioms F1–F6, then the graph G obtained by means of the TA algorithm is isomorphic to M.*

Proof. Taking into account that for any triangular norm $\otimes$, the equality $a \otimes 0 = 0$ is true for every non-negative real number a, it is clear that $Dep_t(\alpha, \beta) = 0$ if, and only if, $Dep(\alpha, \beta | \gamma) = 0$ (and consequently $I(\alpha, \beta | \gamma)$) for some node γ. Therefore the edges linking pairs of nodes α, β such that $I(\alpha, \beta | \gamma)$ will never appear in G. Thus, only the edges linking pairs of conditionally dependent nodes given any other node will appear in G ($n - 1$ edges for the case of a tree, and less than $n - 1$ for a forest). □

The TA algorithm, for the case of a probabilistic dependency model, is similar to Chow–Liu's algorithm (Chow and Liu 1968): both use an MWST algorithm where the weight of an edge, $Dep_t(\alpha, \beta)$, is a measure of the dependency degree between the linked variables (particularly the Kullback–Leibler cross entropy measure, for the case of Chow–Liu's algorithm). However, the algorithm by Chow and Liu (1968) computes the weight $Dep_t(\alpha, \beta)$ directly, it does not use a conjunction of conditional weights $Dep(\alpha, \beta | \gamma)$ but the single marginal weight $Dep(\alpha, \beta | \ )$ (and this fact gives rise to less complexity, $O(n^2 \log(n))$). Should the probabilistic dependency model be tree-isomorphic or forest-isomorphic, both algorithms return the same output, but this is not necessarily true in the general case. It would be interesting to compare these two algorithms by evaluating their performance from several points of view (e.g. success rates obtained in classification problems, as in Acid and de Campos (1995), or the

Tree Approximation Algorithm

1. Start with an empty graph G
2. **for** every pair of nodes $\alpha, \beta \in U$ **do**
   2.1. $Dep_t(\alpha, \beta) \leftarrow 1$
   2.2. **for** every node $\gamma \in U \setminus \{\alpha, \beta\}$ **do**
      2.2.1. Compute $Dep(\alpha, \beta | \gamma)$
      2.2.2. $Dep_t(\alpha, \beta) \leftarrow Dep_t(\alpha, \beta) \otimes Dep(\alpha, \beta | \gamma)$
3. Rank the $n(n-1)/2$ values $Dep_t(\alpha, \beta)$ in decreasing order
4. **repeat**
   Select the next largest value $Dep_t(\alpha, \beta)$ and add the edge α—β to G
      unless it creates a cycle in G **or** $Dep_t(\alpha, \beta) \leq \epsilon$
   until $n - 1$ edges have been added to G **or**
   the last value which has been considered verifies $Dep_t(\alpha, \beta) \leq \epsilon$
5. **return** G

Figure 4. Algorithm building an SCN approximation of a dependency model.

Table 1. Average Hamming distances between the true and learnt trees.

Sample size	*50*	*100*	*250*	*500*	*1000*	*2000*
CL	7·6	4·2	3·1	1·5	0·5	0·0
TA	8·0	4·2	3·7	1·5	0·4	0·0

robustness of the two algorithms, for different sizes of the data sets, measured as the degree of similarity between the topologies of the SCNs obtained by the algorithms and that of the underlying SCN). In the next subsection, the results of several experiments carried out with both algorithms are reported.

4.2. *Experimental results*

We have designed two different kinds of experiments to evaluate the performance of the previous approximate learning algorithms, the TA and CL (Chow–Liu) algorithms: the first one tries to give an idea of how robust the two algorithms are with respect to the size of the database used to learn the tree, i.e. how well can the two algorithms recover a true tree structure, depending on the sample size. The second experiment uses several real databases (training sets), relating to some classification problems, to learn bayesian trees (using both algorithms, TA and CL), and use them to estimate the success rates of classification on a different test set.

Now, let us describe the first experiment more precisely: we have randomly generated 10 bayesian trees (random topology and random probability tables), with each one having 10 binary variables. Next, we have obtained databases of different sizes, generated from each one of these trees using the probabilistic logic sampling technique (Henrion 1988) (the specific sizes we have used are 50, 100, 250, 500, 1000 and 2000). Then we applied the two learning algorithms to each data set to generate a learnt tree. For the TA algorithm, we used the minimum as the conjunctive operator $\otimes$, and we chose the Kullback–Leibler cross entropy as the dependency measure in both cases, i.e.

$$Dep(\alpha, \beta \mid \ ) = \sum_{\alpha_i, \beta_j} P(\alpha_i, \beta_j) \ln \left(\frac{P(\alpha_i, \beta_j)}{P(\alpha_i)\, P(\beta_j)} \right)$$

and

$$Dep(\alpha, \beta \mid \gamma) = \sum_{\alpha_i, \beta_j, \gamma_k} P(\alpha_i, \beta_j, \gamma_k) \ln \left(\frac{P(\alpha_i, \beta_j, \gamma_k)\, P(\gamma_k)}{P(\alpha_i, \gamma_k)\, P(\beta_j, \gamma_k)} \right).$$

Later we compared these learnt trees with the original ones by examining any structural differences. The method for comparing graph structures was by using the Hamming distance, i.e. the number of different edges in the learnt tree with respect to the original (either missing or added edges). The average results of the ten experiments for each sample size are summarized in table 1.

Several conclusions may be drawn from these experiments (although, given the small number of trees used for each sample size, 10, the results may be not statistically significant): first, both algorithms exhibit a very similar performance; in most cases (52 from a total of 60 experiments), CL and TA gave the same Hamming distance (in 40 of these 52 cases the outputs of the two algorithms were exactly the same), and in a few cases the Hamming distances were different (2 times TA was better than CL, and

6 times CL obtained trees having smaller Hamming distance than these generated by TA); anyway, the CL algorithm seems slightly more robust for small sample sizes. The reason may be that the CL algorithm computes the dependency degrees $Dep(\alpha, \beta \mid)$ by estimating the bidimensional probability distributions $P(\alpha, \beta)$ from the data set, whereas the TA algorithm computes the dependency degrees $Dep(\alpha, \beta \mid \gamma)$ by estimating the tri-dimensional distributions $P(\alpha, \beta, \gamma)$, which are less reliably estimated for small sample sizes. Second, from the results in table 1, we can see that both algorithms perform quite well, even for relatively small sample sizes (bear in mind that the number of distinct trees of n vertices is n^{n-2} (Christofides 1975), i.e. 10^8 in our case, so that the size of the search space is rather large).

For the other kind of experiment, which tries to evaluate the performance of the TA and CL algorithms as automatic classifiers, we selected two different classification problems: the First MONK problem (Thrun *et al.* 1991) and the Heart Disease problem (King *et al.* 1995). The two databases used are public ones, and are part of the collection of databases at the University of California, Irvine, collated by David Aha.

MONK problems rely on an artificial robot domain, in which robots are described by six different attributes (attributes A1, A2 and A4 all have 3 values, attributes A3 and A6 are binary, and attribute A5 has 4 values). The learning task is a binary classification task. Each problem is given by a logical description of a class. Robots either belong to this class or not, but instead of providing a complete class description for the learning problem, only a subset of all 432 possible robots with its classification is given. We used the First MONK problem, whose class description is: '*the values of attributes A*1 *and A*2 *are equal, or attribute A*5 *takes its first value*'. From 432 possible examples, 124 were randomly selected for the training set. The test set consisted of all 432 examples. The trees learnt by TA and CL from the training set are depicted in figure 5, and the corresponding confusion matrices are shown in table 2. The row of each entry represents the actual classification and the column represents the predicted classification.

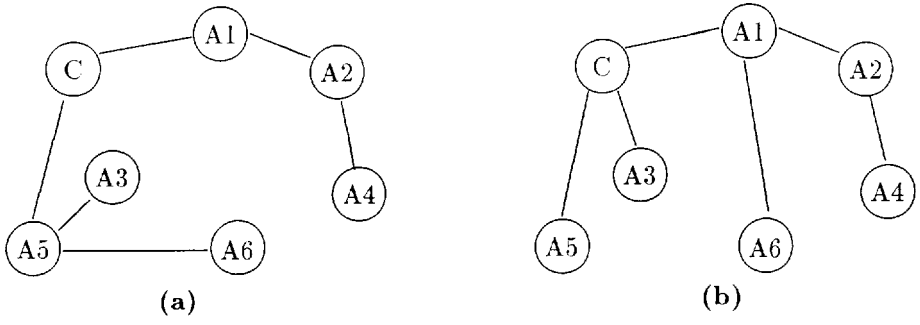

Figure 5. Trees obtained for the MONK problem by (*a*) TA and (*b*) CL.

Table 2. Confusion matrices for TA and CL on the MONK problem.

TA	*Class 0*	*Class 1*	**CL**	*Class 0*	*Class 1*
Class 1	144	72	Class 0	156	60
Class 1	72	144	Class 1	78	138

Table 3. Confusion matrix for TA and CL, for the Heart test set.

TA & CL	*Class 1*	*Class 2*
Class 1	15	1
Class 2	5	9

Table 4. Confusion matrices for TA and CL, for the Heart complete set.

CL	*Class 1*	*Class 2*	**TA**	*Class 1*	*Class 2*
Class 1	132	18	Class 1	133	17
Class 2	21	99	Class 2	22	98

It may be observed that the results obtained by the two algorithms are rather poor (although CL performs slightly better than TA). The overall success rates for the test set are 66·66% and 68·05% for TA and CL, respectively. It is clear that this problem cannot be appropriately approximated by a dependency tree.

In the other classification problem, Heart Disease, the purpose is to predict the presence or absence of heart disease given the results of various medical tests carried out on a patient. The original database comes from the Cleveland Clinic Foundation and was supplied by Robert Detrano, MD PhD of the V. A. Medical Center, Long Beach, CA. We used the same database employed within the Statlog Project (King *et al.* 1995, Michie *et al.* 1994), which contains 13 attributes (and the class variable) and 270 examples. There are 5 continuous attributes, that were discretized in four categories using quartiles. We used the first 240 cases in the database as the training set, and the last 30 cases as the test set.

The trees induced by CL and TA from the training set were not the same (they differed in 3 edges), but the success rates were identical: 80·00% for the test set and 85·60% for the complete set (training plus test). The confusion matrices are displayed in tables 3 and 4. The columns represent the predicted class, and the rows the true class.

In this case, the performance of both algorithms is quite good. The only noticeable difference between CL and TA in this problem is the number of attributes adjacent to the class variable (which are the only relevant attributes for the classification): in the tree created by TA the class variable has three adjacent nodes (*Chest pain type*, *Number of major vessels coloured by fluoroscopy*, *and Thal*), whereas there are four nodes adjacent to class in the tree induced by CL (the above three, plus *Maximum heart rate achieved*). So, TA achieves the same success rates using less information than CL.

5. Directed graphs: independency relationships in polytrees

In this section, we try to characterize dependency models isomorphic to polytrees (singly connected dags). This task is harder than its equivalent in undirected graphs, because d-separation is more difficult to manage than separation, and also because for the directed case there is no analogue to Pearl and Paz's characterization theorem (Pearl and Paz 1985). So, we shall need more axioms and more intermediate results.

Let us consider the following set of axioms:

(P1) Symmetry:

$(I(X, Y \mid Z) \Rightarrow I(Y, X \mid Z)) \forall X, Y, Z \subseteq U.$

(P2) Decomposition/composition:

$(I(X, Y \cup W \mid Z) \Leftrightarrow I(X, Y \mid Z)$ and $I(X, W \mid Z)) \forall X, Y, W, Z \subseteq U.$

(P3) Intersection:

$I(X, Y \mid Z \cup W)$ and $I(X, W \mid Z \cup Y) \Rightarrow I(X, Y \cup W \mid Z)) \forall X, Y, W, Z \subseteq U.$

(P4) Weak union:

$(I(X, Y \cup W \mid Z) \Rightarrow I(X, W \mid Z \cup Y)) \forall X, Y, W, Z \subseteq U.$

(P5) Contraction:

$(I(X, W \mid Z \cup Y)$ and $I(X, Y \mid Z) \Rightarrow I(X, Y \cup W \mid Z)) \forall X, Y, W, Z \subseteq U.$

(P6) Weak transitivity:

$(I(X, Y \mid Z)$ and $I(X, Y \mid Z \cup \gamma) \Rightarrow I(X, \gamma \mid Z)$ or

$I(\gamma, Y \mid Z)) \forall X, Y, Z \subseteq U \; \forall \gamma \in U \backslash (X \cup Y \cup Z).$

(P7) Semi-strong union:

$(I(\alpha, \beta \mid Z)$ and $\neg I(\alpha, \beta \mid \;) \Rightarrow I(\alpha, \beta \mid Z \cup W)) \forall \alpha, \beta \in U \; \forall W, Z \subseteq U \backslash \{\alpha, \beta\}.$

(P8) Semi-strong atriangularity:

$(\neg I(\alpha, \gamma \mid Z) \forall Z \subseteq U \backslash \{\alpha, \gamma\}$ and $\neg I(\gamma, \beta \mid Z) \forall Z \subseteq U \backslash \{\gamma, \beta\} \Rightarrow I(\alpha, \beta \mid \gamma)$ or

$I(\alpha, \beta \mid \;)) \forall \alpha, \beta, \gamma \in U.$

(P9) Shrinkage:

$(I(\alpha, \beta \mid Z) \Rightarrow I(\alpha, \beta \mid Z_1)$ or $I(\alpha, \beta \mid Z_2)) \forall Z_1, Z_2$ such that

$Z = Z_1 \cup Z_2) \forall \alpha, \beta \in U, \forall Z \subseteq U \backslash \{\alpha, \beta\}.$

(P10) Blocking:

$(\neg (\alpha, \gamma \mid Z) \forall Z \subseteq U \backslash \{\alpha, \gamma\}$ and $\neg I(\gamma, \beta \mid Z) \forall Z \subseteq U \backslash \{\gamma, \beta\}$

$\Rightarrow I(\alpha, \delta \mid \gamma \cup \beta)$ or $I(\delta, \beta \mid \gamma \cup \alpha)$ or $I(\alpha \cup \beta, \delta \mid \;)) \forall \delta \in U \backslash \{\alpha, \beta, \gamma\}) \forall \alpha, \beta, \gamma \in U.$

(P11) Marginal chordality:

$(I(\alpha, \beta \mid \;)$ and $I(\gamma, \delta \mid \;) \Rightarrow I(\alpha, \beta \mid \gamma)$ or $I(\alpha, \beta \mid \delta)) \forall \alpha, \beta, \gamma, \delta \in U.$

It is simple to check that any polytree, together with the d-separation criterion, defines a dependency model that satisfies P1–P11. The difficult task will be to prove the converse, namely, that every dependency model verifying P1–P11 is polytree-isomorphic.

An important property that can immediately be deduced from P6 and P7 is the following:

(P12) Semi-strong transitivity:

$(I(\alpha, \beta \mid Z)$ and $\neg I(\alpha, \beta \mid \;) \Rightarrow I(\alpha, \gamma \mid Z)$ or

$I(\gamma, \beta \mid Z) \forall \gamma \in U \backslash (Z \cup \{\alpha, \beta\})) \forall \alpha, \beta \in U \; \forall Z \subseteq U \backslash \{\alpha, \beta\}.$

Another useful property that can easily be deduced from P9 (using induction) is the following:

(P13) Singularity:

$$(I(\alpha,\beta|Z), Z \neq \emptyset \Rightarrow \exists \gamma \in Z \text{ such that } I(\alpha,\beta|\gamma)) \forall \alpha,\beta \in U \; \forall Z \in U\backslash\{\alpha,\beta\}.$$

Axioms P1–P6 are well-known, and it has been shown (Pearl 1988) that they constitute a necessary condition for a dependency model to be dag-isomorphic. The meaning of axiom P6 is the following: if X and Y are each dependent on γ, then they must also be dependent on each other, either marginally or conditionally, given γ. In (Pearl 1988), another axiom was also considered, namely chordality:

(P14) Chordality:

$$(I(\alpha,\beta|\gamma \cup \delta) \text{ and } I(\gamma,\delta|\alpha \cup \beta) \Rightarrow I(\alpha,\beta|\gamma) \text{ or } I(\alpha,\beta|\delta)) \forall \alpha,\beta,\delta \in U.$$

In our case this axiom will not be necessary, because it is clearly implies by singularity.

Although P1–P6 are needed in order to obtain a dag-isomorphic dependency model, they are not sufficient. Moreover, it has been reported (Geiger 1987) that dag-isomorphic models are non-axiomatizable by a bounded set of Horn clauses, which suggests that the number of axioms required for a complete characterization of the d-separation in dags is probably unbounded (Pearl 1988). However, we are going to prove that some more restricted models, namely polytree-isomorphic models, can be fully characterized by a finite number of axioms: only five additional axioms, P7–P11, are needed (although, as occurs with P6, P7–P11 are not Horn clauses). Axiom P7 is similar to strong union (F3) but a bit weaker: P7 demands marginal dependency as a premise to increase the conditioning set without destroying independency. Axioms P8 is similar to atriangularity (F6) but it weakens the conclusion to allow the two possible independency patterns among three variables which do not form a triangle: marginal independency of two variables or conditional independency of two variables, given the third one. Axiom P9 establishes another way to reduce the size of the separating set, which is different from contraction: given any covering of the separating set Z, then at least one of the subsets in the covering is still a separating set. P13 represents an extreme case of this property, where the subsets in the covering are all singletons. Note that this property of singularity was deduced from the other axioms in the undirected case (Proposition 2). In the directed case this is no longer true, and it is even necessary to impose a stronger condition. Note also that singularity, together with composition/decomposition, asserts that single variables are sufficient to describe all the independency statements. It is difficult to explain in a few words the meaning of axiom P10; it is related to the following idea: in a dag, two adjacent nodes in a chain always block the chain, regardless of the direction of the arrows. Finally, P11 is quite similar to chordality (P14), but it substitutes the conditioning sets in the antecedents of P14 by the empty set, hence the name of marginal chordality.

As we did for the undirected case, we are going to build a graph (a dag) that represents a dependency model verifying, in this case, the axioms P1–P11. The idea is similar: first, we construct an undirected graph whose edges connect every pair of variables which are not independent in M, given any subset of U; second, we give direction to the edges.

Using semi-strong union, it is easy to show that

$$\neg I(\alpha,\beta|Z) \; \forall Z \subseteq U\backslash\{\alpha,\beta\} \Leftrightarrow \neg I(\alpha,\beta|U\backslash\{\alpha,\beta\}) \text{ and } \neg I(\alpha,\beta|\emptyset). \tag{6}$$

Therefore, the axioms P8 and P10 can be reformulated in the following way:

(P8) Semi-strong atriangularity:

$(\neg I(\alpha, \gamma | U\backslash\{\alpha, \gamma\}), \neg I(\alpha, \gamma | \ ), \neg I(\gamma, \beta | U\backslash\{\beta, \gamma\})$ and

$\neg I(\gamma, \beta | \ ) \Rightarrow I(\alpha, \beta | \gamma)$ or $I(\alpha, \beta | \ )) \forall \alpha, \beta, \gamma \in U$.

(P10) Blocking:

$(\neg I(\alpha, \gamma | U\backslash\{\alpha, \gamma\}), \neg I(\alpha, \gamma | \ ), \neg (\gamma, \beta | U\backslash\{\beta, \gamma\})$ and

$\neg I(\gamma, \beta | \ ) \Rightarrow I(\alpha, \delta | \gamma \cup \beta)$ or $I(\delta, \beta | \gamma \cup \alpha)$ or

$I(\alpha \cup \beta, \delta | \ ) \forall \delta \in U\backslash\{\alpha, \beta, \gamma\}) \forall \alpha, \beta, \gamma \in U$.

Taking into account equation (6), we define the skeleton of a dependency model as follows:

Definition 2. *Given a dependency model $M = (U, I)$ verifying the axioms P1–P11, the skeleton of M is the undirected graph $G_M = (U, E_M)$, where the set of edges E_M is*

$$E_M = \{(\alpha, \beta) | \alpha, \beta \in U, \neg I(\alpha, \beta | U\backslash\{\alpha, \beta\}), \neg I(\alpha, \beta | \ )\}. \tag{7}$$

In order to give direction to the skeleton, we proceed as follows: given any pair of adjacent edges $(\alpha, \gamma), (\gamma, \beta) \in E_M$, $\alpha \rightarrow \gamma$ and $\beta \rightarrow \gamma$ are arrows in the dag if, and only if, $I(\alpha, \beta | \ )$ (we say that γ is a head to head node); next, the rest of the edges must be directed, with the restriction of not creating more head to head nodes. We may have some degree of freedom to complete the process of giving direction to the remaining edges: some of these edges will be unambiguously directed, but some others may have any direction. In the following definition, we describe this process more formally:

Definition 3. *Let $M = (U, I)$ be a dependency model verifying the axioms P1–P11, and let $G_M = (U, E_M)$ be its skeleton. Let us define*:

- *The set B_M of basic arrows of G_M as*
  $B_M = \{\alpha \rightarrow \gamma | (\alpha, \gamma) \in E_M, \exists \beta \in U\backslash\{\alpha, \gamma\}$ *such that* $(\gamma, \beta) \in E_M$ *and* $I(\alpha, \beta | \ )\}$
- *The set R_M of remaining edges of G_M as*
  $R_M = \{(\alpha, \gamma) \in E_M | \alpha \rightarrow \gamma \notin B_M$ *and* $\gamma \rightarrow \alpha \notin B_M\}$
- *A set of arrows C_M is said to be a set of arrows compatible with B_M if, and only if,*
  - *(i)* $\forall (\alpha, \gamma) \in R_M$, *either* $\alpha \rightarrow \gamma \in C_M$ *and* $\gamma \rightarrow \alpha \notin C_M$, *or* $\alpha \rightarrow \gamma \notin C_M$ *and* $\gamma \rightarrow \alpha \in C_M$,
  - *(ii) If* $\alpha \rightarrow \gamma \in C_M$ *then* $(\alpha, \gamma) \in R_M$,
  - *(iii) If* $\alpha \rightarrow \gamma \in C_M$ *then* $\forall \delta \in U\backslash\{\alpha, \gamma\}, \delta \rightarrow \gamma \notin (C_M \cup B_M)$.
- *A (directed) graph $D_M = (U, A_M)$, where $A_M = B_M \cup C_M$ is any set of arrows compatible with B_M, is said to be a dag associated with M.*

B_M is the set of arrows that define head to head nodes; the edges in R_M must be directed taking into account that no more head to head nodes can be formed; each set C_M represents a choice that completes the assignment of directions in a way consistent with this restriction: the condition (i) ensures that all the edges in R_M are directed one-way; the condition (ii) guarantees that only the edges in R_M are directed; and finally, the condition (iii) guarantees that we do not create more head to head nodes. Any dag containing all the basic arrows and a set of compatible arrows is a dag associated with M. All the different dags associated with a dependency model are equivalent (Verma and Pearl 1990a).

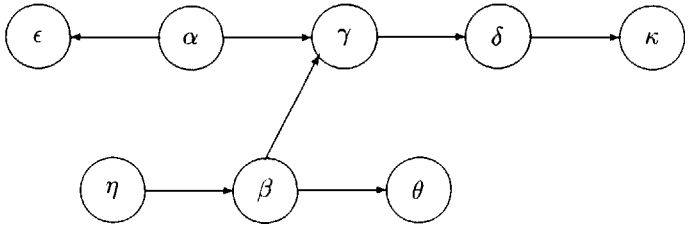

Figure 6. Dag D including a dependency model M.

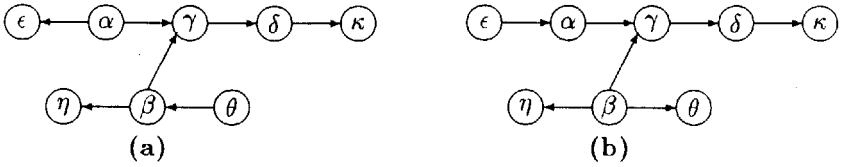

Figure 7. Dags associated with (*a*) C^1_M and (*b*) C^2_M.

For example, consider the dependency model M induced by the dag D in figure 6 through the d-separation criterion. In this case we have:

$$E_M = \{(\alpha, \varepsilon), (\alpha, \gamma), (\beta, \gamma), (\beta, \eta), (\beta, \theta), (\gamma, \delta), (\delta, \kappa)\}$$
$$B_M = \{\alpha \rightarrow \gamma, \beta \rightarrow \gamma\}$$
$$R_M = \{(\alpha, \varepsilon), (\beta, \eta), (\beta, \theta), (\gamma, \delta), (\delta, \kappa)\}$$

Observe that the edges (γ, δ) and (δ, κ) have to be necessarily directed as $\gamma \rightarrow \delta$ and $\delta \rightarrow \kappa$. However, the other edges in R_M may have more than one direction. There are six different sets of compatible arrows; so, there are six different dags associated with M (including D). Figure 7 shows two of them, corresponding to the sets $C^1_M = \{\gamma \rightarrow \delta, \delta \rightarrow \kappa, \alpha \rightarrow \varepsilon, \beta \rightarrow \eta, \theta \rightarrow \beta\}$ and $C^2_M = \{\gamma \rightarrow \delta, \delta \rightarrow \kappa, \varepsilon \rightarrow \alpha, \beta \rightarrow \eta, \beta \rightarrow \theta\}$.

Now, we have to prove three main results: first, that the concept of dag D_M associated to a dependency model M verifying the axioms P1–P11 is well-defined (i.e. that definition 3 actually defines a dag); second, that the independency relationships in M are equivalent to d-separation relationships in D_M, and third that D_M is in fact a polytree. The rest of this section is devoted to showing the truth of these assertions.

The next two lemmas establish some basic properties which are necessary for subsequent development.

Lemma 1. *Given a dependency model, M, verifying P1–P11, and its skeleton G_M, if $\alpha\gamma\beta$ is a chain in G_M, then it is either $I(\alpha, \beta \mid \gamma)$ or $I(\alpha, \beta \mid \theta)$ but the two statements cannot both be true.*

Lemma 1 states that if a node γ in a chain $\alpha\gamma\beta$ is not head to head (i.e. $\neg I(\alpha, \beta \mid)$), then its instantiation makes α and β independent. This creates a clear distinction between head to head nodes (that allow us to define the basic arrows) and not head to head nodes, in terms of the different independency relationships they represent: marginal independency or conditional independency, respectively.

Lemma 2. *Given a dependency model, M, verifying P1–P11, and its skeleton G_M, if $t_1 t_2 \cdots t_{n-1} t_n$ is a chain in G_M such that $I(t_{i-1}, t_{i+1} \mid t_i) \forall i = 2, \ldots, n-1$, then*

(*a*) $I(t_1, t_n \mid t_i), \forall i = 2, \ldots, n-1$.
(*b*) $\neg I(t_1, t_n \mid Z), \forall Z \subseteq U \backslash \{t_1, \ldots, t_n\}$.

Lemma 2 states that whenever a chain does not contain head to head nodes, the extreme nodes of the chain are independent, given any intermediate node. It also asserts that the instantiation of elements outside a chain without head to head nodes does not make the chain's two extreme nodes independent; in particular, the extreme nodes are not marginally independent ($\neg I(t_1, t_n \mid)$).

Now, we are in a position to prove that definition 3 is consistent, i.e. it always defines a dag:

Proposition 5. *Let $M = (U, I)$ be a dependency model verifying the axioms P1–P11, and let G_M be its skeleton. Then the graph $D_M = (U, A_M)$, where $A_M = B_M \cup C_M$, B_M is the set of basic arrows of G_M, and C_M is any set of arrows compatible with B_M, is a directed acyclic graph.*

Proof. If we examine Definition 3, we may observe that there are only three situations that could prevent D_M from being a dag: first, some basic arrows could be bidirected, i.e. $\alpha \rightarrow \gamma \in B_M$ and $\gamma \rightarrow \alpha \in B_M$ (however, observe that none of the arrows in C_M can be bidirected). Second, if G_M has a cycle and no edge in this cycle is found to be a basic arrow, then this would lead to a directed cycle in D_M, because of the restriction of not creating any other head to head nodes than those defined by the basic arrows. Third, we could find no set C_M verifying the restriction above. We will prove that none of these three cases appears.

Let us start with the first case: the only way to get a basic arrow bidirected is to have a chain $\delta\alpha\gamma\beta$ in G_M such that $I(\alpha, \beta \mid)$ and $I(\delta, \gamma \mid)$. In that case we would direct the edges as $\alpha \rightarrow \gamma$, $\beta \rightarrow \gamma$, $\delta \rightarrow \alpha$ and $\gamma \rightarrow \alpha$. Let us show that this situation is not possible: by applying (P10) to the chain $\alpha\gamma\beta$ we know that $I(\alpha, \delta \mid \gamma \cup \beta)$ or $I(\delta, \beta \mid \gamma \cup \alpha)$ or $I(\alpha \cup \beta, \delta \mid)$. The first and third possibilities are false because α and δ are adjacent. Therefore we have $I(\delta, \beta \mid \gamma \cup \alpha)$, and using singularity we achieve $I(\delta, \beta \mid \gamma)$ or $I(\delta, \beta \mid \alpha)$. On the other hand, since α and δ are adjacent we have $\neg I(\alpha, \delta \mid \gamma)$, and as we also have $I(\alpha, \beta \mid)$, from Lemma 1 we deduce $\neg I(\alpha, \beta \mid \gamma)$. Now, by using weak transitivity, we obtain $\neg I(\delta, \beta \mid \gamma)$ or $\neg I(\delta, \beta \mid \gamma \cup \alpha)$. So, we have $\neg I(\delta, \beta \mid \gamma)$ and therefore $I(\delta, \beta \mid \alpha)$. Now, from $I(\delta, \beta \mid \alpha)$ and $I(\delta, \beta \mid \gamma \cup \alpha)$ we obtain $I(\delta, \gamma \mid \alpha)$ or $I(\gamma, \beta \mid \alpha)$, using weak transitivity. Finally, as γ and β are adjacent, we have $\neg I(\gamma, \beta \mid \alpha)$ and thus $I(\delta, \gamma \mid \alpha)$. Now, from Lemma 1 we get $\neg I(\delta, \gamma \mid)$, which contradicts the hypothesis. What we have proven is that a node adjacent to a head to head node cannot be head to head as well.

Now, let us look at the second case: suppose that the skeleton G_M contains a cycle $t_1 t_2 \cdots t_{n-1} t_n t_1$ and that there is no head to head node in the cycle, i.e. $\neg I(t_{i-1}, t_{i+1} \mid)$, $\forall i = 2, \ldots, n-1$, $\neg I(t_{n-1}, t_1 \mid)$ and $\neg I(t_n, t_2 \mid)$. Then, by using Lemma 1, we obtain $I(t_{i-1}, t_{i+1} \mid t_i), \forall i = 2, \ldots, n-1, I(t_{n-1}, t_1 \mid t_n)$ and $I(t_n, t_2 \mid t_1)$. So, we can apply Lemma 2(*a*) to the chain $t_1 t_2 \cdots t_{n-1} t_n$, and we get $I(t_1, t_n \mid t_i)$. But this means that the nodes t_1 and t_n cannot be adjacent in G_M, which contradicts the hypothesis.

Finally, let us study the third case: the only situation in which is not possible to complete the assignment of directions to the edges in R_M without introducing more head to head nodes is the following: we have a chain without head to head nodes, $\alpha t_1 t_2 \cdots t_n \gamma$, in G_M (i.e. $\neg I(t_{i-1}, t_{i+1} \mid)$, $\forall i = 2, \ldots, n-1, \neg I(\alpha, t_2 \mid), \neg I(t_{n-1}, \gamma \mid)$), and the extreme edges in this chain have been directed as $\alpha \rightarrow t_1$ and $\gamma \rightarrow t_n$. But in this case, these arrows must have been directed because there are nodes β and δ such that the chains $\alpha t_1 \beta$ and $\gamma t_n \delta$ are in G_M, and moreover $I(\alpha, \beta \mid)$ and $I(\gamma, \delta \mid)$. In these circumstances, by using marginal chordality (P11), we obtain $I(\alpha, \beta \mid \gamma)$ or $I(\alpha, \beta \mid \delta)$.

If $I(\alpha, \beta \mid \gamma)$ (the other case is completely analogous), this statement, together with $I(\alpha, \beta \mid\ )$, produces $I(\alpha, \gamma \mid\ )$ or $I(\gamma, \beta \mid\ )$ by using weak transitivity; in either case, between α and γ (or between β and γ) there is a chain without head to head nodes, and then Lemma 2 (b) asserts $\neg I(\alpha, \gamma \mid\ )$ ($\neg I(\gamma, \beta \mid\ )$, respectively); hence we obtain a contradiction in every case. □

In the light of the result above, we can rightly speak about a dag D_M associated to a dependency model M verifying P1–P11. The next task will be to show that M is isomorphic to D_M. To do so, we still need a previous result, which is stated in the following lemma:

Lemma 3. *Given a dependency model, M, verifying P1–P11, and an associated dag D_M, if $I(\alpha, \beta \mid \gamma)$ and $\neg I(\alpha, \beta \mid\ )$, then there is a chain in D_M linking α and β which does not have head to head nodes and does contain γ.*

The previous lemma allows us to directly prove a partial result about the isomorphism between the model M and its associated dag D_M, that we could call marginal isomorphy. Moreover, this result will make it possible to prove the full isomorphism between M and D_M in a relatively simple way.

Lemma 4. *Let M be a dependency model verifying P1–P11, and let D_M be an associated dag. Then*

$$\forall \alpha, \beta \in U\ (I(\alpha, \beta \mid\ ) \Leftrightarrow \langle \alpha, \beta \mid\ \rangle_{D_M}).$$

Now, we are ready to prove the equivalence between independence statements in the model and d-separation statements in the graph.

Theorem 4. *Let M be a dependency model verifying P1–P11, and let D_M be an associated dag. Then*

$$\forall X, Y, Z \subseteq U, (I(X, Y \mid Z) \Leftrightarrow \langle X, Y \mid Z \rangle_{D_M}).$$

Proof. First, it is clear that this is enough to prove

$$I(\alpha, \beta \mid Z) \Leftrightarrow \langle \alpha, \beta \mid Z \rangle_{D_M}, \forall Z \in U \backslash \{\alpha, \beta\} \forall \alpha, \beta \in U$$

because of the composition/decomposition property (P2).

From the previous lemma, we know that $I(\alpha, \beta \mid\ ) \Leftrightarrow \langle \alpha, \beta \mid\ \rangle_{D_M}$. The proof considers two different cases: the two variables α and β are marginally independent ($I(\alpha, \beta \mid\ )$) or they are not ($\neg I(\alpha, \beta \mid\ )$). In the first case, the result is proven by ascending induction, whereas in the second case it is proven using descending induction, on the size of the separating set Z.

Suppose that $\neg I(\alpha, \beta \mid\ )$ (and therefore we also have $\neg \langle \alpha, \beta \mid\ \rangle_{D_M}$): we are going to prove that $\langle \alpha, \beta \mid Z \rangle_{D_M} \Rightarrow I(\alpha, \beta \mid Z)$, by using descending induction.

If $|Z| = n - 2$, i.e. $Z = U \backslash \{\alpha, \beta\}$, we have $\langle \alpha, \beta \mid U \backslash \{\alpha, \beta\} \rangle_{D_M}$. If $\neg I(\alpha, \beta \mid U \backslash \{\alpha, \beta\})$, as we also have $\neg I(\alpha, \beta \mid\ )$, then α and β are adjacent in D_M, and therefore $\neg \langle \alpha, \beta \mid U \backslash \{\alpha, \beta\} \rangle_{D_M}$, which contradicts the hypothesis.

Now, let us suppose that the result is true for every set S of size $k + 1 \leqslant |S| \leqslant n - 2$, and let Z be a set such that $|Z| = k$ and $\langle \alpha, \beta \mid Z \rangle_{D_M}$. Let γ be any element that does not belong to Z. From $\langle \alpha, \beta \mid Z \rangle_{D_M}$ and $\neg \langle \alpha, \beta \mid\ \rangle_{D_M}$, and using semi-strong union, we obtain $\langle \alpha, \beta \mid Z \cup \gamma \rangle_{D_M}$. By using semi-strong transitivity we also get $\langle \alpha, \gamma \mid Z \rangle_{D_M}$ or $\langle \gamma, \beta \mid Z \rangle_{D_M}$, and using composition we have $\langle \alpha, \beta \cup \gamma \mid Z \rangle_{D_M}$ or $\langle \alpha \cup \gamma, \beta \mid Z \rangle_{D_M}$. Now, using weak union we deduce $\langle \alpha, \gamma \mid Z \cup \beta \rangle_{D_M}$ or $\langle \gamma, \beta \mid Z \cup \alpha \rangle_{D_M}$. As the size of the set $Z \cup \gamma$ is $k + 1$, we can apply the induction hypothesis and obtain $I(\alpha, \beta \mid Z \cup \gamma)$.

The size of the sets $Z \cup \alpha$ and $Z \cup \beta$ is also $k+1$. So, if it should be $\neg \langle \alpha, \gamma | \ \rangle_{D_M}$ and $\neg \langle \gamma, \beta | \ \rangle_{D_M}$, we could also apply the induction hypothesis and obtain either $I(\alpha, \gamma | Z \cup \beta)$ or $I(\gamma, \beta | Z \cup \alpha)$. Now, using intersection, we get $I(\alpha, \beta \cup \gamma | Z)$ or $I(\alpha \cup \gamma, \beta | Z)$; in any case, from decomposition we obtain $I(\alpha, \beta | Z)$.

Should we have either $\langle \alpha, \gamma | \ \rangle_{D_M}$ or $\langle \gamma, \beta | \ \rangle_{D_M}$, we could not apply the induction hypothesis. But we can follow the following reasoning: from $I(\alpha, \beta | Z \cup \gamma)$, we obtain $I(\alpha, \beta | Z)$ or $I(\alpha, \beta | \gamma)$ by using shrinkage; in the first case we get the desired result directly; in the second case, from $I(\alpha, \beta | \gamma)$ and either $\langle \alpha, \gamma | \ \rangle_{D_M}$ or $\langle \gamma, \beta | \ \rangle_{D_M}$ (which imply either $I(\alpha, \gamma | \ )$ or $I(\gamma, \beta | \ )$), we obtain, by using contraction and decomposition, $I(\alpha, \beta | \ )$, which contradicts the hypothesis. So, this second case cannot occur.

Therefore, we have proven that $\langle \alpha, \beta | Z \rangle_{D_M} \Rightarrow I(\alpha, \beta | Z)$. The converse implication may be proven in a completely similar way.

Suppose that $I(\alpha, \beta | \ )$ (and therefore we also have $\langle \alpha, \beta | \ \rangle_{D_M}$): we are going to prove that $I(\alpha, \beta | Z) \Rightarrow \langle \alpha, \beta | Z \rangle_{D_M}$, by using ascending induction.

If $|Z| = 1$, i.e. $Z = \{\gamma\}$, we have $I(\alpha, \beta | \gamma)$ and $I(\alpha, \beta | \ )$. By using weak transitivity we obtain $I(\alpha, \gamma | \ )$ or $I(\gamma, \beta | \ )$. But in this case we also have $\langle \alpha, \gamma | \ \rangle_{D_M}$ or $\langle \gamma, \beta | \ \rangle_{D_M}$. Now, by using composition we obtain $\langle \alpha, \beta \cup \gamma | \ \rangle_{D_M}$ or $\langle \alpha \cup \gamma, \beta | \ \rangle_{D_M}$; in any case, from weak union, we obtain $\langle \alpha, \beta | \gamma \rangle_{D_M}$.

Now, let us suppose that the result is true for every set S of size $1 \leqslant |S| \leqslant k$, and let Z be a set such that $|Z| = k+1$ and $I(\alpha, \beta | Z)$. Let z_1, z_2 be any two elements in Z, and $Z_1 = Z \backslash \{z_1\}, Z_2 = Z \backslash \{z_2\}$. It is clear that $Z = Z_1 \cup Z_2$. Therefore, by using shrinkage we get $I(\alpha, \beta | Z_1)$ or $I(\alpha, \beta | Z_2)$. Let us suppose that it is $I(\alpha, \beta | Z_1)$ (the other case is completely analogous). So, by using the induction hypothesis, we deduce $\langle \alpha, \beta | Z_1 \rangle_{D_M}$. On the other hand, from $I(\alpha, \beta | Z)$, $I(\alpha, \beta | Z_1)$ and weak transitivity, we obtain $I(\alpha, z_1 | Z_1)$ or $I(z_1, \beta | Z_1)$. Suppose that $I(\alpha, z_1 | Z_1)$ (once again, the other case is analogous). If $I(\alpha, z_1 | \ )$, using the induction hypothesis, we deduce $\langle \alpha, z_1 | Z_1 \rangle_{D_M}$; otherwise, if $\neg I(\alpha, z_1 | \ )$, then the first part of this proof also ensures the same result $\langle \alpha, z_1 | Z_1 \rangle_{D_M}$. Now, from $\langle \alpha, \beta | Z_1 \rangle_{D_M}$ and $\langle \alpha, z_1 | Z_1 \rangle_{D_M}$ we obtain $\langle \alpha, \beta \cup z_1 | Z_1 \rangle_{D_M}$ by composition, and $\langle \alpha, \beta | Z_1 \cup z_1 \rangle_{D_M} \equiv \langle \alpha, \beta | Z \rangle_{D_M}$ by weak union.

So, we have proven that $I(\alpha, \beta | Z) \Rightarrow \langle \alpha, \beta | Z \rangle_{D_M}$. The proof for the opposite result, namely $\langle \alpha, \beta | Z \rangle_{D_M} \Rightarrow I(\alpha, \beta | Z)$, is completely similar. □

The only remaining task is to prove that the dags associated to dependency models verifying P1–P11 are always singly connected.

Theorem 5. *Let M be a dependency model verifying P1–P11, and let D_M be an associated dag. Then D_M is a polytree (D_M has no undirected cycle).*

Proof. From Proposition 5 we know that D_M is a dag, and from Theorem 4 we know that the independency statements in M are equivalent to the d-separation statements in D_M. Let us suppose that D_M is not a polytree. Then D_M has at least one undirected cycle, and this cycle must have at least one head to head node, γ; let α and β be the parents of γ in the cycle, and let δ be the other node adjacent to α in the cycle. If $\delta = \beta$, then we would have a triangle in D_M, and this is not possible because of semi-strong atriangularity. So, we deduce that $\delta \neq \beta$. As we have the chain $\delta\alpha\gamma\beta$ in D_M and γ is a head to head node, we know that α cannot be head to head (this was stated in the proof for Proposition 5); so, we deduce $\neg I(\delta, \gamma | \ )$ and $I(\delta, \gamma | \alpha)$.

Now, we distinguish two cases: if in the chain linking δ and γ (that does not pass through α but passes through β) there is no head to head node, then we have a chain linking δ and γ which is not blocked by α, and this implies $\neg \langle \delta, \gamma | \alpha \rangle_{D_M}$ and $\neg I(\delta, \gamma | \alpha)$,

which is a contradiction. Otherwise, in the chain linking δ and γ there are head to head nodes. Let W be the set containing all these head to head nodes. Then we have $\neg \langle \delta, \gamma | W \cup \alpha \rangle_{D_M}$ and $\neg I(\delta, \gamma | W \cup \alpha)$. However, from $I(\delta, \gamma | \alpha)$ and $\neg I(\delta, \gamma |\ )$ we deduce $I(\delta, \gamma | W \cup \alpha)$ by using semi-strong union, and once again we get a contradiction. Therefore D_M cannot contain any cycle. □

Theorem 6. *A dependency model is polytree-isomorphic if, and only if, it verifies* P1–P11.

The proof follows directly from Theorems 4 and 5.

With respect to the minimality of the set of axioms characterizing polytree-isomorphic dependency models, in order to prove that P1–P11 constitute such a minimal set, and bearing in mind the result of theorem 6, it would be suffice to find, for each axiom Pk, a dependency model verifying all the axioms P1–P11, except Pk. Alternatively, to show that the set P1–P11 is not minimal, we could try to prove that one of these axioms may be deduced from the others. So far, neither of these two approaches has been successful, hence we can only ponder on the minimality of the set P1–P11, but no proof is in sight.

6. Directed graphs: learning algorithms

In this section, we are going to follow, for directed graphs, a development similar to that carried out in Section 4 for undirected graphs: first, we develop exact and approximate algorithms for learning singly connected dags, and second, the results of several experiments carried out with these algorithms are reported.

6.1. *Algorithms for learning polytrees*

Starting out from the results obtained in Section 5, the next proposition will establish the basic property needed to develop an efficient algorithm for learning polytrees:

Proposition 6. *Let M be a dependency model verifying the axioms* P1–P11. *Let D_M be a directed graph associated with M, and G_M its skeleton. Then, for every pair $\alpha, \beta \in U$,*

$$\text{the edge } \alpha\text{–}\beta \text{ is in } G_M \Leftrightarrow \neg I(\alpha, \beta | \gamma)\ \forall \gamma \in U \setminus \{\alpha, \beta\}, \text{ and } \neg I(\alpha, \beta |\ ).$$

Proof. The edge α–β is in G_M if, and only if, $\neg I(\alpha, \beta | U \setminus \{\alpha, \beta\})$ and $\neg I(\alpha, \beta |\ )$. But we also have

$$\neg I(\alpha, \beta | U \setminus \{\alpha, \beta\}) \text{ and } \neg I(\alpha, \beta |\ ) \Leftrightarrow \neg I(\alpha, \beta | \gamma)\ \forall \gamma \text{ and } \neg I(\alpha, \beta |\ ),$$

the necessary condition being deduced from semi-strong union (P7) and the sufficient condition from singularity (P13) (which is a consequence of (P9)). □

This result allows us to design an efficient Exact Polytree (EP) algorithm to recover a directed graph associated with a dependency model verifying P1–P11, which is shown in figure 8. The algorithm first finds the skeleton of the graph by removing edges from an initially complete undirected graph G: for each pair of variables, it iteratively checks the marginal independency and the conditional independency of these two variables, given any other single variable, and removes the corresponding edge from G if at least one of these relationships is found to be true. Next, the algorithm finds the head to head patterns by testing for marginal independency of any pair of variables having a common adjacent variable. Finally, the remaining edges are directed without introducing directed cycles or new head to head connections.

Exact Polytree Algorithm

1. Start with a complete undirected graph G
2. **for** every pair of nodes $\alpha, \beta \in U$ **do**
   2.1. *removed* $\leftarrow$**false**
   2.2. $S \leftarrow U \setminus \{\alpha, \beta\}$
   2.3. **if** $I(\alpha, \beta|\emptyset)$ =**true**
   then
   2.3.1. Remove the edge α—β from G
   2.3.2. *removed* $\leftarrow$**true**
   2.4. **while** $S \neq \emptyset$ **and** *removed* =**false do**
   2.4.1. Select a node $\gamma \in S$
   2.4.2. $S \leftarrow S \setminus \{\gamma\}$
   2.4.3. **if** $I(\alpha, \beta|\gamma)$ =**true**
   then
   2.4.3.1. Remove the edge α—β from G
   2.4.3.2. *removed* $\leftarrow$**true**
3. **for** every triplet of nodes $\alpha, \beta, \gamma \in U$ such that α—γ, γ—$\beta \in G$ **do**
   3.1. **if** $I(\alpha, \beta|\emptyset)$ =**true**
   then direct the subgraph α—γ—β as $\alpha \rightarrow \gamma \leftarrow \beta$
4. Direct the remaining edges without introducing directed cycles **or** new head to head connections.
   if this is possible **then return** G, **else return 'fail'**

Figure 8. Algorithm for learning polytrees.

Proposition 7. *If M is a dependency model verifying the axioms P1–P11, then the graph G obtained by means of the EP algorithm is a polytree isomorphic to M.*

Proof. Taking into account the results in Theorems 4 and 5, it is sufficient to prove that the graph G obtained by the algorithm is a dag D_M associated with M. From Proposition 6 we deduce that the algorithm (steps 1 and 2) correctly recover the skeleton, G_M, of any associated dag, D_M. Moreover, step 3 of the algorithm produces the set B_M of basic arrows of G_M. Finally, in step 4, a set, C_M, of arrows compatible with B_M is identified. From Proposition 5, the algorithm never fails, and outputs a polytree which is one of the dags, D_M, associated with M. □

The previous proposition proves that the EP algorithm produces a polytree which is isomorphic to the dependency model if such a one exists. With respect to complexity, it is clear that the algorithm is efficient, it takes, at most, $O(n^3)$; even if we have to estimate the truth values of the independence statements from a data set, the complete process can be done in $O(n^3m)$, where m is the size of the data set. Moreover, to reliably test independency statements like $I(\alpha, \beta|\ )$ and $I(\alpha, \beta|\gamma)$, we do not need an enormous amount of data, so that the value of m may be feasible for practical situations.

The next proposition asserts that, for dag-isomorphic dependency models which are not polytree-isomorphic, the EP algorithm either returns 'fail' (meaning that the model is not polytree-isomorphic) or produces a dag (not necessarily singly connected) as output, which is at least an I-map of the model.

Proposition 8. *If M is a dependency model which is isomorphic to a dag, then the EP algorithm returns either 'fail' or a dag which is at least an I-map of M.*

Proof. If the model M is polytree-isomorphic, then Proposition 7 asserts that the algorithm does not fail and builds a dag isomorphic to M. So, let us consider a model isomorphic to a dag, D, which is not polytree-isomorphic. This means that the dag D has at least one (undirected) cycle. We shall prove that if the algorithm does not fail, then it constructs a dag, G, which contains all the edges in D (and possibly some additional edges); moreover, the head to head patterns in D will also be head to head patterns in G, and the patterns which are not head to head in D will be non-head to head in G as well. In these circumstances, G is clearly an I-map of D. First, as the algorithm only removes from G those edges α–β such that $I(\alpha, \beta|\ )$ or $I(\alpha, \beta|\gamma)$ for some node γ, then all the edges which are in the graph D remain in G.

Second, let us see that if a chain $\alpha\gamma\beta$ in D is not head to head at γ, then this chain is not head to head at γ in G: suppose that $\alpha\gamma\beta$ is head to head at γ in G. Then it is either $I(\alpha, \beta|\ )$ (but this is not possible because there is at least one chain without head to head nodes in D linking α and β) or there are nodes δ_1 and δ_2, which are adjacent to γ in G, and verifying $I(\delta_1, \alpha|\ )$ and $I(\delta_2, \beta|\ )$. But in this case, we have $\neg I(\delta_1, \gamma|\ )$ and $\neg I(\delta_2, \gamma|\ )$ (because of the adjacency of δ_1 and γ, and δ_2 and γ in G) and this means that in D there are chains without head to head nodes linking δ_1 and γ, and δ_2 and γ. On the other hand, as γ is not head to head in the chain $\alpha\gamma\beta$ in D, then at least one of the edges α–γ or γ–β must be directed as $\alpha \leftarrow \gamma$ or $\gamma \rightarrow \beta$; suppose that $\gamma \rightarrow \beta$ (the other case is analogous). Then by composing the chain without head to head nodes from δ_2 to γ with the arrow $\gamma \rightarrow \beta$ we find a chain in D without head to head nodes linking δ_2 to β, and this implies $\neg I(\delta_2, \beta|\ )$, which is a contradiction.

Third, suppose that we have one head to head pattern $\alpha \rightarrow \gamma \leftarrow \beta$ in D. We shall see that the algorithm either produces this head to head pattern in G too, or fails: if $I(\alpha, \beta|\ )$, then the algorithm clearly directs the edges α–γ and γ–β in G towards γ. If $\neg I(\alpha, \beta|\ )$, then in D there is at least one chain without head to head nodes linking α and β. If this chain contains only one arrow, i.e. α and β are adjacent in D, then the chains $\gamma\alpha\beta$ and $\gamma\beta\alpha$ are in D and are not head to head. Then, using the result stated in the second part of this proof, we can deduce that the algorithm does not create head to head connections at α and β. So, the algorithm either has to direct the edges creating the head connection at γ or has to direct in another way, thus creating a directed cycle, and returning 'fail'. If the chain linking α and β contains more than one arrow, the same reasoning may be applied: the algorithm will never create head to head connections at the nodes in this chain, and then again there are two options: creating the head to head connection at γ, or creating a directed cycle. The proof is complete. □

The previous proposition shows that, in some cases, the EP algorithm is capable of constructing a non-singly connected network which is at least an I-map of the model, even if the model is not polytree-isomorphic but it is dag-isomorphic. An example of this is shown in figure 9, which depicts a dag and the corresponding I-map constructed by the EP algorithm. It is worth noting the case of any dependency model isomorphic to a dag that contains only cycles having three or more head to head nodes (a special type of the so-called simple graphs (Geiger *et al.* 1993)). In this case, it is easy to see that any two non-adjacent nodes in the graph are either independent given the empty set or independent given some other single node. So, steps 1 and 2 of the algorithm will recover the skeleton of the graph, step 3 will correctly direct the set of basic arrows, and step 4 will orient the remaining edges without introducing new head to head connections. Therefore, the resultant graph, G, is not only an I-map, but it is also isomorphic to the model. Should we want to modify the EP algorithm, in order to

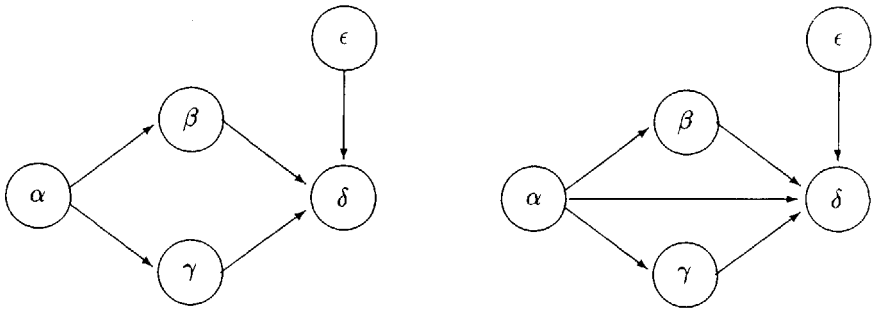

Figure 9. (*a*) Dag-isomorphic model and (*b*) I-map constructed by EP.

know whether a dependency model can be exactly represented by a polytree, we could not add a simple check for triangular structures, as we did in the undirected case: we need to add a more complex check for general undirected cycles, although the overall complexity of the modified algorithm is still $O(n^3)$.

There is another algorithm, developed by Geiger *et al.* (1990), which recovers a polytree I-map of a dependency model, if such a one exists. It is quite similar to the EP algorithm, the main practical difference being that the algorithm in Geiger *et al.* (1990) constructs the skeleton of the dag by using the independency tests $I(\alpha, \beta|\ )$ and $I(\alpha, \beta|\ U\backslash\{\alpha, \beta\})$ (this last test replaces the set of tests $I(\alpha, \beta|\ \gamma)\ \forall \gamma$ in the EP algorithm). As we commented in Section 4, using tests like $I(\alpha, \beta|\ U\backslash\{\alpha, \beta\})$ entails exponential complexity and very low reliability, if we have to estimate their true values from a data set. So, the EP algorithm may be more appropriate in these circumstances.

In order to complete the parallelism with the undirected case, we now consider the task of constructing a polytree approximation of any dependency model. As we have already said, this kind of algorithm may be more interesting in practice than the previous ones, because it can quickly find an approximate model, which is useful, at least as a first approach to the problem. We start out from the same idea of using a dependency function which measures the degree of dependency between variables. However, in this case, in the light of Proposition 6, we should employ not only the degree of conditional dependency between two variables, given a third variable, $Dep(\alpha, \beta|\ \gamma)$, but also the marginal degree of dependency $Dep(\alpha, \beta|\ )$, and then aggregate them using a triangular norm. In other words, the global degree of dependency between two variables α and β is defined in this case as

$$Dep_p(\alpha, \beta) = Dep(\alpha, \beta|\ ) \otimes \left(\underset{\gamma \in U\backslash\{\alpha,\beta\}}{\otimes} Dep(\alpha, \beta|\ \gamma) \right)$$

The proposed algorithm uses a MWST algorithm to obtain the skeleton of the polytree, like the TA algorithm does, the only difference being the different measure of global dependency used. Next the algorithm tries to direct the edges of the skeleton by using the following scheme: in a head to head pattern $\alpha \rightarrow \gamma \leftarrow \beta$, the instantiation of the head to head node γ should normally increase the degree of dependency between the variables α and β, whereas in a non-head to head pattern such as $\alpha \leftarrow \gamma \rightarrow \beta$, the instantiation of the middle node γ should produce the opposite effect, decreasing the degree of dependency between α and β. So, the idea is to compare the degree of

dependency between α and β after the instantiation of γ, $Dep(\alpha, \beta | \gamma)$, with the degree of dependency between α and β before the instantiation of γ, $Dep(\alpha, \beta | \ )$, and directing the edges towards γ if the former is greater than the latter. The proposed algorithm, called the Polytree Approximation (PA) algorithm, is depicted in figure 10.

Bearing in mind the definition of the dependency degree $Dep_p(.,.)$ that the algorithm uses, the properties of any triangular norm $\otimes$, and the result of Proposition 6, it is clear that for polytree-isomorphic dependency models the PA algorithm recovers the correct polytree (up to isomorphism). Anyway, the PA algorithm finds a polytree approximation of any dependency model. It is also obvious that the complexity of the PA algorithm is $O(n^3)$.

In order to finish this subsection, let us look at an example in which a dag-isomorph probabilistic dependency model is approximated by a polytree, using the PA algorithm. The dag representing the model (which has four binary variables) is depicted in figure 11, and the description of the quantitative part of the model (the probability distributions) is shown in table 5.

Polytree Approximation Algorithm

1. Start with an empty graph G
2. **for** every pair of nodes $\alpha, \beta \in U$ **do**
   2.1. Compute $Dep(\alpha, \beta|\emptyset)$
   2.2. $Dep_p(\alpha, \beta) \leftarrow Dep(\alpha, \beta|\emptyset)$
   2.3. **for** every node $\gamma \in U \setminus \{\alpha, \beta\}$ **do**
      2.3.1. Compute $Dep(\alpha, \beta|\gamma)$
      2.3.2. $Dep_p(\alpha, \beta) \leftarrow Dep_p(\alpha, \beta) \otimes Dep(\alpha, \beta|\gamma)$
3. Rank the $n(n-1)/2$ values $Dep_p(\alpha, \beta)$ in decreasing order
4. **repeat**
   Select the next largest value $Dep_p(\alpha, \beta)$ and add the edge α—β to G
      unless it creates a cycle in G **or** $Dep_p(\alpha, \beta) \leq \epsilon$
   until $n-1$ edges have been added to G **or**
   the last value which has been considered verifies $Dep_p(\alpha, \beta) \leq \epsilon$
5. **for** every triplet of nodes $\alpha, \beta, \gamma \in U$ such that α—γ, γ—$\beta \in G$ **do**
   5.1. **if** $Dep(\alpha, \beta|\emptyset) < Dep(\alpha, \beta|\gamma)$
      then direct the subgraph α—γ—β as $\alpha \rightarrow \gamma \leftarrow \beta$
6. Direct the remaining edges without introducing new head to head connections.
7. **return** G

Figure 10. Algorithm building a polytree approximation of a dependency model.

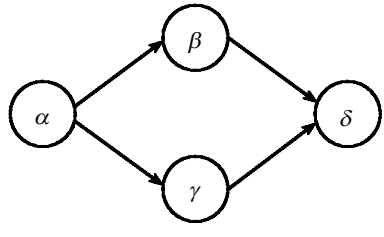

Figure 11. Dag-isomorphic dependency model D.

Table 5. Probability distributions for dag D.

$P(\alpha_1) = 0{\cdot}6$	$P(\alpha_2) = 0{\cdot}4$
$P(\beta_1 \mid \alpha_1) = 0{\cdot}4$	$P(\beta_2 \mid \alpha_1) = 0{\cdot}6$
$P(\beta_1 \mid \alpha_2) = 0{\cdot}2$	$P(\beta_2 \mid \alpha_2) = 0{\cdot}8$
$P(\gamma_1 \mid \alpha_1) = 0{\cdot}1$	$P(\gamma_2 \mid \alpha_1) = 0{\cdot}9$
$P(\gamma_1 \mid \alpha_2) = 0{\cdot}9$	$P(\gamma_2 \mid \alpha_2) = 0{\cdot}1$
$P(\delta_1 \mid \beta_1, \gamma_1) = 0{\cdot}3$	$P(\delta_2 \mid \beta_1, \gamma_1) = 0{\cdot}7$
$P(\delta_1 \mid \beta_1, \gamma_2) = 0{\cdot}1$	$P(\delta_2 \mid \beta_1, \gamma_2) = 0{\cdot}9$
$P(\delta_1 \mid \beta_2, \gamma_1) = 0{\cdot}8$	$P(\delta_2 \mid \beta_2, \gamma_1) = 0{\cdot}2$
$P(\delta_1 \mid \beta_2, \gamma_2) = 0{\cdot}2$	$P(\delta_2 \mid \beta_2, \gamma_2) = 0{\cdot}8$

The dependency measure Dep we use in this example is the L_1 norm defined by means of

$$Dep(\alpha, \beta \mid\ ) = \sum_{\alpha_i, \beta_j} \left| P(\alpha_i, \beta_j) - P(\alpha_i)\, P(\beta_j) \right|$$

and

$$Dep(\alpha, \beta \mid \gamma) = \sum_{\gamma_k} P(\gamma_k) \sum_{\alpha_i, \beta_j} \left| P(\alpha_i, \beta_j \mid \gamma_k) - P(\alpha_i \mid \gamma_k)\, P(\beta_j \mid \gamma_k) \right|$$

The values obtained for $Dep(.,. \mid\ )$, $Dep(.,. \mid .)$ and $Dep_p(.,.)$ are displayed in table 6, where we use the minimum operator as the triangular norm.

Therefore steps 1–4 of the PA algorithm produce a skeleton, which has the edges α–γ, β–δ and γ–δ. Since step 5 does not find any head to head pattern, step 6 directs all the edges without introducing head to head connections. There are several options for doing this; one of them is depicted in figure 12 (*a*). Figure 12 (*b*) shows another possible polytree approximation that has not been considered by the PA algorithm.

In order to estimate the quality of the resultant polytree as an approximation of the dag in figure 11, we have measured the distance between the joint probability distributions P and P_a, associated with the original dag and its polytree approximation given by the PA algorithm, respectively. The same procedure was carried out for P and the distribution P_b associated with the other approximation being considered. The distance measure used has also been the L_1 norm:

$$dip(P, P') = \sum_{\alpha_i, \beta_j, \gamma_k, \delta_l} \left| P(\alpha_i, \beta_j, \gamma_k, \delta_l) - P'(\alpha_i, \beta_j, \gamma_k, \delta_l) \right|$$

The results are the following: $dist(P, P_a) = 0{\cdot}095$, and $dist(P, P_b) = 0{\cdot}192$, so it is clear that the polytree obtained by the PA algorithm is better although it modifies the topology of the original dag more than the polytree in figure 12 (*b*).

There is another algorithm, developed by Rebane and Pearl (1989), that constructs polytree approximations for probabilistic dependency models and also guarantees finding a polytree isomorphic to the model if such a one exists. It uses Chow–Liu's algorithm for building the skeleton, and next directs it by using tests of marginal independency. For the model of the previous example, the algorithm in Rebane and Pearl (1989) produces the same result as the PA algorithm, although this is not always the case. In the next subsection, we compare the performance of both algorithms from an empirical point of view.

Table 6. Dependency degrees between variables.

	$Dep(.,.\mid\ )$	$Dep(.,.\mid\alpha)$	$Dep(.,.\mid\beta)$	$Dep(.,.\mid\gamma)$	$Dep(.,.\mid\delta)$	$Dep_p(.,.)$
α, β	0·192			0·071	0·064	0·064
α, γ	0.768		0.734		0.537	0.537
α, δ	0·426		0·364	0·024		0·024
β, γ	0·154	0·0			0·079	0·0
β, γ	0·284	0·198		0·203		0·198
γ, δ	0·511	0·170	0·461			0·170

6.2. *Experimental results*

We have designed three types of experiments, aiming to evaluate the behaviour of the PA and RP (Rebane–Pearl) algorithms from different points of view. The first two experiments are similar to the ones designed for the undirected case in Section 4.2: the first experiment intends to assess the robustness of the algorithms, depending on the size of the data set used for learning the polytree. The second experiment uses the polytrees learnt from a given training set to calculate the success rates for classifying instances from a different test set. The third experiment compares a multiply connected network with the singly connected approximations obtained by PA and RP.

For the first experiment, we have randomly generated 10 polytrees (random skeleton, random directions, and random probability tables), each one having 10 binary variables. Then we simulated data sets with sizes of 50, 100, 250, 500, 1000 and 2000, from each one of these polytrees, once again using probabilistic log sampling. After applying the RP and PA algorithms to each data set to obtain the learnt polytrees, we compared the resultant structures with the original ones: we measured the Hamming distances between the true and the learnt polytrees, as well as the

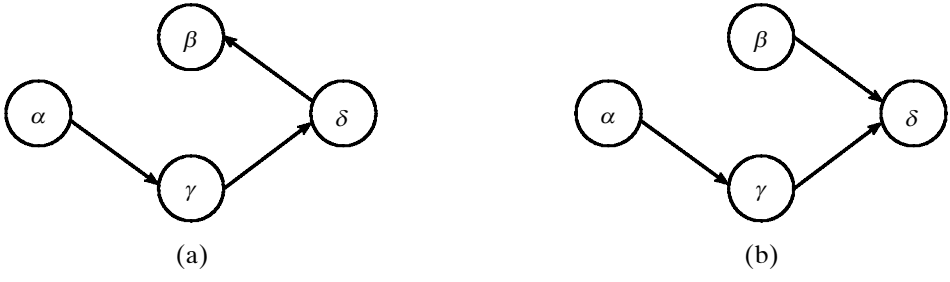

Figure 12. (*a*) Result of the PA algorithm on dag D. (*b*) Another polytree approximation of D.

Table 7. Average Hamming distances between the skeletons of the true and learnt polytrees.

Sample size	*50*	*100*	*250*	*500*	*1000*	*2000*
RP	8·4	6·5	4·7	3·7	2·7	1·3
PA	8·4	7·1	5·1	3·9	3·1	1·5

Table 8. Average Hamming distances between the true and learnt polytrees.

Sample size	*50*	*100*	*250*	*500*	*1000*	*2000*
RP	11·6	9·9	8·7	7·5	5·6	5·1
PA	10·4	9·8	7·5	6·7	5·9	2·9

Hamming distances between their skeletons (without considering the directions of the arrows). The mean values of the Hamming distances are displayed in tables 7 and 8. As in the undirected case, the dependency measure selected was the Kullback–Leibler entropy, and the conjunctive operator $\otimes$ used by the PA algorithm was the minimum.

After comparing the results displayed in tables 1 and 7, it is clear that it is easier to estimate a tree than the skeleton of a polytree (which is a tree): we obtain worse results for polytrees than for trees, using the same sample sizes (or, in other words, in order to reliably estimate the skeleton of a polytree, the number of data required is greater than for the case of trees). If we examine tables 7 and 8, we can also see that the task of directing the edges of the skeleton, in order to obtain the complete polytree, is complicated: the Hamming distances between the learnt and the original polytrees are much greater than the distances between the corresponding skeletons. These facts are not surprising, because polytrees represent models that are more complex than those represented by trees. Moreover, the erroneous estimation of some edges of the skeleton may frequently produce a cascaded erroneous estimation of the directions of other (correct) edges, which explains the higher values in table 8, with respect to those in table 7.

We can also conclude that, although the two algorithms, RP and PA, perform quite similarly, RP is slightly more robust than PA for estimating the skeleton of the polytree, the reason being, once again, that RP only needs to estimate bidimensional distributions, and PA also needs to estimate three-dimensional distributions (in 49 of the 60 experiments, the Hamming distances between skeletons were the same for both algorithms). However, with respect to the polytree itself, i.e. taking into account the direction of the arrows, the converse situation arises: the PA algorithm produces values of the Hamming distance lower than the corresponding values for the RP algorithm (and, in this case, the algorithms reveal more differences: only in 12 of the 60 cases the Hamming distances between polytrees were the same for the two algorithms, and only in 1 case were the polytrees identical).

We have also used the First MONK problem (same training and test sets used in the undirected case) for testing the performance of the PA and RP algorithms on classification problems. The polytrees obtained after running PA and RP on the training set are displayed in figure 13.

In this case the confusion matrices display a markedly different behavior. Table 9 shows these matrices. The row of each entry represents the actual classification and the column represents the predicted classification. We may note that PA behaves considerably better than RP: PA reaches a success rate of 94·44%, whereas RP only achieves 68·05%.

It is also interesting to note that by removing the weakest arcs (i.e. those arcs representing the weakest dependencies) from the polytree obtained by PA, we can improve the results: if we eliminate the weakest arc (the arc $A3 \rightarrow C$), we obtain a

success rate of 95·83%, and if we remove the two weakest arcs ($A3 \rightarrow C$ and $A6 \rightarrow A1$) we get a 100% success rate. This may be comparable to the common practice employed by many classification tree based learning algorithms for pruning the trees obtained initially.

The results obtained by PA and RP for the Heart Disease database are identical to those reported for TA and CL, respectively, in Section 4.2 (i.e. although the learnt networks were different, the success rates and confusion matrices were the same).

Finally, to test the PA and RP algorithms in more realistic and complex domains, we selected the Alarm Monitoring System (Beinlich *et al.* 1989) for the third

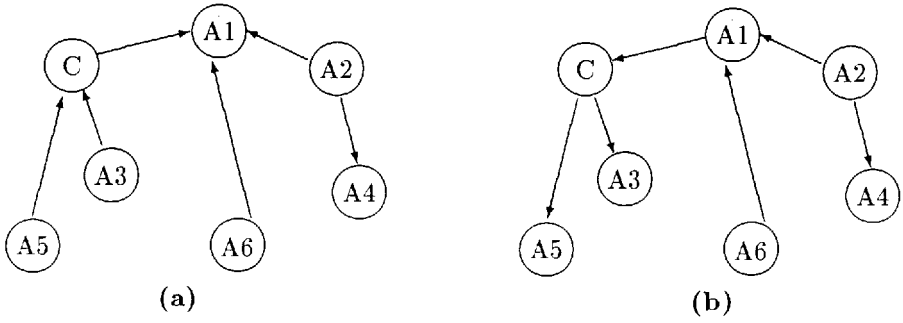

Figure 13. Polytrees obtained for the MONK problem by (*a*) PA and (*b*) RP.

Table 9. Confusion matrices for PA and RP on the MONK problem.

PA	*Class 0*	*Class 1*	**RP**	*Class 0*	*Class 1*
Class 0	192	24	Class 0	156	60
Class 1	0	216	Class 1	78	138

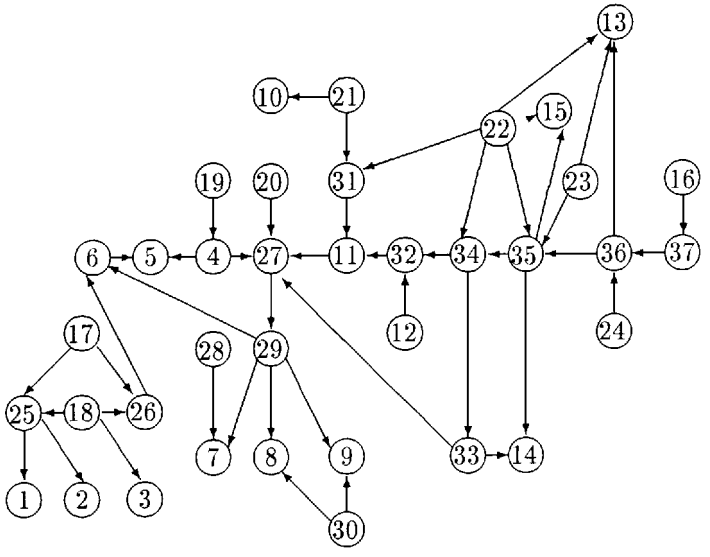

Figure 14. The Alarm network structure.

experiment. This is a diagnostic application for patient monitoring, based on belief networks. The Alarm network, which displays the relevant variables and relationships in this domain, is depicted in figure 14.

The performance of any data-driven learning algorithm on the Alarm network has become one of the standard ad hoc procedures for testing the capabilities of the algorithm. The input data commonly used are subsets of the Alarm database (Herskovits 1991), which contains 20000 cases that were stochastically generated using the Alarm network.

Given the nature of the PA and RP algorithms, we cannot expect to recover the Alarm network exactly (because it is not singly connected), but it is interesting to compare the learnt polytrees with the original network. In our experiment, we used the first 2000 cases from the Alarm database. The results obtained by PA and RP when applied to this data set are depicted in figures 15 and 16, respectively.

When comparing figures 15 and 16 with the Alarm network, we can see that the two algorithms perform well, but in this case the PA algorithm performs much better than RP: both algorithms introduce 32 correct edges and 4 incorrect edges (edges 12–14, 24–10, 9–8 and 34–15) in the skeleton; however, PA reverses only 1 arc ($8 \rightarrow 30$) with respect to the Alarm network, whereas RP reverses the directions of 12 arcs. Taking into account that the algorithms do not use any information relating to ordering variables, the number of incorrect directions is surprisingly small (especially for PA).

In order to assess the performance of the algorithms, not only from the perspective of how much of the target network is reconstructed, but also of how closely the probability distribution learnt approximates the empirical frequency distribution, we would like to compute the cross-entropy (CE) between the empirical frequency distribution, P, and each of the distributions associated to the true Alarm network (P_T) and to the networks generated by PA (P_{PA}) and RP (P_{RP}). Unfortunately, this computation is unfeasible (because it involves summing over the exponentially many atomic events). However, there is a formula (Lam and Bacchus 1994), which takes

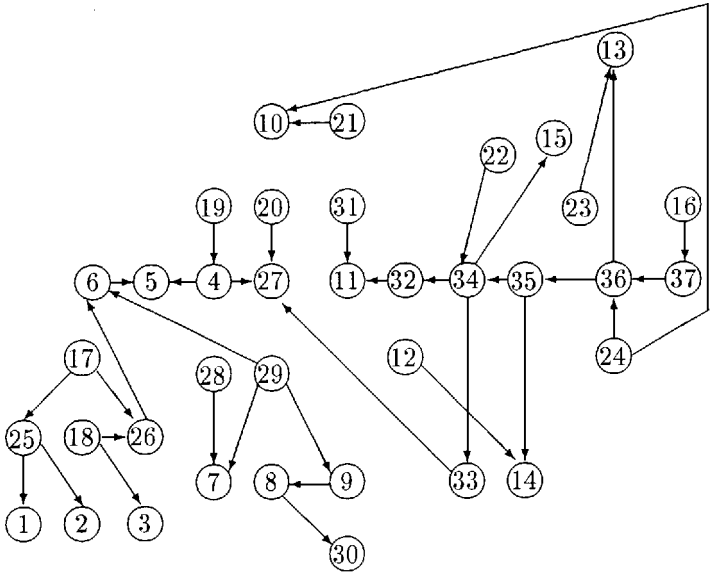

Figure 15. The Alarm polytree obtained using PA.

advantage of the fact that a distribution specified by a bayesian network has a special form, which allows us to calculate efficiently the difference of cross-entropy between two different networks: If P_G is the joint probability distribution associated to a network G whose set of nodes is $U = \{x_1, \ldots, x_n\}$, then the cross-entropy $CE(P, P_G)$ can be written in the following way:

$$CE(P, P_G) = -H_P(U) + \sum_{i=1}^{n} H_P(x_i) - \sum_{i=1, \pi_G(i) \neq} Dep(x_i, \pi_G(i) \mid)$$

where H_P denotes Shannon entropy with respect to the distribution P, $\pi_G(i)$ is the parent set of x_i in the graph G, and $Dep(x_i, \pi_G(i) \mid)$ is the Kullback–Leibler cross-entropy between x_i and $\pi_G(i)$. As the first two terms of the expression above do not depend on the graph G, then the difference of cross-entropy between two networks G_1 and G_2 can be calculated as follows:

$$CE(P, P_{G_1}) - CE(P, P_{G_2}) = \sum_{i=1, \pi_{G_2}(i) \neq}^{n} Dep(x_i, \pi_{G_2}(i) \mid) - \sum_{i=1, \pi_{G_1}(i) \neq}^{n} Dep(x_i, \pi_{G_1}(i) \mid)$$

For the Alarm network, the results are displayed in table 10. We have also included for comparative purposes, the difference of cross-entropy between the distribution associated to the true Alarm network and the one corresponding to the empty network, P (which represents marginal independency among all the variables).

From these results we can see that the distributions associated to the polytrees

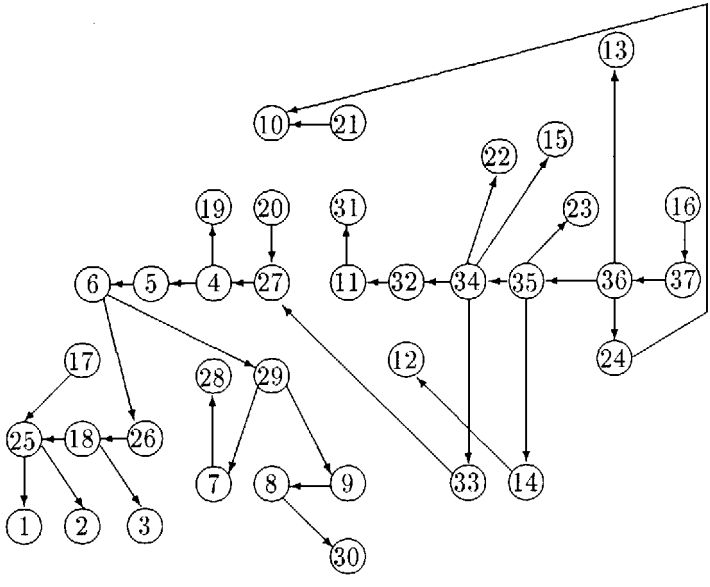

Figure 16. The Alarm polytree obtained using RP.

Table 10. Differences of cross-entropy.

$CE(P, P_{PA}) - CE(P, P_T)$	0·96
$CE(P, P_{RP}) - CE(P, P_T)$	1·40
$CE(P, P) - CE(P, P_T)$	9·04

generated by PA and RP, P_{PA} and P_{RP}, respectively, represent good approximations of the target distribution P. It can also be observed that, from a cross-entropy point of view, PA still performs better than RP.

There are several algorithms for learning bayesian networks which are quite reliable and reasonably efficient on sparse networks (Cooper and Herskovits 1992, Lam and Bacchus 1994, Spirtes *et al.* 1993). Obviously, these algorithms perform better than PA or RP on the Alarm network. However, in general, learning bayesian networks is NP-Hard (Chickering *et al.* 1994). The purpose of this example is only to show that PA and RP can extract a lot of useful and reliable information from data at low cost. For managing domains with a large number of variables (for example, in Information Retrieval applications (Ghazfan *et al.* 1996, Savoy 1991), we have to deal with thousands of terms or concepts, and each one represents a variable) or very large databases (where the number of times the data set needs to be read is very important (Ezawa and Schuermann 1995)), singly connected networks and other kinds of simplified models could represent an interesting alternative.

7. Concluding remarks

We have studied in detail the classes of dependency models which are associated with singly connected networks. The main results we obtained are the identification of finite sets of axioms that characterize dependency models isomorphic to undirected SCNs (forests and trees) through the separation criterion, and to directed SCNs (polytrees) through the d-separation criterion. In more practical terms, our theoretical study has been the basis for the development of simple and efficient learning algorithms for SCNs: we have developed algorithms that recover in polynomial (cubic) time the graphs associated with dependency models isomorphic to SCNs, as well as algorithms that find SCN approximations of any dependency model, also in polynomial time. We have also tested the performance of the approximate learning algorithms from different points of view, obtaining good results in general. Our experiments on classification problems also reveal that belief networks, and particularly polytrees, may represent a good technique for the construction of automatic classifiers, although the learning algorithms should probably be modified in some way to give them a more classification oriented perspective.

With respect to future research work, we are interested in studying the independency properties that characterize other kinds of graphical structures, such as chordal graphs and simple graphs. Chordal graphs are undirected graphs in which every cycle of length four or more has a chord, and they are the kind of graphs associated with the so-called decomposable models (which have the property of being representable by means of both undirected and directed graphs). We have already proven (de Campos 1996) that dependency models isomorphic to chordal graphs may be characterized by the axioms F1–F5 and one additional axiom similar to P14 (chordality).

On the other hand, simple graphs (Geiger *et al.* 1993) are directed acyclic graphs where every pair of nodes with a common direct child have no common ancestor nor is one an ancestor of the other. The additional axioms we have introduced to characterize polytree-isomorphic dependency models, P7–P11, are no longer valid for simple graphs. However, a weaker version of P7 is valid for simple graphs:

Weak semi-strong union:

$I(\alpha, \beta | Z)$ and $\neg I(\alpha, \beta | \ )) \Rightarrow I(\alpha, \beta | U \backslash \{\alpha, \beta\})$.

A weaker version of P8 is also valid for simple graphs:

Weak atriangularity:

$\neg I(\alpha, \gamma | Z) \forall Z \subseteq U \backslash \{\alpha, \gamma\}$ and $\neg I(\gamma, \beta | Z) \forall Z \subseteq U \backslash \{\gamma, \beta\} \Rightarrow I(\alpha, \beta | \ )$ or $\exists Z_0 \subseteq U \backslash \{\alpha, \beta, \gamma\}$ such that $I(\alpha, \beta | Z_0 \cup \gamma)$.

So, we should look for the additional axioms needed to characterize dependency models isomorphic to simple graphs. The detailed study of these topics, as well as their possible consequences which are relevant for the design of learning algorithms, will be the object of future research.

Acknowledgements

This work has been supported by the Spanish Comisión Interministerial de Ciencia y Tecnología (CICYT) under Project TIC96-0781. I should like to thank Milan Studený for his helpful comments and suggestions. I am especially grateful to Silvia Acid, for her invaluable help with the painstaking algorithm work. I am most obliged to Gregory Cooper for providing the Alarm database.

References

Acid, S., de Campos, L. M., González, A., Molina, R., and Pérez de la Blanca, N., 1991, Learning with CASTLE. *Symbolic and Quantitative Approaches to Uncertainty*, *Lecture Notes in Computer Science 548*, edited by R. Kruse and P. Siegel (Berlin: Springer Verlag), pp. 99–106.

Acid, S. and de Campos, L. M., 1995, Approximations of causal networks by polytrees: an empirical study. *Advances in Intelligent Computing*, *Lecture Notes in Computer Science 945*, edited by B. Bouchon Meunier, R. R. Yager and L. A. Zadeh (Berlin: Springer Verlag), pp. 149–158.

Aho, A. V., Hopcroft, J. E., and Ullman, J. D., 1987, *Data Structures and Algorithms* (Addison-Wesley).

Becker, A., and Geiger, D., 1994, Approximation algorithms for the loop cutset problem. *Uncertainty in Artificial Intelligence*, *Proceedings of the Tenth Conference*, edited by D. Poole and R. López de Mántaras (San Francisco: Morgan Kaufmann, 1994), pp. 60–68.

Beinlich, I. A., Suermondt, H. J., Chavez, R. M., and Cooper, G. F., 1989, The Alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. *Proceedings of the Second European Conference on Artificial Intelligence in Medicine*, pp. 247–256.

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

de Campos, L. M., 1996, Characterizations of decomposable dependency models. *Journal of Artificial Intelligence Research*, **5**, 289–300.

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

de Campos, L. M., and Huete, F., 1993b, Learning non probabilistic belief networks. *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, *Lecture Notes in Computer Science 747,* edited by M. Clarke, R. Kruse and S. Moral (Berlin: Springer Verlag), pp. 57–64.

Cano, J. E., Moral, S., and Verdegay, J. F., 1993, Propagation of convex sets of probabilities in directed acyclic networks. *Uncertainty in Intelligent Systems*, edited by B. Bouchon-Meunier, L. Valverde and R. R. Yager (Amsterdam: North-Holland), pp. 15–26.

Chickering, D. M., Geiger, D., and Heckerman, D., 1994, Learning bayesian networks is NP-Hard. Technical Report MSR-TR-94-17 (Microsoft Research).

Chow, C. K., and Liu, C. N., 1968, Approximating discrete probability distribution with dependence trees. *IEEE Transactions on Information Theory*, **14**, 462–467.

Christofides, N., 1975, *Graph Theory, an Algorithmic Approach* (London: Academic Press).

Cooper, G. F., and Herskovitz, E., 1992, A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, **9**, 309–347.

Dawid, A. P., 1979, Conditional independence in statistical theory, *Journal of the Research Statistical Society Series* B, **41**, 1–31.

Ezawa, K., and Schuermann, T., 1995, Fraud/Uncollectible debt detection using bayesian network based learning system: a rare binary outcome with mixed data structures. *Uncertainty in Artificial Intelligence*, *Proceedings of the Eleventh Conference*, edited by P. Besnard and S. Hanks (San Mateo: Morgan and Kaufmann), pp. 157–166.

Fagin, R., 1977, Multivalued dependencies and a new form for relational databases. *ACM Transactions on Database Systems*, **2**, 262–278.

Friedman, N., and Goldszmidt, M., 1996, Building classifiers using bayesian networks. *Proceedings of the National Conference on Artificial Intelligence* (Menlo Park : AAAI Press), pp. 1277–1284.

Geiger, D., 1987, The non-axiomatizability of dependencies in directed acyclic graphs. Technical Report R-83 (Cognitive Systems Laboratory, UCLA).

Geiger, D., 1992, An entropy-based learning algorithm of bayesian conditional trees. *Uncertainty in Artificial Intelligence, Proceedings of the Eighth Conference*, edited by D. Dubois, M. P. Wellman, B. D. D'Ambrosio and P. Smets (San Mateo: Morgan and Kaufmann), pp. 92–97.

Geiger, D., Paz, A., and Pearl, J., 1990, Learning causal trees from dependence information. *Proceedings of the Eighth National Conference on Artificial Intelligence*, pp. 770–776.

Geiger, D., Paz, A., and Pearl, J., 1991, Axioms and algorithms for inferences involving probabilistic independence. *Information Computing*, **91**, 128–141.

Geiger, D., Paz, A., and Pearl, J., 1993, Learning simple causal structures. *International Journal of Intelligent Systems*, **8**, 231–247.

Ghazfan, D., Indrawan, M., and Srinivasan, B., 1996, Towards meaningful bayesian network for information retrieval systems. *Proceedings of the Sixth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pp. 841–846.

Henrion, M., Propagating uncertainty in bayesian networks by logic sampling. *Uncertainty in Artificial Intelligence 2*, edited by J. F. Lemmer and L. N. Kanal (Amsterdam : North-Holland), pp. 149–163.

Herskovitz, E. H., 1991, Computer-based probabilistic networks construction. PhD Thesis, Medical Information Sciences, Stanford University, USA.

Huete, J. F., and de Campos, L. M., 1993, Learning causal polytrees. *Symbolic and Quantitative Approaches to Reasoning and Uncertainty, Lecture Notes in Computer Science 747*, edited by M. Clarke, R. Kruse and S. Moral (Berlin: Springer Verlag), pp. 180–185.

Hunter, D., 1991, Graphoids and natural conditional functions, *International Journal of Appropriate Reasoning*, **5**, 489–504.

King, R. D., Feng, C., and Sutherland, A., 1995, Statlog: comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence*, **9**, 289–333.

Kullback, S., and Leibler, R. A., 1951, Information and sufficiency. *Annals of Mathematical Statistics*, **22**, 79–86.

Lam, W., and Bacchus, F., 1994, Learning belief networks: an approach based on the MDL principle. *Computational Intelligence*, **10**, 269–293.

Lauritzen, S. L., 1982, *Lectures on Contingency Tables*, 2nd edition (Aalborg, Denmark : University of Aalborg Press).

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

Lauritzen, S. L., Dawid, A. P., Larsen, B. N., and Leimer, H.-G., 1990, Independence properties of directed Markov fields. *Networks*, **20**, 491–505.

Michie, D., Spiegelhalter, D. J., and Taylor, C. C., (eds), 1994, *Machine Learning, Neural and Statistical Classification* (London: Ellis Horwood).

Ng, K., and Levitt, I. S., 1994, Incremental dynamic construction of layered polytree networks. *Uncertainty in Artificial Intelligence, Proceedings of the Tenth Conference*, edited by R. López de Mántaras and D. Poole (San Mateo: Morgan and Kaufmann), 440–446.

Pearl, J., 1986, Fusion, propagation and structuring in belief networks. *Artificial Intelligence* **29**, 241–288.

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

Pearl, J., Geiger, D., and Verma, T., 1989, Conditional independence and its representation. *Kybernetika*, **25**, 33–44.

Pearl, J., and Paz, A., 1985, Graphoids: a graph-based logic for reasoning about relevance relations, Technical Report 850038 (R-53-L) (Cognitive Systems Laboratory, UCLA).

Pearl, J., and Verma, T. S., 1991, A theory of inferred causation. *Principles of Knowledge Representation and Reasoning, Proceedings of the Second International Conference*, J. A. Allen, R. Fikes and E. Sandwall (San Mateo: Morgan and Kaufman), pp. 441–452.

Rebane, G., and Pearl, J., 1989, The recovery of causal polytrees from statistical data. *Uncertainty in Artificial Intelligence 3*, edited by L. N. Kanal, T. S. Levitt and J. F. Lemmer (Amsterdam : North-Holland), pp. 175–182.

Sarkar, S., 1993, Using tree-decomposable structures to approximate belief networks. *Uncertainty in Artificial Intelligence, Proceedings of the Ninth Conference*, edited by D. Heckerman and A. Mamdani (San Mateo: Morgan and Kaufmann), pp. 376–382.

Savoy, J., 1991, Information retrieval in hypertext systems: an approach using bayesian networks. *Electronic Publishing*, **4**, 87–108.

Schachter, R. D., 1988, Probabilistic inference and influence diagrams, *Operational Research* **36**, 589–604.

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

Smith, J. Q., 1989, Influence diagrams for statistical modeling. *Annals of Statistics*, **17**, 654–672.
Spirtes, P., Detecting causal relations in the presence of unmeasured variables. *Uncertainty in Artificial Intelligence, Proceedings of the Seventh Conference*, edited by B. D'Ambrosio, P. Smets and P. P. Bonissone (San Mateo: Morgan and Kaufmann), pp. 392–397.
Spirtes, P., Glymour, C., and Scheines, R., 1993, *Causation, Prediction and Search, Lecture Notes in Statistics 81* (New York: Springer Verlag).
Spohn, W., 1980, Stochastic independence, causal independence and shieldability, *Journal of Philosophical Logic*, **9**, 73–99.
Studený, M., 1989, Attempts at axiomatic description of conditional independence, *Kybernetika*, **25** (Suppl. 3), 72–79.
Studený, M., 1990, Conditional independence relations have no finite complete characterization. *Transactions of the Eleventh Prague Conference on Information Theory, Statistical Decision Functions and Random Processes*, Vol. B. (Prague: Academia), pp. 377–396.
Studený, M., formal properties of conditional independence in different calculi of AI. *Symbolic and Quantitative Approaches to Reasoning and Uncertainty, Lecture Notes in Computer Science 747,* edited by M. Clarke, R. Kruse and S. Moral (Berlin: Springer Verlag), pp. 341–348.
Schweizer, B., and Sklar, A., 1983, *Probabilistic Metric Spaces* (New York: North-Holland).
Thrun, S. B., Bala, J., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., De Jong, K., Dzeroski, S., Fahlman, S., Fisher, D., Hamann, R., Kaufmann, K., Keller, S., Kononenko, I., Kreuziger, J., Michalsi, R., Mitchell, T., Pachowicz, P., Reich, Y., Vafaie, H., Van de Welde, W., Wenzel, W., Wnek, J., and Zhang, J., 1991, The Monk's problems: a performance comparison of different learning algorithms. Technical Report CMU-CS-91-197 (Carnegie Mellon University).
Verma, T., and Pearl, J., 1990a, Equivalence and synthesis of causal models. *Uncertainty in Artificial Intelligence, Proceedings of the Sixth Conference*, Mass, pp. 220–227.
Verma, T., and Pearl, J., 1990b, Causal networks: semantics and expressiveness. *Uncertainty in Artificial Intelligence 4*, edited by R. D. Schachter, T. S. Levitt, L. N. Kanal and J. F. Lemmer (Amsterdam: North-Holland), pp. 69–76.
Wilson, N., 1994, Generating graphoids from generalized conditional probability. *Uncertainty in Artificial Intelligence, Proceedings of the Tenth Conference*, edited by D. Poole and R. López de Mántaras (San Francisco: Morgan Kaufmann), pp. 583–590.

Appendix: proof of lemmas

In this appendix, we prove all the lemmas stated in the main body of the paper, as well as some other intermediate results.

Lemma 1. *Given a dependency model, M, verifying $P1$–$P11$, and its skeleton G_M, if $\alpha\gamma\beta$ is a chain in G_M, then it is either $I(\alpha, \beta \mid \gamma)$ of $I(\alpha, \beta \mid \ )$ but the two statements cannot both be true.*

Proof. Since $\alpha\gamma\beta$ is a chain in G_M we have $\neg I(\alpha, \gamma \mid U\backslash\{\alpha, \gamma\})$, $\neg I(\alpha, \gamma \mid \ )$, $\neg I(\gamma, \beta \mid U\backslash\{\beta, \gamma\})$ and $\neg I(\gamma, \beta \mid \ )$. Then using semi-strong atriangularity we obtain $I(\alpha, \beta \mid \gamma)$ or $I(\alpha, \beta \mid \ )$. On the other hand, from $\neg I(\alpha, \gamma \mid \ )$, $\neg I(\gamma, \beta \mid \ )$ and weak transitivity (in contrapositive form) we obtain $\neg I(\alpha, \beta \mid \gamma)$ or $\neg I(\alpha, \beta \mid \ )$. Therefore we have $I(\alpha, \beta \mid \gamma)$ and $\neg I(\alpha, \beta \mid \ )$, or $I(\alpha, \beta \mid \ )$ and $\neg I(\alpha, \beta \mid \gamma)$. □

Lemma 2. *Given a dependency model, M, verifying $P1$–$P11$, and its skeleton G_M, if $t_1 t_2 \cdots t_{n-1} t_n$ is a chain in G_M such that $I(t_{i-1}, t_{i+1} \mid t_i)\ \forall i = 2, \ldots, n-1$, then*

(*a*) $I(t_1, t_n \mid t_i), \forall i = 2, \ldots, n-1$.
(*b*) $\neg I(t_1, t_n \mid Z)\ \forall Z \subseteq U\backslash\{t_1, \ldots, t_n\}$.

Proof. First, we are going to prove that $I(t_1, t_n \mid t_{n-1})$ and $\neg I(t_1, t_n \mid \ )$ by using induction. For $n = 3$ the result is true because of Lemma 1. Suppose that the result is true for $n-1$, that is to say, $I(t_1, t_{n-1} \mid t_{n-2})$ and $\neg I(t_1, t_{n-1} \mid \ )$. As $I(t_{n-2}, t_n \mid t_{n-1})$, then $\neg I(t_{n-2}, t_n \mid \ )$, again using Lemma 1. By applying semi-strong transitivity to these two statements, we obtain $I(t_1, t_n \mid t_{n-1})$ or $I(t_1, t_{n-2} \mid t_{n-1})$. If $I(t_1, t_{n-2} \mid t_{n-1})$ is true, as we have $I(t_1, t_{n-1} \mid t_{n-2})$, then intersection and decomposition produce $I(t_1, t_{n-1} \mid \ )$, which contradicts the induction hypothesis; hence we have $I(t_1, t_n \mid t_{n-1})$. Moreover, if

$I(t_1, t_n \mid)$ then we can apply weak transitivity, which gives $I(t_1, t_{n-1} \mid)$ or $I(t_{n-1}, t_n \mid)$, and both statements are false, the first one according to the induction hypothesis and the second one because t_{n-1} and t_n are adjacent in G_M. Therefore $\neg\, I(t_1, t_n \mid)$.

Now, let us prove the part (a) of the lemma: from the previous result, applied to the (sub)chain $t_1 \cdots t_i\, t_{i+1}$, we obtain as a result $I(t_1, t_{i+1} \mid t_i)$ and $\neg\, I(t_1, t_{i+1} \mid)$. Now, by using semi-strong transitivity we obtain $I(t_1, t_n \mid t_i)$ or $I(t_{i+1}, t_n \mid t_i)$. If we again apply the previous result to the (sub)chain $t_n \cdots t_{i+1}\, t_i$ we obtain $I(t_i, t_n \mid t_{i+1})$ and $\neg\, I(t_i, t_n \mid)$. In the case that $I(t_{i+1}, t_n \mid t_i)$ is true, then from $I(t_{i+1}, t_n \mid t_i)$ and $I(t_i, t_n \mid t_{i+1})$, by using intersection and decomposition, we obtain $I(t_i, t_n \mid)$, which is a contradiction. So, we have $\neg\, I(t_{i+1}, t_n \mid t_i)$ and therefore $I(t_1, t_n \mid t_i)$.

Finally, let us prove the part (b) of the lemma: we are going to prove first $\neg\, I(t_1, t_n \mid \gamma)\ \forall \gamma \in U \backslash \{t_1, \ldots, t_n\}$, using induction. For $n = 3$, since t_2 is adjacent to t_1 and t_3, we have $\neg\, I(t_1, t_2 \mid \gamma)$ and $\neg\, I(t_2, t_3 \mid \gamma)$, $\forall \gamma \in U \backslash \{t_1, t_2, t_3\}$. Moreover, from Lemma 1 we deduce $\neg\, I(t_1, t_3 \mid)$. Therefore, from semi-strong transitivity we obtain $\neg\, I(t_1, t_3 \mid \gamma)$. Now, let us suppose that the result is true for $n-1$, i.e. $\neg\, I(t_1, t_{n-1} \mid \gamma)$. If $I(t_1, t_n \mid \gamma)$, as we already know that $\neg\, I(t_1, t_n \mid)$, then we can apply semi-strong transitivity, and we obtain $I(t_1, t_{n-1} \mid \gamma)$ or $I(t_{n-1}, t_n \mid \gamma)$, and both statements are false (the first one according to the induction hypothesis and the second one because of the adjacency of t_{n-1} and t_n). Therefore we have $\neg\, I(t_1, t_n \mid \gamma)\ \forall \gamma \in U \backslash \{t_1, \ldots, t_n\}$. Now, using singularity we obtain $\neg\, I(t_1, t_n \mid Z)\ \forall Z \subseteq U \backslash \{t_1, \ldots, t_n\}$. □

To prove Lemma 3, we need the following auxiliary result:

Auxiliary Lemma. *Given a dependency model, M, verifying P1–P11, and an associated dag D_M, if $t_n \cdots t_1\, \gamma s_1 \cdots s_m$ is a chain in D_M such that $I(t_{i-1}, t_{i+1} \mid t_i)\ \forall i = 2, \ldots, n-1$, $I(s_{j-1}, s_{j+1} \mid s_j)\ \forall j = 2, \ldots, m-1$, and $I(t_1, s_1 \mid)$, then $I(t_n, s_m \mid)$.*

Proof. We have already proven (in Proposition 5) that a node adjacent to a head to head node cannot be head to head as well. In our case, this implies $\neg\, I(t_2, \gamma \mid)$ because t_1 is adjacent to the head to head node γ; then from Lemma 1 we deduce $I(t_2, \gamma \mid t_1)$.

From the hypothesis and the previous result we see that the chain $t_n \cdots t_2\, t_1\, \gamma$ satisfies the conditions in Lemma 2. Therefore we deduce $I(t_n, \gamma \mid t_1)$ and $\neg\, I(t_n, \gamma \mid)$. By applying semi-strong transitivity, we obtain either $I(t_n, s_1 \mid t_1)$ or $I(\gamma, s_1 \mid t_1)$. As γ and s_1 are adjacent nodes, then we have $\neg\, I(\gamma, s_1 \mid t_1)$ and therefore we have $I(t_n, s_1 \mid t_1)$; this statement, together with $I(t_1, s_1 \mid)$, gives $I(t_n, s_1 \mid)$ by using contradiction and decomposition.

On the other hand, the chain $s_m \cdots s_2\, s_1\, \gamma$ also verifies the conditions of Lemma 2. Then we obtain $I(s_m, \gamma \mid s_1)$ and $\neg\, I(s_m, \gamma \mid)$. Once again using semi-strong transitivity, we obtain $I(t_n, s_m \mid s_1)$ or $I(t_n, \gamma \mid s_1)$. However, $I(t_n, \gamma \mid s_1)$ is false because of Lemma 2(b), since s_1 is not in the chain without head to head nodes linking t_n and γ. So, we have $I(t_n, s_m \mid s_1)$, that together with $I(t_n, s_1 \mid)$ gives $I(t_n, s_m \mid)$ by contraction and decomposition. □

This auxiliary lemma states that, in a chain of D_M having only one head to head node, the extreme nodes are marginally independent.

Lemma 3. *Given a dependency model, M, verifying P1–P11, and an associated dag D_M, if $I(\alpha, \beta \mid \gamma)$ and $\neg\, I(\alpha, \beta \mid)$, then there is a chain in D_M linking α and β which does not have head to head nodes and does contain γ.*

Proof. The proof is constructive. Let us define the set S, as follows

$$S = \{t \in U \backslash \{\alpha\} \mid \neg\, I(\alpha, t \mid U \backslash \{\alpha, \beta\})\}.$$

Firstly, we are going to prove that $S \neq$.

If $S =$ then $\forall t \in U\backslash\{\alpha\}$ is $I(\alpha, t \mid U\backslash\{\alpha, t\})$. Let $t_1, t_2 \in U\backslash\{\alpha\}$; using intersection, from $I(\alpha, t_1 \mid U\backslash\{\alpha, t_1\})$ and $I(\alpha, t_2 \mid U\backslash\{\alpha, t_2\})$ we can deduce $I(\alpha, t_1 \cup t_2 \mid U\backslash\{\alpha, t_1, t_2\})$. By repeating this process for the rest of variables in $U\backslash\{\alpha\}$, we obtain $I(\alpha, U\backslash\{\alpha\} \mid)$ in a finite number of steps. In particular, we have (by using decomposition) $I(\alpha, \beta \mid)$ which contradicts the hypothesis. Hence $S \neq$. The same reasoning used before allows us to prove $I(\alpha, U\backslash(S \cup \{\alpha\}) \mid S)$.

On the other hand, from $I(\alpha, \beta \mid \gamma)$ and $\neg I(\alpha, \beta \mid)$, we obtain $I(\alpha, \beta \mid U\backslash\{\alpha, \beta\})$ using semi-strong union, and therefore $\beta \notin S$. So, we also deduce $I(\alpha, \beta \mid S)$ by using decomposition.

Now, from $I(\alpha, \beta \mid S)$ and singularity we deduce that $\exists t_0 \in S$ such that $I(\alpha, \beta \mid t_0)$. If $I(\alpha, t_0 \mid)$, then using contraction we obtain $I(\alpha, \beta \mid)$ which contradicts the hypothesis; hence $\neg I(\alpha, t_0 \mid)$. In graph terms this means that the set $T = \{t \in U\backslash\{\alpha\} \mid \neg I(\alpha, t \mid U\backslash\{\alpha, t\}), \neg I(\alpha, t \mid)\}$ of nodes adjacent to α in the graph D_M is not empty, because t_0 is adjacent to α. If $I(t_0, \beta \mid)$, then from $I(\alpha, \beta \mid t_0)$ and contraction we obtain $I(\alpha, \beta \mid)$; therefore $\neg I(t_0, \beta \mid)$. On the other hand, from $\neg I(\alpha, \beta \mid)$ and $I(\alpha, \beta \mid \gamma)$, using semi-strong transitivity, we obtain $I(\alpha, t_0 \mid \gamma)$ or $I(t_0, \beta \mid \gamma)$ (we suppose that $t_0 \neq \gamma$). Since α and t_0 are adjacent nodes, the statement $I(\alpha, t_0 \mid \gamma)$ is false and then we have $I(t_0, \beta \mid \gamma)$. Once again, by using semi-strong transitivity, from $\neg I(\alpha, \beta \mid)$ and $I(\alpha, \beta \mid t_0)$ we obtain $I(\alpha, \gamma \mid t_0)$ or $I(\gamma, \beta \mid t_0)$. But $I(\gamma, \beta \mid t_0)$, together with $I(t_0 \beta \mid \gamma)$ gives, using intersection and decomposition, $I(t_0, \beta \mid)$, which is false; therefore we have $I(\alpha, \gamma \mid t_0)$. At the moment, we have found a node t_0 such that $\neg I(\alpha, t_0 \mid U\backslash\{\alpha, t_0\}), \neg I(\alpha, t_0 \mid), \neg I(t_0, \beta \mid), I(\alpha, \beta \mid t_0)$, $I(\alpha, \gamma \mid t_0)$ and $I(t_0, \beta \mid \gamma)$ (in fact, t_0 is the single node having these properties, but this is not important for our purposes).

Now, we can apply the previous development to the nodes t_0 and β (because $\neg I(t_0, \beta \mid)$ and $I(t_0, \beta \mid \gamma)$). So, we find a single node t_1 such that $\neg I(t_0, t_1 \mid U\backslash\{t_0, t_1\})$, $\neg I(t_0, t_1 \mid)$, $\neg I(t_1, \beta \mid)$, $I(t_0, \beta \mid t_1), I(t_0 . \gamma \mid t_1)$ and $I(t_1, \beta \mid \gamma)$. Moreover, from $\neg I(\alpha, \beta \mid)$ and $I(\alpha, \beta \mid t_0)$ we obtain $I(\alpha, t_1 \mid t_0)$ or $I(t_1, \beta \mid t_0)$ by using semi-strong transitivity. If $I(t_1, \beta \mid t_0)$, since we also have $I(t_0, \beta \mid t_1)$, intersection and decomposition give $I(t_0, \beta \mid)$, which is a contradiction. Therefore, we have $I(\alpha, t_1 \mid t_0)$.

By repeating this process we can find nodes $t_2, t_3 \ldots$ until some t_i coincides with γ (as the number of nodes is finite, this process must stop within a finite number of steps, provided that all the variables t_j are always different from each other). These nodes verify $\neg I(t_{j-1}, t_j \mid U\backslash\{t_{j-1}, t_j\})$, $\neg I(t_{j-1}, t_j \mid)$, $\neg I(t_j, \beta \mid)$, $I(t_{j-1}, \beta \mid t_j)$, $I(t_{j-1}, \gamma \mid t_j)$ and $I(t_j, \beta \mid \gamma)$, $\forall j = 1, \ldots, i-1$ and $\neg I(t_{i-1}, \gamma \mid U\backslash\{t_{i-1}, \gamma\})$, $\neg I(t_{i-1}, \gamma \mid)$, $I(t_{i-1}, \beta \mid \gamma)$. So, we have built a chain $\alpha t_0 t_1 \cdots t_{i-1} \gamma$ in D_M. Let us see that this chain does not have head to head nodes: from $\neg I(t_{j-1}, \beta \mid)$ and $I(t_{j-1}, \beta \mid t_j)$ we deduce either $I(t_{j-1}, t_{j+1} \mid t_j)$ or $I(t_{j+1}, \beta \mid t_j)$ by using semi-strong transitivity. If $I(t_{j+1}, \beta \mid t_j)$, as it is also $I(t_j, \beta \mid t_{j+1})$, we obtain $I(t_j, \beta \mid)$ by intersection and decomposition, and this statement is false. Hence $I(t_{j-1}, t_{j+1} \mid t_j)$ and from Lemma 1, $\neg I(t_{j-1}, t_{j+1} \mid)$, which means that there are no head to head nodes.

To prove that we never find a variable t_r which coincides with some other previous t_k, we use the following reasoning: if $t_r = t_k$ with $k < r-1$, then from $I(t_{r-2}, \beta \mid t_{r-1})$ and $\neg I(t_{r-2}, \beta \mid)$ we find $I(t_{r-3}, t_{r-2} \mid t_{r-1})$ (which is not possible) or $I(t_{r-3}, \beta \mid t_{r-1})$ by using semi-strong transitivity. By repeating this process, we obtain $I(t_k, \beta \mid t_{r-1}) \equiv I(t_r, \beta \mid t_{r-1})$, which, together with $I(t_{r-1}, \beta \mid t_r)$, gives (using intersection and decomposition) $I(t_r, \beta \mid)$, which is a false statement. Therefore the variables t_j are all different.

If we apply exactly the same process as before, but starting from β instead of α, we obtain another chain $\gamma s_{h-1} \cdots s_0 \beta$ without head to head nodes. If for some k and l

it is $t_k = s_l$, then we have $I(\alpha, \gamma | t_k)$ (from Lemma 2(a)) and $I(\alpha, s_l | \gamma) \equiv I(\alpha, t_k | \gamma)$, which produce by intersection and decomposition the false statement $I(\alpha, \gamma |\ )$. So, $t_k \neq s_l \forall k, l$, and we have a chain $\alpha t_0 t_1 \cdots t_{i-1} \gamma s_{h-1} \cdots s_0 \beta$ where all the nodes (except perhaps γ) are not head to head. But γ cannot be the single head to head node in the chain because in that case we could obtain $I(\alpha, \beta |\ )$ from the auxiliary lemma. The proof is complete. □

Lemma 4. *Let M be a dependency model verifying $P1$–$P11$, and let D_M be an associated dag. Then*

$$\forall \alpha, \beta \in U\,(I(\alpha, \beta |\ ) \Leftrightarrow \langle \alpha, \beta |\ \rangle_{D_M}).$$

Proof. Let us suppose that $\neg \langle \alpha, \beta |\ \rangle_{D_M}$. Then there is a chain in D_M without head to head nodes, and by using Lemma 2(b), we obtain $\neg I(\alpha, \beta |\ )$. This proves the necessary condition.

Now let us suppose that $\neg I(\alpha, \beta |\ )$. If $\neg I(\alpha, \beta | U \setminus \{\alpha, \beta\})$ is also true, then α and β are adjacent nodes in D_M and therefore $\neg \langle \alpha, \beta |\ \rangle_{D_M}$. The other possibility is that $I(\alpha, \beta | U \setminus \{\alpha, \beta\})$. In this case, from singularity we can find a node γ such that $I(\alpha, \beta | \gamma)$, and from Lemma 3 we deduce that there is a chain linking α and β which does not contain head to head nodes; thus we have $\neg \langle \alpha, \beta |\ \rangle_{D_M}$. This proves the sufficient condition. □